LEARNING IN NONSTATIONARY ENVIRONMENTS: PERSPECTIVES AND APPLICATIONS

Giacomo Boracchi[1] and Gregory Ditzler[2]

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

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

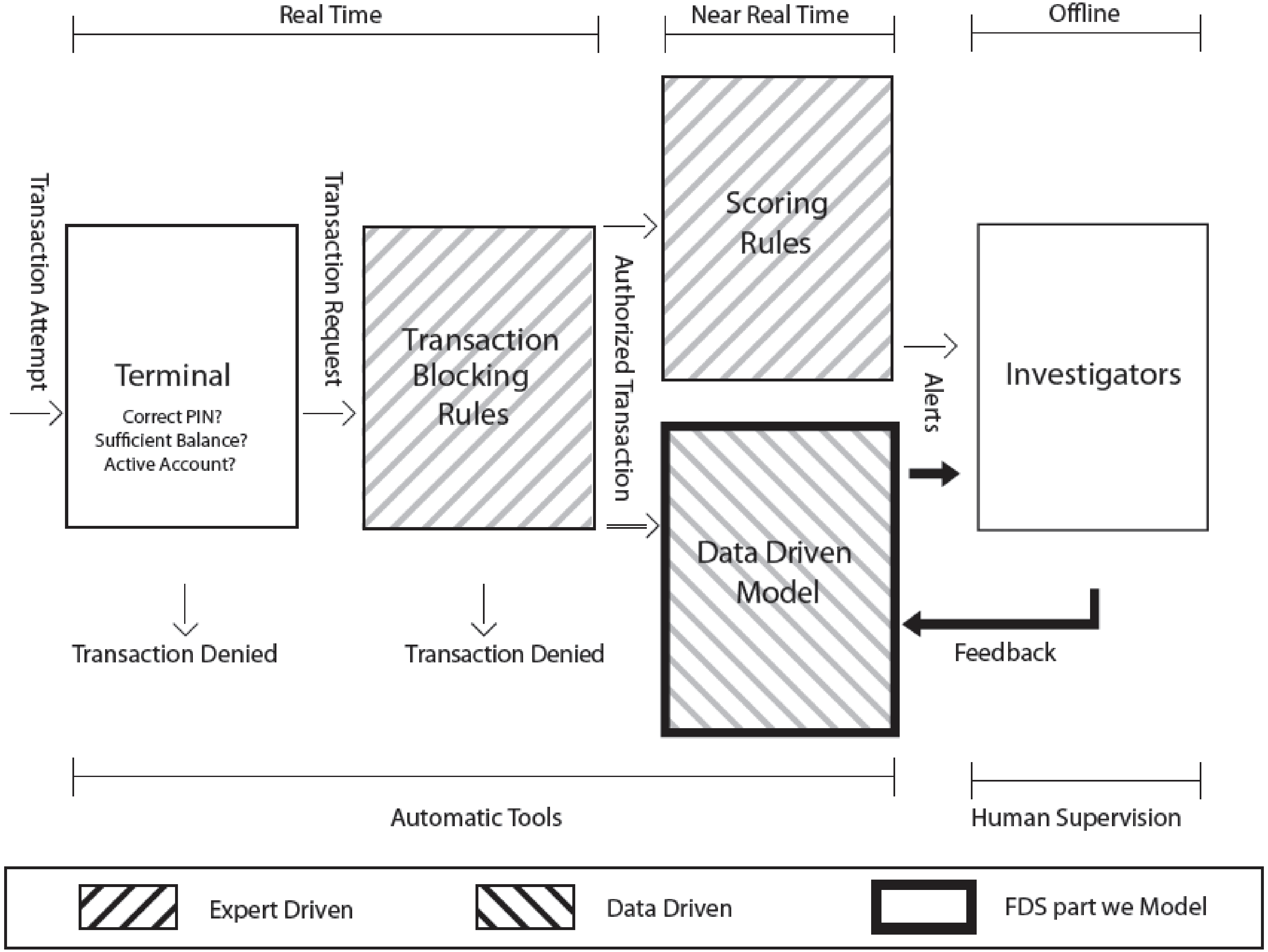
giacomo.boracchi@polimi.it, ditzler@email.arizona.edu

AN INTRODUCTORY EXAMPLE

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

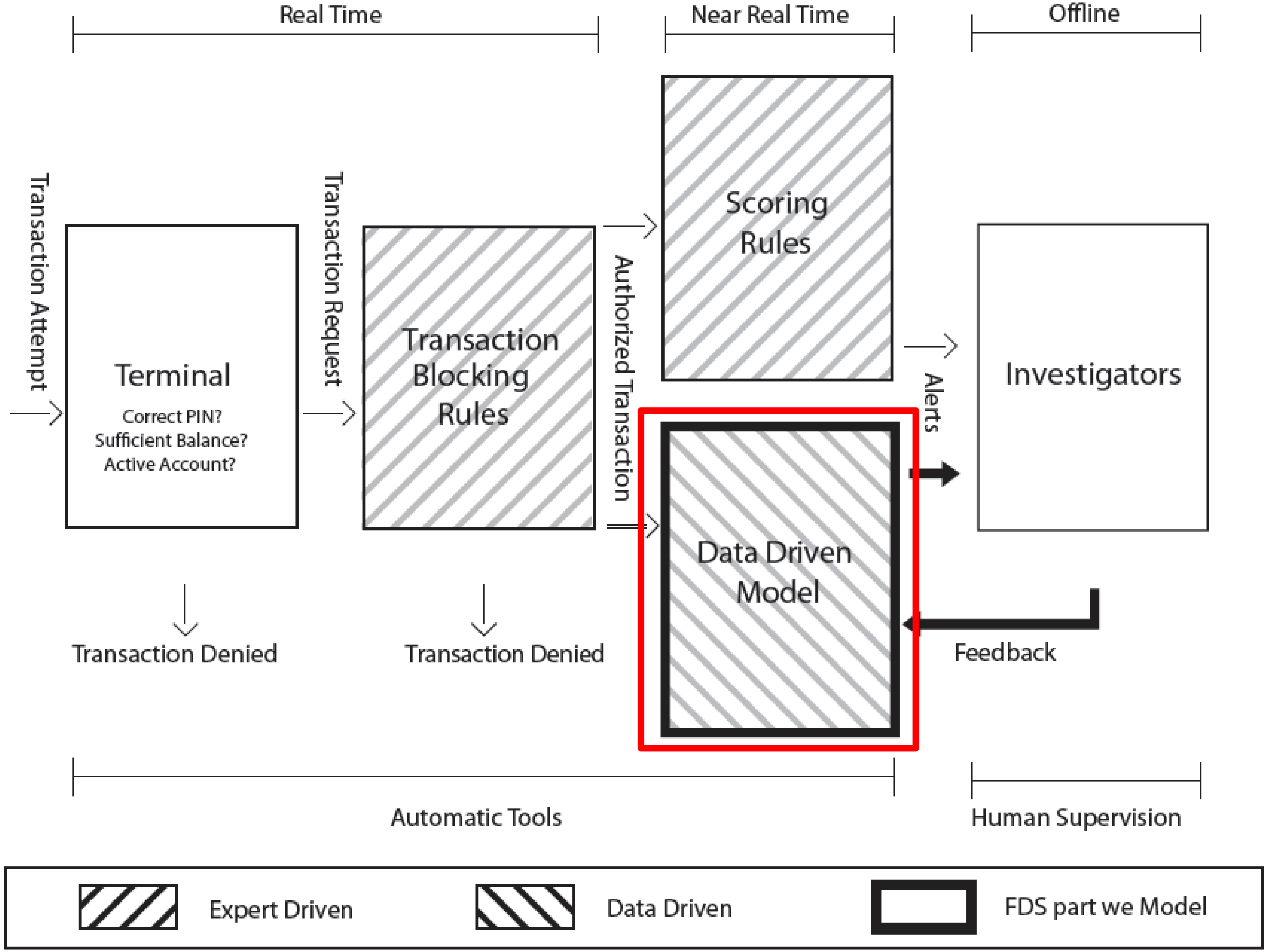
Fraud Detection

A REAL WORLD FRAUD-DETECTION SYSTEM



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

A REAL WORLD FRAUD-DETECTION SYSTEM



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

AN INTRODUCTORY EXAMPLE

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

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

Challenging classification problem because of

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

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

CONCEPT DRIFT IN LEARNING PROBLEMS

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

- Recommendation systems
- Spam / email filtering

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

Spam Classification

CONCEPT DRIFT IN LEARNING PROBLEMS

Concept Drift often occurs in prediction problems like :

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

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

Financial Markets

IN PRACTICE…

In all **application scenarios** where

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

The data-driven model should **adapt** to preserve its performance in case of **Concept Drift (CD)**

THIS TUTORIAL

This tutorial focuses on:

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

DISCLAIMER

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

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

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

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

DISCLAIMER

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

The tutorial will be unbalanced towards active methods but

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

DISCLAIMER

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

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

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

TUTORIAL OUTLINE

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

PROBLEM STATEMENT

Learning in Nonstationary (Streaming) Environments

CLASSIFICATION OVER DATASTREAMS

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

$$X = \{\boldsymbol{x_0}, \boldsymbol{x_1}, \dots, \}$$

Data-generating process $\mathcal{X}$ generates tuples $(\boldsymbol{x}_t, y_t) \sim \mathcal{X}$

- $\boldsymbol{x}_t$ is the observation at time t (e.g., $\boldsymbol{x}_t \in \mathbb{R}^d$)
- y_t is the associated label which is (often) unknown ($y_t \in \Lambda$)

CLASSIFICATION OVER DATASTREAMS

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

$$X = \{\boldsymbol{x_0}, \boldsymbol{x_1}, \dots ,\}$$

Data-generating process $\mathcal{X}$ generates tuples $(\boldsymbol{x}_t, y_t) \sim \mathcal{X}$

- $\boldsymbol{x}_t$ is the observation at time t (e.g., $\boldsymbol{x}_t \in \mathbb{R}^d$)
- y_t is the associated label which is (often) unknown ($y_t \in \Lambda$)

The task: learn an **adaptive classifier** K_t to predict labels

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

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

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

CLASSIFICATION OVER DATASTREAMS

Typically, one **assumes**

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

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

- TR contains data generated in stationary conditions

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

CLASSIFICATION OVER DATASTREAMS

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

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

We say that **concept drift** occurs at time t if

$$\phi_t(\boldsymbol{x}, y) \neq \phi_{t+1}(\boldsymbol{x}, y)$$

(we also say $\mathcal{X}$ becomes **nonstationary**)

ASSUMPTIONS: SUPERVISED SAMPLES

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

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

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

DRIFT TAXONOMY

DRIFT TAXONOMY

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

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

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

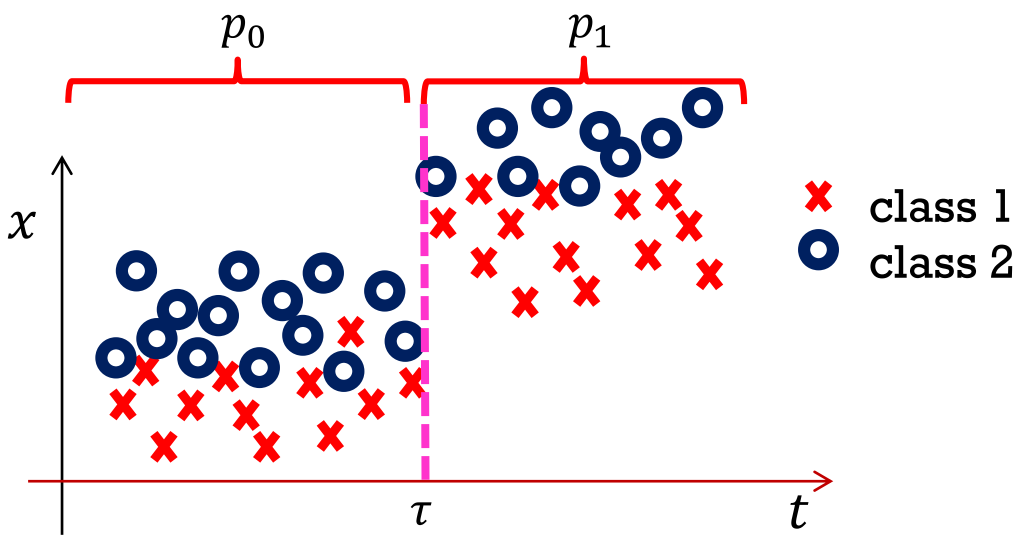
DRIFT TAXONOMY: WHAT IS CHANGING?

