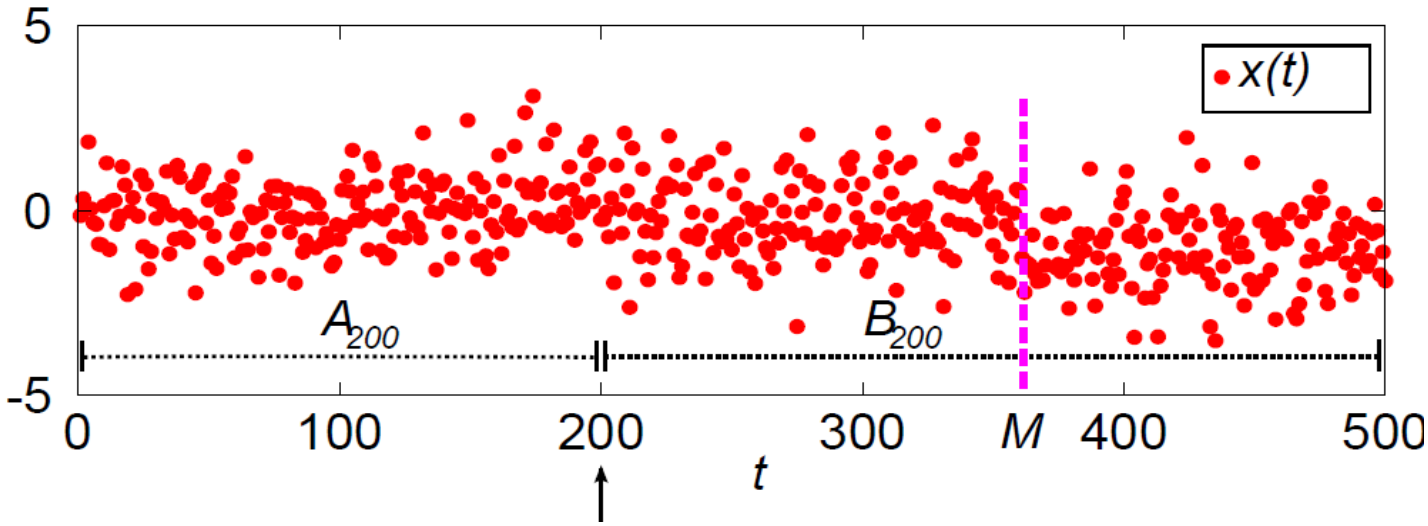
Learning in Nonstationary environments

«It takes a life to learn how to live, a life to learn how to die»
Seneca

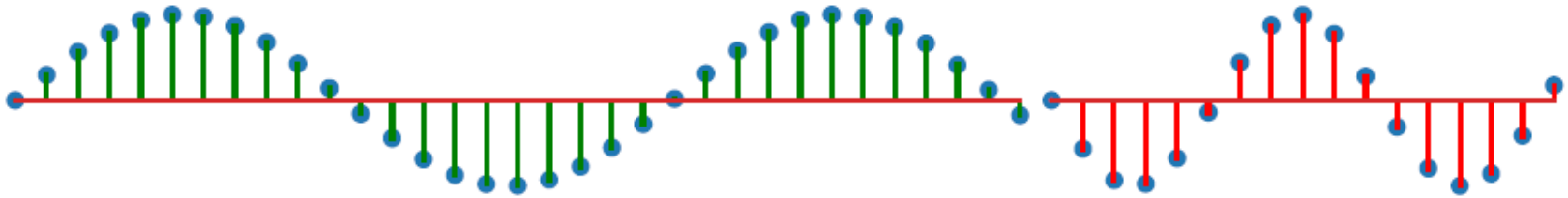
«Panta Rei», Heraclitus

Our (data) streams …

- **Sequences of i.i.d data**

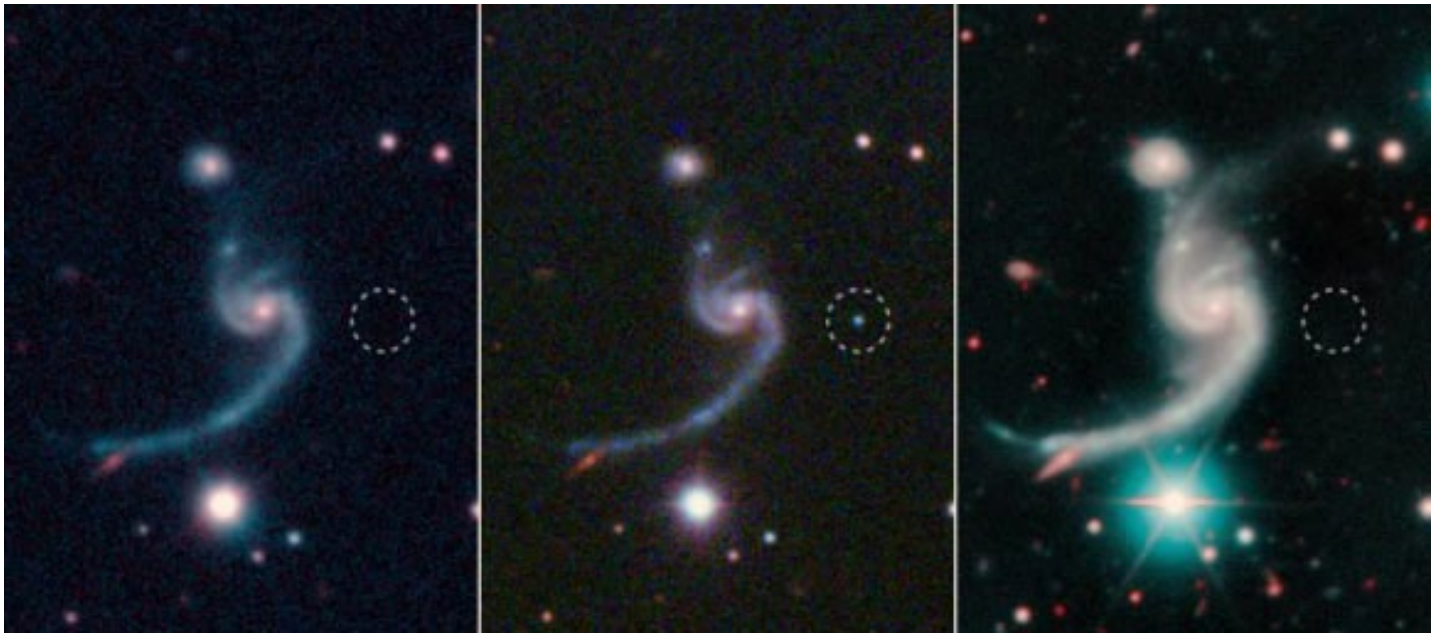


- **Signals**



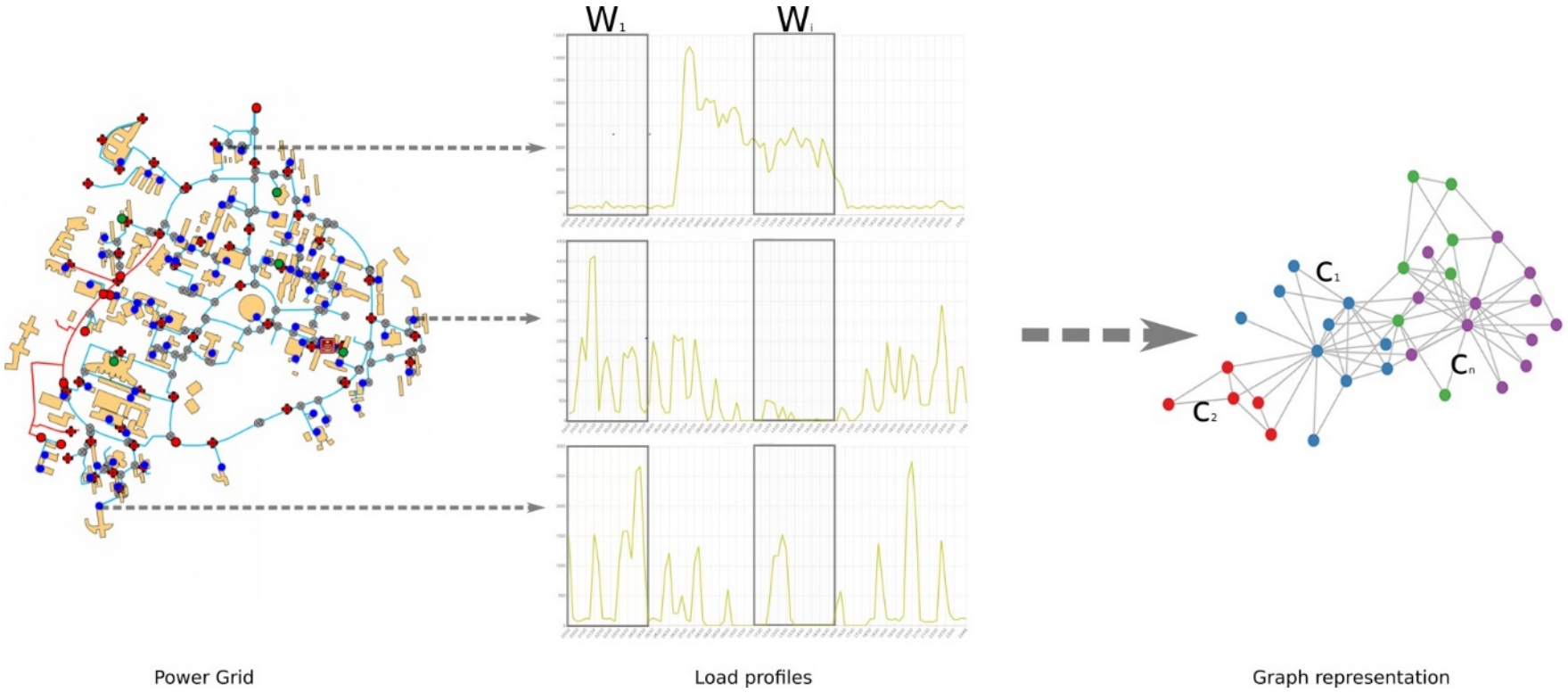
Our (data) streams …

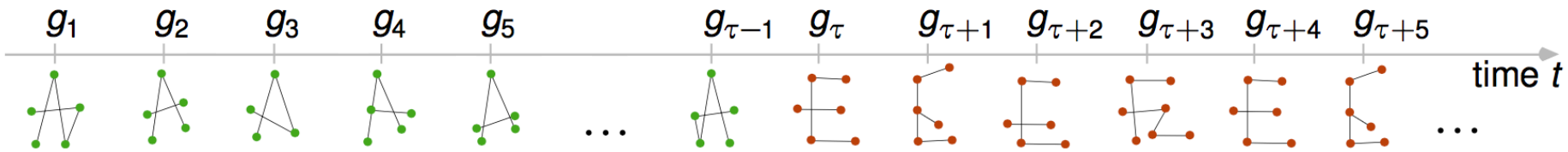
- **Images**



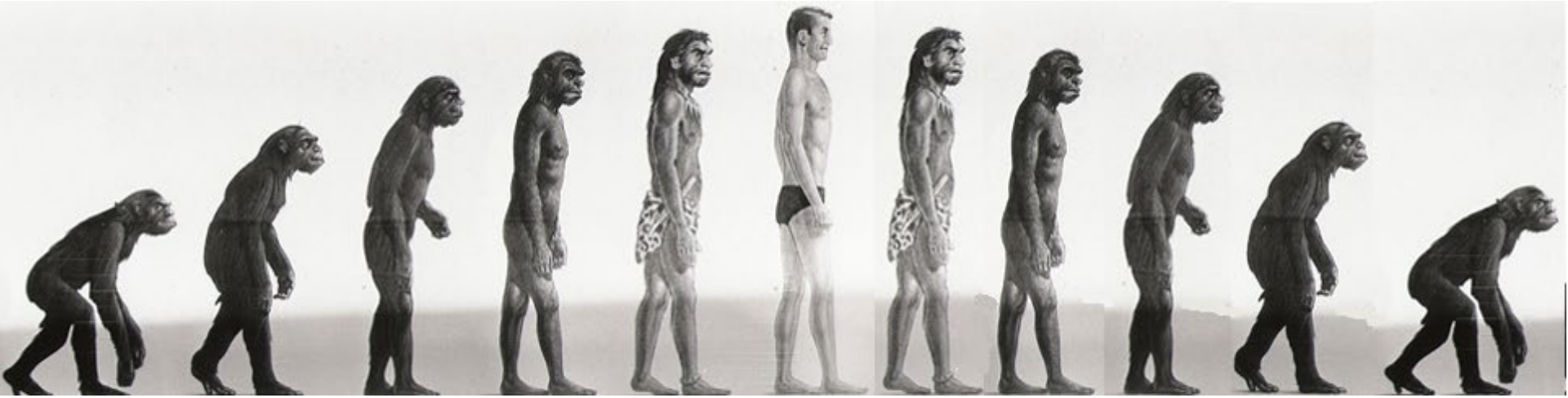
Our (data) streams …

- **Graphs**





Everything and everybody *evolve* with *time*



What about our applications?

