LEARNING IN NONSTATIONARY ENVIRONMENTS: PERSPECTIVES AND APPLICATIONS

Giacomo Boracchi[1] and Gregory Ditzler[2]

[1] Politecnico di Milano
Dipartimento Elettronica e Informazione
Milano, Italy

[2] The University of Arizona
Department of Electrical & Computer Engineering
Tucson, AZ USA

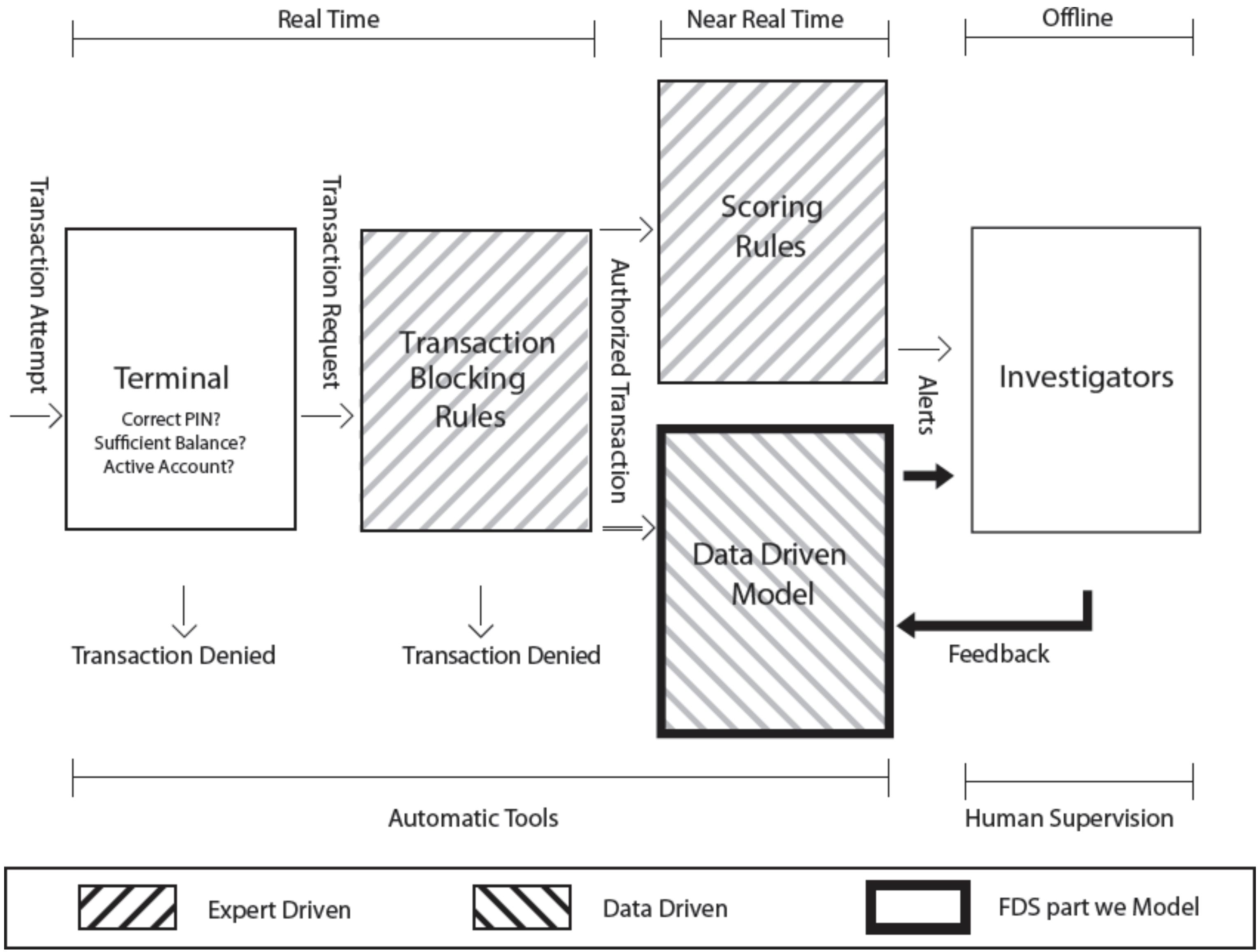
giacomo.boracchi@polimi.it, ditzler@email.arizona.edu

AN INTRODUCTORY EXAMPLE

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

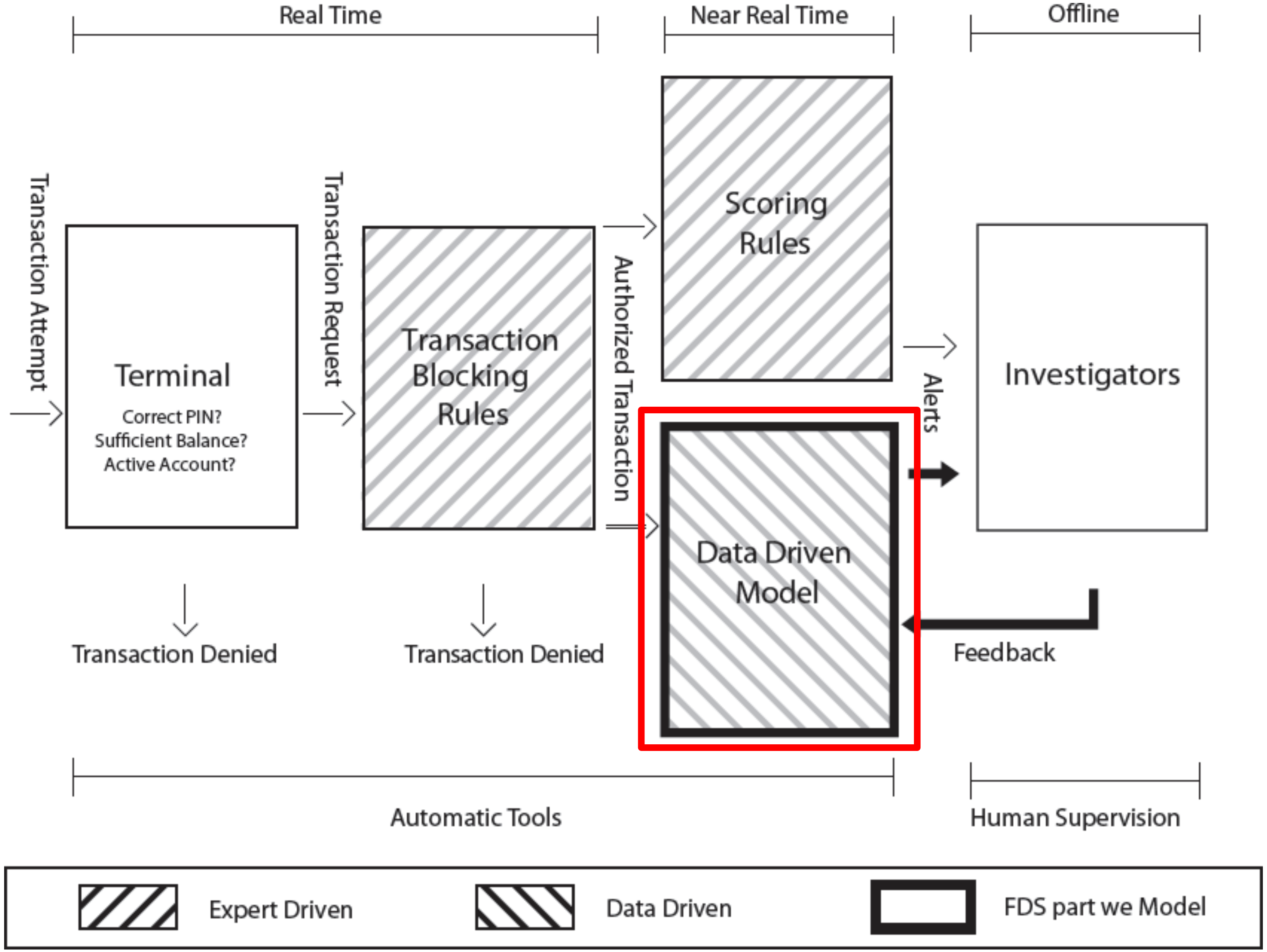
Fraud Detection

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 **detecting frauds**

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

Challenging classification problem because:

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

Concept drift "changes the problem" the classifier has to address

CONCEPT DRIFT IN LEARNING PROBLEMS

Learning problems related to **predicting user preferences / interests,** such as:

- Recommendation systems
- Spam / email filtering

..when the user change his/her own preferences, the classification problem changes

Spam Classification

CONCEPT DRIFT IN LEARNING PROBLEMS

Prediction problems like :

- Financial markets analysis
- Environmental monitoring
- Critical infrastructure monitoring / management

where data are often in a form of **time-series** and the **data-generating process** typically **evolves** over time.

Financial Markets

IN PRACTICE…

In practice **Concept Drift (CD)** is a problem in all **application scenarios** where:

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

Since in these cases, the data-driven model should **autonomously adapt** to preserve its performance over time

THIS TUTORIAL

This tutorial focuses on:

- **methodologies and general approaches** for **adapting data-driven models** to Concept Drift (i.e. in Nonstationary Environments)
- **learning aspects**, while related issues like change/outlier/anomaly detection are not discussed in detail
- **classification** as an example of **supervised learning problem**. Regression problems are not considered here even though similar issues applies
- the **most important frameworks** that can be implemented using **any classifier,** rather than solutions for specific classifiers
- illustrations typically refer to scalar and numerical data, even though methodologies often apply to **multivariate and numerical or 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

We **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

TUTORIAL OUTLINE

- Problem Statement
  - Drift taxonomy
  - The issue
- Active Approaches
  - CD detection monitoring classification error
  - CD detection monitoring raw data
  - JIT classifiers
  - Window comparison methods
- Passive Approaches
  - Single model methods
  - Ensemble methods
  - Initially labelled environments
- Datasets and Codes
- Concluding Remarks and Research Perspectives

PROBLEM STATEMENT

Learning in Nonstationary (Streaming) Environments

CLASSIFICATION OVER DATASTREAMS

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

$$X = \{\boldsymbol{x}_{\boldsymbol{0}}, \boldsymbol{x}_{\boldsymbol{1}}, \dots, \}$$

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

Typical assumptions:

- Independent and identically distributed (**i.i.d.**) **inputs**
$$(\boldsymbol{x}_t, y_t) \sim \phi(\boldsymbol{x}, y)$$
- An **initial training set** $TR = \{(\boldsymbol{x}_0, y_0), \dots, (\boldsymbol{x}_n, y_n)\}$ 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.

The data generating process is then modeled as:

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

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 supervised samples might arrive:

- In single instances
- Batch-wise

Fresh, new supervised samples are necessary to:

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

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

DRIFT TAXONOMY

Different Types of Concept Drift

DRIFT TAXONOMY

Drift taxonomy according to two characteristics:

1. **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

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

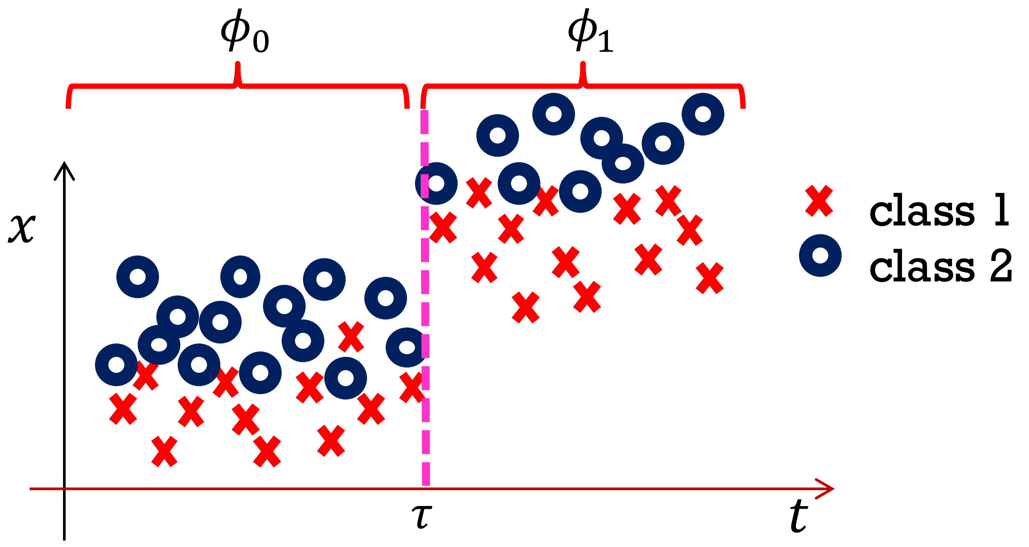
DRIFT TAXONOMY: WHAT IS CHANGING?

Real Drift

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

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

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



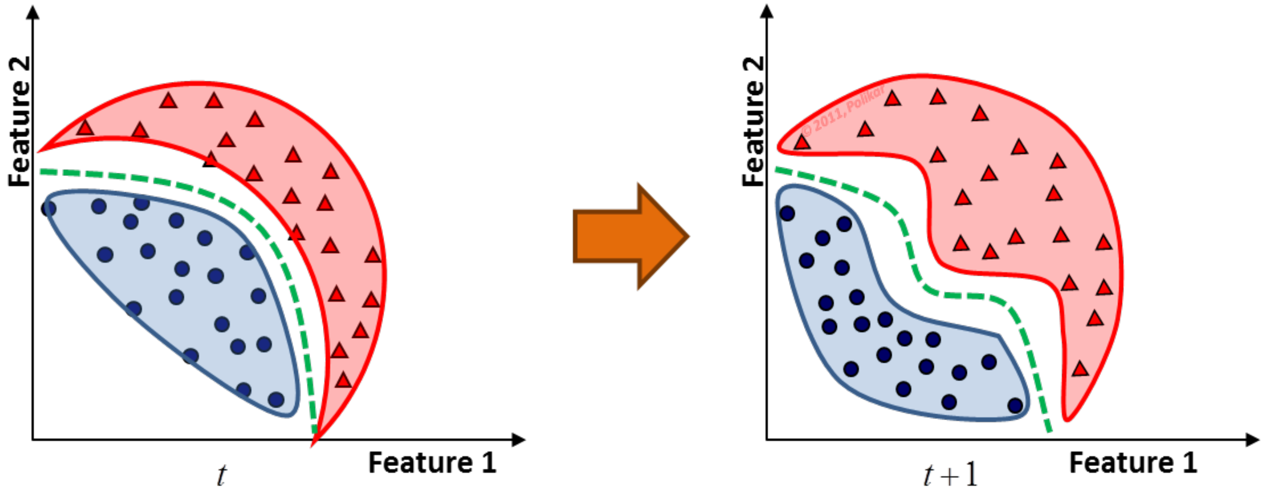
DRIFT TAXONOMY: WHAT IS CHANGING?

Real Drift

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

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

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



DRIFT TAXONOMY: WHAT IS CHANGING?

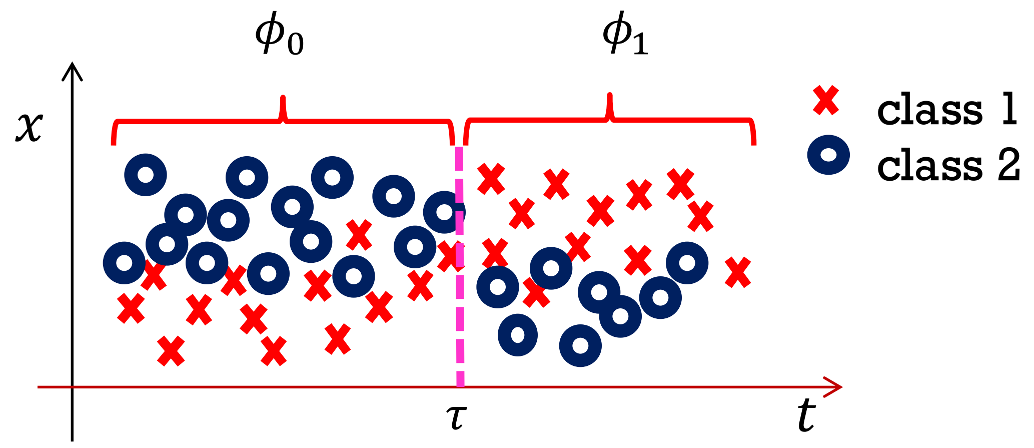
Real Drift

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

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

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

E.g. classes swap



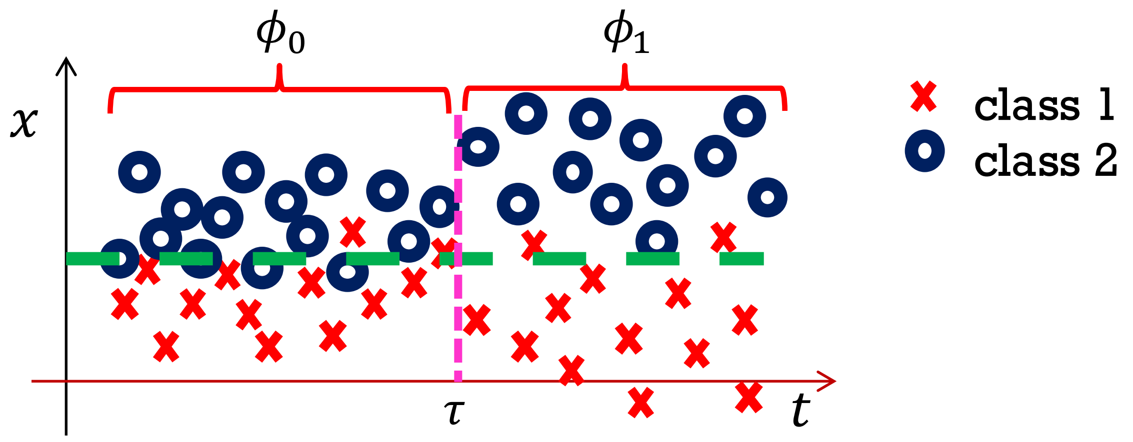
DRIFT TAXONOMY: WHAT IS CHANGING?