Real Drift

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

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

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



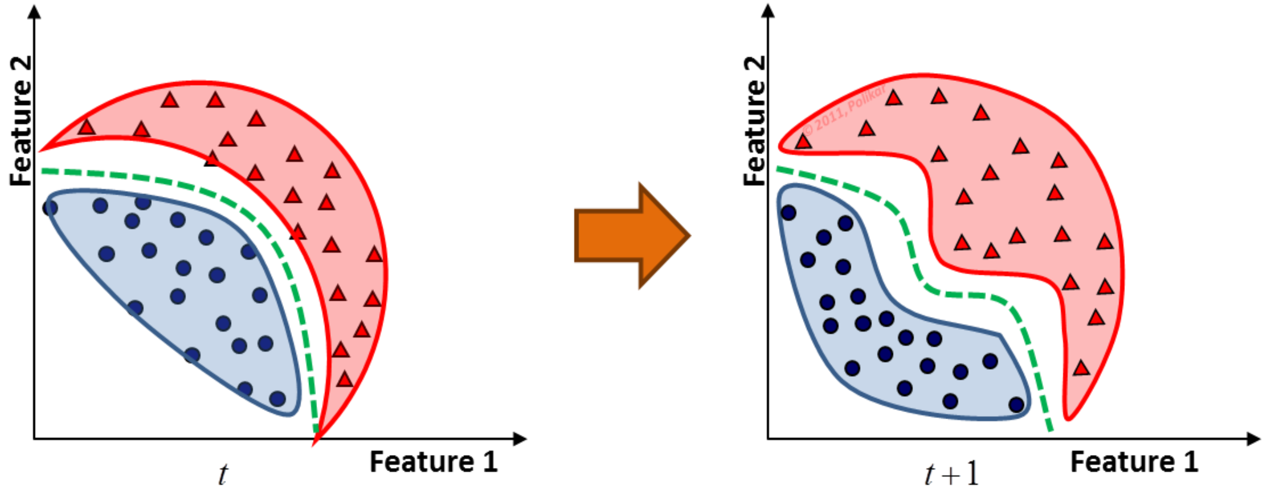
DRIFT TAXONOMY: WHAT IS CHANGING?

Real Drift

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

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

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



DRIFT TAXONOMY: WHAT IS CHANGING?

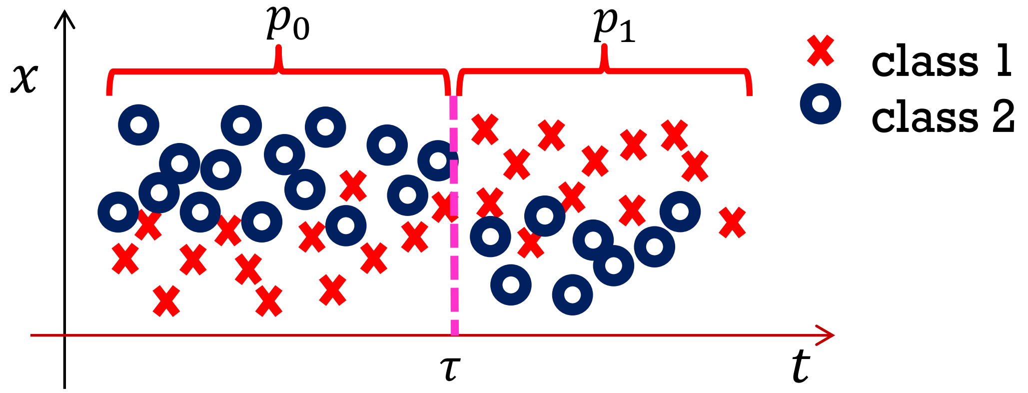
Real Drift

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

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

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

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



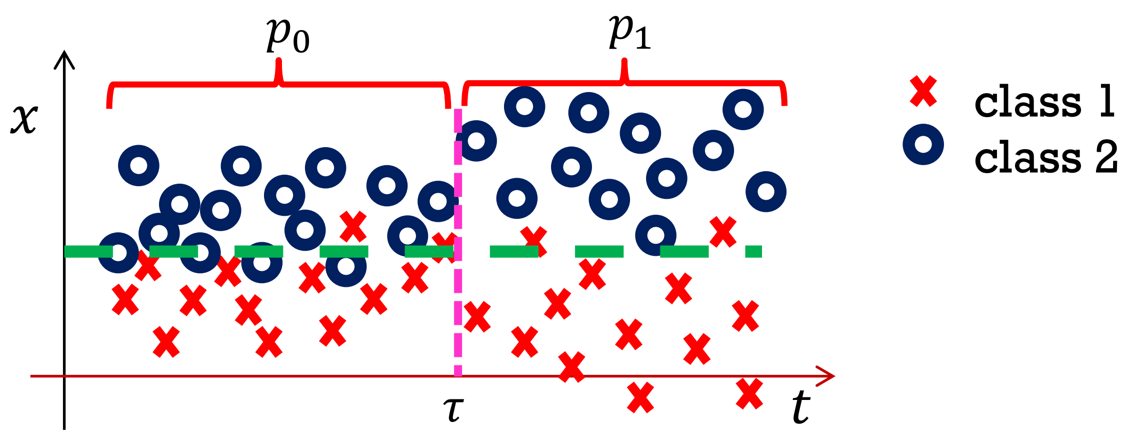
DRIFT TAXONOMY: WHAT IS CHANGING?

Virtual Drift

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

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

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

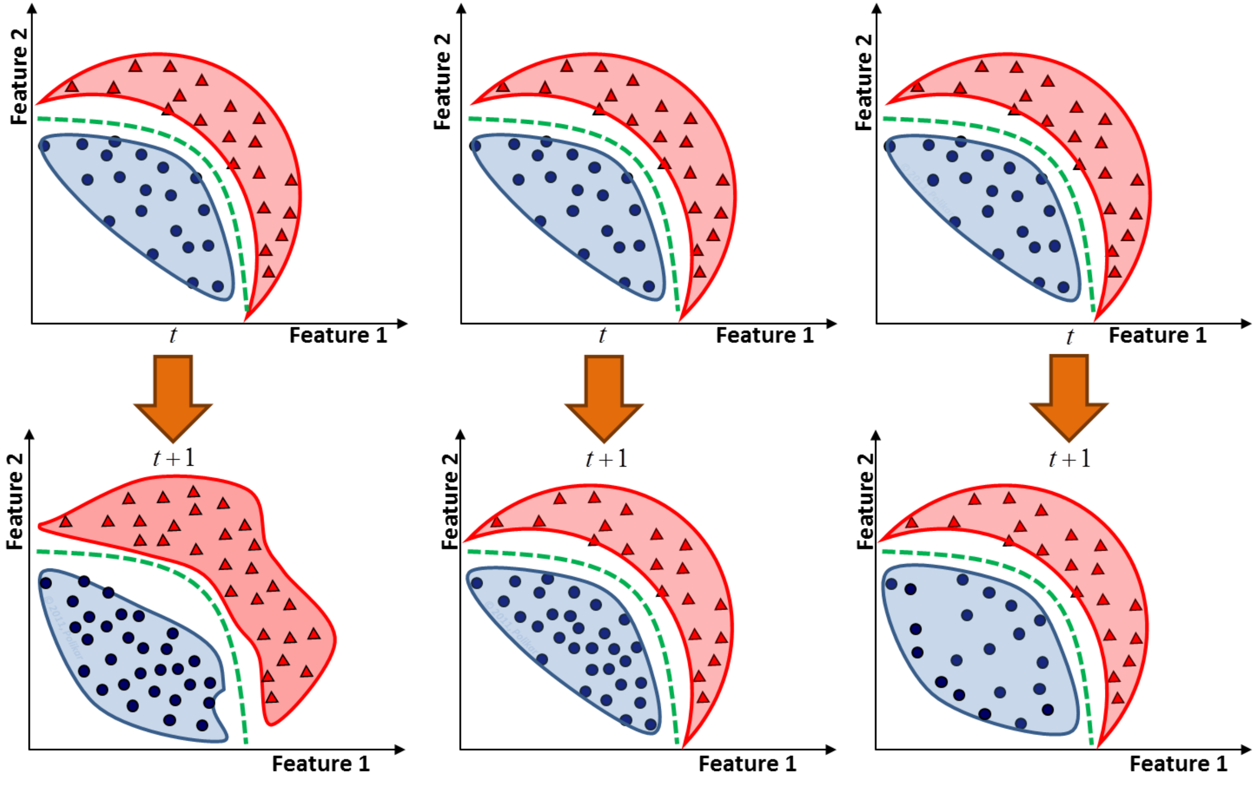


DRIFT TAXONOMY: WHAT IS CHANGING?

