Just-in-Time Classifiers For Recurrent Concepts

Giacomo Boracchi
Politecnico di Milano,
giacomo.boracchi@polim.it
September, 16th 2015
Universitè Libre de Bruxelles

Joint work with Cesare Alippi and Manuel Roveri

Presentation Outline

- Problem Statement
  - Drift Taxonomy
- Just In Time Classifiers at a Glance
  - Few more details
- Experiments
- Conclusions

PROBLEM FORMULATION

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 ($y_t \in \Lambda$)

Typically, one **assumes**

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

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

- a **training set** is provided

$$TR = \{(\boldsymbol{x}_0, y_0), \dots, (\boldsymbol{x}_n, y_n)\}$$

CLASSIFICATION OVER DATASTREAMS

The task: learn a classifier K to predict labels

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

in an **online manner** having a low **classification error**,

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

CLASSIFICATION OVER DATASTREAMS

The task: learn a classifier K to predict labels

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

in an **online manner** having a low **classification error**,

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

Unfortunately, datastreams $\mathcal{X}$ might change during operations. From time t onward

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

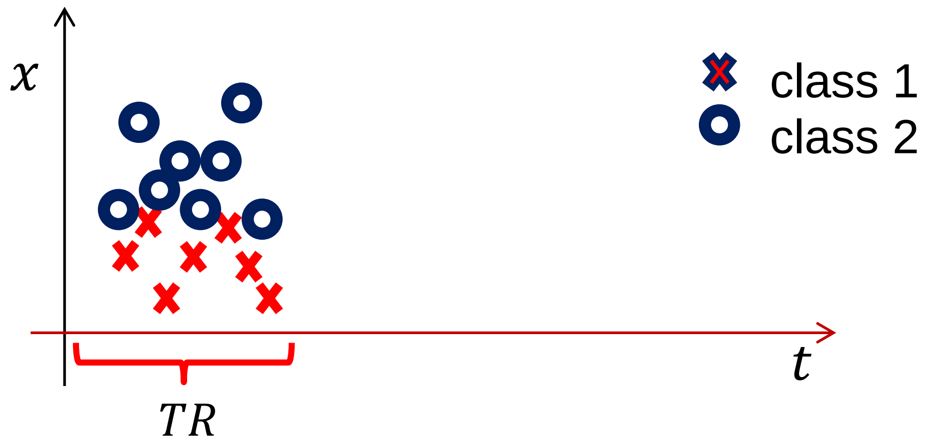
and $\mathcal{X}$ becomes **nonstationary** (undergoes a change) at t if

$$p_t(\boldsymbol{x}, y) \neq p_{\{t+1\}}(\boldsymbol{x}, y)$$

Changes in $\mathcal{X}$ are referred to as **concept drift**

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

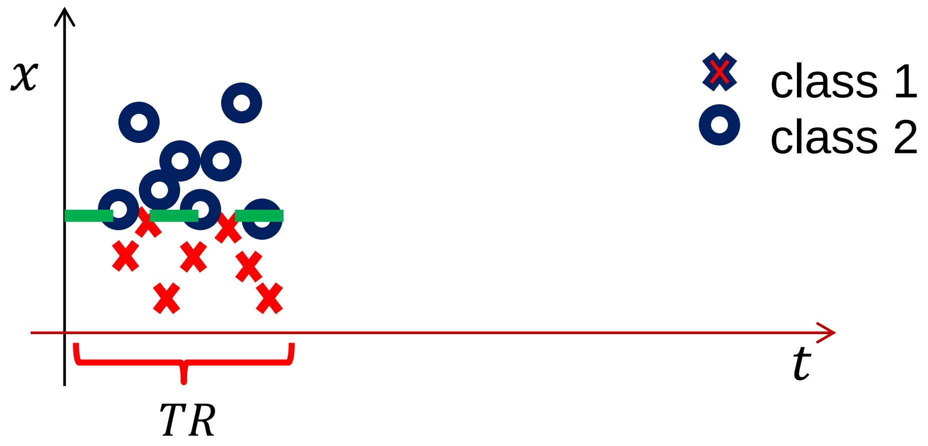
- The initial part of the stream is provided for training
- K is simply a threshold



CLASSIFICATION OVER DATASTREAMS

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

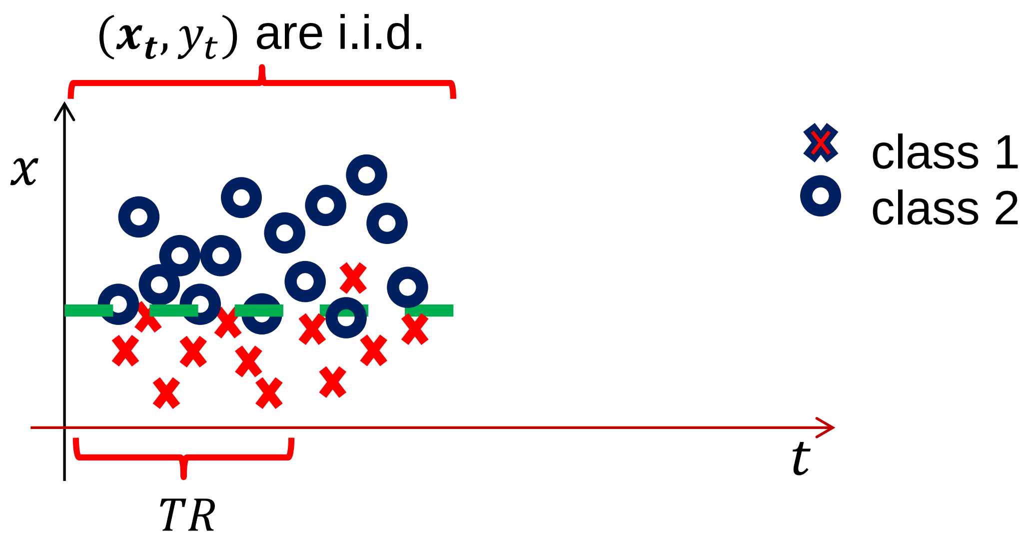
- The initial part of the stream is provided for training
- K is simply a threshold



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

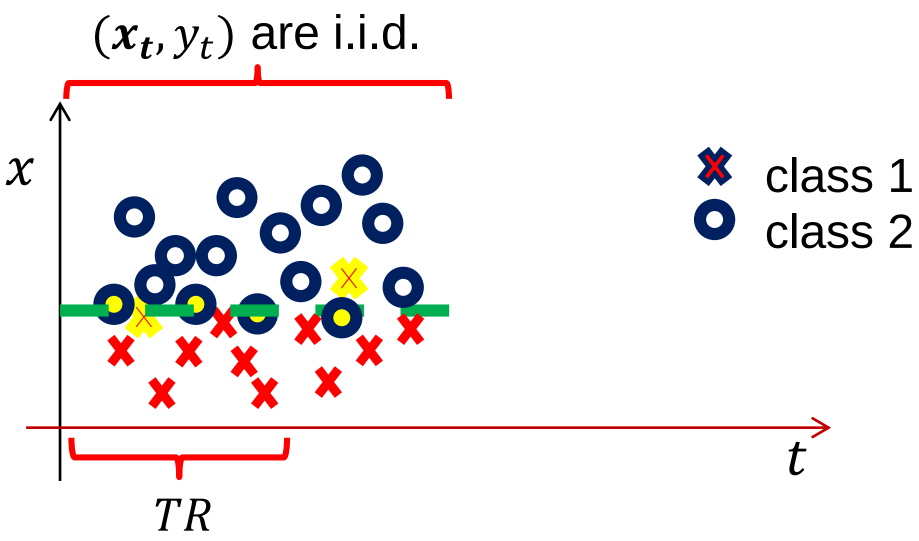


CLASSIFICATION OVER DATASTREAMS

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

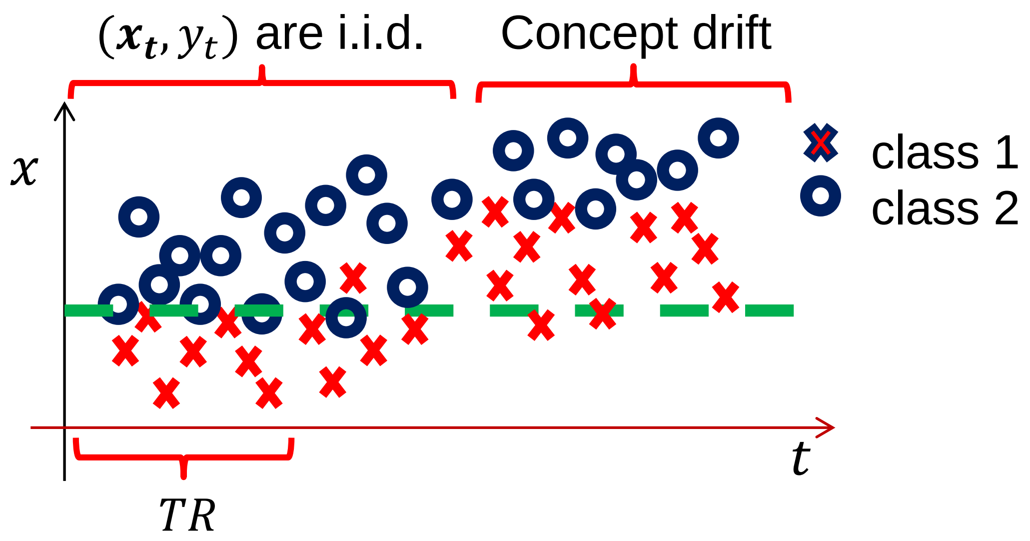
- The initial part of the stream is provided for training
- K is simply a threshold

