Learning Under Concept Drift: Methodologies and Applications

Giacomo Boracchi
DEIB, Politecnico di Milano,
giacomo.boracchi@polim.it

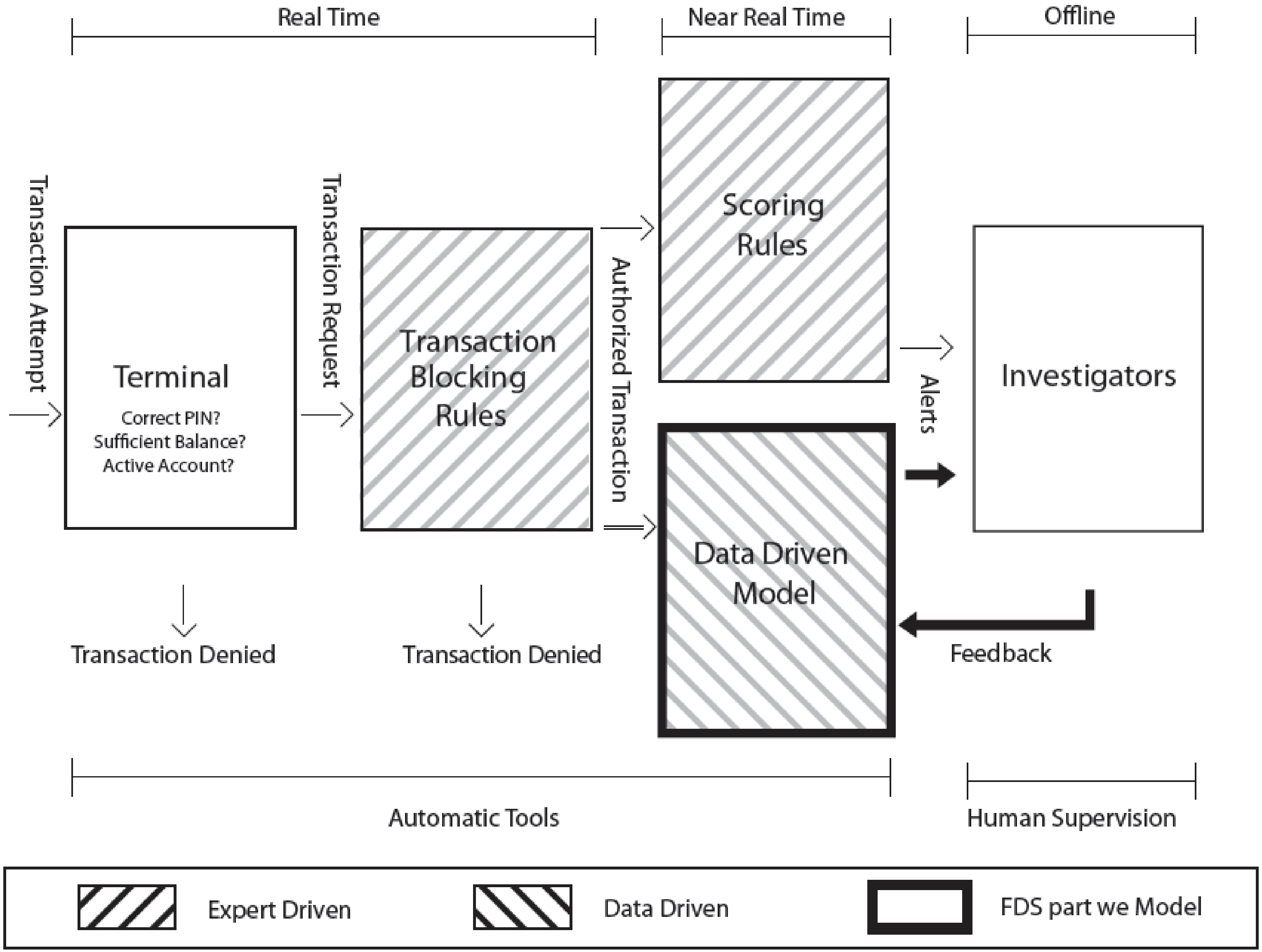
September 26, 2015

EANN 2015, Island of Rhodes,Greece

AN INTRODUCTORY EXAMPLE

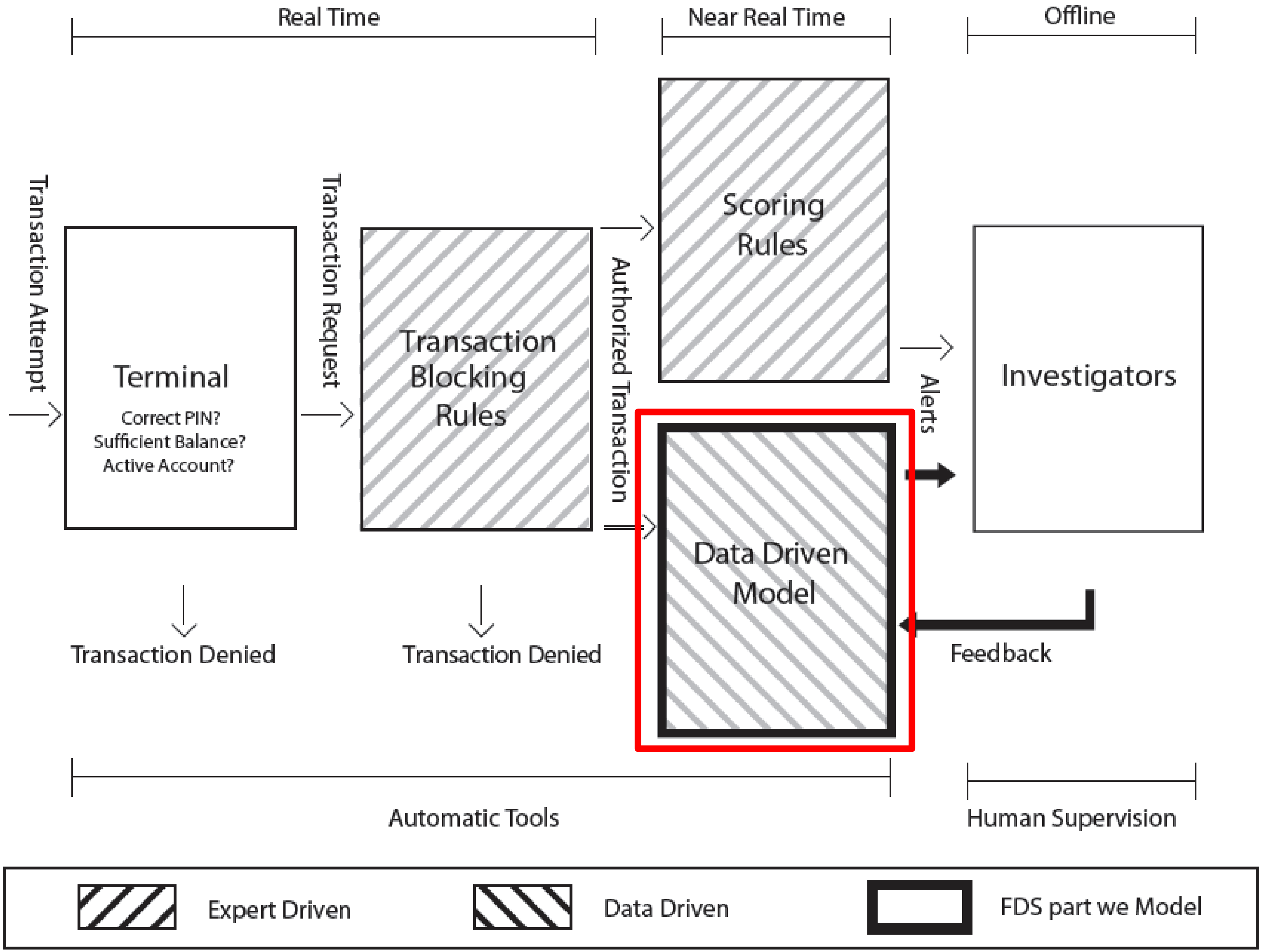
Everyday millions of **credit card transactions** are processed by **automatic systems** that are in charge of **authorizing, analyzing** and eventually **detect frauds**

A REAL WORLD FRAUD-DETECTION SYSTEM



Dal Pozzolo A., Boracchi G., Caelen O., Alippi C. and Bontempi G., ***Credit Card Fraud Detection and Concept-Drift Adaptation with Delayed Supervised Information*** **, Proceedings of IJCNN 2015,**

A REAL WORLD FRAUD-DETECTION SYSTEM



Dal Pozzolo A., Boracchi G., Caelen O., Alippi C. and Bontempi G., *Credit Card Fraud Detection and Concept-Drift Adaptation with Delayed Supervised Information* , Proceedings of IJCNN 2015,

AN INTRODUCTORY EXAMPLE

Everyday millions of **credit card transactions** are processed by **automatic systems** that are in charge of **authorizing, analyzing** and eventually **detect frauds**

Fraud detection is performed by a **classifier** that associates to each transaction a label «*genuine*» or «*fraudulent*»

Challenging classification problem because of

- High dimensional data (considering the number of supervised samples)
- Class unbalance
- A massive amount of transactions comes in a stream
- **Concept drift**: new fraudulent strategies appear
- **Concept drift**: genuine transactions evolves over time

CONCEPT DRIFT IN LEARNING PROBLEMS

Relevant examples of learning problem in presence of Concept Drift includes:

- recommendation systems and spam / email filtering where learning task consists in **predicting user preferences / interests**

Spam Classification

CONCEPT DRIFT IN LEARNING PROBLEMS

Relevant examples of learning problem in presence of Concept Drift includes:

- recommendation systems and spam / email filtering where learning task consists in **predicting user preferences / interests**
- Financial market analysis, where the learning task is to predict trends
- Environmental, and smart grids monitoring (anomaly detection and prediction tasks). Security (anomaly detection tasks)

Need to **retrain/update** the model to **keep performance**

IN PRACTICE…

In all **application scenarios** where

- **data-driven models** are used
- the **data-generating process** might evolve over time
- data come in the form of **stream** (acquisition over time)

Concept Drift (CD) should be taken into account.

THIS TUTORIAL

This tutorial focuses on:

- **methodologies and algorithms** for **adapting data-driven models** when CD occurs
- **learning aspects**, change/outlier/anomaly detection algorithms are not discussed
- **classification** as an example of **supervised learning problem**. Regression problems are not considered here even though similar issues applies
- the **most important approaches/frameworks** that can be implemented using any classifier, rather than solutions for specific classifiers
- Illustrations refer to scalar and numerical data, even though methodologies often applies to **multivariate and numerical/categorical data** as well

DISCLAIMER

The tutorial is **far from being exhaustive**… please have a look at the very good surveys below

J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," ACM Computing Surveys (CSUR), vol. 46, no. 4, p. 44, 2014

G. Ditzler, M. Roveri, C. Alippi, R. Polikar, "Adaptive strategies for learning in nonstationary environments," IEEE Computational Intelligence Magazine, November 2015

C.Alippi, G.Boracchi, G.Ditzler, R.Polikar, M.Roveri, "Adaptive Classifiers for Nonstationary Environments" Contemporary Issues in Systems Science and Engineering, IEEE/Wiley Press Book Series, 2015

DISCLAIMER

The tutorial is **far from being exhaustive**… please have a look at the very good surveys below

The tutorial will be unbalanced towards active methods but

- passive methods are very popular
- this is because of time limitation and a biased perspective (from my research activity)

DISCLAIMER

The tutorial is **far from being exhaustive**… please have a look at the very good surveys below

The tutorial will be unbalanced towards active methods but

- passive methods are very popular
- this is because of time limitation and a biased perspective (from my research activity)

I **hope** this tutorial will help researcher from other disciplines to familiarize with the problem and possibly contribute to the development of this research filed

Let’s try to make this tutorial as **interactive** as possible

PRESENTATION OUTLINE

- Problem Statement
  - Drift Taxonomy
- Active Approaches
  - CD detection monitoring Classification Error
  - CD detection monitoring raw data
  - JIT classifiers
  - Window comparison methods
- Passive Approaches
  - Single Model Methods
  - Ensemble Methods
- Concluding Remarks

PROBLEM STATEMENT

Learning in Nonstationary (Streaming) Environments

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 $(\boldsymbol{x}_t, y_t) \sim \mathcal{X}$

- $\boldsymbol{x}_t$ 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

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 $(\boldsymbol{x}_t, y_t) \sim \mathcal{X}$

- $\boldsymbol{x}_t$ 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 ($y_t \in \Lambda$)

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**,

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

CLASSIFICATION OVER DATASTREAMS

Typically, one **assumes**

- Independent and identically distributed (i.i.d.) inputs

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

- a **training set** is provided $TR = \{(\boldsymbol{x}_0, y_0), \dots, (\boldsymbol{x}_n, y_n)\}$

An **initial training set** TR is provided for learning K_0

- TR contains data generated in stationary conditions

A **stationary condition of $\mathcal{X}$** is also denoted **concept**

CLASSIFICATION OVER DATASTREAMS

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

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

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**)

ASSUMPTIONS: SUPERVISED SAMPLES

We assume that **few supervised samples** are provided also during **operations**. These are necessary to:

- **React/adapt to concept drift**
- **Increase classifier accuracy** in stationary conditions

The classifier K_0 is **updated** during operation, thus will be denoted by K_t.

DRIFT TAXONOMY

DRIFT TAXONOMY

- Drift taxonomy according to two characteristics:
- What is changing?

$$\phi_t(\boldsymbol{x}, y) = \phi_t(y|\boldsymbol{x})\ \phi_t(\boldsymbol{x})$$

- Drift might affect $\phi_t(y|\boldsymbol{x})$ and/or $\phi_t(\boldsymbol{x})$
  - Real
  - Virtual
- How does process change over time?
  - Abrupt
  - Gradual
  - Incremental
  - Recurring

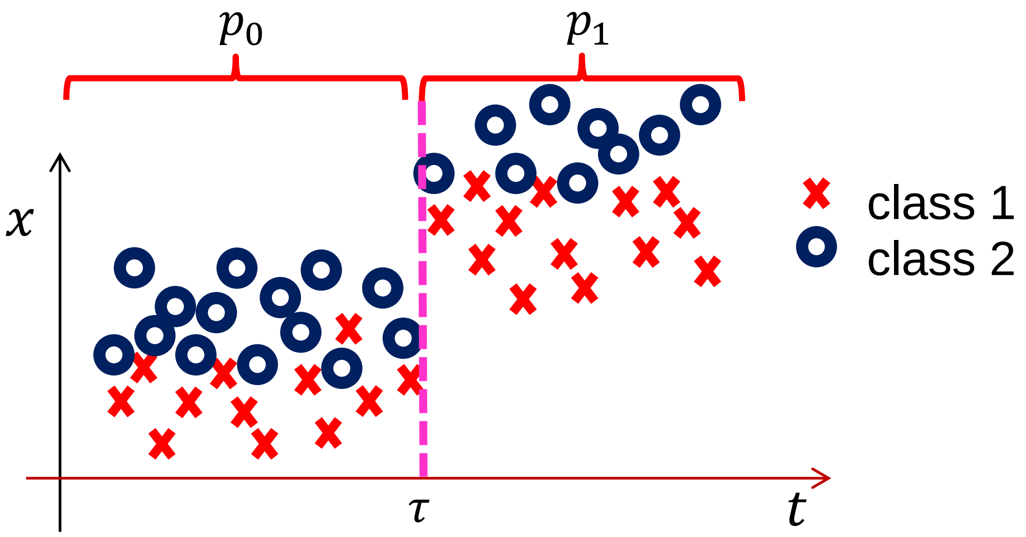
DRIFT TAXONOMY: WHAT IS CHANGING?

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.

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

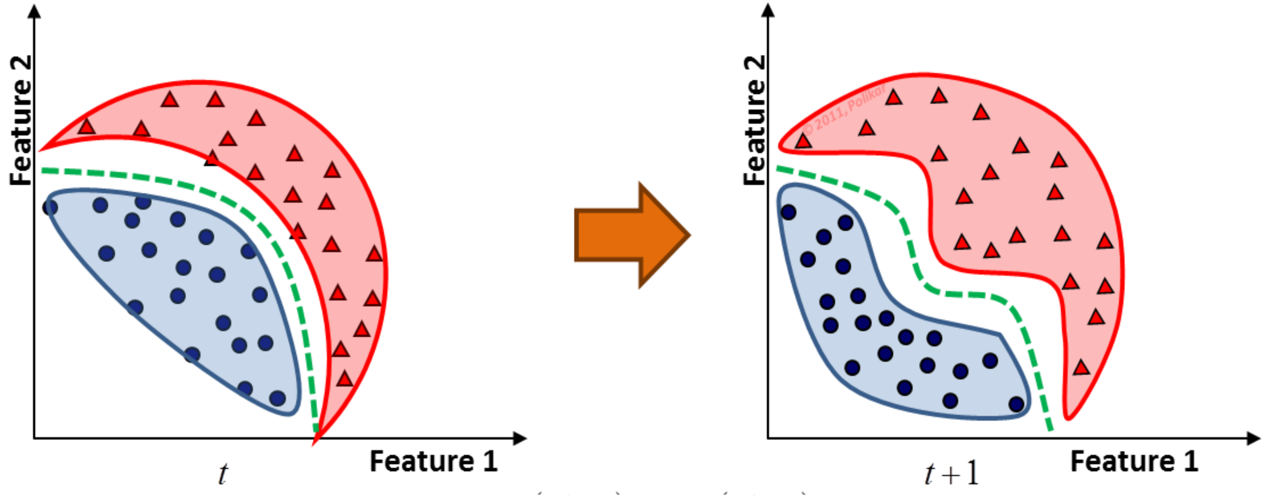


Real Drift

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

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?

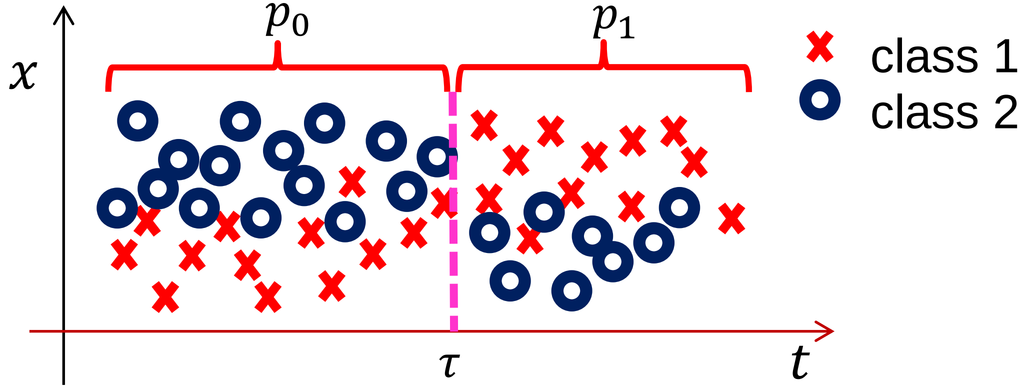
Real Drift

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

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. changes in the "class function", classes swap



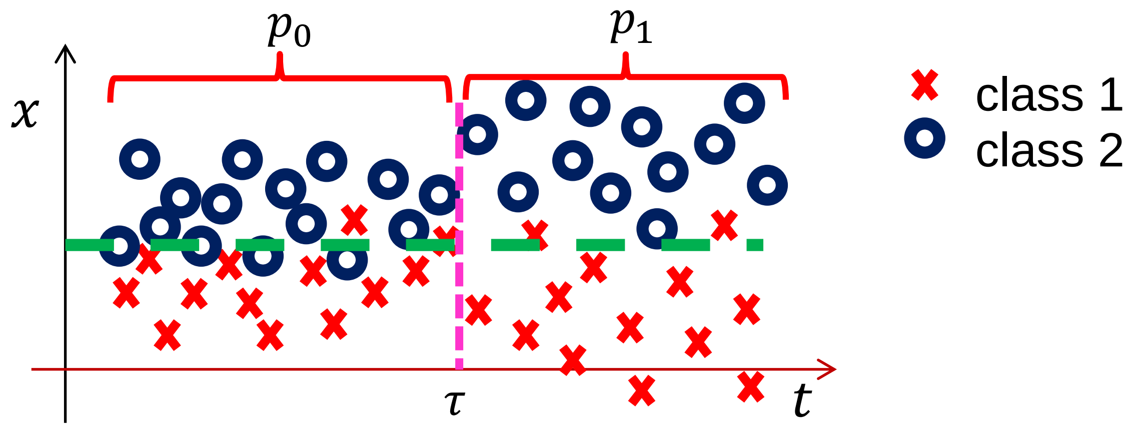
DRIFT TAXONOMY: WHAT IS CHANGING?

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

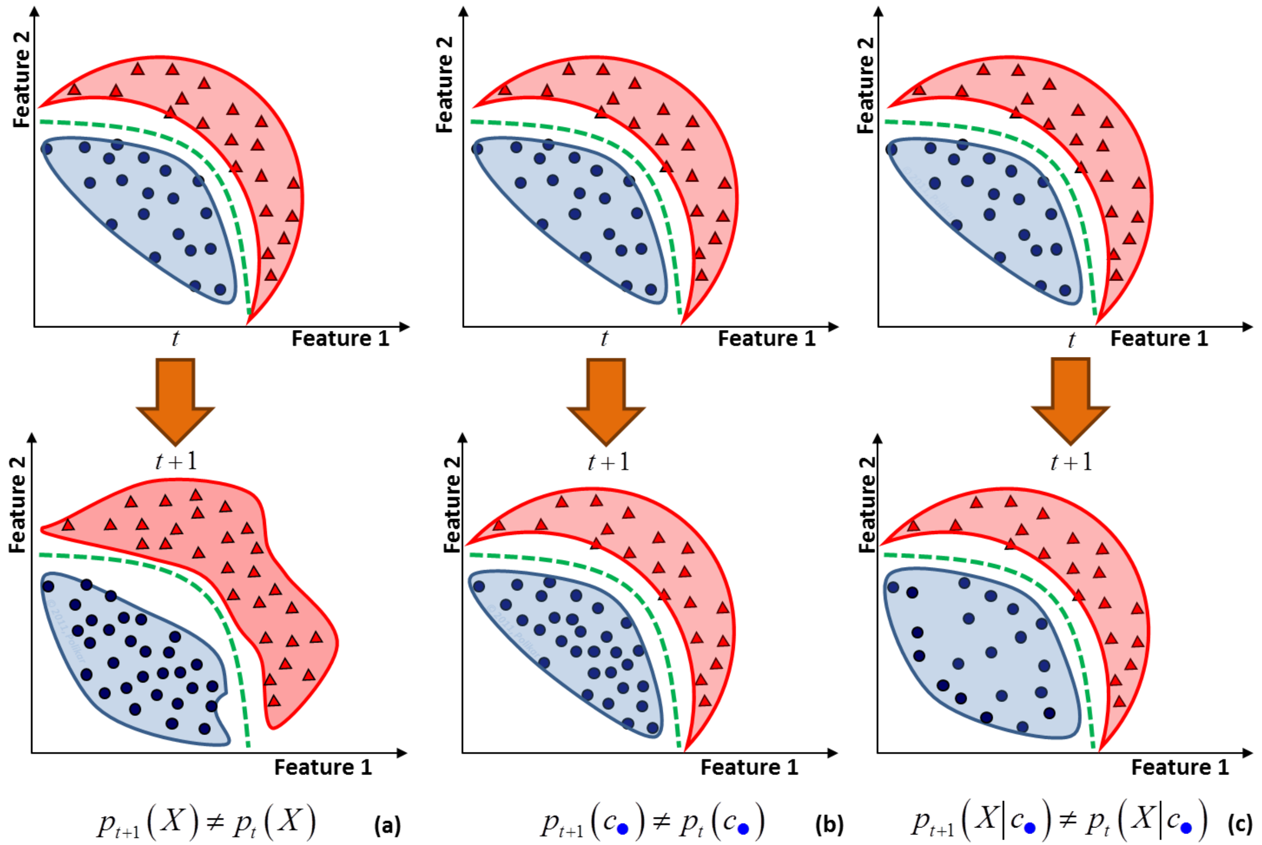


DRIFT TAXONOMY: WHAT IS CHANGING?

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.

