Online Learning And Monitoring

Giacomo Boracchi, Francesco Trovò

May 13, 2020

Politecnico di Milano, DEIB

giacomo.boracchi@polimi.it

Refernces

Tutorials:

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

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

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

Surveys:

B. Krawczyk, L. Minku, J. Gama, J. Stefanowski, M. Wozniak. "Ensemble Learning for Data Stream Analysis: a survey", Information Fusion, 37, 132-156, 2017

G. Ditzler, M. Roveri, C. Alippi, R. Polikar. "Learning in Nonstationary Environments: A survey", IEEE Computational Intelligence Magazine, 10 (4), 12-25, 2015.

J Gama, I Žliobaitė, A Bifet, M Pechenizkiy, A Bouchachia . "A Survey on Concept Drift Adaptation", ACM Computing Surveys, 46 (4), 2014.

Adaptation Strategies

Adaptation Strategies Under Concept Drift

Two main solutions in the literature:

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

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

Active Approaches

Peculiarities:

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

Pro:

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

Cons:

- Difficult to handle incremental and gradual drifts

Passive Approaches

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:

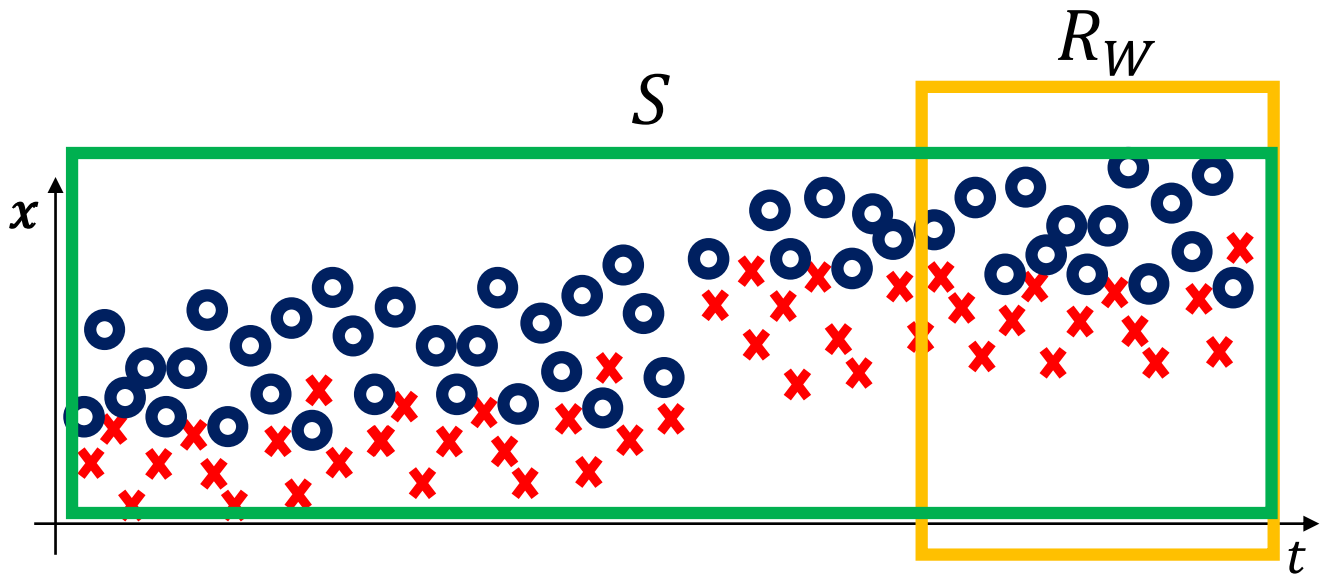
- **Single model** methods
- **Ensemble** methods

Adaptation in Active Approaches

Methods Based on Windows Comparison

Paired Learners

To cope with concept drift, we paired a stable online learner with a reactive one. A ***stable learner*** S *predicts based on all of its experience, whereas a* ***reactive learner*** R_W *predicts based on its experience over a short, recent window of time.*



Bach, S.H.; Maloof, M., *"Paired Learners for Concept Drift"* ICDM '08.

Paired Learners

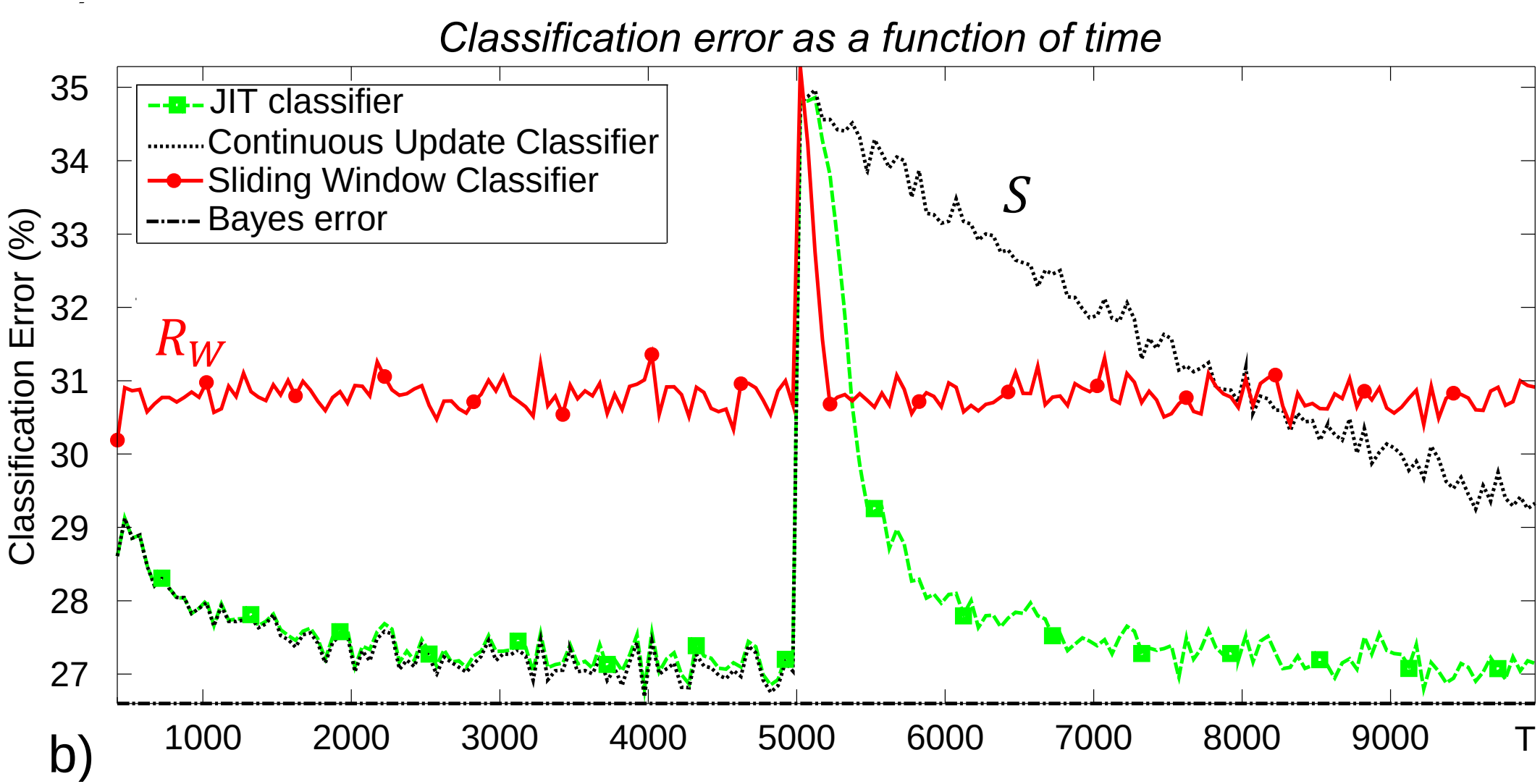
To cope with concept drift, we paired a stable online learner with a reactive one. A ***stable learner*** S *predicts based on all of its experience, whereas a* ***reactive learner*** R_W *predicts based on its experience over a short, recent window of time.*

Paired Learning copes with concept drift by:

- *Leveraging the interplay between reactive and stable learners*
- *Analyze the differences in their accuracy to cope with concept drift*

Bach, S.H.; Maloof, M., *"Paired Learners for Concept Drift"* ICDM '08.

Paired Learners



Paired Learners

Limitations:

- A too reactive R_W may have difficulty acquiring *any* target concept
- A too stable learner S may be overly burdened by knowledge of a previous concept to learn a new one.

Strengths:

- S outperforms R_W when acquiring a stationary concept,
- R_W outperforms S when the concept changes.

Idea:

- Detect a change when R_W outperforms S over a short window of time
- Adapt to the new concept by replacing S with R_W

Boracchi, Trovò

Paired Learners

Two classifiers are trained and steadily updated

- 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

- Only S provides the outputs of the model
- Predictions of R_w are computed but not provided
- As soon as, on the most recent samples, $\boldsymbol{R_w}$ **outperforms** S over a test window of length w, then **detect CD**

Adaptation consists in **replacing** S by R_w

Bach, S.H.; Maloof, M., *"Paired Learners for Concept Drift"* ICDM '08.