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



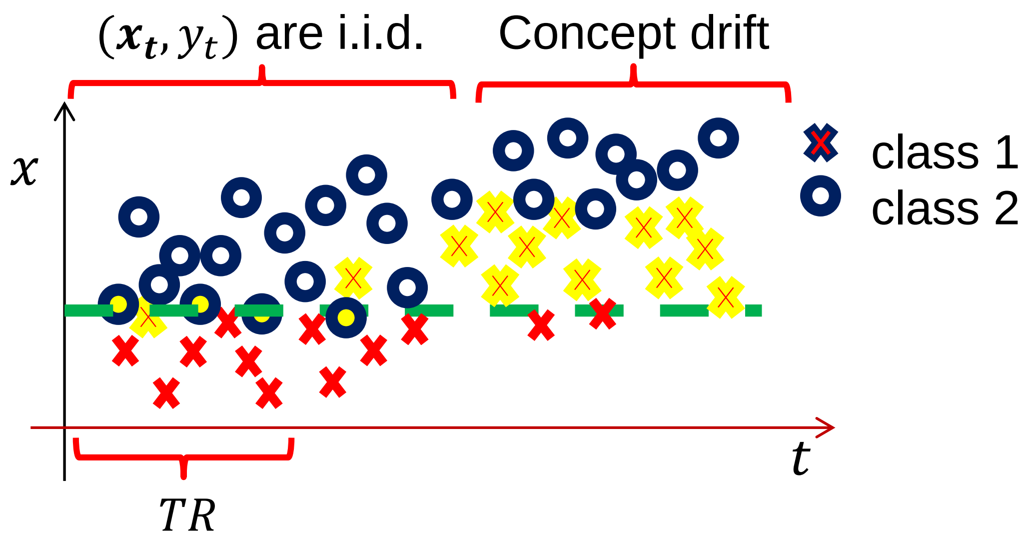
CLASSIFICATION OVER DATASTREAMS

Unfortunately, when concept drift occurs, and pdf p of $\mathcal{X}$ changes,

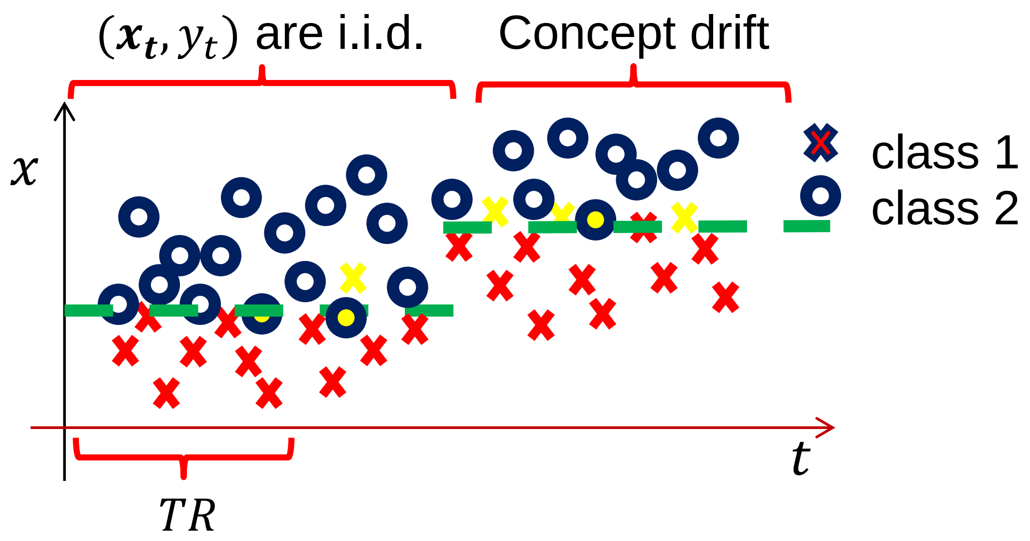


CLASSIFICATION OVER DATASTREAMS

Unfortunately, when concept drift occurs, and pdf p of $\mathcal{X}$ changes, things can be terribly worst.



Adaptation is needed to **preserve** classifier performance



SUPERVISED SAMPLES

We assume that **few supervised samples** are provided during **operations**.

Supervised samples enable the classifier to:

- **React to concept drift** to preserve its performance.
- **Increase its accuracy** in stationary conditions.

The classifier have to be **updated**, thus K becomes K_t

ADAPTATION STRATEGIES

ADAPTATION STRATEGIES

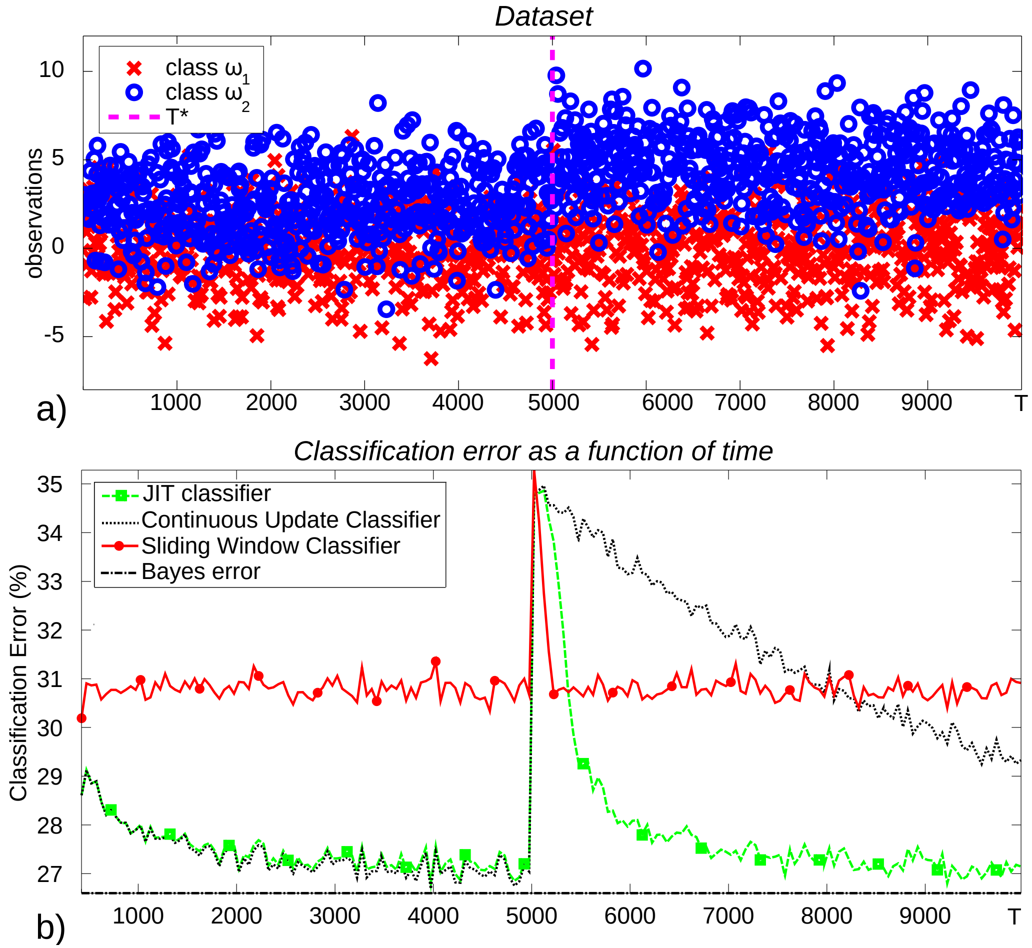
Consider two straightforward adaptation strategies

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

ADAPTATION STRATEGIES

Consider two straightforward adaptation strategies

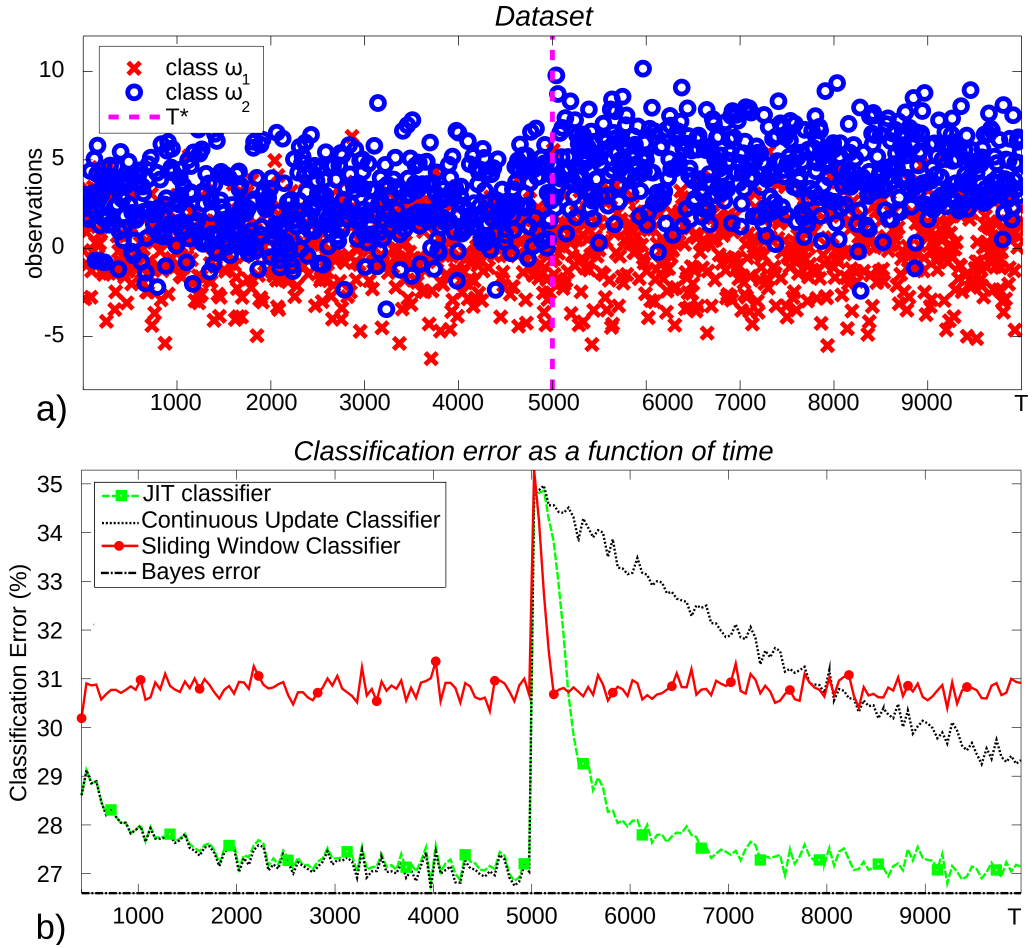
- Contiunously update K_t using all supervised couples
- Train K_t using only the last δ supervised couples



ADAPTATION STRATEGIES

Consider two straightforward adaptation strategies

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



Just including "fresh" training samples is not enough

Adaptation Strategies

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

DRIFT TAXONOMY

DRIFT TAXONOMY

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

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

- Drift might affect $p_t(y|\boldsymbol{x})$ and/or $p_t(\boldsymbol{x})$
  - Real
  - Virtual
- How does it changes over time?
  - Abrupt
  - Gradual
  - Recurring
  - .....