Virtual Drift

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

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



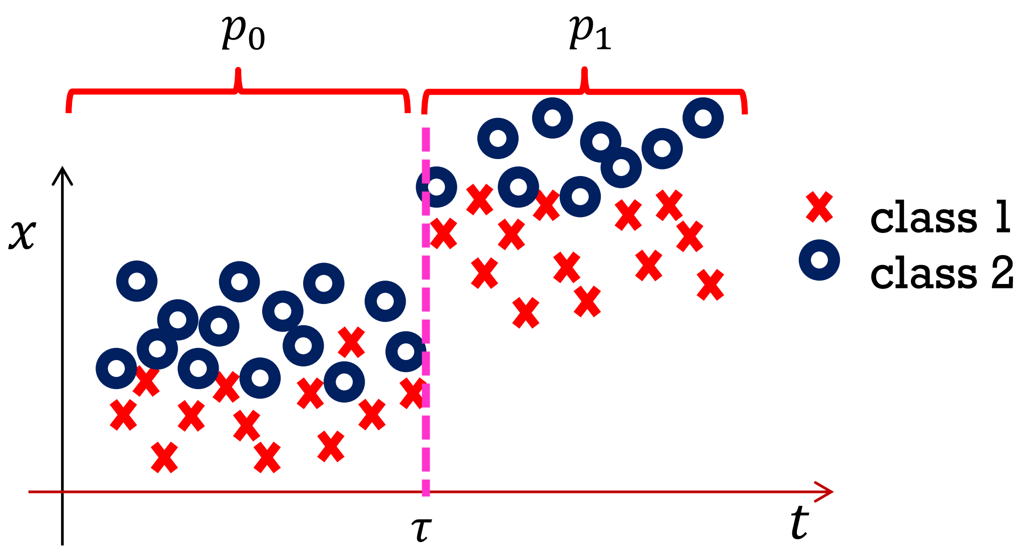
$p_{t+1}(X) \neq p_t(X)$ (a) $p_{t+1}(c_\bullet) \neq p_t(c_\bullet)$ (b) $p_{t+1}(X|c_\bullet) \neq p_t(X|c_\bullet)$ (c)