The unsupervised framework we are familiar with

The *stationary* process generates the i.i.d sequence

$$x_1, \cdots, x_n$$

Applications e.g.,

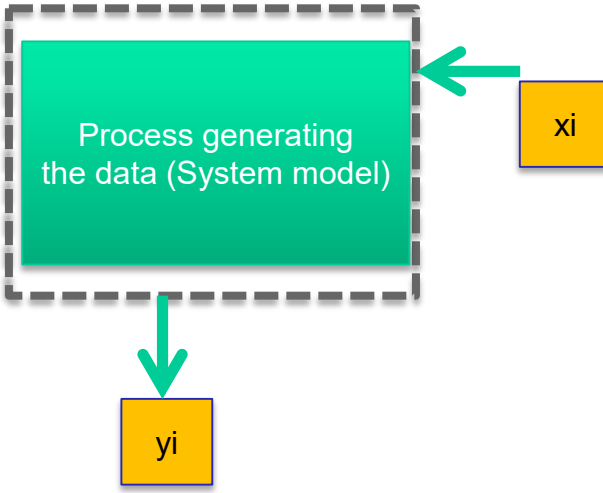
- Cluster data
- Detect anomalies (rare events, faults)

and the supervised –regression- one

The *stationary* process generating the data

$$y = g(x) + \eta$$

.. i.i.d. couples for training, validation and test sets



- In many cases the i.i.d. assumption holds for short time only
- We need to update our application to track changes in the process: **Learning in non-stationary environments (NSE)**

$$f(\hat{\theta}, x)$$

How to?

Dealing with NSE

The simplest strategy to deal with NSE within a stream-based application is to retrain the application from the scratch over time (windowing) in order to adapt to the new framework.

- The strategy is time consuming and requires to fully re-train a model (or part of)

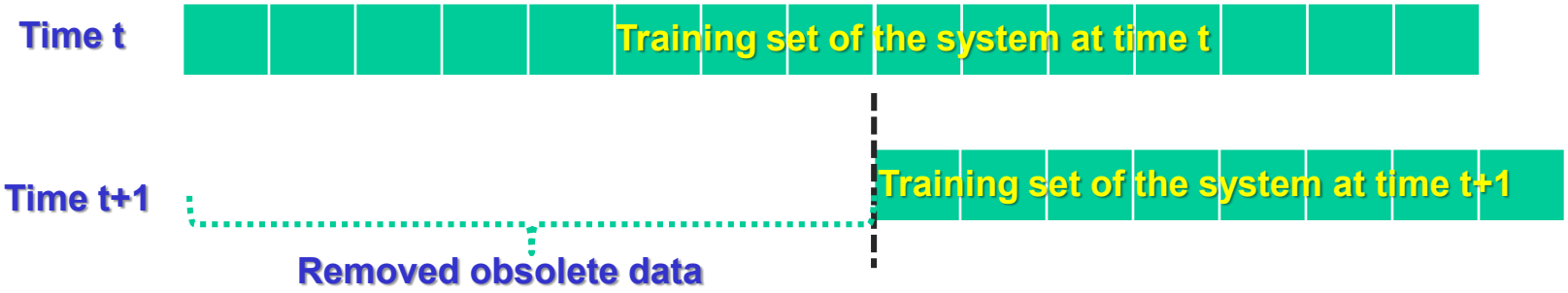
- As available data are not i.i.d. in a nonstationary framework they must be processed with care to generate the model

How to?

- Instance selection
- Instance weighting
- Multiple models (ensembles)

Instance selection

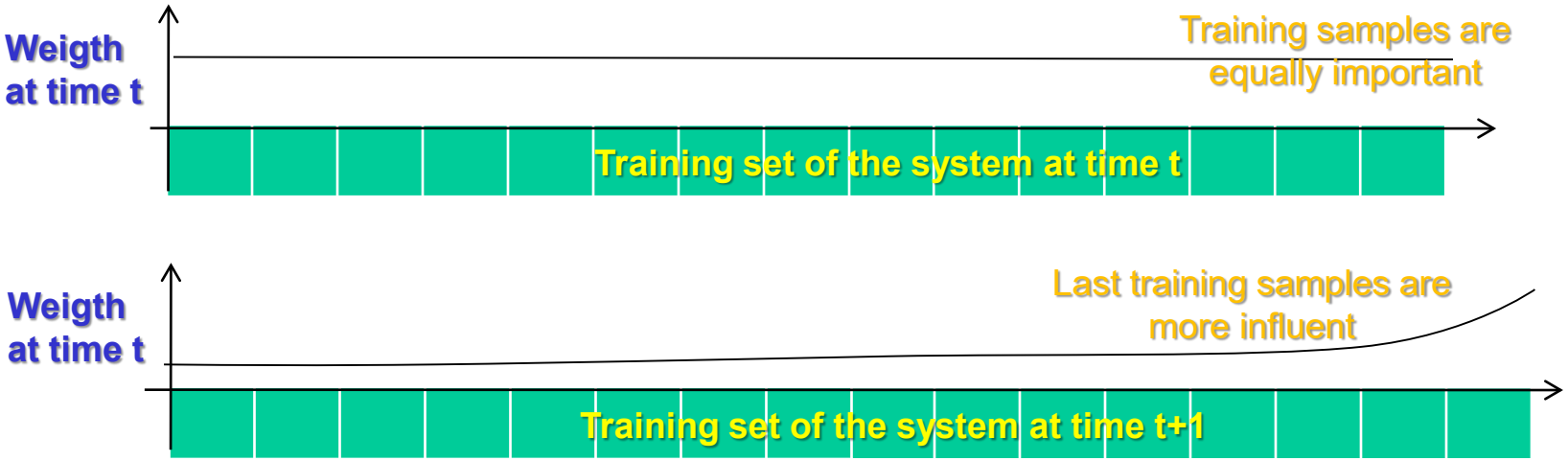
- *Keep data that are relevant to the current state of the process.*



- Many adaptive systems take advantage of most recent training samples
  - **fixed window approach:** the length of the window is fixed a-priori by the user
  - **heuristic approaches:** adapt the window length over the latest samples to maximize pertinence to the new state

Instance weighting

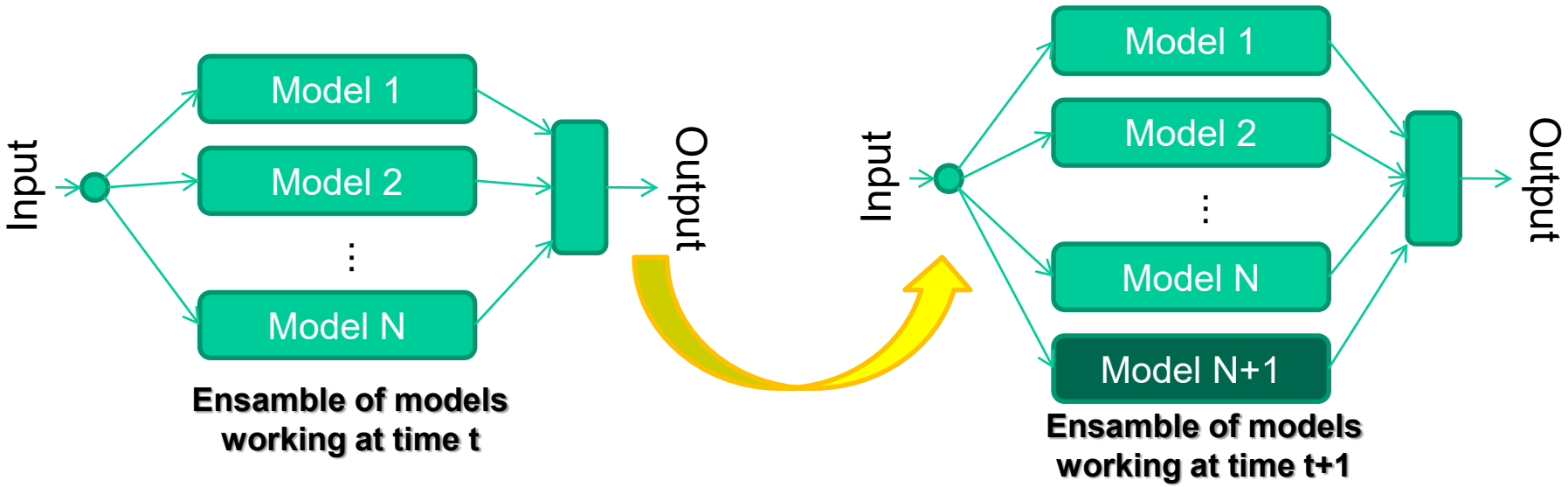
- Old data *are not removed but suitably weighted*



- E.g., weighting depends on
  - Time (age)
  - relevance to the current state

Multiple models

- *Outputs of an ensemble of models are combined by means of voting or weighted mechanisms to form the final output*



- All these systems includes *techniques for dynamically integrating new models in the system and/or deleting obsolete ones* (i.e., pruning techniques aiming at removing the oldest model or the one with the lowest accuracy).

The need for adaptation

Instead of just fully retraining the application from the scratch we can consider incremental approaches, i.e., opting for adaptation mechanisms which depend on (and can adapt to) the change in dynamics

$$\theta_{i+1} = \theta_i - \eta \frac{\partial L\left(y_i, f(\theta, x_i)\right)}{\partial \theta}|_{\theta_i} \quad \Rightarrow \quad f(\hat{\theta}_t, x) \Rightarrow f(\hat{\theta}_{t+1}, x)$$

Passive approach

ProActive approach

Always (compulsive)

When needed (lazy)

When (extremes)

Passive learning

We continuously adapt the application by updating its parameters.

Passive Learning

Online (incremental) learning

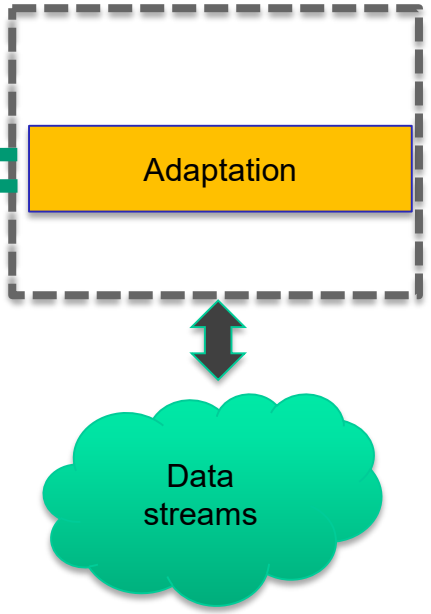
$$V_N(\theta, \{(x_i, y_i)\}) = L(y_i, f(\theta, x_i))$$

$$\theta_{i+1} = \theta_i - \eta \frac{\partial L(y_i, f(\theta, x_i))}{\partial \theta}|_{\theta_i}$$

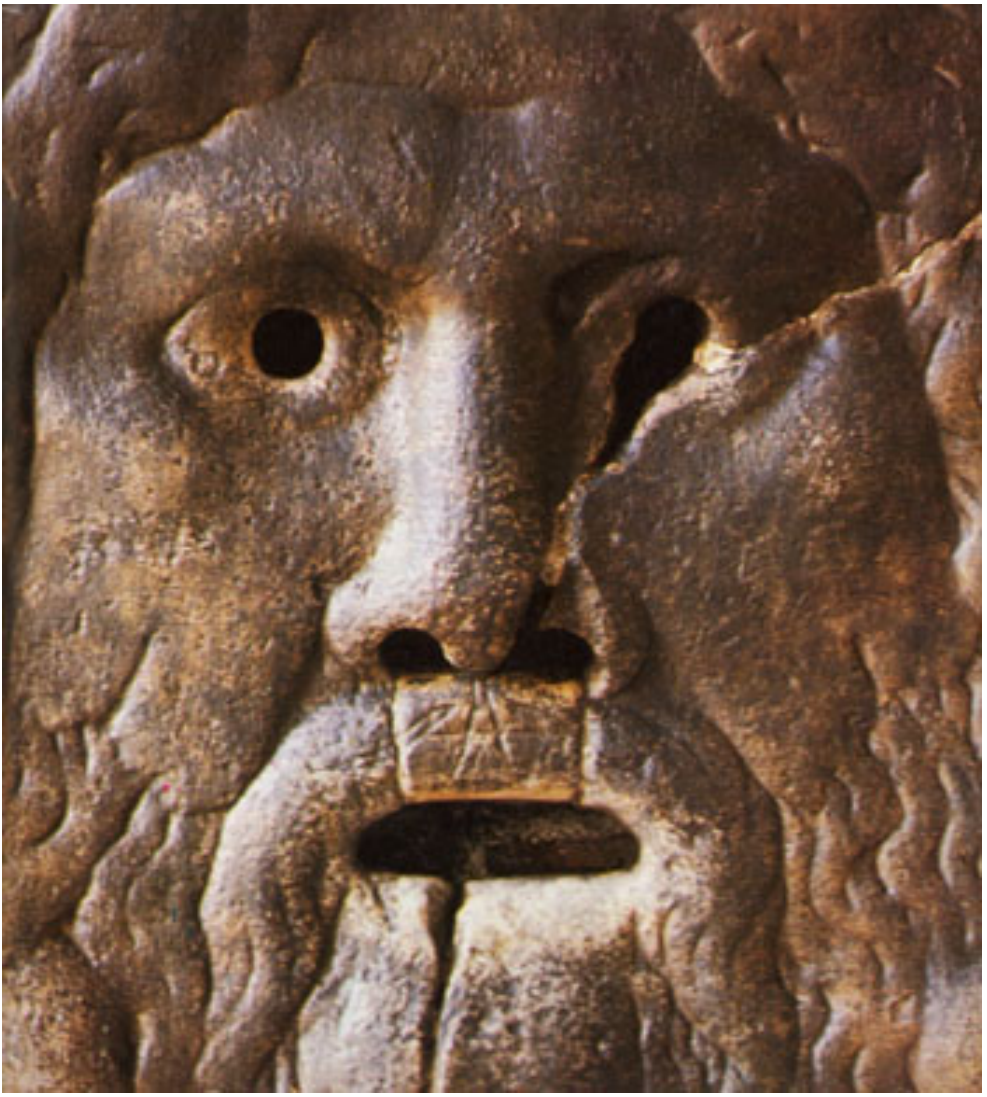
Batch learning

$$Z_{n,i} = \{(x_i, y_i), (x_{i-1}, y_{i-1}), \cdots (x_{i-n+1}, y_{i-n+1})\}$$

$$\theta_{i+1} = \theta_i - \eta \frac{\partial V_N(\theta, Z_{n,i})}{\partial \theta}|_{\theta_i}$$