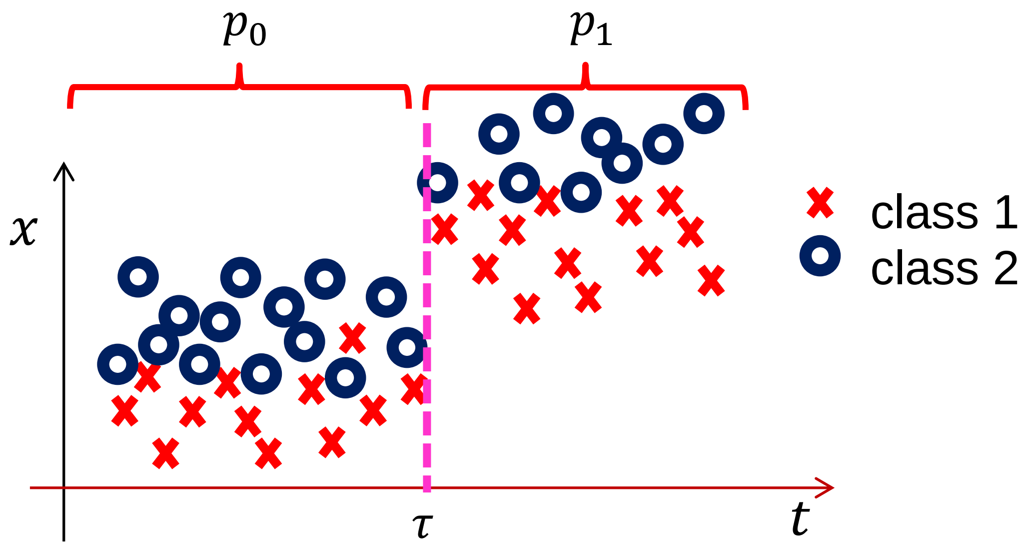


DRIFT TAXONOMY: TIME EVOLUTION

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

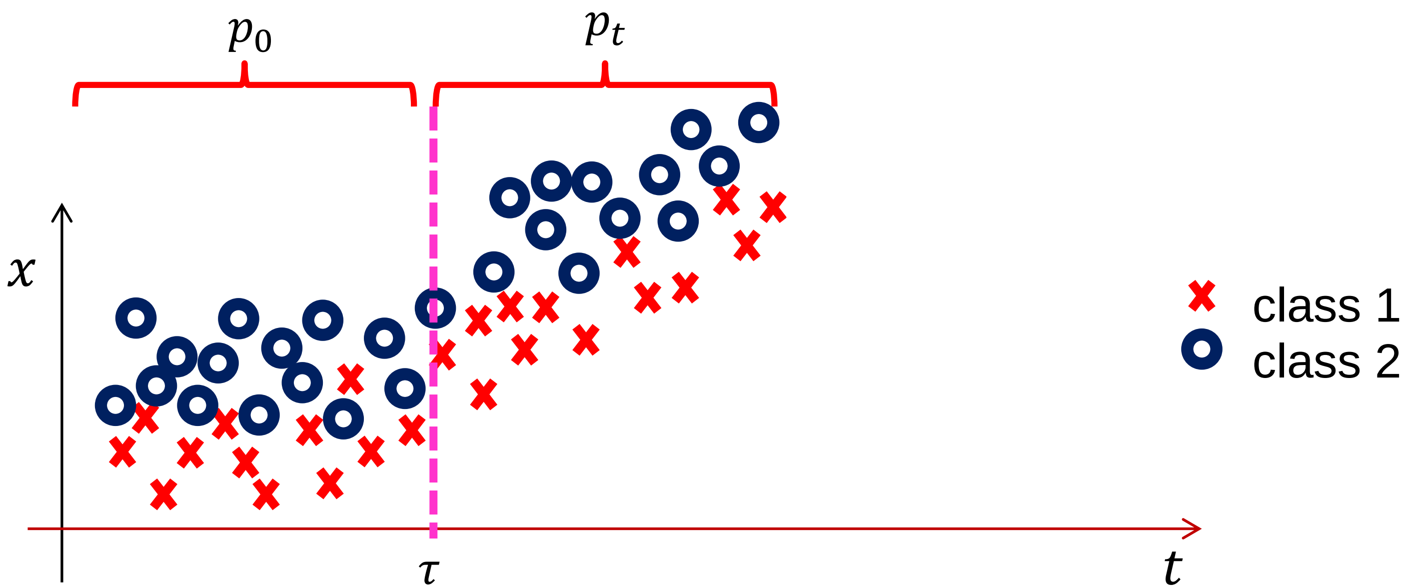


DRIFT TAXONOMY: TIME EVOLUTION

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 that *might* end up in another stationary state

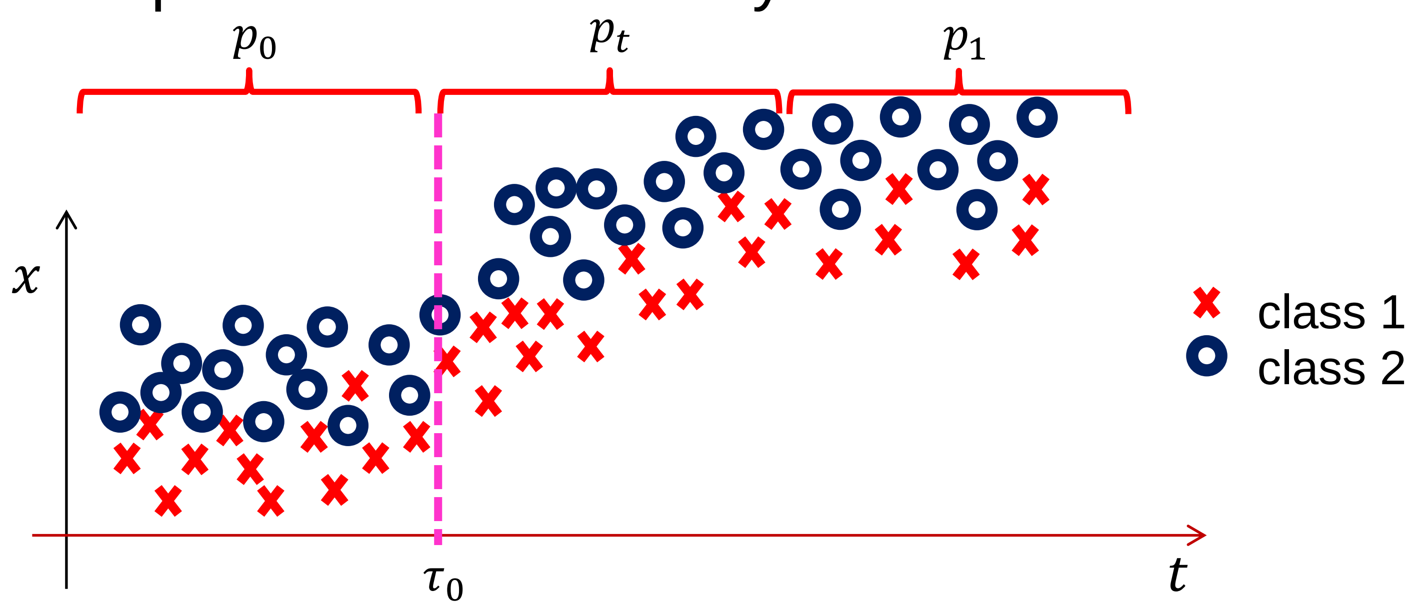


DRIFT TAXONOMY: TIME EVOLUTION

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

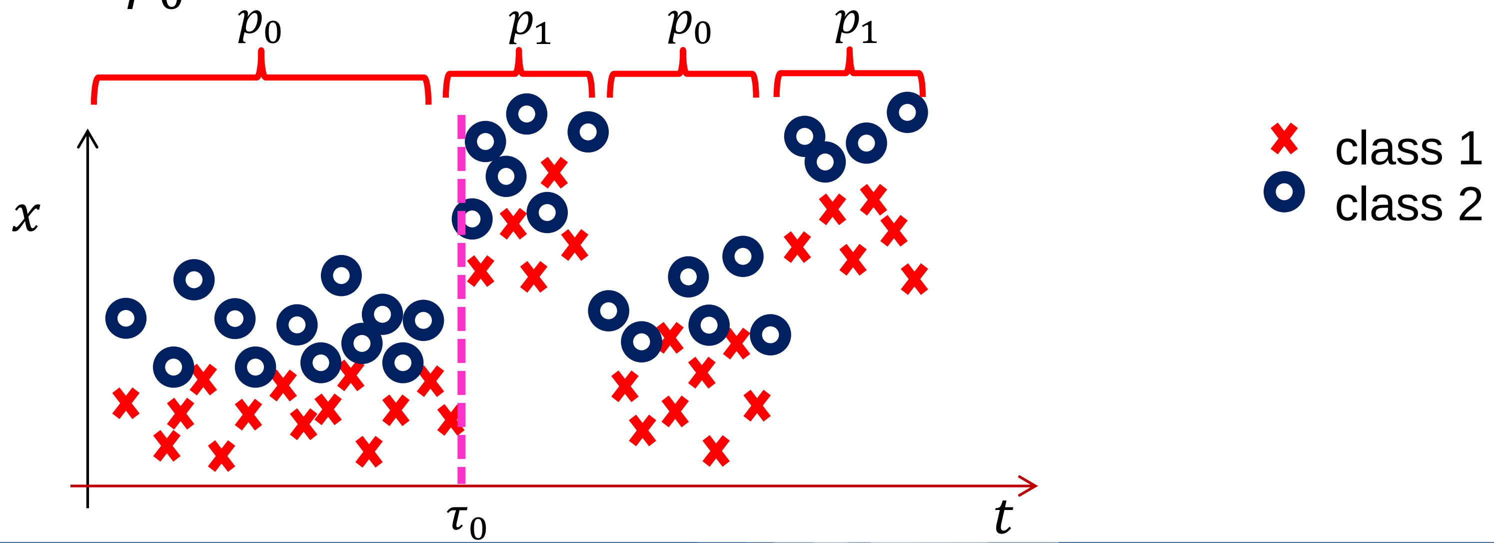


DRIFT TAXONOMY: TIME EVOLUTION

Recurring

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

After concept drift, it is possible that $\mathcal{X}$ goes back in its initial conditions ϕ_0

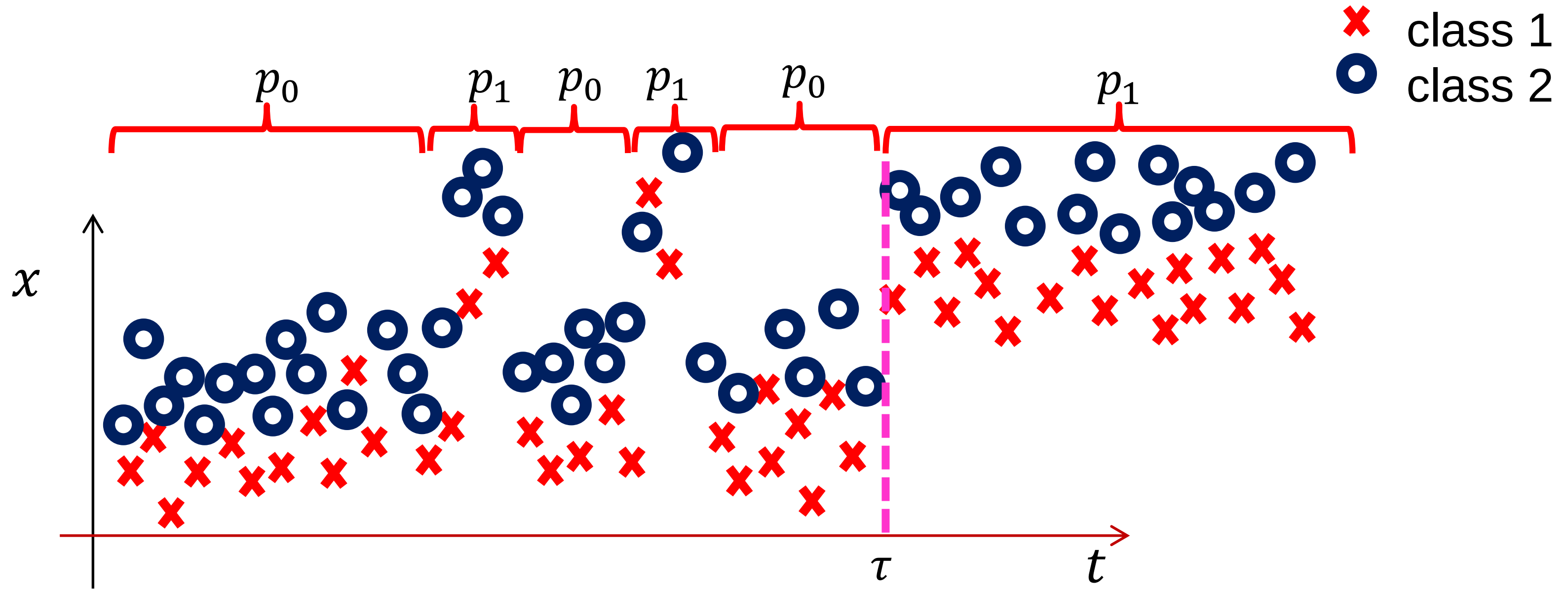


DRIFT TAXONOMY: TIME EVOLUTION

Gradual

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

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

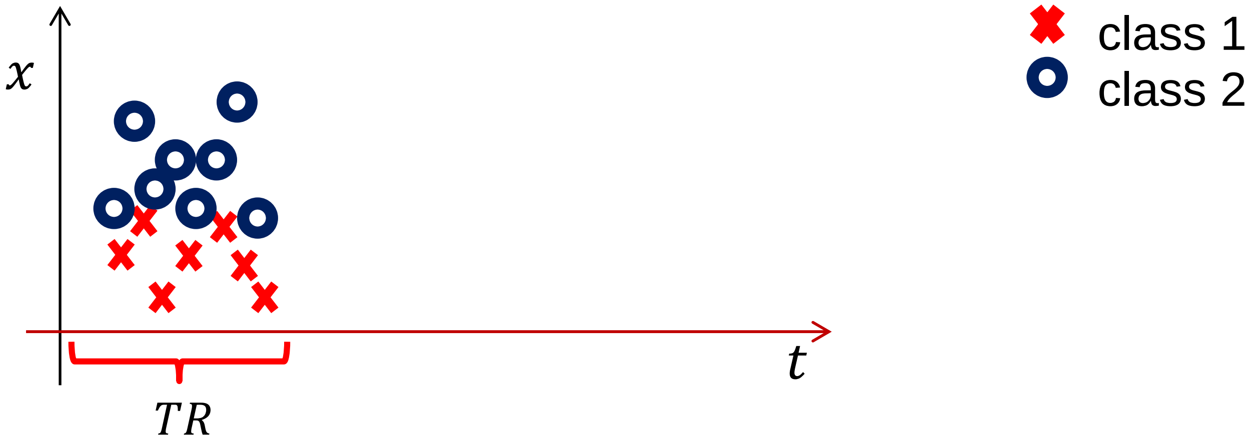


IS CONCEPT DRIFT A PROBLEM?

CLASSIFICATION OVER DATASTREAMS

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

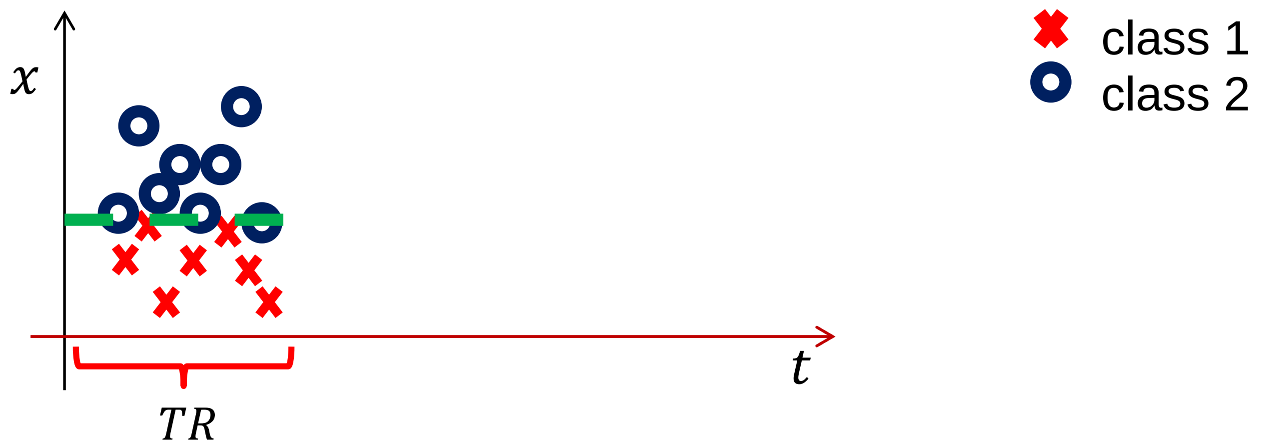
- The initial part of the stream is provided for training
- K is simply a threshold



CLASSIFICATION OVER DATASTREAMS

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

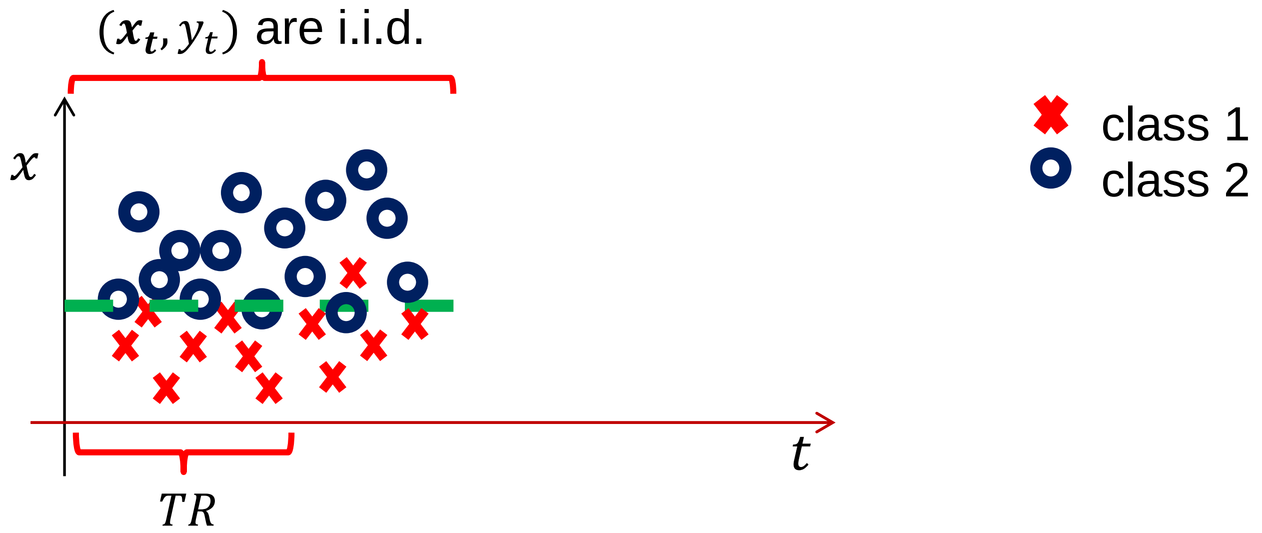
- The initial part of the stream is provided for training
- K is simply a threshold



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

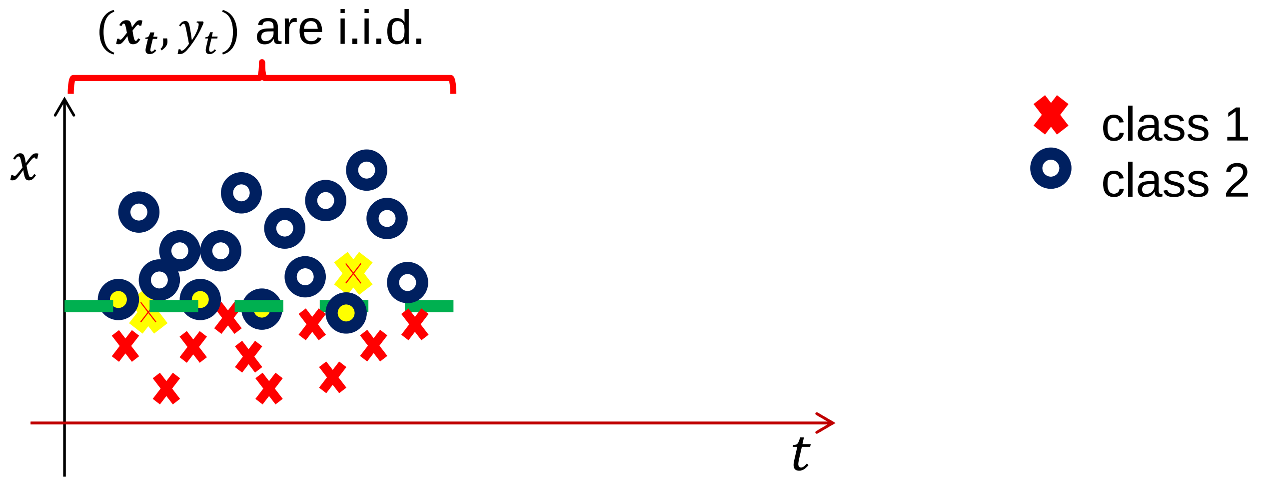


CLASSIFICATION OVER DATASTREAMS

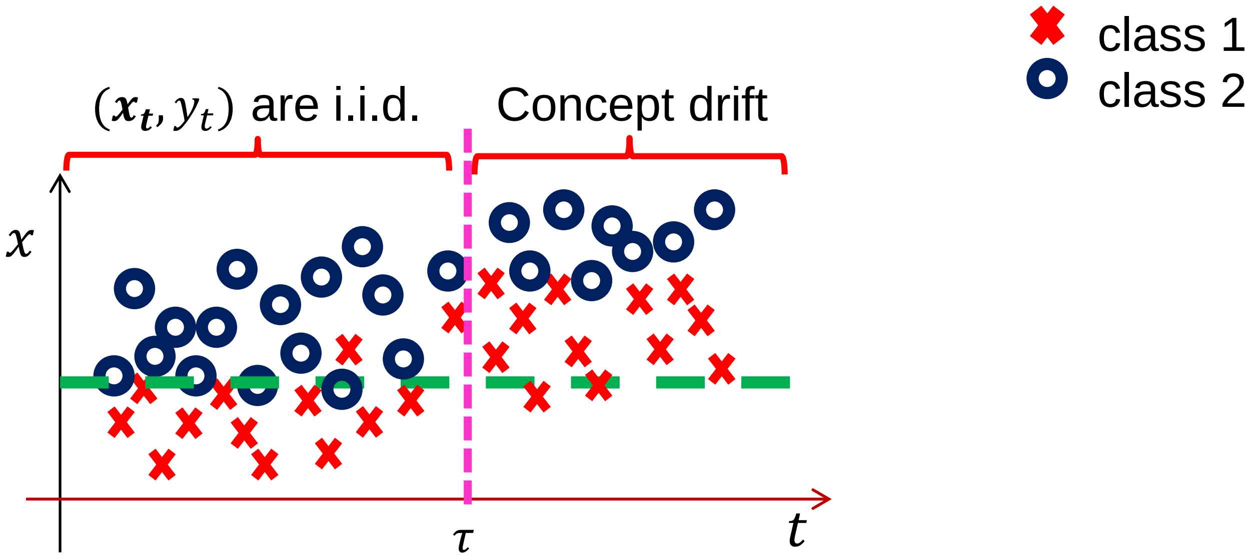
Consider as, an illustrative example, a simple 1-dimensional classification problem, where