DRIFT TAXONOMY: TIME EVOLUTION

Abrupt

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

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

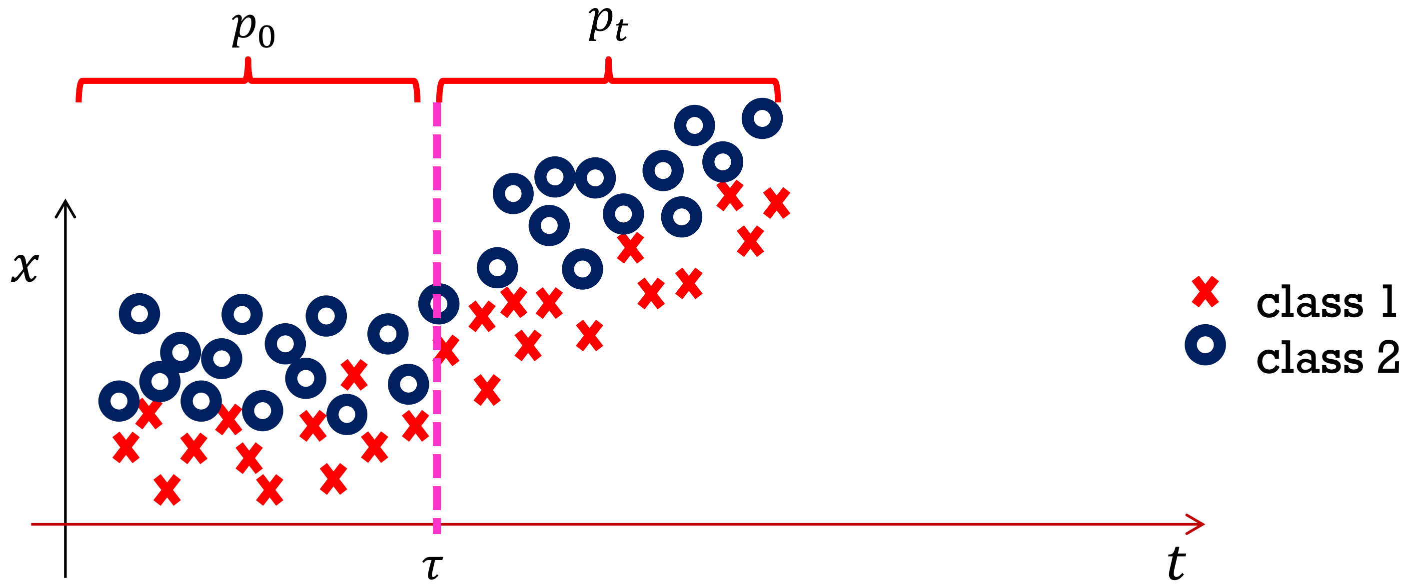


DRIFT TAXONOMY: TIME EVOLUTION

Incremental

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

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

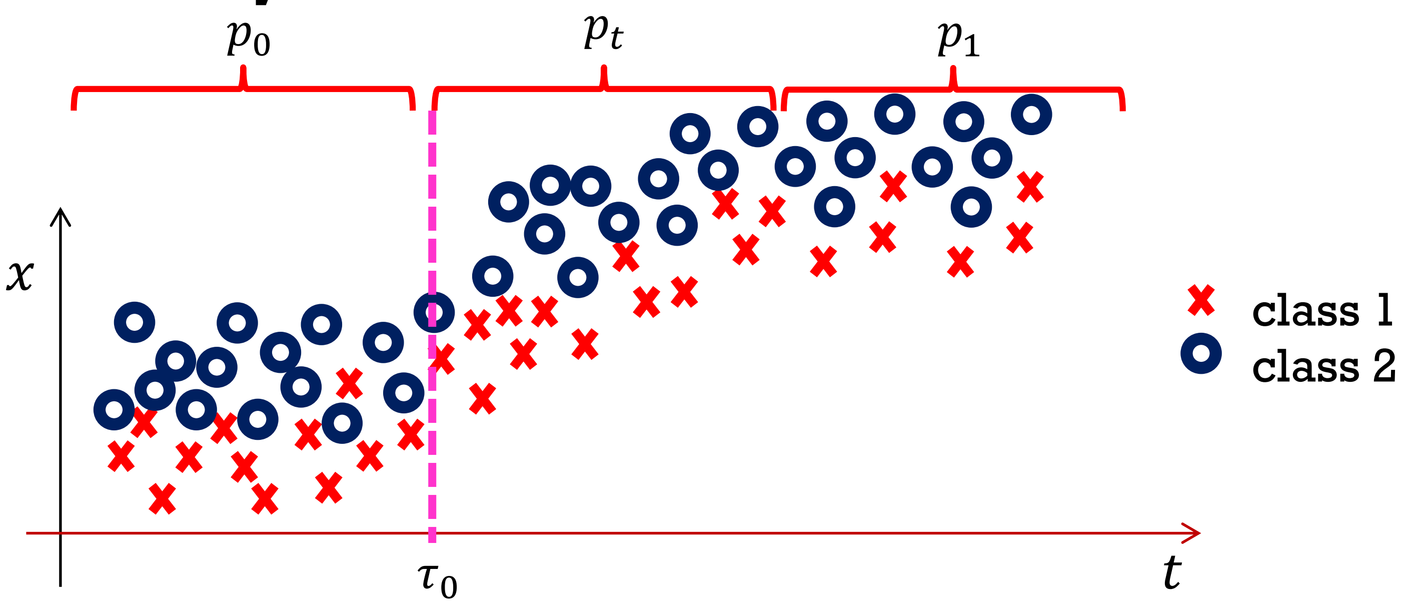


DRIFT TAXONOMY: TIME EVOLUTION

Incremental

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

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

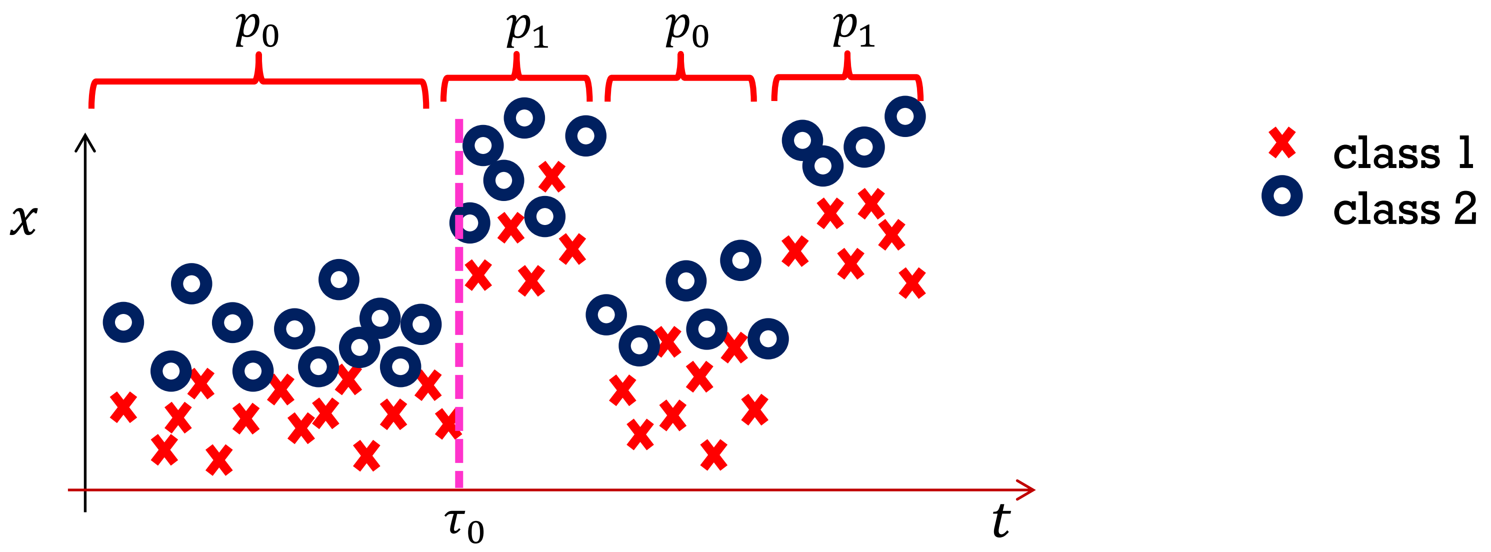


DRIFT TAXONOMY: TIME EVOLUTION

Recurring

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

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

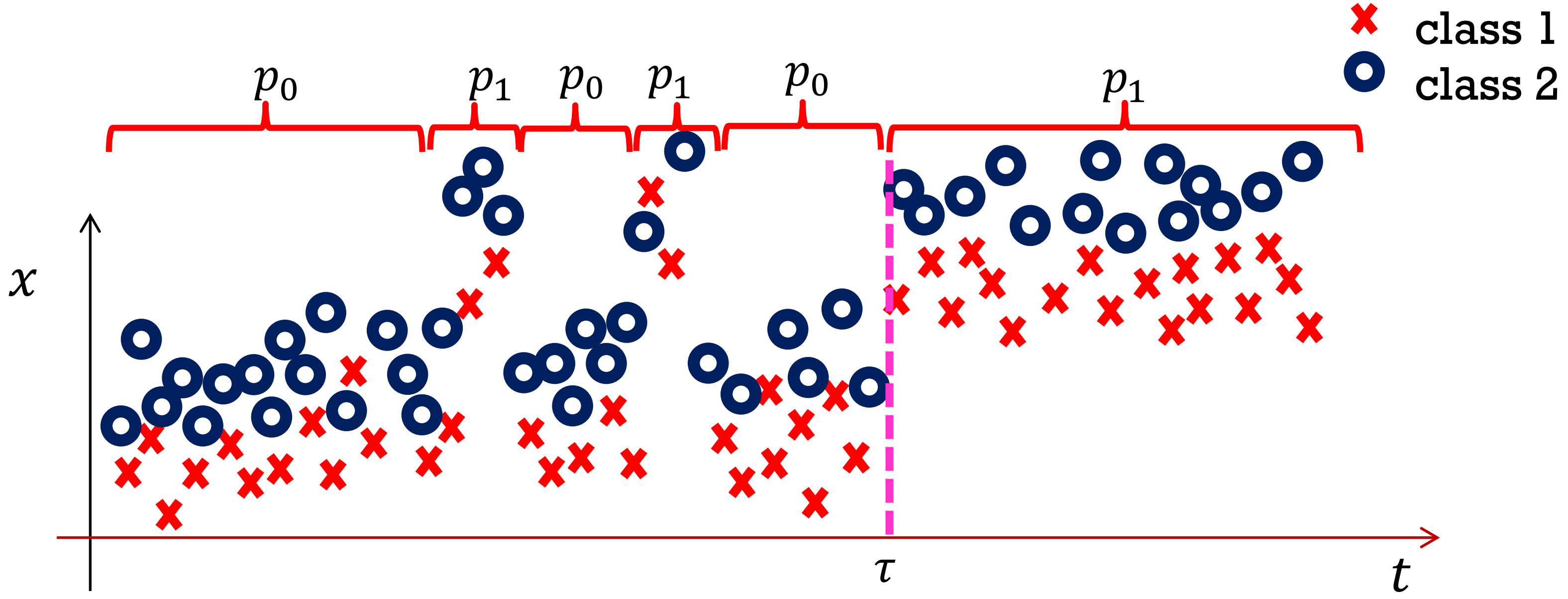


DRIFT TAXONOMY: TIME EVOLUTION

Gradual

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

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

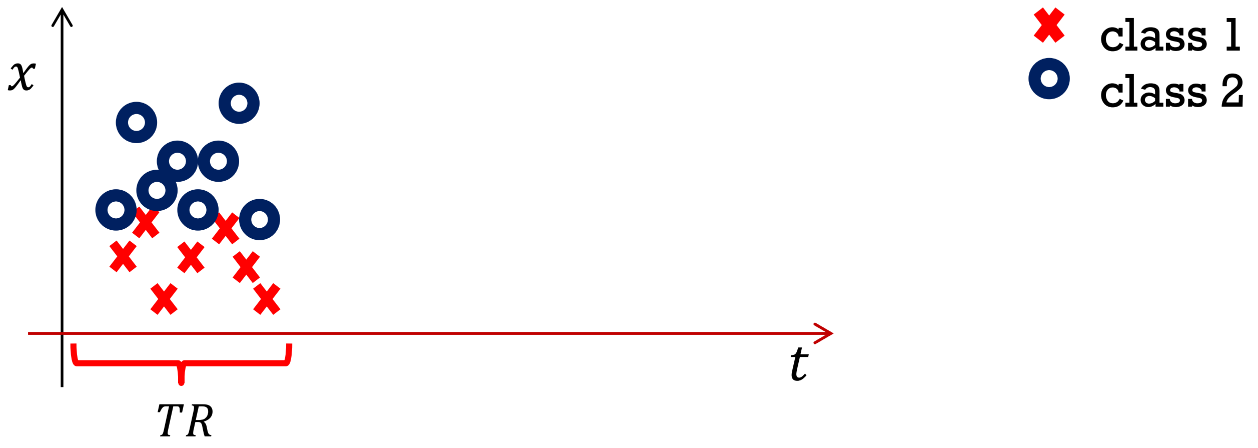


IS CONCEPT DRIFT A PROBLEM?

CLASSIFICATION OVER DATASTREAMS

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

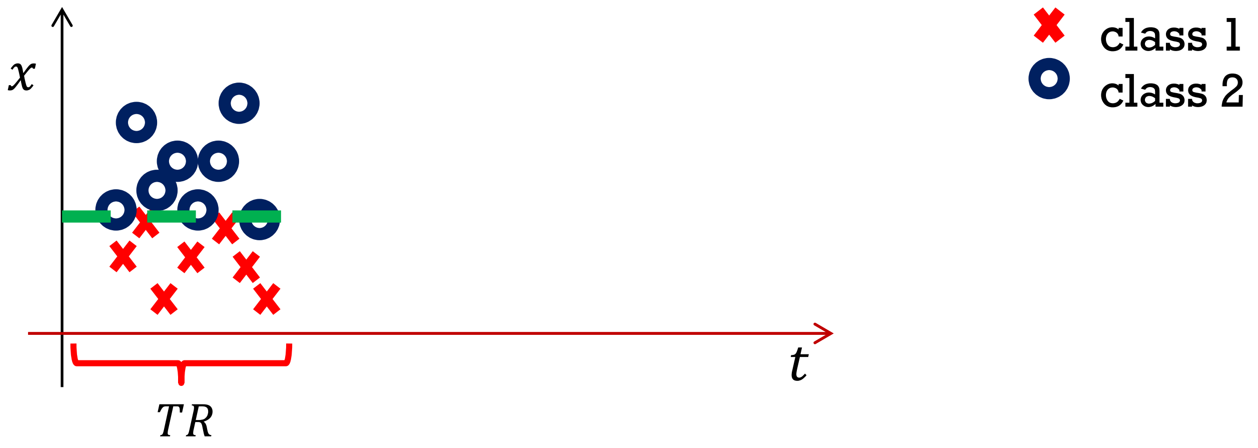
- The initial part of the stream is provided for training
- K is simply a threshold



CLASSIFICATION OVER DATASTREAMS

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

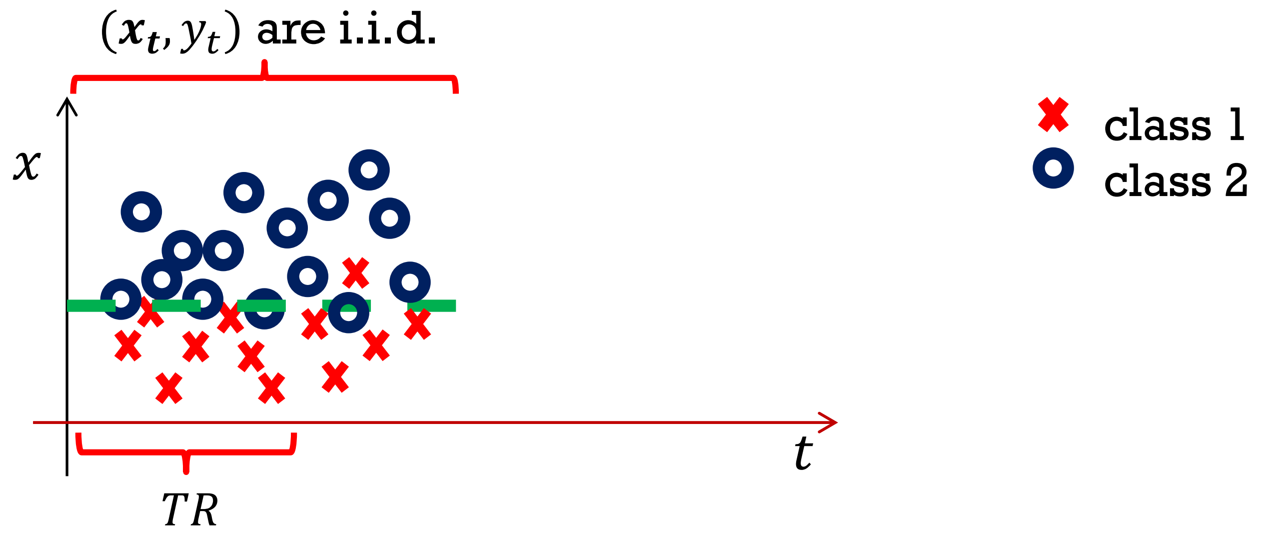
- The initial part of the stream is provided for training
- K is simply a threshold



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

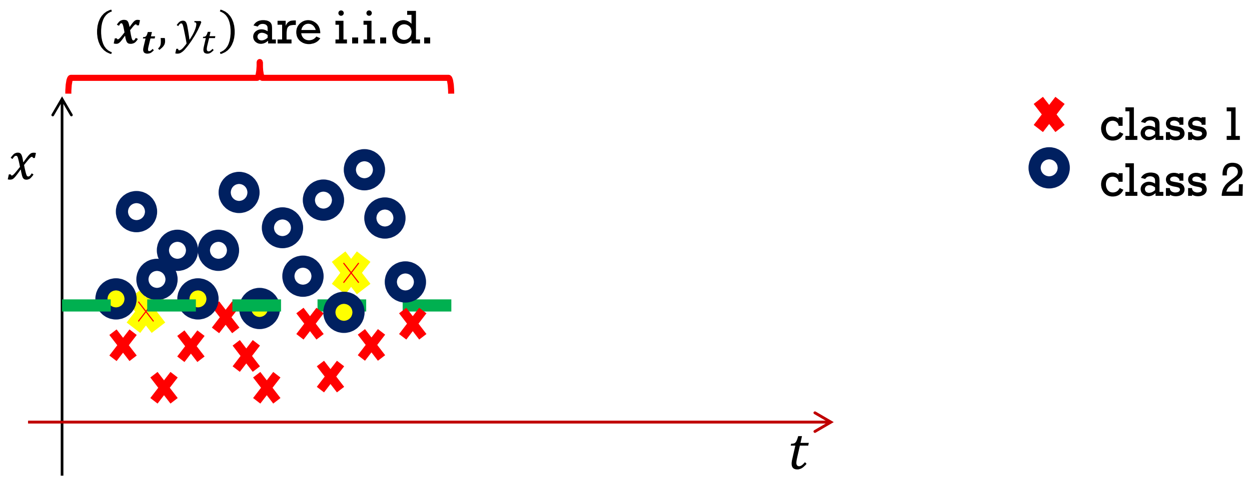


CLASSIFICATION OVER DATASTREAMS

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

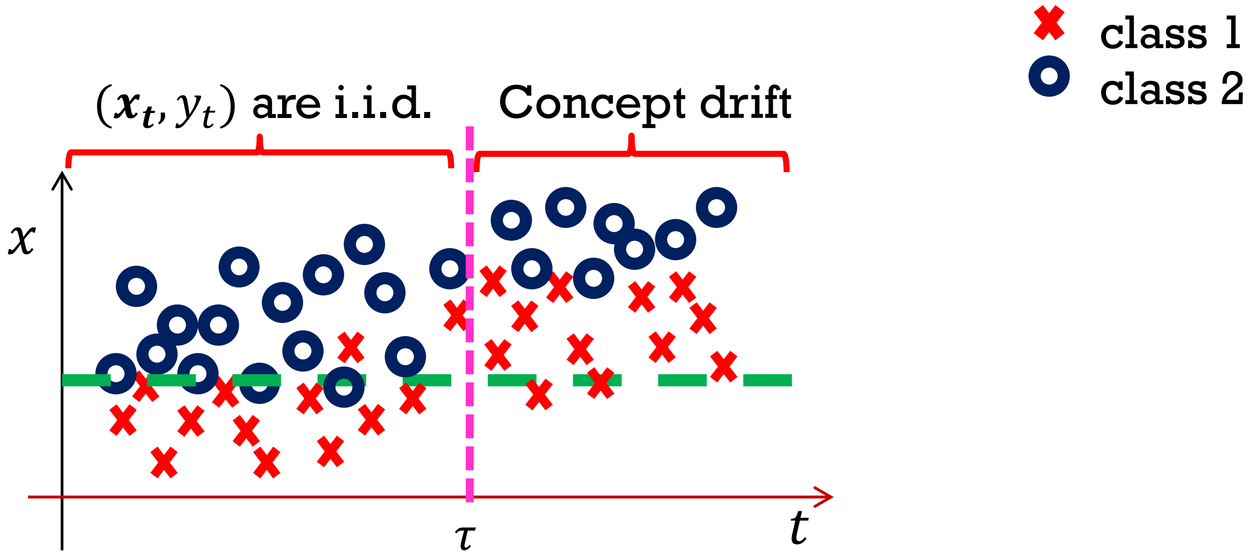
- The initial part of the stream is provided for training
- K is simply a threshold