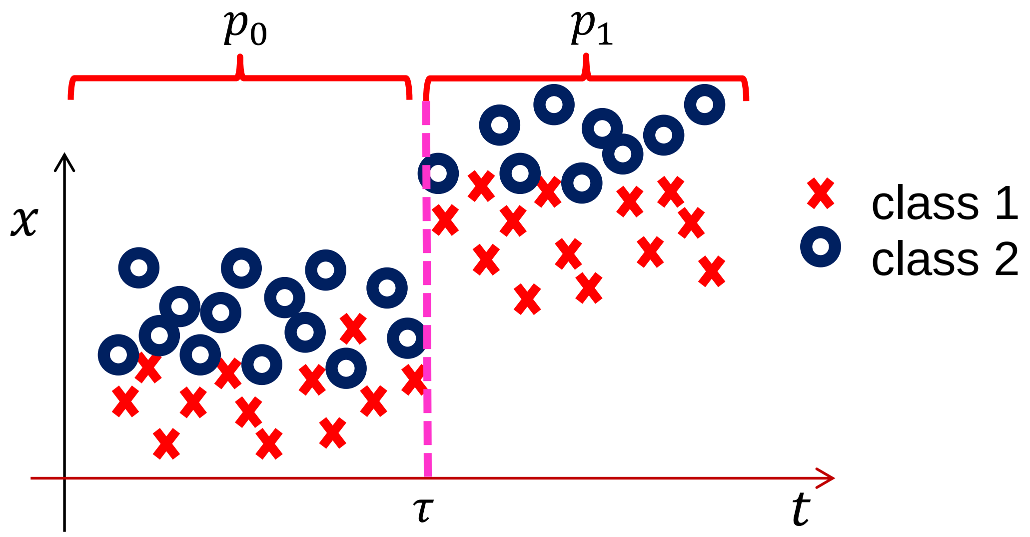
Drift taxonomy: What is changing?

Real Drift

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

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

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



Drift taxonomy: What is changing?

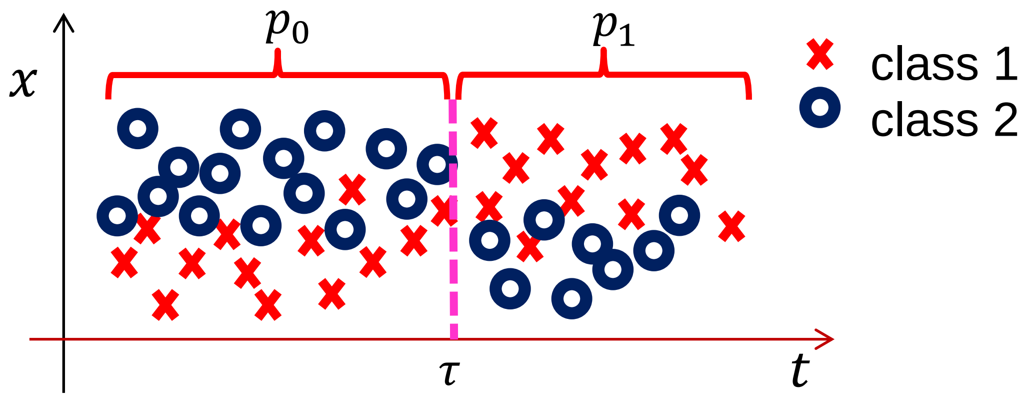
Real Drift

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

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

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

E.g. changes in the "class function", classes swap



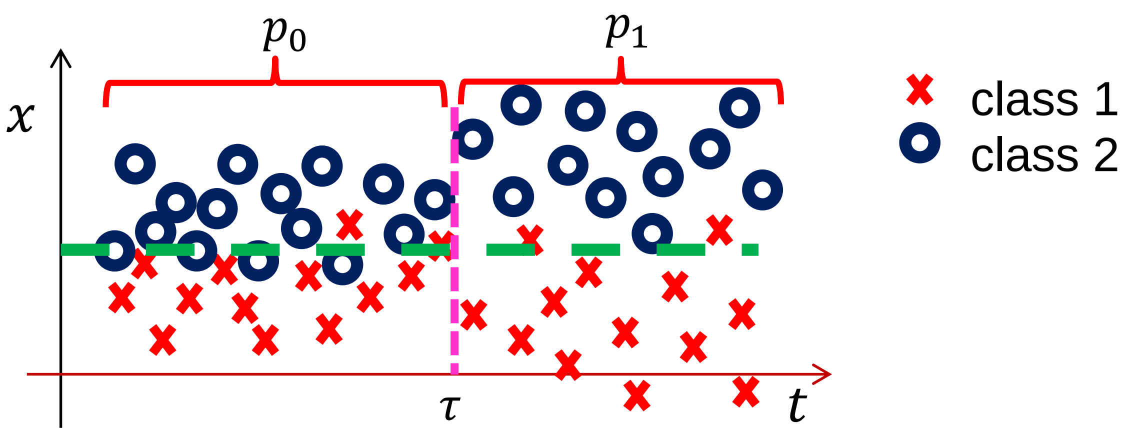
Drift taxonomy: What is changing?

Virtual Drift

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

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

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

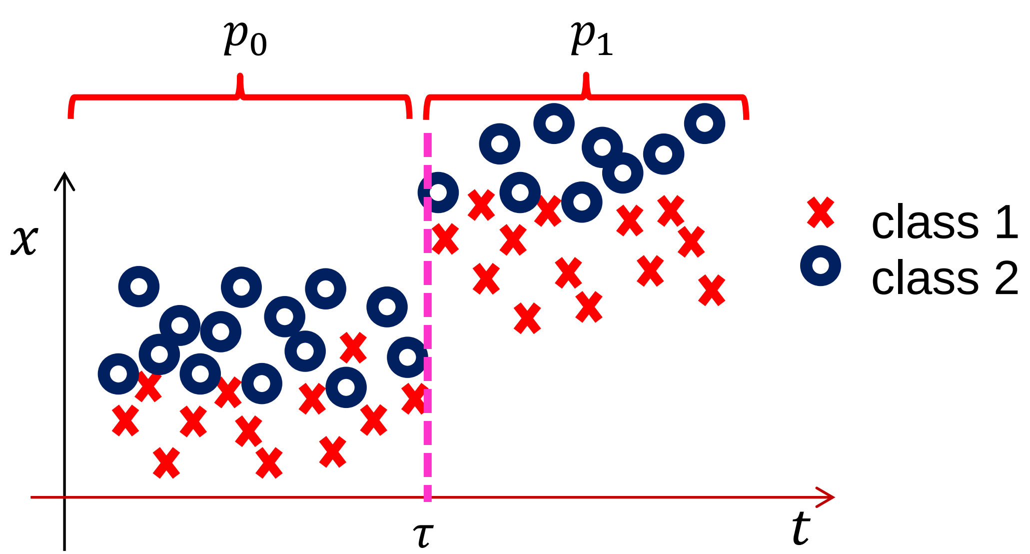


Drift taxonomy: time evolution

Abrupt

$$p_t(\boldsymbol{x}, y) = \begin{cases} p_0(\boldsymbol{x}, y) & t < \tau \\ p_1(\boldsymbol{x}, y) & t \geq \tau \end{cases}$$

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

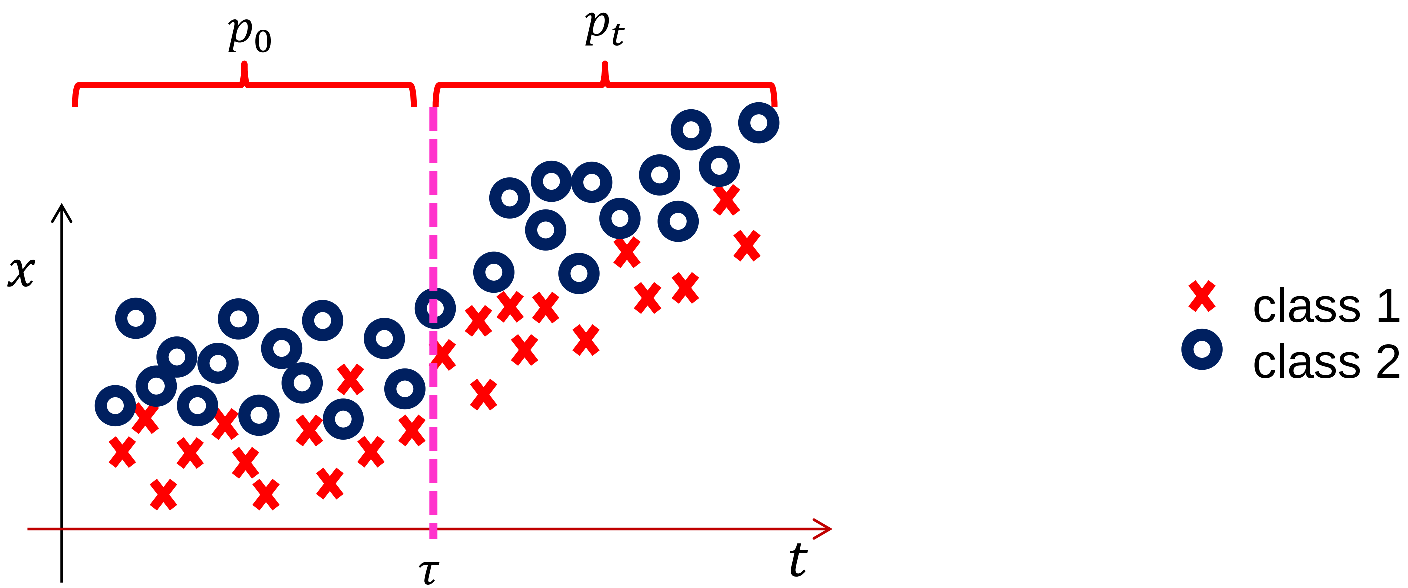


Drift taxonomy: time evolution

Gradual

$$p_t(\boldsymbol{x}, y) = \begin{cases} p_0(\boldsymbol{x}, y) & t < \tau \\ p_t(\boldsymbol{x}, y) & t \geq \tau \end{cases}$$