Paired Learners

This is to keep track of classification errors ϵ_t over the window w

Algorithm 1 Paired Learner

Input: $\{\vec{x}_t, y_t\}_{t=1}^{T}$, w, θ
$\{\vec{x}_t, y_t\}_{t=1}^{T}$: training data
w: window size for the reactive learner
θ: threshold for creating a new stable learner
Let S be a stable learner
Let R_w be a w-reactive learner
Let C be a circular list of w bits, each initially 0
for $t \leftarrow 1$ **to** T **do**
  $\hat{y}_S \leftarrow S$.Classify$(\vec{x}_t)$
  output $\hat{y}_S$
  $\hat{y}_R \leftarrow R_w$.Classify$(\vec{x}_t)$
  if $\hat{y}_S \neq y_t \wedge \hat{y}_R = y_t$ **then**
    C.set(t)
  else
    C.unset(t)
  end if
  if $\theta < C$.proportionOfSetBits$()$ **then**
    $S \leftarrow$ **new** StableLearner$()$
    $S \leftarrow R_w$.getConceptDescription$()$
    C.unsetAll$()$
  end if
  S.Train$(\vec{x}_t, y_t)$
  R_w.Train$(\vec{x}_t, y_t)$
end for

Bach, S.H.; Maloof, M., *"Paired Learners for Concept Drift"* ICDM '08.

Paired Learners

Predictions are only provided by *S*

Algorithm 1 Paired Learner

1: **Input:** $\{\vec{x}_t, y_t\}_{t=1}^{T}$, w, θ
2: $\{\vec{x}_t, y_t\}_{t=1}^{T}$: training data
3: w: window size for the reactive learner
4: θ: threshold for creating a new stable learner
5: Let S be a stable learner
6: Let R_w be a w-reactive learner
7: Let C be a circular list of w bits, each initially 0
8: **for** $t \leftarrow 1$ **to** T **do**
9: $\quad \hat{y}_S \leftarrow S.\text{Classify}(\vec{x}_t)$
10: $\quad$ **output** $\hat{y}_S$
11: $\quad \hat{y}_R \leftarrow R_w.\text{Classify}(\vec{x}_t)$
12: $\quad$ **if** $\hat{y}_S \neq y_t \wedge \hat{y}_R = y_t$ **then**
13: $\quad\quad C.\text{set}(t)$
14: $\quad$ **else**
15: $\quad\quad C.\text{unset}(t)$
16: $\quad$ **end if**
17: $\quad$ **if** $\theta < C.\text{proportionOfSetBits}()$ **then**
18: $\quad\quad S \leftarrow$ **new** StableLearner()
19: $\quad\quad S \leftarrow R_w.\text{getConceptDescription}()$
20: $\quad\quad C.\text{unsetAll}()$
21: $\quad$ **end if**
22: $\quad S.\text{Train}(\vec{x}_t, y_t)$
23: $\quad R_w.\text{Train}(\vec{x}_t, y_t)$
24: **end for**

Bach, S.H.; Maloof, M., *"Paired Learners for Concept Drift"* ICDM '08.

Paired Learners

Drift deteceted when more than θ times over the latest w samples, R_W provides a correct prediction while S does not

Algorithm 1 Paired Learner

Input: $\{\vec{x}_t, y_t\}_{t=1}^{T}$, w, θ
$\{\vec{x}_t, y_t\}_{t=1}^{T}$: training data
w: window size for the reactive learner
θ: threshold for creating a new stable learner
Let S be a stable learner
Let R_w be a w-reactive learner
Let C be a circular list of w bits, each initially 0
for $t \leftarrow 1$ **to** T **do**
  $\hat{y}_S \leftarrow S.\text{Classify}(\vec{x}_t)$
  output $\hat{y}_S$
  $\hat{y}_R \leftarrow R_w.\text{Classify}(\vec{x}_t)$
  if $\hat{y}_S \neq y_t \wedge \hat{y}_R = y_t$ **then**
    $C.\text{set}(t)$
  else
    $C.\text{unset}(t)$
  end if
  if $\theta < C.\text{proportionOfSetBits}()$ **then**
    $S \leftarrow$ **new** StableLearner()
    $S \leftarrow R_w.\text{getConceptDescription}()$
    $C.\text{unsetAll}()$
  end if
  $S.\text{Train}(\vec{x}_t, y_t)$
  $R_w.\text{Train}(\vec{x}_t, y_t)$
end for

Bach, S.H.; Maloof, M., *"Paired Learners for Concept Drift"* ICDM '08.

Paired Learners

Algorithm 1 Paired Learner

1: **Input:** $\{\vec{x}_t, y_t\}_{t=1}^{T}$, w, θ
2: $\{\vec{x}_t, y_t\}_{t=1}^{T}$: training data
3: w: window size for the reactive learner
4: θ: threshold for creating a new stable learner
5: Let S be a stable learner
6: Let R_w be a w-reactive learner
7: Let C be a circular list of w bits, each initially 0
8: **for** $t \leftarrow 1$ **to** T **do**
9: $\quad \hat{y}_S \leftarrow S.\text{Classify}(\vec{x}_t)$
10: $\quad$ **output** $\hat{y}_S$
11: $\quad \hat{y}_R \leftarrow R_w.\text{Classify}(\vec{x}_t)$
12: $\quad$ **if** $\hat{y}_S \neq y_t \wedge \hat{y}_R = y_t$ **then**
13: $\quad\quad C.\text{set}(t)$
14: $\quad$ **else**
15: $\quad\quad C.\text{unset}(t)$
16: $\quad$ **end if**
17: $\quad$ **if** $\theta < C.\text{proportionOfSetBits}()$ **then**
18: $\quad\quad S \leftarrow$ **new** StableLearner()
19: $\quad\quad S \leftarrow R_w.\text{getConceptDescription}()$
20: $\quad\quad C.\text{unsetAll}()$
21: $\quad$ **end if**
22: $\quad S.\text{Train}(\vec{x}_t, y_t)$
23: $\quad R_w.\text{Train}(\vec{x}_t, y_t)$
24: **end for**

Adaptation: R_W replaces S and the error computation is reset
Detection suggests that samples before $t - w$ do not conform with the current status of the process

Bach, S.H.; Maloof, M., *"Paired Learners for Concept Drift"* ICDM '08.

Fourth Matlab Assignment

Boracchi, Trovò

Fourth Matlab Assignment

Implement Paired Learners to cope with a concept-driven environment

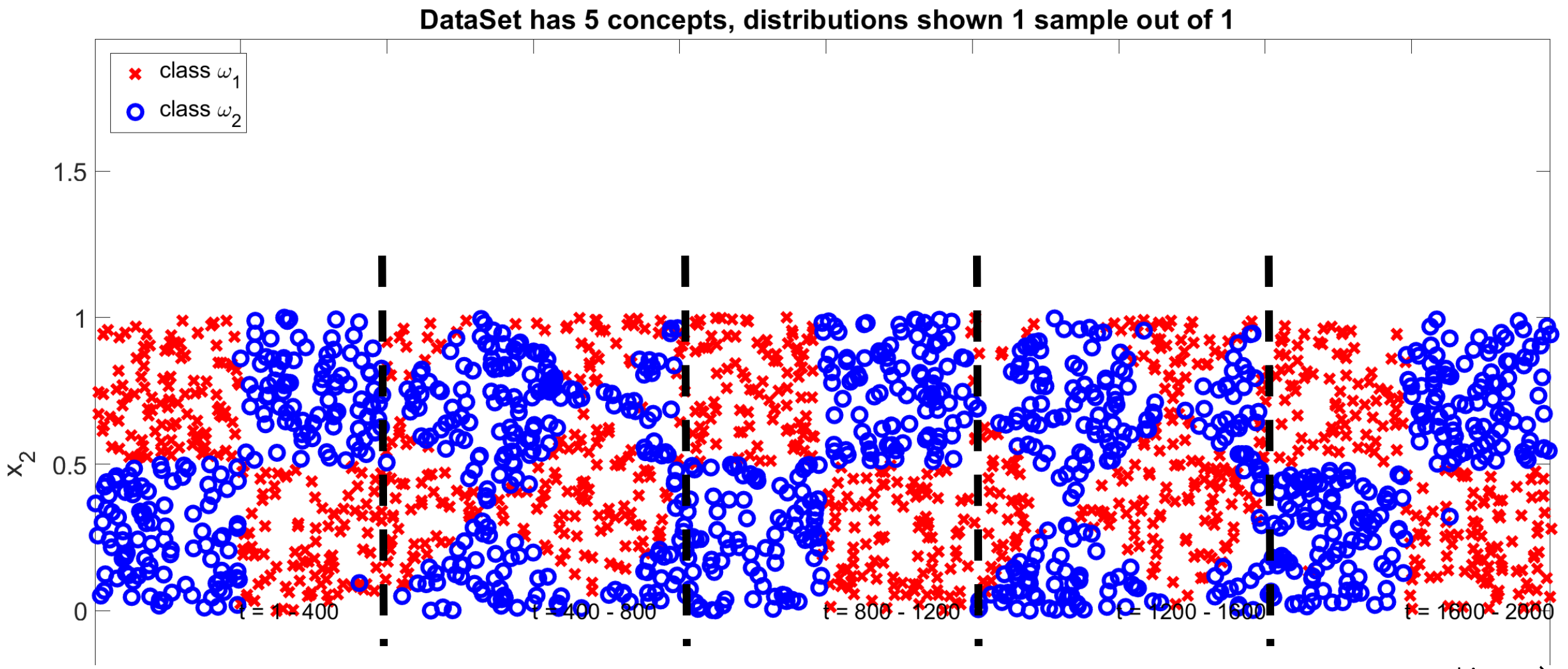
- Watch out to correcty update the classifiers accordingly