Virtual Drift

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

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

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

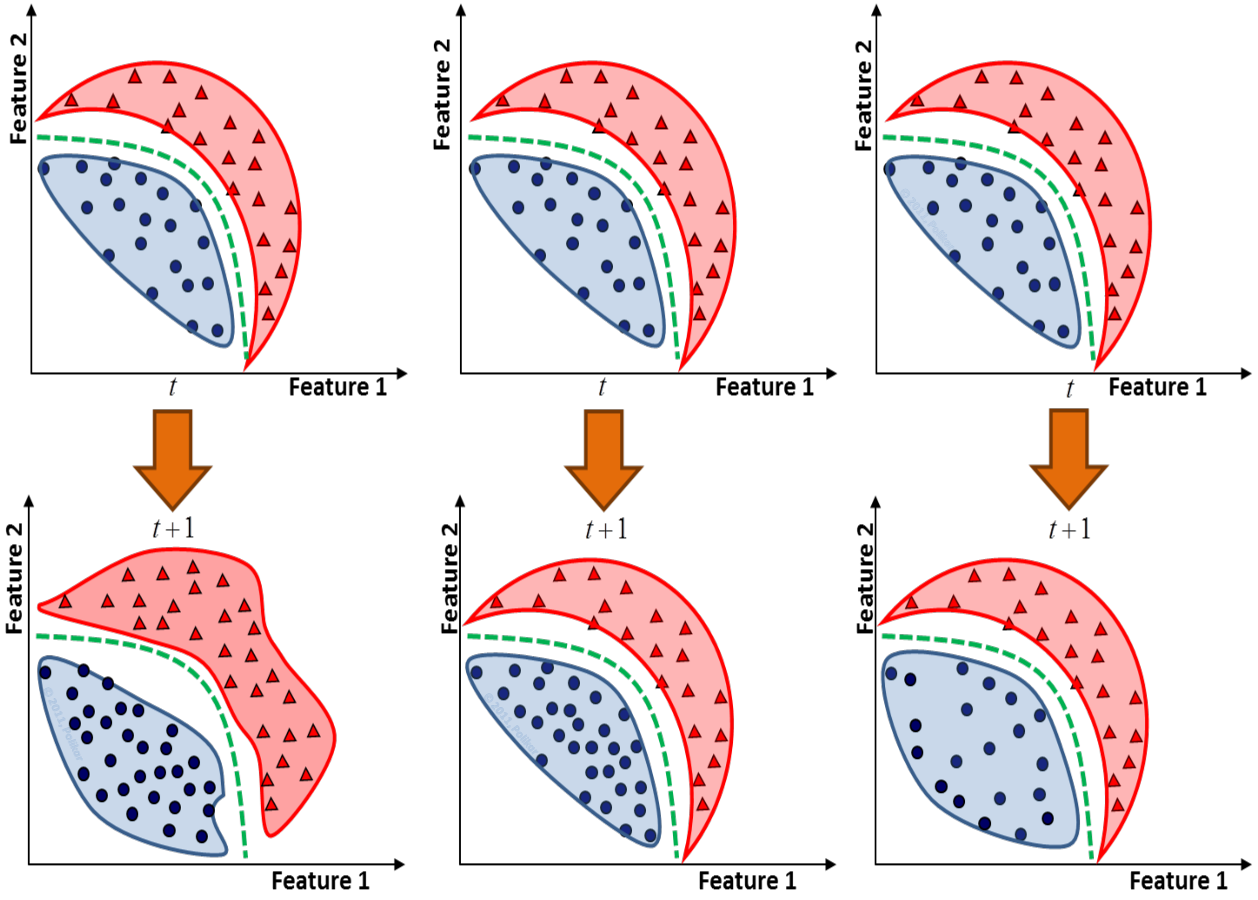


DRIFT TAXONOMY: WHAT IS CHANGING?

Virtual Drift

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

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

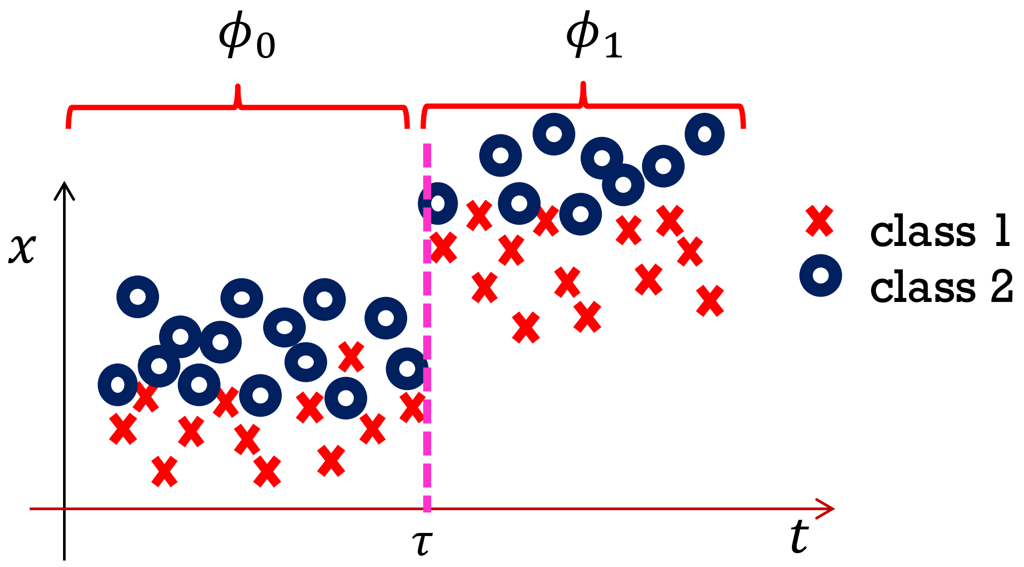


DRIFT TAXONOMY: TIME EVOLUTION

Abrupt

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

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

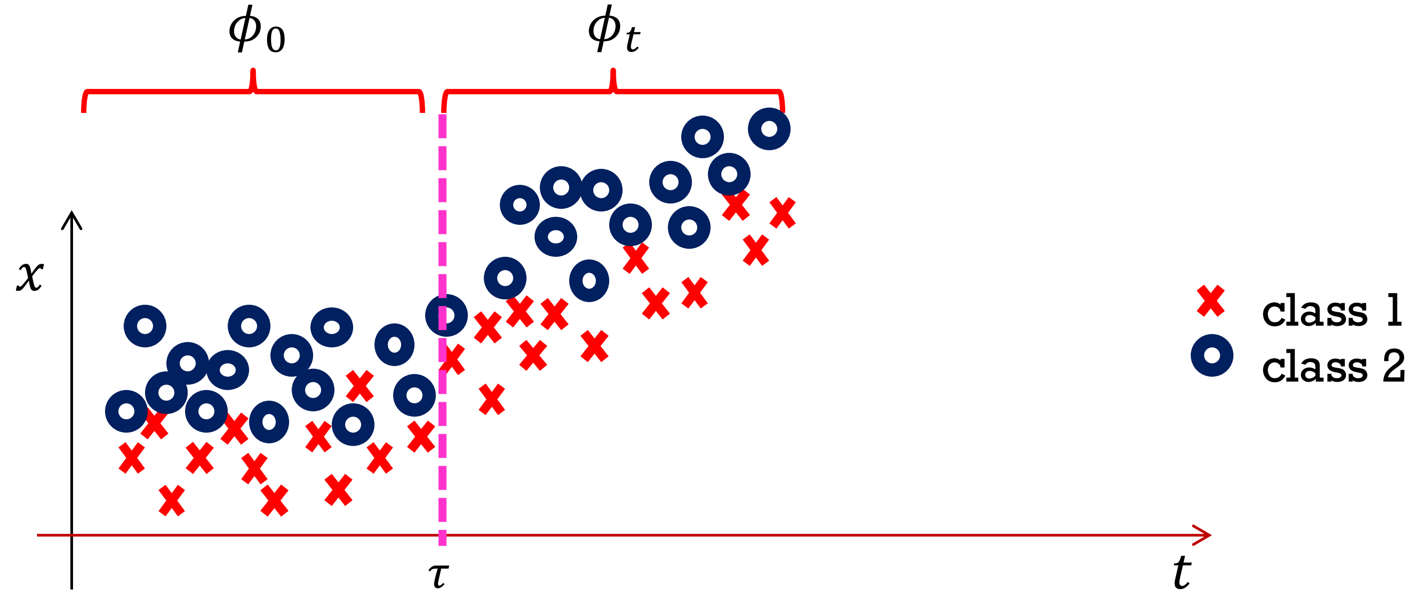


DRIFT TAXONOMY: TIME EVOLUTION

Incremental

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

There is a continuously drifting condition after the change

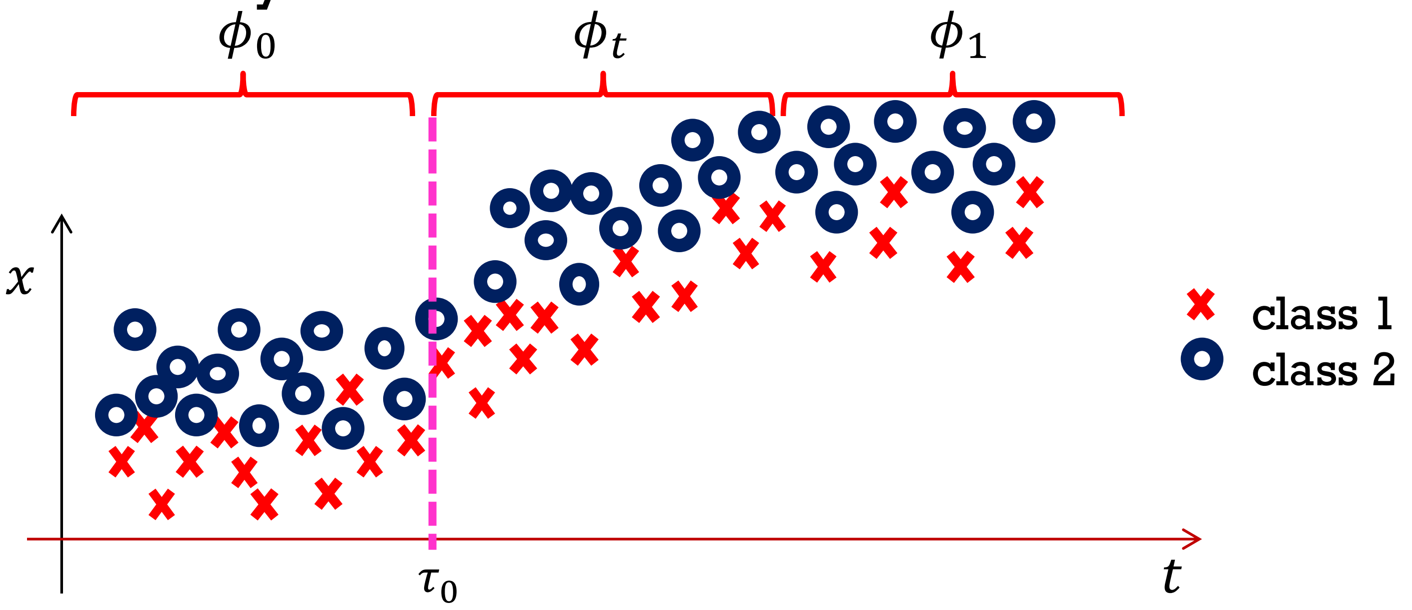


DRIFT TAXONOMY: TIME EVOLUTION

Incremental

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

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

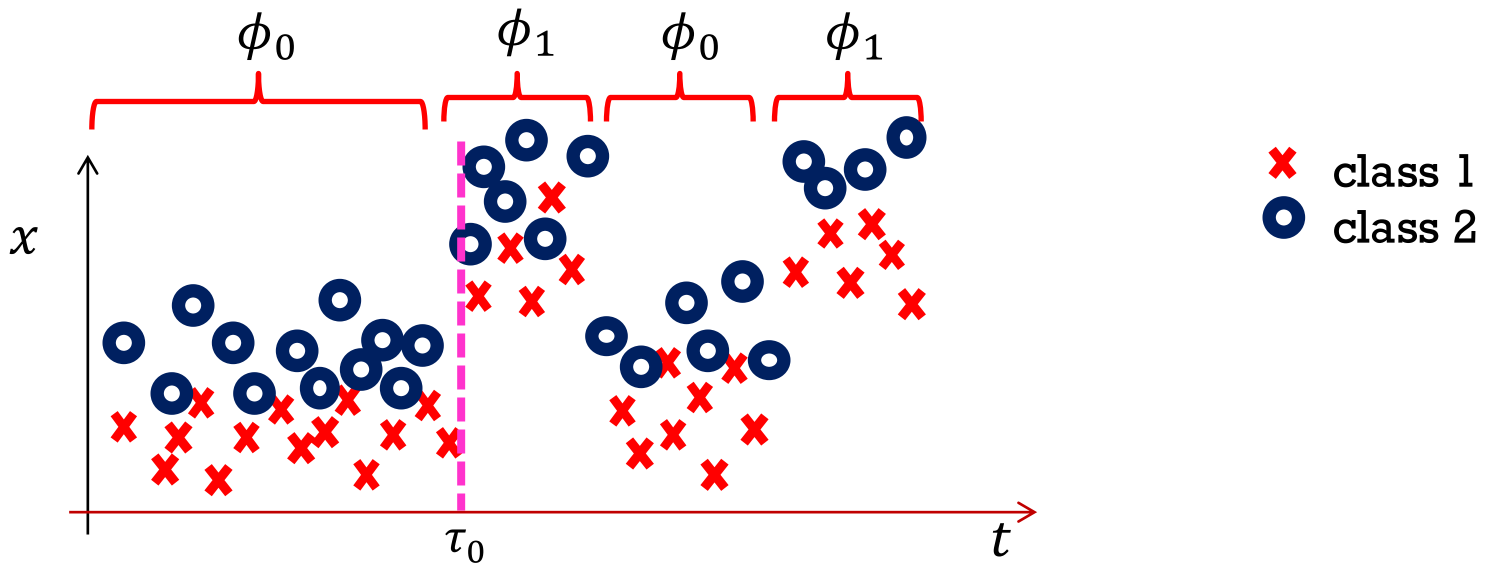


DRIFT TAXONOMY: TIME EVOLUTION

Recurring

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

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

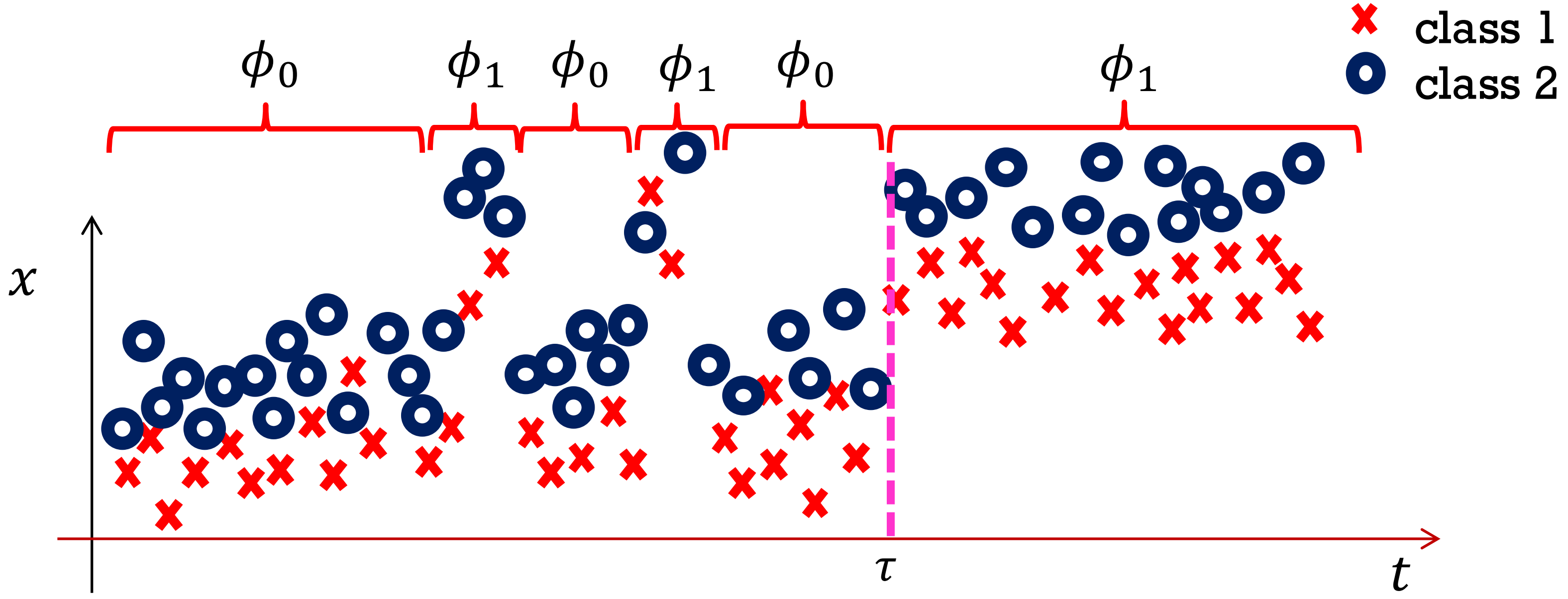


DRIFT TAXONOMY: TIME EVOLUTION

Gradual

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

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

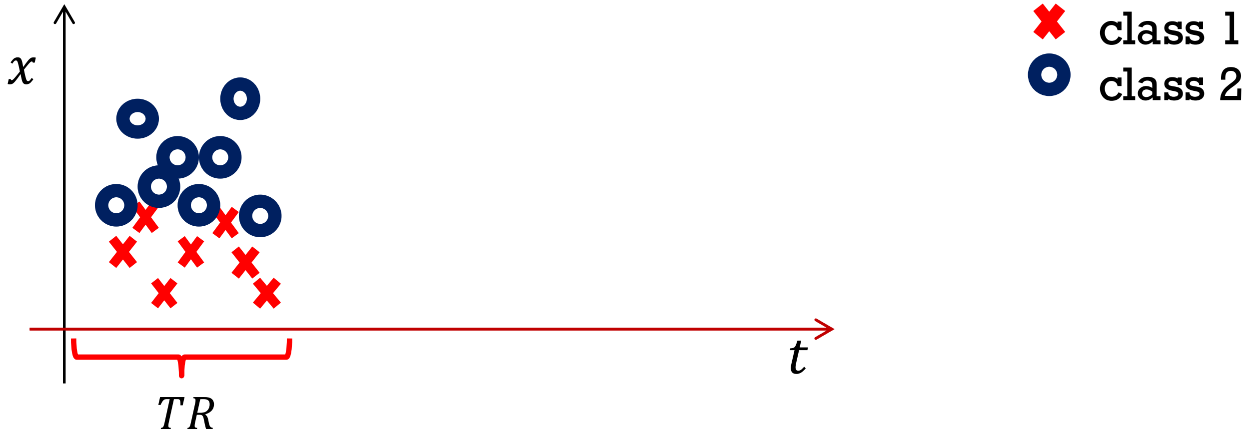


IS CONCEPT DRIFT A PROBLEM?

