Online Learning And Monitoring

Giacomo Boracchi, Francesco Trovò

April 20th, 2022

Politecnico di Milano, DEIB

giacomo.boracchi@polimi.it

Today Outline

- Practical Information and Course Logistics
- The General Picture
- An Illustrative Example: Fraud Detection
- Problem Formulation Learning in NSE
- Setting up the Stage for Learning in NSE
- First Assignment
- Controlling False Alarms
- Concept Drift Detection monitoring classification error
  - DDM
  - EWMA

Practical Information:
Click on Google Calendar Link

Practical Information: Schedule

Lecture Dates:

- *Intro and Learning in NSE: Monitoring*. 20th April 2022 - 14.15 to 18.00 Sala Conferenze, DEIB Edificio 20. Giacomo Boracchi
- *Learning in NSE: Adaptation*. 27th April 2022 - 14.15 to 18.00, Sala Conferenze, DEIB Edificio 20. Giacomo Boracchi
- *Learning with Experts*. 4th May 2022 - 14.15 to 18.00, Sala Conferenze, DEIB Edificio 20. Francesco Trovò
- *Learning with MAB*. 11th May 2022 - 14.15 to 18.00, Room TDB. Francesco Trovò
- *Anomaly Detection and Domain Adaptation*. 18th May 2022 - 14.15 to 18.00, Sala Conferenze, DEIB Edificio 20. Giacomo Boracchi
- *Applications*. 25th May 2022 - 14.15 to 18.00, Sala Conferenze, DEIB Edificio 20. Giacomo Boracchi

Online Streaming in the Webex Room of the speaker

https://politecnicomilano.webex.com/meet/giacomo.boracchi or https://politecnicomilano.webex.com/meet/francesco1.trovo)

Practical Information: Teaching Modality

Lectures will be held in presence, and streamed via the presenter webex room.

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

- During Lecture, ask questions, share your doubts
- During Lecture, answer to our questions
- During the «lab» part of the lecture… be proactive… these are very helpful to make sure you understood everything

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 entire course materials
- 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 and Diego Carrera *Tutorial at ICPR 2020*
https://boracchi.faculty.polimi.it/Tutorials/AnomalyAndChangeDetectionTutorial_ICPR2020.html

Anomaly Detection in Images, Giacomo Boracchi and Diego Carrera, *Tutorial at ICIP 2020*
https://boracchi.faculty.polimi.it/Tutorials/AnomalyDetectionInImagesTutorial.html

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*

https://boracchi.faculty.polimi.it/docs/2015_12_LNSE_tutorial_SSCI2015_Part1_Boracchi.pdf
https://boracchi.faculty.polimi.it/docs/2015_12_LNSE_tutorial_SSCI2015_Part2_Ditzler.pdf

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

Questions?

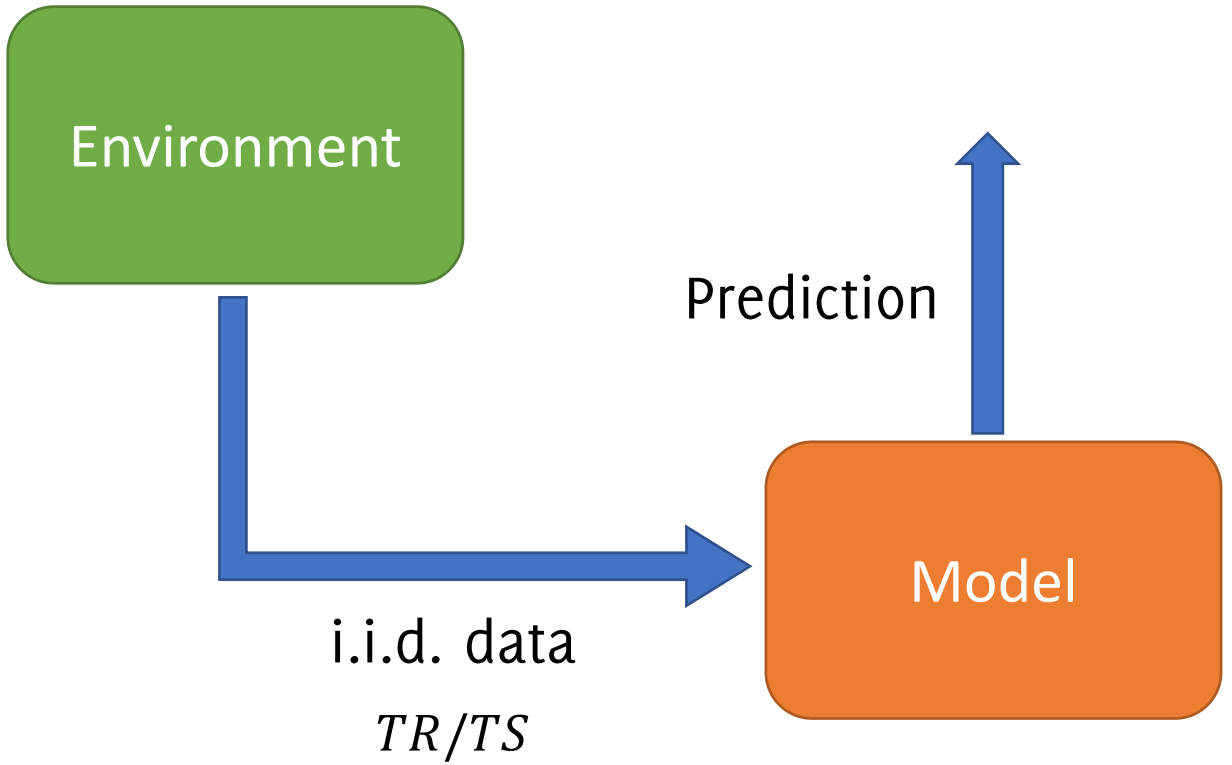
The General Picture

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*

Rmk: Predictions does not influence the environment, nor TR/TS



The major focus is towards making data-driven models able to extract information out of training data (TR) to perform inference on test data (TS)

This Course Framework

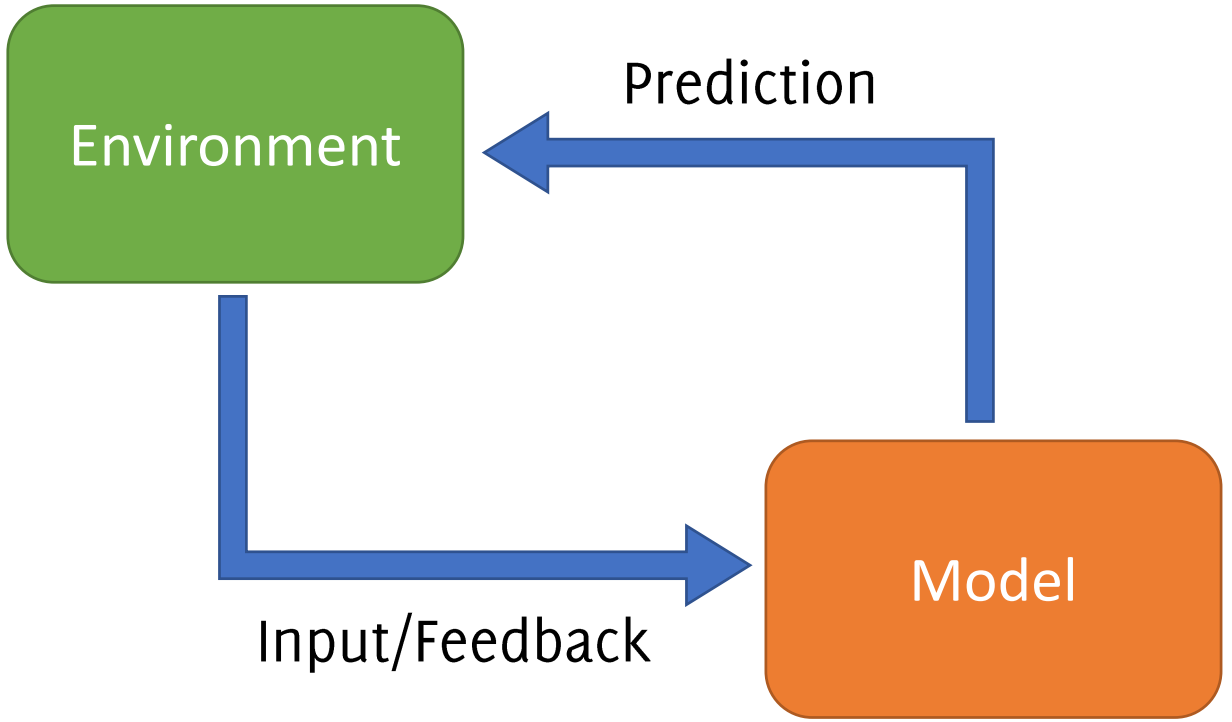
In a **streaming** scenario i.i.d. assumption often does not hold:

- the **environment** might be **changing or adversarial**
- It is not possible to ignore the **model-environment interactions,** since model outcomes are influencing the environment or the supervision provided (feedback)

These settings call for:

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

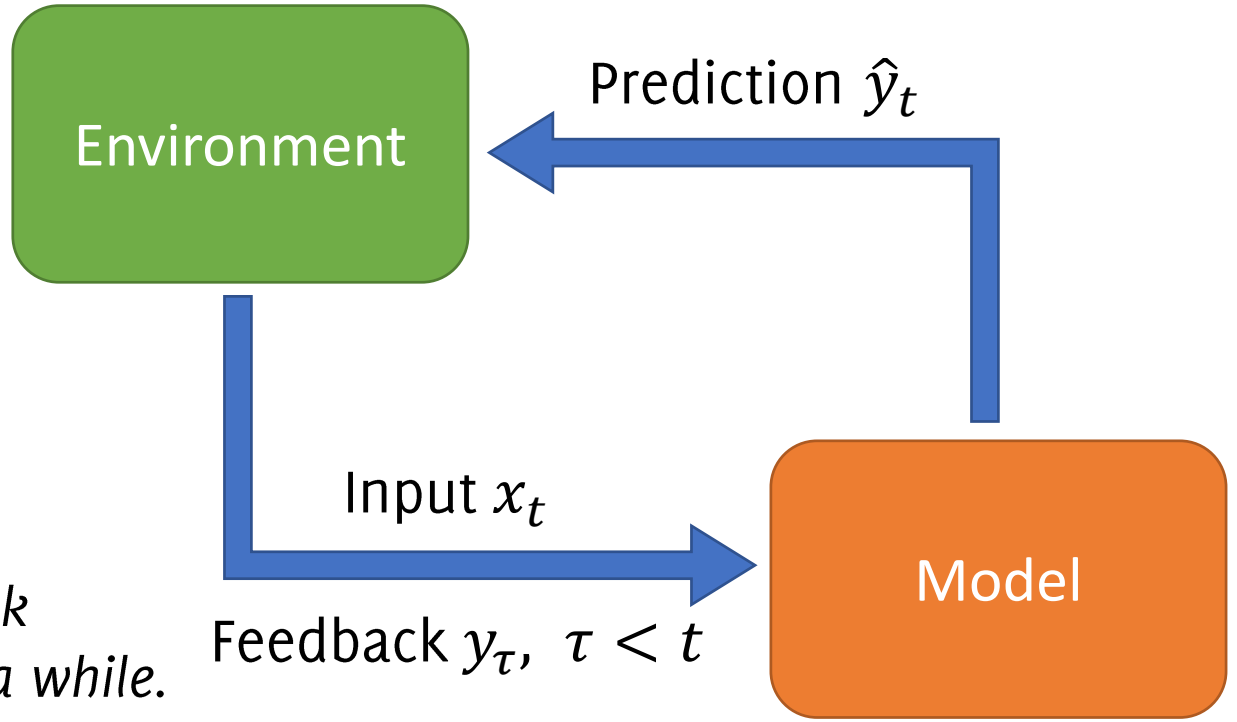
Rmk: Predictions might influence the environment, and/or TR/TS



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



Example: **Fraud Detection**

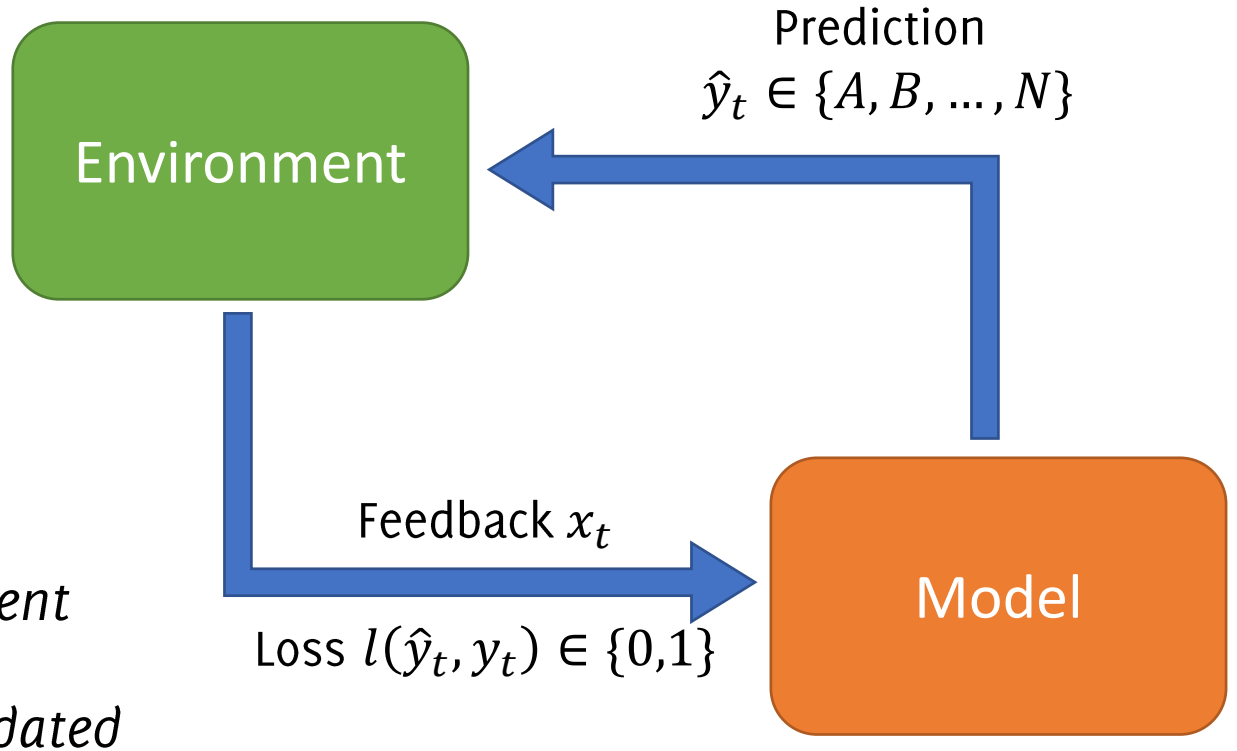
You classify each transaction x_t assigning a label $\hat{y}_t$(genuine/fraudulent), investigators check only those labels and return a feedback y_t after a while. Feedbacks are not representative of the entire stream, and possibly delayed

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 $l(\hat{y}_t, y_t)$
- the model gets feedback x_t as additional information
- the model is updated based on $l(\hat{y}_t, y_t)$ and x_t

Rmk: Model has to enable
- fast and frequent update
- Operation with limited storage



Example: **Clinical Trials**

You chose which treatment $\hat{y}_t$ to assign to a patient x_{t-1} (the environment). The environment replies positively or negatively $l(\hat{y}_t, y_t)$, the model is updated according to this feedback to define treatment to next patient x_t

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 provide theoretical guarantees. Major emphasis on the application.
- **Online learning: formal approach to the problem**, requiring strong theoretical requirements on the algorithms (focus on the analysis). Using these models in real-life requires substantial adjustments and approximations.

This is the second edition of the course. Still, 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 (Matlab Assignment)

Monitoring

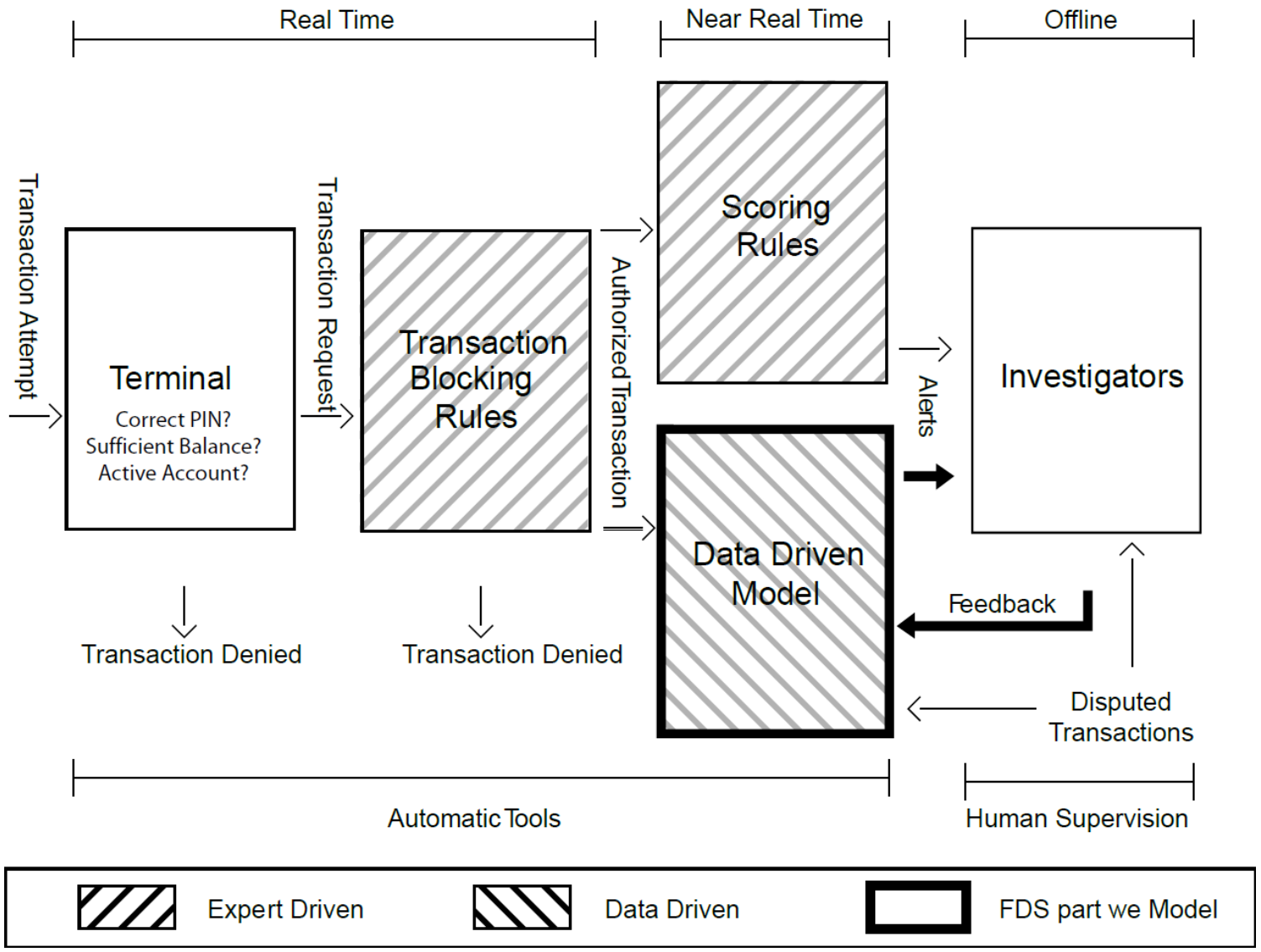
- Change Detection Test on the classification error (Lab session)
- Change Detection test on the input distribution (Lab session)

Fraud Detection

A Cool Example for Learning in NSE

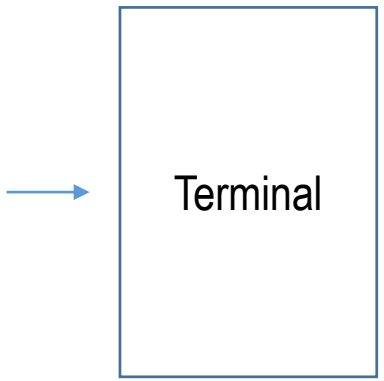
Boracchi, Trovò

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

The Terminal

Acceptance checks like:

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

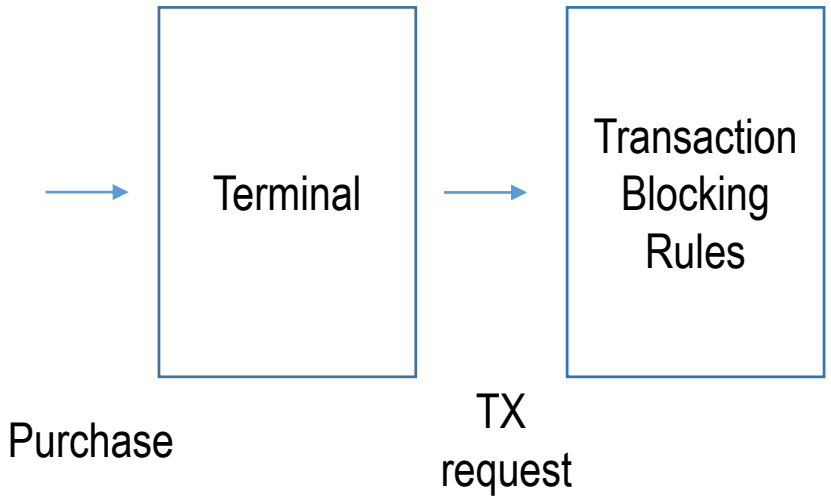
are immediately performed.

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



Transaction Blocking Rules

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

IF Internet transactions AND compromised website THEN deny the transaction

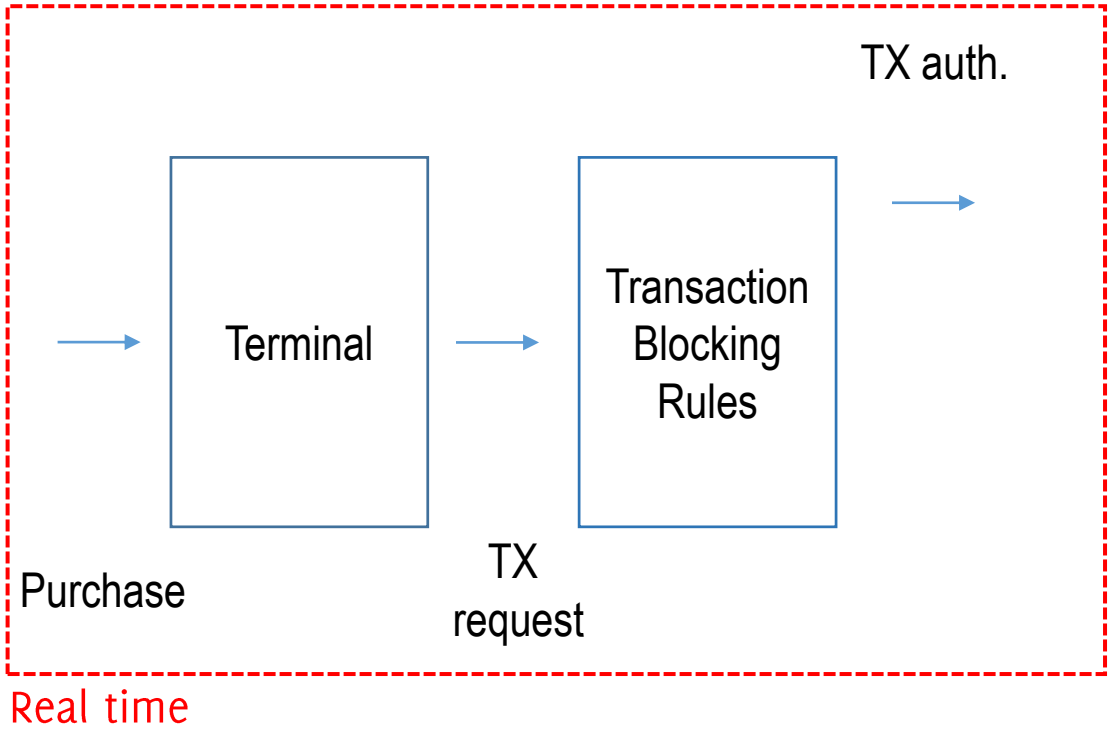
These rules:

- 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 (otherwise the transaction is accepted)
- 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

Near Real Time Processing



Feature Augmentation

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

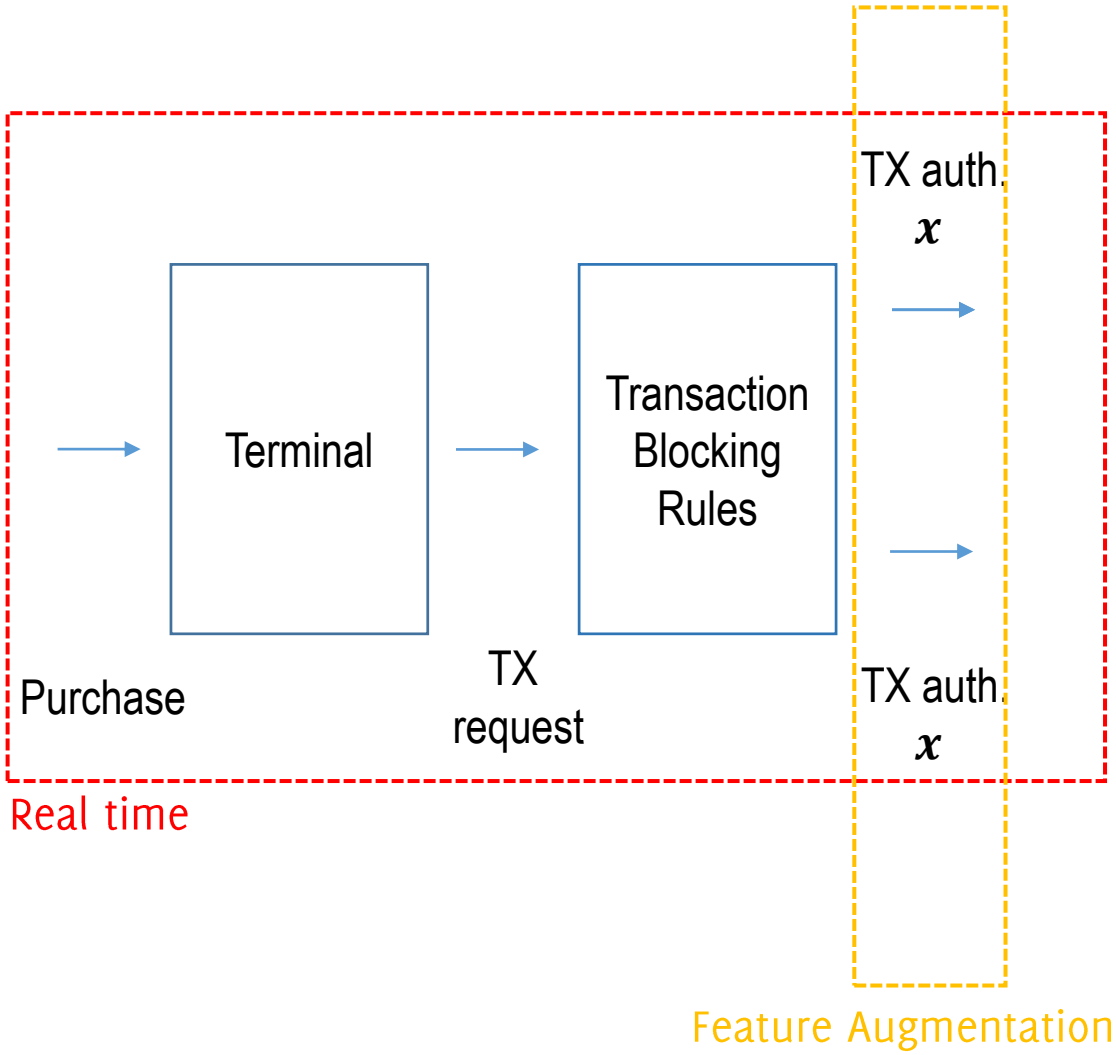
The components of $\boldsymbol{x}$ include data about the current transaction and customary shopping habits of the cardholder, e.g.:

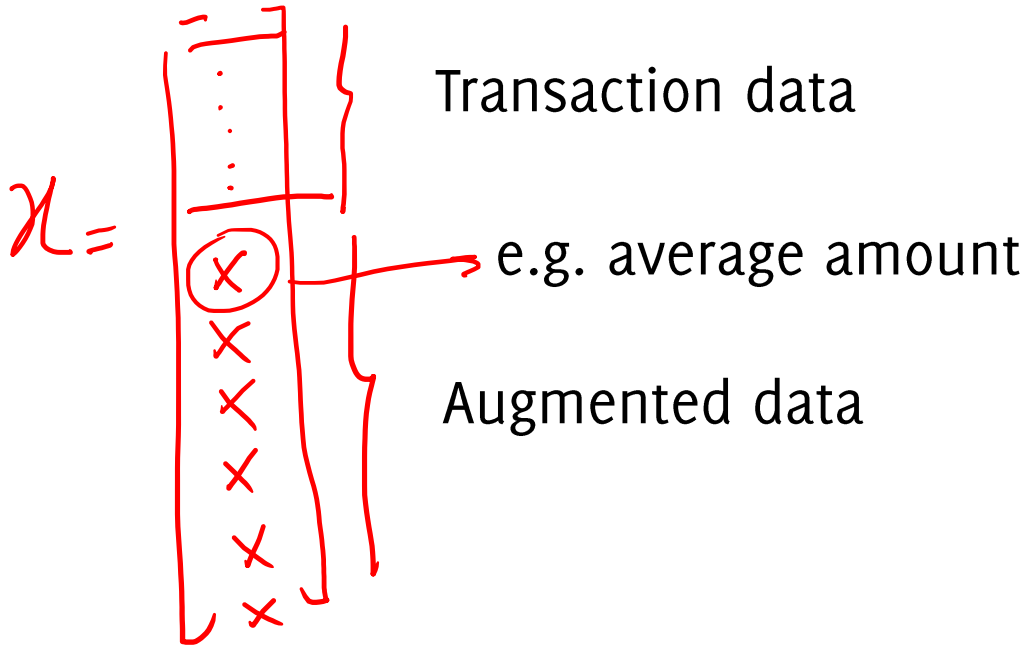
- the average expenditure
- the average number of transactions per day
- the cardholder age
- the location of the last purchases
- …

and are **very informative** for fraud-detection purposes

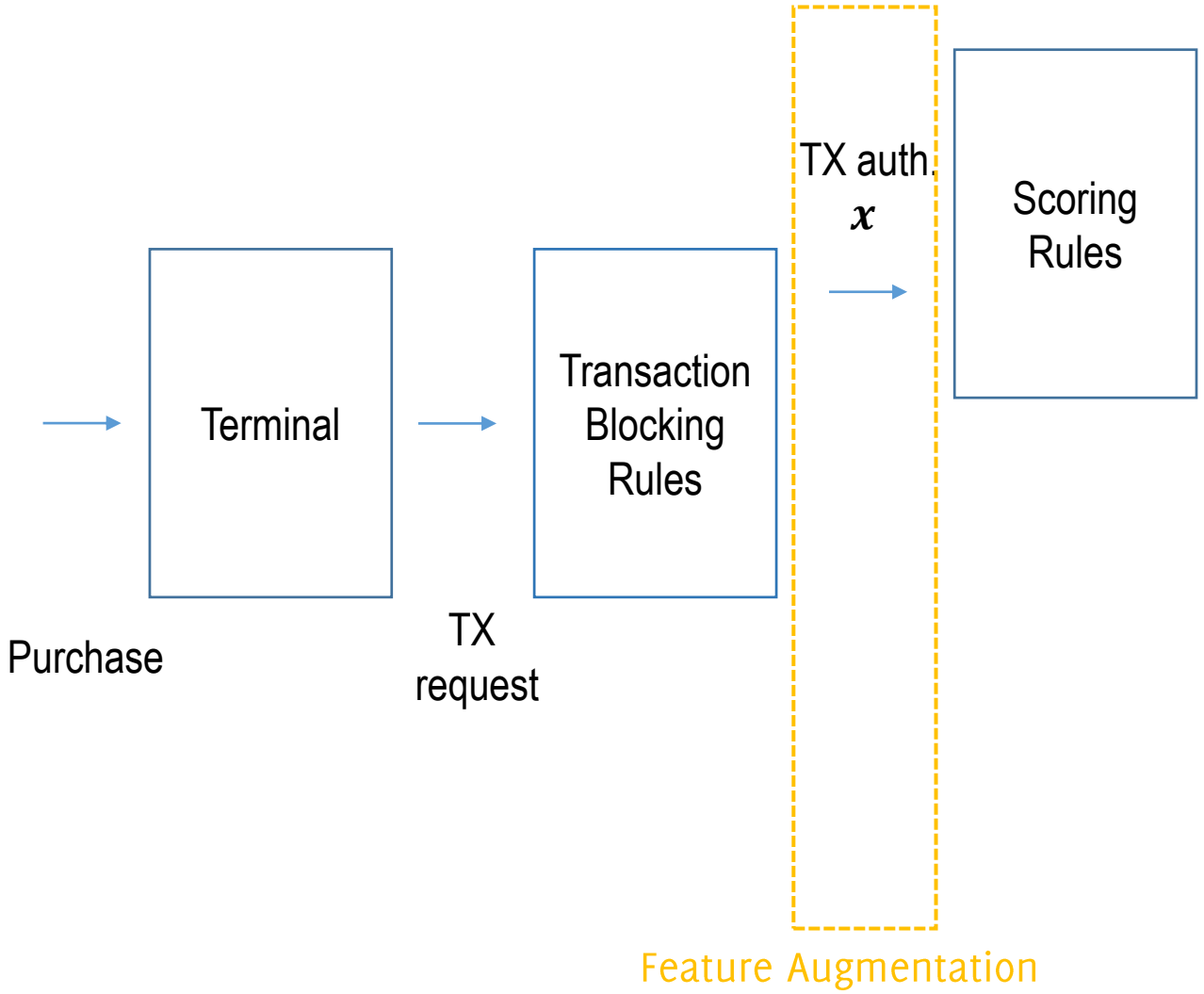
Overall, about 40 features are extracted in near-real time.

Near Real Time Processing





Scoring Rules



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

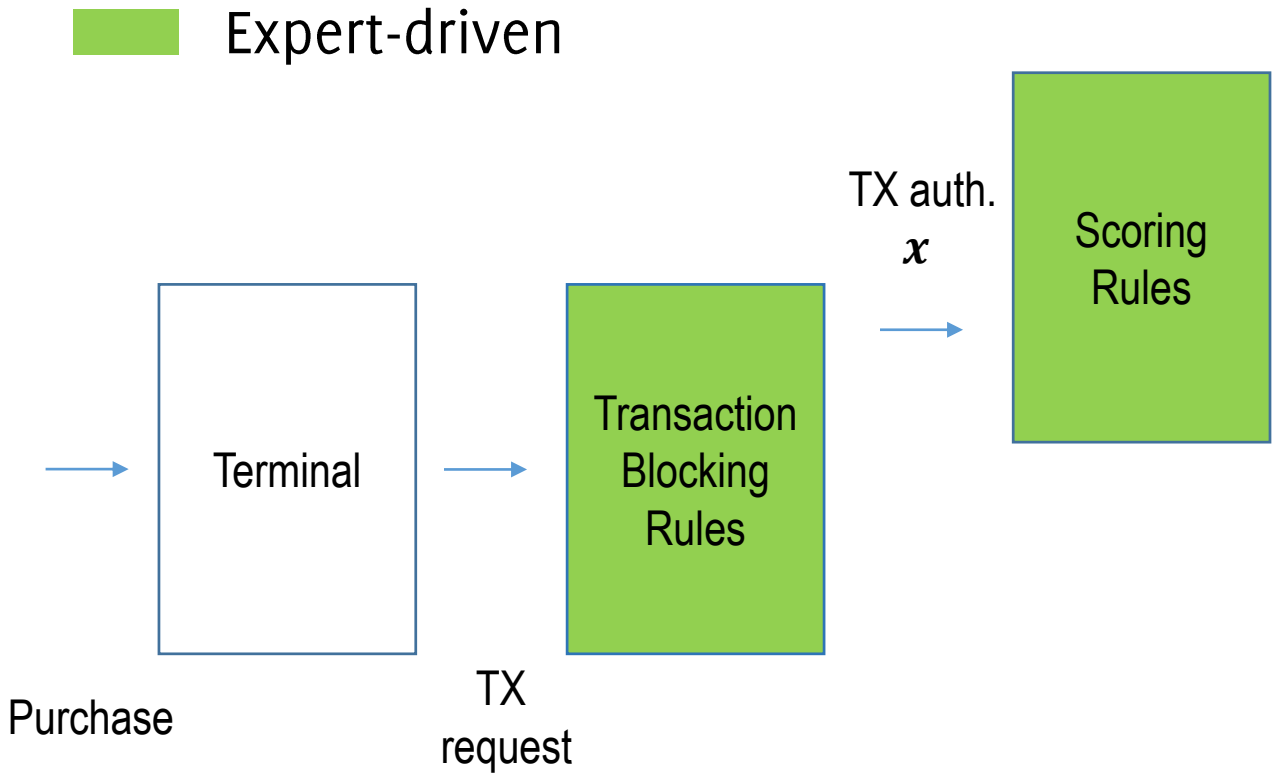
Scoring Rules

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 rules

Allows interaction

Expert-Driven vs Data-Driven models

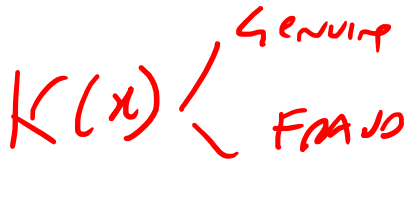
Scoring rules are an **expert-driven model**, thus:

- 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



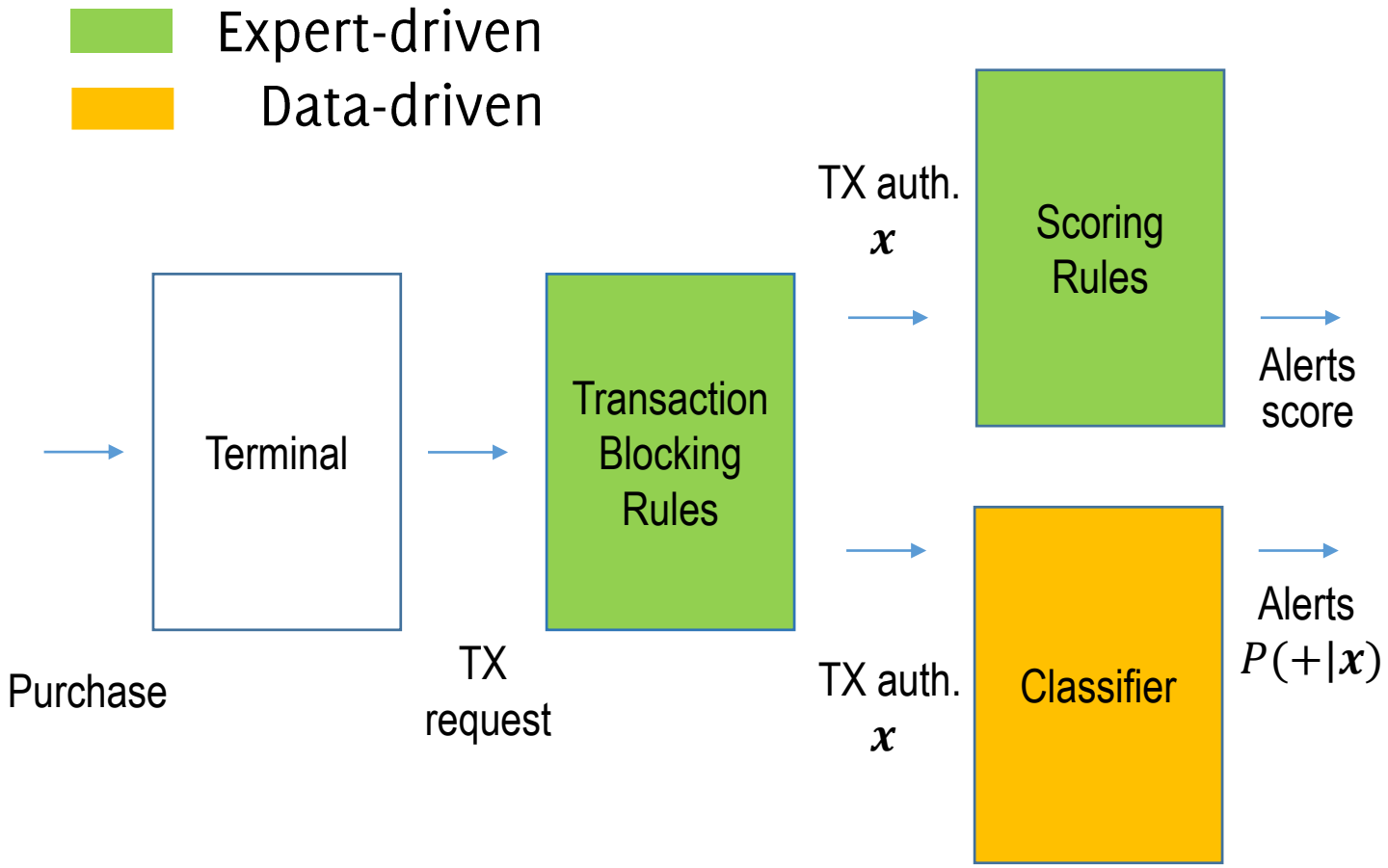
Fraudulent patterns can be directly **learned from data**, by means of a **data-driven model**.