Display the classification error of the

- Paired learner output
- The stable classifier
- The reactive classifier
- A classifier that is never updated

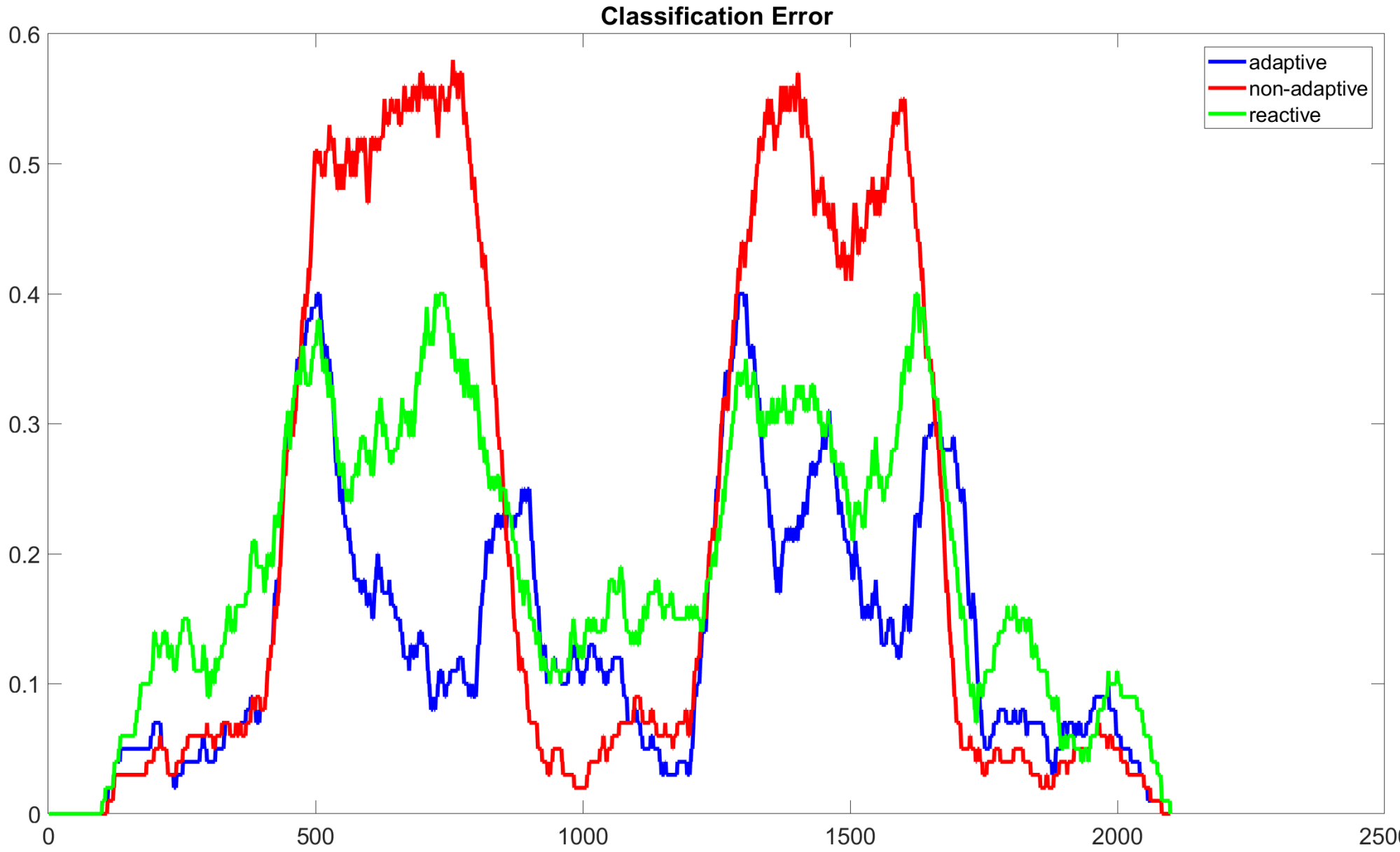
Test paired learners on different concept drift sequences, possibly including recurrent concepts

Checkerboard Dataset



Boracchi, Trovò

Classification Error Paired Learners



Boracchi, Trovò

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 TNNLS 2016

An example of Concept Representations

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

$Z_i = \{(\boldsymbol{x_0}, y_0), \ldots, (\boldsymbol{x_n}, y_n)\}$: **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 $C_0 = (Z_0, F_0, D_0)$ 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

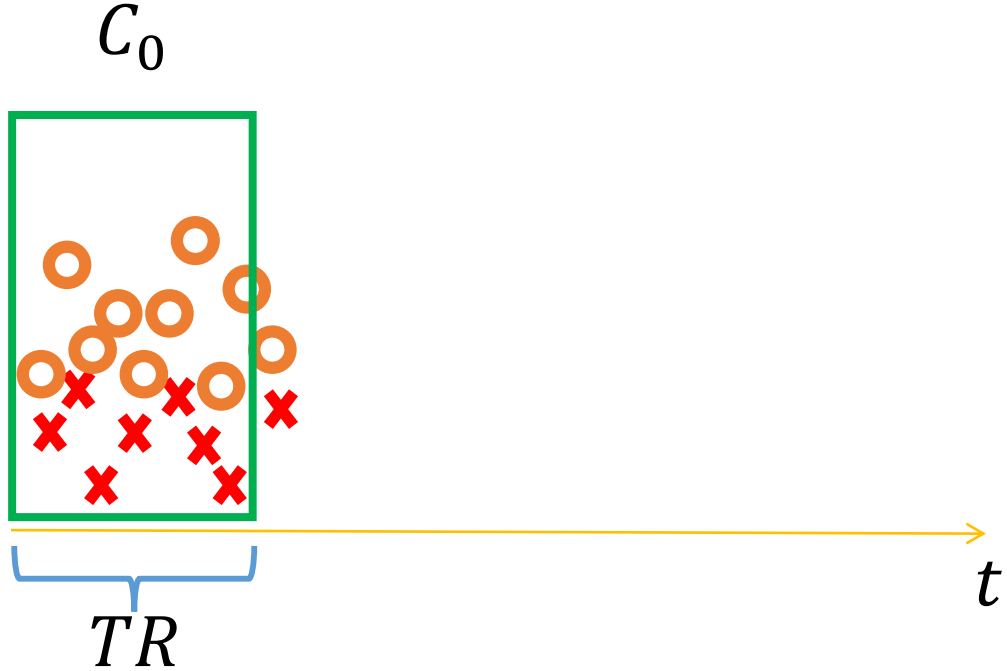
Initial Training

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

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- $\quad\quad \hat{y}_t = K(Z_i \cup Z_{\text{rec}}, x_t)$.
$\quad$ **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

Boracchi, Trovò

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:

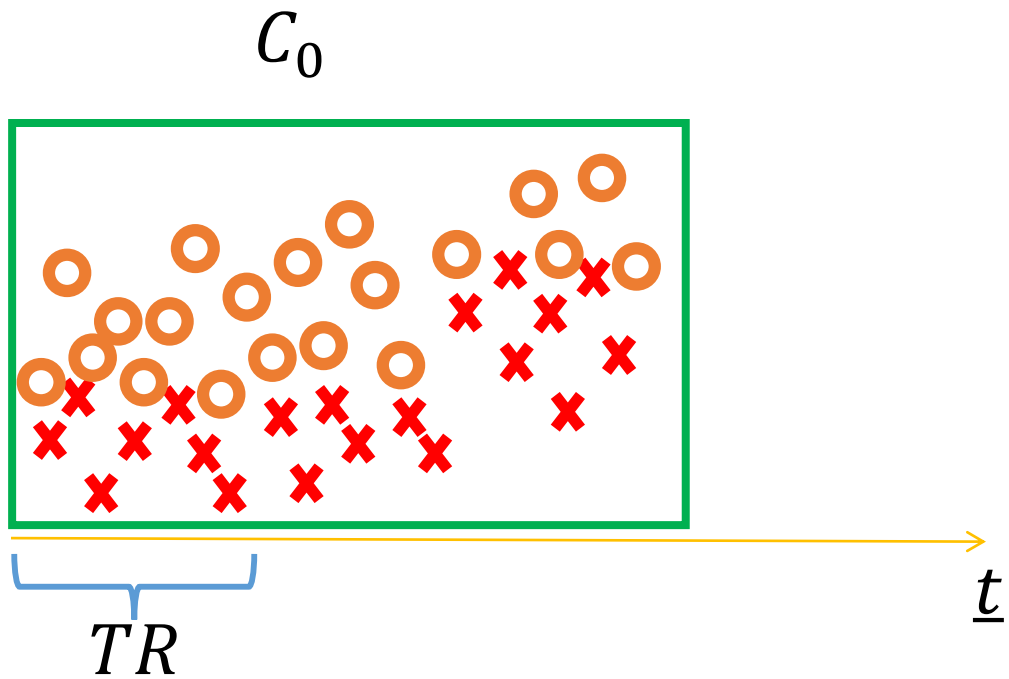
- Extract features that are appended to F_i
- Append supervised information in Z_i