CLASSIFICATION OVER DATASTREAMS

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

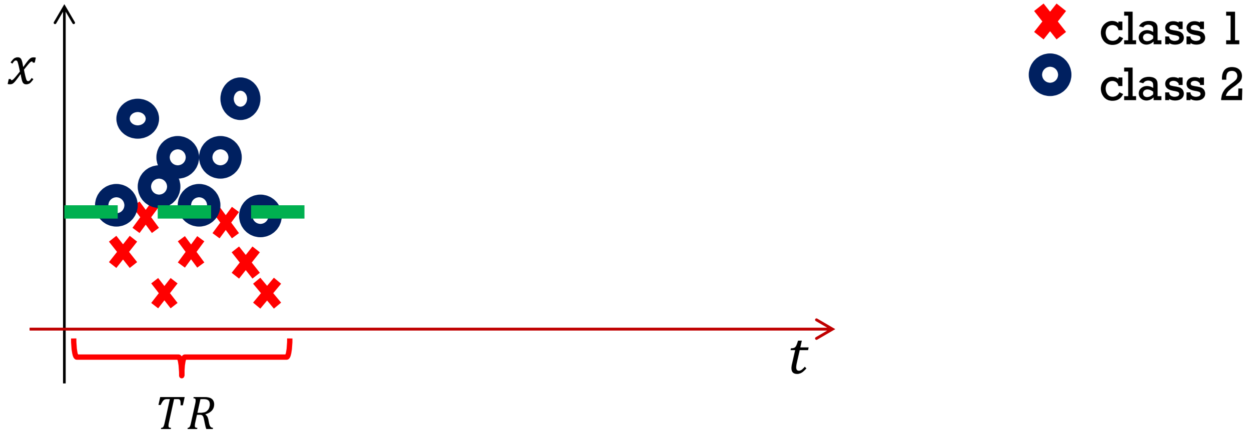
- The initial part of the stream is provided for training



CLASSIFICATION OVER DATASTREAMS

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

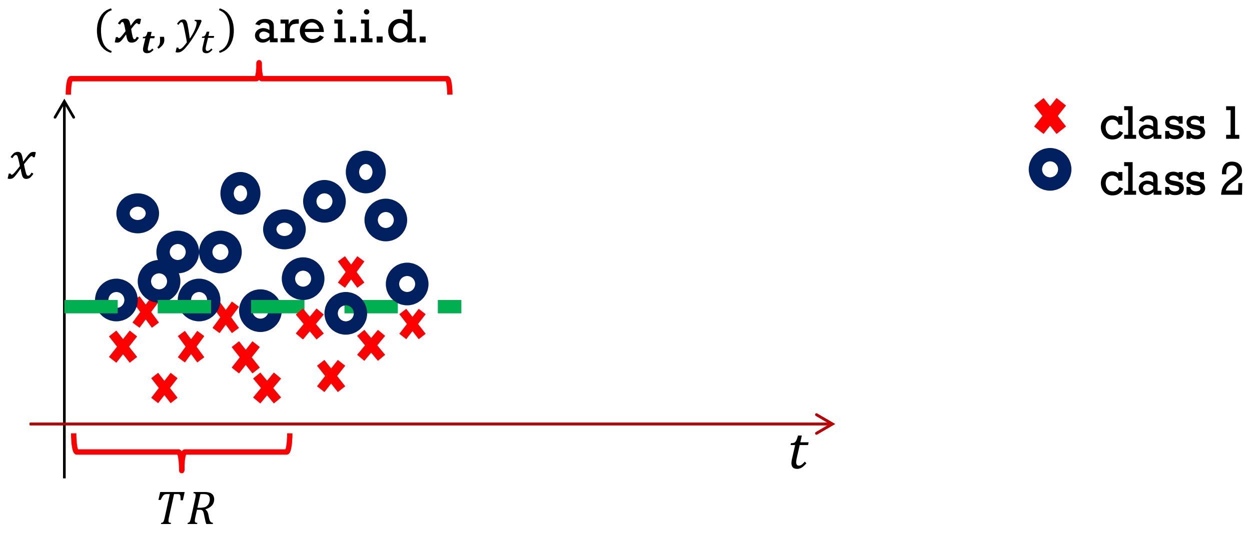
- The initial part of the stream is provided for training
- K is simply a threshold



CLASSIFICATION OVER DATASTREAMS

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

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

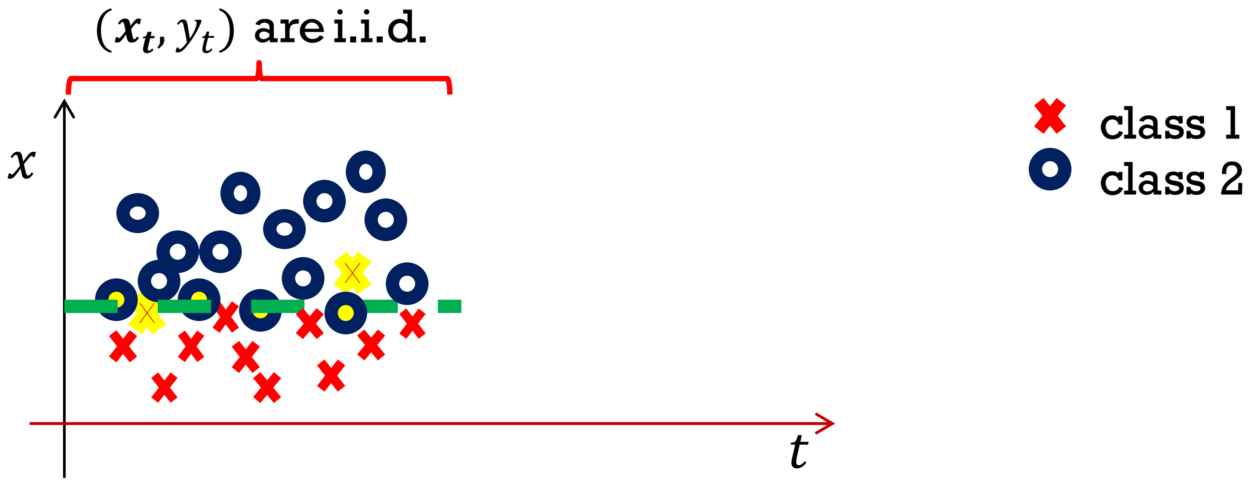


CLASSIFICATION OVER DATASTREAMS

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

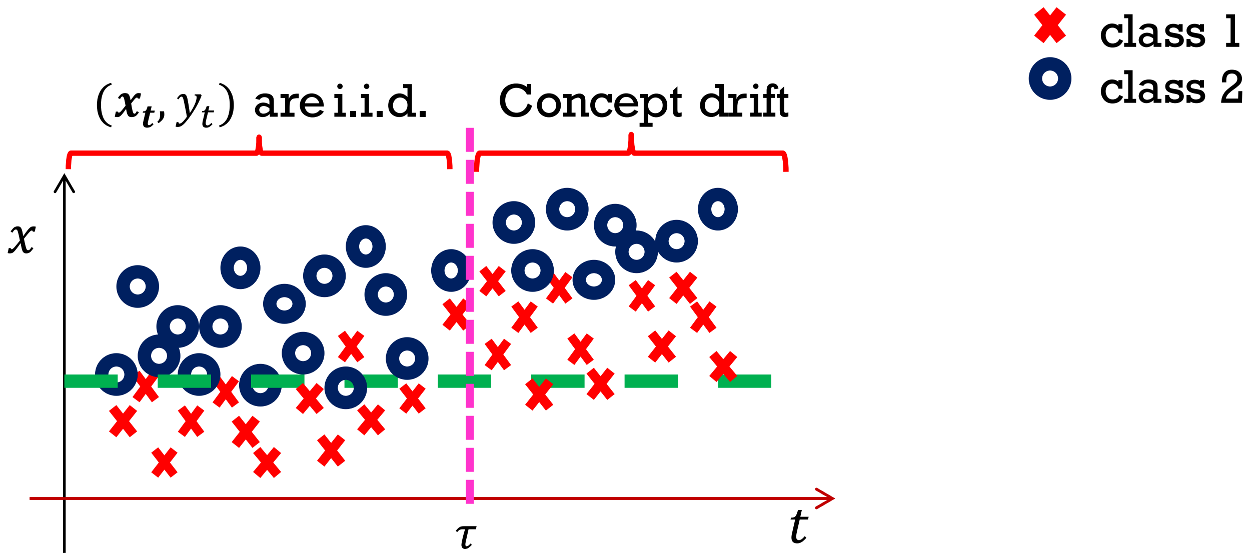
- The initial part of the stream is provided for training
- K is simply a threshold