ProActive Learning



The Oracle detects changes in stationarity

ProActive Learning

Online (incremental) learning

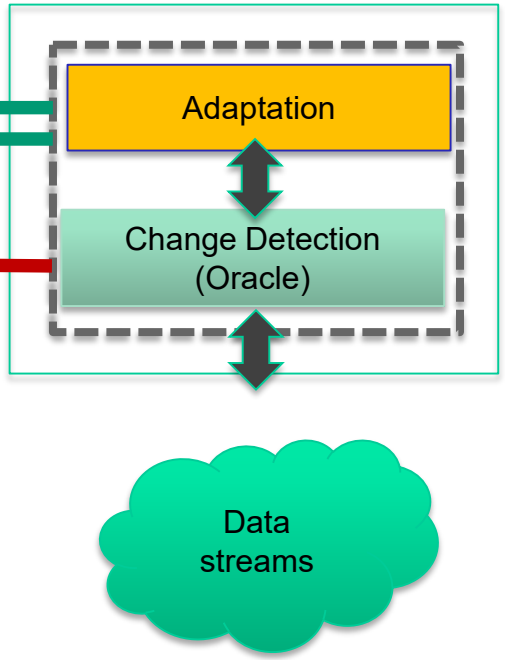
$$V_N(\theta, \{(x_i, y_i)\}) = L(y_i, f(\theta, x_i))$$

$$\theta_{i+1} = \theta_i - \eta \frac{\partial L(y_i, f(\theta, x_i))}{\partial \theta}|_{\theta_i}$$

Batch learning

$$Z_{n,i} = \{(x_i, y_i), (x_{i-1}, y_{i-1}), \cdots (x_{i-n+1}, y_{i-n+1})\}$$

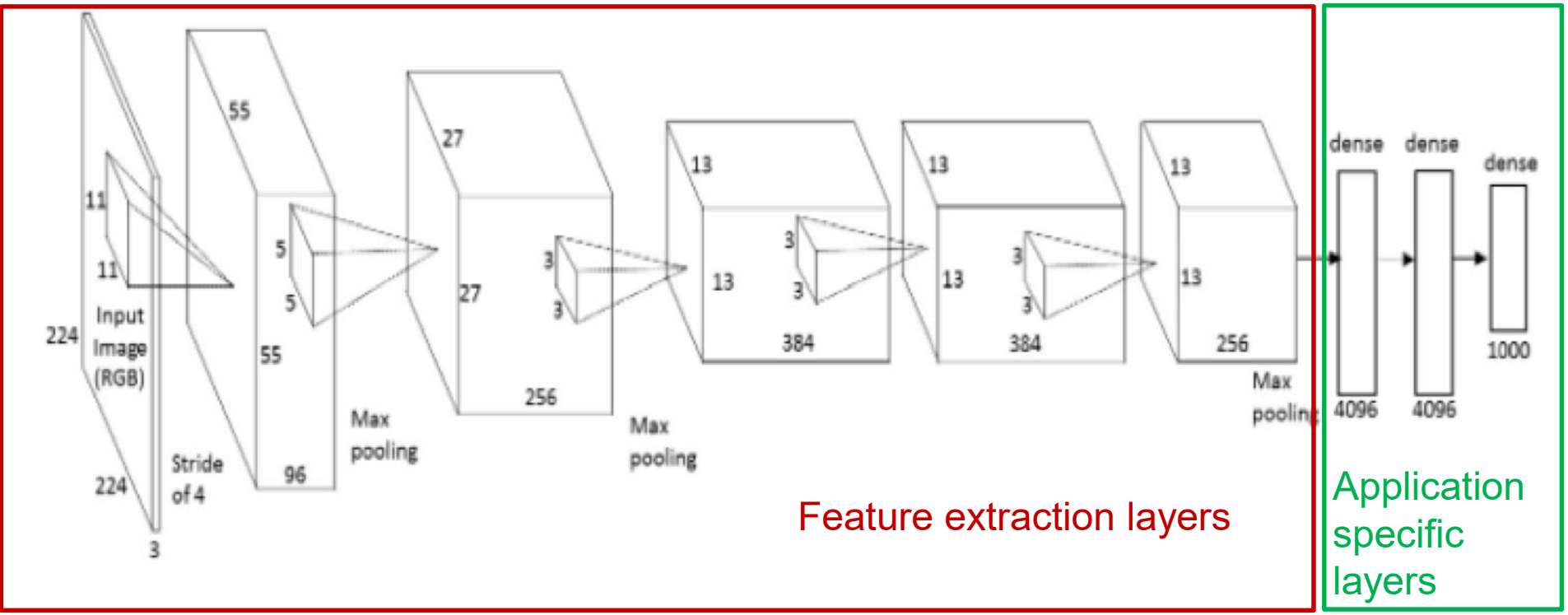
$$\theta_{i+1} = \theta_i - \eta \frac{\partial V_N(\theta, Z_{n,i})}{\partial \theta}|_{\theta_i}$$



What about false negatives/positives?

Update only the last layers

- Adaptation can operate on application specific layer(s) only until performance degenerate (if they do) and a full retraining is needed.
  This is straightforward with the AlexNet (same holds for others)

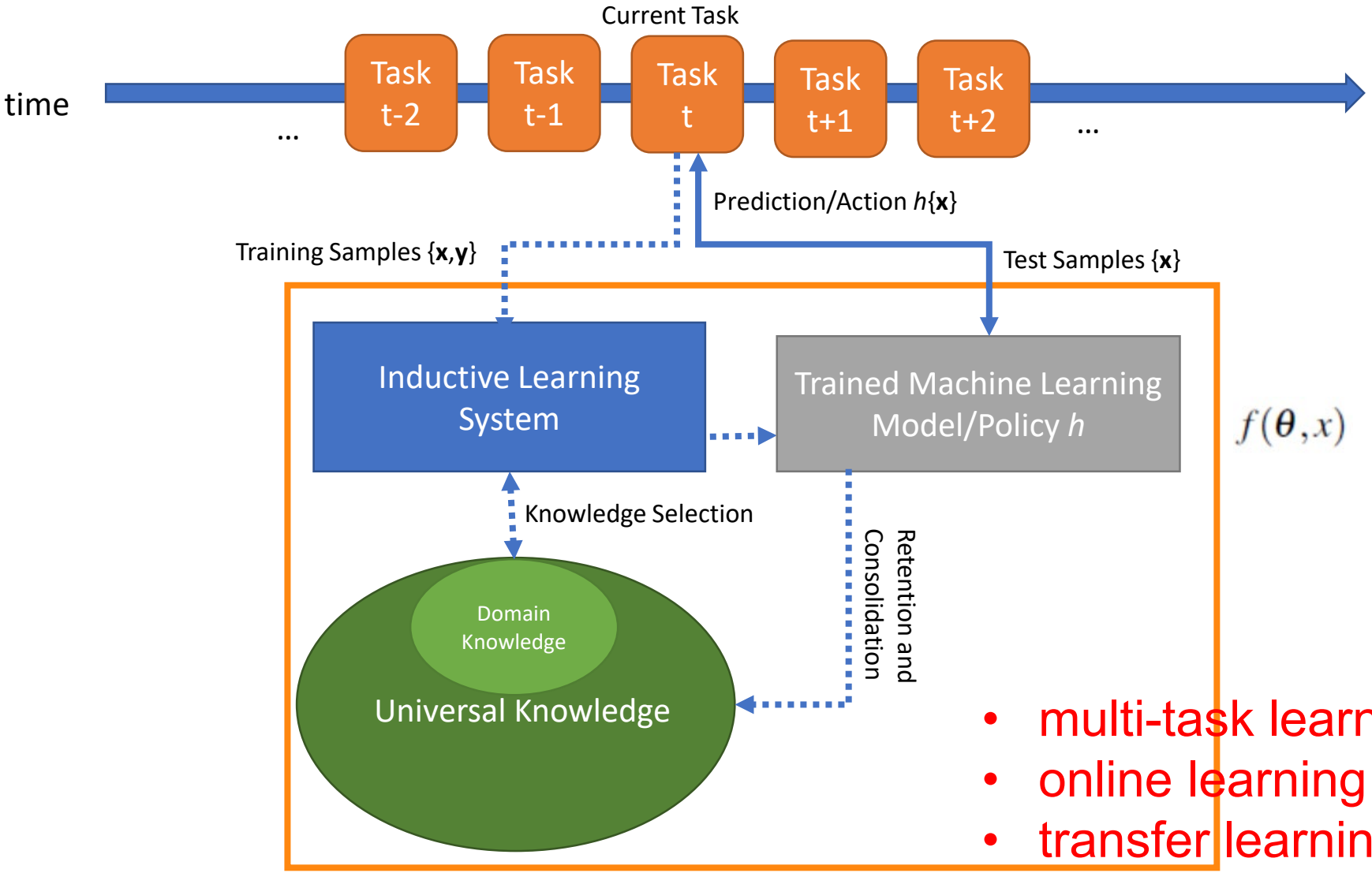


A Lifelong learning perspective

DARPA's definition:
Lifelong Learning Machines are systems that can learn continuously during execution and become increasingly expert while performing tasks, are subject to safety limits, and apply previous skills and knowledge to new situations - without forgetting previous learning.

Key topics

- multi-task learning
- Learning in NSE
- online learning
- transfer learning
- knowledge representation and maintenance.



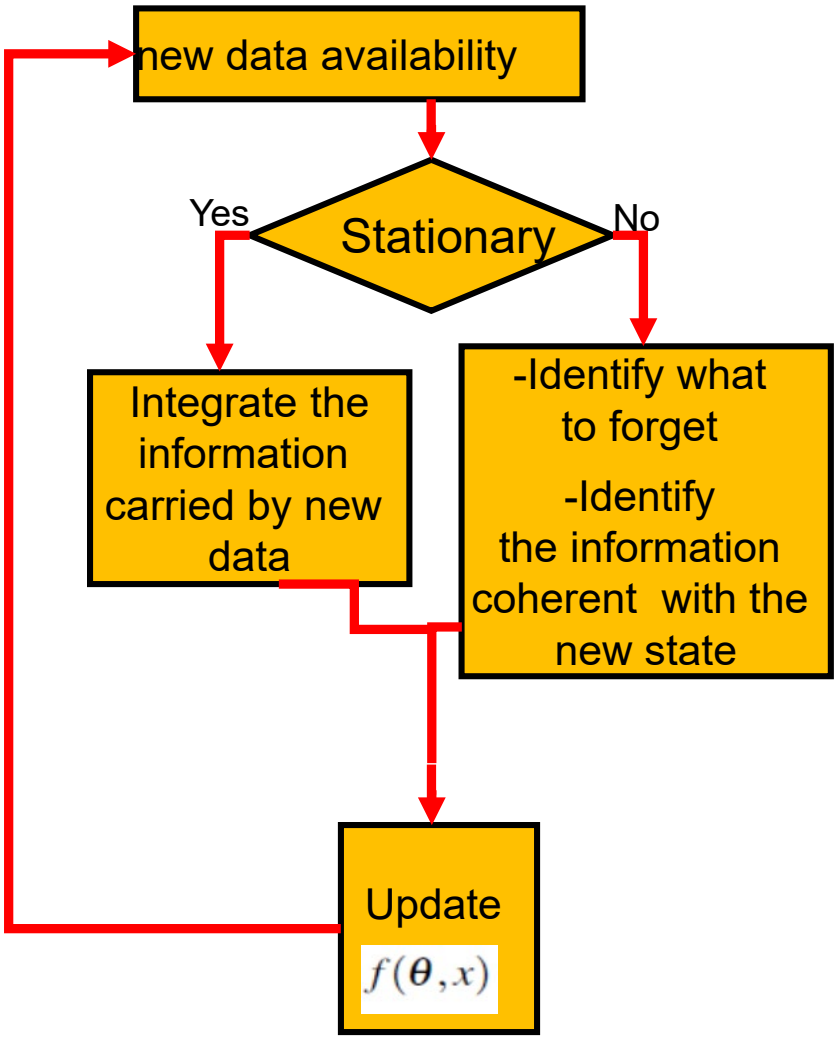
- multi-task learning
- online learning
- transfer learning
- knowledge maintenance.