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

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

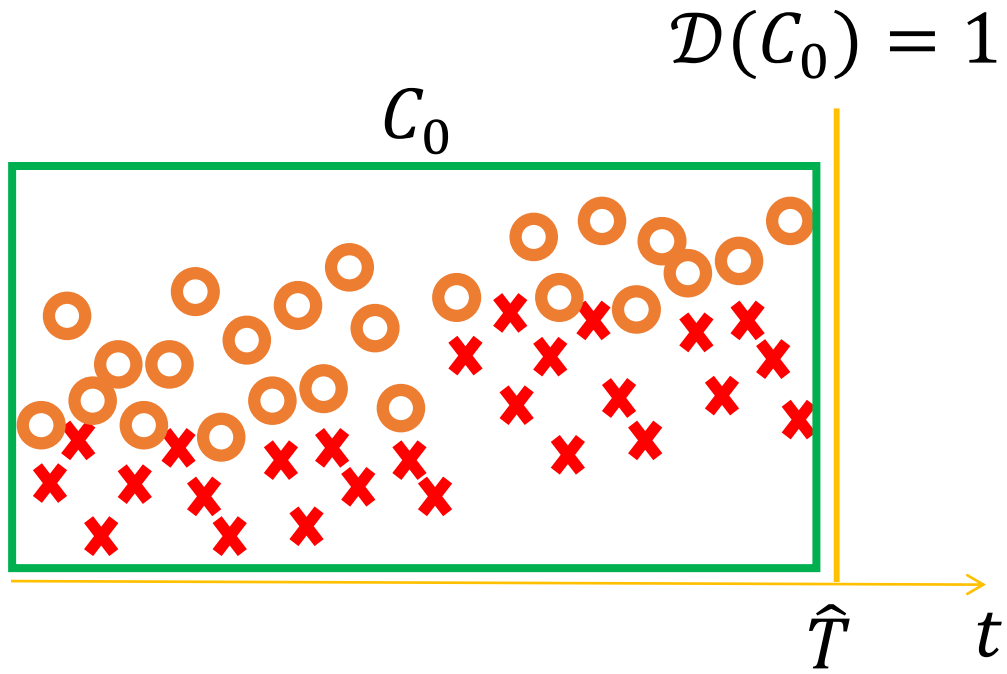
Concept Drift Detection:

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 $\boldsymbol{D}$** 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 $\phi(y|\boldsymbol{x})$ and $\phi(\boldsymbol{x})$ 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 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,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- $\quad \mathcal{U}(C_i, \{x_t\}) \rightarrow C_i$;
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 Split:

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

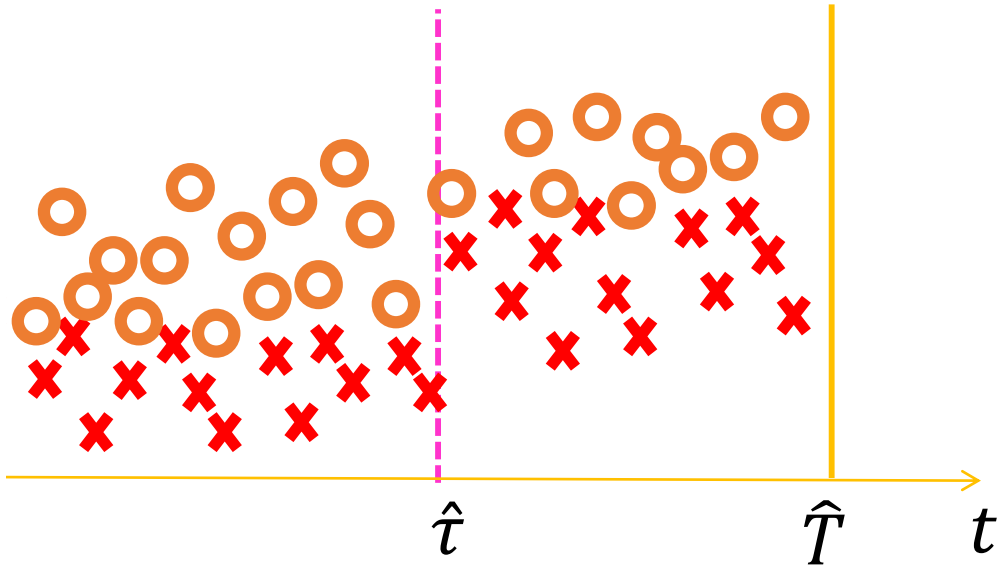
A new concept description is built

JIT Classifiers: Concept Split

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

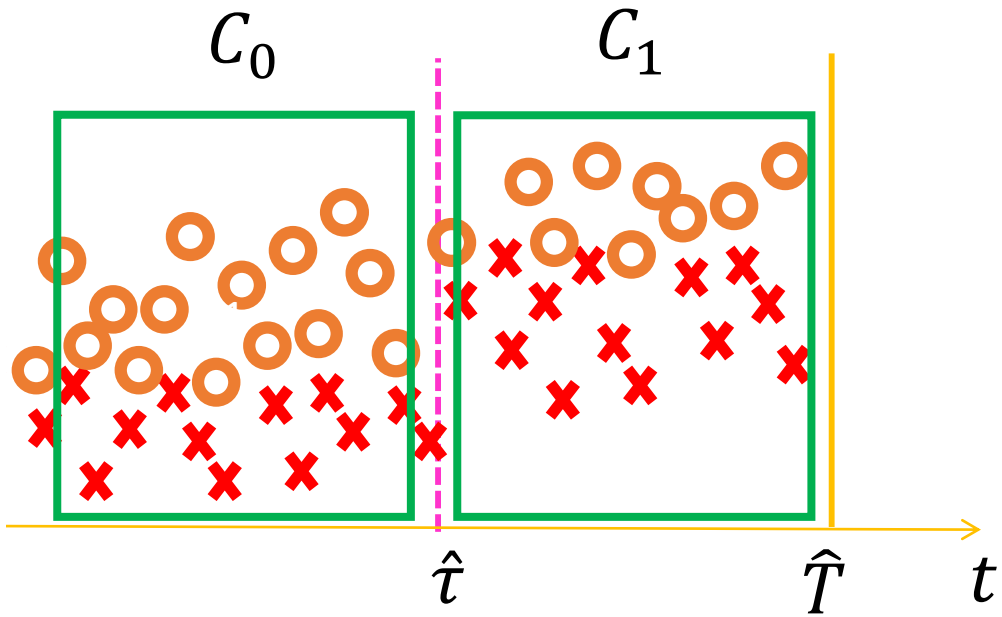
Uses statistical tools for performing an **offline** and **retrospective analysis** over the recent data, like:

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



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:

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

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

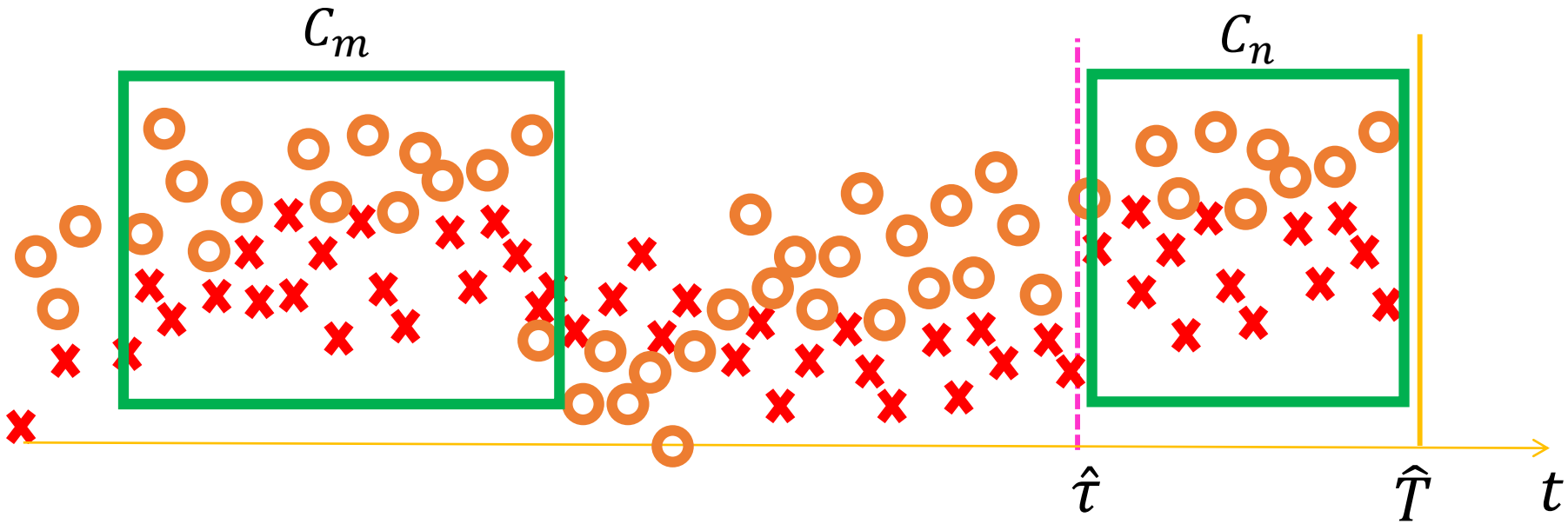
Gather supervised samples from all the representations C_j that refers to the same concept

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

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



Testing for Equivalence

Conventional HTs are meant to assess if two populations are different and **assume under H_0 that "they are the same"**

When there is not enough statistical evidence to conclude that the two populations are different, there is no hint on whether the two populations are the same.