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



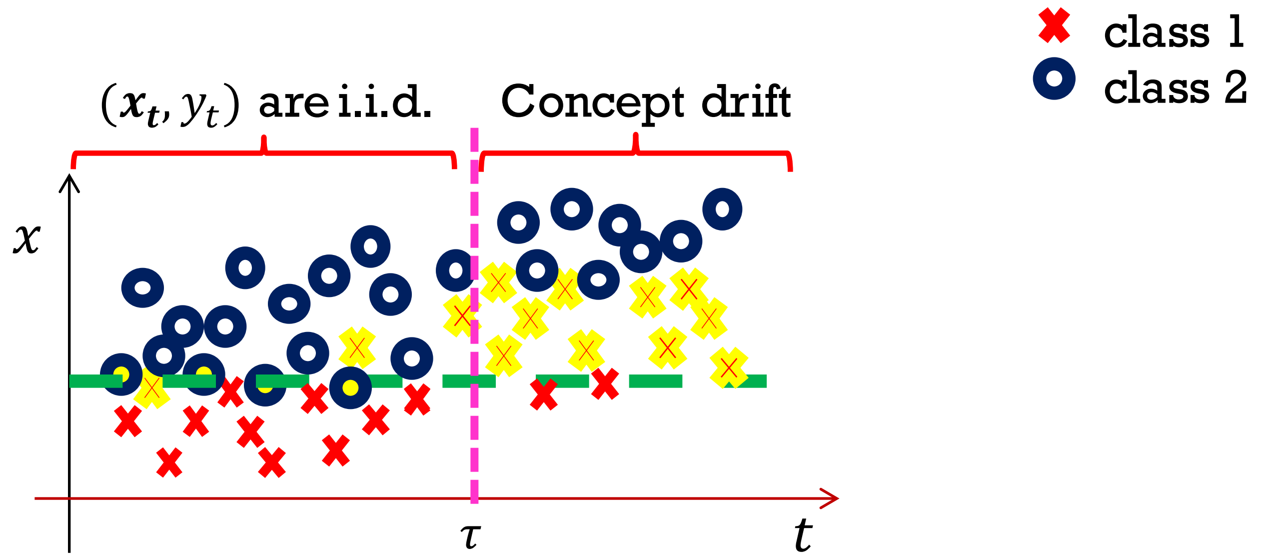
CLASSIFICATION OVER DATASTREAMS

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



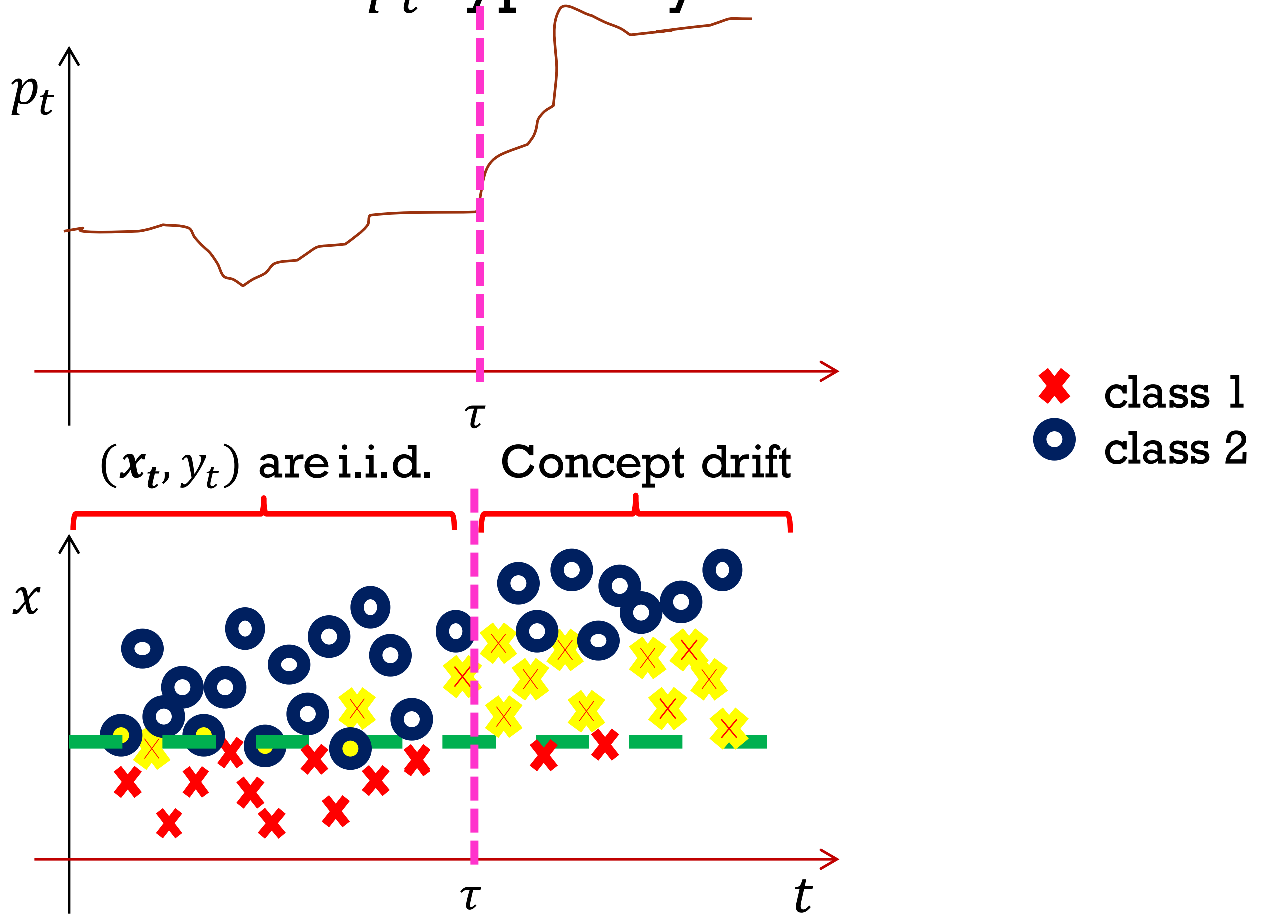
CLASSIFICATION OVER DATASTREAMS

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



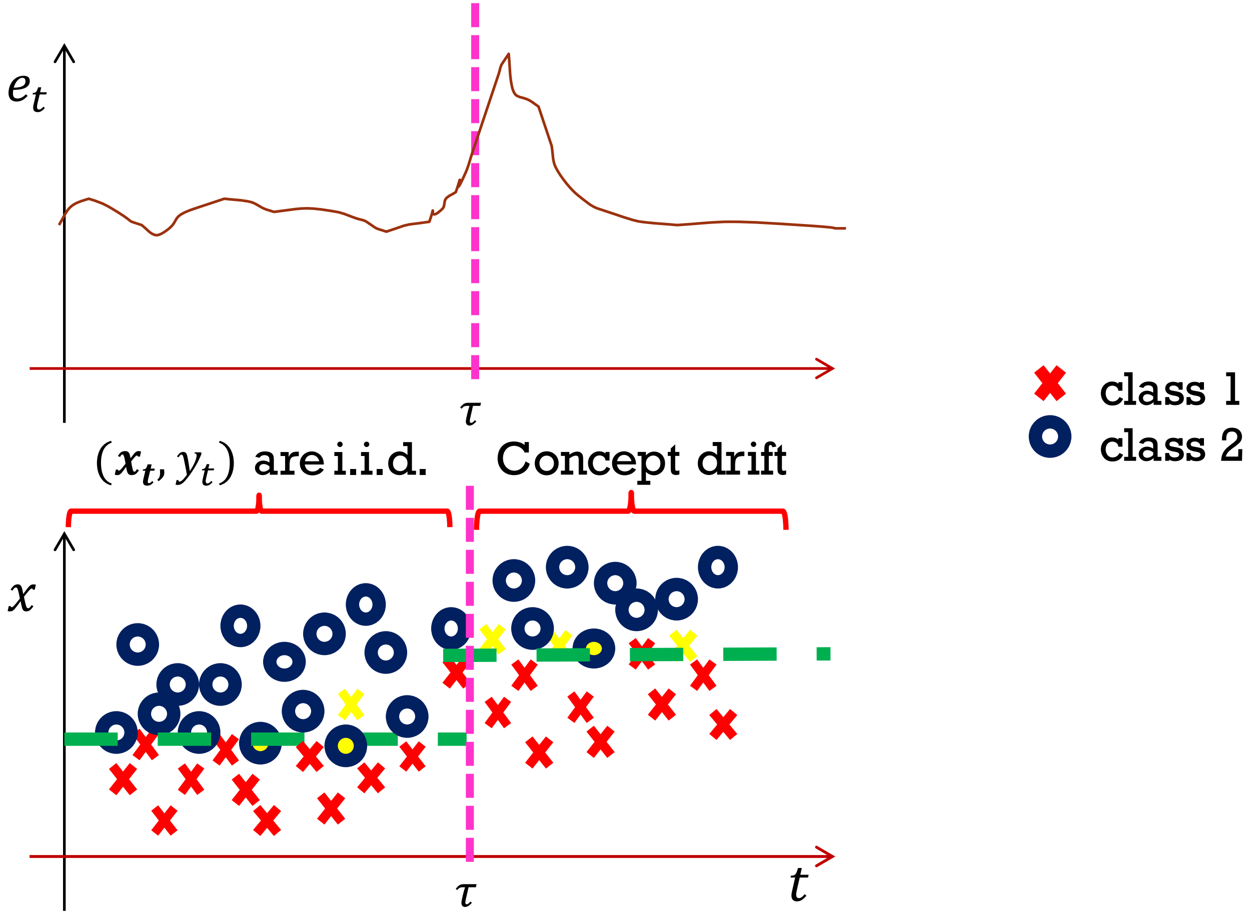
CLASSIFICATION OVER DATASTREAMS

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



NEED FOR ADAPTATION

Adaptation is needed to **preserve** classifier performance



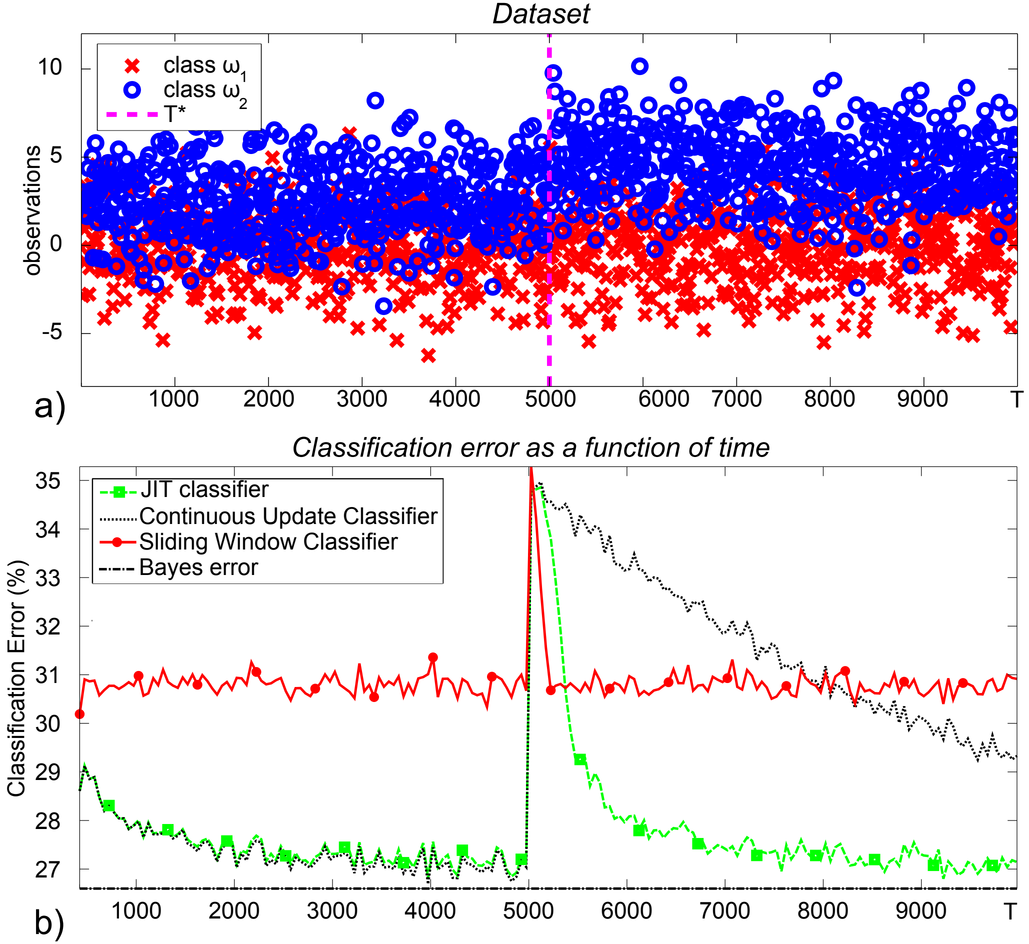
ADAPTATION

Do we Really Need Smart Adaptation Strategies?

SIMPLE ADAPTATION STRATEGIES

Consider two simple adaptation strategies and a simple concept drift

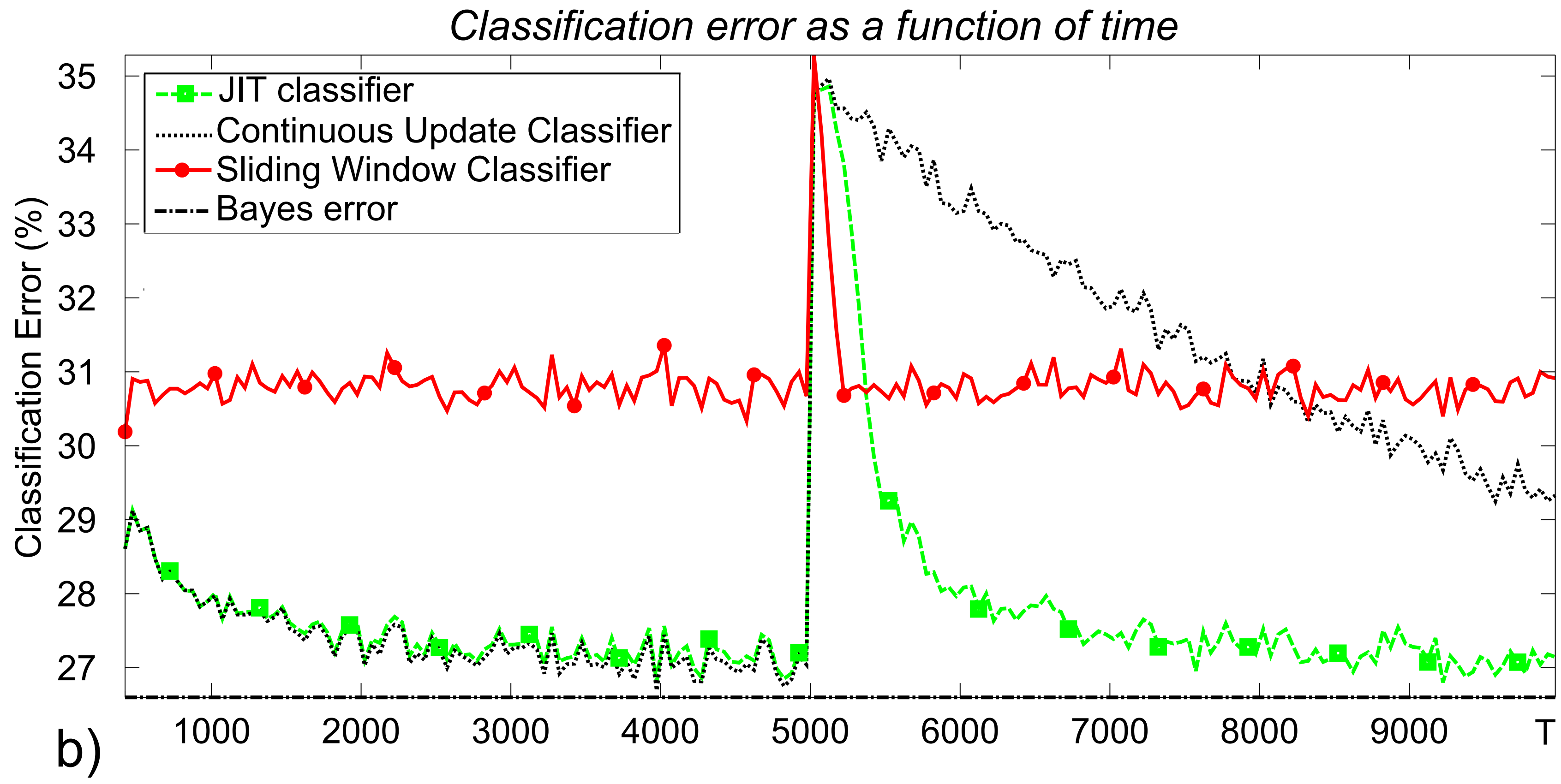
- Continuously update K_t using all supervised couples
- Train K_t using only the last δ supervised couples



SIMPLE ADAPTATION STRATEGIES

Classification error of two simple adaptation strategies

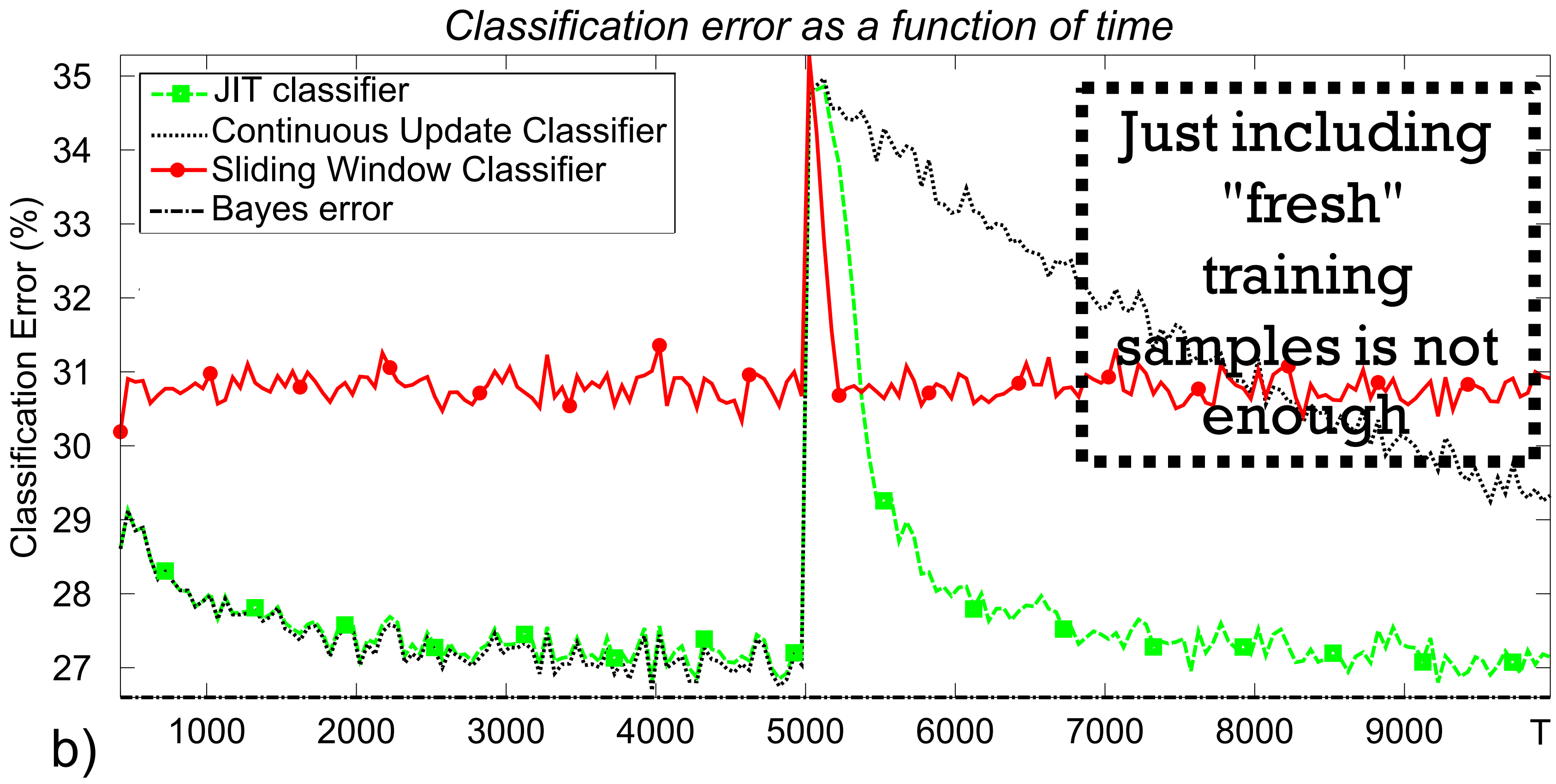
- Black dots: K_t uses all supervised couples at time t
- Red line: K_t uses only the last δ supervised couples



SIMPLE ADAPTATION STRATEGIES

Classification error of two simple adaptation strategies

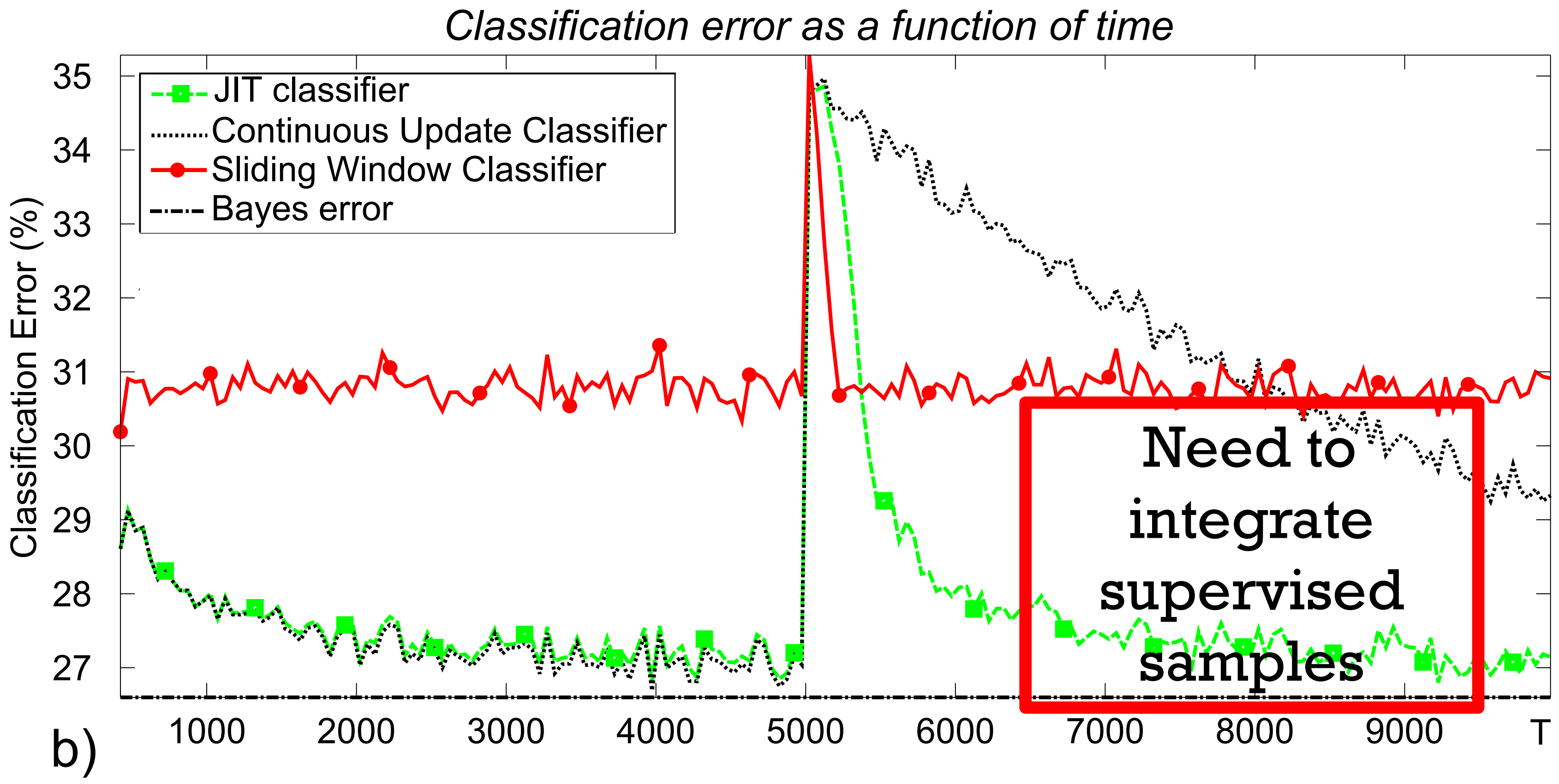
- Black dots: K_t uses all supervised couples at time t
- Red line: K_t uses only the last δ supervised couples