- 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*

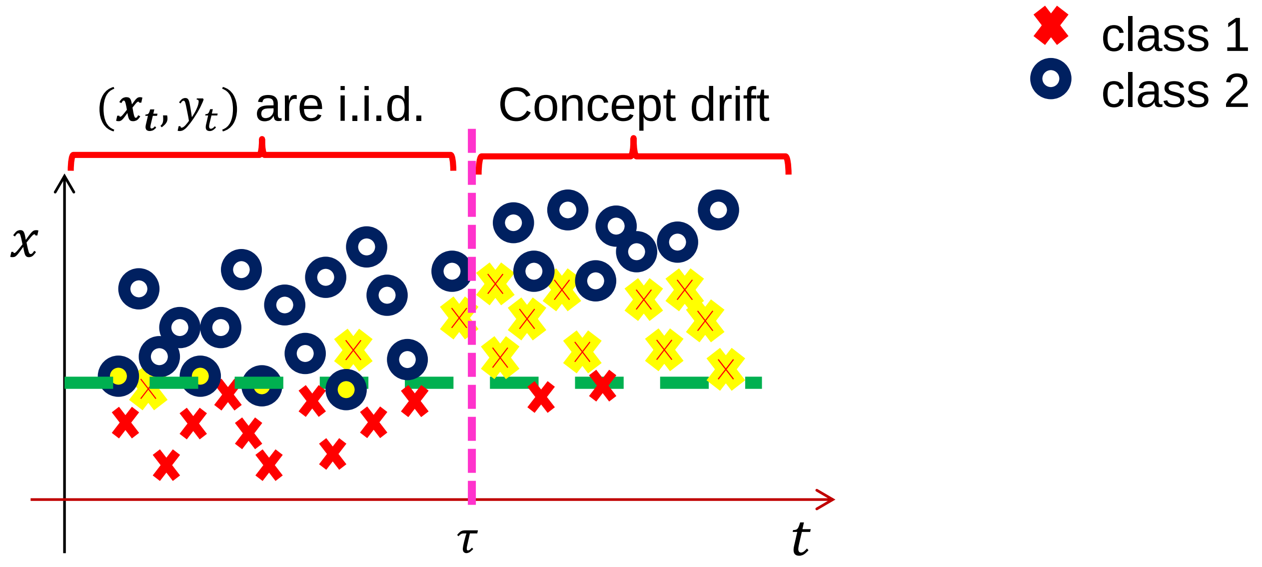


Unfortunately, when concept drift occurs, and ϕ changes,



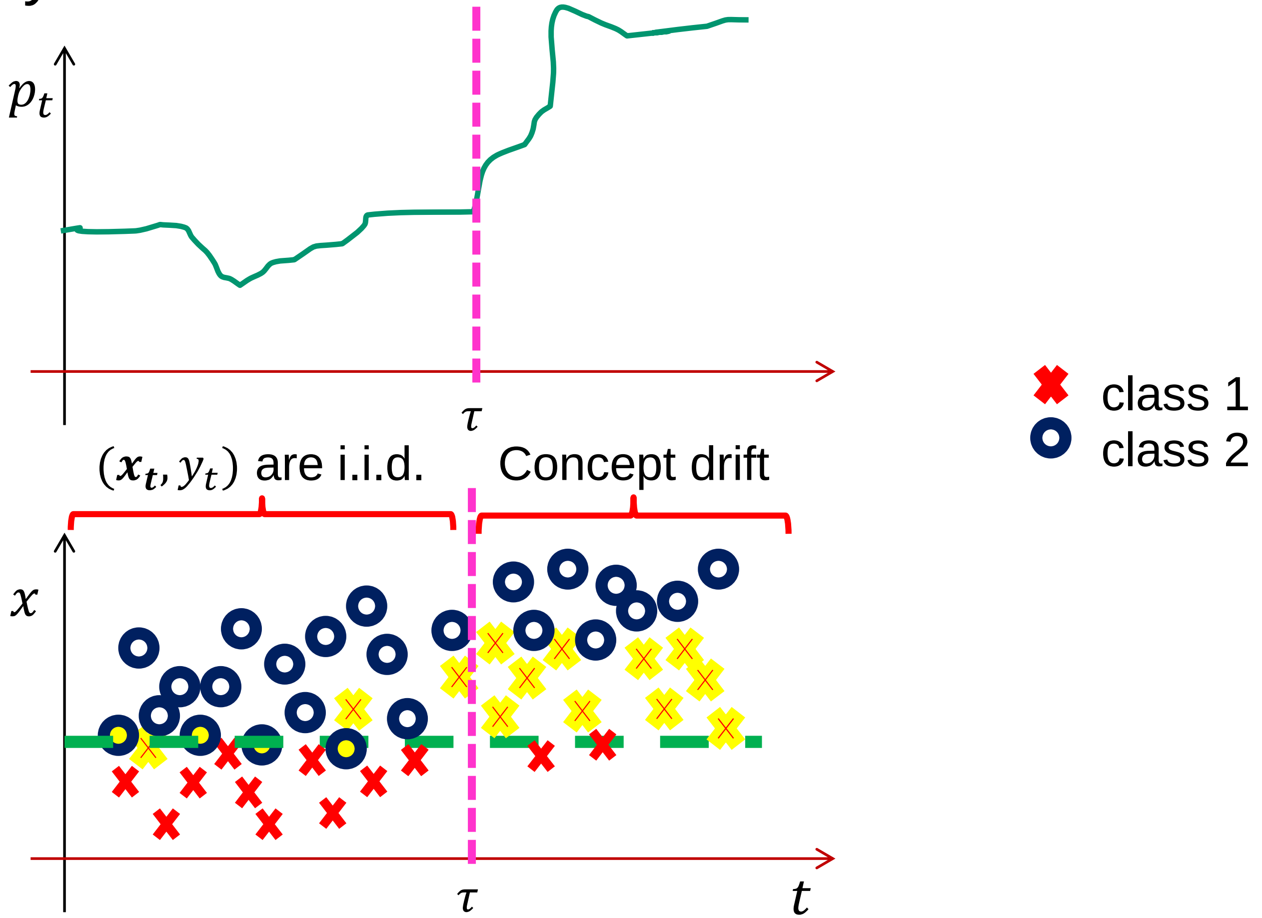
CLASSIFICATION OVER DATASTREAMS

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



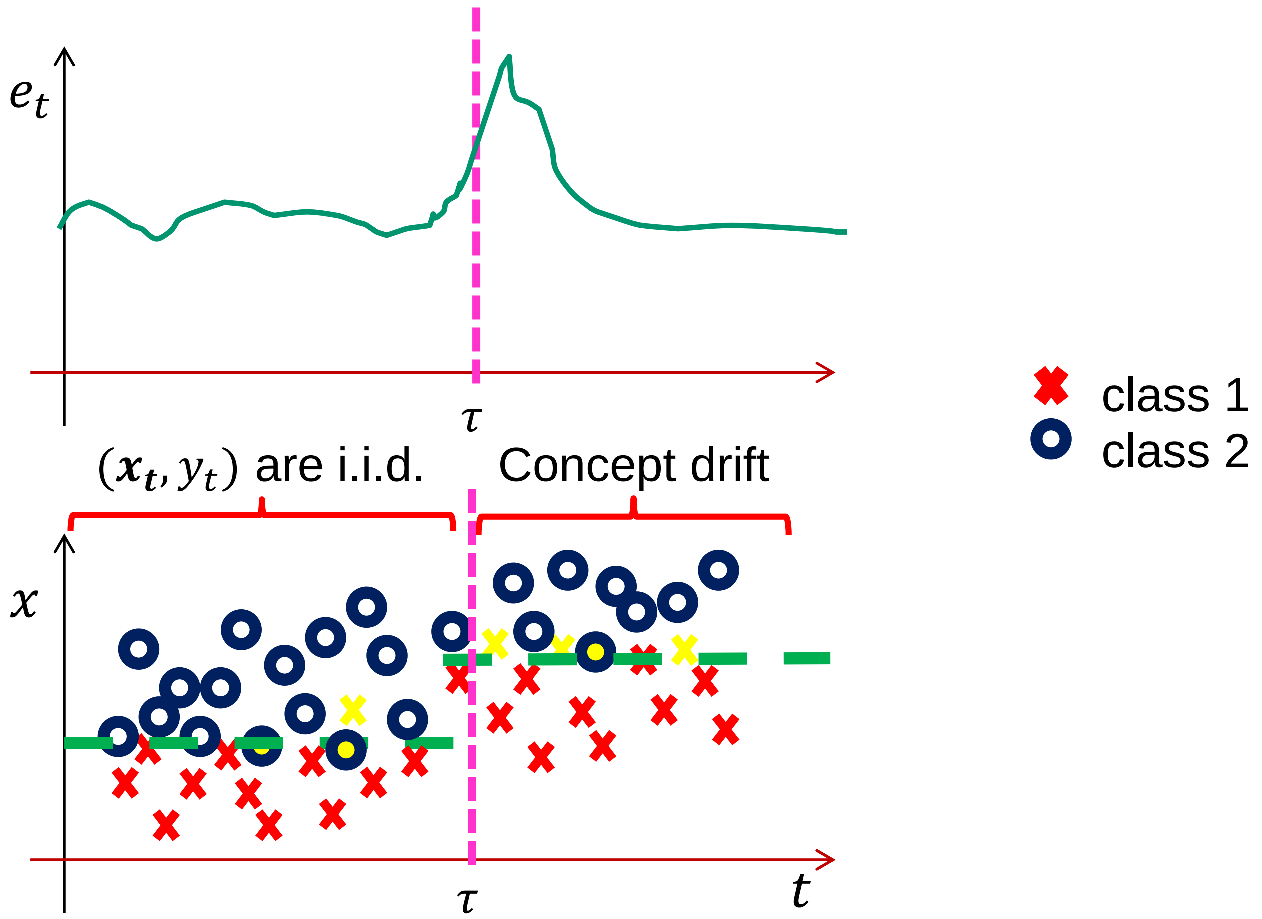
CLASSIFICATION OVER DATASTREAMS

Unfortunately, when **concept drift occurs**, and ϕ changes, things can be terribly worst, and the **average classification error** p_t typically **increases**



NEED FOR ADAPTATION

Adaptation is needed to **preserve** classifier performance



ADAPTATION

SIMPLE ADAPTATION STRATEGIES

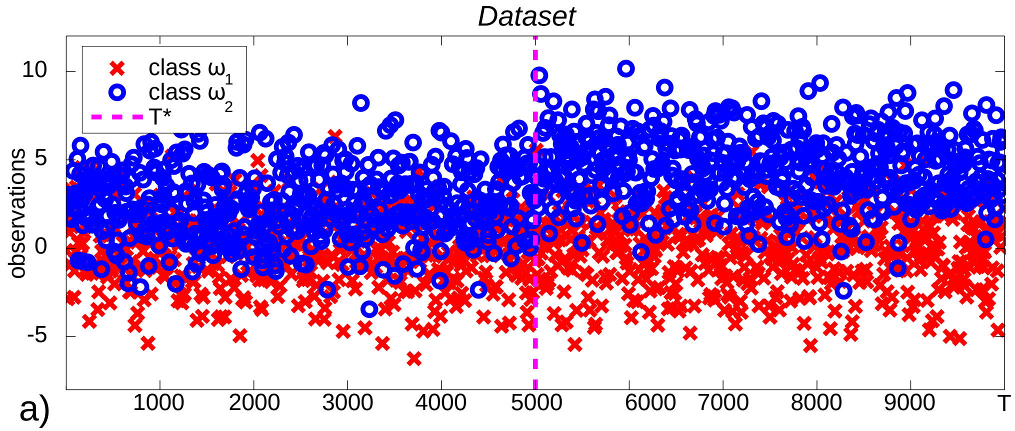
Consider two simple adaptation strategies

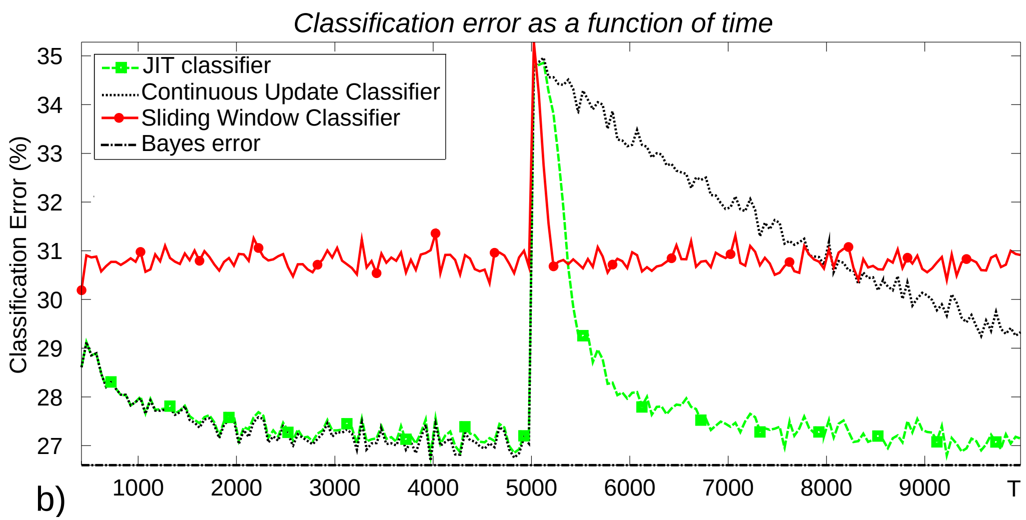
- Continuously update K_t using all supervised couples
- Train K_t using only the last δ supervised couples

SIMPLE ADAPTATION STRATEGIES

Consider two simple adaptation strategies

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

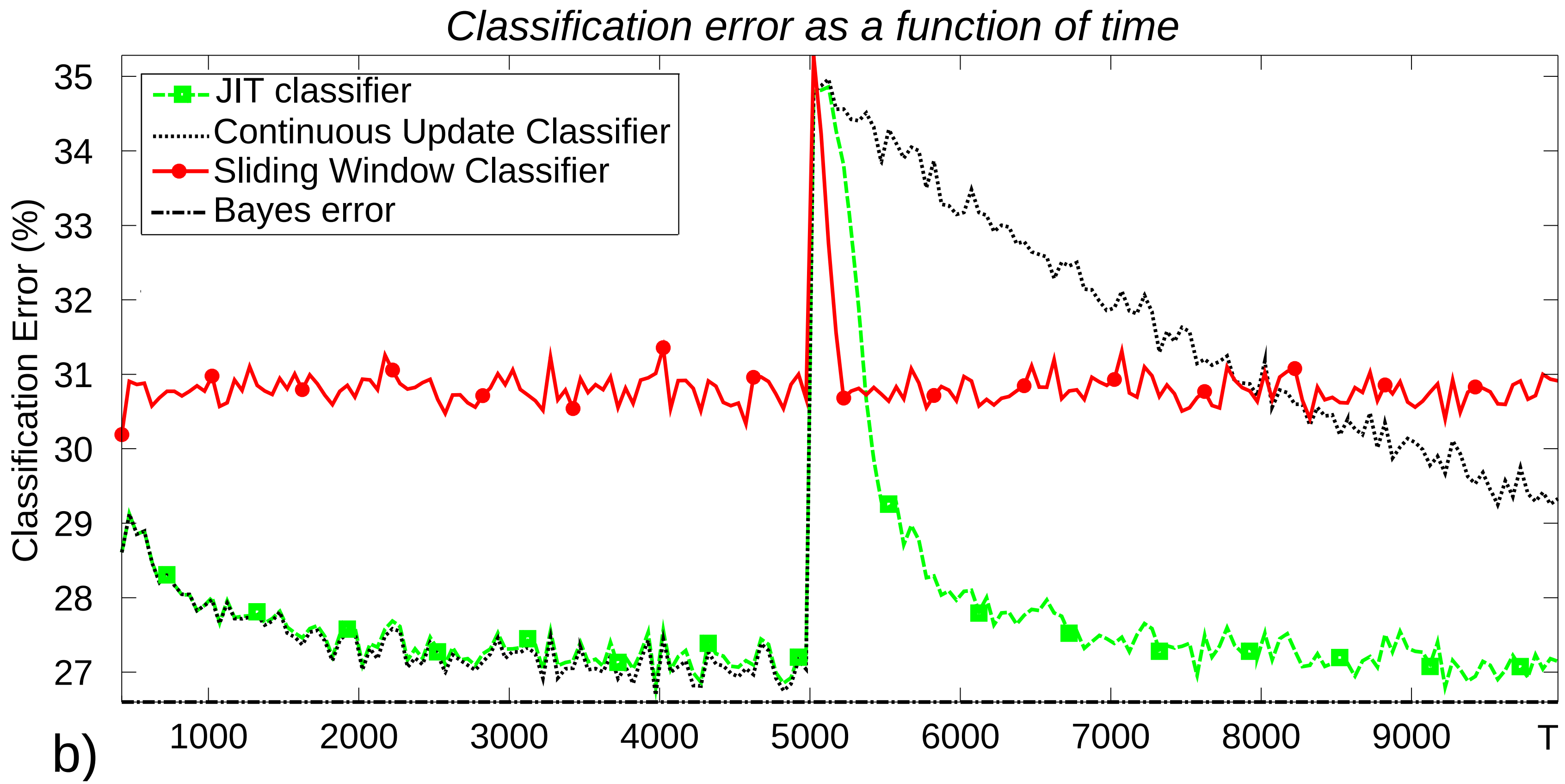




SIMPLE ADAPTATION STRATEGIES

Classification error of two simple adaptation strategies

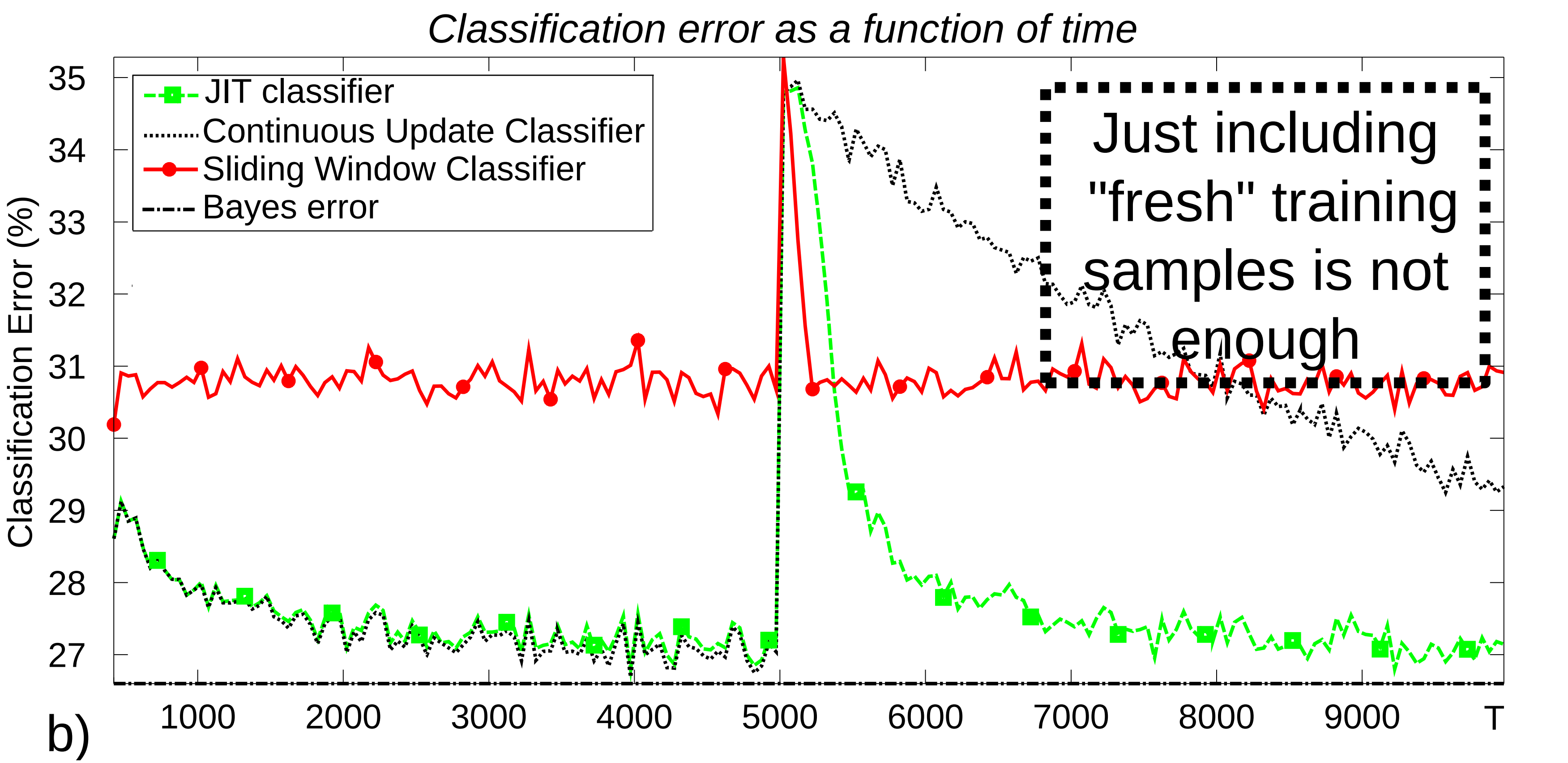
- Black dots: K_t uses all supervised couples at time t
- Red line: K_t uses only the last δ supervised couples



SIMPLE ADAPTATION STRATEGIES

Classification error of two simple adaptation strategies

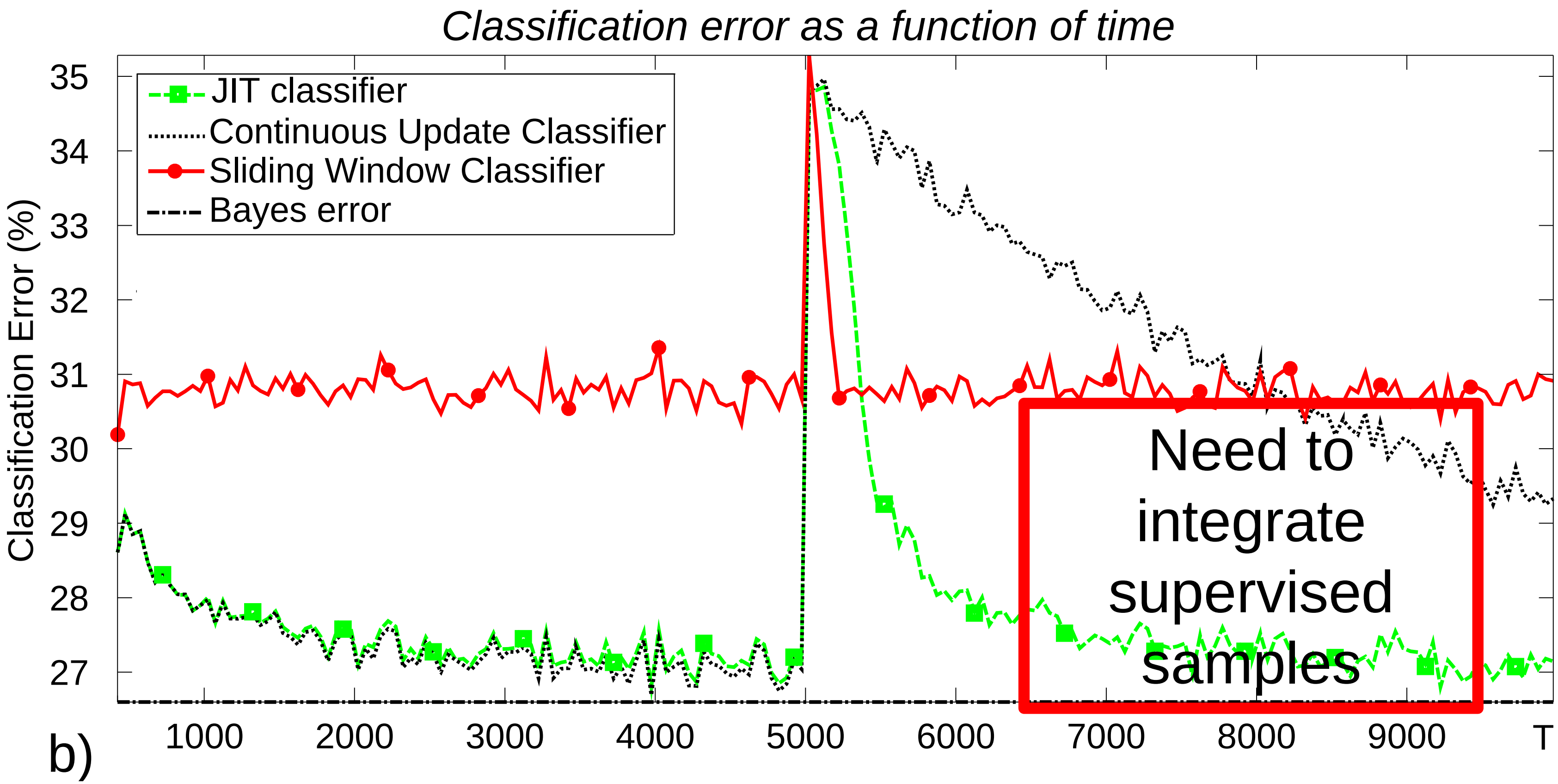
- Black dots: K_t uses all supervised couples at time t
- Red line: K_t uses only the last δ supervised couples



