Online Learning And Monitoring

Giacomo Boracchi, Francesco Trovò

May 6th, 2020

Politecnico di Milano, DEIB

giacomo.boracchi@polimi.it

Practical Information: Schedule

Lecture Dates:

- Intro and Learning in NSE (monitoring): 6 May 2020 from 10.00 to 13.30
- Learning in NSE (adaptation): 13 May 2020 from 10.00 to 13.30
- Learning with Experts: 20 May 2020 from 10.00 to 13.30
- Learning with MAB: 27 May 2020 from 10.00 to 13.30
- Anomaly Detection and Domain Adaptation: 3 June 2020 from 10.00 to 13.30
- Applications: 10 June 2020 from 10.00 to 13.30

Practical Information: Teaching Modality

We will always use the same Teams meeting, so you will have access to all the lectures through the same link

Lectures will be recorded: so you might want to go through the materials later. You should find videos on stream.

We will provide you a few Matlab snippets to fill in, to better familiarize with the course material. During lectures we will give you a short «lab» time to develop these codes, then we will go through them.

Let us promote interaction as much as possible

- Directly unmuting yourself if you feel brave and if lecture timing allows
- Using the chat (please be polite)
- During the «lab» part of the lecture

Practical Information: Exam

PhD Students: Pass/Fail

- Discussion about the exam topics
- Short oral exam where to discuss a few assignments and one extension of these

MSc Students: Grades (18-30L)

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

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

Practical Information: Teaching Material

There is not a unique reference book for this teaching material, but relevant papers are cited in each slide

- Concerning the Learning in NSE part of the course, you can refer to:

Cesare Alippi "Intelligence for Embedded Systems. A Methodological Approach", Springer 2014, Chapter 9: Learning in Nonstationary and Evolving Environments

- Concerning Change Detection:

Basseville, Michèle, and Igor V. Nikiforov. *Detection of abrupt changes: theory and application*. Vol. 104. Englewood Cliffs: prentice Hall, 1993.

- Concerning the Online Learning part of the course, you can refer to:

Nicolò Cesa-Bianchi, Gábor Lugosi "Prediction, learning, and games" Cambridge university press 2006, Chapters 1-4

Sébastien Bubeck, Nicolò Cesa-Bianchi "Regret analysis of stochastic and nonstochastic multi-armed bandit problems" Foundations and Trends in Machine Learning 2012, Chapters 1-3

Tutorials relevant to Learning NSE

Change and Anomaly Detection in Signals, Images, and General Data Streams Giacomo Boracchi *Tutorial at IEEE ICASSP 2018; April 16th, 2018*

Learning in Nonstationary Environments: Perspective and Applications Giacomo Boracchi and Gregory Ditzer *Tutorial at SSCI 2015, Symposium Series in Computational Intelligence; December 8th, 2015, Cape Town, South Africa*

Learning Class Imbalanced Data Streams, Leandro Minku, Shuo Wang and Giacomo Boracchi *World Congress on Computational Intelligence (WCCI)*, July 2018.

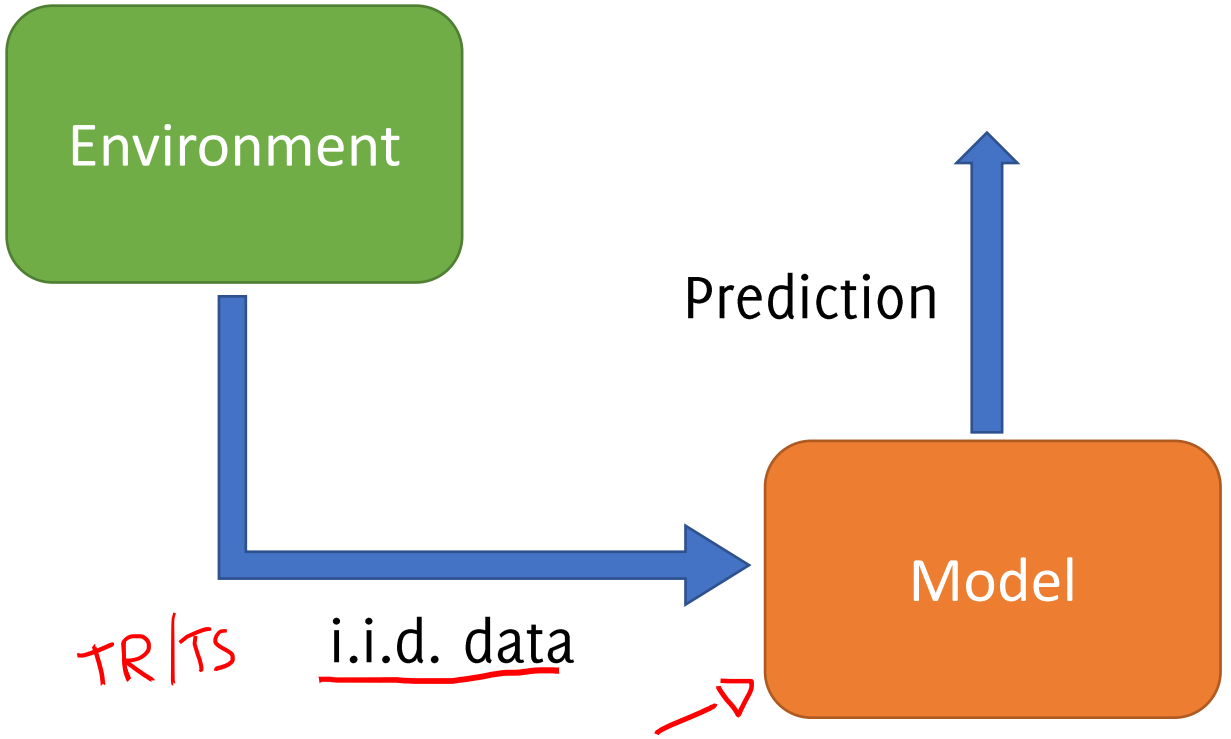
Questions?

General ML Framework

Typical assumption in ML:

*Incoming data (both training or testing) are **independent and identically distributed** (i.i.d.) realizations of an unknown process*

The major focus is towards how data-driven models able to extract information out of these training data



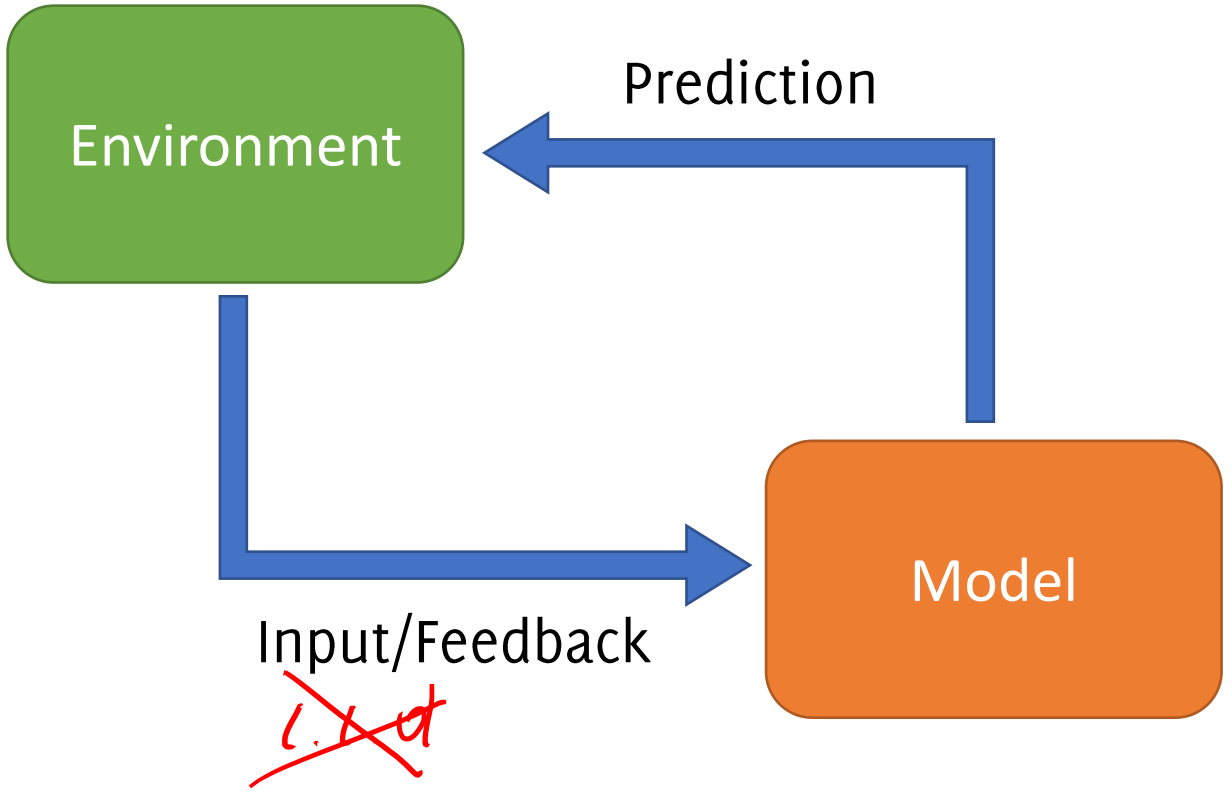
This Course Framework

In a **streaming** scenario:

- the **environment** might be **changing or adversarial**
- the **model-environment interactions** can not be disregarded

These settings call for:

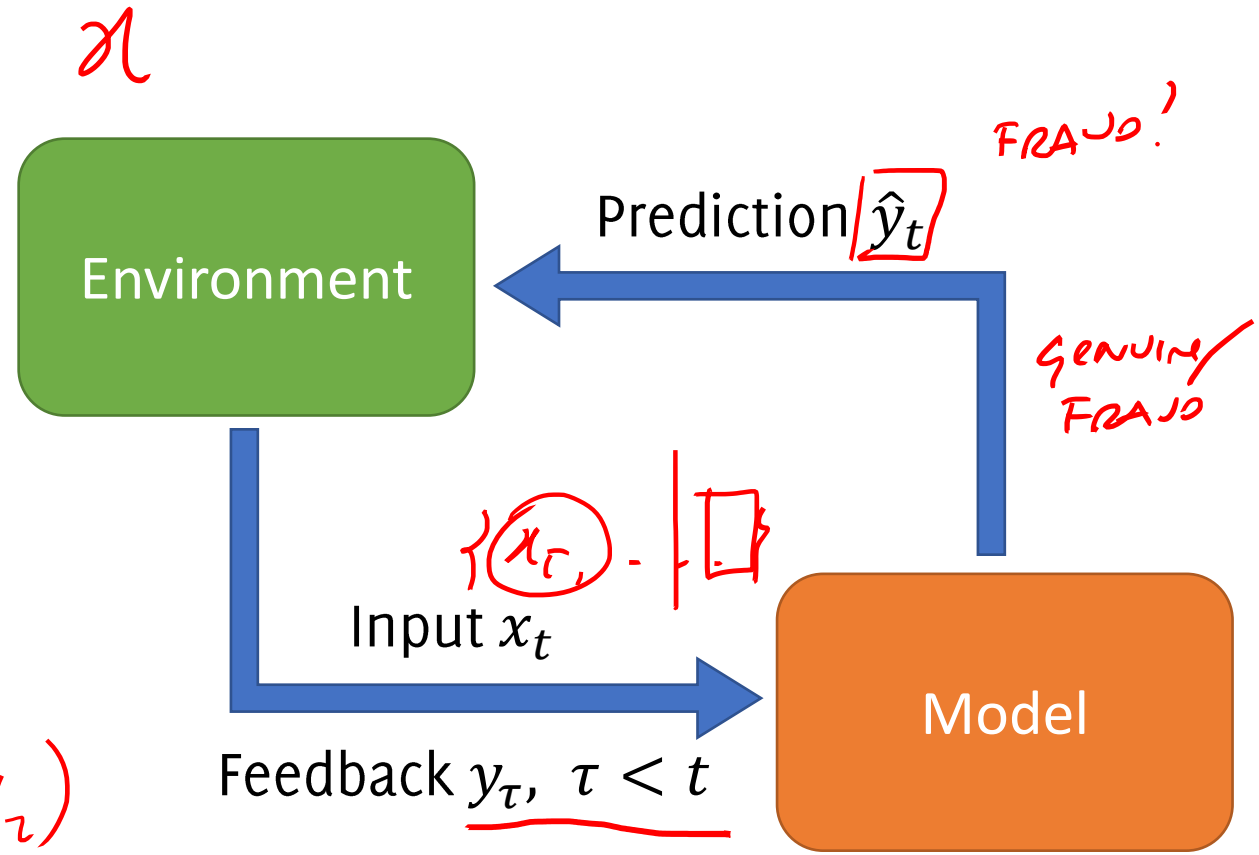
- Techniques to learn-adapt the data-driven model
- Techniques to monitor the interaction model-environment



Learning in Non-Stationary Environment

At each time instant t

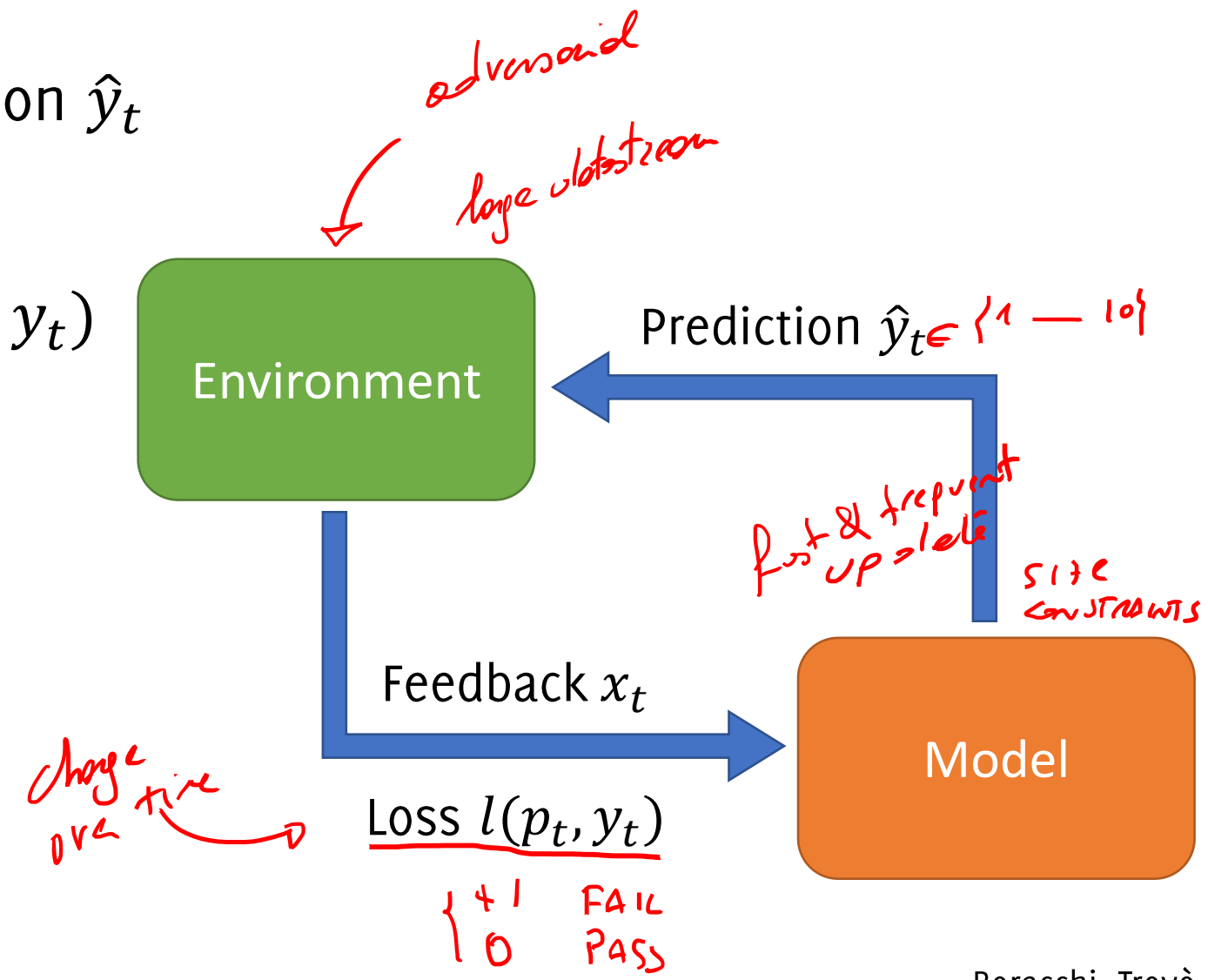
- we get an input x_t from a stream
- we generate a prediction $\hat{y}_t$
- we get feedback y_τ $(\tau < t)$
- we update the model



Online Learning Framework

At each round t

- The model generates a prediction $\hat{y}_t$
- the environment chooses y_t
- the model receives a loss $l(p_t, y_t)$
- the model gets feedback x_t
- the model is updated



Course Overview

Typical assumption in ML:

Training and incoming data are i.i.d.

This course:

Data are either nonstationary or chosen by an adversarial

These settings are often encountered in **real-world applications on streaming data,** e.g., to select sponsored links for Internet advertising, or to detect frauds in credit card transaction.

The course provides an **overview of techniques to employ data-driven models** in these **streaming settings**

Disclaimer

The course deals two different approaches, based on different models:

- Learning in NonStationary Environment (NSE): driven by practical problems, rarely the developed methods can be handled analytically to prove theoretical guarantees. The major emphasis is over the application.
- Online learning: formal approach to the problem, requiring strong theoretical requirements on the algorithms (focus on the analysis). The usage of these models in real-life scenarios requires substantial adjustments/approximations.

This is the first edition of the course: notation might not be consistent and slides might contain (minor?) errors. Please report these to us

Today's outline

Learning in Non-stationary Environment

- Fraud Detection
- Problem Formulation and Concept Drift
- Learning in NSE Approaches

Monitoring

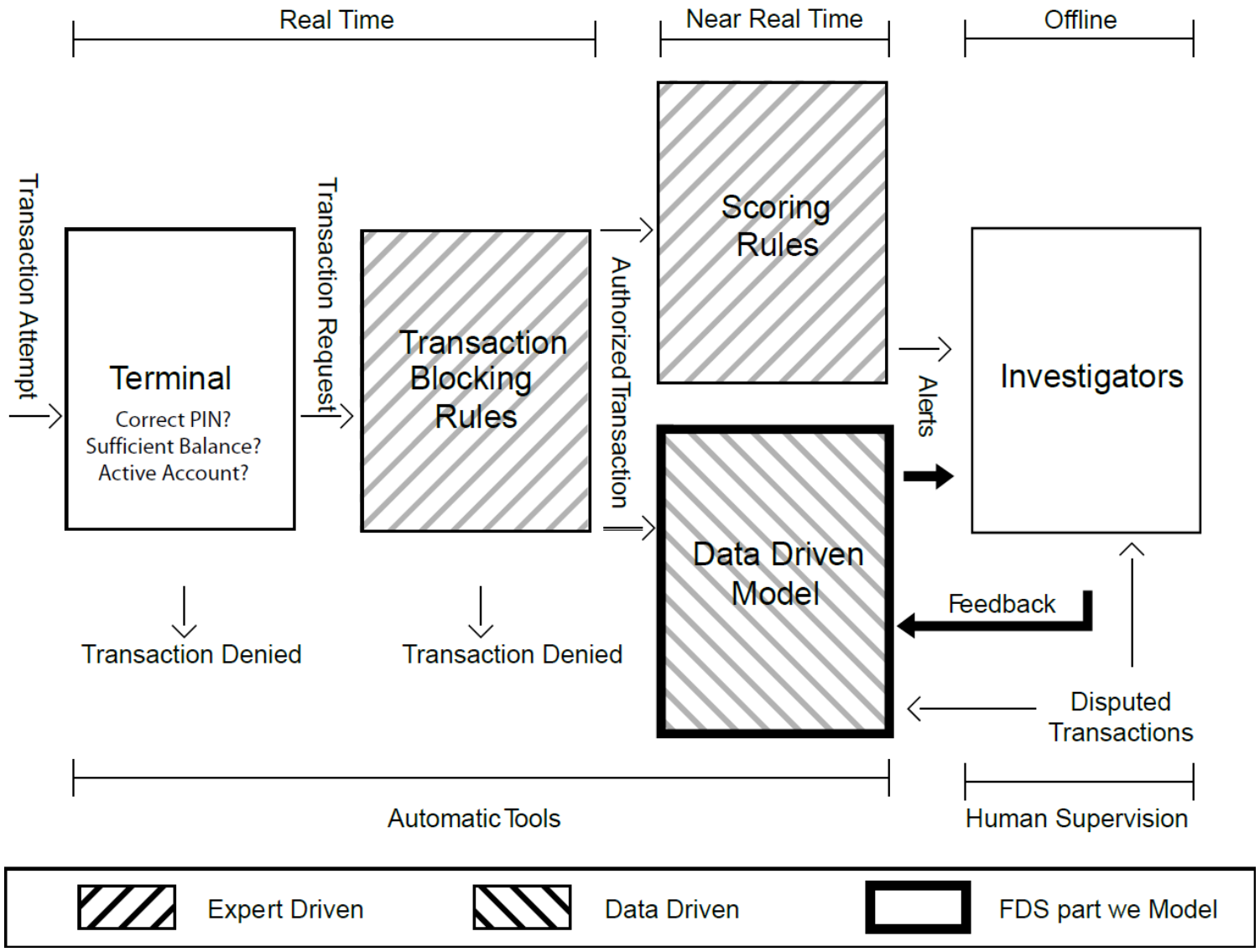
- Change Detection Test on the classification error
- Change Detection test on the input distribution

Fraud Detection

A Cool Example for Learning in NSE

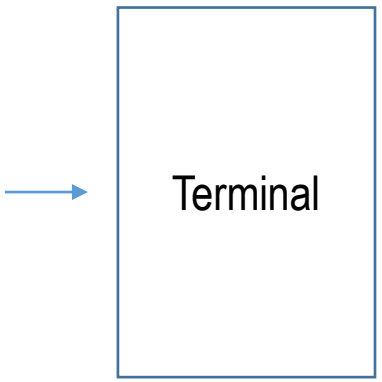
Boracchi, Trovò

Fraud Detection



Dal Pozzolo A., Boracchi G., Caelen O., Alippi C. and Bontempi G., *"Credit Card Fraud Detection: a Realistic Modeling and a Novel Learning Strategy"* , IEEE TNNLS 2017

The Terminal



Purchase

ò

The Terminal

Acceptance checks like:

- Correct PIN
- Number of attempts
- Card status (active, blocked)
- Card balance / availability

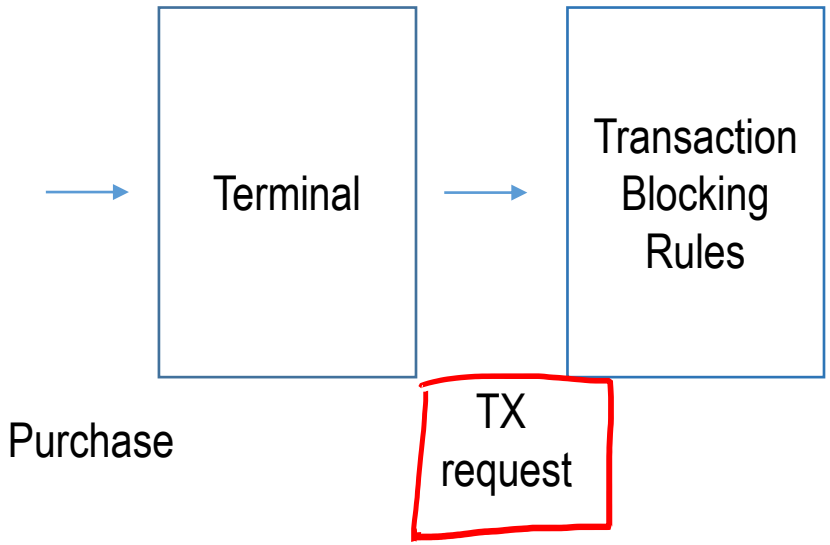
are immediately performed.

These checks are done in **real time**, and **preliminary filter** our purchases: when these checks are not satisfied, the card/transaction can be blocked.

Otherwise, a **transaction request** is entered in the system that include information of the actual purchase:

- *transaction amount, merchant id, location, transaction type, date time, …*

Blocking rules



ò

Transaction Blocking Rules

Association rules (if-then-else statements) like*

IF Internet transactions AND compromised website THEN deny the transaction

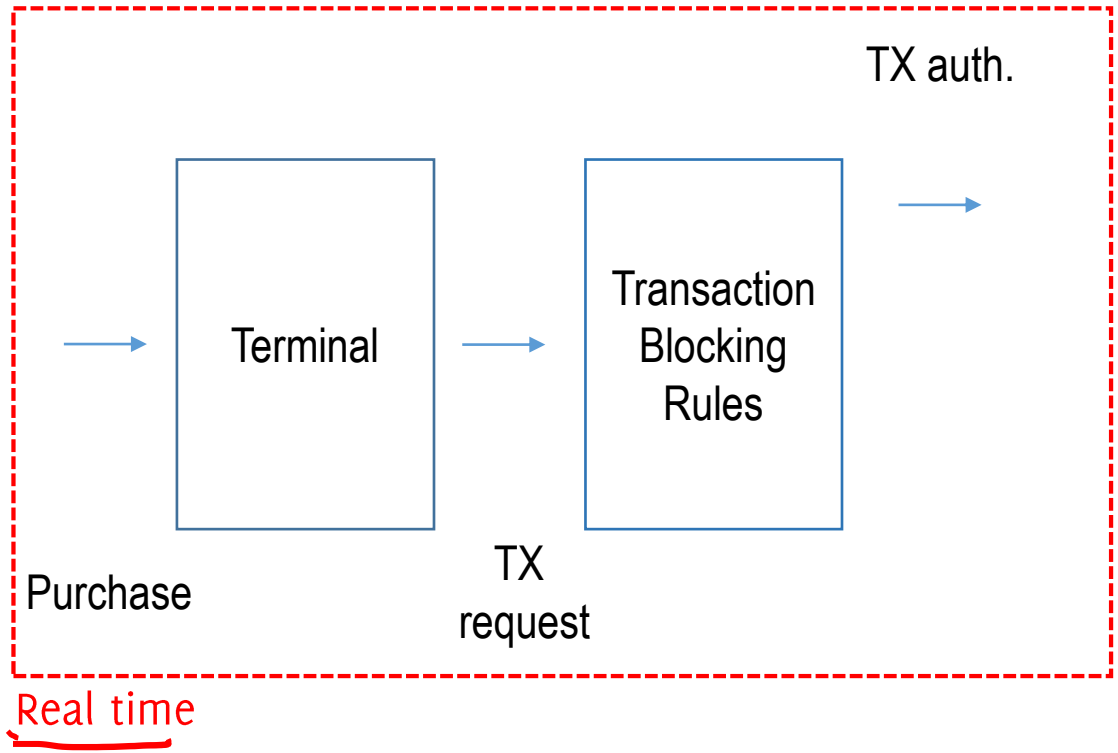
These rules:

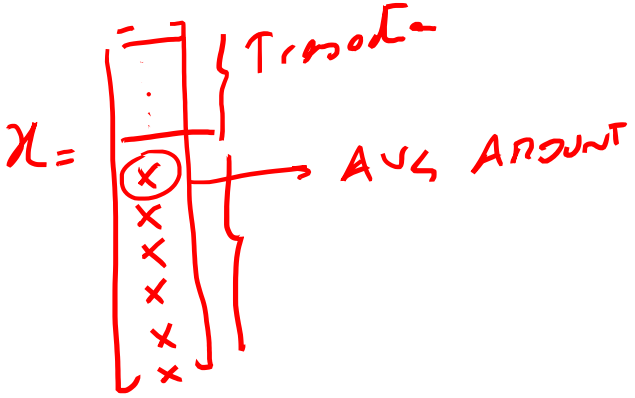
- are **expert-driven**, designed by investigators
- involves quite simple expressions with a few data
- are easy to interpret
- have always «deny the transaction» as statement
- are executed in real time

All the transaction RX passing these rules are **authorized transactions** and further analyzed by the FDS

(*) Transaction blocking rules are confidential and this is just a likely example

Real Time Processing





ò

Feature Augmentation

A feature vector $\boldsymbol{x}$ is associated to each authorized transaction.

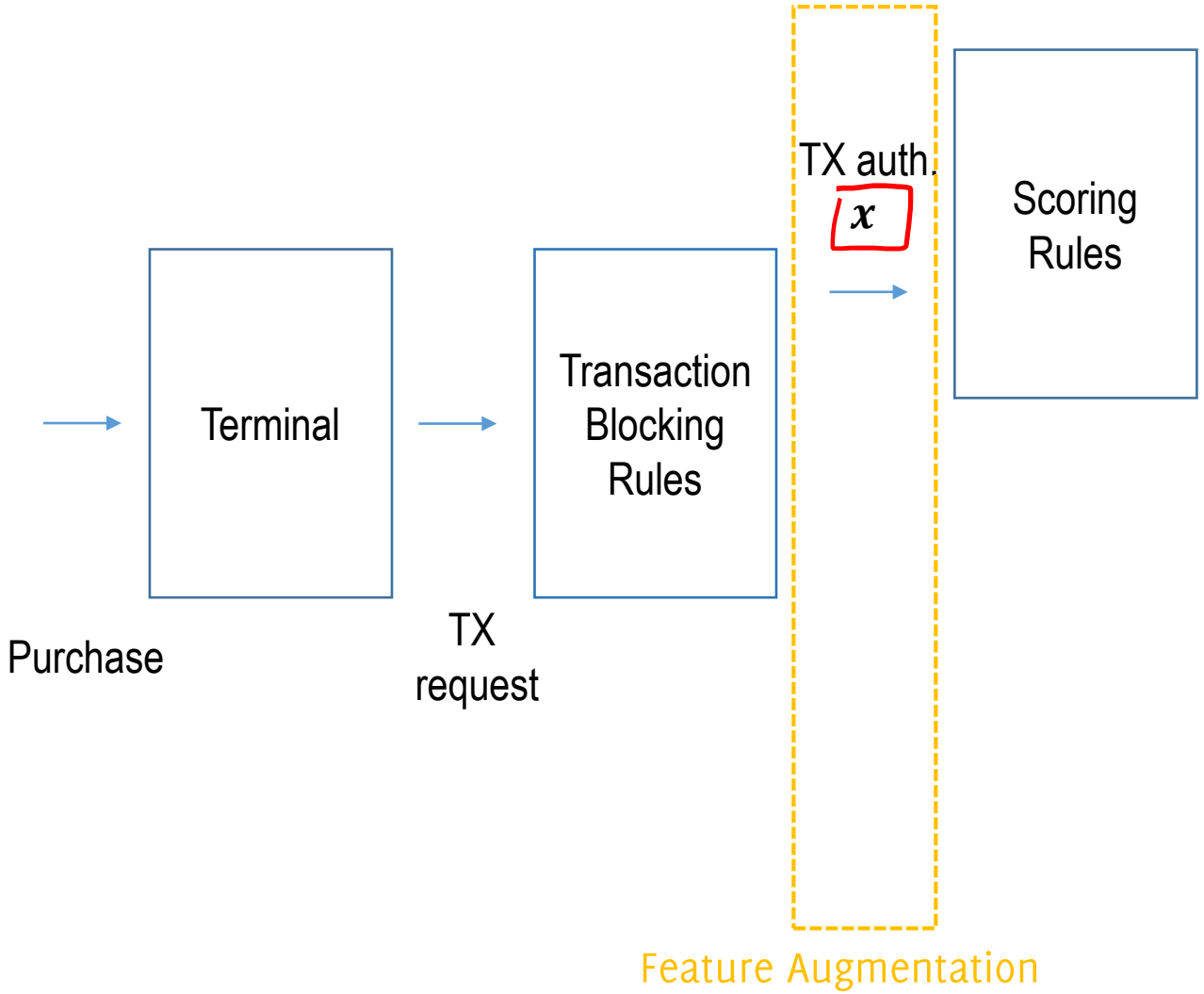
The components of $\boldsymbol{x}$ include data about the current transaction and customary shopping habits of the cardholder, e.g.:

- the average expenditure
- the average number of transactions per day
- the cardholder age
- the location of the last purchases
- …

and are very informative for fraud-detection purposes

Overall, about 40 features are extracted in near-real time.

Scoring Rules



ò

Scoring Rules

Scoring rules are if-then-else statements that:

- are being processed in near-real time
- are **expert-driven**, designed by investigators.
- Operate on augmented features (components of $\boldsymbol{x}$)
- Assign a **score**: the larger the score the more risky the transaction (an estimate of the probability for $\boldsymbol{x}$ to be a fraud, according to investigator expertise)
- Feature vector receiving large scores are alerted
- Are easy to interpret and are designed by investigators

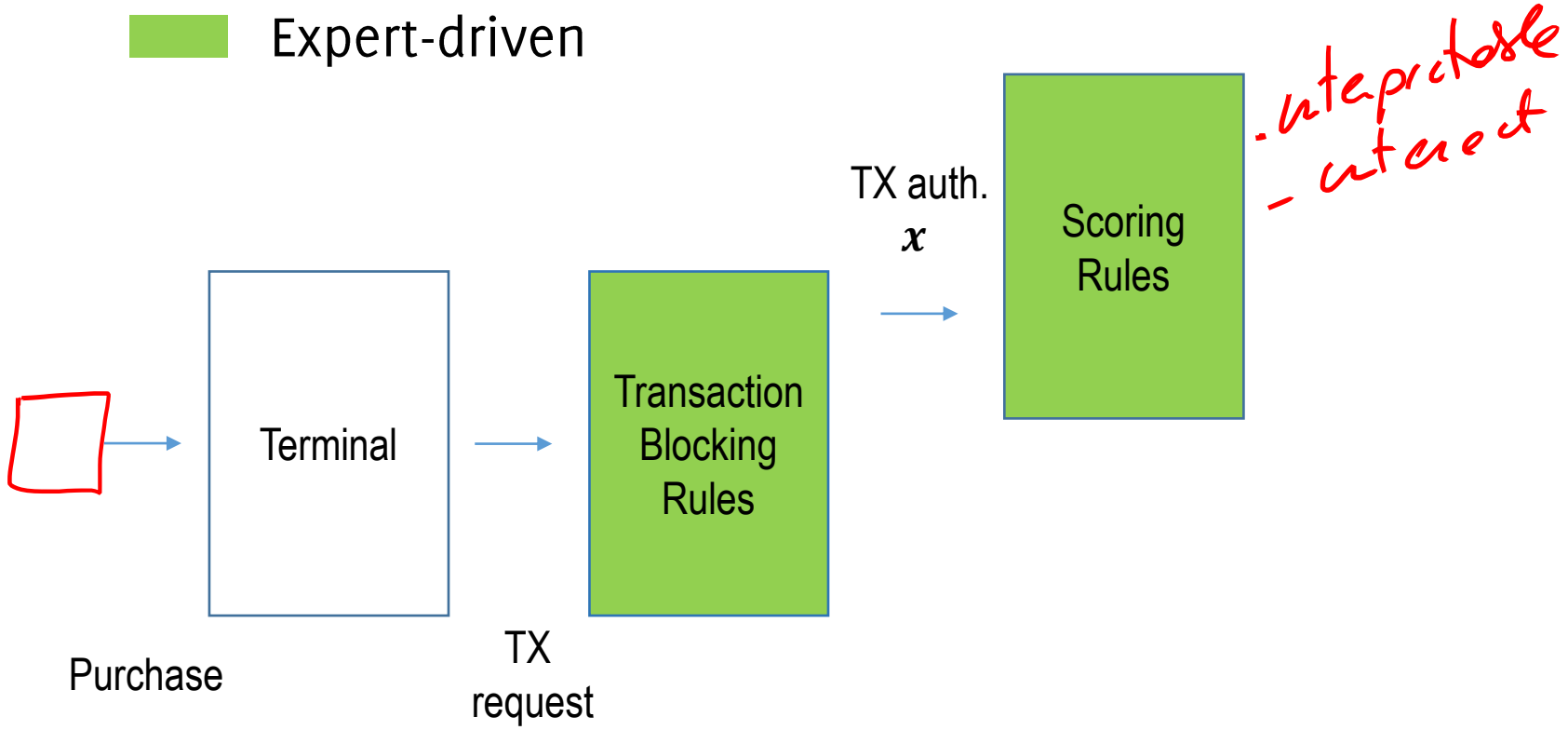
Scoring Rules

Examples* of scoring rules might be:

- *IF previous transaction in a different country AND less than 2 hours since the previous transaction, AND operation using PIN THEN fraud score = 0.95*
- *IF amount > average of transactions* $+ 3\sigma$ *AND country is a fiscal paradise AND customer travelling habits low THEN fraud score = 0.75*

(*) Scoring rules are confidential and these are just likely examples

Expert-Driven Models in fraud detection



- interpretable
- interact

Expert-Driven vs Data-Driven models

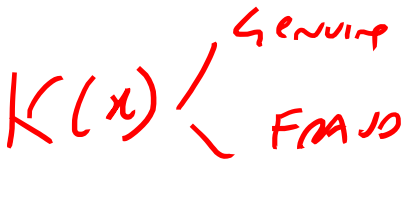
Scoring rules are an **expert-driven model**, thus:

- Can detect **well-known / reasonable** frauds
- Involve **few components** of the feature vector
- **Difficult** to **exploit correlation** among features

Expert-Driven vs Data-Driven models

Scoring rules are an **expert-driven model**, thus:

- Can detect **well-known / reasonable** frauds
- Involve **few components** of the feature vector
- **Difficult** to **exploit correlation** among features

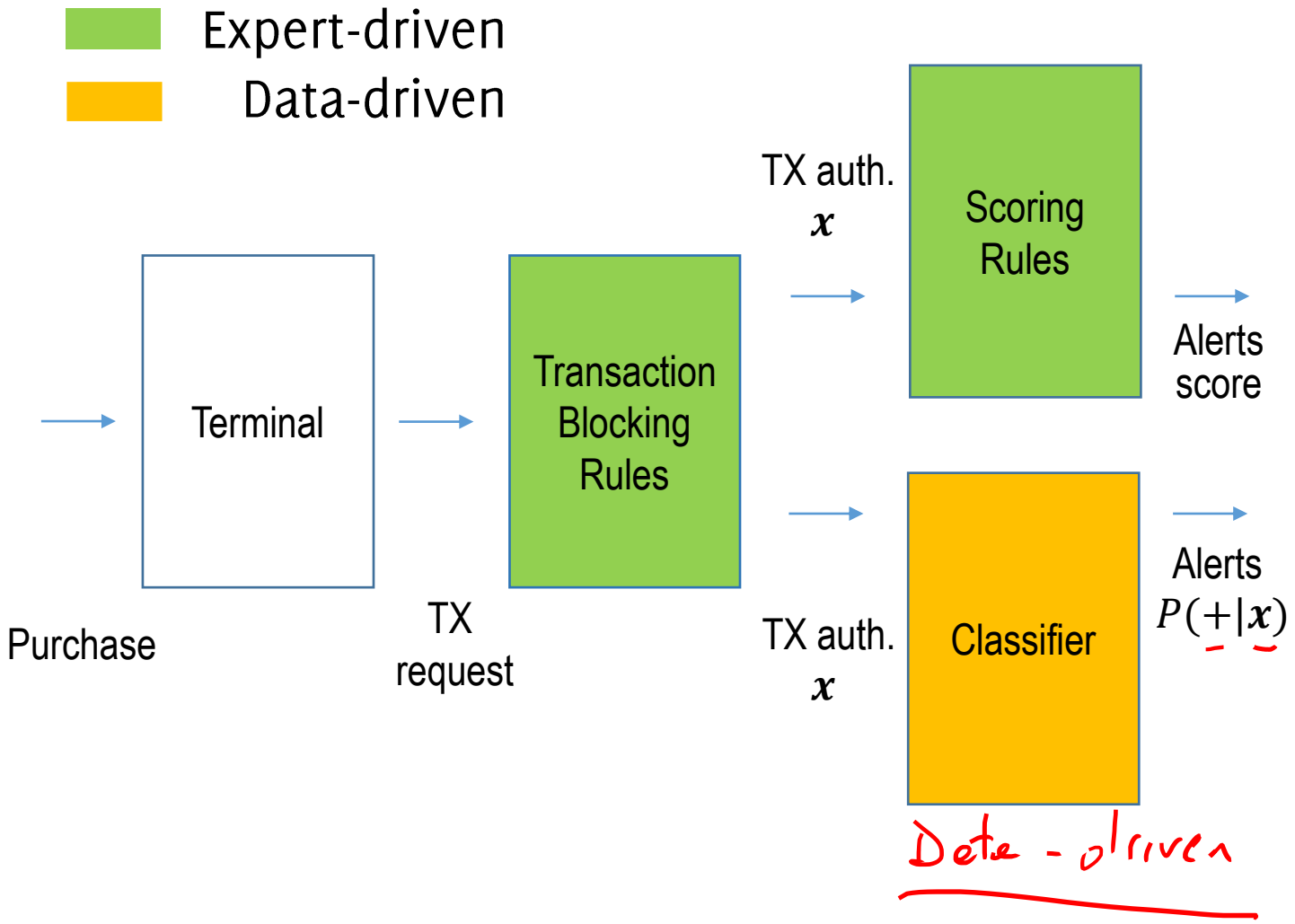


Fraudulent patterns can be directly **learned from data**, by means of a **data-driven model**. This should:

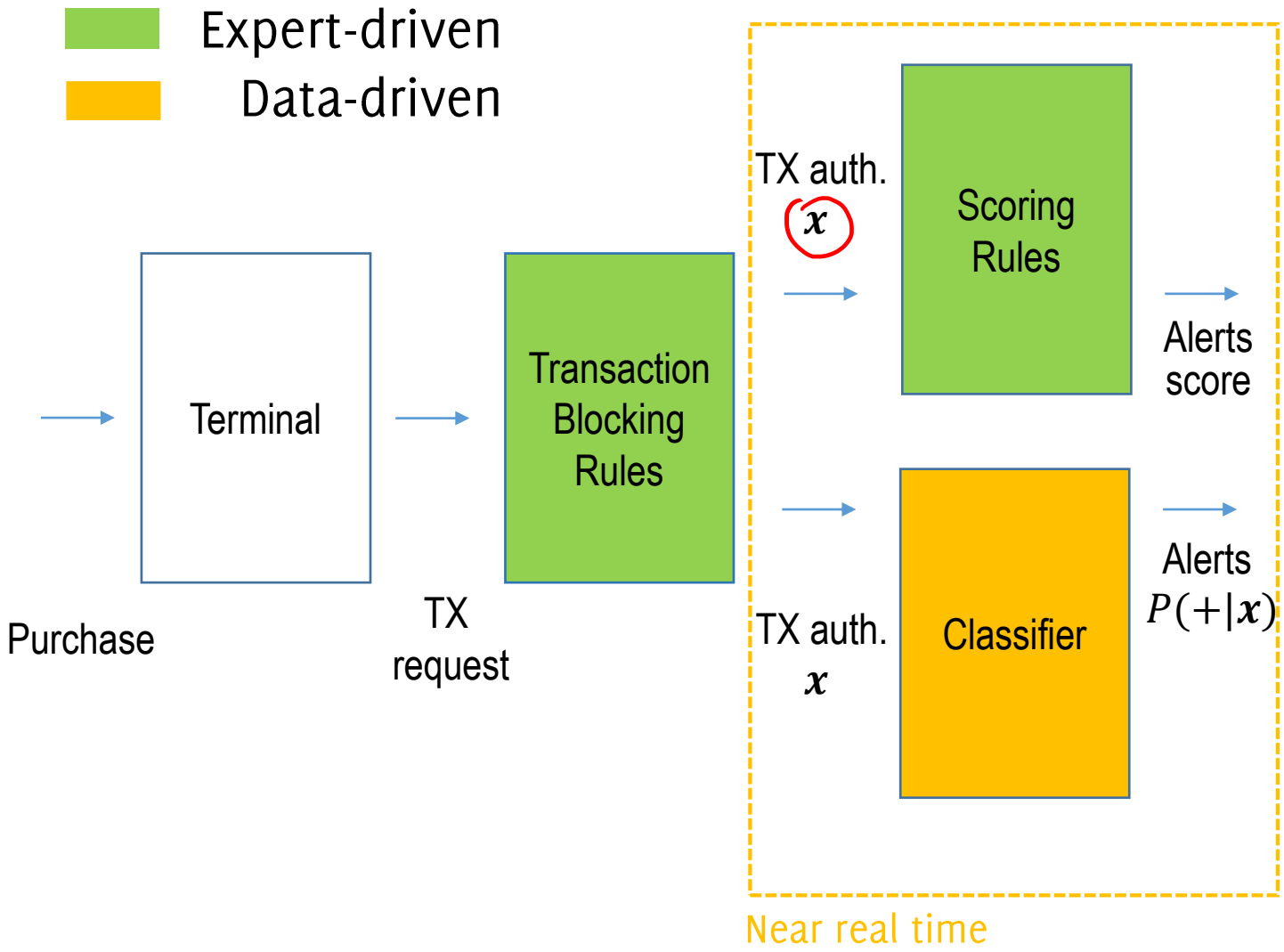
- Simultaneously analyze **several components** of the feature vector
- Uncover **complex relations among features** that cannot be identified by investigator

These relations can be meaningful for separating frauds from genuine transactions

Data-driven models in fraud detection

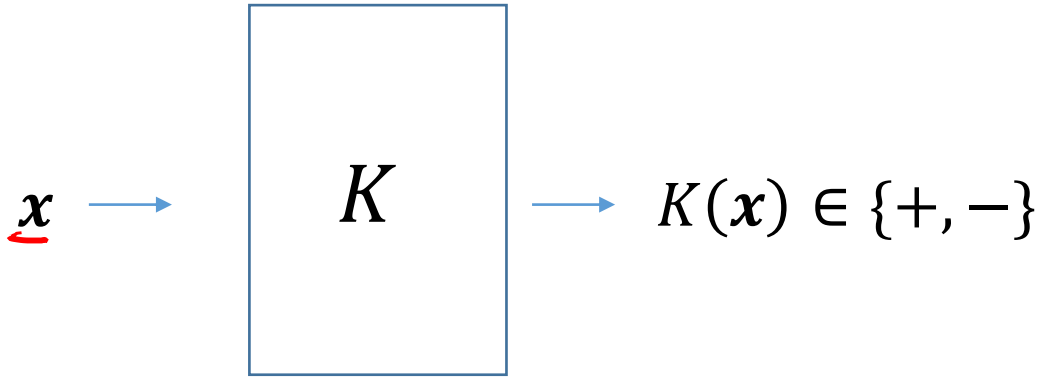


Data-driven models in fraud detection



Classifiers in Fraud Detection

In practice, the classifier K then can assign a label where the label $\hat{y} \in \{+, -\}$ i.e., $\{«fraud», «genuine»\}$ to each incoming feature vector $\boldsymbol{x}$



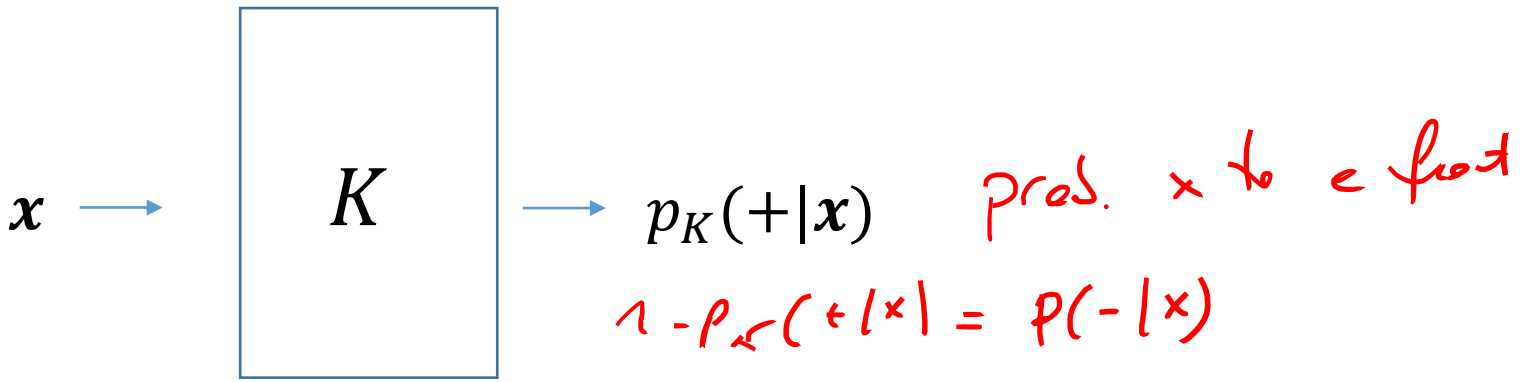
K considers transactions labeled as '+' as frauds

Classifiers in Fraud Detection

It is not feasible to alert all transactions labeled as frauds.

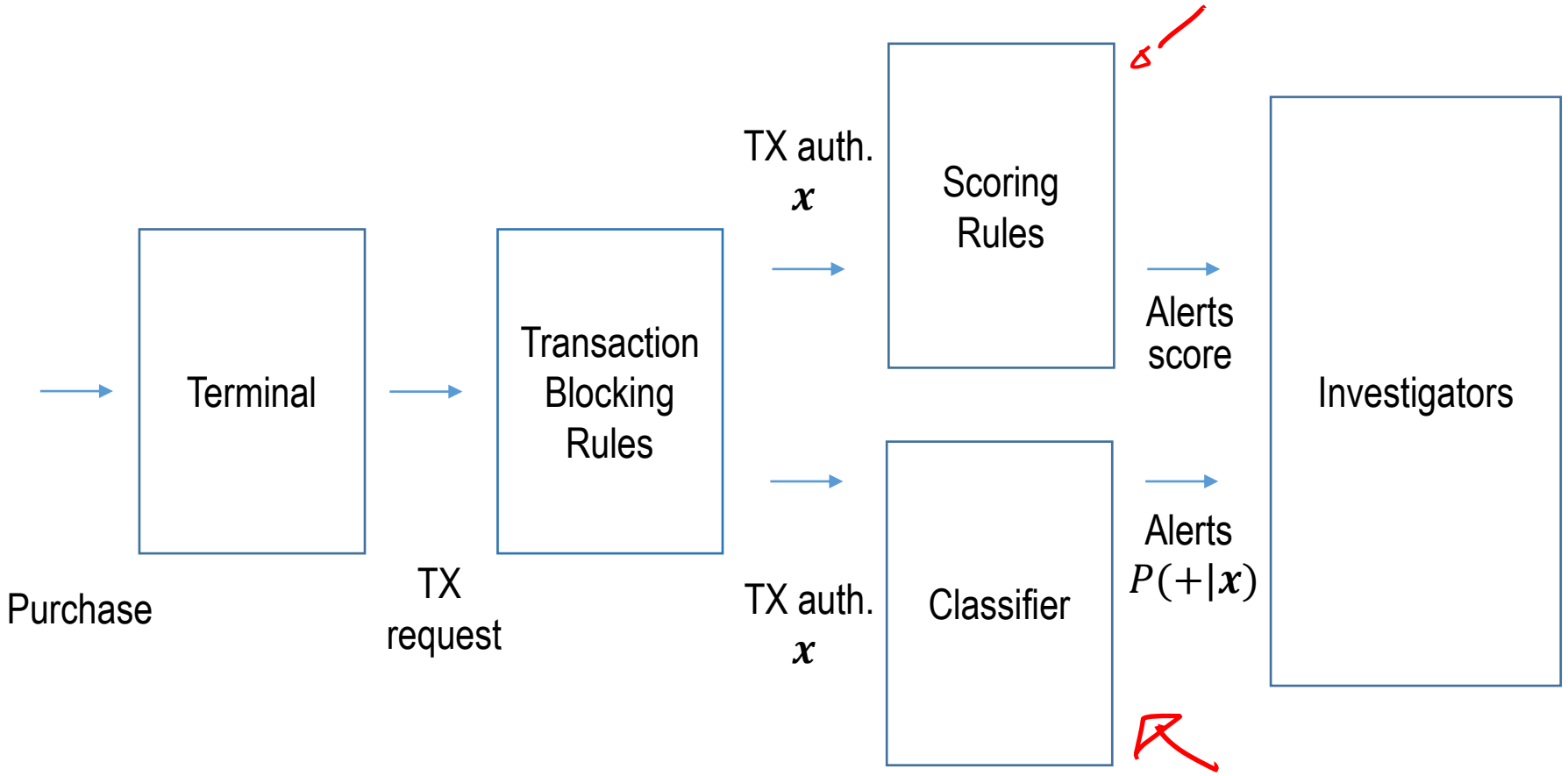
Only few transactions that are **very likely** to be frauds can be alerted.

Thus, the FDS typically consider $p_K(+|\boldsymbol{x})$, **an estimate of the probability** for $\boldsymbol{x}$ to be a fraud according to K



and only transactions yielding $p_K(+|\boldsymbol{x}) \approx 1$ raise an alert

Investigators



Investigators

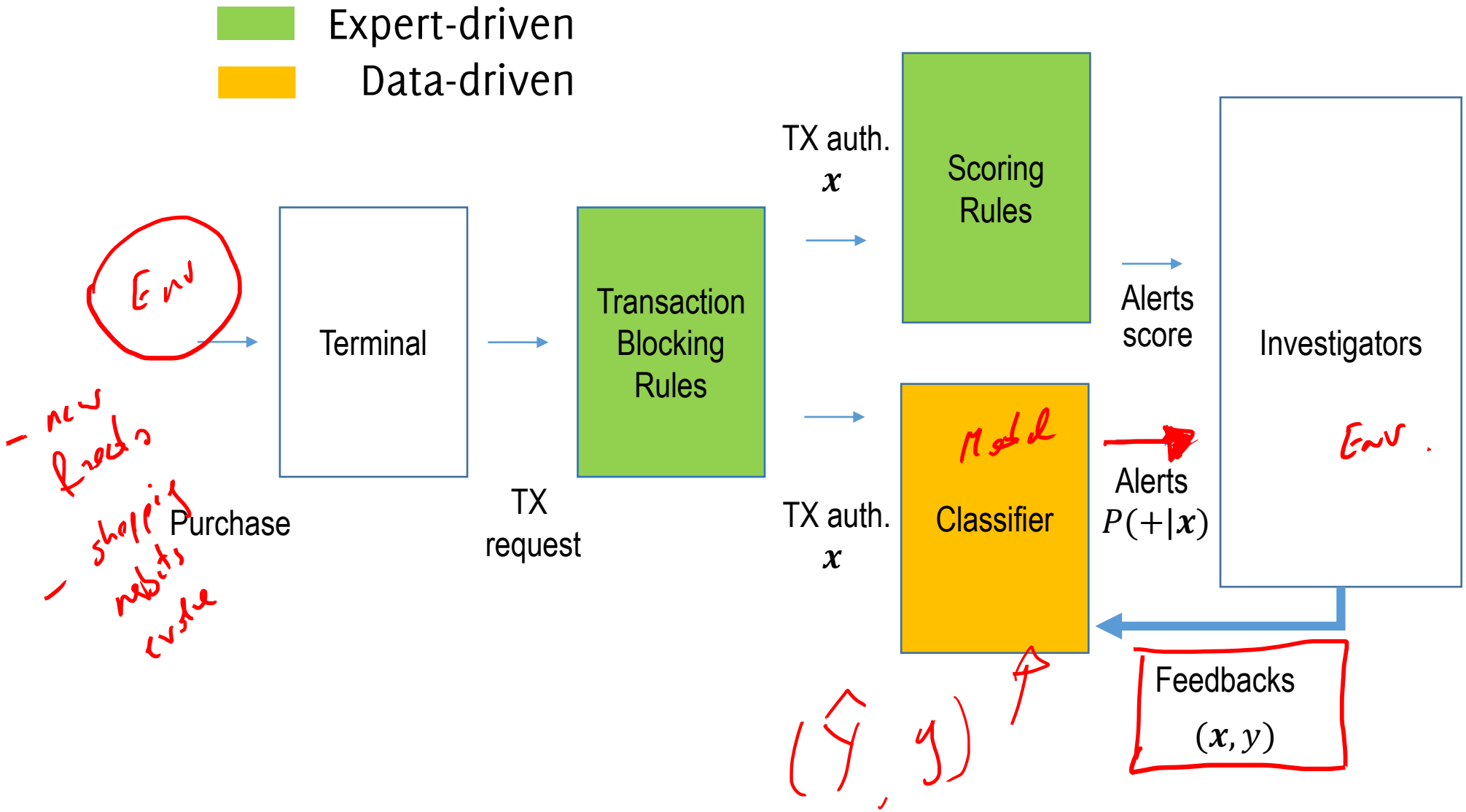
Investigators are **professionals** that are experienced in analyzing credit card transactions:

- they **design blocking/scoring rules**
- they **call cardholders** to check whether alerts correspond to frauds
- as soon as they detect a fraud, they block the card
- they annotate the **true label** of checked transactions

The labels associated to transactions comes in the form of **feedbacks** and can be used to re-train/update K

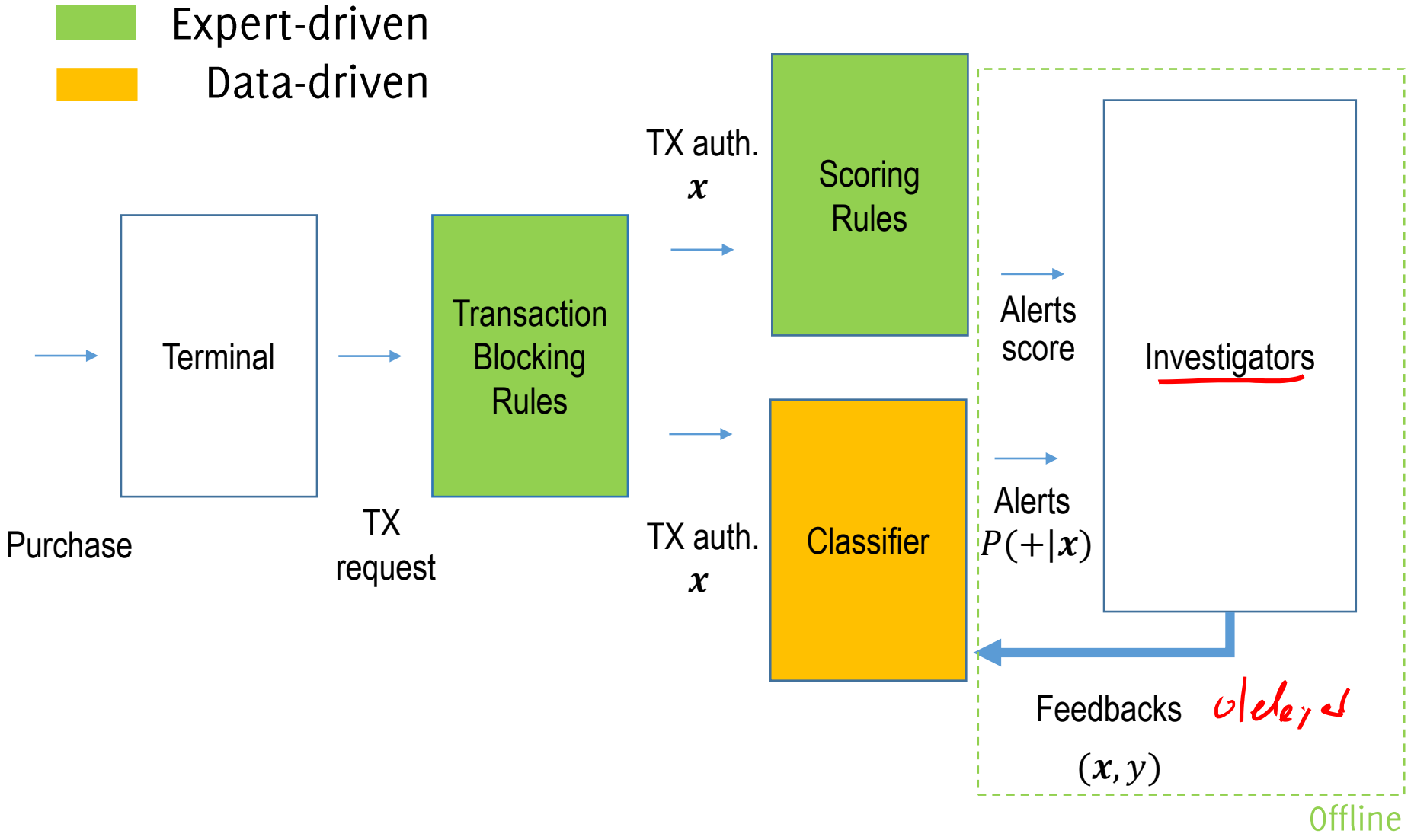
Given the limited number of investigators, the large number of transactions, the multiple sources of alerts, etc … it is important to provide **very precise alerts**

Investigators' feedback: Supervised Information



ò

Investigators' feedback: Supervised Information



ò

Problem Formulation

Classification over Datastreams

Classification Over Datastreams

The problem: classification over a potentially infinitely long **stream of data**

$$X = \{\boldsymbol{x_0}, \boldsymbol{x_1}, \dots, \}$$

Data-generating process $\mathcal{X}$ generates tuples $(\boldsymbol{x}_t, y_t) \sim \phi_{\boldsymbol{x},y}$

- $\boldsymbol{x}_t$ is the observation at time t (e.g., $\boldsymbol{x}_t \in \mathbb{R}^d$)
- y_t is the associated label which is (often) unknown ($y_t \in \Lambda$)

Classification Over Datastreams

Typical assumptions:

- Inputs are Independent and identically distributed (**i.i.d.**)

$$(\boldsymbol{x_t}, y_t) \sim \phi_{\boldsymbol{x},y}$$

- An **initial training set** $TR = \{(\boldsymbol{x}_0, y_0), \ldots, (\boldsymbol{x}_n, y_n)\}$ is provided for learning K
- TR contains data generated in stationary conditions

The classifier is trained (i.e. its parameters are estimated) by optimizing some loss function (e.g. binary cross-entropy, hinge loss, …) over TR.

A **stationary condition of $\boldsymbol{x}$** is denoted as **concept.**

Classification error

A classifier estimates for each input $\boldsymbol{x}$ a label $\hat{y}$ (*)

$$\hat{y} = K(\boldsymbol{x})$$

And – hopefully – it often happens that $\hat{y} = y$.