SIMPLE ADAPTATION STRATEGIES

Classification error of two simple adaptation strategies

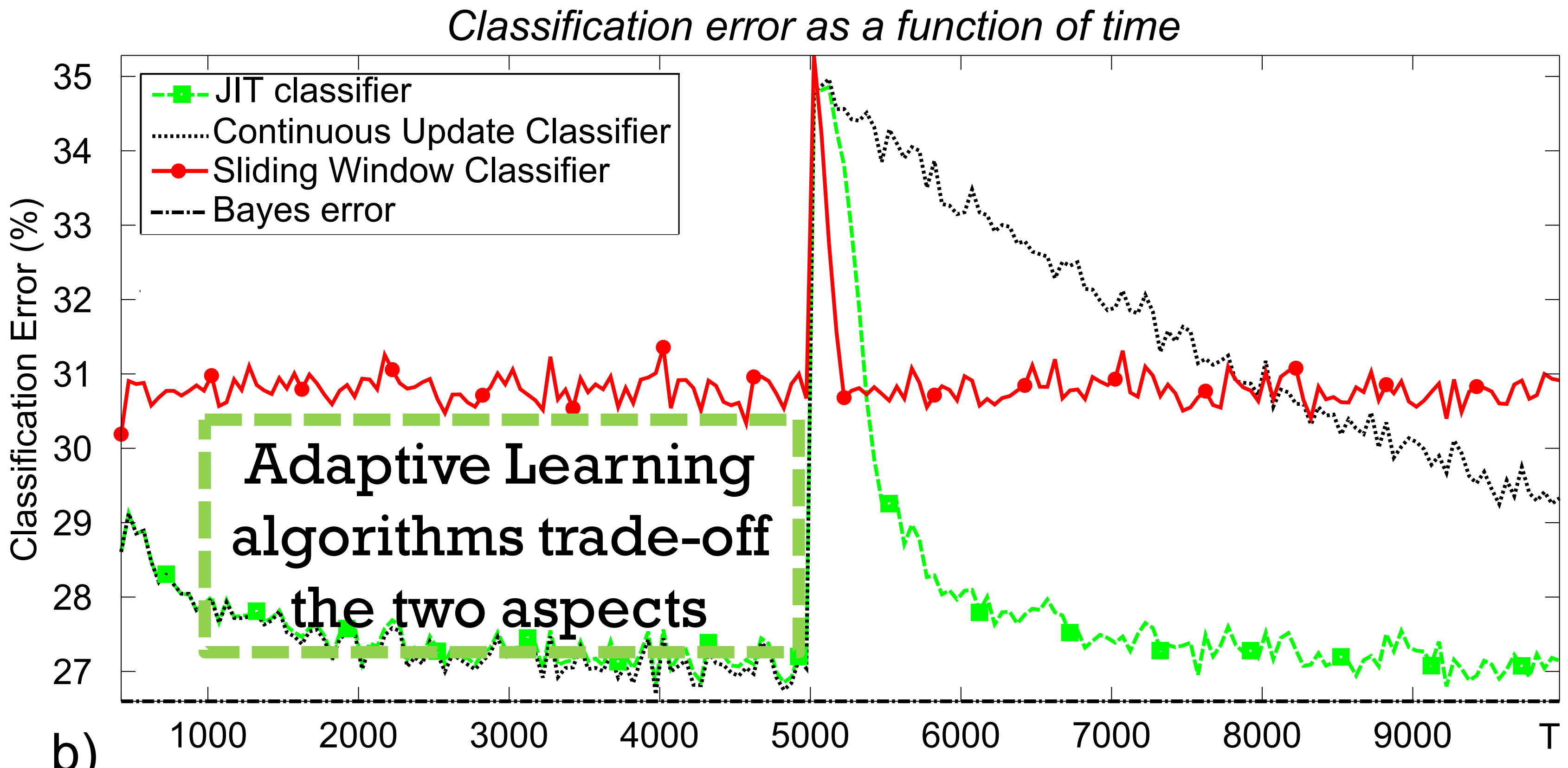
- Black dots: K_t uses all supervised couples at time t
- Red line: K_t uses only the last δ supervised couples



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



b)

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

LNSE: ACTIVE APPROACHES

Giacomo Boracchi[1] and Gregory Ditzler[2]

[1] Politecnico di Milano
Dipartimento Elettronica e Informazione
Milano, Italy

[2] The University of Arizona
Department of Electrical & Computer Engineering
Tucson, AZ USA

giacomo.boracchi@polimi.it, ditzler@email.arizona.edu

ACTIVE APPROACHES

Peculiarities:

- Relies on an **explicit drift-detection mechanism**: the 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 THE CLASSIFICATION ERROR

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

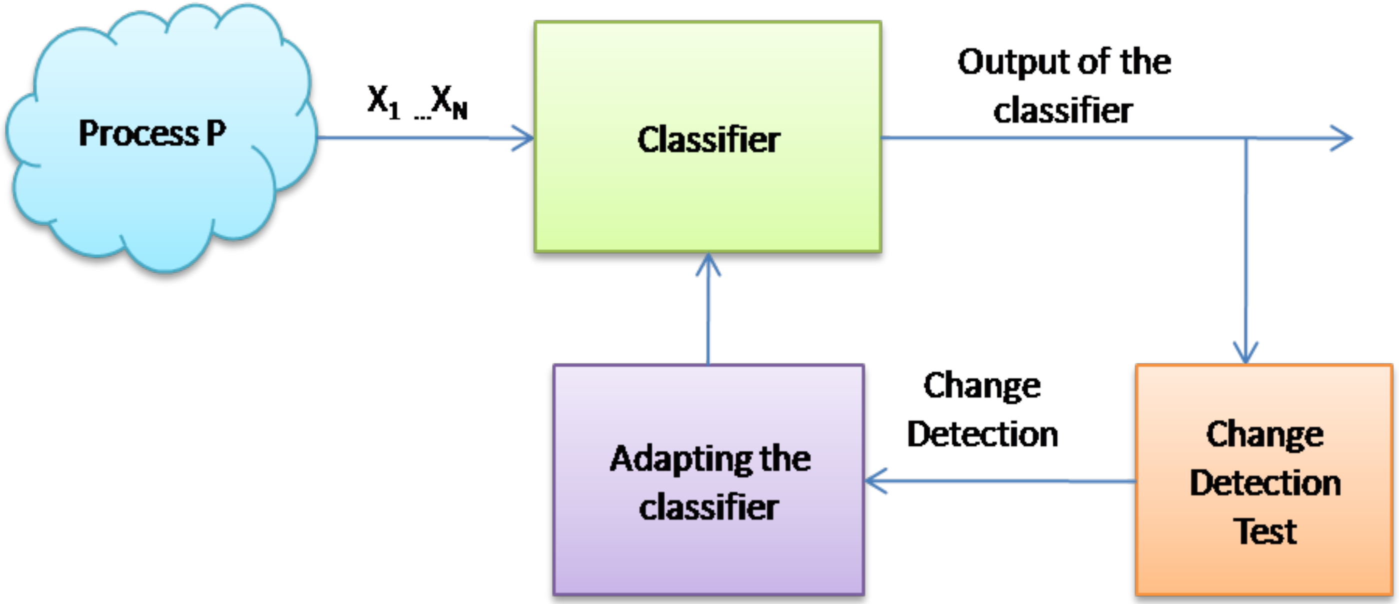
Pro:

- 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



MONITORING THE CLASSIFICATION ERROR

- The element-wise classification error follows a **Bernoulli** pdf

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

π_0 is the expected classification error in stationary conditions

- The sum of e_t in a window follows a **Binomial** pdf

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

- Gaussian approximation 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

MONITORING THE CLASSIFICATION ERROR: DDM

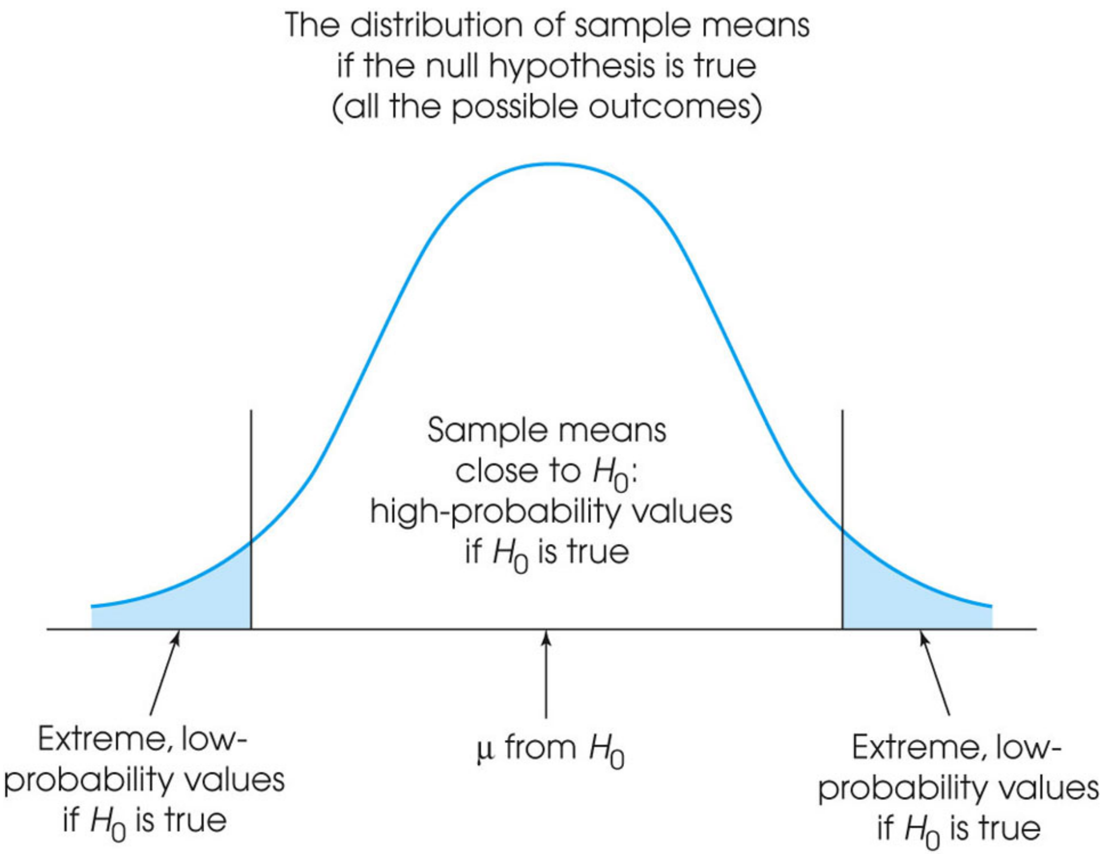
Basic idea behind Drift Detection Method (DDM):

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

MONITORING THE CLASSIFICATION ERROR: DDM

Basic idea behind Drift Detection Method (DDM):

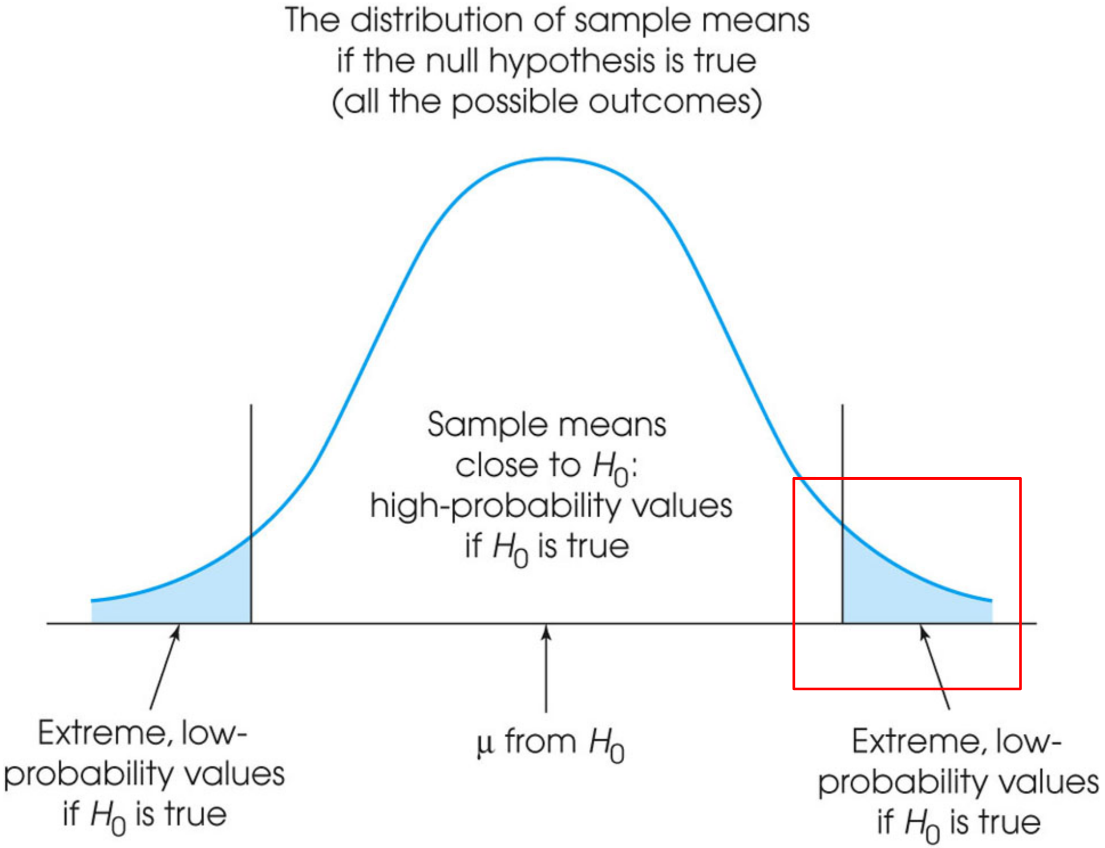
- Detect CD as **outliers** in the classification error



MONITORING THE CLASSIFICATION ERROR: DDM

Basic idea behind Drift Detection Method (DDM):

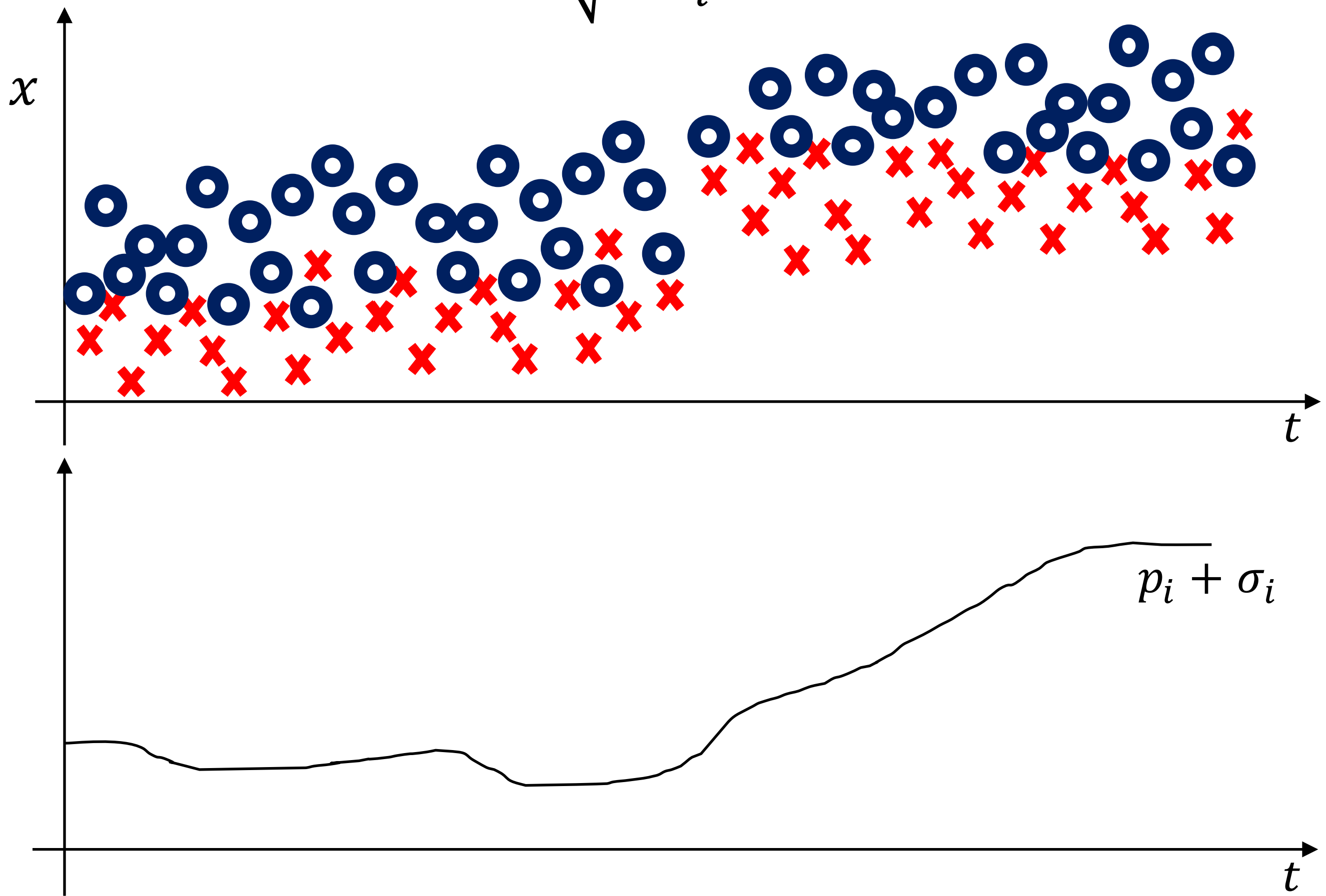
- Detect CD as **outliers** in the classification error
- Since in stationary conditions error will decrease, look for outliers in the right tail only