SIMPLE ADAPTATION STRATEGIES

Classification error of two simple adaptation strategies

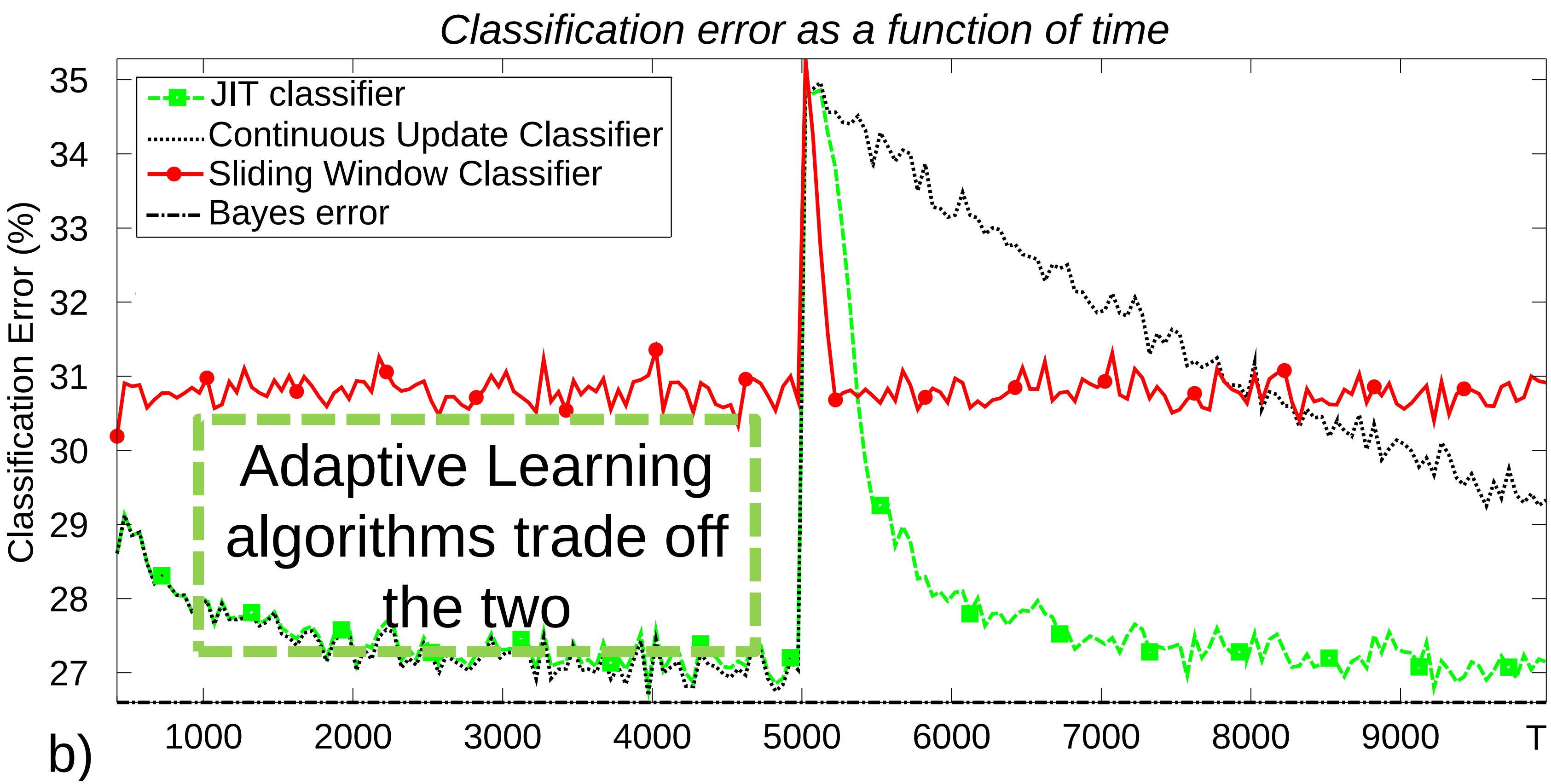
- Black dots: K_t uses all supervised couples at time t
- Red line: K_t uses only the last δ supervised couples



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



ADAPTATION 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

THE ACTIVE APPROACH

Detect-React Classifiers

ACTIVE APPROACHES

Peculiarities:

- Relies on an **explicit drift-detection mechanism**, change detection tests (CDTs)
- Specific **post-detection adaptation** procedures to isolate recent data generated after the change

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 CLASSIFICATION ERROR

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

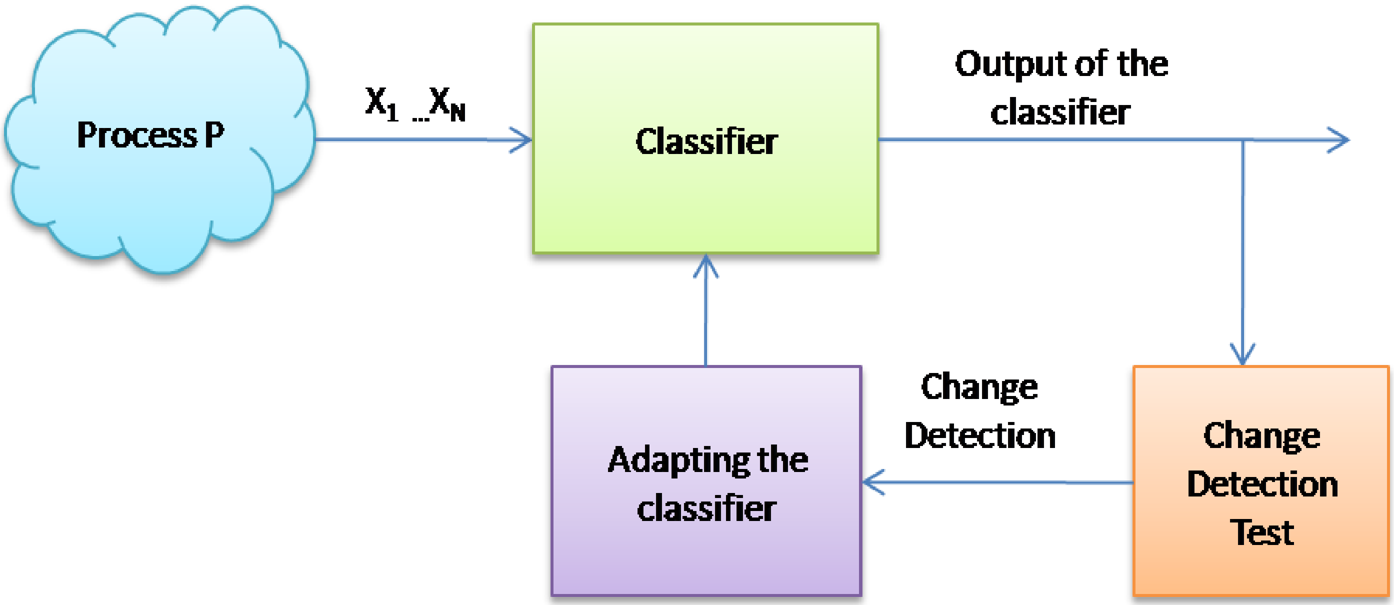
Pro:

- 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 CLASSIFICATION ERROR



MONITORING 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 sliding window follows a **Binomial** pdf

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

- Gaussian approximation when ν is sufficiently large

$$p_t = \frac{1}{\nu} \sum_{t=T-\nu}^{T} e_t \sim \frac{1}{\nu} \mathcal{B}(\pi_0, \nu) \approx \mathcal{N}\left(\pi_0, \frac{\pi_0(1 - \pi_0)}{\nu}\right)$$

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

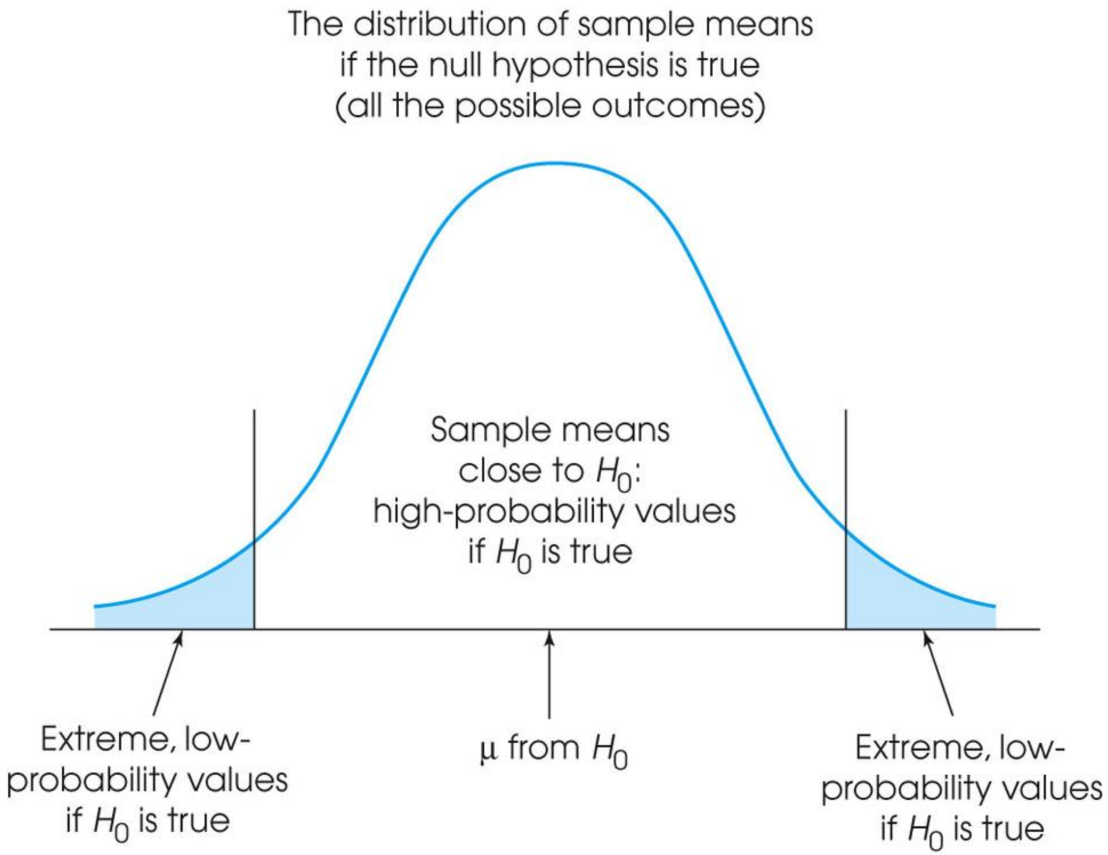
Basic idea behind Drift Detection Method (DDM):

J. Gama, P. Medas, G. Castillo, and P. Rodrigues*. "Learning with Drift Detection"* In Proc. of the 17th Brazilian Symp. on Artif. Intell. (SBIA). Springer, Berlin, 286–295, 2004

MONITORING CLASSIFICATION ERROR: DDM

Basic idea behind Drift Detection Method (DDM):

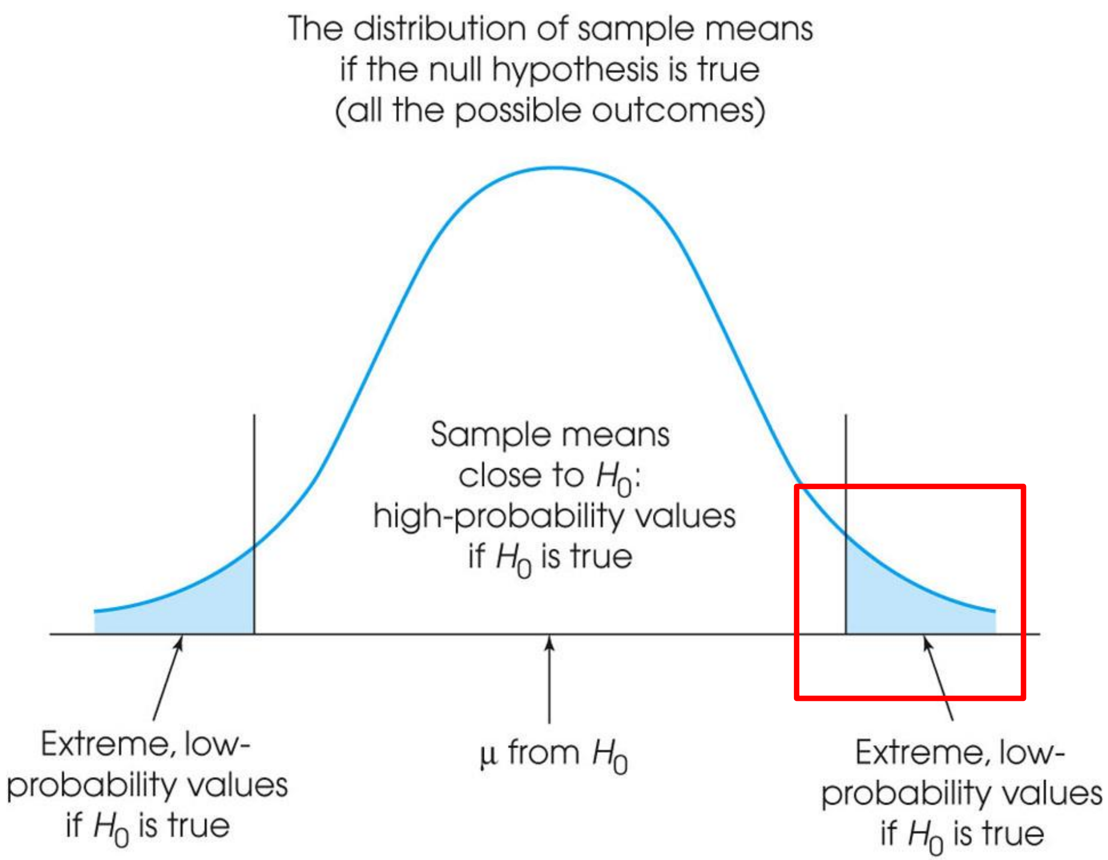
- Detect CD as **outliers** in the classification error



MONITORING CLASSIFICATION ERROR: DDM

Basic idea behind Drift Detection Method (DDM):

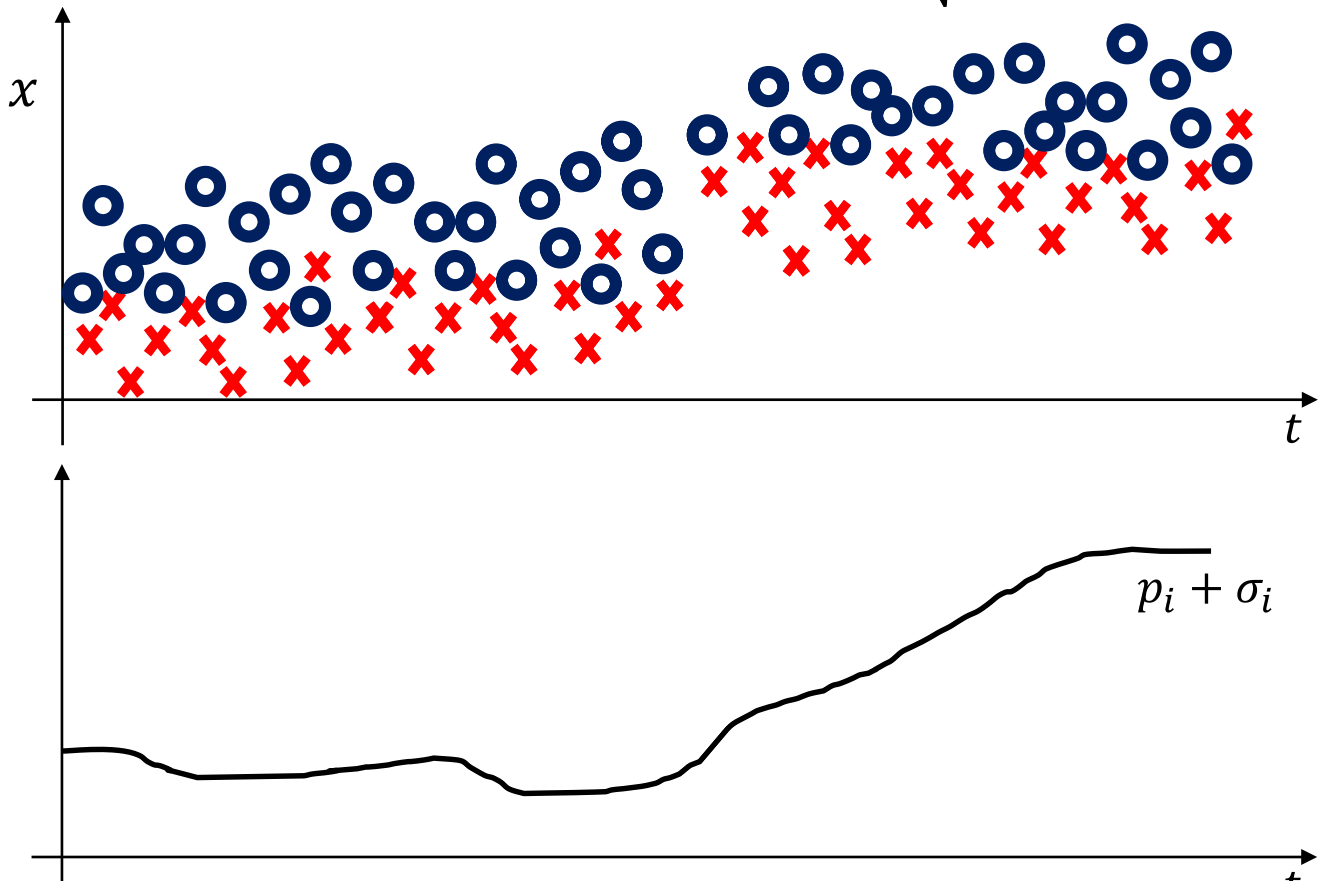
- Detect CD as **outliers** in the classification error
- Since in stationary conditions error will decrease, look for outliers in the right tail only



MONITORING CLASSIFICATION ERROR: DDM

Basic idea behind Drift Detection Method (DDM):

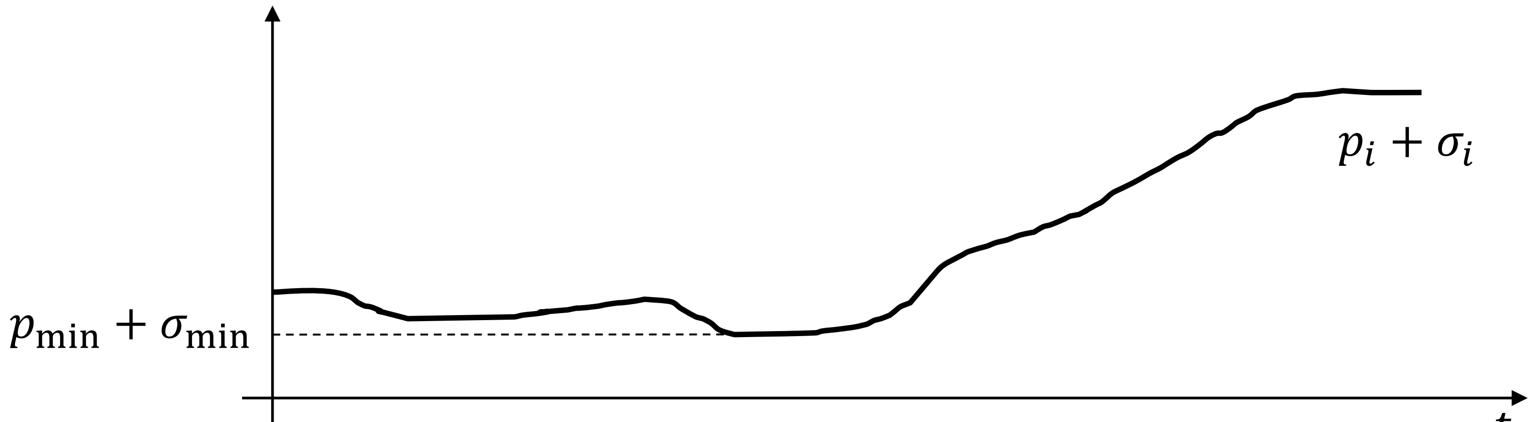
- Detect CD as **outliers** in the classification error
- Compute, over time p_i, and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$



MONITORING CLASSIFICATION ERROR: DDM

Basic idea behind Drift Detection Method (DDM):

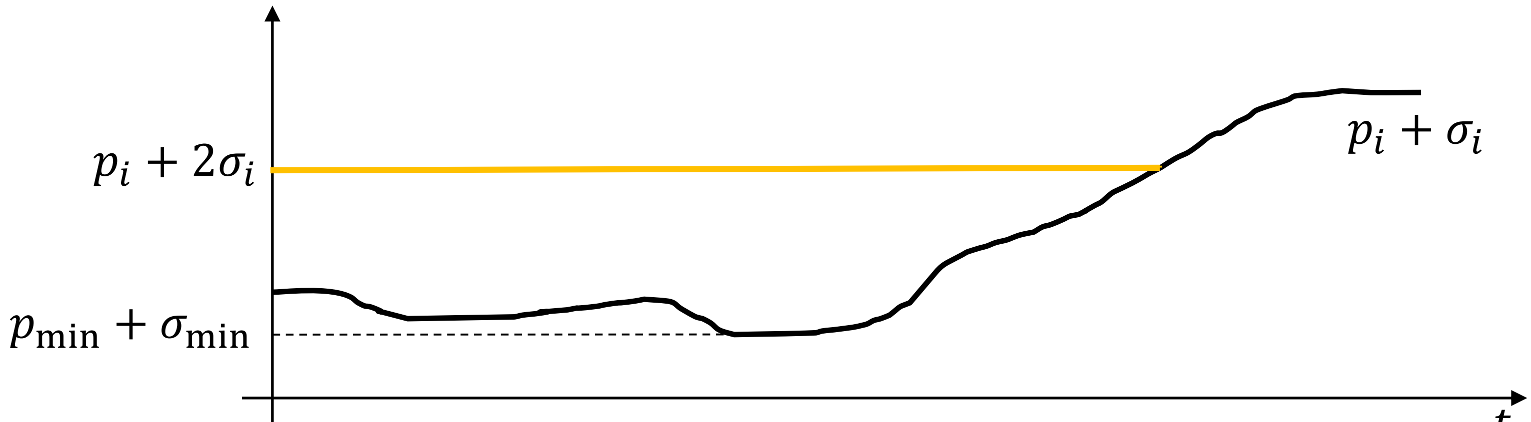
- Detect CD as **outliers** in the classification error
- Compute, over time p_i, and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$
- Let $p_{\min}$ be the minimum error, $\sigma_{\min} = \sqrt{\frac{p_{\min}(1-p_{\min})}{i}}$



MONITORING CLASSIFICATION ERROR: DDM

Basic idea behind Drift Detection Method (DDM):