LIFELONG MACHINE LEARNING SYSTEM

NSE: a Lifelong learning perspective

- *Stationary case:*
  New information is integrated in $f(\theta, x)$ to asymptotically improve the model performance

- *Non-stationary case:*
  Innovation is brought by new data:
  - What to forget?
  - What to learn?



Example: Just-in-Time (JIT) classifiers

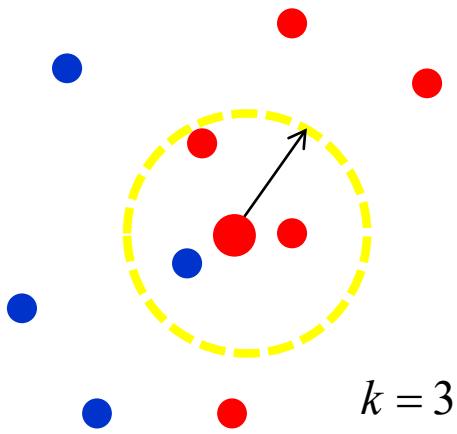
- Apply ProActive learning to classifiers
- the novel information enables weight update

$$\theta_{i+1} = \theta_i - \eta \frac{\partial L\left(y_i, f(\theta, x_i)\right)}{\partial \theta}|_{\theta_i}$$

- The classifier might be inadequate to host the new information content
  - Retrain the classifier
  - Consider incremental light learning strategies, e.g., those based on the k-NN classifier

JIT: Stationary case

- k-NN classifier:
  - ↑ No training phase
  - ↑ ↓ Complexity (Condensing and editing rules)
  - ↓ Selection of *k*
- Optimal value estimation of *k*
  - LOO: computationally expensive
  - Fukunaga: requires the joint (x,y) pdf
- Adaptive estimate of *k*

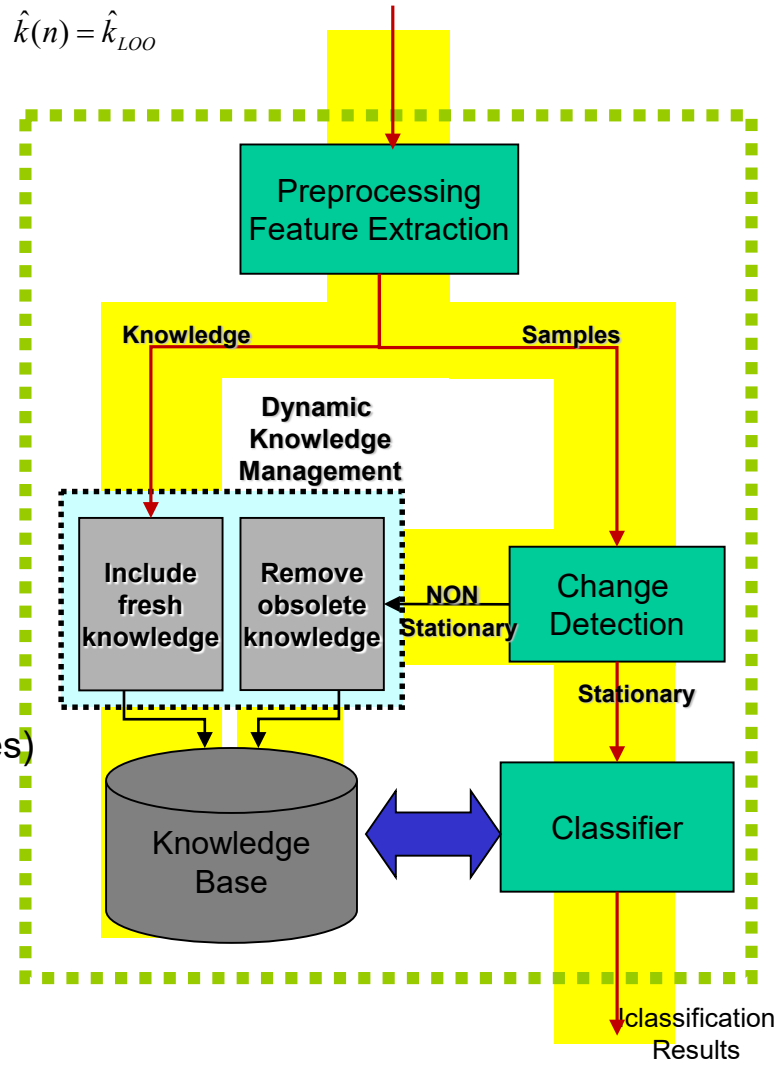


$$\hat{k}(n+\Delta n) = \hat{k}_{LOO}(n)\left(\frac{n+\Delta n}{n}\right)^{\frac{4}{d+4}}$$

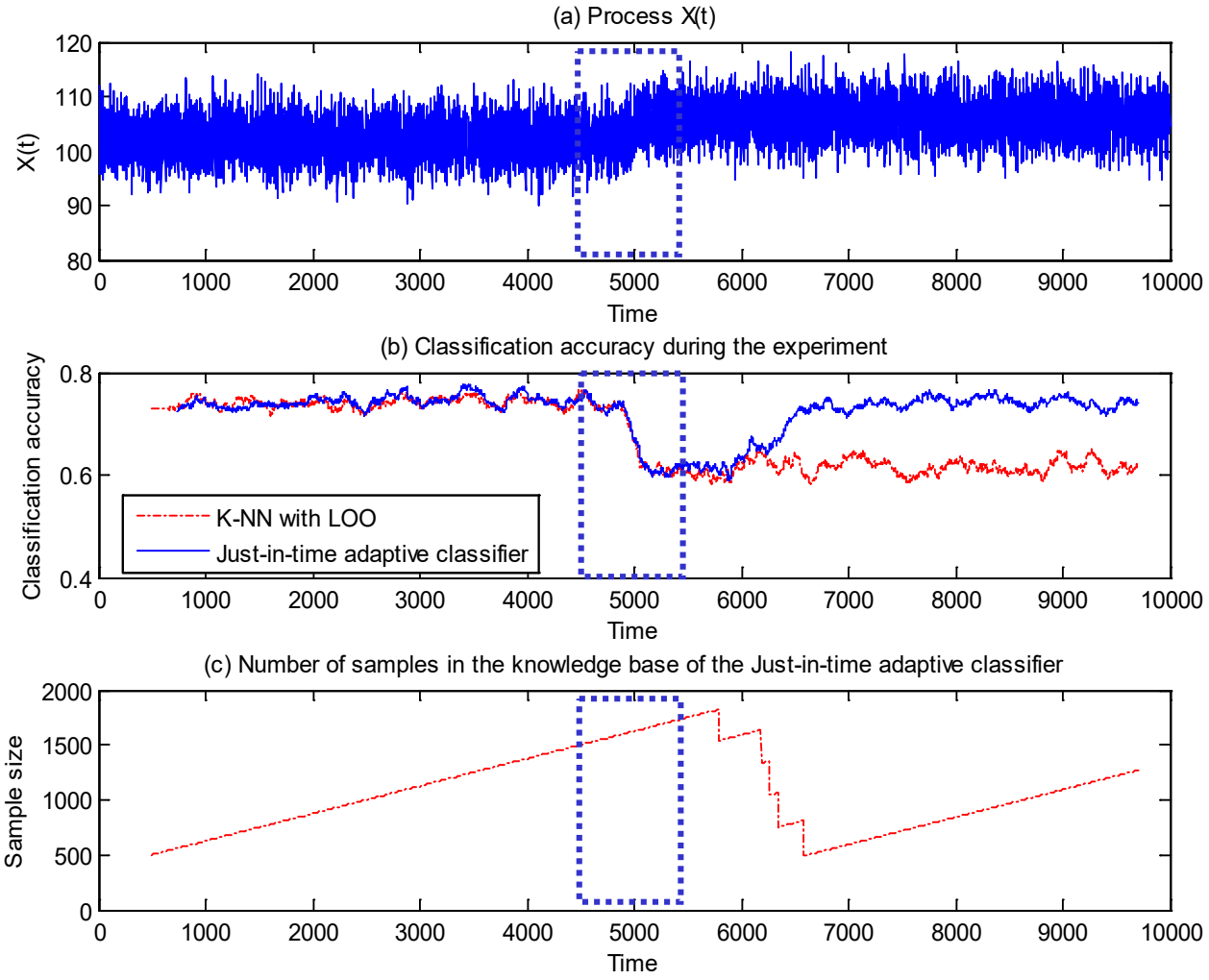
What about if we have other ML architectures?

JIT Classifier

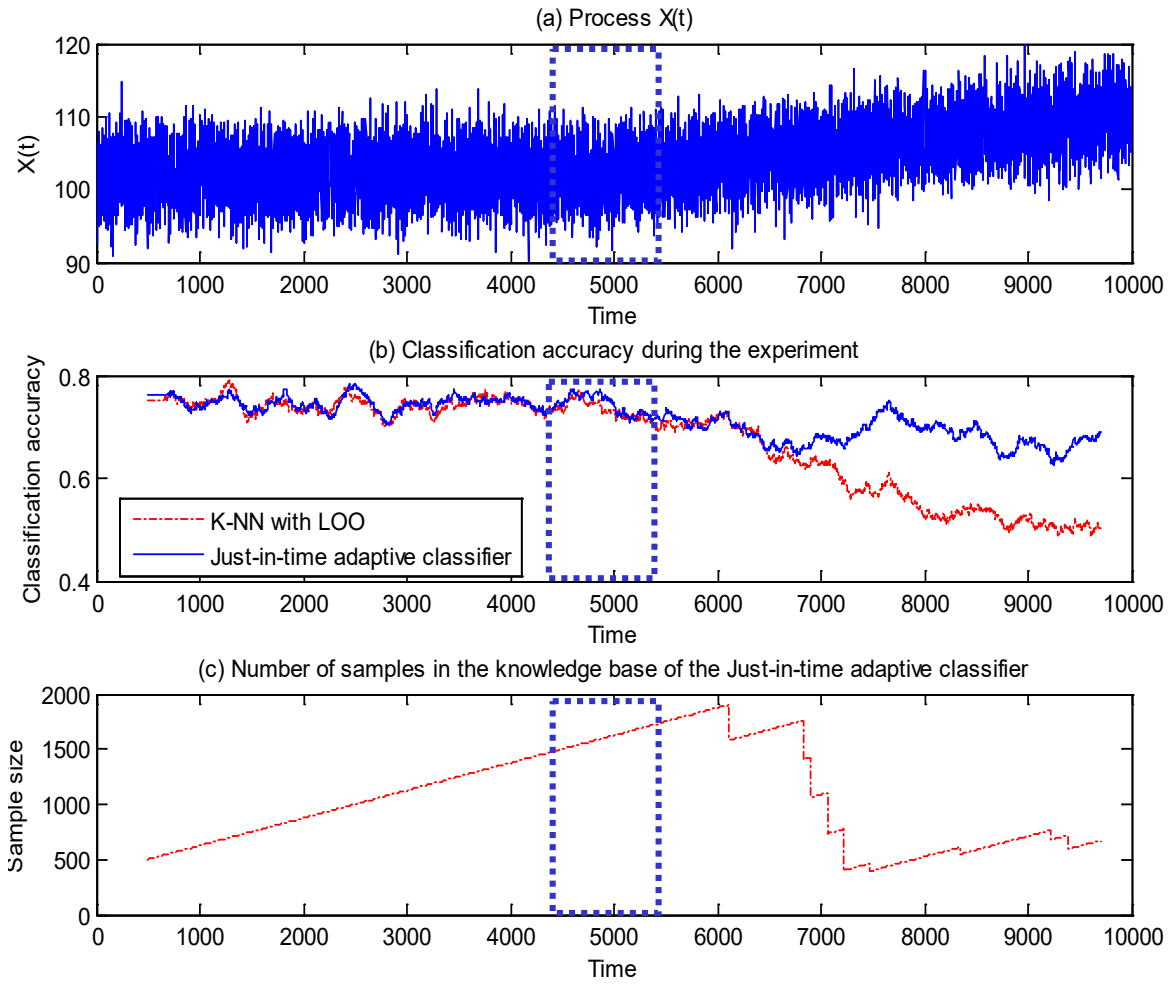
1. Estimate $\hat{k}_{LOO}$ by means of LOO applied to KB_0 and set $\hat{k}(n) = \hat{k}_{LOO}$
2. Configure the k-NN classifier on KB_0
3. Configure the CDT test on KB_0
4. $KB(0) = KB_0$ and $n = |KB_0|$
5. $i = 1$
6. **while (1) do**
7. **if** (new knowledge $IKB(i)$ is available) **then**
8. $KB(i) = Add(KB(i-1), IKB(i))$
9. $\Delta n = |KB(i)| - |KB_0|$; if needed estimate k with LOO
10. $\hat{k}(n+\Delta n) = \hat{k}_{LOO}\left(\frac{n+\Delta n}{n}\right)^{\frac{4}{d+4}}$
11. $i = i+1$
12. **end if**
13. **if** (CDT (sample x)== Stationary) **then**
14. classification=k-NN ($x, KB, \hat{k}(n)$) ;
15. **else**
16. KB_{OBS} = old knowledge (more than the last T samples)
17. $KB(0) = \mathrm{Re}\,m(KB(i), KB_{OBS})$
18. $n = |KB(0)|$
19. Estimate $\hat{k}_{LOO}$ on $KB(0)$ and set $\hat{k}(n) = \hat{k}_{LOO}$
20. Configure the classifier on $KB(0)$
21. Configure the CDT on $KB(0)$
22. $i = 1$ classification=k-NN ($x, KB, \hat{k}(n)$) ;
23. **end if**
24. **end while**