We use Two One-Sided t-Test (TOST) to assess equivalence of F_0 and F_1.

In TOST, H_0 corresponds to the non-equivalence of the two populations. Discarding H_0 implies accepting that the two populations are equivalent.

$$H_0 : \text{“} \left(\overline{F}_j^{\kappa} - \overline{F}_i^{\kappa}\right) < \theta_L \text{ or } \left(\overline{F}_j^{\kappa} - \overline{F}_i^{\kappa}\right) > \theta_U, \text{”}$$

$$H_1 : \text{“} \theta_L \leq \overline{F}_j^{\kappa} - \overline{F}_i^{\kappa} \leq \theta_U, \text{”}$$

[40] P. Bauer and M. Kieser, "A unifying approach for confidence intervals and testing of equivalence and difference," Biometrika, vol. 83, no. 4, pp. pp. 934–937, 1996. http://www.jstor.org/stable/2337298

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

Label Prediction:

The classifier K is reconfigured using all the available supervised couples

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 / any rate
- **Perform continuous adaptation** of their model(s) parameters at each new arrival

They can be divided in:

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

Passive Approach

- Overcomes the potential disadvantage of active methods that can get false alarms or delays in the detection of drift.
- Potentially **more adequate** for **slow concept drifts.**
- **Potential disadvantage** of not using concept drift detection: don’t inform users of whether concept drift is occurring.

Single Classifier Models

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 Models

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 Models

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

A Dilemma of Sorts

Stability

The ability of an algorithm to recall old information that it has learned in the past

Plasticity

The ability for an algorithm to learn new information when data are available

Sounds like we could have two opposing ideas!

Boracchi, Trovò

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 different models

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

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

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

Ensemble Methods

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

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

Each **i** aining set an

Final **of the indivi**

The weight α_i encode how reliable the prediction from h_i is. Different methods set different weighting schemes, which are typically based on the posterior of h_i or the accuracy of h_i over recent data

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

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

Ensemble Methods and Concept Drift

Each individual h_i implicitly refers to a component of a mixture distribution characterizing a **concept**

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 α_i
- **structural update**: including new (pruning old) 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 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

Adaptive strategies can be applied to add/remove classifiers by on individual classifier and the ensemble error

Kuncheva, L. I. *"Classifier ensembles for changing environments"* In Workshop on Multiple Classifier Systems. MCS. 1–15 2004.

Streaming Ensemble Algorithm: SEA

An **ensemble of a fixed number** of individuals performs

- **batch learning**
- **structural update** to adapt to concept drift

Two additional classifiers are stored h_t and h_{t-1}

- h_t is being trained on the current batch
- h_{t-1} is the classifier trained on the previous batch

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

Streaming Ensemble Algorithm: SEA

When a new batch $S = \{(\boldsymbol{x_0^t}, y_0^t), (\boldsymbol{x_1^t}, y_1^t), \ldots, (\boldsymbol{x_B^t}, y_B^t)\}$ 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})

Classifier h_t is never added as its performance are affected by overfitting

Adaptation to concept drift is performed by replacing individuals (no update of each individual instead)

Pruning the ensemble to improve the overall performance

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

Streaming Ensemble Algorithm: SEA

The **individuals are decision trees**, which enables fast processing

Majority voting as aggregation strategy over the ensemble

"Quality" of an individual is an indicator to **favor individuals that correctly classify recent samples where the ensemble was "undecided"** providing a score close to 0.5 (in two class problems)

```
while more data points are available
  read d points, creating training set D
  build classifier C_i using D
  evaluate classifier C_{i-1} on D
  evaluate all classifiers in ensemble E on D
  if E not full
    insert C_{i-1}
  else if Quality(C_{i-1}) > Quality(E_j) for some j
    replace E_j with C_{i-1}
  end
end
```

Figure 1: Pseudocode for streaming ensembles. E_j represents the jth tree in ensemble E.

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

Dynamic Weighted Majority: DWM

Dynamic weighted majority (DWM) is an ensemble where:

- **Individuals** are trained on different batches of data and **regularly updated** at a pre-defined frequency
- 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**
- When **the ensemble makes a wrong guess**, a **new individual** is added
- Predictions are made by **weighted majority voting**
- **The ensemble size is also dynamic as it might vary over time**

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:

- Two different **weighting schemes** for **individuals** and **training samples**
- **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

Elwell R. and Polikar R., "Incremental Learning of Concept Drift in Nonstationary Environments" IEEE TNNLS , vol. 22, 2011.

Diversity, Diversity, Diversity: DDD

Intuitively, the **key to the success** of an ensemble of classifiers is that the **individual classifiers perform diversely**

There is no consensus on the best measure of diversity between classifiers. A practical option is Q statistic to compare h_i and h_k

$$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}$$

Where $N^{a,b} = \#\{\boldsymbol{x}, h_i(\boldsymbol{x}) = a \text{ and } h_k(\boldsymbol{x}) = b\}$ or $0, 1$ indicates wrong / correct prediction.

When h_i and h_k are identical, $Q_{i,k} = 1$, negative values of $Q_{i,k}$ indicates more disagreement and more diverse classifiers

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,

N Oza, "On-line Ensemble Learning," PhD Thesis, University of California, Berkeley, 2001.

Diversity, Diversity, Diversity: DDD

Intuitively, the key to the success of an ensemble of classifiers is that the individual classifiers perform diversely

There is no consensus on the best measure of diversity between classifiers. A practical option is Q statistic to compare h_i and h_k

$$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}$$

Wher rong / corre

In ensembles, the average value of $Q_{i,k}$ indicates the diversity of the ensemble

When h_i and h_k are identical, $Q_{i,k} = 1$, negative values of $Q_{i,k}$ indicates more disagreement and more diverse classifiers

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,

N Oza, "On-line Ensemble Learning," PhD Thesis, University of California, Berkeley, 2001.

Diversity, Diversity, Diversity: 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,

N Oza, "On-line Ensemble Learning," PhD Thesis, University of California, Berkeley, 2001.

Initially Labeled Environments

Initially Labeled Environments

All previous learning scenarios assumed a supervised learning setting

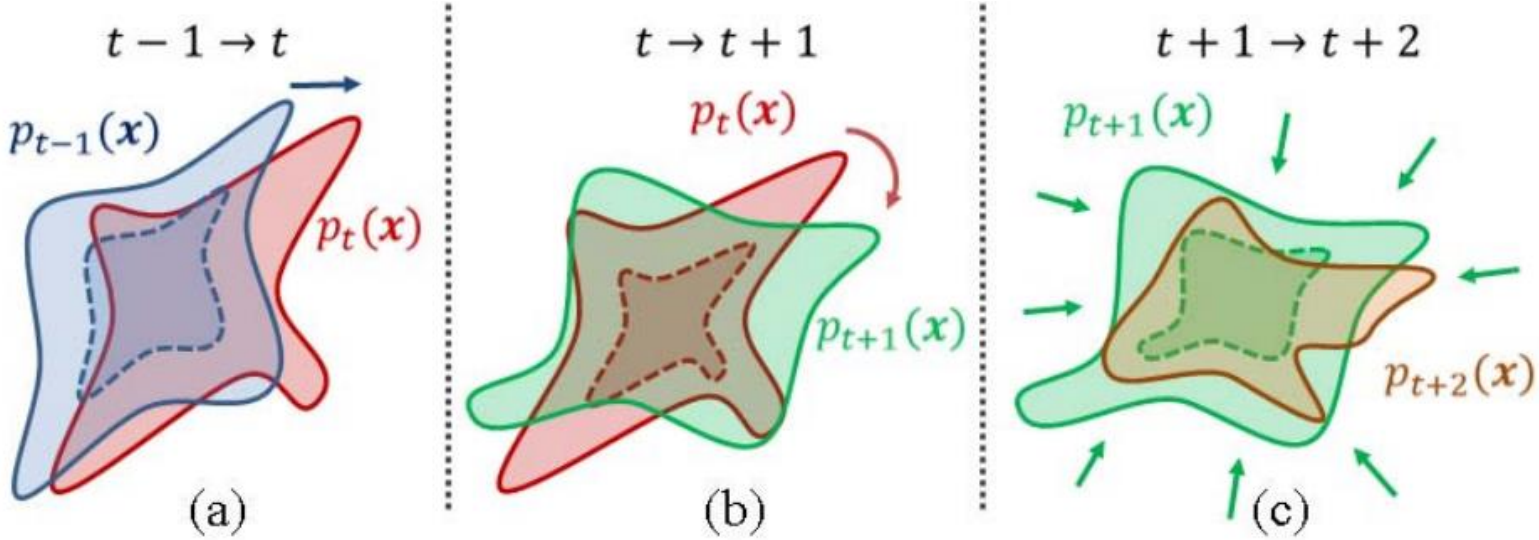
What if we're only provide some labeled data at T_0 and all future time points are unlabeled?