There is not a stationary state of $\mathcal{X}$ after the change

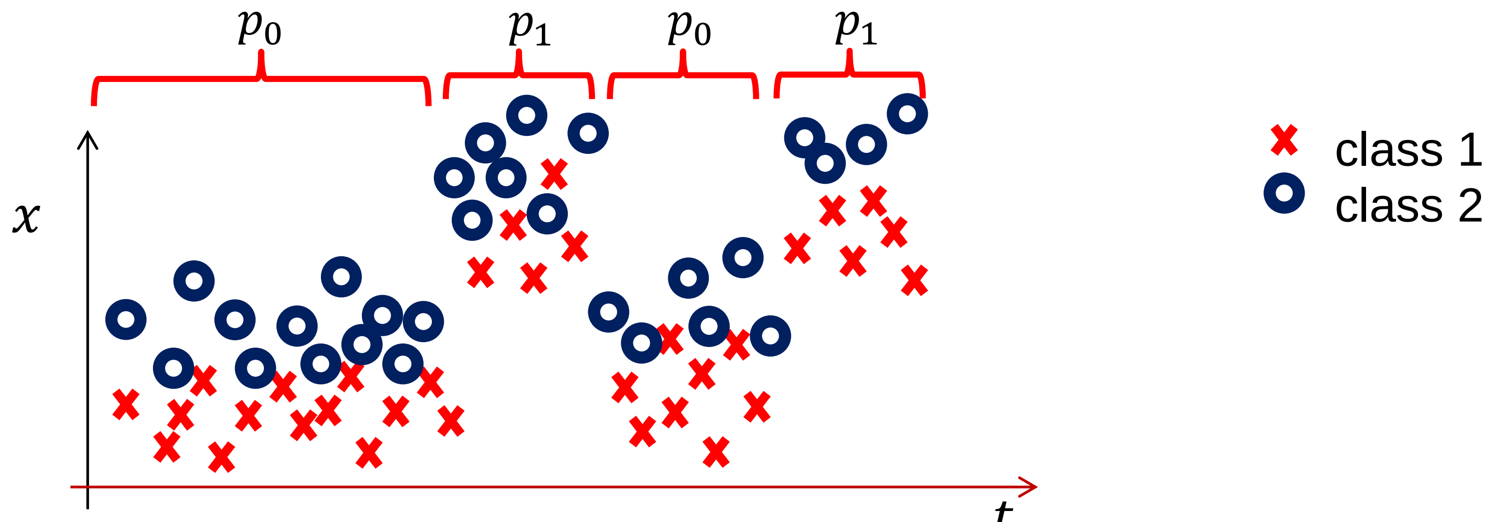


Drift taxonomy: time evolution

Recurring

$$p_t(\boldsymbol{x}, y) = \begin{cases} p_0(\boldsymbol{x}, y) & t < \tau \\ p_1(\boldsymbol{x}, y) & t \geq \tau \\ \quad\ldots & \\ \quad p_1(\boldsymbol{x}, y) & \end{cases}$$

After τ, another concept drift might bring back $\mathcal{X}$ in p_0



What we address here

We present a framework to design adaptive classifiers able to operate on concept drifts that are

- abrupt
- possibly recurrent
- real
- virtual

JUST-IN-TIME CLASSIFIERS

A methodology for designing adaptive classifiers

JIT Classifiers: the Algorithm

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- $\quad \mathcal{U}(C_i, \{x_t\}) \rightarrow C_i$;

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

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- $\quad \boxed{\mathcal{U}}(C_i, \{x_t\}) \rightarrow C_i$;
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** $(\boxed{\mathcal{D}}(C_i) = 1)$ **then**
8- $\quad\quad i = i + 1$;
9- $\quad\quad \boxed{\Upsilon}(C_{i-1}) \rightarrow (C_k, C_l)$;
10- $\quad\quad C_i = C_l$;
11- $\quad\quad C_{i-1} = C_k$;
12- $\quad\quad Z_{\text{rec}} = \bigcup_{\substack{\boxed{\mathcal{E}}(C_i, C_j) = 1 \\ 0 \leq j < i}} Z_j$;
$\quad$ **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- $\quad \mathcal{U}(C_i, \{x_t\}) \rightarrow C_i$;

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

JIT classifiers can be built upon specific classifier (like svm, decision trees, naive Bayes, knn, etc..)

JIT Classifiers: the Algorithm

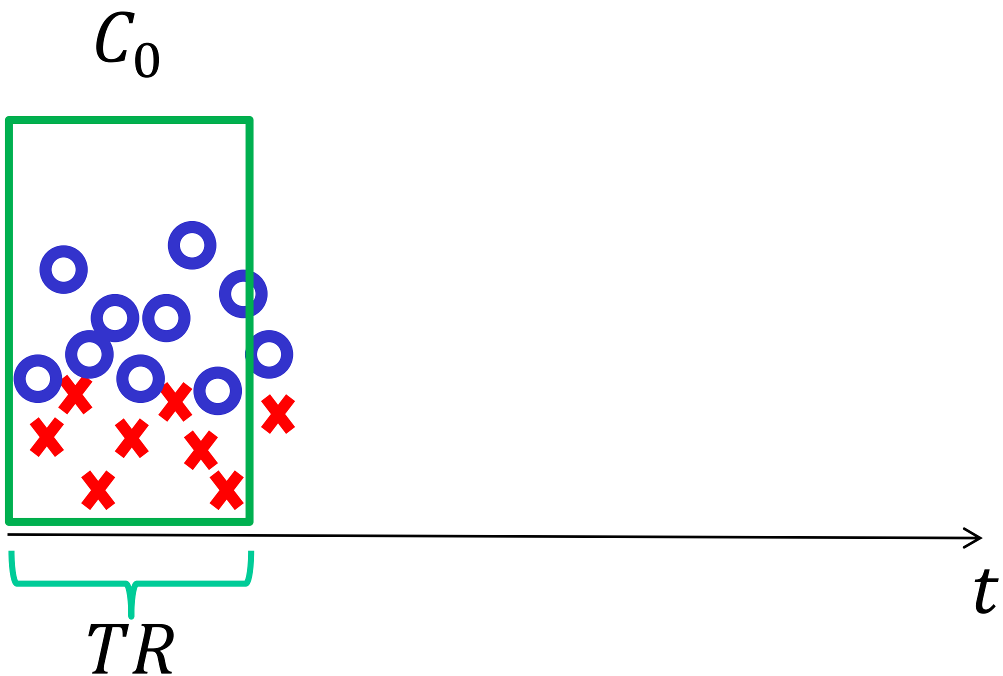
Build concept $C_0 = (Z_0, F_0, D_0)$ from the training sequence;
$Z_{\text{rec}} = \emptyset$ and $i = 0$;
while *(x_t is available)* **do**
  $\mathcal{U}(C_i, \{x_t\}) \rightarrow C_i$;
  if *(y_t is available)* **then**
    $\mathcal{U}(C_i, \{(x_t, y_t)\}) \rightarrow C_i$;
  end
  if *($\mathcal{D}(C_i)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

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

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

During operations, each input sample is analyzed to

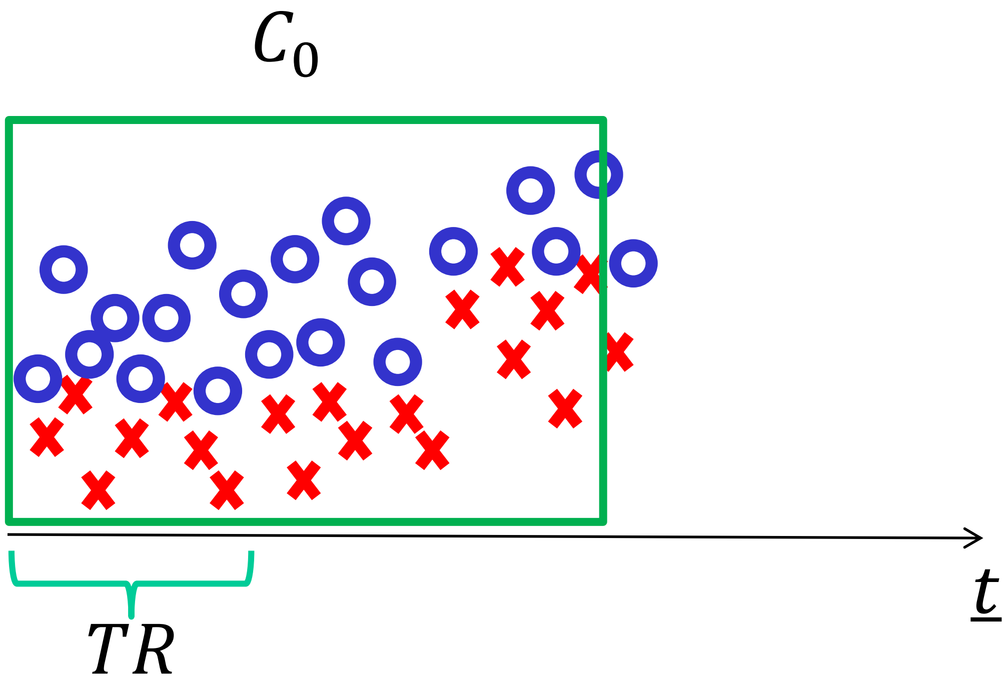
- Extract features that are appended to F_i
- Append supervised information in Z_i

thus updating the current concept representation

JIT Classifiers: Concepts 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

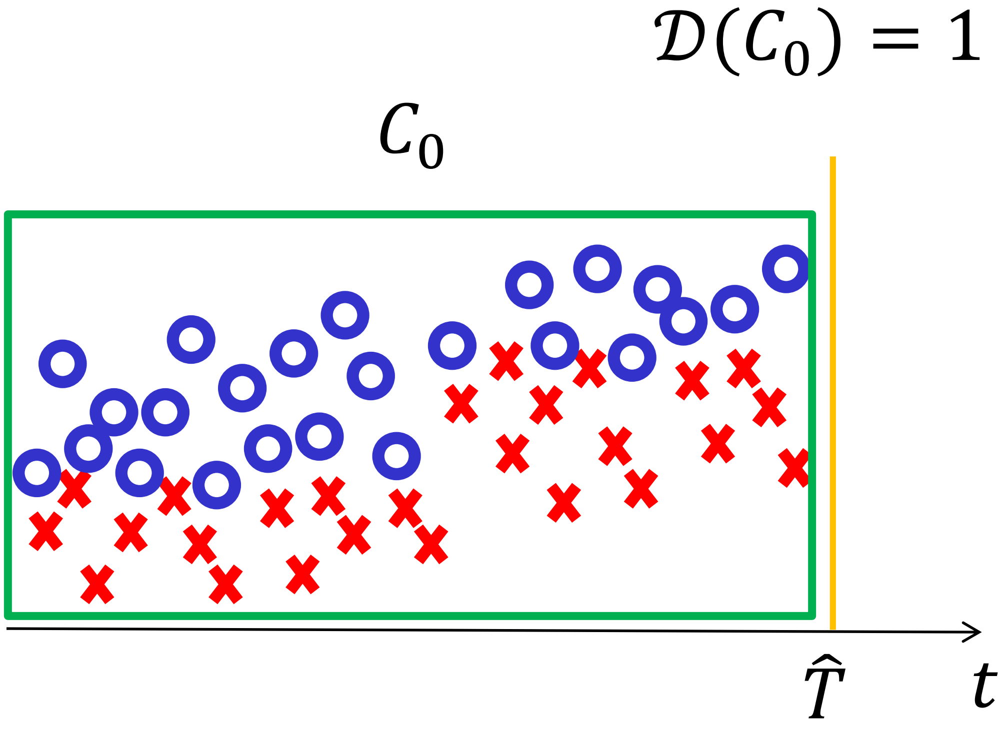
1- Build concept $C_0 = (Z_0, F_0, D_0)$ from the training sequence;
2- $Z_{\text{rec}} = \emptyset$ and $i = 0$;
3- **while** *(x_t is available)* **do**
4- $\quad \mathcal{U}(C_i, \{x_t\}) \rightarrow C_i$;
5- $\quad$ **if** *(y_t is available)* **then**
6- $\quad\quad \mathcal{U}(C_i, \{(x_t, y_t)\}) \rightarrow C_i$;
$\quad$ **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- $\quad\quad Z_{\text{rec}} = \bigcup_{\substack{\mathcal{E}(C_i, C_j) = 1 \\ 0 \le j < i}} Z_j$;
$\quad$ **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