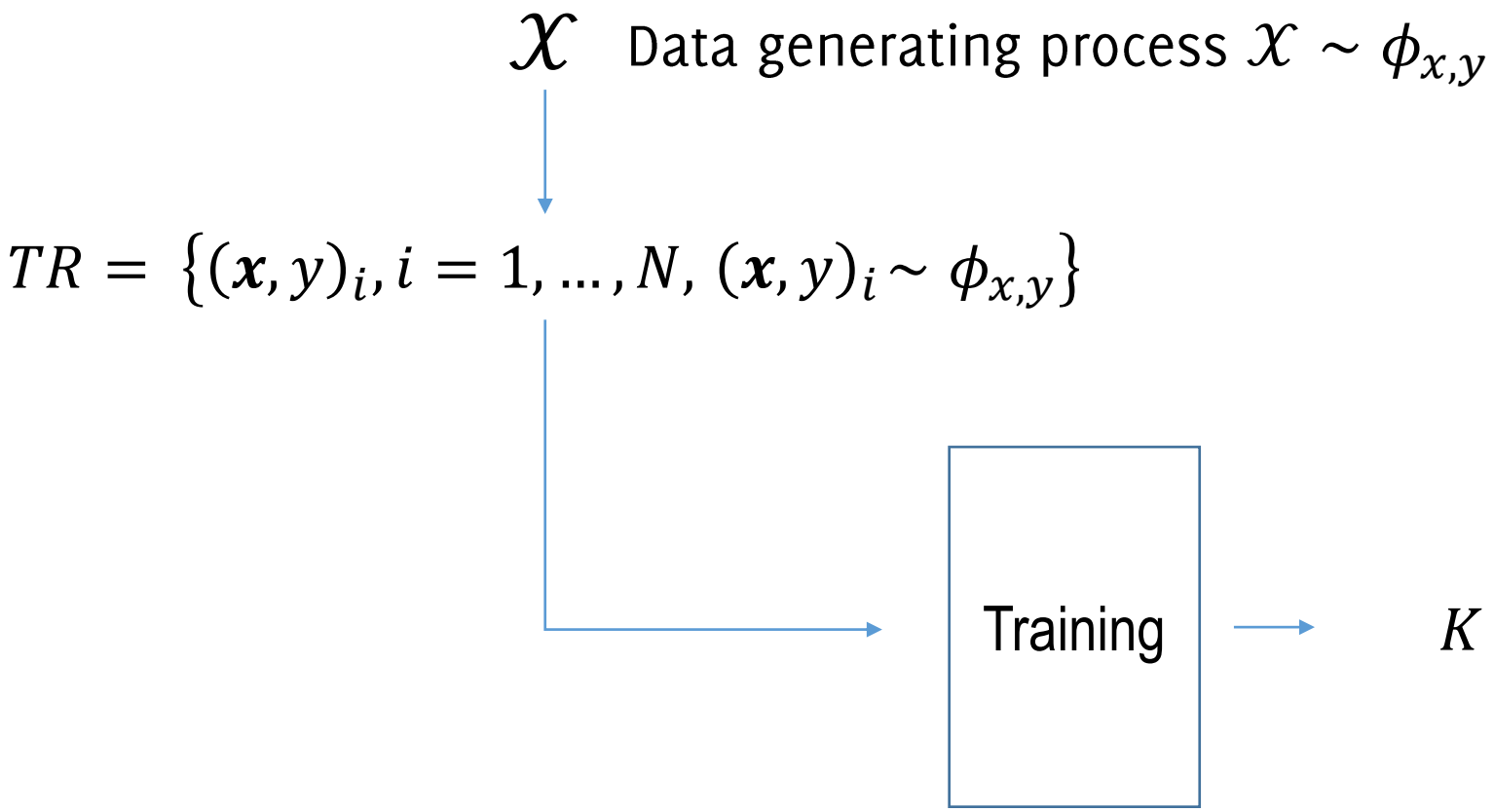
Here, we consider the classification error to measure how good a learned model K matches the distribution $\phi_{\boldsymbol{x},y}$, namely

$$e = \#\{\hat{y}_i \neq y_i, i \in R\}$$

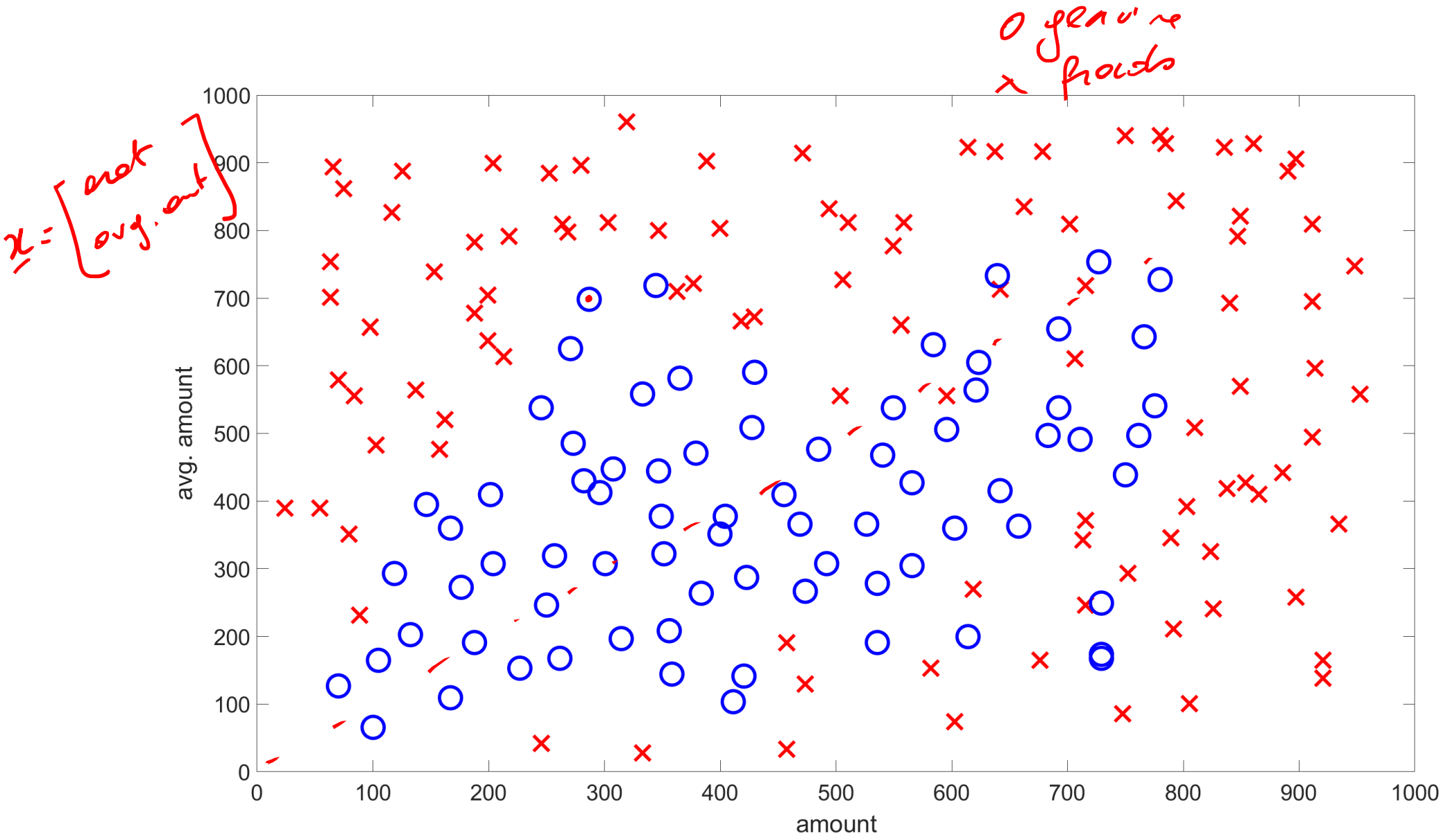
being R «a reference set» for assessing the error

(*) Classifiers typically return the posterior probability of each class

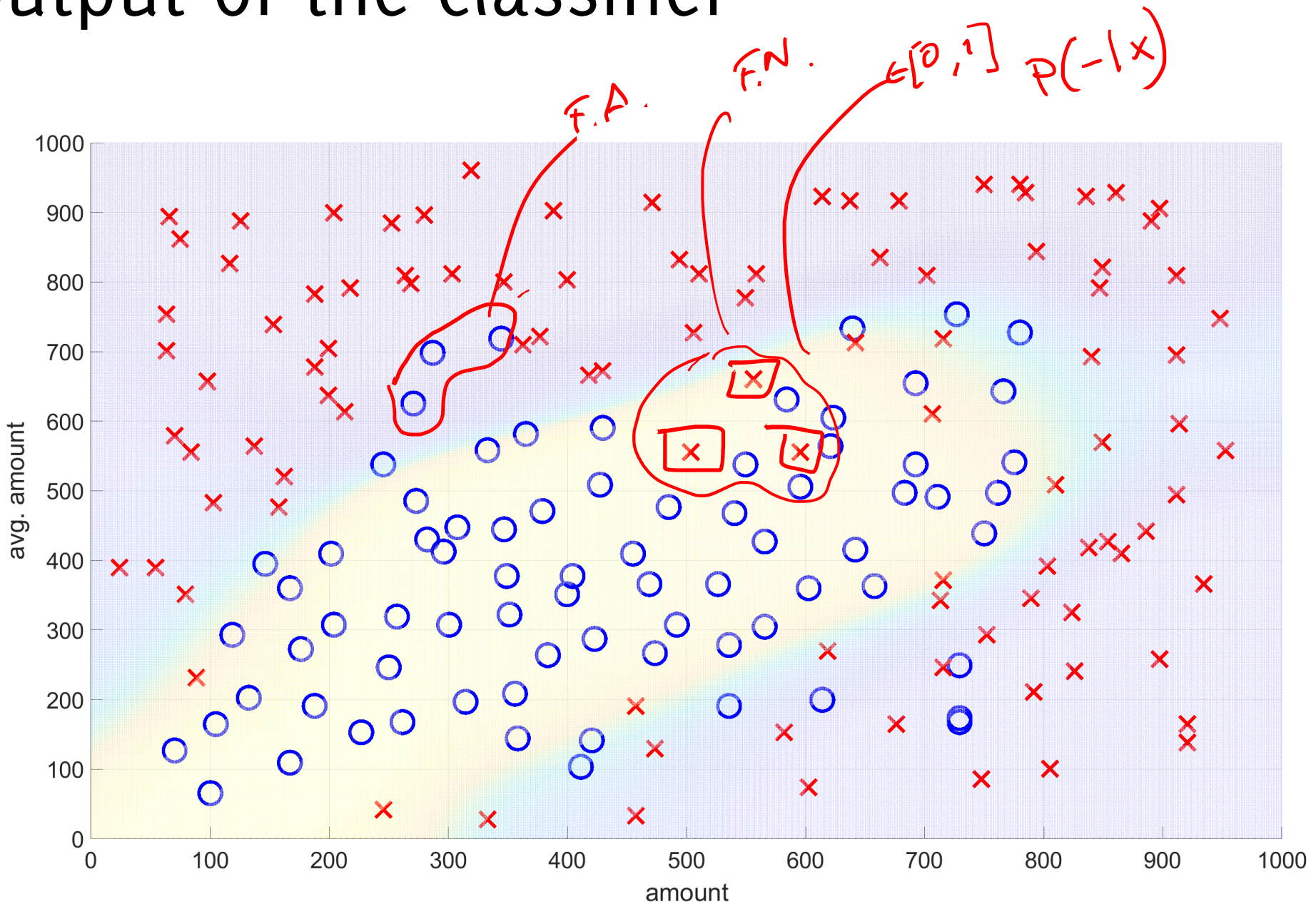
Training the Classifier



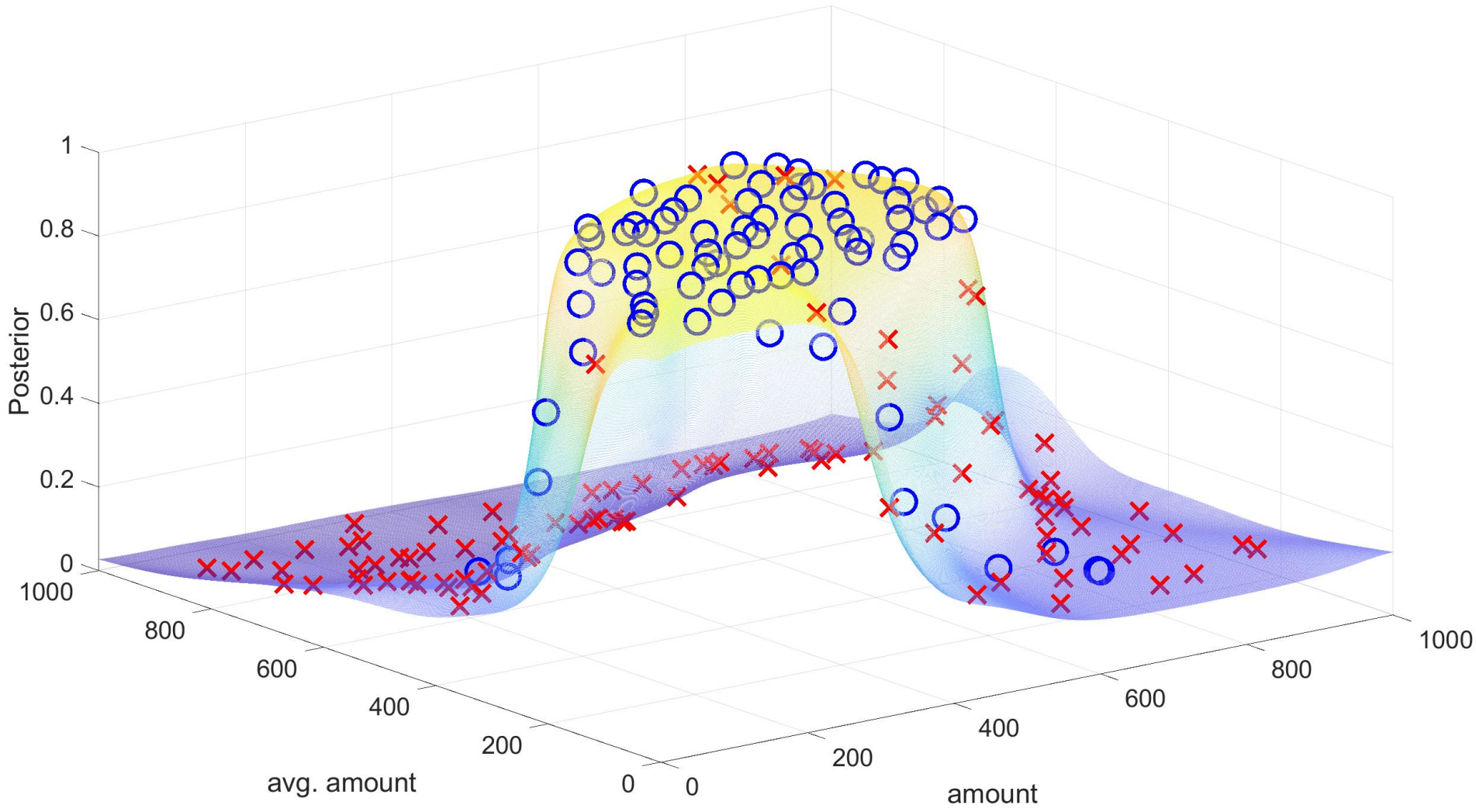
Training Set



The output of the classifier

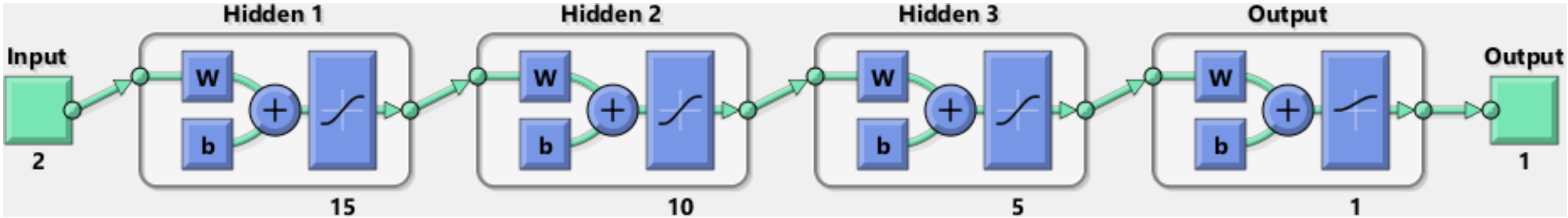


The output of the classifier

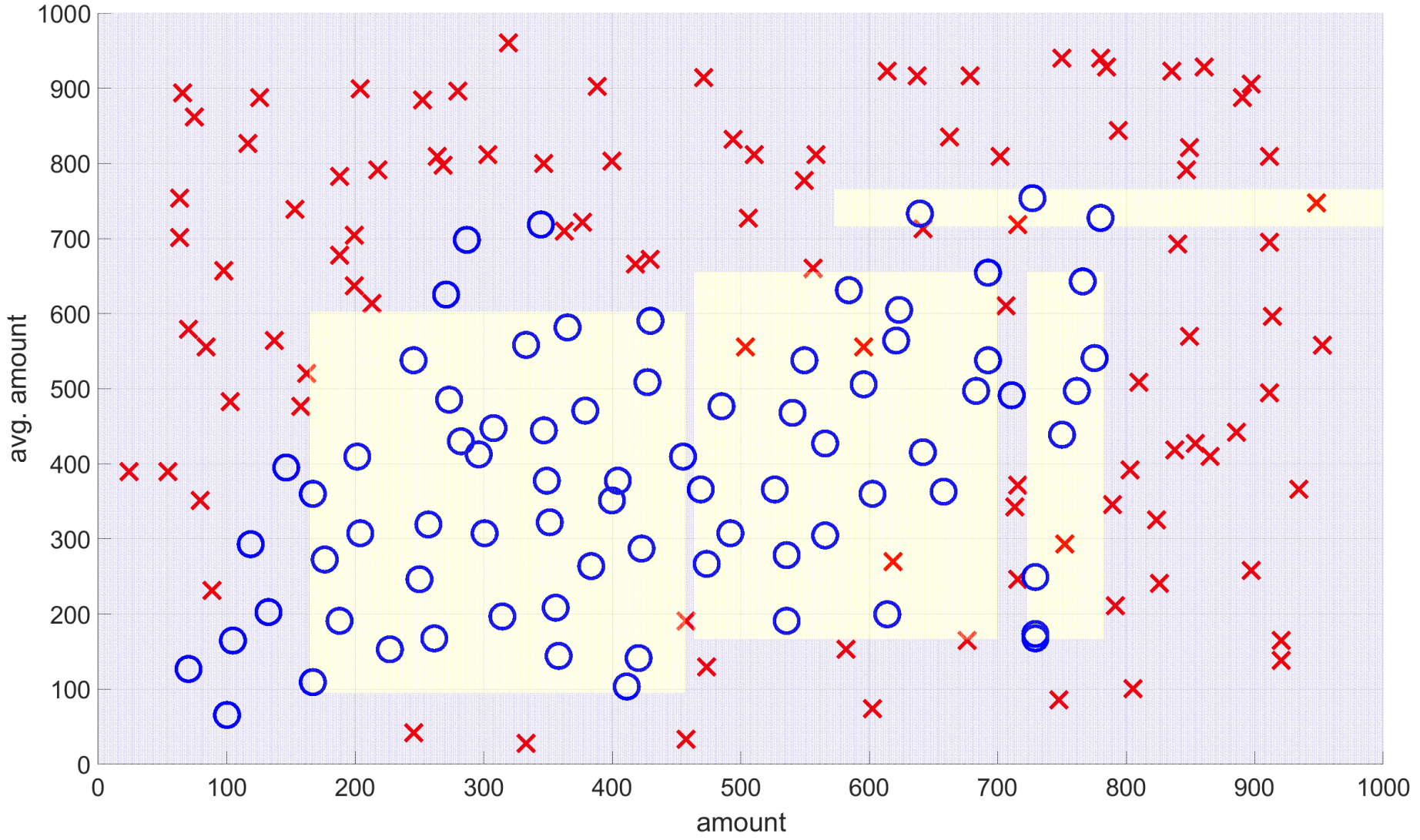


Bonacchi, Trovò

Btw… that was a Neural Network



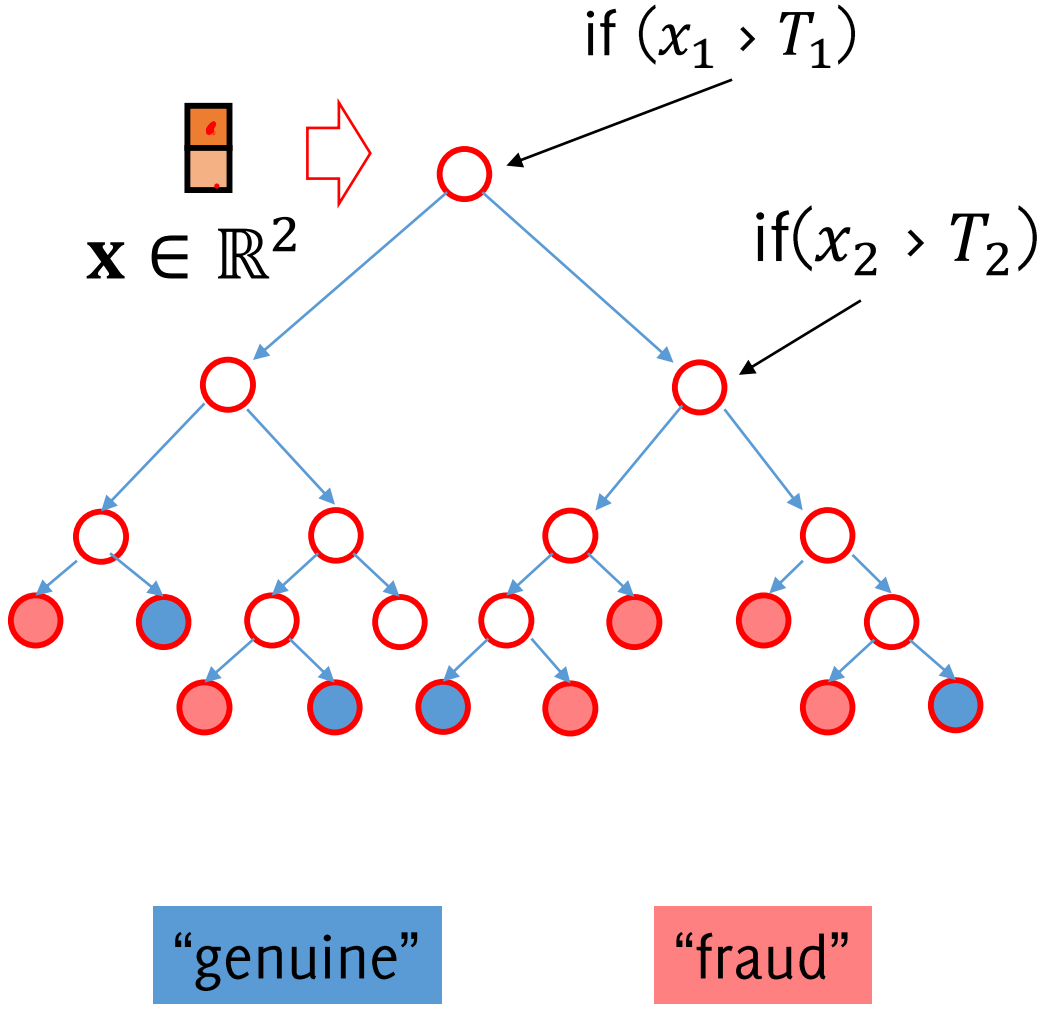
The output of another classifier



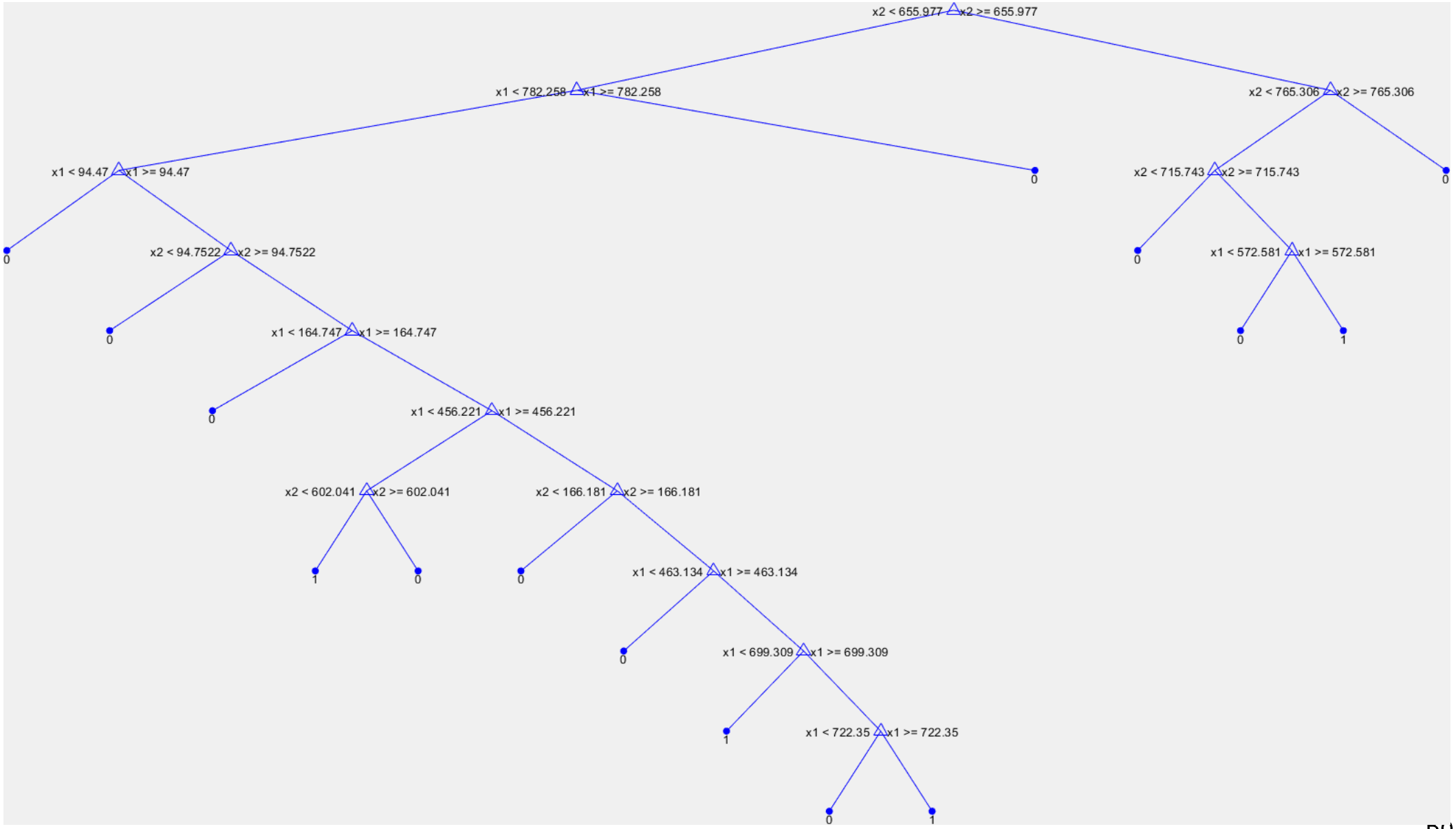
Decision Tree

The classifier has a few patameters:

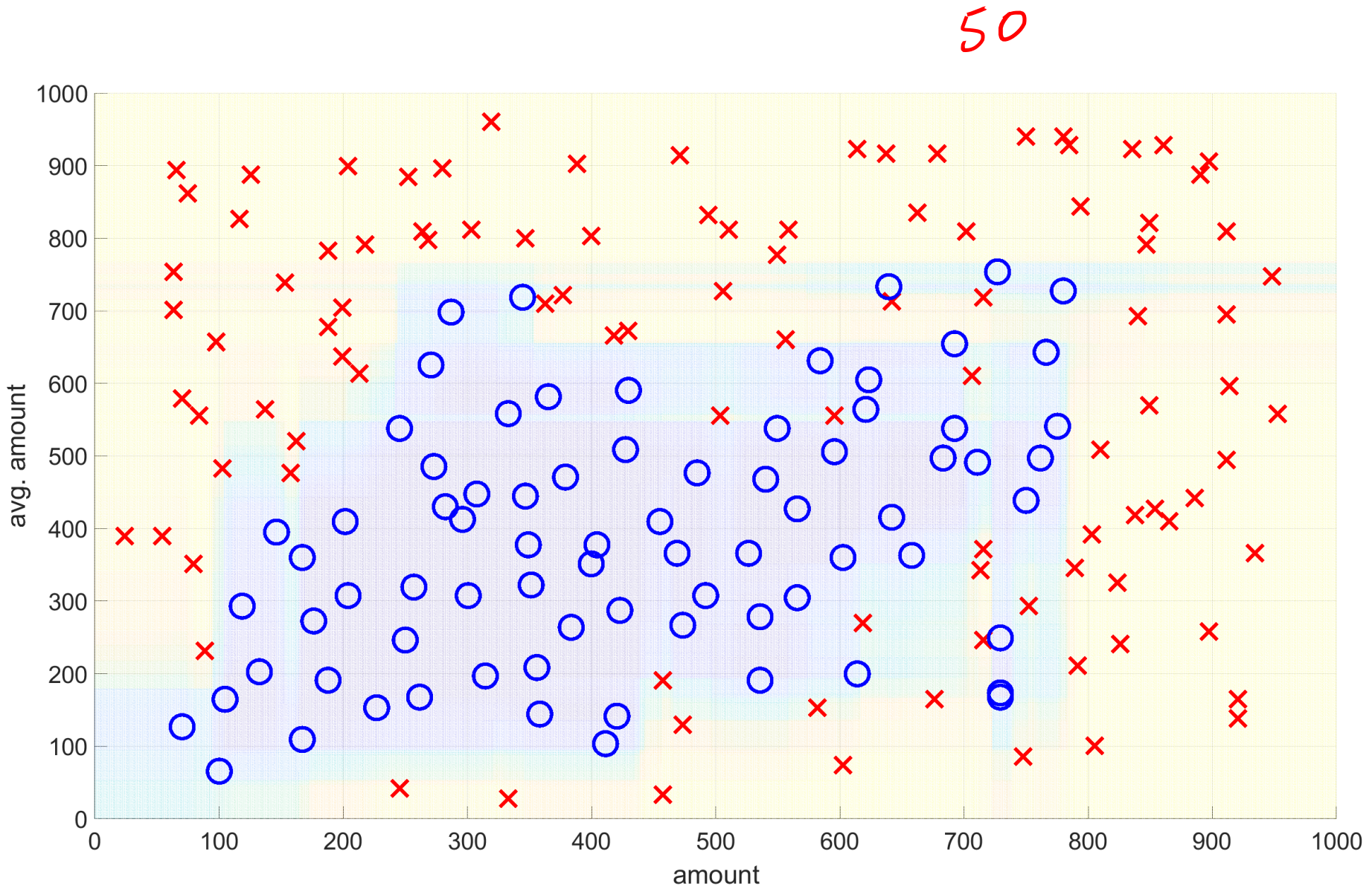
- The **splitting criteria**
- The **splitting thresholds T_i**



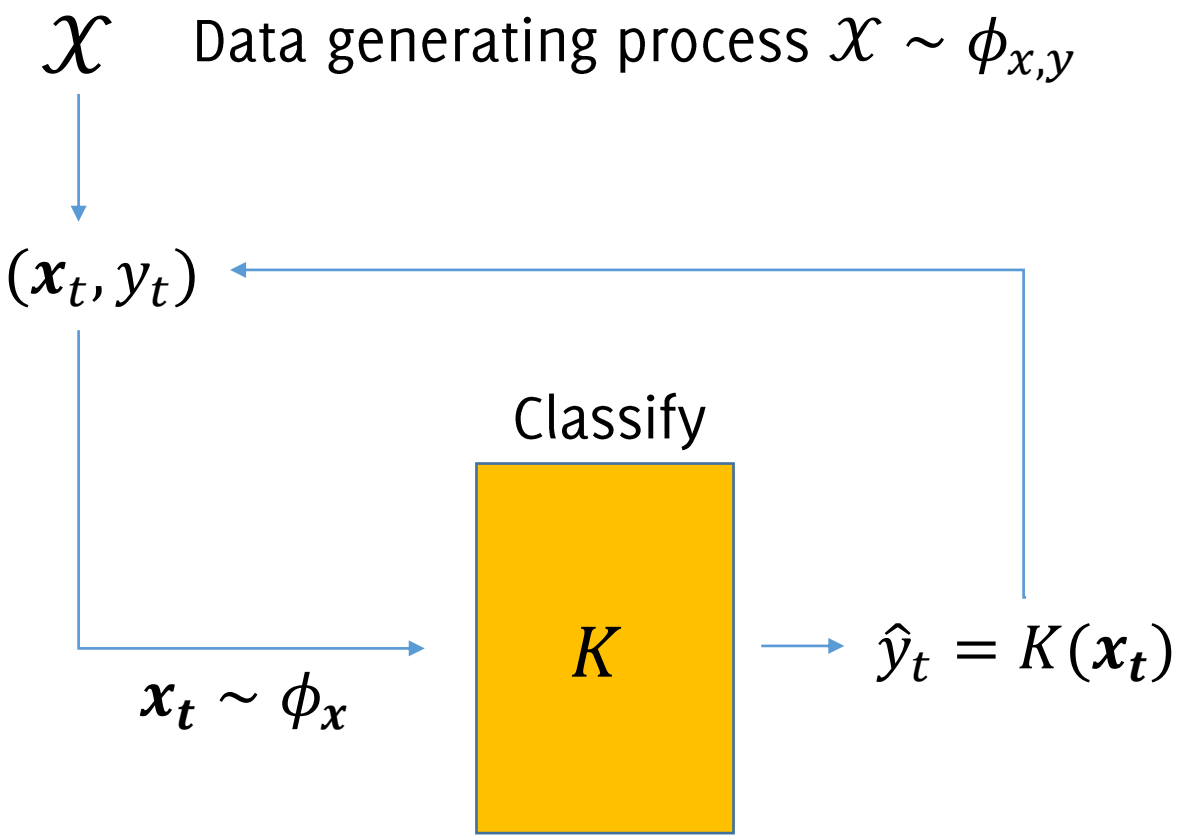
Yes, that was a decision tree



… better with many trees (random forest)



Classification



Supervised Information

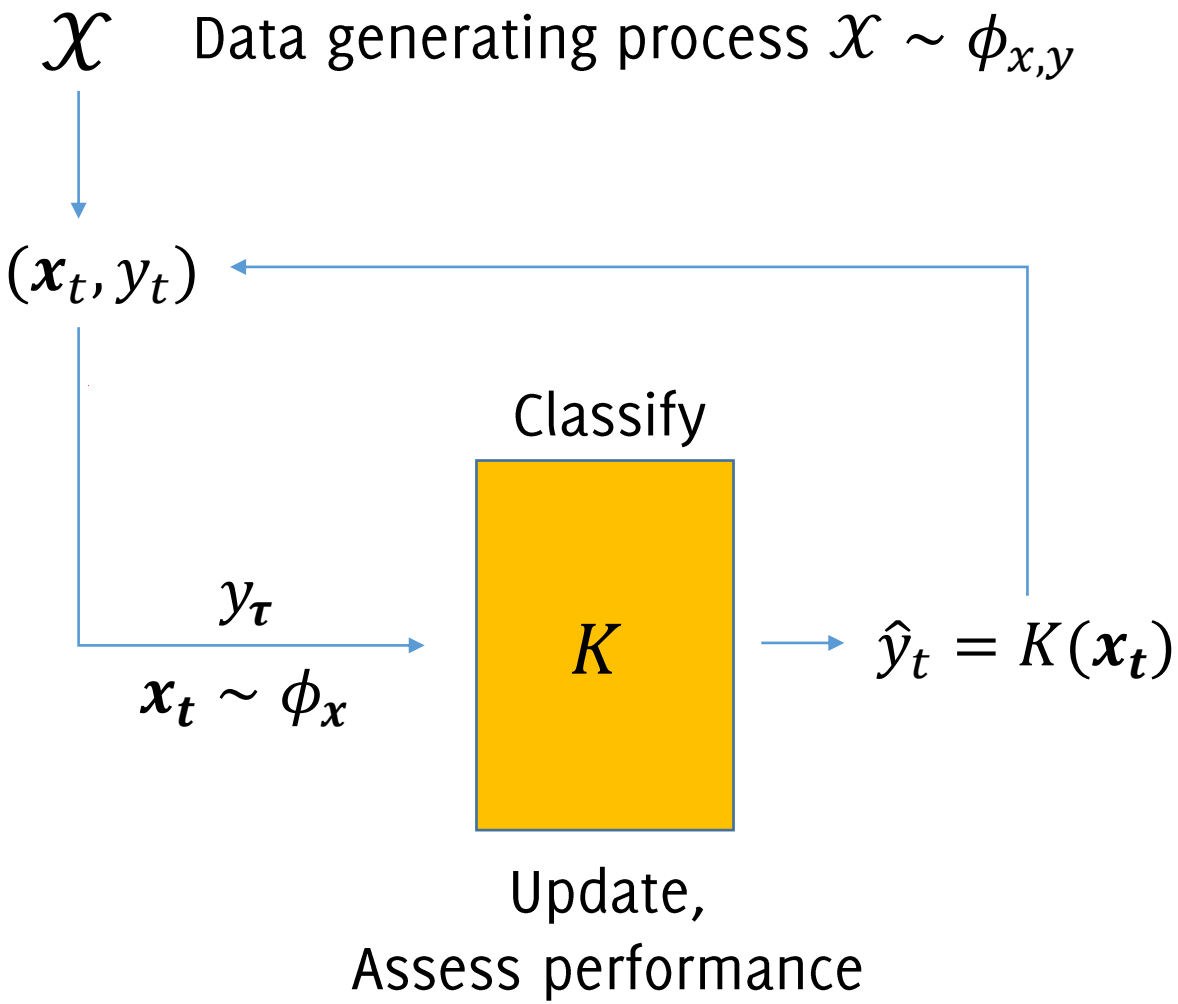
Gather all the supervised annotations $\{(\boldsymbol{x_t}, y_t) \sim \phi_{x,y}\}$

- assess performance of K

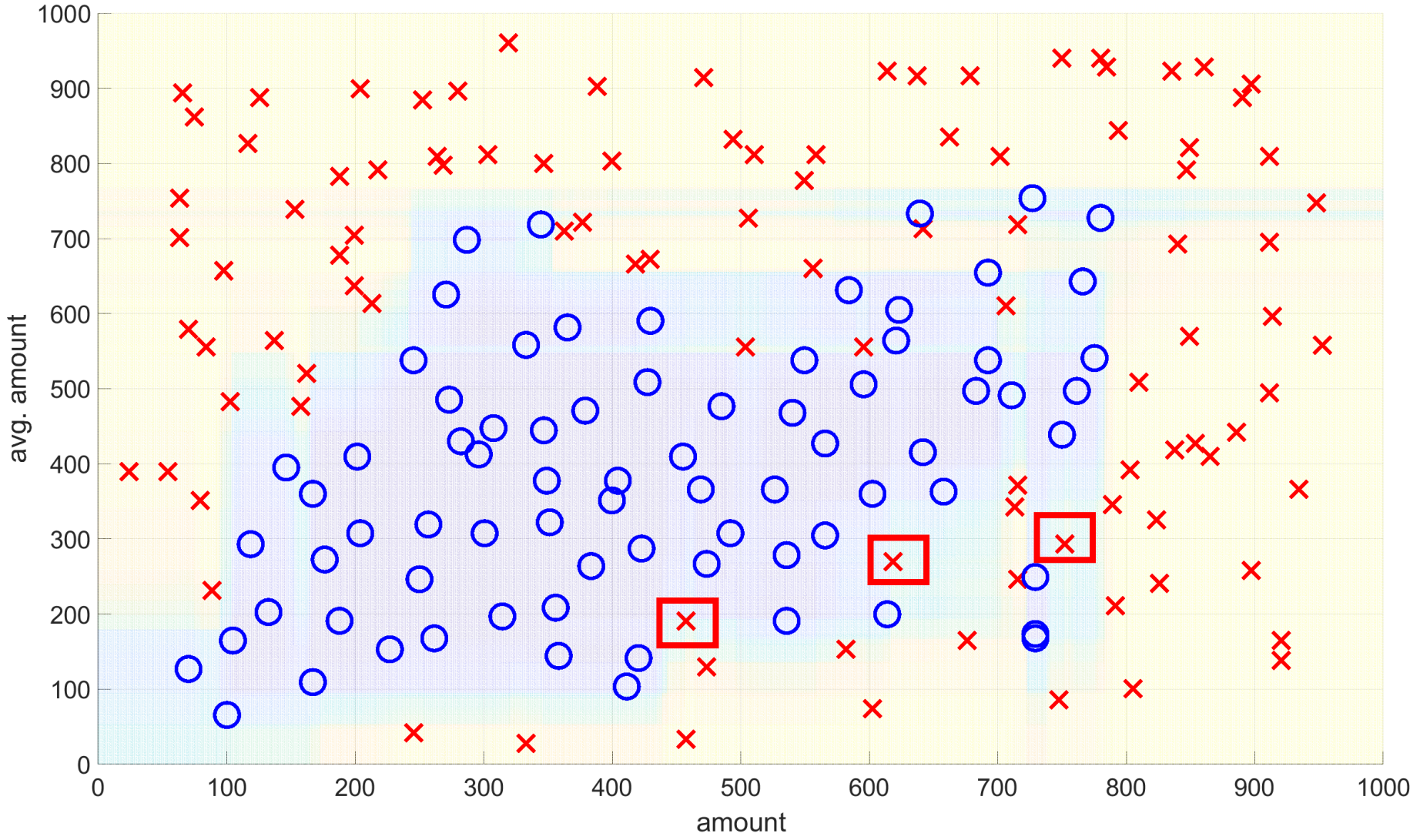
$$p(T) = \frac{1}{T}\sum_t e_t$$

$$\text{where } e_t = \begin{cases} 0, & \text{if } \hat{y}_t = y_t \\ 1, & \text{if } \hat{y}_t \neq y_t \end{cases}$$

- use supervised information for updating K



Classification Errors



Learning in Nonstationary (Streaming) Environments

The Problem Formulation

Boracchi, Trovò

Concept Drift

Unfortunately, in **the real world**, datastream $\mathcal{X}$ might **change unpredictably** during operation.

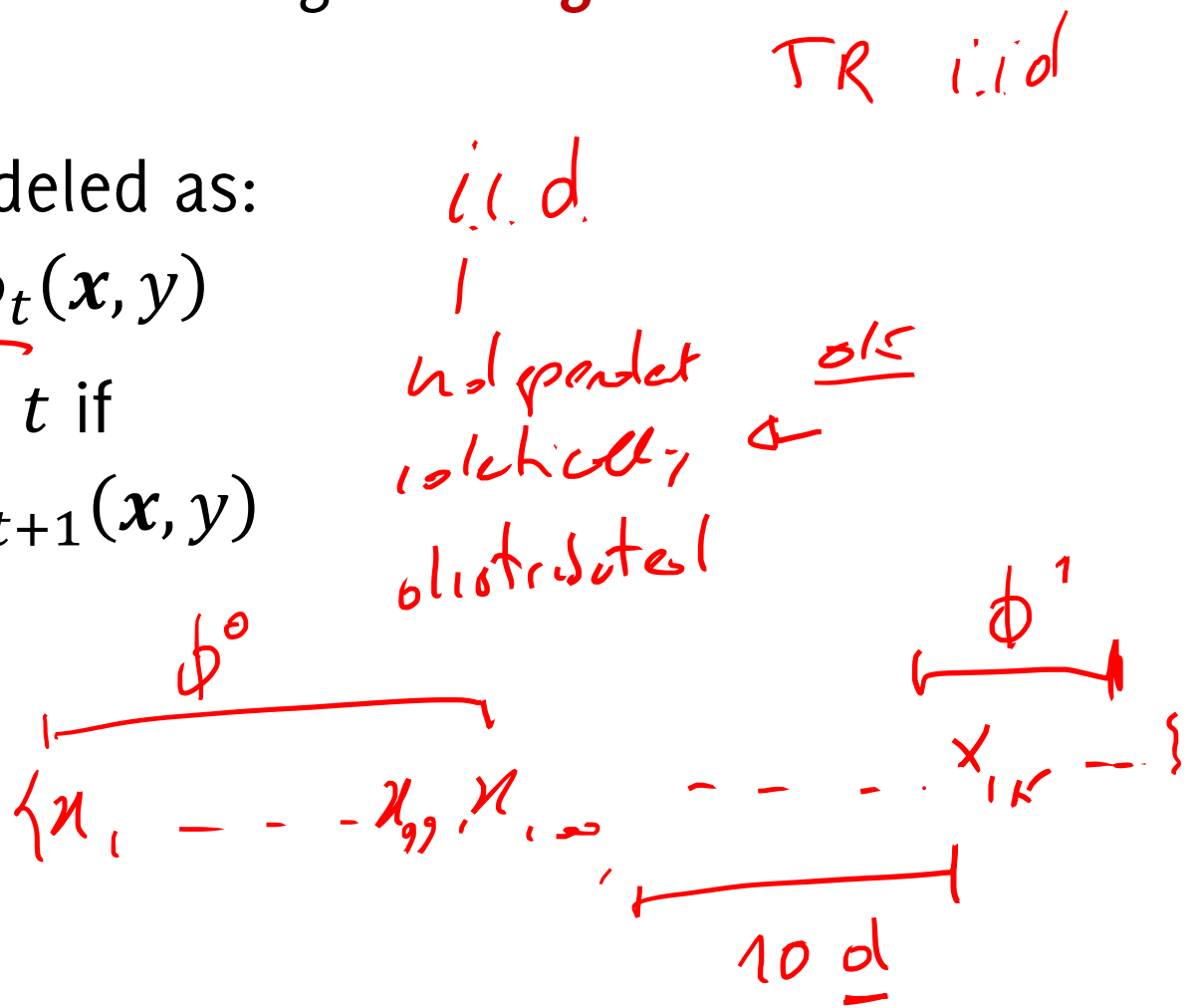
The data generating process is then modeled as:

$$(\boldsymbol{x}_t, y_t) \sim \phi_t(\boldsymbol{x}, y)$$

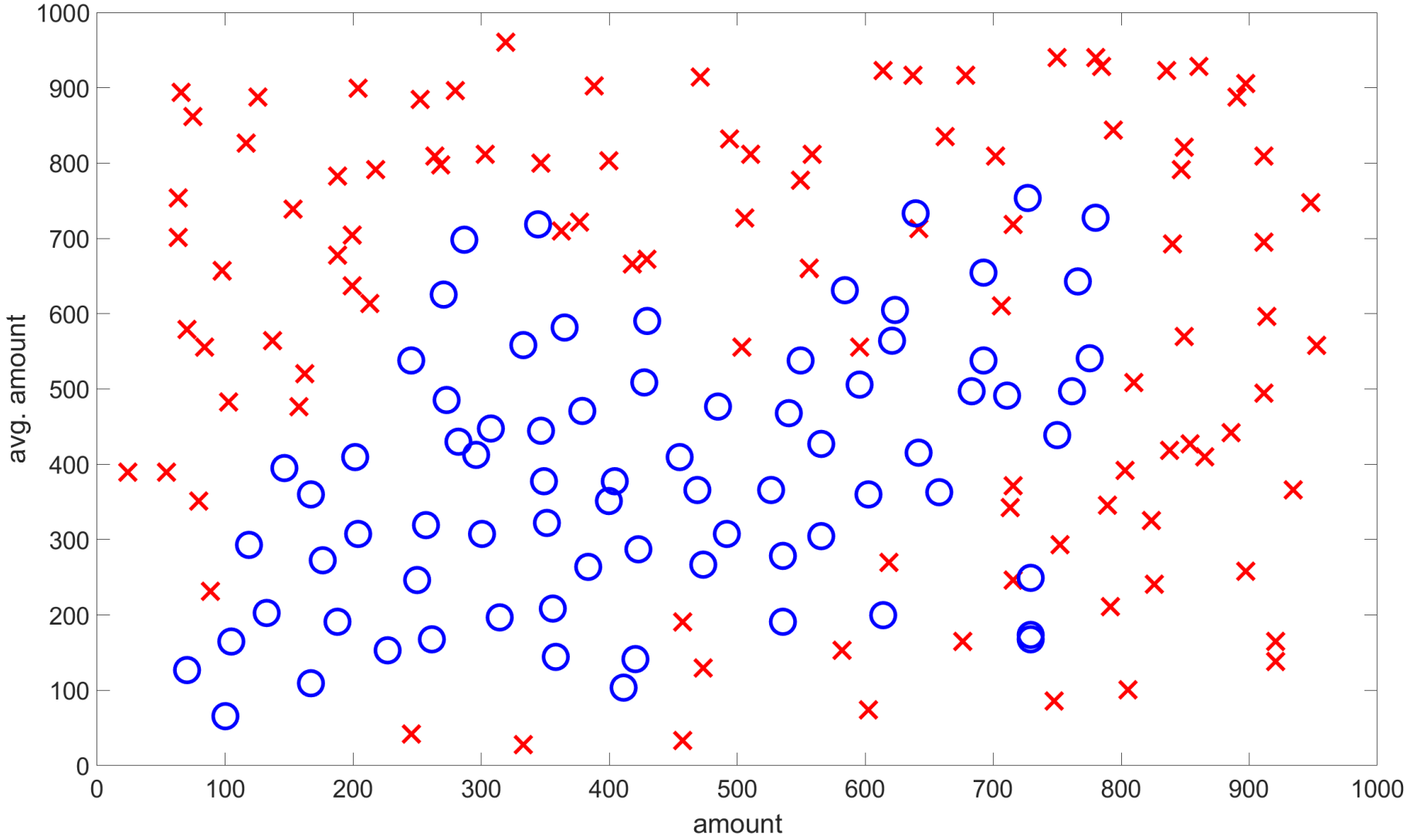
We say that **concept drift** occurs at time t if

$$\phi_t(\boldsymbol{x}, y) \neq \phi_{t+1}(\boldsymbol{x}, y)$$

We also say $\mathcal{X}$ becomes **nonstationary**.

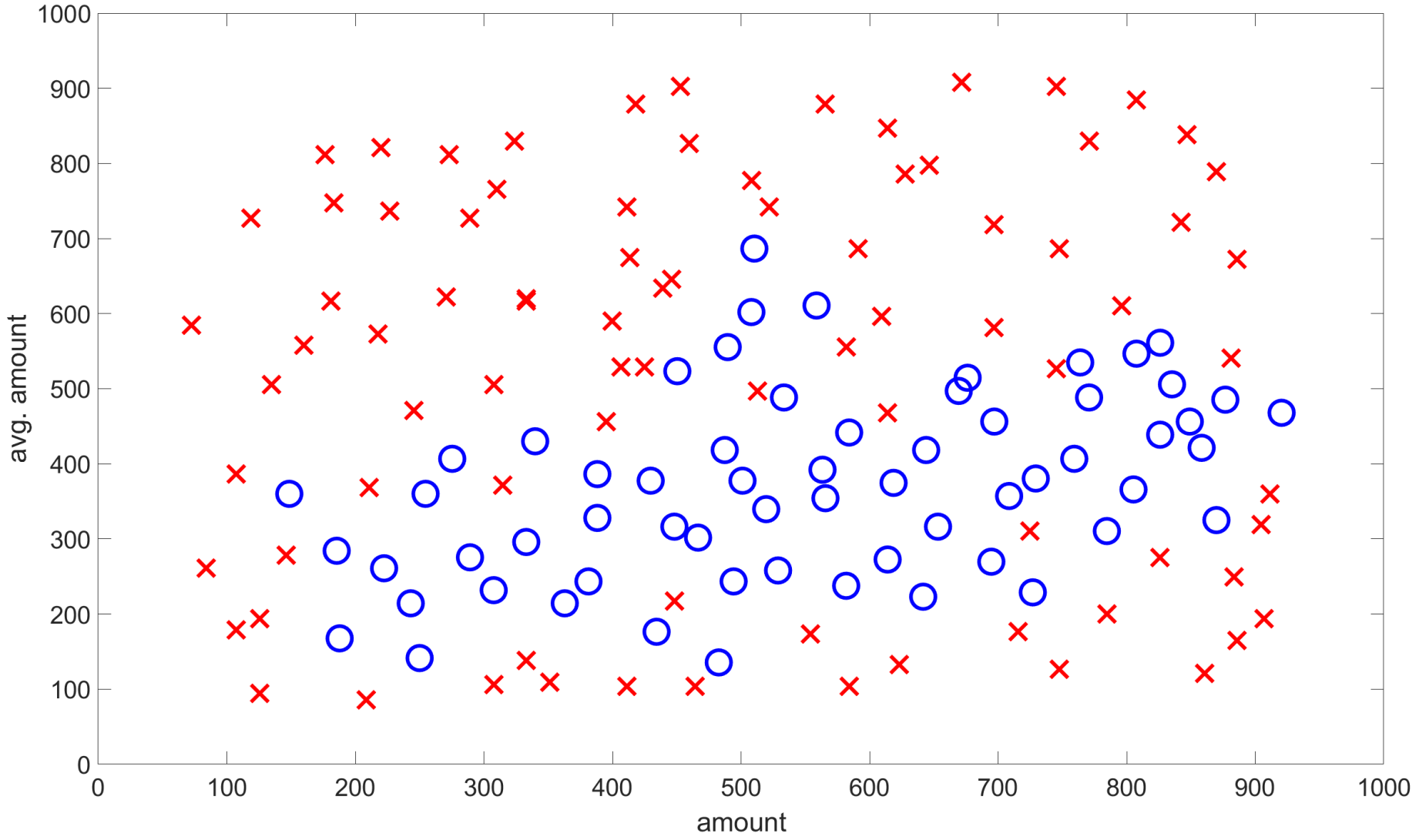


Distribution Changes



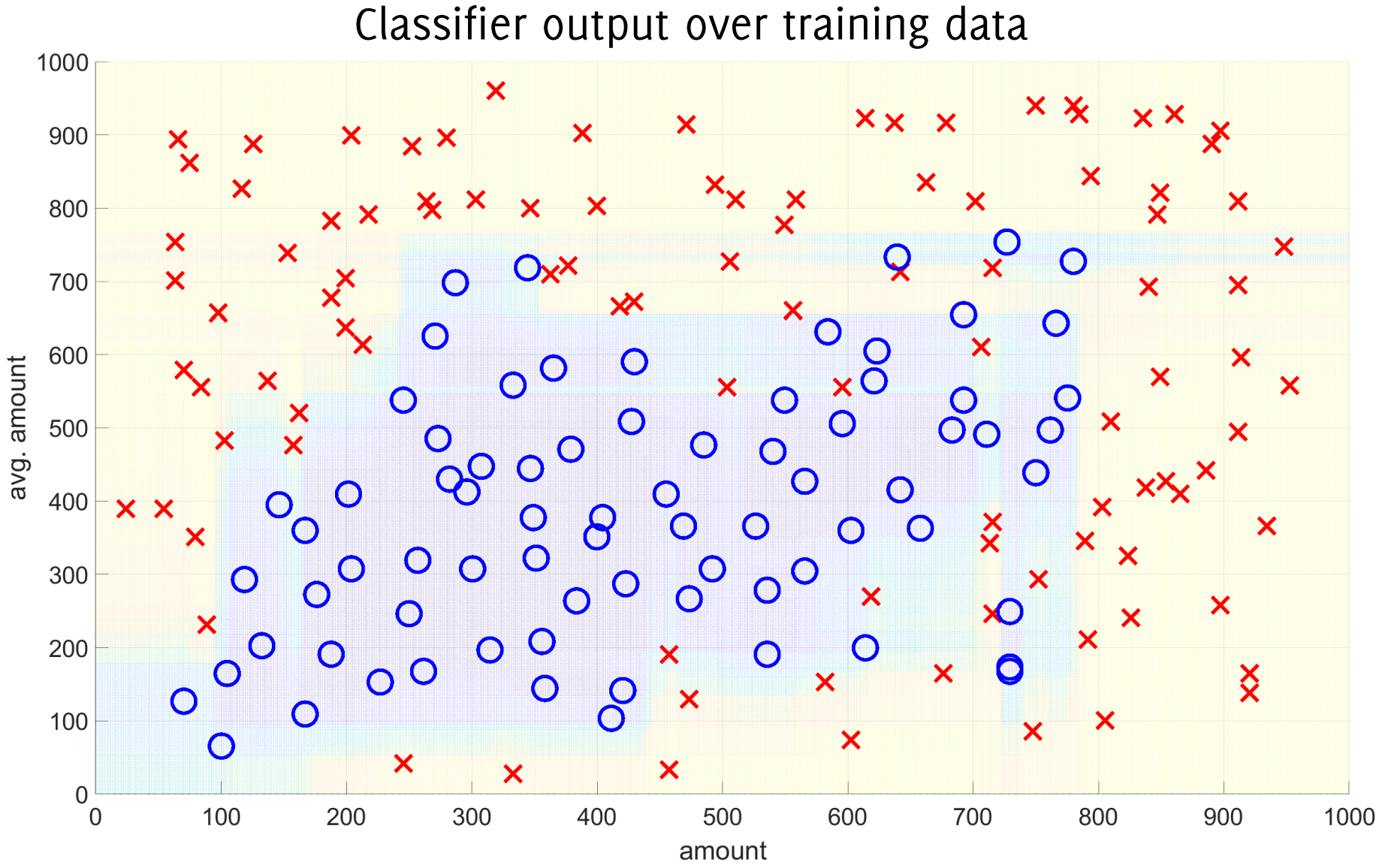
$\phi^0_{x,y}$

Distribution Changes



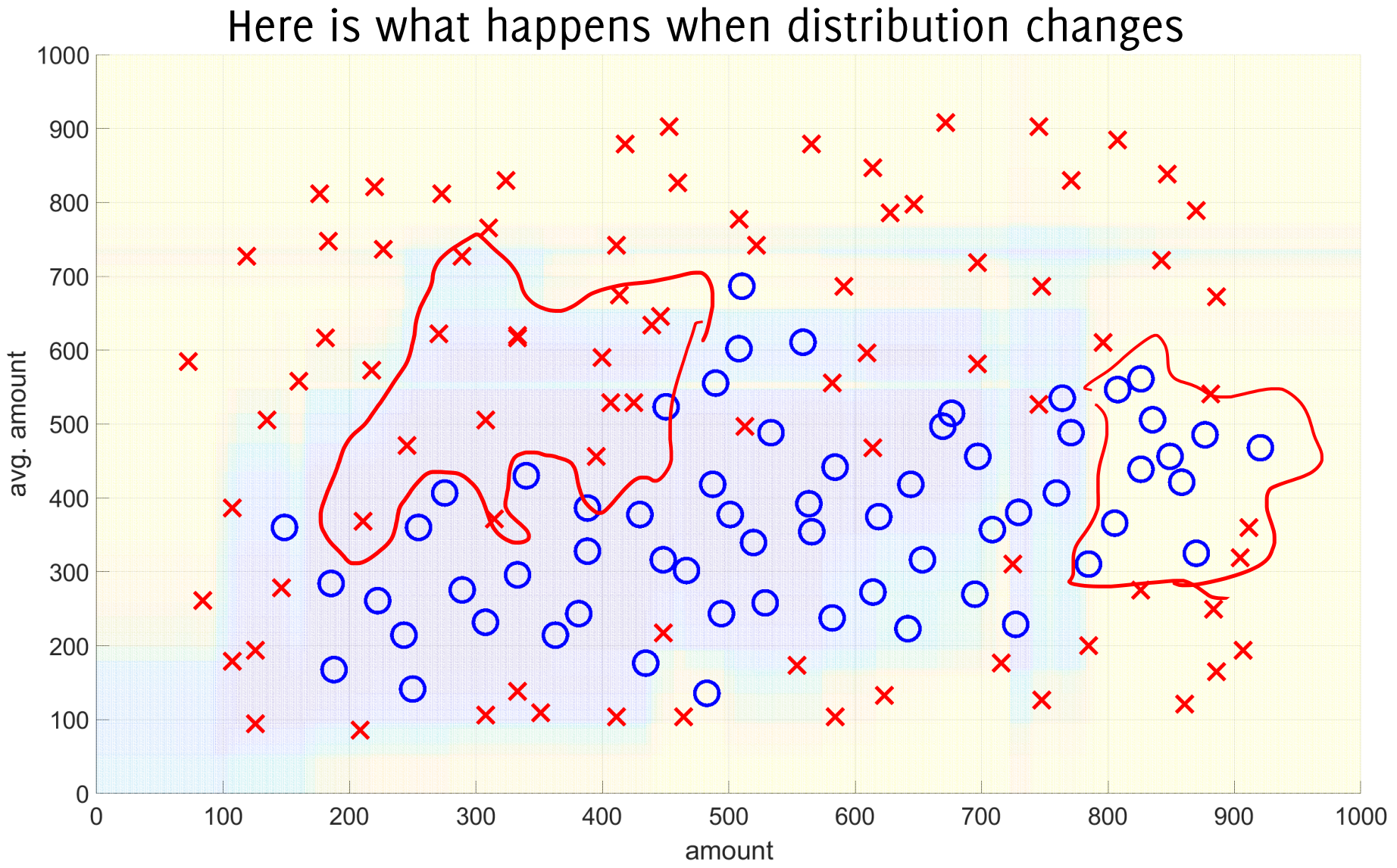
$\phi^1_{x,y}$

What happens when $\phi^0_{x,y} \rightarrow \phi^1_{x,y}$?



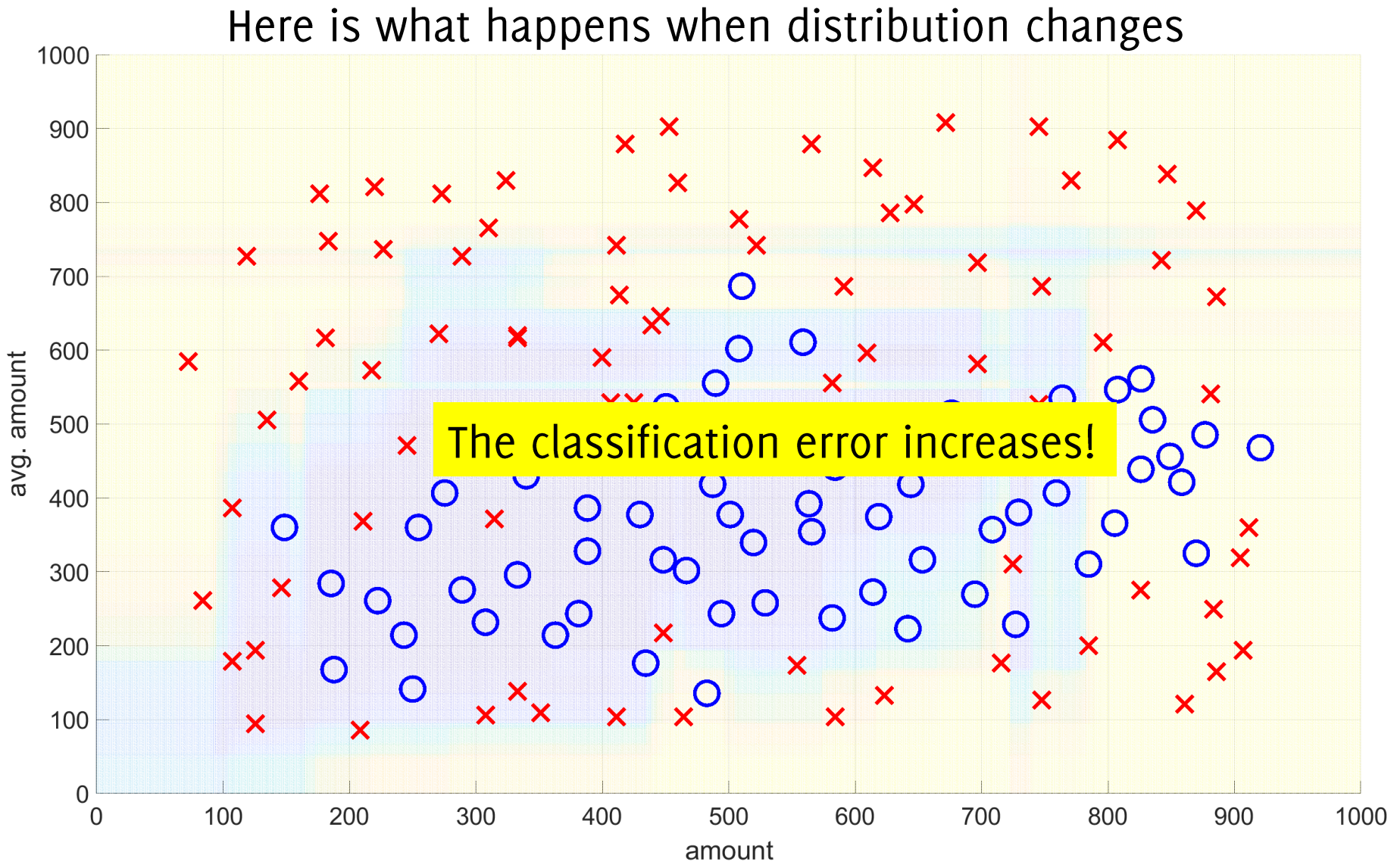
$\phi^0_{x,y}$

What happens when $\phi^0_{x,y} \rightarrow \phi^1_{x,y}$?



$\phi^1_{x,y}$

What happens when $\phi^0_{x,y} \rightarrow \phi^1_{x,y}$?



$\phi^1_{x,y}$

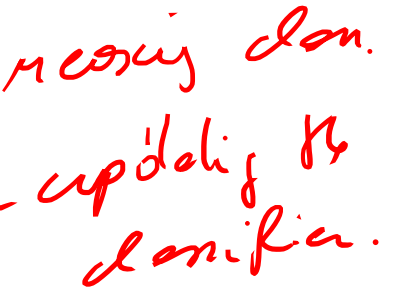
Problem formulation learning in NSE

The task: learn an **adaptive classifier** K_t to predict labels

$$\hat{y}_t = K_t(\boldsymbol{x}_t)$$

in an **online manner** having a low **classification error** over time:

$$\frac{1}{T}\sum_{t=1}^{T} e_t \text{, where } e_t = \begin{cases} 0, & \text{if } \hat{y}_t = y_t \\ 1, & \text{if } \hat{y}_t \neq y_t \end{cases}$$



This classifier should also operate when the distribution generating the input data changes.

Concept Drift Taxonomy

Drift Taxonomy

Drift taxonomy according to two characteristics:

1. **What** is changing?

$$\phi_{x,y}(x, y) = \phi_{y|x}(y|x)\ \phi_x(x)$$

Drift might affect $\phi_{y|x}(y|x)$ and/or $\phi_x(x)$

- Real concept drift affects $\phi_{y|x}(y|x)$
- Virtual concept drift affects $\phi_x(x)$

2. **How** does process change **over time**?
   - Abrupt
   - Gradual
   - Incremental
   - Recurring

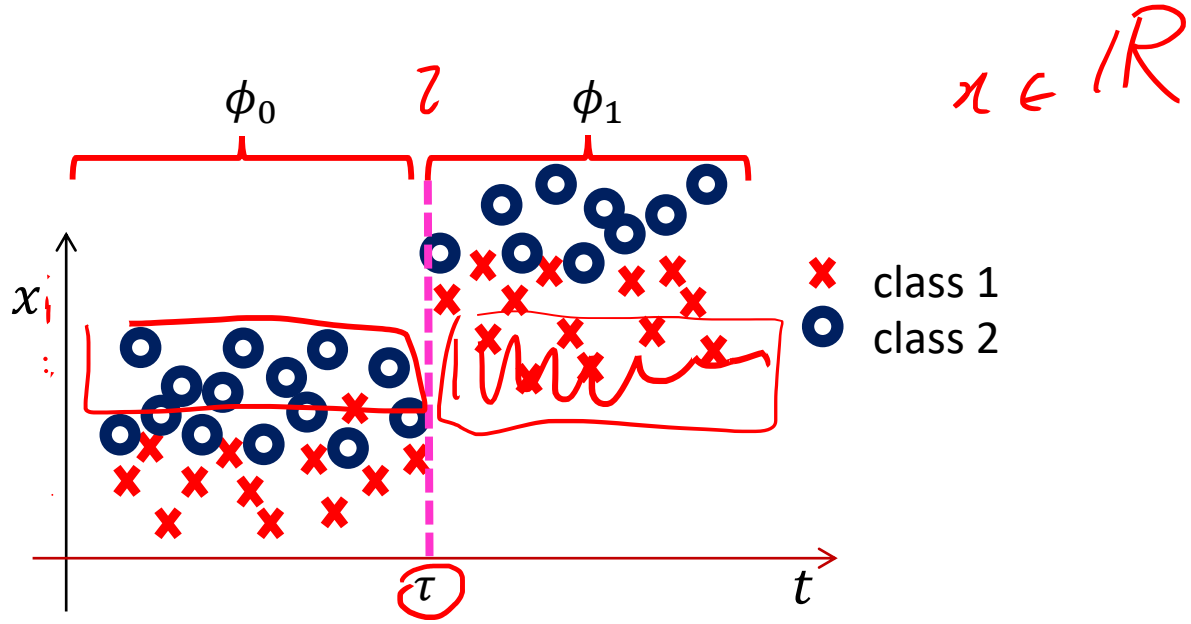
Drift Taxonomy: What Is Changing?

Real Drift

$$\phi_{\tau+1}(y|\boldsymbol{x}) \neq \phi_{\tau}(y|\boldsymbol{x})$$

affects $\phi_t(y|\boldsymbol{x})$ while $\phi_t(\boldsymbol{x})$ – the distribution of unlabeled data – *might* change or not.

$$\phi_{\tau+1}(\boldsymbol{x}) \neq \phi_{\tau}(\boldsymbol{x})$$



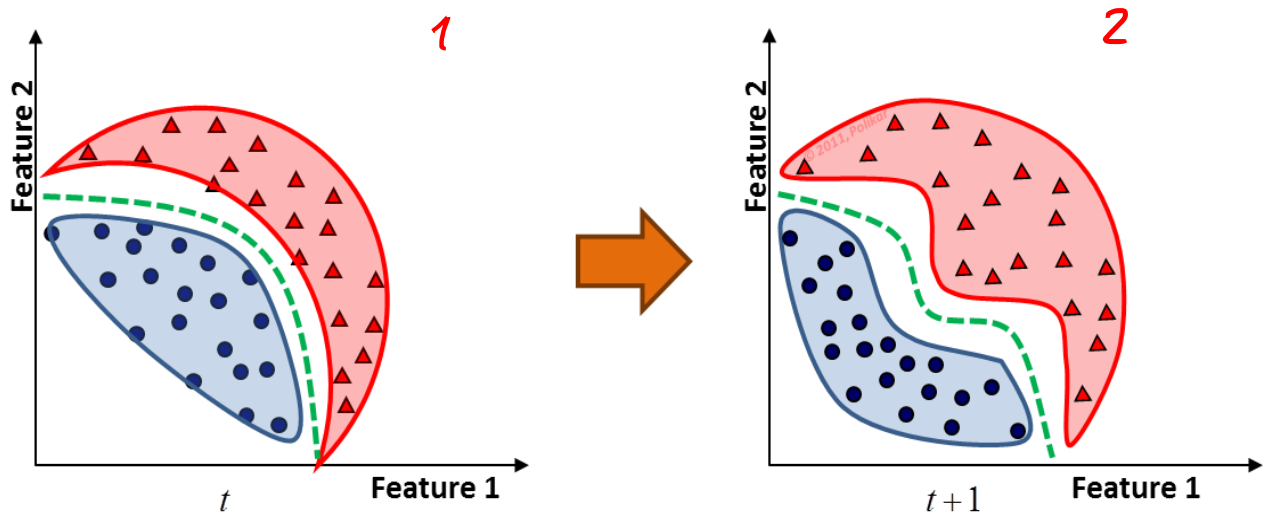
Drift Taxonomy: What Is Changing?

Real Drift

$$\phi_{\tau+1}(y|\boldsymbol{x}) \neq \phi_{\tau}(y|\boldsymbol{x})$$

affects $\phi_t(y|\boldsymbol{x})$ while $\phi_t(\boldsymbol{x})$ – the distribution of unlabeled data – *might* change or not.

$$\phi_{\tau+1}(\boldsymbol{x}) \neq \phi_{\tau}(\boldsymbol{x})$$



Drift Taxonomy: What Is Changing?

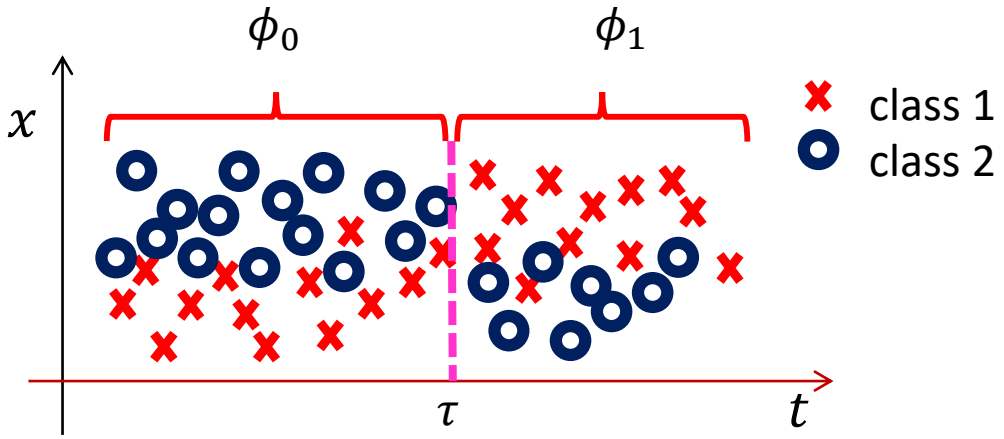
Real Drift

$$\phi_{\tau+1}(y|\boldsymbol{x}) \neq \phi_{\tau}(y|\boldsymbol{x})$$

affects $\phi_t(y|\boldsymbol{x})$ while $\phi_t(\boldsymbol{x})$ – the distribution of unlabeled data – *might* change or not.

$$\phi_{\tau+1}(\boldsymbol{x}) = \phi_{\tau}(\boldsymbol{x})$$

E.g. classes swap



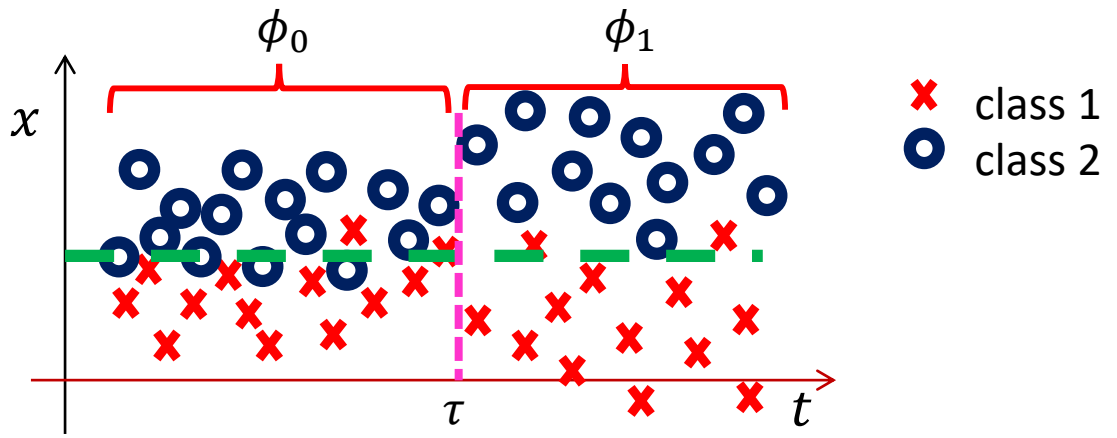
Drift Taxonomy: What Is Changing?

Virtual Drift

$$\phi_{\tau+1}(y|\boldsymbol{x}) = \phi_{\tau}(y|\boldsymbol{x}) \text{ while } \phi_{\tau+1}(\boldsymbol{x}) \neq \phi_{\tau}(\boldsymbol{x})$$

affects only $\phi_t(\boldsymbol{x})$ and leaves the class posterior probability unchanged.

These are not relevant from a predictive perspective, classifier accuracy is not affected



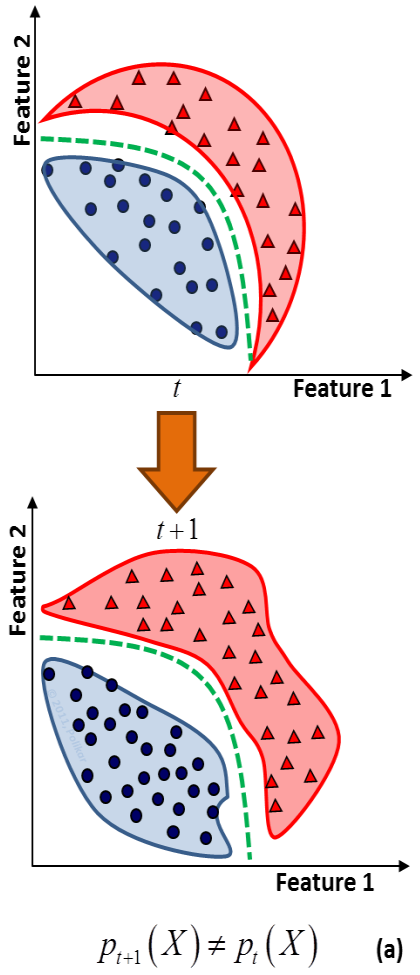
Drift Taxonomy: What Is Changing?