This has the potential to:

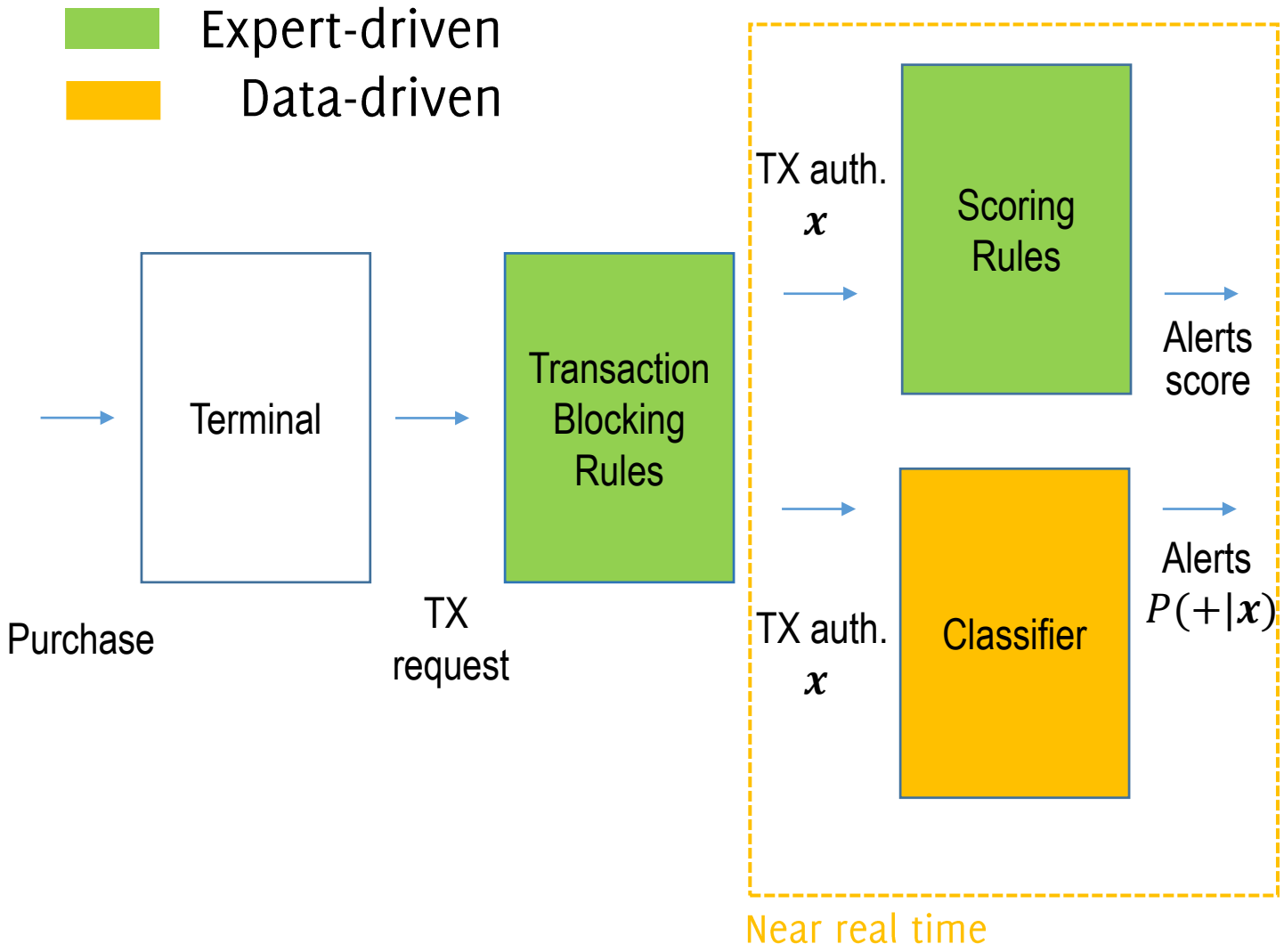
- Simultaneously analyze **several components** of the feature vector
- Uncover **complex relations among features** that cannot be identified by investigator

These relations can be meaningful for separating frauds from genuine transactions

Data-driven models in fraud detection

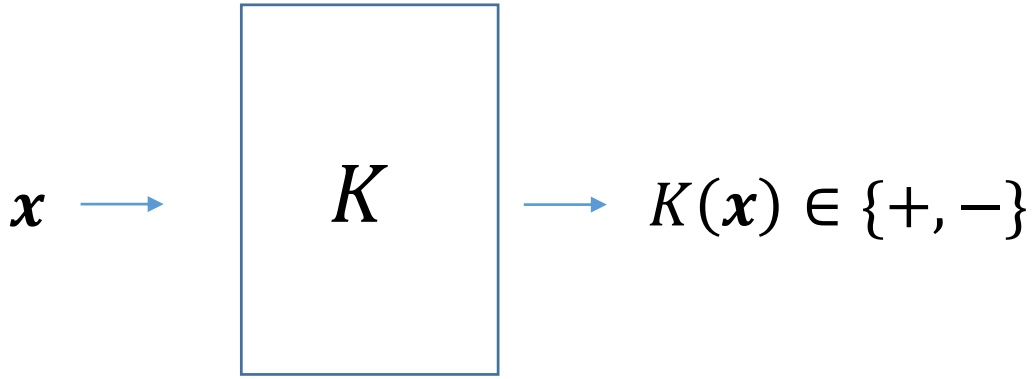


Data-driven models in fraud detection



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**

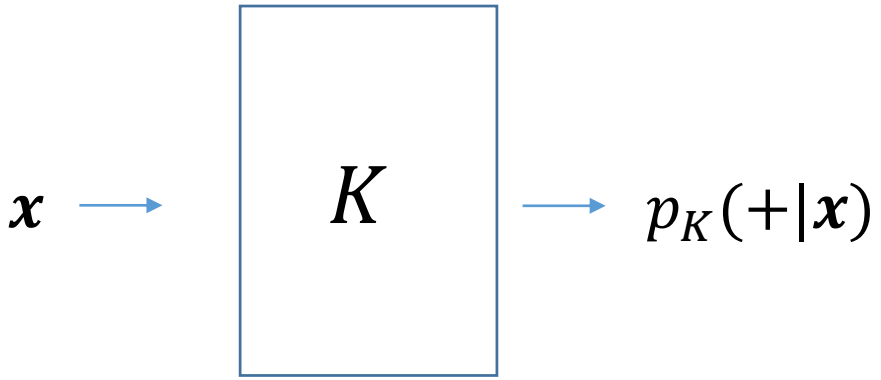
Classifiers in Fraud Detection

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

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

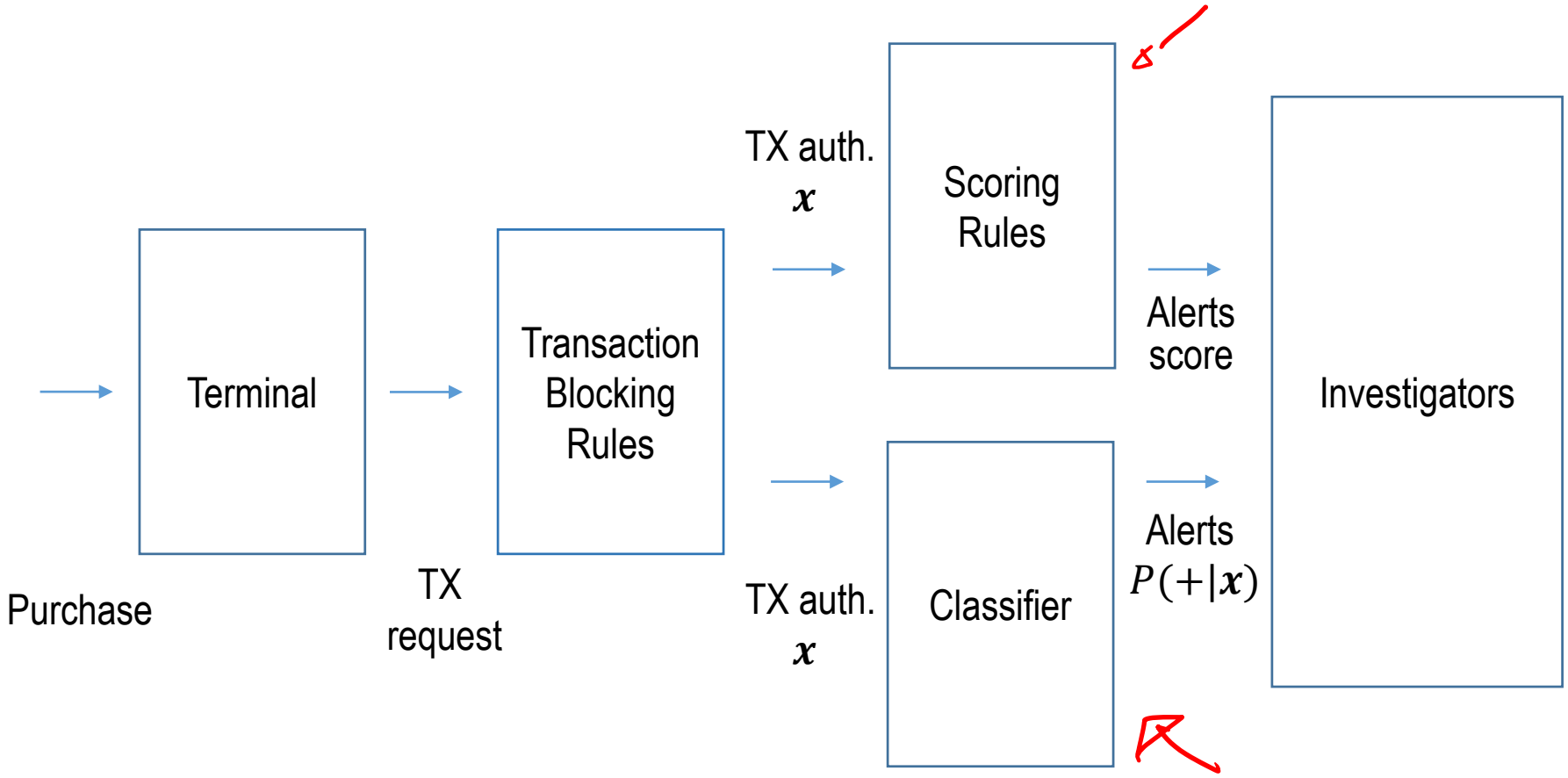
Thus, the FDS typically consider $p_K(+|\boldsymbol{x})$, **an estimate of the probability** for $\boldsymbol{x}$ to be a fraud according to K

Since this is a binary classification problem $p_K(-|\boldsymbol{x}) = 1 - p_K(+|\boldsymbol{x})$



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

Investigators Provide Feedbacks



Investigators

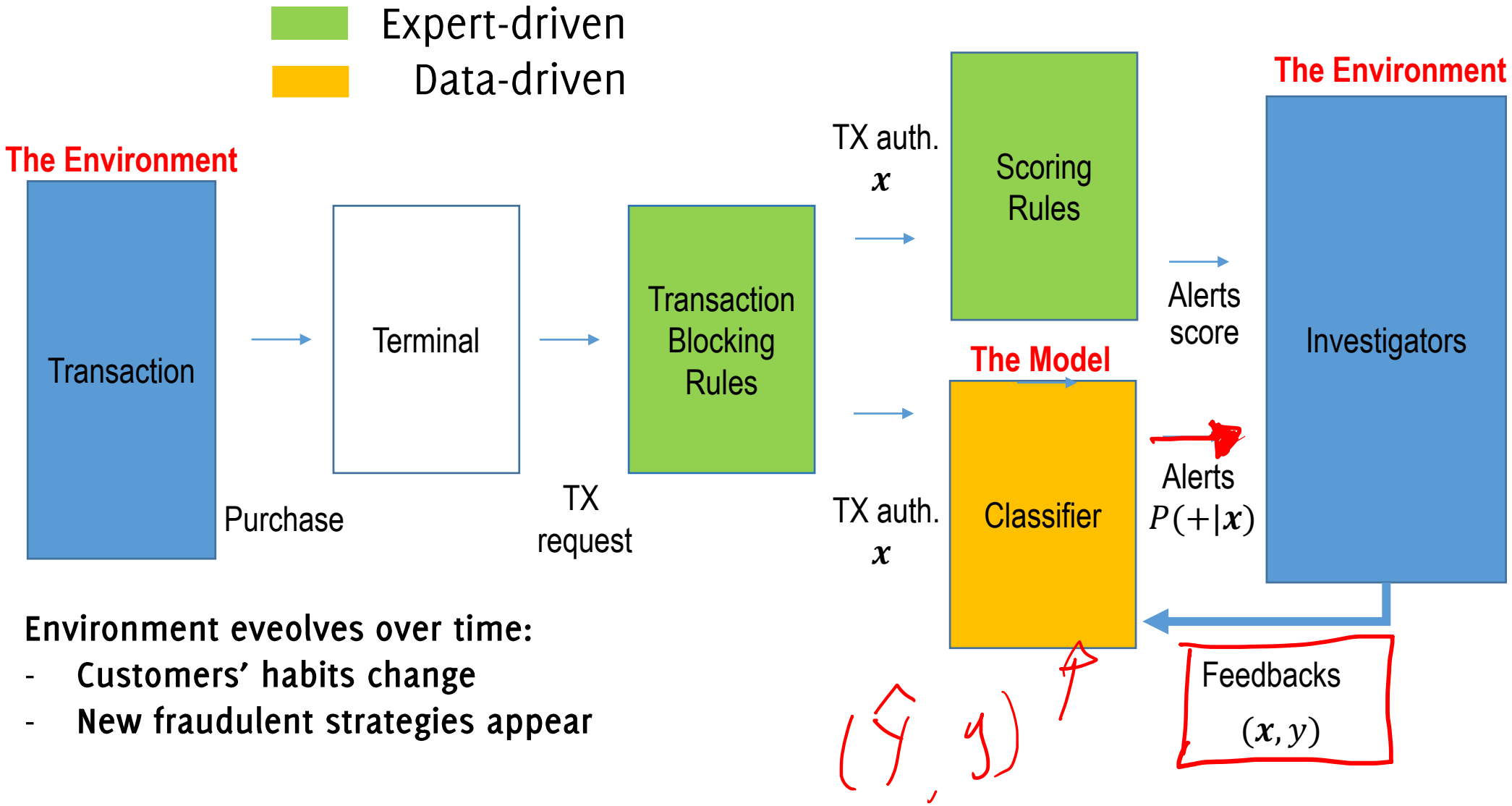
Investigators are **professionals** that are experienced in analyzing credit card transactions:

- 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

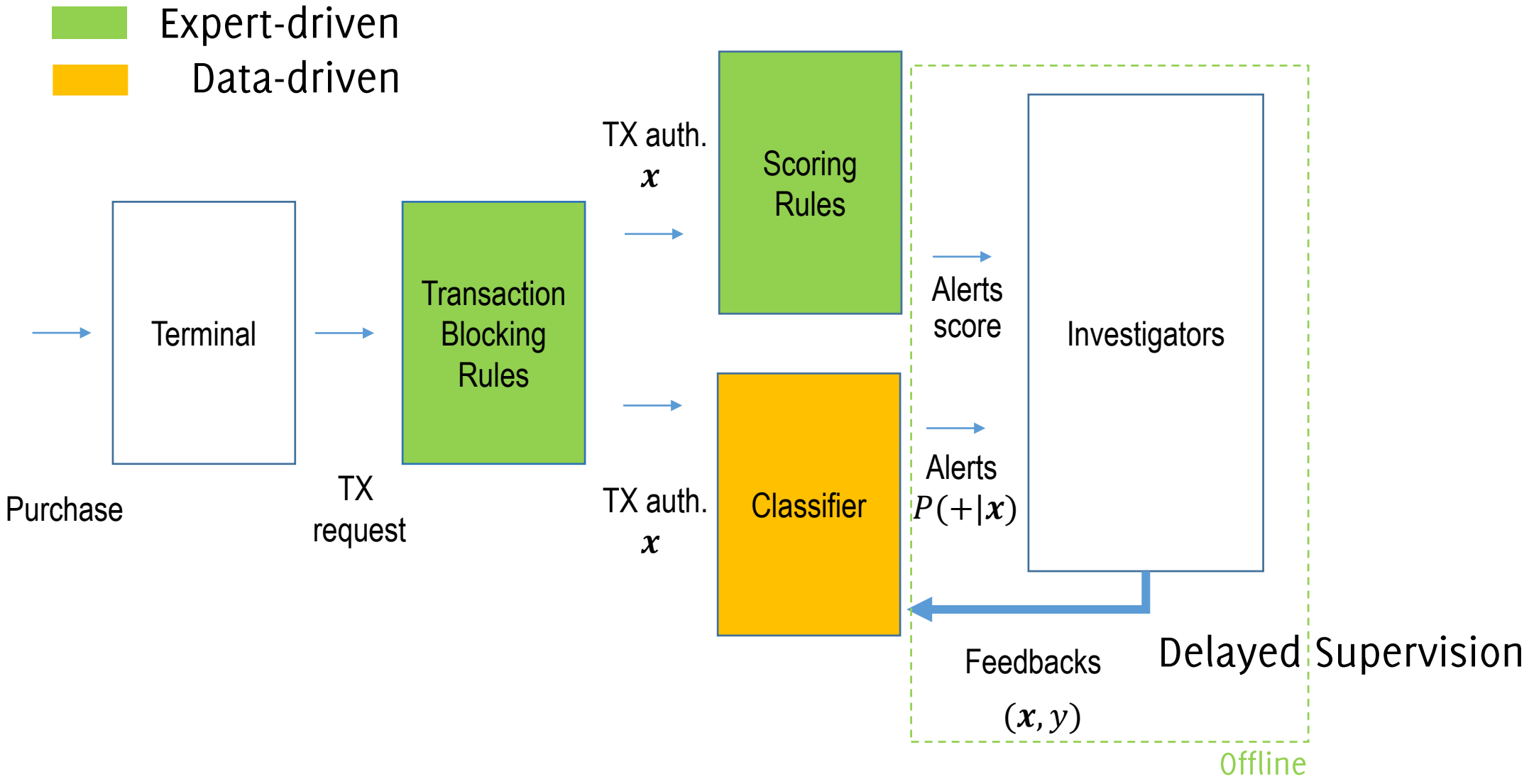


Environment eveolves over time:

- **Customers' habits change**
- **New fraudulent strategies appear**

ovò

Investigators' feedback: Supervised Information



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 $\hat{y} = y$.

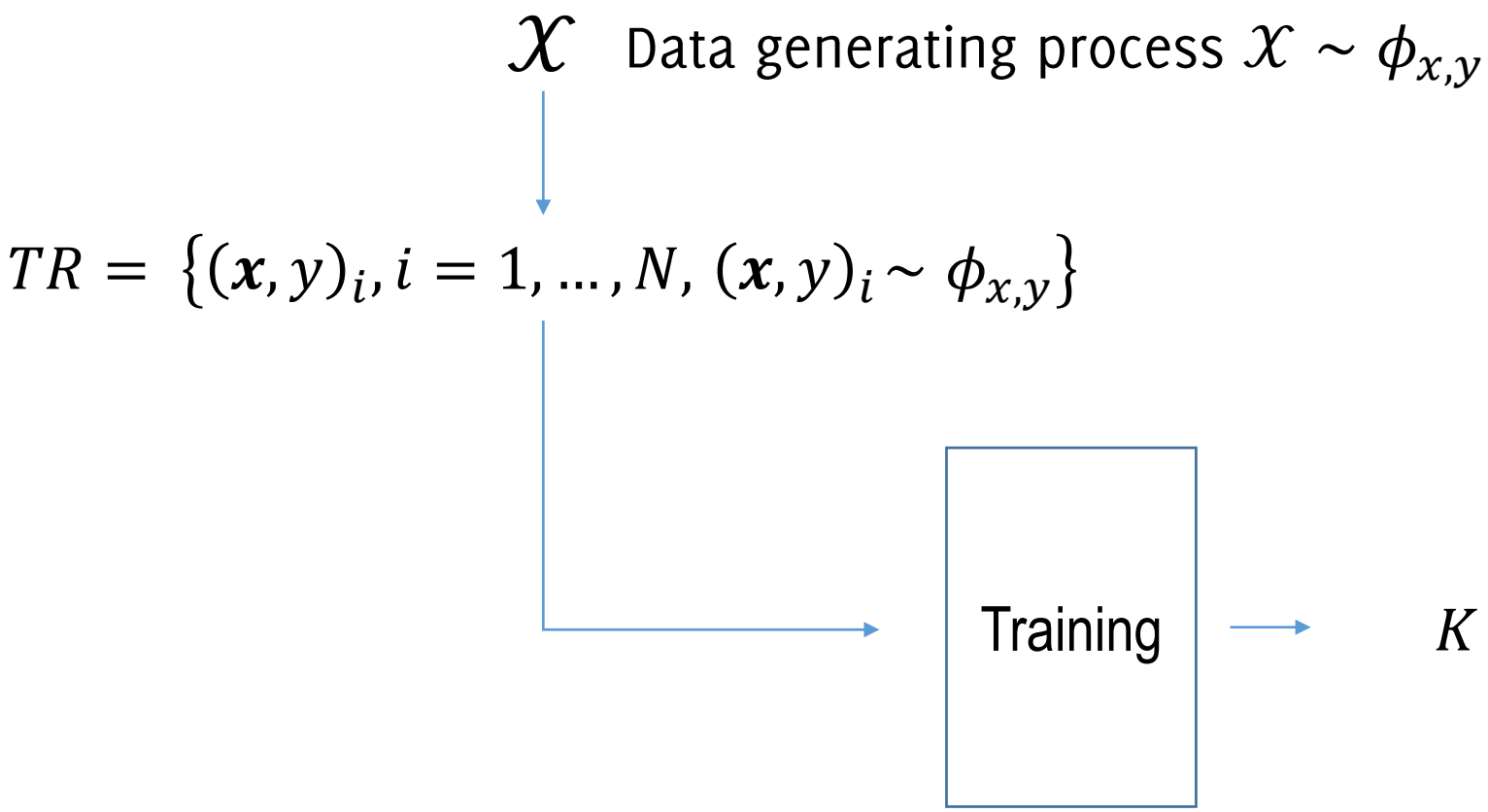
Here, we consider the **classification error** to measure how good a learned model K matches the distribution $\phi_{\boldsymbol{x},y}$, namely

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

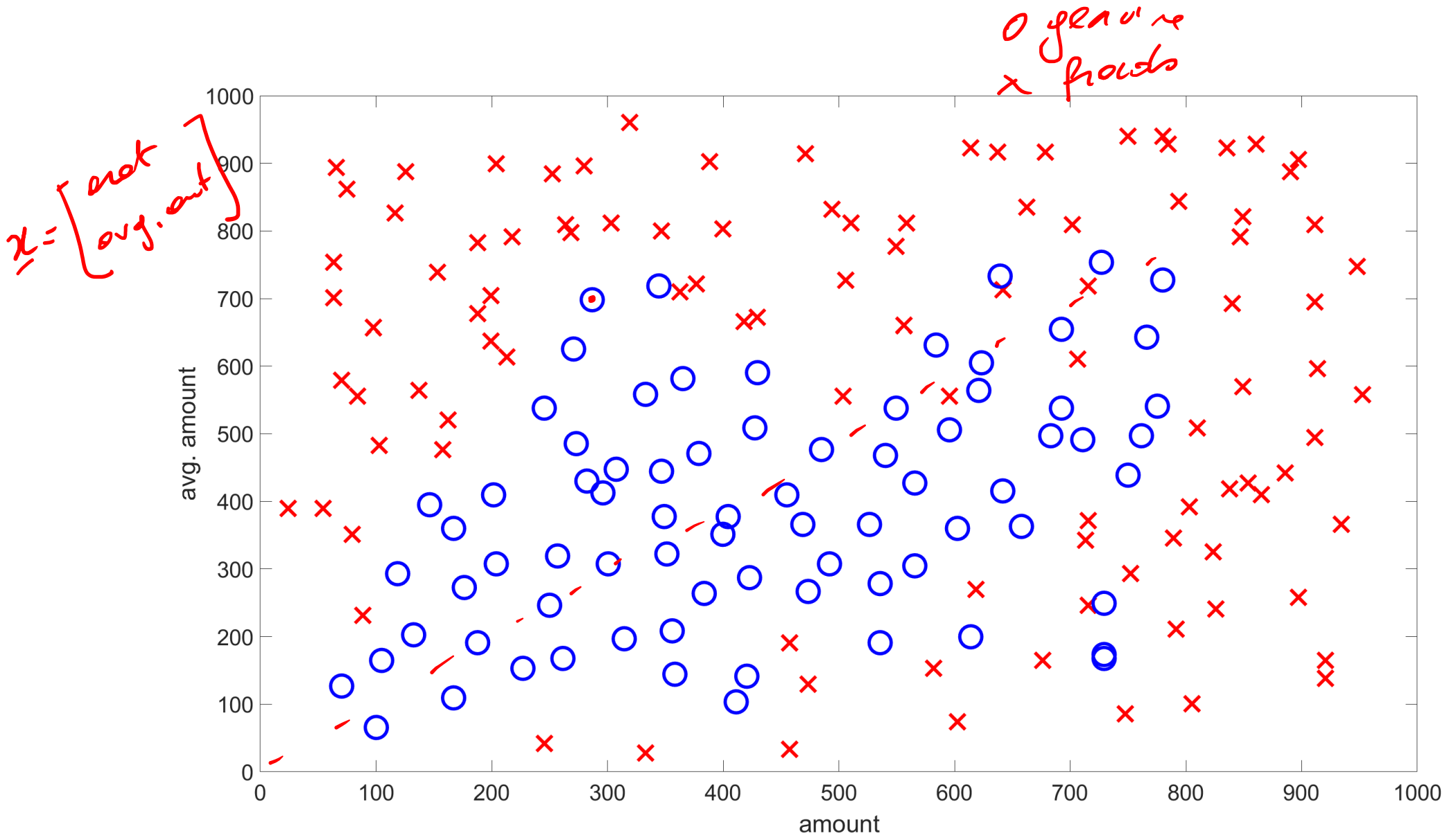
being R «a reference set» for assessing the error

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

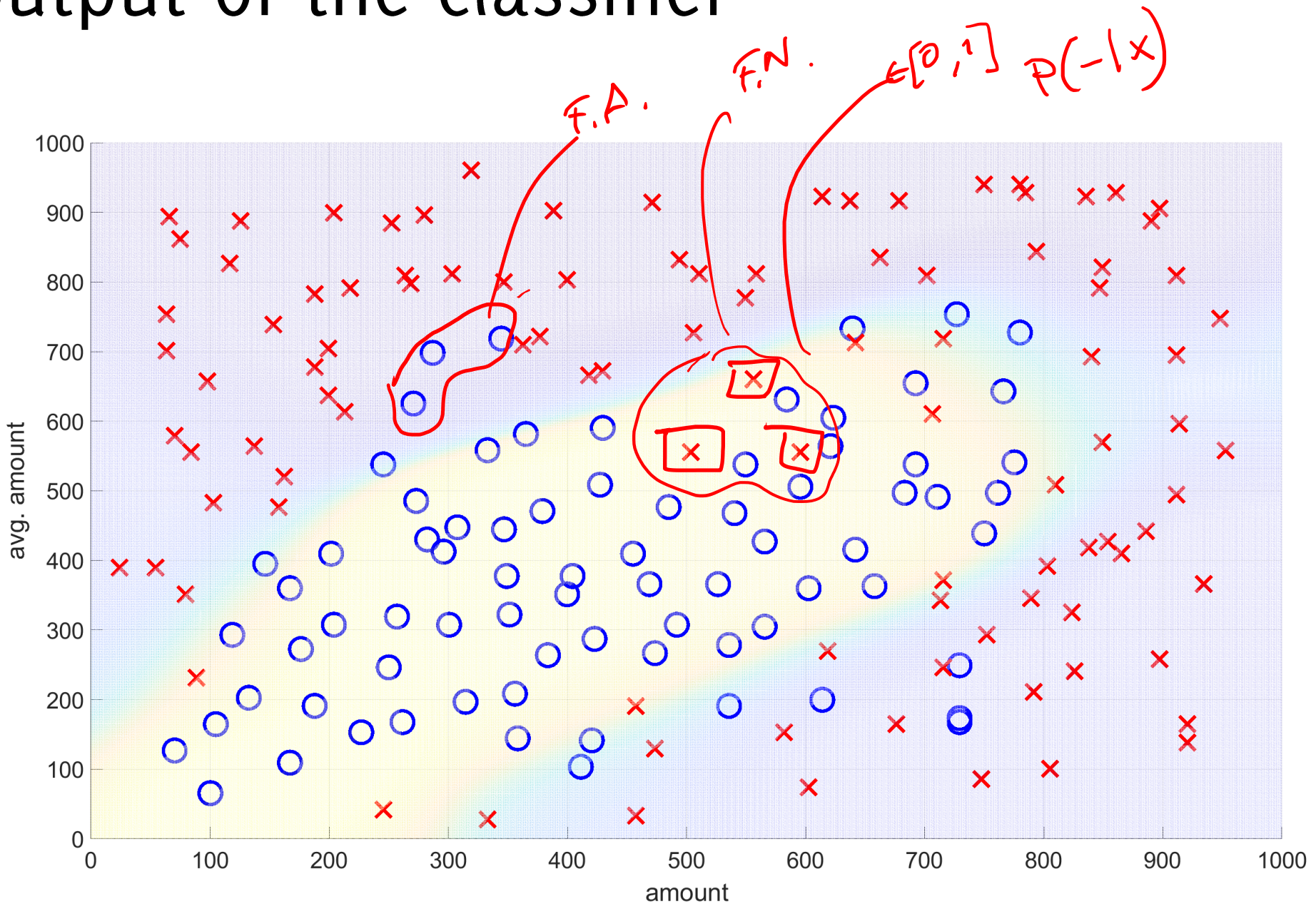
Training the Classifier



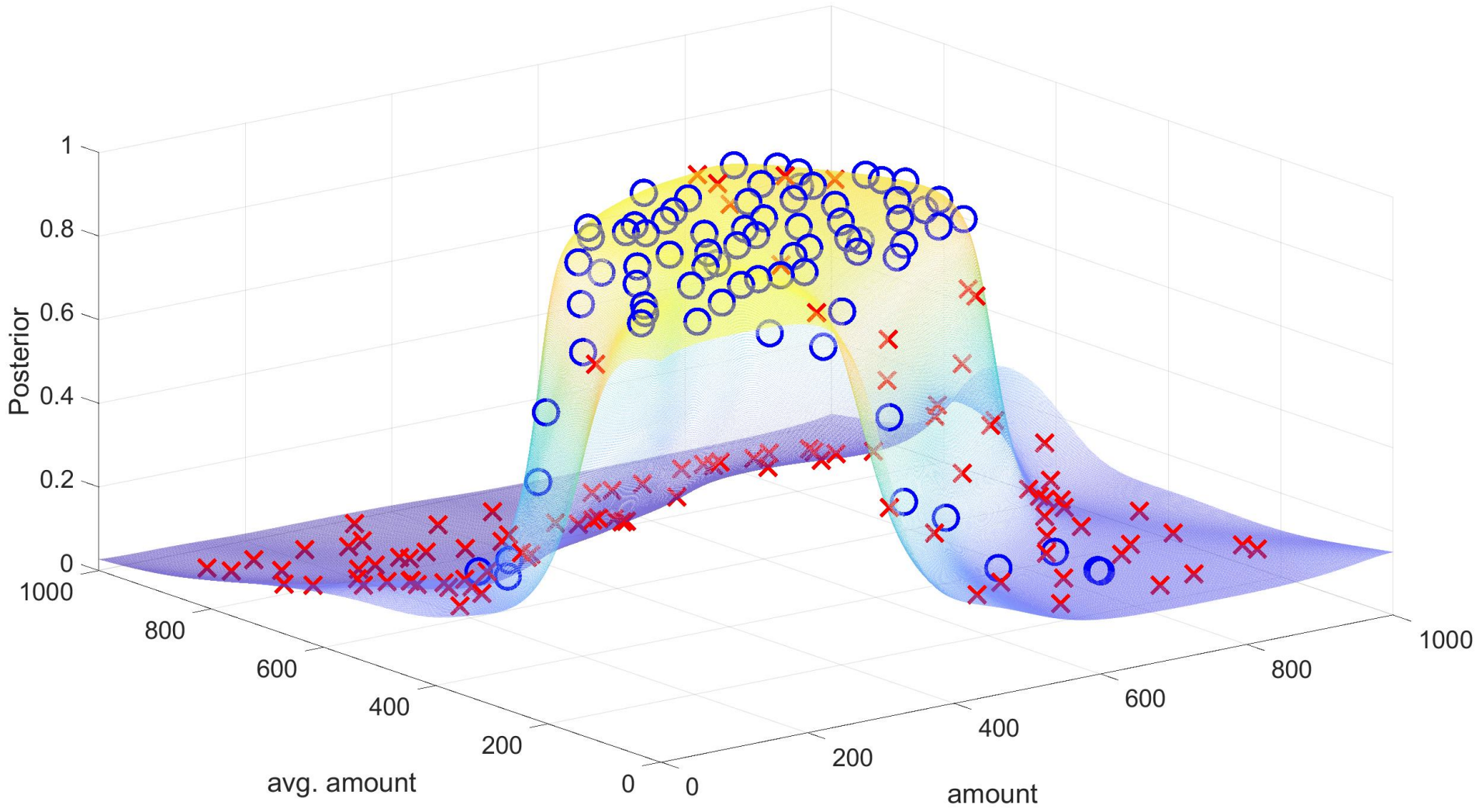
Training Set



The output of the classifier

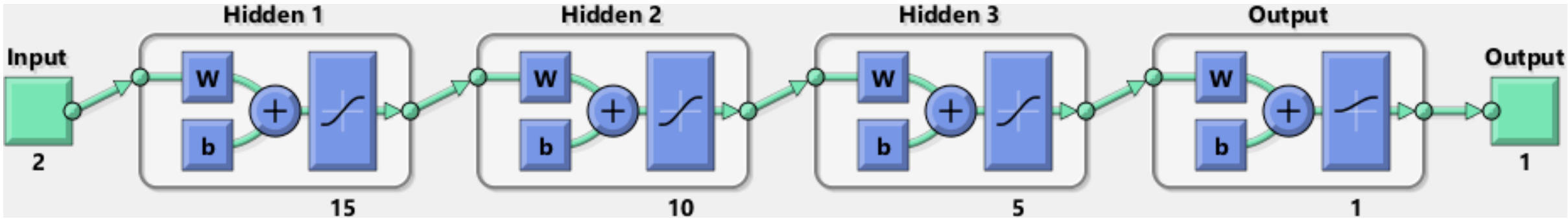


The output of the classifier

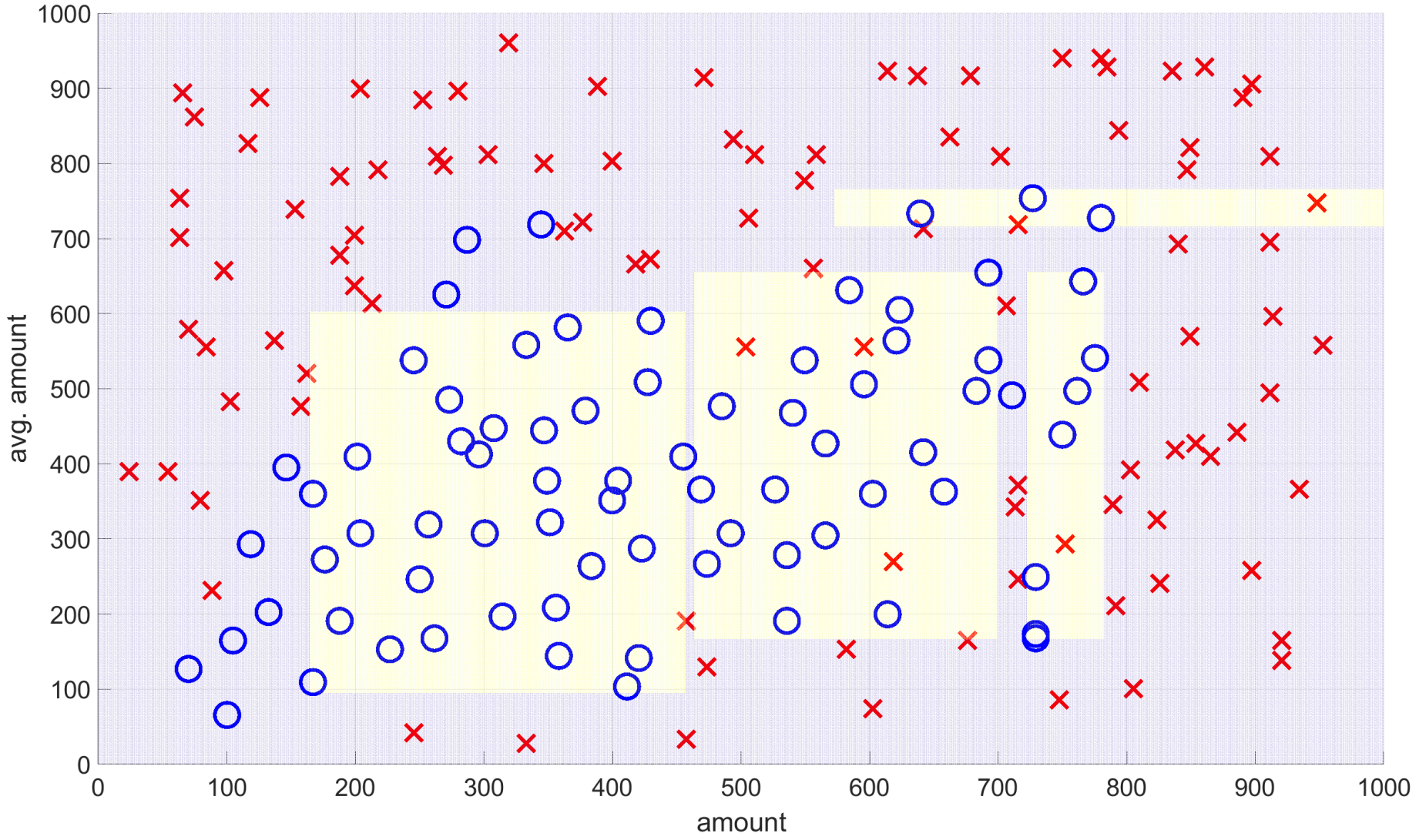


Boracchi, Trovò

Btw… that was a Neural Network

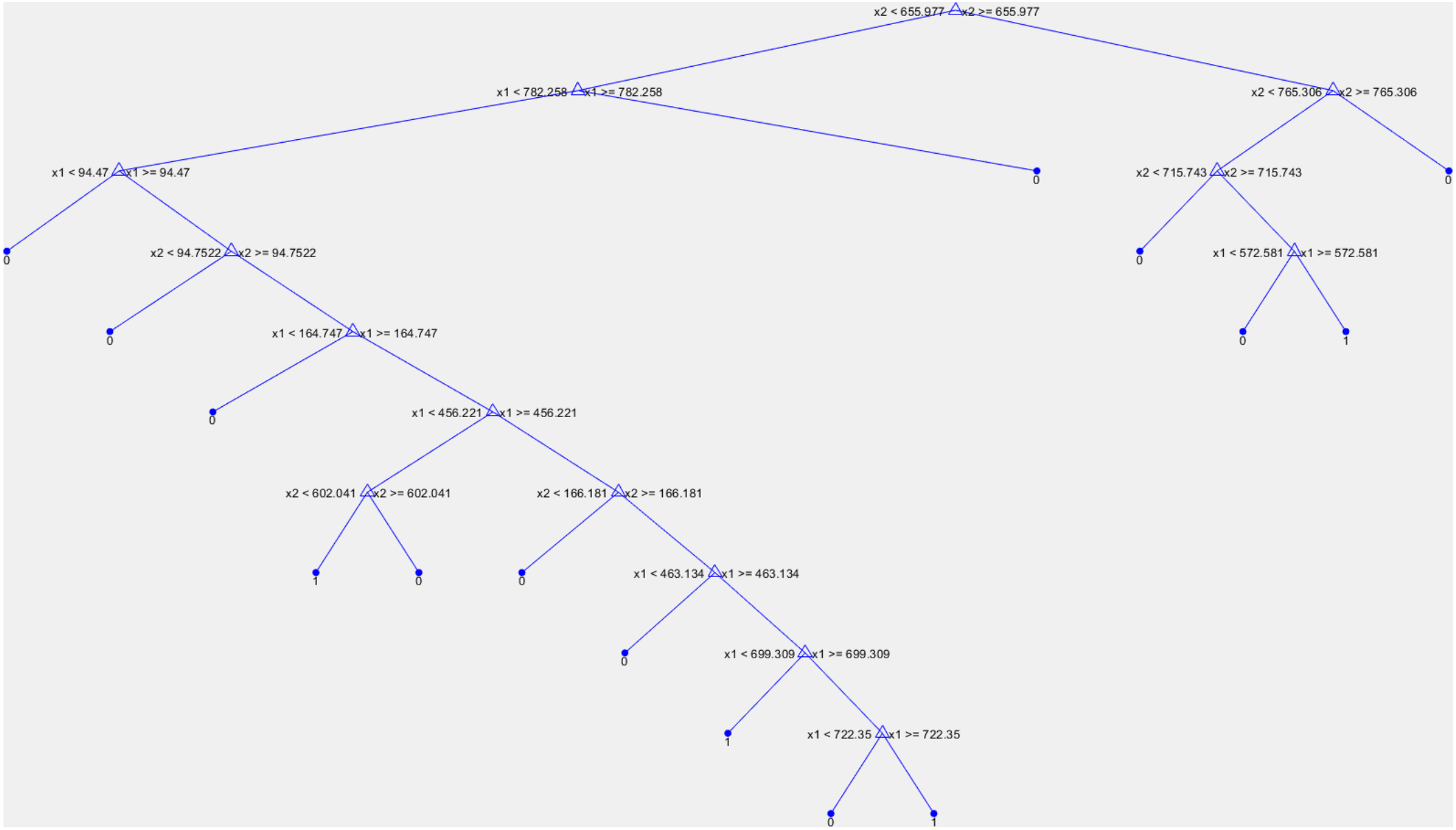


The output of another classifier



Boracchi, Trovò

Yes, that was a decision tree

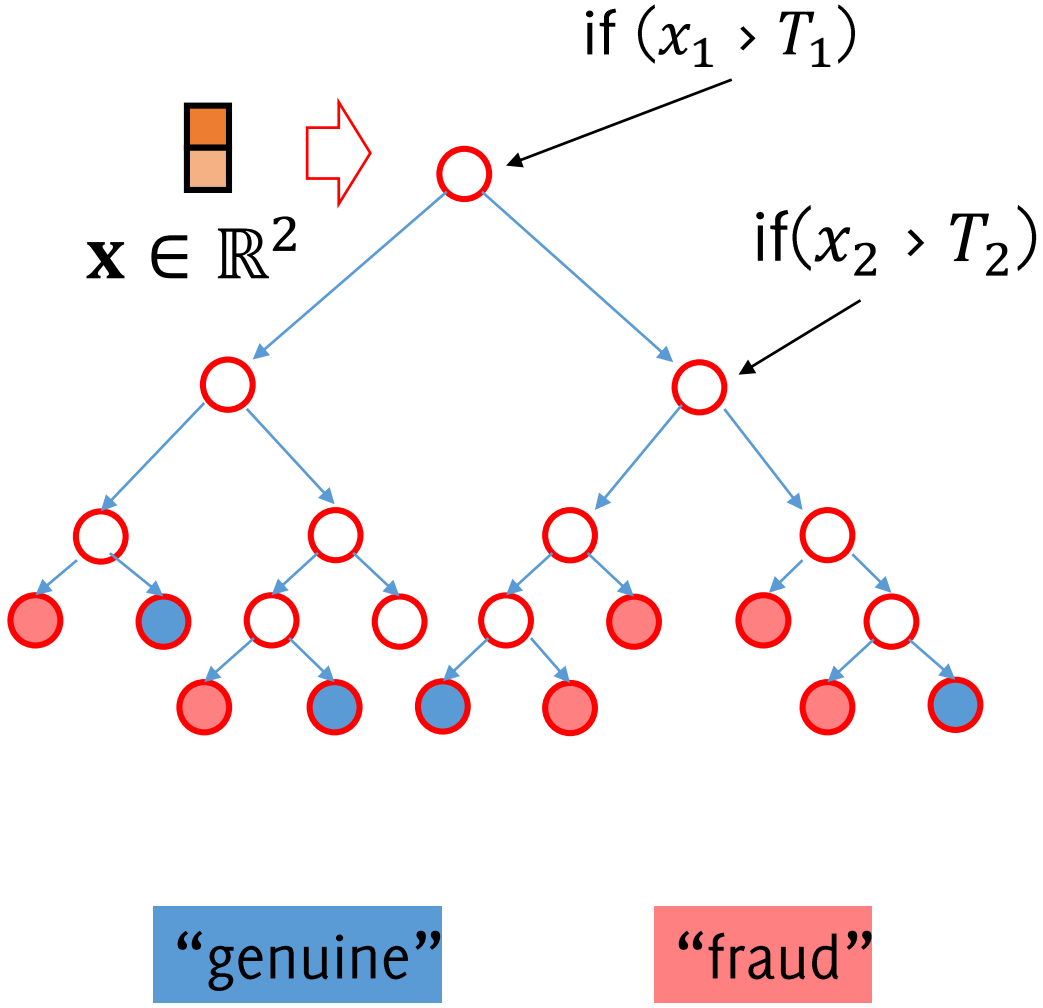


Boracchi, Trovò

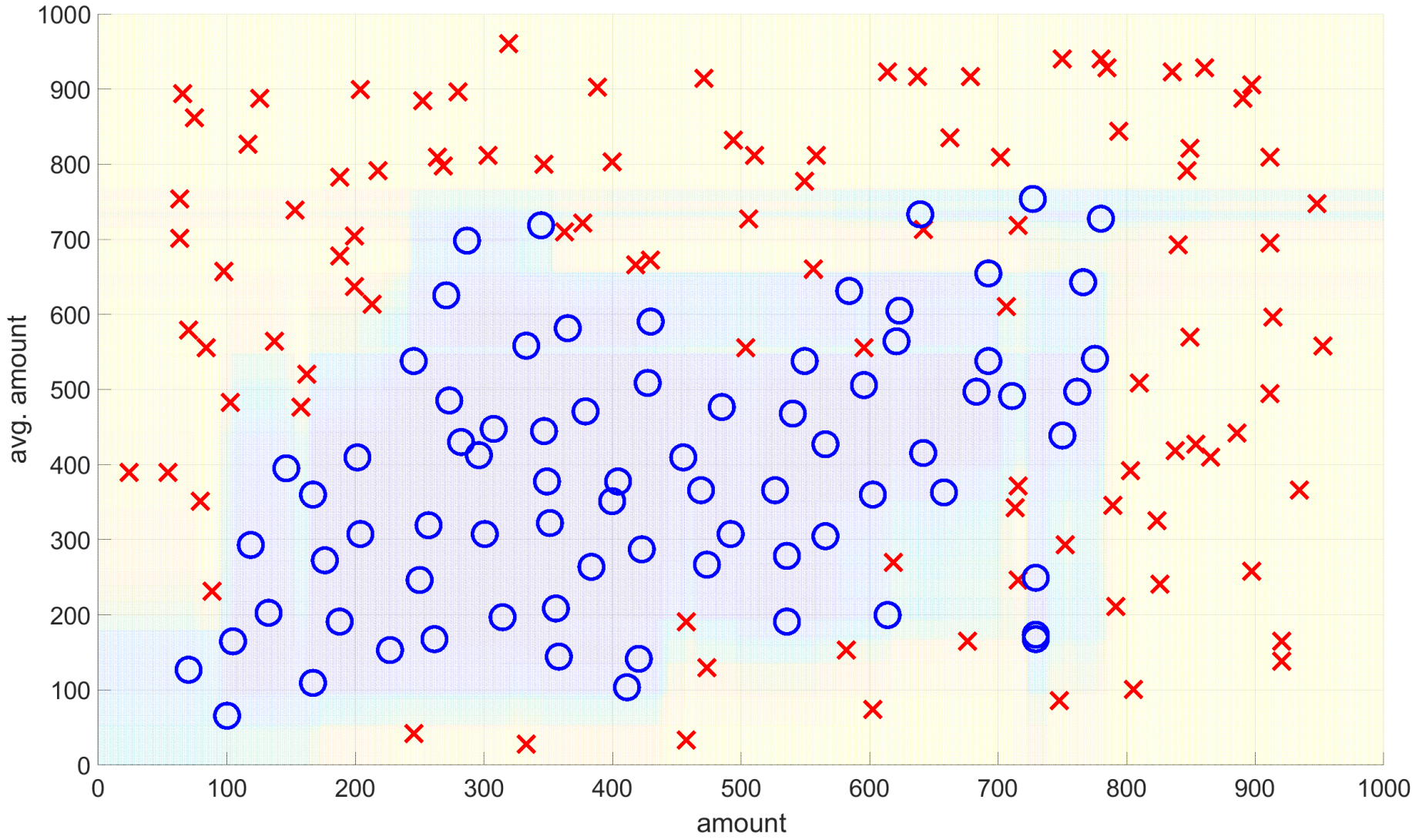
Decision Tree

The classifier has a few patameters:

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

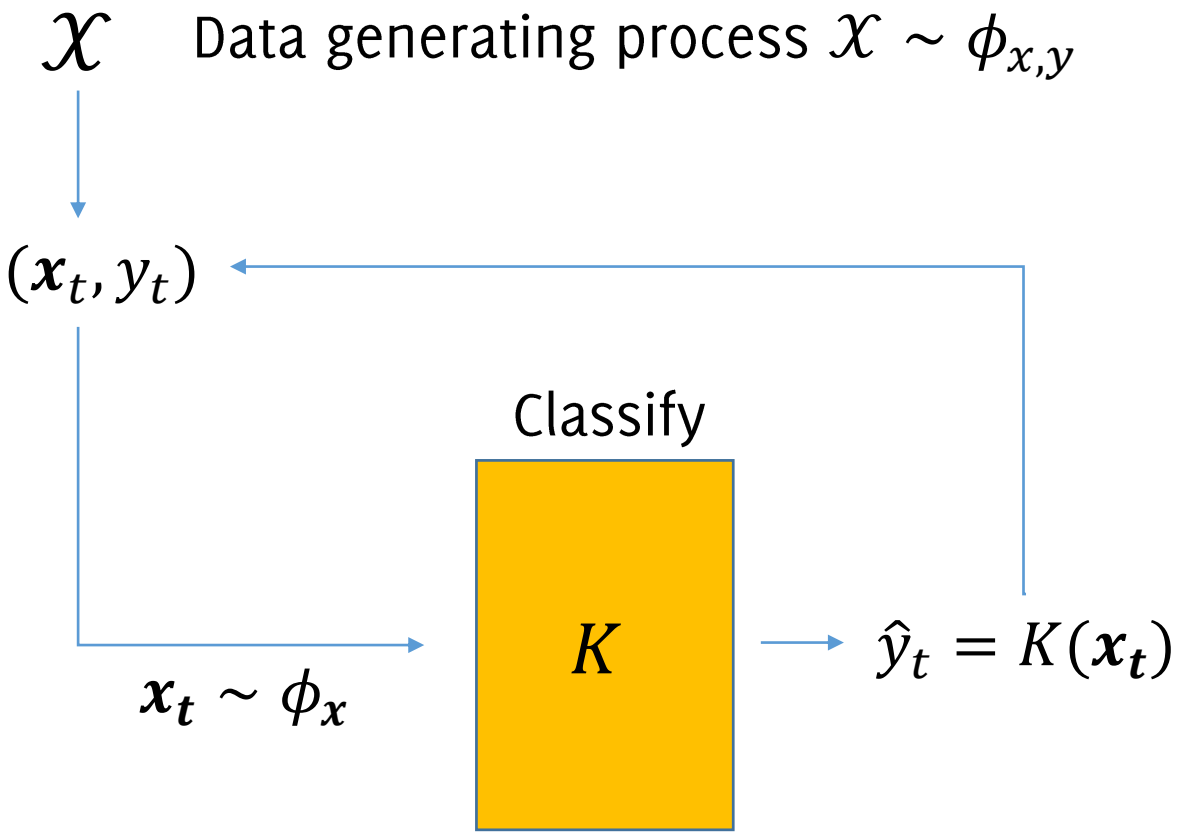


… better with many (50) trees, random forest



Boracchi, Trovò

Classification (Inference)



Supervised Information (performance assessment)

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

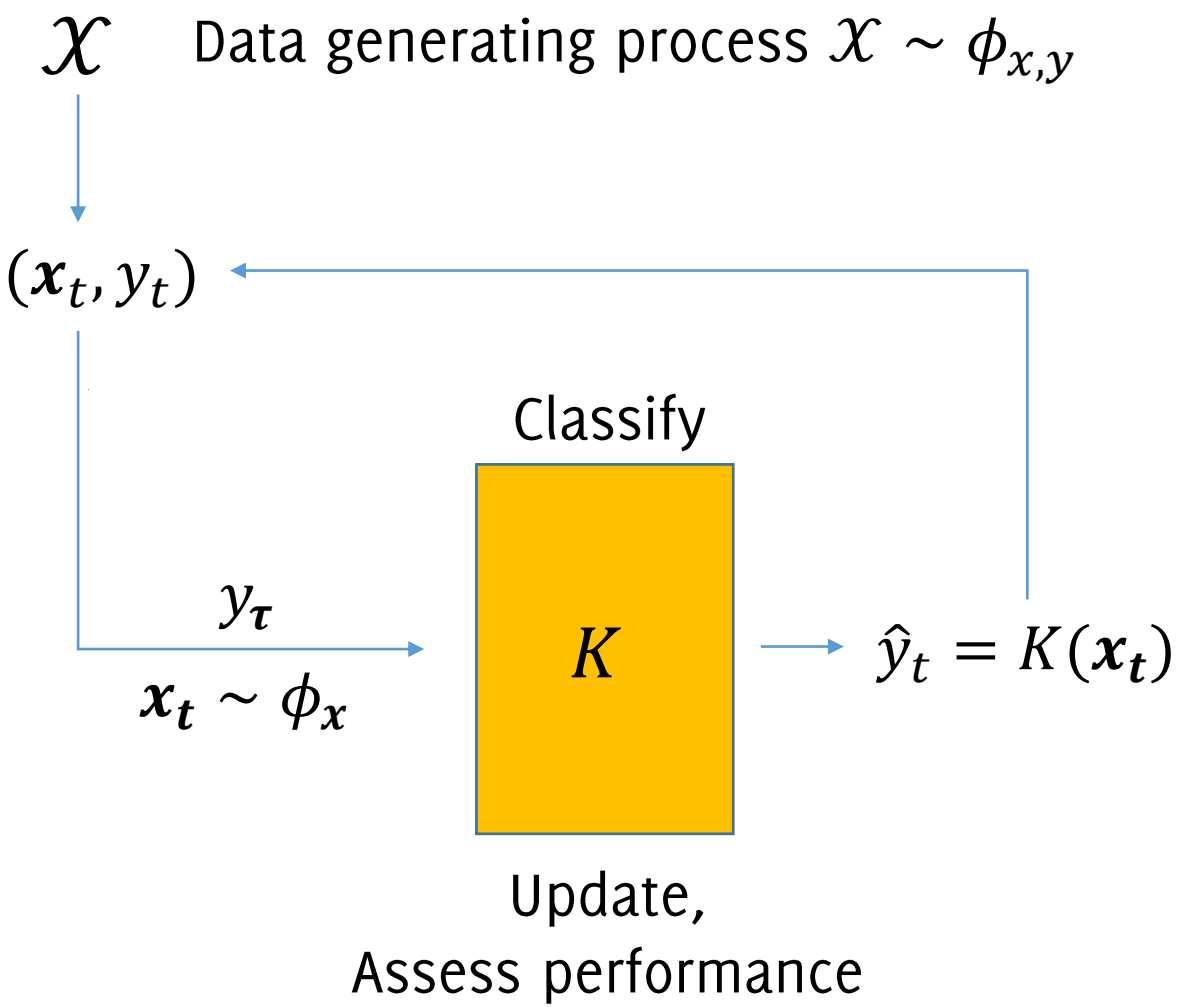
These are very useful for:

- assessing performance of K

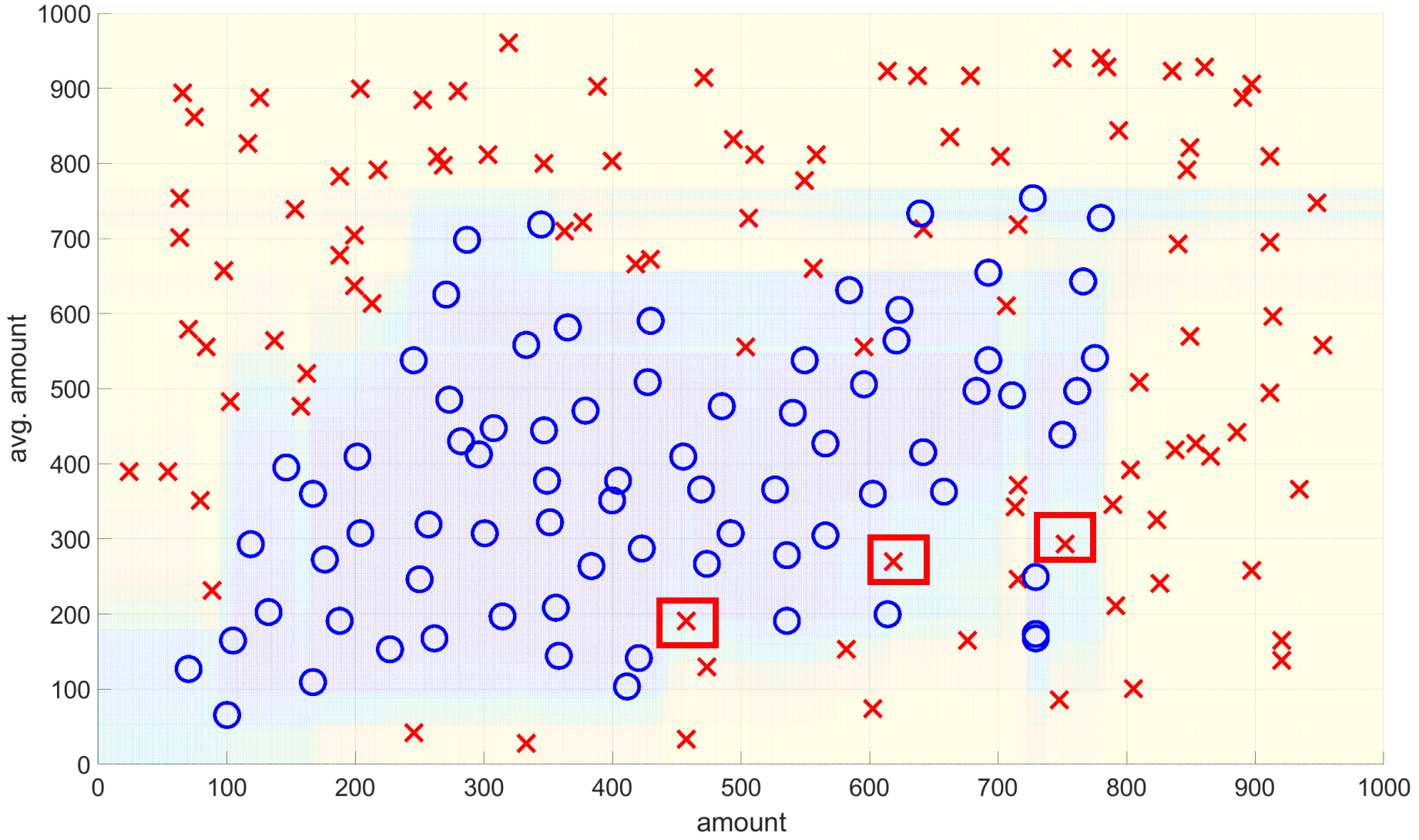
$$p(T) = \frac{1}{T}\sum_t e_t$$

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

- updating K



Classification Errors



Learning in Nonstationary (Streaming) Environments

The Problem Formulation

Concept Drift

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

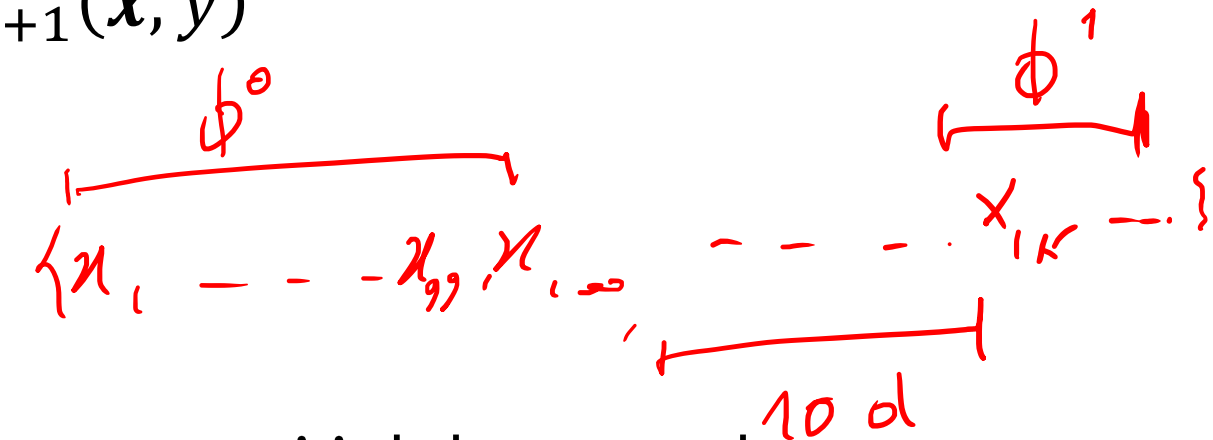
The data generating process is then modeled as:

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

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

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

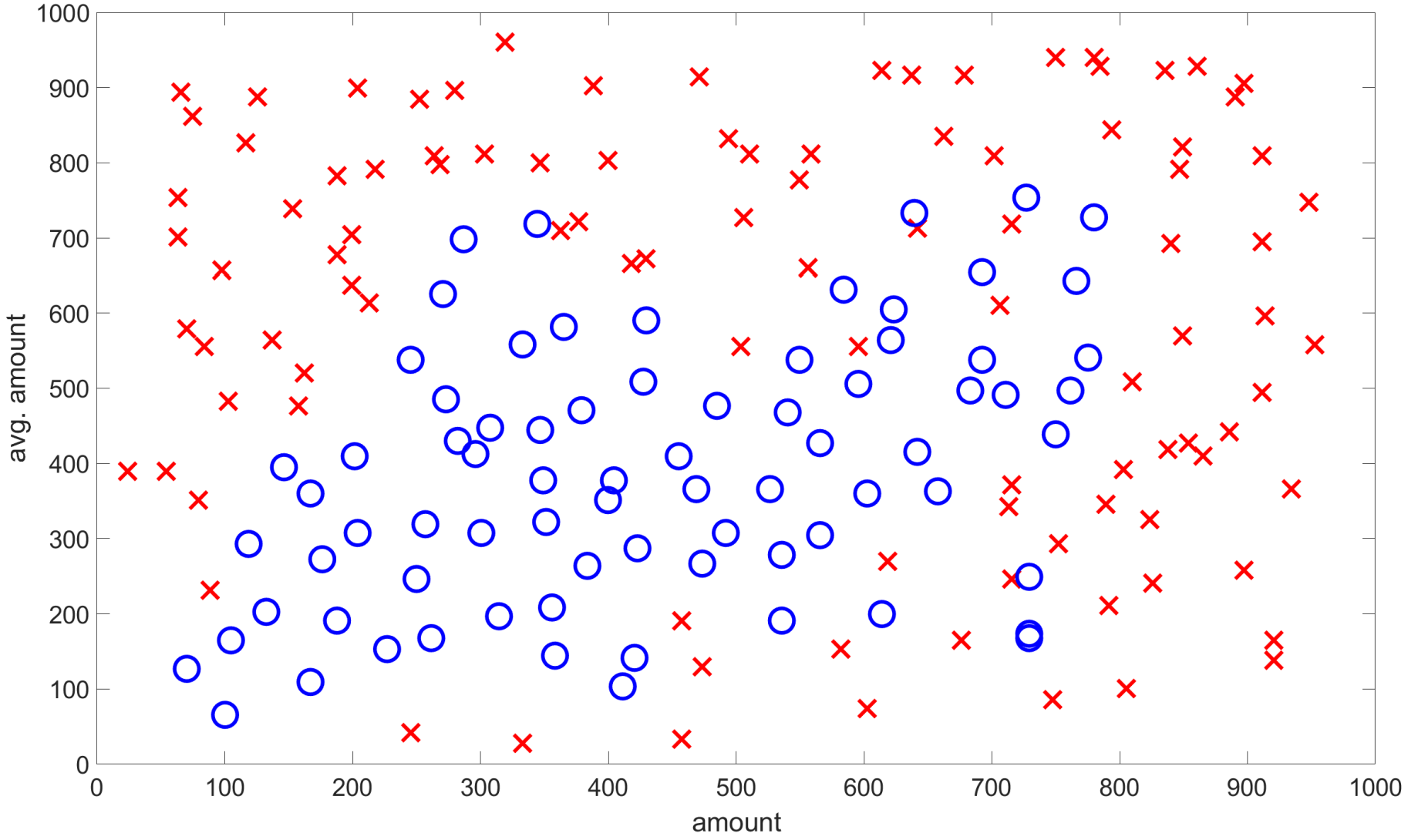
We also say $\mathcal{X}$ becomes **nonstationary**.



TR is always assumed i.i.d.

After the change (e.g. 10 days), data are no more i.i.d. because data are not identically distributed after

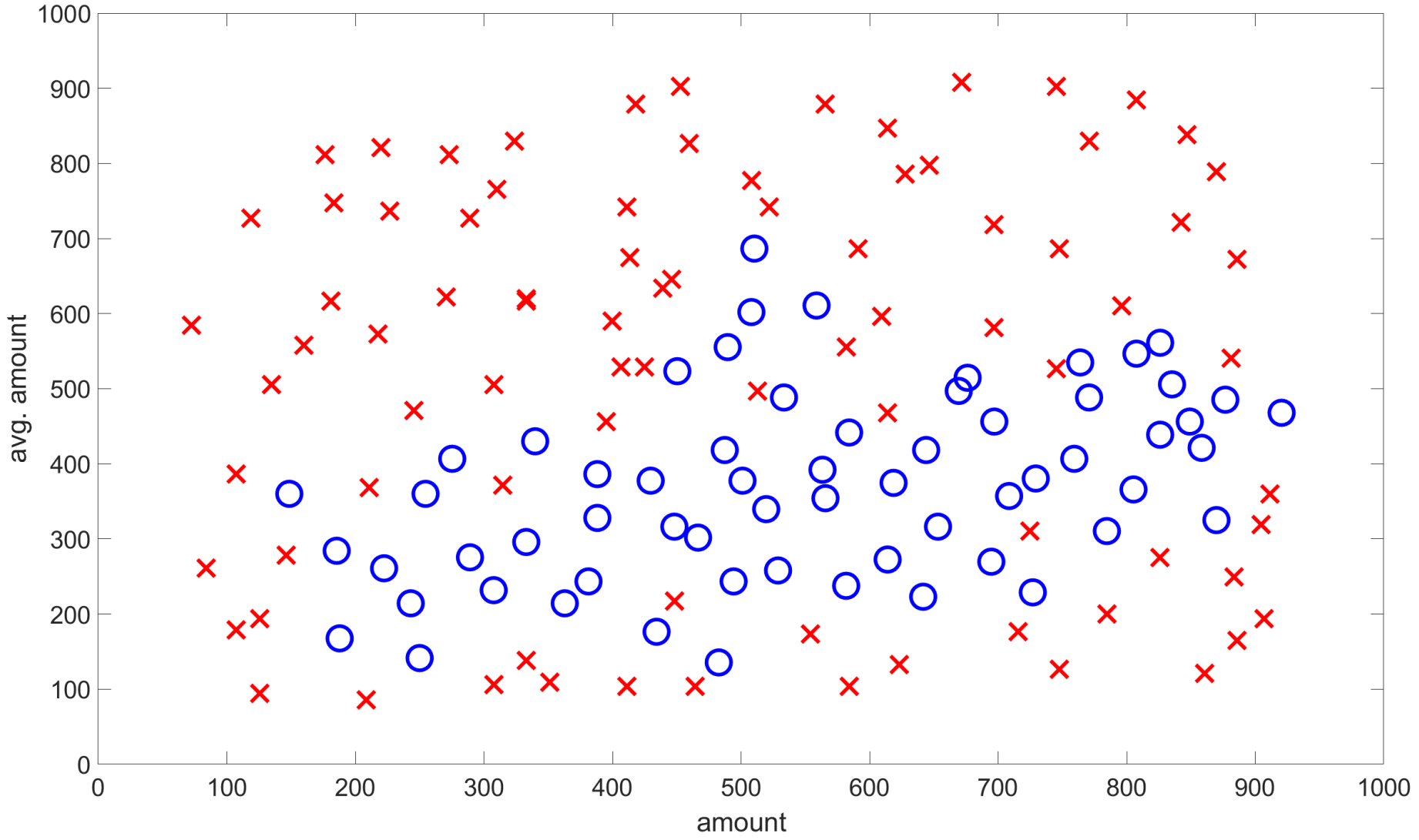
Distribution Changes



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

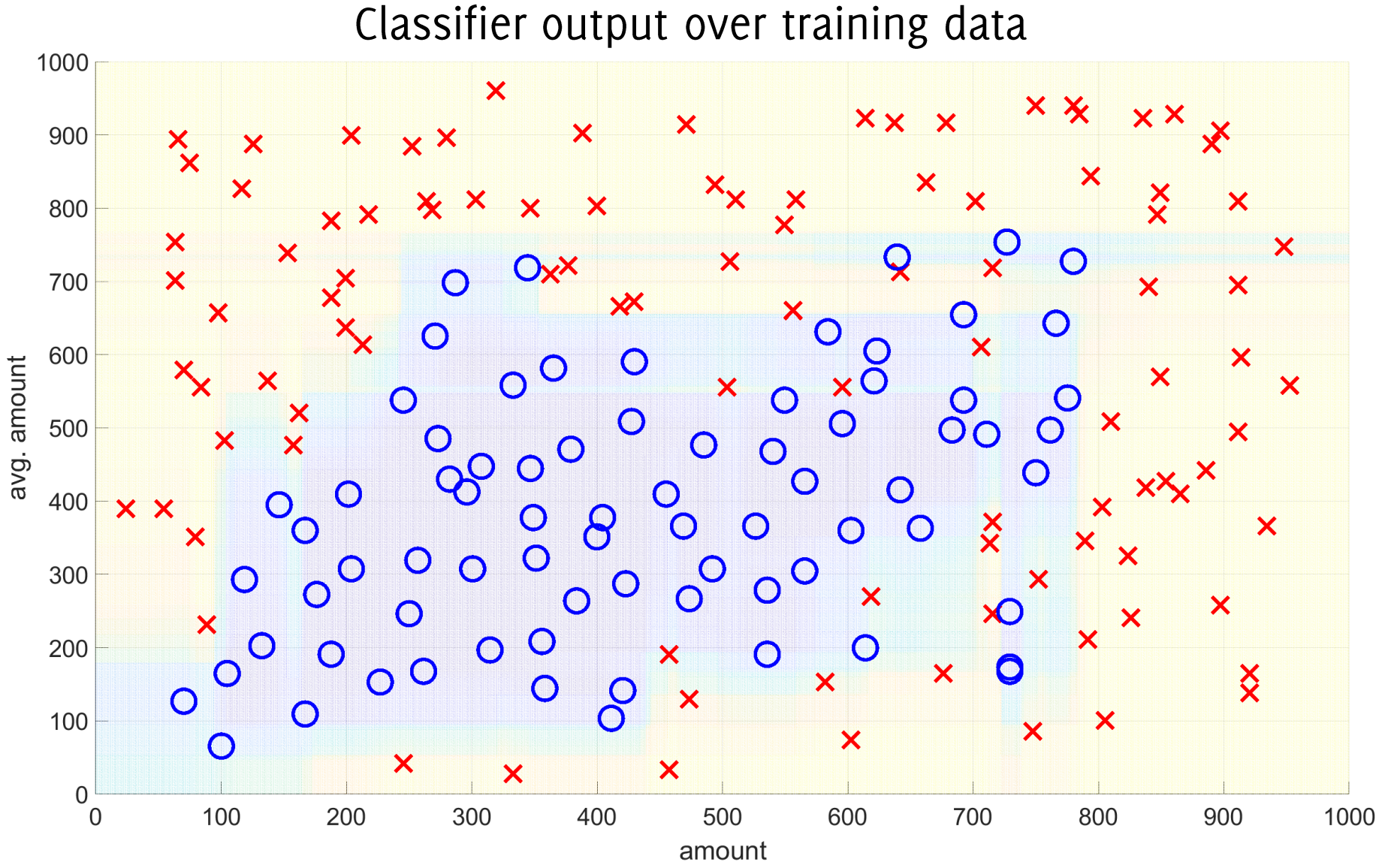
Boracchi, Trovò

Distribution Changes



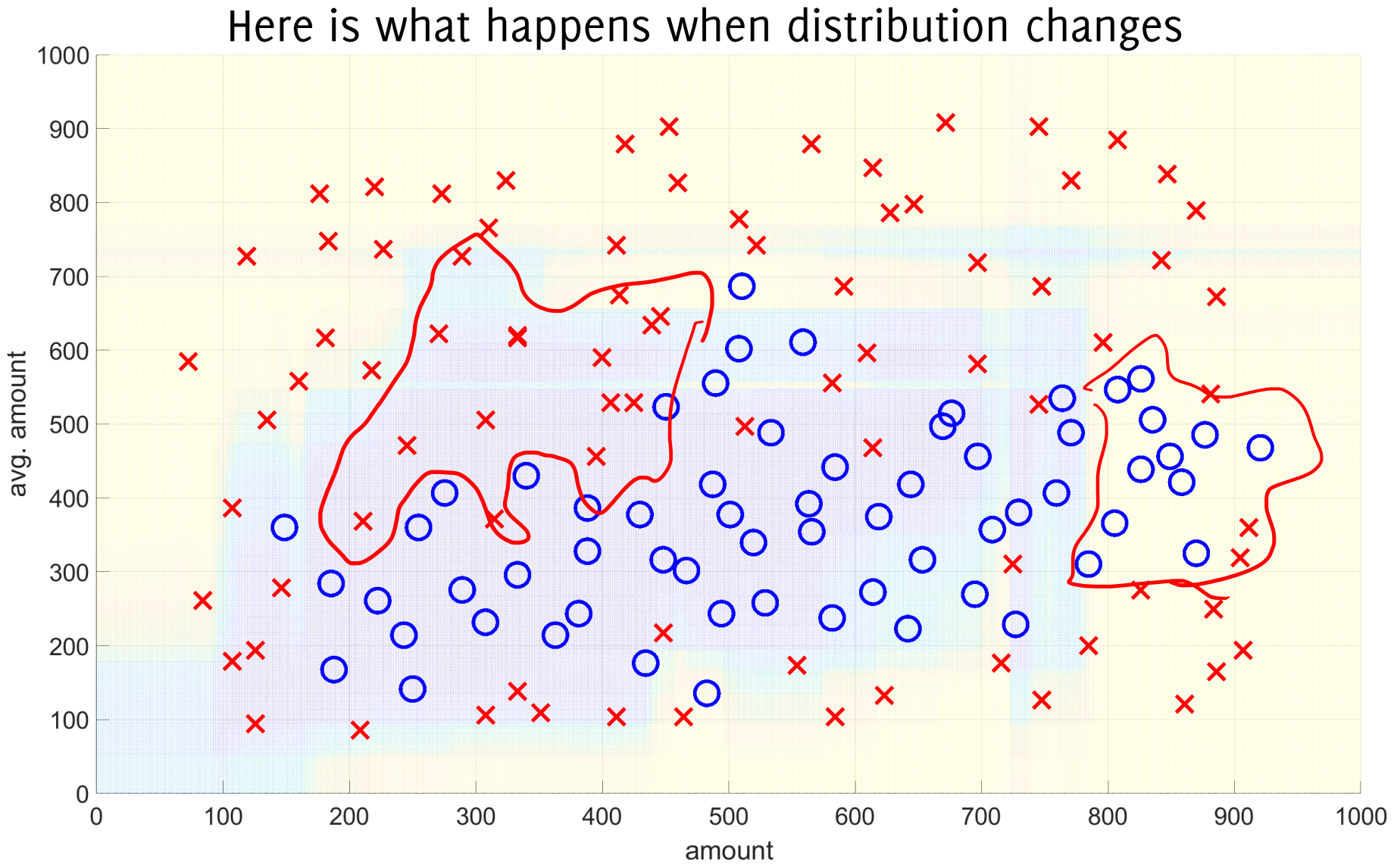
$\phi^1_{x,y}$

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



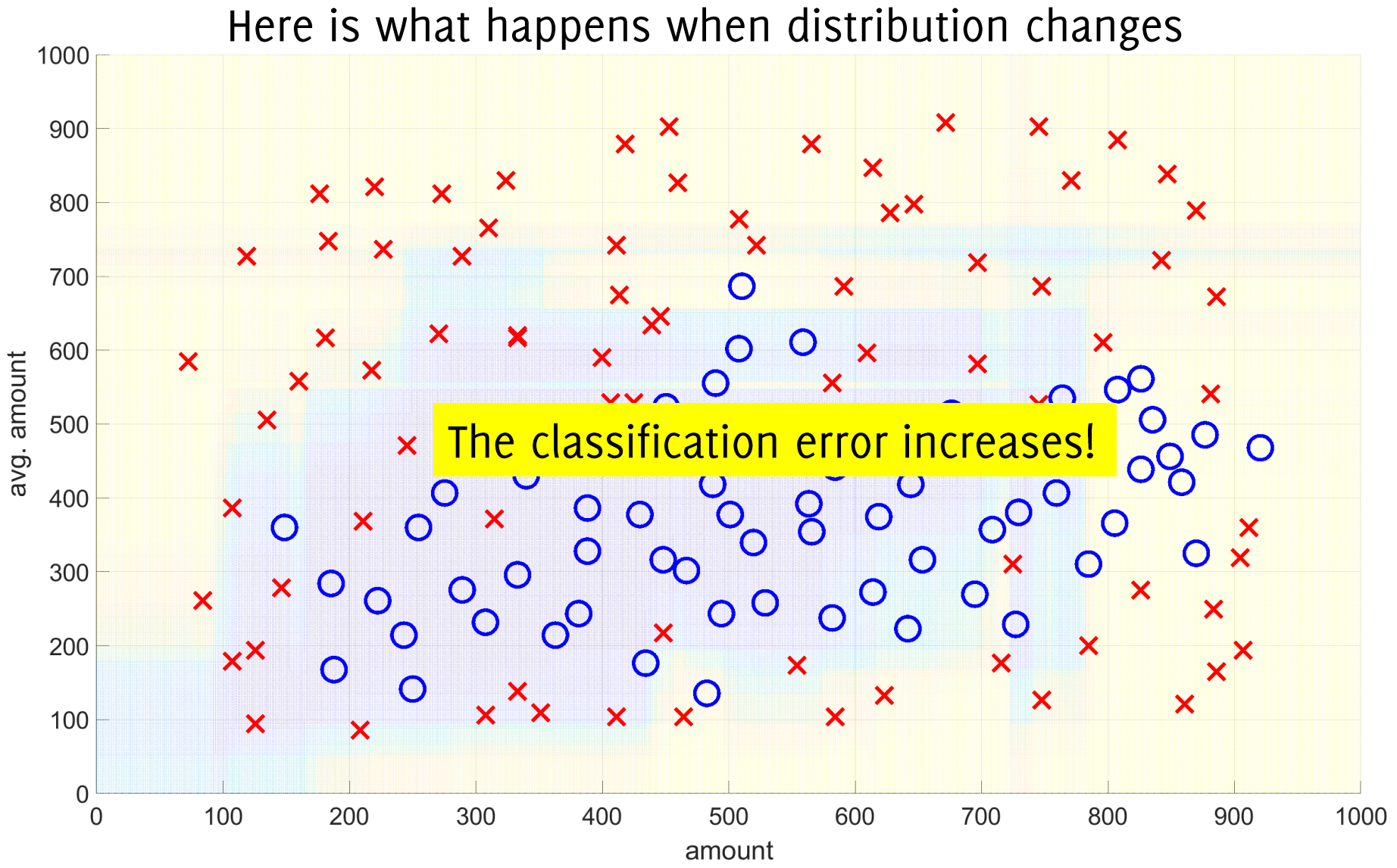
$\phi^0_{x,y}$

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.

- **Measuring and Monitoring the classification error**
- **Updating the classifier**

Concept Drift Taxonomy

Setting up the Stage

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?

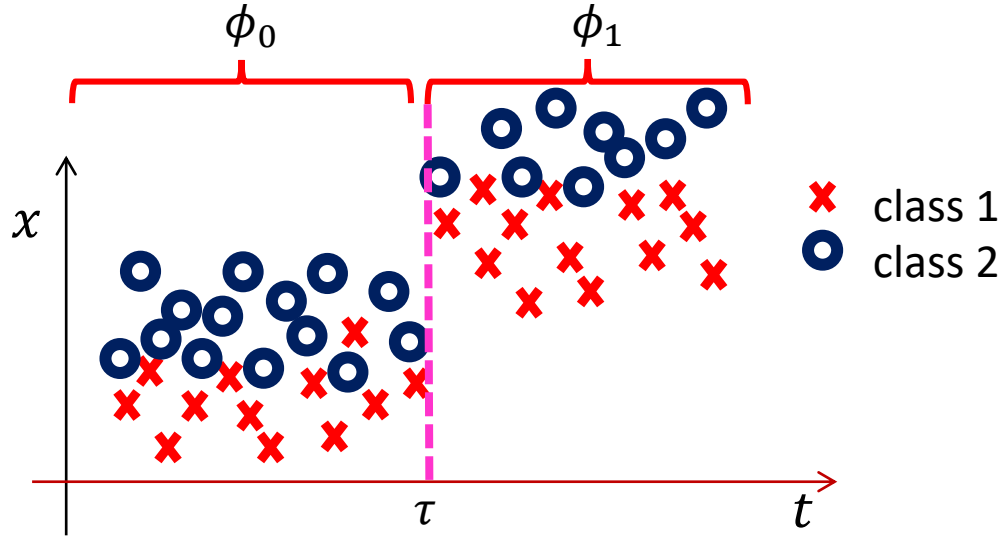
Real Drift

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

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

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

Rmk. Here inputs $\boldsymbol{x}$ are scalar



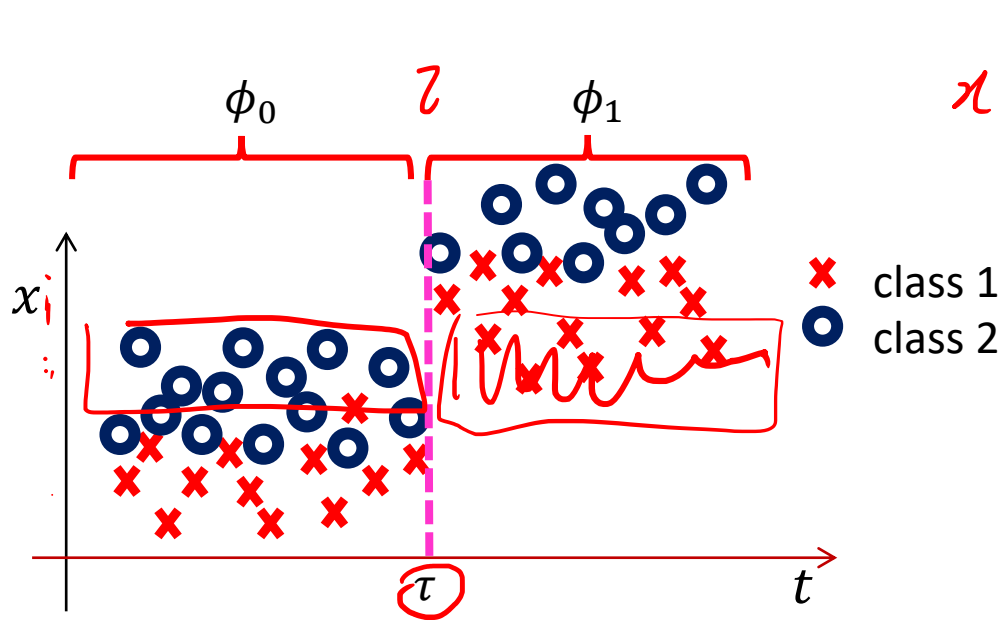
Drift Taxonomy: What Is Changing?

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?

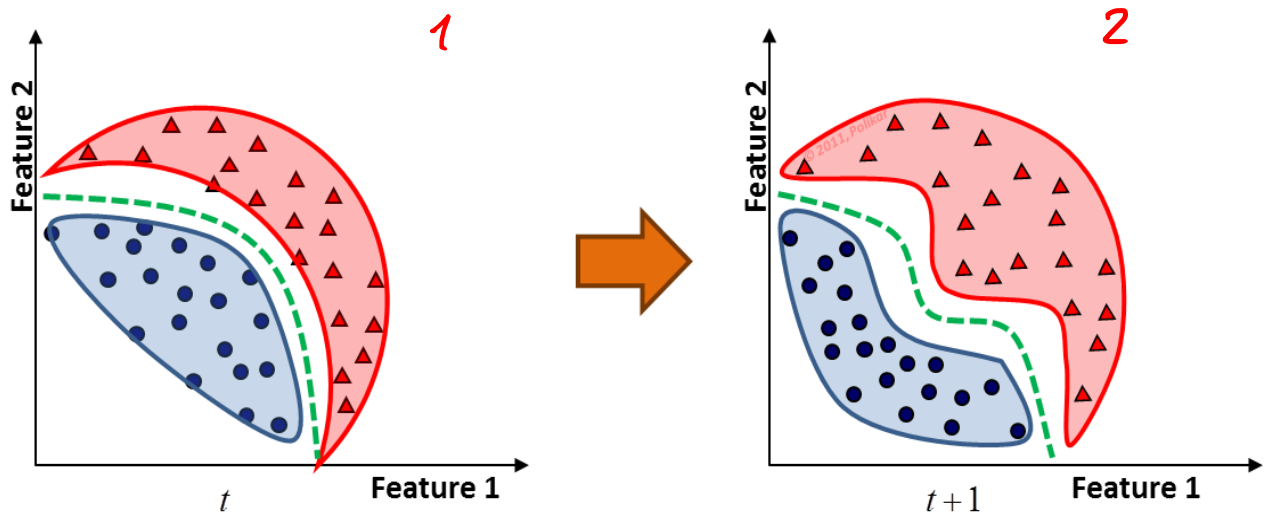
Real Drift

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

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

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

Rmk. Here inputs $\boldsymbol{x}$ are two dimensional vectors



Drift Taxonomy: What Is Changing?

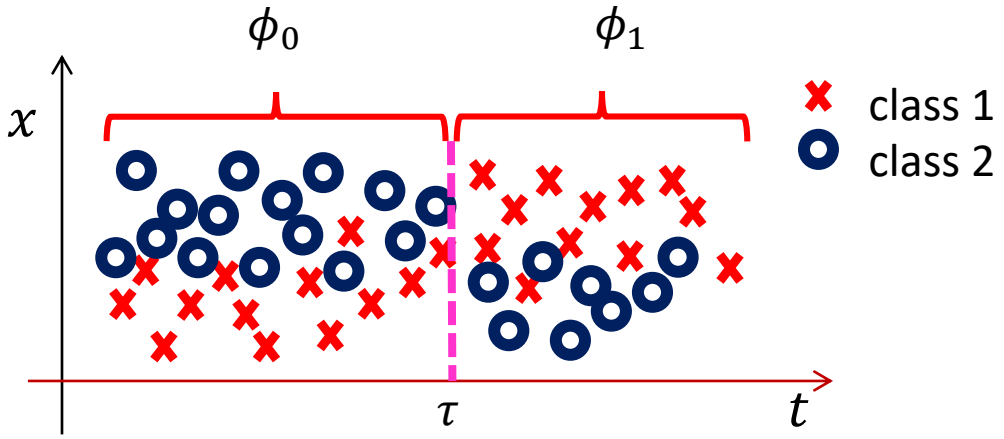
Real Drift

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

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



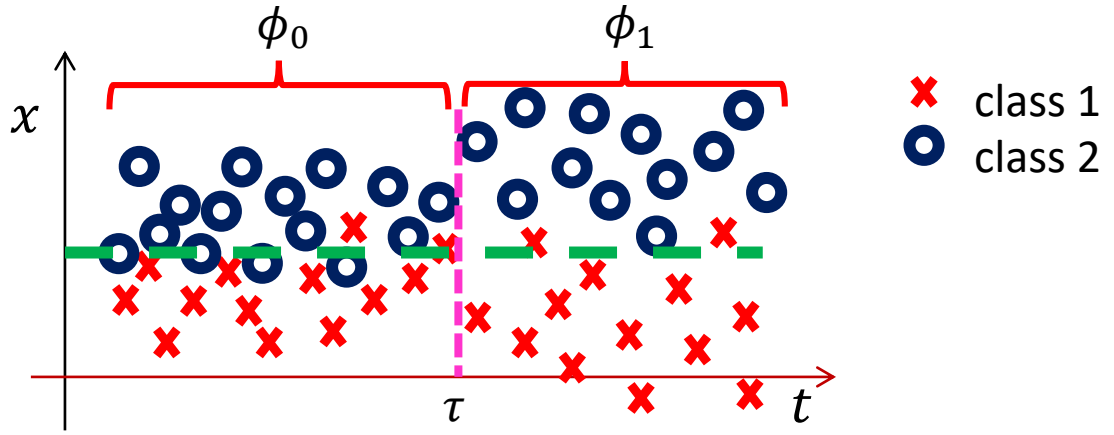
Drift Taxonomy: What Is Changing?