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



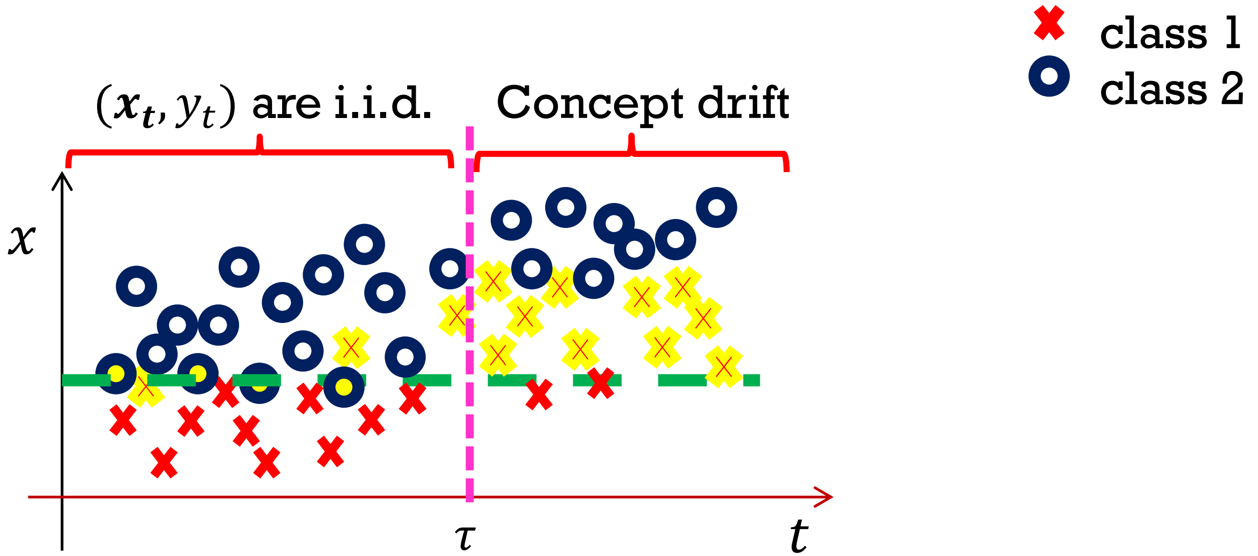
CLASSIFICATION OVER DATASTREAMS

Unfortunately, when concept drift occurs, and ϕ changes,



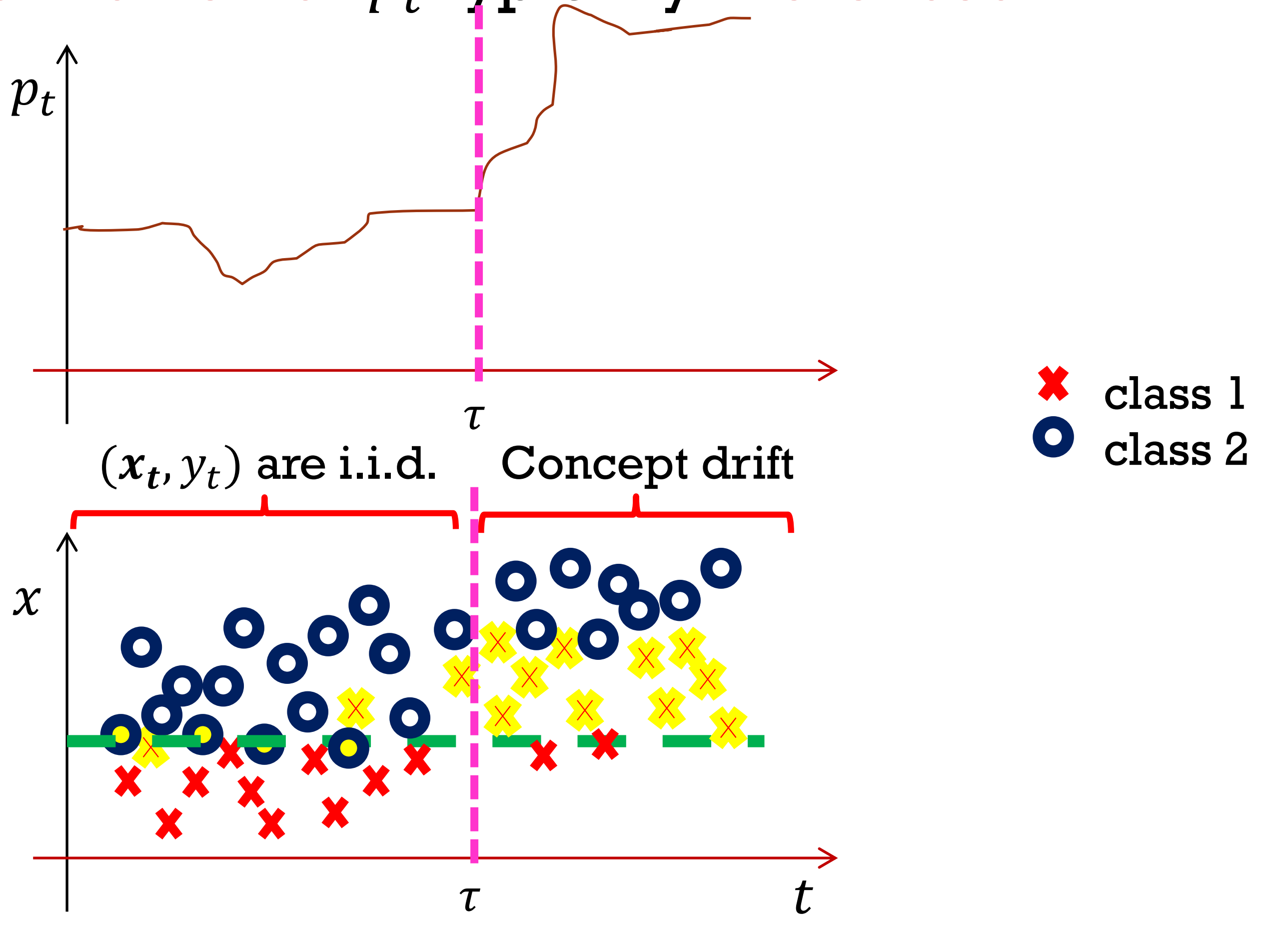
CLASSIFICATION OVER DATASTREAMS

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



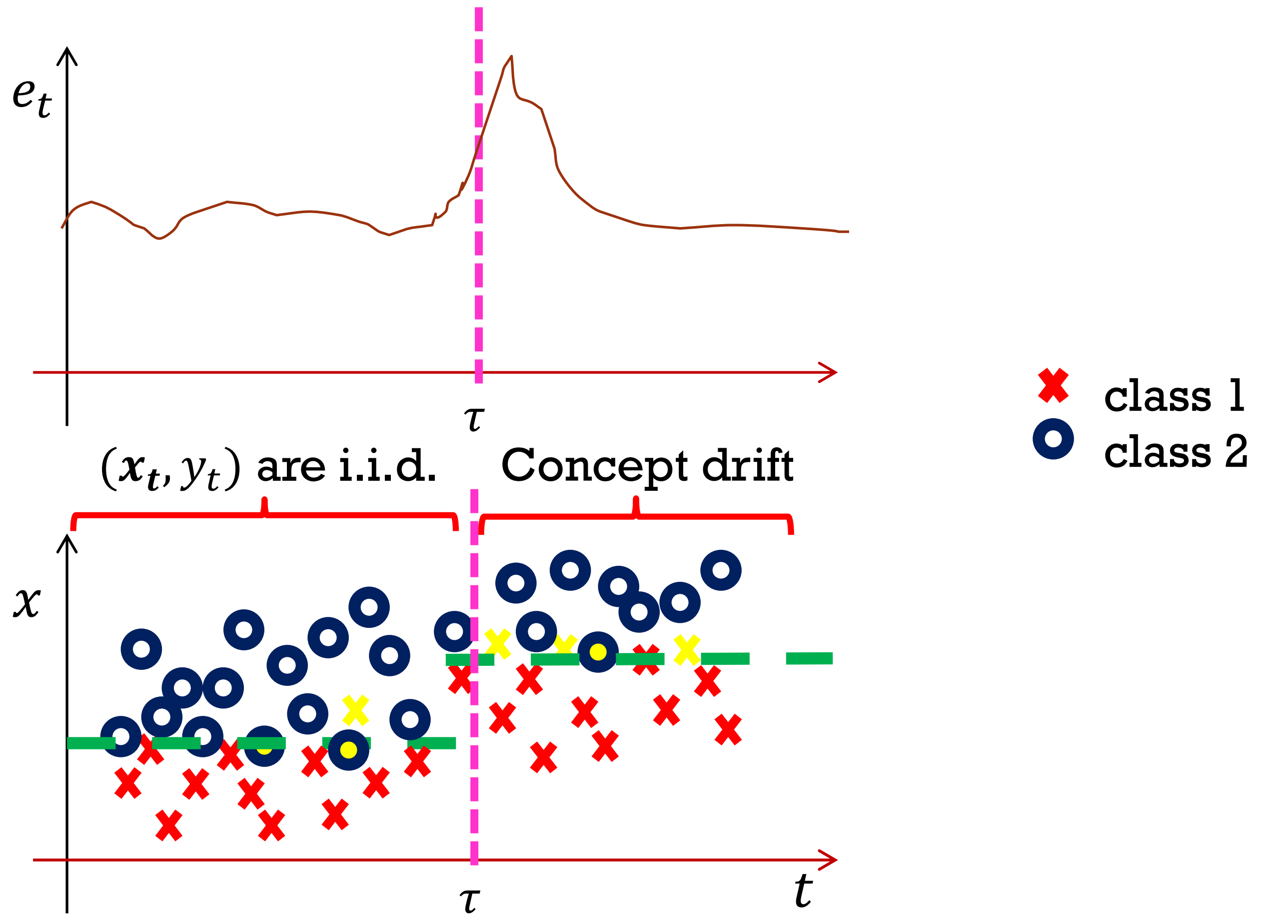
CLASSIFICATION OVER DATASTREAMS

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



NEED FOR ADAPTATION

Adaptation is needed to **preserve** classifier performance



ADAPTATION

SIMPLE ADAPTATION STRATEGIES

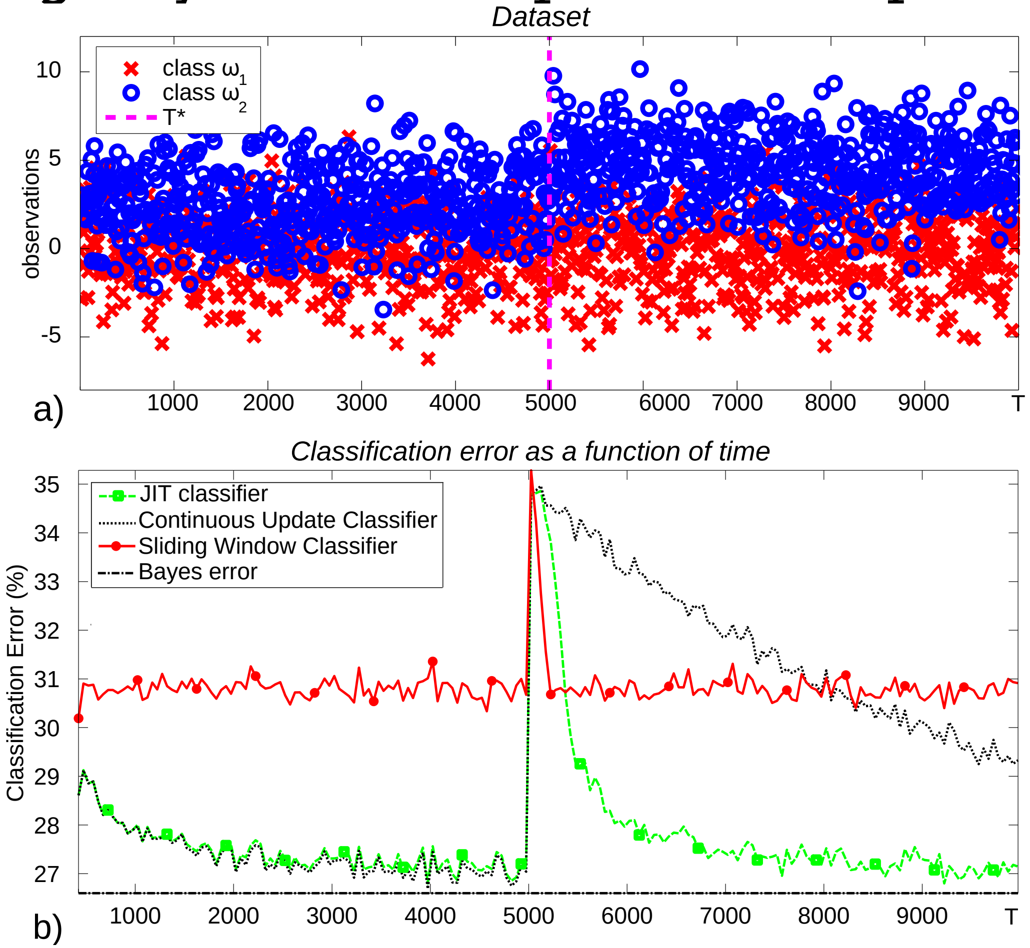
Consider two simple adaptation strategies