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

JIT Classifier: Drift Detection

$\mathcal{D}$ monitoring the datastream by means of **online** and **sequential** change-detection tests (CDTs)

- Changes are detected monitoring $p(y|\boldsymbol{x})$ and $p(\boldsymbol{x})$



JIT Classifiers: the Algorithm

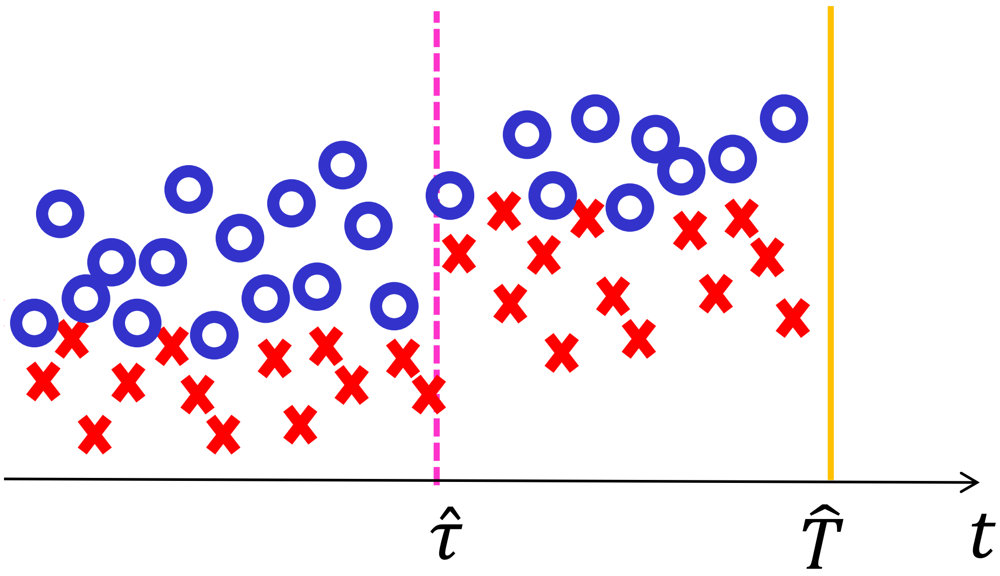
Build concept $C_0 = \left(Z_0, F_0, D_0\right)$ from the training sequence;
$Z_{\text{rec}} = \emptyset$ and $i = 0$;
while *(x_t is available)* **do**
  $\mathcal{U}(C_i, \{x_t\}) \rightarrow C_i$;
  if *(y_t is available)* **then**
    $\mathcal{U}(C_i, \{(x_t, y_t)\}) \rightarrow C_i$;
  end
  if *($\mathcal{D}(C_i) = 1$)* **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

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 Splits

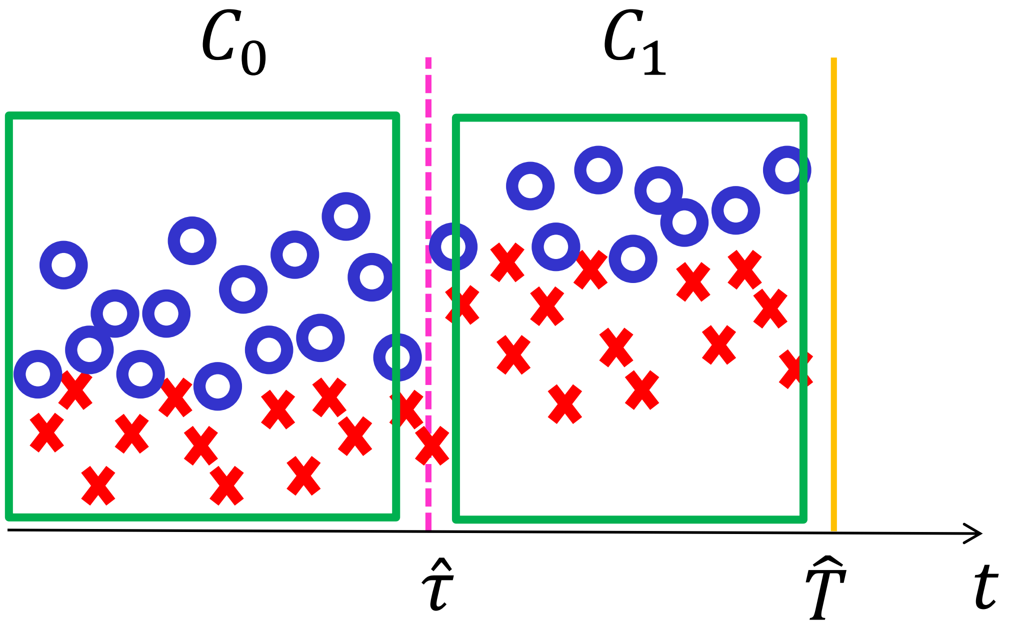
Offline and **retrospective** statistical tools such as hypothesis tests (HT) or change-point methods (CPM) can be used to estimate the change point.



JIT Classifiers: Concept Splits

Two concept descriptions are constructed

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



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

Look for concepts that are equivalent to the current one.

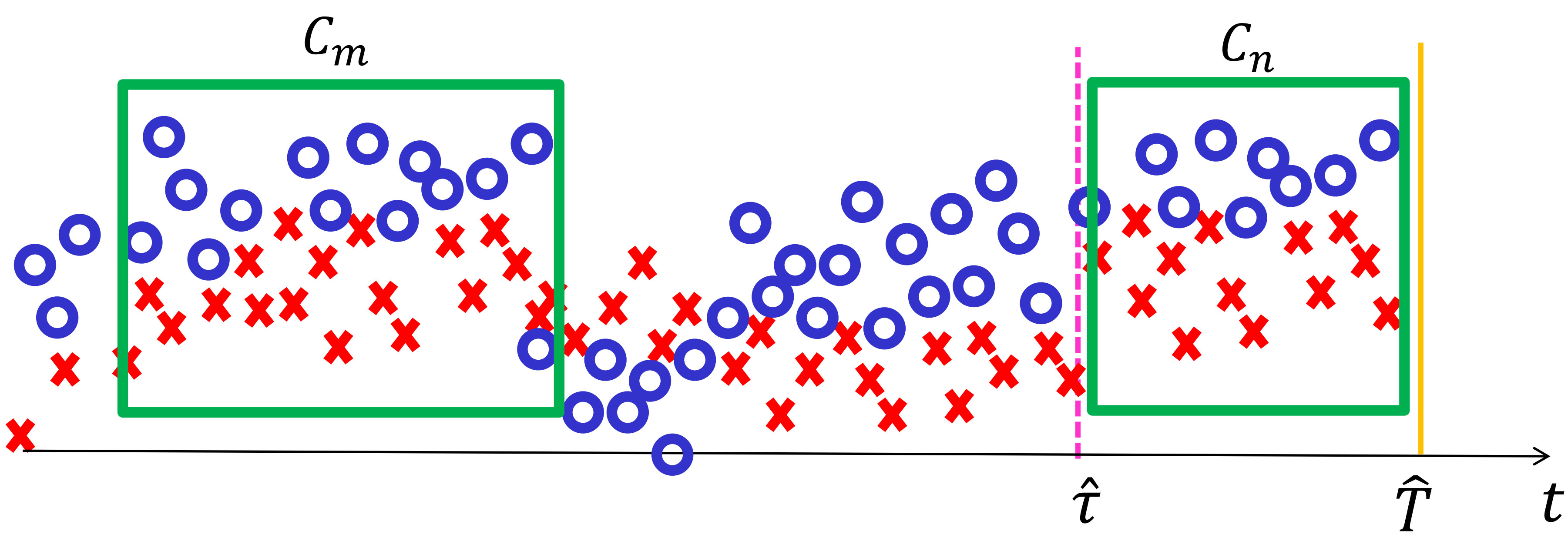
Gather supervised samples from all the representations C_j that refers to the same concept

JIT Classifiers: Comparing Concepts

Concept equivalence is assessed by

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

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



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

JUST-IN-TIME CLASSIFIERS

Few more details about a specific example

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 **nonoverlapping 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 $\widehat{err}$

  extracted from **nonoverlapping sequences**

Update Operator

Update operator

$$\mathcal{U}(C_i, \{(\boldsymbol{x_0}, y_0)\}) = C_i$$

insert the **supervised couple** $(\boldsymbol{x_0}, y_0)$ in Z_i and

$$\mathcal{U}(C_i, \{\boldsymbol{x_0}, \dots, \boldsymbol{x_n}\}) = C_i$$

Takes a sequence of unsupervised data as input, **extracts features** values and **appends** them to F_i

Concept Drift 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**

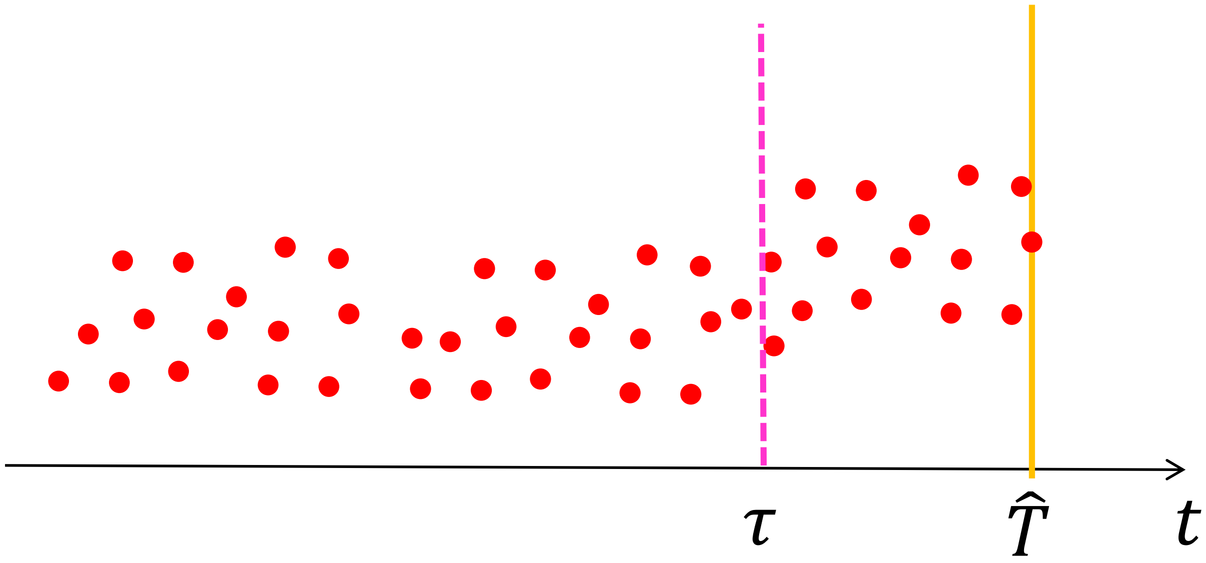
[1] A. Goldenshluger and A. Nemirovski, "On spatial adaptive estimation of nonparametric regression," Math. Meth. Statistics, vol. 6, pp. 135–170,1997.