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



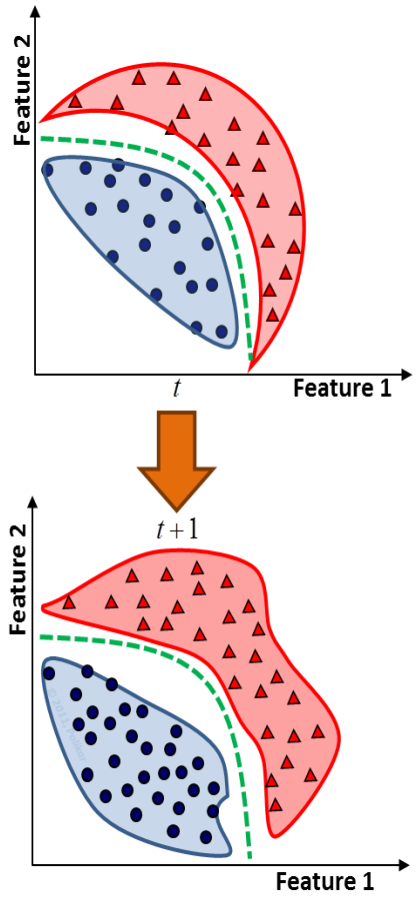
Drift Taxonomy: What Is Changing?

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)

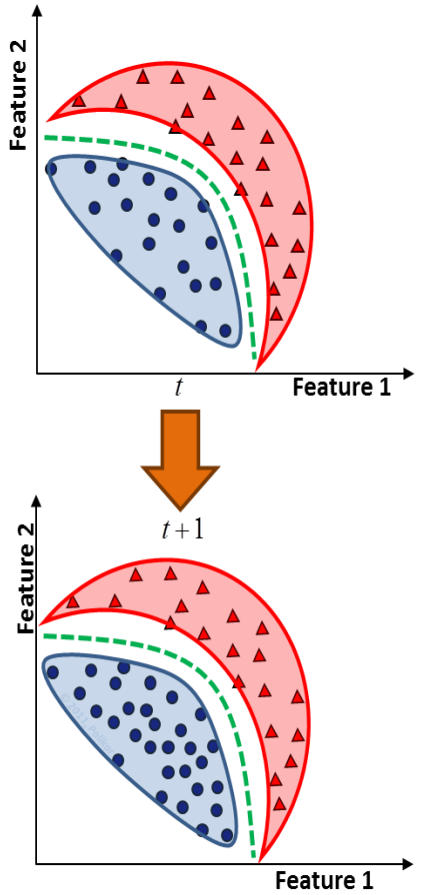
Drift Taxonomy: What Is Changing?

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.

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



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

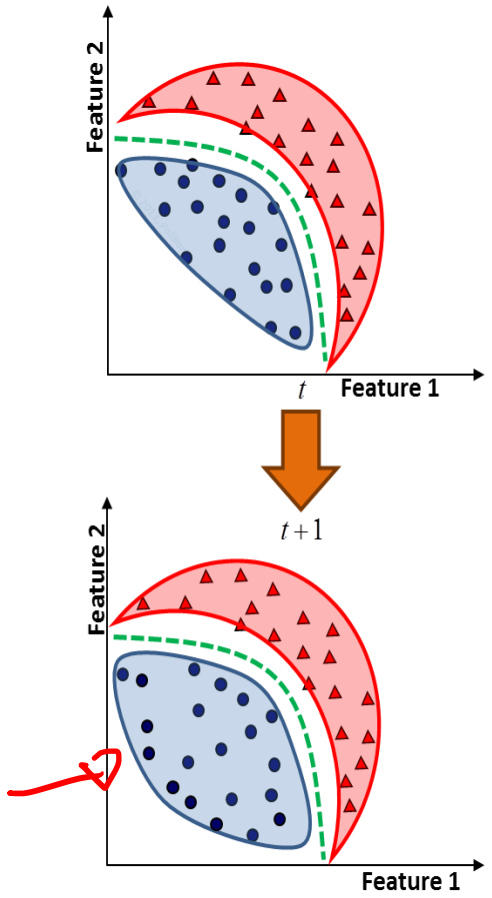
Drift Taxonomy: What Is Changing?

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.

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



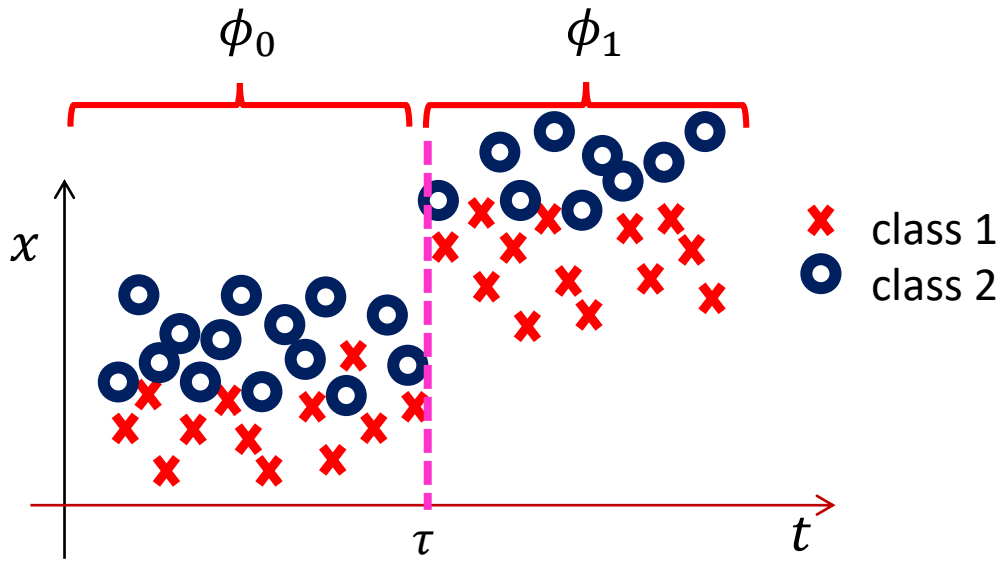
$$\phi_{\tau+1}(\boldsymbol{x}|y = c) \neq \phi_{\tau}(\boldsymbol{x}|y = c)$$

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



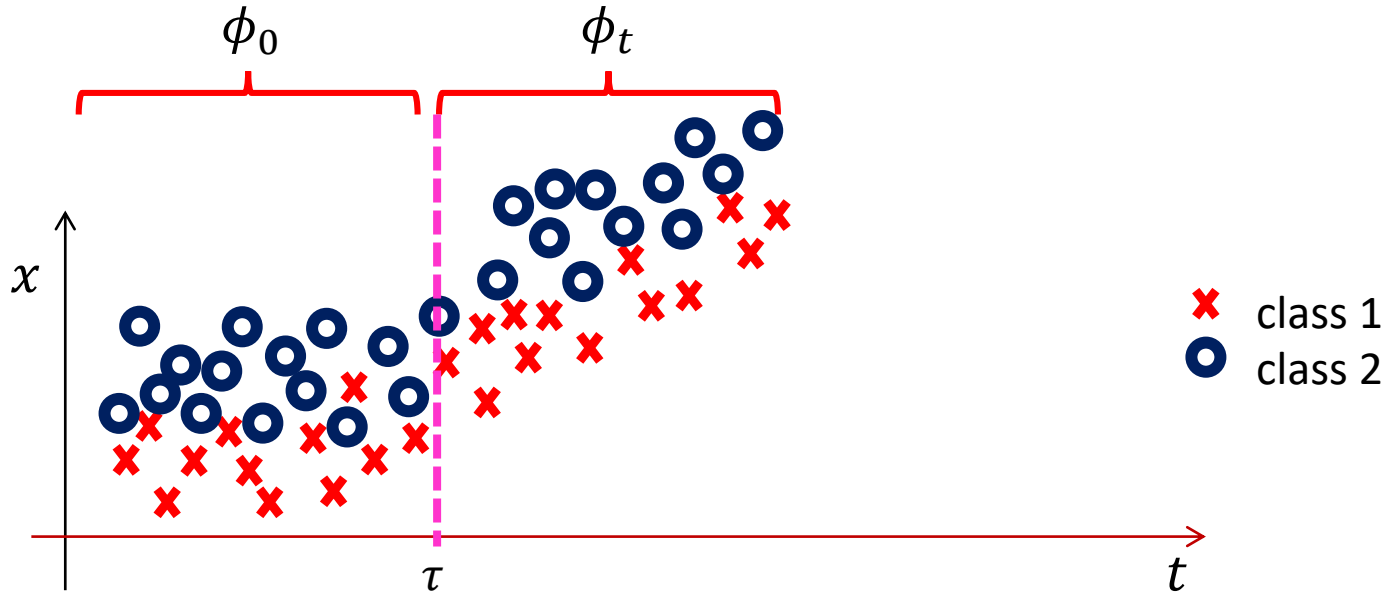
Boracchi, Trovò

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

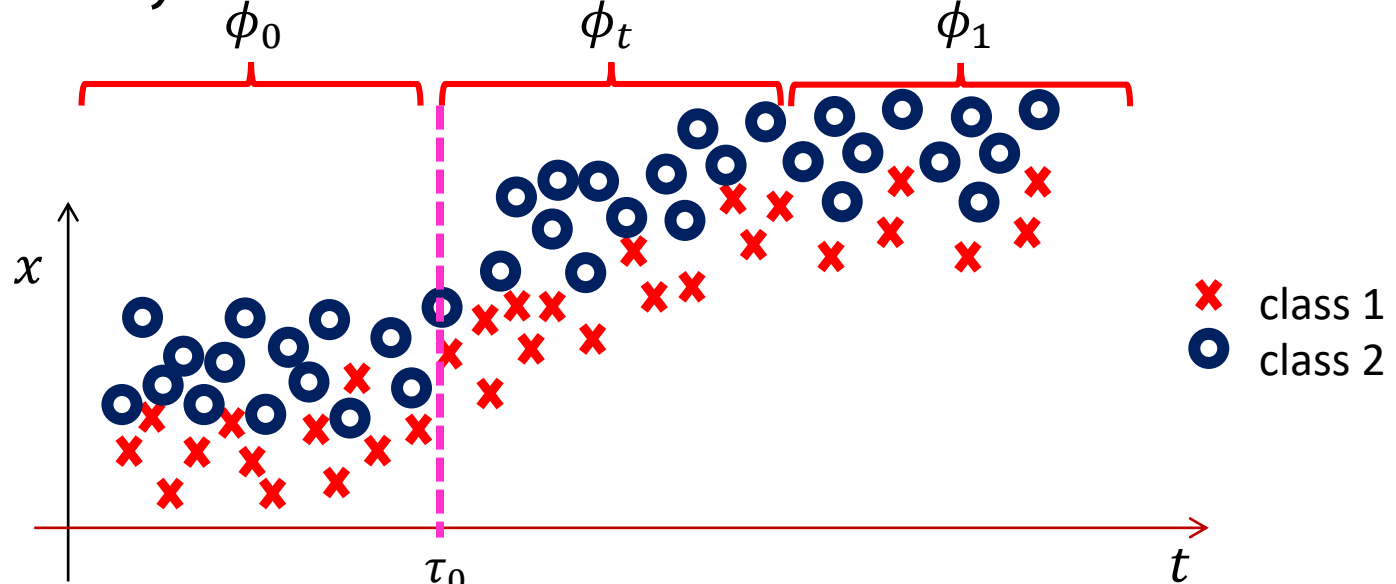


Drift Taxonomy: Time Evolution

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

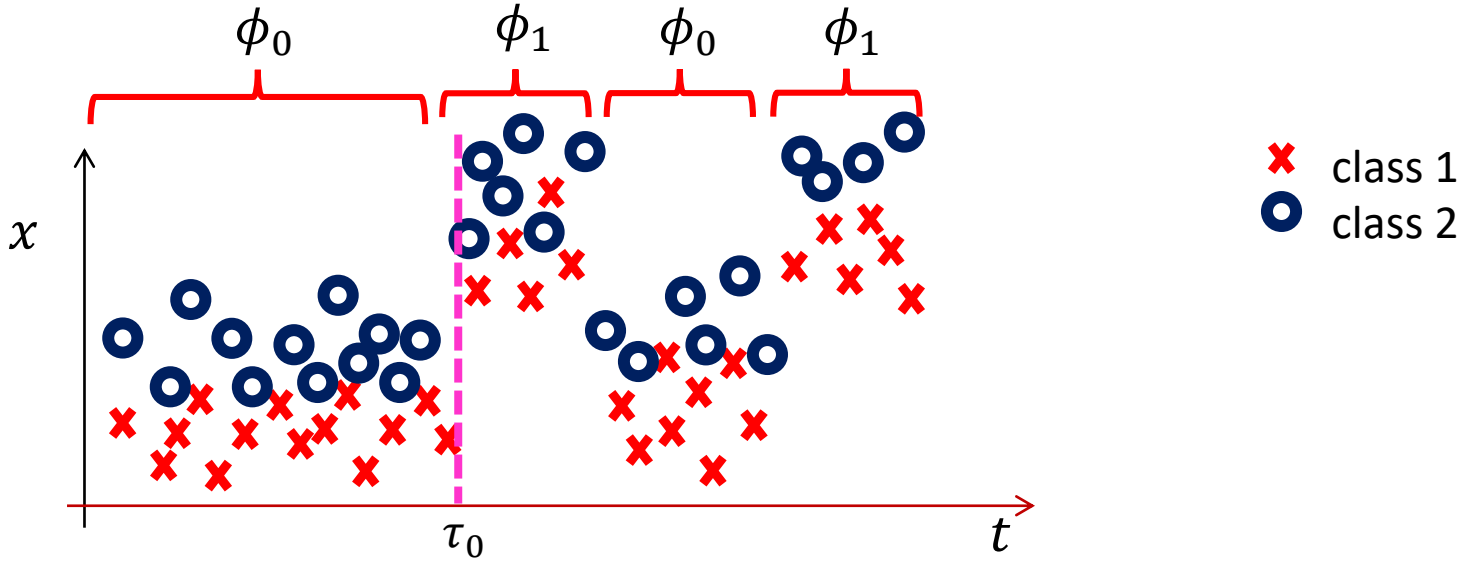


Drift Taxonomy: Time Evolution

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

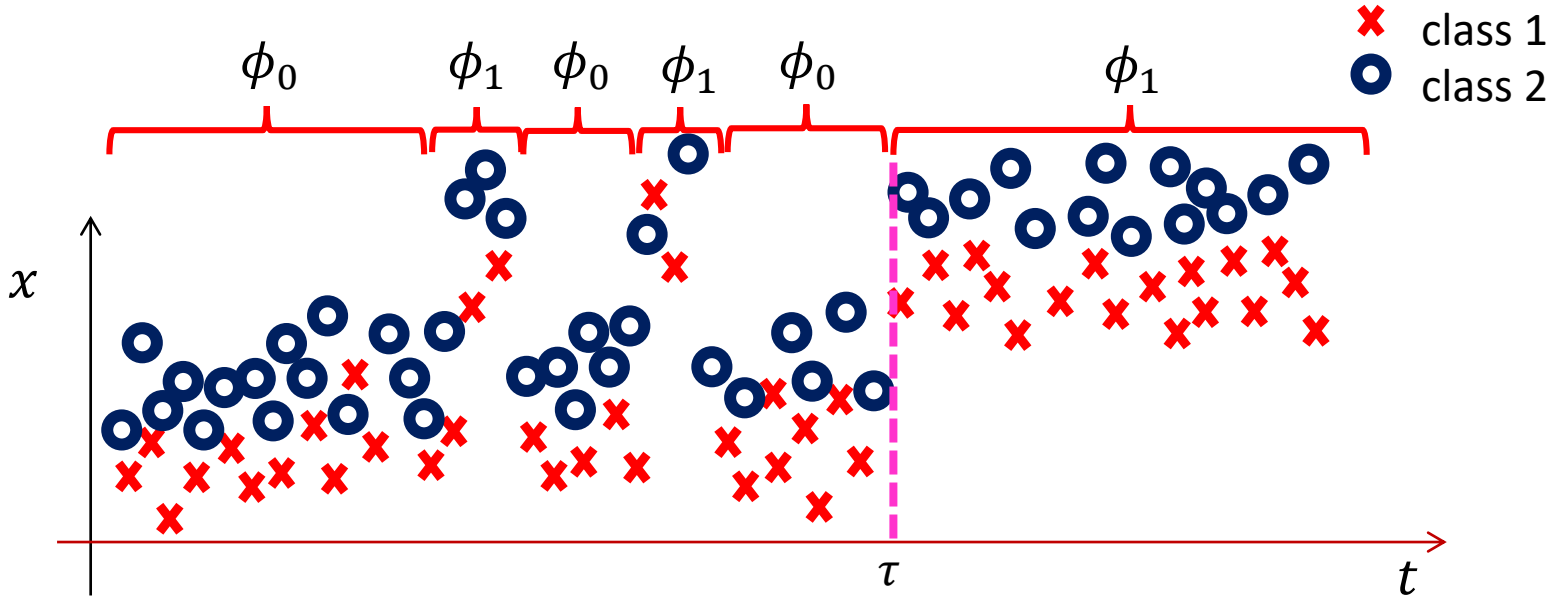


Drift Taxonomy: Time Evolution

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



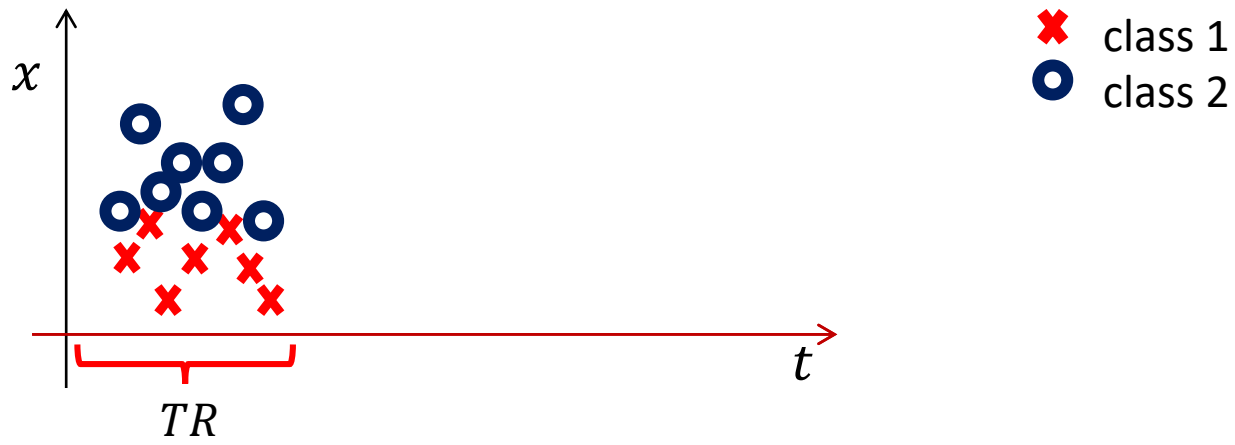
Is Concept Drift a Problem?

The need for Adaptation

Classification Over Datastreams

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

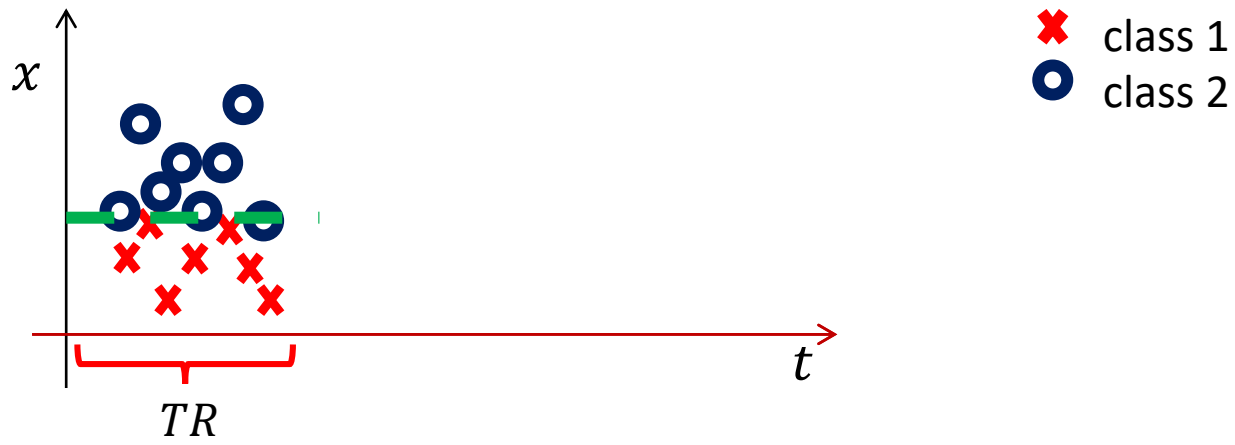
- The initial part of the stream is provided for training



Classification Over Datastreams

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

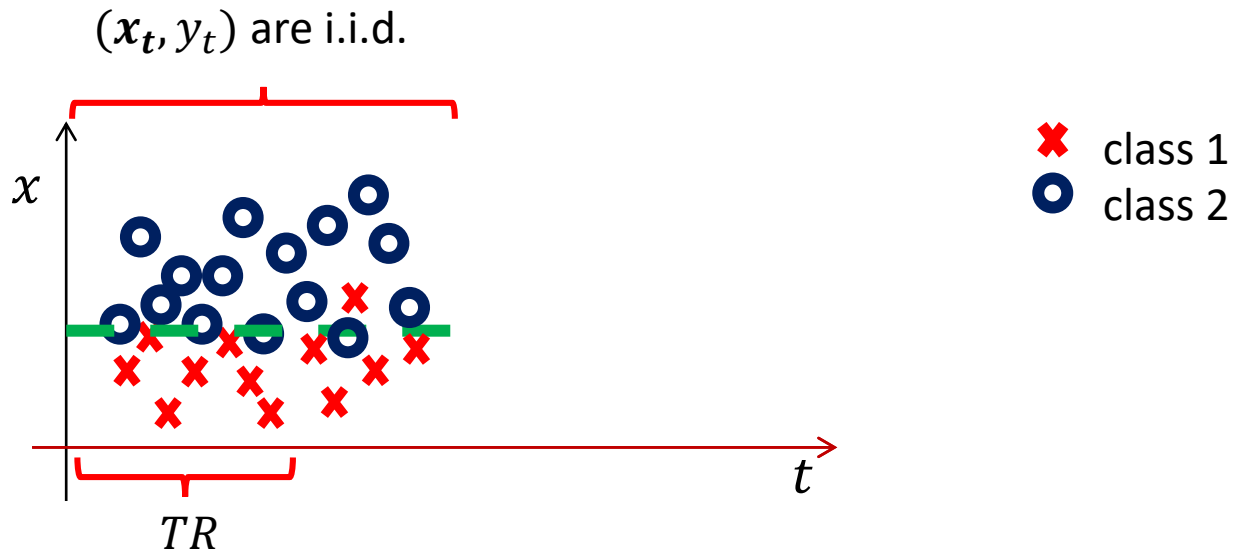
- The initial part of the stream is provided for training
- K is simply a threshold



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

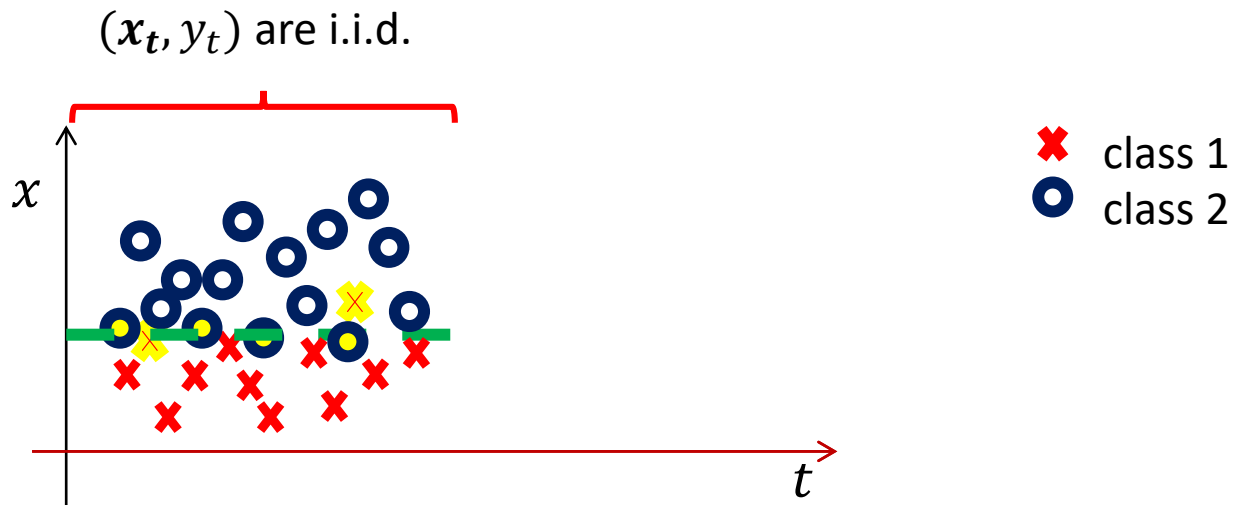


Classification Over Datastreams

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

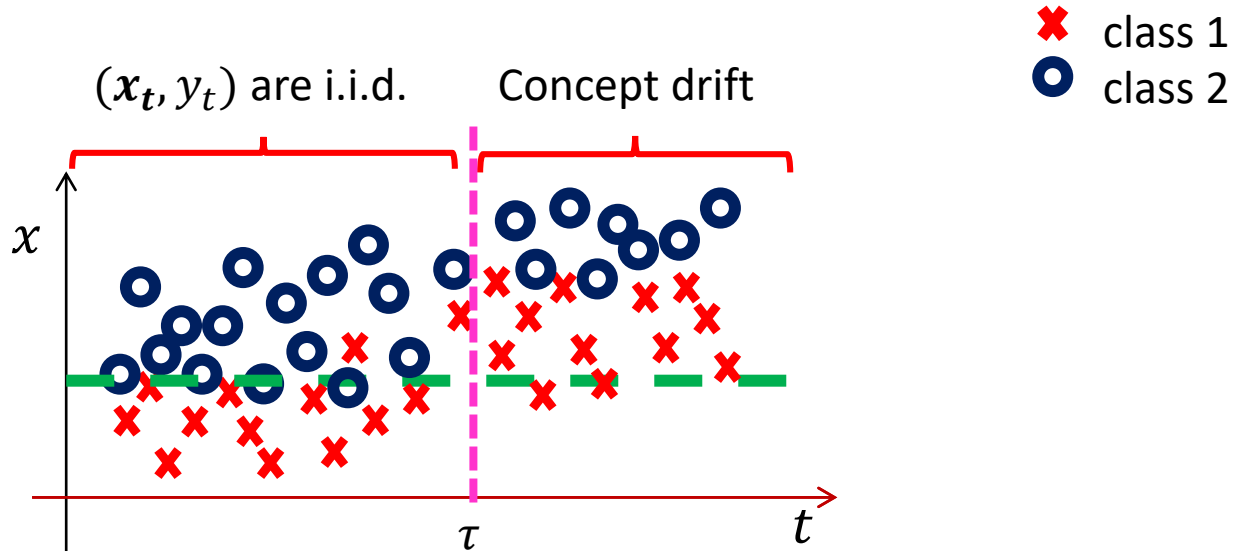
- The initial part of the stream is provided for training
- K is simply a threshold

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



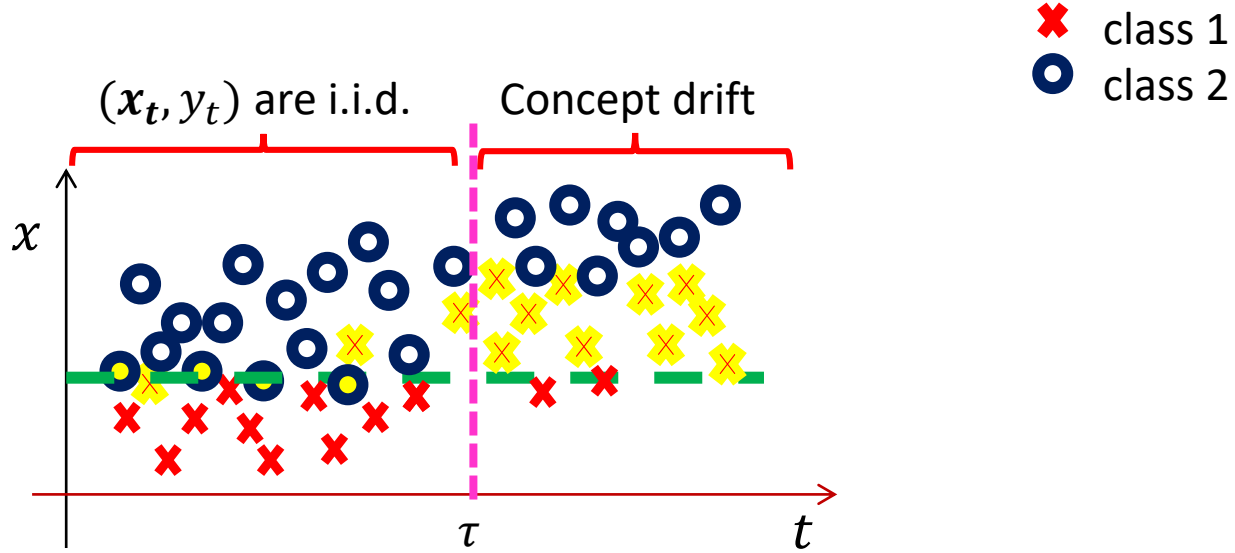
Classification Over Datastreams

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



Classification Over Datastreams

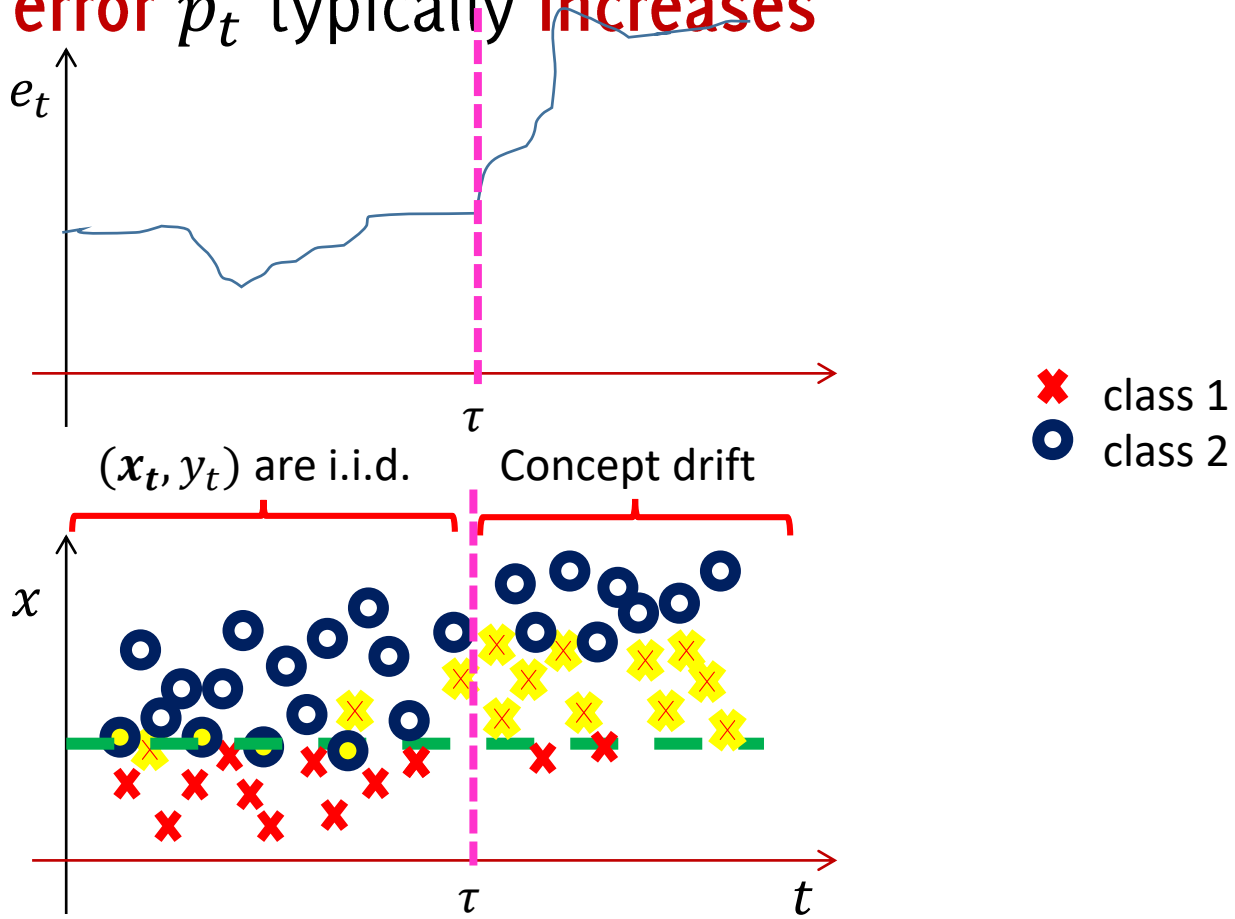
Unfortunately, when **concept drift occurs**, and ϕ changes, things can be terribly worst,



Classification Over Datastreams

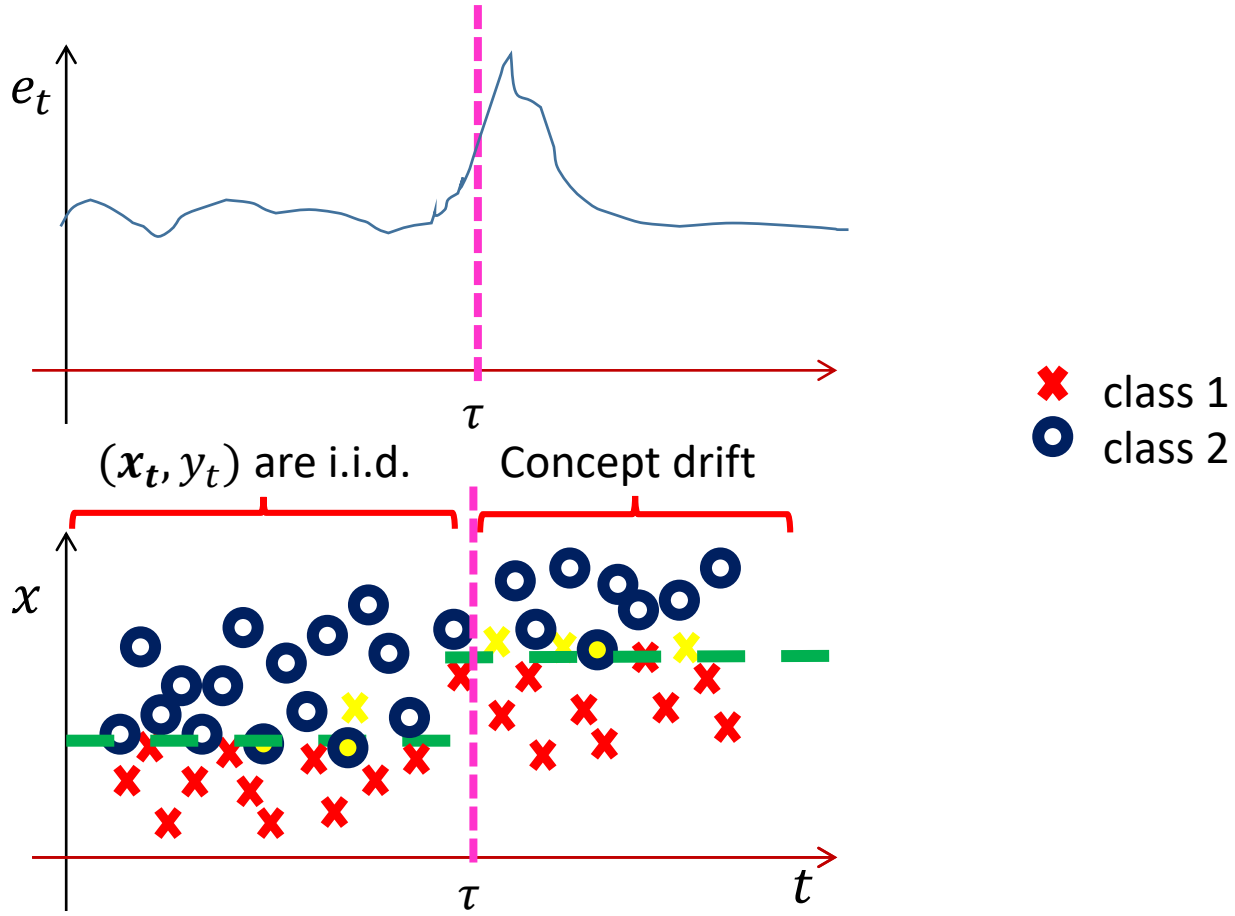
Unfortunately, when **concept drift occurs**, and ϕ changes, things can be terribly worst,

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



Need For Adaptation

Adaptation is needed to **preserve** classifier performance



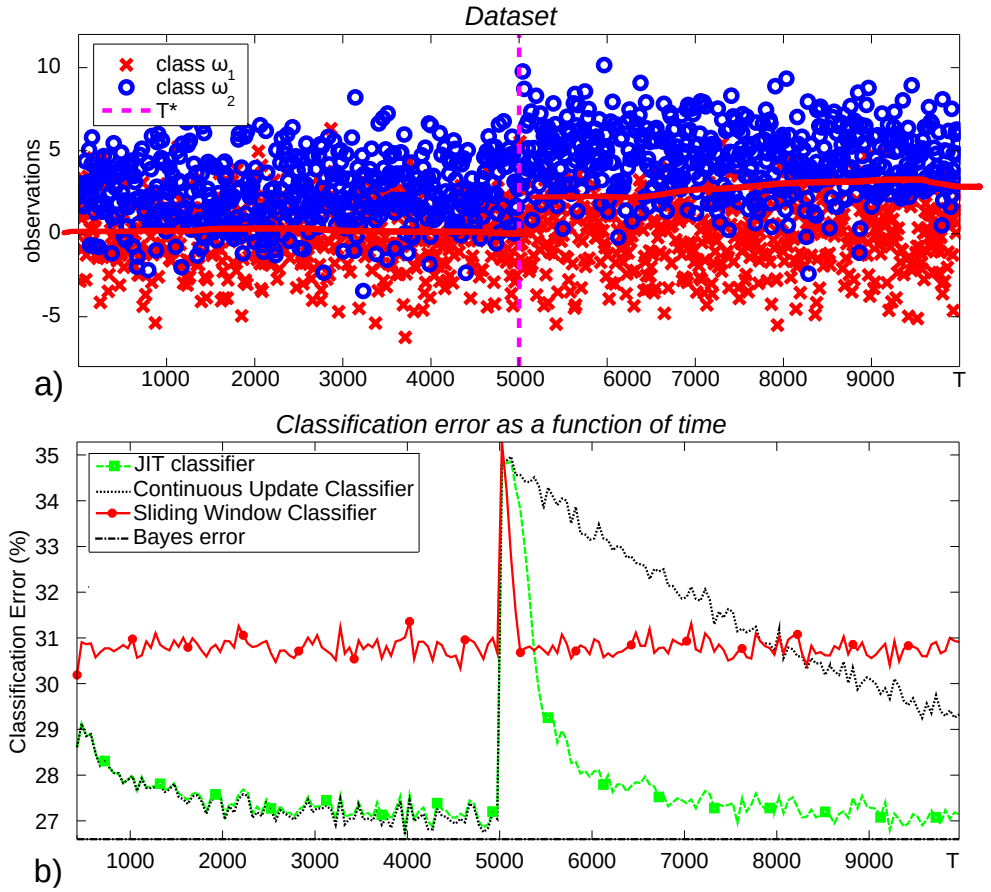
Adaptation

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

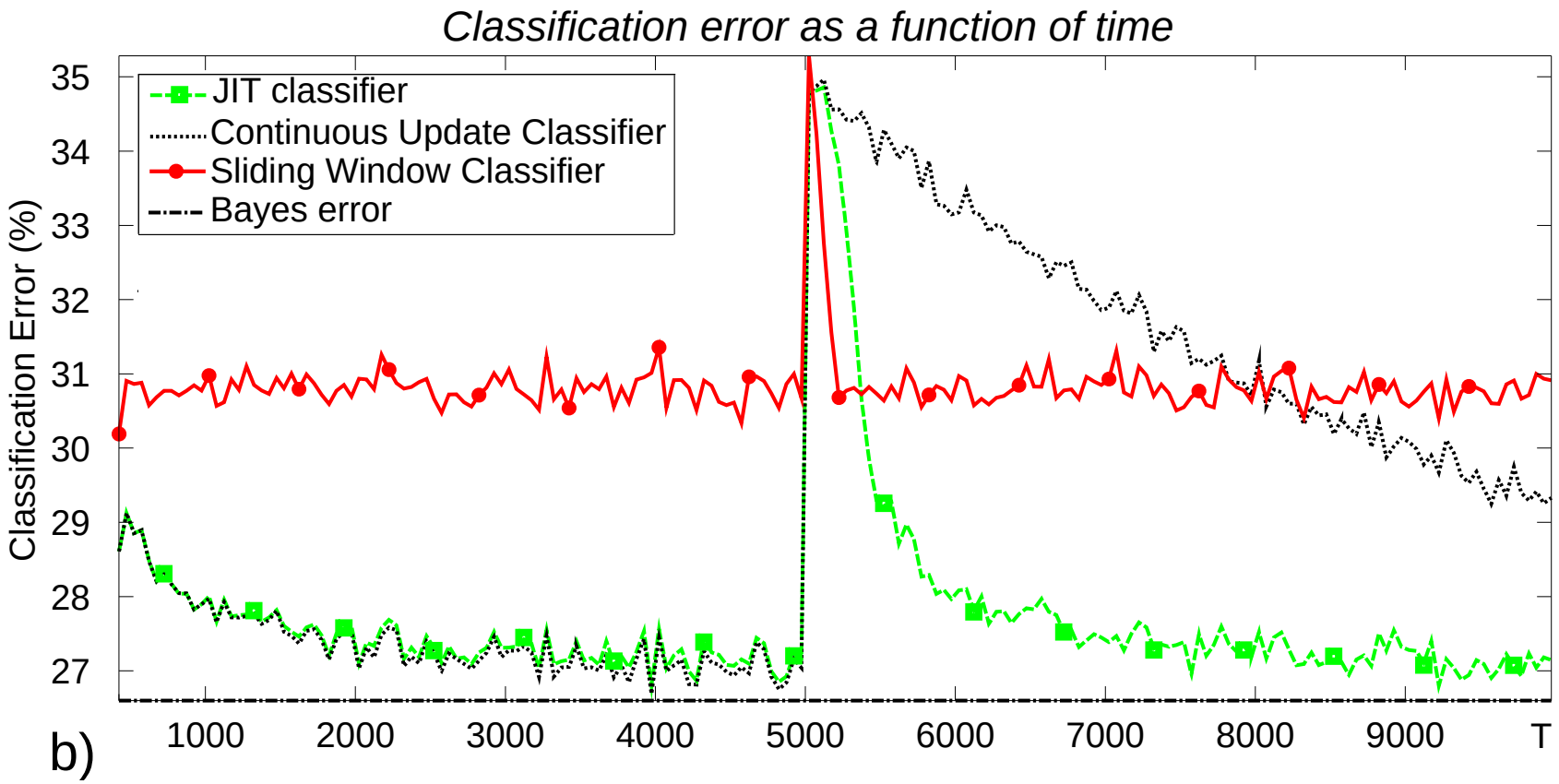


Boracchi, Trovò

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

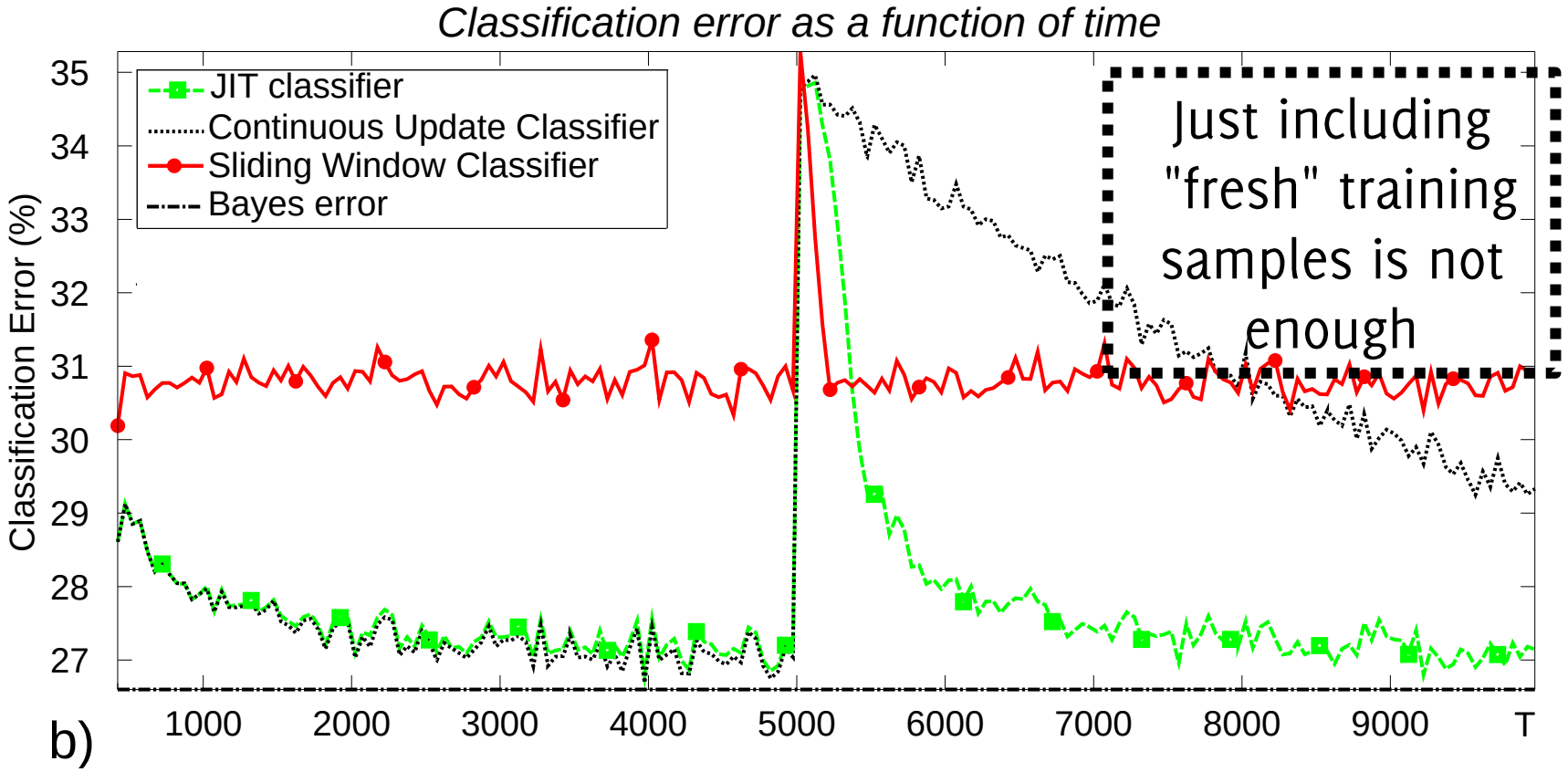


Alippi, C., Boracchi, G., Roveri, M. (2013). Just-in-time classifiers for recurrent concepts. *IEEE TNNLS* 620-634.