Virtual Drift

$$\phi_{\tau+1}(y|\boldsymbol{x}) = \phi_{\tau}(y|\boldsymbol{x})$$
$$\text{while } \phi_{\tau+1}(\boldsymbol{x}) \neq \phi_{\tau}(\boldsymbol{x})$$

affects only $\phi_t(\boldsymbol{x})$ and leaves the class posterior probability unchanged.

$$\phi_{\tau+1}(\boldsymbol{x}) \neq \phi_{\tau}(\boldsymbol{x})$$



$p_{t+1}(X) \neq p_t(X)$ (a)

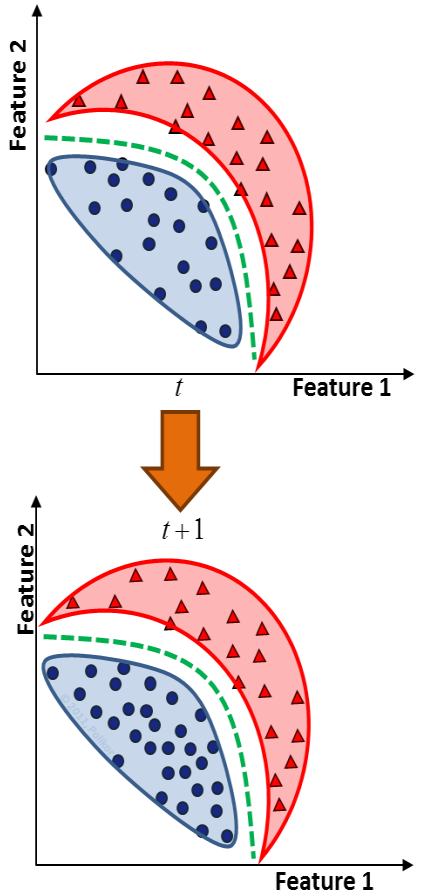
Drift Taxonomy: What Is Changing?

Virtual Drift

$$\phi_{\tau+1}(y|\boldsymbol{x}) = \phi_{\tau}(y|\boldsymbol{x})$$
$$\text{while } \phi_{\tau+1}(\boldsymbol{x}) \neq \phi_{\tau}(\boldsymbol{x})$$

affects only $\phi_t(\boldsymbol{x})$ and leaves the class posterior probability unchanged.

$$\phi_{\tau+1}(\boldsymbol{x}) \neq \phi_{\tau}(\boldsymbol{x})$$



$$p_{\tau+1}(y = c) \neq p_{\tau}(y = c)$$

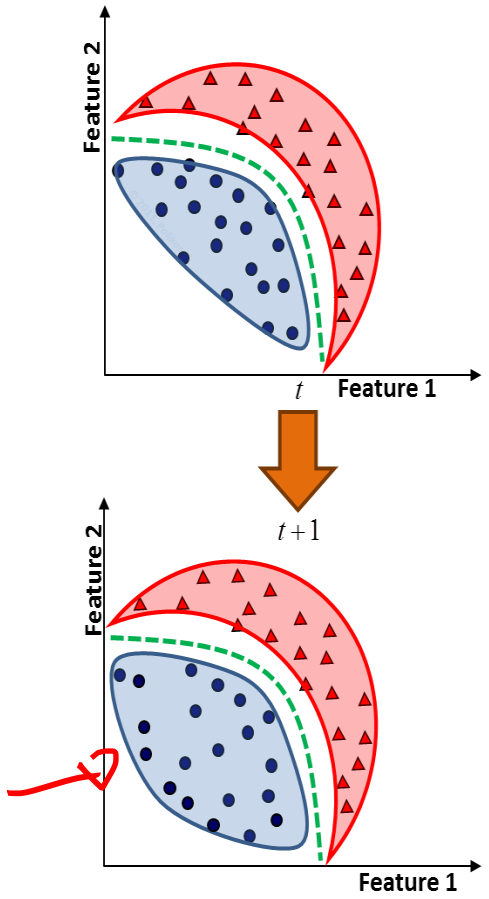
Drift Taxonomy: What Is Changing?

Virtual Drift

$$\phi_{\tau+1}(y|\boldsymbol{x}) = \phi_{\tau}(y|\boldsymbol{x})$$
$$\text{while } \phi_{\tau+1}(\boldsymbol{x}) \neq \phi_{\tau}(\boldsymbol{x})$$

affects only $\phi_t(\boldsymbol{x})$ and leaves the class posterior probability unchanged.

$$\phi_{\tau+1}(\boldsymbol{x}) \neq \phi_{\tau}(\boldsymbol{x})$$



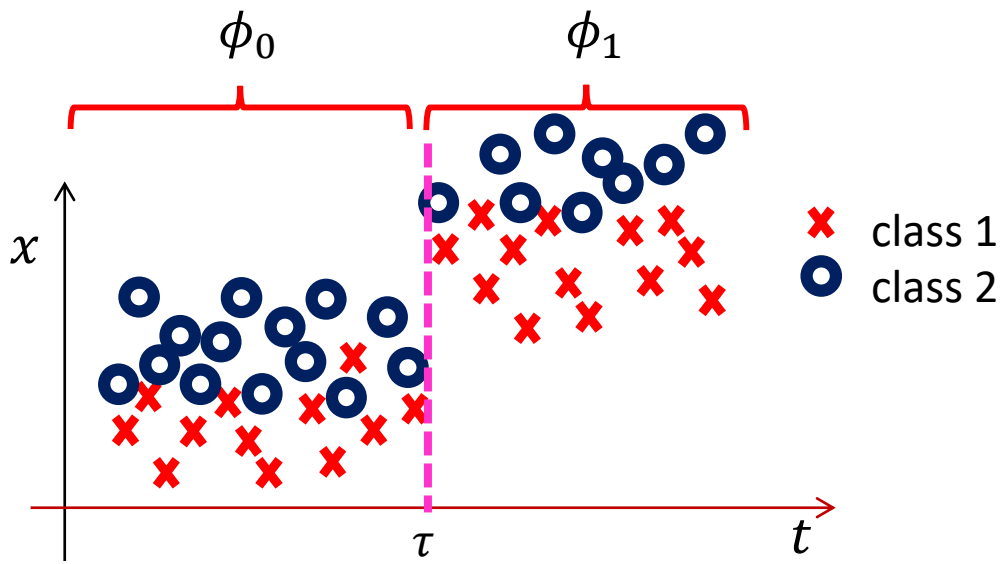
$$\phi_{\tau+1}(\boldsymbol{x}|y = c) \neq \phi_{\tau}(\boldsymbol{x}|y = c)$$

Drift Taxonomy: Time Evolution

Abrupt

$$\phi_t(\boldsymbol{x}, y) = \begin{cases} \phi_0(\boldsymbol{x}, y) & t < \tau \\ \phi_1(\boldsymbol{x}, y) & t \geq \tau \end{cases}$$

Permanent shift in the state of $\mathcal{X}$, e.g. a faulty sensor, or a system turned to an active state

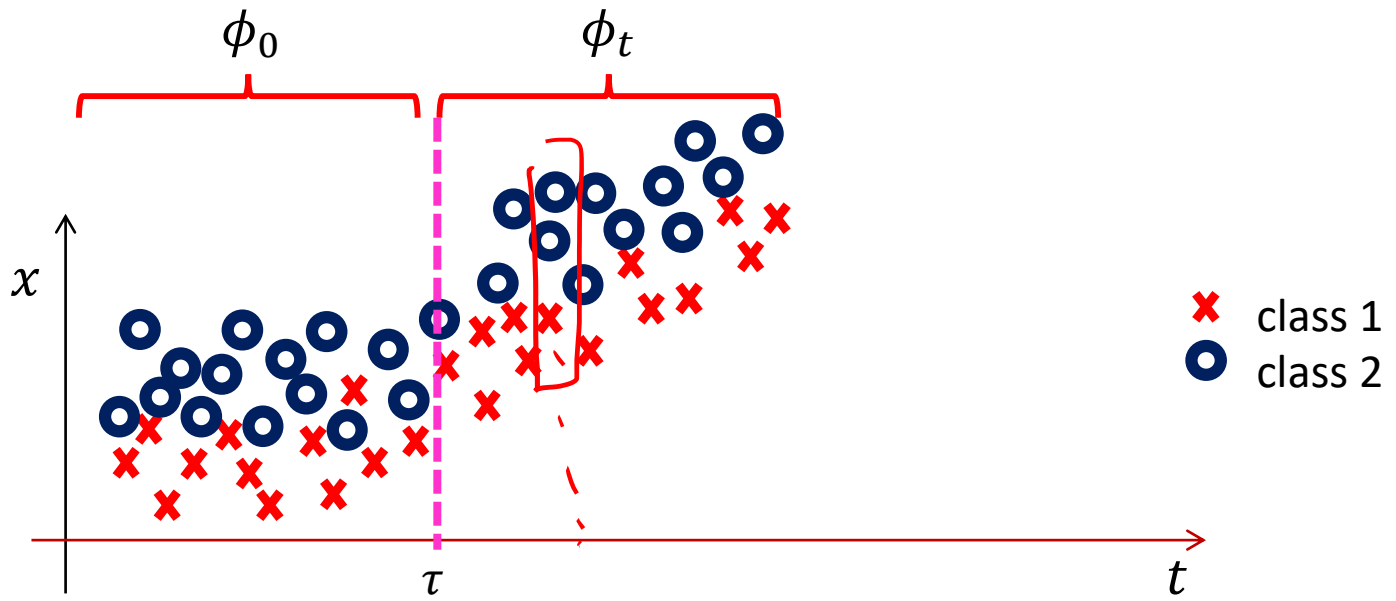


Drift Taxonomy: Time Evolution

Incremental

$$\phi_t(\boldsymbol{x}, y) = \begin{cases} \phi_0(\boldsymbol{x}, y) & t < \tau \\ \phi_t(\boldsymbol{x}, y) & t \geq \tau \end{cases}$$

There is a continuously drifting condition after the change

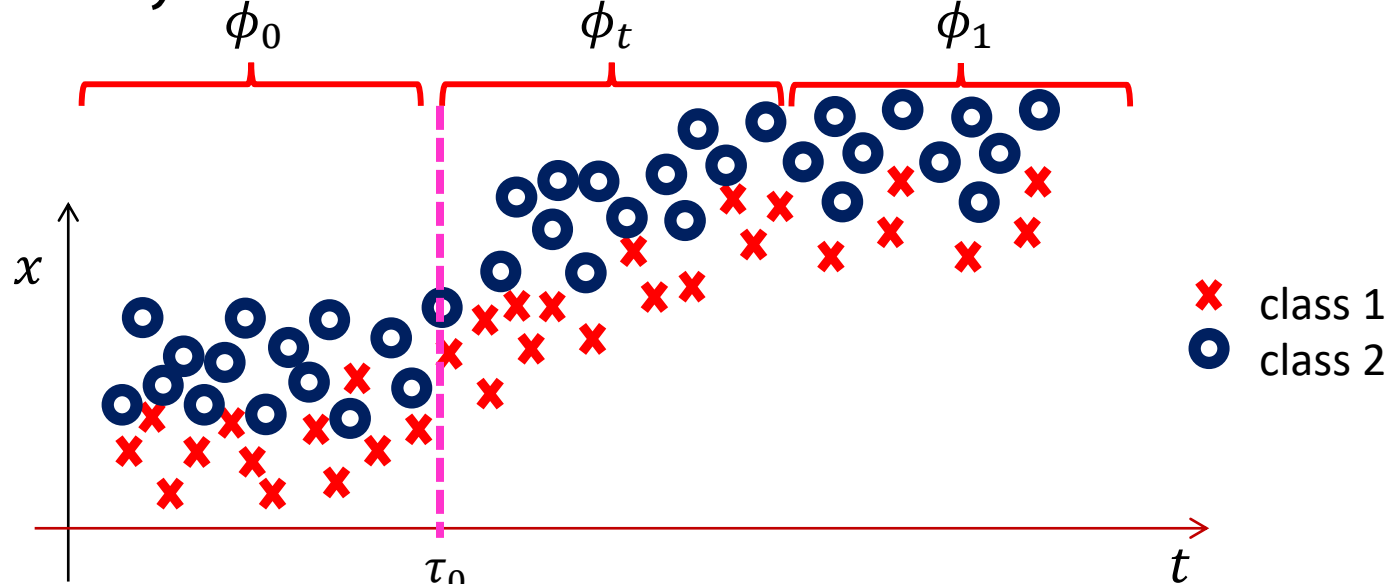


Drift Taxonomy: Time Evolution

Incremental

$$\phi_t(\boldsymbol{x}, y) = \begin{cases} \phi_0(\boldsymbol{x}, y) & t < \tau_0 \\ \phi_t(\boldsymbol{x}, y) & \tau_0 \leq t < \tau_1 \\ \phi_1(\boldsymbol{x}, y) & t \geq \tau_1 \end{cases}$$

There is a continuously drifting condition after the change that might end up in another stationary state

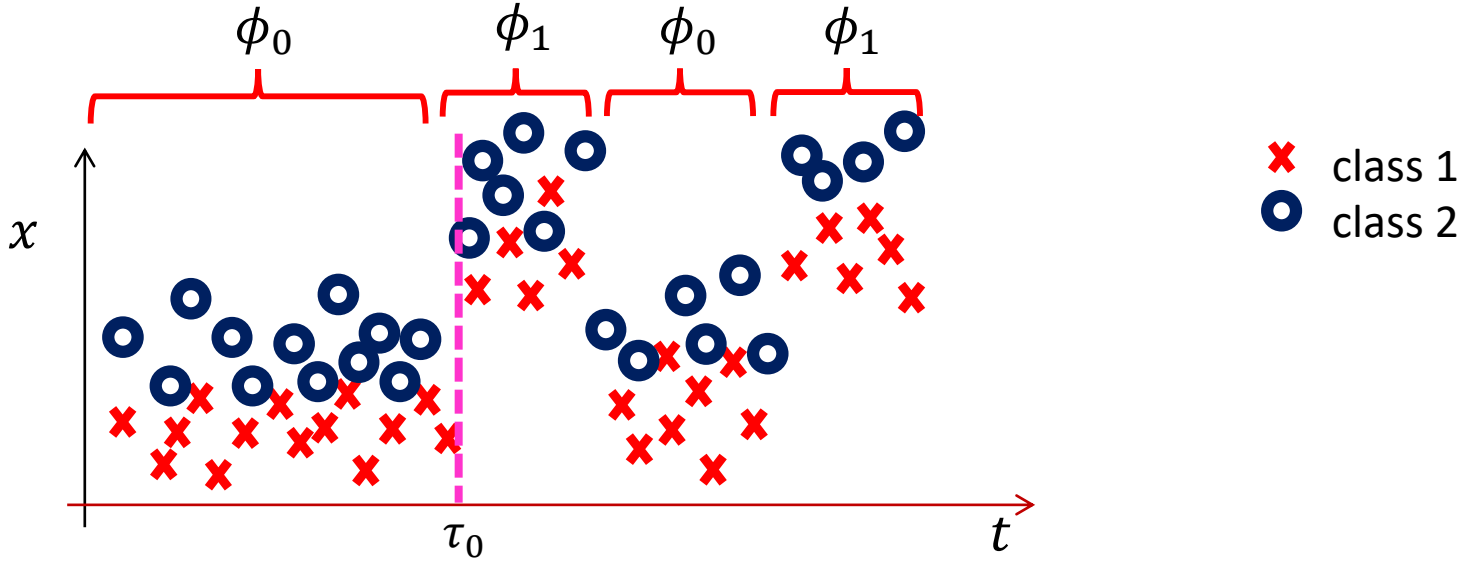


Drift Taxonomy: Time Evolution

Recurring

$$\phi_t(\boldsymbol{x}, y) = \begin{cases} \phi_0(\boldsymbol{x}, y) & t < \tau_0 \\ \phi_1(\boldsymbol{x}, y) & \tau_0 \leq t < \tau_1 \\ & \dots \\ \phi_0(\boldsymbol{x}, y) & t \geq \tau_n \end{cases}$$

After concept drift, it is possible for $\mathcal{X}$ to go back to previous concept ϕ_0

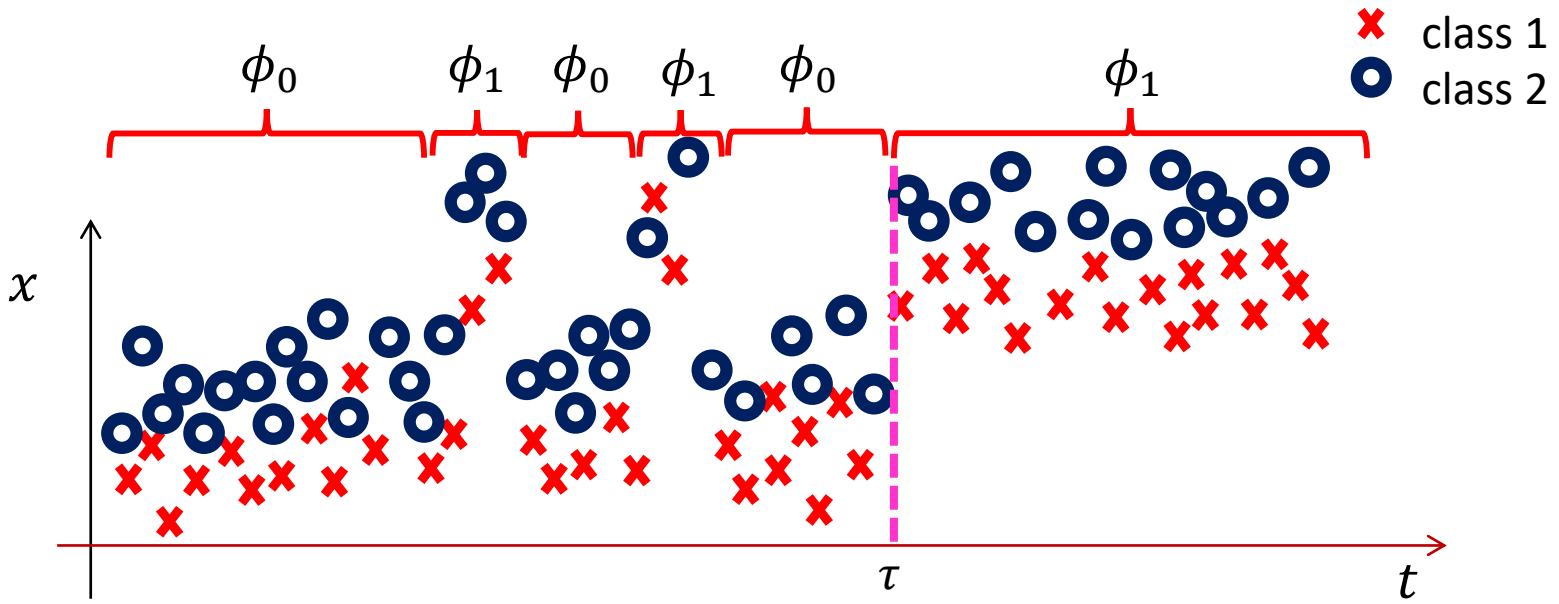


Drift Taxonomy: Time Evolution

Gradual

$$\phi_t(\boldsymbol{x}, y) = \begin{cases} \phi_0(\boldsymbol{x}, y) \; or \; \phi_1(\boldsymbol{x}, y) & t < \tau \\ \phi_1(\boldsymbol{x}, y) & t \geq \tau \end{cases}$$

The process definitively switches in the new conditions after having anticipated some short drifts



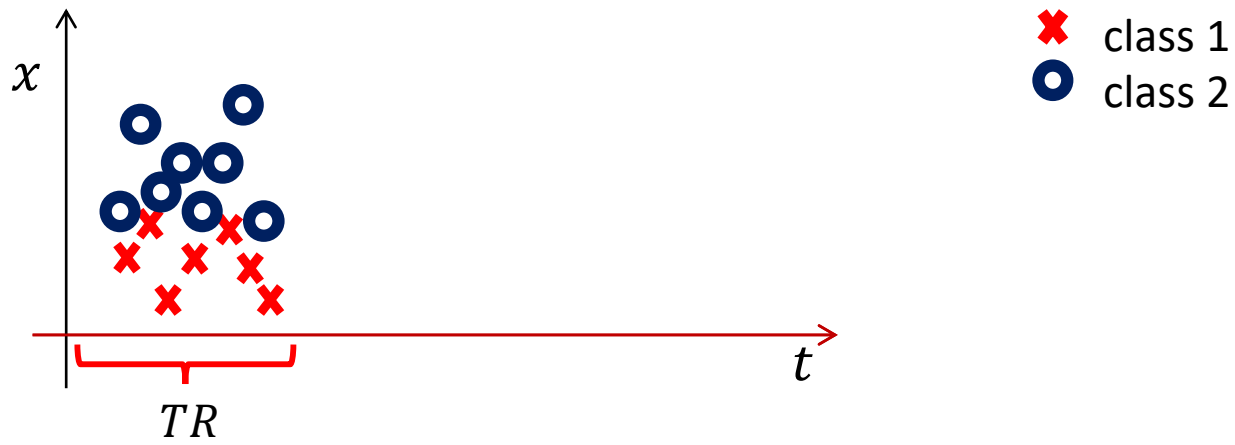
Is Concept Drift a Problem?

The need for Adaptation

Classification Over Datastreams

Consider as, an illustrative example, a simple 1-dimensional classification problem, where

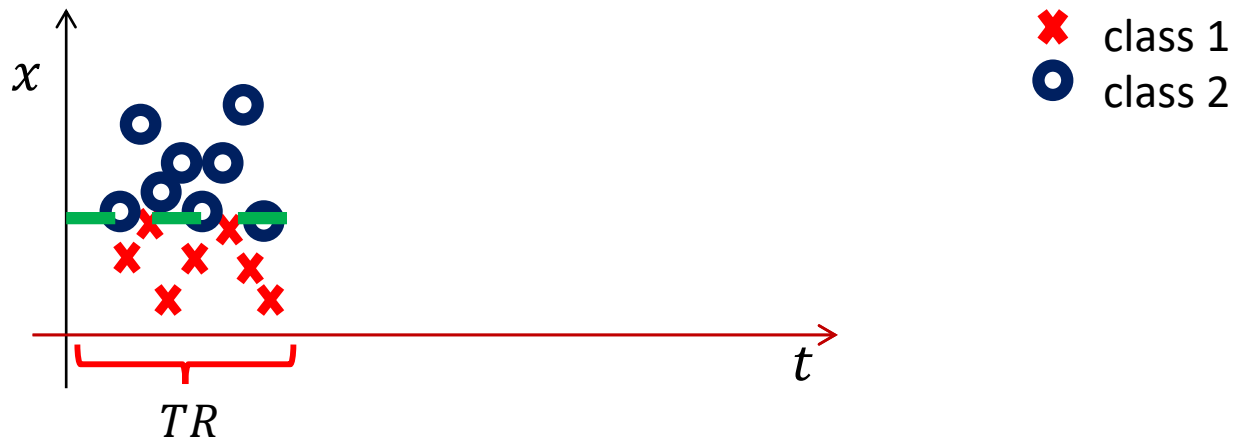
- The initial part of the stream is provided for training



Classification Over Datastreams

Consider as, an illustrative example, a simple 1-dimensional classification problem, where

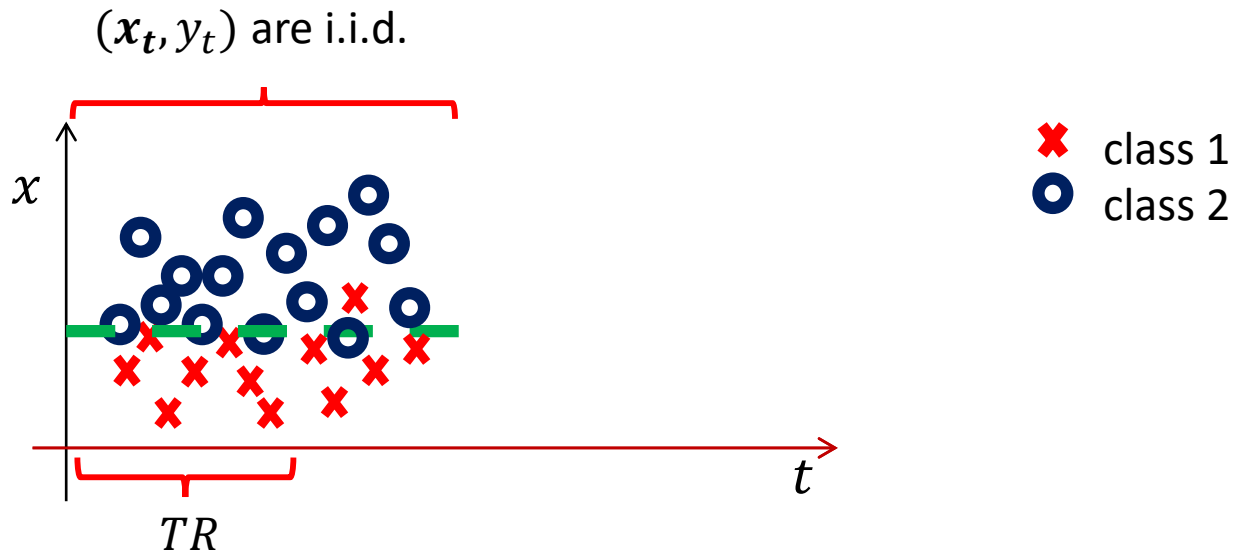
- The initial part of the stream is provided for training
- K is simply a threshold



Classification Over Datastreams

Consider as, an illustrative example, a simple 1-dimensional classification problem, where

- The initial part of the stream is provided for training
- K is simply a threshold

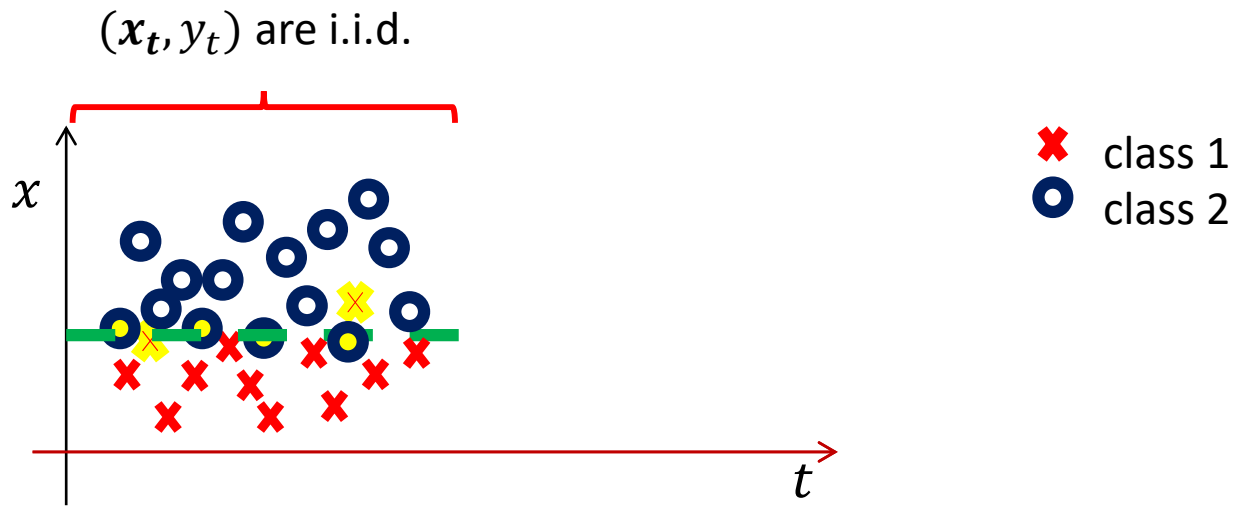


Classification Over Datastreams

Consider as, an illustrative example, a simple 1-dimensional classification problem, where

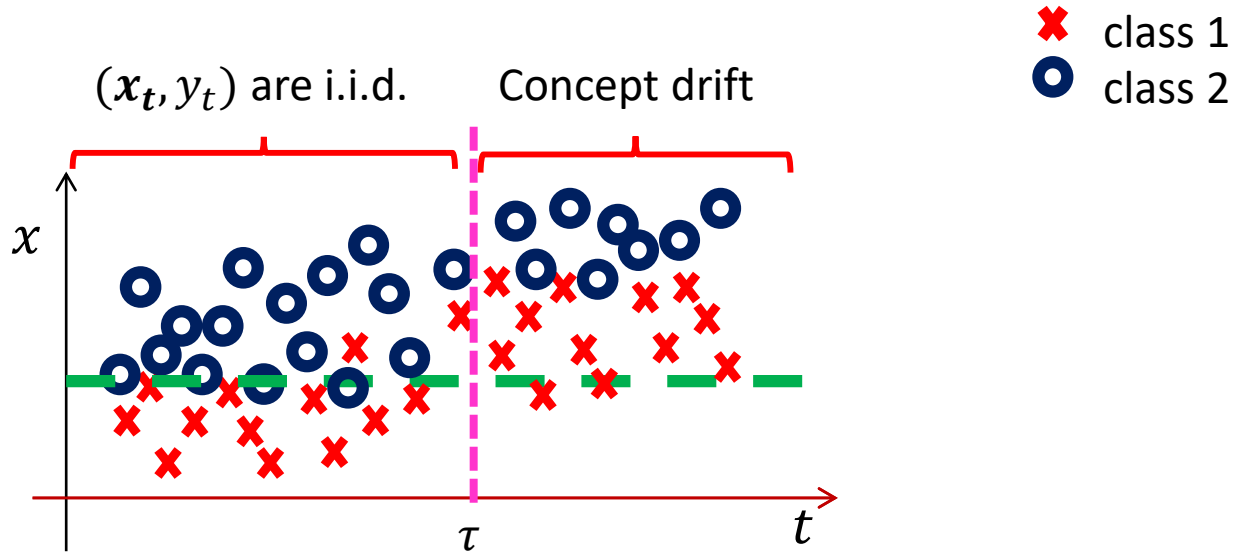
- The initial part of the stream is provided for training
- K is simply a threshold

As far as data are i.i.d., the classification error is *controlled*



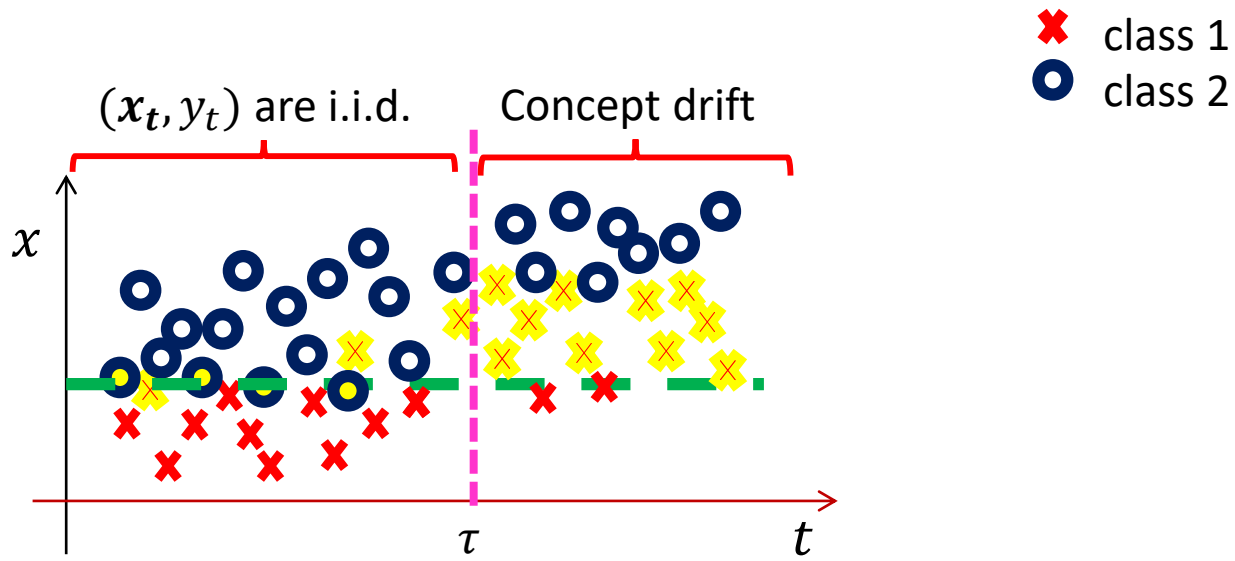
Classification Over Datastreams

Unfortunately, when **concept drift occurs**, and ϕ changes,



Classification Over Datastreams

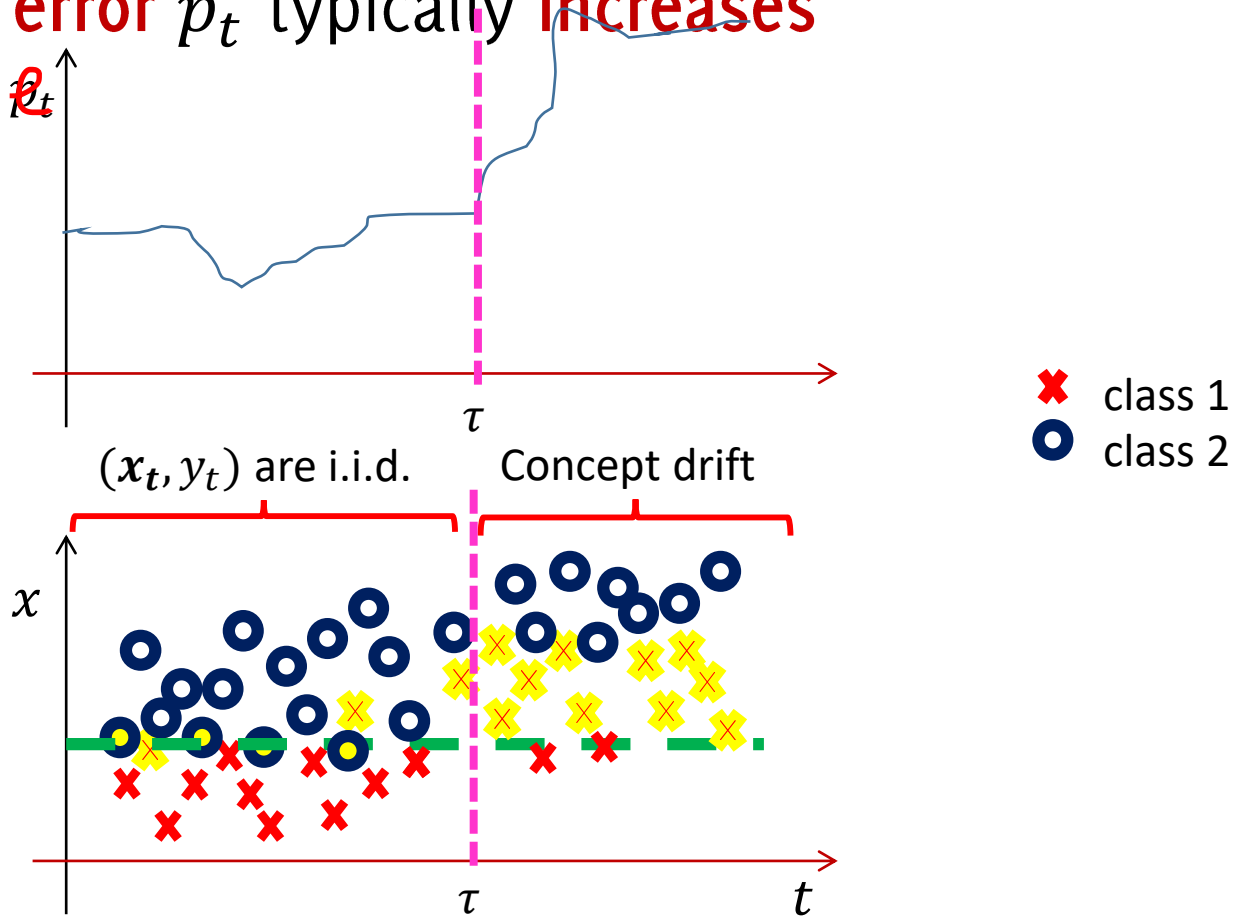
Unfortunately, when **concept drift occurs**, and ϕ changes, things can be terribly worst,



Classification Over Datastreams

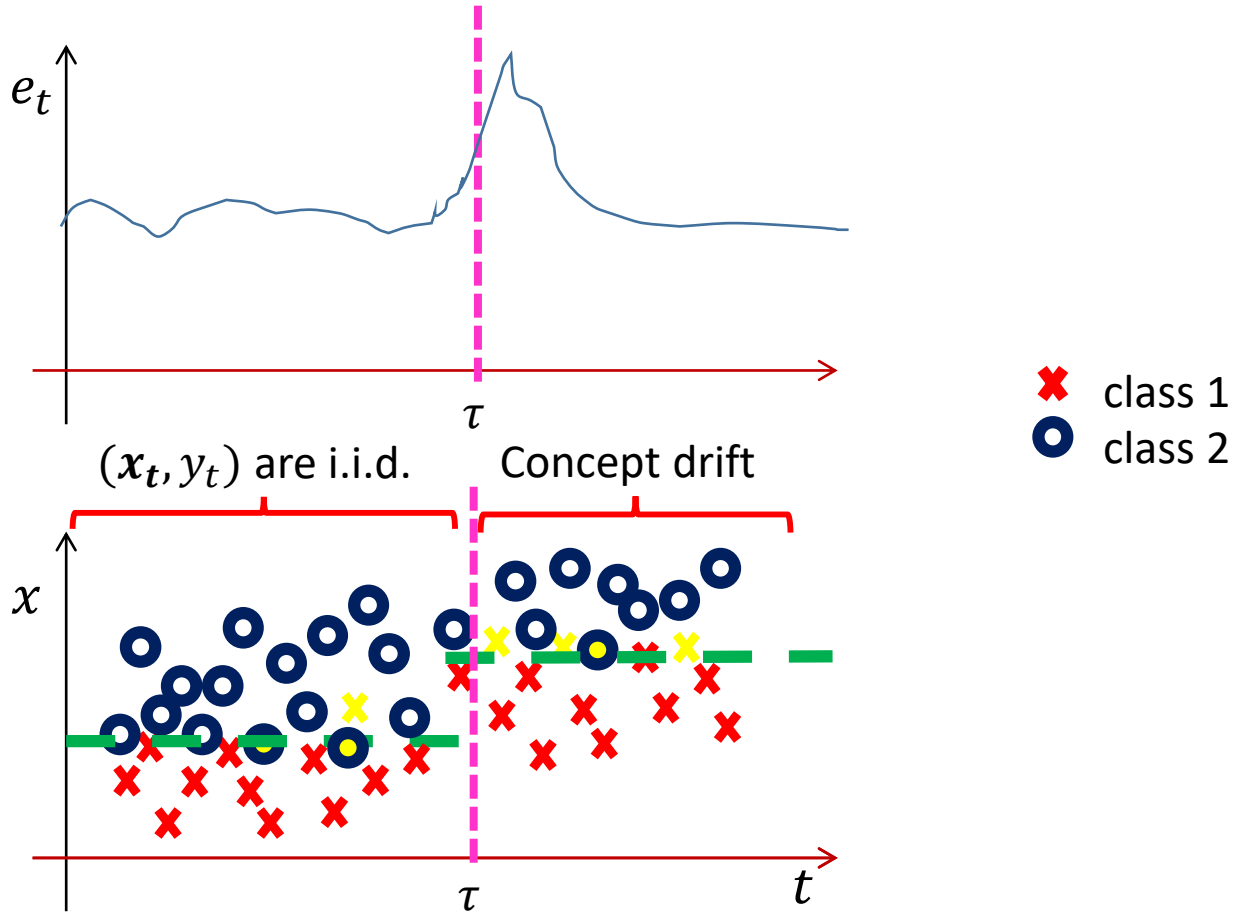
Unfortunately, when **concept drift occurs**, and ϕ changes, things can be terribly worst,

The **average classification error** p_t typically **increases**



Need For Adaptation

Adaptation is needed to **preserve** classifier performance



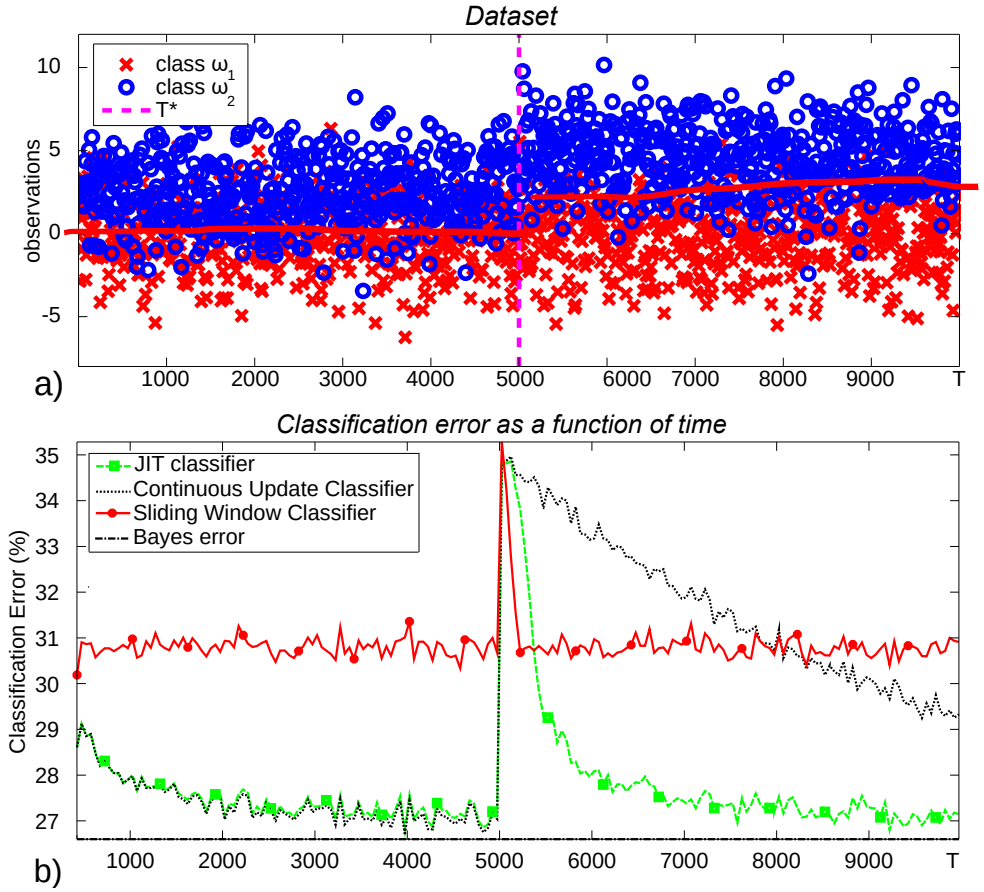
Adaptation

Do we Really Need Smart Adaptation Strategies?

Simple Adaptation Strategies

Consider two simple adaptation strategies and a simple concept drift

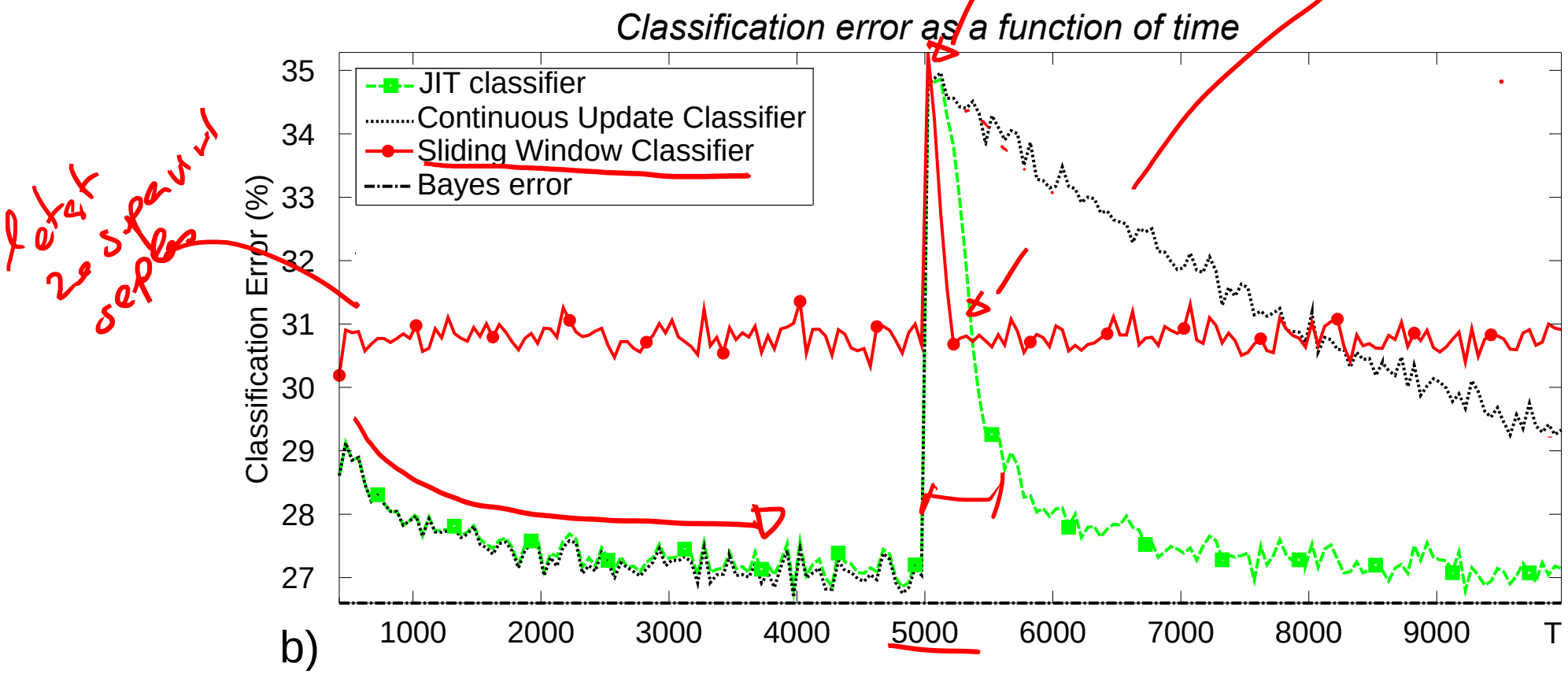
- Continuously update K_t using all supervised couples
- Train K_t using only the last δ supervised couples



Simple Adaptation Strategies

Classification error of two simple adaptation strategies

- Black dots: K_t uses all supervised couples at time t
- Red line: K_t uses only the last δ supervised couples

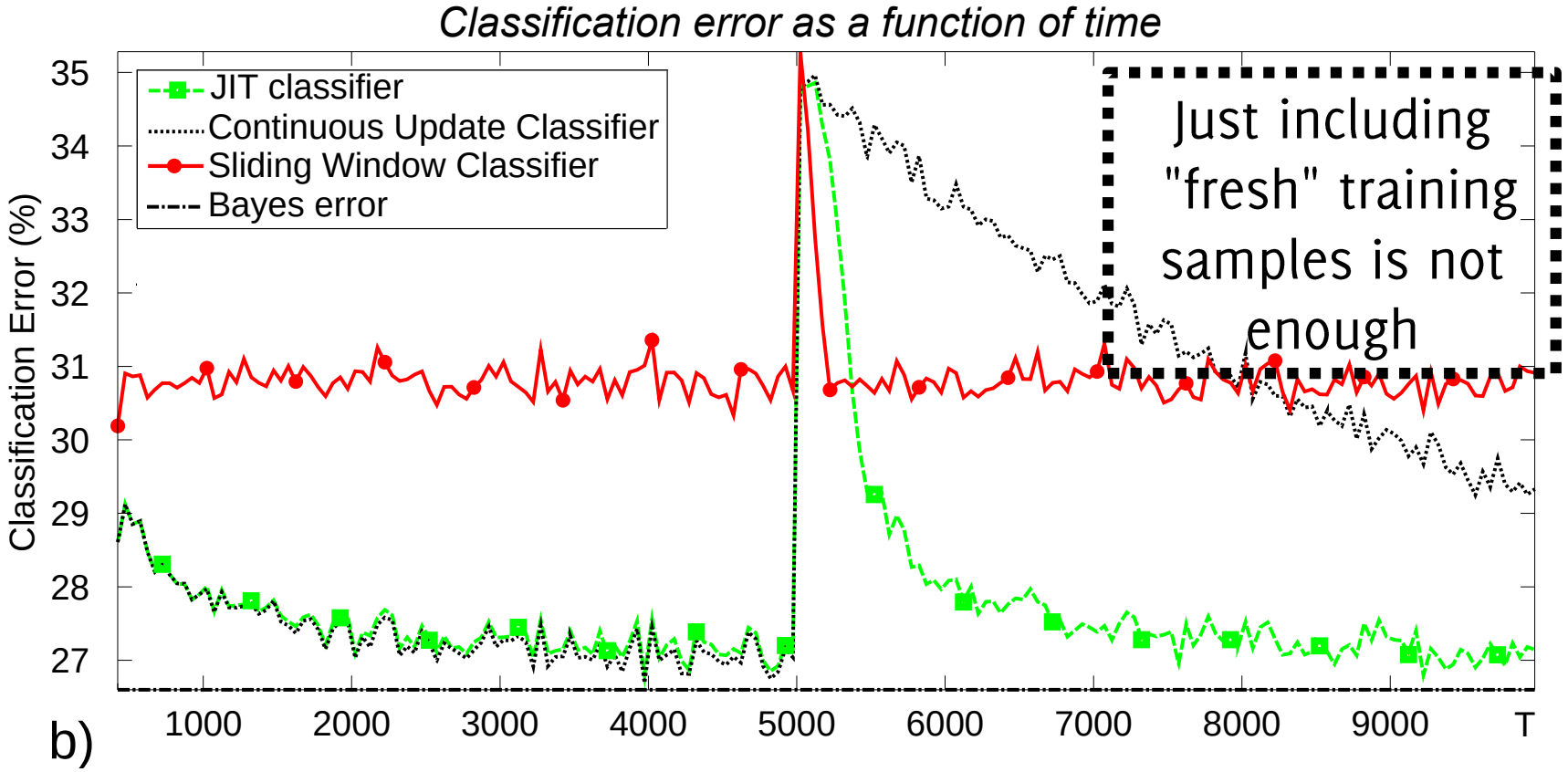


Alippi, C., Boracchi, G., Roveri, M. (2013). Just-in-time classifiers for recurrent concepts. *IEEE TNNLS* 620-634.

Simple Adaptation Strategies

Classification error of two simple adaptation strategies

- Black dots: K_t uses all supervised couples at time t
- Red line: K_t uses only the last δ supervised couples

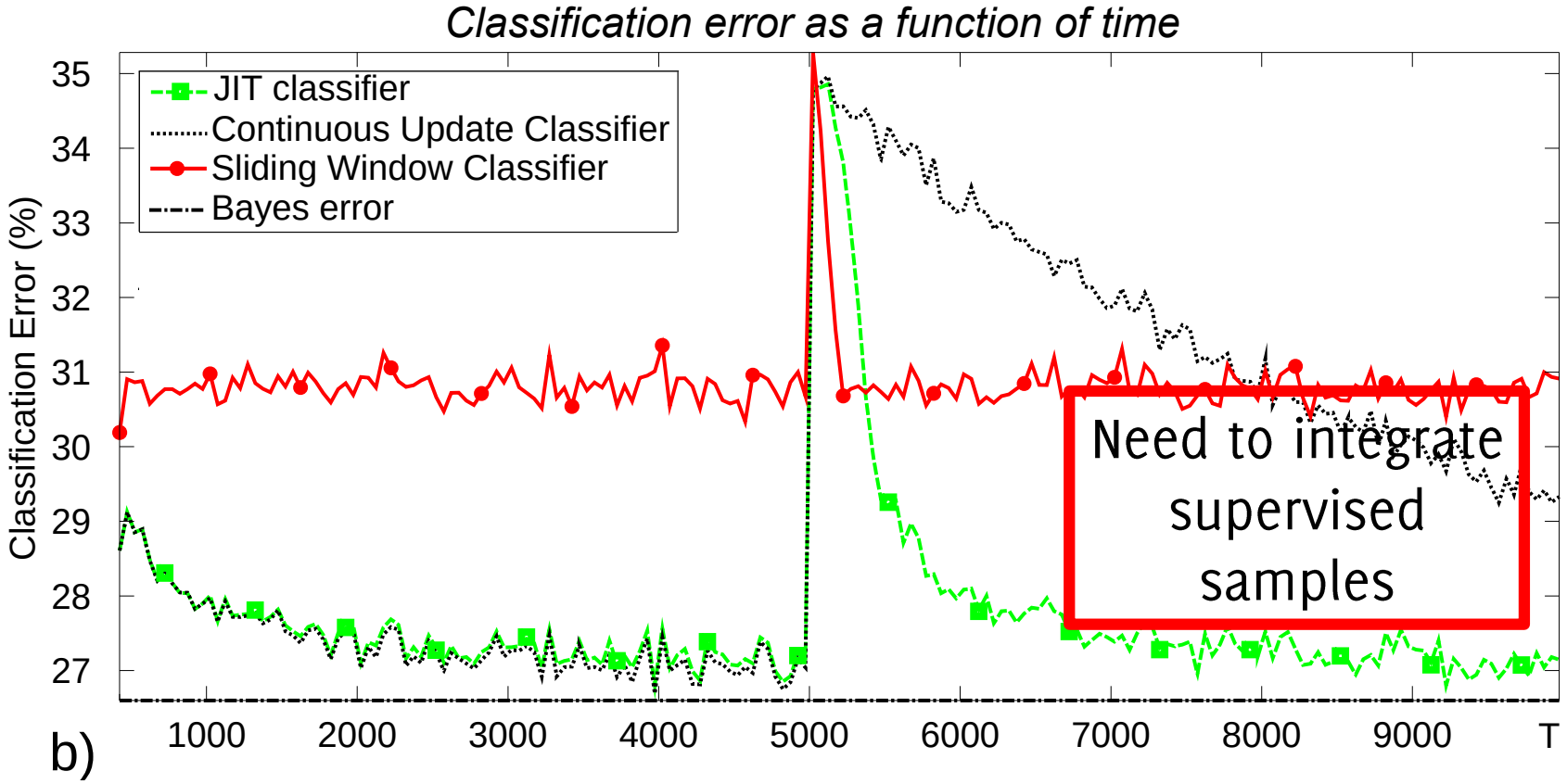


Alippi, C., Boracchi, G., Roveri, M. (2013). Just-in-time classifiers for recurrent concepts. *IEEE TNNLS* 620-634.

Simple Adaptation Strategies

Classification error of two simple adaptation strategies

- Black dots: K_t uses all supervised couples at time t
- Red line: K_t uses only the last δ supervised couples

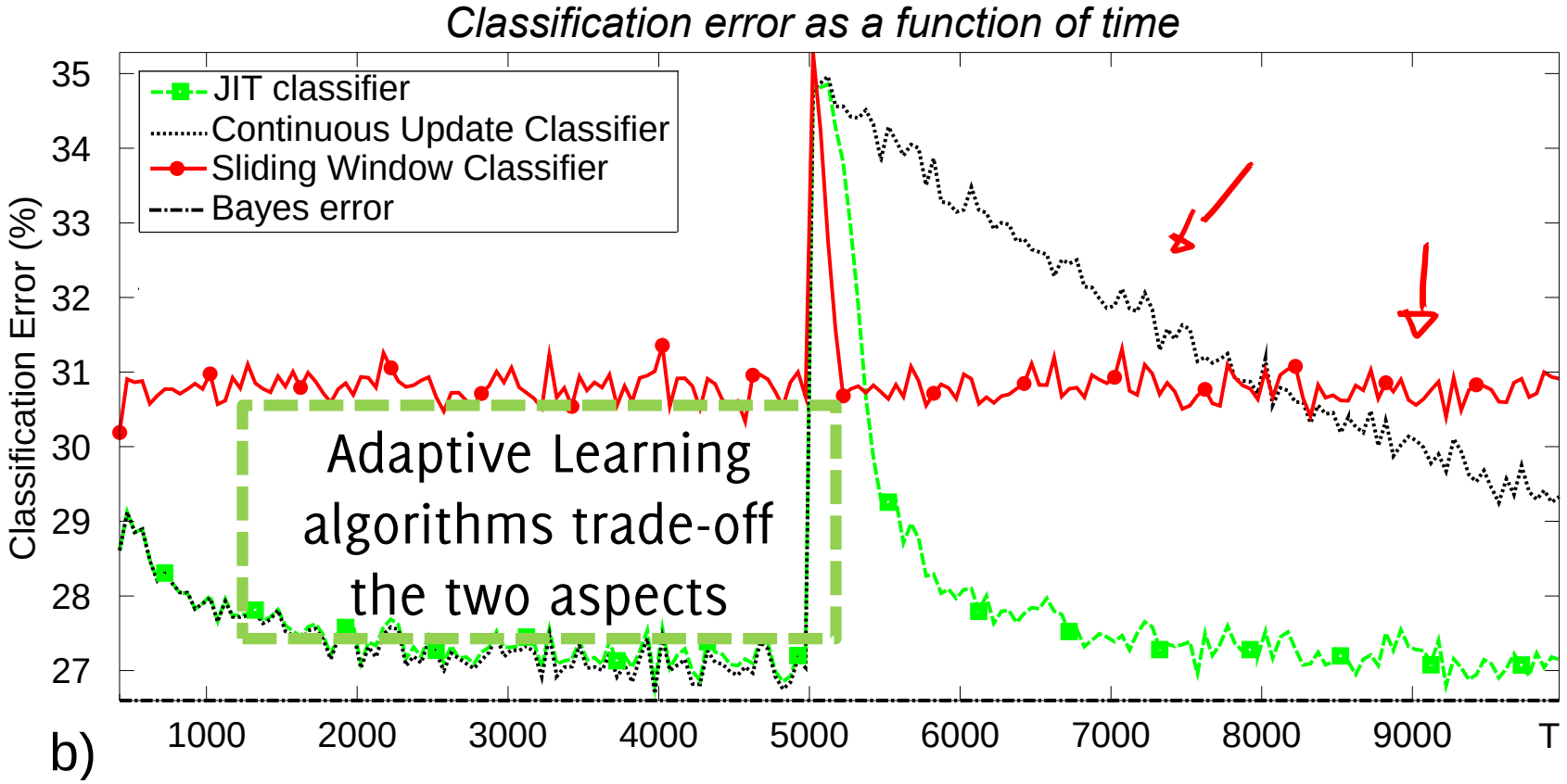


Alippi, C., Boracchi, G., Roveri, M. (2013). Just-in-time classifiers for recurrent concepts. *IEEE TNNLS* 620-634.

Simple Adaptation Strategies

Classification error of two simple adaptation strategies

- Black dots: K_t uses all supervised couples at time t
- Red line: K_t uses only the last δ supervised couples



Alippi, C., Boracchi, G., Roveri, M. (2013). Just-in-time classifiers for recurrent concepts. *IEEE TNNLS* 620-634.

First Matlab Assignment

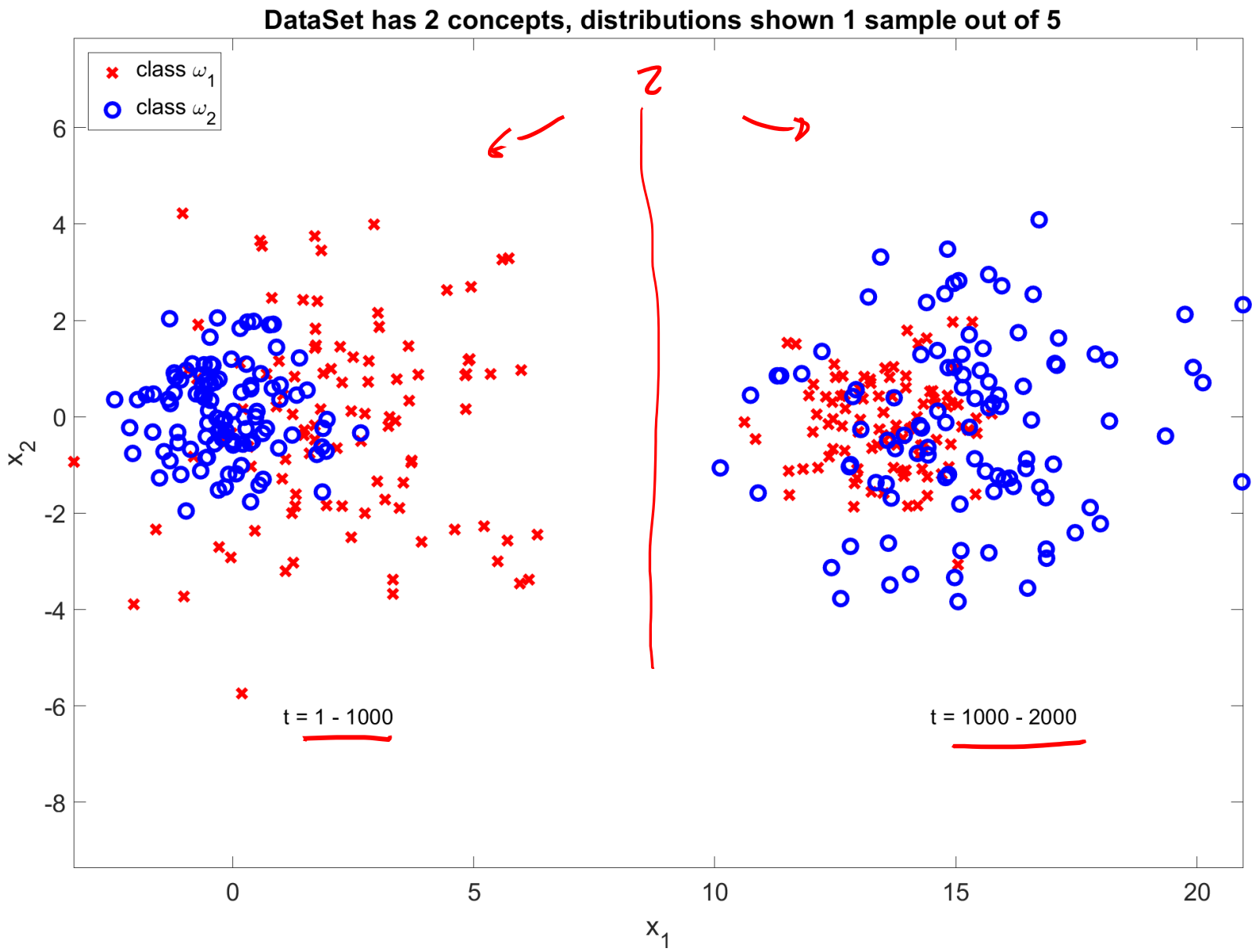
Get the first matlab snippet

And develop

- a classifier that is always updated every time a feedback is provided
- A classifier that is always updated over the latest ν training samples
- Compute and display the classification error of these classifiers over the whole datastream

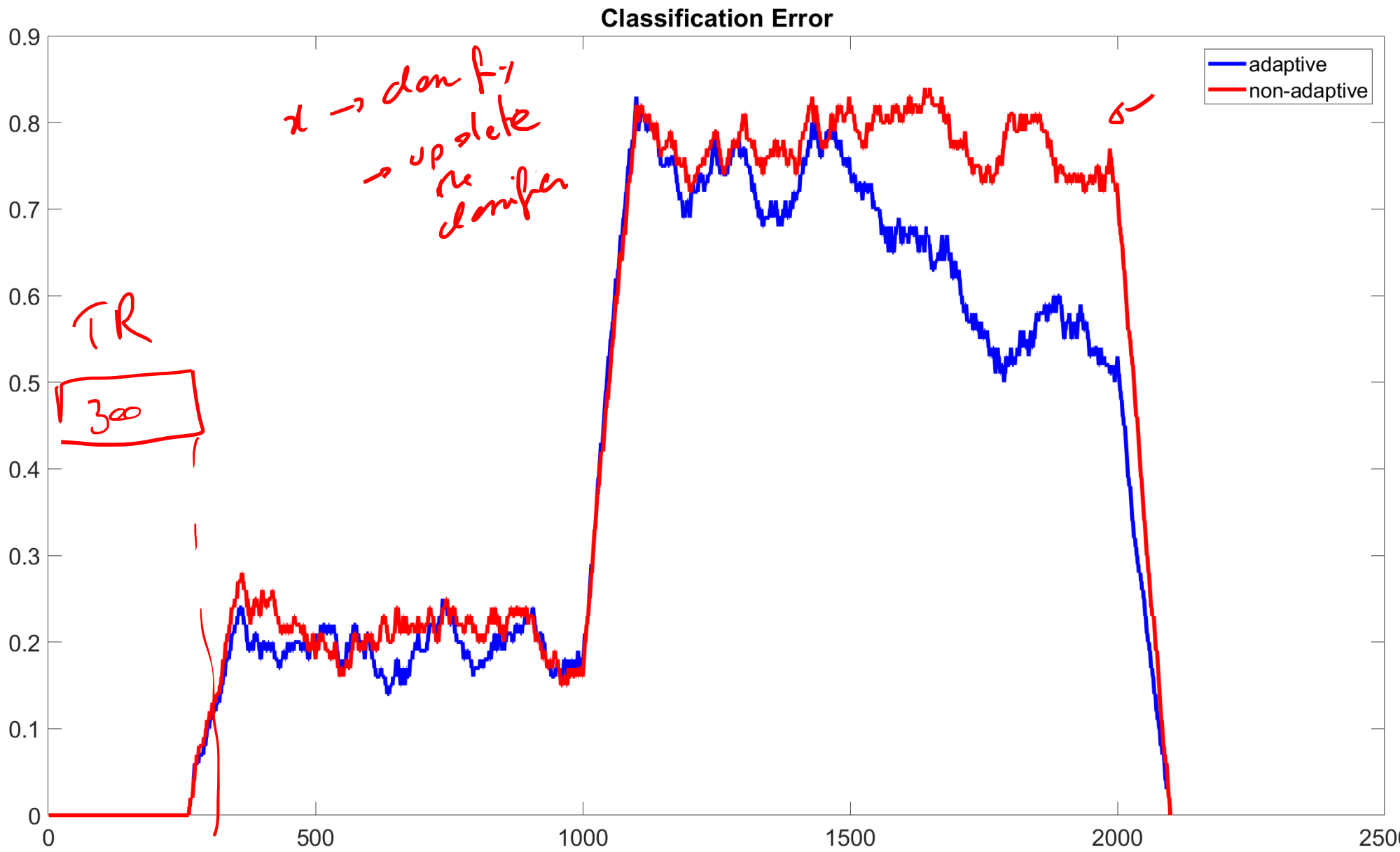
Compare the performance with a classifier that is never updated

Generate a dataset like this one



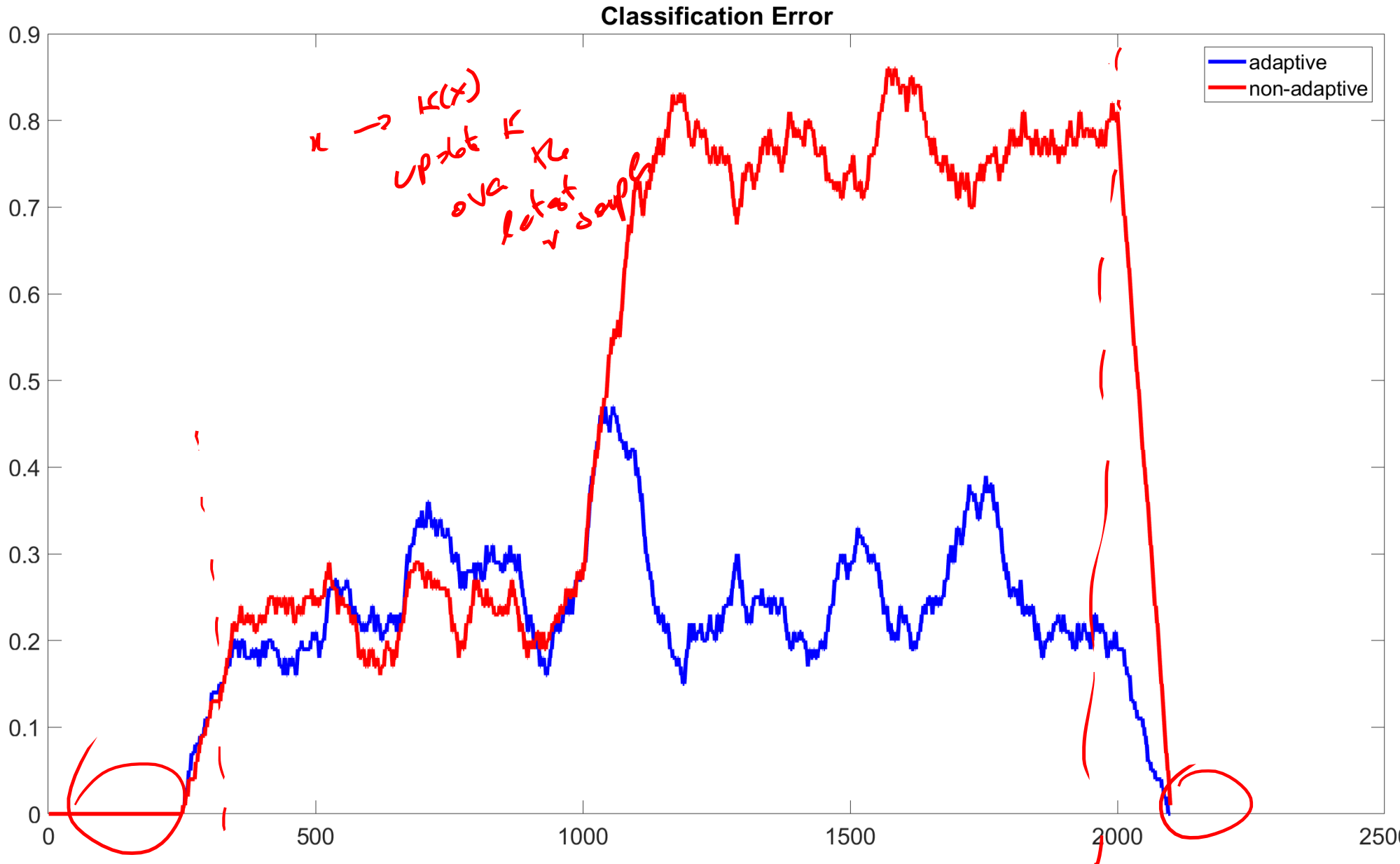
abrupt real drift

Classifier always updated



Boracchi, Trovò

Sliding window classifier (50 samples)