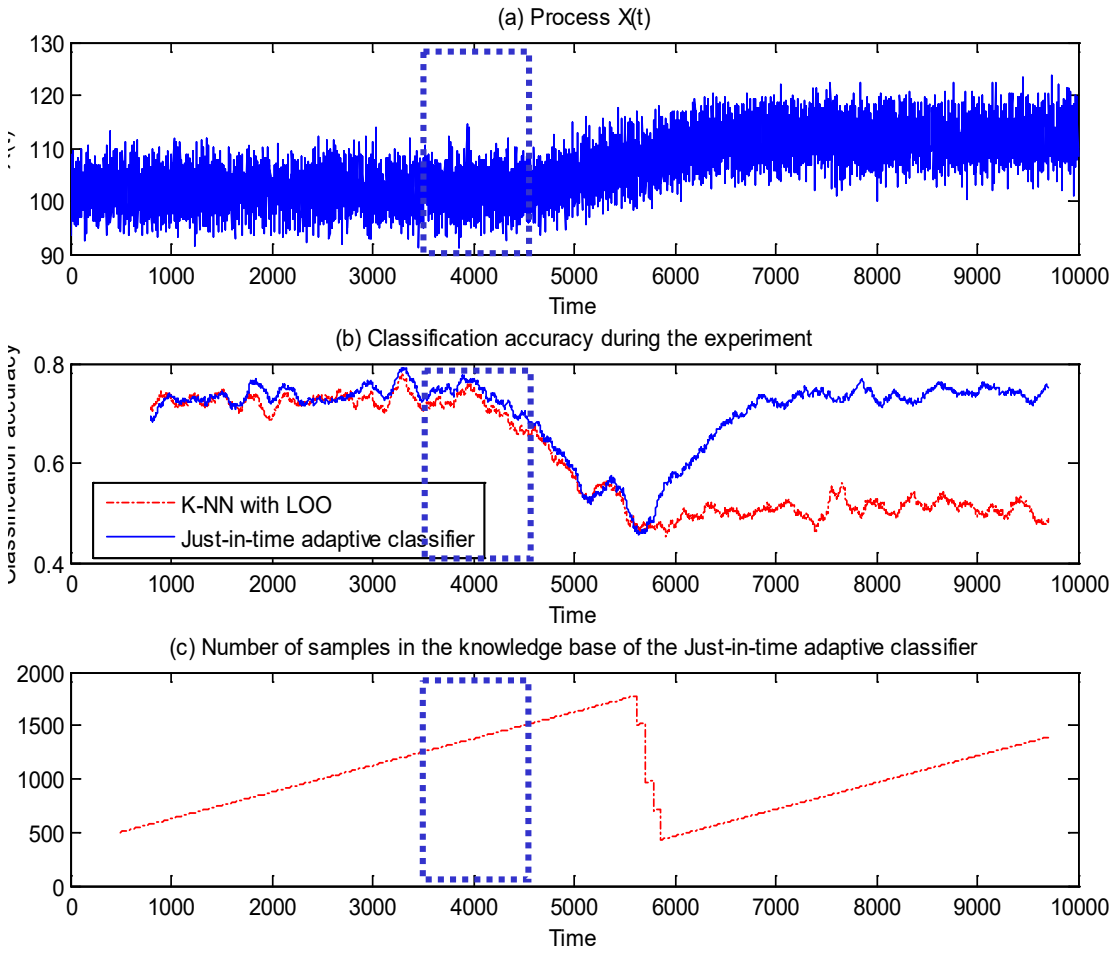
JIT Classifier: Abrupt change



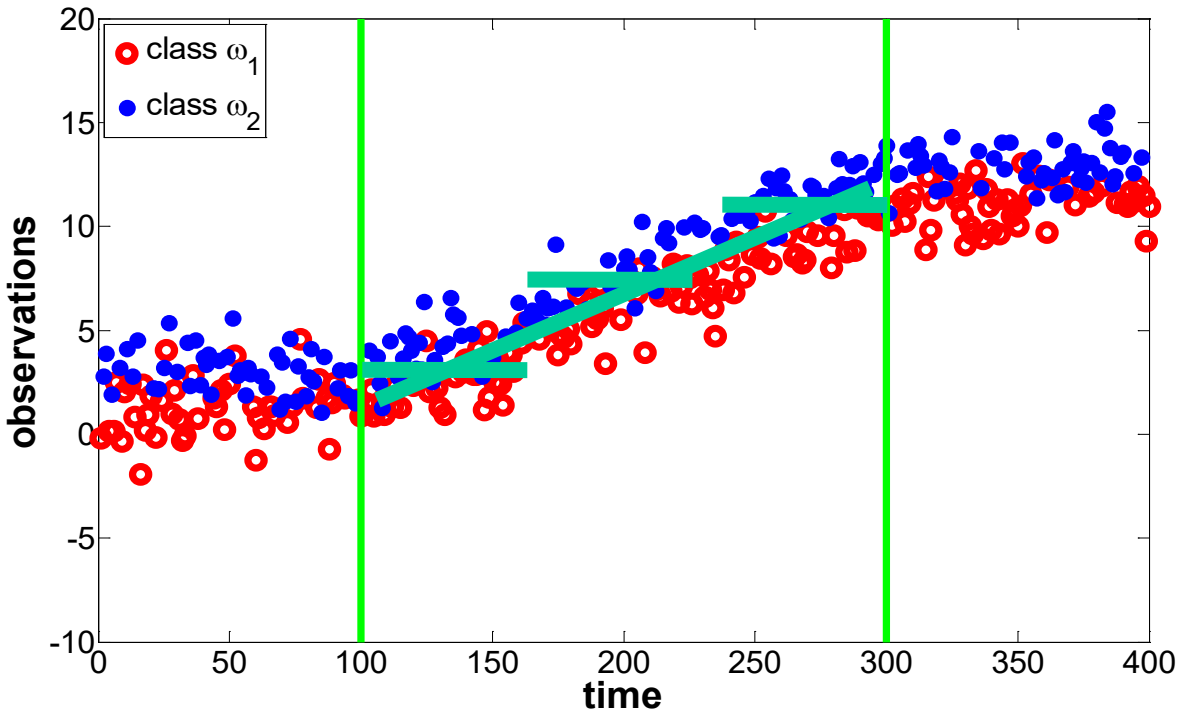
JIT Classifier: drift



JIT Classifier: smooth transition to stationarity



More on drifts…



✓ Gradual concept drift modeled as a **polynomial trend in the expectation** of the conditional probability density function

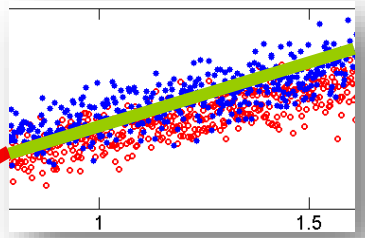
- Drifts are generally seen as a sequence of abrupt changes

Allow the **expectation of the conditional pdf to evolve over time** as a piece-wise polynomial

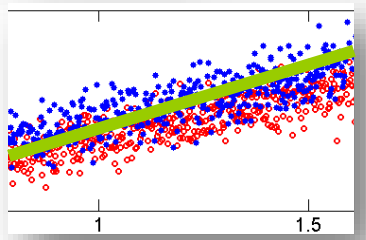
More on drift: the classifier

Algorithm 1: the general JIT Adaptive Classifier for gradual concept drifts

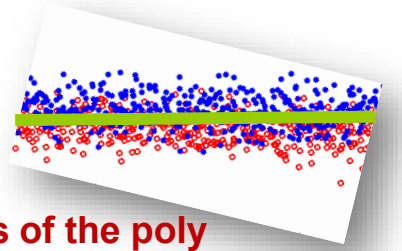
1. Configure the classifier and the change detection test;
2. **while** (1){
3. New observations are received;
4. Solve the polynomial regression problem;
5. **if** (change-detection test detects a change in the process distribution or a change in the expectation trend) {
6. Characterize the new process state;
7. Configure the classifier and the test on the new process state; }
8. **else** integrate the new information (if available) in the knowledge base;
9. Classify the new input samples by exploiting the output of the polynomial regression technique;
10. }



Estimate the parameters of the poly trend



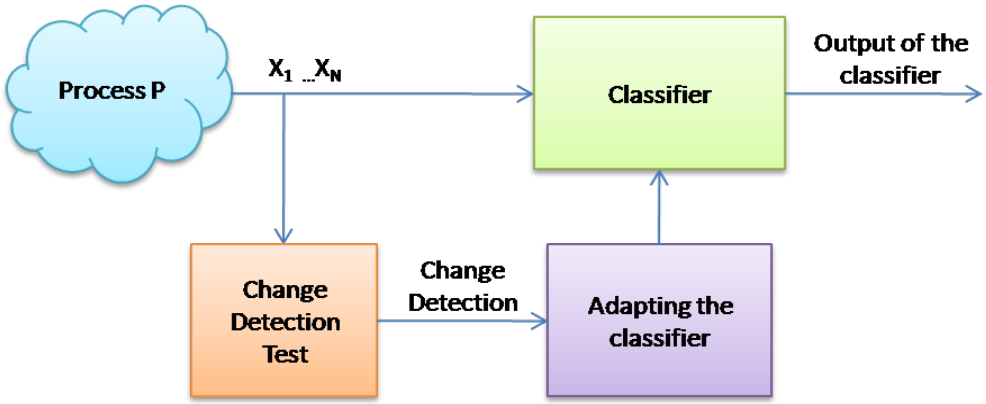
Detect changes



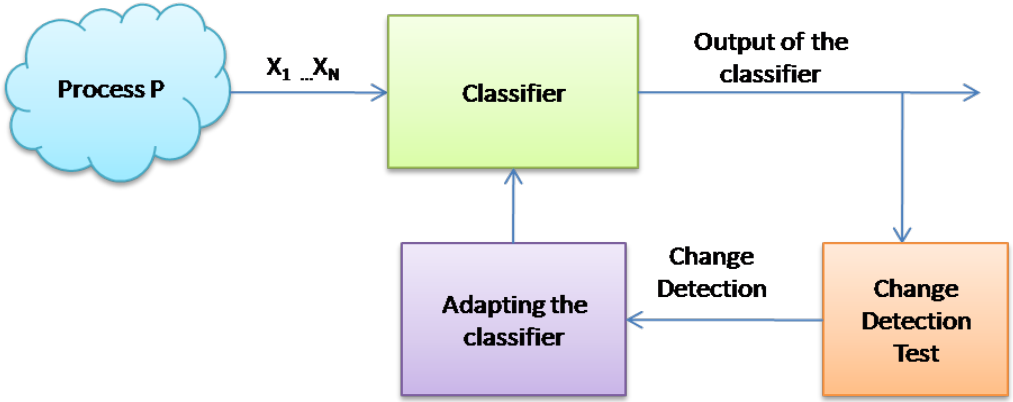
Detrend with the parameters of the poly regression and classification

Where to apply the Oracle, classification case

Change detection on the pdf of the inputs:



Change detection on the classification error



How to design the Oracle (to detect changes)

Ad hoc triggers designed to detect changes by inspecting sequences of data or derived features

- **Feature-based methods**
  - Limit checking
- **Statistical-based methods**
  - Traditional hypothesis tests
  - Change-point methods
  - Change detection tests

Change Point Methods

CPMs inspect a sequence of data and check for concept drift

Given sequence

$$\mathscr{X} = \{x(t), t = 1, \ldots, n\}$$

Produce a generic partitioning

$$\mathscr{A}_\tau = \{x(t),\ t = 1, \ldots, \tau\},$$
$$\mathscr{B}_\tau = \{x(t),\ t = \tau + 1, \ldots, n\}$$

and

$$\tau \text{ is a change point if } x(t) \sim \begin{cases} \mathscr{F}_0, & \text{for } t < \tau \\ \mathscr{F}_1, & \text{for } t \geq \tau \end{cases}$$

In practice

$$\begin{cases} \text{The estimated change-point in } \mathscr{X} \text{ is } M & \text{if } \mathscr{T}_M \geq h_{n,\alpha} \\ \text{No change-point identified in } \mathscr{X}, & \text{if } \mathscr{T}_M < h_{n,\alpha} \end{cases}$$