MONITORING THE CLASSIFICATION ERROR: DDM

Basic idea behind Drift Detection Method (DDM):

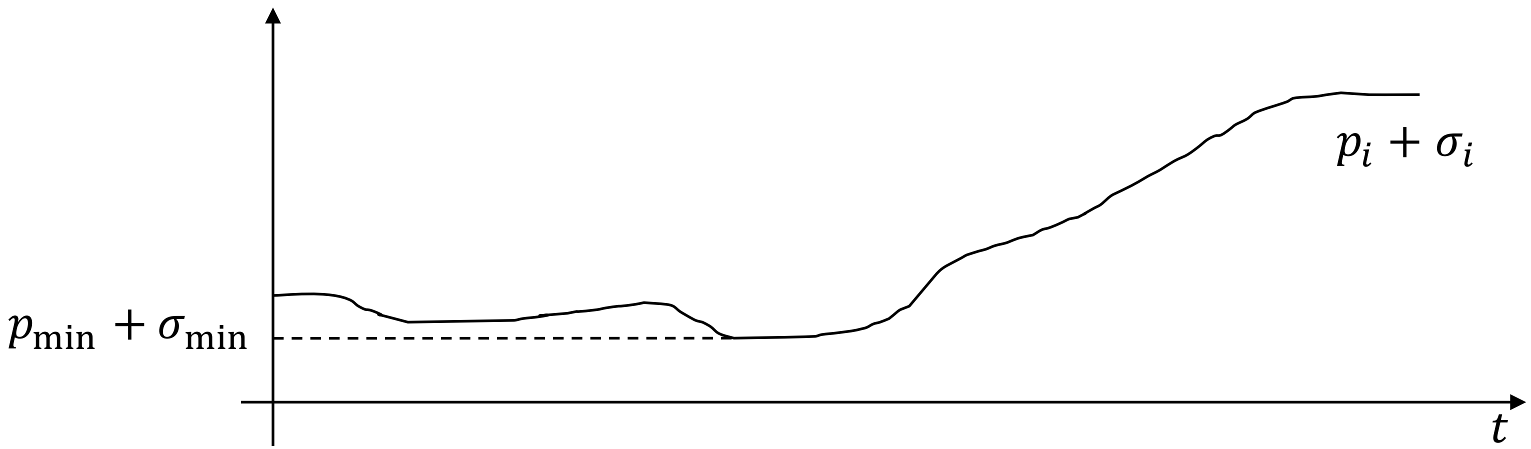
- Detect CD as **outliers** in the classification error
- Compute, over time p_i, and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$



MONITORING THE CLASSIFICATION ERROR: DDM

Basic idea behind Drift Detection Method (DDM):

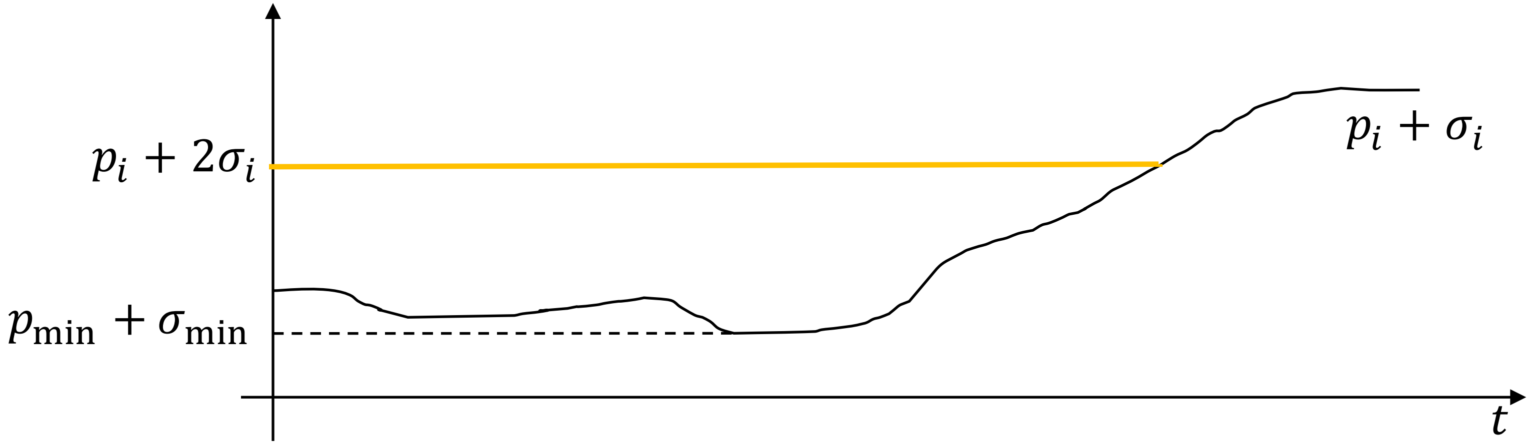
- Detect CD as **outliers** in the classification error
- Compute, over time p_i, and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$
- Let $p_{\min}$ be the minimum error, $\sigma_{\min} = \sqrt{\frac{p_{\min}(1-p_{\min})}{i}}$



MONITORING THE CLASSIFICATION ERROR: DDM

Basic idea behind Drift Detection Method (DDM):

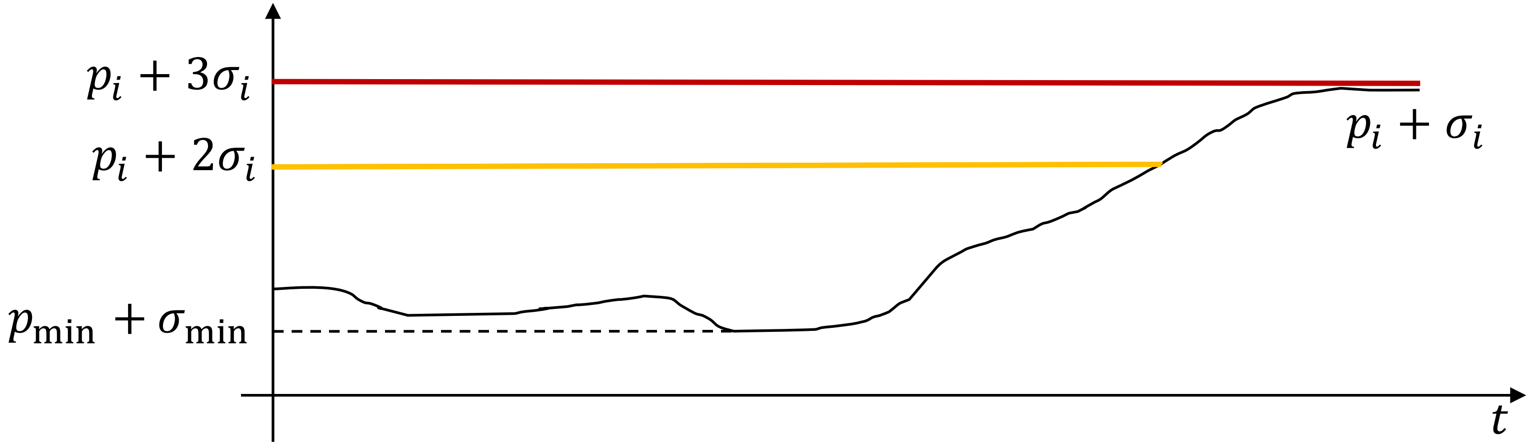
- Detect CD as **outliers** in the classification error
- Compute, over time p_i, and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$
- Let $p_{\min}$ be the minimum error, $\sigma_{\min} = \sqrt{\frac{p_{\min}(1-p_{\min})}{i}}$
- When $p_i + \sigma_i > p_{\min} + 2 * \sigma_{\min}$ raise a warning alert



MONITORING THE CLASSIFICATION ERROR: DDM

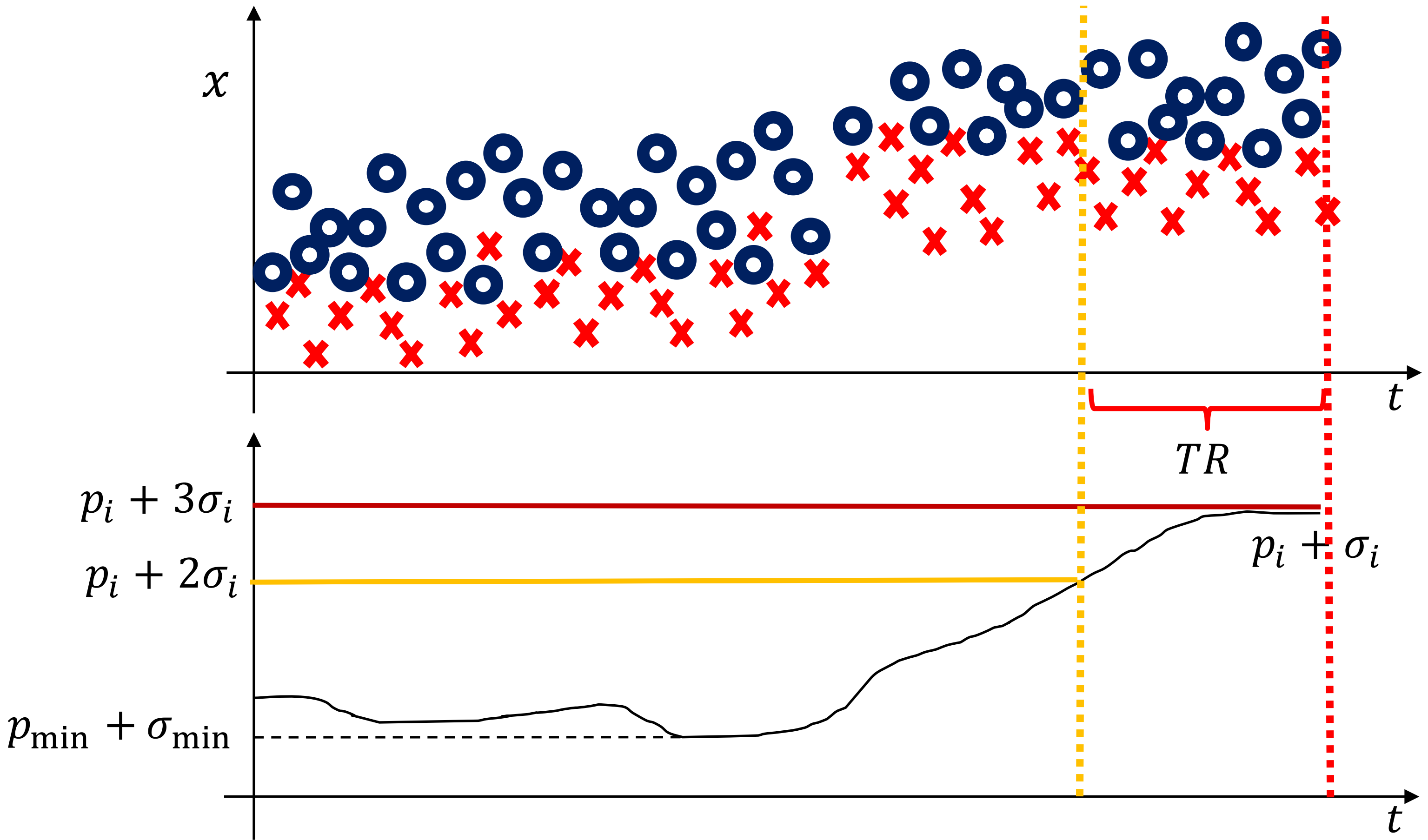
Basic idea behind Drift Detection Method (DDM):

- 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

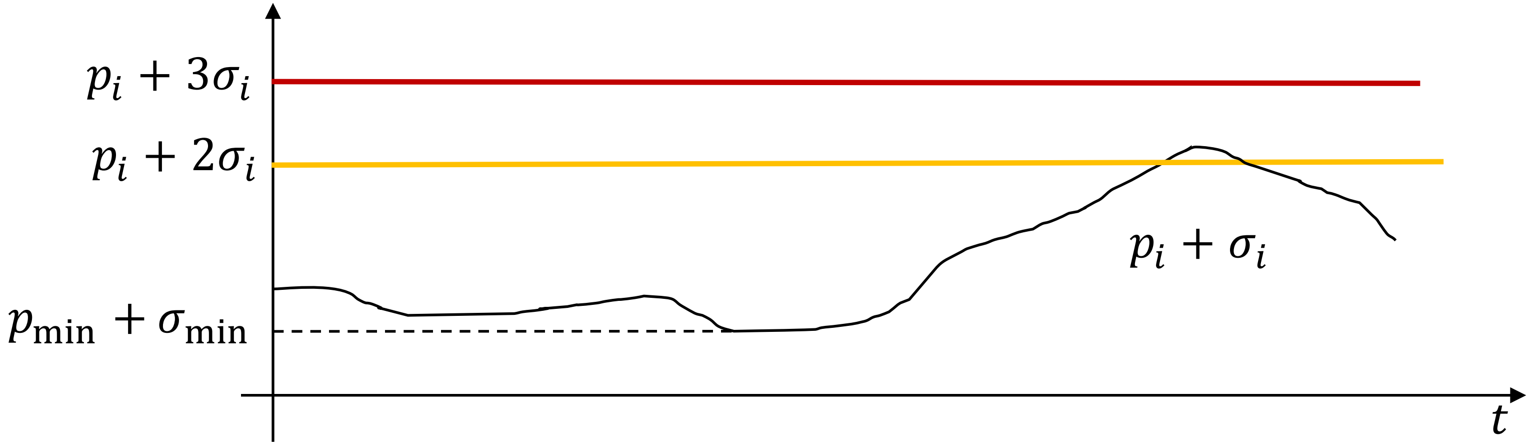
Use supervised samples in **between warning** and **drift** alert to **reconfigure** the classifier



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 THE CLASSIFICATION ERROR: EDDM

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 THE 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 THE 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*