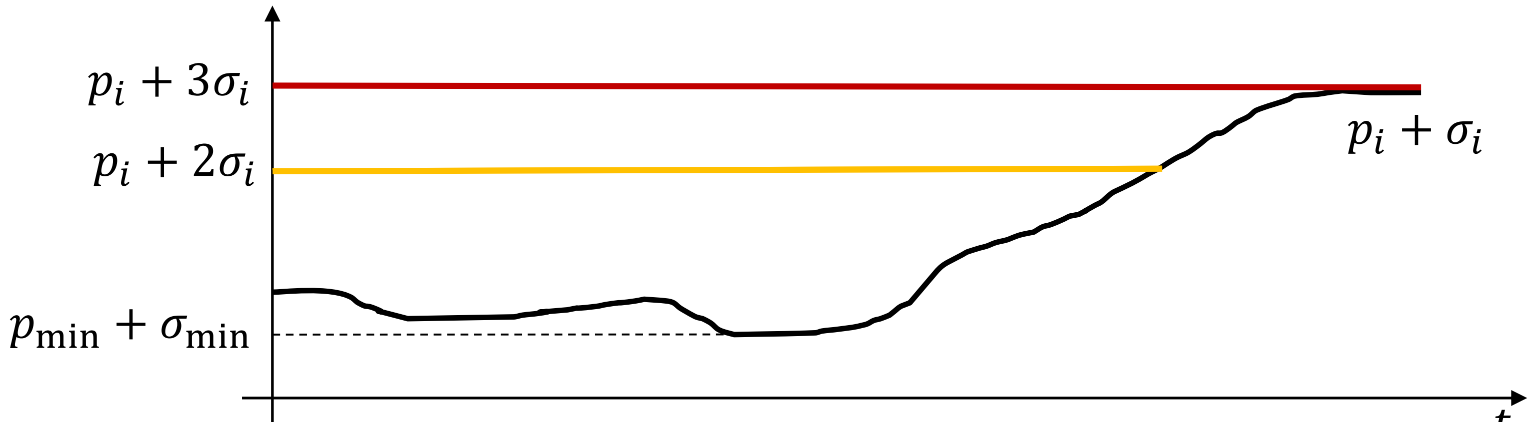
- Detect CD as **outliers** in the classification error
- Compute, over time p_i, and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$
- Let $p_{\min}$ be the minimum error, $\sigma_{\min} = \sqrt{\frac{p_{\min}(1-p_{\min})}{i}}$
- When $p_i + \sigma_i > p_{\min} + 2 * \sigma_{\min}$ raise a warning alert



MONITORING CLASSIFICATION ERROR: DDM

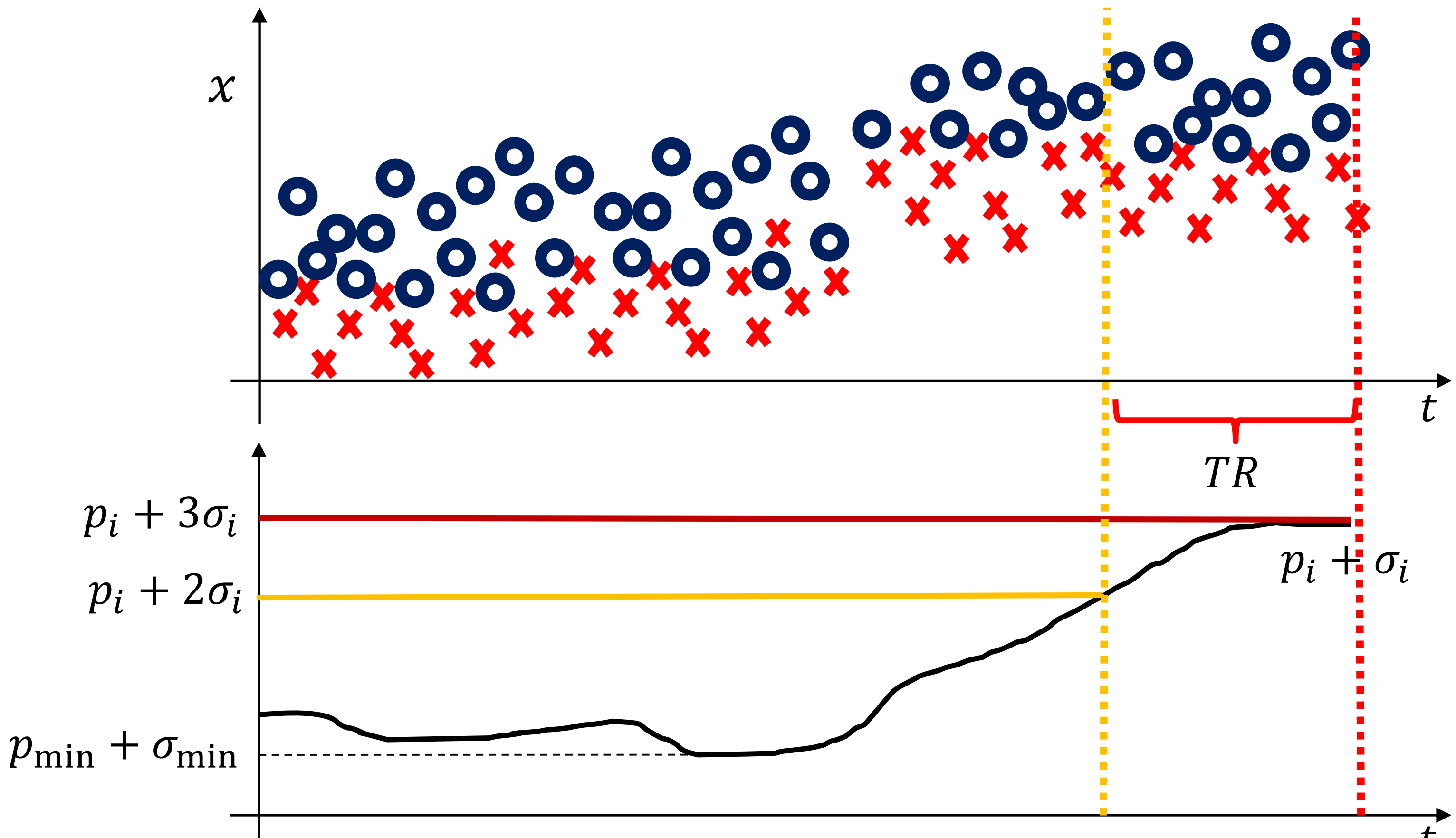
Basic idea behind Drift Detection Method (DDM):

- Detect CD as **outliers** in the classification error
- Compute, over time p_i, and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$
- Let $p_{\min}$ be the minimum error, $\sigma_{\min} = \sqrt{\frac{p_{\min}(1-p_{\min})}{i}}$
- When $p_i + \sigma_i > p_{\min} + 2 * \sigma_{\min}$ raise a warning alert
- When $p_i + \sigma_i > p_{\min} + 3 * \sigma_{\min}$ detect concept drift



POST-DETECTION RECONFIGURATION: DDM

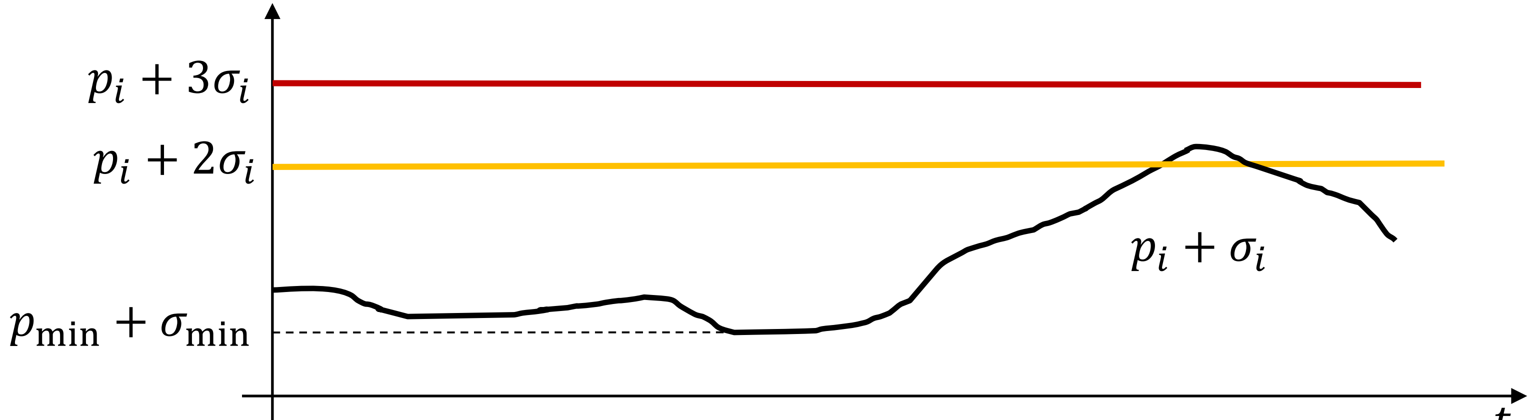
Use supervised samples in between warning and drift alert to reconfigure the classifier



POST-DETECTION RECONFIGURATION: 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



MONITORING CLASSIFICATION ERROR

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

- Average distance 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 CLASSIFICATION ERROR: EWMA

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

Compute EWMA statistic

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

Detect concept drift when

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

- $p_{0,t}$ is the average error estimated until time t
- σ_t is its standard deviation of the above estimator
- L_t is a threshold parameter

EWMA statistic is mainly influenced by **recent data**. CD is detected when the error on recent samples departs from $p_{0,t}$

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 CLASSIFICATION ERROR: EWMA

Most importantly:

- L_t can be set to **control the average run length** (ARL) of the test (the expected time between false positives)
- Like DDM, classifier **reconfiguration** is performed by monitoring Z_t also at a *warning level*

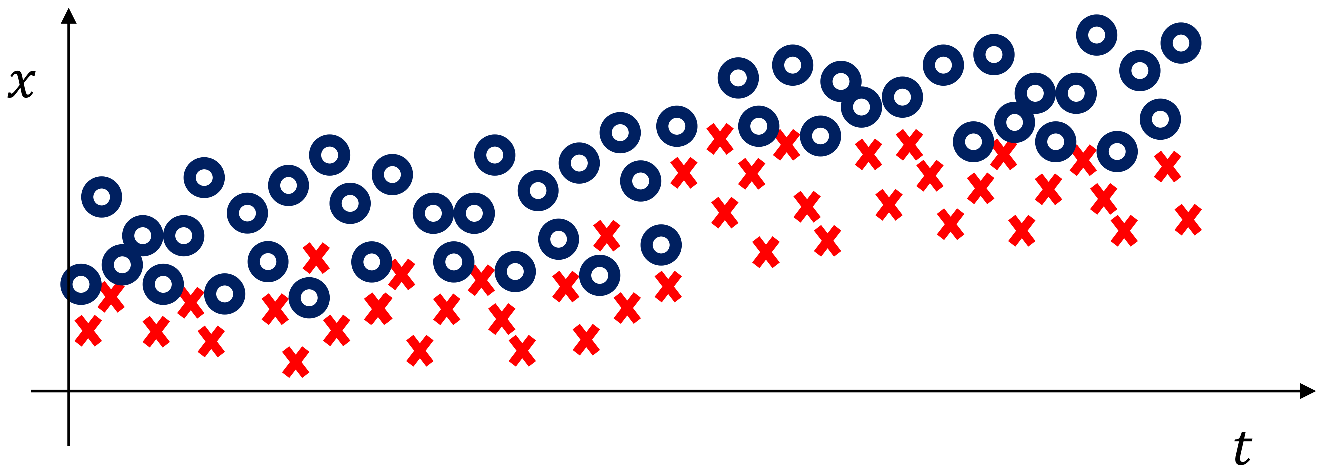
$$Z_t > p_{0,t} + 0.5\, L_t \sigma_t$$

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

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

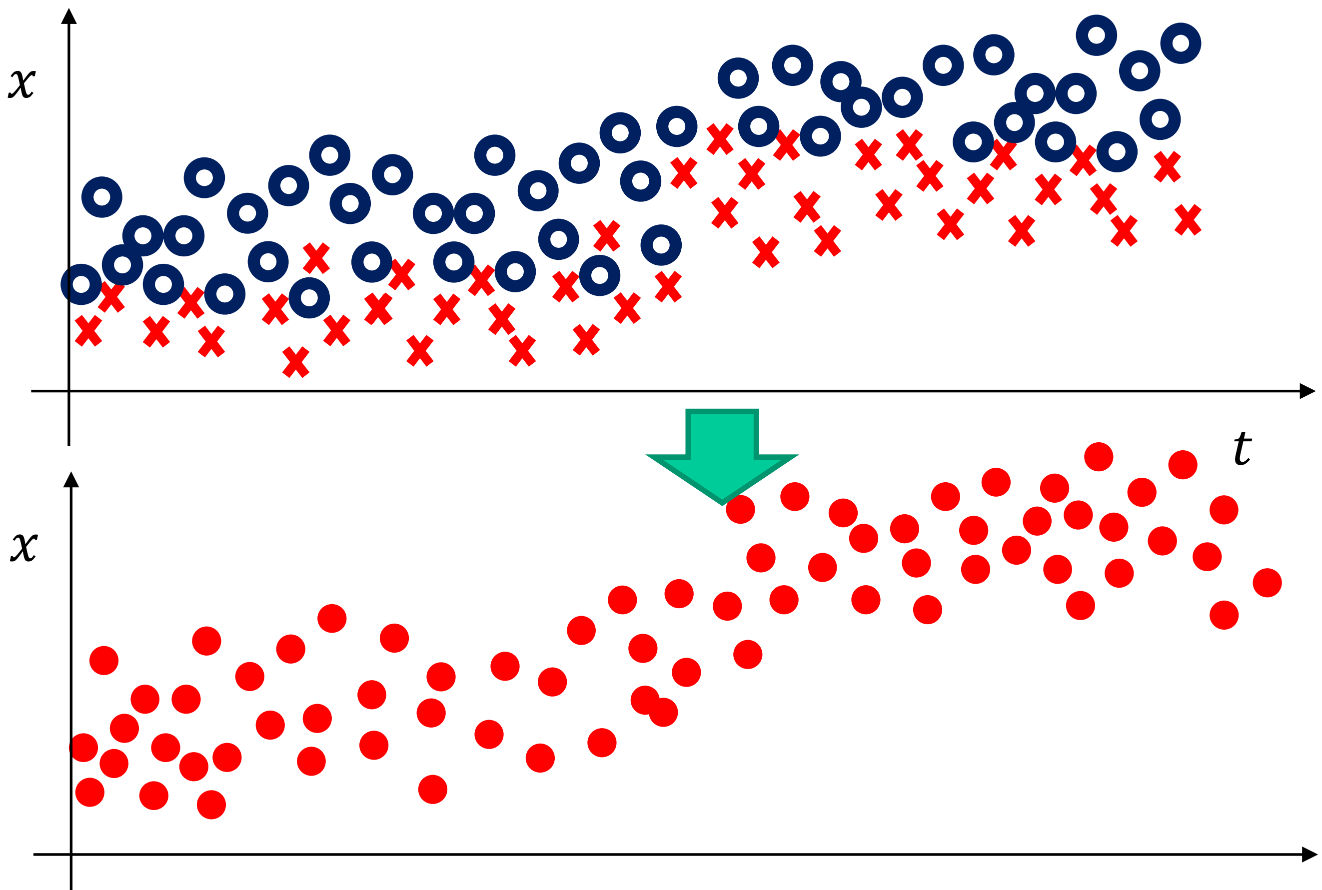
MONITORING THE RAW DATA

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

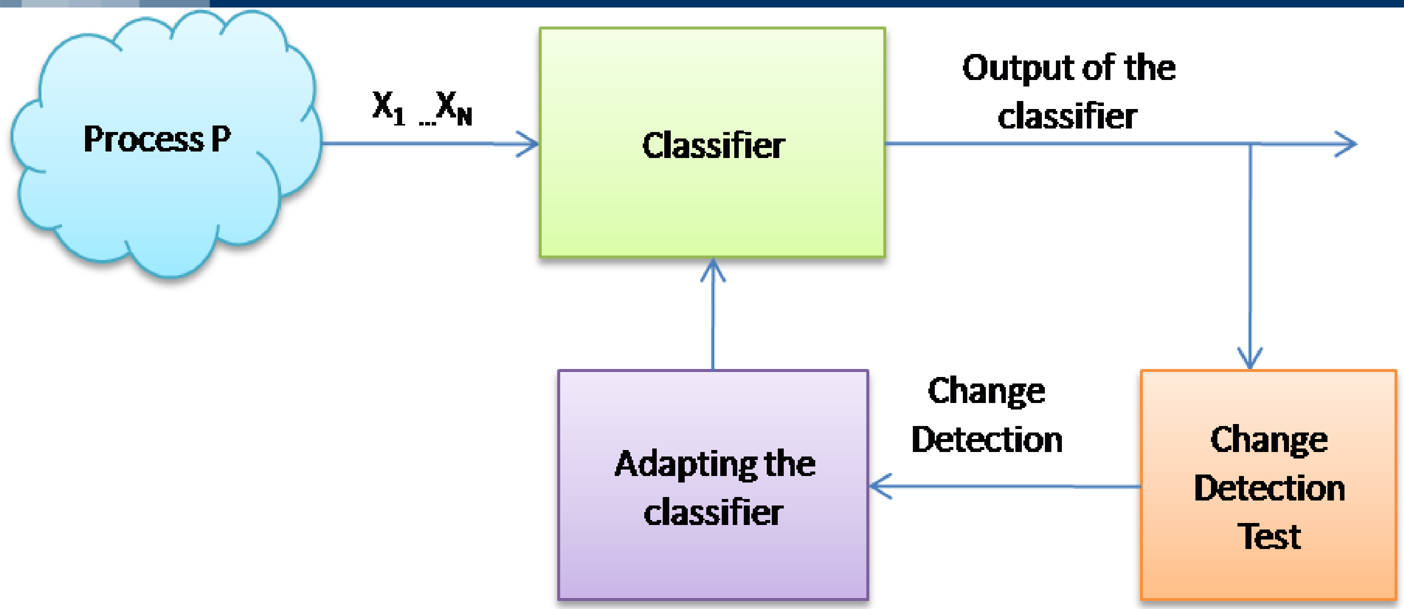


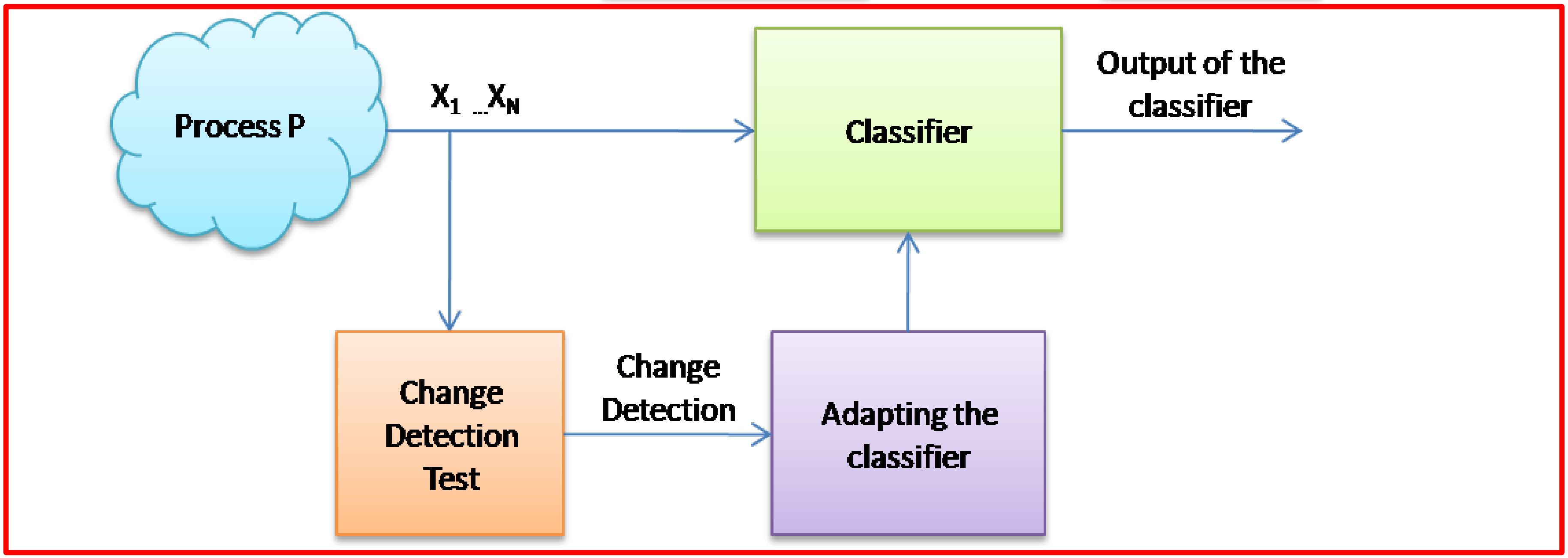
MONITORING THE RAW DATA

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



MONITORING THE RAW DATA





MONITORING THE RAW DATA

Pros:

- Monitoring $\phi(\boldsymbol{x})$ **does not require supervised samples**
- Enables the detection of both **real and virtual 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 CDTs on streams of multivariate data whose distribution is unknown

MONITORING MULTIVARIATE THE RAW DATA

One typically resort to:

- Operating component-wise (thus not performing a multivariate analysis)
- **Monitoring the log-likelihood** w.r.t. an additional model describing approximating $\phi(\boldsymbol{x})$ in stationary conditions

WHAT TO MONITOR: ICI-BASED CDT

Extracts Gaussian-distributed **features from non-overlapping windows** (such that they are **i.i.d.**)

- the sample mean over data windows

$$\mathrm{M}(s) = \sum_{t=(s-1)\nu+1}^{s\,\nu} x_t$$

- a power-law transform of the sample variance

$$\mathrm{V}(s) = \left(\frac{\mathrm{S}(s)}{\nu - 1}\right)^{h_0}$$

$S(s)$ is the sample variance over window yielding $M(s)$

Detection criteria: the Intersection of Confidence Intervals rule, an adaptive filtering technique for polynomial regression

C. Alippi, G. Boracchi, M. Roveri ***"A just-in-time adaptive classification system based on the intersection of confidence intervals rule"*****, Neural Networks, Elsevier vol. 24 (2011), pp. 791-800**

WHAT TO MONITOR: CI-CUSUM

Several features from non-overlapping windows including

- Sample moments
- Projections over the principal components
- Mann-Kendal statistic

Detection criteria: the cumulative sum of each of this feature is monitored to detect change in a CUSUM-like scheme

C. Alippi and M. Roveri, *"Just-in-time adaptive classifiers–part I: Detecting nonstationary changes,"* IEEE Transactions on Neural Networks, vol. 19, no. 7, pp. 1145–1153, 2008.

C. Alippi, M. Roveri, *"Just-in-time adaptive classifiers — part II: Designing the classifier,"* IEEE Transactions on Neural Networks, vol. 19, no. 12, pp. 2053–2064, 2008.

WHAT TO MONITOR: LOG-LIKELIHOOD

Fit a model (e.g. by GMM or KDE) $\hat{\phi}_0$ to **describe distribution** of **raw** (multivariate) data in stationary conditions

For each sample $\boldsymbol{x}$ **compute the log-likelihood** w.r.t. $\hat{\phi}_0$

$$\mathcal{L}(\boldsymbol{x}_t) = \log\left(\hat{\phi}_0(\boldsymbol{x}_t)\right) \in \mathbb{R}$$

Idea: Changes in the distribution of **the log-likelihood** indicate that $\hat{\phi}_0$ **is unfit** in describing unsupervised data, thus concept drift (possibly virtual) has occurred.