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

SIMPLE ADAPTATION STRATEGIES

Consider two simple adaptation strategies

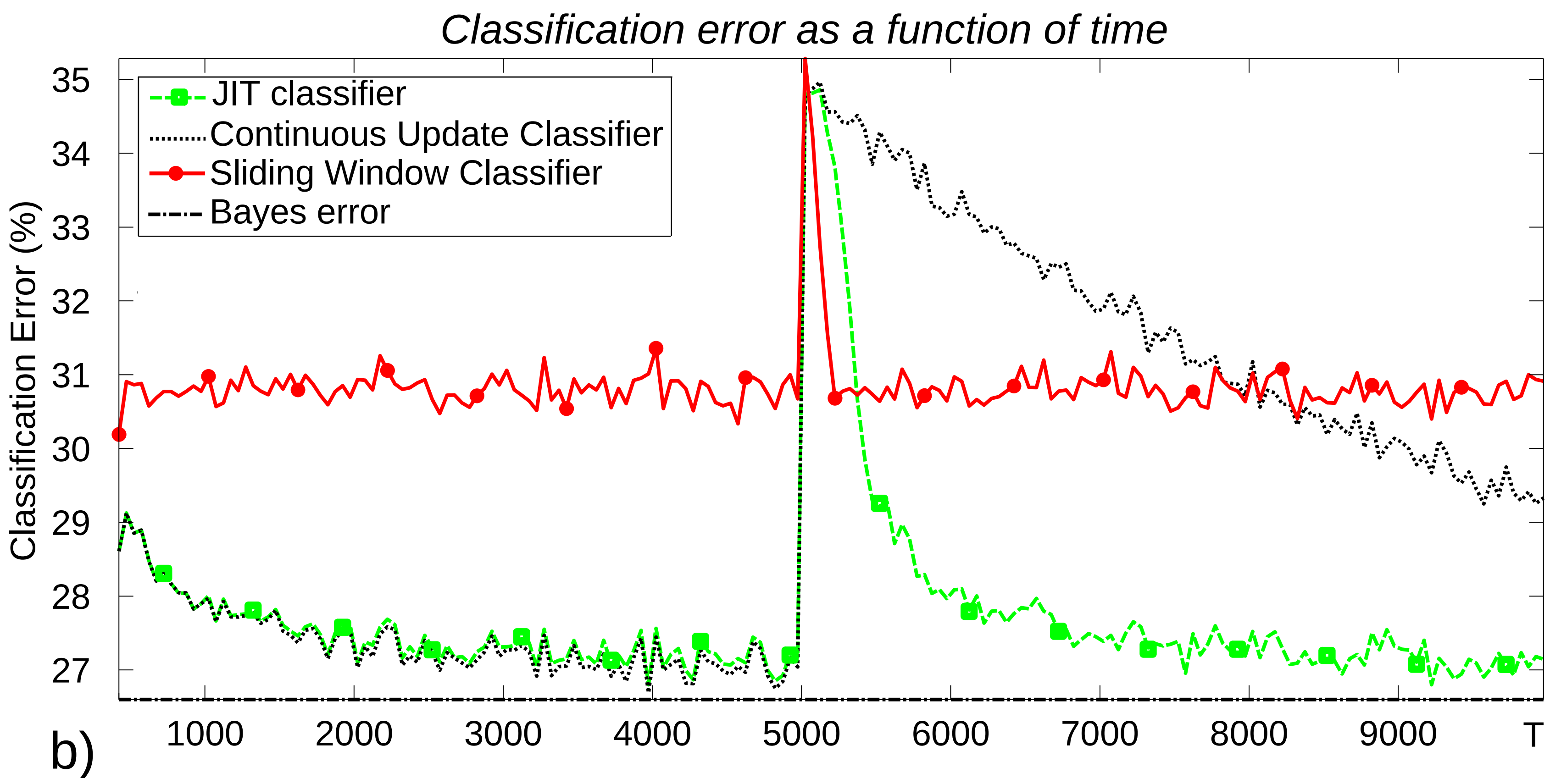
- Continuously update K_t using all supervised couples
- Train K_t using only the last δ supervised couples



SIMPLE ADAPTATION STRATEGIES

Classification error of two simple adaptation strategies

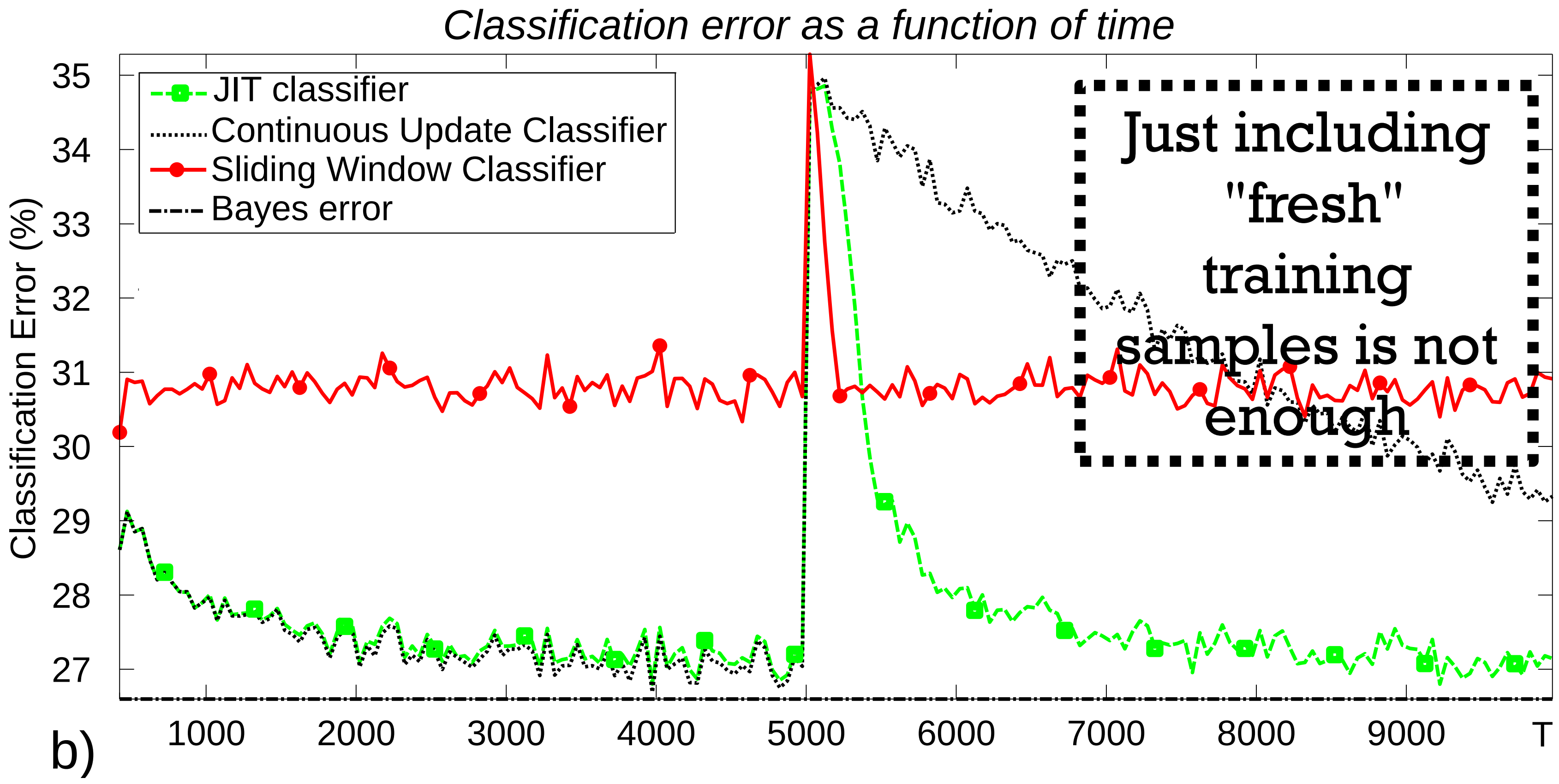
- Black dots: K_t uses all supervised couples at time t
- Red line: K_t uses only the last δ supervised couples