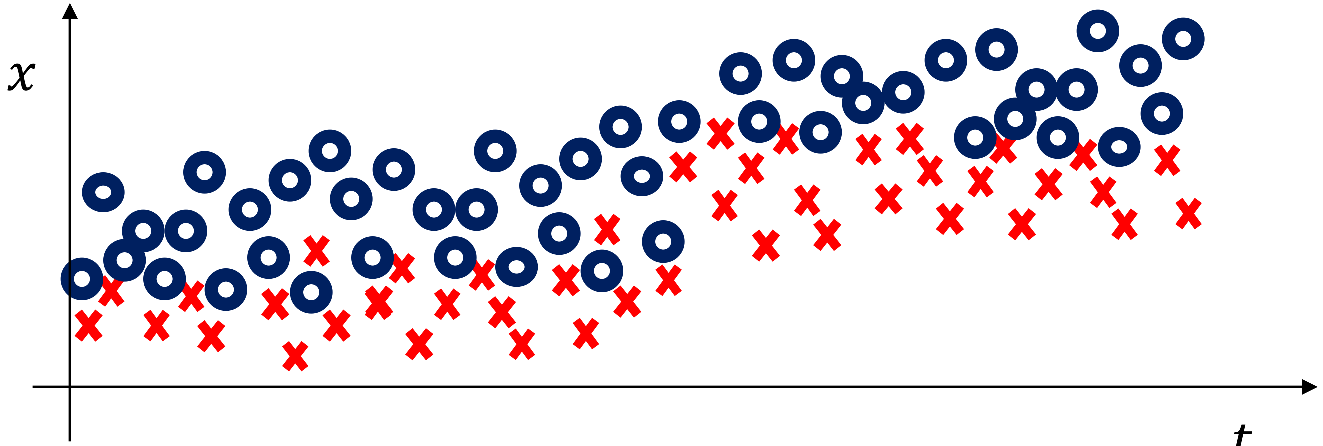
$$Z_t > p_{0,t} + 0.5\, L_t \sigma_t$$

- 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

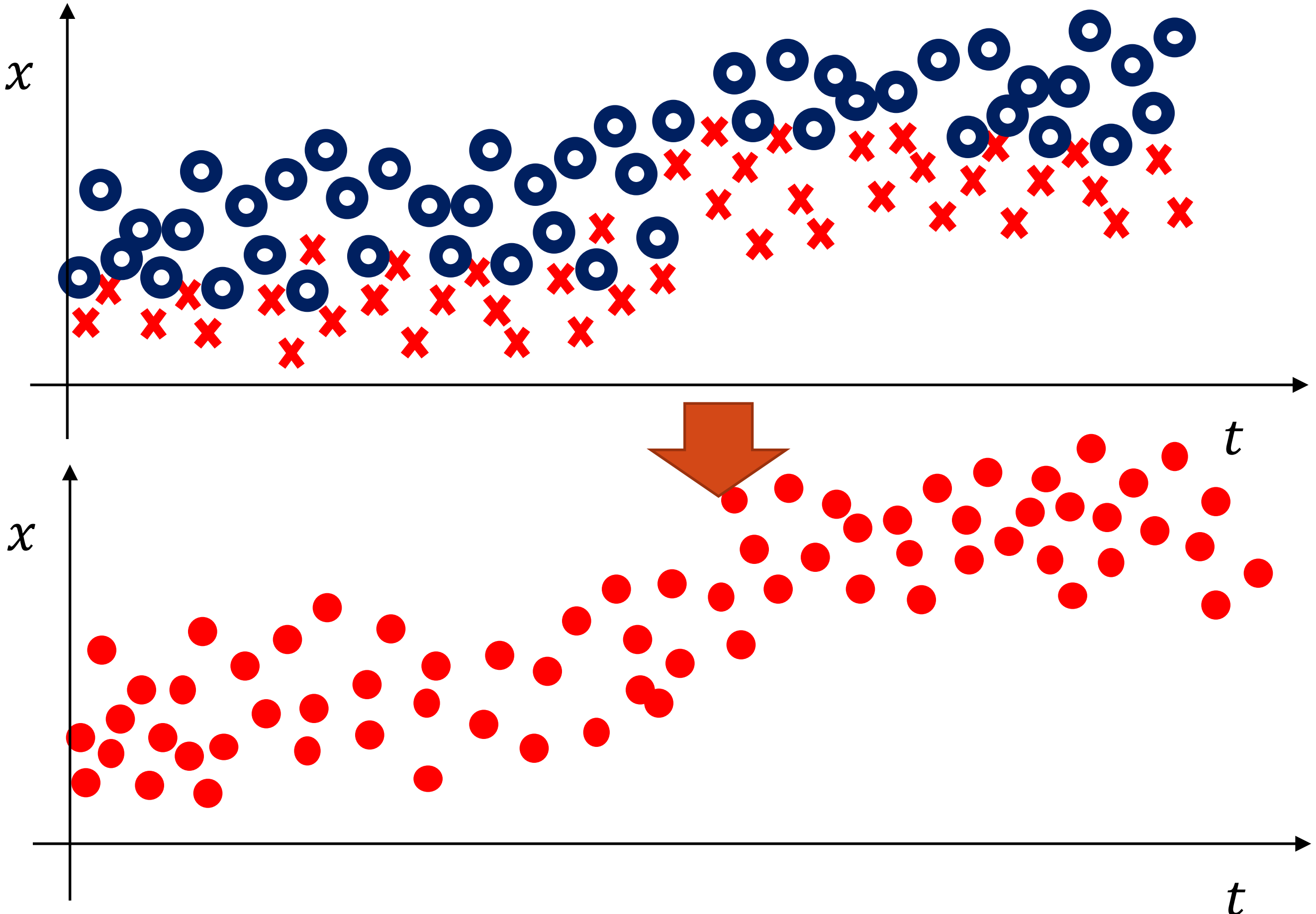
MONITORING THE RAW DATA

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

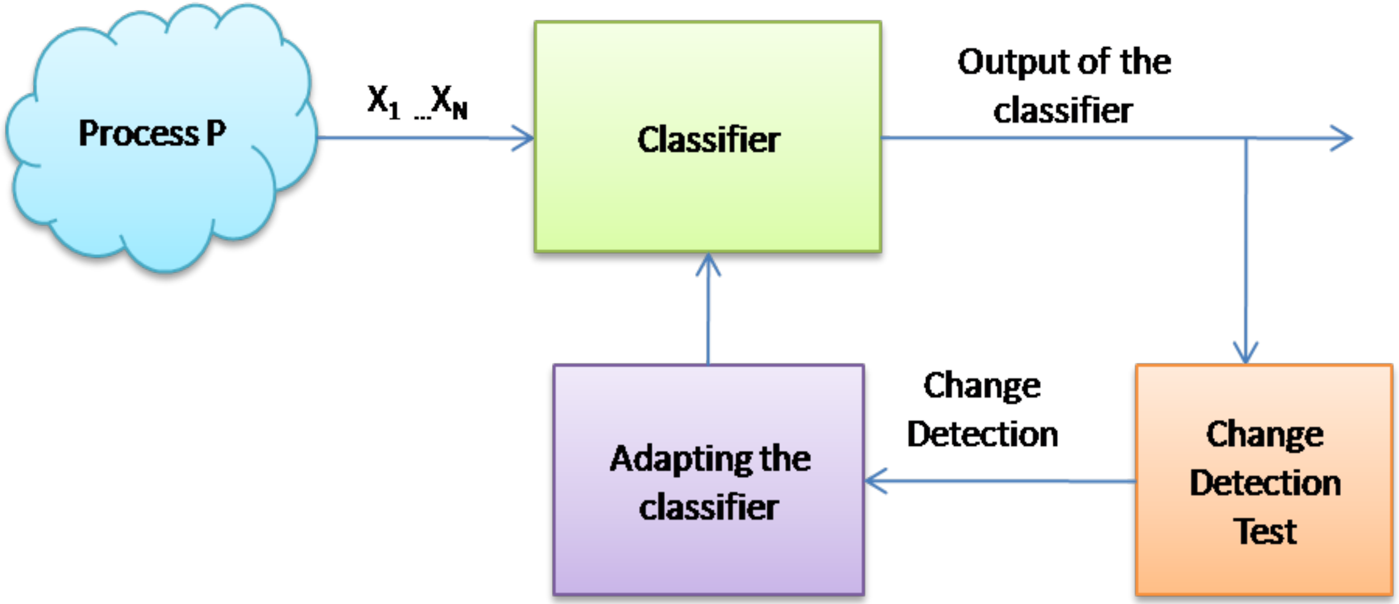


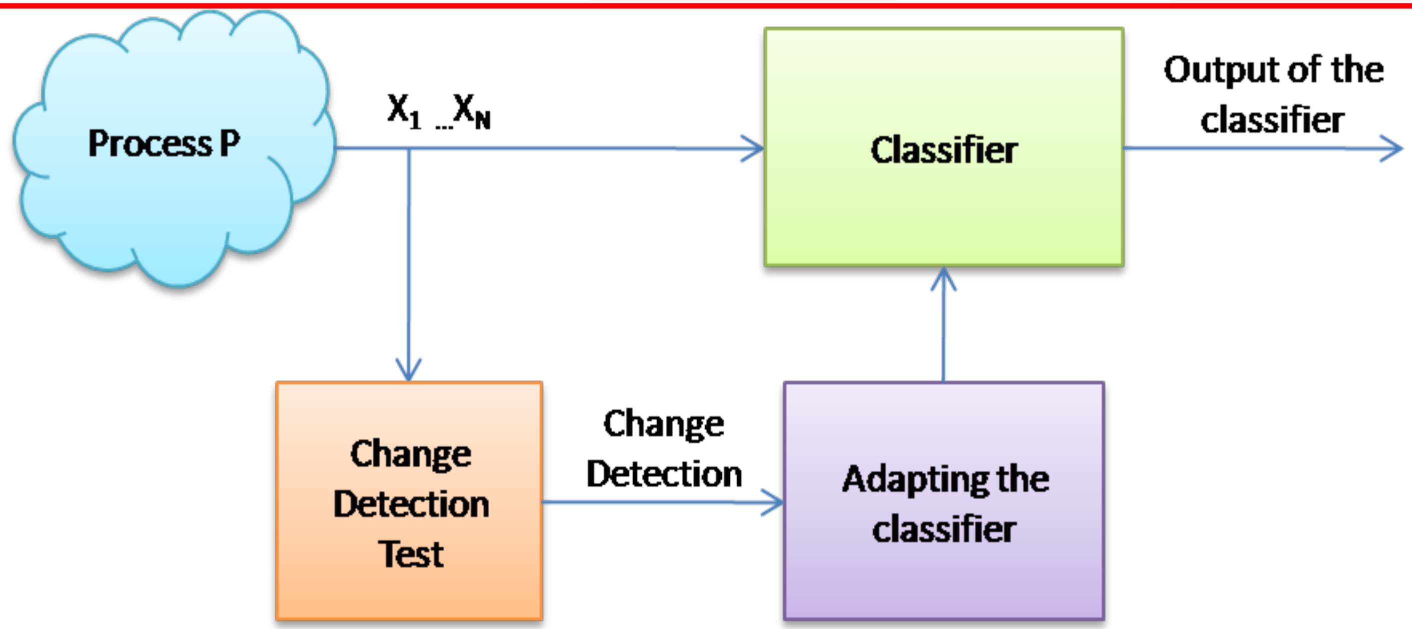
MONITORING THE RAW DATA

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 $\phi(x)$ **does not require supervised samples**
- Enables the detection of both **real and virtual concept drift**

Cons:

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

DETECTION TOOLS: ICI-BASED CDT

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

- 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

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

$\mathrm{S}(s)$ 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

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

DETECTION TOOLS: 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.

MONITORING (MULTIVARIATE) 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 $\phi(x)$ in stationary conditions

MONITORING THE 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 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)

C Alippi, G Boracchi, D Carrera, M Roveri *Change Detection in Multivariate Datastreams: Likelihood and Detectability Loss* - arXiv preprint arXiv:1510.04850, 2015

JUST-IN-TIME 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 a new training for the new concept and exploit of recurrent concepts

JIT classifiers leverage:

- **sequential techniques to detect CD,** monitoring both classification error and raw data distribution
- **statistical techniques to identify the new concept and possibly recurrent ones**

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 Outstanding Paper Award 2016

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

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

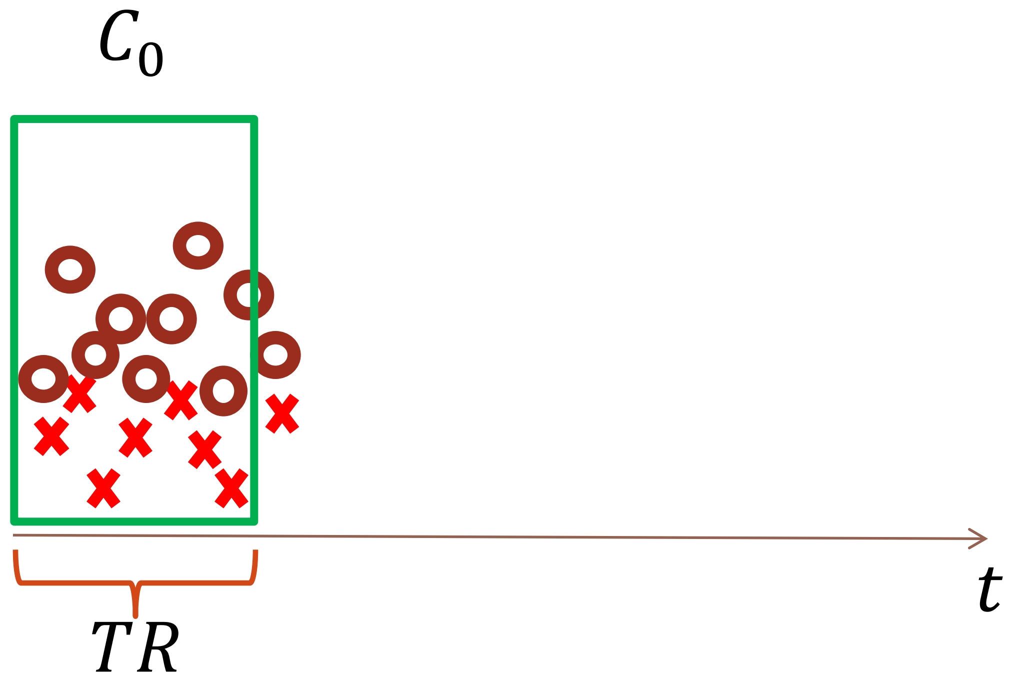
Initial Training

Use the initial training sequence to build the concept representation C_0

JIT CLASSIFIERS: INITIAL TRAINING

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

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



JIT CLASSIFIERS: THE ALGORITHM

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

During operations, each input sample is analyzed to:

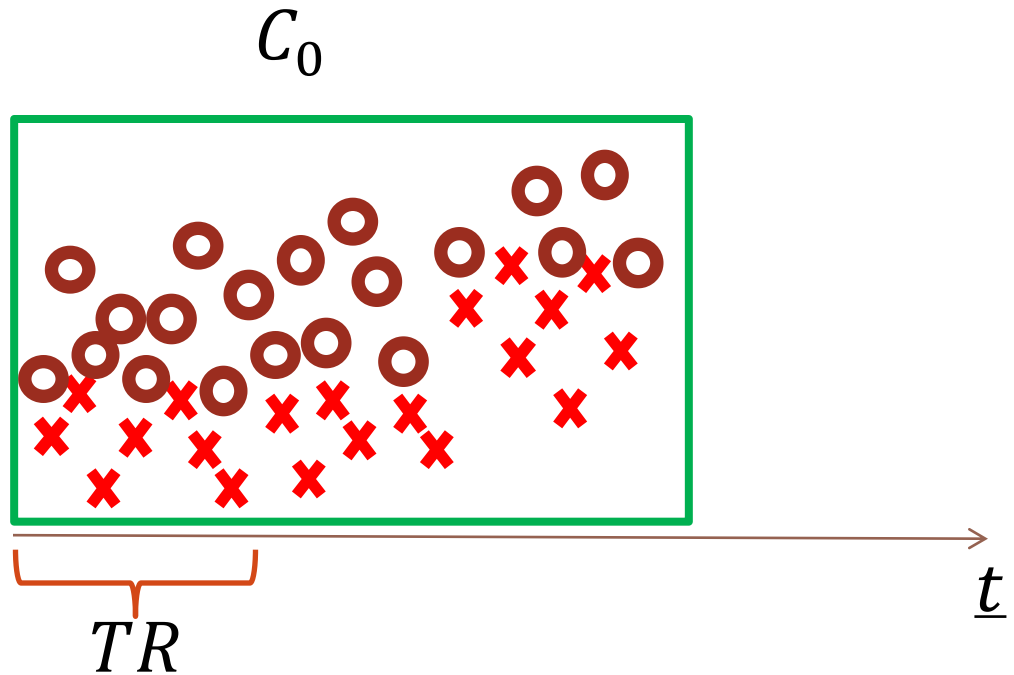
- Extract features that are appended to F_i
- Append supervised information in Z_i

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 $p(\boldsymbol{x})$)
- Computing feature D_0



JIT CLASSIFIERS: THE ALGORITHM

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

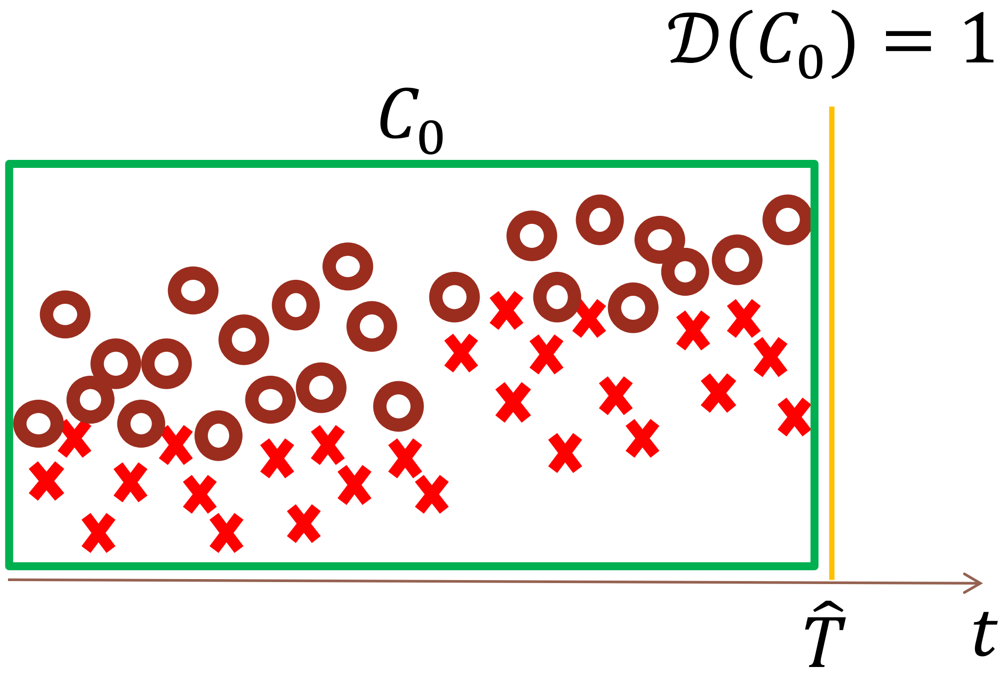
Concept Drift Detection:

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

JIT CLASSIFIERS: CONCEPT DRIFT DETECTION

Determine when **features in *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 $\phi(y|x)$ and $\phi(x)$ 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 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 $C_0 = (Z_0, F_0, D_0)$ 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- $\quad\quad \mathcal{U}(C_i, \{(x_t, y_t)\}) \to C_i$;
$\quad$ **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- $\quad\quad \widehat{y}_t = K(Z_i \cup Z_{\text{rec}}, x_t)$.
$\quad$ **end**
end

Concept Split:

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

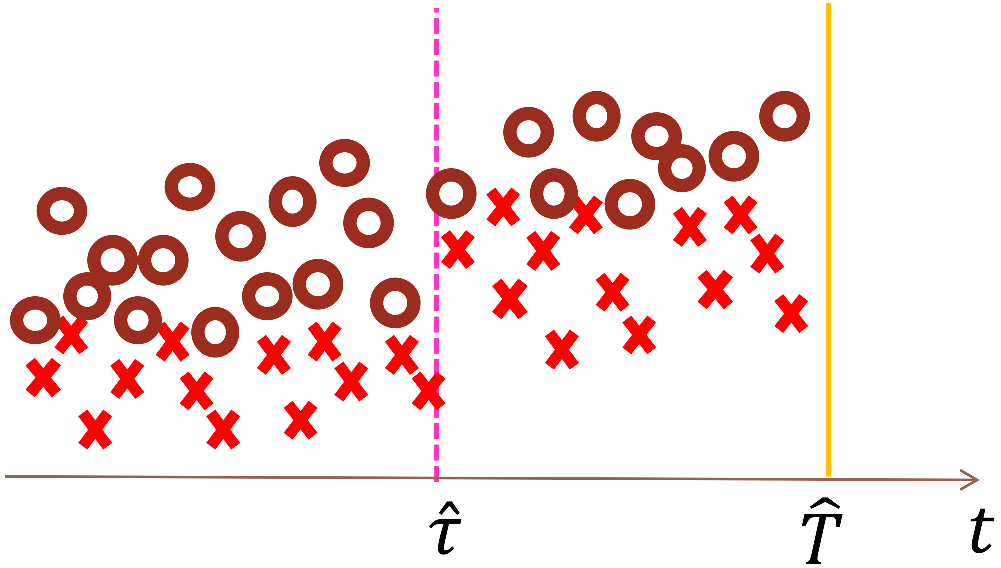
A new concept description is built

JIT CLASSIFIERS: CONCEPT SPLIT

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

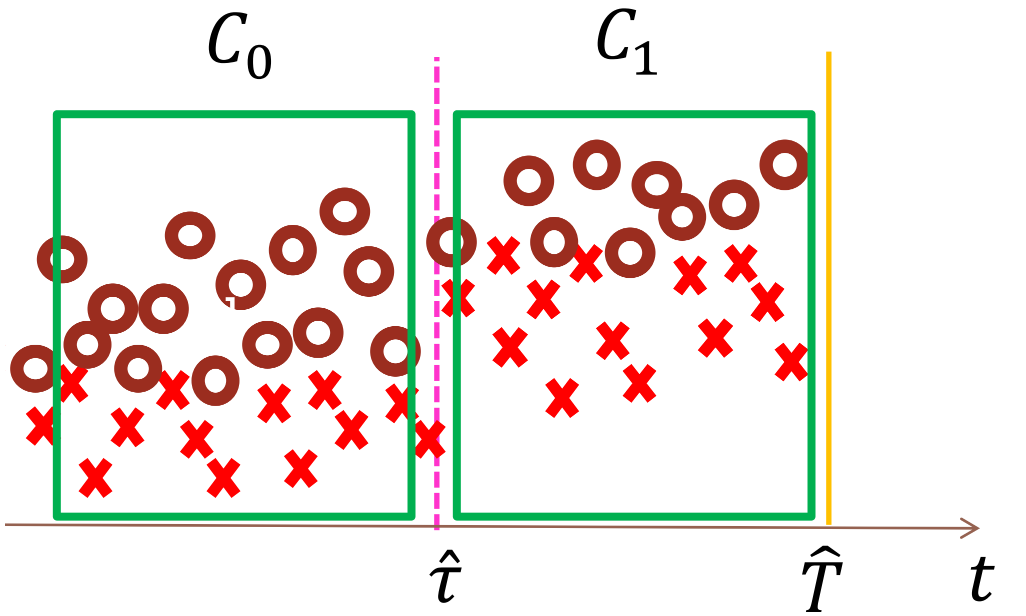
Uses statistical tools for performing an **offline** and **retrospective analysis** over the recent data, like:

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



JIT CLASSIFIERS: CONCEPT SPLIT

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



EXAMPLES OF CONCEPT SPLIT OPERATOR

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

- 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: 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- $\quad \mathcal{U}(C_i, \{x_t\}) \rightarrow C_i$;
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- $\quad\quad \boxed{Z_{\text{rec}} = \bigcup_{\substack{\mathcal{E}(C_i, C_j) = 1 \\ 0 \leq j < i}} Z_j;}$
$\quad$ **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 Equivalence

Look for concepts that are equivalent to the current one.

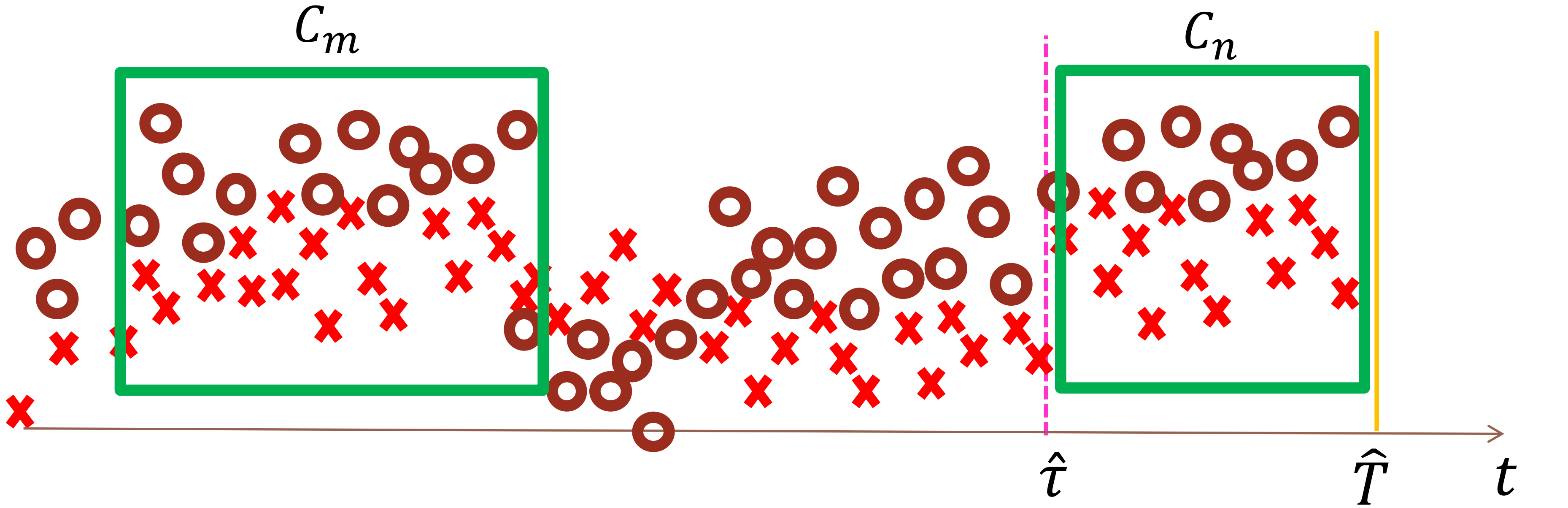
Gather supervised samples from all the representations C_j that refers to the same concept

JIT CLASSIFIERS: COMPARING CONCEPTS

Concept equivalence is assessed by

- comparing features F to determine whether $\phi(\boldsymbol{x})$ 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 $\phi(y|\boldsymbol{x})$ is the same

$$\mathcal{E}(C_m, C_n) = 1$$



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- $\quad \mathcal{U}(C_i, \{x_t\}) \rightarrow C_i$;
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- $\quad\quad \boxed{\widehat{y}_t = K(Z_i \cup Z_{\text{rec}}, x_t).}$
$\quad$ **end**
end

Label Prediction:

The classifier K is reconfigured using all the available supervised couples

COMPARING WINDOWS

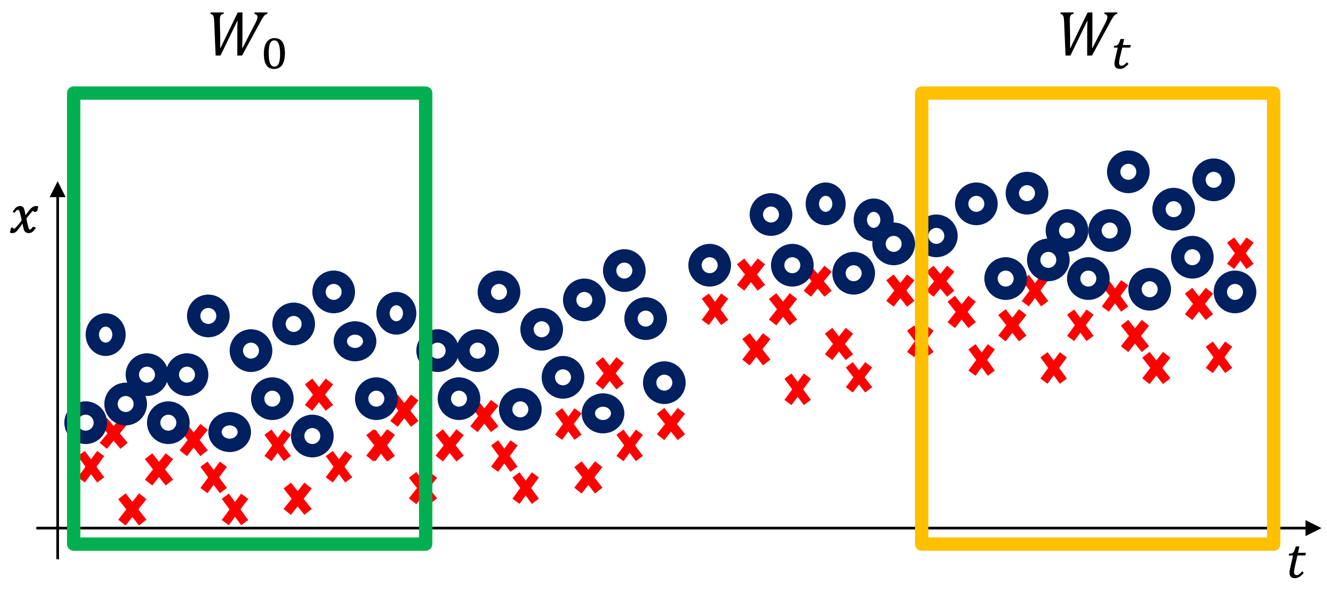
THE MOTIVATING IDEA

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



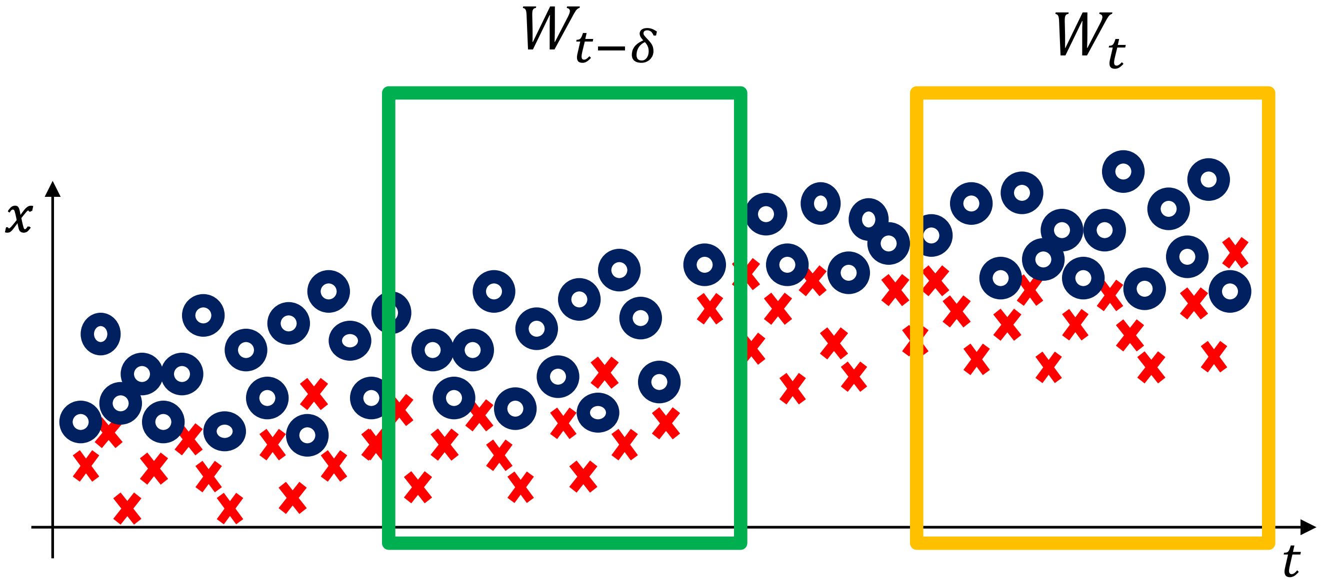
THE MOTIVATING IDEA

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 the data distribution on two different 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

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

WINDOW COMPARISON: 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

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
  - The density ratio over the two windows using kernel methods (to overcome curse of dimensionality problems when computing empirical distributions)

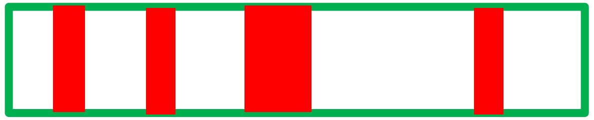
Kawahara, Y. and Sugiyama, M. "*Sequential change-point detection based on direct density-ratio estimation*". Statistical Analysis and Data Mining, 5(2):114–127, 2012.

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, on the most recent samples, $\boldsymbol{R_w}$ **correctly classifies** enough samples that S **misclassifies**, then, **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

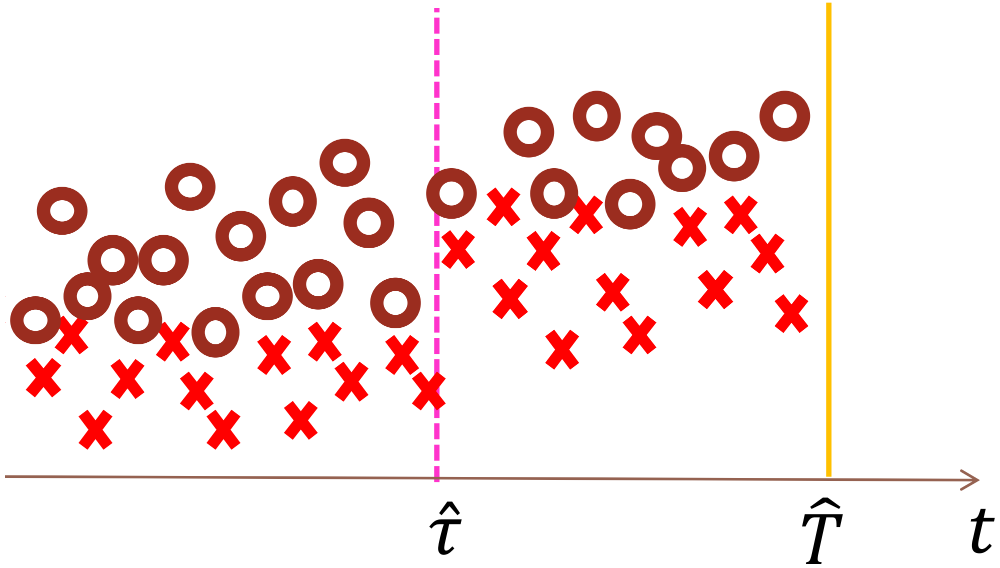
REMARKS ON ACTIVE APPROACHES

COMMENTS FROM MY PERSONAL EXPERIENCE

- Typically, when monitoring the classification error, false positives hurt less than detection delay
  - Things might change when classes are unbalanced

COMMENTS FROM MY PERSONAL EXPERIENCE

- Typically, when monitoring the classification error, false positives hurt less than detection delay
  - Things might change when classes are unbalanced
- 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 when classes are unbalanced
- Providing i.i.d. samples for reconfiguration seems more critical. When estimating the change-time:
  - Overestimating of τ provide too few samples
  - Underestimating 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 when classes are unbalanced
- Providing i.i.d. samples for reconfiguration seems more critical. When estimating the change-time:
  - Overestimating of τ provide too few samples
  - Underestimating of τ provide non i.i.d. data
  - Worth using accurate SPC methods like change-point methods (CPMs)
- Exploiting recurrent concepts is important
  - Providing additional samples could really make the difference
  - Mitigate the impact of false positives