Active Learning versus Learning in Initially Labeled Environments

- AL: Assume that we have access to an oracle that can label the unlabeled data at a cost
- ILNSE: Extreme latency verification! No labeled data are received after T_0

Compacted Object Sample Extraction: COMPOSE

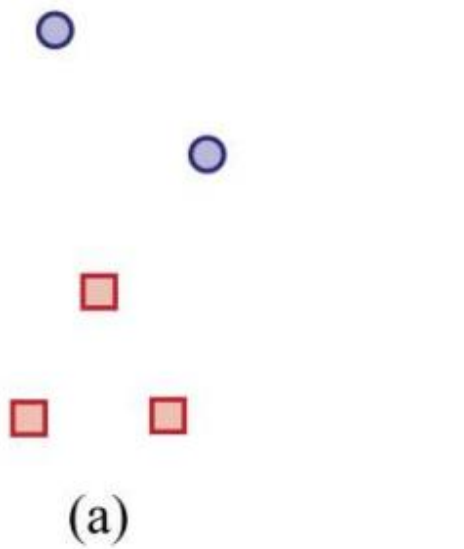
Intuition: often, the «central» part of the distribution is preserved under concept drift. **Exploit unlabled data to identify the «core set» of each class**



Progression of a single class experiencing (a) translational, (b) rotational, and (c) volumetric drift.

Dyer, Karl B., Robert Capo, and Robi Polikar. "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data." *IEEE transactions on neural networks and learning systems* 25.1 (2013): 12-26.

Compacted Object Sample Extraction: COMPOSE

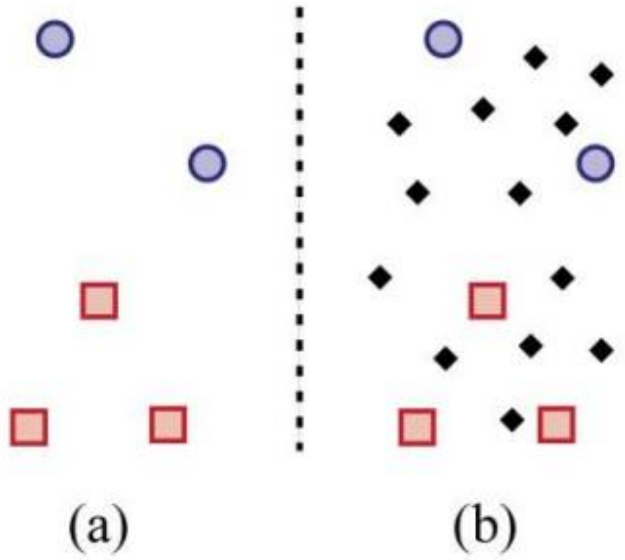


(a)

Initially Labeled Data

Dyer, Karl B., Robert Capo, and Robi Polikar. "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data." *IEEE transactions on neural networks and learning systems* 25.1 (2013): 12-26.

Compacted Object Sample Extraction: COMPOSE

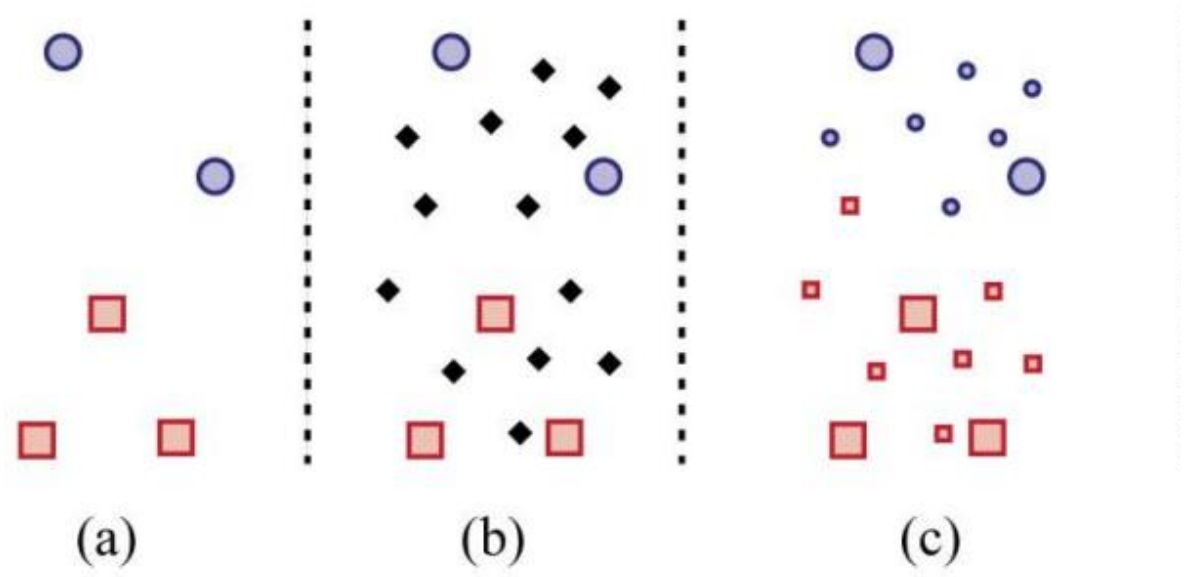


Initially Labeled Data

Receive Unlabeled Data

Dyer, Karl B., Robert Capo, and Robi Polikar. "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data." *IEEE transactions on neural networks and learning systems* 25.1 (2013): 12-26.

Compacted Object Sample Extraction: COMPOSE



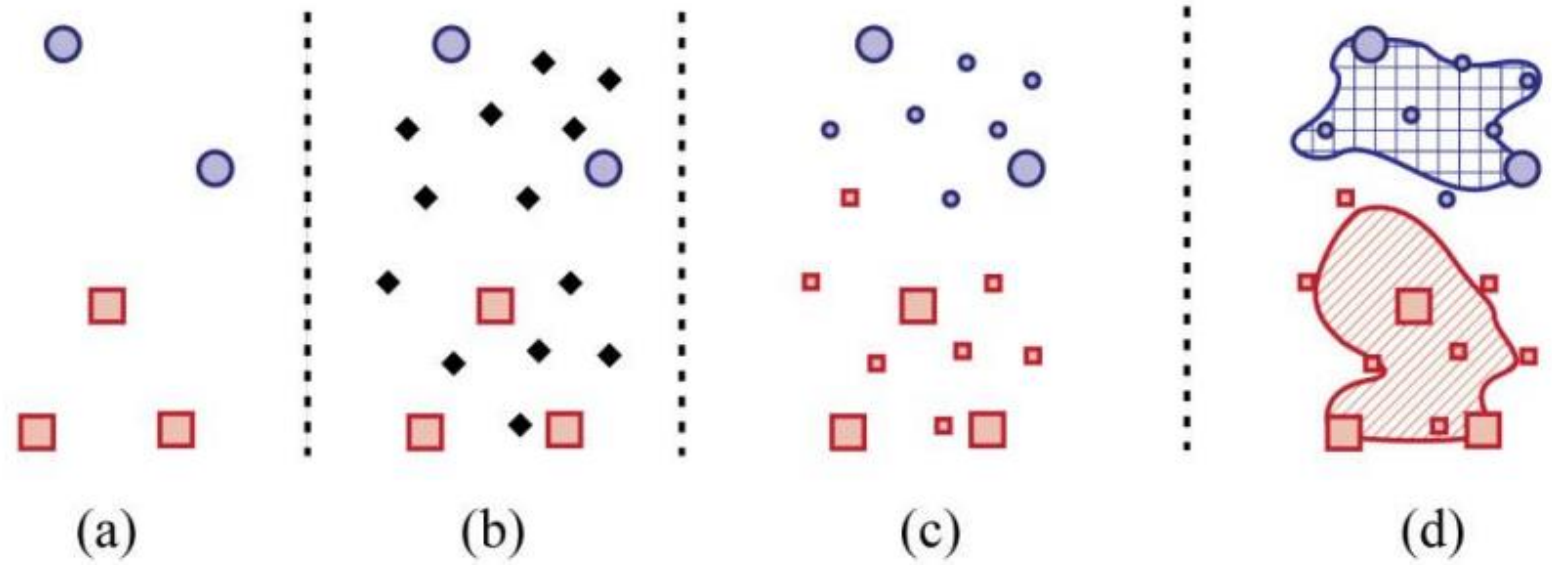
Initially Labeled Data

Receive Unlabeled Data

Classify Using SSL

Dyer, Karl B., Robert Capo, and Robi Polikar. "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data." *IEEE transactions on neural networks and learning systems* 25.1 (2013): 12-26.