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

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



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
- ICI-based CDTs implement a refinement procedure to stimate τ after having detected a change at $\hat{T}$.
- Change-Point Methods implement the following Hypothesis test on the feature sequence:

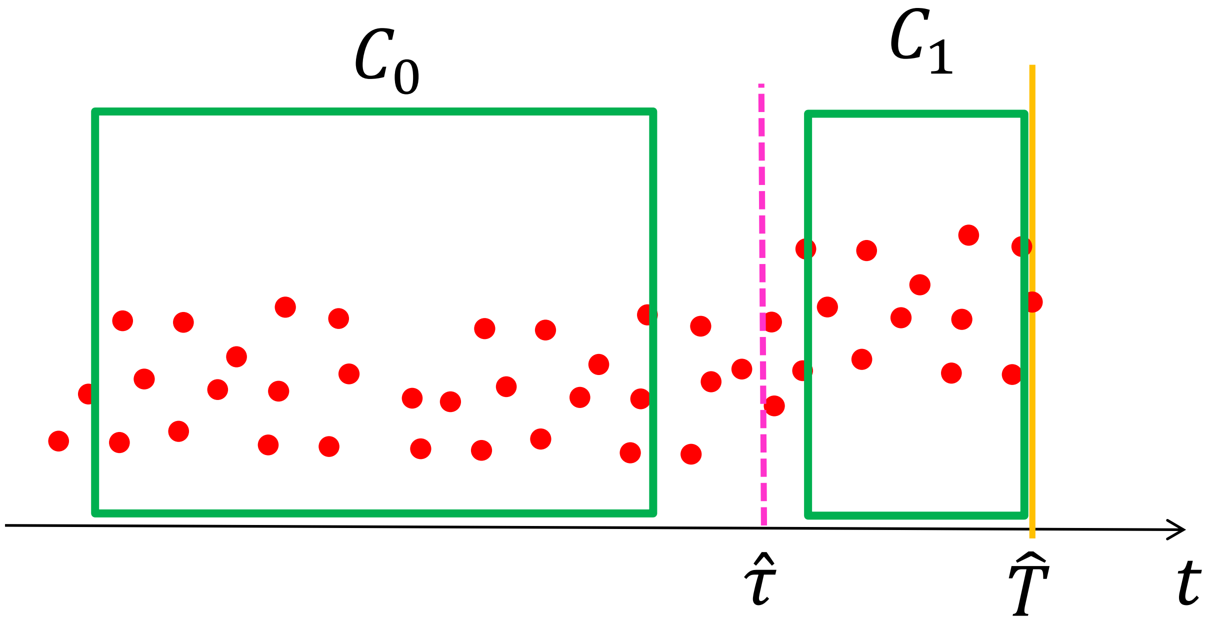
$$\begin{cases} H_0: \text{"}F_i \text{ contains i.i.d. samples"} \\ H_1: \text{"}F_i \text{ contains a change point"} \end{cases}$$

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

Split Operator

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

- In both cases, it is convenient to exclude data close to the estimated change point $\hat{\tau}$, implementing some heuristic



Equivalence Operator

$$\mathcal{E}(C_0, C_1) \in \{0,1\}$$

- **Determines if** C_0 and C_1 **refer** to the **same concept**
  - Performs an **equivalence testing** problem to determine whether F_0 and F_1 refer to the same $p(\boldsymbol{x})$
  - **Compares classifiers** trained on Z_0 and Z_1 on the same validation set to determine if $p(y|\boldsymbol{x})$ was the same
- Recurrent concepts are identified by performing a **pair-wise comparison** against the previously encountered concepts

EXPERIMENTS

Considered Classifiers

We considered the following adaptive classifiers:

- JIT for recurrent concepts
- JIT without recurrent concepts handling
- W: a **sliding window** classifier
- E: a two-individuals **ensemble** which pairs JIT and W
- U: a classifier trained on **all** the available **data**

that have been tested on KNN, and Naive Bayes Classifiers

Considered Classifiers

We considered the following adaptive classifiers:

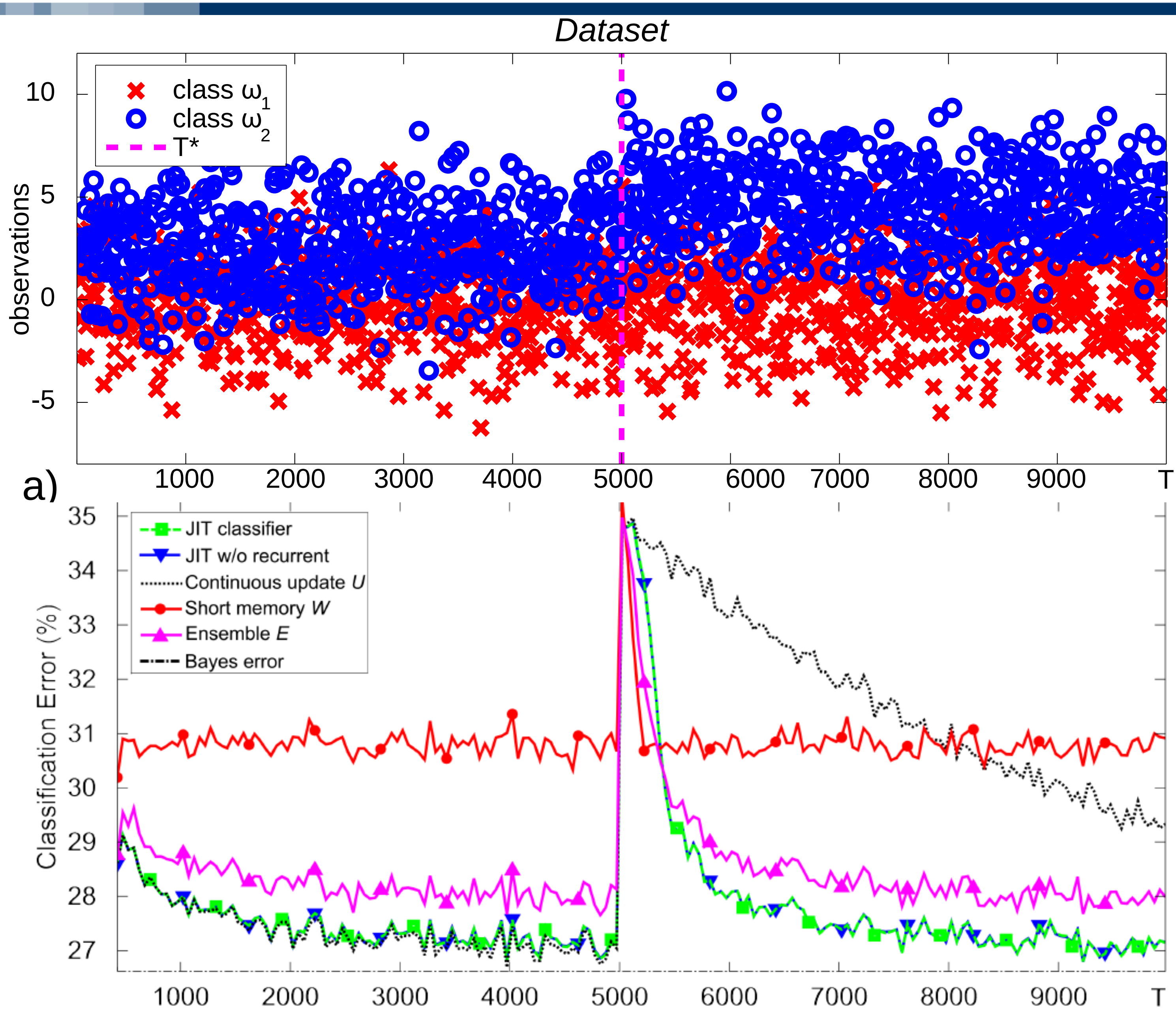
- JIT for recurrent concepts
- JIT without recurrent concepts handling
- W: a **sliding window** classifier
- E: a two-individuals **ensemble** which pairs JIT and W
- U: a classifier trained on **all** the available **data**

that have been tested on KNN, and Naive Bayes Classifiers

In the ensemble E, the output is defined by **selecting** the most accurate classifier over the last 20 samples (like in paired learners)

The **ensemble** is meant to **improve reaction promptness** to concept drift. In stationary conditions JIT outperforms E

The Ensemble E



Synthetic Datasets

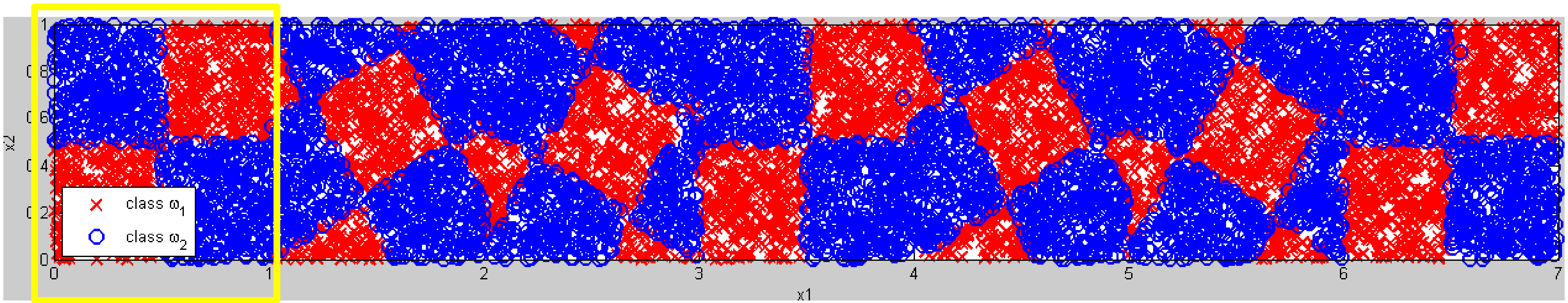
Checkerboard: sequences are composed of

- 10000 samples uniformly distributed in $[0, 1] \times [0,1]$
- Classification function is a checkerboard of side 0.5
- Concept drift affects classification function by rotating the checkerboard every 2000 samples.
- One sample every 5 is supervised