Detection Criteria: any monitoring scheme for **scalar i.i.d. datastream**

Kuncheva L.I., *" Change detection in streaming multivariate data using likelihood detectors"*, IEEE Transactions on Knowledge and Data Engineering, 2013, 25(5), 1175-1180

X. Song, M. Wu, C. Jermaine, S. Ranka *"Statistical change detection for multi-dimensional data"* In Proceedings of the 13th ACM SIGKDD (KDD 2007)

JUST-IN-TIME CLASSIFIERS

A methodology for designing adaptive classifiers

JUST-IN-TIME CLASSIFIERS

JIT classifiers are described in terms of :

- **concept representations**
- **operators** for concept representations

JIT classifiers are able to:

- detect abrupt CD (both real or virtual)
- Identify and take advantage of recurrent concepts

JIT classifiers leverage:

- **sequential techniques to detect CD,** monitoring both classification error and raw data distribution
- **statistical techniques to identify recurrent concepts**

Most of solutions for recurrent concepts are among passive approaches (see reference below for a survey)

C. Alippi, G. Boracchi, M. Roveri "*Just In Time Classifiers for Recurrent Concepts*" IEEE Transactions on Neural Networks and Learning Systems, 2013. vol. 24, no.4, pp. 620 -634

1- Build concept $\boxed{C_0 = (Z_0, F_0, D_0)}$ from the training sequence;
2- $Z_{\text{rec}} = \emptyset$ and $i = 0$;
3- **while** *(x_t is available)* **do**
4- $\quad \mathcal{U}(C_i, \{x_t\}) \to C_i$;
5- $\quad$ **if** *(y_t is available)* **then**
6- $\qquad \mathcal{U}(C_i, \{(x_t, y_t)\}) \to C_i$;
$\quad$ **end**
7- $\quad$ **if** *($\mathcal{D}(C_i) = 1$)* **then**
8- $\qquad i = i + 1$;
9- $\qquad \Upsilon(C_{i-1}) \to (C_k, C_l)$;
10- $\qquad C_i = C_l$;
11- $\qquad C_{i-1} = C_k$;
12- $\qquad Z_{\text{rec}} = \bigcup_{\substack{\mathcal{E}(C_i, C_j)=1 \\ 0 \le j < i}} Z_j$;
$\quad$ **end**
13- $\quad$ **if** *(y_t is not available)* **then**
14- $\qquad \widehat{y}_t = K(Z_i \cup Z_{\text{rec}}, x_t)$.
$\quad$ **end**
end

Concept Representations

$$C = (Z, F, D)$$

- Z : set of supervised samples
- F : set of features for assessing concept equivalence
- D : set of features for detecting concept drift

AN EXAMPLE OF CONCEPT REPRESENTATIONS

$$C_i = (Z_i, F_i, D_i)$$

- $Z_i = \{(\boldsymbol{x_0}, y_0), \dots, (\boldsymbol{x_n}, y_n)\}$: **supervised samples** provided during the i^{th} concept
- F_i **features describing** $p(\boldsymbol{x})$ of the i^{th} concept. We take:
  - the sample mean $M(\cdot)$
  - the power-low transform of the sample variance $V(\cdot)$

  extracted from **non-overlapping sequences**
- D_i **features for detecting** concept drift. These include:
  - the sample mean $M(\cdot)$
  - the power-low transform of the sample variance $V(\cdot)$
  - the average classification error $p_t(\cdot)$

  extracted from **non-overlapping sequences**

In **stationary conditions** features are **i.i.d.**

JIT CLASSIFIERS: THE ALGORITHM

1- Build concept $C_0 = \big(Z_0, F_0, D_0\big)$ from the training sequence;
2- $Z_{\text{rec}} = \emptyset$ and $i = 0$;
3- **while** *(x_t is available)* **do**
4- $\quad \mathcal{U}(C_i, \{x_t\}) \rightarrow C_i$;
5- $\quad$ **if** *(y_t is available)* **then**
6- $\quad\quad \mathcal{U}(C_i, \{(x_t, y_t)\}) \rightarrow C_i$;
$\quad$ **end**
7- $\quad$ **if** *($\mathcal{D}(C_i) = 1$)* **then**
8- $\quad\quad i = i + 1$;
9- $\quad\quad \Upsilon(C_{i-1}) \rightarrow (C_k, C_l)$;
10- $\quad\quad C_i = C_l$;
11- $\quad\quad C_{i-1} = C_k$;
12- $\quad\quad Z_{\text{rec}} = \bigcup_{\substack{\mathcal{E}(C_i, C_j)=1 \\ 0 \leq j < i}} Z_j$;
$\quad$ **end**
13- $\quad$ **if** *(y_t is not available)* **then**
14- $\quad\quad \widehat{y}_t = K(Z_i \cup Z_{\text{rec}}, x_t)$.
$\quad$ **end**
end

Concept Representations

$$C = (Z, F, D)$$

- Z : set of supervised samples
- F : set of features for assessing concept equivalence
- D : set of features for detecting concept drift

Operators for Concepts

- $\mathcal{D}$ concept-drift detection
- Υ concept split
- $\mathcal{E}$ equivalence operators
- $\mathcal{U}$ concept update

JIT CLASSIFIERS: THE ALGORITHM

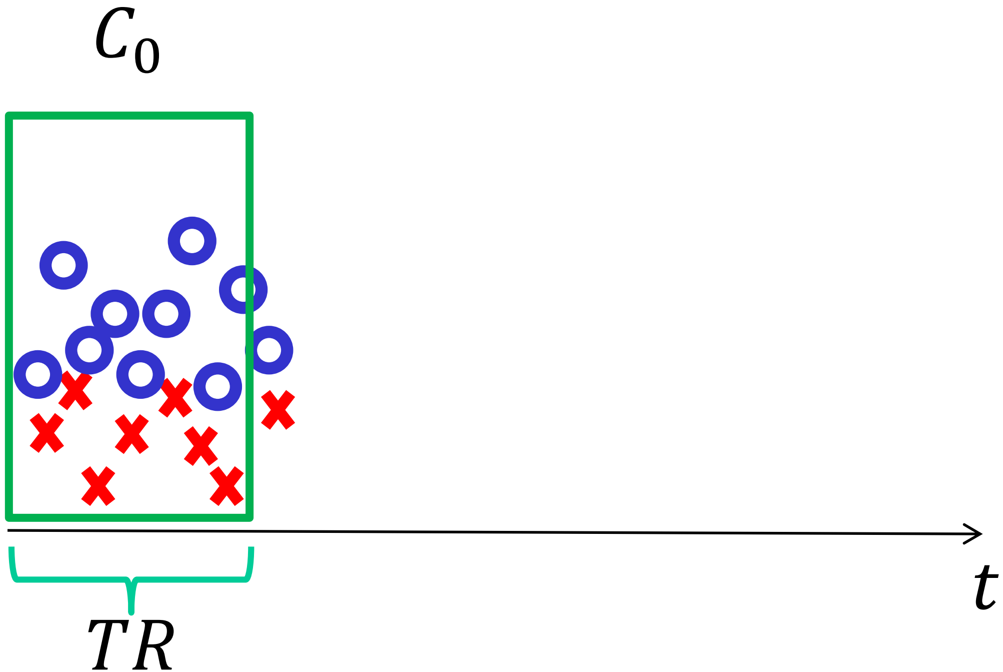
1- Build concept $C_0 = (Z_0, F_0, D_0)$ from the training sequence;
2- $Z_{\text{rec}} = \emptyset$ and $i = 0$;
3- **while** *(x_t is available)* **do**
4- $\mathcal{U}(C_i, \{x_t\}) \rightarrow C_i$;
5- **if** *(y_t is available)* **then**
6- $\mathcal{U}(C_i, \{(x_t, y_t)\}) \rightarrow C_i$;
end
7- **if** *($\mathcal{D}(C_i) = 1$)* **then**
8- $i = i + 1$;
9- $\Upsilon(C_{i-1}) \rightarrow (C_k, C_l)$;
10- $C_i = C_l$;
11- $C_{i-1} = C_k$;
12- $Z_{\text{rec}} = \bigcup_{\substack{\mathcal{E}(C_i, C_j) = 1 \\ 0 \leq j < i}} Z_j$;
end
13- **if** *(y_t is not available)* **then**
14- $\widehat{y}_t = K(Z_i \cup Z_{\text{rec}}, x_t)$.
end
end

Use the initial training sequence to build the concept representation C_0

JIT CLASSIFIER: CONCEPT REPRESENTATIONS

Build C_0, a **practical representation** of the **current concept**

- Characterize both $p(\boldsymbol{x})$ and $p(y|\boldsymbol{x})$ in stationary conditions



JIT CLASSIFIERS: THE ALGORITHM

1- Build concept $C_0 = \big(Z_0, F_0, D_0\big)$ from the training sequence;
2- $Z_{\text{rec}} = \emptyset$ and $i = 0$;
3- **while** *(x_t is available)* **do**
4- $\quad \mathcal{U}(C_i, \{x_t\}) \rightarrow C_i$;
5- $\quad$ **if** *(y_t is available)* **then**
6- $\qquad \mathcal{U}(C_i, \{(x_t, y_t)\}) \rightarrow C_i$;
$\quad$ **end**
7- $\quad$ **if** *($\mathcal{D}(C_i) = 1$)* **then**
8- $\qquad i = i + 1$;
9- $\qquad \Upsilon(C_{i-1}) \rightarrow (C_k, C_l)$;
10- $\qquad C_i = C_l$;
11- $\qquad C_{i-1} = C_k$;
12- $\qquad Z_{\text{rec}} = \bigcup_{\substack{\mathcal{E}(C_i, C_j) = 1 \\ 0 \leq j < i}} Z_j$;
$\quad$ **end**
13- $\quad$ **if** *(y_t is not available)* **then**
14- $\qquad \widehat{y}_t = K(Z_i \cup Z_{\text{rec}}, x_t)$.
$\quad$ **end**
end

During operations, each input sample is analyzed to

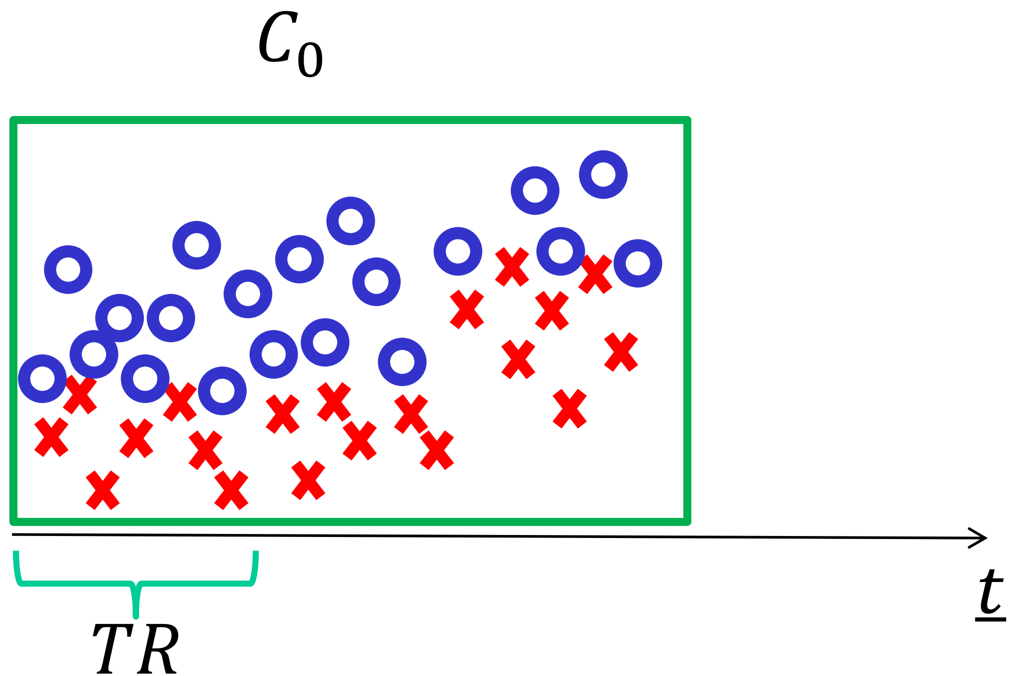
- Extract features that are appended to F_i
- Append supervised information in Z_i

thus updating the current concept representation

JIT CLASSIFIERS: CONCEPT UPDATE

The **concept representation** C_0 is **always updated** during operation,

- Including supervised samples in Z_0 (to describe $p(y|\boldsymbol{x})$)
- Computing feature F_0 (to describe $p(\boldsymbol{x})$)



JIT CLASSIFIERS: THE ALGORITHM

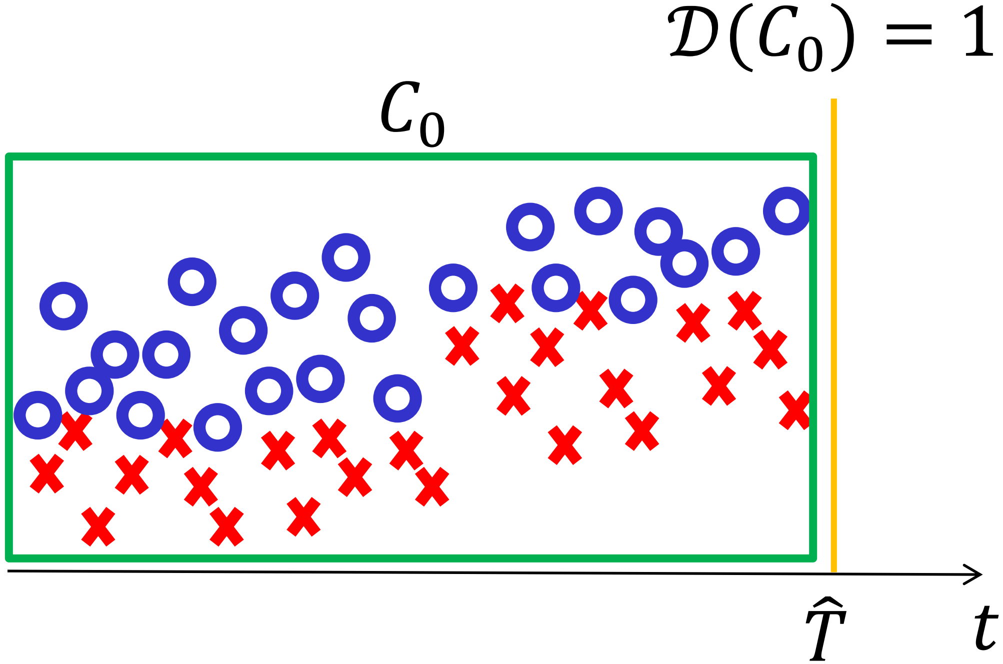
1- Build concept $C_0 = \big(Z_0, F_0, D_0\big)$ from the training sequence;
2- $Z_{\text{rec}} = \emptyset$ and $i = 0$;
3- **while** *(x_t is available)* **do**
4- $\quad \mathcal{U}(C_i, \{x_t\}) \rightarrow C_i$;
5- $\quad$ **if** *(y_t is available)* **then**
6- $\quad\quad \mathcal{U}(C_i, \{(x_t, y_t)\}) \rightarrow C_i$;
$\quad$ **end**
7- $\quad$ **if** *($\mathcal{D}(C_i) = 1$)* **then**
8- $\quad\quad i = i + 1$;
9- $\quad\quad \Upsilon(C_{i-1}) \rightarrow (C_k, C_l)$;
10- $\quad\quad C_i = C_l$;
11- $\quad\quad C_{i-1} = C_k$;
12- $\quad\quad Z_{\text{rec}} = \bigcup_{\substack{\mathcal{E}(C_i, C_j)=1 \\ 0 \leq j < i}} Z_j$;
$\quad$ **end**
13- $\quad$ **if** *(y_t is not available)* **then**
14- $\quad\quad \widehat{y}_t = K(Z_i \cup Z_{\text{rec}}, x_t)$.
$\quad$ **end**
end

The current concept representation is analyzed by $\mathcal{D}$ to determine whether concept drift has occurred

JIT CLASSIFIER: DRIFT DETECTION

Determine when **features in $\boldsymbol{D}$** are no more stationary

- $\mathcal{D}$ monitoring the datastream by means of **online** and **sequential** change-detection tests (CDTs)
- Depending on features, both changes in $p(y|\boldsymbol{x})$ and $p(\boldsymbol{x})$ can be detected
- $\hat{T}$ is the detection time



AN EXAMPLE OF DETECTION OPERATOR

$$\mathcal{D}(C_i) \in \{0,1\}$$

- Implements **online** change-detection tests (**CDTs**) based on the **Intersection of Confidence Intervals** (ICI) rule
- The ICI-rule is an adaptation technique used to define adaptive supports for polynomial regression
- **The ICI-rule determines** when feature sequence (D_i) cannot be fit by a zero-order polynomial, thus **when $\boldsymbol{D_i}$ is non stationary**
- ICI-rule requires **Gaussian**-distributed **features** but **no assumptions on the post-change distribution**

A. Goldenshluger and A. Nemirovski, *"On spatial adaptive estimation of nonparametric regression"* Math. Meth. Statistics, vol. 6, pp. 135–170,1997.

V. Katkovnik, *"A new method for varying adaptive bandwidth selection"* IEEE Trans. on Signal Proc, vol. 47, pp. 2567–2571, 1999.

JIT CLASSIFIERS: THE ALGORITHM

1- Build concept $C_0 = \big(Z_0, F_0, D_0\big)$ from the training sequence;
2- $Z_{\text{rec}} = \emptyset$ and $i = 0$;
3- **while** *(x_t is available)* **do**
4- $\quad \mathcal{U}(C_i, \{x_t\}) \rightarrow C_i$;
5- $\quad$ **if** *(y_t is available)* **then**
6- $\quad\quad \mathcal{U}(C_i, \{(x_t, y_t)\}) \rightarrow C_i$;
$\quad$ **end**
7- $\quad$ **if** *($\mathcal{D}(C_i) = 1$)* **then**
8- $\quad\quad i = i + 1$;
9- $\quad\quad \Upsilon(C_{i-1}) \rightarrow (C_k, C_l)$;
10- $\quad\quad C_i = C_l$;
11- $\quad\quad C_{i-1} = C_k$;
12- $\quad\quad Z_{\text{rec}} = \bigcup_{\substack{\mathcal{E}(C_i, C_j)=1 \\ 0 \leq j < i}} Z_j$;
$\quad$ **end**
13- $\quad$ **if** *(y_t is not available)* **then**
14- $\quad\quad \widehat{y}_t = K(Z_i \cup Z_{\text{rec}}, x_t)$.
$\quad$ **end**
end

If concept drift is detected, the concept representation is split, to isolate the recent data that refer to the new state of $\mathcal{X}$

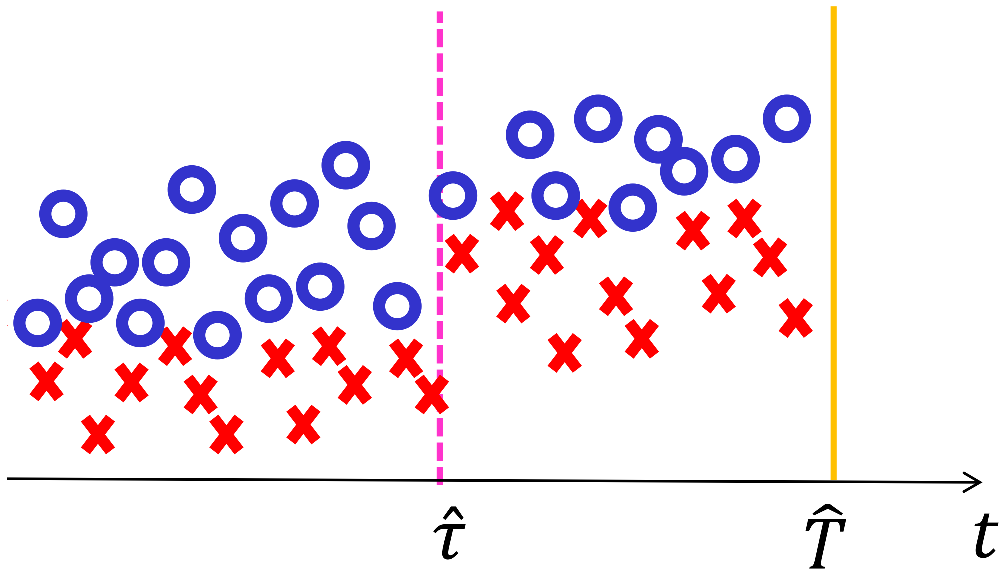
A new concept description is built

JIT CLASSIFIERS: CONCEPT SPLIT

Goal: estimating the change point τ (detections are always delayed). Samples in between $\hat{\tau}$ and $\hat{T}$

Uses statistical tools for performing an **offline** and **retrospective** analysis over the recent data, like:

- as hypothesis tests (HT)
- change-point methods (CPM) can



EXAMPLES OF SPLIT OPERATORS

$$\Upsilon(C_0) = (C_0, C_1)$$

- It performs an **offline analysis** on F_i (just the feature detecting the change) to estimate **when concept drift has actually happened**
- Detections $\hat{T}$ are delayed w.r.t. the actual change point τ
- **Change-Point Methods** implement the following Hypothesis test on the feature sequence:

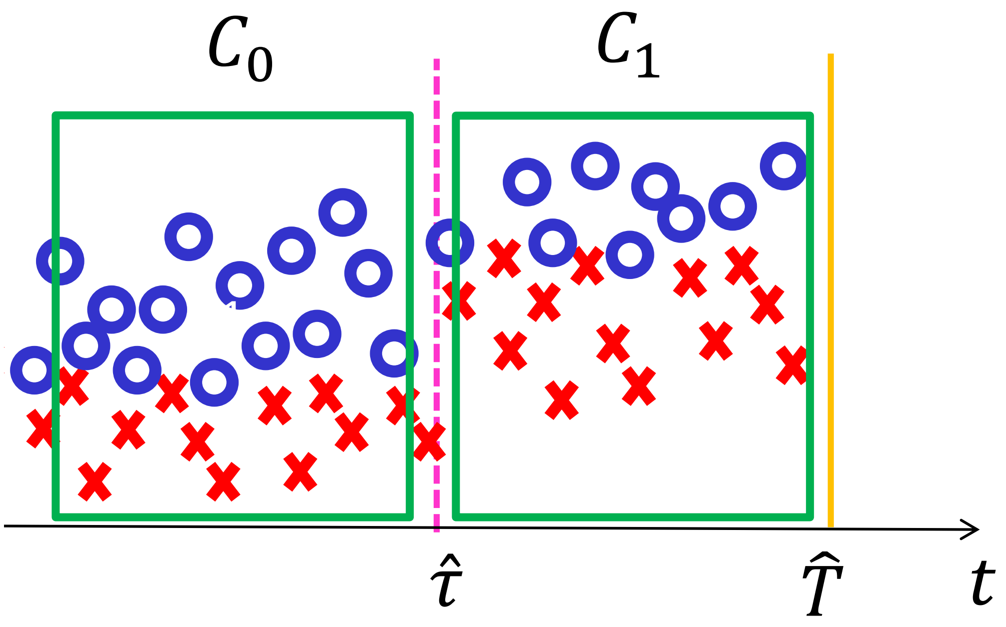
$$\begin{cases} H_0: \text{"}F_i \text{ contains i.i.d. samples"} \\ H_1: \text{"}F_i \text{ contains a change point"} \end{cases}$$

  testing all the possible partitions of F_i and determining the most likely to contain a change point

- ICI-based CDTs implement a refinement procedure to estimate τ after having detected a change at $\hat{T}$.