Compacted Object Sample Extraction: COMPOSE



Initially Labeled Data

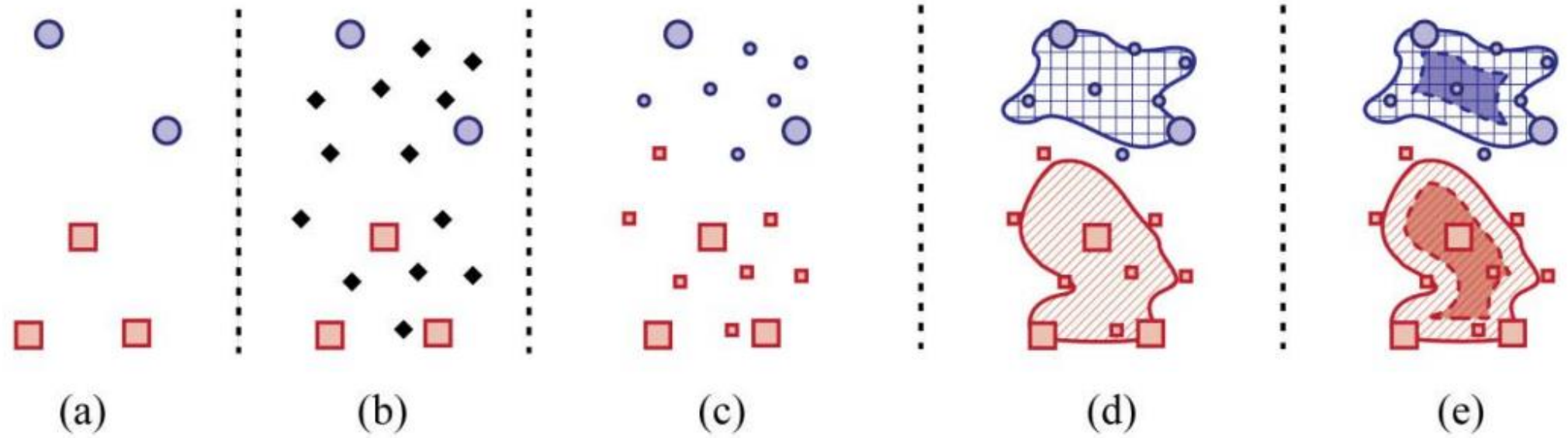
Receive Unlabeled Data

Classify Using SSL

Construct a Boundary

Dyer, Karl B., Robert Capo, and Robi Polikar. "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data." *IEEE transactions on neural networks and learning systems* 25.1 (2013): 12-26.

Compacted Object Sample Extraction: COMPOSE



Initially Labeled Data

Receive Unlabeled Data

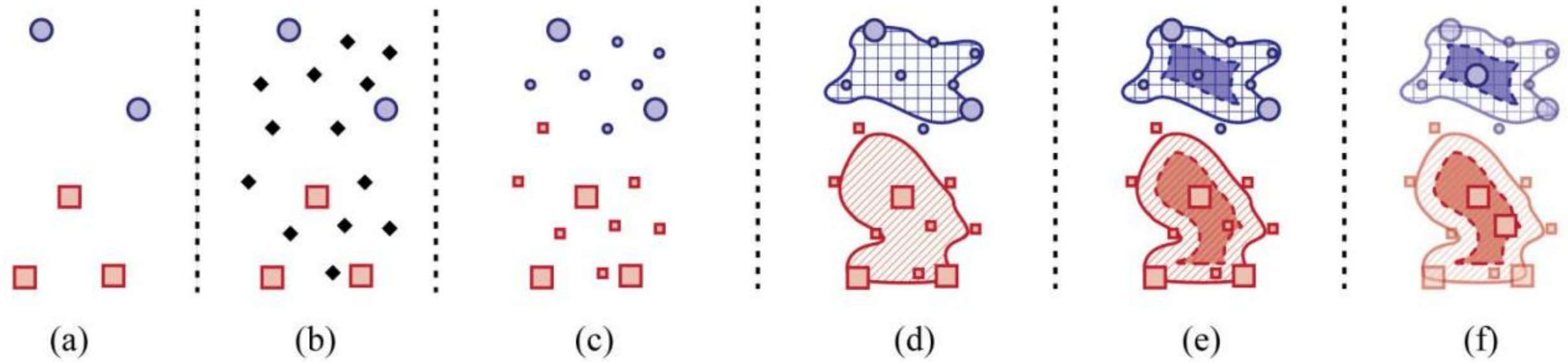
Classify Using SSL

Construct a Boundary

Compact the Boundary

Dyer, Karl B., Robert Capo, and Robi Polikar. "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data." *IEEE transactions on neural networks and learning systems* 25.1 (2013): 12-26.

Compacted Object Sample Extraction: COMPOSE



Initially Labeled Data

Receive Unlabeled Data

Classify Using SSL

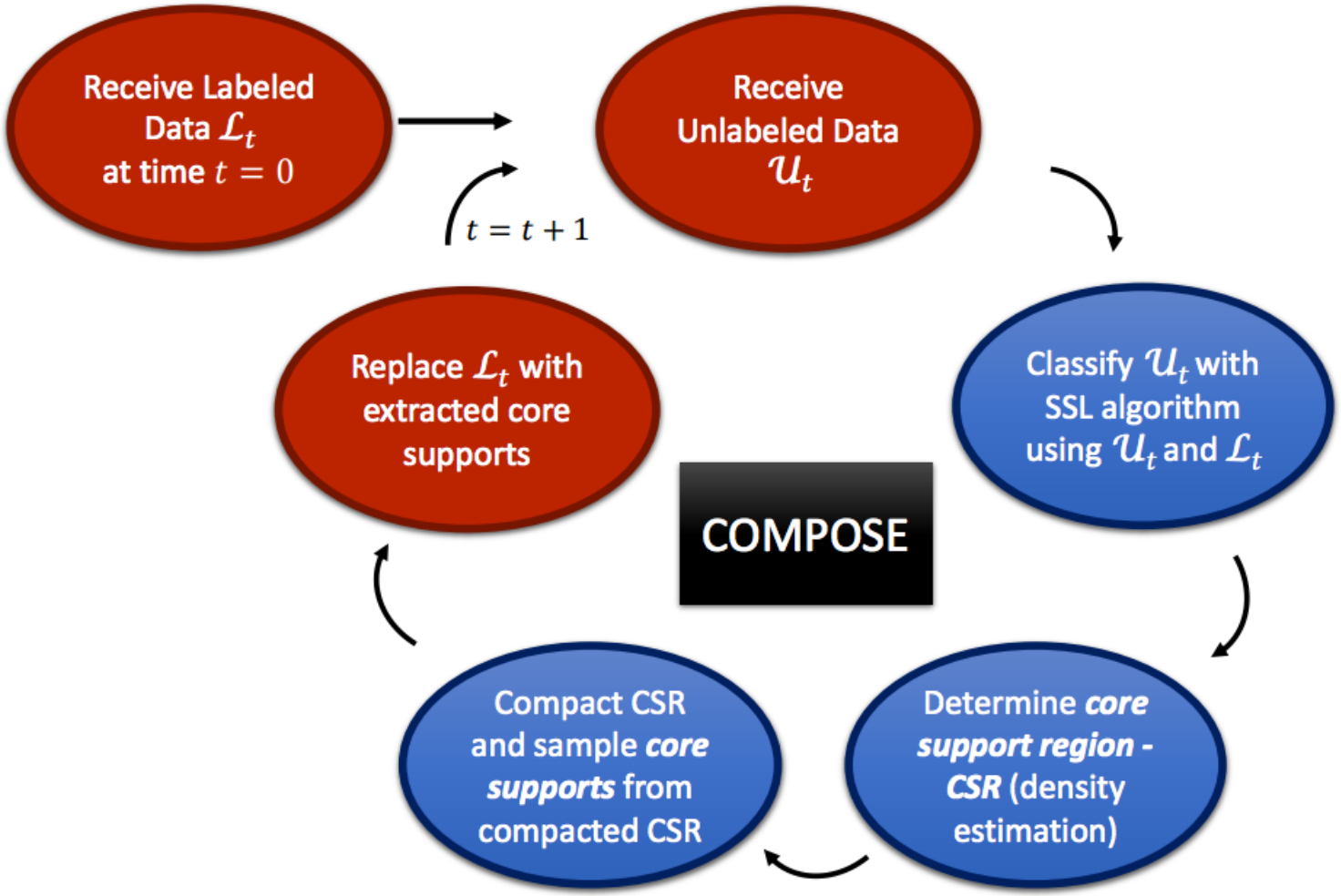
Construct a Boundary

Compact the Boundary

Extract Core Set

Dyer, Karl B., Robert Capo, and Robi Polikar. "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data." *IEEE transactions on neural networks and learning systems* 25.1 (2013): 12-26.