Synthetic Datasets

Checkerboard: sequences are composed of

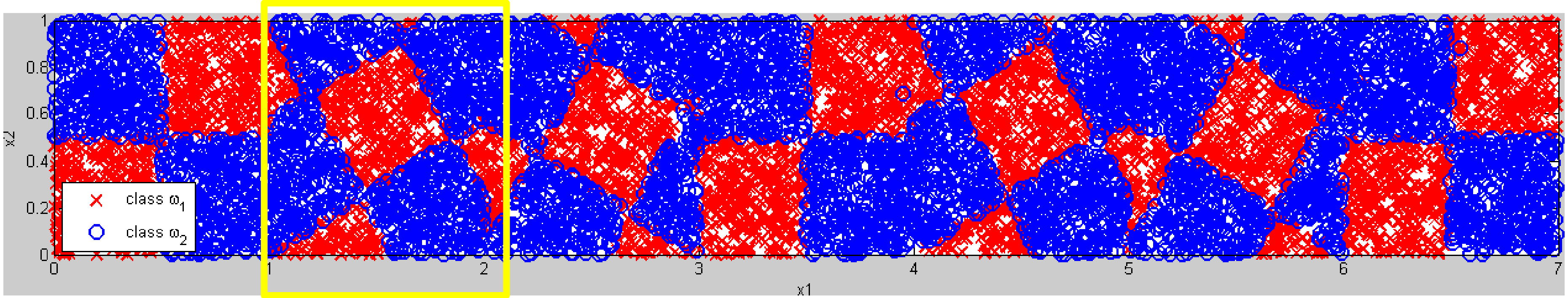
- 10000 samples uniformly distributed in $[0, 1] \times [0,1]$
- Classification function is a checkerboard of side 0.5
- Concept drift affects classification function by rotating the checkerboard every 2000 samples.
- One sample every 5 is supervised



Synthetic Datasets

Checkerboard: sequences are composed of

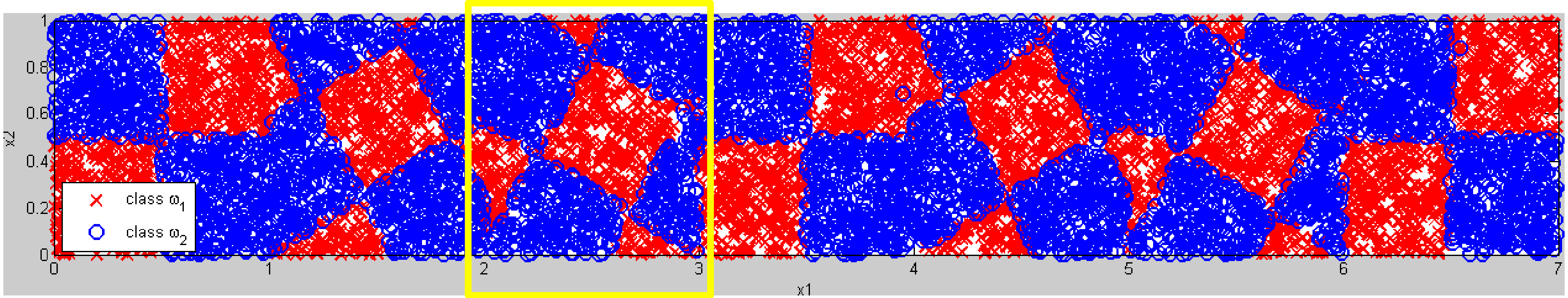
- 10000 samples uniformly distributed in $[0, 1] \times [0,1]$
- Classification function is a checkerboard of side 0.5
- Concept drift affects classification function by rotating the checkerboard every 2000 samples.
- One sample every 5 is supervised



Synthetic Datasets

Checkerboard: sequences are composed of

- 10000 samples uniformly distributed in $[0, 1] \times [0,1]$
- Classification function is a checkerboard of side 0.5
- Concept drift affects classification function by rotating the checkerboard every 2000 samples.
- One sample every 5 is supervised



R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," Neural Networks, IEEE Transactions on, vol. 22, no. 10, pp. 1517 –1531, oct. 2011

Synthetic Datasets

Checkerboard: sequences are composed of

- 10000 samples uniformly distributed in $[0, 1] \times [0,1]$
- Classification function is a checkerboard of side 0.5
- Concept drift affects classification function by rotating the checkerboard every 2000 samples.
- One sample every 5 is supervised

Sine:

- Similar to CB, class function is a sine
- Tested introducing irrelevant components and class noise

W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, ser. KDD '01

Figures of Merit

- Classification error averaged over 2000 runs
- Precision and Recall for the identification of recurrent concept (JIT classifier only)

$$precision = \frac{tp}{tp+fp} \text{ and } recall = \frac{tp}{tp+fn}$$

Experiment	Base classifier	JIT	Ensemble	JIT w/o recurrent	Short Memory (W)	Continuous Update (U)	Precision Recurrent	Recall Recurrent
CHECKERBOARD_1	k-NN	21.45	**17.06**	21.41	21.77	44.58	0.422	0.724
CHECKERBOARD_2	k-NN	19.92	**14.32**	20.37	18.93	24.48	1	0.799
CHECKERBOARD_3	k-NN	18.60	**15.60**	18.83	20.48	25.67	0.977	0.833
MULTIVARIATE GAUSSIAN	k-NN	23.60	**21.74**	23.61	25.00	47.85	1	0.947
	NB	21.52	**19.97**	21.52	21.08	49.03	1	1
SINE_2	k-NN	14.33	**11.09**	15.50	15.59	44.07	1	0.987
SINE_2A	k-NN	19.49	**12.80**	20.55	18.10	44.43	1	0.932
SINE_IRREL_2	k-NN	23.76	**18.37**	24.79	24.19	45.49	1	0.793
SINE_IRREL_2A	k-NN	31.23	**22.05**	31.64	27.33	45.83	1	0.415
EMAIL_LIST	k-NN	42.00	**36.65**	42.00	36.55	37.03	-	0
	SVM	22.34	**17.31**	22.90	22.62	42.83	1	0.250

Figures of Merit

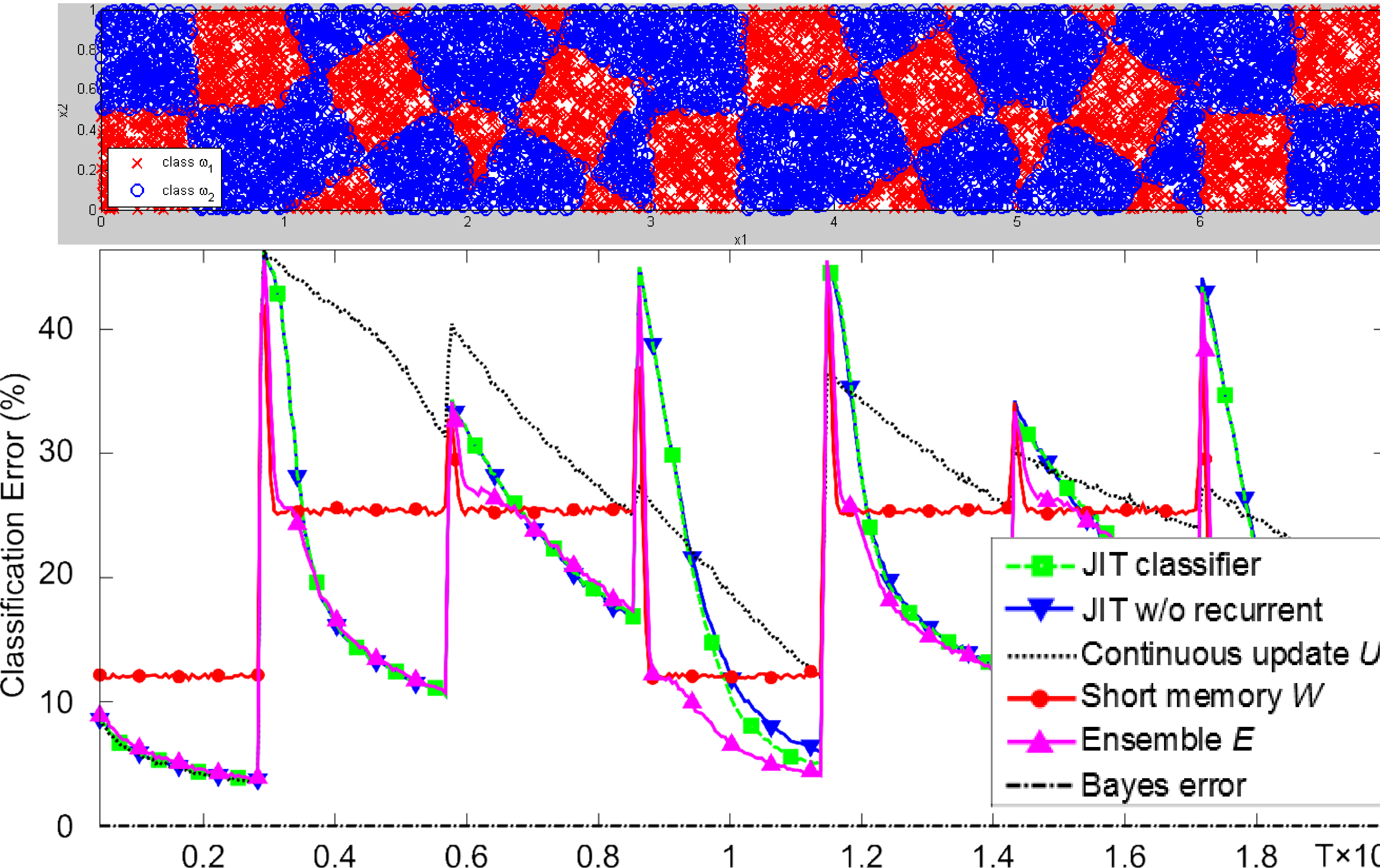
- **Classification error** averaged over 2000 runs
- **Precision** and **Recall** for the identification of recurrent concept (JIT classifier only)