Boracchi, Trovò

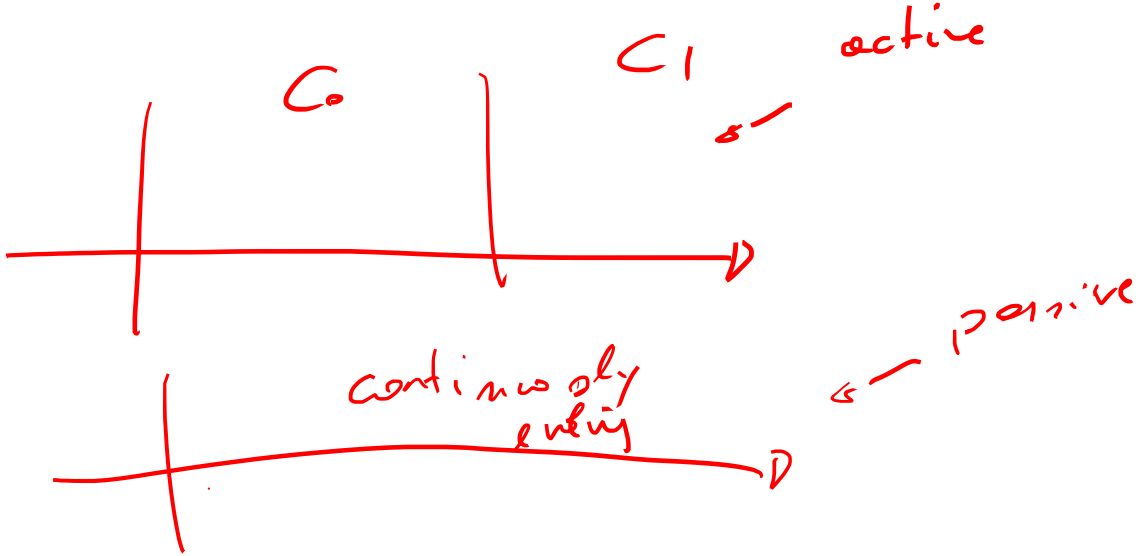
Active Approaches: The General Picture

Adaptation Strategies Under Concept Drift

Two main solutions in the literature:

- **Active**: the classifier K_t is combined with statistical tools **to detect concept drift and pilot the adaptation**
- **Passive**: the classifier K_t undergoes **continuous adaptation** determining every time which supervised information to preserve

Which is best depends on the expected change rate and memory/computational availability



Active Approaches

Peculiarities:

- Relies on an **explicit drift-detection mechanism**: such as an outlier detection or a change detection test (CDT)
- Specific **post-detection adaptation** procedures to isolate recent data generated after the change, thus that are coherent with the new concept

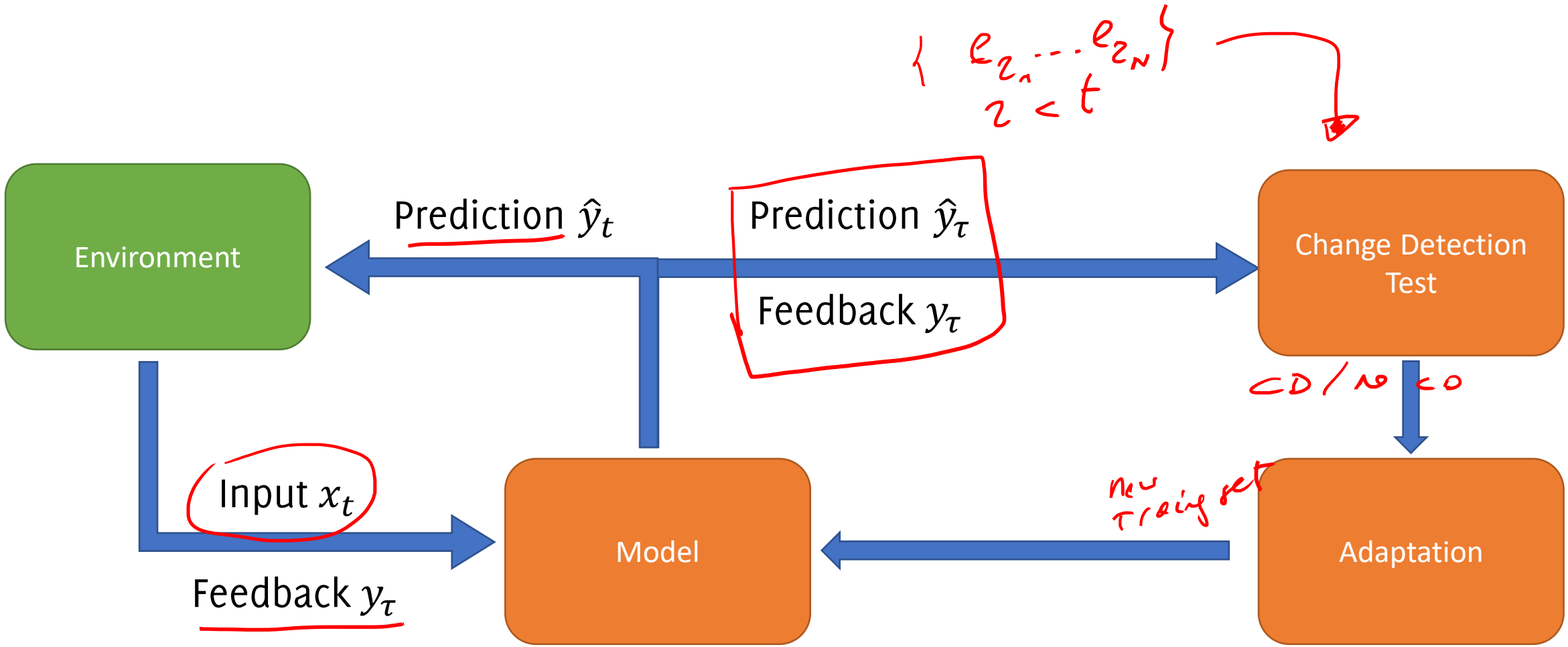
Pro:

- Also provide information that CD has occurred
- Can improve their performance in stationary conditions
- Alternatively, classifier adapts only after detection

Cons:

- Difficult to handle incremental and gradual drifts

Monitoring the Classification Error



Monitoring the Classification Error

The simplest approach consist in monitoring the classification error (or similar performance measure)

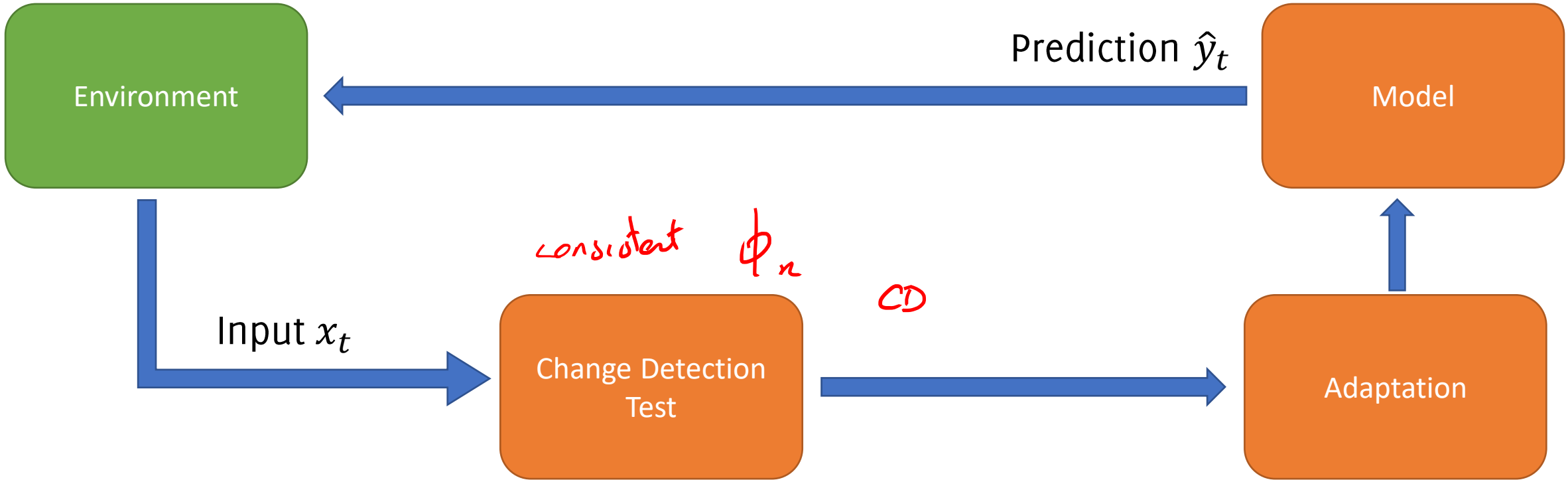
Pro:

- It is the most straightforward figure of merit to monitor
- Changes in p_t prompts adaptation only when performance are affected

Cons:

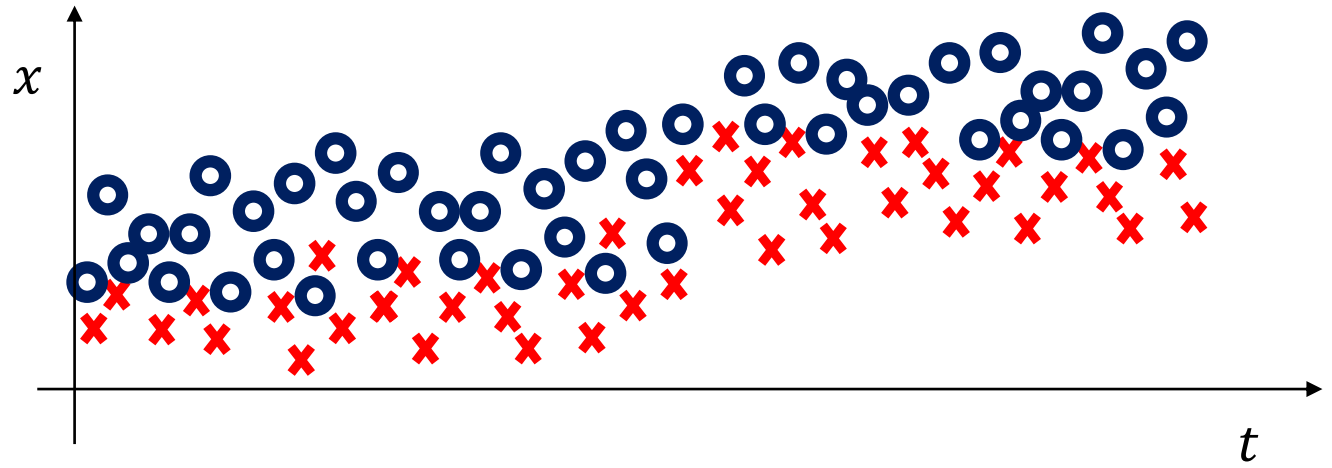
- CD detection from supervised samples only

Monitoring Input Distribution



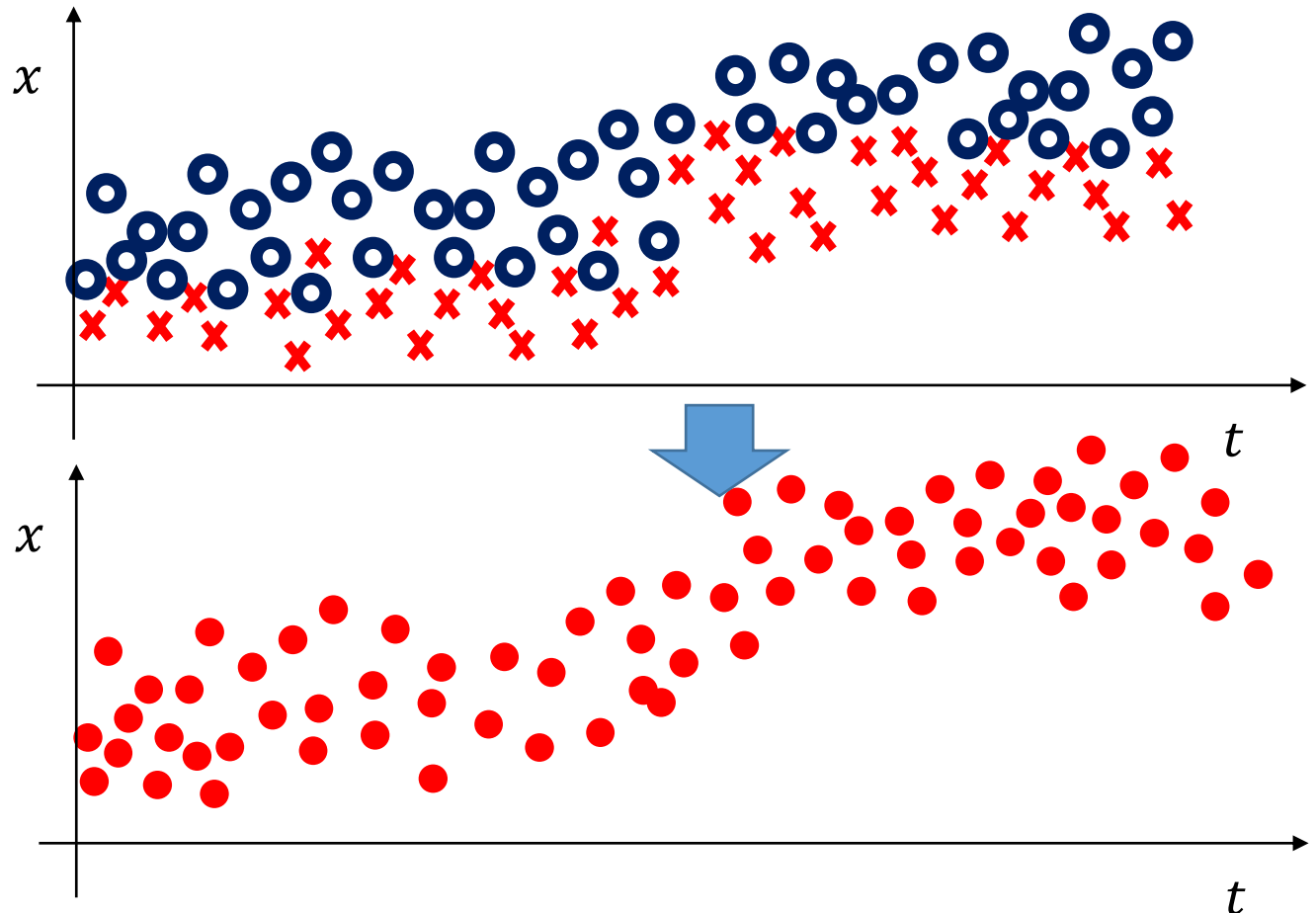
Monitoring Input Distribution

In some cases, CD can be detected by ignoring class labels and **monitoring the distribution of the input,** unsupervised, raw data.



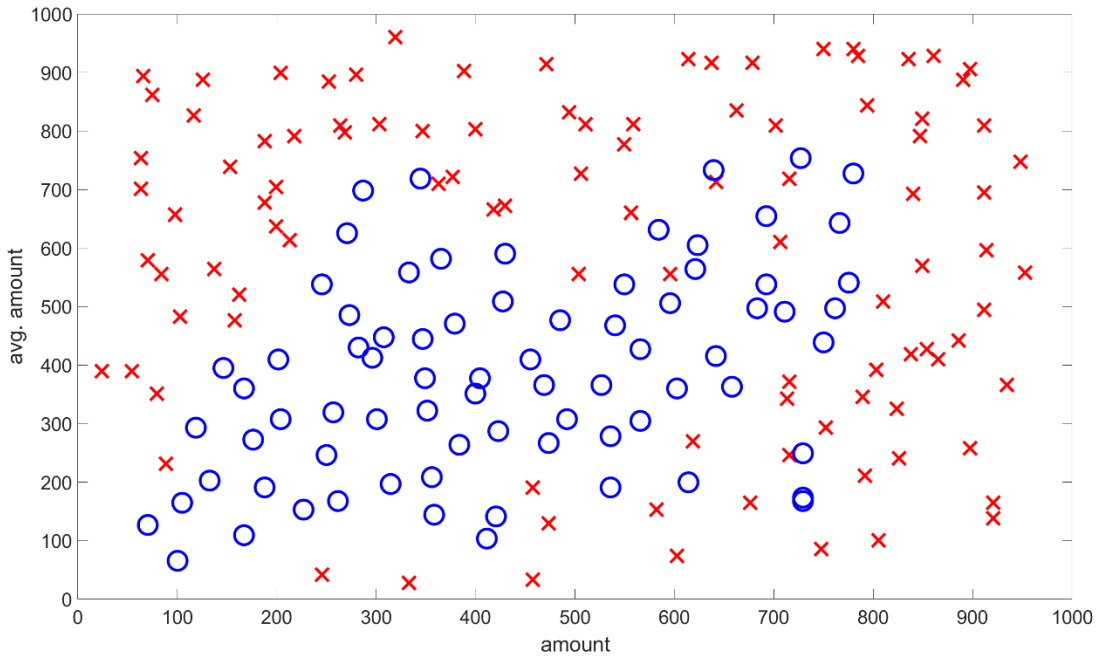
Monitoring Input Distribution

In some cases, CD can be detected by ignoring class labels and **monitoring the distribution of the input,** unsupervised, raw data.

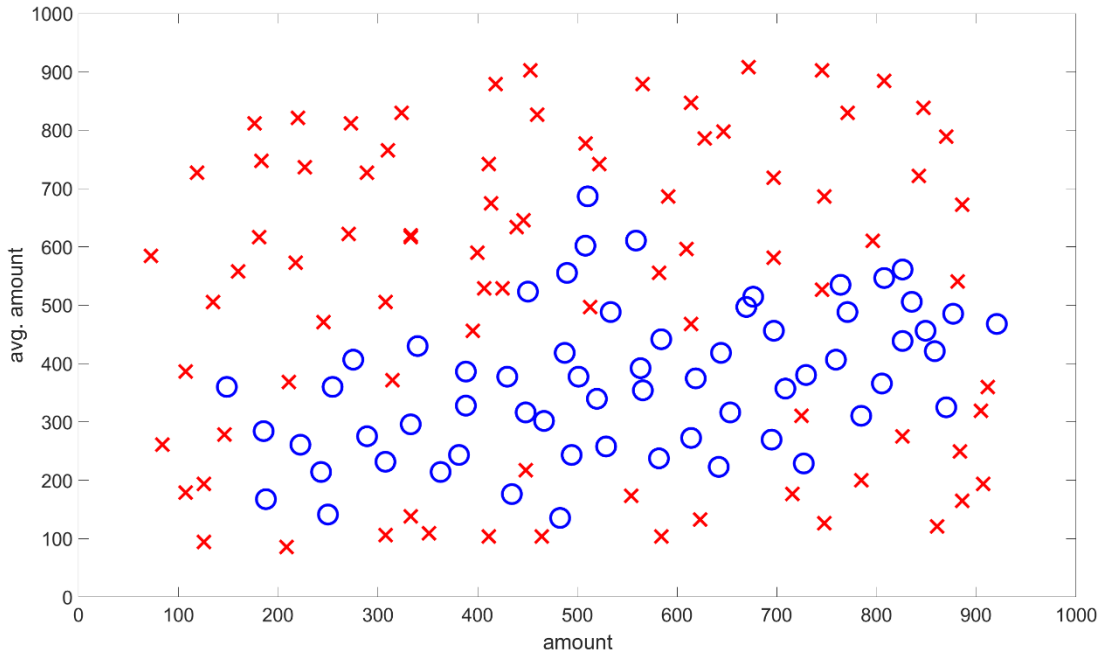


Monitoring Input Distribution

$$(\boldsymbol{x_t}, y_t) \sim \phi^0_{x,y}, t < \tau_0$$



$$(\boldsymbol{x_t}, y_t) \sim \phi^1_{x,y}, t > \tau_0$$



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**

Cons:

- CD that does not affect $\phi(\boldsymbol{x})$ are not perceivable
- In principle, changes not affecting $\phi(y|\boldsymbol{x})$ do not require reconfiguration.
- Difficult to design **sequential detection tools,** i.e., **change-detection tests** (CDTs) when streams are multivariate and distribution unknown

Monitoring

Change Detection Test

Change Detection: Problem Formulation
.. In a statistical framework

Process Changes

Normal data are generated in **stationary** conditions, i.e. **are i.i.d. realizations** of a process $\mathcal{P}_N$

After the change, **data are** generated from a **different process** $\mathcal{P}_A \neq \mathcal{P}_N$, which **persists over time**

Examples:

- Quality inspection system: **faults** producing flawed components
- Environmental monitoring: **persistent changes in the morphology** of measured signals
- Change of **user interests** in on-demand platform

Change-Detection in a Statistical Framework

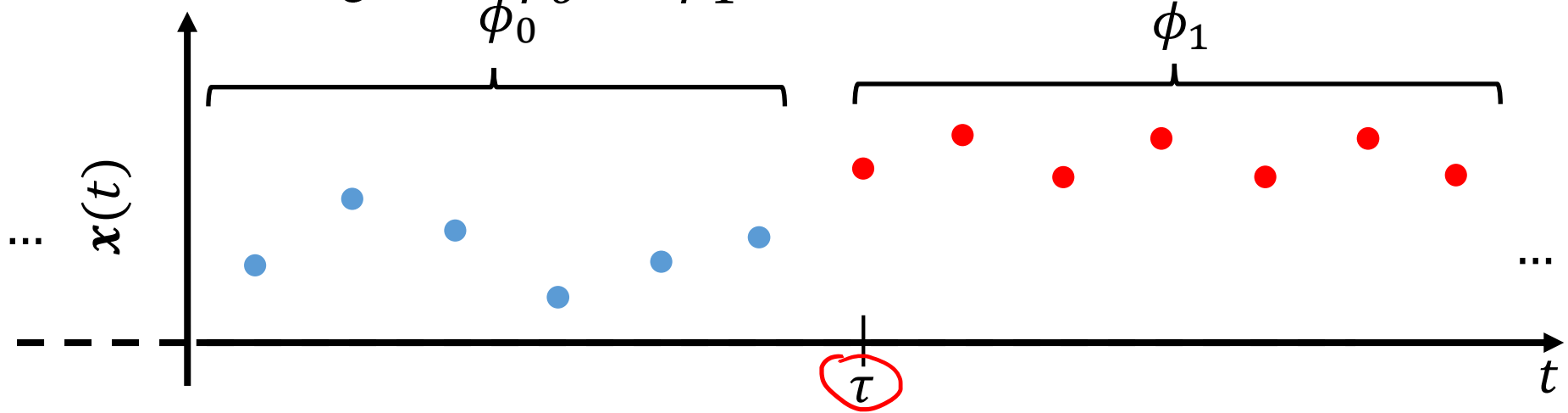
Often, the change-detection problem **boils down to:**

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 in a Statistical Framework

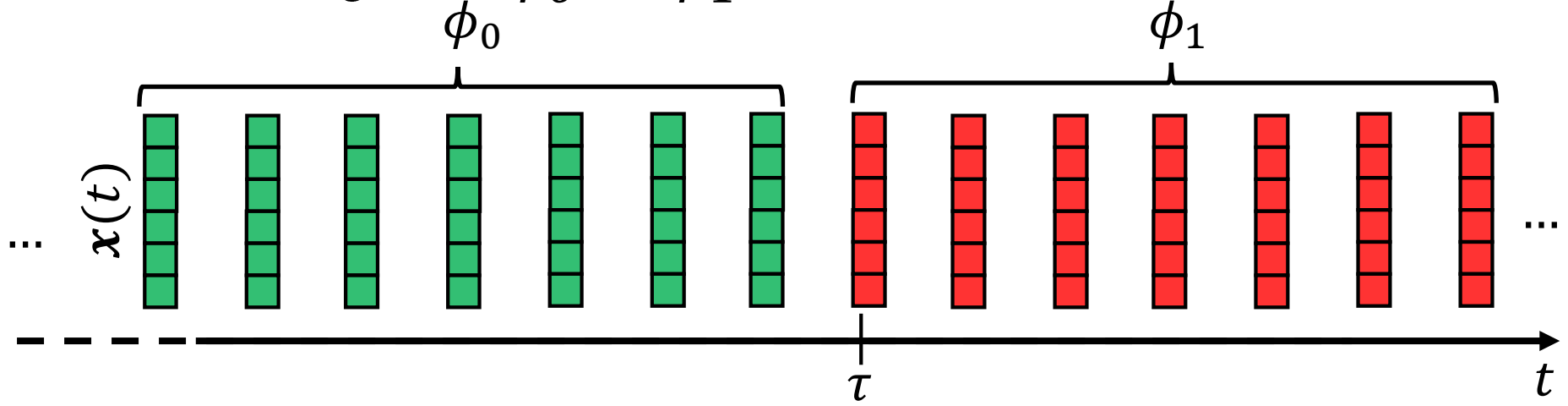
Often, the change-detection problem **boils down to:**

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 $\phi_0 \neq \phi_1$

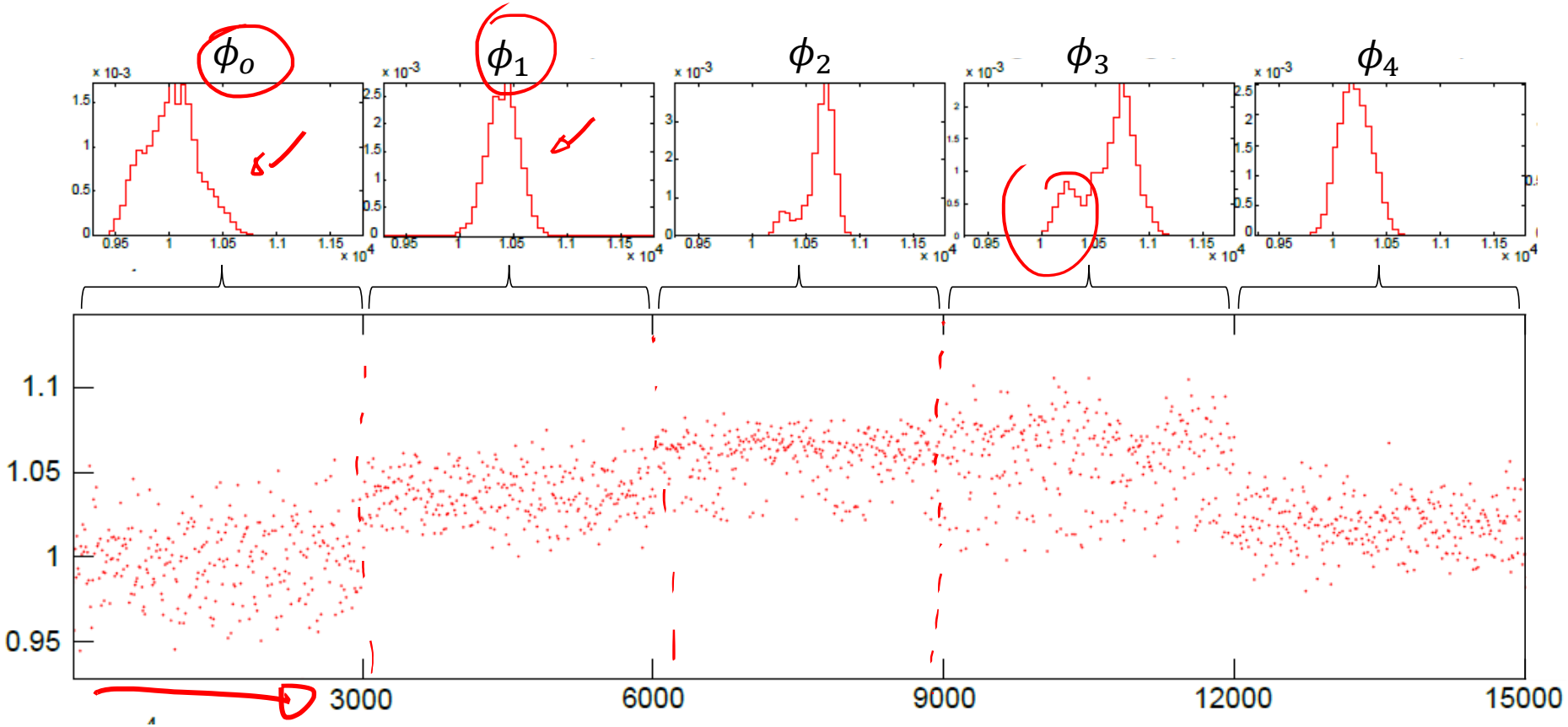
We denote such change as: $\phi_o \rightarrow \phi_1$



Change-Detection in a Statistical Framework

Here are data from an X-ray monitoring apparatus.

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



Boracchi, Trovò

Anomalies

"Anomalies are patterns in data that do not conform to a well defined notion of normal behavior"

Thus:

- **Normal data** are generated from a **stationary process** $\mathcal{P}_N$
- **Anomalies** are from a **different process** $\mathcal{P}_A \neq \mathcal{P}_N$

Examples:

- **Frauds** in the stream of all the credit card transactions
- **Arrhythmias** in ECG tracings
- **Defective regions in an image**, which do not conform a reference pattern

Anomalies might appear as **spurious** elements, and are typically the most **informative** samples in the stream

V. Chandola, A. Banerjee, V. Kumar. "*Anomaly detection: A survey*". ACM Comput. Surv. 41, 3, Article 15 (July 2009), 58 pages.

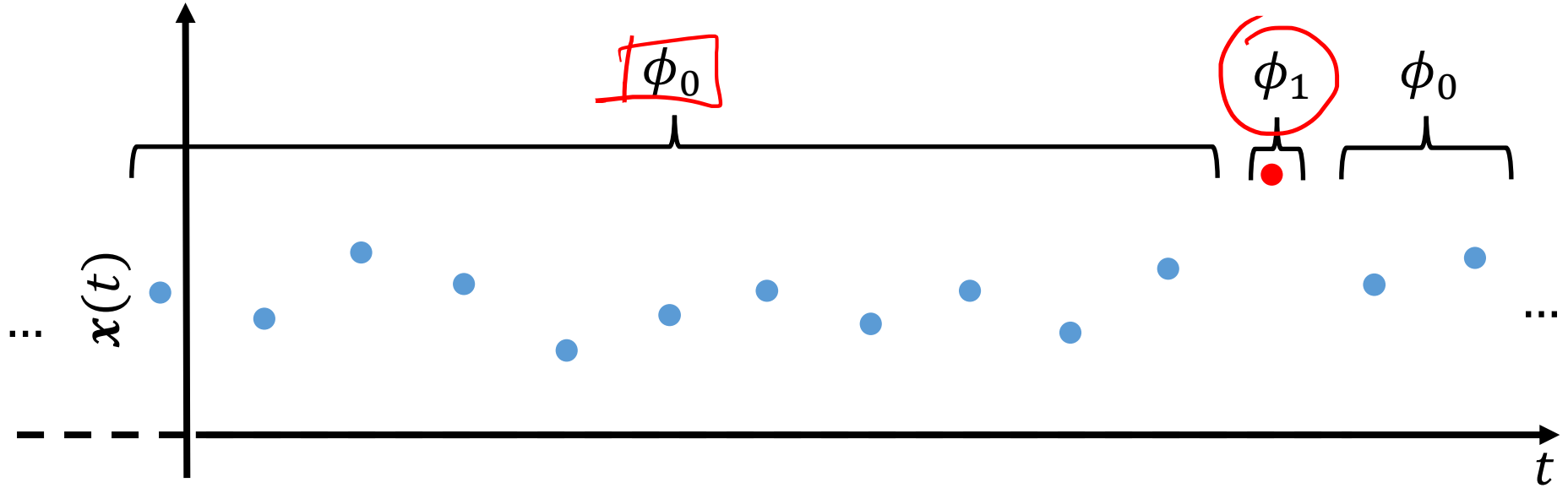
Anomaly-Detection in a statistical framework

Often, the anomaly-detection problem **boils down to monitoring a stream**

$$\{\boldsymbol{x}(t), \quad t = t_0, \dots\}, \quad \boldsymbol{x}(t) \in \mathbb{R}^d$$

where $x(t)$ are realizations of a **random variable having pdf** ϕ_o, and detect those points that are **outliers** i.e.,

$$\boldsymbol{x}(t) \sim \begin{cases} \phi_0 & \text{normal data} \\ \phi_1 & \text{anomalies} \end{cases},$$



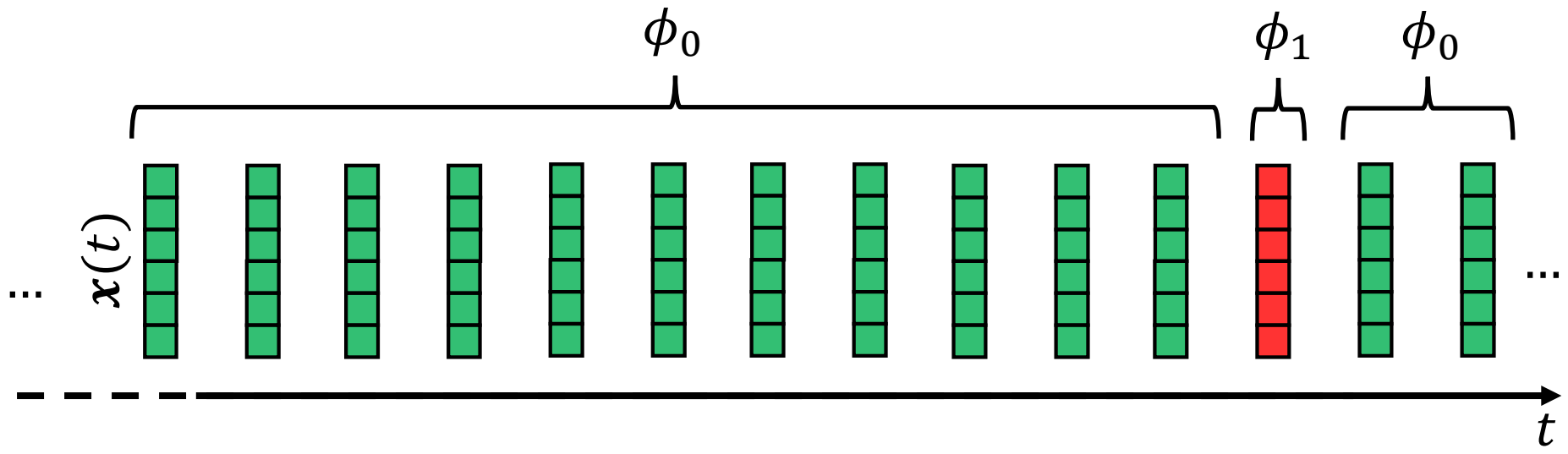
Anomaly-Detection in a statistical framework

Often, the anomaly-detection problem **boils down to monitoring a stream**

$$\{\boldsymbol{x}(t),\ t = t_0, \dots\},\quad \boldsymbol{x}(t) \in \mathbb{R}^d$$

where $x(t)$ are realizations of a **random variable having pdf** ϕ_o, and detect those points that are **outliers** i.e.,

$$\boldsymbol{x}(t) \sim \begin{cases} \phi_0 & \text{normal data} \\ \phi_1 & \text{anomalies} \end{cases},$$



The legal case of Mr Hadlum v. Mrs Hadlum (1949)

The sole evidence of adultery consisted of the birth of a child 349 days after Mr Hadlum had left for military service abroad.

Vic Barnett *"The Study of Outliers: Purpose and Model"* Journal of the Royal Statistical Society. Series C (Applied Statistics) Vol. 27, No. 3 (1978), pp. 242-250

The legal case of Mr Hadlum v. Mrs Hadlum (1949)

The sole evidence of adultery consisted of the birth of a child 349 days after Mr Hadlum had left for military service abroad.

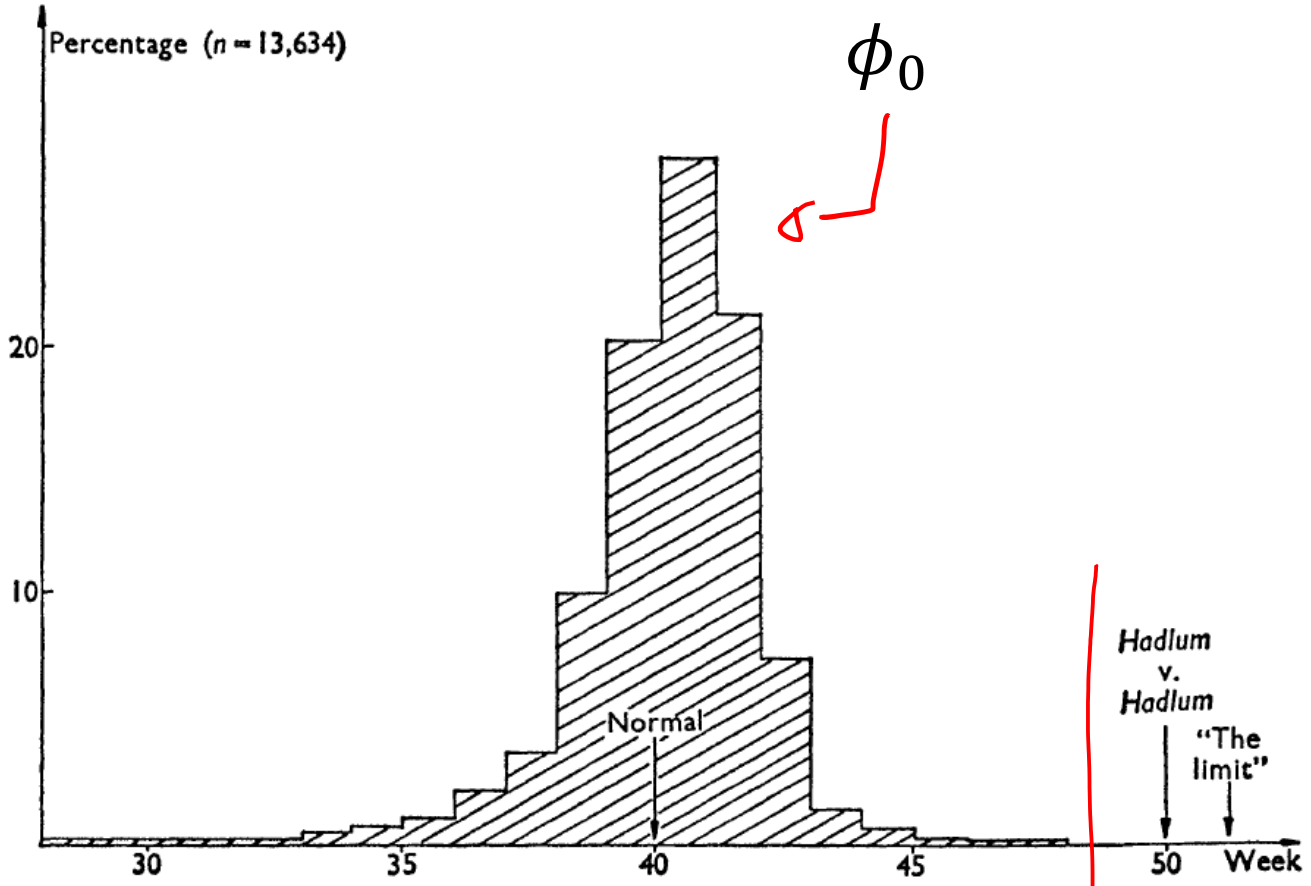


FIG. 1. Distribution of human gestation periods.

Vic Barnett *"The Study of Outliers: Purpose and Model"* Journal of the Royal Statistical Society. Series C (Applied Statistics) Vol. 27, No. 3 (1978), pp. 242-250

The legal case of Mr Hadlum v. Mrs Hadlum (1949)

The sole evidence of adultery consisted of the birth of a child 349 days after Mr Hadlum had left for military service abroad.

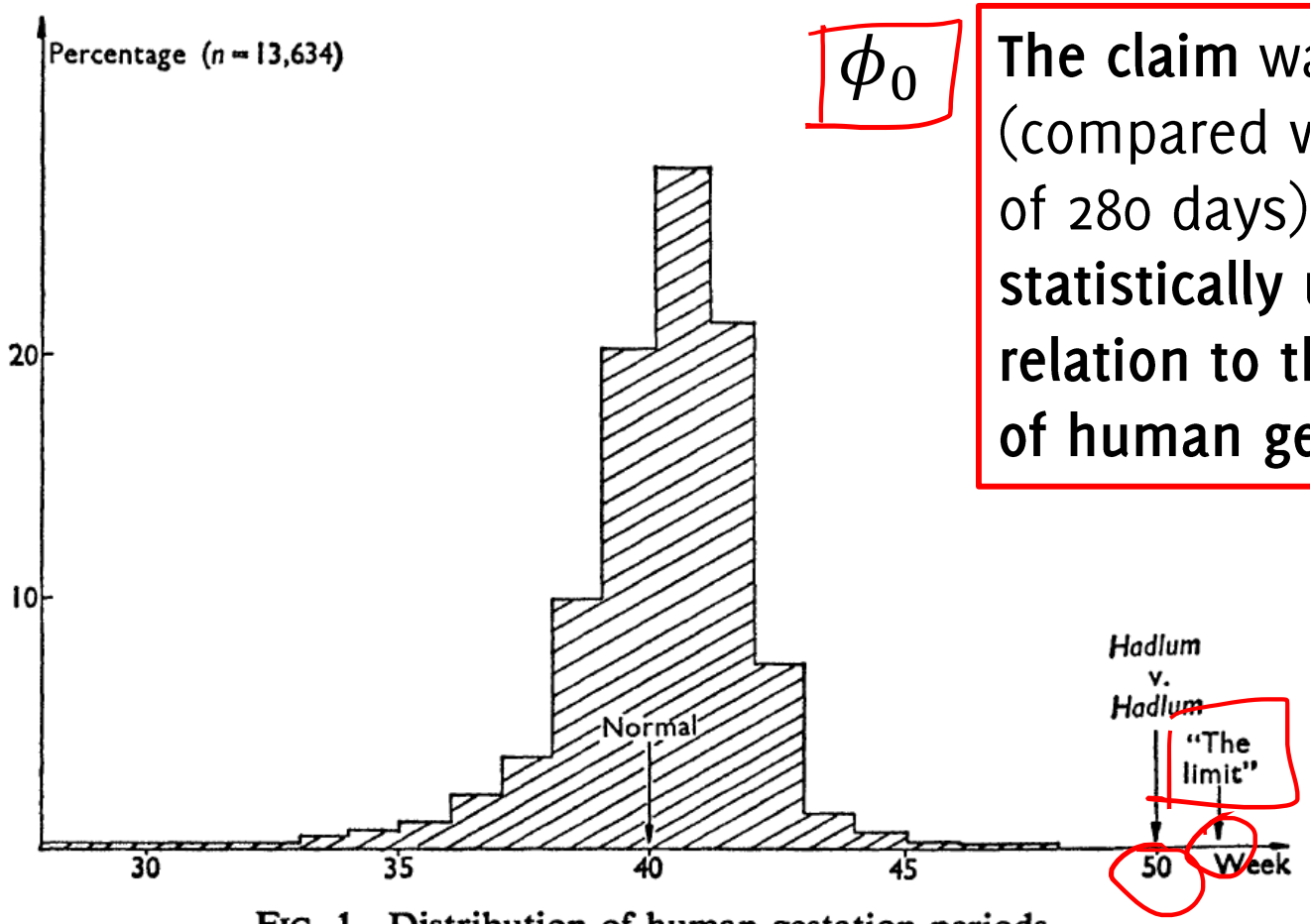


FIG. 1. Distribution of human gestation periods.

ϕ_0

The claim was that 349 days (compared with an average of 280 days) was discordant: **statistically unreasonable in relation to the distribution of human gestation periods.**

Vic Barnett *"The Study of Outliers: Purpose and Model"* Journal of the Royal Statistical Society. Series C (Applied Statistics) Vol. 27, No. 3 (1978), pp. 242-250

The legal case of Mr Hadlum v. Mrs Hadlum (1949)

The sole evidence of adultery consisted of the birth of a child 349 days after Mr Hadlum had left for military service abroad.

ϕ_0

The claim was that 349 days
h an average
as discordant:
reasonable in
distribution
tation periods.

Credits:
"Outlier Detection Techniques"
Hans-Peter Kriegel, Peer Kröger, Arthur Zimek

Tutorial in 2010 SIAM International Conference on Data Mining

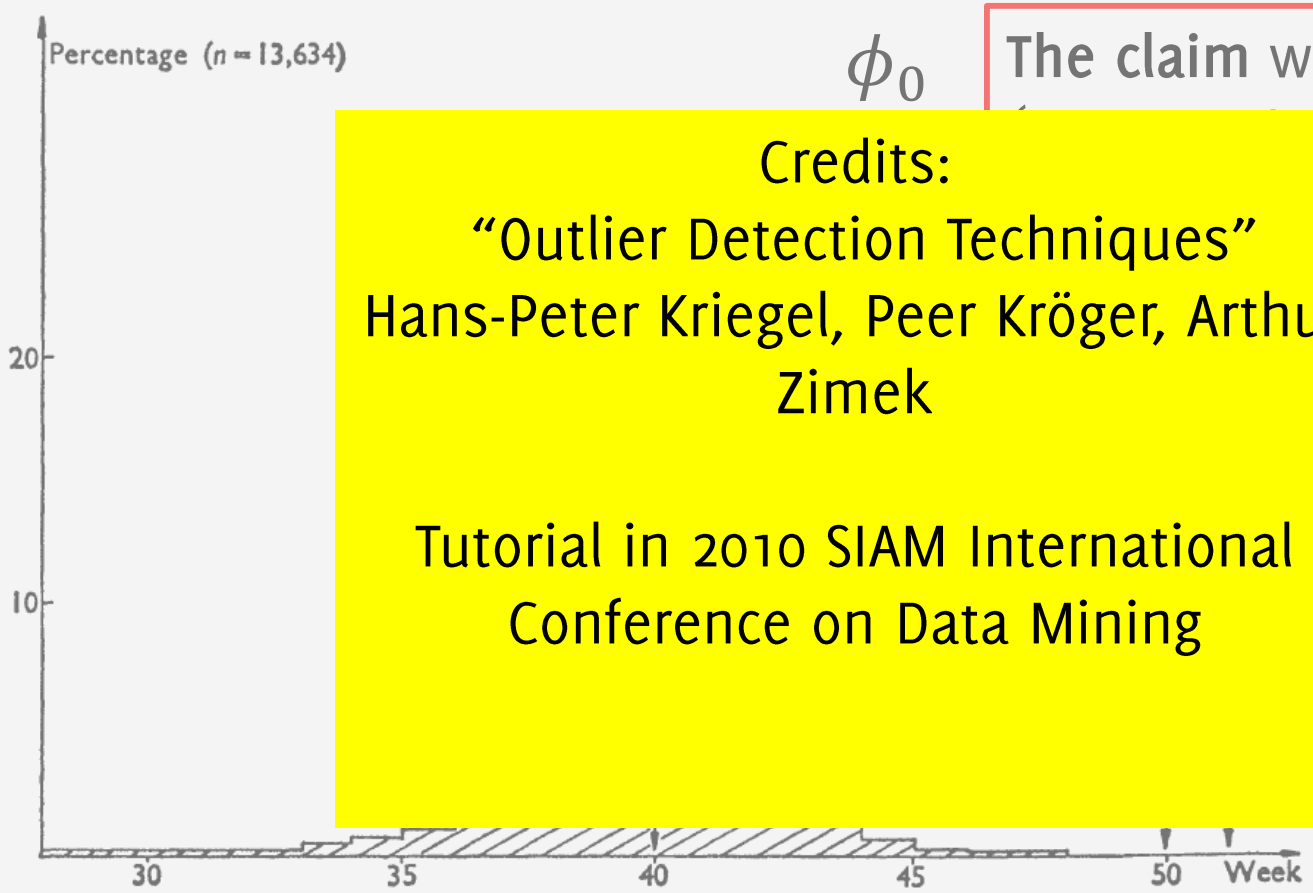
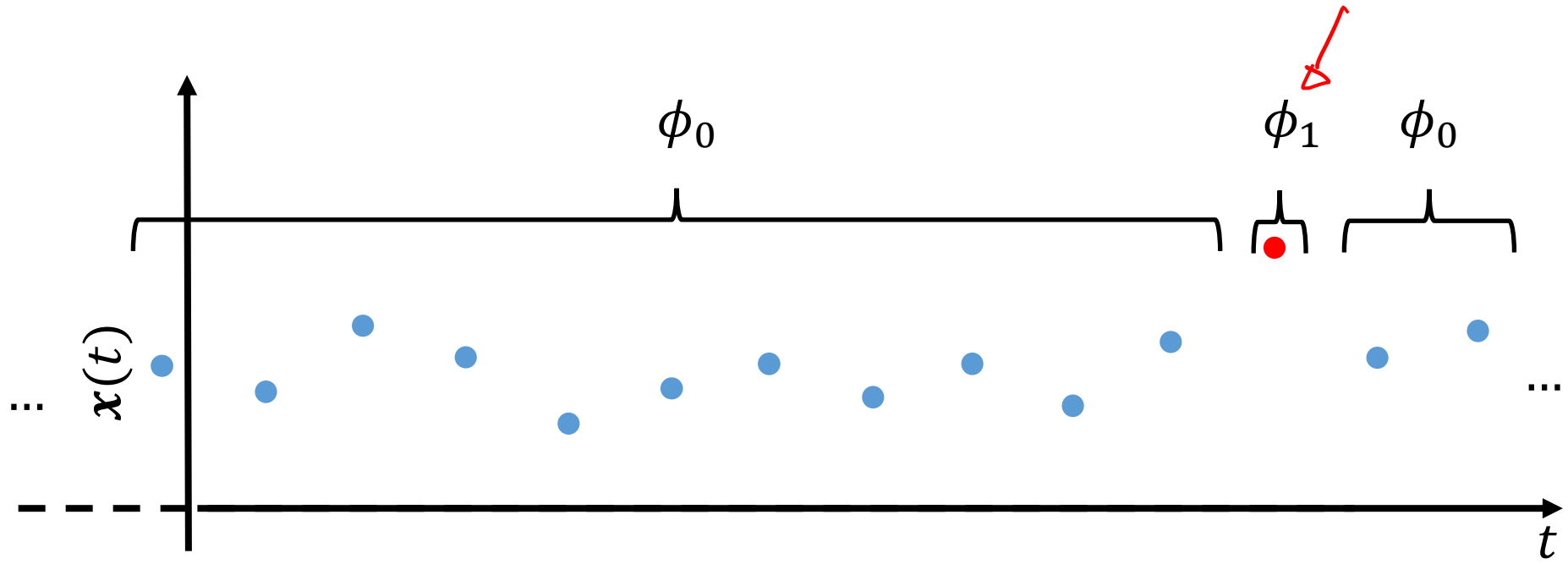


FIG. 1. Distribution of human gestation periods.

Vic Barnett *"The Study of Outliers: Purpose and Model"* Journal of the Royal Statistical Society. Series C (Applied Statistics) Vol. 27, No. 3 (1978), pp. 242-250

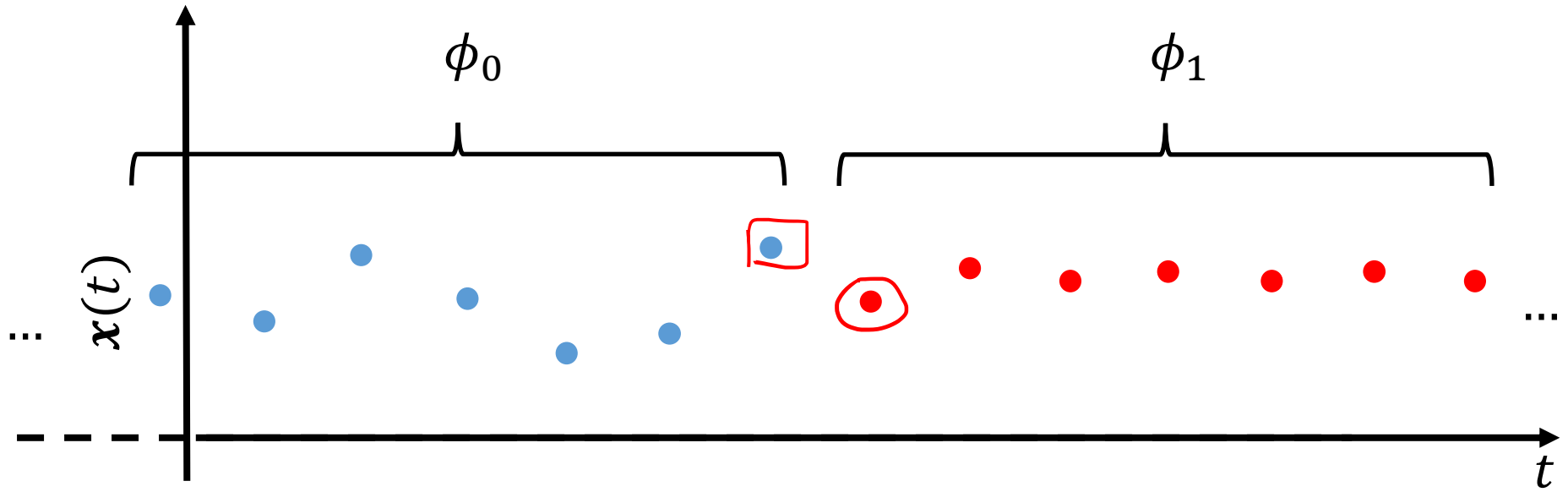
Process changes vs anomalies

Not all anomalies are due to process changes



Process changes vs anomalies

Not all process changes result in anomalies



General Approaches

To detect changes and anomalies

Change / Anomaly Detection Questions

Change-detection question:

Given the previously estimated model, the arrival of new data invites the question: "Is yesterday's model capable of explaining today's data?"

Detecting process changes is important to understand the monitored phenomenon

V. Chandola, A. Banerjee, V. Kumar. "*Anomaly detection: A survey*". ACM Comput. Surv. 41, 3, Article 15 (2009), 58 pages.

C. J. Chu, M. Stinchcombe, H. White "*Monitoring Structural Change*" Econometrica Vol. 64, No. 5 (Sep., 1996), pp. 1045-1065.

Change / Anomaly Detection Questions

Change-detection question:

Given the previously estimated model, the arrival of new data invites the question: "Is yesterday's model capable of explaining today's data?"

Detecting process changes is important to understand the monitored phenomenon

Anomaly-detection question:

Locate those samples that do not conform the normal ones or a model explaining normal ones

Anomalies in data translate to significant information

V. Chandola, A. Banerjee, V. Kumar. "*Anomaly detection: A survey*". ACM Comput. Surv. 41, 3, Article 15 (2009), 58 pages.

C. J. Chu, M. Stinchcombe, H. White "*Monitoring Structural Change*" Econometrica Vol. 64, No. 5 (Sep., 1996), pp. 1045-1065.

The Typical Solutions

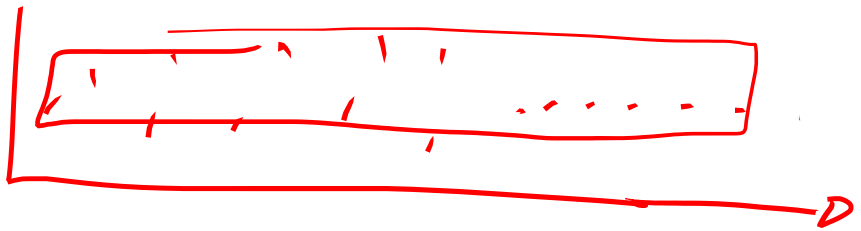
Most algorithms are composed of:

- A **statistic** that has a known response to normal data (e.g., the average, the sample variance, the log-likelihood, the confidence of a classifier, an "anomaly score"…)
- A **decision rule** to analyze the statistic (e.g., an adaptive threshold, a confidence region)

Statistics and Decision Rules

Change-detection algorithms:

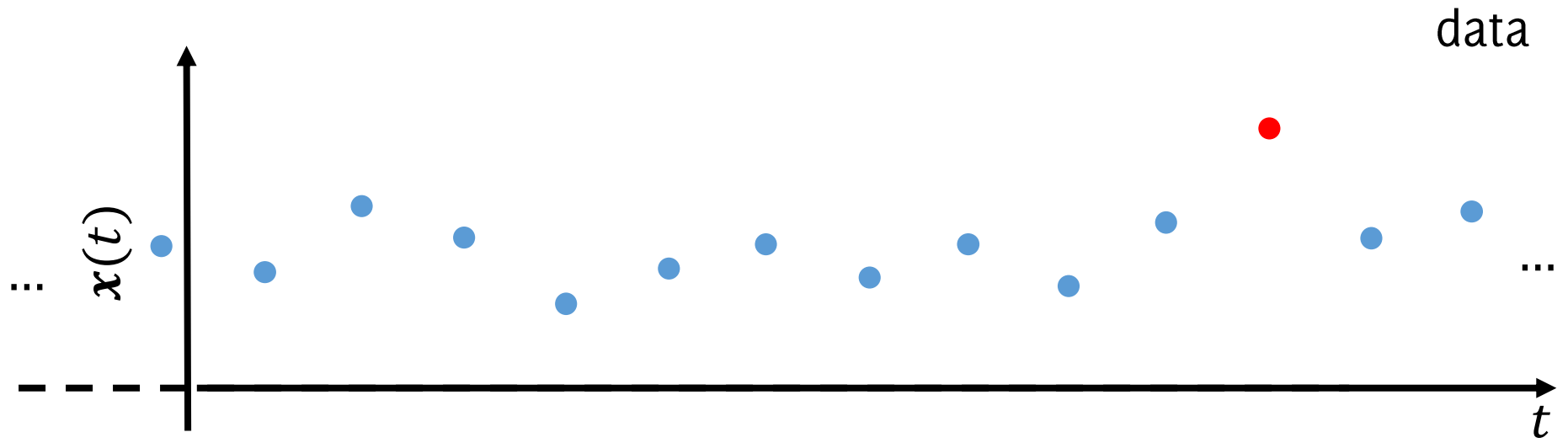
Statistics and decision rules are **sequential**, as they make a decision considering --in principle-- all the data received so far. Integrating information over time makes these algorithms able to detect subtle changes as well



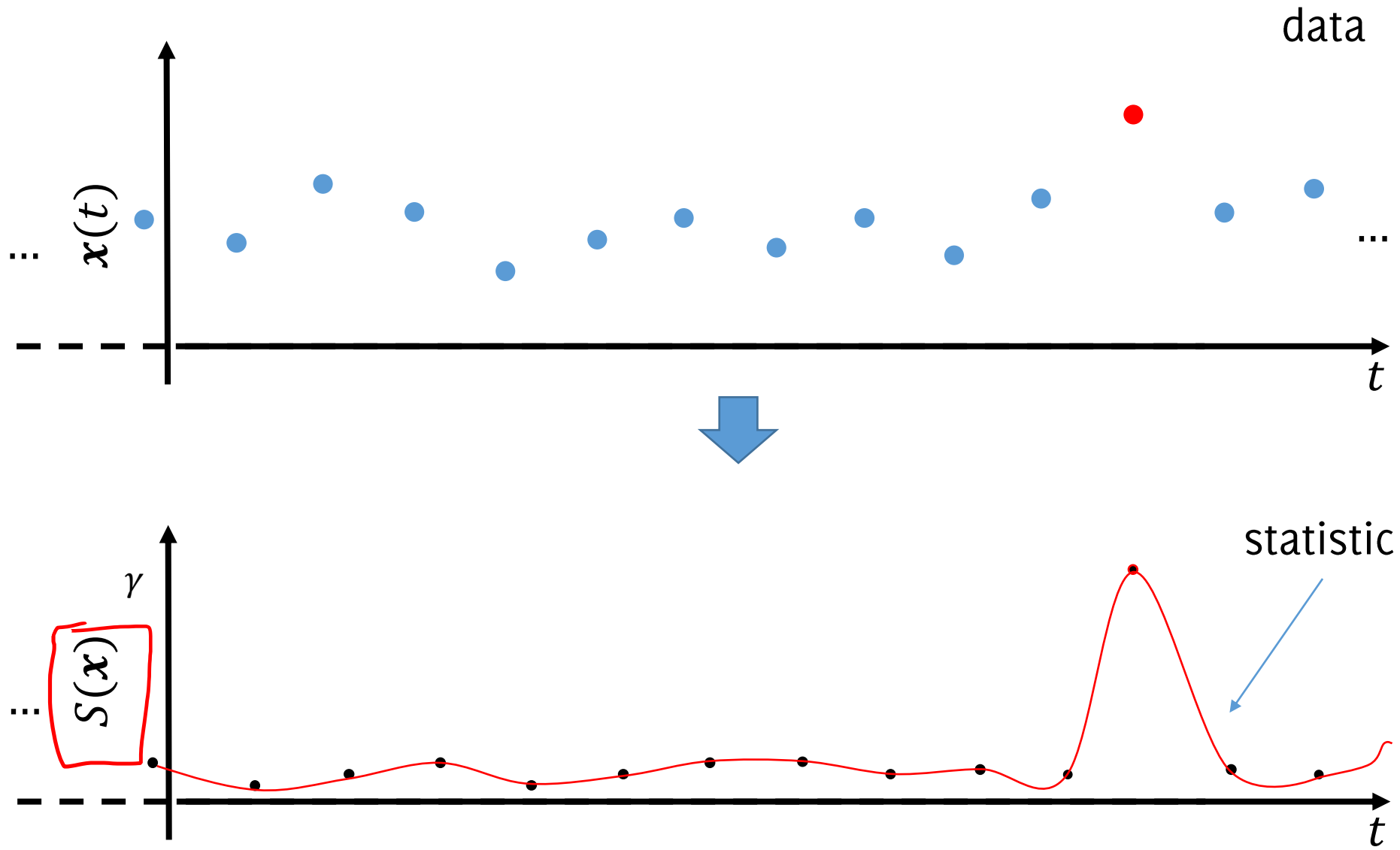
Anomaly-detection algorithms:

Statistics and decision rules are **“one-shot”**, analyzing a set of historical data or each new data --or chunk– independently. Different samples are not jointly analyzed.

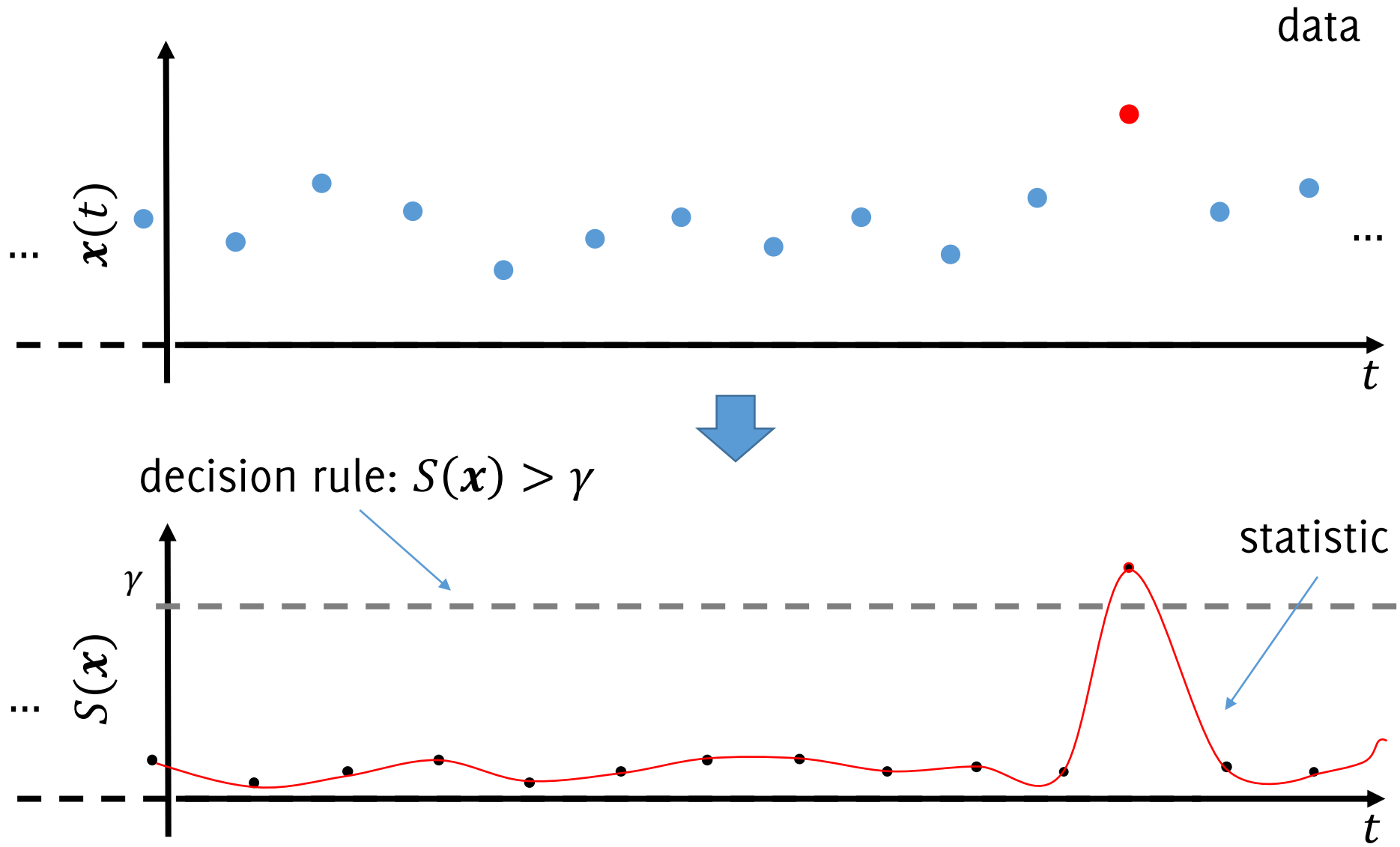
The Typical Solutions



The Typical Solutions

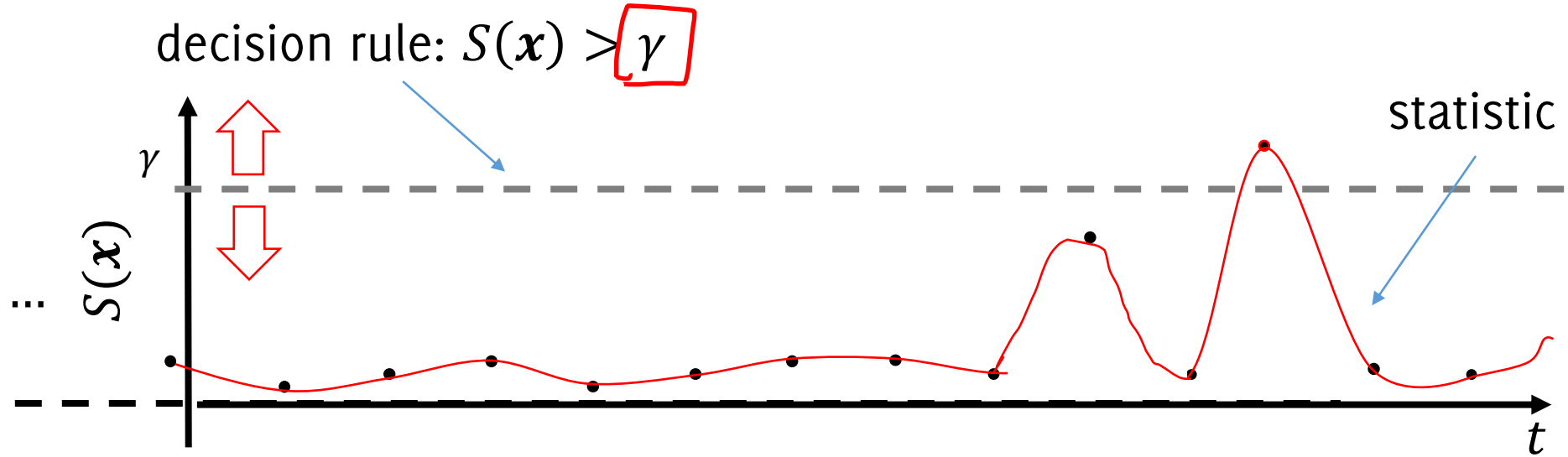


The Typical Solutions



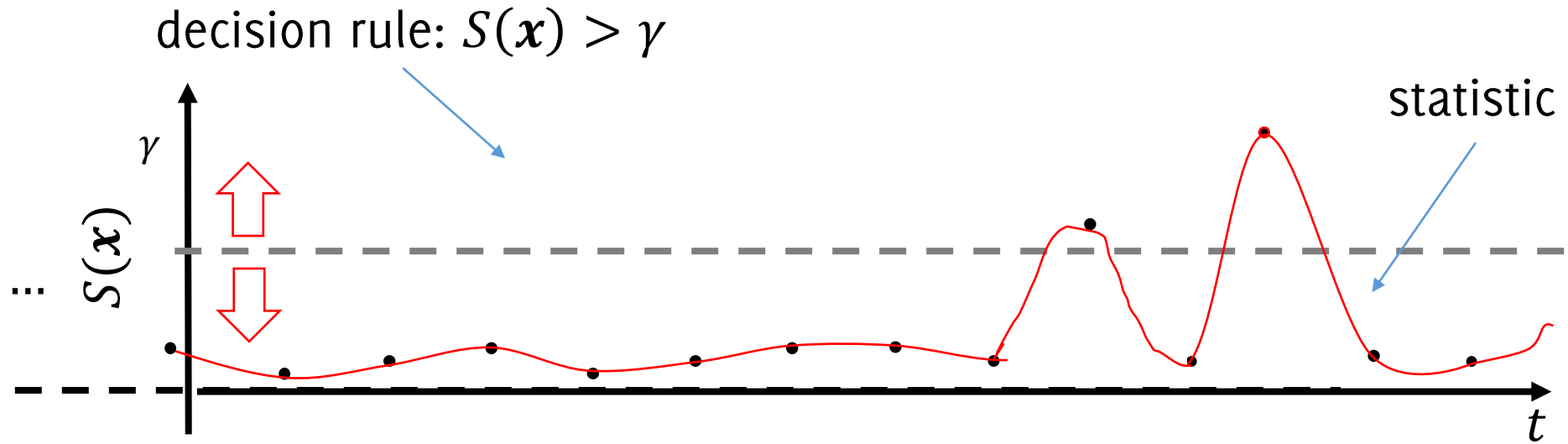
The Typical Change-Detection Solution

By changing γ it is possible to achieve different detection performance (e.g. more true positive, more false positives)



The Typical Change-Detection Solution

By changing γ it is possible to achieve different detection performance (e.g. more true positive, more false positives)



Statistics and Decision Rules

Detection rules often **rely on thresholds,** namely γ

In both these cases: It is of primary concern to control *false positives,* namely "how often" a change/anomaly is detected within stationary data.

Controlling False Positives

Controlling False Positives in Anomaly Detection

In anomaly detection the notion of False Positive Rates corresponds to type I errors in hypothesis testing (the probability of rejecting null hypothesis when this holds).