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

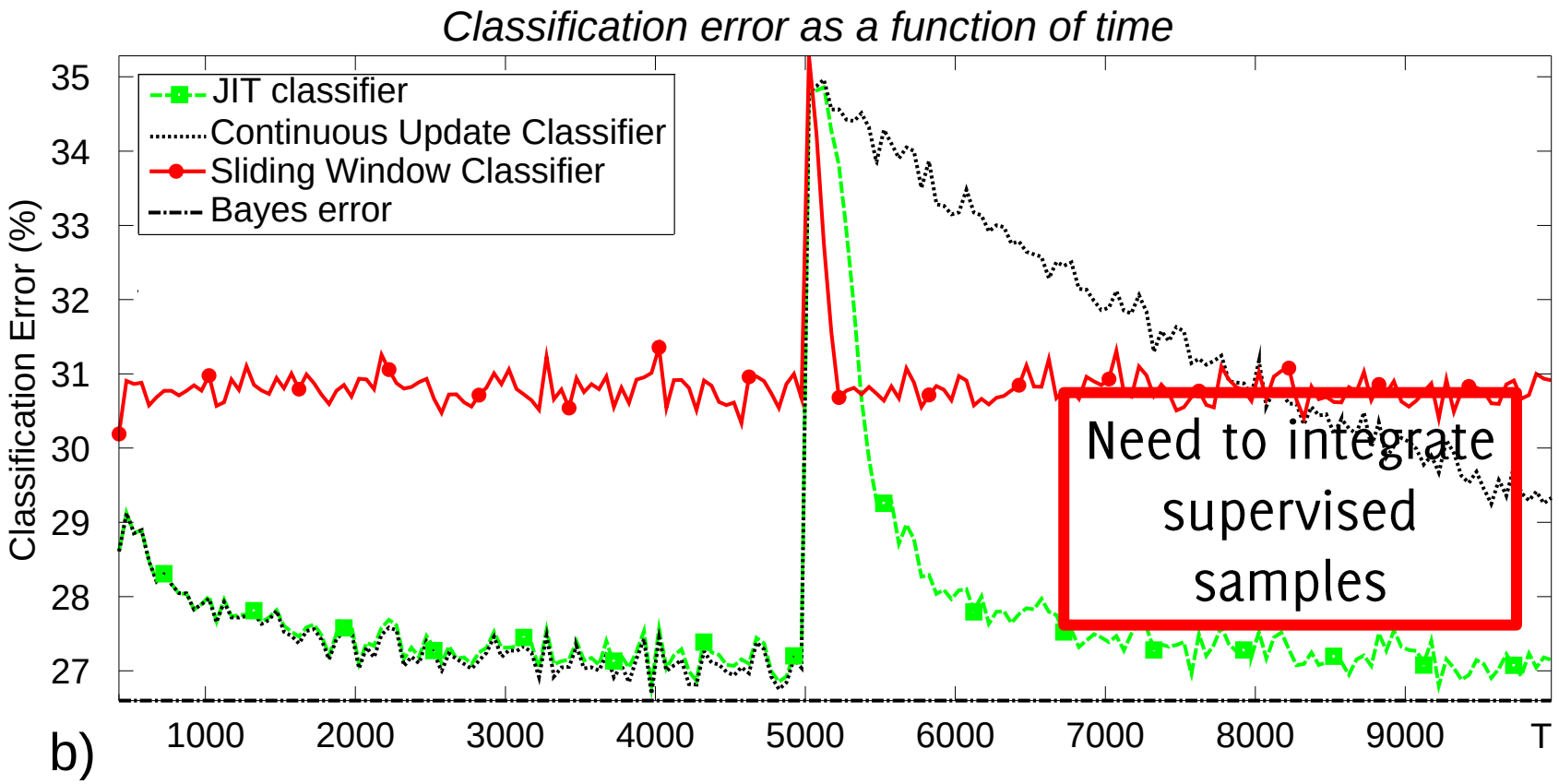


Alippi, C., Boracchi, G., Roveri, M. (2013). Just-in-time classifiers for recurrent concepts. *IEEE TNNLS* 620-634.

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

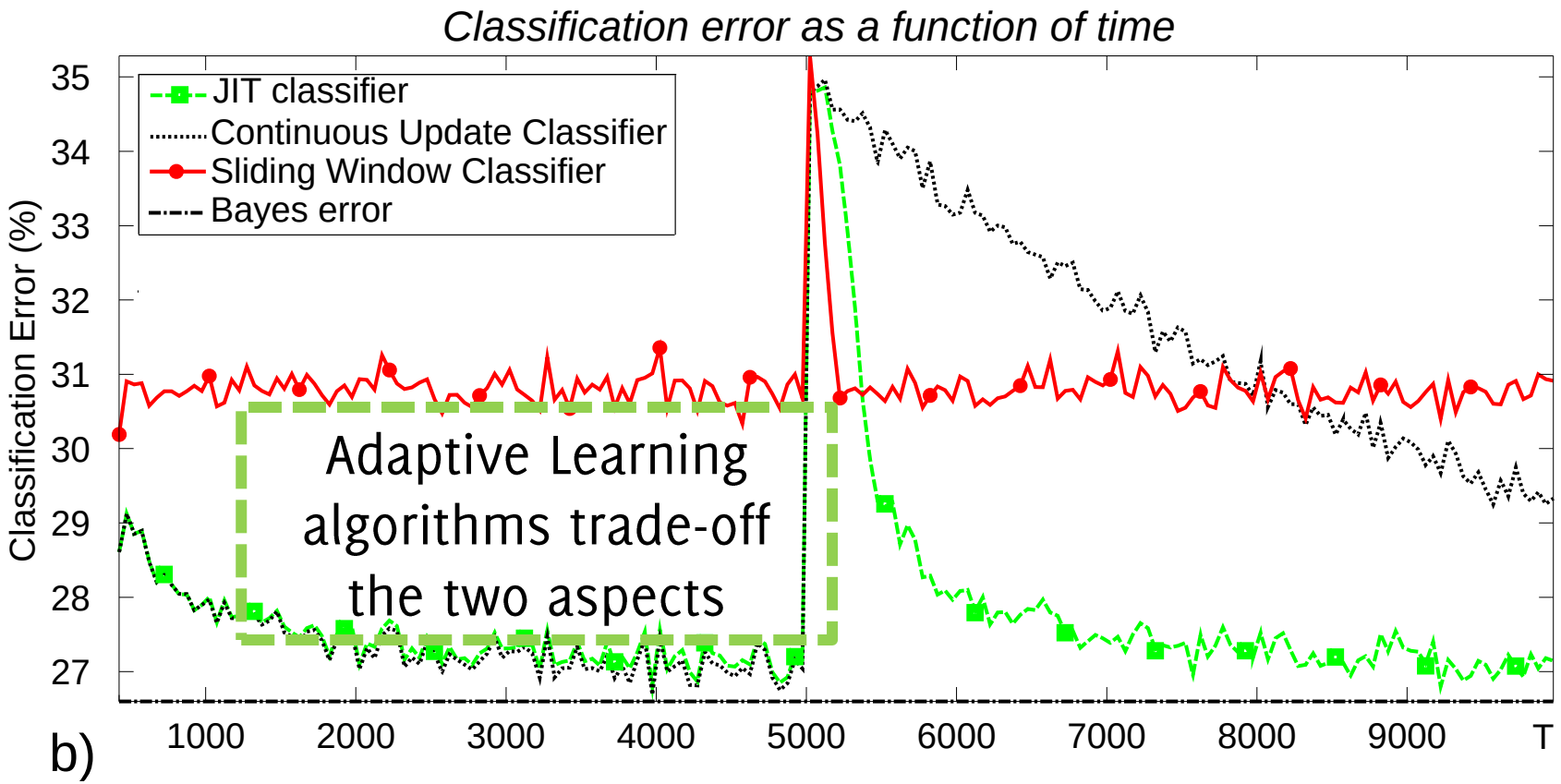


Alippi, C., Boracchi, G., Roveri, M. (2013). Just-in-time classifiers for recurrent concepts. *IEEE TNNLS* 620-634.

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



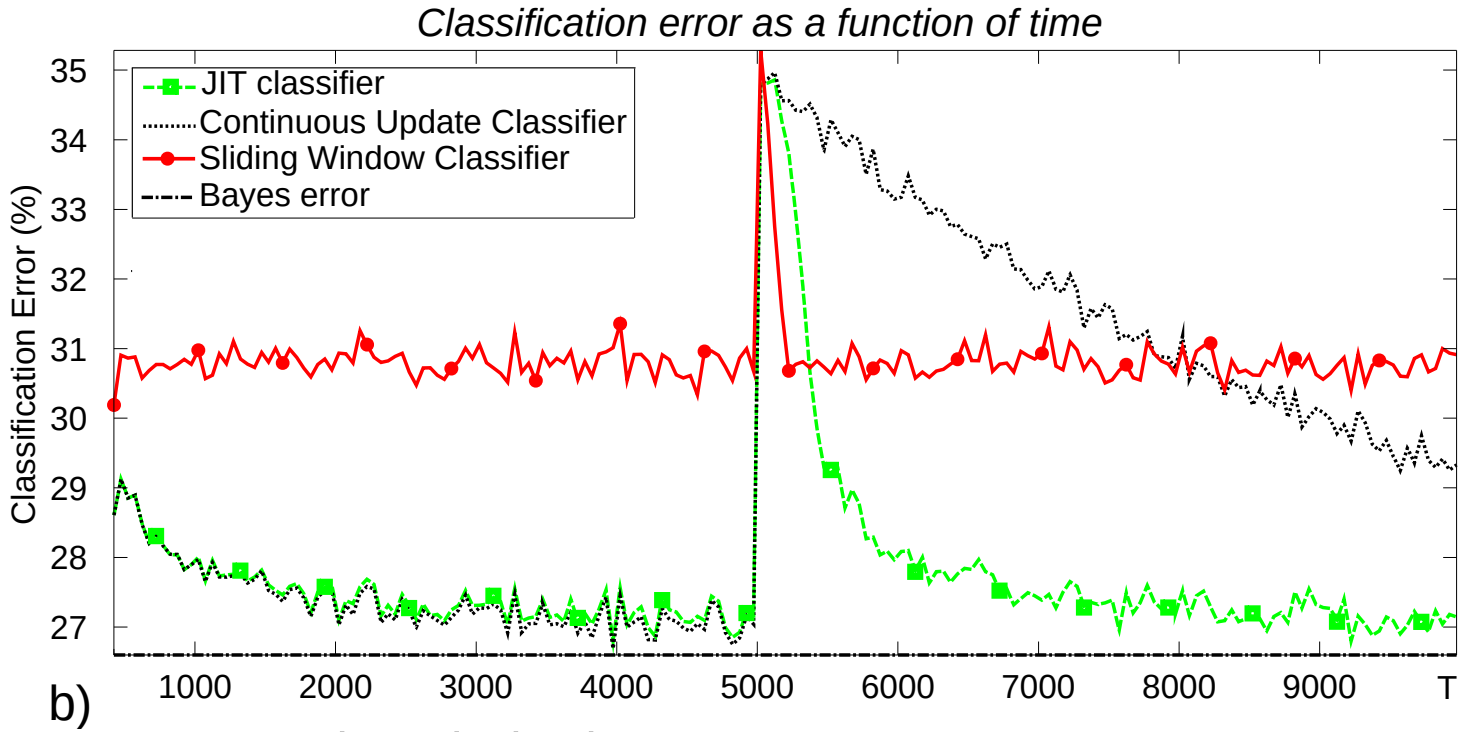
Alippi, C., Boracchi, G., Roveri, M. (2013). Just-in-time classifiers for recurrent concepts. *IEEE TNNLS* 620-634.

First Matlab Assignment

Get the first matlab snippet

… develop and test

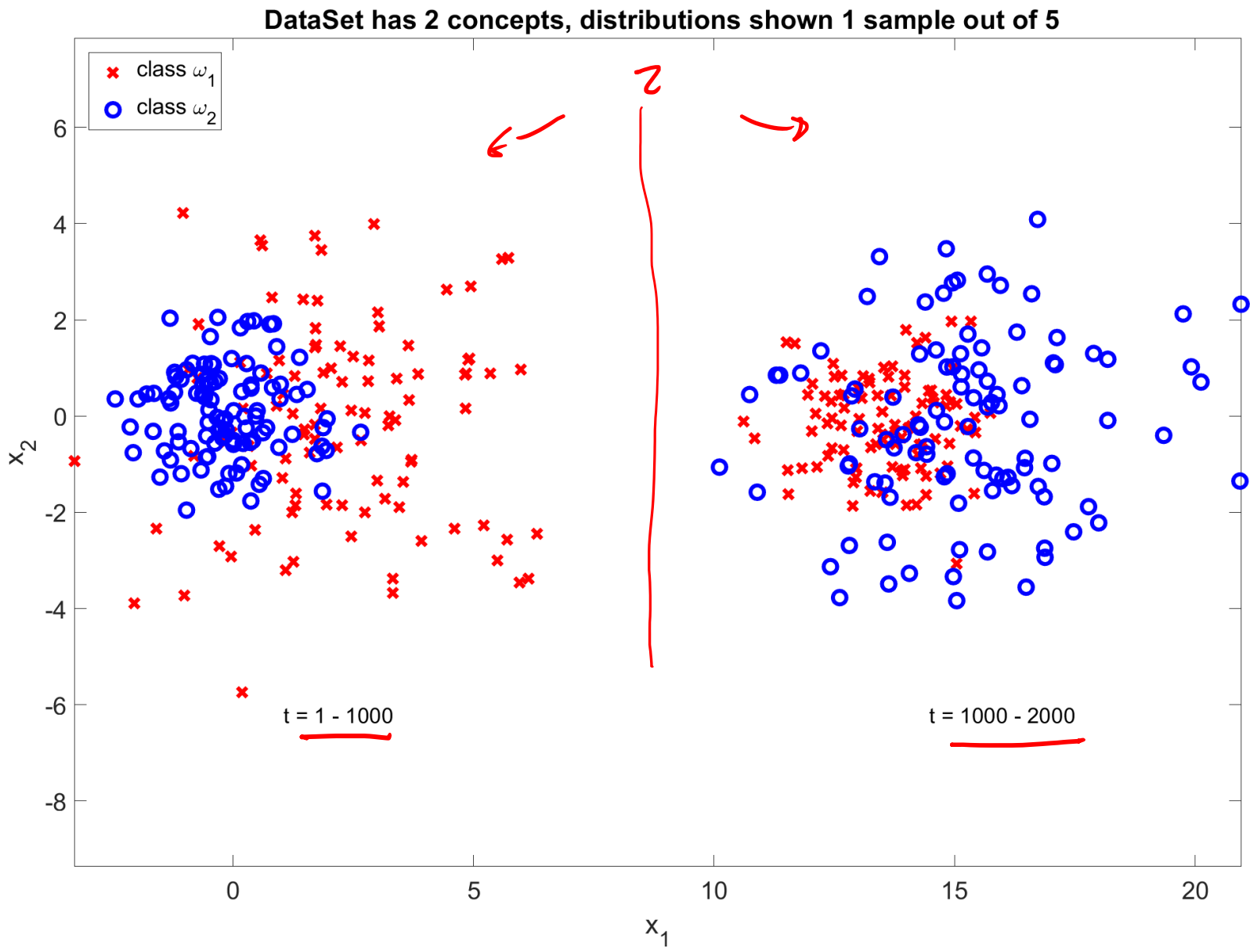
- An adaptive classifier that is always updated every time a feedback is provided (black)
- An adaptive classifier that is always updated over the latest ν training samples (red)
- A classifier that is never updated (not shown)



b)

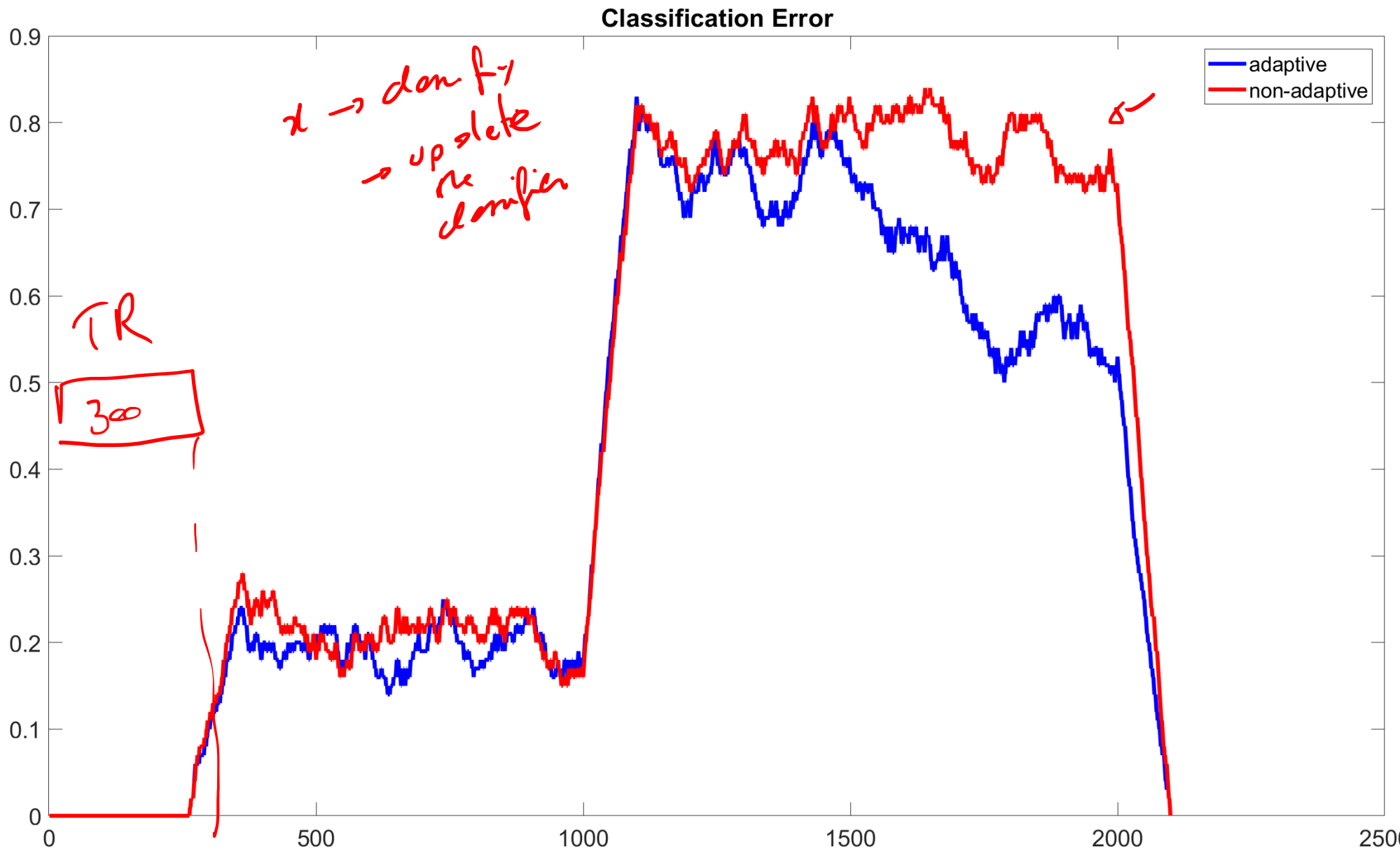
- Compute and display the classification error over the whole datastream

Generate a dataset like this one



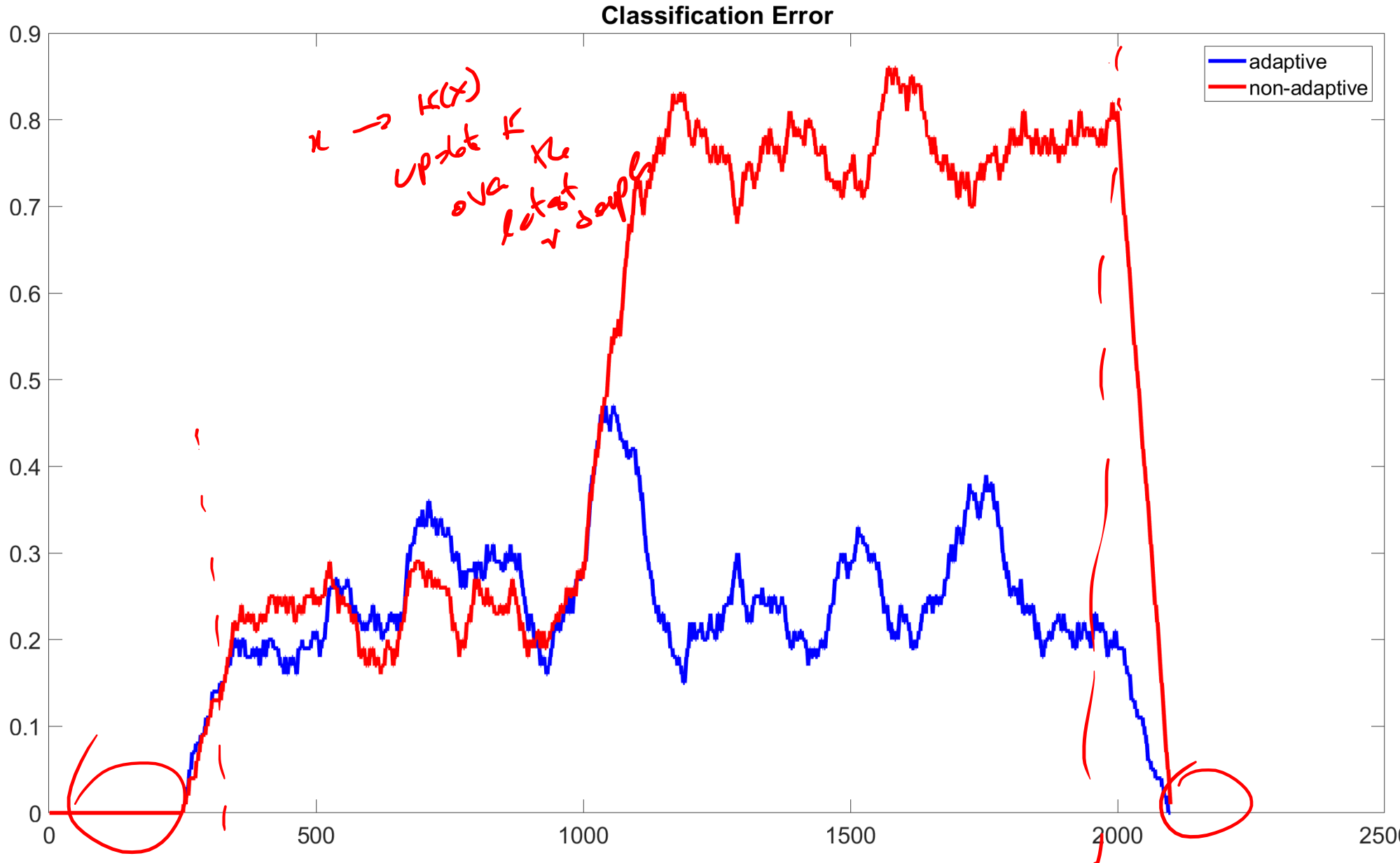
abrupt real drift

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 (Change Detection Tests / Anomaly Detection Algorithms) **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 recently data generated after the change, thus that are coherent with the new concept

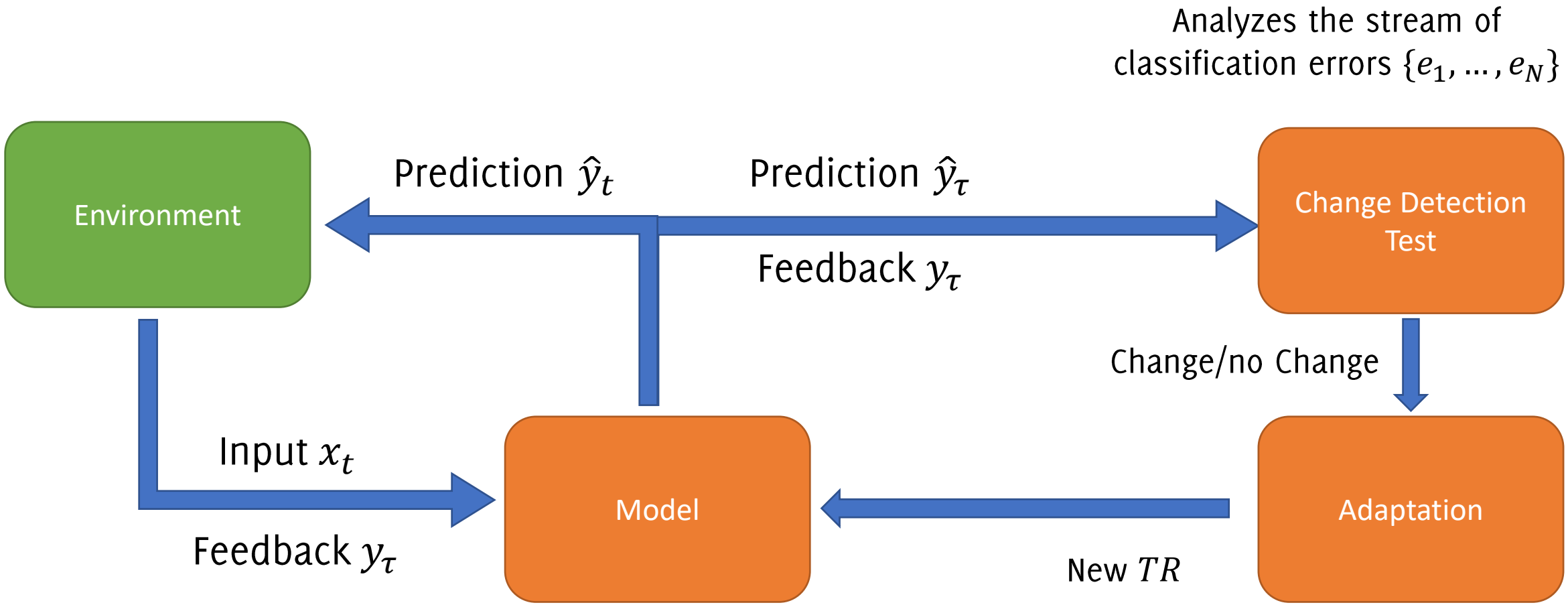
Pro:

- 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)

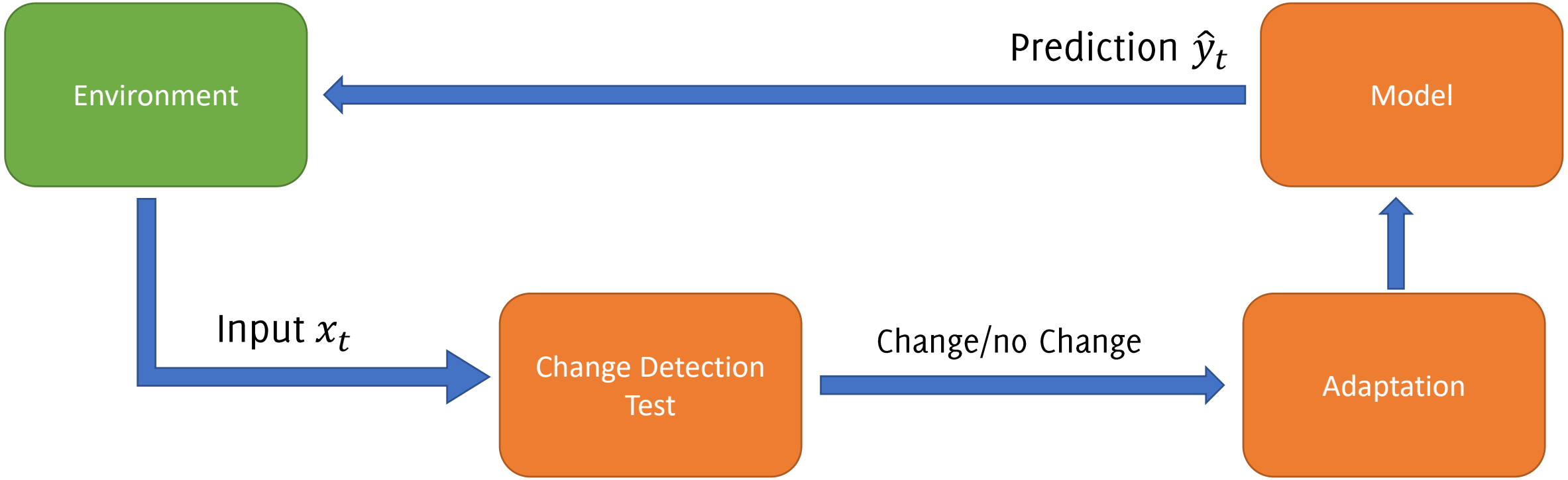
Pro:

- 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

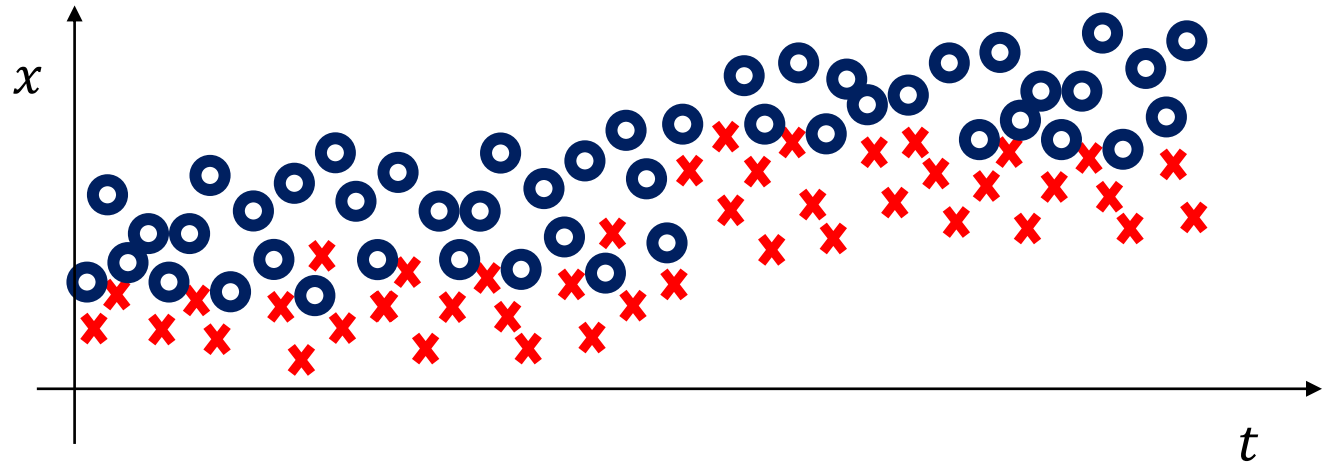
Monitoring Input Distribution



Boracchi, Trovò

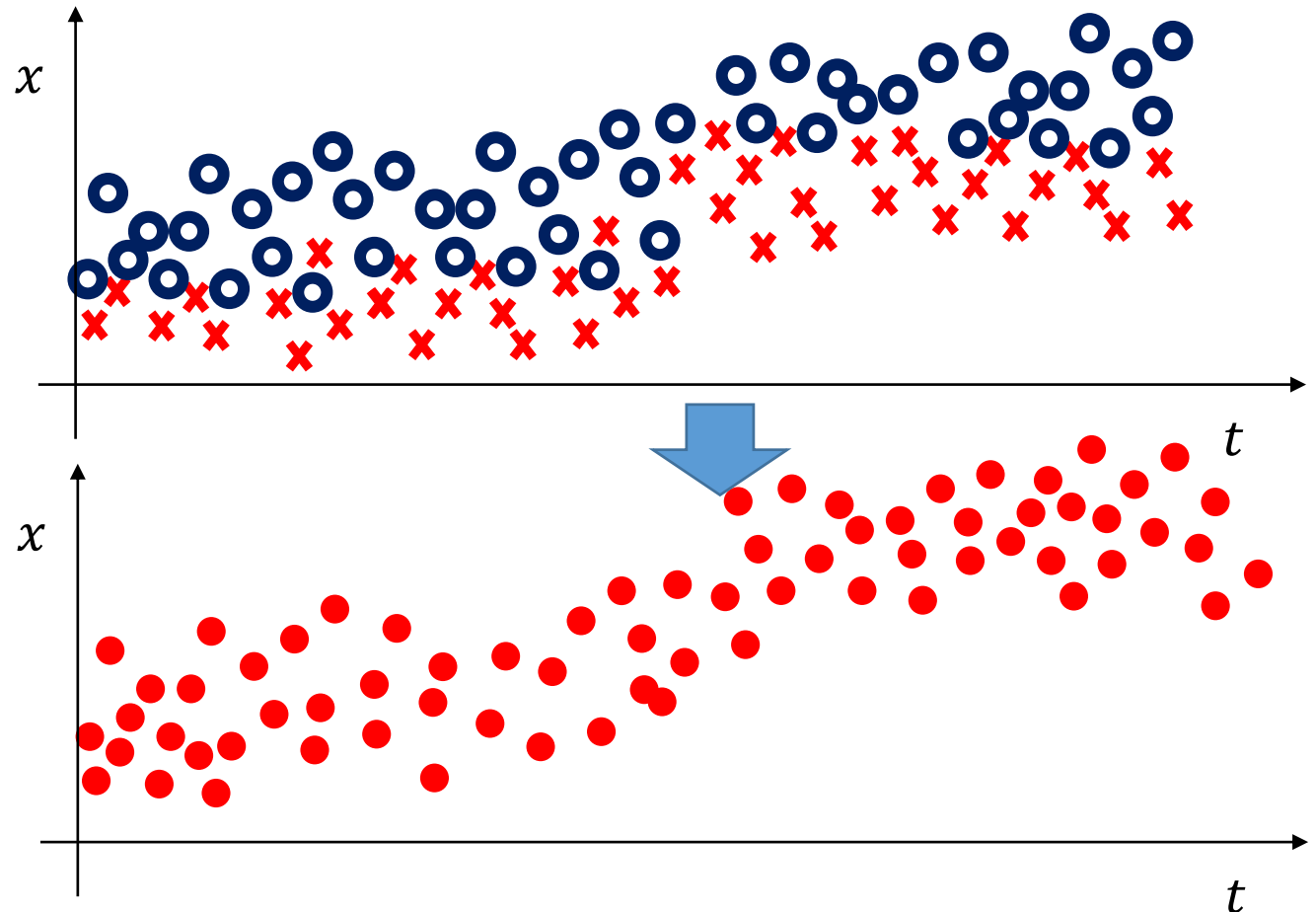
Monitoring Input Distribution

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



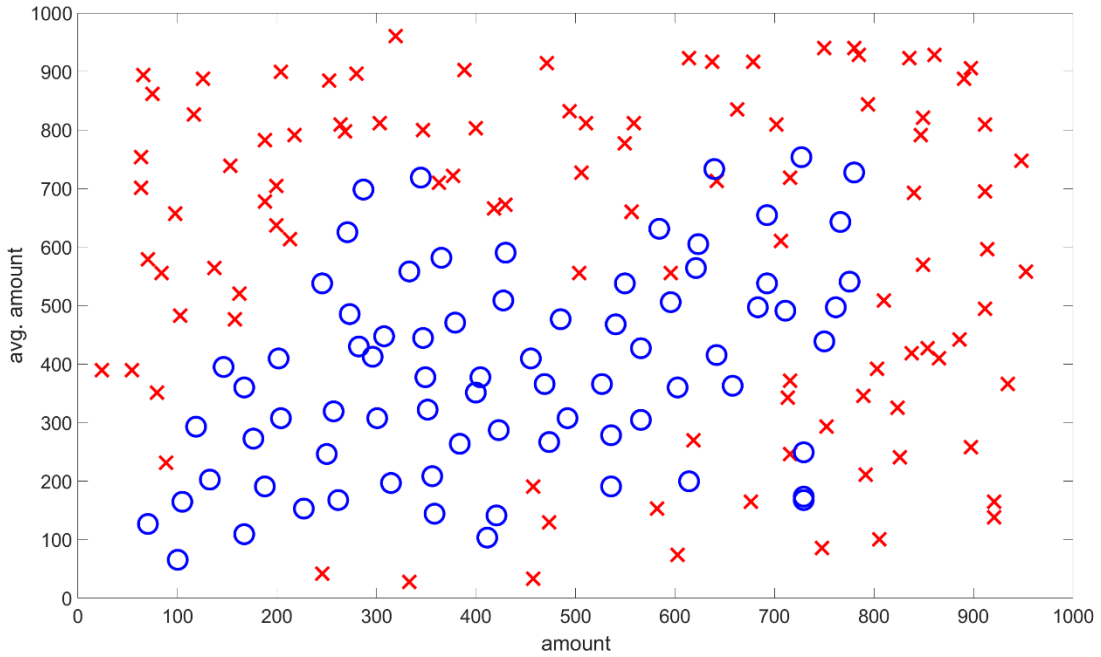
Monitoring Input Distribution

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

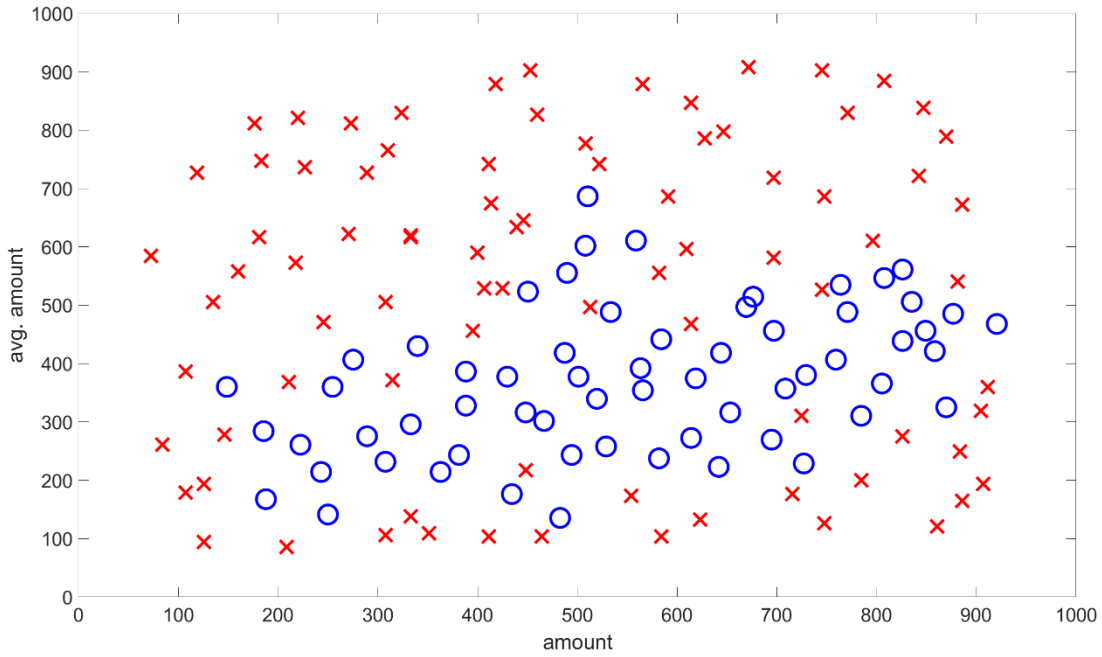


Monitoring Input Distribution

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



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



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 (e.g. classes swap)
- 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 is 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