$$precision = \frac{tp}{tp+fp} \text{ and } recall = \frac{tp}{tp+fn}$$

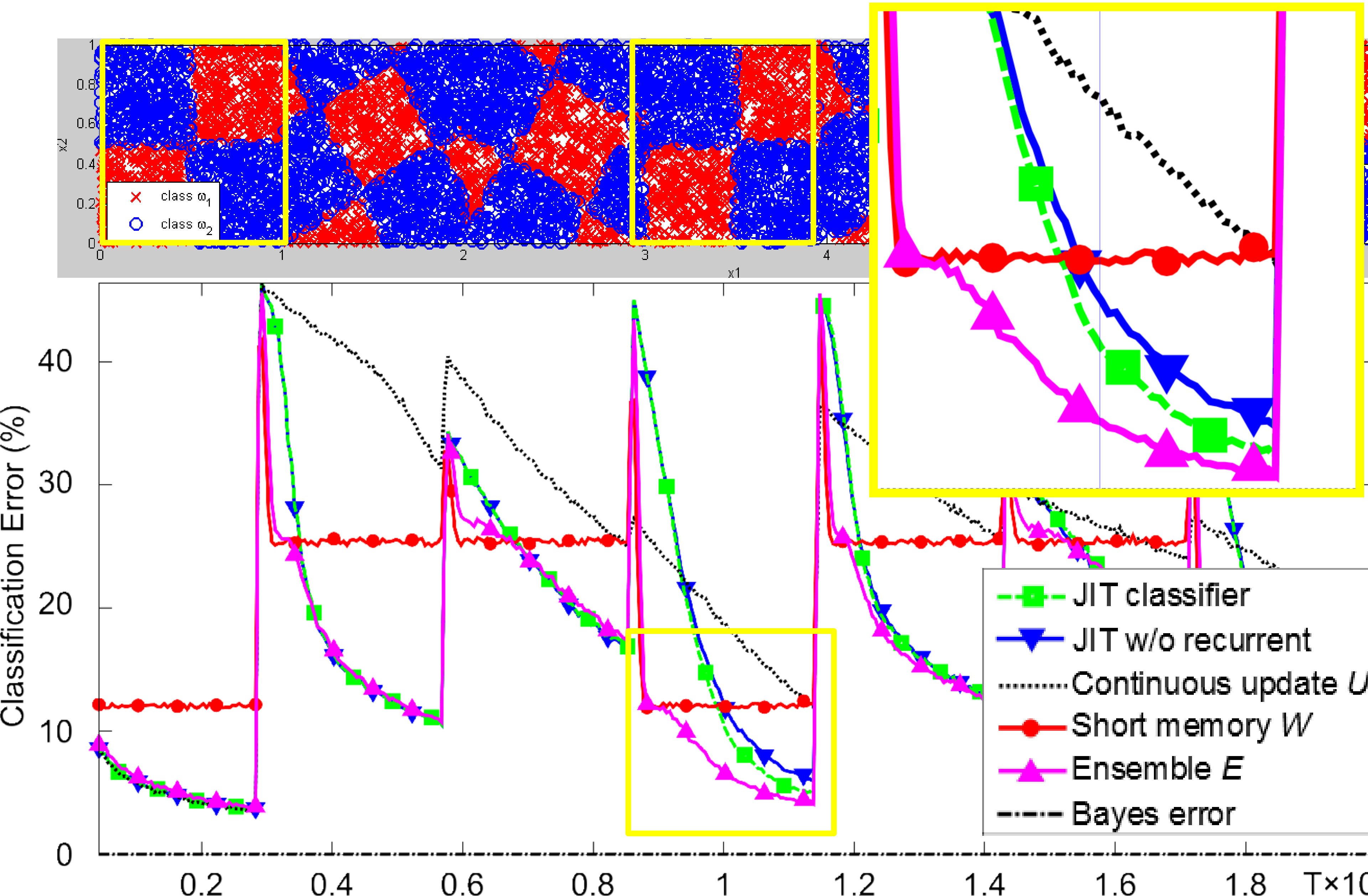
CHECKERBOARD_1 dataset does not contain recurrent concepts. Equivalence operator can correctly associate concepts that have been split by FP of $\mathcal{D}$

Experiment	Base classifier	JIT	Ensemble	JIT w/o recurrent	Short Memory (W)	Continuous Update (U)	Precision Recurrent	Recall Recurrent
CHECKERBOARD_1	k-NN	21.45	**17.06**	21.41	21.77	44.58	0.422	0.724
CHECKERBOARD_2	k-NN	19.92	**14.32**	20.37	18.93	24.48	1	0.799
CHECKERBOARD_3	k-NN	18.60	**15.60**	18.83	20.48	25.67	0.977	0.833
MULTIVARIATE	k-NN	23.60	**21.74**	23.61	25.00	47.85	1	0.947
GAUSSIAN	NB	21.52	**19.97**	21.52	21.08	49.03	1	1
SINE_2	k-NN	14.33	**11.09**	15.50	15.59	44.07	1	0.987
SINE_2A	k-NN	19.49	**12.80**	20.55	18.10	44.43	1	0.932
SINE_IRREL_2	k-NN	23.76	**18.37**	24.79	24.19	45.49	1	0.793
SINE_IRREL_2A	k-NN	31.23	**22.05**	31.64	27.33	45.83	1	0.415
EMAIL_LIST	k-NN	42.00	**36.65**	42.00	36.55	37.03	-	0
	SVM	22.34	**17.31**	22.90	22.62	42.83	1	0.250

Exploiting Recurrent Concepts

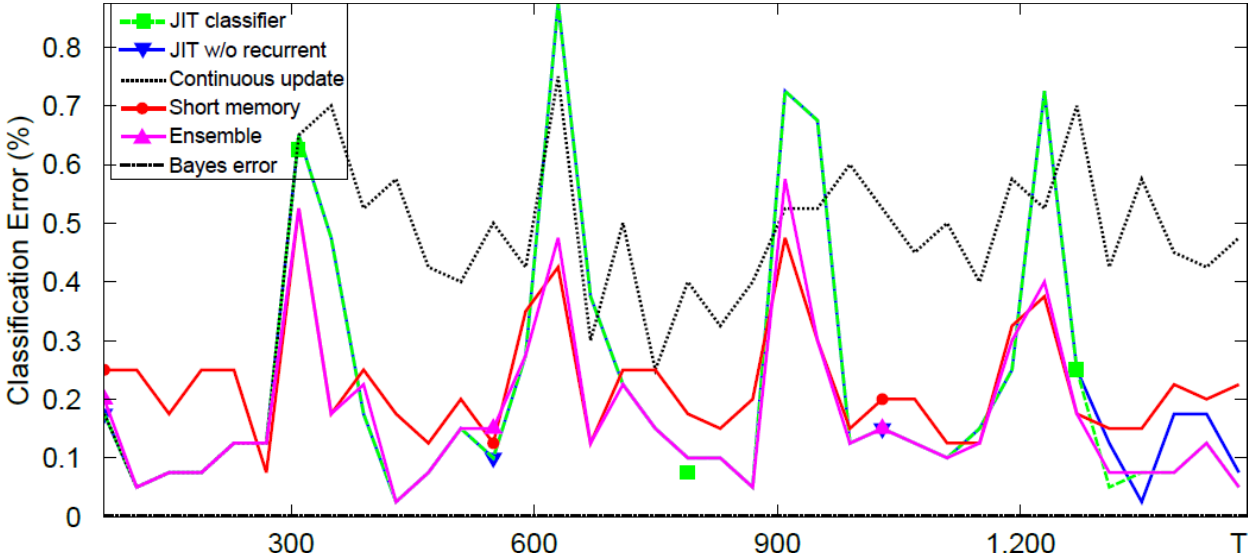


Exploiting Recurrent Concepts



Spam Email Dataset

- Inputs x are email text in the bag-of-words representation (913 Boolean attributes)
- Each email refers to a specific topic. Some topics are considered of interest, the remaining are considered spam
- Concept drift is introduced every 300 emails by swapping spam/ham labels, simulating a change in user interests



I. Katakis, G. Tsoumakas, and I. Vlahavas, "Tracking recurring contexts using ensemble classifiers: an application to email filtering," Knowl. Inf. Syst., vol. 22, no. 3, pp. 371–391, Mar. 2010

CONCLUDING REMARKS

Conclusions

- We proposed a **general methodology** for designing different JIT Classifiers based on different
  - concept representations
  - techniques to detect concept drift, split concept representations and assess concept equivalence
  - base classifiers
- Concept representations have to be *condensed* for the JIT classifiers to be efficient in the real-world
  - Pruning / down sampling Z, F, D
  - Learn models describing data distributions in Z, F, D

  not investigated yet

  Similarly, very old concept representations might be dropped if necessary

Conclusions

- Unfortunately, most of nonparametric techniques for analyzing $p(\boldsymbol{x})$ are meant for scalar data
  - These can be though applied to multivariate data by **monitoring the log-likelihood** of a models learned to describe unsupervised data

Kuncheva L.I., Change detection in streaming multivariate data using likelihood detectors, IEEE Transactions on Knowledge and Data Engineering, 2013, 25(5), 1175-1180 (DOI: 10.1109/TKDE.2011.226).

Conclusions

- Unfortunately, most of nonparametric techniques for analyzing $p(\boldsymbol{x})$ are meant for scalar data
  - These can be though applied to multivariate data by **monitoring the log-likelihood** of a models learned to describe unsupervised data
- Monitoring the **classification error** is straightforward **but**: the error of K_t is nonstationary, since K_t is updated.
  - It is more convenient to monitor the error of a second classifier K_0 that is never updated

Kuncheva L.I., Change detection in streaming multivariate data using likelihood detectors, IEEE Transactions on Knowledge and Data Engineering, 2013, 25(5), 1175-1180 (DOI: 10.1109/TKDE.2011.226).

Conclusions

- Extension to **gradual drifts**
  - «detection / adaptation» paradigm is not optimal since the post-change conditions are nonstationary
  - Need to interpret and compensate drift as in semi-supervised learning methods

Dyer K., Capo R., Polikar R., "COMPOSE: A Semi-Supervised Learning Framework for Initially Labeled Non-Stationary Streaming Data" IEEE Transactions on Neural Networks and Learning Systems, Special issue on Learning in Nonstationary and Dynamic Environments – Systems, vol. 25, no. 1, pp. 12-26, 2014

Thank you, questions?

Preprint and (some) codes available from

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

Just In Time Classifiers for Recurrent Concepts
Cesare Alippi, Giacomo Boracchi and Manuel Roveri,
IEEE Transactions on Neural Networks and Learning Systems, 2013. vol. 24, no.4, pp. 620 -634 *doi:10.1109/TNNLS.2013.2239309*

A just-in-time adaptive classification system based on the intersection of confidence intervals rule,
Cesare Alippi, Giacomo Boracchi, Manuel Roveri
Neural Networks, Elsevier vol. 24 (2011), pp. 791-800 *doi:10.1016/j.neunet.2011.05.012*

Thank you, questions?