JIT CLASSIFIERS: CONCEPT SPLIT

Given $\hat{\tau}$, two different concept representations are built



JIT CLASSIFIERS: THE ALGORITHM

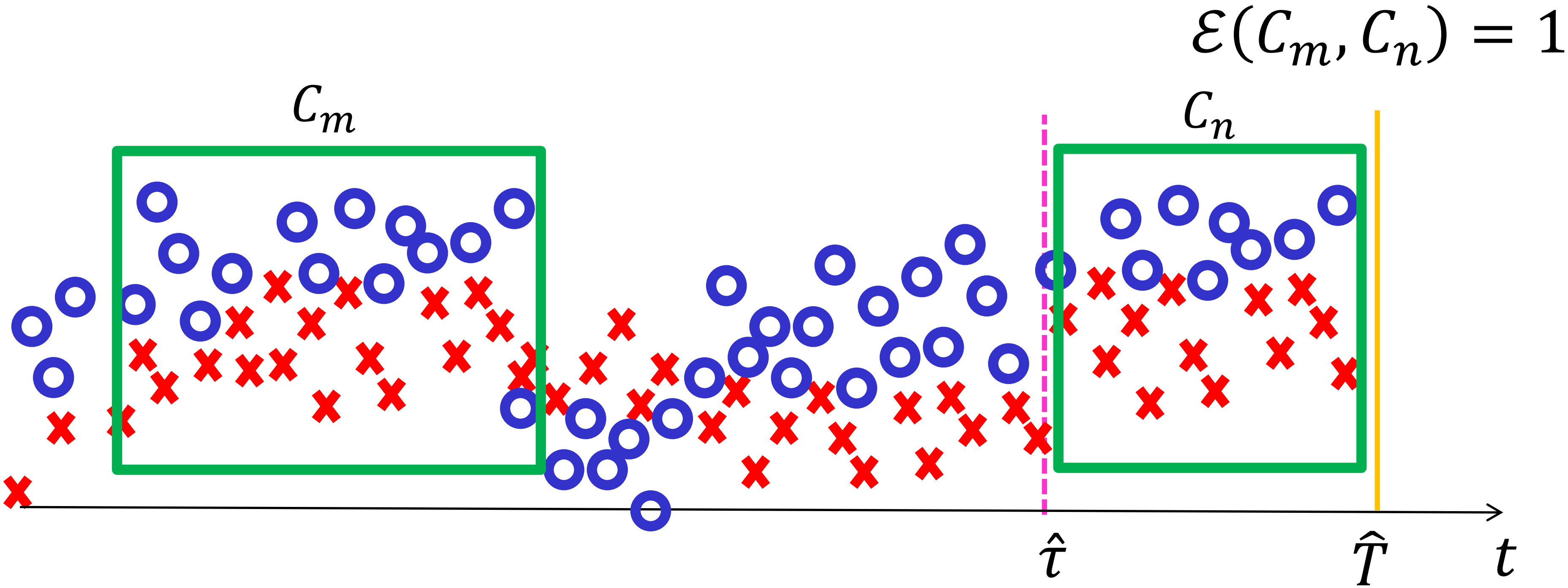
1- Build concept $C_0 = \left(Z_0, F_0, D_0\right)$ from the training sequence;
2- $Z_{\text{rec}} = \emptyset$ and $i = 0$;
3- **while** *(x_t is available)* **do**
4- $\quad \mathcal{U}(C_i, \{x_t\}) \to C_i$;
5- $\quad$ **if** *(y_t is available)* **then**
6- $\quad\quad \mathcal{U}(C_i, \{(x_t, y_t)\}) \to C_i$;
$\quad$ **end**
7- $\quad$ **if** *($\mathcal{D}(C_i) = 1$)* **then**
8- $\quad\quad i = i + 1$;
9- $\quad\quad \Upsilon(C_{i-1}) \to (C_k, C_l)$;
10- $\quad\quad C_i = C_l$;
11- $\quad\quad C_{i-1} = C_k$;
12- $\quad\quad Z_{\text{rec}} = \bigcup_{\substack{\mathcal{E}(C_i, C_j) = 1 \\ 0 \le j < i}} Z_j$;
$\quad$ **end**
13- $\quad$ **if** *(y_t is not available)* **then**
14- $\quad\quad \widehat{y}_t = K(Z_i \cup Z_{\text{rec}}, x_t)$.
$\quad$ **end**
end

Look for concepts that are equivalent to the current one.

Gather supervised samples from all the representations C_j that refers to the same concept

Concept equivalence is assessed by

- comparing features F to determine whether $p(\boldsymbol{x})$ is the same on C_m and C_n (using a test of equivalence)
- comparing classifiers trained on C_m and C_n to determine whether $p(y|\boldsymbol{x})$ is the same



JIT CLASSIFIERS: THE ALGORITHM

1- Build concept $C_0 = \big(Z_0, F_0, D_0\big)$ from the training sequence;
2- $Z_{\text{rec}} = \emptyset$ and $i = 0$;
3- **while** *(x_t is available)* **do**
4- $\quad \mathcal{U}(C_i, \{x_t\}) \to C_i$;
5- $\quad$ **if** *(y_t is available)* **then**
6- $\quad\quad \mathcal{U}(C_i, \{(x_t, y_t)\}) \to C_i$;
$\quad$ **end**
7- $\quad$ **if** *($\mathcal{D}(C_i) = 1$)* **then**
8- $\quad\quad i = i + 1$;
9- $\quad\quad \Upsilon(C_{i-1}) \to (C_k, C_l)$;
10- $\quad\quad C_i = C_l$;
11- $\quad\quad C_{i-1} = C_k$;
12- $\quad\quad Z_{\text{rec}} = \bigcup_{\substack{\mathcal{E}(C_i, C_j)=1 \\ 0 \leq j < i}} Z_j$;
$\quad$ **end**
13- $\quad$ **if** *(y_t is not available)* **then**
14- $\quad\quad \boxed{\widehat{y}_t = K(Z_i \cup Z_{\text{rec}}, x_t).}$
$\quad$ **end**
end

The classifier K is reconfigured using all the available supervised couples

COMPARING WINDOWS

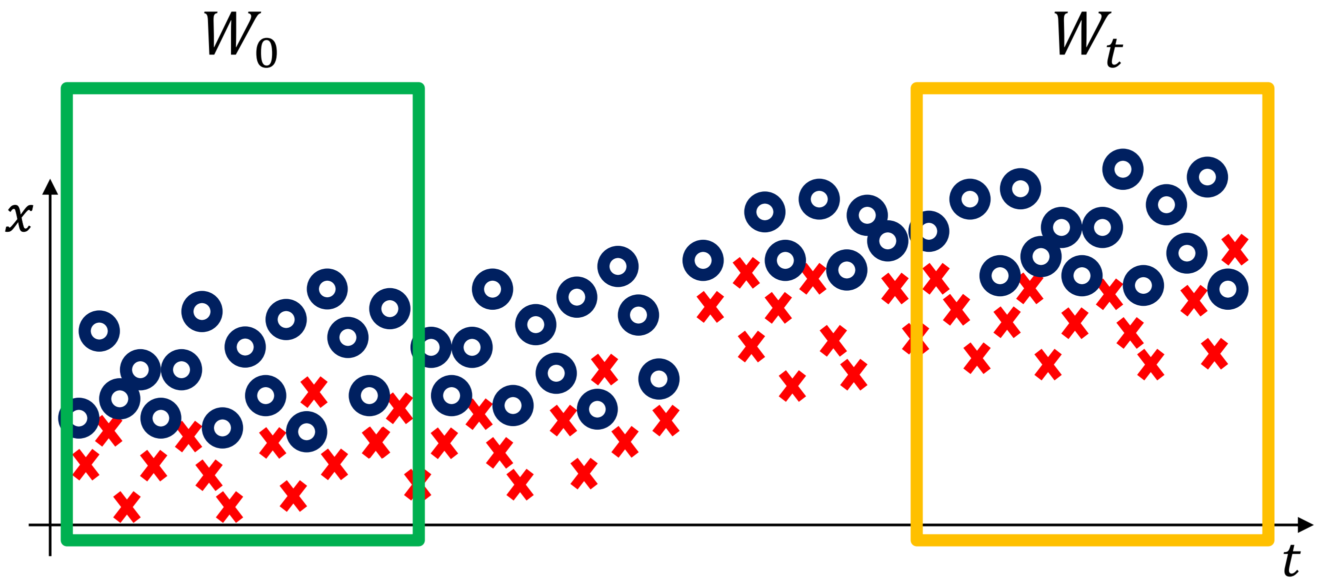
THE MOTIVATING IDEA

Detect CD at time t by comparing two different windows.

In practice, one computes:

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

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



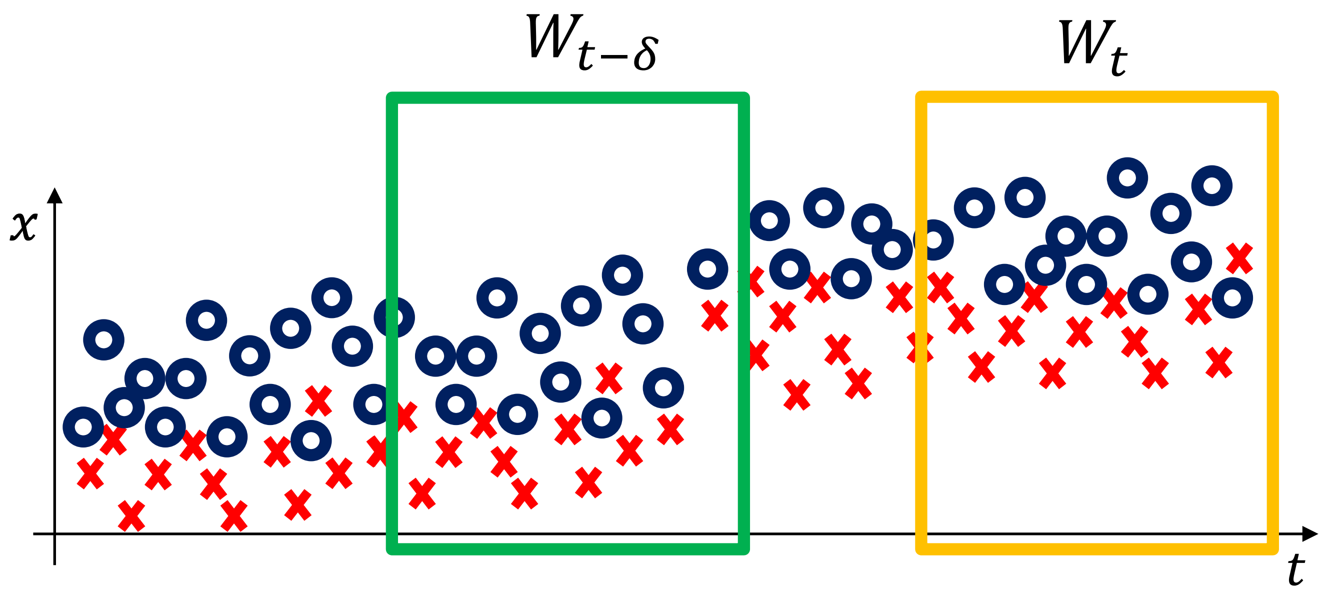
THE MOTIVATING IDEA

Detect CD at time t by comparing two different windows.

In practice, one computes:

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

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



THE MOTIVATING IDEA

Pro:

- there are a lot of test statistics to compare data windows

Cons:

- The biggest drawback of comparing windows is that subtle CD might not be detected (this is instead the main advantage of sequential techniques)
- More computational demanding than sequential technique
- Window size definition is an issue

WINDOW COMPARISON: MAIN APPROACHES

- The averages over two adjacent windows (ADWIN)

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

WINDOW COMPARISON: MAIN APPROACHES

- The averages over two adjacent windows (ADWIN)
- Comparing the classification error over W_t and W_0

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

WINDOW COMPARISON: MAIN APPROACHES

- The averages over two adjacent windows (ADWIN)
- Comparing the classification error over W_t and W_0
- Compute empirical distributions of raw data over W_0 and W_t and compare
  - The Kullback-Leibler divergence
  - the Hellinger distance

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

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: MAIN APPROACHES

- The averages over two adjacent windows (ADWIN)
- Comparing the classification error over W_t and W_0
- Compute empirical distributions of raw data over W_0 and W_t and compare
  - The Kullback-Leibler divergence
  - the Hellinger distance
  - Compute the density ratio over the two windows using kernel methods (to overcome curse of dimensionality problems when computing empirical distributions)

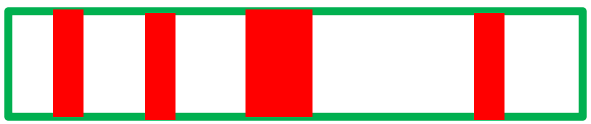
Kawahara, Y. and Sugiyama, M. *"Sequential change-point detection based on direct density-ratio estimation"*. Statistical Analysis and Data Mining, 5(2):114–127, 2012.

WINDOW COMPARISON: TESTING EXCHANGABILITY

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

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

WINDOW COMPARISON: PAIRED LEARNERS

Two classifiers are trained

- a **stable online learner** (S) that predicts based on all the supervised samples
- a **reactive** one (R_w) trained over a short sliding window

During operation

- labels are provided by S
- predictions of R_w are computed but not provided
- as soon as $\boldsymbol{R_w}$ **is more frequently correct** than S, **detect CD**

Adaptation consists in **replacing** S by R_w

Bach, S.H.; Maloof, M., *"Paired Learners for Concept Drift"* in Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on pp.23-32, 15-19 Dec. 2008

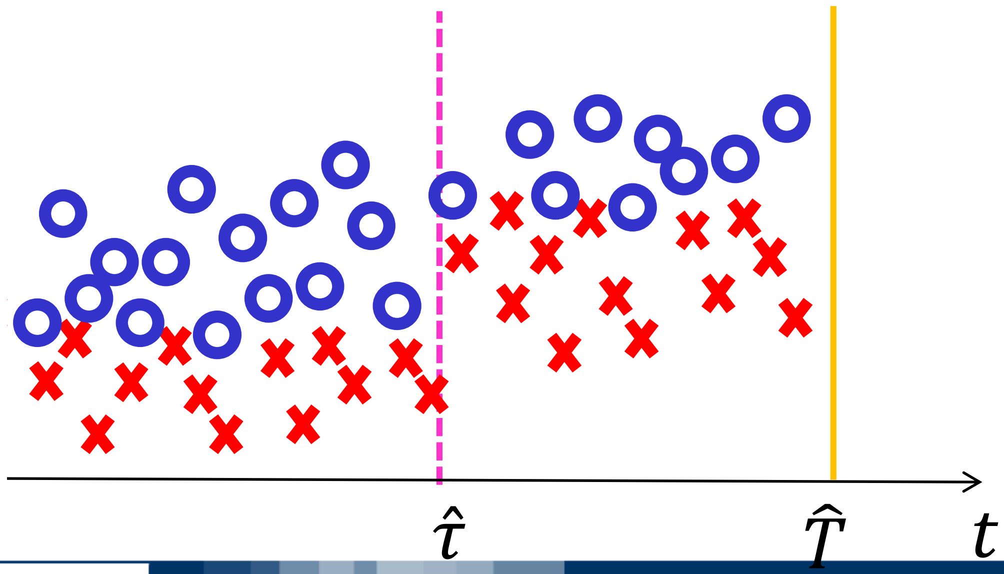
REMARKS ON ACTIVE APPROACHES

COMMENTS FROM MY PERSONAL EXPERIENCE

- Typically, when monitoring the classification error, false positives hurt less than detection delay
  - Things might change on class unbalance

COMMENTS FROM MY PERSONAL EXPERIENCE

- Typically, when monitoring the classification error, false positives hurt less than detection delay
  - Things might change on class unbalance
- Providing i.i.d. samples for reconfiguration seems more critical. When estimating the change-time:



COMMENTS FROM MY PERSONAL EXPERIENCE

- Typically, when monitoring the classification error, false positives hurt less than detection delay
  - Things might change on class unbalance
- Providing i.i.d. samples for reconfiguration seems more critical. When estimating the change-time:
  - Overestimates of τ provide too few samples
  - Underestimates of τ provide non i.i.d. data
  - Worth using accurate SPC methods like change-point methods (CPMs)

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.

COMMENTS FROM MY PERSONAL EXPERIENCE

- Typically, when monitoring the classification error, false positives hurt less than detection delay
  - Things might change on class unbalance
- Providing i.i.d. samples for reconfiguration seems more critical. When estimating the change-time:
  - Overestimates of τ provide too few samples
  - Underestimates of τ provide non i.i.d. data
  - Worth using accurate SPC methods like change-point methods (CPMs)
- Exploitation of recurrent concept is important
  - Providing additional samples could make the difference
  - Mitigate the impact of false positives

THE PASSIVE APPROACH

Classifiers undergoing continuous adaptation

PASSIVE APPROACH

Passive approaches:

- **Do not have an explicit** CD **detection** mechanism
- They are **aware** that $\phi_t(\boldsymbol{x}, y)$ *might* change at any time and at any rate
- **Perform continuous adaptation** of their model(s) parameters at each new arrival

They can be divided in:

- **Single model** methods
- **Ensemble** methods

SINGLE CLASSIFIER MODEL

- Lower computational cost than ensemble methods
- Mainly related to specific classifiers
  - CVFDT: Concept-adapting Very Fast Decision Tree learner, and online decision tree algorithm that incrementally learns from a sliding window

P. Domingos and G. Hulton, *“Mining high-speed data streams”* in Proc. of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 71–80, 2000.

G. Hulten, L. Spencer, and P. Domingos, *“Mining time-changing data streams”* in Proc. of Conference on Knowledge Discovery in Data, pp. 97–106, 2001.

SINGLE CLASSIFIER MODEL

- Lower computational cost than ensemble methods
- Mainly related to specific classifiers
  - CVFDT: Concept-adapting Very Fast Decision Tree learner, and online decision tree algorithm that incrementally learns from a sliding window
  - OLIN: fuzzy-logic based approach that exploits a sliding window

L. Cohen, G. Avrahami-Bakish, M. Last, A. Kandel, and O. Kipersztok, *"Real-time data mining of non-stationary data streams from sensor networks"*, Information Fusion, vol. 9, no. 3, pp. 344–353, 2008.

SINGLE CLASSIFIER MODEL

- Lower computational cost than ensemble methods
- Mainly related to specific classifiers
  - CVFDT: Concept-adapting Very Fast Decision Tree learner, and online decision tree algorithm that incrementally learns from a sliding window
  - OLIN: fuzzy-logic based approach that exploits a sliding window
  - An Extreme Learning Machine has been also combined with a time-varying NN

Y. Ye, S. Squartini, and F. Piazza, *"Online sequential extreme learning machine in nonstationary environments"*, Neurocomputing, vol. 116, no. 20, pp. 94–101, 2013

ENSEMBLE METHODS

ENSEMBLE METHODS

An **ensemble** of **multiple models** is preserved in memory

$$\mathcal{H} = \{h_0, \ldots, h_N\}$$

Each **individual** $h_i, i = 1, \ldots, N$ is typically trained from a different training set and could be from a different model

Final prediction of the ensemble is given by (weighted) **aggregation of the individual predictions**

$$\mathcal{H}(\boldsymbol{x_t}) = \underset{\boldsymbol{\omega \in \Lambda}}{\operatorname{argmax}} \sum_{\boldsymbol{h_i \in \mathcal{H}}} \alpha_i \, [h_i(\boldsymbol{x}_t) = \omega]$$

Typically, one assumes data arrives in **batches** and each classifier is trained over a batch

ENSEMBLE METHODS AND CONCEPT DRIFT

- **Each individual** implicitly refers to a component of a mixture distribution characterizing a **concept**
- In practice, often ensemble methods assume data (supervised and unsupervised) are provided in batches
- **Adaptation** can be achieved by:
  - **updating each individual:** either in batch or online manner
  - **dynamic aggregation**: adaptively defining weights ω_i
  - **structural update:** including/removing new individuals in the ensemble, possibly recovering past ones that are useful in case of recurrent concepts

Kuncheva, L. I. *"Classifier ensembles for changing environments"* In Proc. of the 5th Int. Workshop on Multiple Classifier Systems. MCS. 1–15 2004.

ENSEMBLE METHODS AND CONCEPT DRIFT

Ensemble based approaches provide a **natural fit** to the problem **of learning in nonstationary settings**,

- Ensembles tend to be more accurate than single classifier-based systems due to **reduction in the variance of the error**
- **Stability:** flexible to easily incorporate new data into a classification model, simply by **adding new individuals** to the ensemble (or updating individuals)
- **Plasticity:** provide a natural mechanism **to forget irrelevant knowledge**, simply by **removing** the corresponding old **individual(s)** from the ensemble
- They can operate in continuously drifting environments

SEA

A **fixed-size ensemble** that performs

- batch learning
- structural update to adapt to concept drift

When a new batch $S = \{(\boldsymbol{x}_0^t, y_0^t), (\boldsymbol{x}_1^t, y_1^t), \dots, (\boldsymbol{x}_B^t, y_B^t)\}$ arrives

- train h_t on S
- test h_{t-1} on S
- If the ensemble is not full ($\#\mathcal{H} < N$), **add** h_{t-1} to $\mathcal{H}$
- Otherwise, **remove** $h_i \in \mathcal{H}$ that is **less accurate** on S (as far as this is worst than h_{t-1})

W. N. Street and Y. Kim, *"A streaming ensemble algorithm (SEA) for large scale classification"*, in Proceedings to the 7th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 377–382, 2001

DWM ALGORITHM

Dynamic weighted majority (DWM) algorithm is an ensemble method where:

- Individuals are trained on different batches of data
- Each **individual is associated to a weight**
- **Weights are decreased** to individuals that are **not accurate** on the samples of the current batch
- Individuals having **low weights are dropped**
- Individuals are **created** at each error of the ensemble
- Predictions are made by **weighted majority voting**

Kolter, J. and Maloof, M. ***"Dynamic weighted majority: An ensemble method for drifting concepts".*** **Journal of Machine Learning Research 8, 2755–2790. 2007**

LEARNS++ .NSE

Batch-learning algorithm performing predictions based on a **weighted majority voting** scheme:

- Both **individuals** and **training samples** are **weighted**
- **Misclassified instances** receive **large weights**: samples from the new concept are often misclassified thus they receive large weights.
- **Weights of the individuals** depends on the **time-adjusted errors** on current and past batches: old individuals can be recovered in case of recurrent concepts
- Old individuals are not discarded

DDD

- Diversity for Dealing with Drifts (DDD) combines **two ensembles:**
  - An **High diversity** ensemble
  - A **Low diversity** ensemble

  and a concept-drift detection method.
- Online bagging is used to control ensemble diversity
- In **stationary conditions**, predictions are made by **low-diversity ensemble**
- **After concept drift**, the ensembles are updated and predictions are made by the **high-diversity ensemble**.

Minku, L. L.; Yao, X. *"DDD: A New Ensemble Approach For Dealing With Concept Drift"*, IEEE Transactions on Knowledge and Data Engineering, IEEE, v. 24, n. 4, p. 619-633, April 2012,

COMMENTS FROM MY PERSONAL EXPERIENCE

We have combined

- a JIT classifier using recurrent concepts
- a sliding window classifier

As in paired learners,

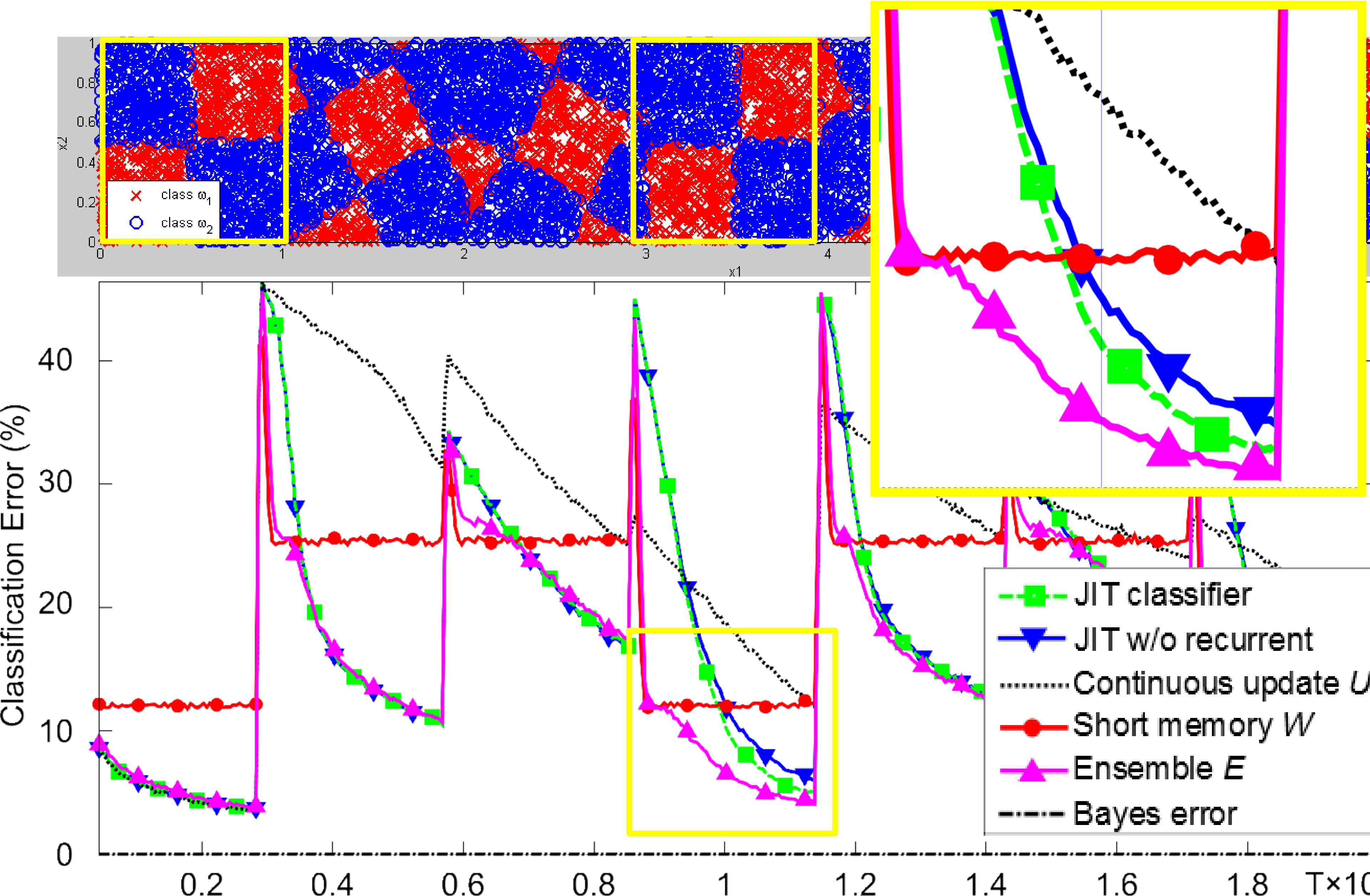
- JIT is meant to provide the best post-detection adaptation and best performance in a stationary state
- The sliding window classifier is meant to provide the quickest reaction to CD

We used a simple aggregation *"Predictions are made by the most accurate classifier over the last 20 samples"*

Actually this ensemble performed very well, combining the advantages of the two classifiers

C. Alippi, G. Boracchi and M. Roveri, "*Just In Time Classifiers for Recurrent Concepts*" IEEE Transactions on Neural Networks and Learning Systems, 2013. vol. 24, no.4, pp. 620 -634

THE ENSEMBLE USING JIT CLASSIFIER



CONCLUDING REMARKS

Recent Trends and Open Issues

RECENT TRENDS

Learning under **concept drift** and **class imbalance**

- Typically resampling are used in ensemble methods to compensate class imbalance (e.g. SMOTE is used in Learn++.CDS, uncorrelated bagging, online bagging, SERA)
- Determine which figure of merit to monitor when classes are imbalanced

G. Ditzler and R. Polikar, *"Incremental learning of concept drift from streaming imbalanced data"* IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 10, pp. 2283–2301, 2013.