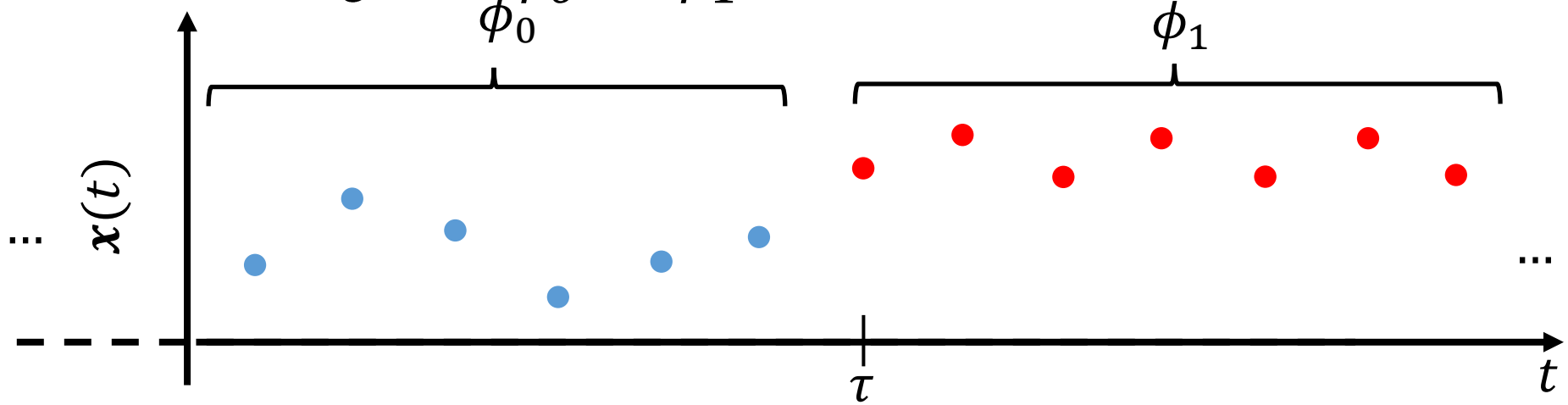
Often, the change-detection problem **boils down to:**

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: $\phi_o \rightarrow \phi_1$



racchi, Trovò

Change-Detection in a Statistical Framework

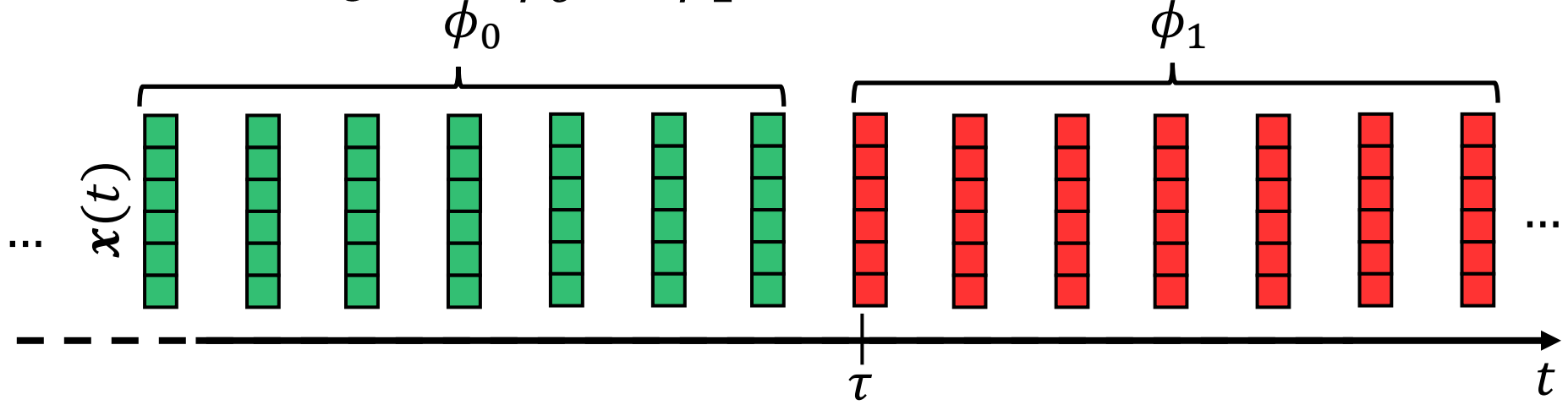
Often, the change-detection problem **boils down to:**

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 $\phi_0 \neq \phi_1$

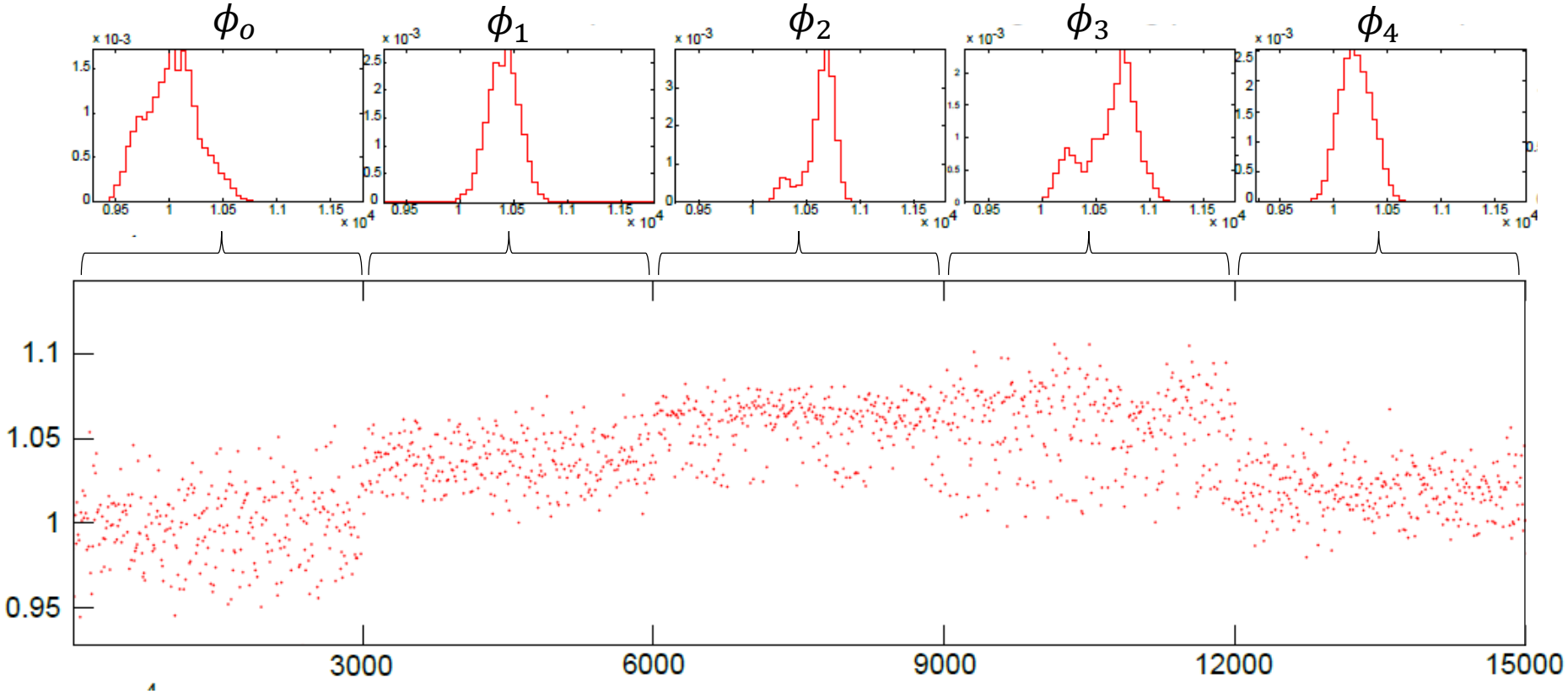
We denote such change as: $\phi_o \rightarrow \phi_1$



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.

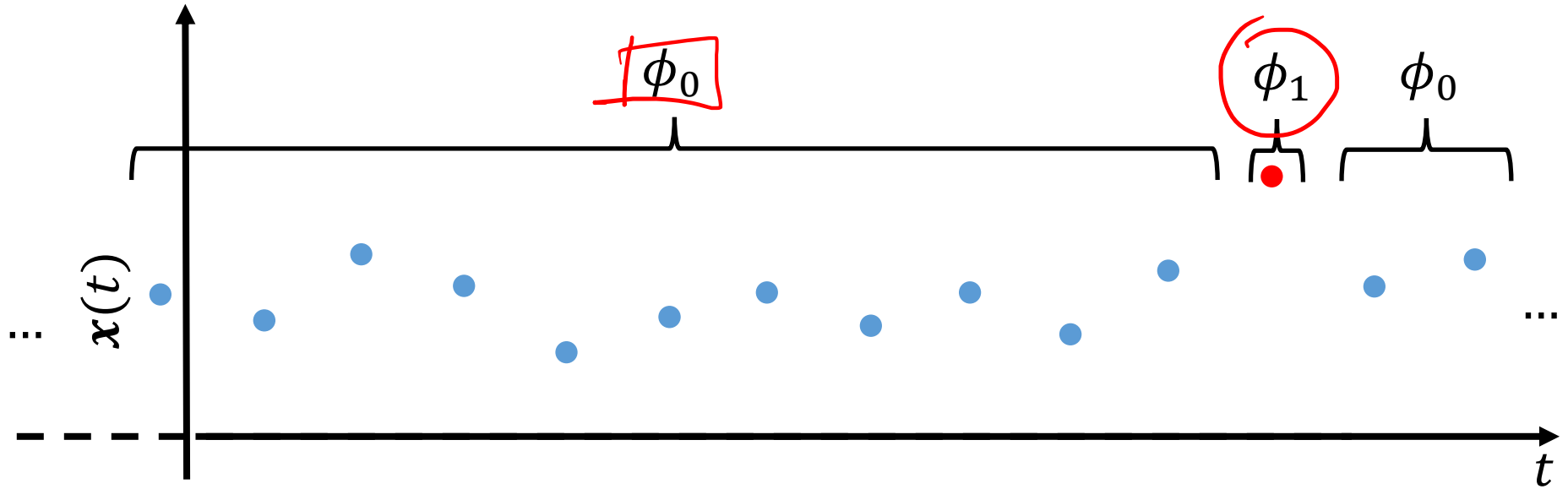
Anomaly-Detection in a statistical framework

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.,

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



racchi, Trovò

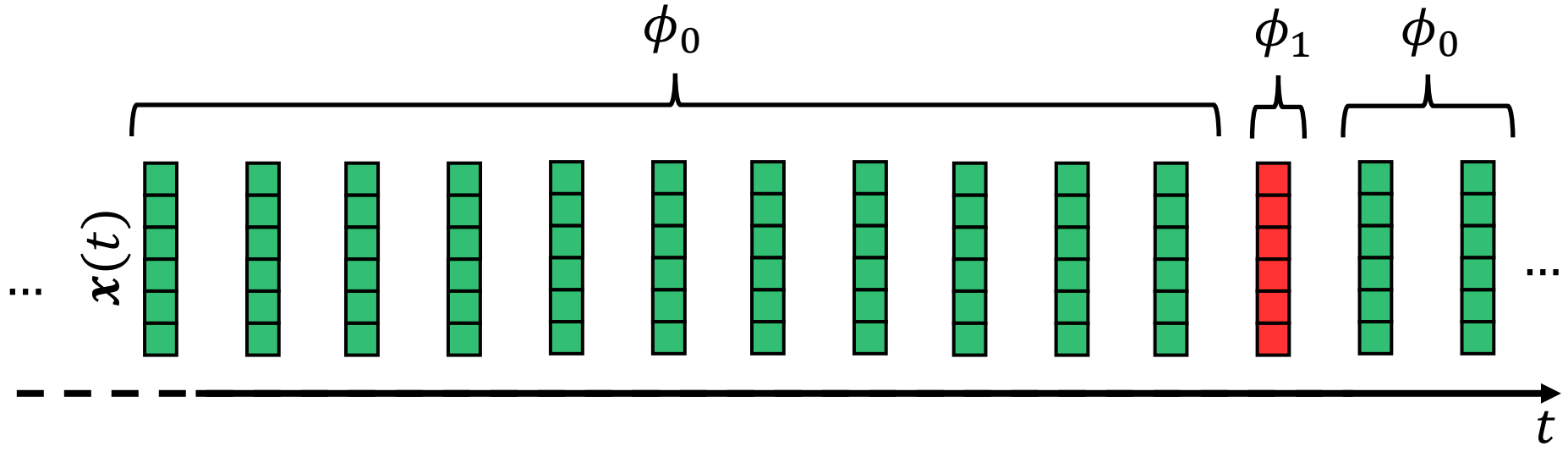
Anomaly-Detection in a statistical framework

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.,

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



racchi, Trovò

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.

ϕ_0

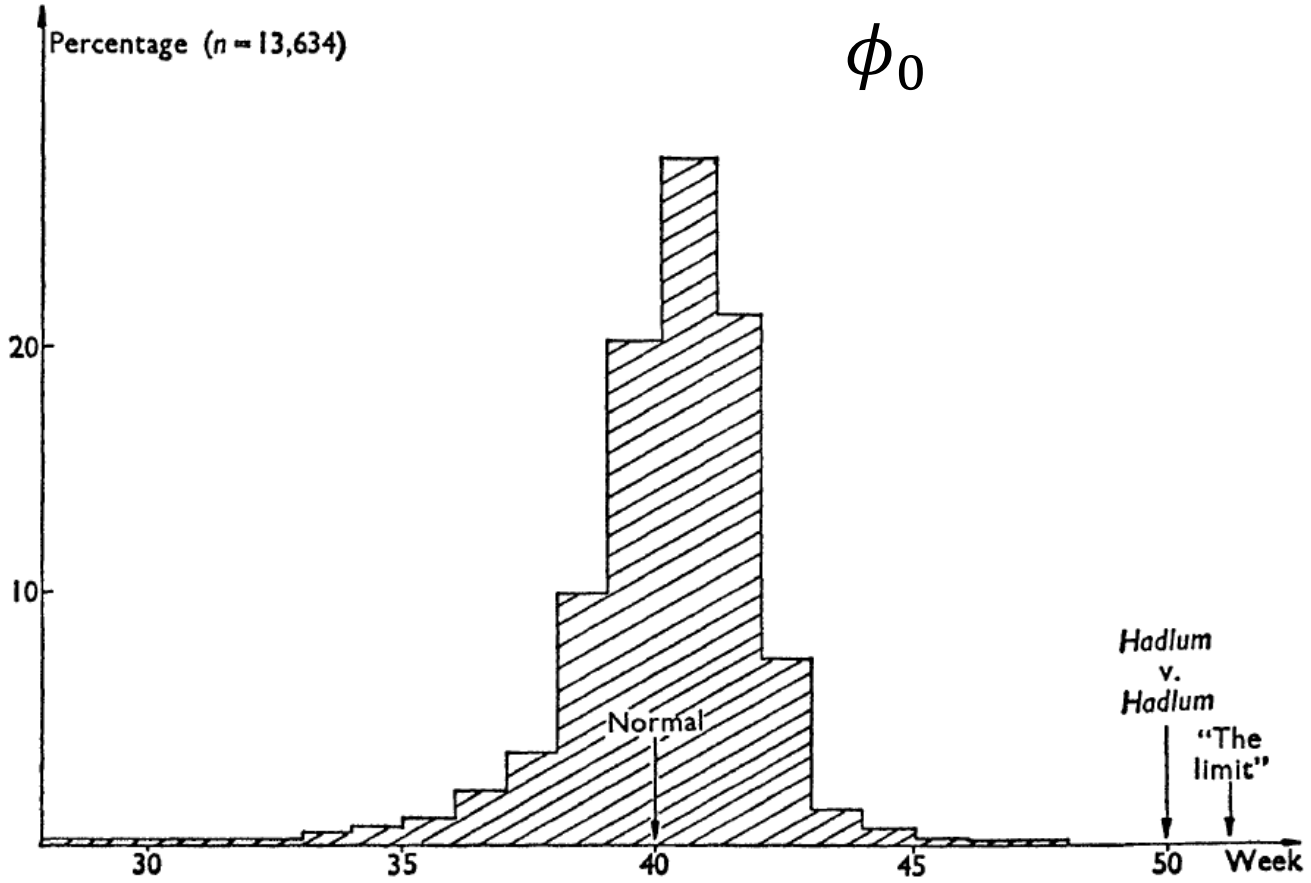


FIG. 1. Distribution of human gestation periods.

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.

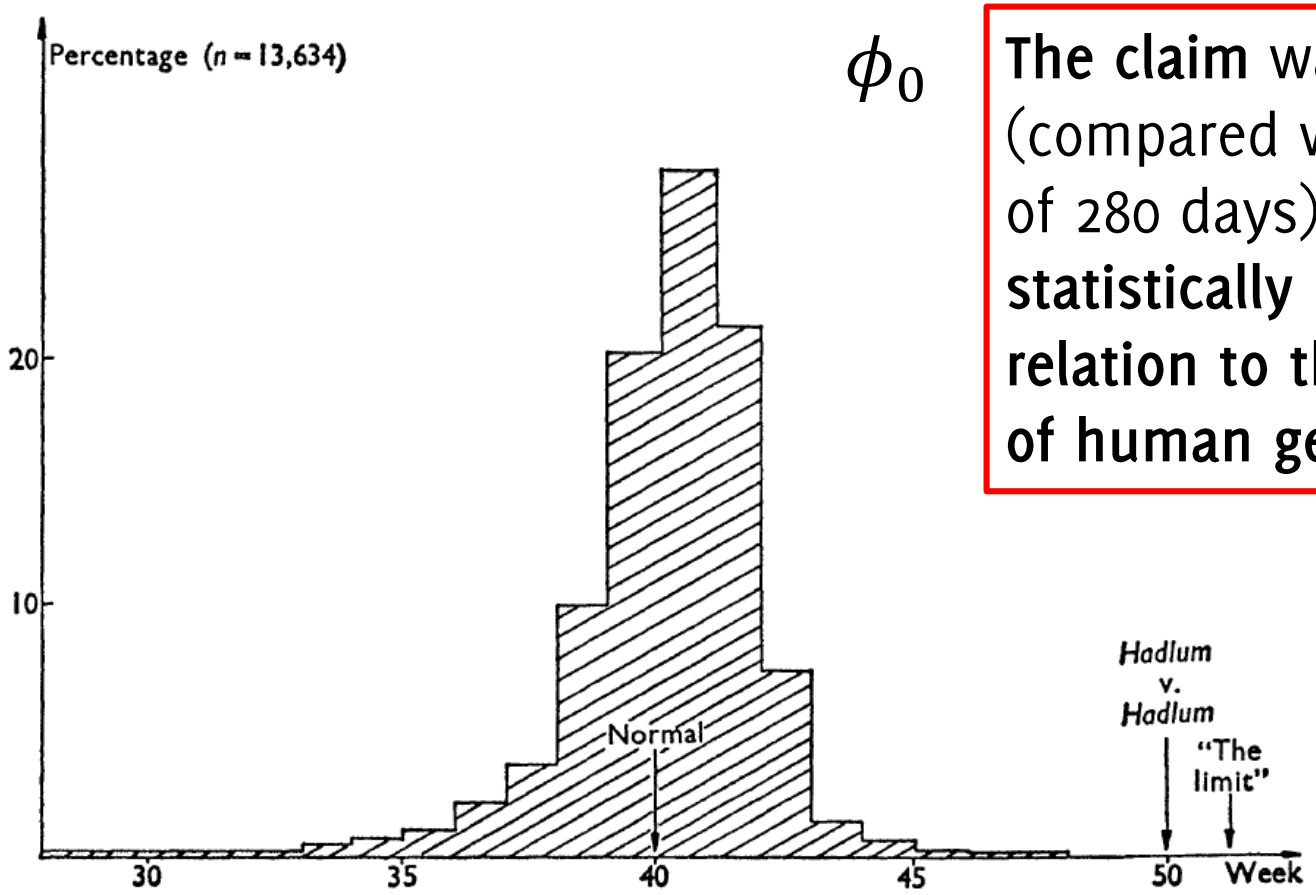


FIG. 1. Distribution of human gestation periods.

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
ation periods.

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

Tutorial in 2010 SIAM International Conference on Data Mining

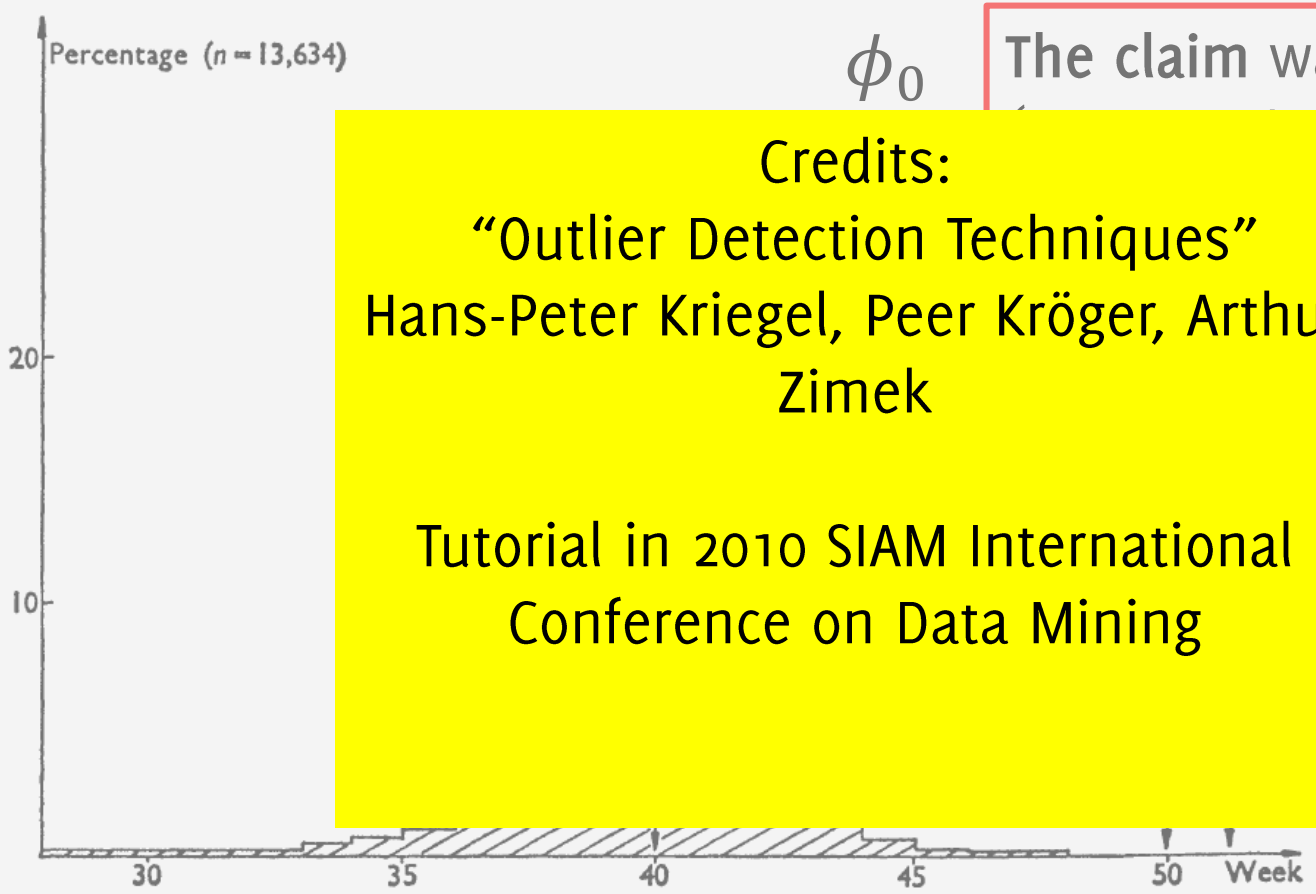
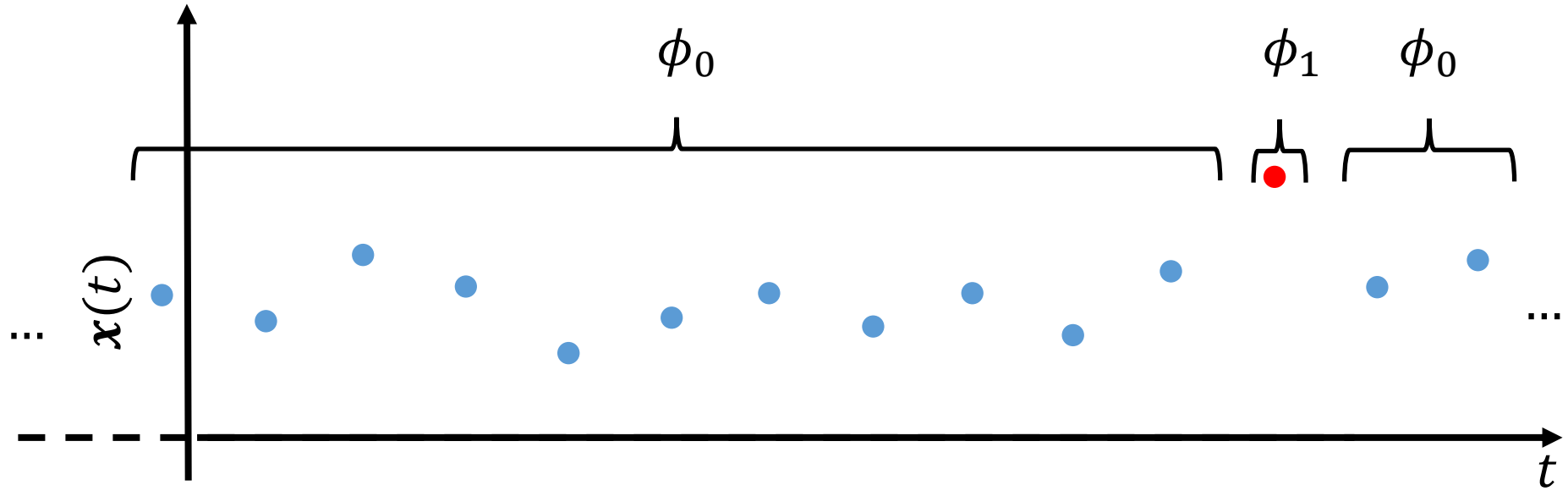


FIG. 1. Distribution of human gestation periods.

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

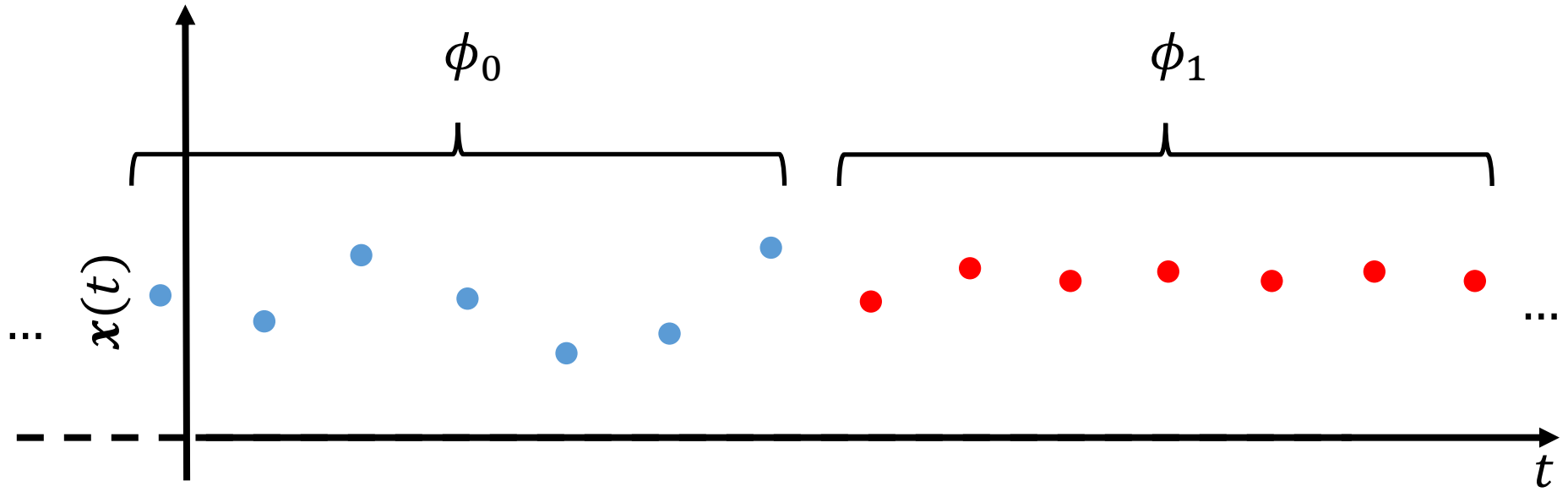
Process changes vs anomalies

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

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.

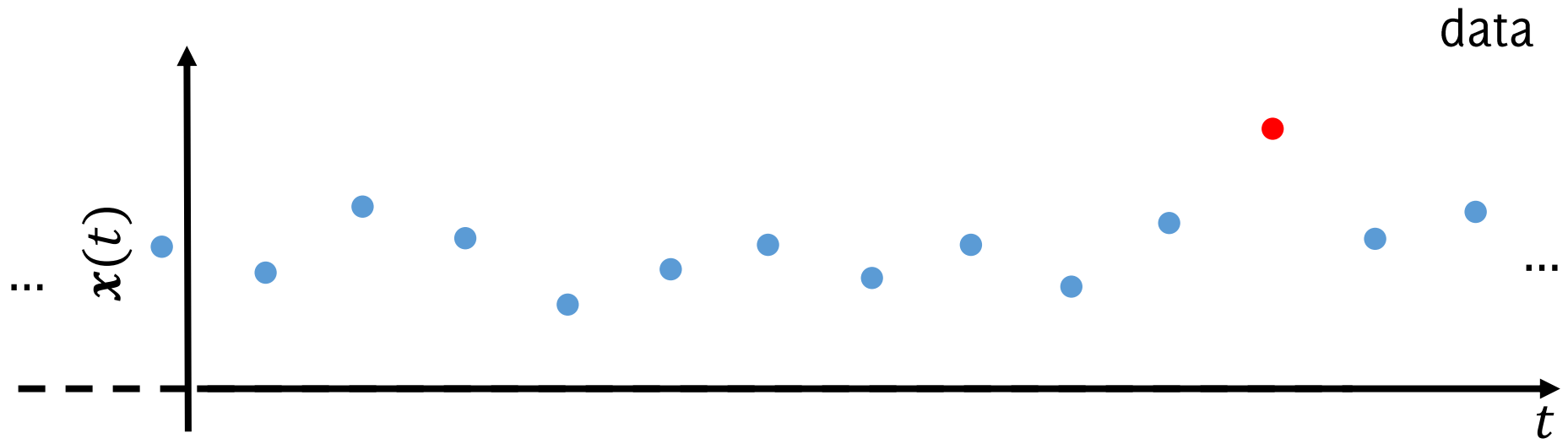
E.g.: The Malhanobis distance

Change-detection algorithms:

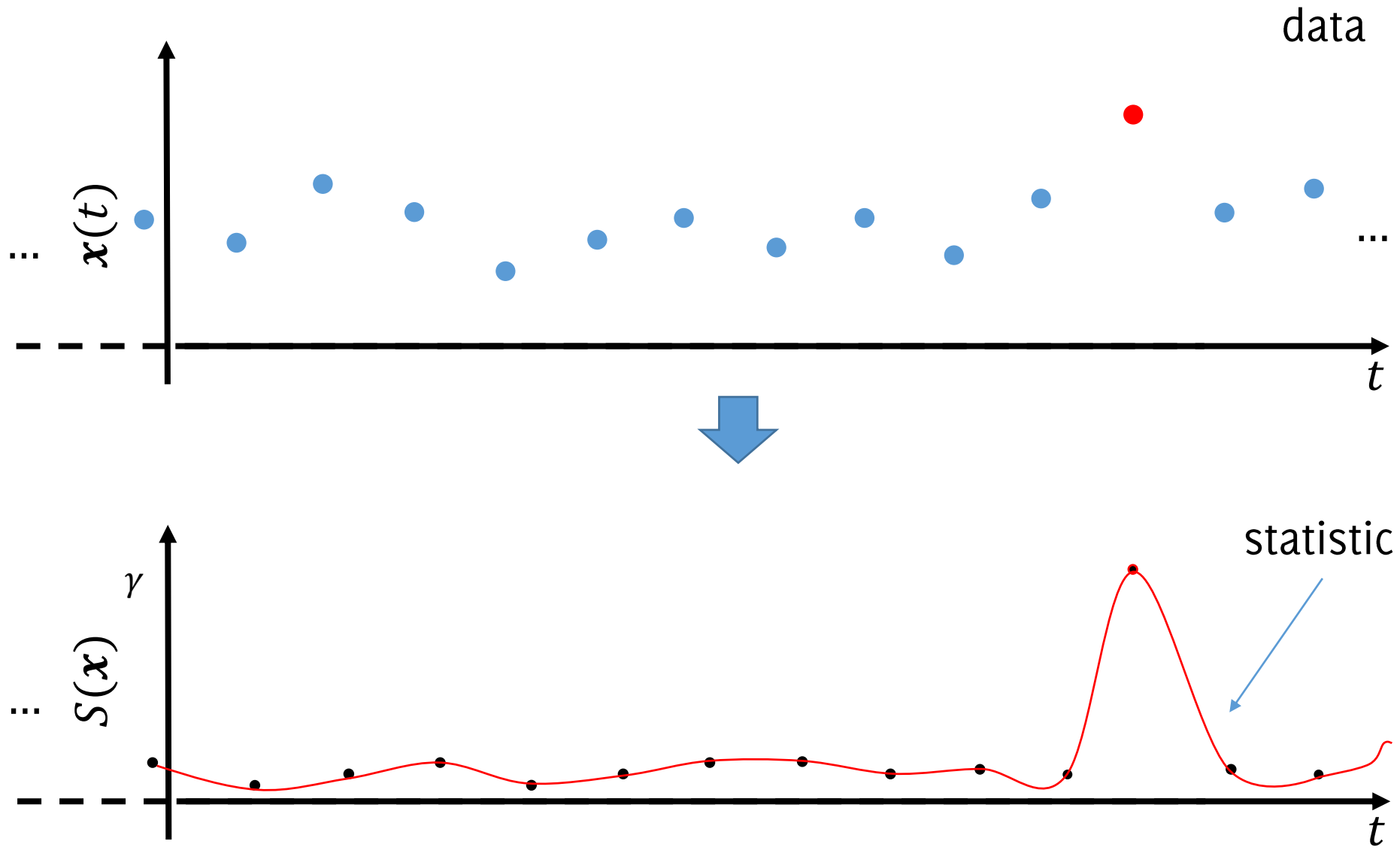
Statistics and decision rules are **sequential**, as they make a decision considering --in principle-- all the data received so far. Integrating information over time makes these algorithms able to detect subtle changes as well.

E.g.: The running average of a scalar input

The Typical Solutions

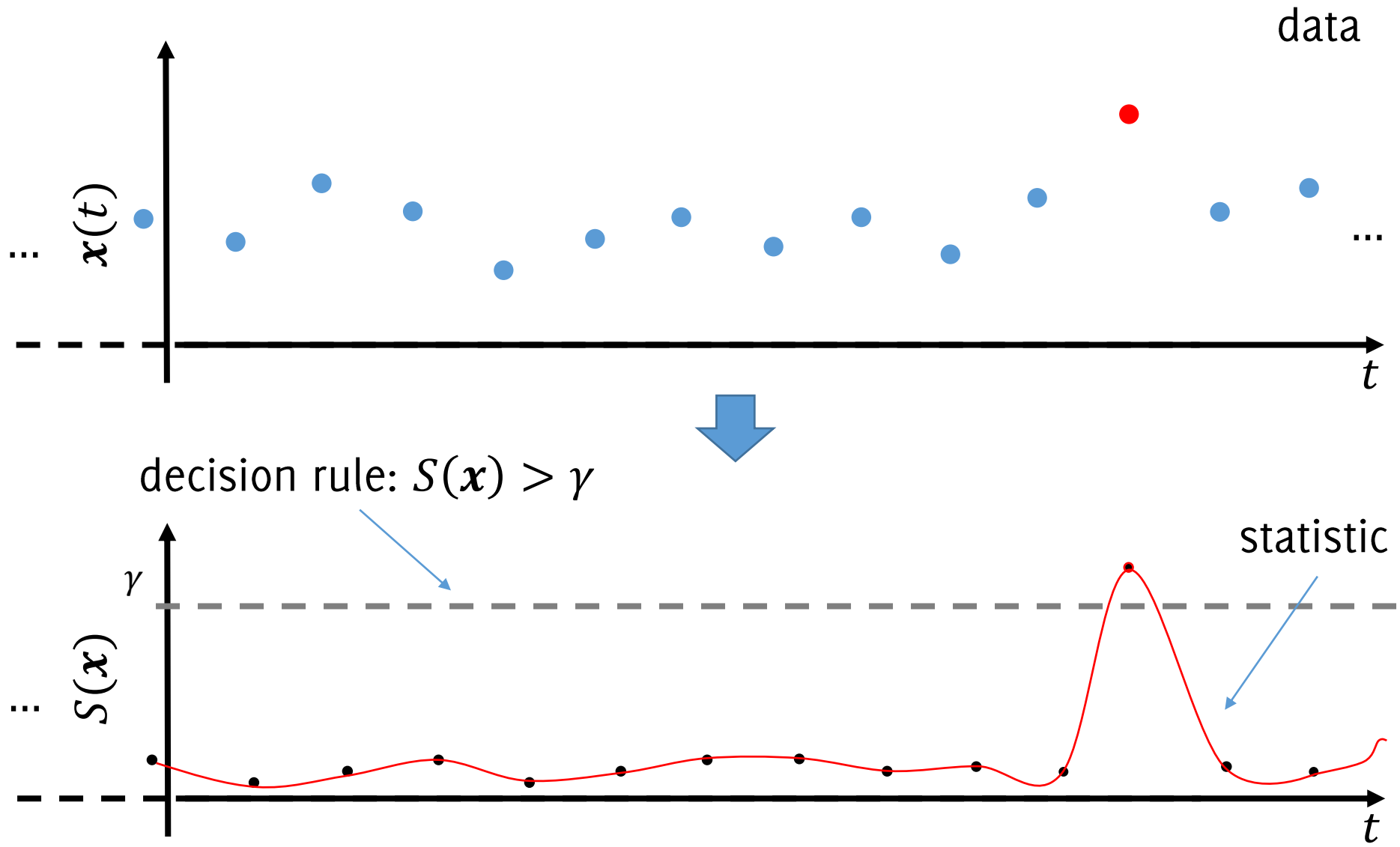


The Typical Solutions



racchi, Trovò

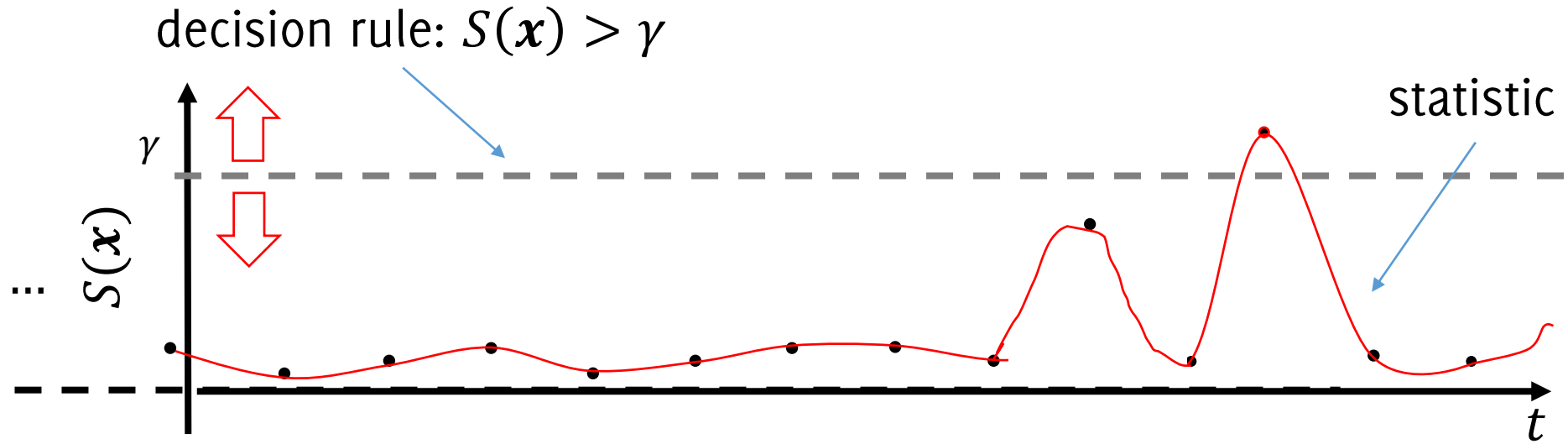
The Typical Solutions



racchi, Trovò

The Typical Solutions

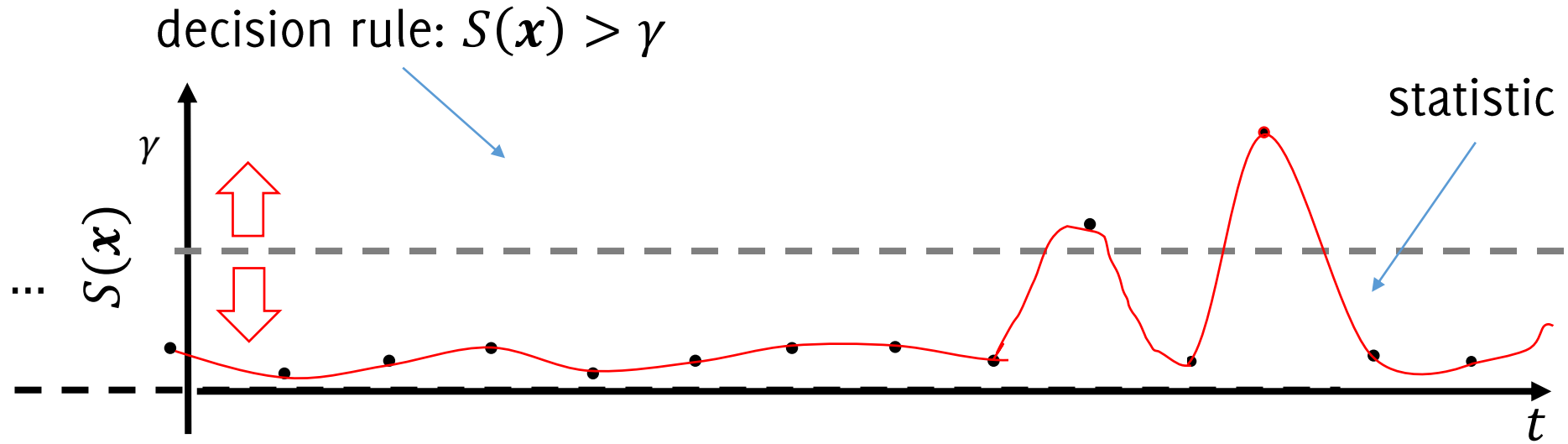
By changing γ it is possible to achieve different detection performance (e.g. more true positive, more false positives)



racchi, Trovò

The Typical Solutions

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



racchi, Trovò

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

Typically FPR is computed empirically

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

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

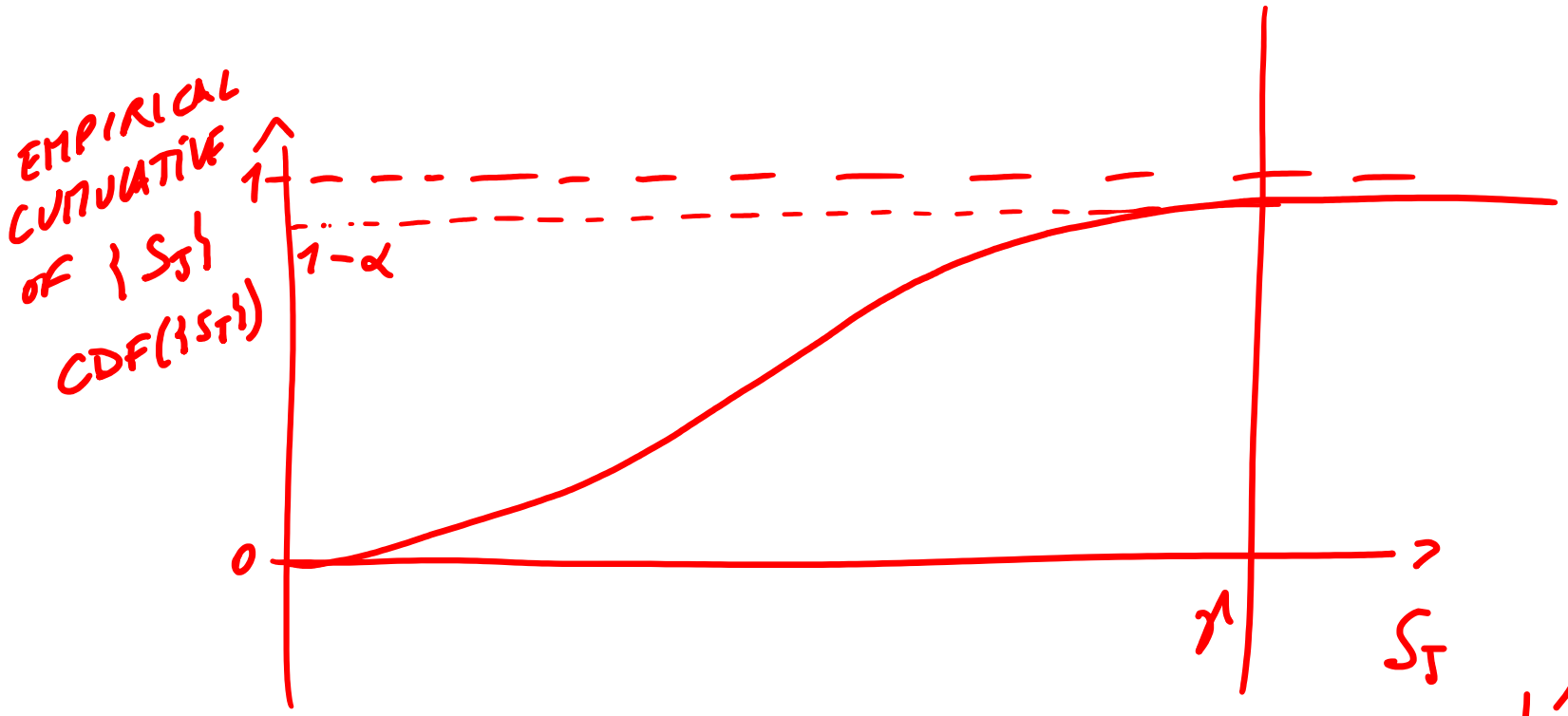
- Extract from TR a random batch b_j of size ν with replacement
- Compute the test statistic $S_j = S(b_j)$ over the batch b_j
- Store the value of the statistics S_j