SIMPLE ADAPTATION STRATEGIES

Classification error of two simple adaptation strategies

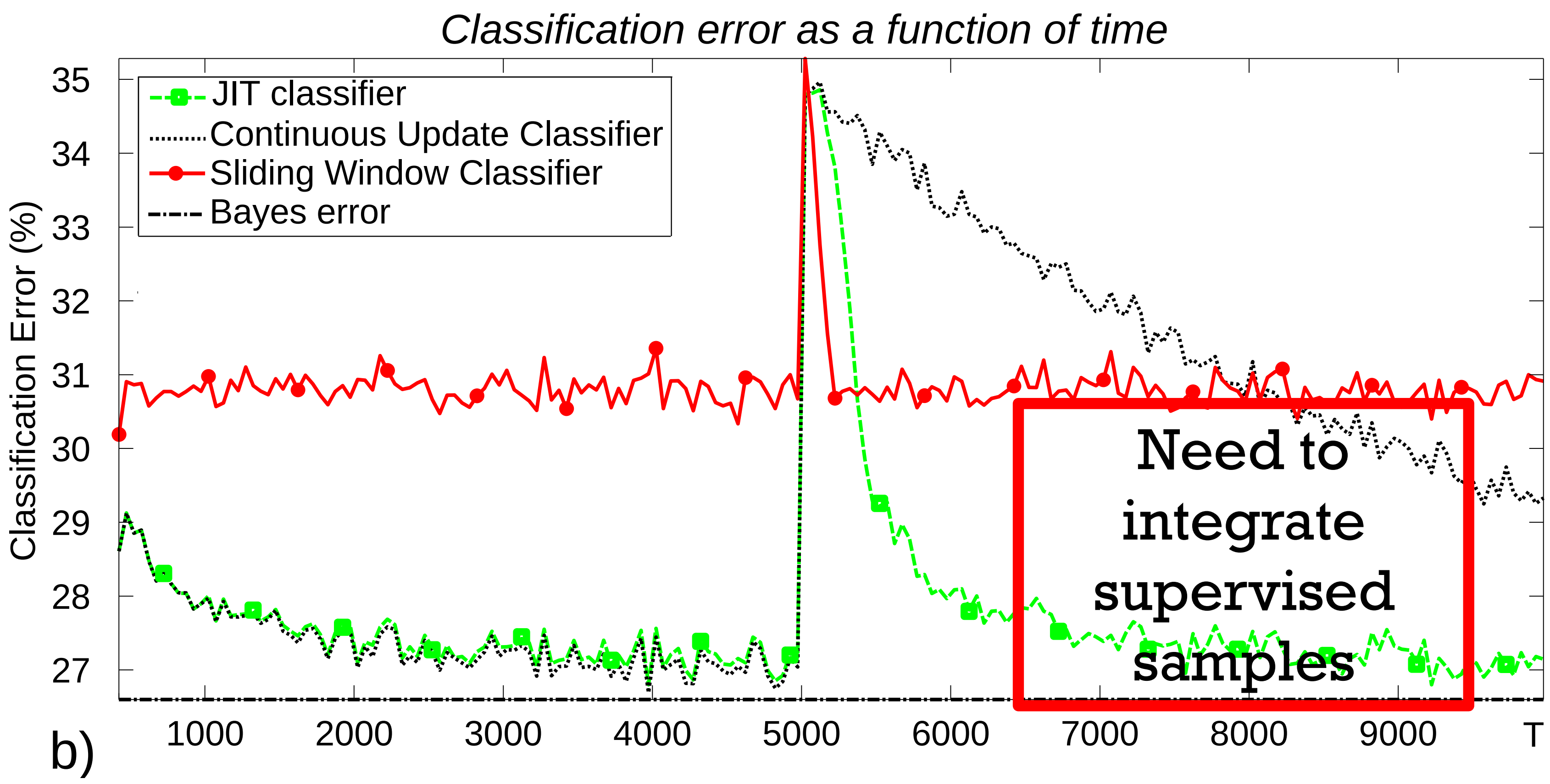
- Black dots: K_t uses all supervised couples at time t
- Red line: K_t uses only the last δ supervised couples



SIMPLE ADAPTATION STRATEGIES

Classification error of two simple adaptation strategies

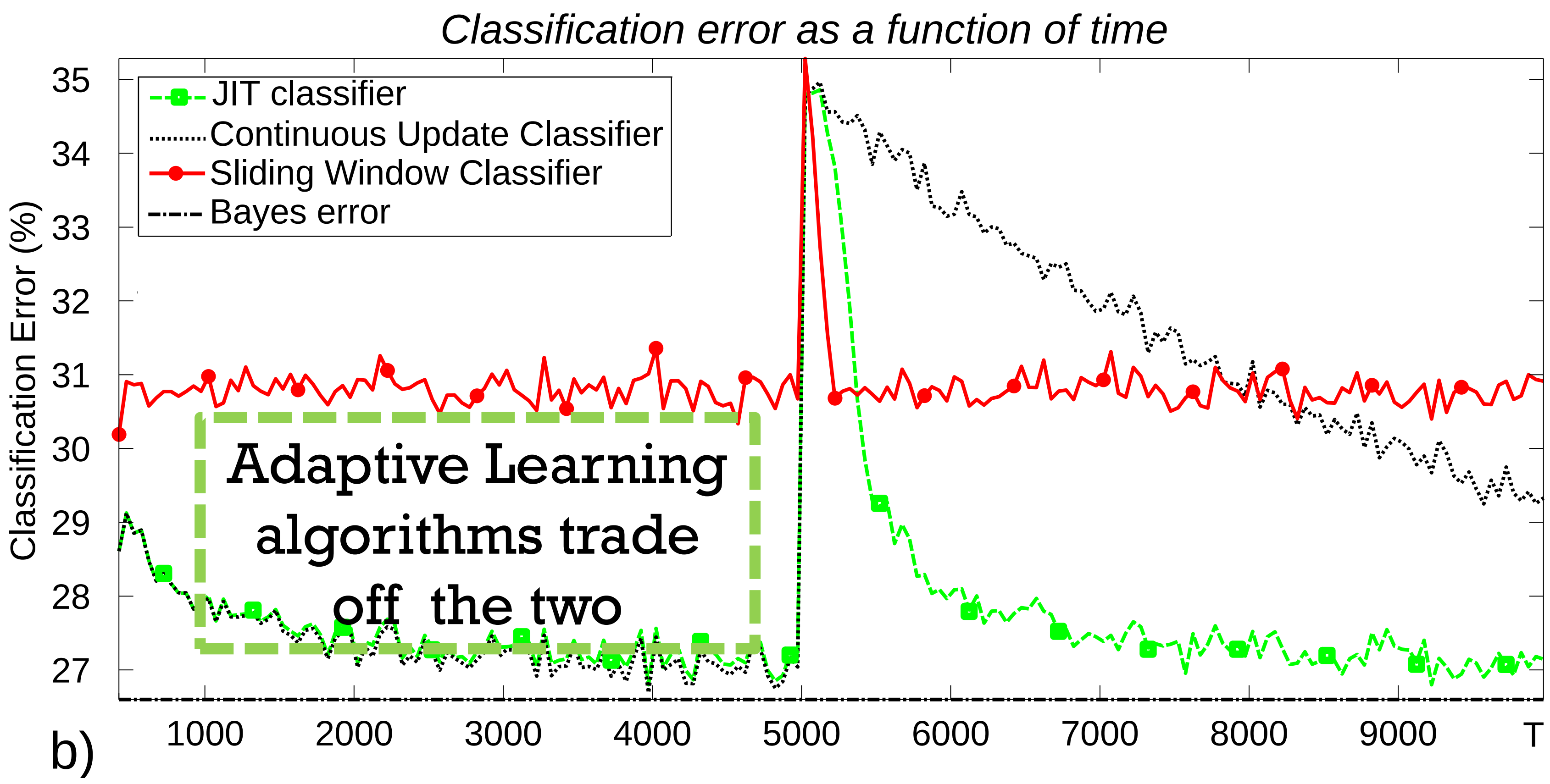
- Black dots: K_t uses all supervised couples at time t
- Red line: K_t uses only the last δ supervised couples



SIMPLE ADAPTATION STRATEGIES

Classification error of two simple adaptation strategies

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



ADAPTATION UNDER CONCEPT DRIFT

Two main solutions in the literature:

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

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

LNSE: ACTIVE APPROACHES

Giacomo Boracchi[1] and Gregory Ditzler[2]

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

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

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

ACTIVE APPROACHES

Peculiarities:

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

Pro:

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

Cons:

- Difficult to handle incremental and gradual drifts

MONITORING THE CLASSIFICATION ERROR

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

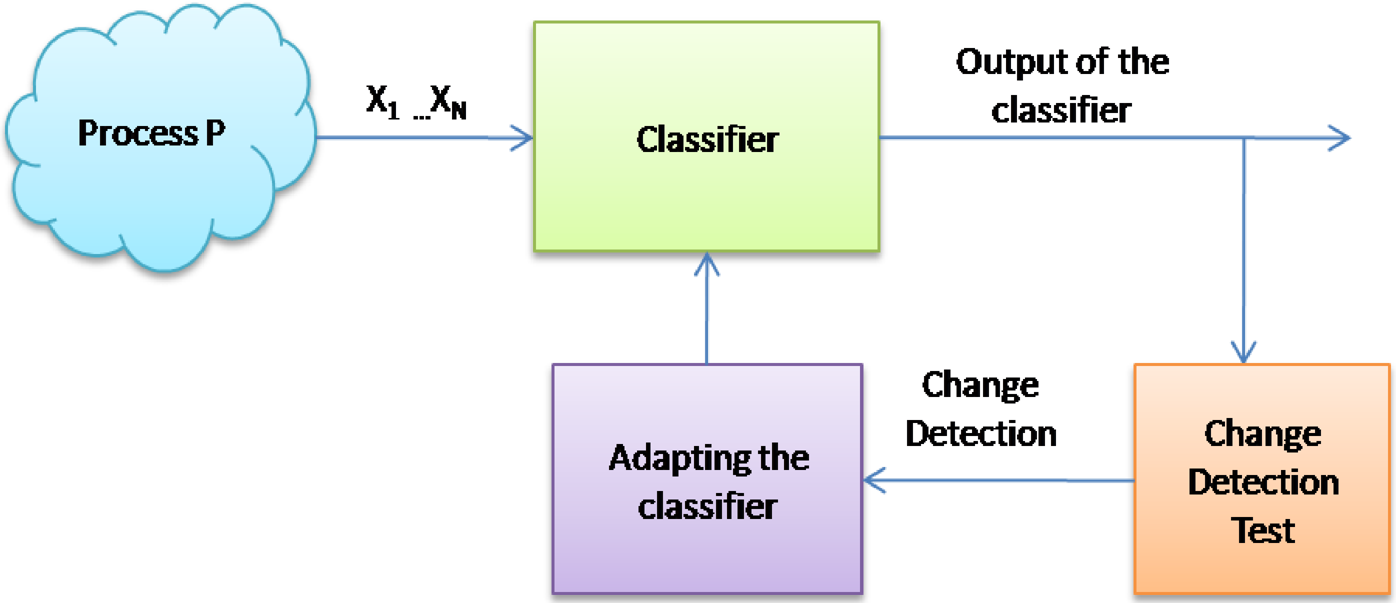
Pro:

- It is the most straightforward figure of merit to monitor
- Changes in p_t prompts **adaptation only when performance are affected**

Cons:

- CD detection from **supervised samples only**

MONITORING THE CLASSIFICATION ERROR



MONITORING THE CLASSIFICATION ERROR

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

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

π_0 is the expected classification error in stationary conditions

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

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

- Gaussian approximation when ν is sufficiently large

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

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

MONITORING THE CLASSIFICATION ERROR: DDM

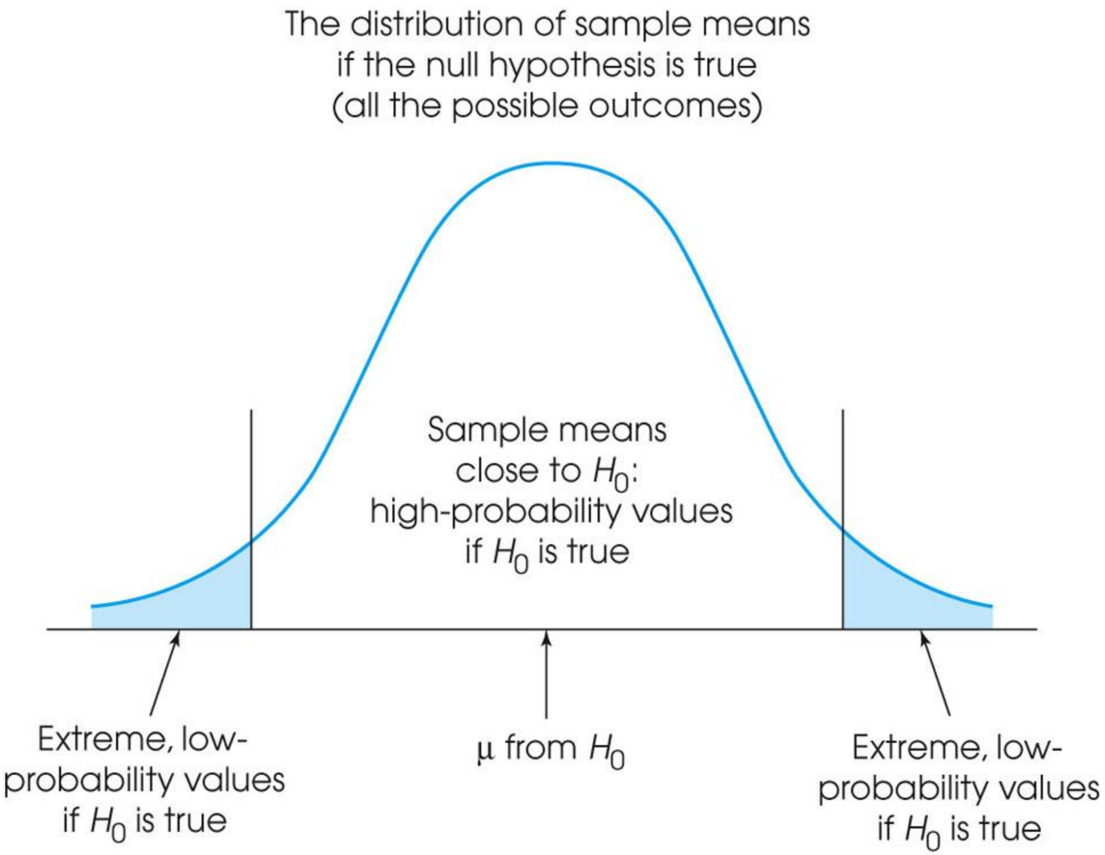
Basic idea behind Drift Detection Method (DDM):