Let the decision rule be $S(\boldsymbol{x}) > \gamma$, type I error is the probability of detecting an anomaly when data are stationary

$$P(S(\boldsymbol{x}) > \gamma | \boldsymbol{x} \sim \phi_0)$$

$P(\cdot)$ denotes probability of an event

$$\text{FPR} : \frac{\#\{S(x) > \gamma \mid x \sim \phi_0\}}{N}$$

Controlling False Positives in Anomaly Detection

A good anomaly-detection algorithm is accompanied with a table/criteria/formula that, provided a maximum acceptable FPR, it provides the corresponding threshold γ

$$\alpha = 0.01 \rightarrow \gamma = ?$$

How to define this threshold?

- When the distribution of S is known, the threshold γ corresponds to the $\alpha - th$ quantile of the distribution (watch out, the distribution of S might in principle depend on ϕ_0)
- When enough stationary data are provided for training, one can just resort to **bootstrap**.

Bootstrapping

Any test or metric that relies on **random sampling with replacement.**

The bootstrap is a flexible and powerful statistical tool that can be used to **quantify the uncertainty associated with a given estimator or statistical learning method**. Primarily used to obtain standard errors of an estimator.

We adopt bootstrapping to estimate the **empirical distribution function of the observed data** and its quantiles.

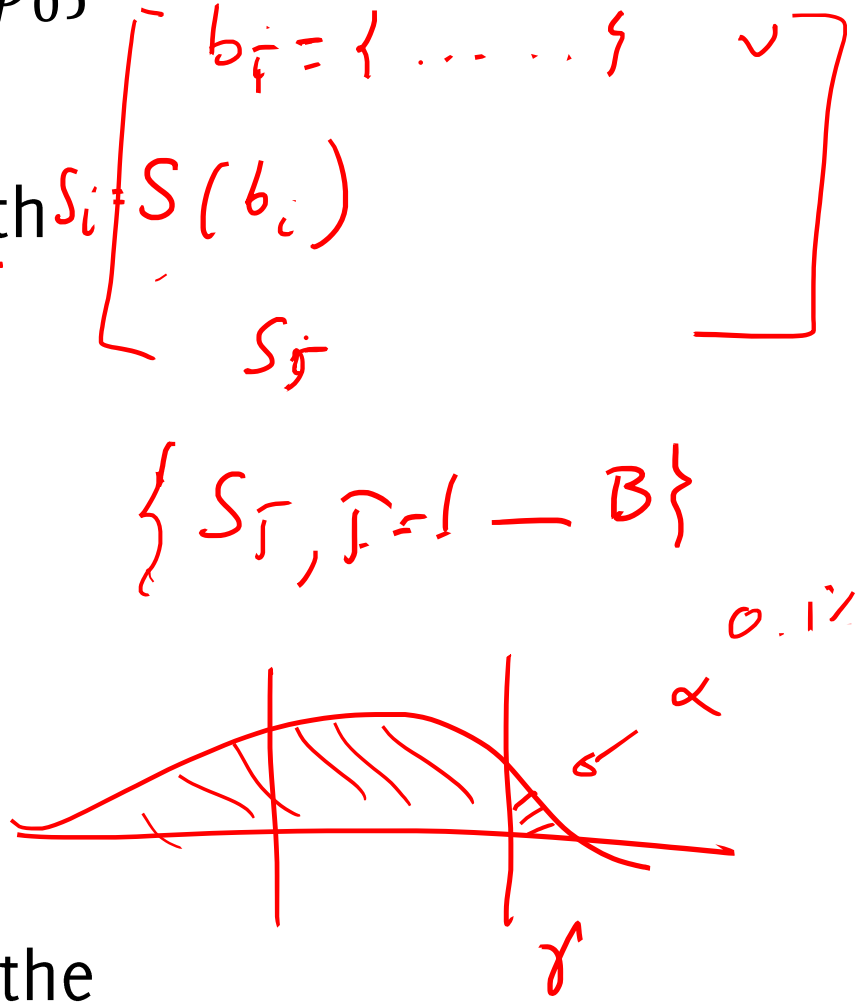
Bootstrapping for Threshold Estimation

Given the training set $TR = \{\boldsymbol{x}_i, i = 1, \ldots, N, \boldsymbol{x} \sim \phi_0\}$

Iterate B times

- Extract from TR a random batch b_j of size ν with replacement
- Compute the test statistic S_j over the batch b_j
- Store the value of the statistics S_j

At the end, compute the $\alpha - \text{th}$ quantile from bootstrap estimates $\{S_j, j = 1, \ldots, B\}$



This quantile is a good estimate of the quantile of the distribution of the test statistic under ϕ_0

Controlling False Positives in Change Detection

In change-detection false positives are controlled by the Average Run Length ARL_0 :

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

Thus **denotes the expected time between false positive detections**

Controlling False Positives in Change Detection

A good change-detection test is accompanied with a table/rule/formula that defines, for a given expected value of ARL_0 the corresponding threshold γ

$$\phi_x^0 ARL_0 = 5000 \rightarrow \gamma?$$

Watch out: thresholds depend on the statistics S, **which in turn might depend on the distribution of the monitored data** ϕ_x^0

Threshold computation for change-detection algorithm is more complicated than in anomaly-detection algorithm since bootstrap procedure has to consider temporal evolution of the analysis

Anomaly/Change Detection in the Ideal Settings

...when ϕ_0 and ϕ_1 are known

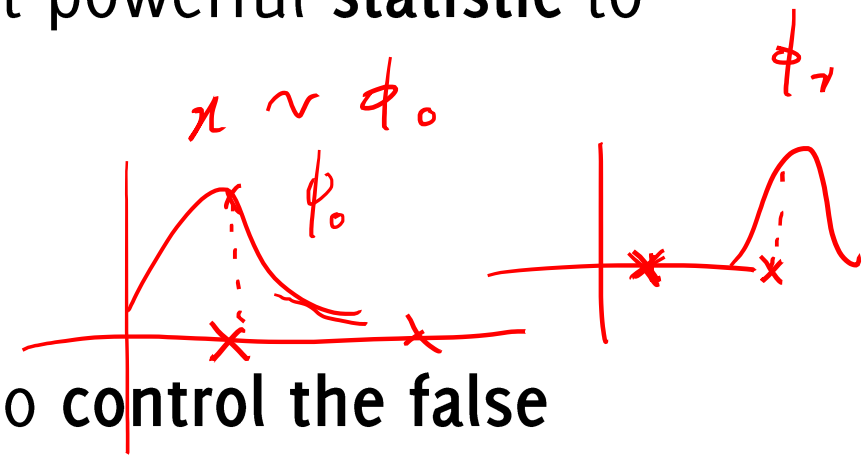
One-shot detector: Newman Pearson test

Assume data are generated from a parametric distribution ϕ_θ and formulate the following hypothesis test

$$H_0: \theta = \theta_0 \text{ vs } H_1: \theta = \theta_1$$

According to the Neumann Pearson lemma, the most powerful **statistic** to detect changes is the **likelihood ratio**

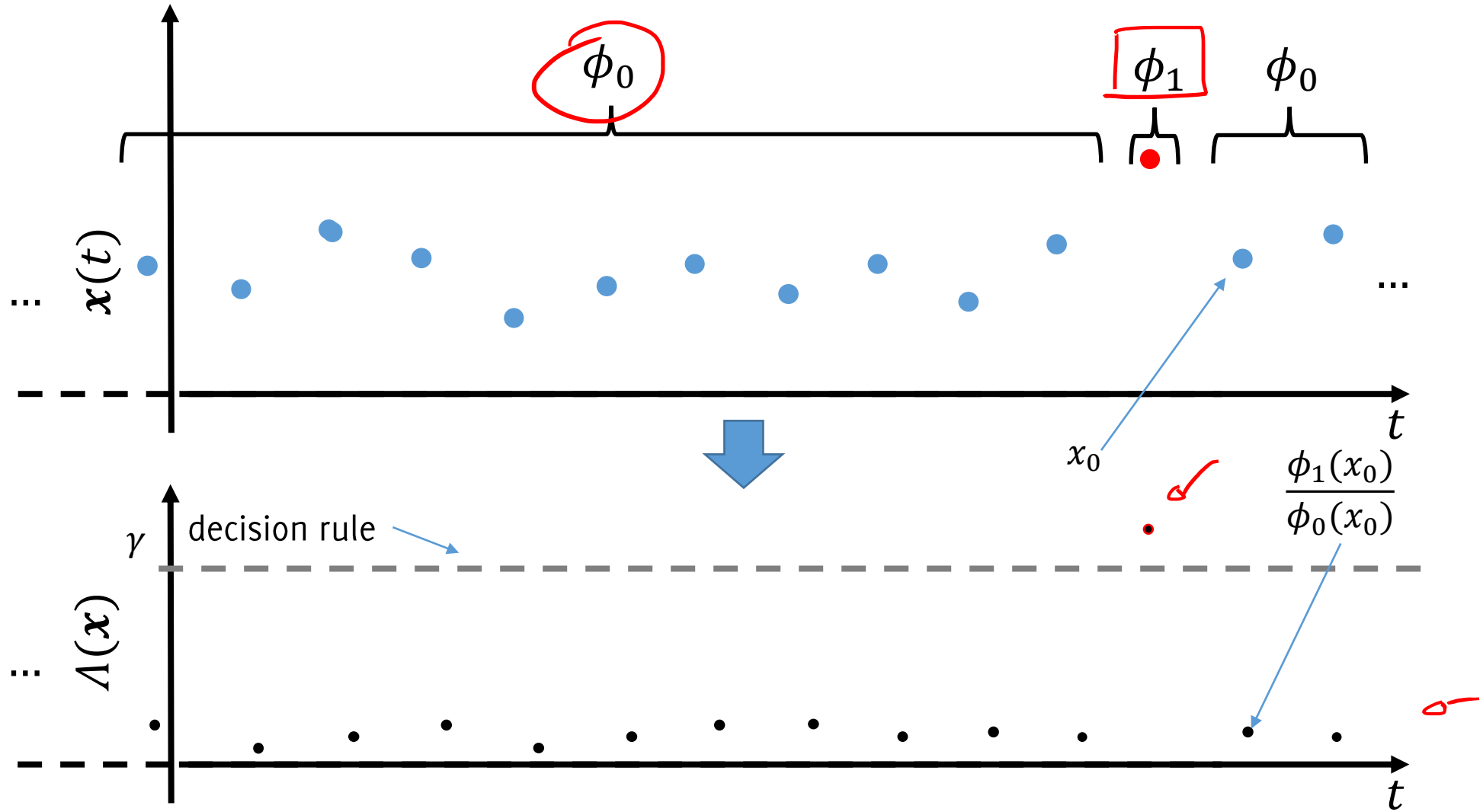
$$\Lambda(x) = \frac{\phi_1(x)}{\phi_0(x)}$$



and the **detection rule** is $\Lambda(x) > \gamma$, where γ is set to **control the false alarm rate** (type I errors of the test).

One-shot detector: Newman Pearson test

Outliers can be detected by a threshold on $\Lambda(\boldsymbol{x})$



The CUSUM test on the likelihood ratio

CUSUM involves the calculation of a **CU**mulative **SUM**, which makes it a sequential monitoring scheme.

It can be applied to the log-likelihood ratio:

$$\log\big(\Lambda(x)\big) = \log\left(\frac{\phi_1(x)}{\phi_0(x)}\right) = \begin{cases} < 0 & \text{when } \phi_0(x) > \phi_1(x) \\ > 0 & \text{otherwise} \end{cases}$$

The CUSUM statistic is:

$$S(t) = \max\Big(0, S(t-1) + \log\big(\Lambda(x(t))\big)\Big)$$

And the decision rule is

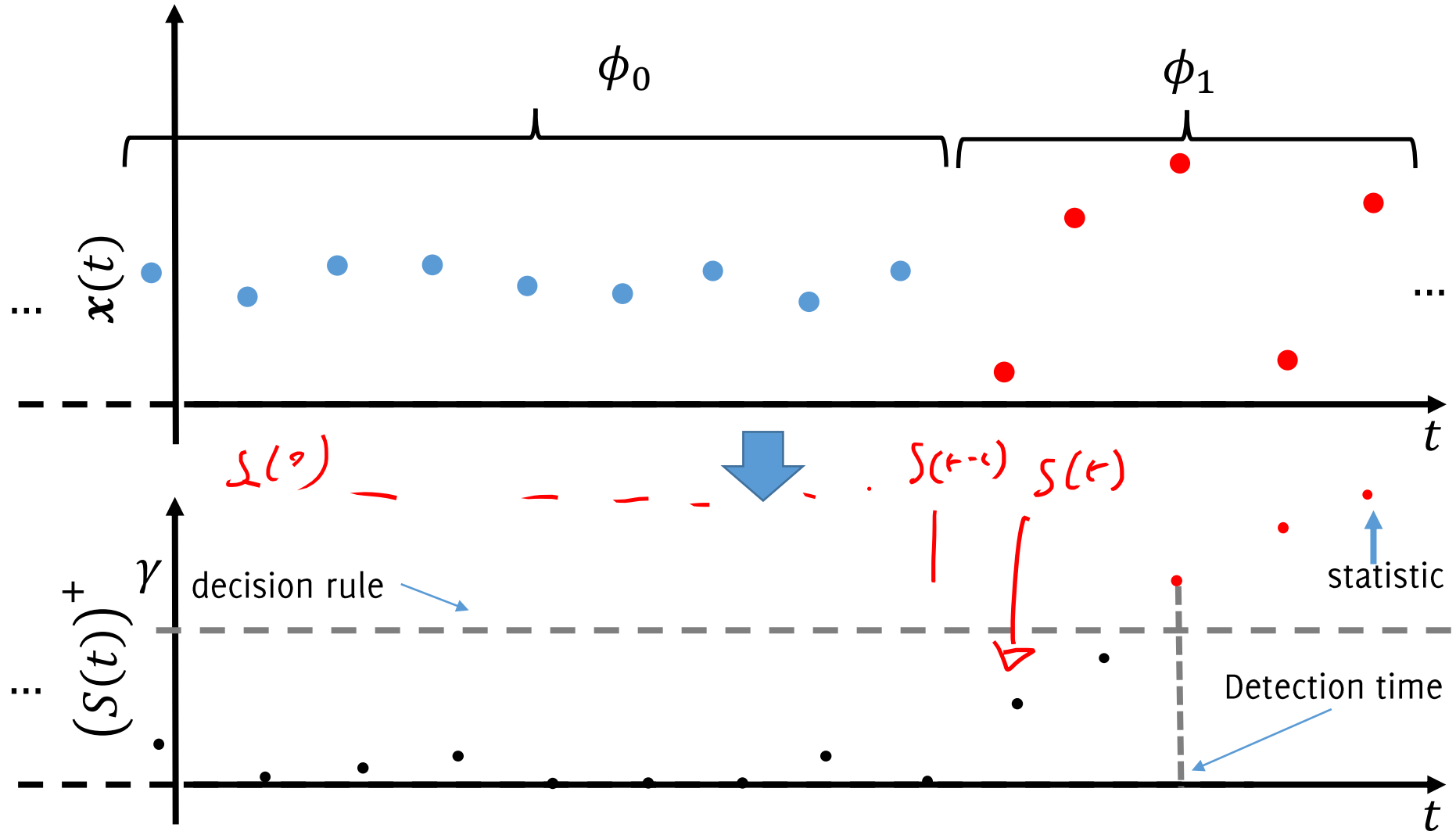
$$S(t) > \gamma$$

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

Page, E. S. "*Continuous Inspection Scheme*". Biometrika. 41 (1/2): 100–115 (June 1954).

CUSUM test

Outliers can be detected by a threshold on $\Lambda(\boldsymbol{x})$

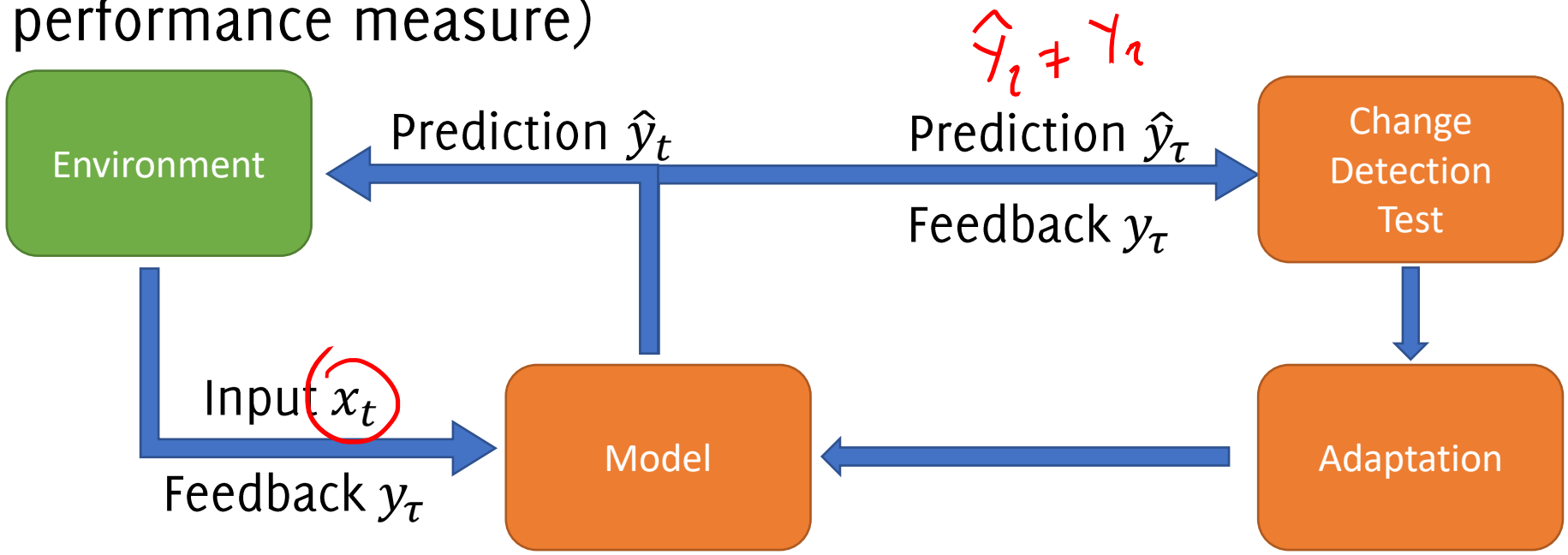


Learning in NSE by Monitoring the Classification Error

...when ϕ_0 and ϕ_1 are "unknown"

Monitoring the Classification Error

The simplest approach consist in monitoring the classification error (or similar performance measure)



Pro:

- It is the most straightforward figure of merit to monitor
- Changes in p_t prompts adaptation only when performance are affected

Cons:

- CD detection from supervised samples only

Monitoring the Classification Error

The element-wise classification error follows a Bernoulli pdf

$$e_t \sim \text{Bernulli}(\pi_0)$$

π_0 is the expected classification error in stationary conditions

The sum of e_t in a window follows a Binomial pdf

$$\sum_{t=T-\nu}^{T} e_t \sim \mathcal{B}(\pi_0, \nu)$$

Gaussian approximation holds when ν is sufficiently large

$$p_t = \frac{1}{\nu} \sum_{t=T-\nu}^{T} e_t \sim \frac{1}{\nu} \mathcal{B}(p_t, \nu) \approx \mathcal{N}\left(p_t, \frac{p_t(1-p_t)}{\nu}\right)$$

We have a sequence of i.i.d. Gaussian distributed values

Monitoring the Classification Error: DDM

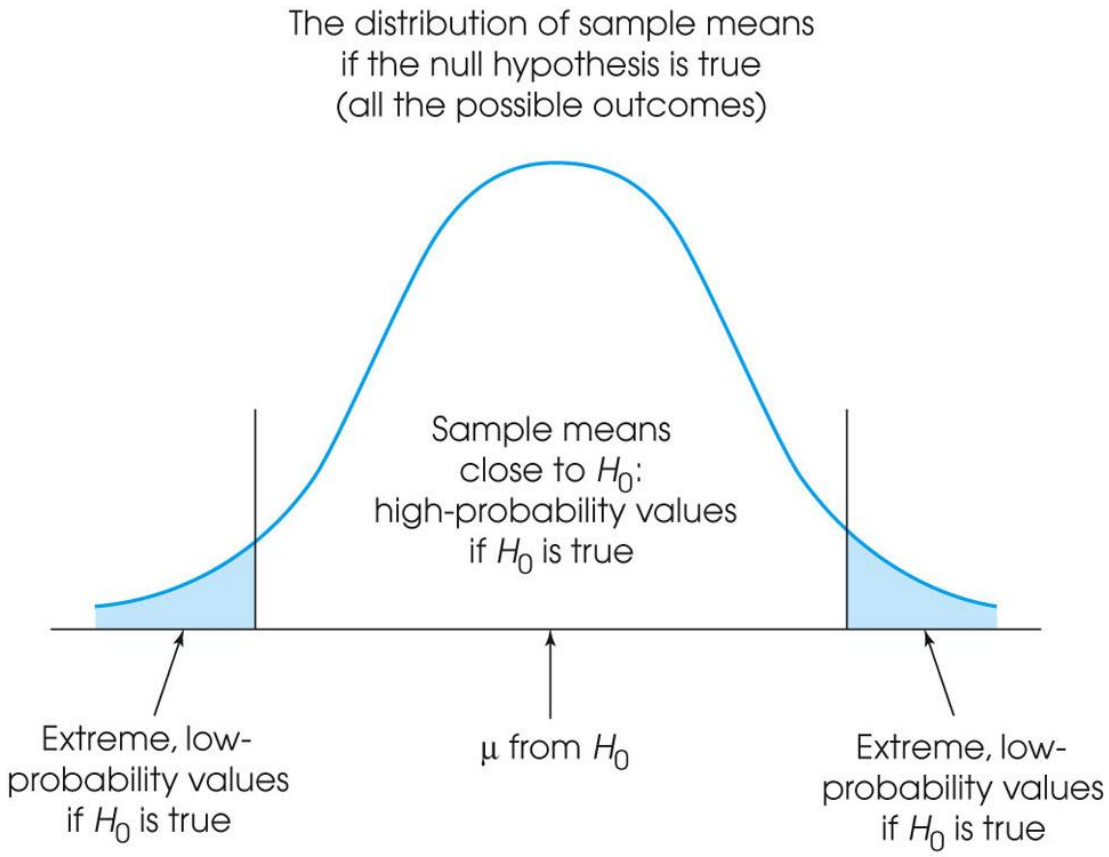
Basic idea behind Drift Detection Method (DDM):

J. Gama, P. Medas, G. Castillo, and P. Rodrigues. *"Learning with Drift Detection"* SBIA. Springer, Berlin, 286–295, 2004

Monitoring the Classification Error: DDM

Basic idea behind Drift Detection Method (DDM):

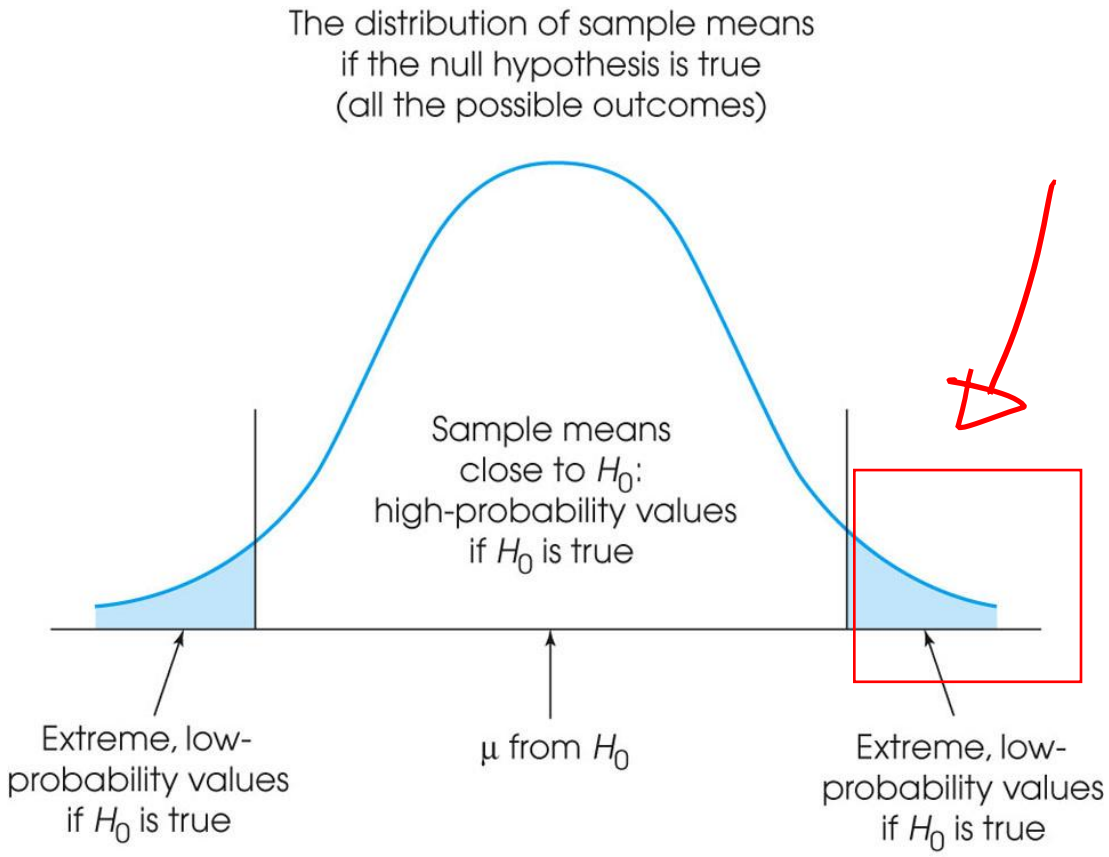
- Detect concept drift as an **outlier** in the classification error



Monitoring the Classification Error: DDM

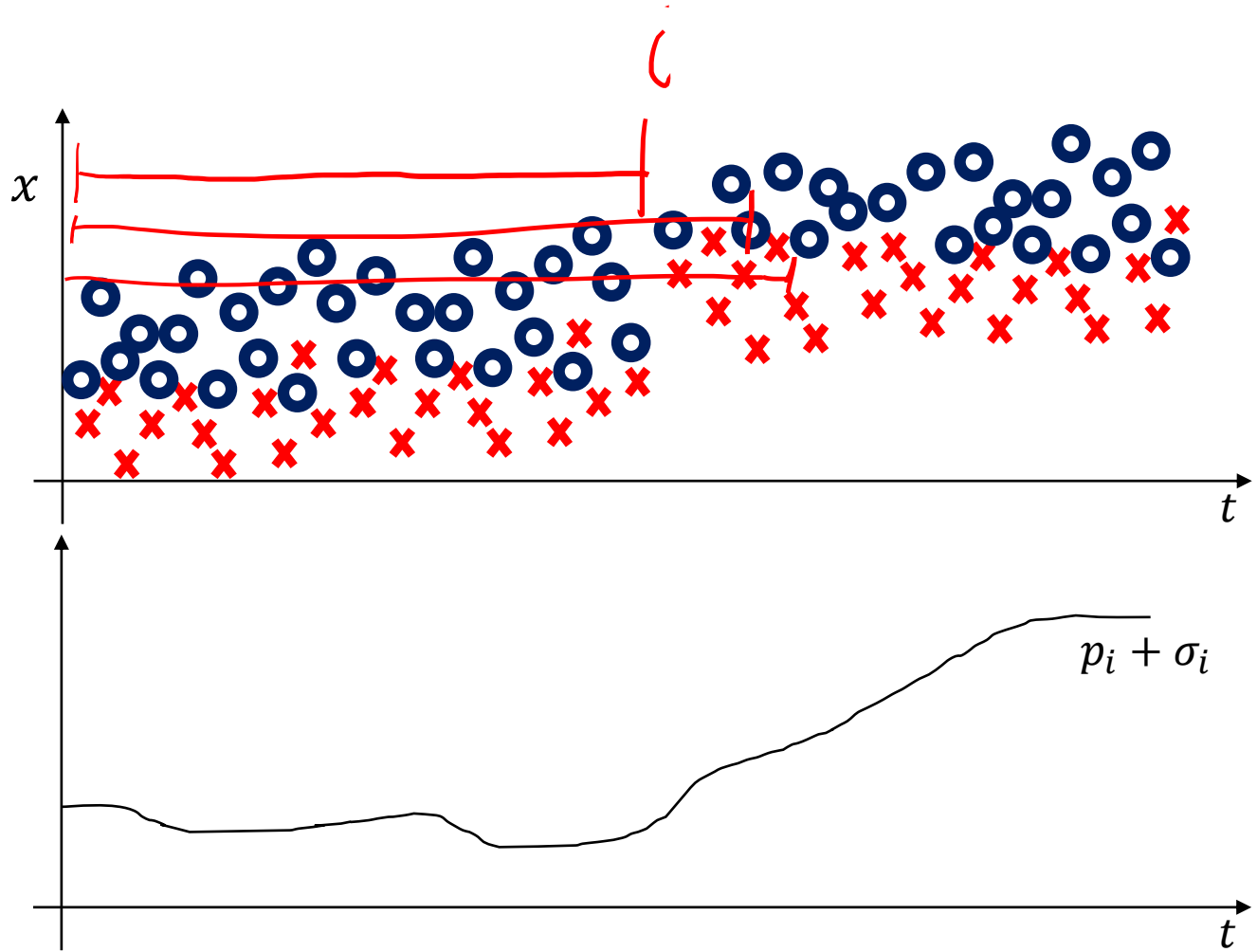
Basic idea behind Drift Detection Method (DDM):

- Detect concept drift as an **outlier** in the classification error
- In stationary conditions error decreases, **look for outliers** in the right tail



Monitoring the Classification Error: DDM

1. During monitoring, steadily compute p_i and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$

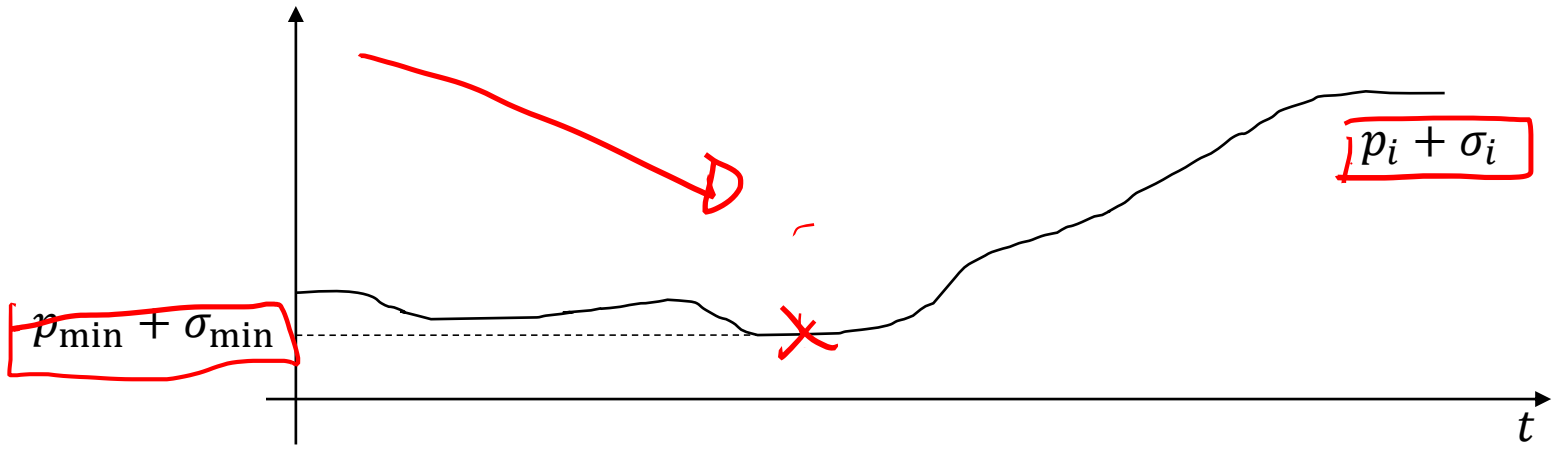


Monitoring the Classification Error: DDM

1. During monitoring, steadily compute p_i and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$

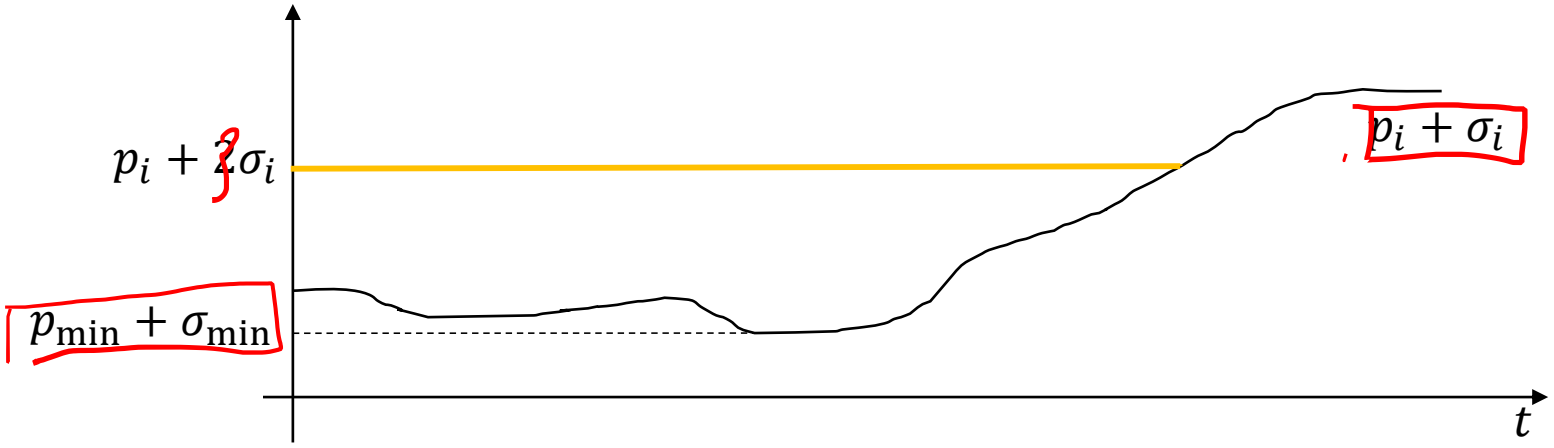
   $\{p_i\}$ ↓ not i.i.d.

2. Let $p_{\min}$ be the minimum error before i and $\sigma_{\min} = \sqrt{\frac{p_{\min}(1-p_{\min})}{i}}$



Monitoring the Classification Error: DDM

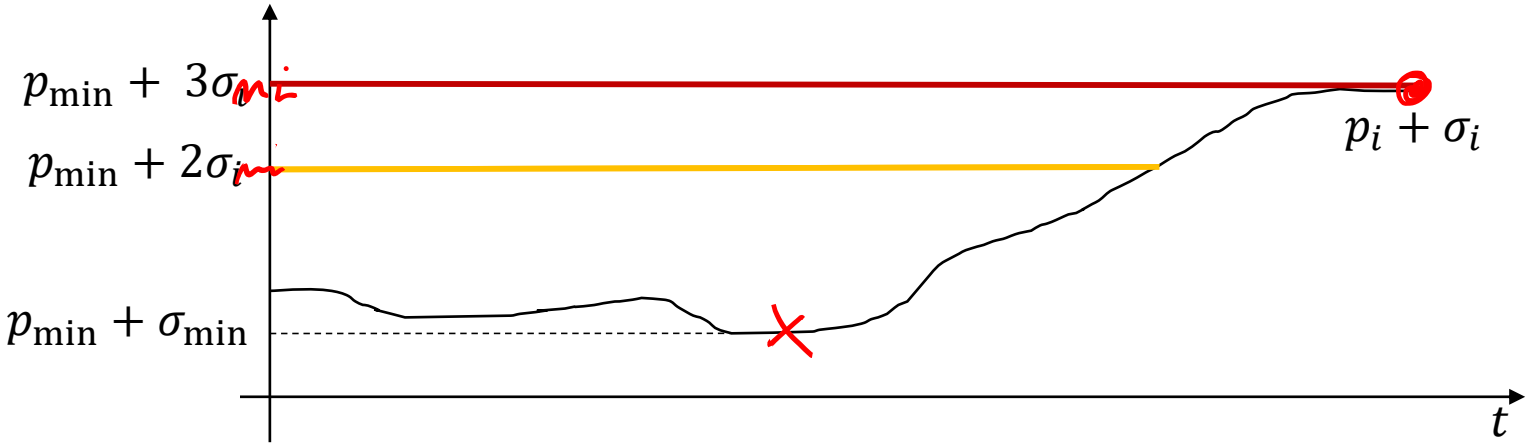
1. During monitoring, steadily compute p_i and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$
2. Let $p_{\min}$ be the minimum error before i and $\sigma_{\min} = \sqrt{\frac{p_{\min}(1-p_{\min})}{i}}$
3. Detect concept drift when $p_i + \sigma_i > p_{\min} + 3 * \sigma_{\min}$



Adaptation Heuristic in DDM

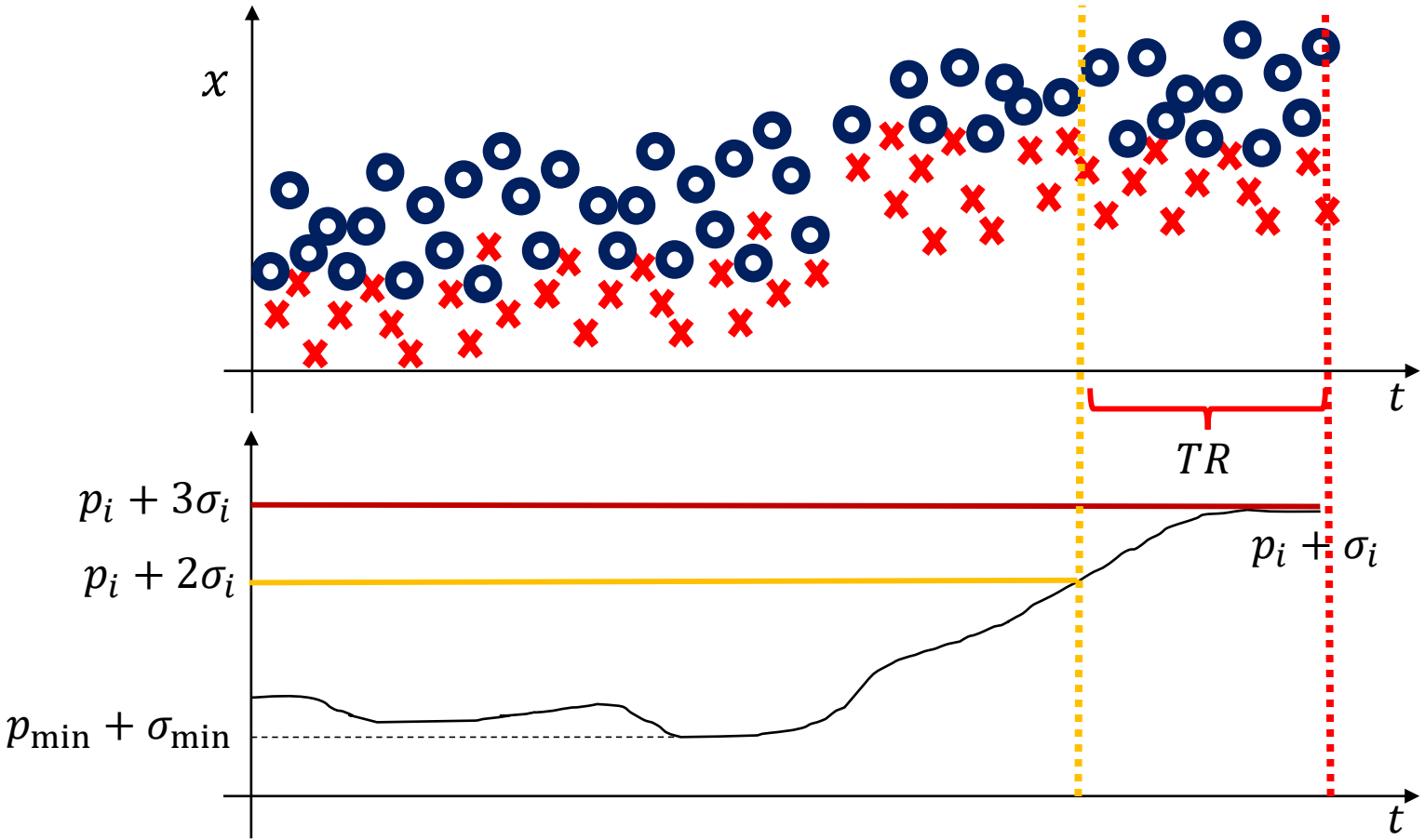
Monitoring the Classification Error: DDM

1. During monitoring, steadily compute p_i and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$
2. Let $p_{\min}$ be the minimum error before i and $\sigma_{\min} = \sqrt{\frac{p_{\min}(1-p_{\min})}{i}}$
3. Raise a "warning" when $p_i + \sigma_i > p_{\min} + 2 * \sigma_{\min}$
4. Detect concept drift when $p_i + \sigma_i > p_{\min} + 3 * \sigma_{\min}$



Post-detection Adaptation: DDM

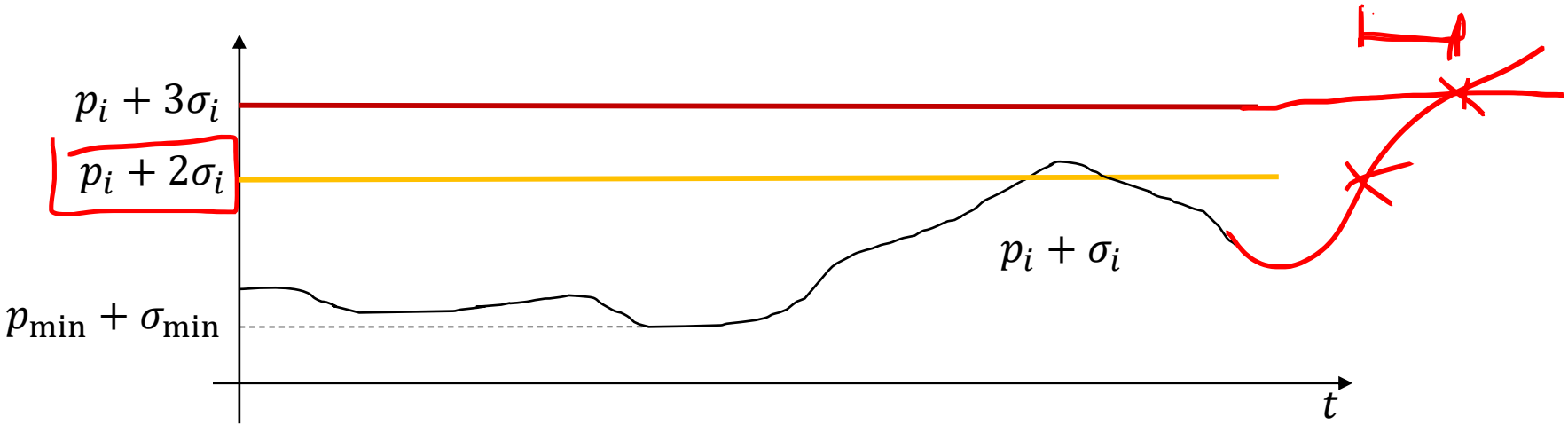
Use supervised samples in **between warning** and **drift** alert to **reconfigure** the classifier



Post-detection Adaptation: DDM

Use supervised samples in **between warning** and **drift** alert to **reconfigure** the classifier

Warning alerts non that are not followed by a drift alert are discarded and considered false-positive detections



Other Monitoring Solutions for the Classification Error

Adaptation

Monitoring the Classification Error: EDDM

Early Drift Detection Methods (EDDM) performs similar monitoring on the **average distance between misclassified samples**

- Average distance between classification error is expected to decrease under CD
- They aim at detecting gradual drifts

M. Baena-García, J. Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldá, R. Morales-Bueno. *"Early drift detection method"* In Fourth International Workshop on Knowledge Discovery from Data Streams (2006)

Monitoring the Classification Error: EWMA

Use the **Exponential Weighted Moving Average** (EWMA) as tests statistic

EWMA statistic

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

Now, as long as $(\boldsymbol{x_t}, y_t) \sim \phi_0(\boldsymbol{x}, y)$, $E[Z_t] = p_0$ the expected classification error in the stationary concept

After the change $\phi_0 \rightarrow \phi_1$ occurs, the average tends towards $p_1 > p_0$

The parameter λ regulates how fast the contribution of past observations decay and how quickly Z_t converges toward p_1 after the change

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

Monitoring the Classification Error: EWMA

A natural choice for a decision rule in our settings consists in:

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

Where σ_{Z_t} can corresponds to

$$\text{std}[Z_t] = \sigma_0 \sqrt{\frac{\lambda}{2-\lambda}(1-(1-\lambda)^{2t})}$$

being σ_0 the standard deviation of the error (this results holds for EWMA monitoring scheme in general)

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

Issues:

EWMA

$$Z_t > p_0 + L\,\sigma_{Z_t}$$

1. How to proceed when p_0 and σ_0 are unknown?
2. How to set L to guarantee a certain ARL_0?

EWMA for Bernoulli Random Variables

Let's make the simplyfing assumption that the error before the change is constant p_0 with the standard deviation $\sigma_0 = \sqrt{p_0(1-p_0)}$

Replace p_0 with its BLUE estimator $\hat{p}_{0,t}$ at time t

$$\hat{p}_{0,t} = \frac{t}{t+1}\hat{p}_{0,t-1} + \frac{1}{t+1}e_t$$

Compute the varaince of $\hat{p}_{0,t}$

$$\hat{\sigma}^2_{0,t} = \hat{p}_{0,t}(1-\hat{p}_{0,t})$$

And plug this in the EWMA statistic (which indeed scales $\hat{\sigma}_{0,t}$)

$$\hat{\sigma}_{Z_t} = \hat{\sigma}_{0,t}\sqrt{\frac{\lambda}{2-\lambda}(1-(1-\lambda)^{2t})}$$

Stopping Rule for EWMA for Bernoulli

When replacing the true values p_0 and σ_0 by their estimates

$$Z_t > \hat{p}_{0,t} + L\hat{\sigma}_{Z_t}$$

The contol limit has to become time-dependent in order to preserve a target ARL_0, thus

$$Z_t > \hat{p}_{0,t} + L_t\hat{\sigma}_{Z_t}$$

Defining the sequence $\{L_t\}_t$ is very complicated as they depend on $\hat{p}_{0,t}$.

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

It is also possible «to revert», provided ARL_0 and p_0 identify L

Sparks, R.S., 2000. CUSUM charts for signalling varying location shifts. J. Qual. Technol. 32.

Stopping Rule for EWMA for Bernoulli

So, the idea is to estimate a function

$$f : (P_0, A_0) \rightarrow L$$

that returns L yielding $ARL_0 = \alpha_0$ over Bernoulli streams having $p_0 = P_0$.

This can be done by **polynomial fit over the results of MonteCarlo simulations** and yields **a function to be invoked at each iteration of the algorithm since $\hat{p}_{0,t}$ does change**

Table 1

Polynomial approximations for L for various choices of ARL_0 and $\lambda = 0.2$.

ARL_0	Regression estimate of L
100	$2.76 - 6.23\hat{p}_0 + 18.12\hat{p}_0^3 - 312.45\hat{p}_0^5 + 1002.18\hat{p}_0^7$
400	$3.97 - 6.56\hat{p}_0 + 48.73\hat{p}_0^3 - 330.13\hat{p}_0^5 + 848.18\hat{p}_0^7$
1000	$1.17 + 7.56\hat{p}_0 - 21.24\hat{p}_0^3 + 112.12\hat{p}_0^5 - 987.23\hat{p}_0^7$

Ross, Adams, Tasoulis, Hand *"Exponentially Weighted Moving Average Charts for Detecting Concept Drift"* Pattern Recogn. Lett. 33, 2012

Monitoring the Classification Error: EWMA

Like DDM, **classifier reconfiguration** is performed by monitoring Z_t also at a *warning level*

$$Z_t > p_{0,t} + 0.5\, L_t \sigma_t$$

WARNING

Once CD is detected, the first sample raising a warning is used to isolate samples from the new distribution and retrain the classifier

Ross, Adams, Tasoulis, Hand *"Exponentially Weighted Moving Average Charts for Detecting Concept Drift"* Pattern Recogn. Lett. 33, 2012

EWMA Monitoring for concept drift

Table 2
Final ECDD algorithm.

Choose a desired value for λ and the ARL_0
Initialize the classifier
$Z_0 = 0$ and $\hat{p}_{0,0} = 0$
For each object f_t
  classify object and update classifier
  Define $X_t = 0$ if the object was correctly classified or X_t if the classification was incorrect,
  $\hat{p}_{0,t} = \frac{t}{t+1}\hat{p}_{0,t-1} + \frac{1}{t+1}X_t$
  $\hat{\sigma}_{X_t} = \sqrt{\hat{p}_{0,t}(1-\hat{p}_{0,t})}$
  $\hat{\sigma}_{Z_t} = \sqrt{\frac{\lambda}{2-\lambda}(1-(1-\lambda)^{2t})}\hat{\sigma}_{X_t}$
  Compute L_t based on current value of $\hat{p}_{0,t}$ using Table 1
  $Z_t = (1-\lambda)Z_{t-1} + \lambda X_t$
  Flag for concept drift if $Z_t > \hat{p}_{0,t} + L_t\hat{\sigma}_{Z_t}$

Ross, Adams, Tasoulis, Hand *"Exponentially Weighted Moving Average Charts for Detecting Concept Drift"* Pattern Recogn. Lett. 33, 2012

Second Matlab Assignment

Boracchi, Trovò

Get the second matlab snippet

And develop a monitoring scheme that

- Classifies each incoming samples
- Once feedback is provided, monitor the classification error using EWMA
- if EWMA detects a change
  - Updates the classifier using all the training samples between the latest sample reaching the warning_level and the sample where the detection occurred

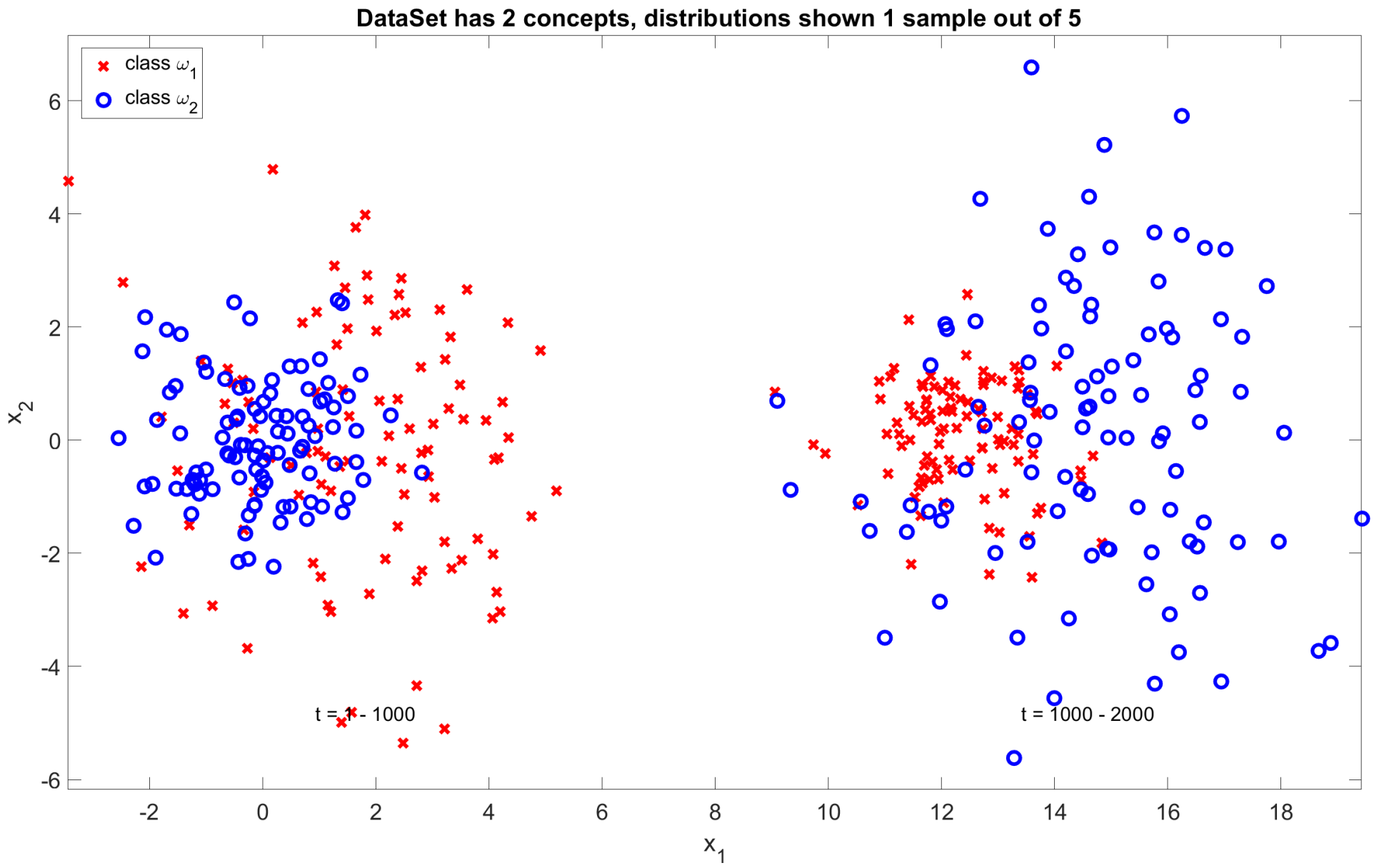
  Else
  - Updates the classifier using all the supervised information from the same concept

Compare the performance with a classifier that is never updated

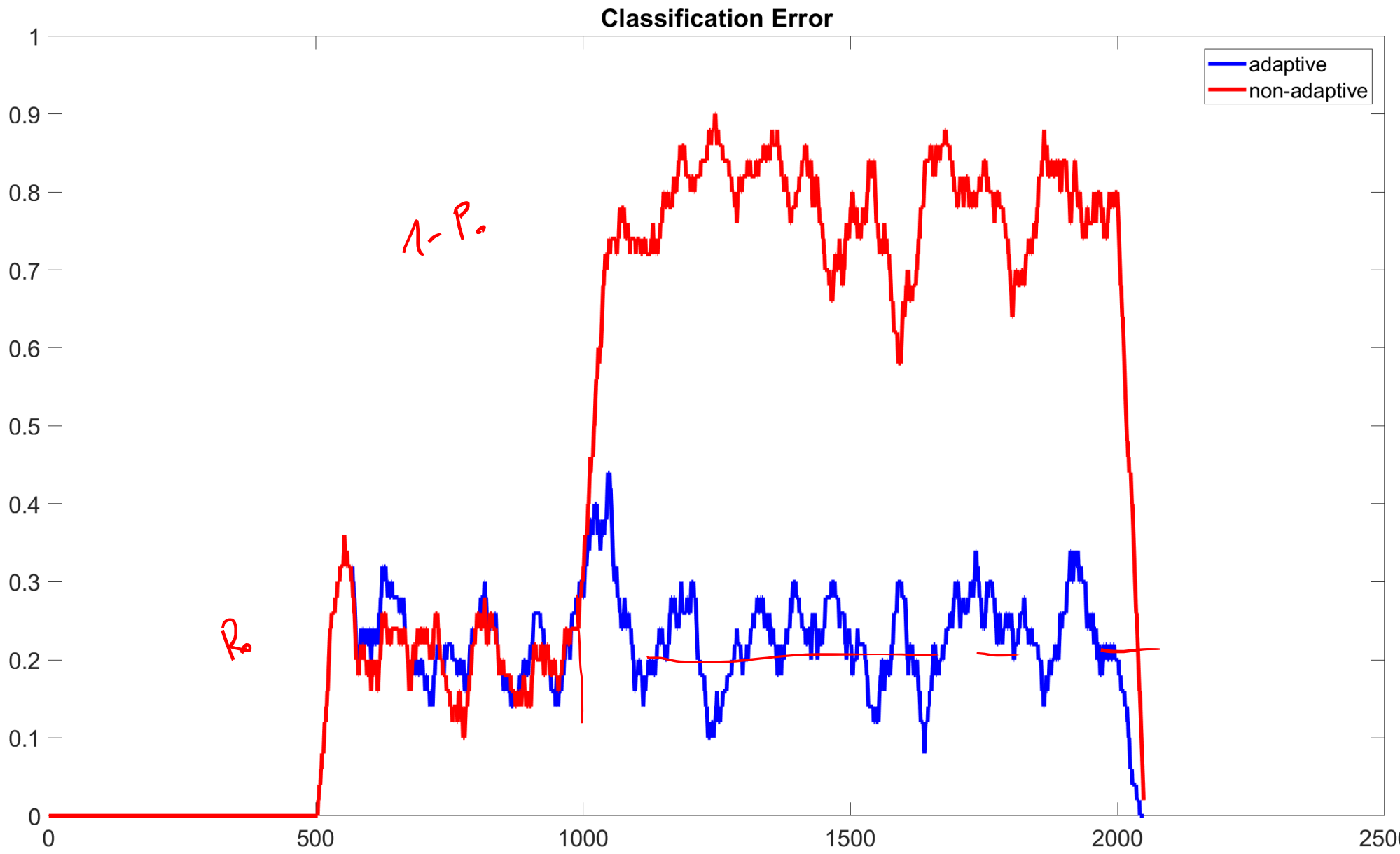
Display the EWMA statistic and its threshold over the whole stream

Boracchi, Trovò

EWMA Example

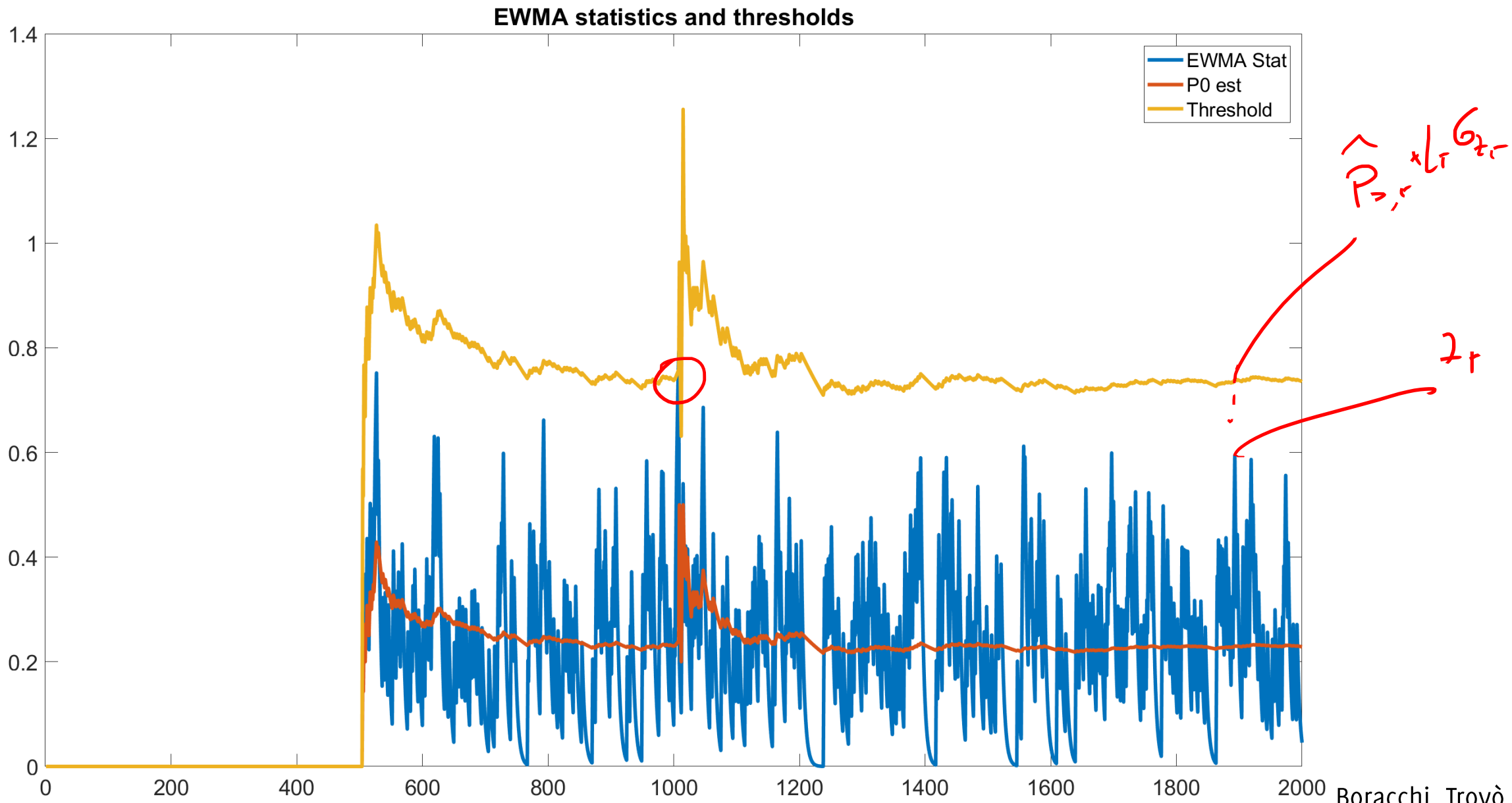


EWMA Example

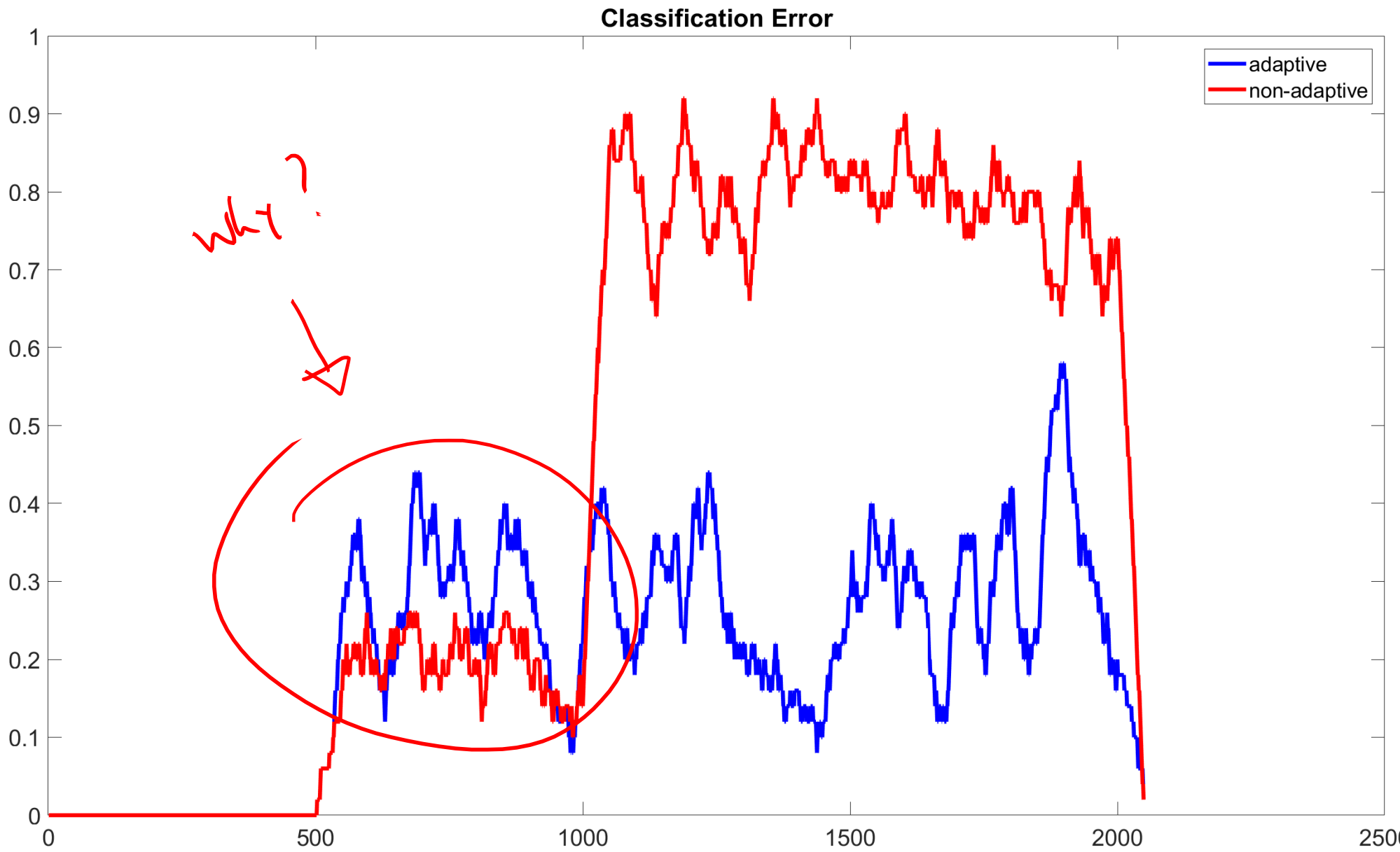


Boracchi, Trovò

EWMA Example

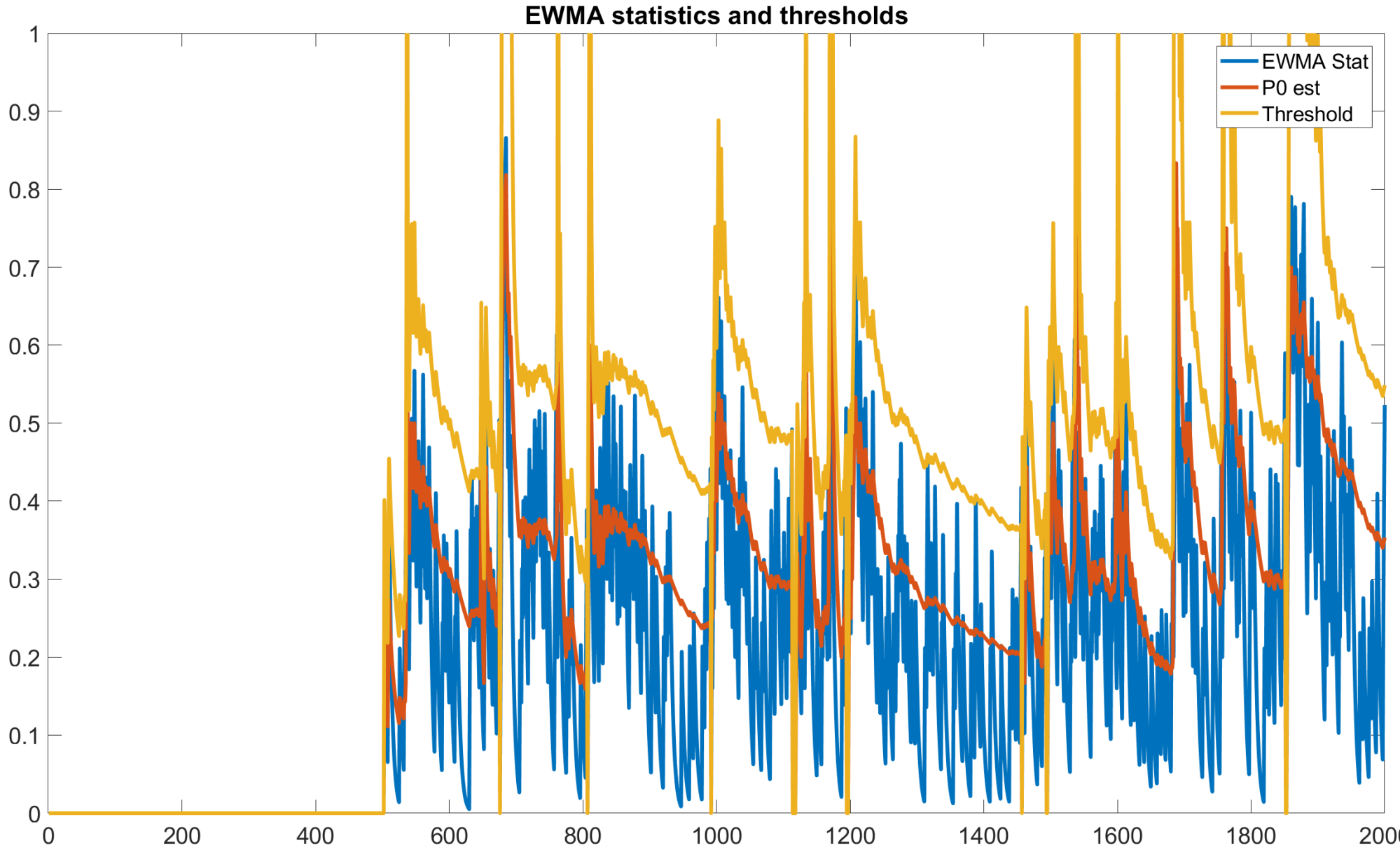


EWMA using wrong thresholds



Boracchi, Trovò

EWMA using wrong thresholds



Boracchi, Trovò

Matlab for Python Users

http://mathesaurus.sourceforge.net/matlab-python-xref.pdf

https://trovo.faculty.polimi.it/02source/olam_2020/matlab-for-dummies.pdf

Monitoring Classification Error By Comparing Windows

Giacomo Boracchi, Francesco Trovò

May 6th, 2020

Politecnico di Milano, DEIB

giacomo.boracchi@polimi.it

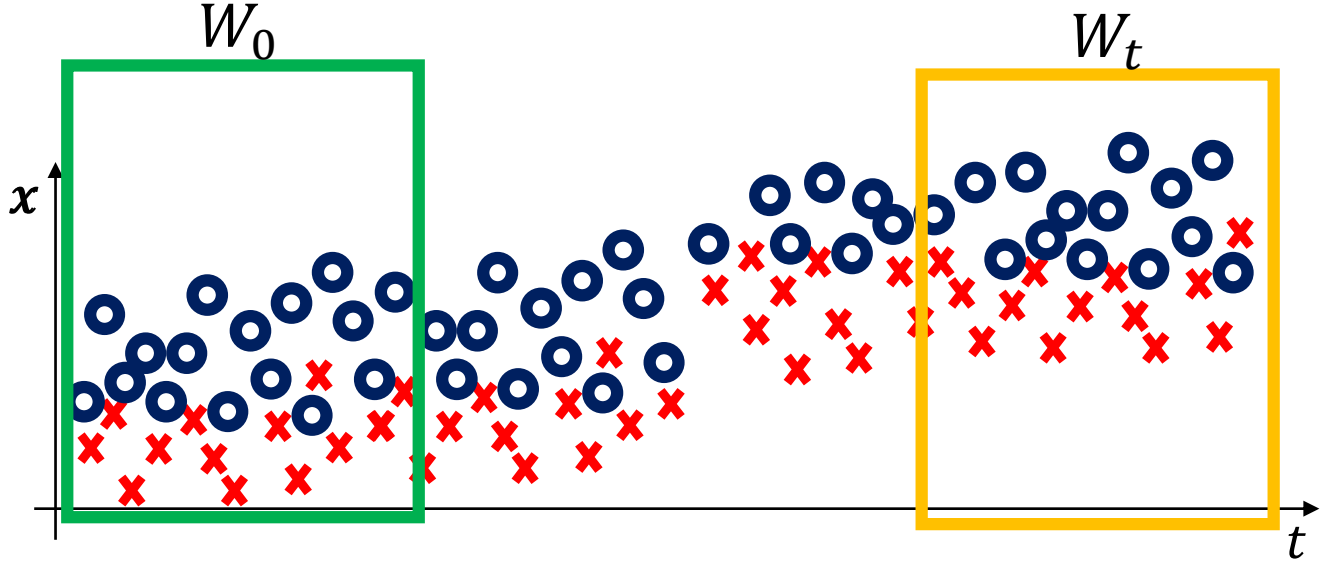
The Motivating Idea

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
- $\mathcal{T}$ is a suitable statistic over the classification error

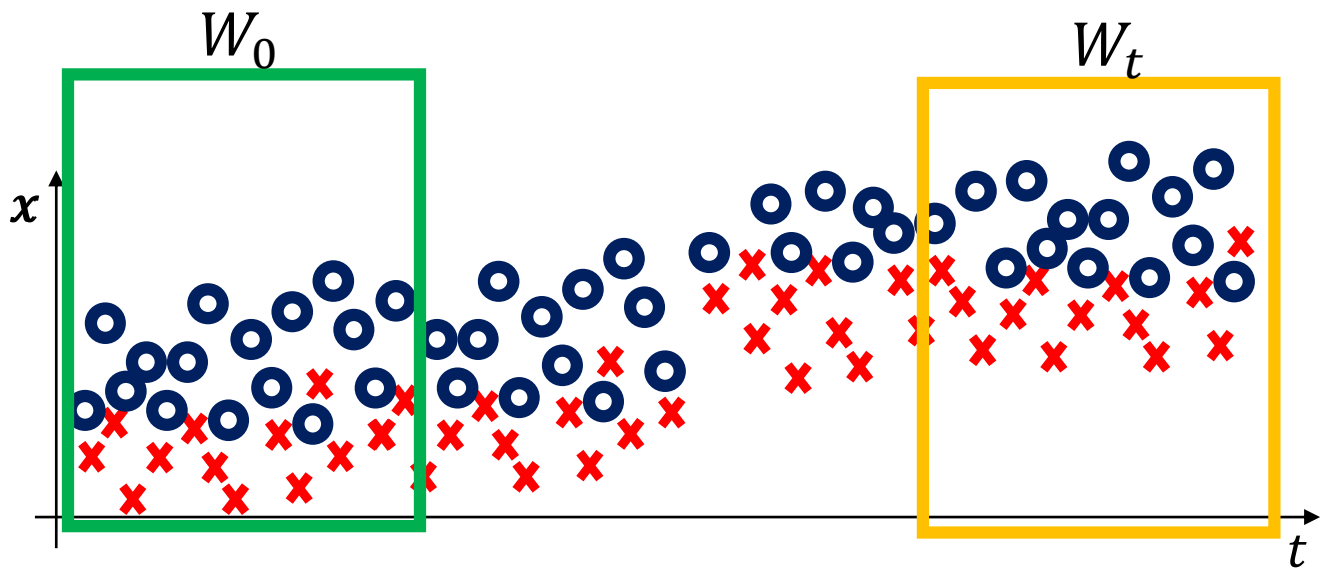


Window Comparison: Major Approaches

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

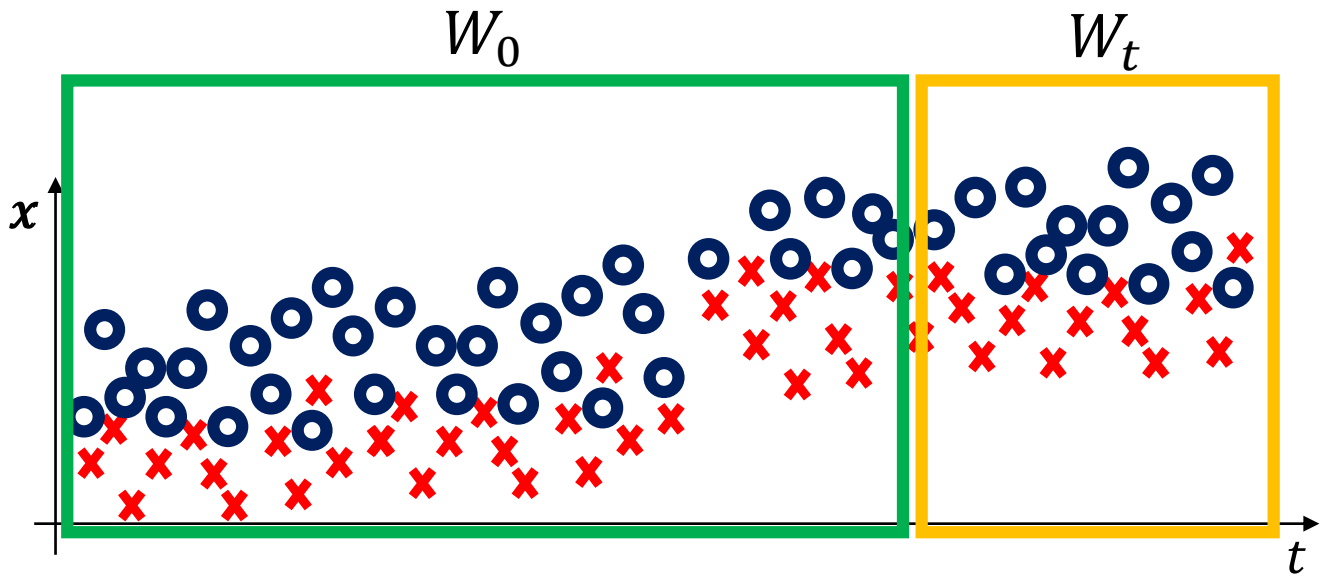
One sided t-test with $H_0 = \{p_t \leq p_0\}$ can detect concept drift



Window Comparison: Major Approaches

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

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

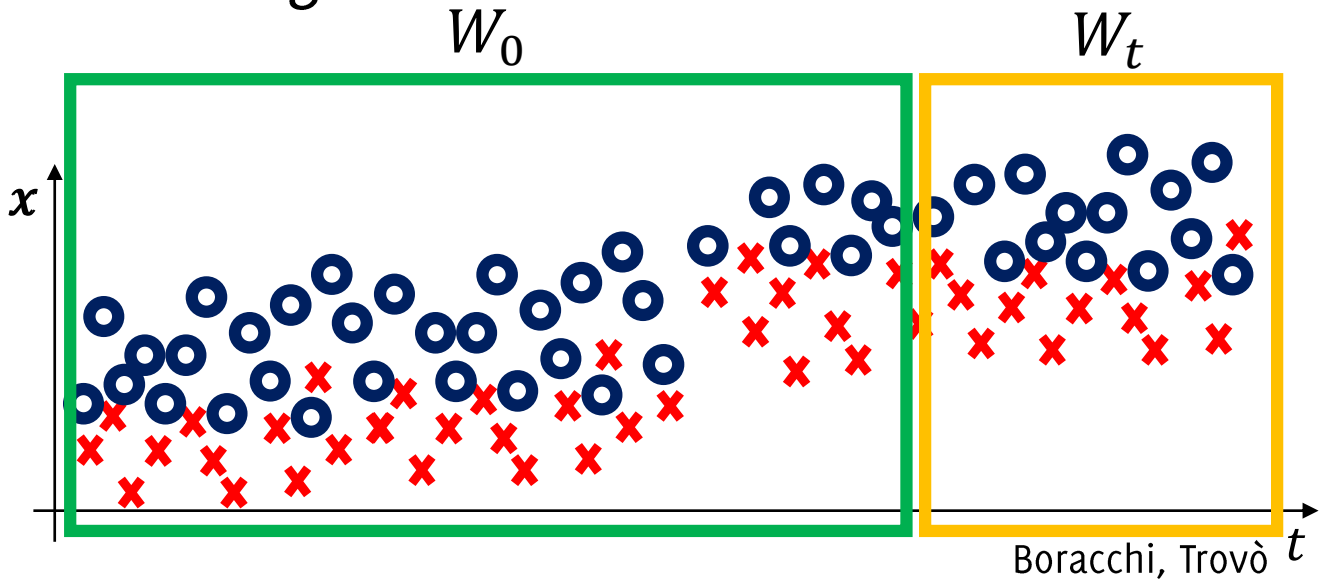


Nishida, K. and Yamauchi, K. *"Detecting concept drift using statistical testing"* In DS, pp. 264–269, 2007

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.