S. Wang, L. L. Minku, and X. Yao, *"Resampling-based ensemble methods for online class imbalance learning"* IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 5, pp. 1356–1368, 2015.

RECENT TRENDS

Semi-supervised and unsupervised methods for CD

- Often, **supervised** information is **scarce**
- *Initially labeled environments* and *verification latency*
- Using **unlabeled** data to **improve model accuracy** (not only to CD detection) is of paramount importance

Typically the problem is addressed by

- Approximating the conditional density of each class $p(\boldsymbol{x}|y)$ by a parametric density models
- Drift is assumed to be smoothly evolving

K. Dyer, R. Capo, and R. Polikar, *"COMPOSE: A semi-supervised learning framework for initially labeled non-stationary streaming data,"* IEEE TKDE , vol. 25, no. 1, pp. 12–26, 2013.

G. Krempl, *"The algorithm apt to classify in concurrence of latency and drift"* Advances in Intelligent Data Analysis, 2011.

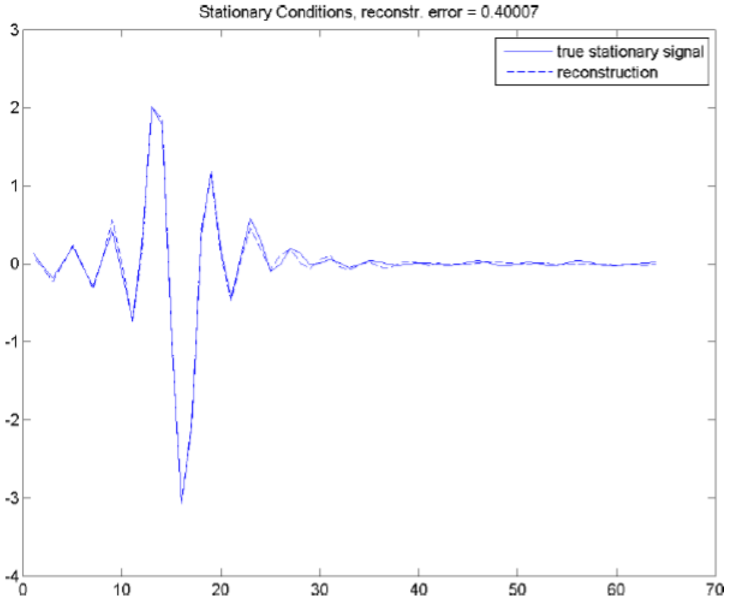
C. Alippi, G. Boracchi, and M. Roveri, *"An effective just-in-time adaptive classifier for gradual concept drifts"* in Proc. of 2011 IJCNN , pp. 1675–1682, IEEE, 2011.

OPEN ISSUES

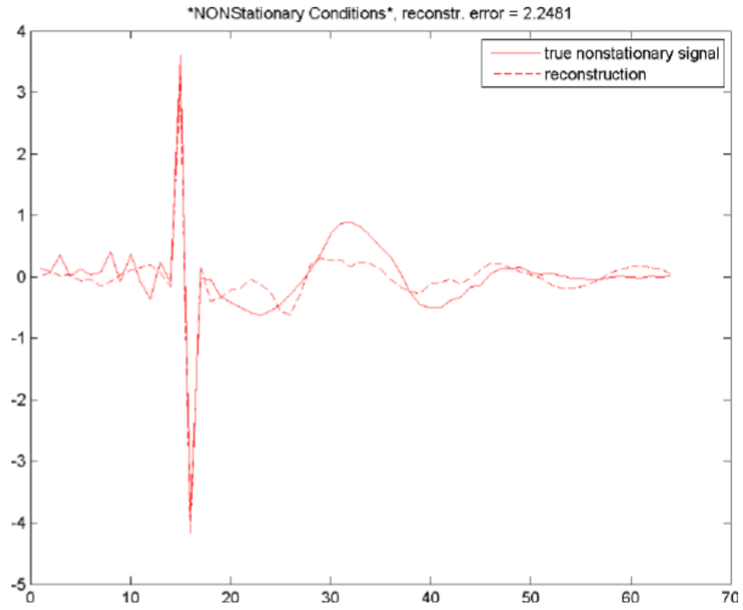
- A suitable **theoretical framework** for learning under CD is missing. This would enable the assessment of performance bounds with respect to specific drift (types, rate, magnitudes)
- Techniques for **handling data** that are not i.i.d. realizations of a random variable but that **feature specific structure under each concept**, like signals and images

Signal acquired from a land-slide monitoring application

Normal Signal



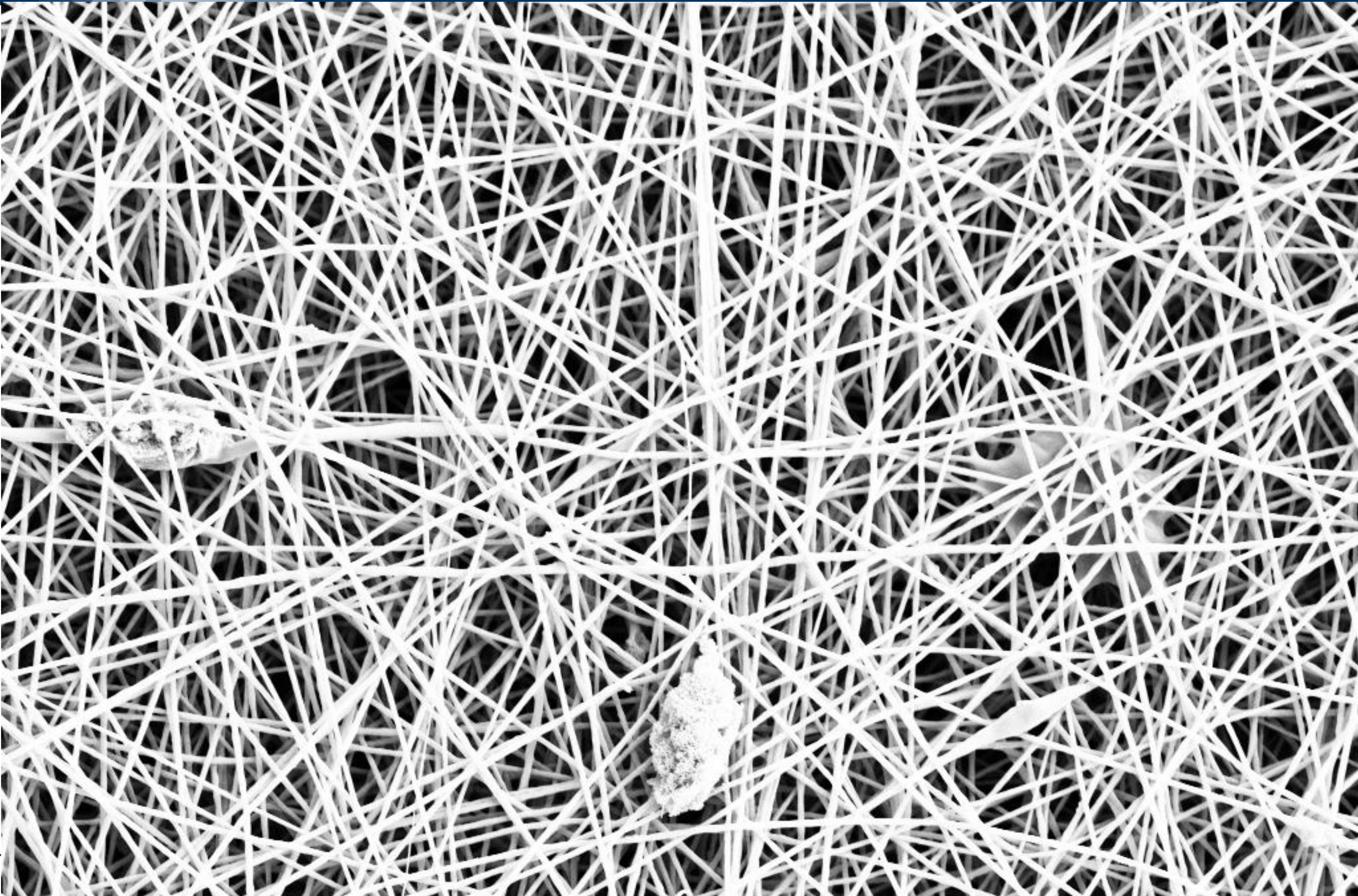
Anomalous Signal



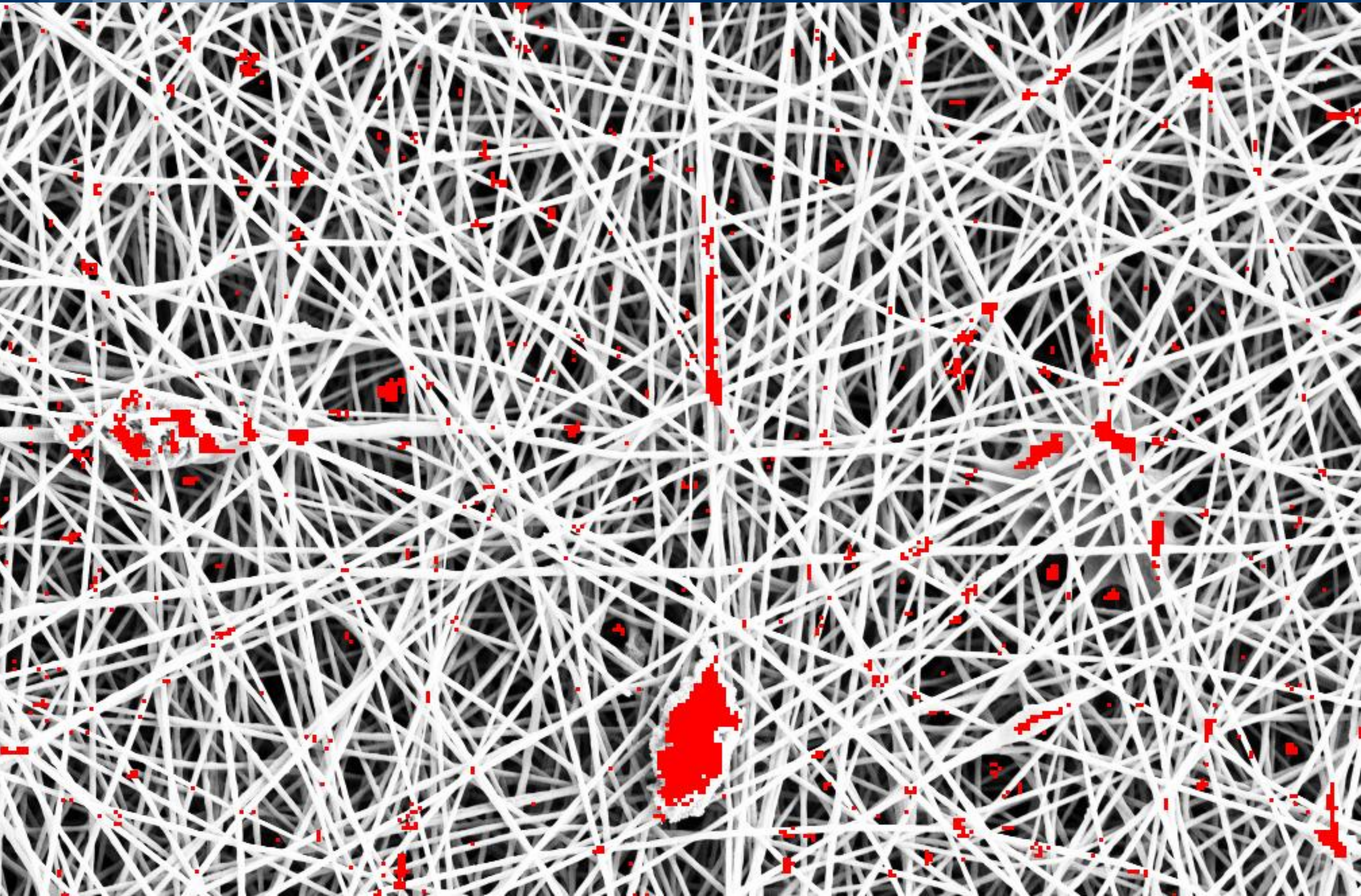
Alippi C., Boracchi G., Roveri M., "*A reprogrammable and intelligent monitoring system for rock-collapse forecasting*" IEEE Systems Journal, Accepted for Publication

Alippi C., Boracchi G., Wohlberg B. "*Change Detection in Streams of Signals with Sparse Representations*" IEEE ICASSP 2014, pp 5252 - 5256

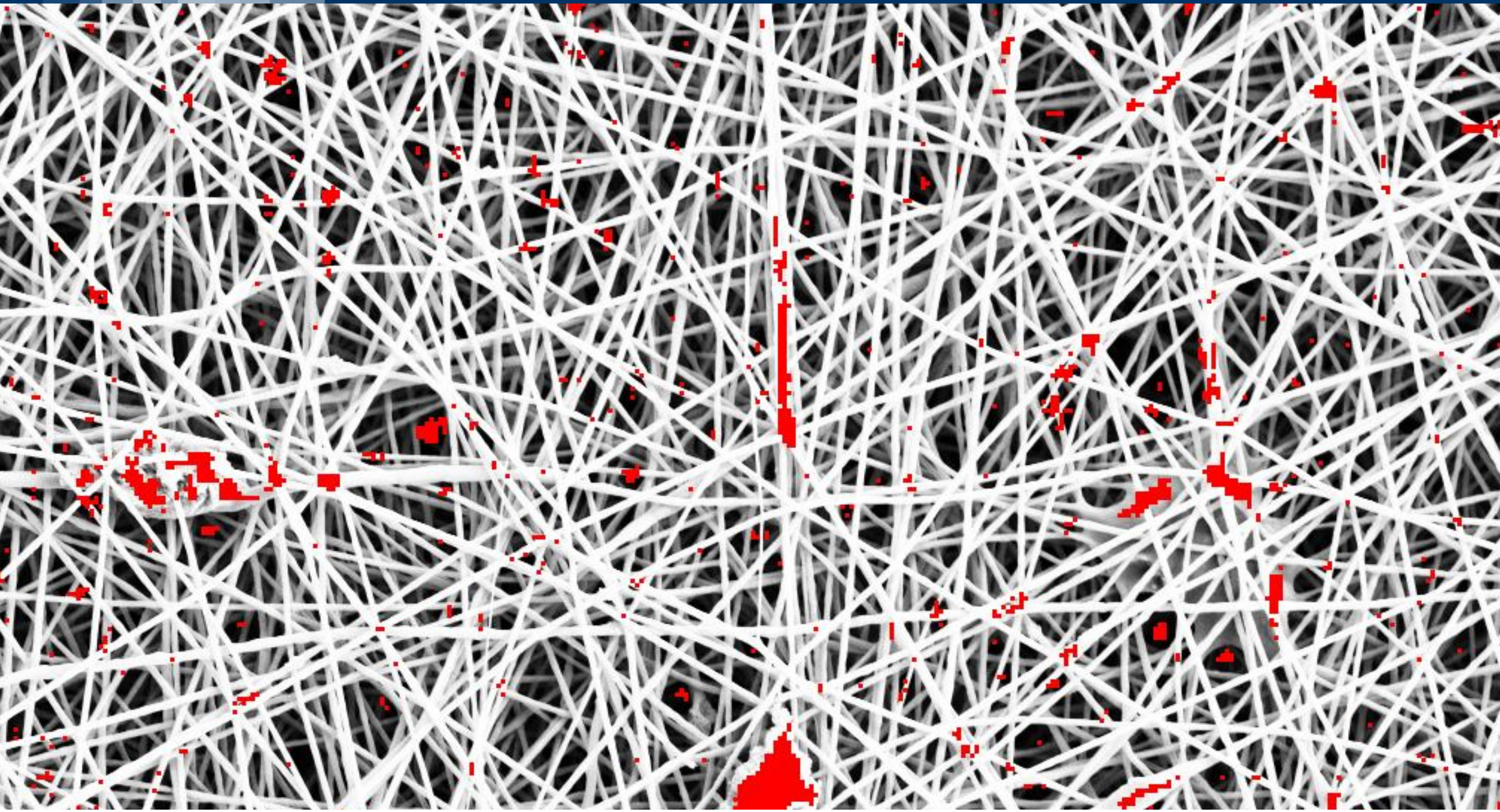
ANOMALY DETECTION IN IMAGES



ANOMALY DETECTION IN IMAGES



ANOMALY DETECTION IN IMAGES



Boracchi G., Carrera D. Wohlberg B. "*Novelty Detection in Images by Sparse Representations*" Proceedings of Intelligent Embedded Systems at IEEE SSCI 2014

Carrera D., Boracchi G., Foi A., Wohlberg B. "*Detecting Anomalous Structures by Convolutional Sparse Models*" Proceedings of IJCNN 2015

OPEN ISSUES

- A suitable **theoretical framework** for learning under CD is missing. This would enable the assessment of performance bounds with respect to specific drift (types, rate, magnitudes)
- Techniques for **handling data** that are not i.i.d. realizations of a random variable but that **feature specific structure under each concept**, like signals and images
  - In this case there is the problem of **learning suitable representations** for **detecting** changes/anomalies in the structure

OPEN ISSUES

- **Integration of expert knowledge and data-driven models for CD handling**
  - Experts are reluctant to rely on outputs of *black-box* models that are difficult to interpret
  - Valuable information from experts could be integrated in CD detection and adaptation
- **Benchmarking**:
  - Statistically significant results (at least for CD detection) often requires synthetically introduced drifts
  - Cross-validation by shuffling data is sometimes not feasible on streaming data
  - A proper validation from historical data is difficult when supervised samples come in the form of feedbacks

A VERY HOT RESEARCH TOPIC

The topic **is quite hot now**, given the **popularity of data-driven models** in real world applications where data-generating processes are **nonstationary**

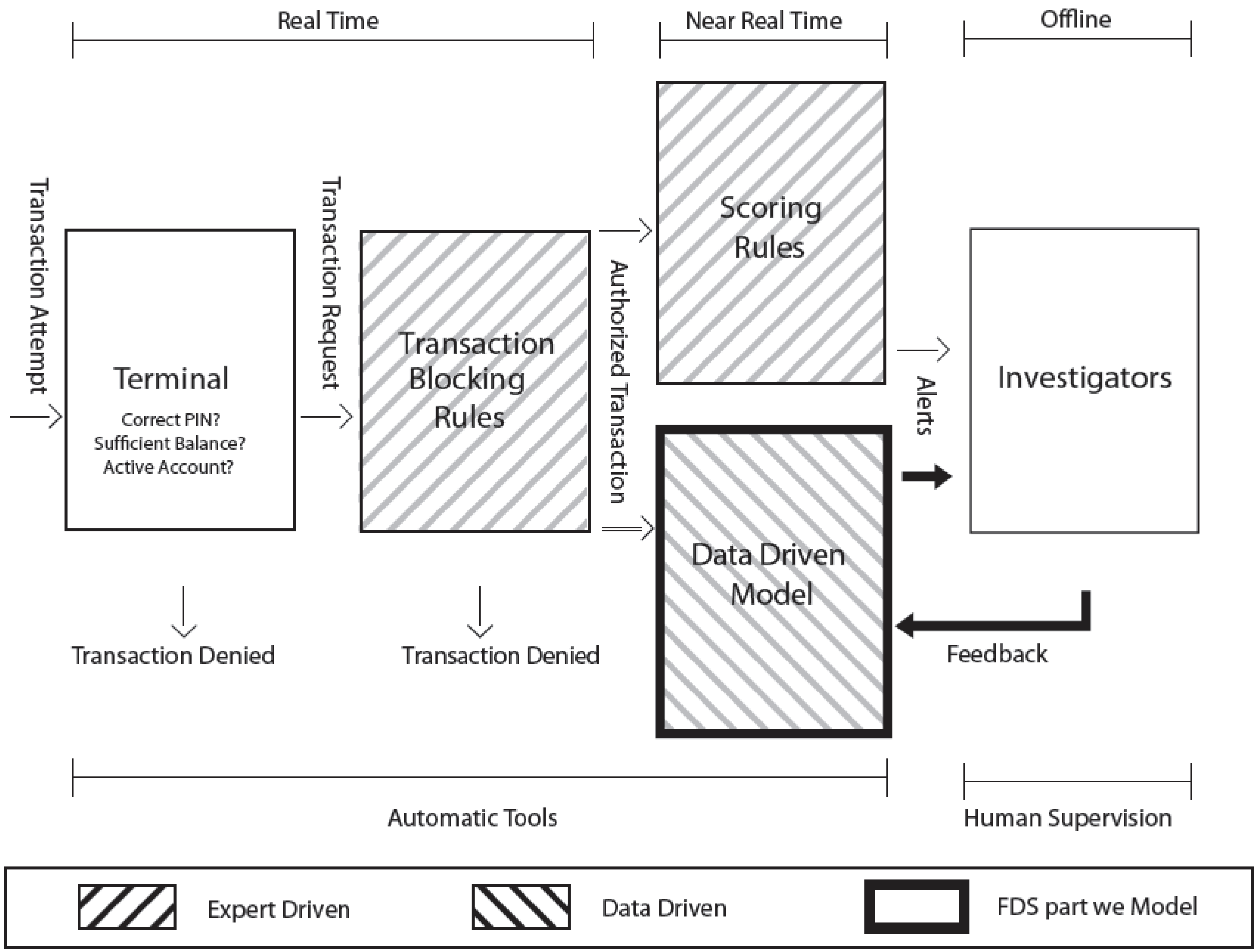
- Special Session at IJCNN 2013, 2014, 2015 … we are organizing together with Robi Polikar, Rowan University
- LEAPS workshop in AIAI 2013
- Special Issue on TNNLS 2013
- Outstanding paper in TNNLS 2016 has been awarded to our paper on JIT recurrent concepts

home.deib.polimi.it/boracchi/index.html

C. Alippi, G. Boracchi and M. Roveri, "*Just In Time Classifiers for Recurrent Concepts*" IEEE Transactions on Neural Networks and Learning Systems, 2013. vol. 24, no.4, pp. 620 -634

QUESTIONS?

THANK YOU VERY MUCH



home.deib.polimi.it/boracchi/index.html