Change Point Methods

Example

$$x(t) \sim \begin{cases} \mathcal{N}(0,1), & \text{if } t < 350 \\ \mathcal{N}(-1,1), & \text{if } t \geq 350 \end{cases}$$

With hypothesis test

$$H_0 : \forall t,\ x(t) \sim \mathscr{F}_0$$

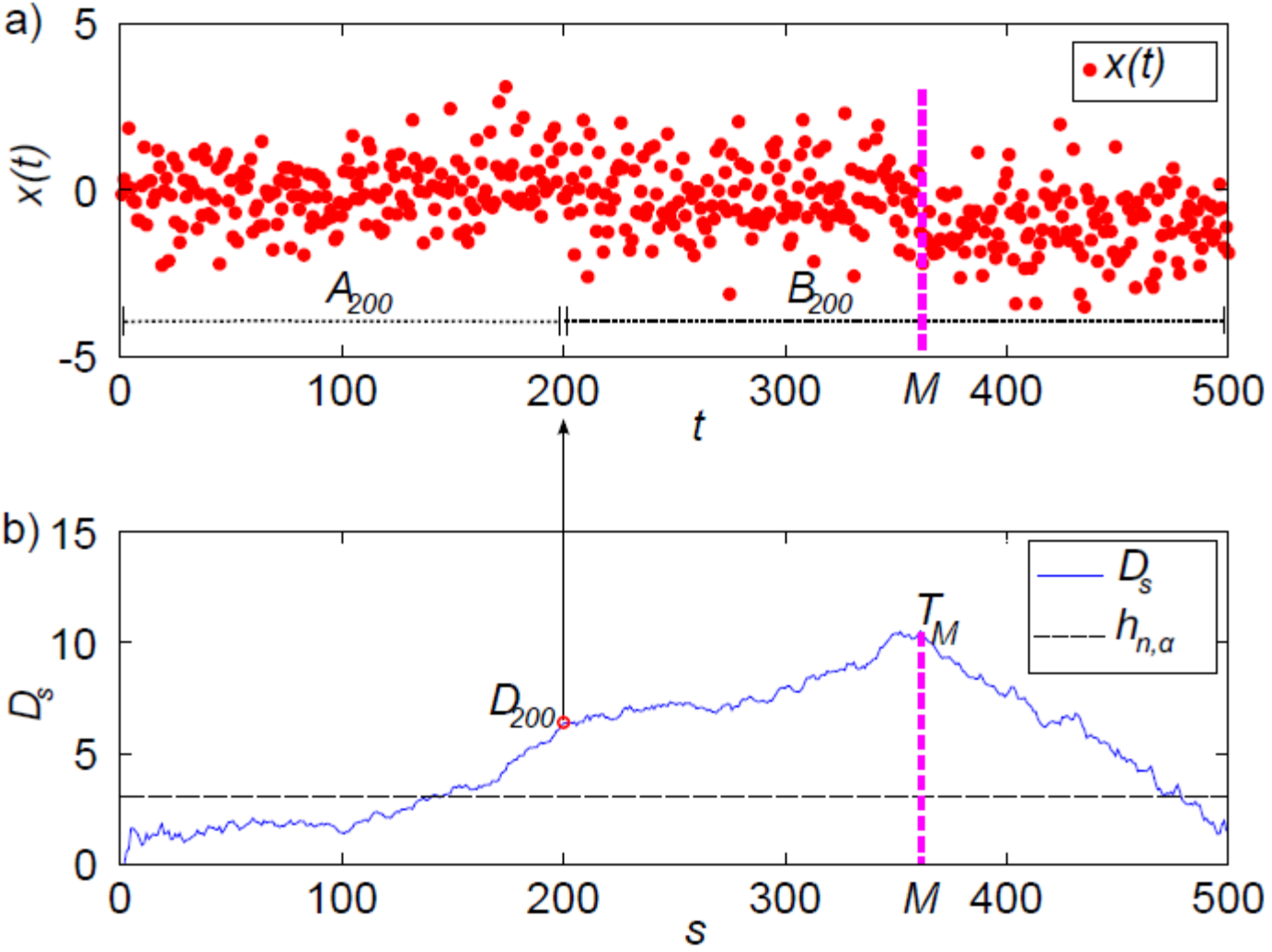
$$H_1 : \exists\ \tau\ x(t) \sim \begin{cases} \mathscr{F}_0, & \text{if } t < \tau \\ \mathscr{F}_1, & \text{if } t \geq \tau \end{cases}$$

For instance consider the Student t statistics for the means

$$D_\tau = \sqrt{\frac{\tau(n-\tau)}{n}}\,\frac{\bar{\mathscr{A}}_\tau - \bar{\mathscr{B}}_\tau}{S_\tau}$$

Change Point Methods

Threshold e.g., $h_{500,0.05} = 3.225$ provided by the CPM package



Change detection CDTε

- We formulate the change-detection procedure on the classification error as an **inference problem on the proportions of two populations**.

$$\epsilon_t = \begin{cases} 1, & \text{if } y_t = K_0(x_t); \\ 0, & \text{otherwise,} \end{cases}$$

$\nu > 20$ ⇒ $\mathcal{B}(p_0, \nu) \sim \mathcal{N}\big(p_0\nu, p_0(1-p_0)\nu\big).$

- The problem becomes **univariate** by evaluating the performance expectation, with the **null hypothesis**

$$\bar{\epsilon}_0 - \bar{\epsilon}_1 = 0$$

$\bar{\epsilon}_0 \Rightarrow [T_{\text{ref}}, \hat{T}]$

$\bar{\epsilon}_1 \Rightarrow [0, T_0]$

Change detection tests CDTx

The Hotelling test

- A multivariate hypothesis test based on the Hotelling T-square statistics

$$S = (\bar{F}^0 - \bar{F}^1)' \left(\left(\frac{1}{n_0} + \frac{1}{n_1} \right) \Sigma \right)^{-1} (\bar{F}^0 - \bar{F}^1),$$

which is distributed as

$$\left(\frac{n_0 + n_1 - 2}{n_0 + n_1 - N - 1} \right) \mathcal{F}(N, n_0 + n_1 - N - 1),$$

- We can test the null hypothesis "the difference between the mean value before and after the estimated change is 0" and possibly reject it at confidence level α.

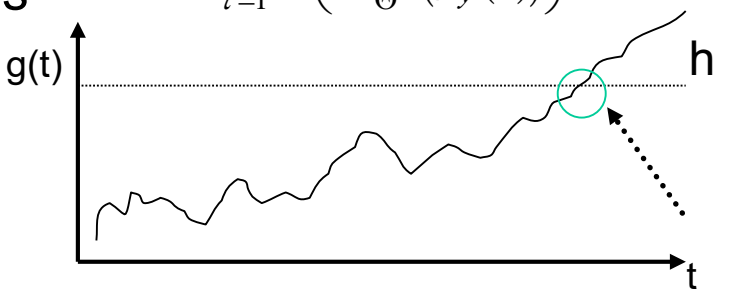
The CI-CUSUM test

Self-configuration procedure

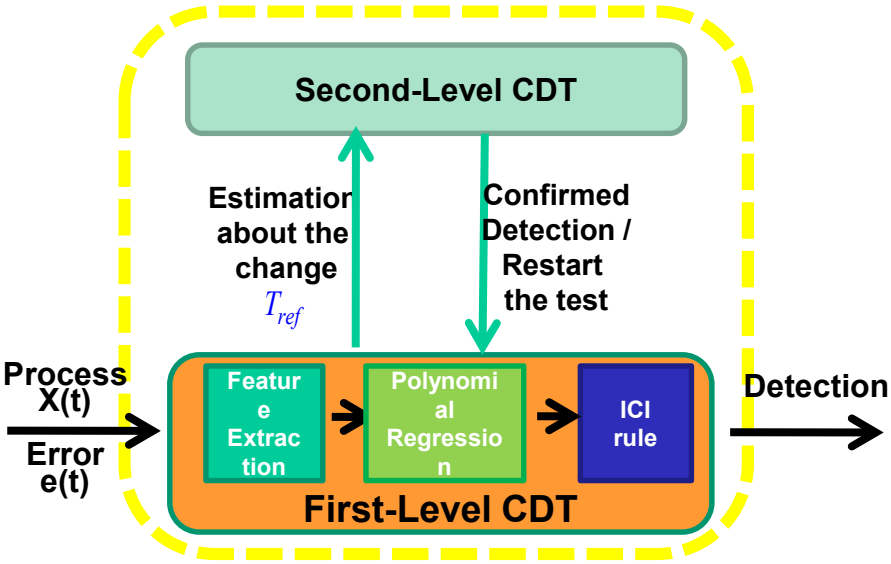
1. Observations $X = \{x(t), t = 1,..,T\}, x(t) \in \Re^d$
2. Partitions of X into disjoint intervals $Y(s) = \{x(t), (v-1)\cdot s \le t < s \cdot v\},$
3. Extract the average feature vector $\varphi_y(s)$ (e.g., mean, var., kur., skew.) from each subsequence $Y(s)$
4. The pdf is gaussian from the central limit theorem
5. Estimate the null hypothesis Θ^0 from $TS = \{\varphi_y(s), s \le s_0\}$
6. Define m alternative hypotheses $\{\Theta^j\}, j = 1,..,m$ as "not being in Θ^0"

Running

7. Measure the discrepancy at time s as $R_j(s) = \sum_{\tau=1}^{s} \ln\left(\frac{N_{\Theta^j}(\varphi_y(\tau))}{N_{\Theta^0}(\varphi_y(\tau))}\right), j = 1,..,m$
8. CI-CUSUM identifies a change at time $\bar{s}$ if $g(\bar{s}) = R_j(\bar{s}) - \min_{1\le\tau\le\bar{s}} R_j(\tau) > h_j$



The hierarchical CDT



Second level change-detection test aiming at confirming (or not) the change hypothesis:

- Multivariate hypothesis test
- Change-point methods

A **multivariate hypothesis** test based on the Hotelling T-square statistics

$$S = (\bar{F}^0 - \bar{F}^1)' \left(\left(\frac{1}{n_0} + \frac{1}{n_1} \right) \Sigma \right)^{-1} (\bar{F}^0 - \bar{F}^1),$$

$$\left(\frac{n_0+n_1-2}{n_0+n_1-N-1} \right) \mathcal{F}(N, n_0 + n_1 - N - 1),$$

Change-point methods: statistical tests able to assess whether a given data-sequence contains (or not) a change point

$$\mathscr{X} = \{x(t), t = 1, \ldots, n\}, \begin{cases} \mathscr{A}_\tau = \{x(t),\ t = 1, \ldots, \tau\}, \\ \mathscr{B}_\tau = \{x(t),\ t = \tau+1, \ldots, n\}, \end{cases}$$

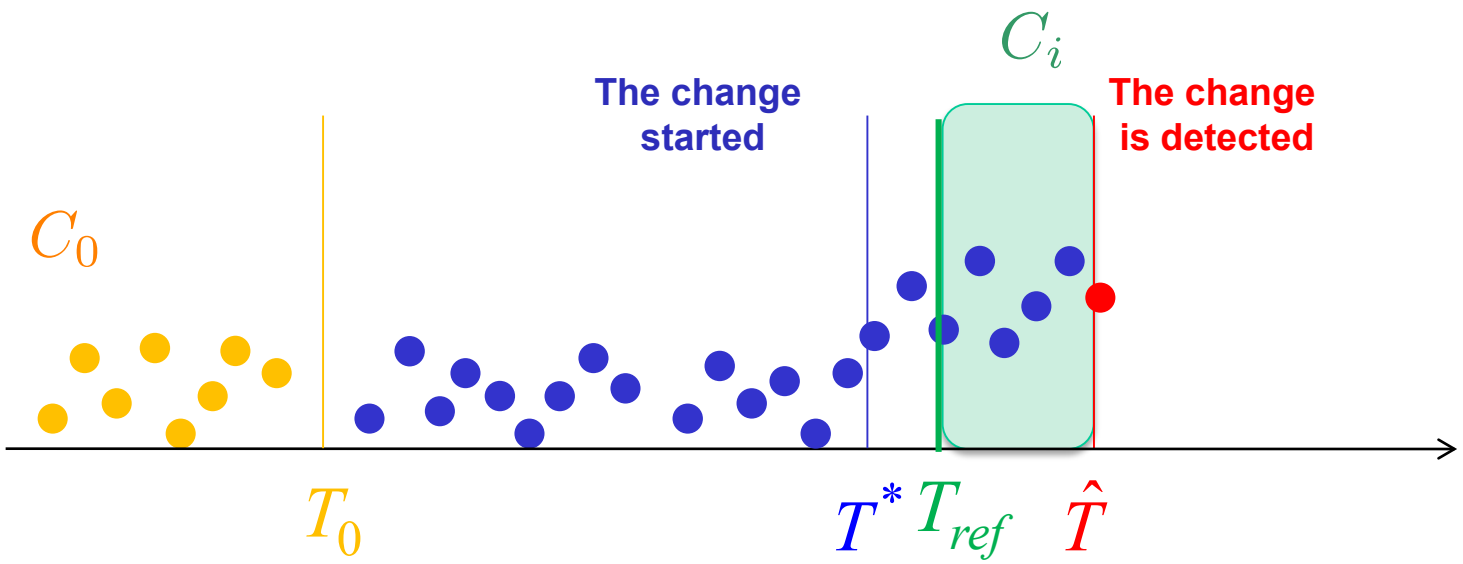
Compute $\mathscr{T}_\tau = \mathscr{T}(\mathscr{A}_\tau, \mathscr{B}_\tau),$

$$\mathscr{T}_M = \max_{s=1,\ldots,n} (\mathscr{T}_\tau)$$

$$\begin{cases} \text{The estimated change-point in } \mathscr{X} \text{ is } M_{\mathscr{X}} & \text{if } \mathscr{T}_M \geq h_{n,\alpha} \\ \text{No change-point identified in } \mathscr{X}, & \text{if } \mathscr{T}_M < h_{n,\alpha} \end{cases}$$