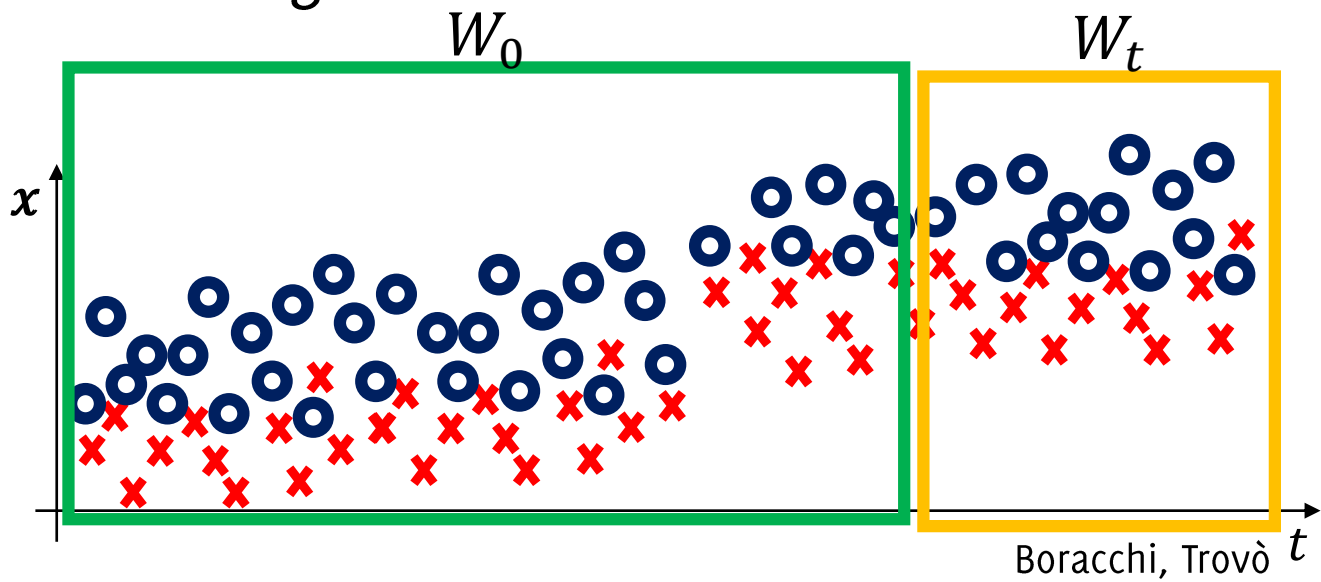


Window Comparison: Major Issues

Issues: Statistical Hypothesis test are "one shot method" 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.

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

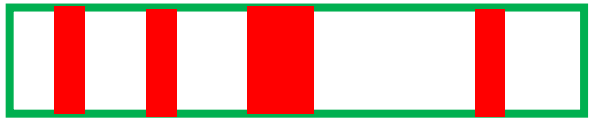


Window Comparison: Testing Exchangeability

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 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)

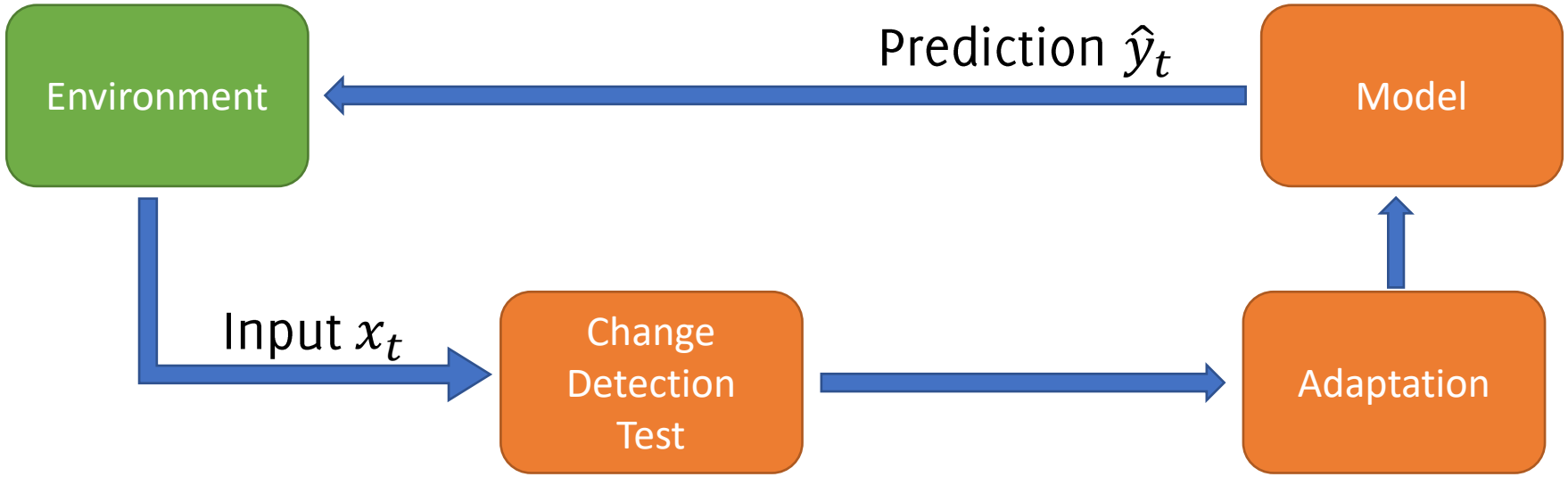
Monitoring the Input Distribution

… 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**

Cons:

- CD that does not affect $\phi(\boldsymbol{x})$ are not perceivable
- 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

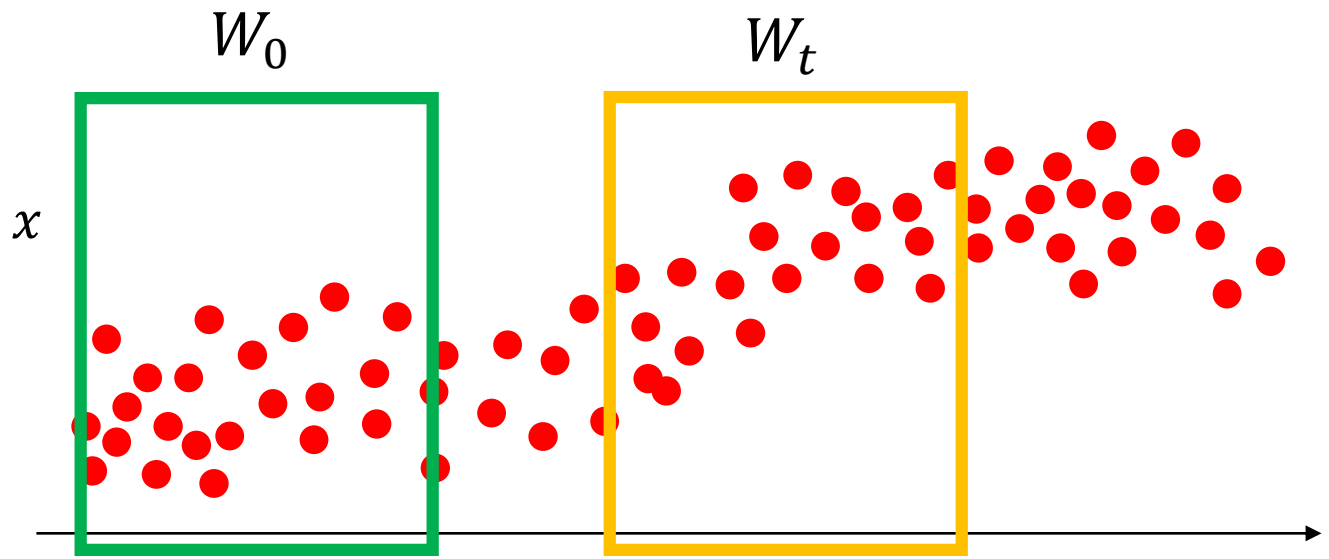
The Motivating Idea

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
- $\mathcal{T}$ is a suitable statistic over the classification error

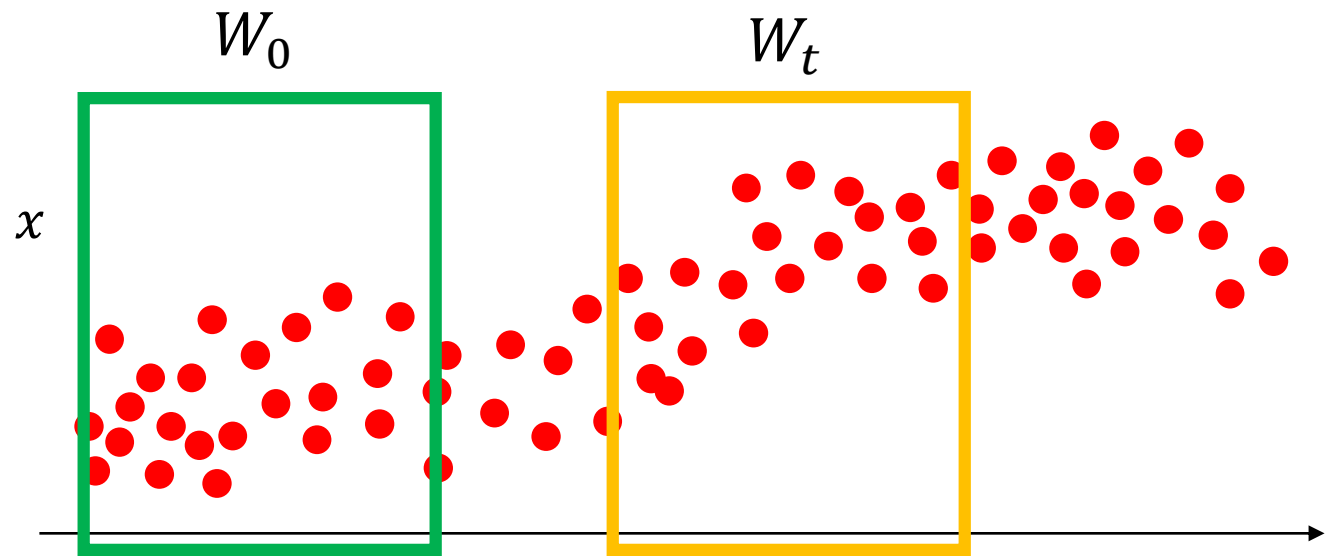


Window Comparison: Major Approaches

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

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



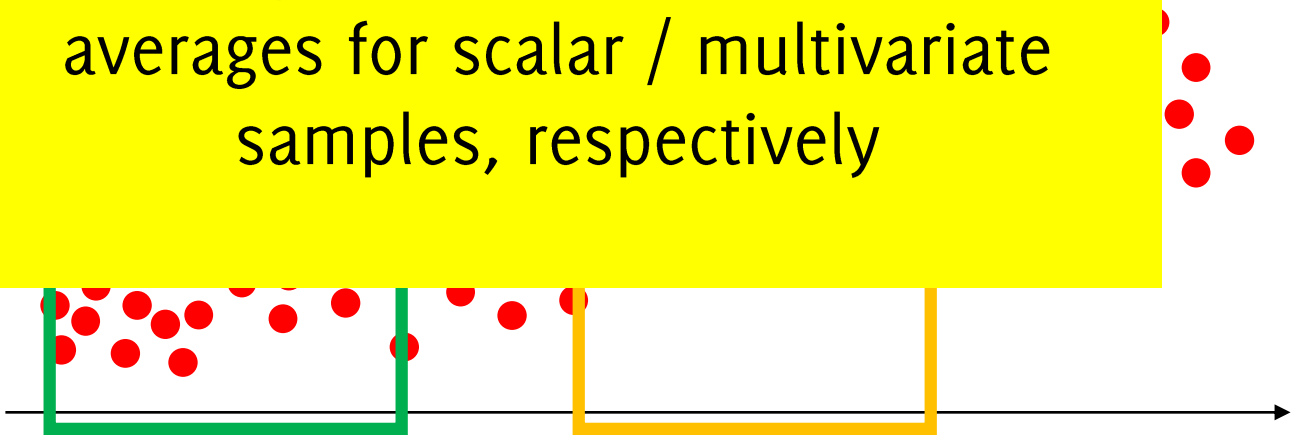
Window Comparison: Major Approaches

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 increasing in 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" 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 increasing in 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

$$W = 101010110111111$$

$$W_0 = 1, \quad \mu_0 \qquad W_1 = 01010110111111, \quad \mu_1$$

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 increasing in 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

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

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

Hypothesis testing

ADWIN: Compare the averages of scalar inputs over two adjacent windows

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

Hypothesis testing

ADWIN: Compare the averages of scalar inputs over two adjacent windows

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

Hypothesis testing

ADWIN: Compare the averages of scalar inputs over two adjacent windows

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)

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

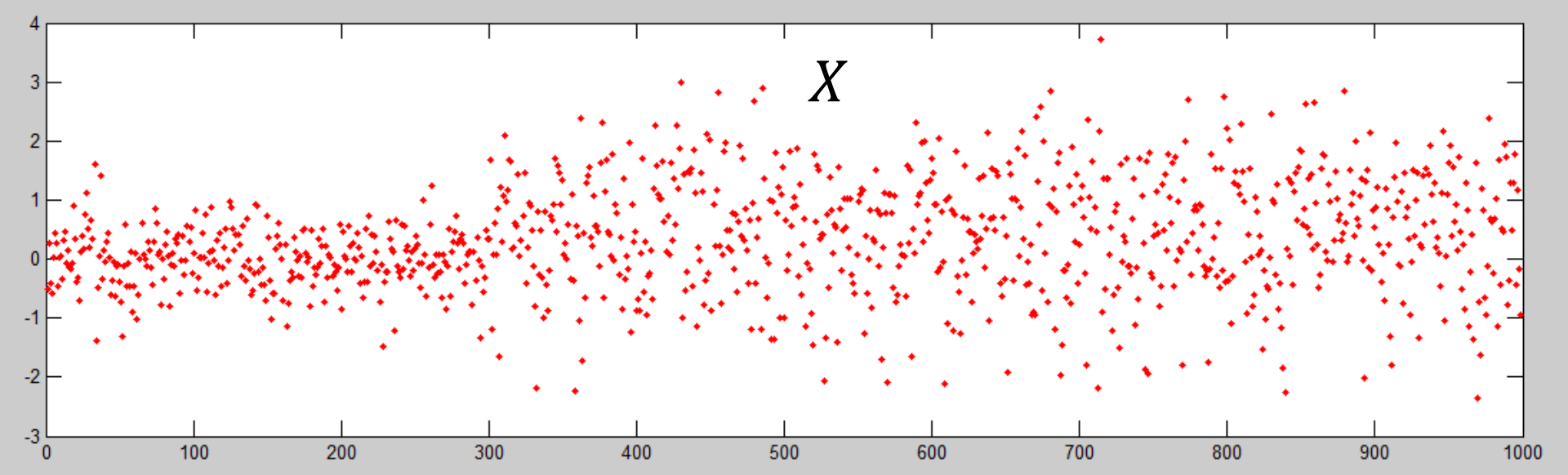
University of Minnesota, Minneapolis, MN 55455

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.

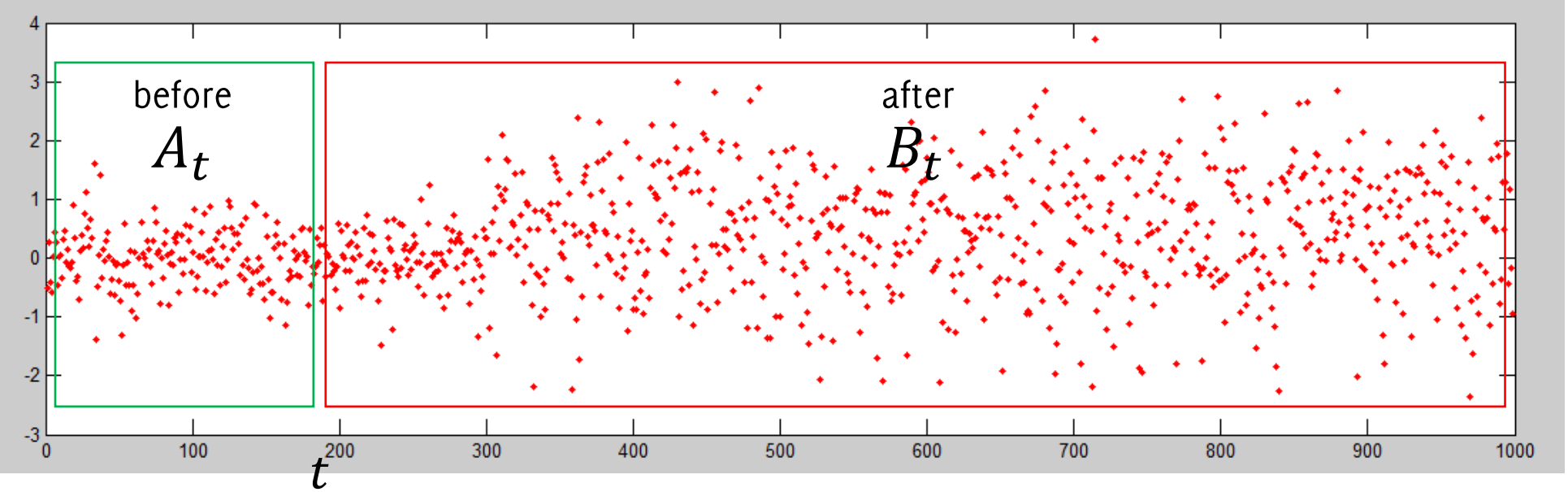
The Change Point Method (CPM)



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

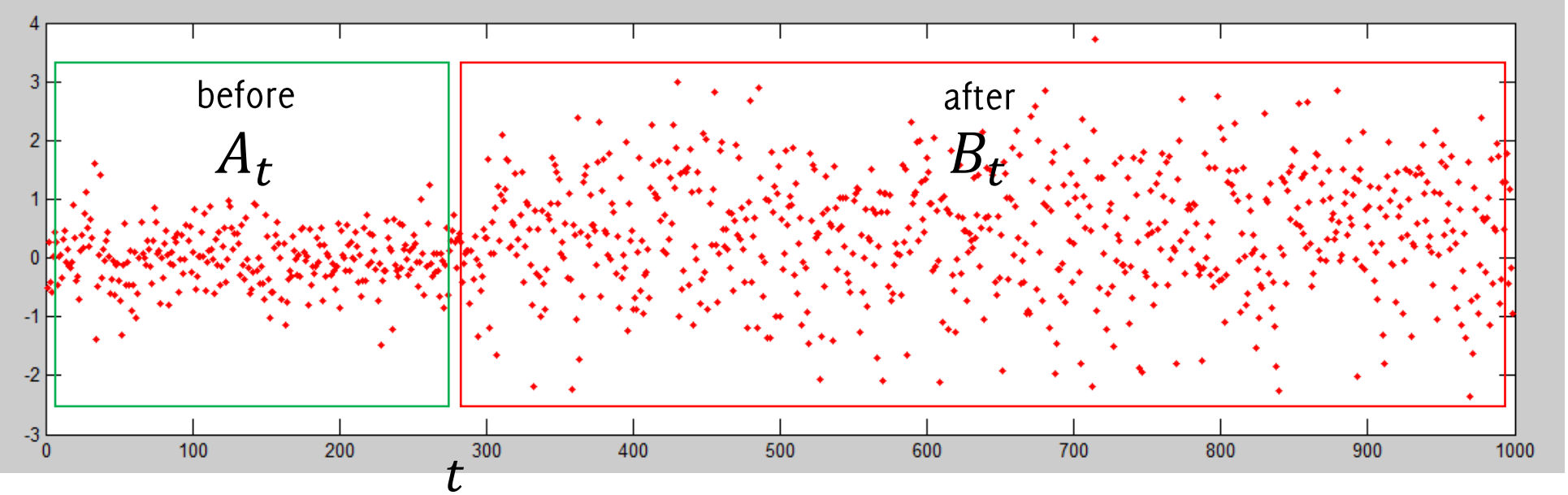
Assume we are given a statistic S_t to compare two datasets $A_t, B_t \subset X$ coming before and after t

The Change Point Method (CPM)



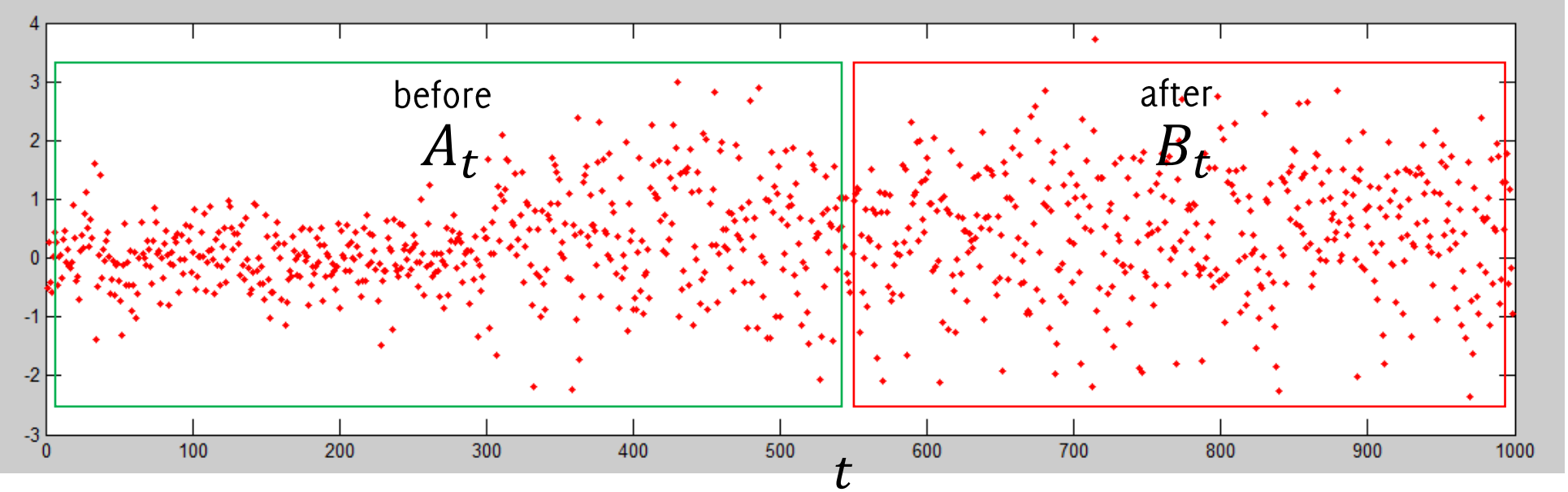
- Test a single point t to be a change point
- Split the dataset in two sets $A_t, B_t \subset X$, namely samples «before» and «after» the putative change at t
- **Compute a test statistic** S_t to determine whether the two sets are from the same distribution (e.g. same mean)
- **Repeat the procedure**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

The Change Point Method (CPM)



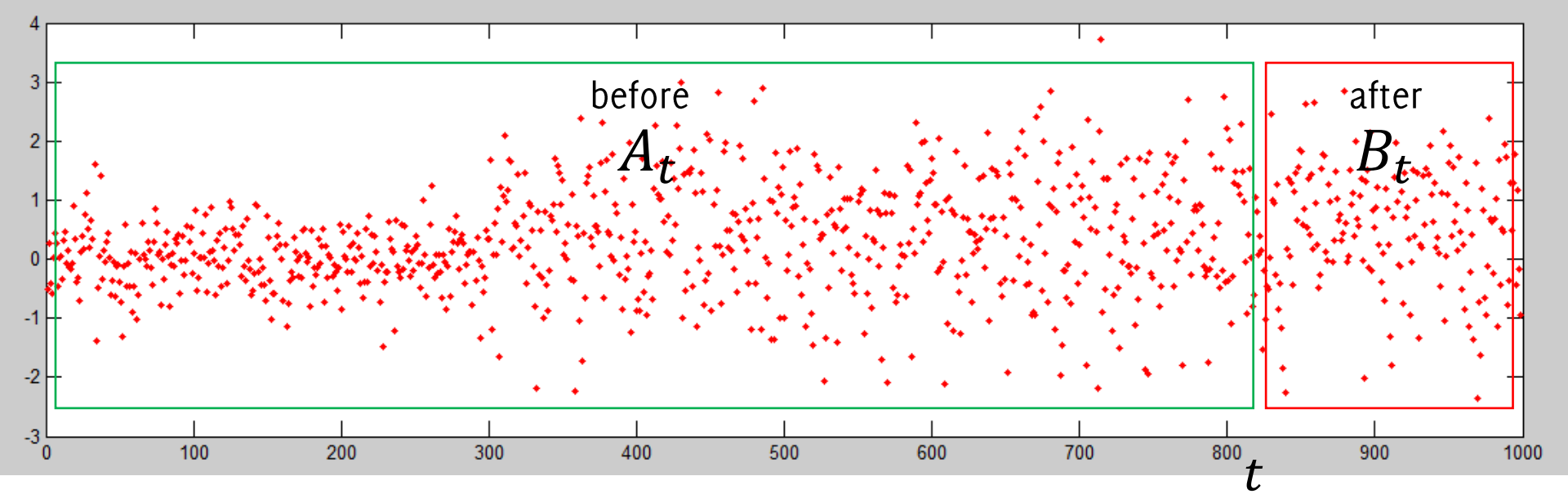
- Test a single point t to be a change point
- Split the dataset in two sets $A_t, B_t \subset X$, namely samples «before» and «after» the putative change at t
- **Compute a test statistic** S_t to determine whether the two sets are from the same distribution (e.g. same mean)
- **Repeat the procedure**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

The Change Point Method (CPM)



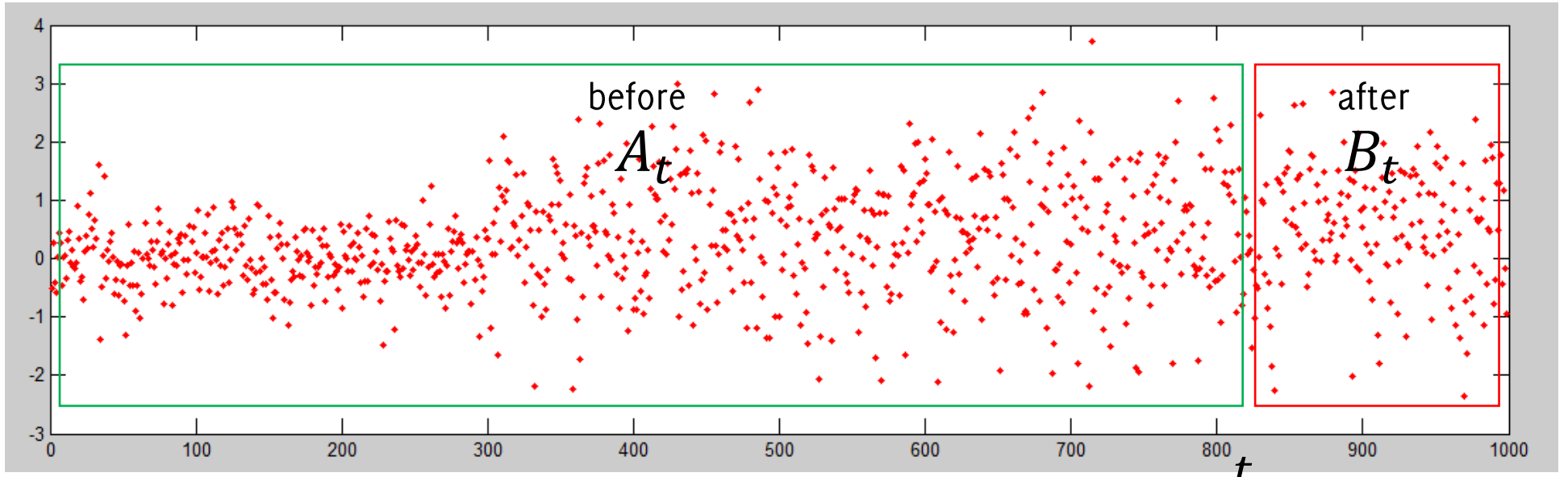
- Test a single point t to be a change point
- Split the dataset in two sets $A_t, B_t \subset X$, namely samples «before» and «after» the putative change at t
- **Compute a test statistic** S_t to determine whether the two sets are from the same distribution (e.g. same mean)
- **Repeat the procedure**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

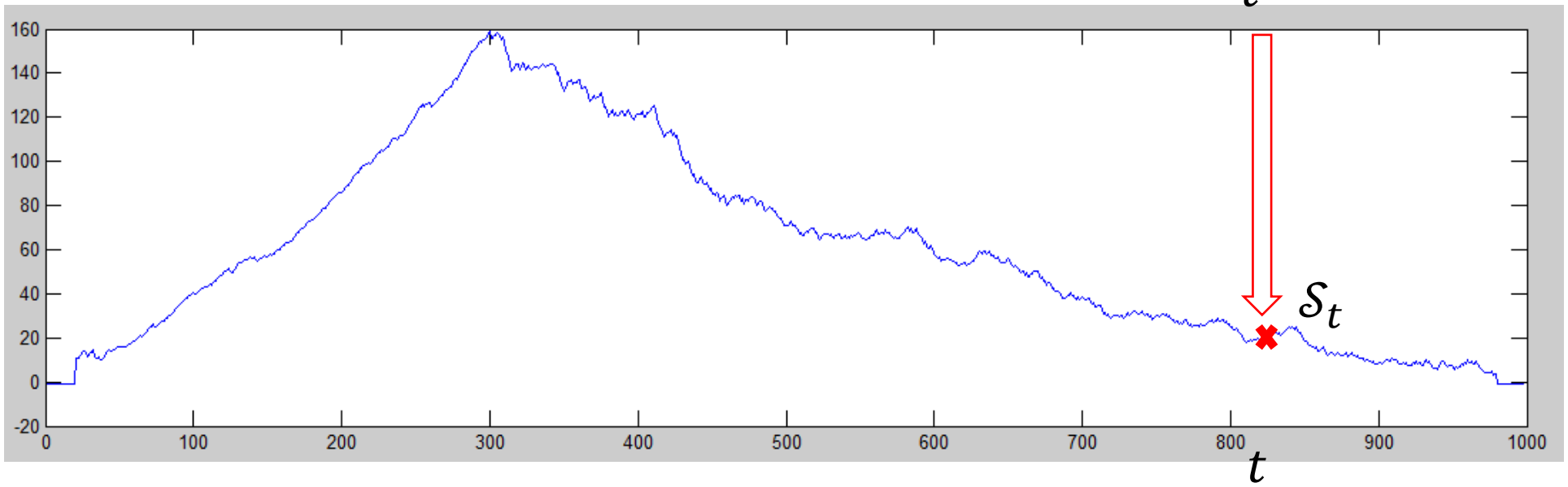
The Change Point Method (CPM)



- 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** S_t to determine whether the two sets are from the same distribution (e.g. same mean)
- **Repeat the procedure** and store the value of the statistic

The Change Point Method (CPM)





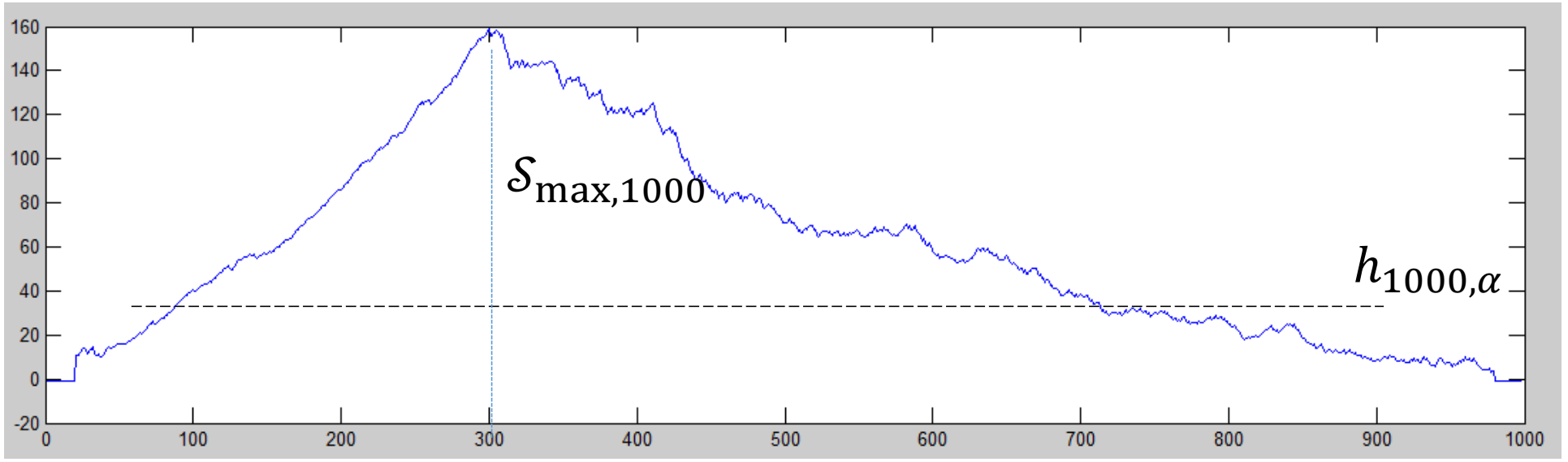
Boracchi, Trovò

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 $\forall\, T \in [1, N]$

- 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

- 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 $\mathcal{S}_{max,N}$**

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

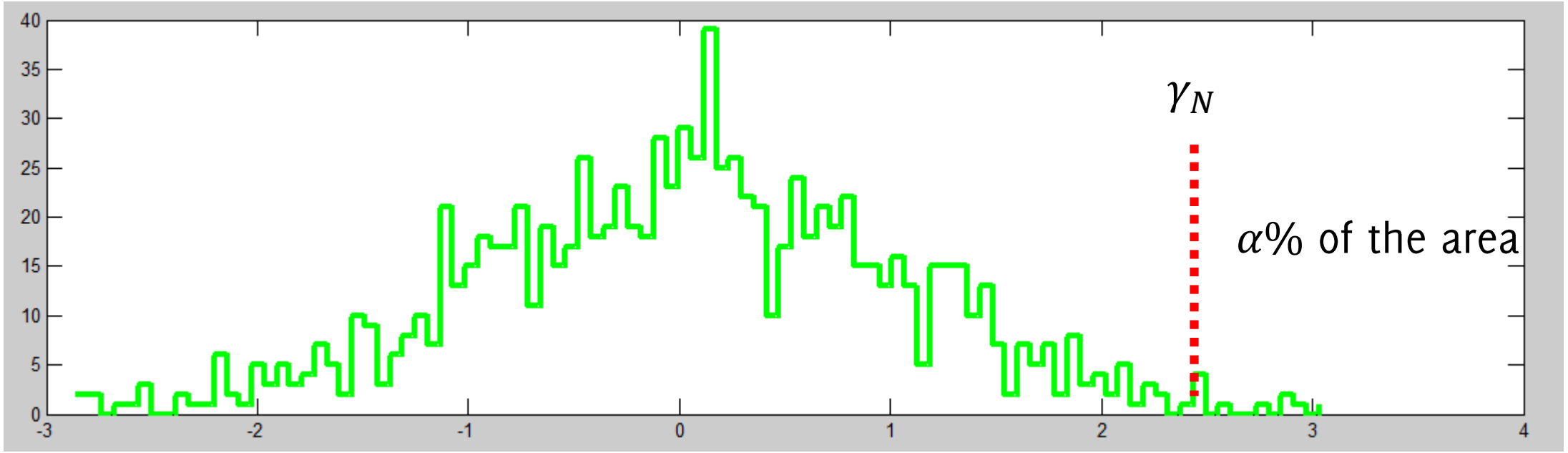
Other options:

- Bonferroni approximations are too loose (many comparison, one per sample of the stream)
- Asymptotic bounds are only available for certain statistics $\mathcal{S}$, 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 multiple sequences (?!)
- Compute the statistic $\mathcal{S}_{\max,N}$ and store these values
- Set the threshold as the quantile of this empirical distribution



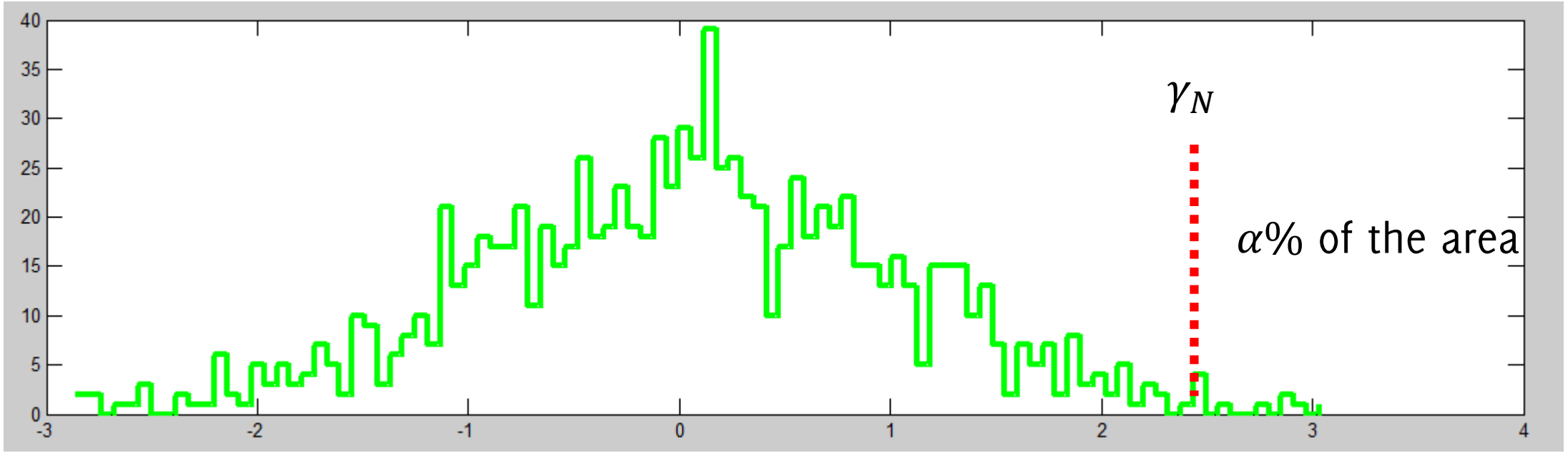
Threshold Computation

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 multiple sequences (?!)
- Compute the statistic $\mathcal{S}_{\max,N}$ and store these values
- Set the threshold as the quantile of this empirical distribution



Threshold Computation

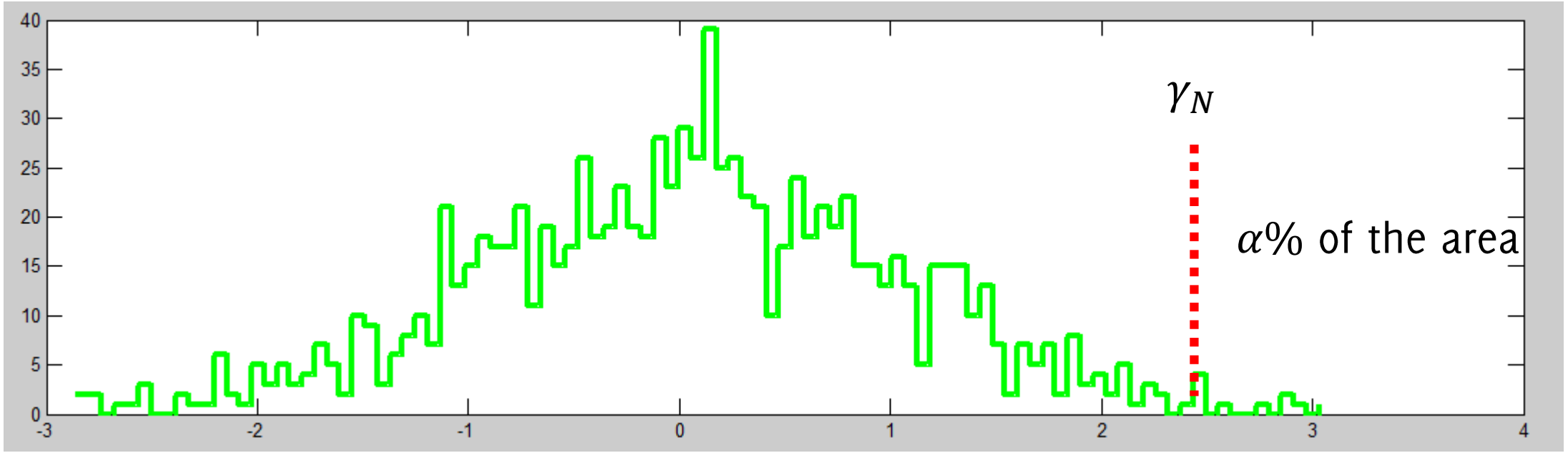
The computed thresholds depends on many factors:

- 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 applied for each ϕ_0 and N

Therefore we resort to bootstrap.

- Draw multiple sequences (?!)
- Compute the statistic $\mathcal{S}_{\max,N}$ an
- 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,

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 that both sets are identically distributed is $(N+1)/2$

M measures the deviation of each rank 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 each rank from this expectation

When $A \sim \phi_0$ 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**.

Assuming 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 to store all the data in memory

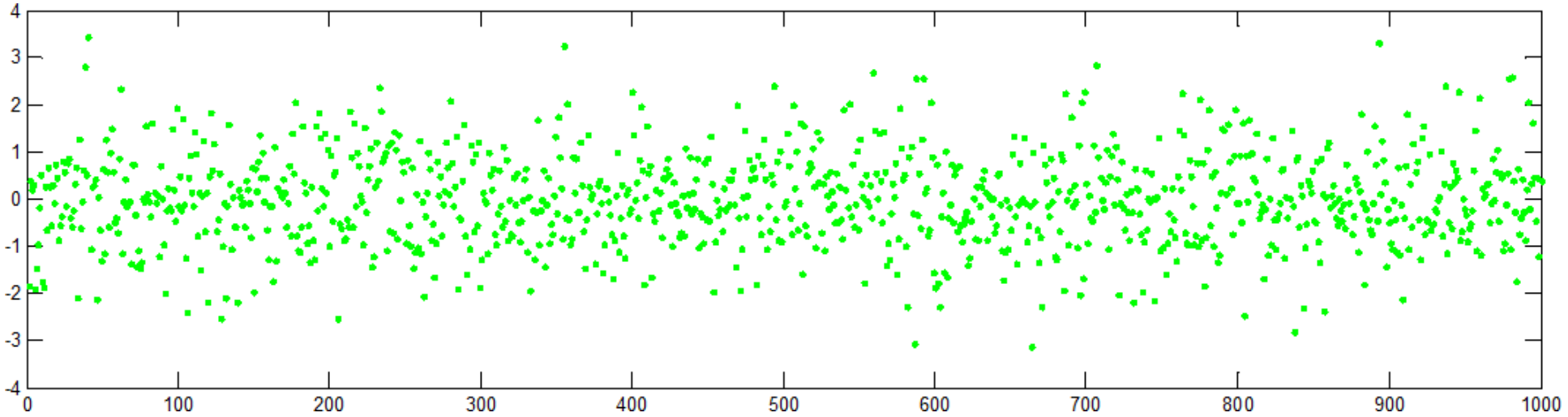
Also time requirement grows at each new observation

This is usually infeasible when working with data streams.

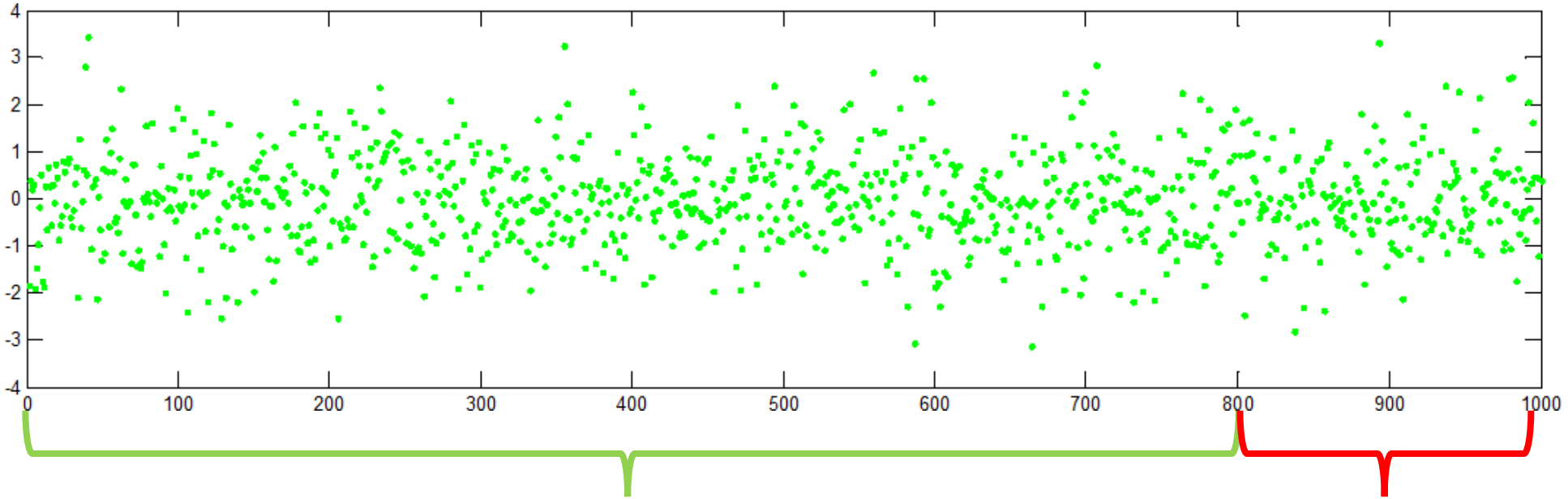
Solution: discretization of the older part of the stream

- 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



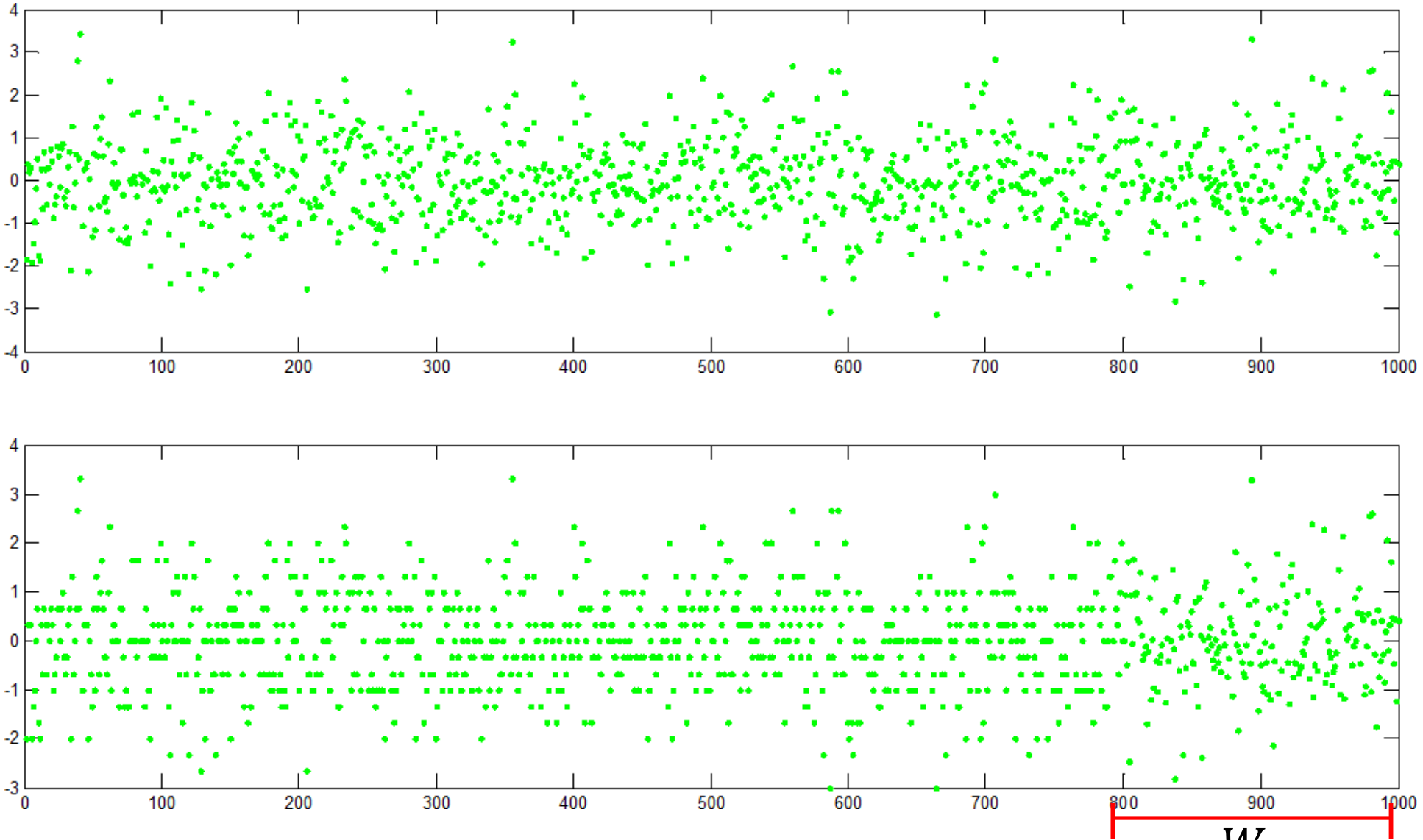
Data Quantization



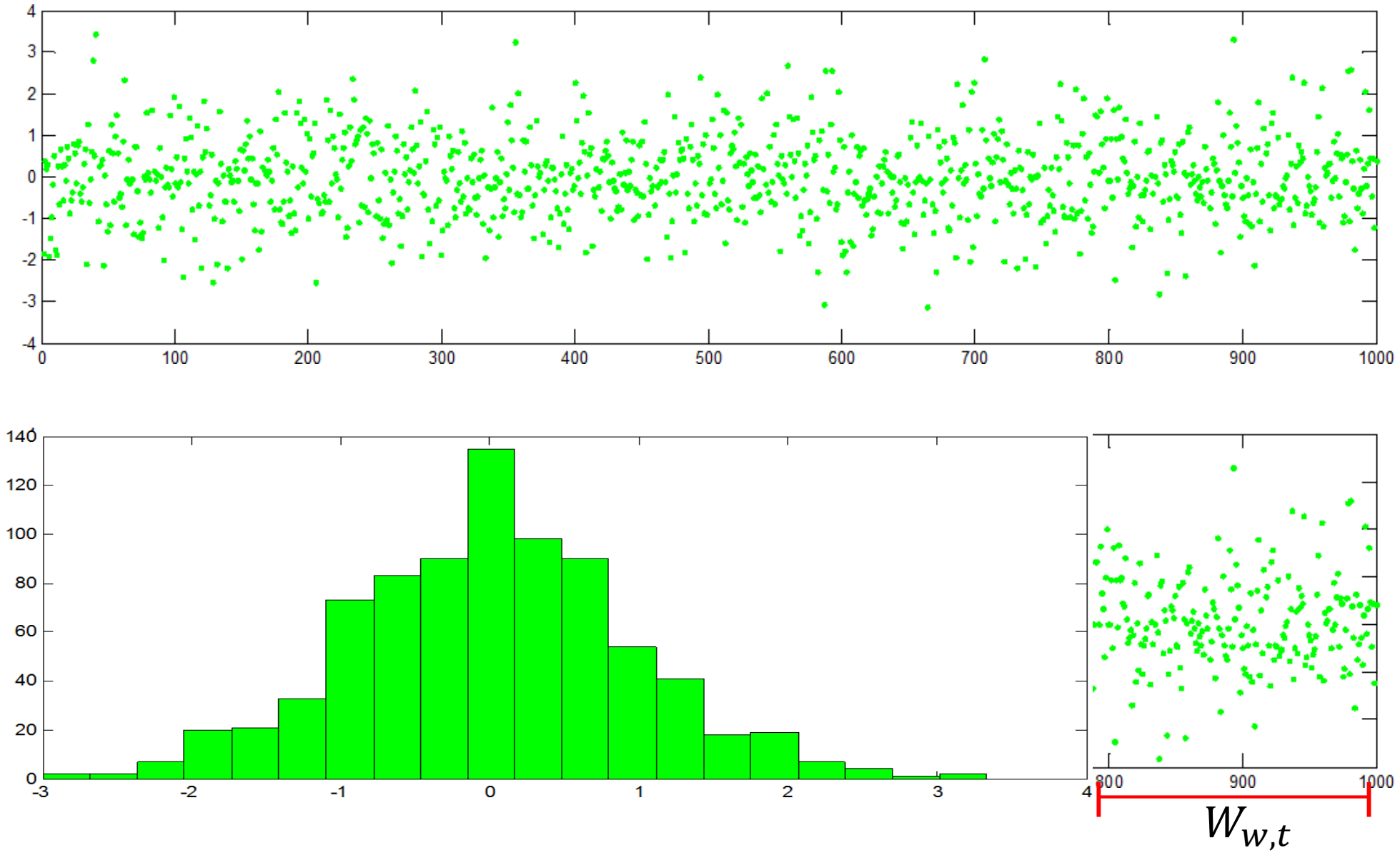
Quantize past data in m values (range is defined from a training sequence)

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

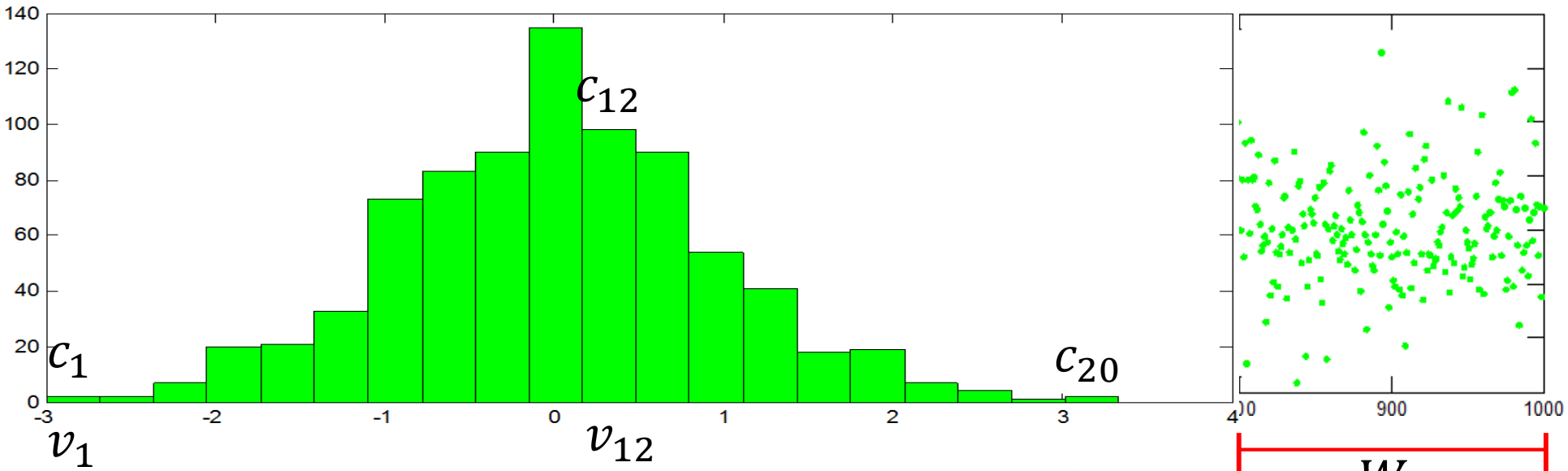
Data Quantization



Data Quantization



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 $W_{w,t}$

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

Change Detection Approaches

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

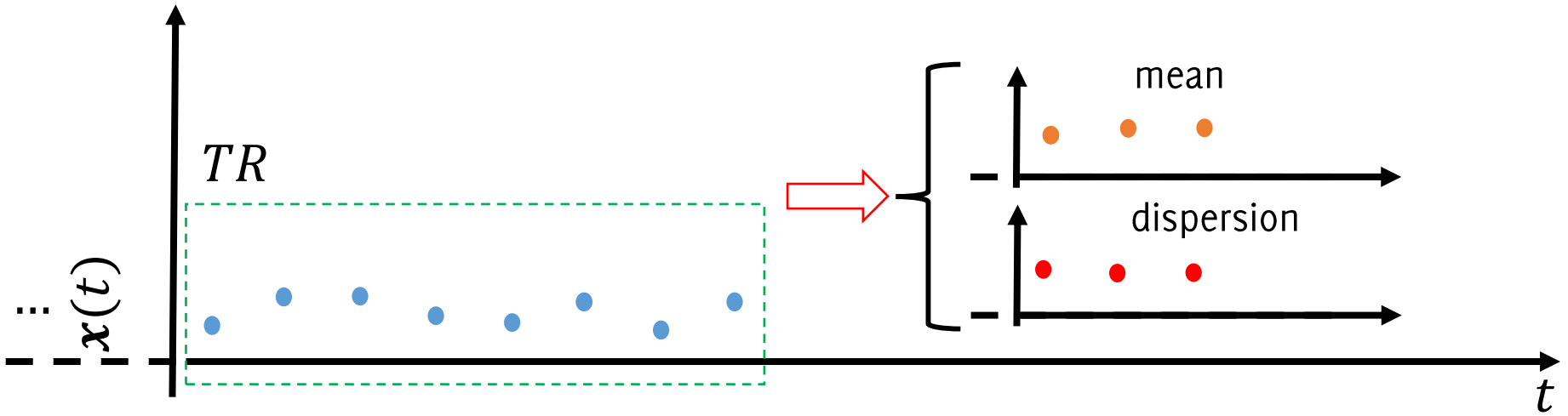
CMPs are nice, but statistics based on sorting holds for scalar sterams

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

Boracchi, Trovò

Three ingredients

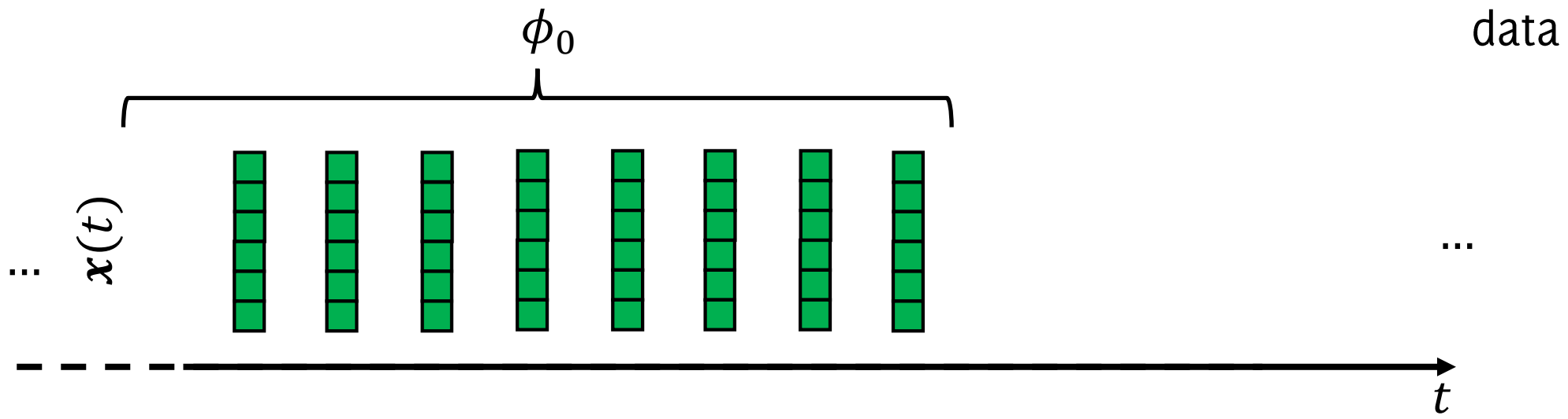
Most change-detection algorithm consists in