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

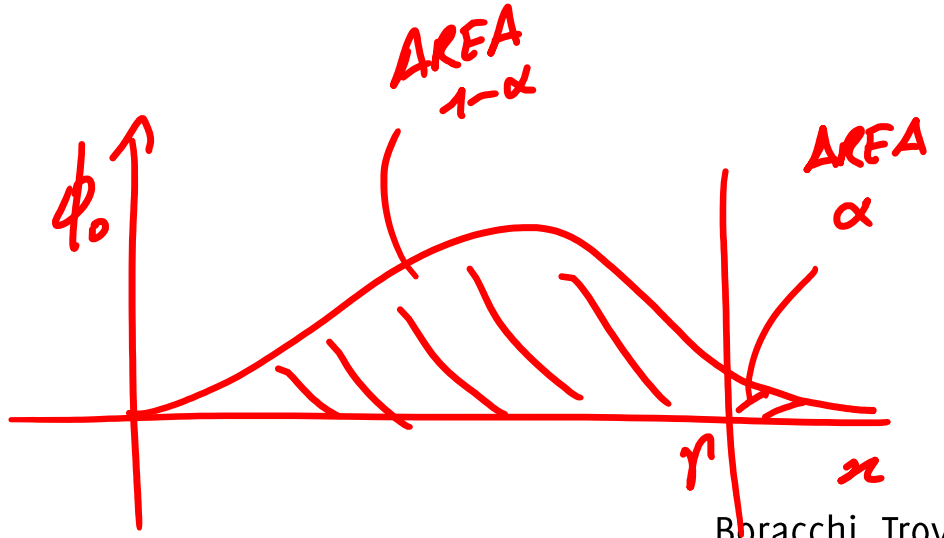
THRESHOLD ESTIMATION BY BOOTSTRAP

$$\{S_T\} = \{S(b_T),\ b_T \sim \phi_0\}$$



$$P(S_T > \gamma \mid b_T \sim \phi_0) \leq \alpha$$

this correspond to



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 target value of ARL_0, the corresponding threshold γ

$$\gamma = \gamma(ARL_0)$$

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

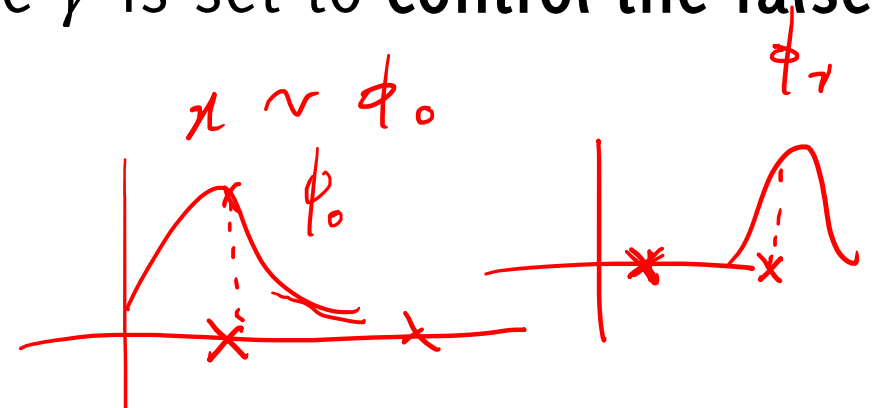
Assume data are generated from a parametric distribution ϕ_θ and formulate the following hypothesis test

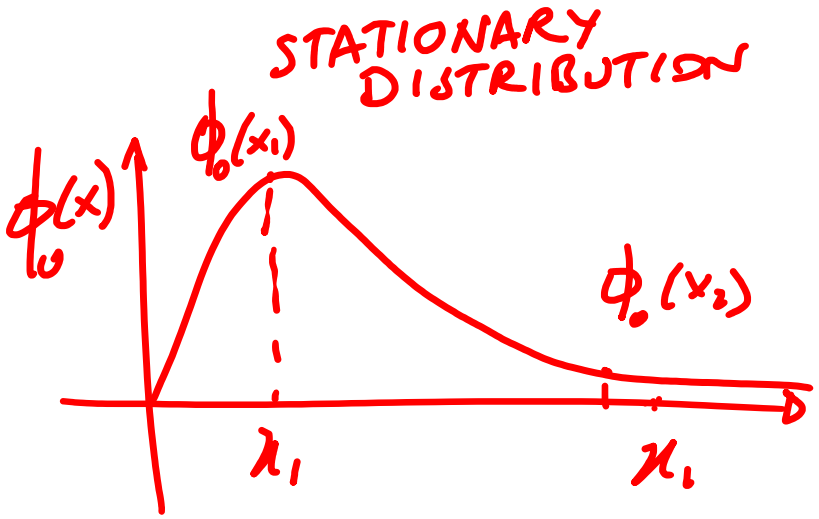
$$H_0: \theta = \theta_0 \text{ vs } H_1: \theta = \theta_1$$

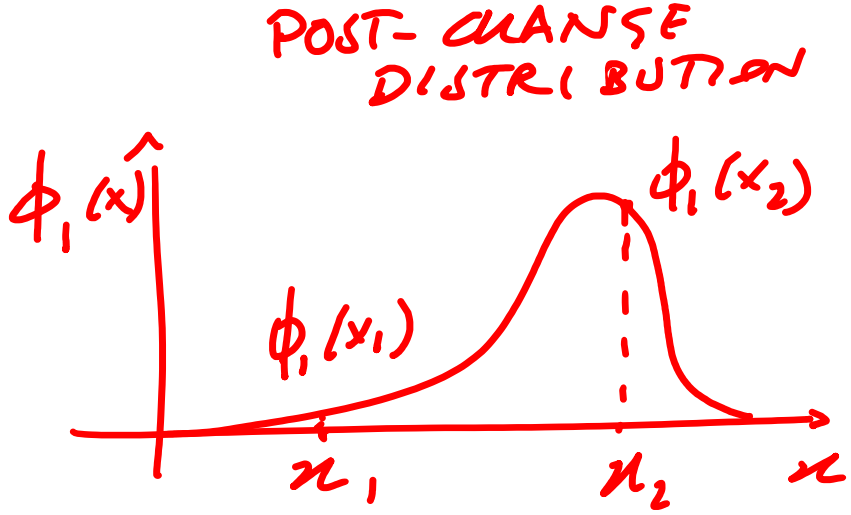
According to the Neumann Pearson lemma, the most powerful **statistic** to detect changes is the **likelihood ratio**

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

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







x_1 is VERY LIKELY UNDER ϕ_0, NOT UNDER ϕ_1,

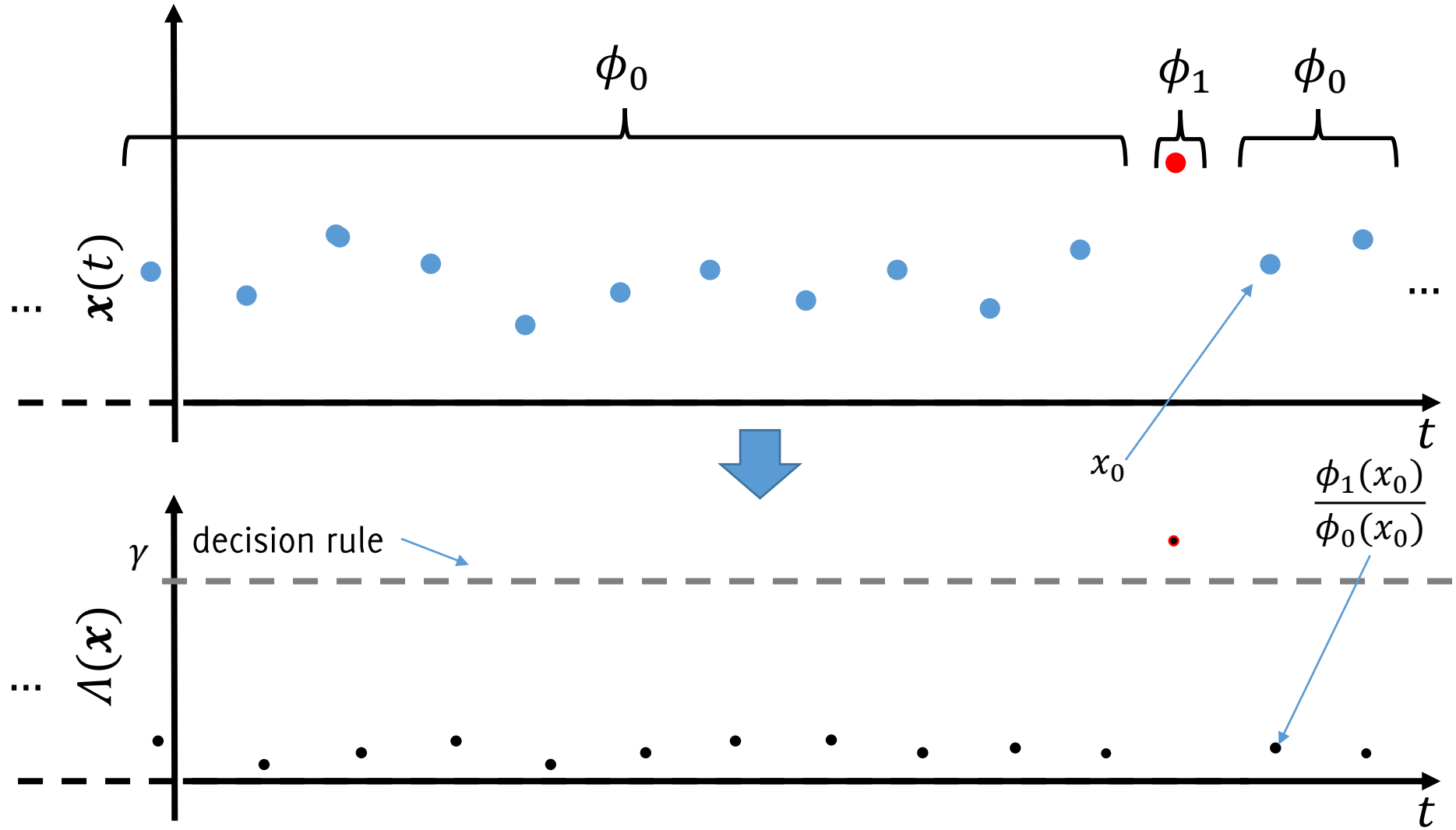
x_2 is " " " ϕ_1, NOT UNDER ϕ_0

$\frac{\phi_1(x_1)}{\phi_0(x_1)} \approx 0$ NORMAL SAMPLE

$\frac{\phi_1(x_2)}{\phi_0(x_2)} \gg 0$ ANOMALOUS SAMPLE

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 is a **sequential statistic.**

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:

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

This statistic includes a reset-mechanism, as it does not become negative when there is evidence of stationarity

And the decision rule is

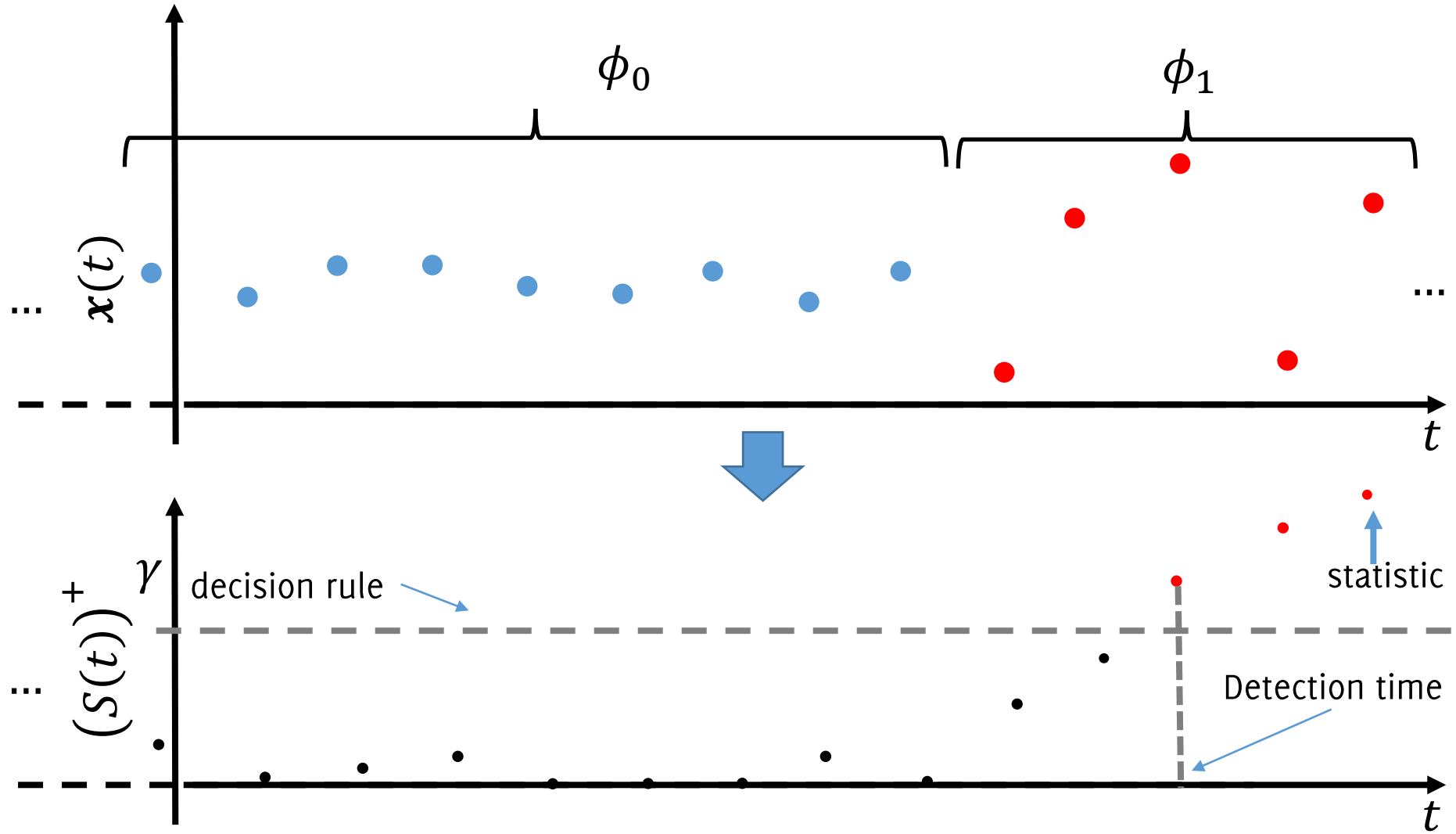
$$S(t) > \gamma$$

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 $\Lambda(\boldsymbol{x})$



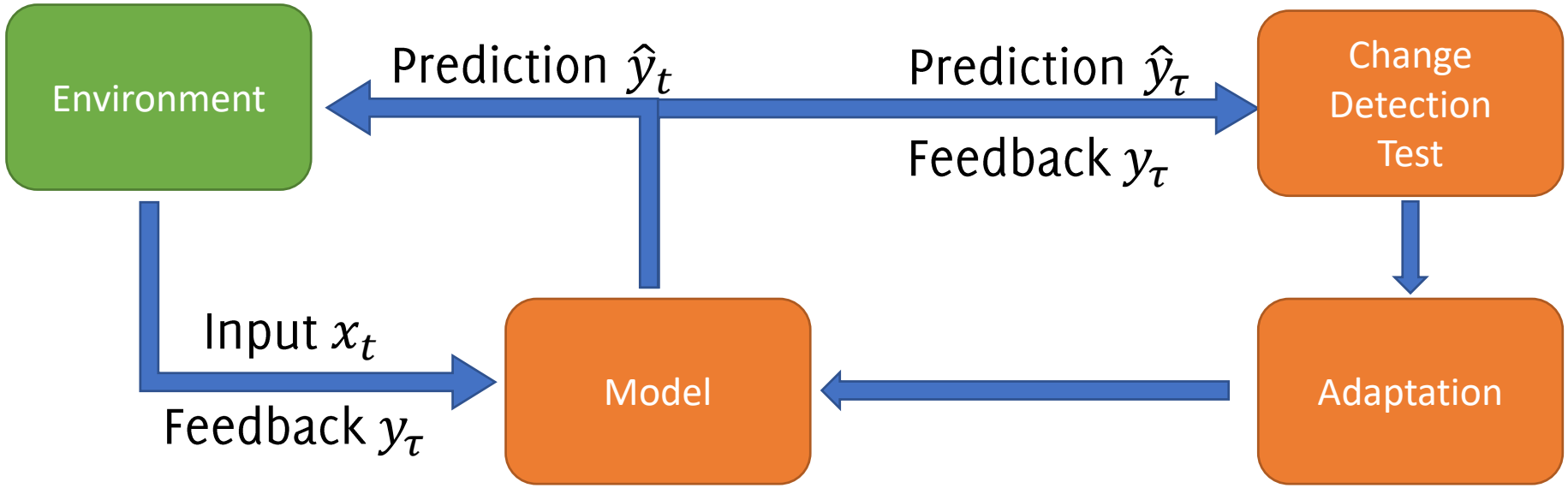
racchi, Trovò

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 e_t follows a Bernoulli pdf

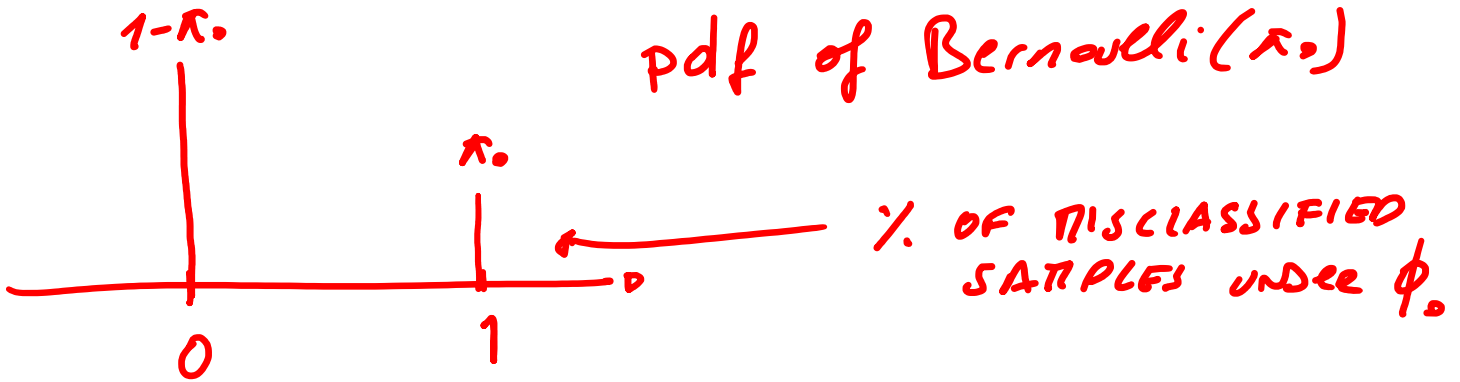
$$e_t \sim \text{Bernulli}(\pi_0)$$

Which is a discrete probability distribution of a random variable which:

- Takes the value 1 with probability π_0
- Takes the value 0 with probability $1 - \pi_0$

Where π_0 is the expected classification error when $\boldsymbol{x} \sim \phi_0$

$$\text{mean}(\text{Bernulli}(\pi_0)\ ) = \pi_0, \qquad \text{variance}(\text{Bernulli}(\pi_0)\ ) = \pi_0(1-\pi_0)$$



Monitoring the Classification Error

The sum of e_t in a window of ν samples follows a Binomial pdf

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

which is also a discrete distribution

$$\text{mean}\big(\mathcal{B}(\pi_0, \nu)\big) = \nu\pi_0, \qquad \text{variance}\big(\mathcal{B}(\pi_0, \nu)\big) = \nu\pi_0(1-\pi_0)$$

Gaussian approximation holds when ν is sufficiently large

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

The average classification error can be considered as a sequence of i.i.d. realization of a Gaussian distributed random value

Monitoring the Classification Error: DDM

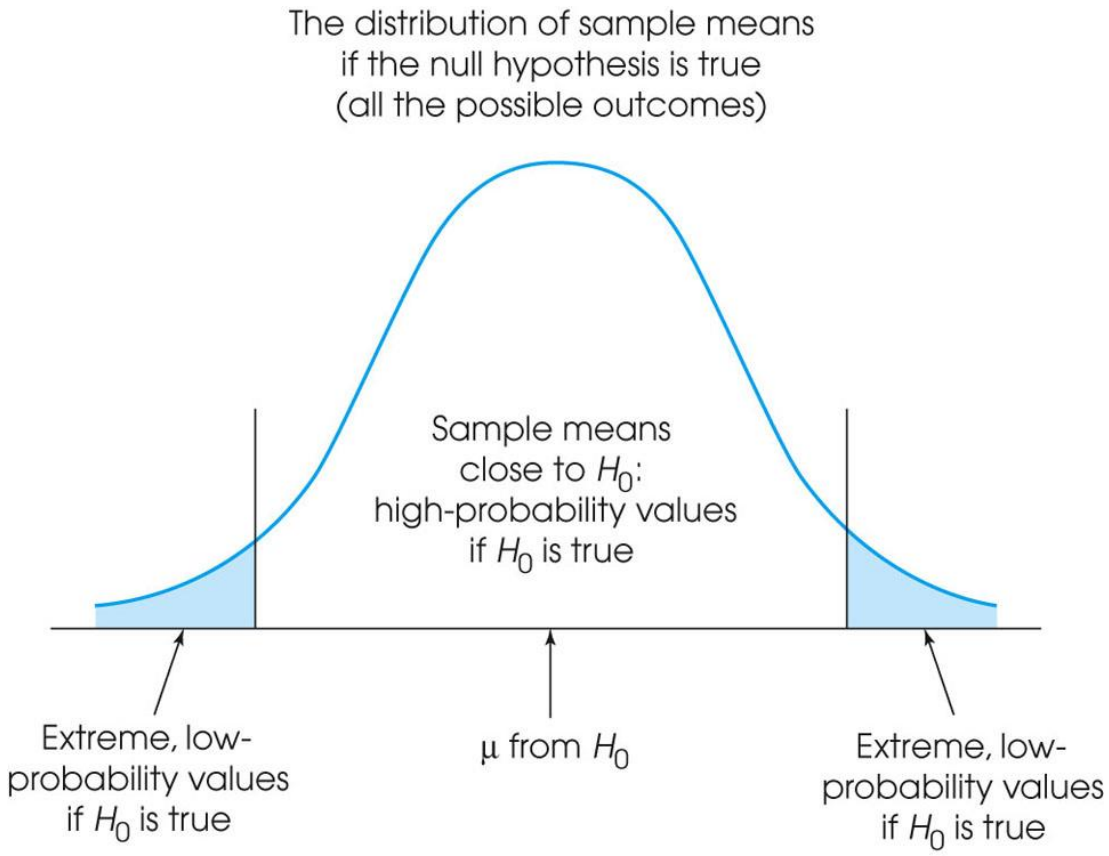
Basic idea behind Drift Detection Method (DDM):

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):

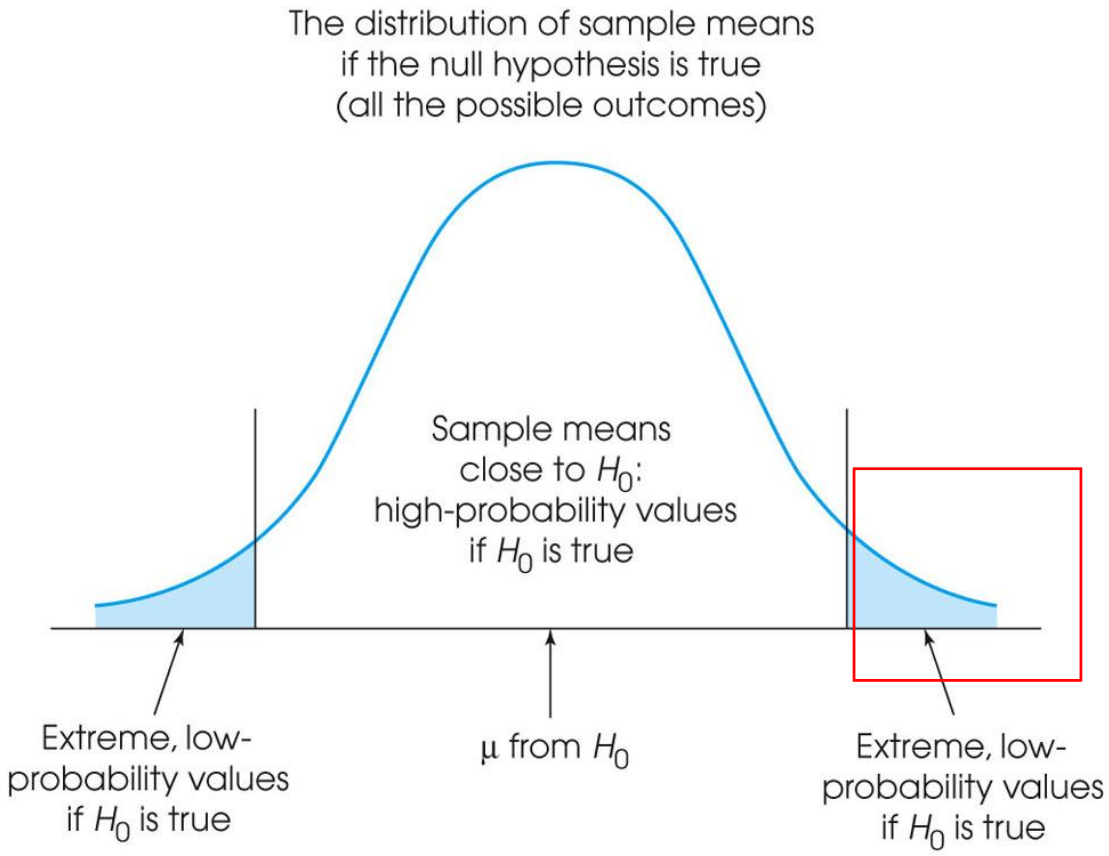
- Detect concept drift as an **outlier** in the classification error



Monitoring the Classification Error: DDM

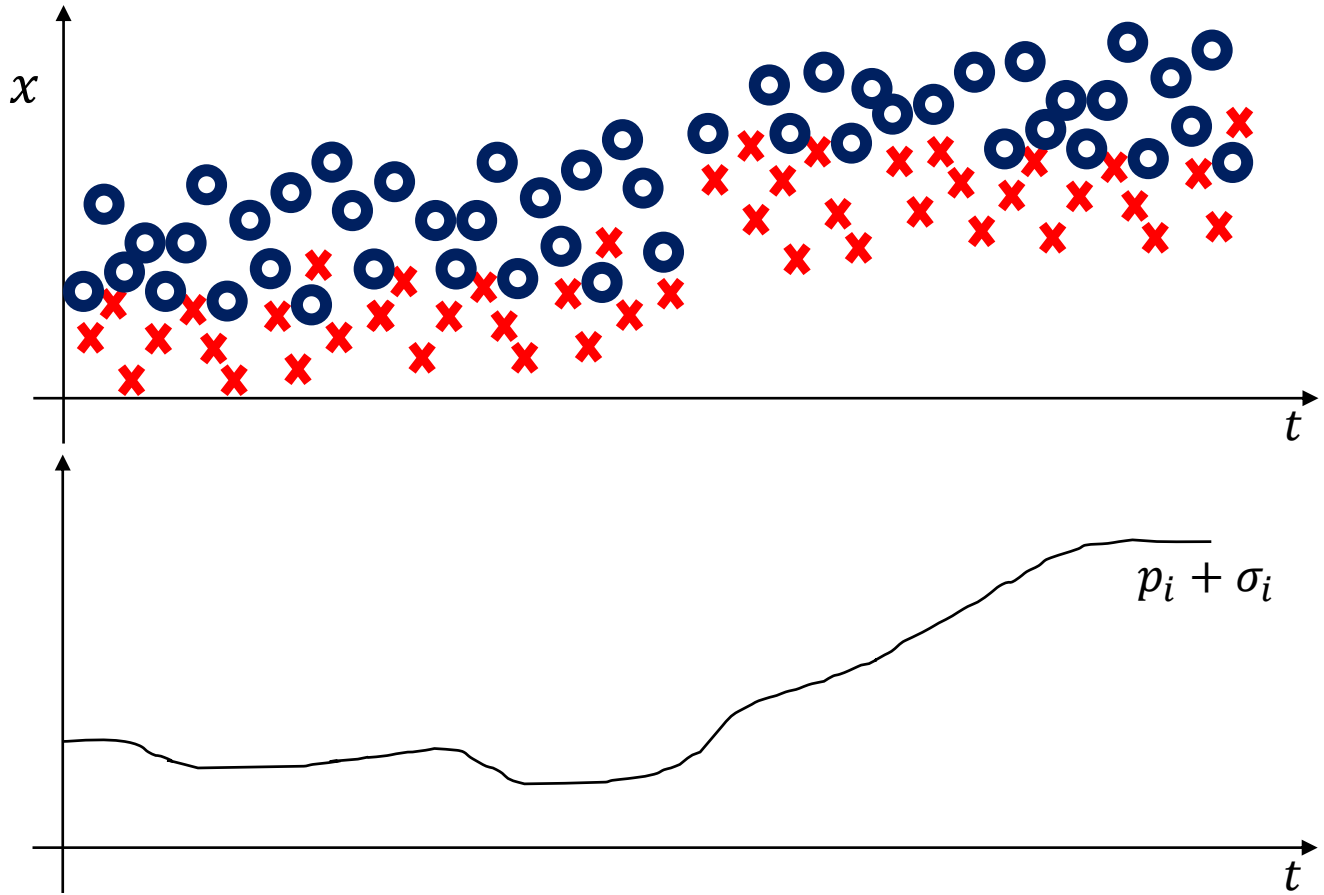
Basic idea behind Drift Detection Method (DDM):

- 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

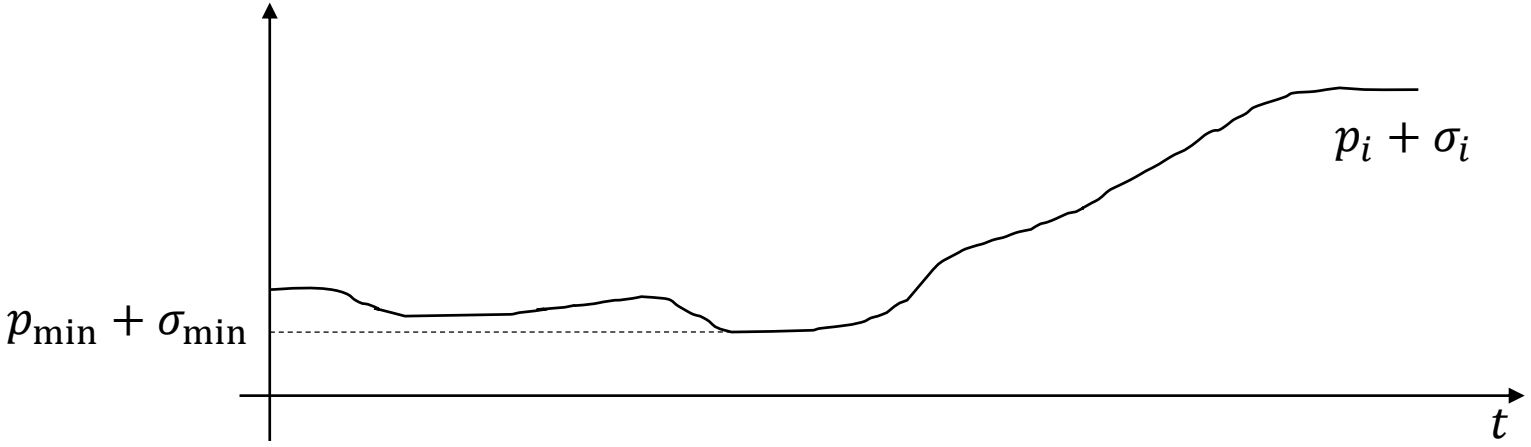
1. During monitoring, steadily compute p_i and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$



These are the sample estimates of the mean and standard deviation of the Gaussian of the average error over the first i samples

Monitoring the Classification Error: DDM

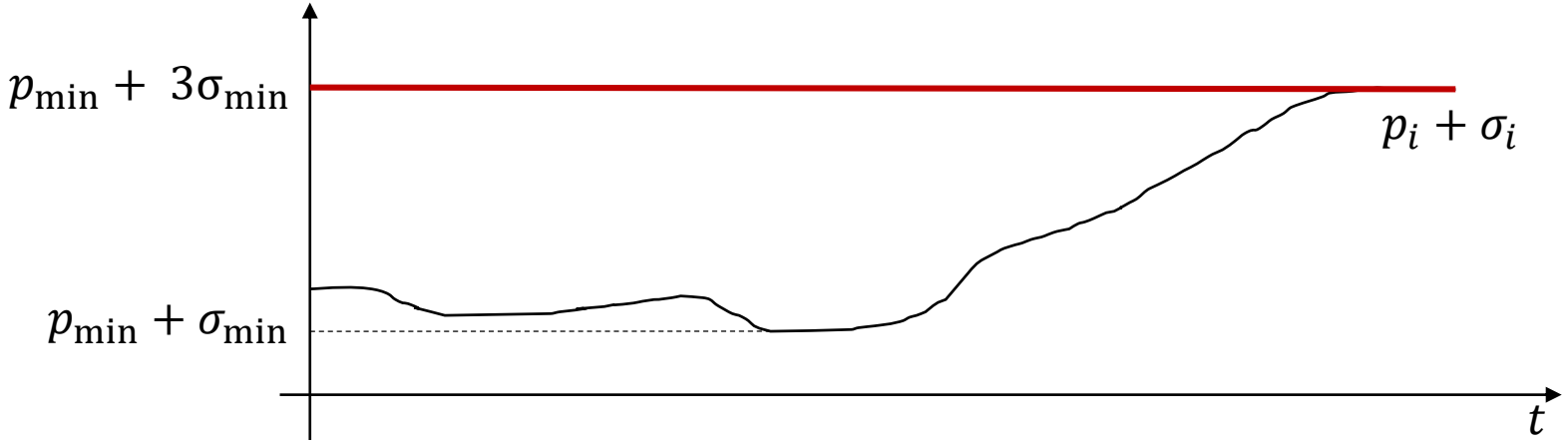
1. During monitoring, steadily compute p_i and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$
2. Let $p_{\min}$ be the minimum error before i and $\sigma_{\min} = \sqrt{\frac{p_{\min}(1-p_{\min})}{i}}$



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 $p_i + \sigma_i > p_{\min} + 3 * \sigma_{\min}$

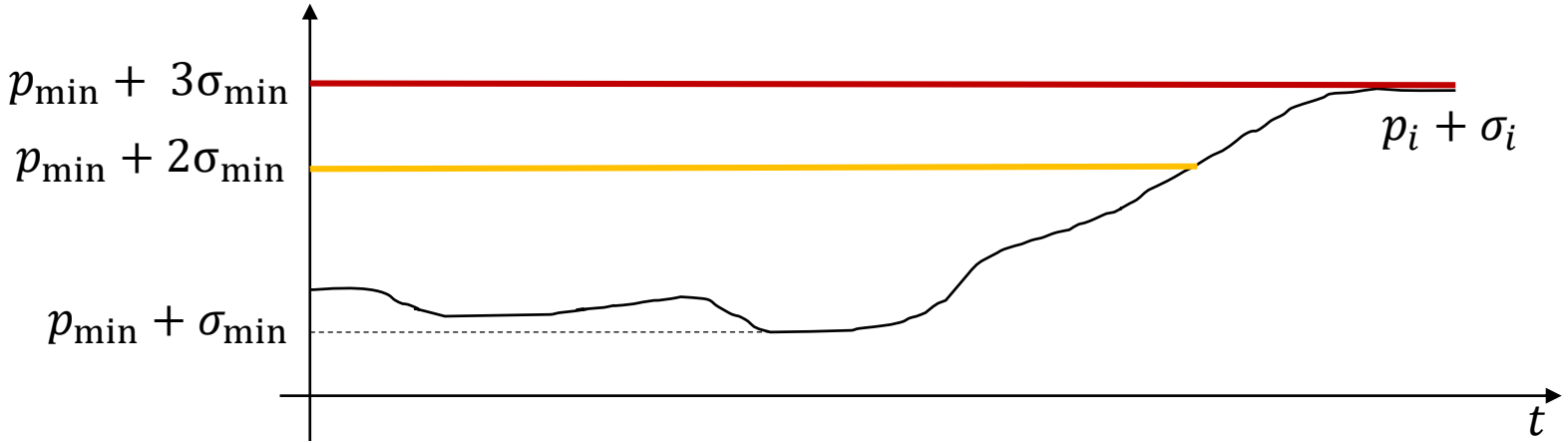
This is an heuristic decision rule, that does not guarantee control over FPR



Adaptation Heuristic in DDM

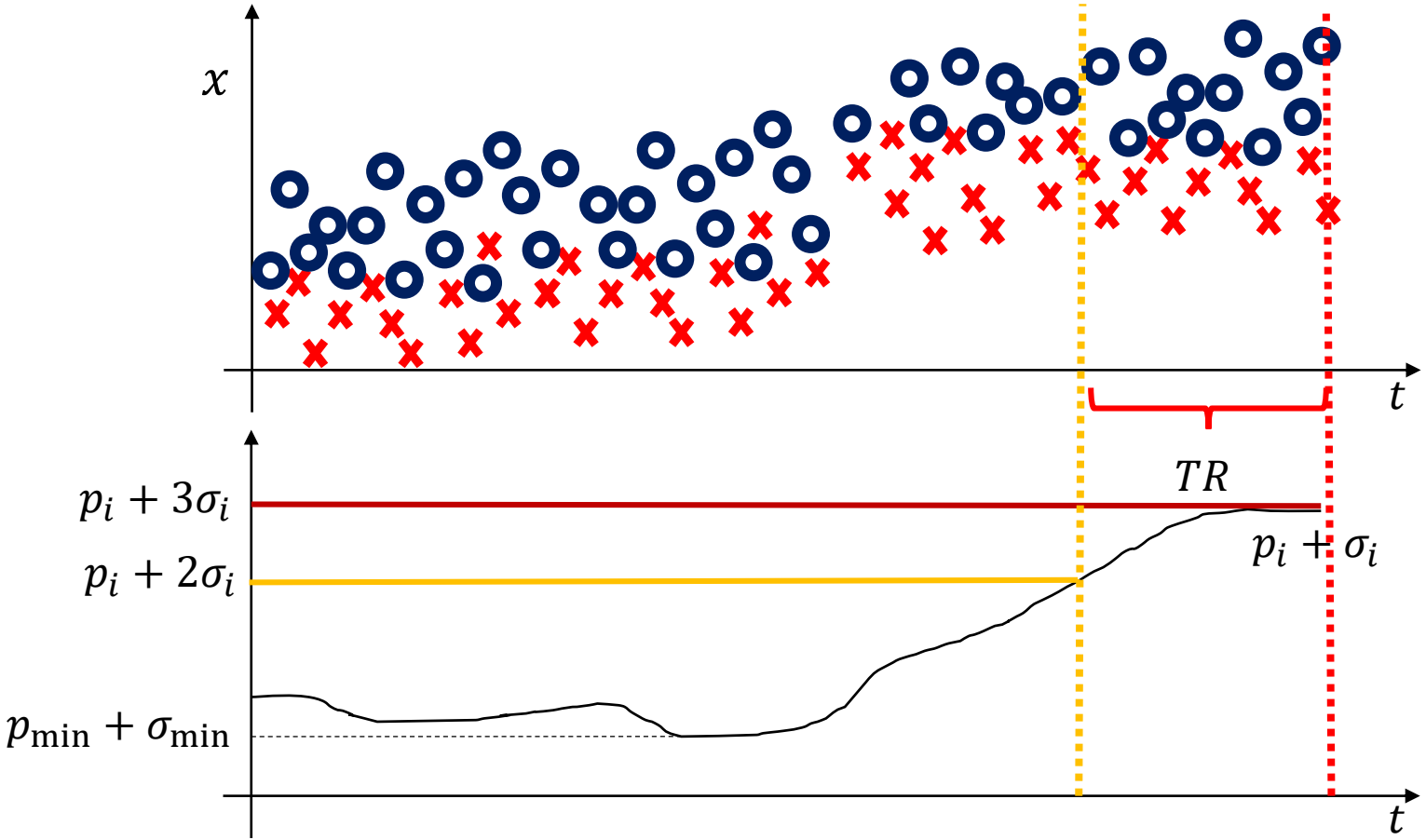
Monitoring the Classification Error: DDM

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

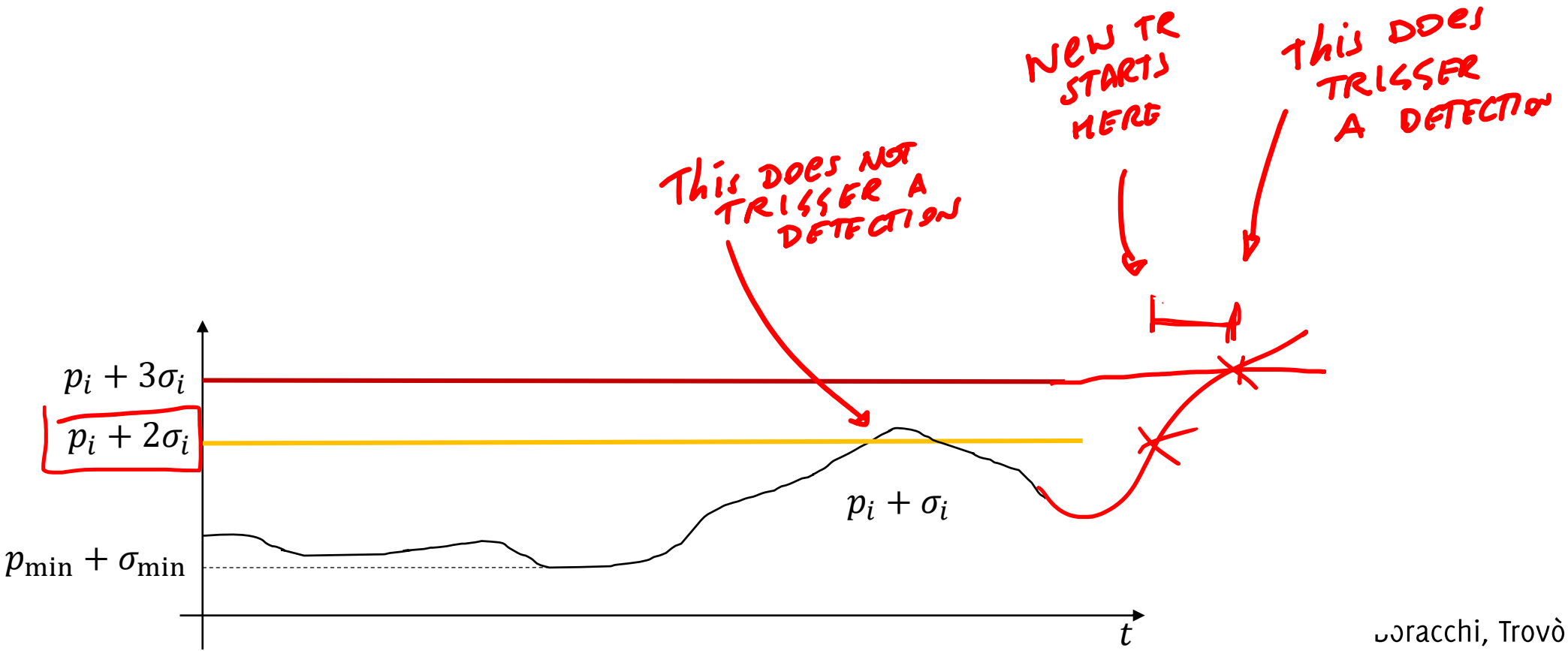


Boracchi, Trovò

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 similarly but **monitors the average distance between misclassified samples**

- Average distance between two mis-classifier samples 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, which is a convex combination of current error and average over the previous ones

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

where $\lambda \in [0,1]$ is a configuration parameter, $e_t \in \{0,1\}$

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

Use the **Exponential Weighted Moving Average** (EWMA) as tests statistic EWMA statistic, which is a convex combination of current error and average over the previous ones

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

where $\lambda \in [0,1]$ is a configuration parameter, $e_t \in \{0,1\}$

Now, if you expand the expression

$$Z_t = (1-\lambda)\big((1-\lambda)Z_{t-2} + \lambda\, e_{t-1}\big) + \lambda\, e_t,$$

...

$$Z_t = \sum (1-\lambda)^{t-i}\lambda e_i$$

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

Use the **Exponential Weighted Moving Average** (EWMA) as tests statistic
EWMA statistic, which is a convex combination of current error and average over the previous ones

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

where $\lambda \in [0,1]$ is a configuration parameter, $e_t \in \{0,1\}$

Now, if you expand the expression

$$Z_t = (1-\lambda)\big((1-\lambda)Z_{t-2} + \lambda\, e_{t-1}\big) + \lambda\, e_t,$$

...

$$Z_t = \sum (1-\lambda)^{t-i}\lambda e_i$$

In stationary conditions Z_t is an estimate of p_0, since all e_i have the same expectation

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

Any change $\phi_0 \rightarrow \phi_1$ introduces a bias in Z_t, as it includes values in the statistic that are generated with expectation $p_1 > p_0$, the classification error after the change.