Compacted Object Sample Extraction: COMPOSE

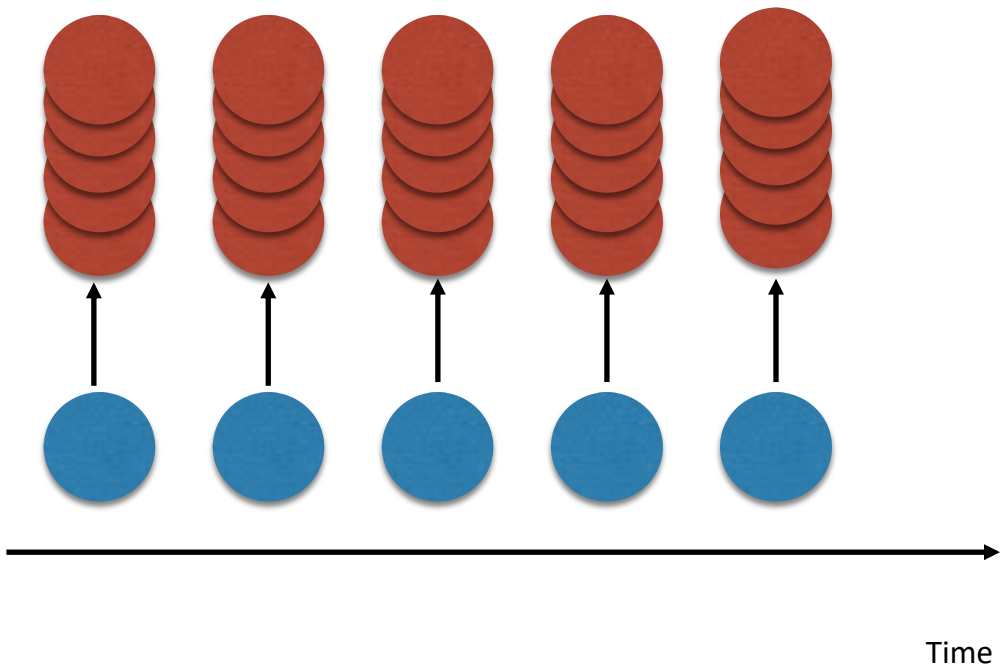


Dyer, Karl B., Robert Capo, and Robi Polikar. "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data." *IEEE transactions on neural networks and learning systems* 25.1 (2013): 12-26.

Performance Measures for Learning in NSE

Performance on a Separate Test Set

Problem: typically infeasible for real world problems on datastreams.

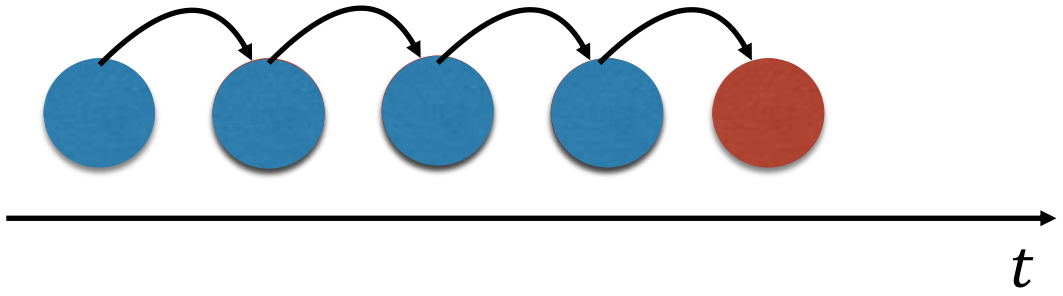


Slide credits «Tutorial on Learning Class Imbalanced Data Streams" Minku, Wang, Boracchi, IJCNN 2018

Prequential Performance

Predict and Train: prequential error is the time-averaged error over all the supervised errors that are first tested, then used for training.

Problem: does not reflect the *current* performance.



$$perf^{(t)} \begin{cases} perf_{ex}^{(t)}, \, if \, t{=}1 \\ \dfrac{(t-1)perf^{(t-1)} + perf_{ex}^{(t)}}{t}, \, otherwise \end{cases}$$

Exponentially Decayed Prequential Performance

A prequential error which takes more into account error on recent samples

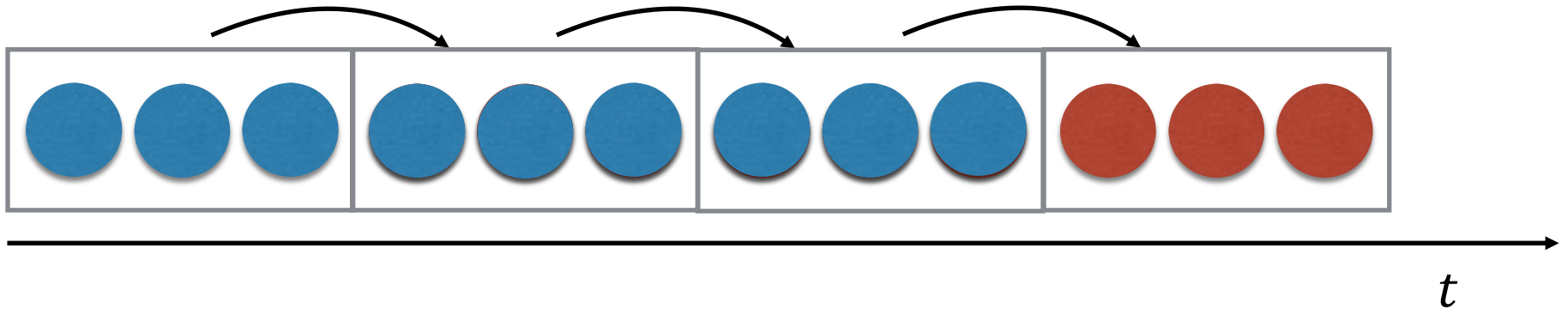
$$\text{perf}^{(t)} \begin{cases} \text{perf}_{ex}^{(t)}, \text{if } t = 1 \\ \eta \cdot \text{perf}^{(t-1)} + (1-\eta) \cdot \text{perf}_{ex}^{(t)}, \textit{ otherwise} \end{cases}$$

Alternative for artificial datasets: reset prequential performance upon known concept drifts.

J.Gama, R.Sebastiao, P.P.Rodrigues. “Issues in Evaluation of Stream Learning Algorithms”, in the ACM SIGKDD international conference on knowledge discovery and data mining, 329338, 2009.

Chunk-Based Performance

These hold only for i.i.d. data



Concluding Remarks

Comments from my personal experience

In Learning problems the classification error is typically the most important figure of merit.

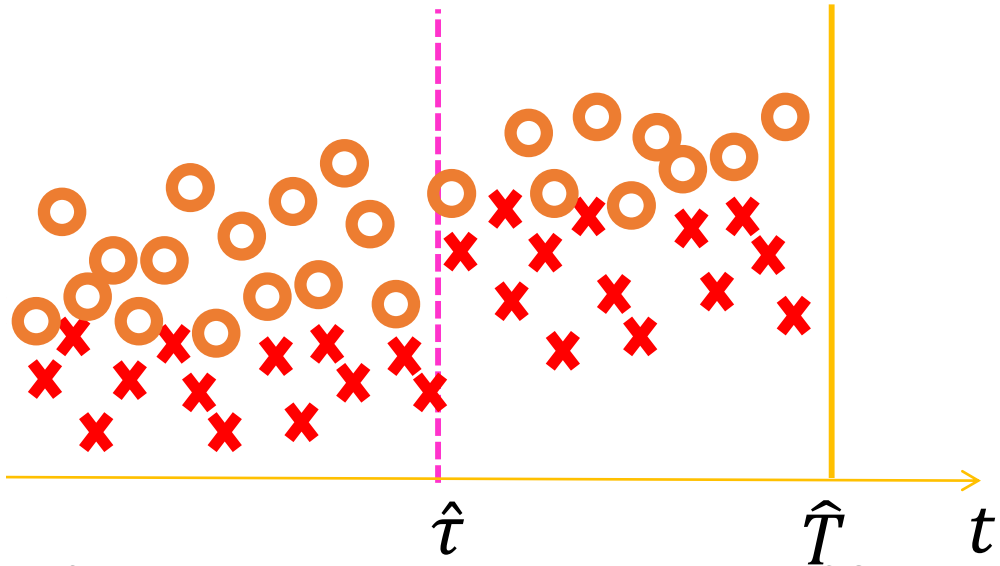
- In this scenario, in general, false positives hurt less than detection delays
- Things might change on class unbalance

Active approaches might be penalized due to their detection delay, while passive approaches might start adaptation earlier

Comments from my personal experience

Providing enough 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, 2003.

Comments from my personal experience

Exploiting recurrent concept is important

- Providing additional samples could make the difference
- Mitigate the impact of false positives

Comments from my personal experience

- Ensemble classifier approaches have had more success that single classifier implementations for nonstationary environments
- Hybrid approaches (active & passive) can be beneficial!
- In practice, a weighted majority vote is a better strategy as long as we have a reliable estimate of a classifiers error

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