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

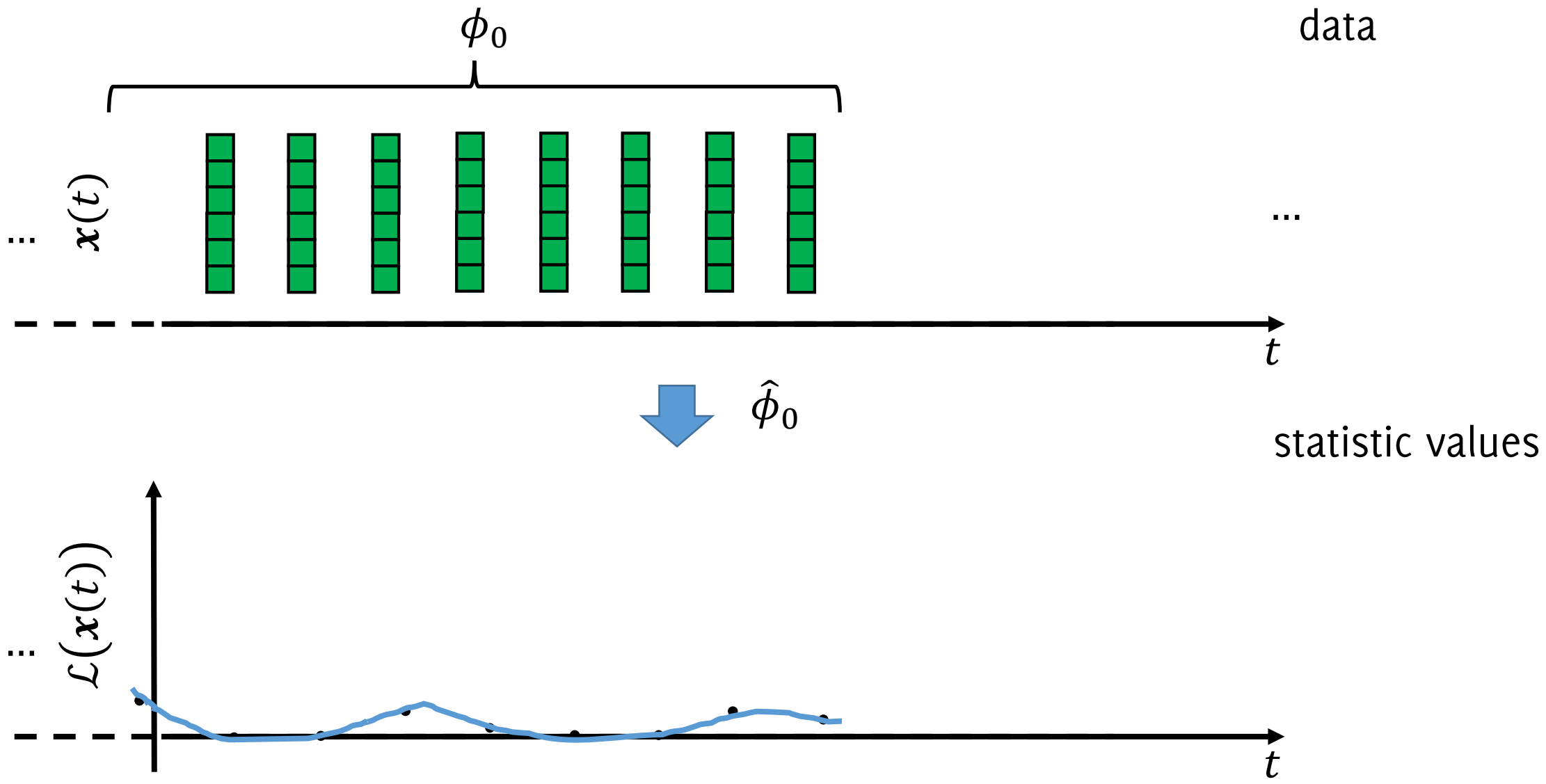
ii. A statistic $\mathcal{T}$ to test incoming data:

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

Illustration

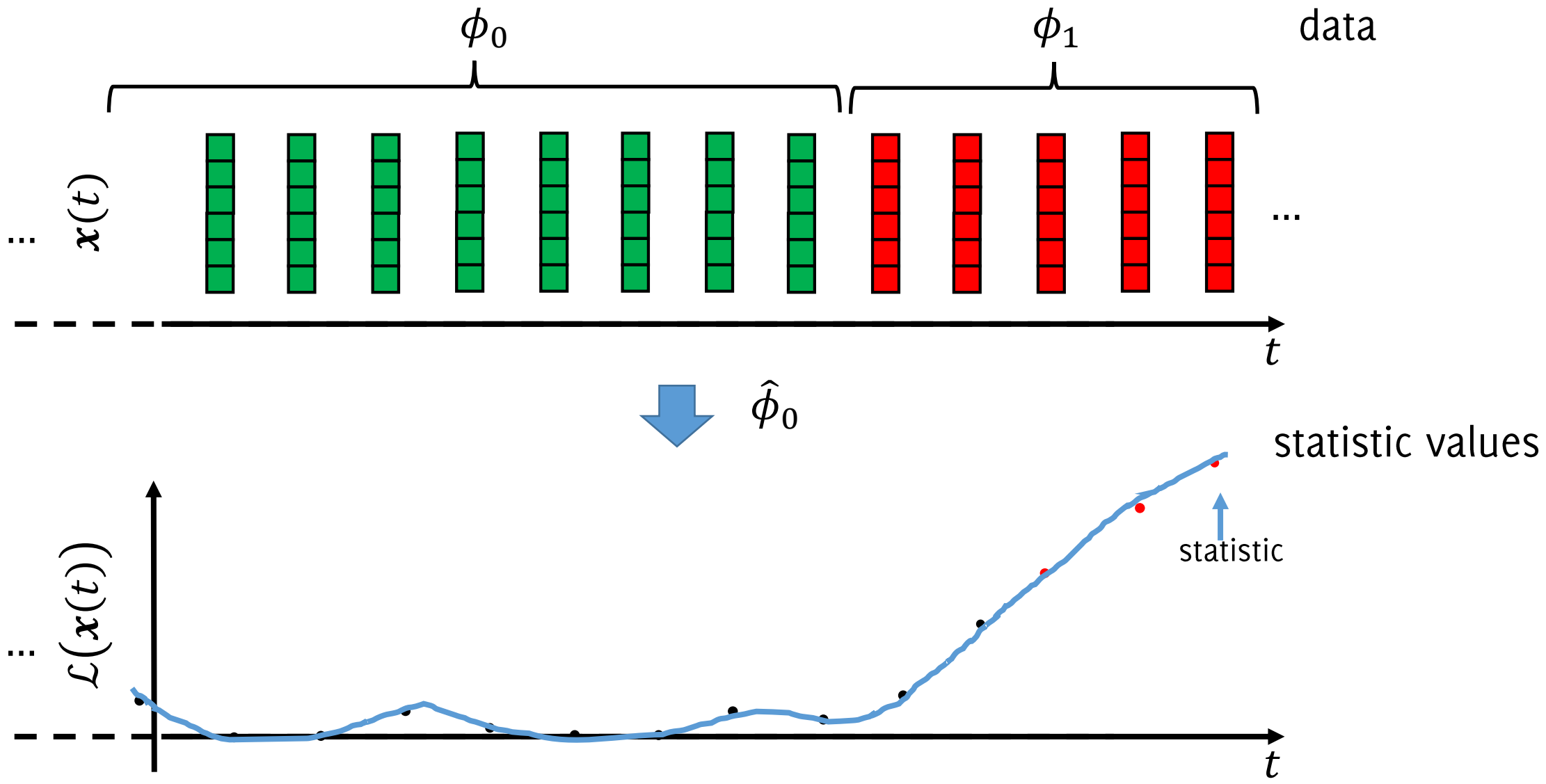


Illustration

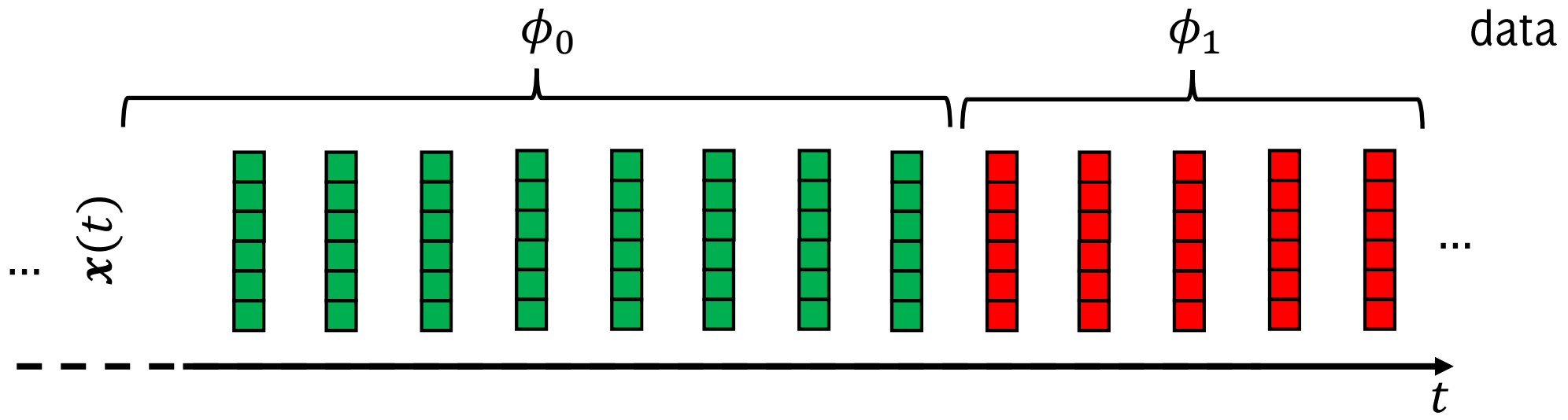


Boracchi, Trovò

Illustration



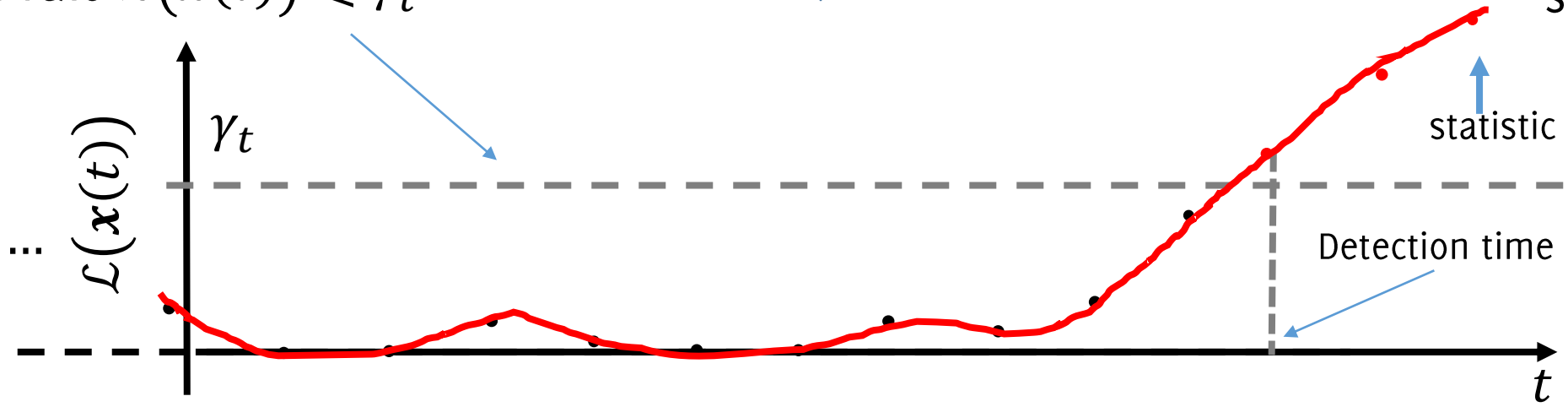
Illustration



$\hat{\phi}_0$

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

statistic values



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

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

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

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

Boracchi, Trovò

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 $S_k \subset \mathcal{X}$

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

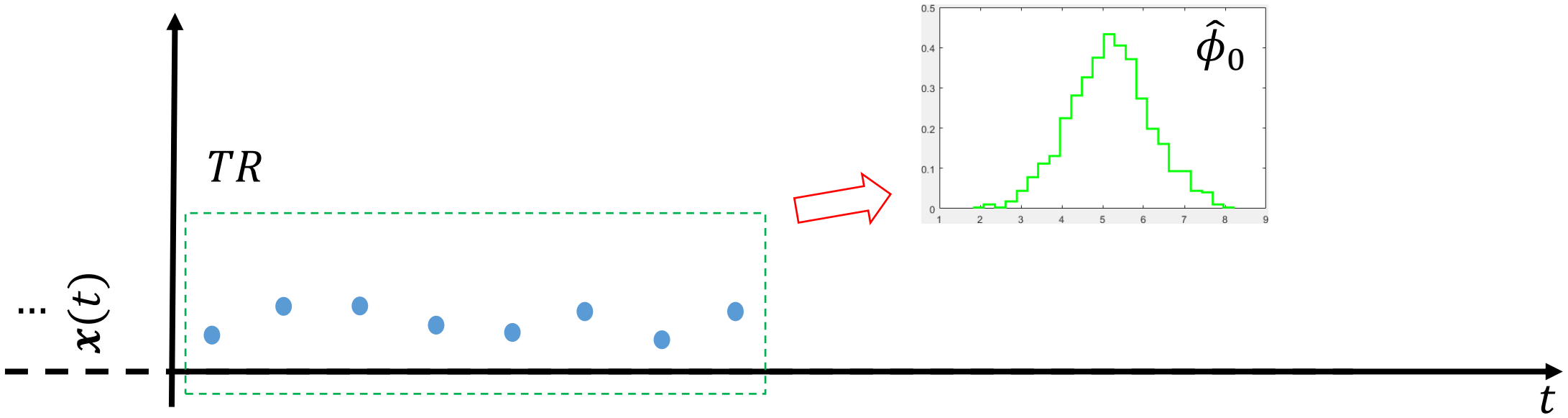
and $p_k^0 \in [0,1]$ is the probability (estimated from TR) for a sample drawn from ϕ_0 to fall inside S_k, i.e.

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

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

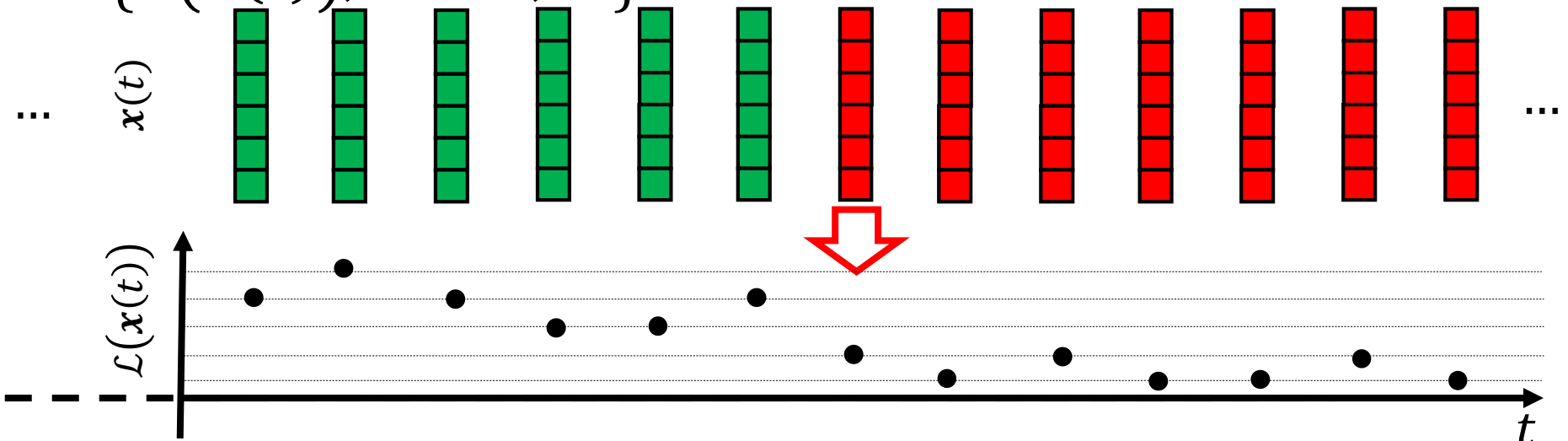
Log-likelihood – Based Monitoring Scheme

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

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

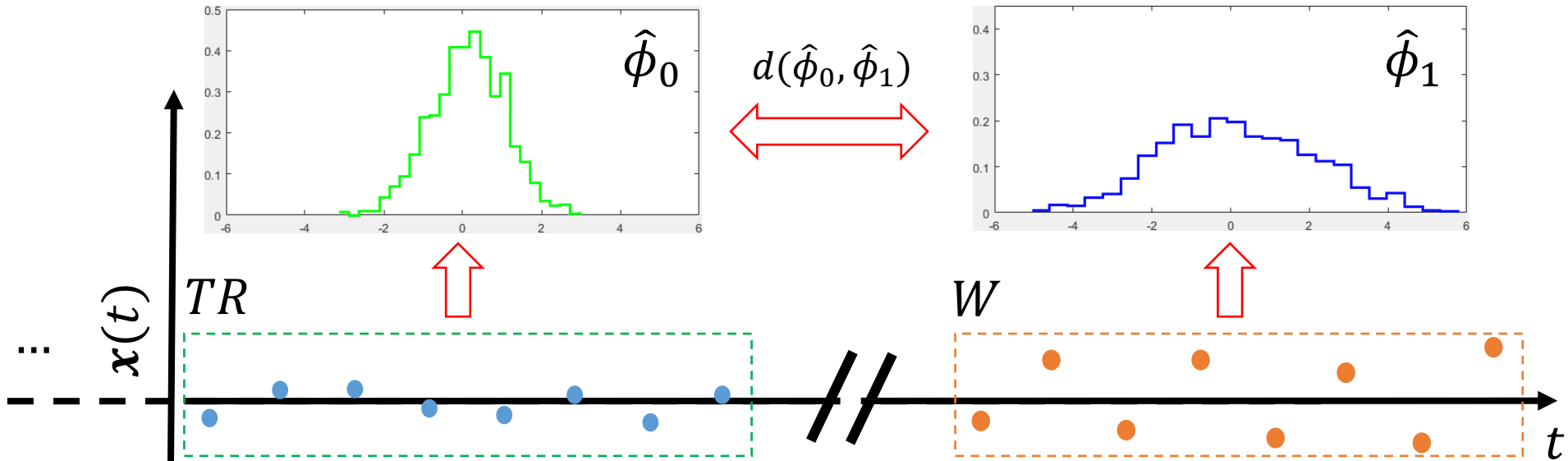
3. Monitor $\{\mathcal{L}(\boldsymbol{x}(t)),\ t = 1, \dots\}$ which is now discrete



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

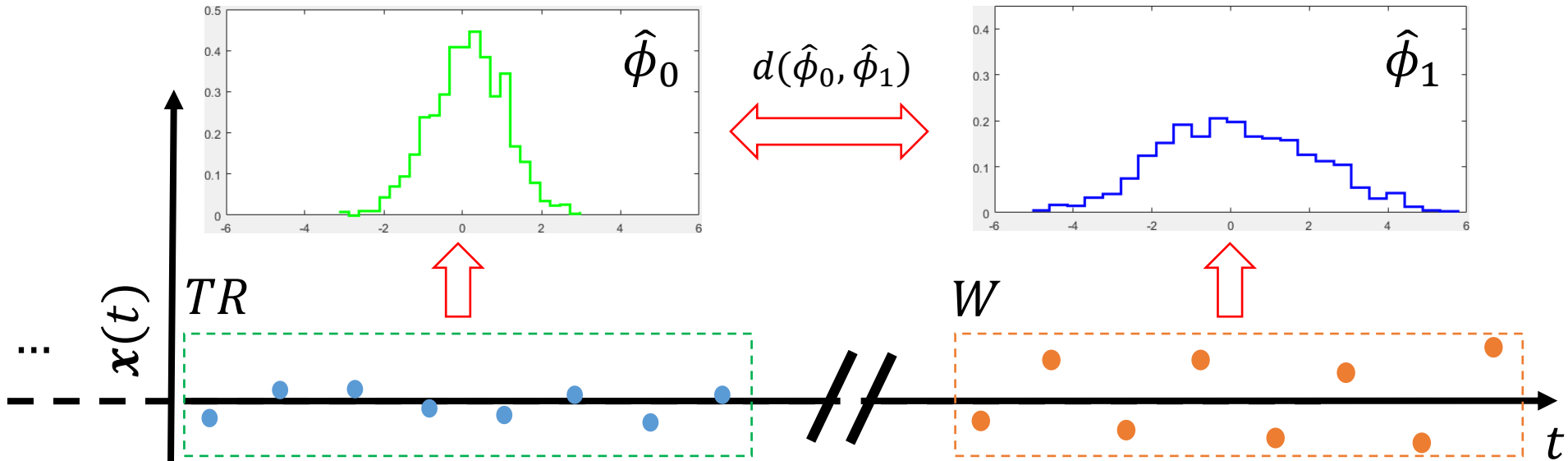


Distance-Based (or batch) Monitoring Scheme

$\hat{\phi}_0$ 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



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

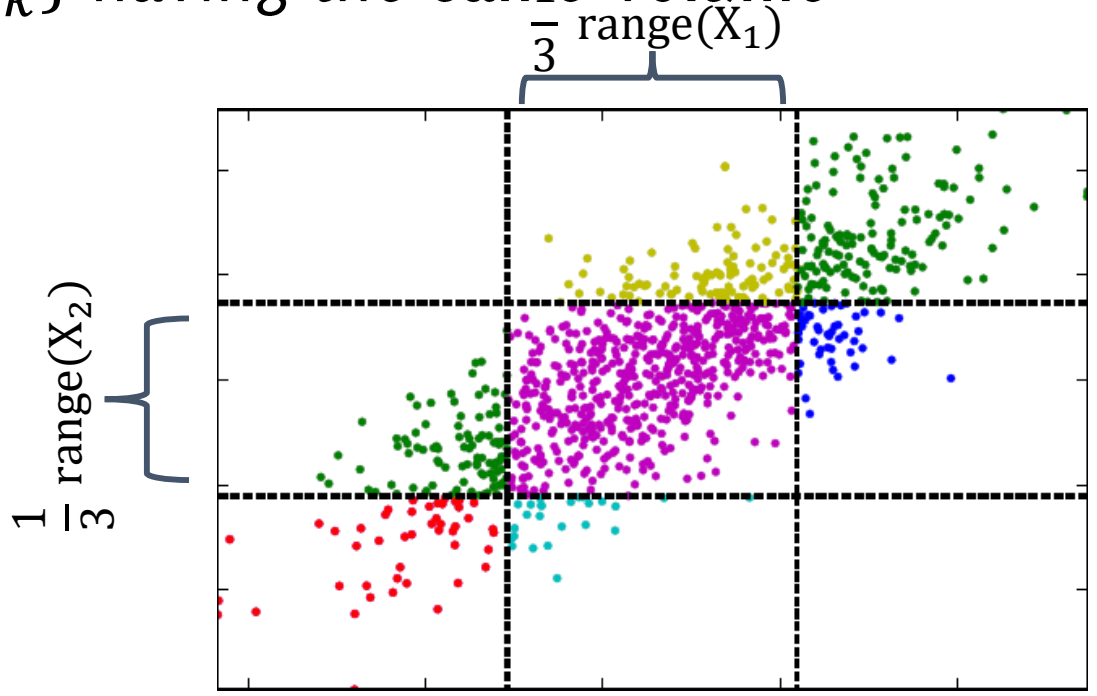
However, there is quite a lot of freedom in designing $\{S_k\}_k$

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.

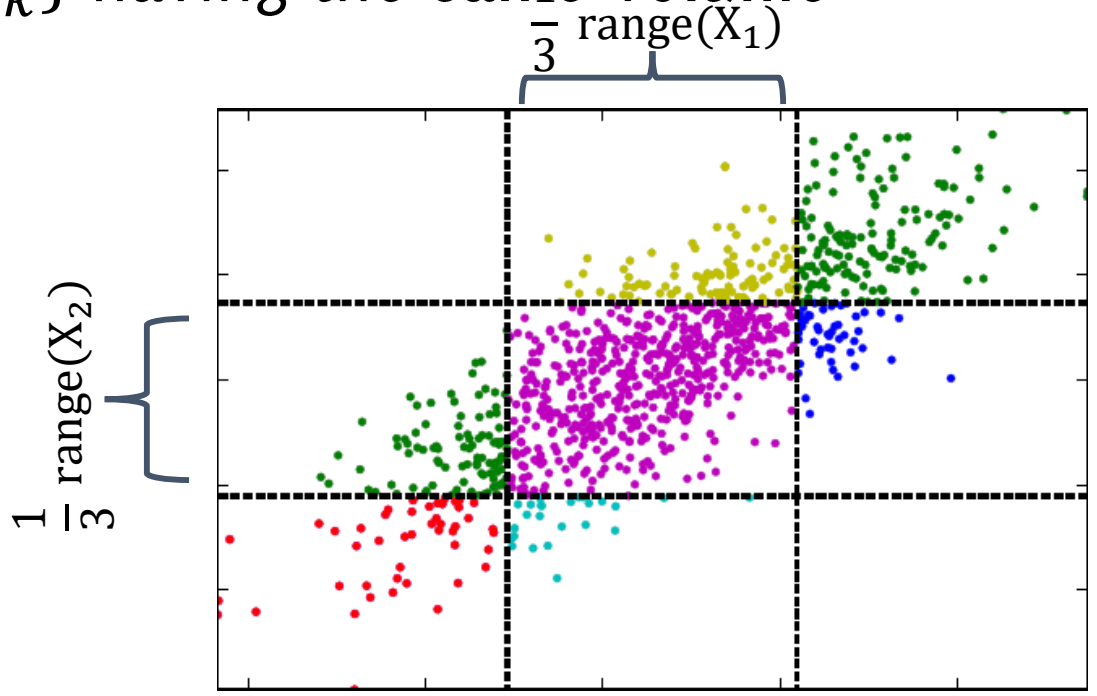
Build a tessellation of $\mathrm{supp}(TR)$ by splitting each component in q equally sized parts.

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 $q = 1/3$

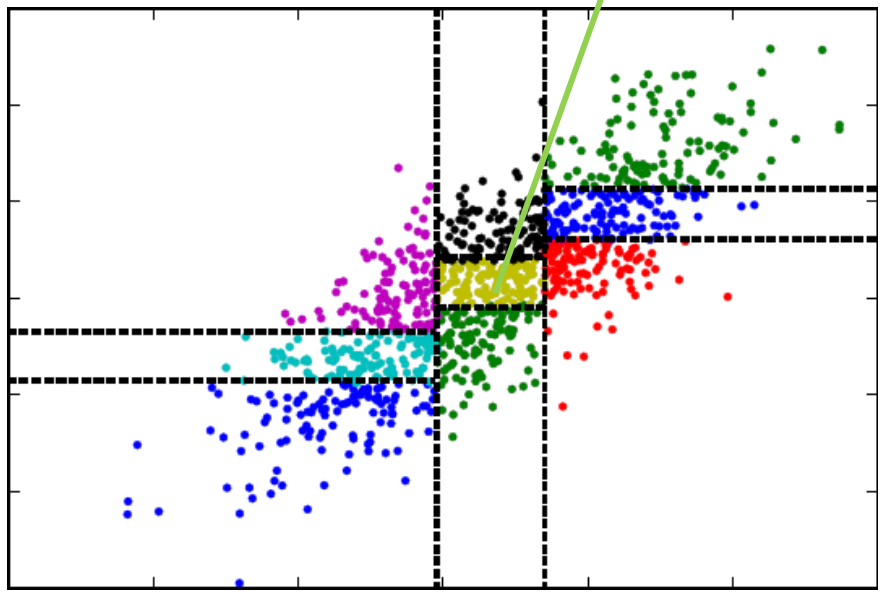
Histograms yielding uniform density

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

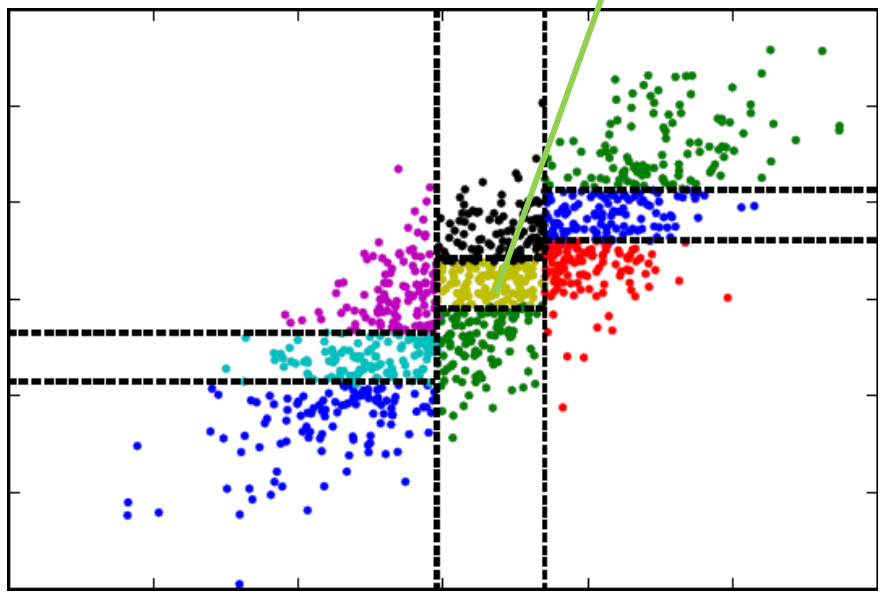
Define the partition $\{S_k\}_k$ in such a way that all the subsets have the **uniform density**, i.e.,

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

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**

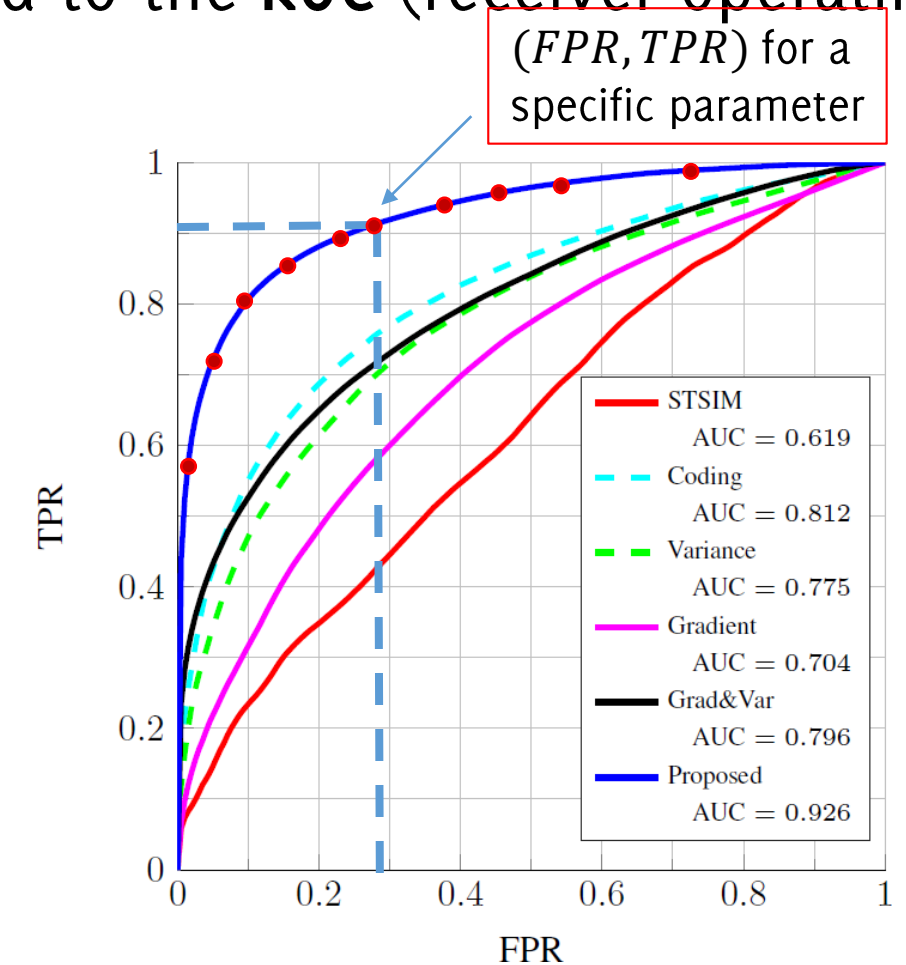
The ideal detector would achieve:

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

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)



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**,

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

and

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

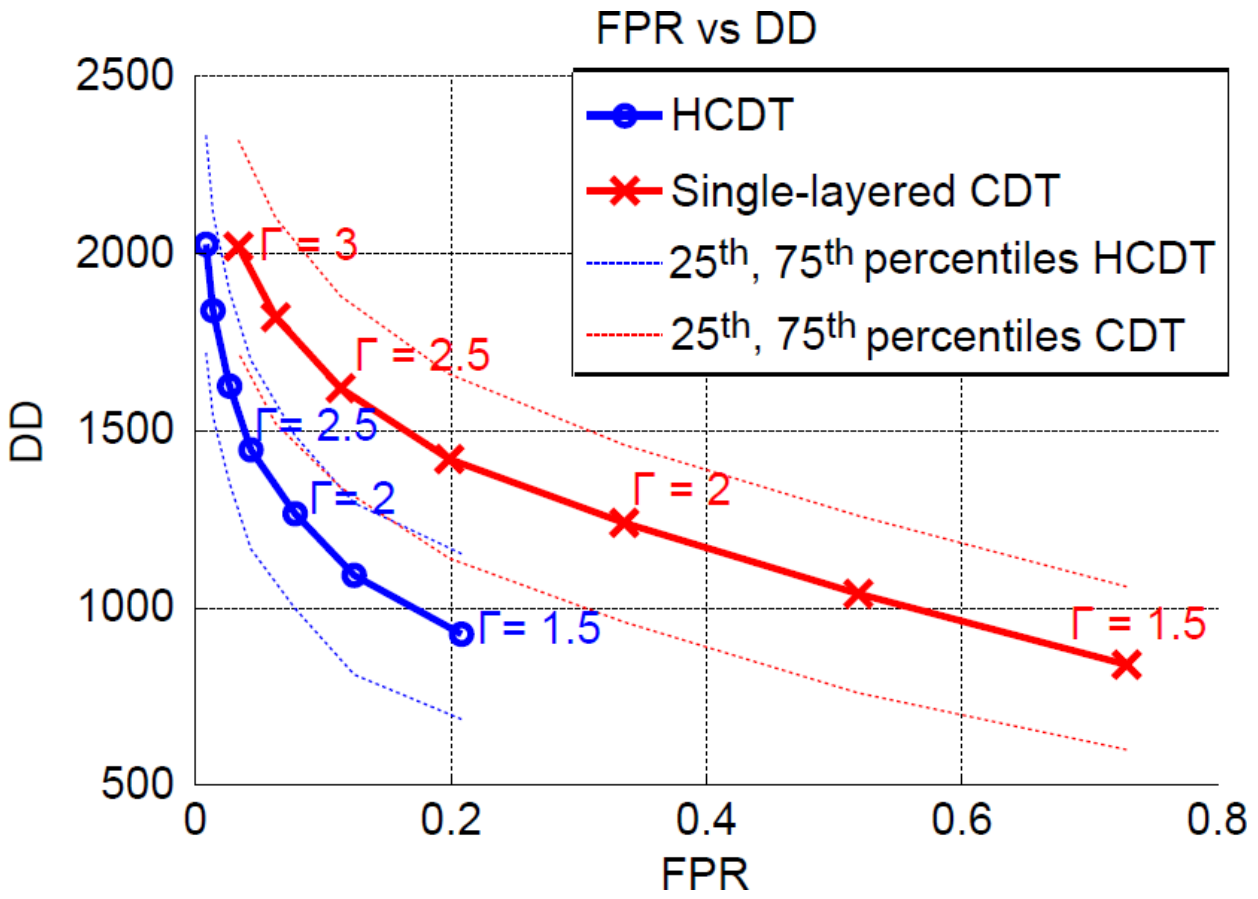
which are defined as in the anomaly detection case, and here depend on the sequence length

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

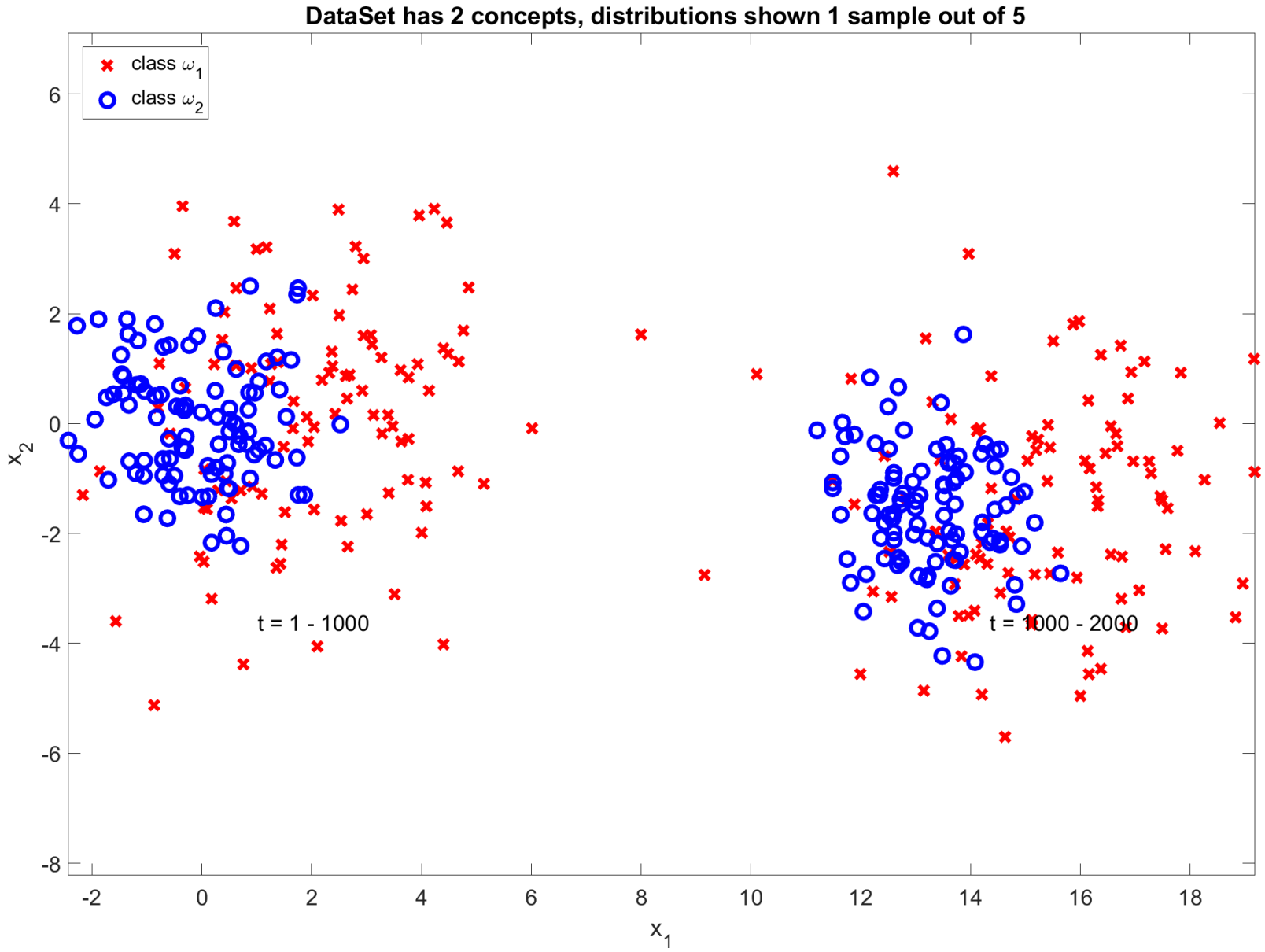
And develop a monitoring scheme that

- **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.

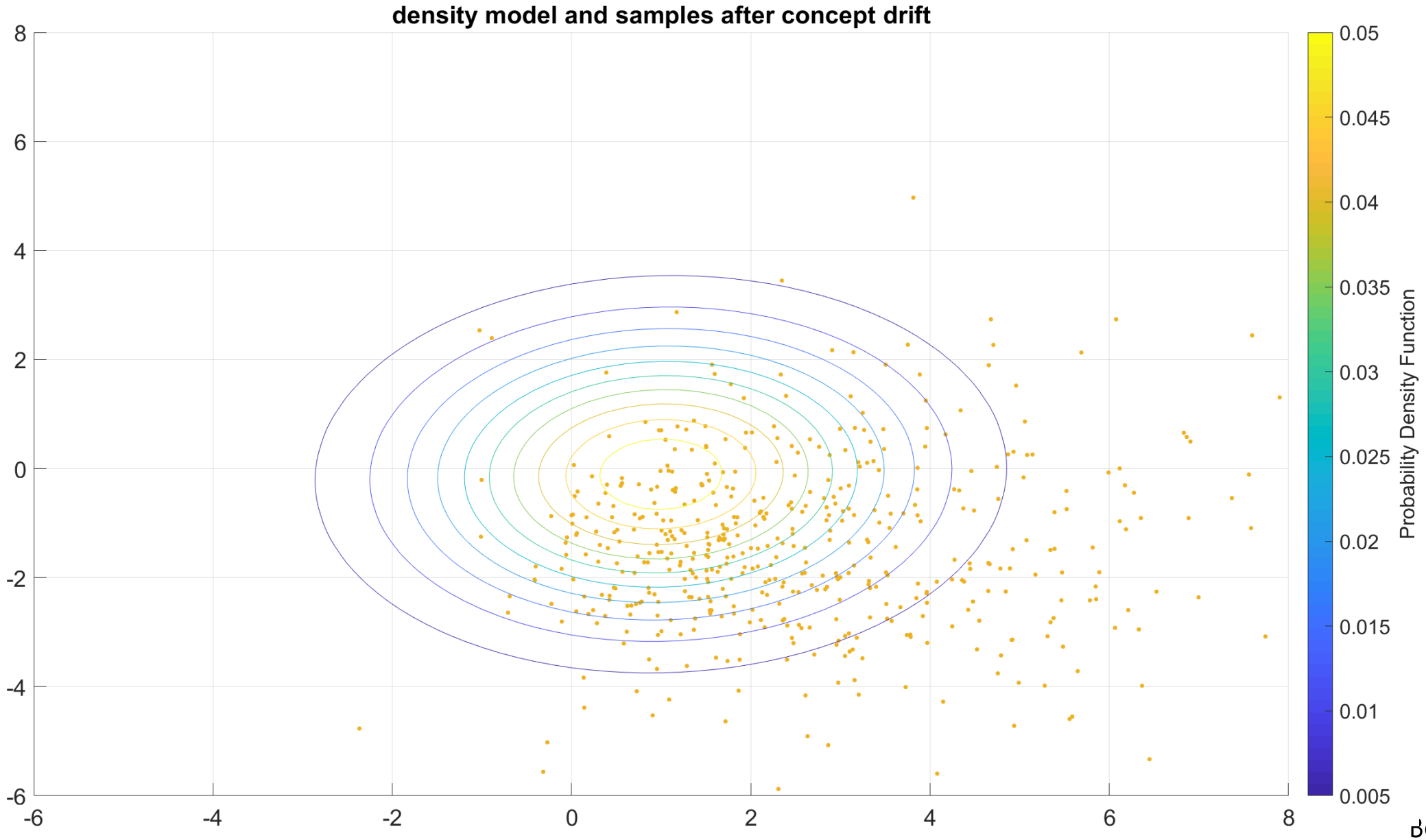
DataSet (change in covariates)



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

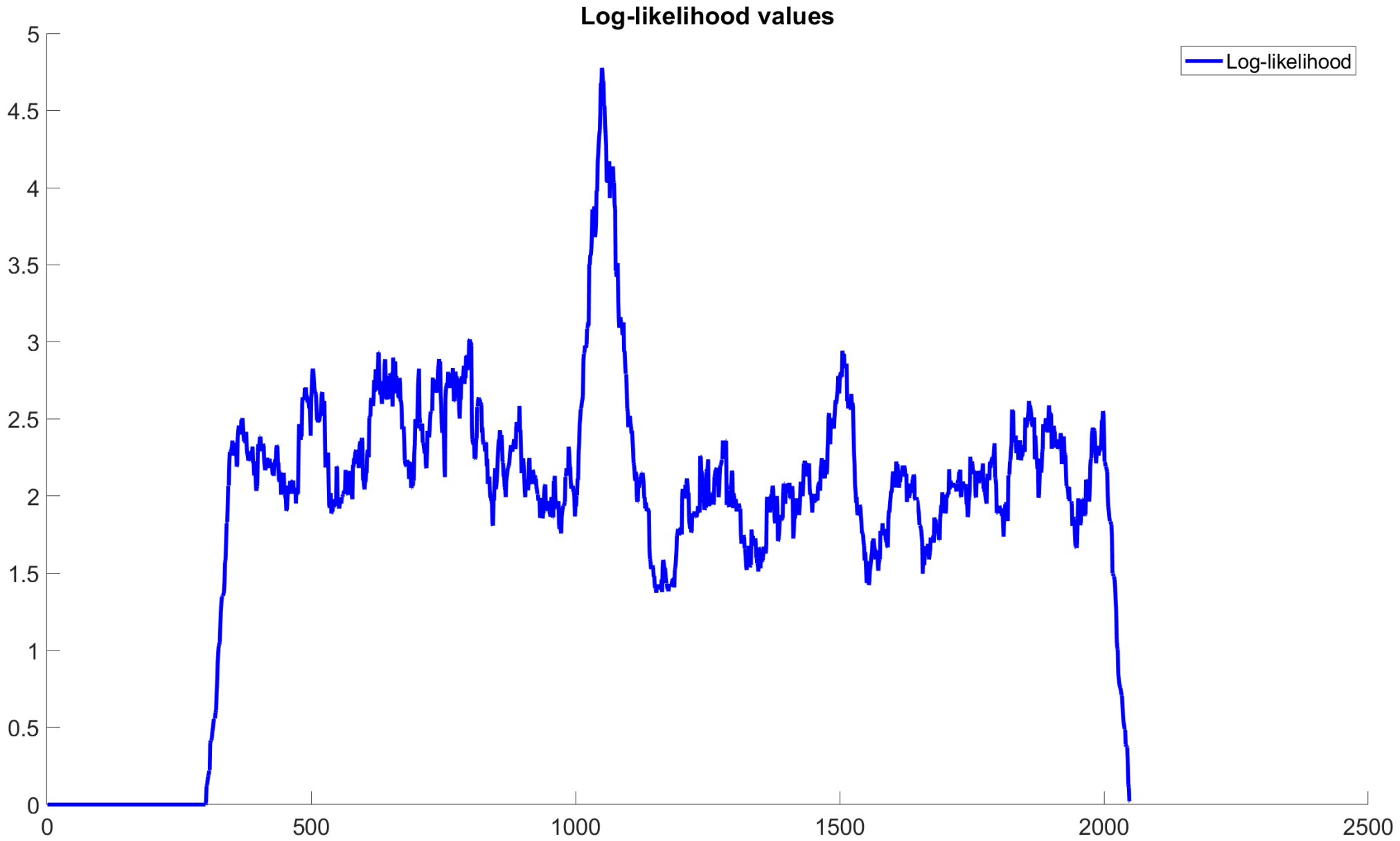
Boracchi, Trovò

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



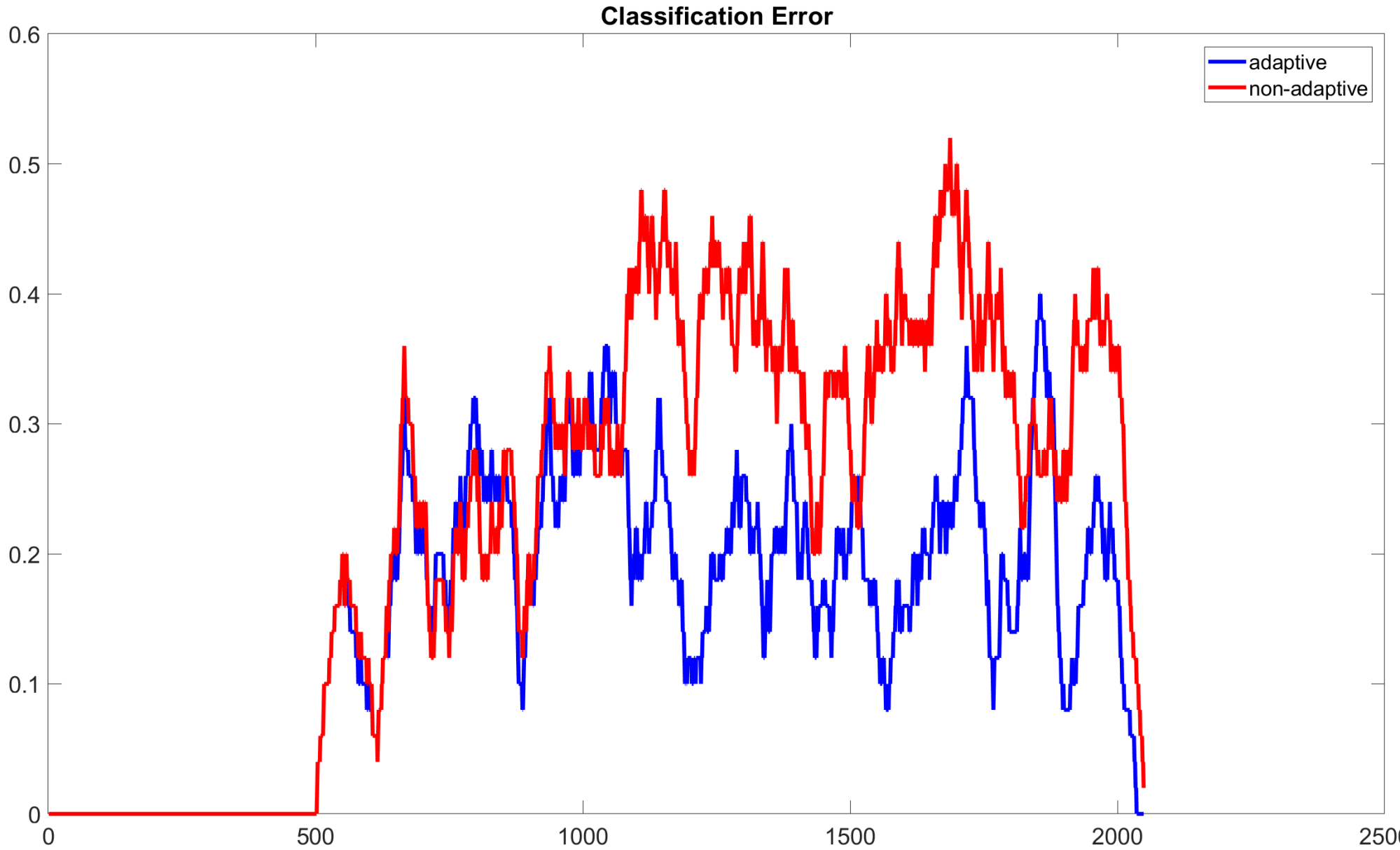
Boracchi, Trovò

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

More on Change Detection

And in particular on high-dimensional data-streams

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

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

Desiderata, Challenge and Goal

Desiderata

i. The model $\hat{\phi}_0$ describing ϕ_0 has to be:
   - general and simple
   - learnable from a training set

ii. The statistic $\mathcal{T}$ used to test incoming data has to:
   - provide a controlled response under ϕ_0
   - provide a different response under ϕ_1

iii. A decision rule that monitors $\mathcal{T}$ has to:
   - promptly detect changes and
   - control FPR (type I error in hypothesis testing) or ARL (averge run length in sequential monitoring)

Research Challenges

Most of the research has been devoted to univariate monitoring schemes

- These were the first case studies in SPC
- Extension to monitoring classification / regression error are straightforward

Challenges for truly multivariate ($d > 1$) monitoring scheme, $d \gg 1$:

- Parametric models $\hat{\phi}_0$ properly matching ϕ_0 are difficult to find
- Non parametric models often require:
  - prohibitively large training sets
  - prohibitively long computing times

Research Challenges

Build a model $\hat{\phi}_0$ and a truly multivariate monitoring scheme that:

- allows change detection in multivariate, possibly high dimensional data
- guarantees a control over the false positives
- it does not require too many training data
- it is very efficient to test

Do I really need $\hat{\phi}^0$?

Of course not….

But it might come handy

Do I really need $\hat{\phi}^0$?

There are a few **non-parametric statistcs** that do need to fit $\hat{\phi}_0$

- Mann Whitney
- Mood
- Lepage
- Kolmogorov-Smirnov
- Cramer Von Mises

Thresholds are either:

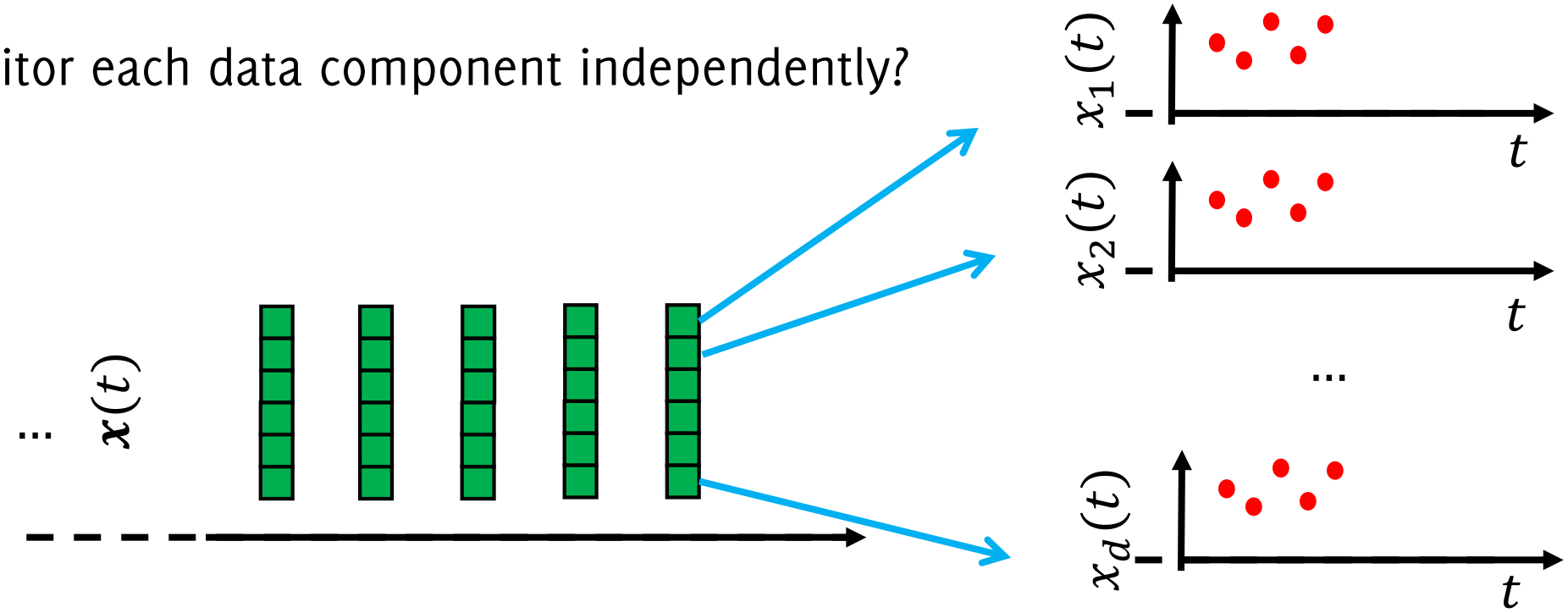
- provided by asymptotic approximation of the statistic
- easily computed from numerical simulations

Unfortunately, most nonparametric statistics relies on sorting and **cannot be extended to multivariate data**

Ross, G. J. and Adams, N. M. "Two nonparametric control charts for detecting arbitrary distribution changes". Journal of Quality Technology, 2012.
Ross, G. J., Tasoulis, D. K., and Adams, N. M. "Nonparametric monitoring of data streams for changes in location and scale". Technometrics, 2011.

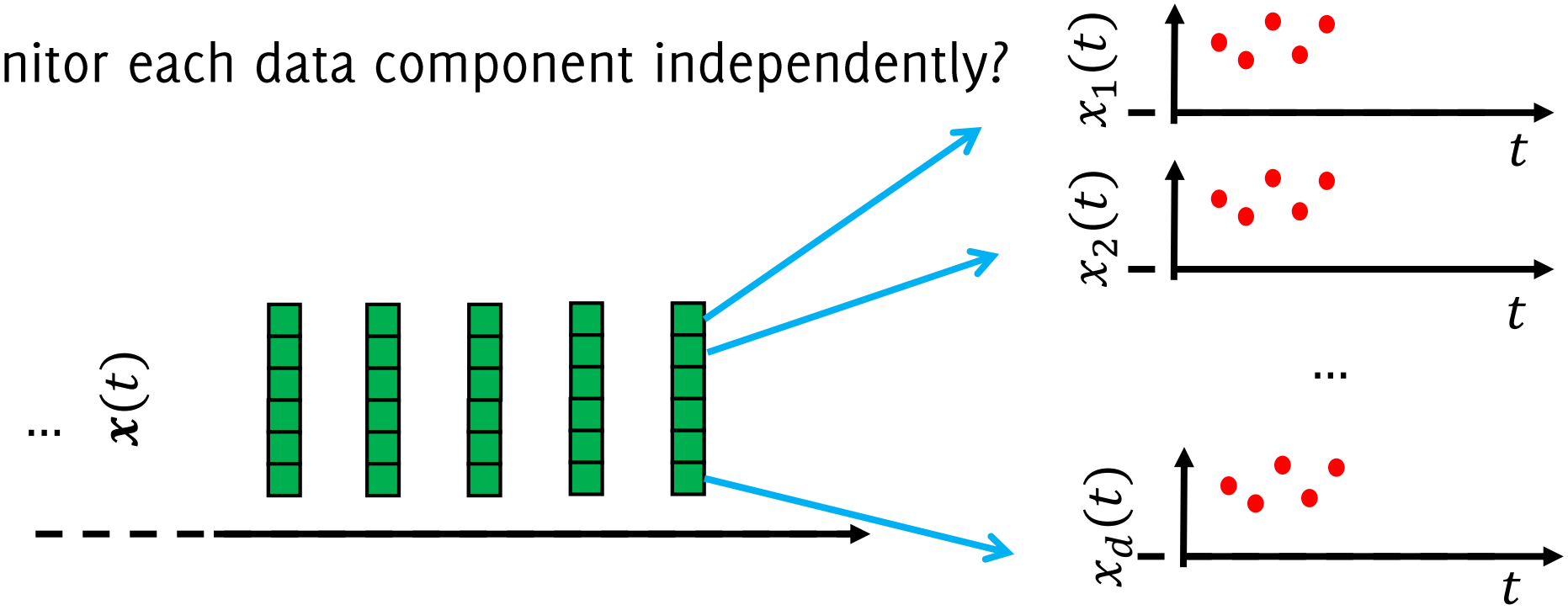
Can’t I go back to a few univariate problems?

Can’t I monitor each data component independently?

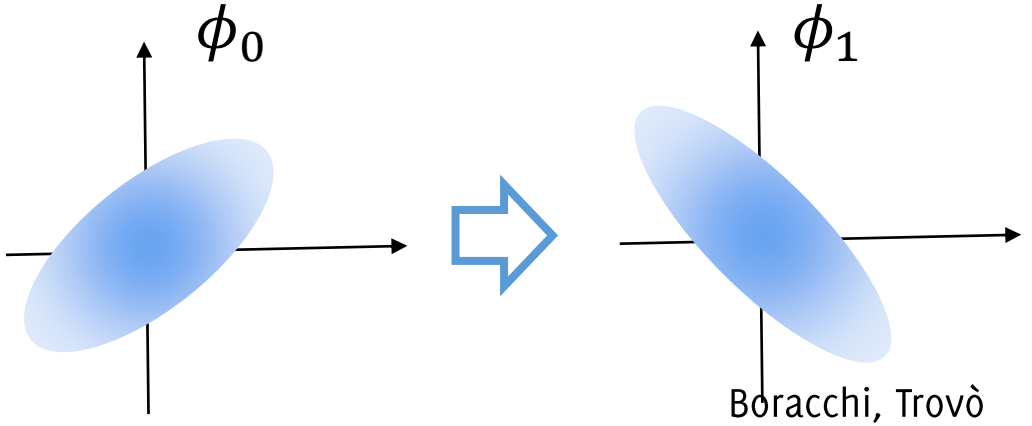


Can’t I go back to a few univariate problems?

Can’t I monitor each data component independently?

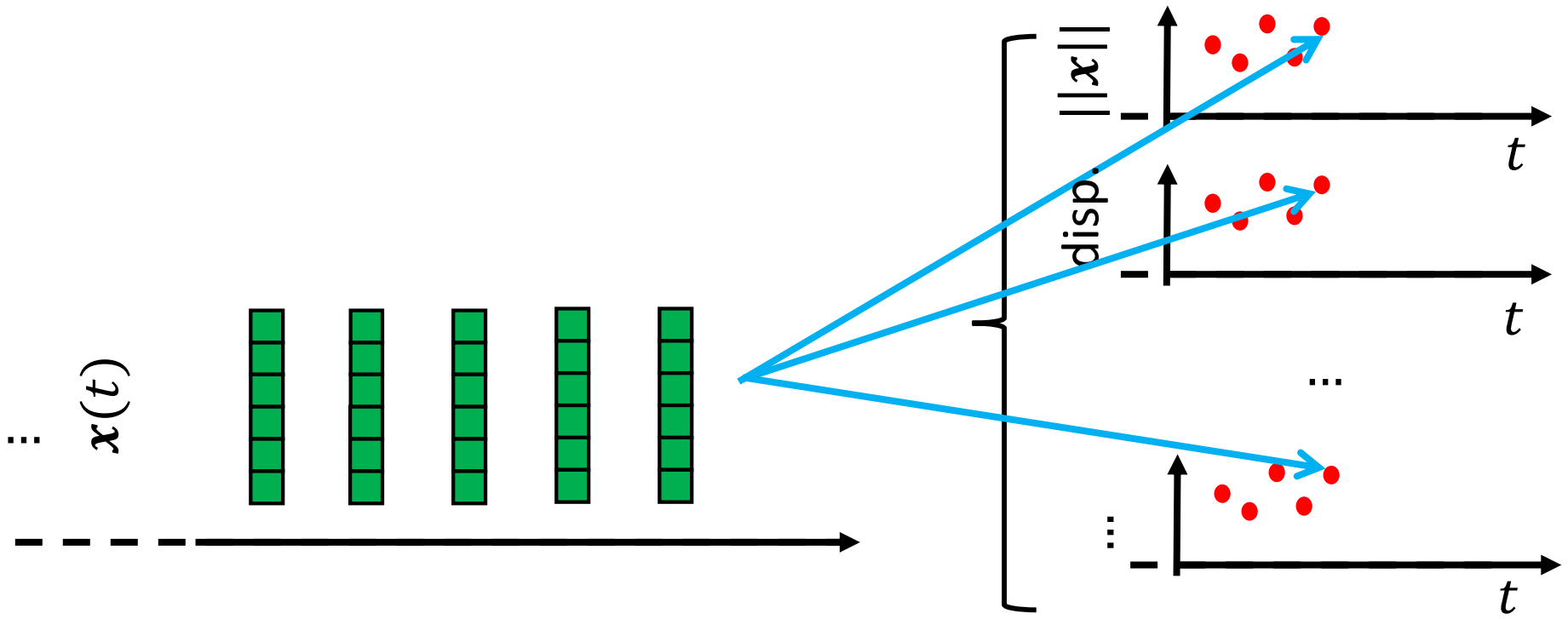


This is **not a truly multivariate** monitoring scheme.
for instance You would not be able to detect
changes affecting correlation



Can’t I go back to a few univariate problems?

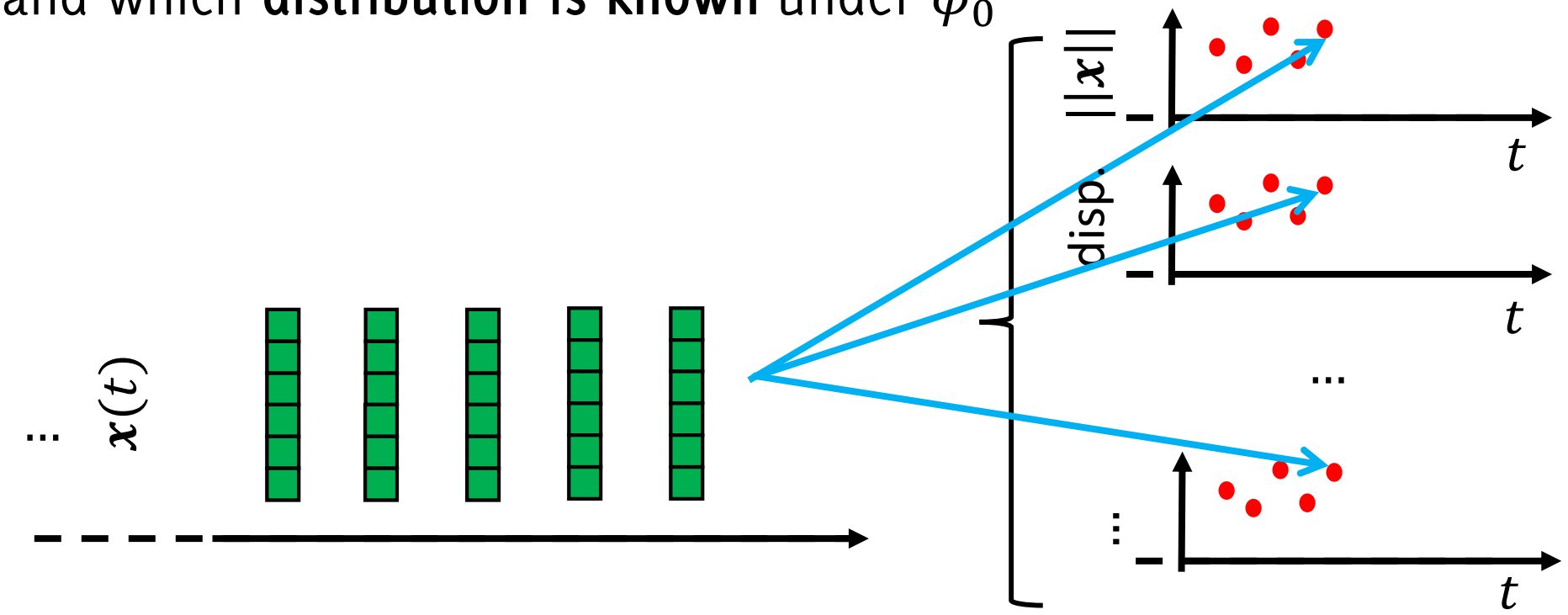
Can’t I **extract a few features / indicators** that are **expected to change** when $\phi_0 \rightarrow \phi_1$ and which **distribution is known** under ϕ_0



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

Can’t I go back to a few univariate problems?