The Exponential Weighted Moving Average expression

$$Z_t = \sum (1-\lambda)^{t-i} \lambda e_i$$

Assigns much smaller weights to old samples e_i $i \ll t$, and is mostly influenced by recent classification errors $e_i, i \approx t$

The parameter λ (typically set in $[0.1, 0.3]$) 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

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

being σ_0 the **standard deviation of the classification error.**

This expression holds for a general EWMA monitoring scheme

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:

Here is the EWMA detection rule

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

1. How to estimate p_0 and σ_0? These are typically unknown..
2. How to set L to guarantee a certain ARL_0?

EWMA for Bernoulli Random Variables

Let's assume that the error before the change has a constant expectation p_0 with the standard deviation $\sigma_0 = \sqrt{p_0(1-p_0)}$

Replace p_0 with its BLUE (Best Linear Unbiased Estimator) $\hat{p}_{0,t}$ at time t

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

Compute the corresponding varaince of $\hat{p}_{0,t}$ from the Bernoulli expression

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

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 in the control chart we have that

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

And, to preserve a target ARL_0 the control limit becomes time-dependent

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

Defining the sequence $\{L_t\}_t$ is very complicated as these 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

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

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

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 by Montecarlo simulations 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*

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

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

$L_t = f(p_0, ARL_0)$

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

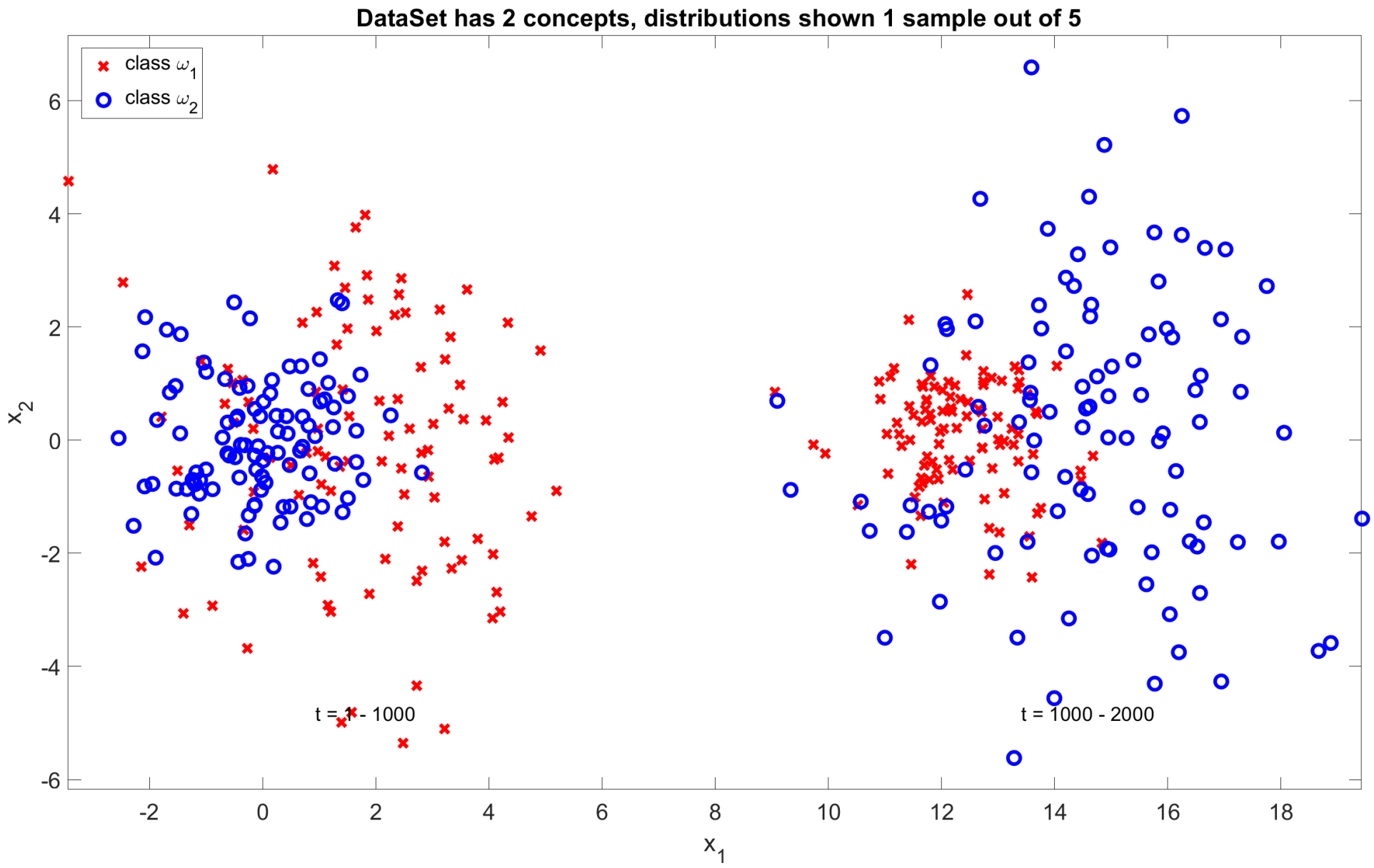
- Classifies each incoming samples
- Once feedback is provided, monitor the classification error using EWMA
- if EWMA detects a change
  - Updates the classifier using all the training samples between the latest sample reaching the warning_level and the sample where the detection occurred

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

Compare the performance with a classifier that is never updated

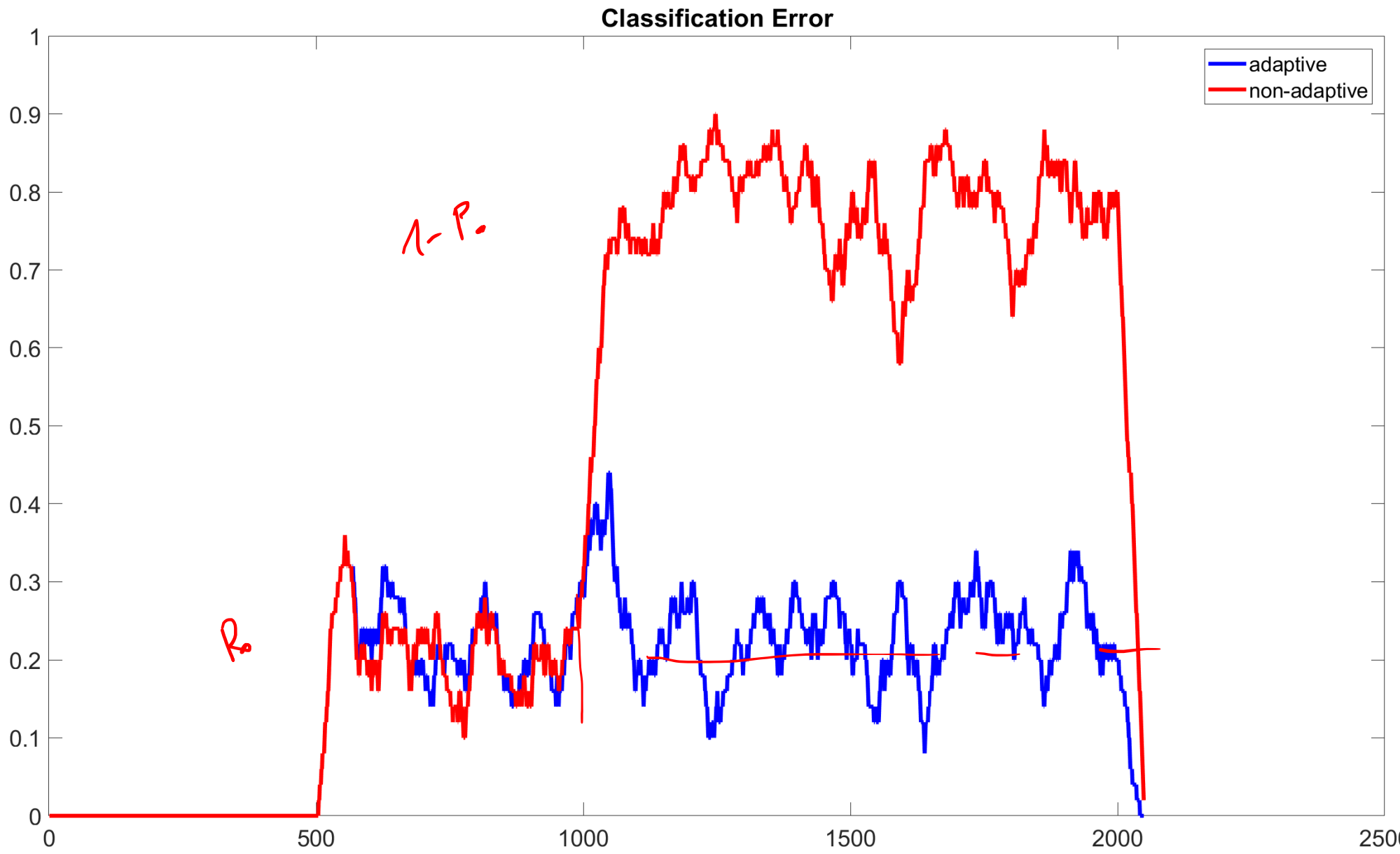
Display the EWMA statistic and its threshold over the whole stream

EWMA Example



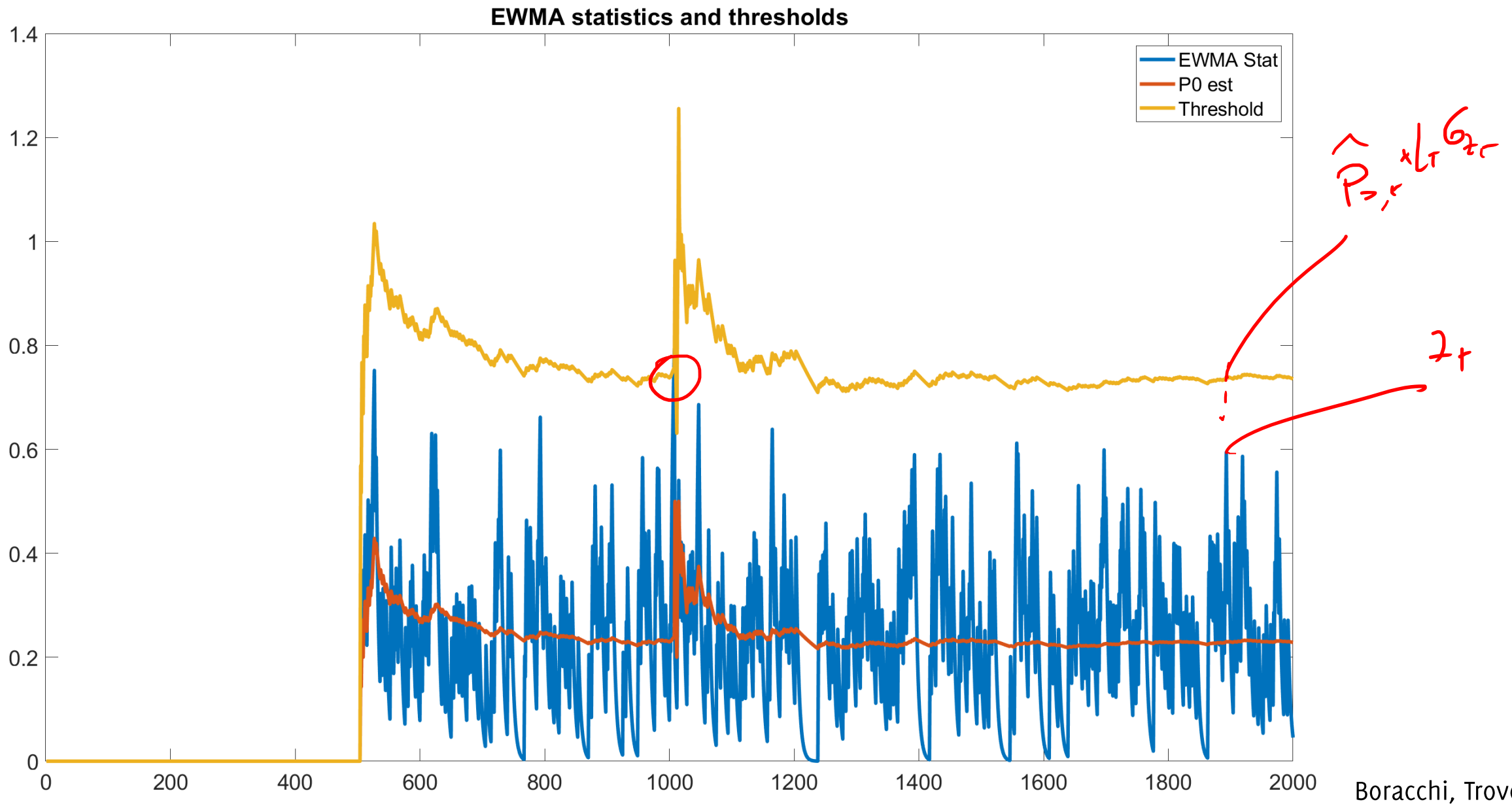
Boracchi, Trovò

EWMA Example (classes' swap)



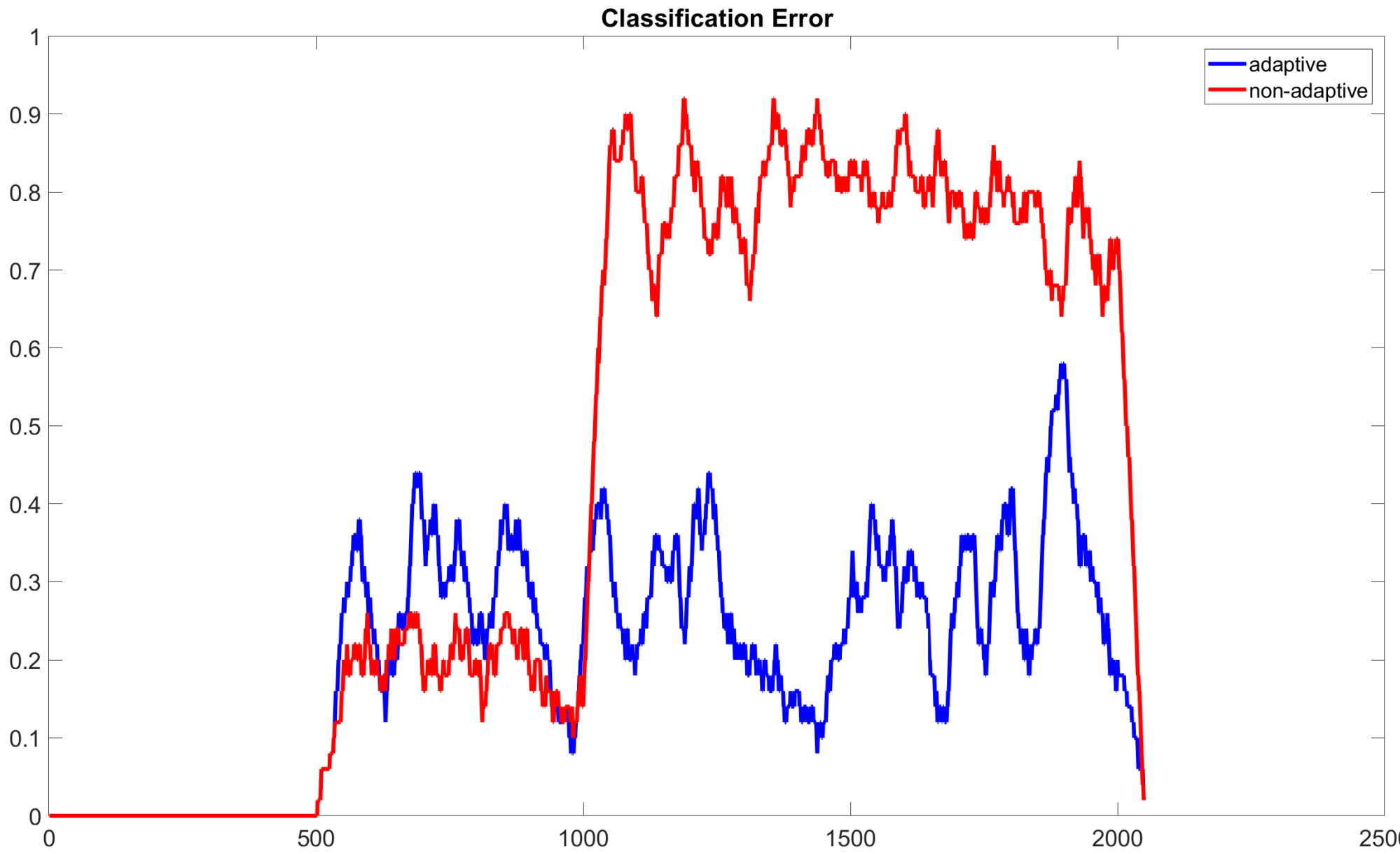
Boracchi, Trovò

EWMA Example



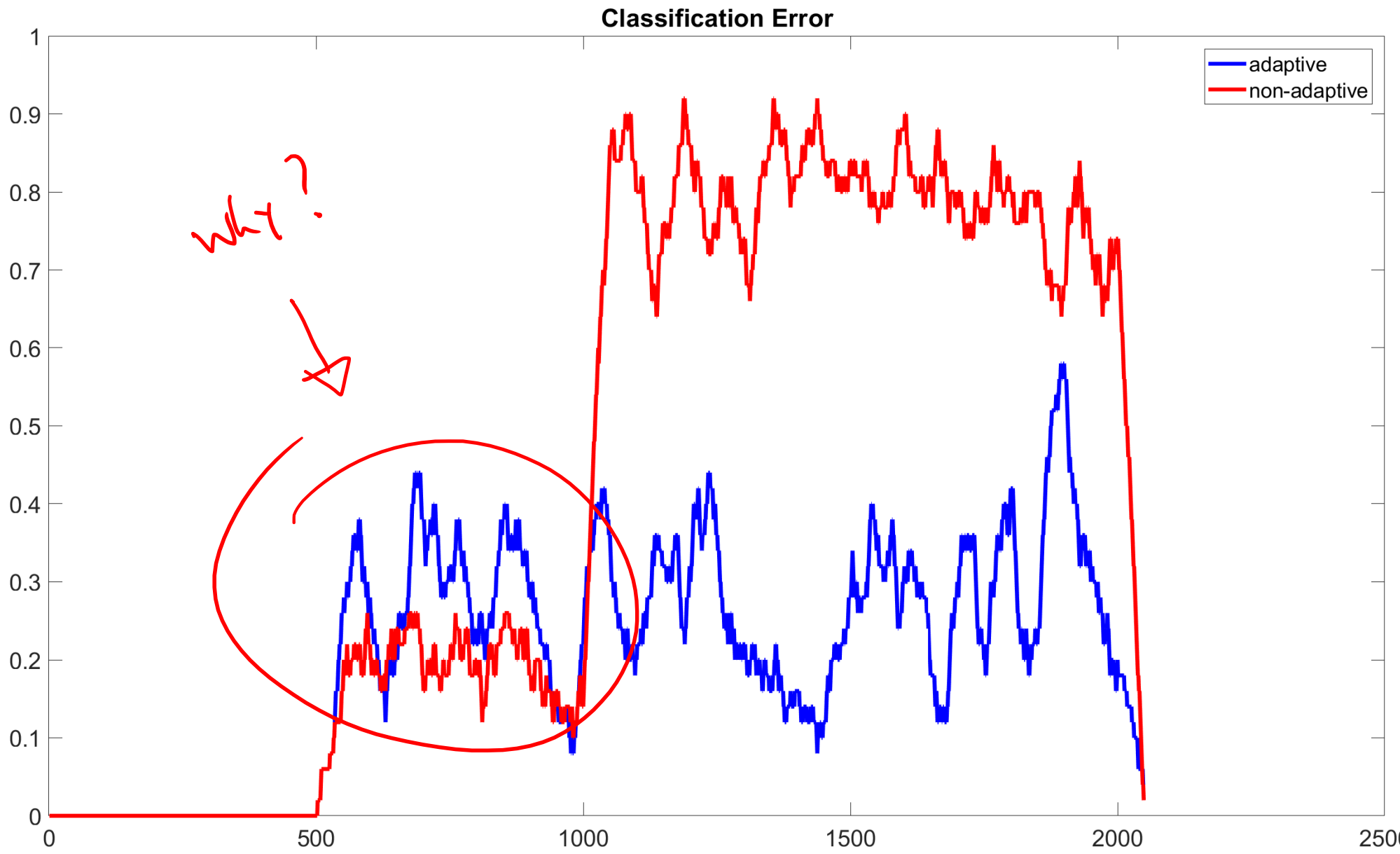
Boracchi, Trovò

EWMA using wrong thresholds



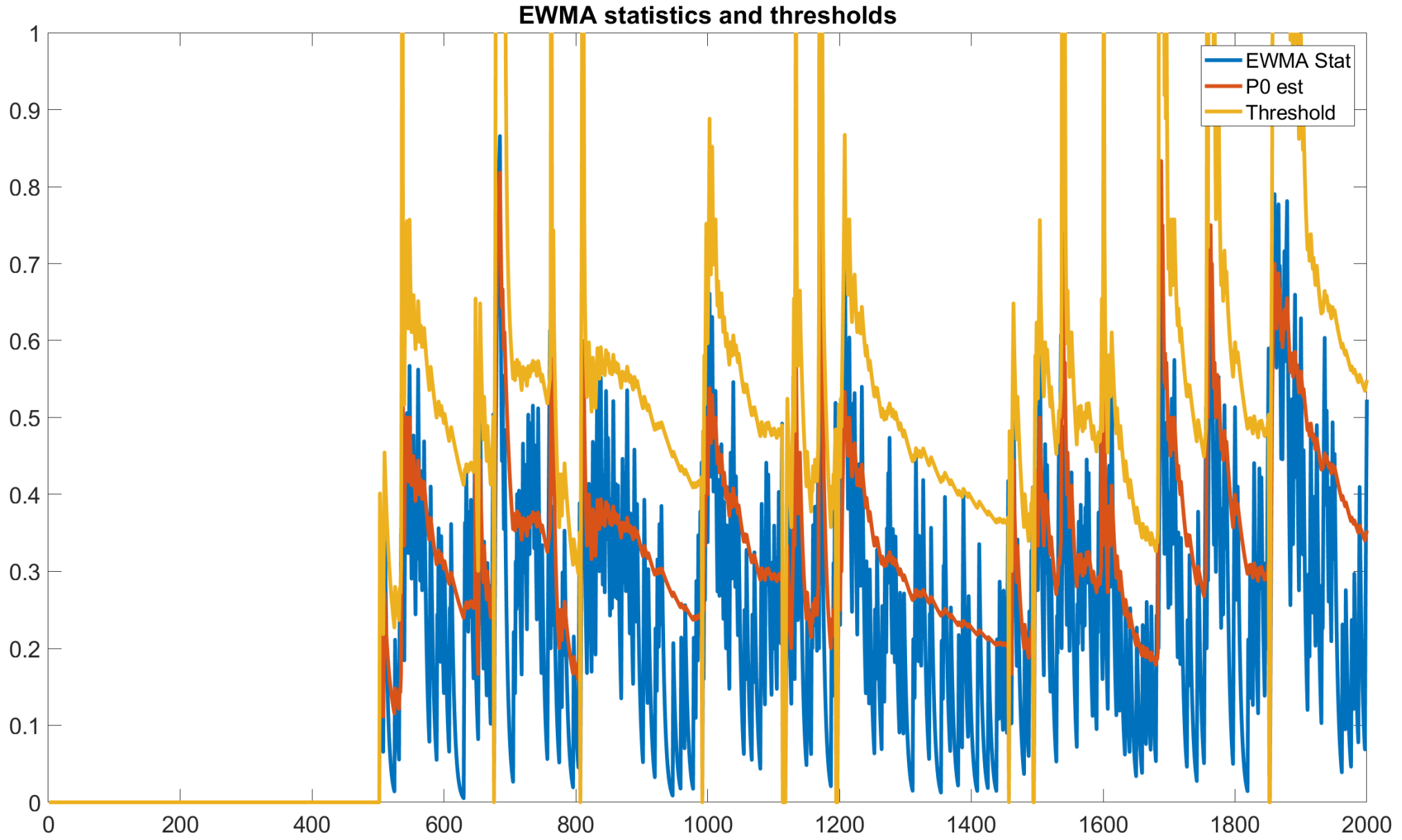
Boracchi, Trovò

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

April 20th, 2022

Politecnico di Milano, DEIB

giacomo.boracchi@polimi.it

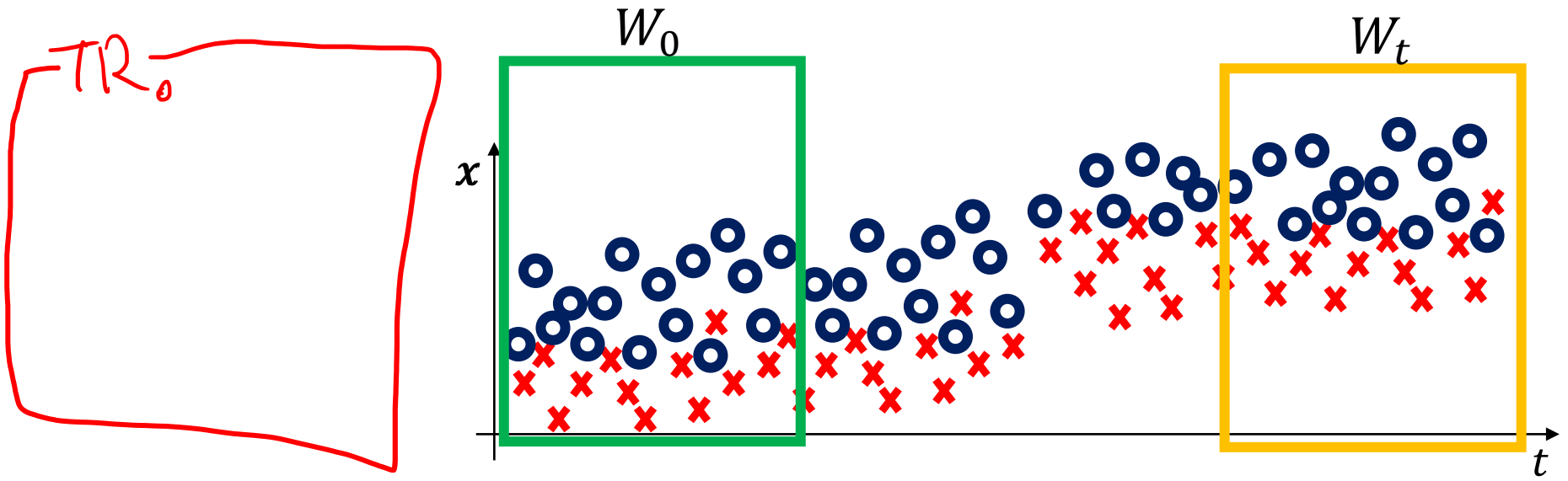
The Motivating Idea

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
- $\mathcal{S}(\cdot,\cdot)$ is a suitable statistic over the classification error

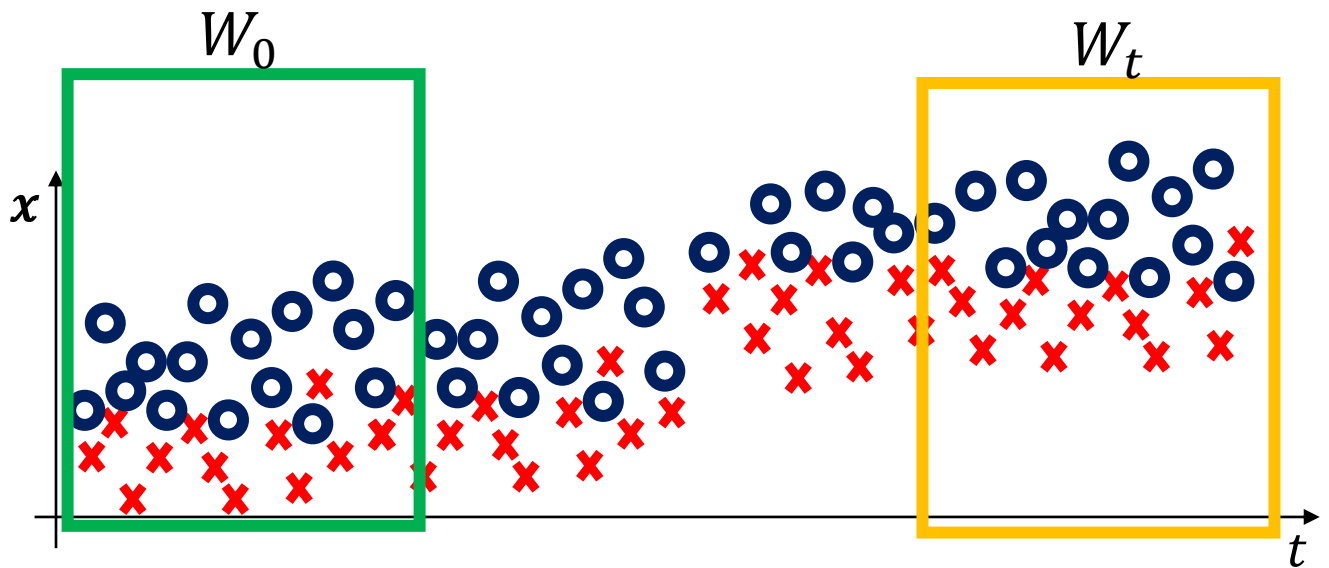


Window Comparison: Major Approaches

Comparing the classification error over W_t and W_0

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

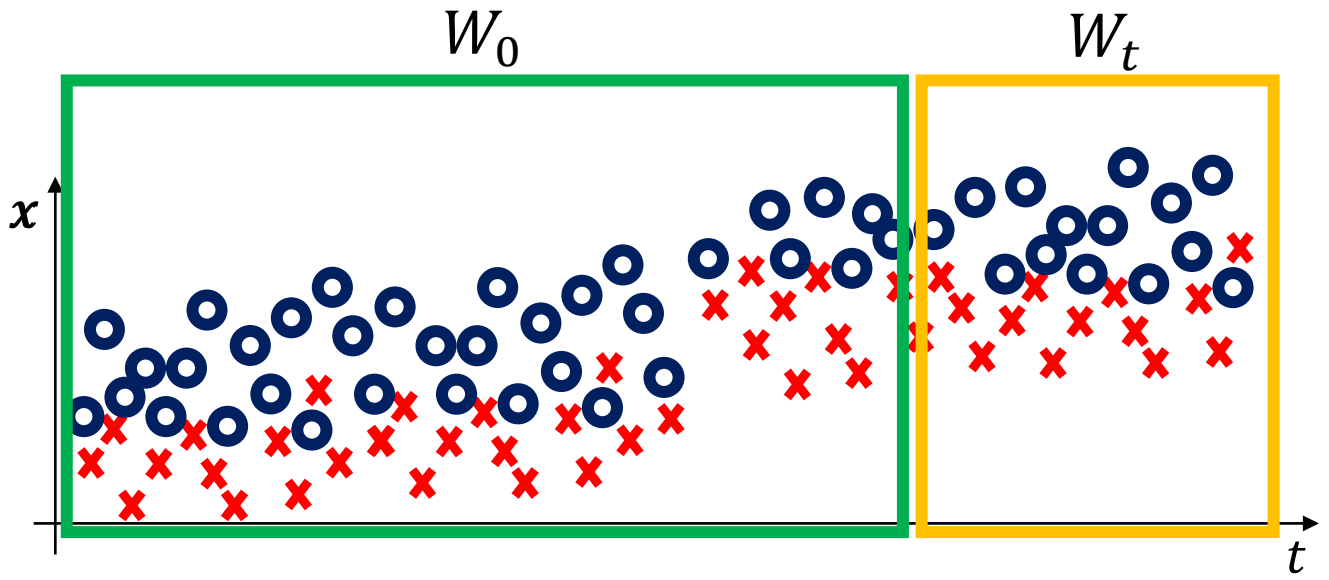
One sided t-test with $H_0 = \{p_t \leq p_0\}$ can detect concept drift



Window Comparison: Major Approaches

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

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

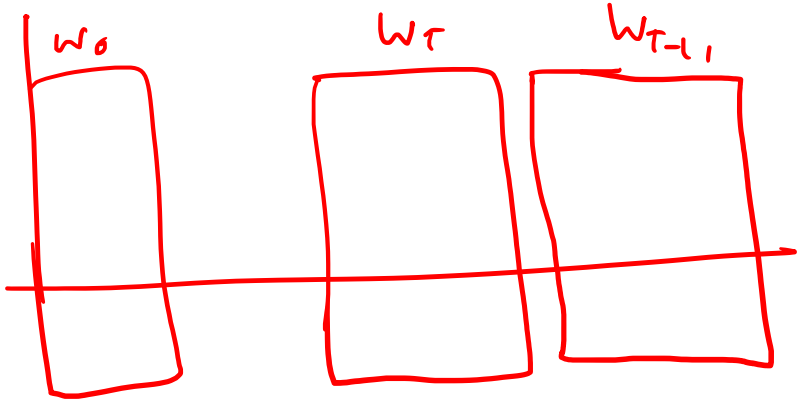


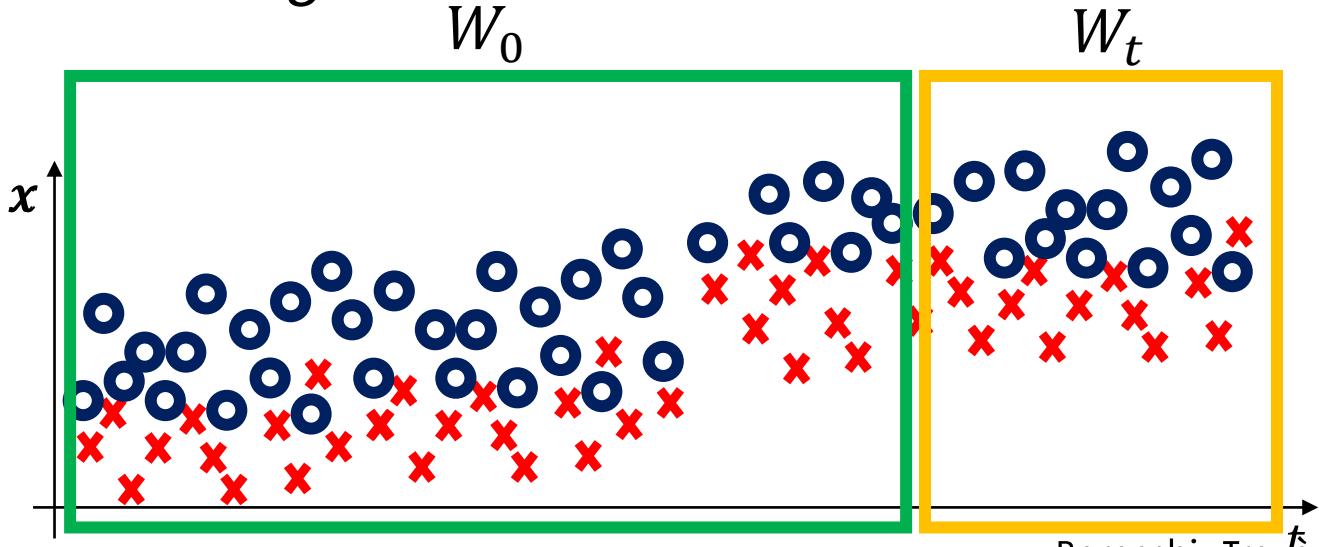
Nishida, K. and Yamauchi, K. *"Detecting concept drift using statistical testing"* In DS, pp. 264–269, 2007

Window Comparison: Major Issues

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

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



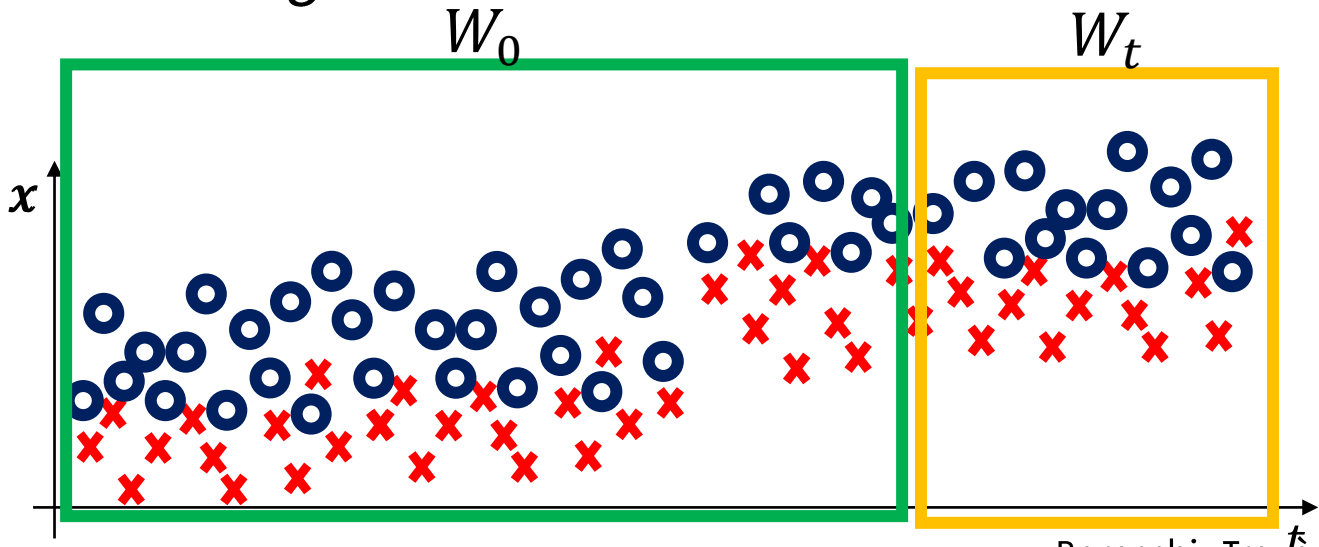


Window Comparison: Major Issues

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

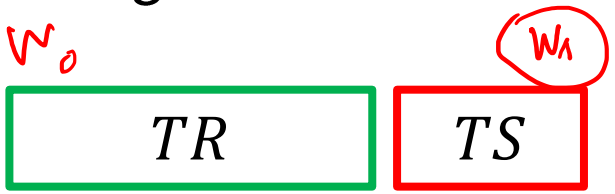
These issues prevent a sound statistical monitoring and give rise to heuristic schemes like DDM, EDDM, STEPD



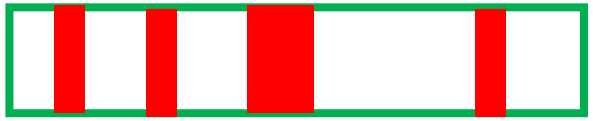
Window Comparison: Testing Exchangeability

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)