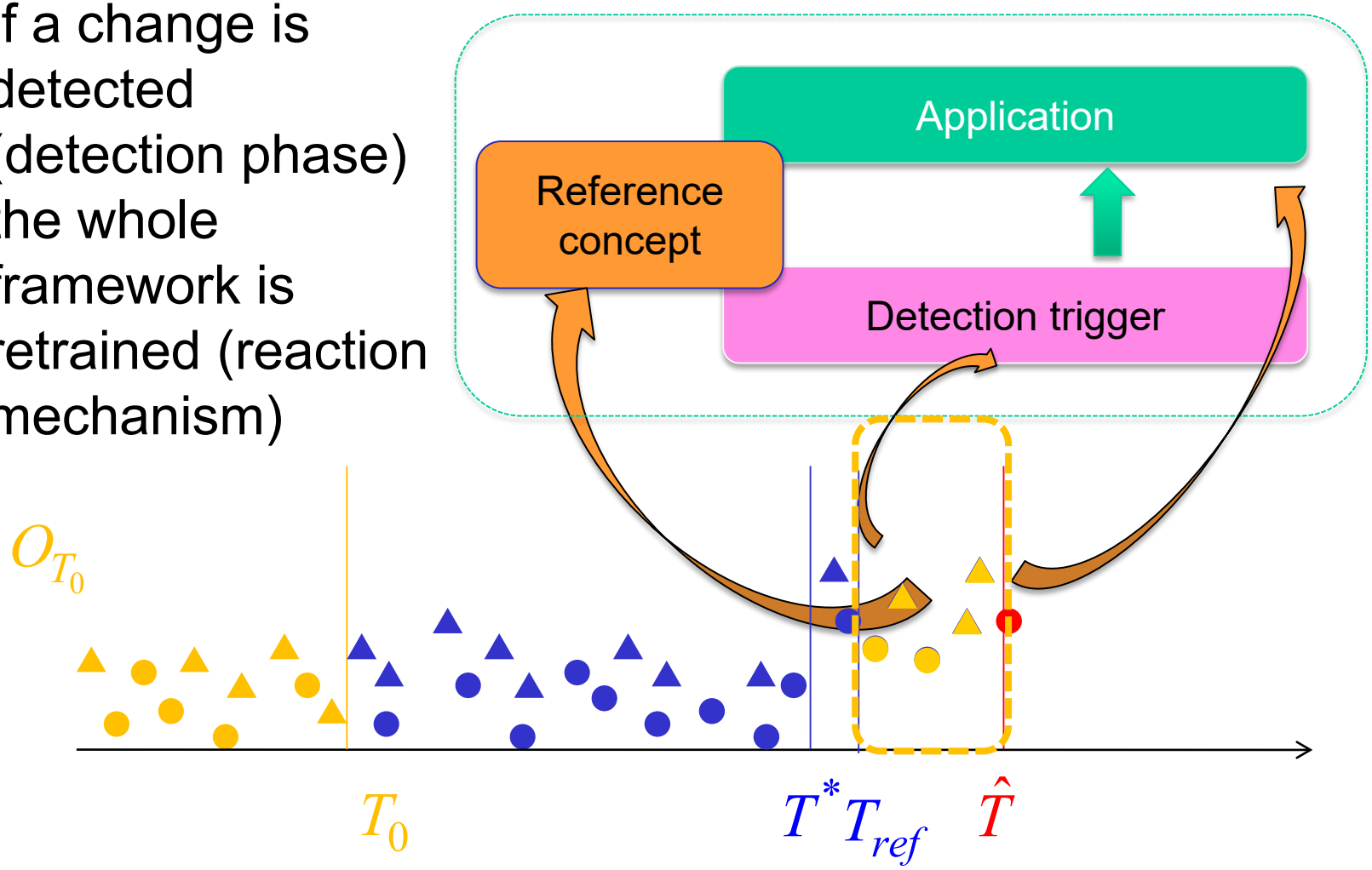
Detecting the change

- Estimate the change time instant $\hat{T}$
- Improve the detection estimate with T_{ref}

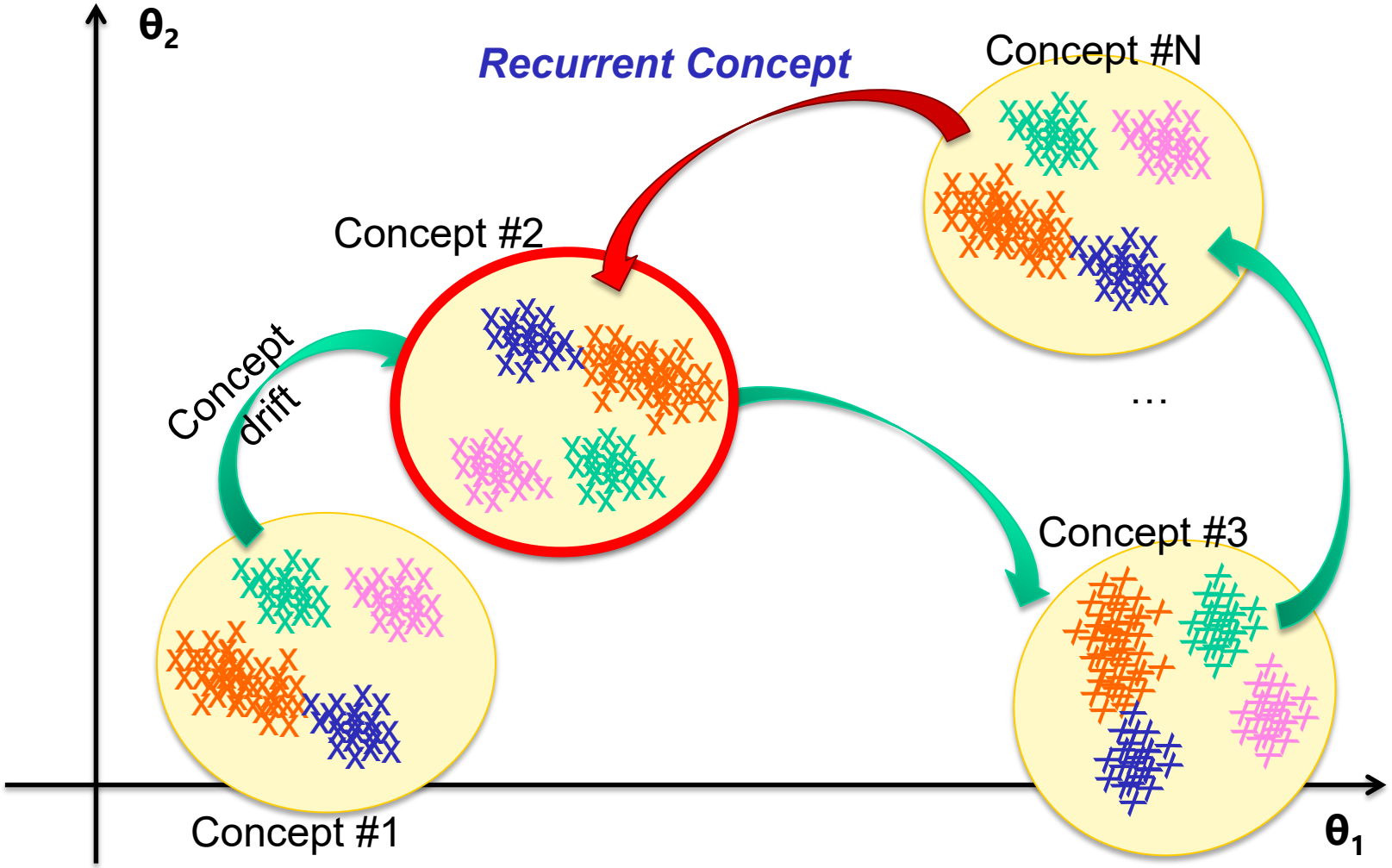


The reaction mechanism

- If a change is detected (detection phase) the whole framework is retrained (reaction mechanism)



Recurrent concepts



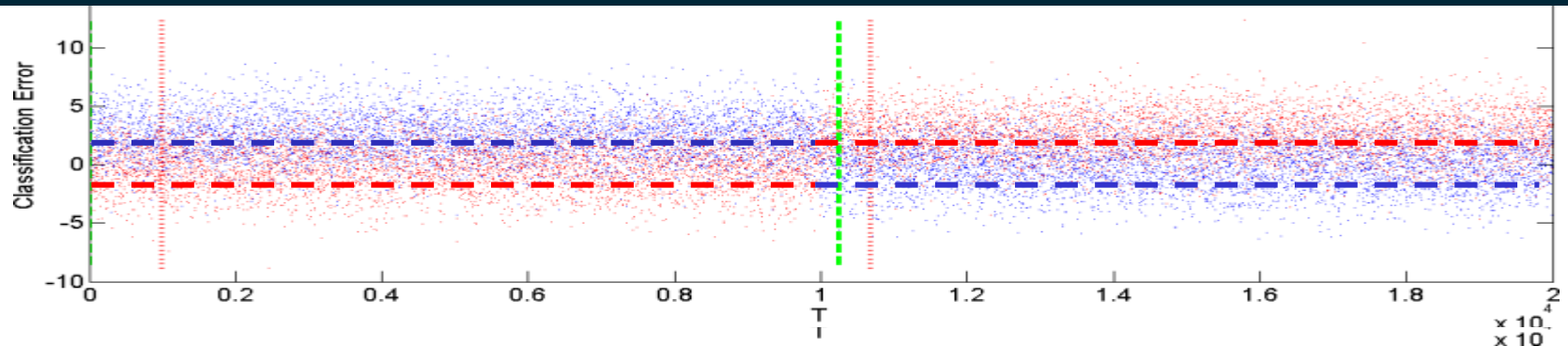
Identifying recurrent concepts

- To compare two concepts C_i and C_j , we verify if both the average of the features $\overline{F_i}, \overline{F_j}$ and the classification errors computed in $[T_{\text{ref},i}, \hat{T}_i]$ correspond to those computed in $[T_{\text{ref},j}, \hat{T}_j]$

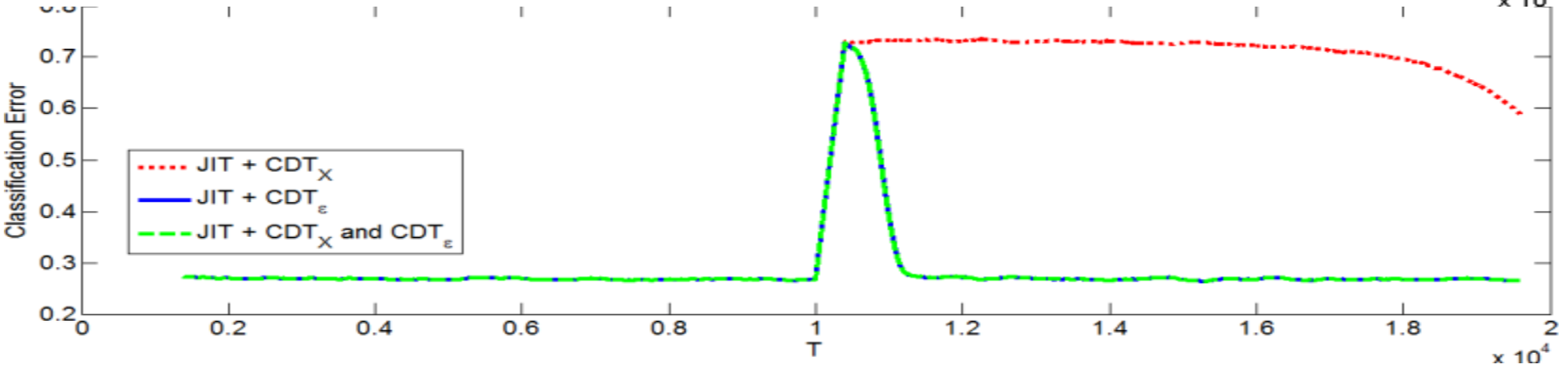
$$\begin{cases} \dfrac{\|\overline{F_i} - \overline{F_j}\|_2}{\|\overline{F_i}\|_2 + \|\overline{F_j}\|_2} < \gamma, \\ \dfrac{|\overline{\epsilon}_i - \overline{\epsilon}_j|}{\overline{\epsilon}_i + \overline{\epsilon}_j} < \gamma \end{cases}$$