Can’t I **extract a few features / indicators** that are **expected to change** when $\phi_0 \rightarrow \phi_1$ and which **distribution is known** under ϕ_0



Not truly multivariate: **only changes affecting features are detectable.**

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

To warp-up: limitations

When d increases,

- Defining models $\hat{\phi}_0$ that match ϕ_0 becomes more difficult
- Non parametric models often require
  - prohibitively large training sets
  - prohibitively long computing times
- Most nonparametric statistics are based on ranking and do not apply when $d > 1$
- «component-wise monitoring» is too coarse a solution
- «feature-extraction and monitoring» does not lead to a truly multivariate monitoring
- Overall, it is not easy to control FPR / ARL

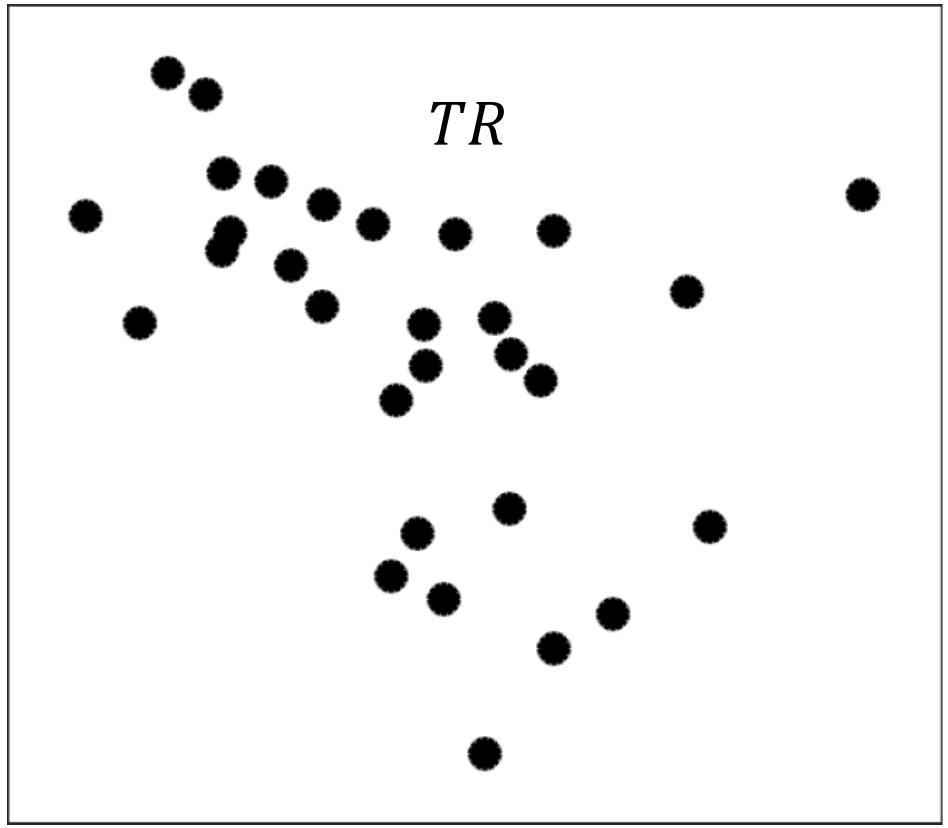
QuantTrees

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

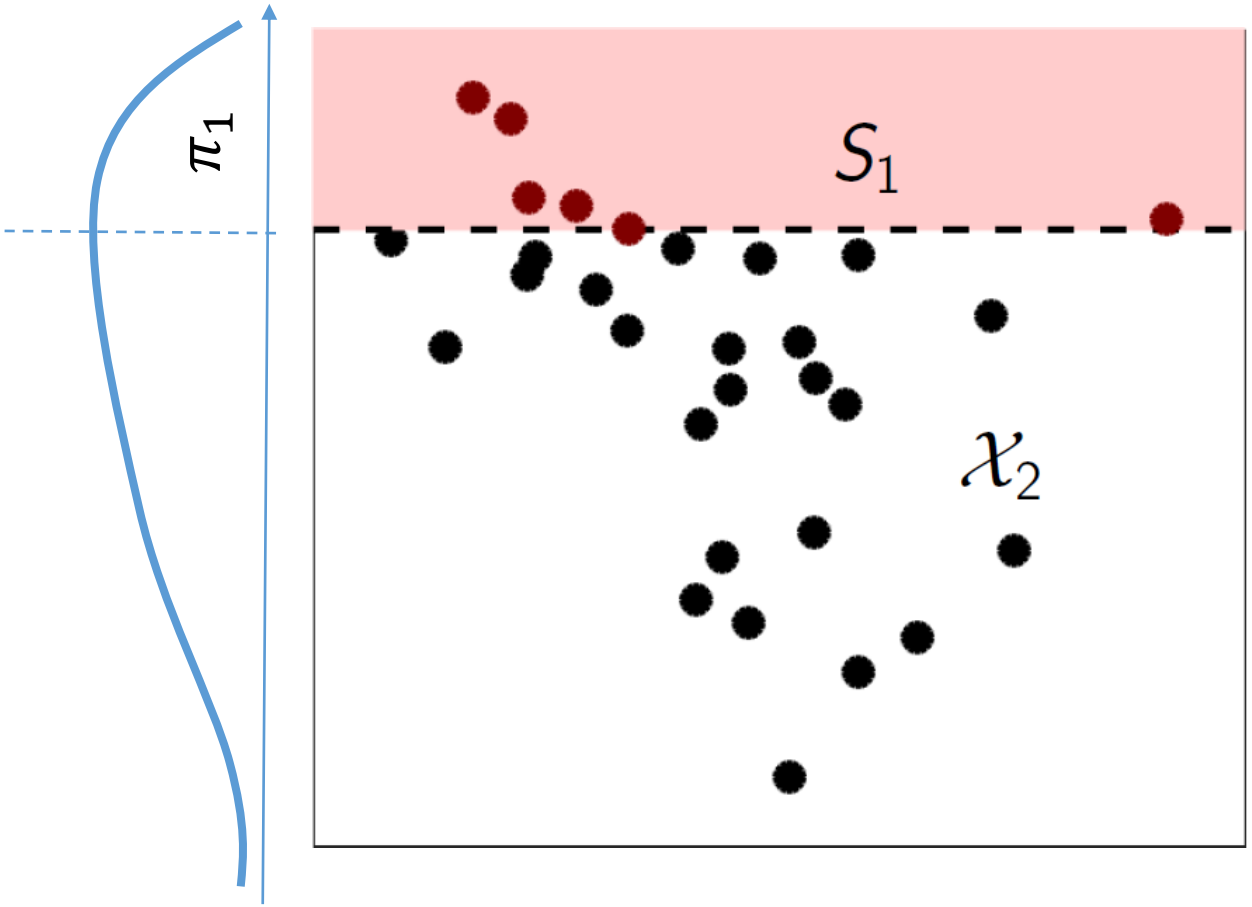
Assume you are given a set of target probabilities $\{\pi_i\}_{i=1,..,K}$ and a training set TR

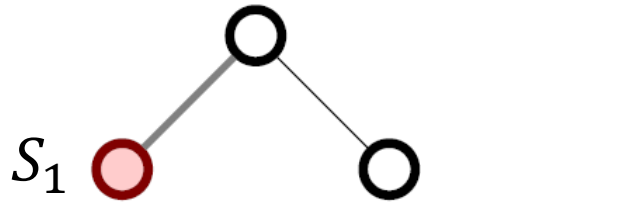


G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

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



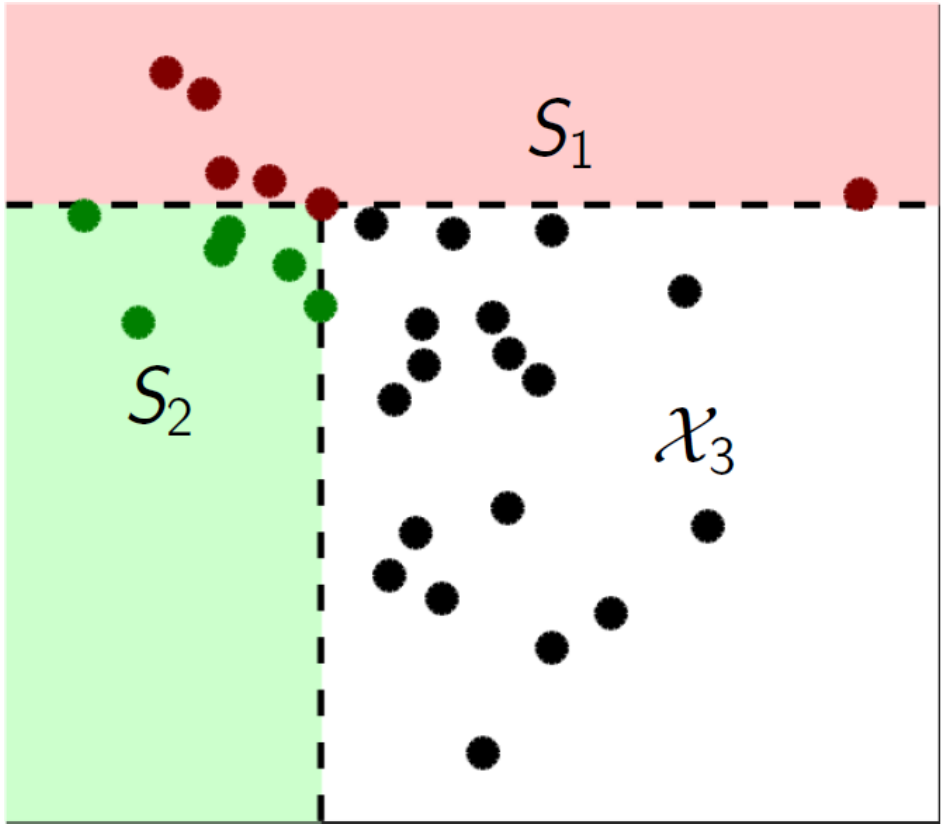


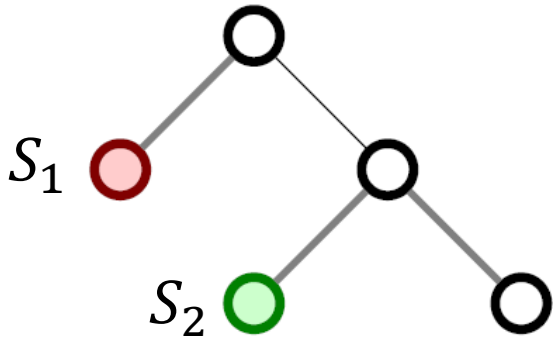
Call $\mathcal{X}_2$ the remaining samples

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.

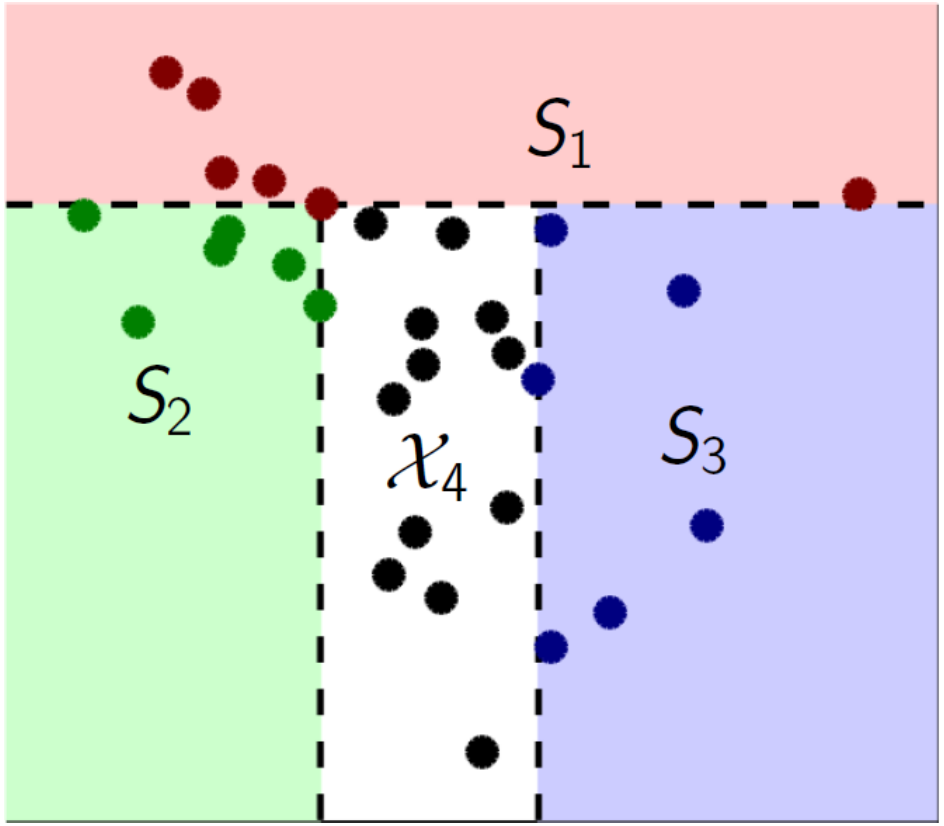


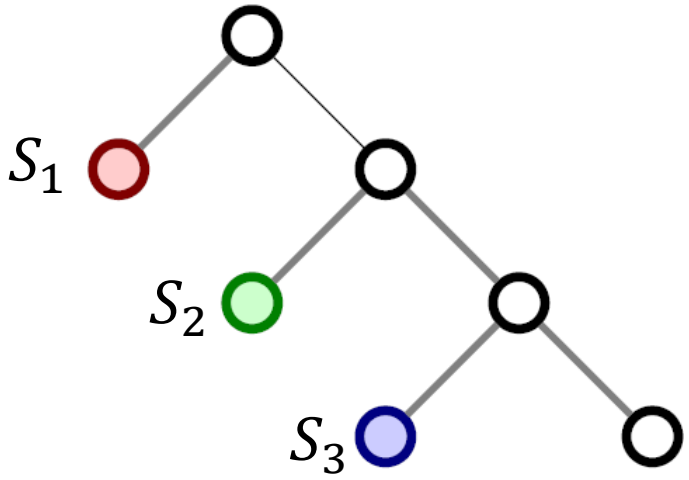


G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

QuantTrees: Histograms for change detection

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

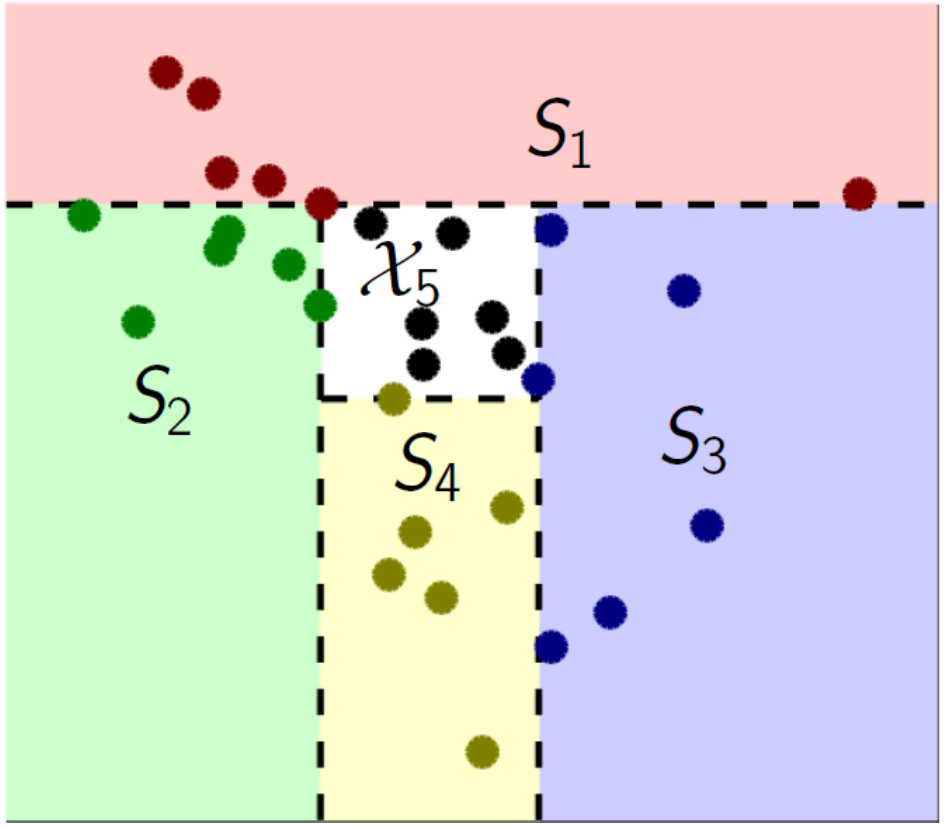


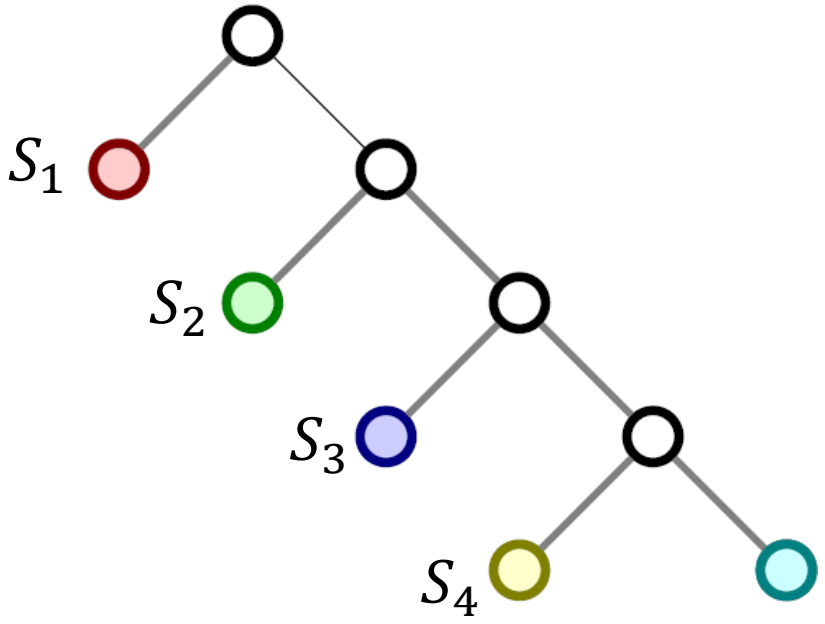


G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

QuantTrees: Histograms for change detection

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

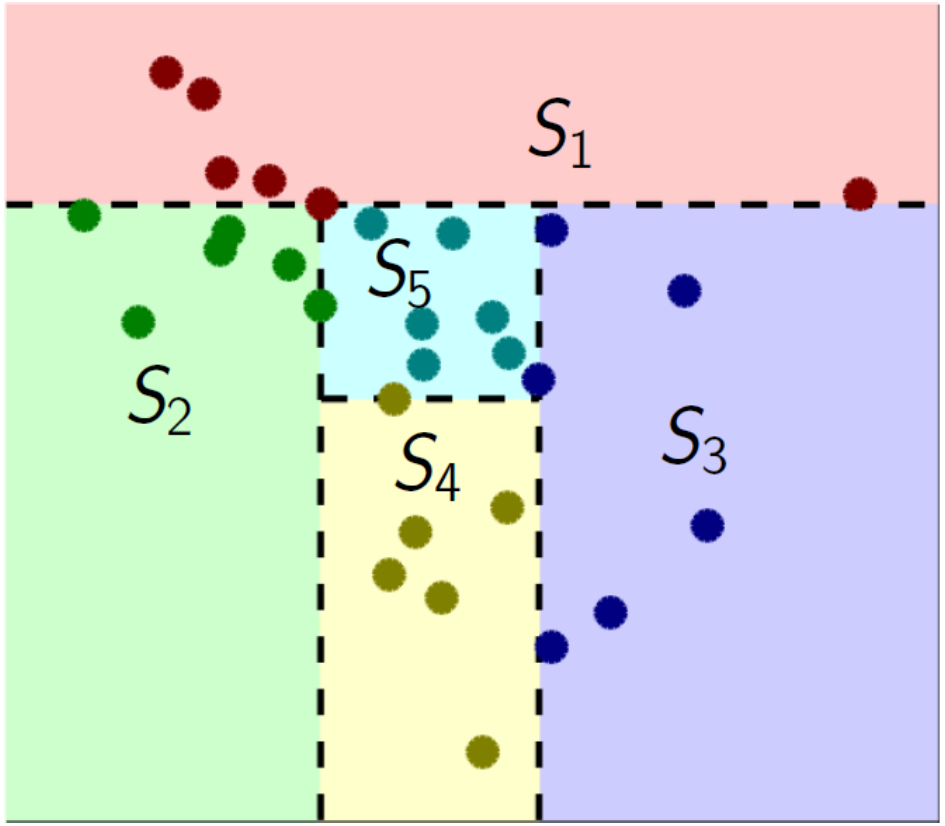


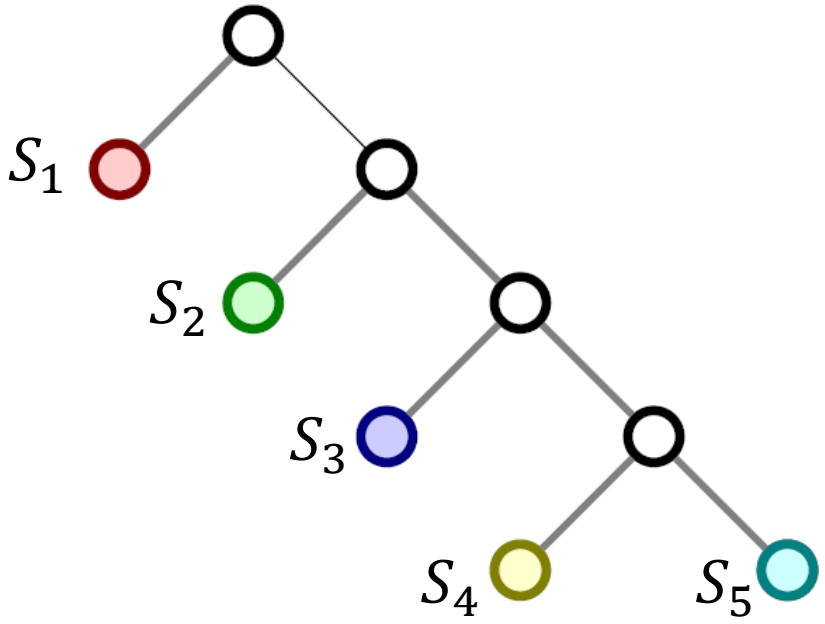


G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

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

QuantTrees: Histograms for change detection

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 $k = 1, \ldots, K$ **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 $i \in \{1, \ldots, d\}$.
    Define $z_n = [\mathbf{x}_n]_i$ for each $\mathbf{x}_n \in \mathcal{X}_k$.
    Sort $\{z_n\}$: $z_{(1)} \leq z_{(2)} \leq \ldots z_{(N_k)}$.
    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

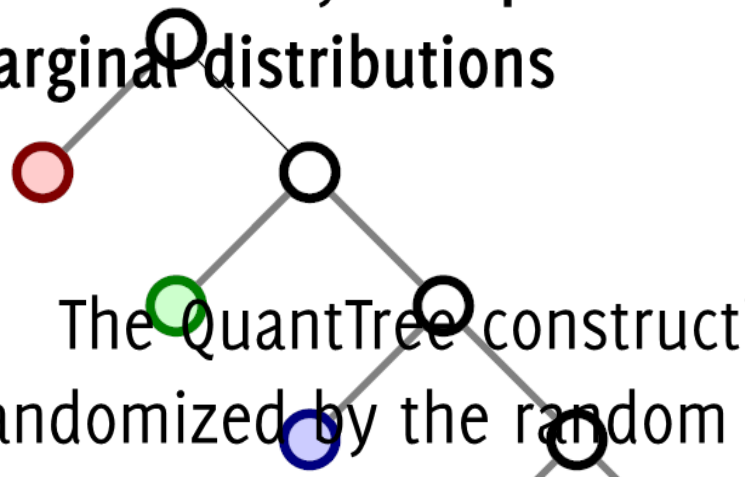
G. Boracchi, D. Carrera, C. Cervellera, D. Macciò *"QuantTree: Histograms for Change Detection in Multivariate Data Streams"* ICML 2018

QuantTrees: Histograms for change detection

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

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**

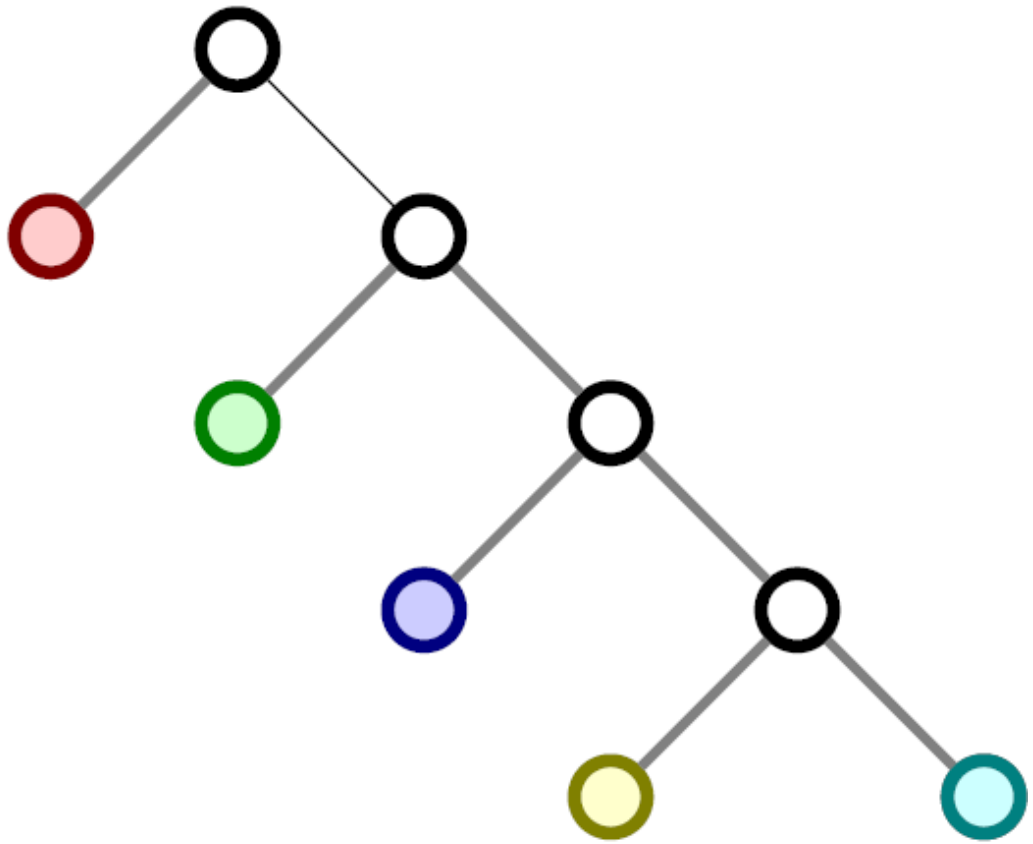


The QuantTree construction is randomized by the random selection of the component for each split and whether to take the π_i or $1 - \pi_i$ quantile

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 resulting histogram corresponds to a tree, which is highly unbalanced (thus not efficient to be scanned) that is very advantageous for change-detection purposes.



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

QuantTrees: Histograms for change detection

Theorem (ICML18)

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

When $W \sim \phi_0$, the distribution of $T_h(W)$ depends only on:

- the number of training samples N,
- the size of window W,
- the expected probabilities in each bin $\{\pi_i\}_{i=1,\dots,K}$

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, the bin probabilities do not depend on ϕ_0, nor data dimension d.

Thus, **thresholds** of tests statistics **can be numerically computed from univariate data** that have been synthetically generated, yet guaranteeing a controlled false positive rate.

	$d > 1$	$d = 1$
Training	$O(KN \log N)$	$O(N \log N)$
Test	$O(K)$	$O(\log K)$

Example of Thresholds Computed for QuantTrees (from 1D Montecarlo simulation)

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

QuantTrees

Provide a model $\hat{\phi}_0$ and a truly multivariate monitoring scheme that:

- allows change detection in multivariate, possibly high dimensional data
- guarantees a control over the false positives
- it does not require too many training data
- it is very efficient to test

Hierarchical Change Detection

Hierarchical Change-detection tests

In nonparametric sequential monitoring it is convenient to

- **online sequential CDTs** for detection purposes
- **offline hypothesis tests** for validation purposes.

Alippi, C., Boracchi, G., Roveri, M. *"Hierarchical Change-Detection Tests"* TNNLS 2016 pp 1- 13

Hierarchical Change-detection tests

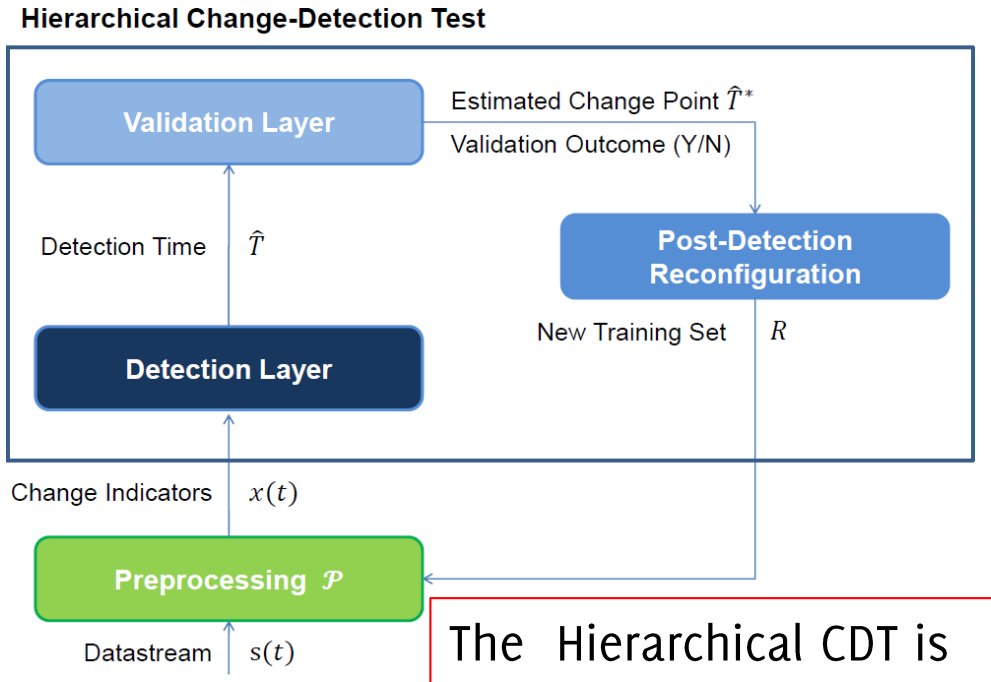
In nonparametric sequential monitoring it is convenient to

- **online sequential CDTs** for detection purposes
- **offline hypothesis tests** for validation purposes.

This results in two-layered (hierarchical) CDTs

Offline HT is activated to validate any detection

Online CDT detects process changes in the input datastream

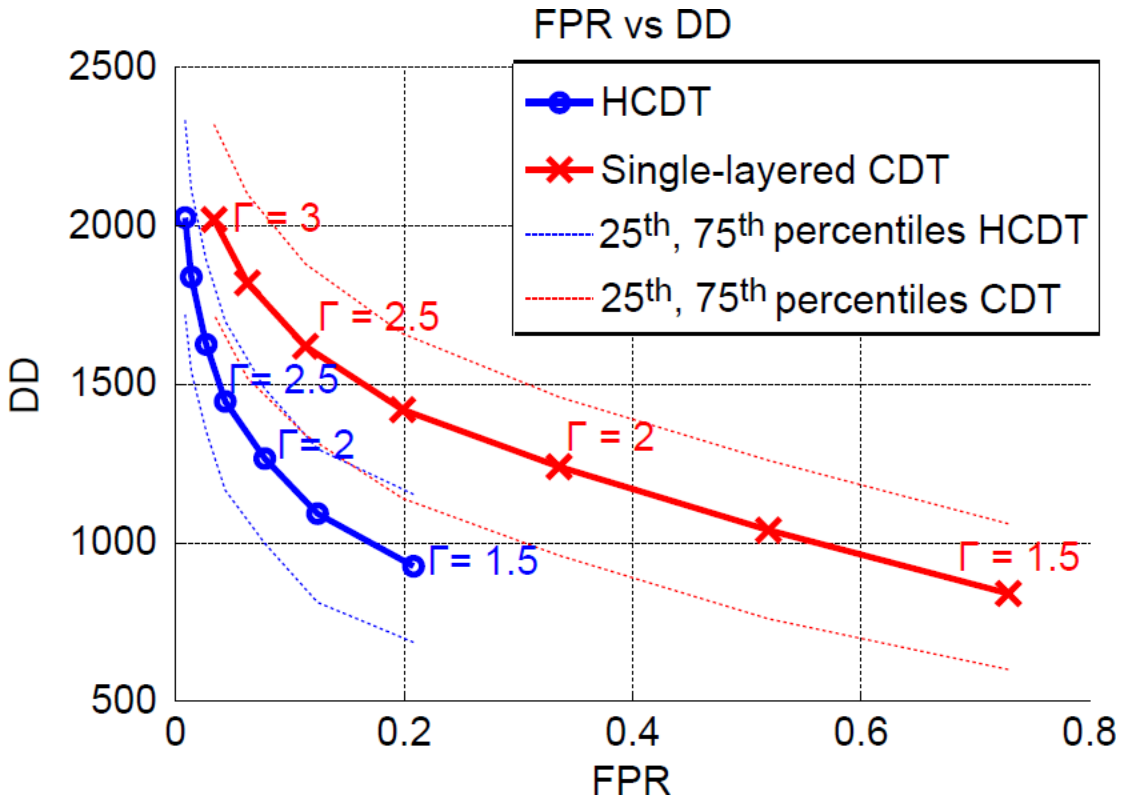


The Hierarchical CDT is automatically reconfigured

Alippi, C., Boracchi, G., Roveri, M. *"Hierarchical Change-Detection Tests"* TNNLS 2016 pp 1- 13

Hierarchical Change-detection tests

Hierarchical CDTs can achieve a far **more advantageous trade-off** between false-positive rate and detection delay **than their single-layered**, more traditional, **counterpart**.



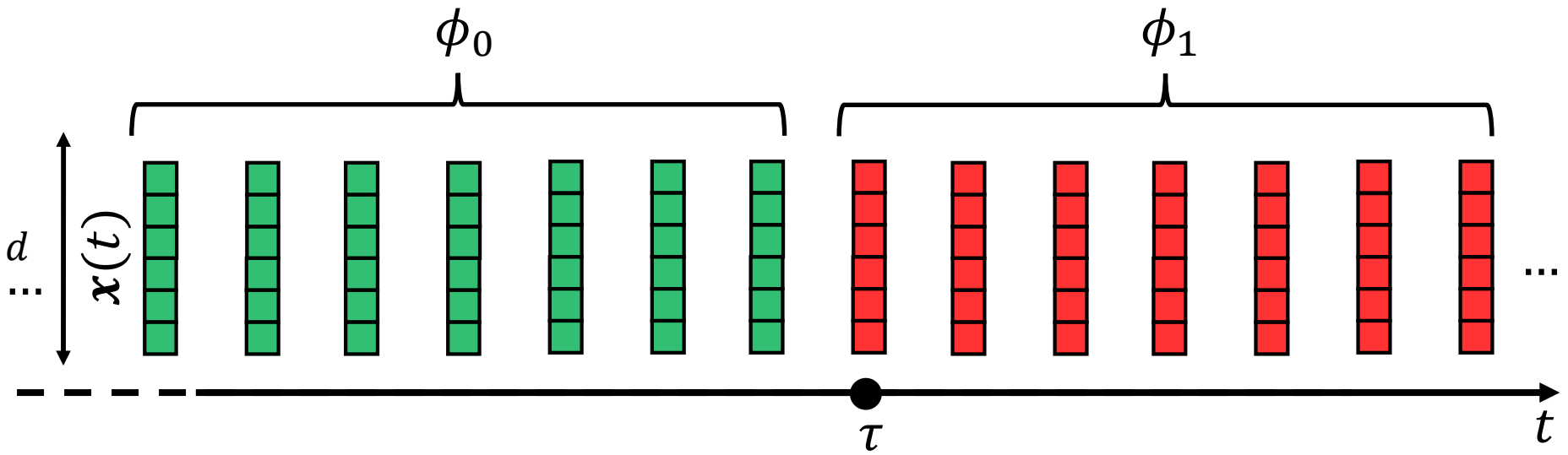
Alippi, C., Boracchi, G., Roveri, M. *"Hierarchical Change-Detection Tests"* TNNLS 2016 pp 1- 13

Detectability loss in high-dimensional data

How data dimension affects monitoring the Log-likelihood

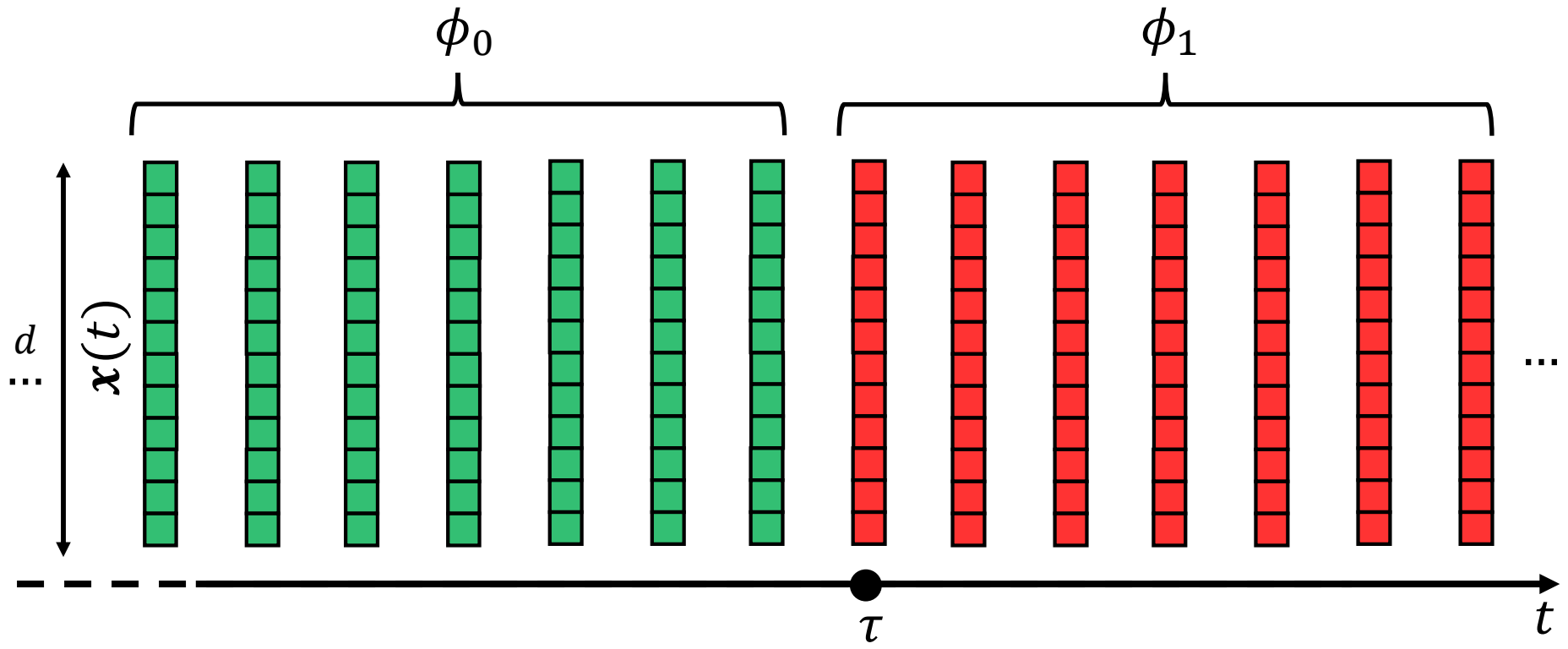
Our Goal

*Study how the **data dimension** d **influences** the **change detectability**, i.e., how difficult is to solve change/anomaly detection problems*



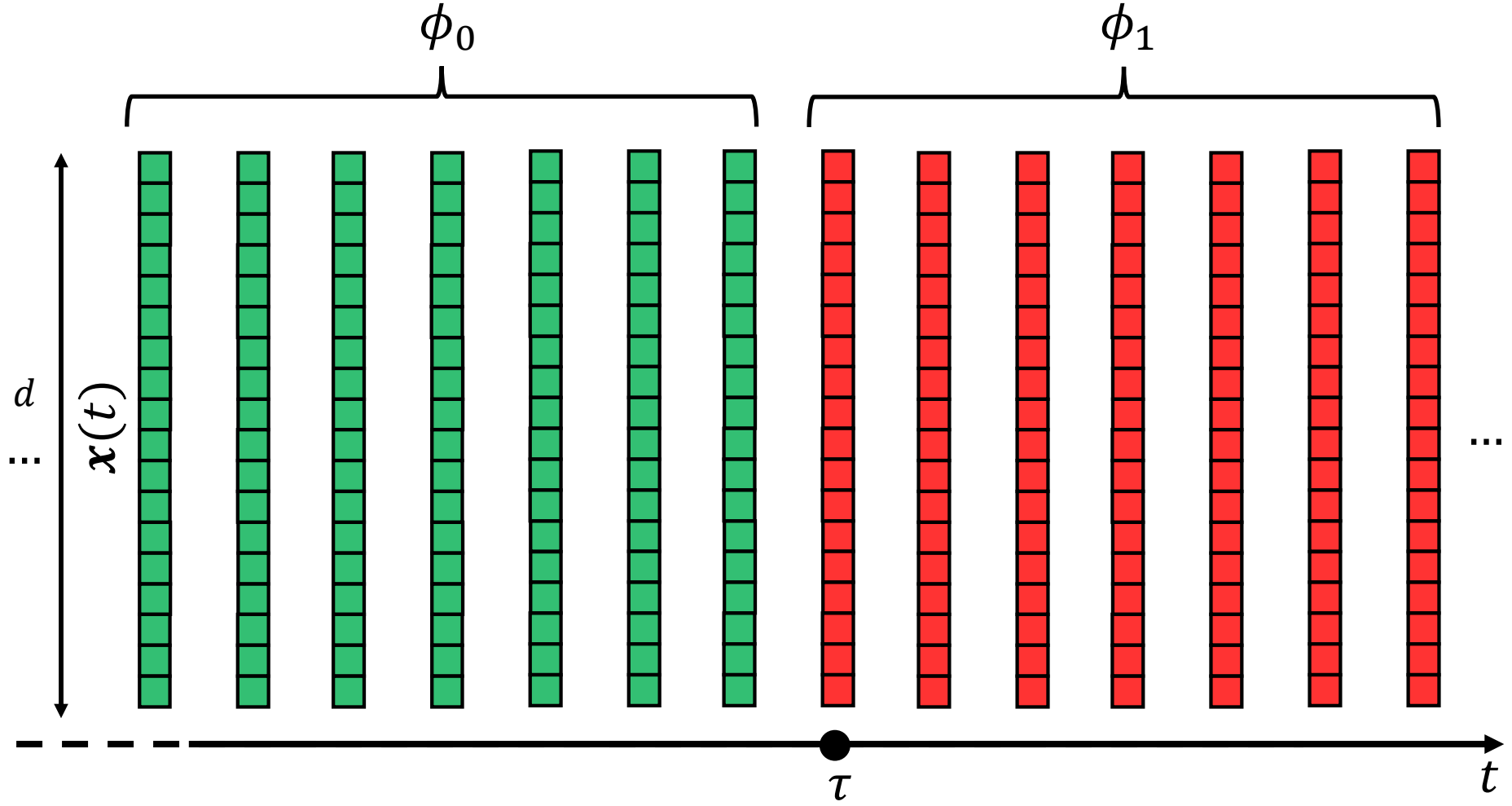
Our Goal

*Study how the **data dimension** d **influences** the **change detectability**, i.e., how difficult is to solve change/anomaly detection problems*



Our Goal

*Study how the **data dimension** d **influences** the **change detectability**, i.e., how difficult is to solve change/anomaly detection problems*



Our Approach

To study the impact of the **sole data dimension** d in **change-detection problems**:

1. Consider a **change-detection approach**
2. Define a measure of **change detectability** that well correlates with traditional performance measures
3. Define a measure of **change magnitude** that refers only to differences between ϕ_0 and ϕ_1

Our Result

We show there is a **detectability loss** problem, i.e. that change **detectability** steadily **decreases** when d increases.

Detectability loss is shown by:

- Analytical derivations: when ϕ_0 and ϕ_1 are **Gaussians**
- Empirical analysis on real data: measuring the **power of hypothesis tests**

Roadmap to detectability loss

Preliminaries:

- The change-detection approach
- The measure of change detectability
- The change magnitude

The *detectability loss*

- Analytical results
- Empirical analysis

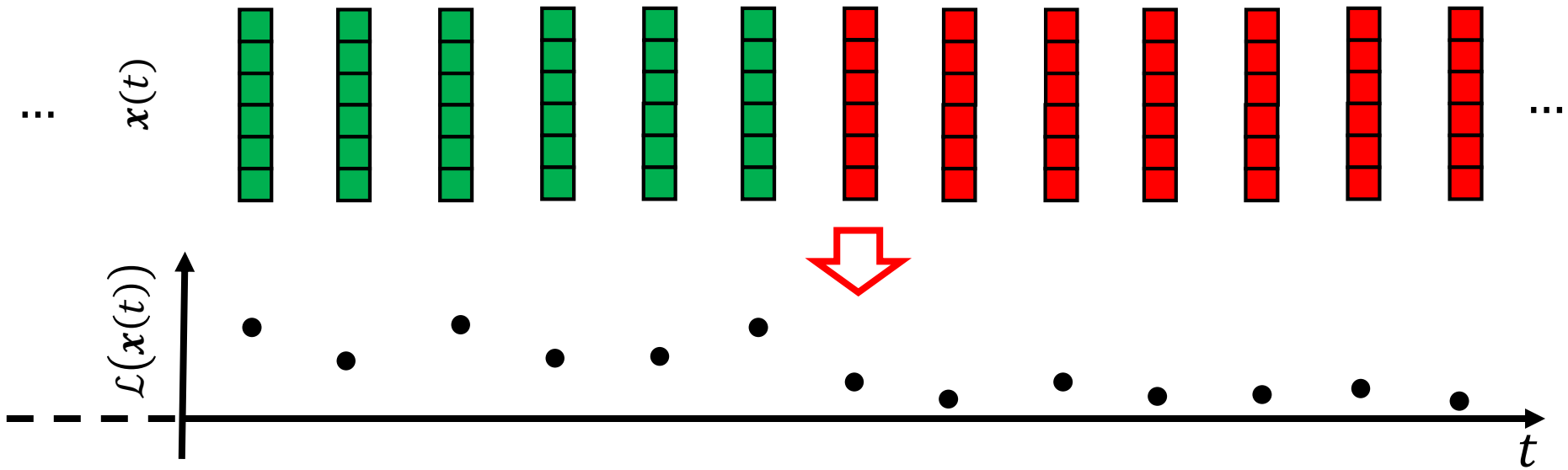
How? Monitoring the Log-likelihood

A typical approach to monitor the log-likelihood

1. During training, estimate $\hat{\phi}_0$ from TR
2. During testing, compute

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

3. Monitor $\{\mathcal{L}\big(\boldsymbol{x}(t)\big),\ t = 1, \ldots\}$



Roadmap to detectability loss

Preliminaries:

- The change-detection approach
- The measure of change detectability
- The change magnitude

The *detectability loss*

- Analytical results
- Empirical analysis

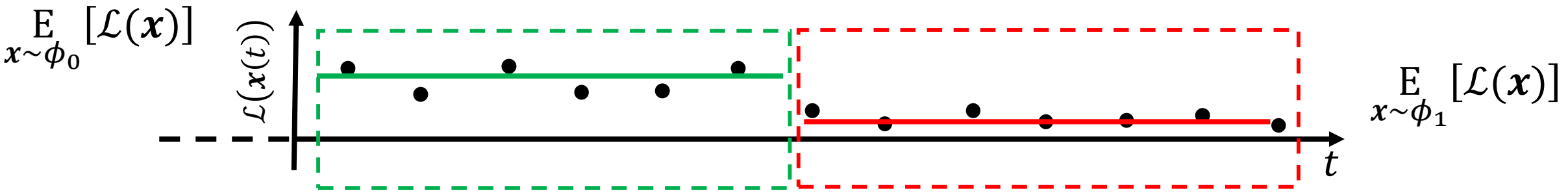
C. Alippi, G. Boracchi, D. Carrera, M. Roveri, "*Change Detection in Multivariate Datastreams: Likelihood and Detectability Loss*" IJCAI 2016,

The Change Detectability

The ***Signal to Noise Ratio of the change***

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

measures the extent to which $\phi_0 \to \phi_1$ is **detectable by statistical tools** designed to **detect changes** in $\mathrm{E}[\mathcal{L}(\boldsymbol{x})]$



Roadmap to detectability loss

Preliminaries:

- The change-detection approach
- The measure of change detectability
- The change magnitude

The *detectability loss*

- Analytical results
- Empirical analysis

The Change Magnitude

We measure the **magnitude of a change** $\phi_0 \rightarrow \phi_1$ by the *symmetric Kullback-Leibler divergence*

$$\mathrm{sKL}(\phi_0, \phi_1) = \mathrm{KL}(\phi_0, \phi_1) + \mathrm{KL}(\phi_1, \phi_0) =$$
$$= \int \log\left(\frac{\phi_0(\boldsymbol{x})}{\phi_1(\boldsymbol{x})}\right)\phi_0(\boldsymbol{x})d\boldsymbol{x} + \int \log\left(\frac{\phi_1(\boldsymbol{x})}{\phi_0(\boldsymbol{x})}\right)\phi_1(\boldsymbol{x})d\boldsymbol{x}$$

In practice, **large values** of $\mathrm{sKL}(\phi_0, \phi_1)$ correspond to **changes** $\phi_0 \rightarrow \phi_1$that are very apparent, since $\mathrm{sKL}(\phi_0, \phi_1)$ identifies an upperbound of the power of hypothesis tests designed to detect either $\phi_0 \rightarrow \phi_1$ or $\phi_1 \rightarrow \phi_0$

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

Our Approach

To study the impact of the **sole data dimension** d in **change-detection problems** we need to:

1. Consider a **change-detection approach**
2. Define a measure of **change detectability** that well correlates with traditional performance measures
3. Define a measure of **change magnitude** that refers only to differences between ϕ_0 and ϕ_1

Our goal (reformulated):
Studying how the **change detectability** $\mathrm{SNR}(\phi_0 \to \phi_1)$ **varies** in **change-detection problems** that have

- **different data dimensions** d
- **constant change magnitude** $\mathrm{sKL}(\phi_0, \phi_1)$

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

Roadmap to detectability loss

Preliminaries:

- The change-detection approach
- The measure of change detectability
- The change magnitude

The *detectability loss*

- Analytical results
- Empirical analysis

The Detectability Loss

Theorem (IJCAI16)

Let $\phi_0 = \mathcal{N}(\mu_0, \Sigma_0)$ *and let* $\phi_1(\boldsymbol{x}) = \phi_0(Q\boldsymbol{x} + \boldsymbol{v})$ *where* $Q \in \mathbb{R}^{d \times d}$ *and orthogonal ,* $\boldsymbol{v} \in \mathbb{R}^d$*, then*

$$\mathrm{SNR}(\phi_0 \to \phi_1) < \frac{C}{d}$$

Where C *is a constant that depends only on* $\mathrm{sKL}(\phi_0, \phi_1)$

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

The Detectability Loss: Remarks

Theorem

Let $\phi_0 = \mathcal{N}(\mu_0, \Sigma_0)$ *and let* $\phi_1(\boldsymbol{x}) = \phi_0(Q\boldsymbol{x} + \boldsymbol{v})$ *where* $Q \in \mathbb{R}^{d \times d}$ *and orthogonal ,* $\boldsymbol{v} \in \mathbb{R}^d$*, then*

$$\mathrm{SNR}(\phi_0 \to \phi_1) < \frac{C}{d}$$

where C *is a constant that depends only on* $\mathrm{sKL}(\phi_0, \phi_1)$

Remarks:

- Changes of a given magnitude, $\mathrm{sKL}(\phi_0, \phi_1)$, become more difficult to detect when d increases
- DL does not depend on the change parameters
- DL does not depend on the specific detection rule
- DL does not depend on estimation errors on $\hat{\phi}_0$

The Detectability Loss: The Change Model

Theorem

Let $\phi_0 = \mathcal{N}(\mu_0, \Sigma_0)$ *and let* $\phi_1(\boldsymbol{x}) = \phi_0(Q\boldsymbol{x} + \boldsymbol{v})$ *where* $Q \in \mathbb{R}^{d \times d}$ *and orthogonal ,* $\boldsymbol{v} \in \mathbb{R}^d$*, then*

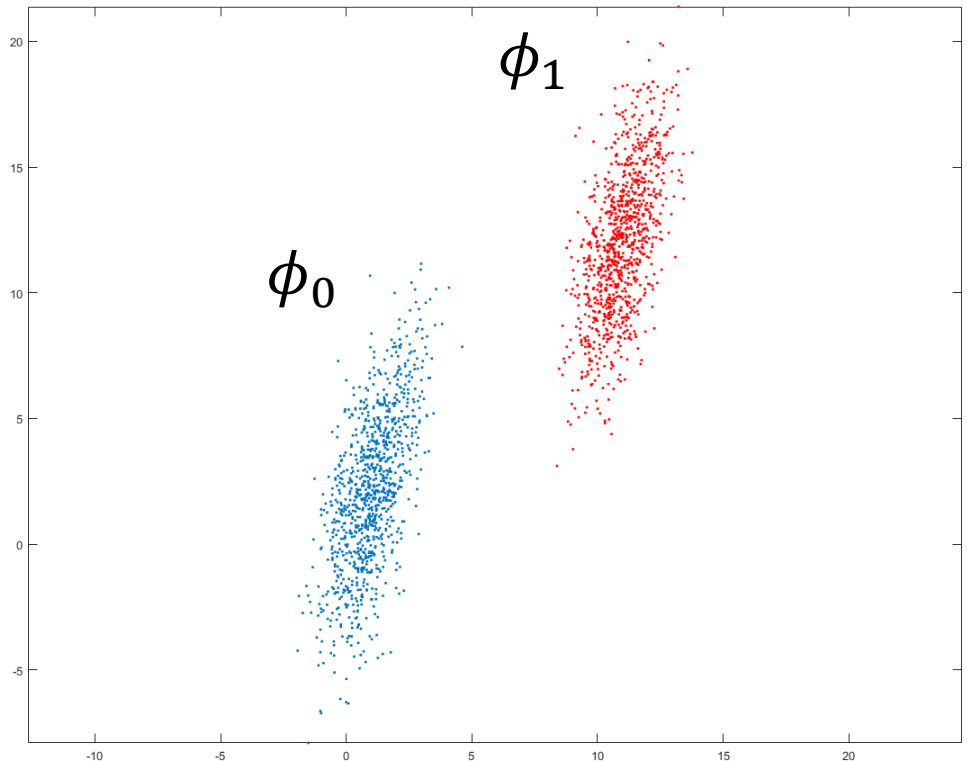
$$\mathrm{SNR}(\phi_0 \to \phi_1) < \frac{C}{d}$$

where C *is a constant that depends only on* $\mathrm{sKL}(\phi_0, \phi_1)$

The Detectability Loss: The Change Model

The change model $\phi_1(\boldsymbol{x}) = \phi_0(Q\boldsymbol{x} + \boldsymbol{v})$ includes:

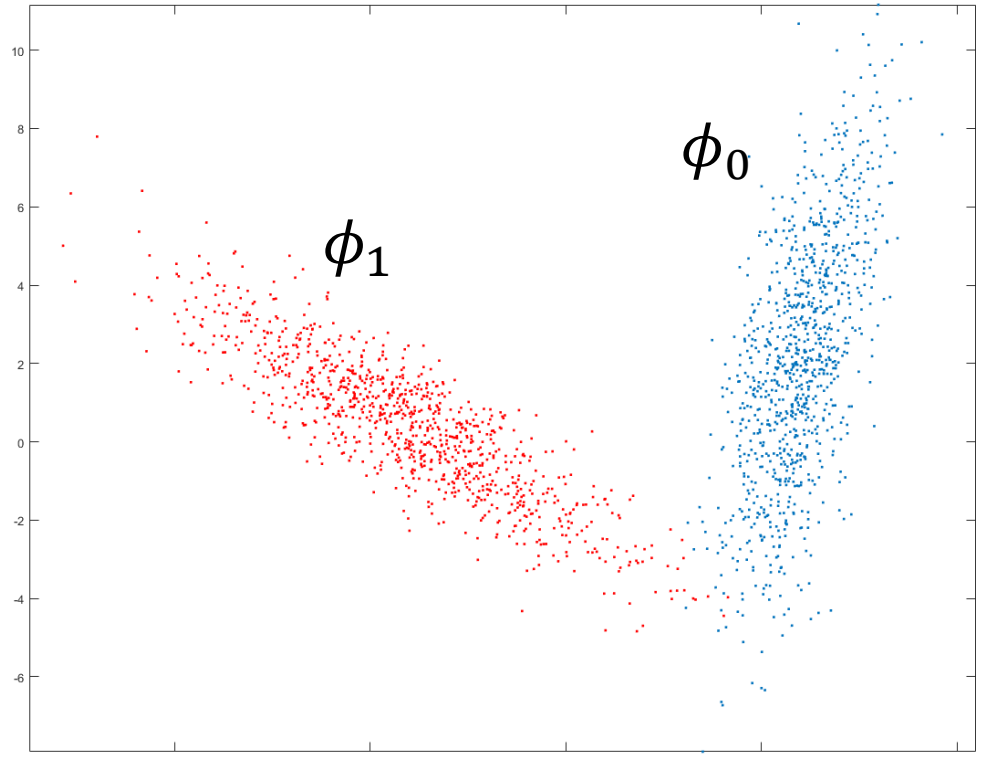
- Changes in the location of ϕ_0 (i.e, $+\boldsymbol{v}$)



The Detectability Loss: The Change Model

The change model $\phi_1(\boldsymbol{x}) = \phi_0(Q\boldsymbol{x} + \boldsymbol{v})$ includes:

- Changes in the location of ϕ_0 (i.e, $+\boldsymbol{v}$)
- Changes in the correlation of $\boldsymbol{x}$ (i.e, $Q\boldsymbol{x}$)



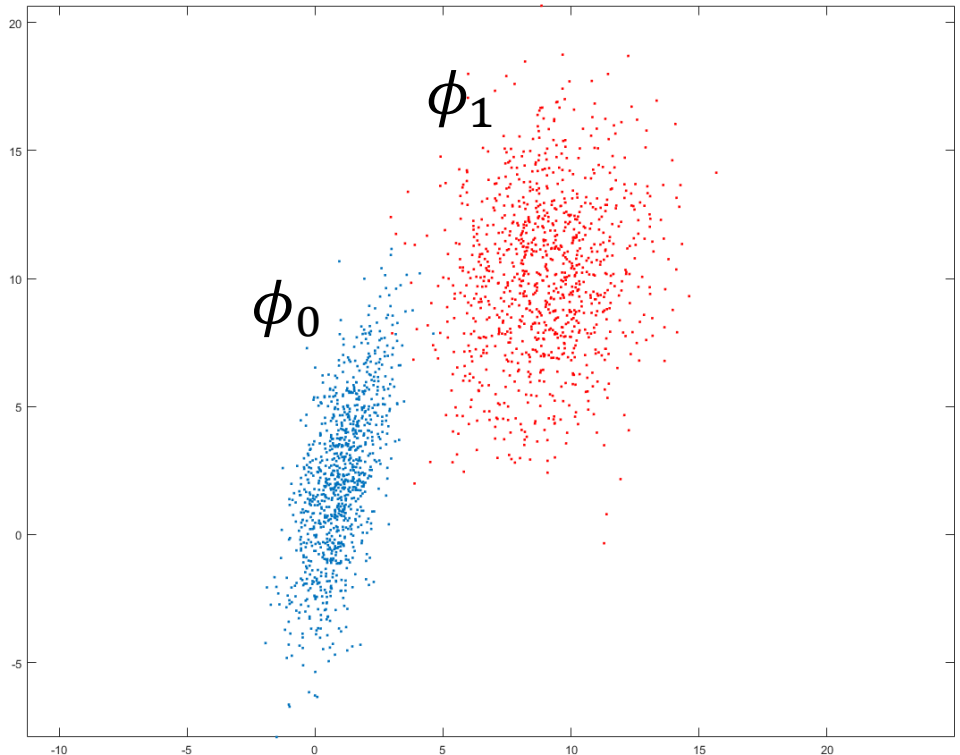
Boracchi, Trovò

The Detectability Loss: The Change Model

The change model $\phi_1(\boldsymbol{x}) = \phi_0(Q\boldsymbol{x} + \boldsymbol{v})$ includes:

- Changes in the location of ϕ_0 (i.e, $+\boldsymbol{v}$)
- Changes in the correlation of $\boldsymbol{x}$ (i.e, $Q\boldsymbol{x}$)

It does not include changes in the scale of ϕ_0 that can be however detected monitoring $||\boldsymbol{x}||$



Boracchi, Trovò

The Detectability Loss: The Gaussian Assumption

Theorem

Let $\phi_0 = \mathcal{N}(\mu_0, \Sigma_0)$ *and let* $\phi_1(\boldsymbol{x}) = \phi_0(Q\boldsymbol{x} + \boldsymbol{v})$ *where* $Q \in \mathbb{R}^{d \times d}$ *and orthogonal ,* $\boldsymbol{v} \in \mathbb{R}^d$*, then*

$$\mathrm{SNR}(\phi_0 \to \phi_1) < \frac{C}{d}$$

where C *is a constant that depends only on* $\mathrm{sKL}(\phi_0, \phi_1)$

The Detectability Loss: The Gaussian Assumption

Assuming $\phi_0 = \mathcal{N}(\mu_0, \Sigma_0)$ looks like a severe limitation.

- Other distributions are not easy to handle analytically
- We can prove that DL occurs **also in random variables having independent components**
- The result have been **empirically confirmed** in case of approximations of $\mathcal{L}(\cdot)$ typically used **for Gaussian mixtures**

Roadmap to detectability loss

Preliminaries:

- The change-detection approach
- The measure of change detectability
- The change magnitude

The *detectability loss*

- Analytical results
- Empirical analysis

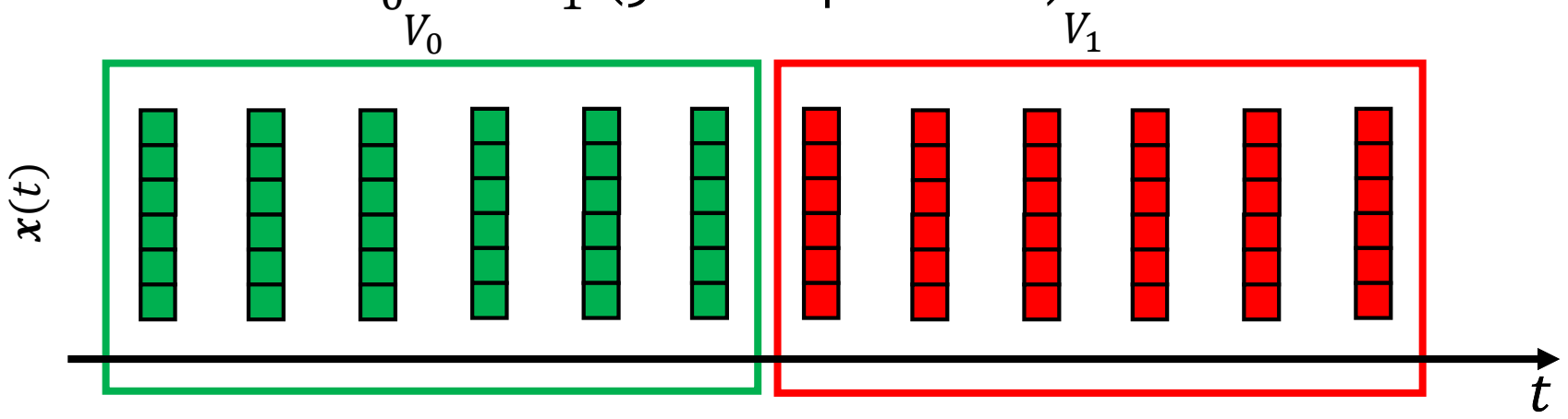
The Detectability Loss: Empirical Analysis

The data

- Synthetically generate streams with different dimensions d
- Estimate $\hat{\phi}_0$ by GM from a **stationary training set**
- In each stream we introduce $\phi_0 \to \phi_1$ such that

$$\phi_1(\boldsymbol{x}) = \phi_0(Q\boldsymbol{x} + \boldsymbol{v}) \text{ and } \mathrm{sKL}(\phi_0, \phi_1) = 1$$

- **Test data: two windows** V_0 and V_1 (500 samples each) before and after the change.



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

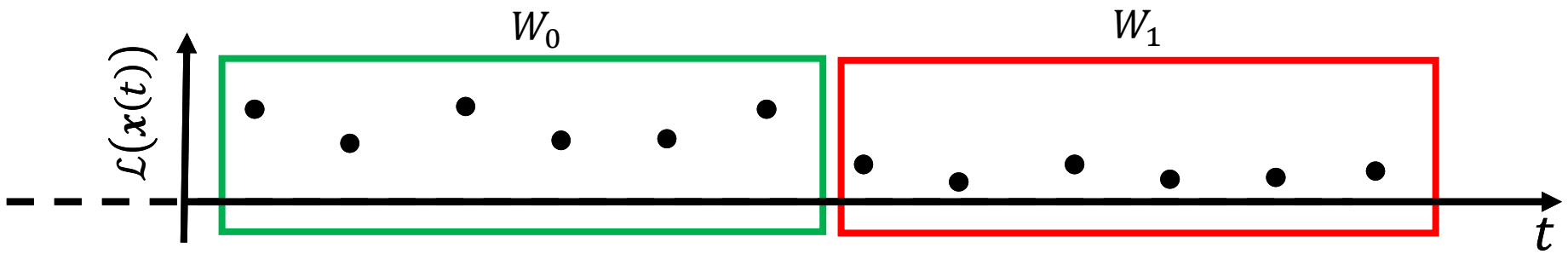
The Detectability Loss: Empirical Analysis

We measure the change-detectabiltity as:

- Compute $\mathcal{L}\left(\hat{\phi}_0(\boldsymbol{x})\right)$ from V_0 and V_1, obtaining W_0 and W_1
- **Compute a test statistic** $\mathcal{T}(W_0, W_1)$ to compare the two
- Detect a change by an **hypothesis test**

$$\mathcal{T}(W_0, W_1) \lessgtr h$$

  where h controls the amount of false positives
- Use the **power** of this **test** to assess change detectability



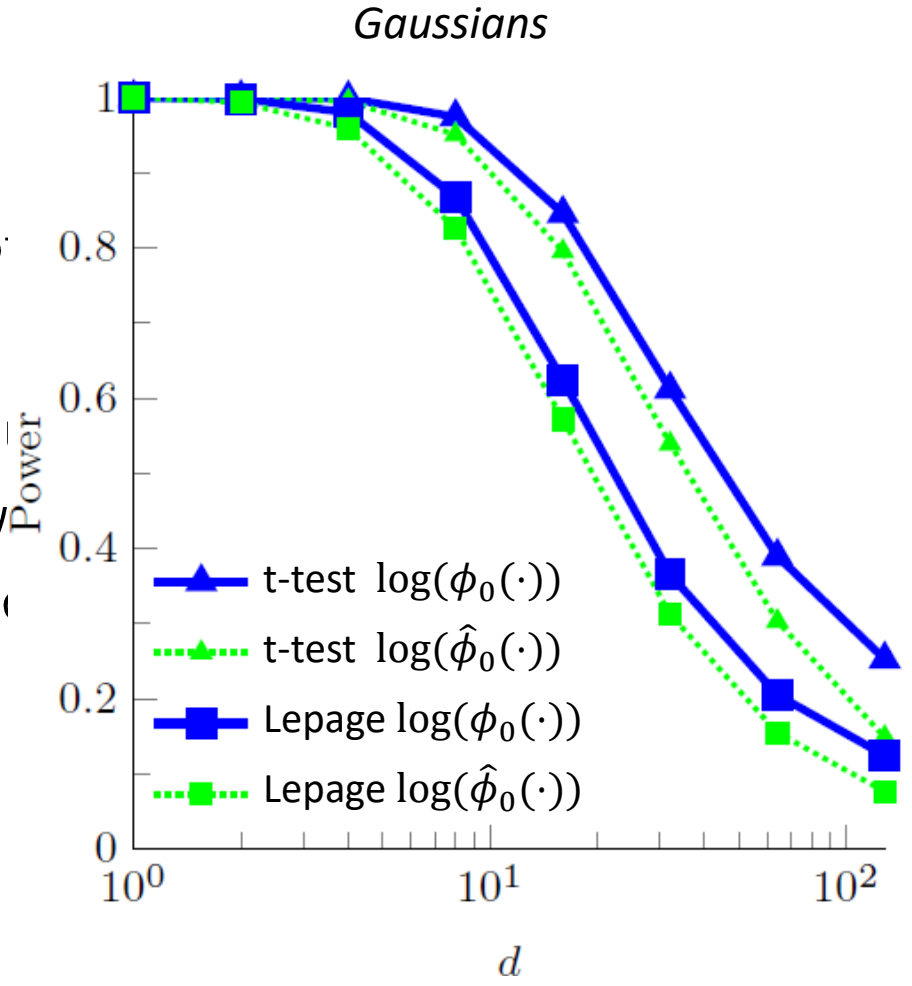
Power of Hypothesis Tests on Gaussian Streams

Remarks:

- ϕ_1 is defined analytically
- The t-test detects changes in the expectation of $\log(\phi_0(\cdot))$
- The Lepage test detects changes in the location and scale o

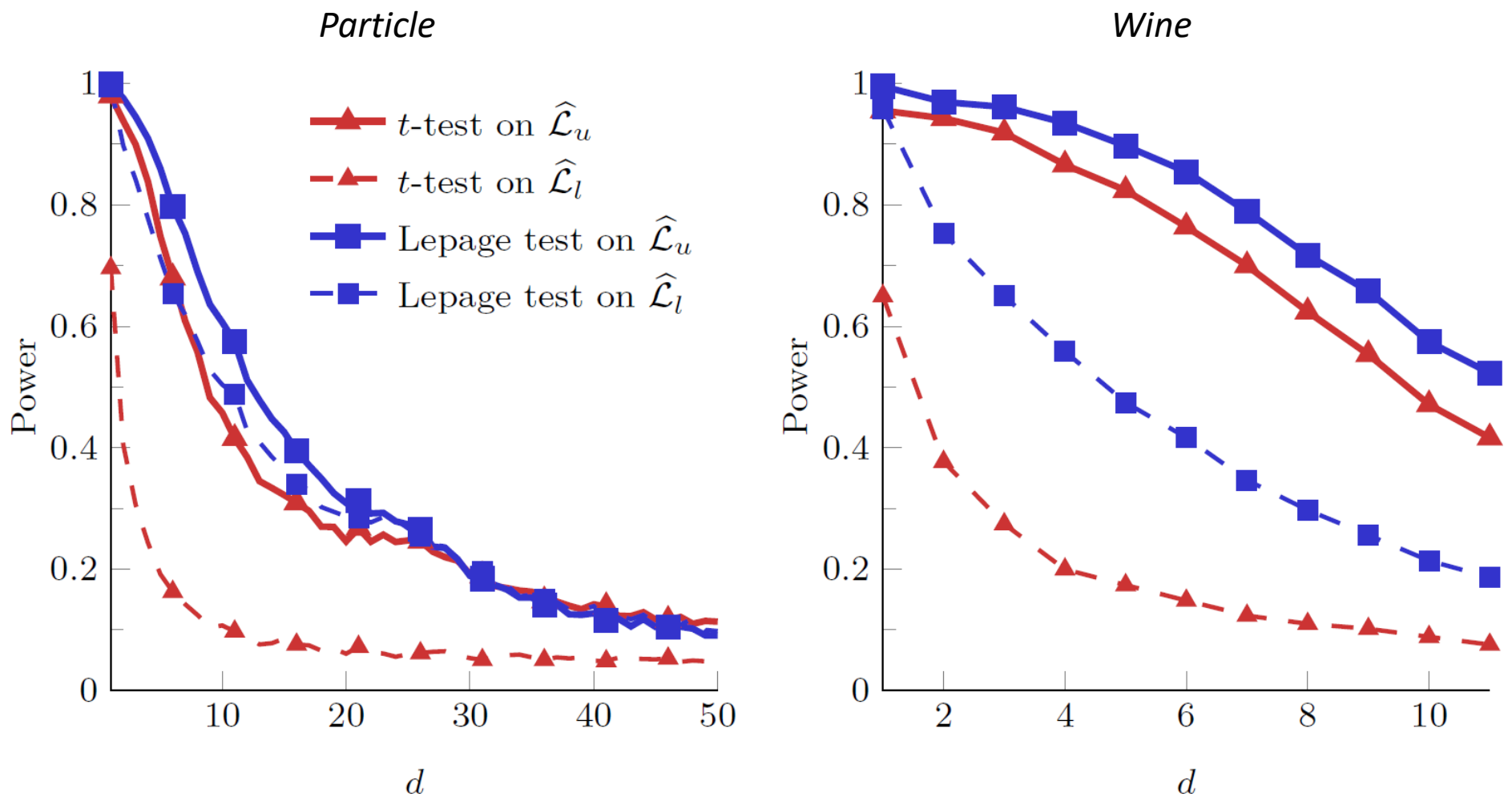
Results

- The HT power decays with d: DL does not only concern the
- DL is not due to estimation errors, but these make things w
- Also the power of the Lepage HT decreases, which indicat detect even when monitoring the variance



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

Power of Hypothesis Tests on UCI datasets



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

The Power on Particle Dataset

Remarks:

- ϕ_1 is defined through CCM a framework to control $\mathrm{sKL}(\phi_0, \phi_1) \approx 1$
- $\hat{\phi}_0$ is a Gaussian Mixture where k is selected by cross-valida
- Approximated expression of $\mathcal{L}(\cdot)$ to prevent numerical appro

Results:

- DL occurs also in non-Gaussian data approximated by GM
- DL is clearly visible at quite a low dimensions