where $\gamma \in [0, 1]$ is a tuning parameter

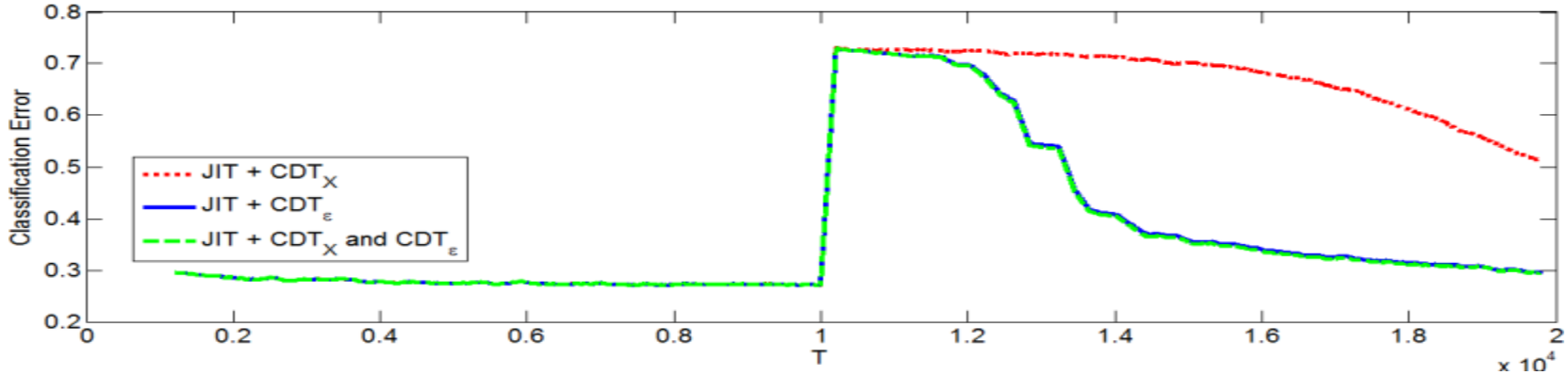
Gaussian scenario: classes' swap



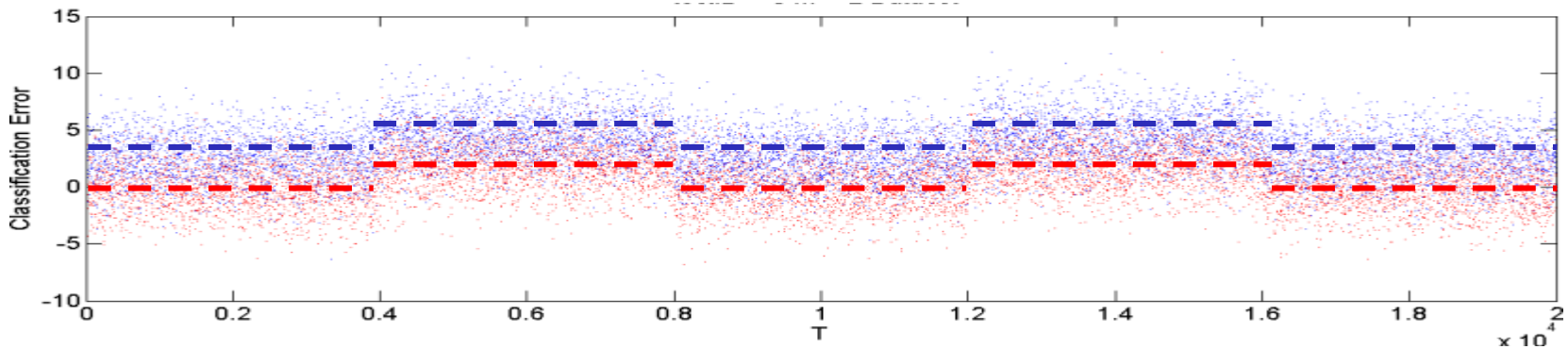
m=2



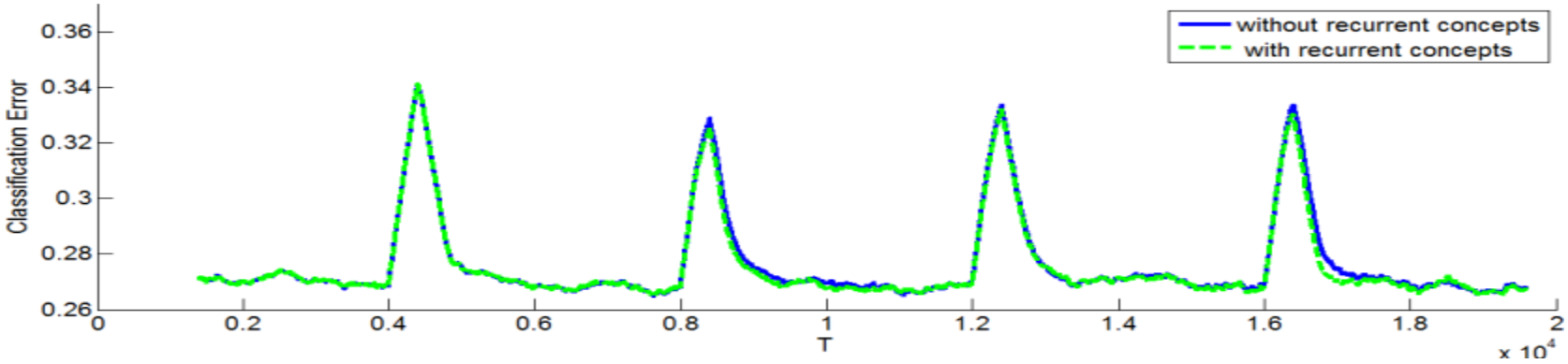
m=20



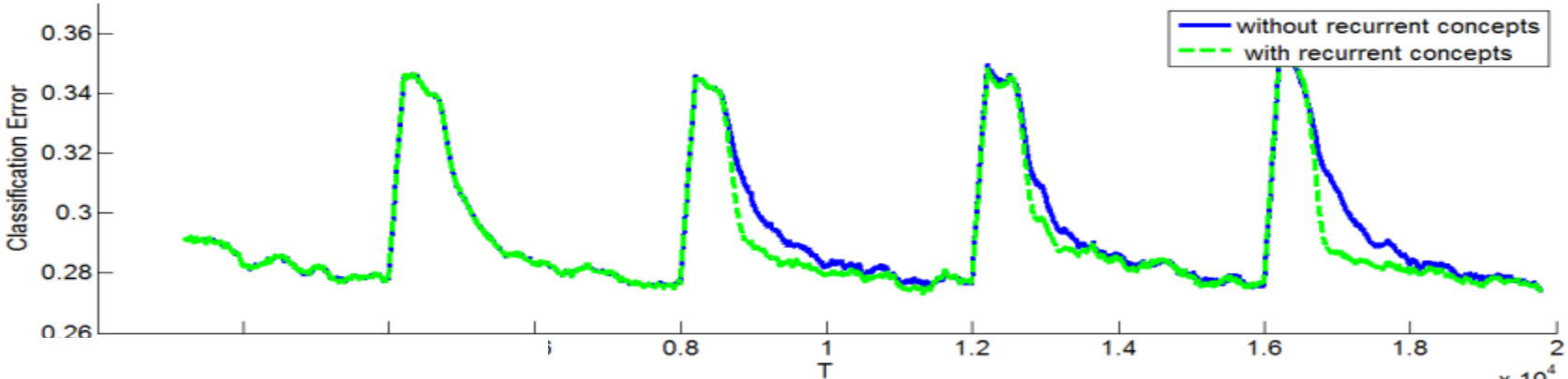
Example: Recurrent Transient Change



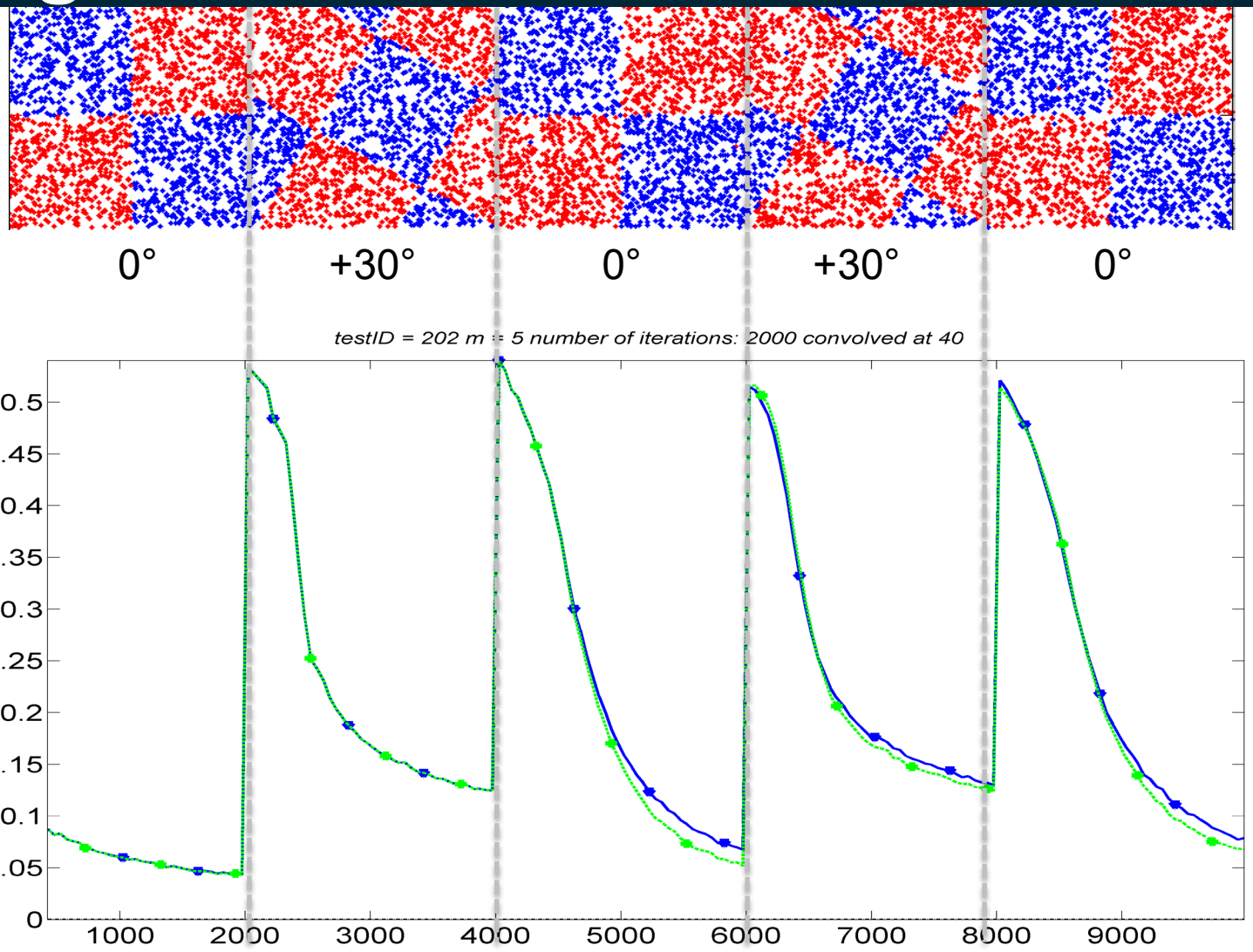
m=2



m=20



Gaussian scenario: Recurrent Transient Change



Conclusions

- The stationary assumption acts as a first order hypothesis of reality
- In lifelong learning the application, decisions and reactions evolve with time
- Recurrence in concepts is fundamental
- Lifelong (nonstationary) learning is a key research issue in machine learning

Principal references

- L.Bu, D.Zhao, C.Alippi, An Incremental Change Detection Test Based on Density Difference Estimation, IEEE Trans on Systems, Man and Cybernetics: Systems, pp 2714 – 2726, Vol 47, No 10, 2017
- C. Alippi, G. Boracchi, M. Roveri, Hierarchical Change-Detection Tests, IEEE Transactions on Neural Networks and Learning Systems, Vol. 28, No. 2, pp 246–258, 2017
- L. Bo, C.Alippi, D.Zhao, A Pdf-free Change Detection Test Based on Density Difference Estimation, IEEE Transactions on Neural Networks and Learning Systems, Vol 29, No 2, 2018, pp 324-334
- G. Ditzler, M. Roveri, C. Alippi, R. Polikar, Adaptive Strategies for Learning in Nonstationary Environments: a Survey, IEEE Computational Intelligence Mag., vol. 10, no. 4, pp. 12-25, 2015
- C.Alippi. M.Roveri. F.Trovo', A Self-building and Cluster-based Cognitive Fault Diagnosis System for Sensor Networks, IEEE Transactions on Neural Networks and Learning Systems, Vol. 25, No.6, pp. 1021-1032, June 2014
- C.Alippi, D.Liu, D.Zhao, L.Bu, Detecting and Reacting to Changes in Sensing Units: the Active Classifier Case, IEEE Trans. on System, Man, Cybernetics: Systems, Vol. 44, No. 3, pp.353-362, 2014