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

MONITORING THE CLASSIFICATION ERROR: DDM

Basic idea behind Drift Detection Method (DDM):

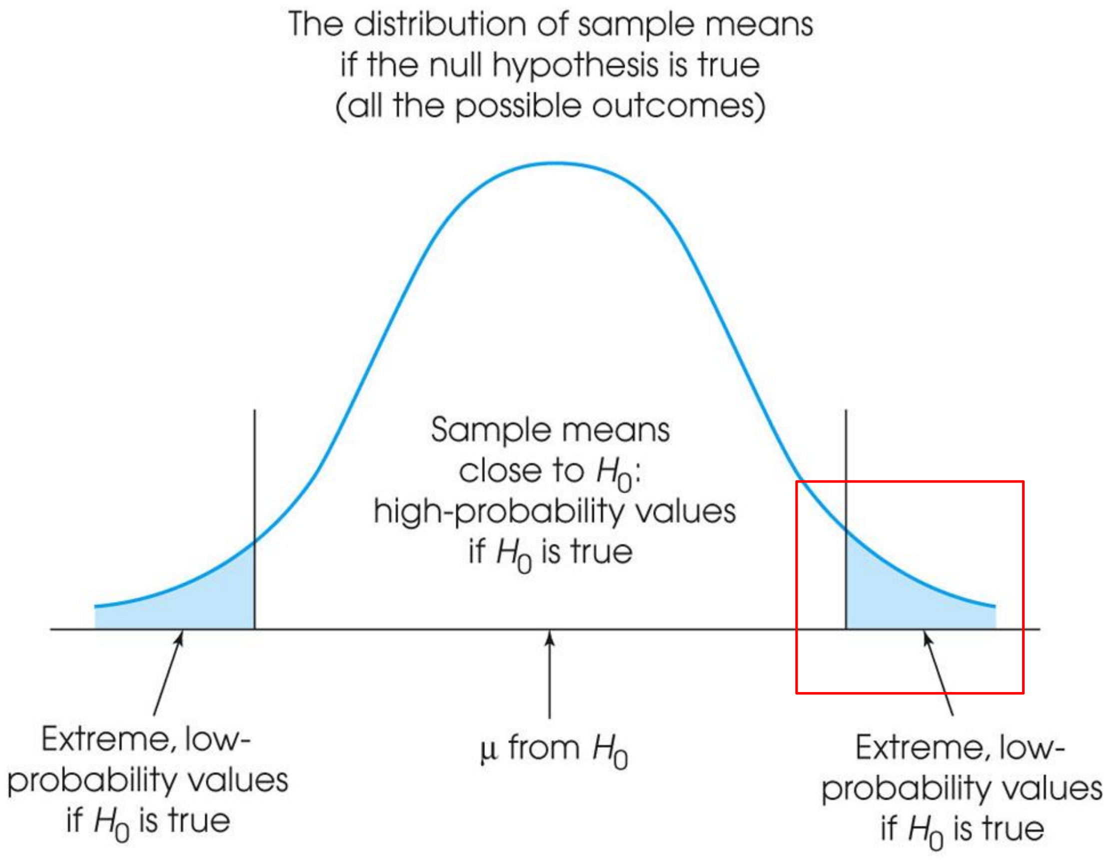
- Detect CD as **outliers** in the classification error



MONITORING THE CLASSIFICATION ERROR: DDM

Basic idea behind Drift Detection Method (DDM):

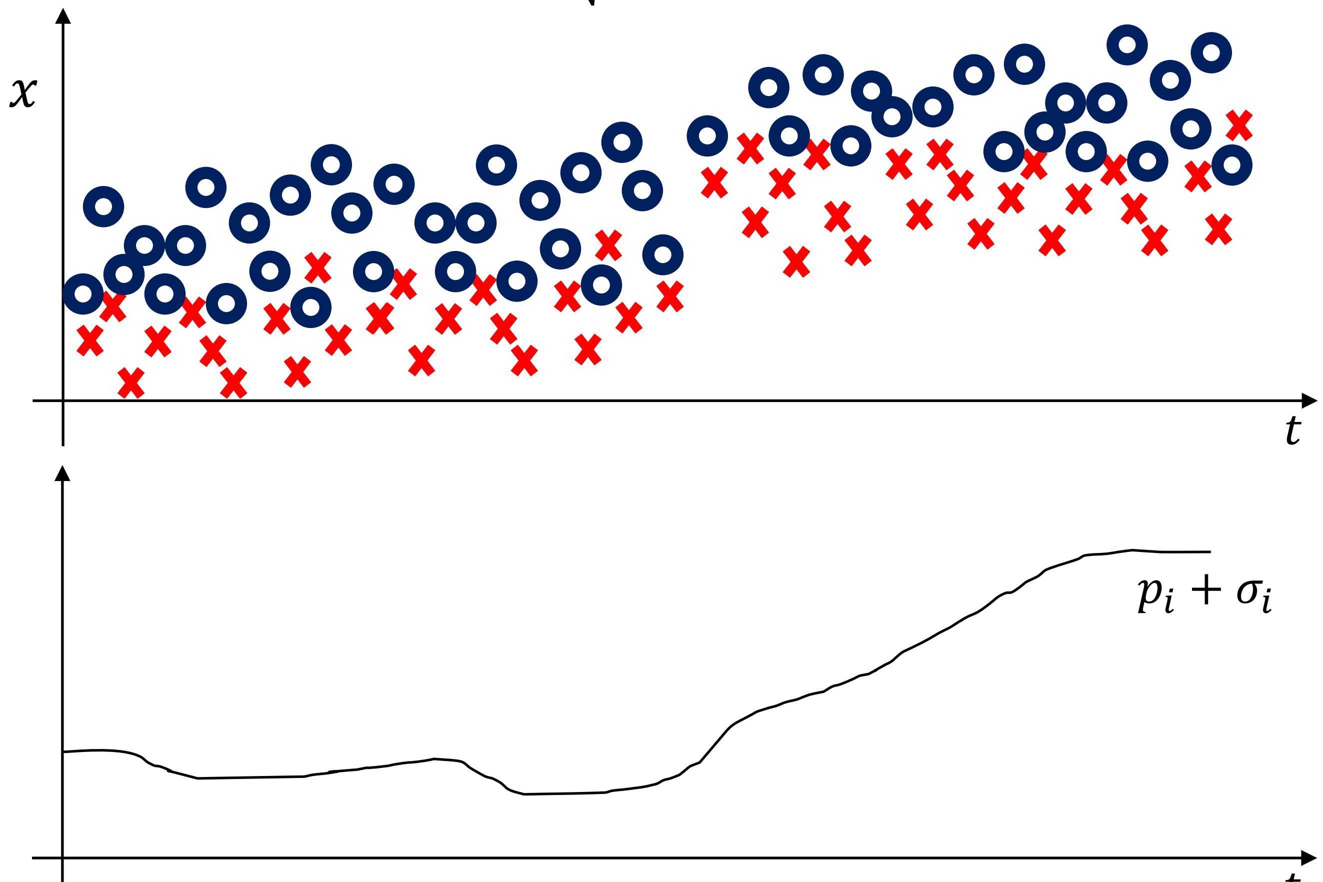
- Detect CD as **outliers** in the classification error
- Since in stationary conditions error will decrease, look for outliers in the right tail only



MONITORING THE CLASSIFICATION ERROR: DDM

Basic idea behind Drift Detection Method (DDM):

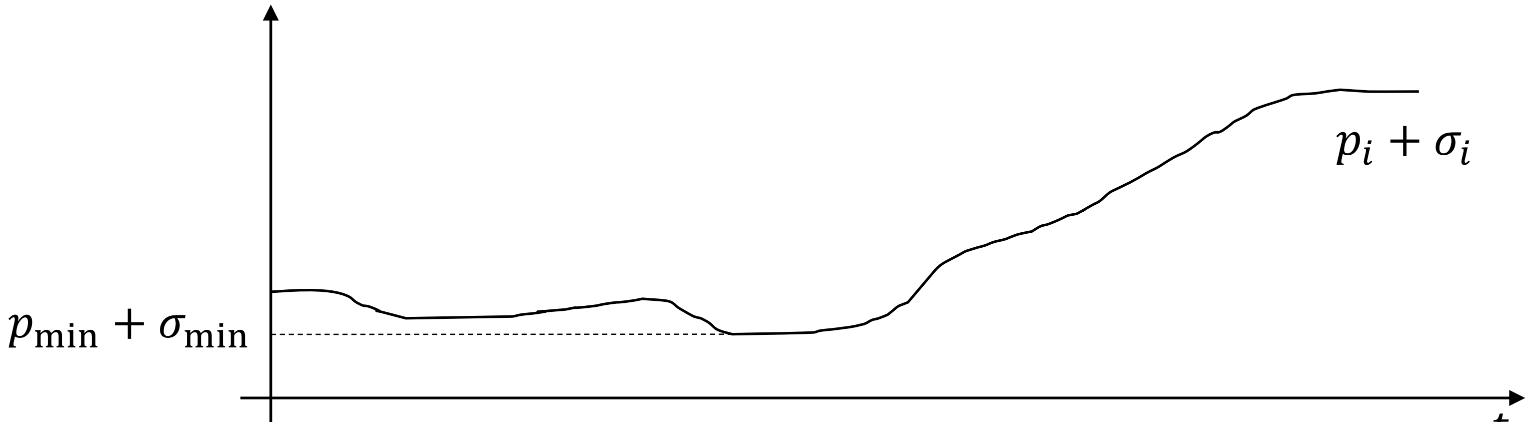
- Detect CD as **outliers** in the classification error
- Compute, over time p_i, and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$



MONITORING THE CLASSIFICATION ERROR: DDM

Basic idea behind Drift Detection Method (DDM):

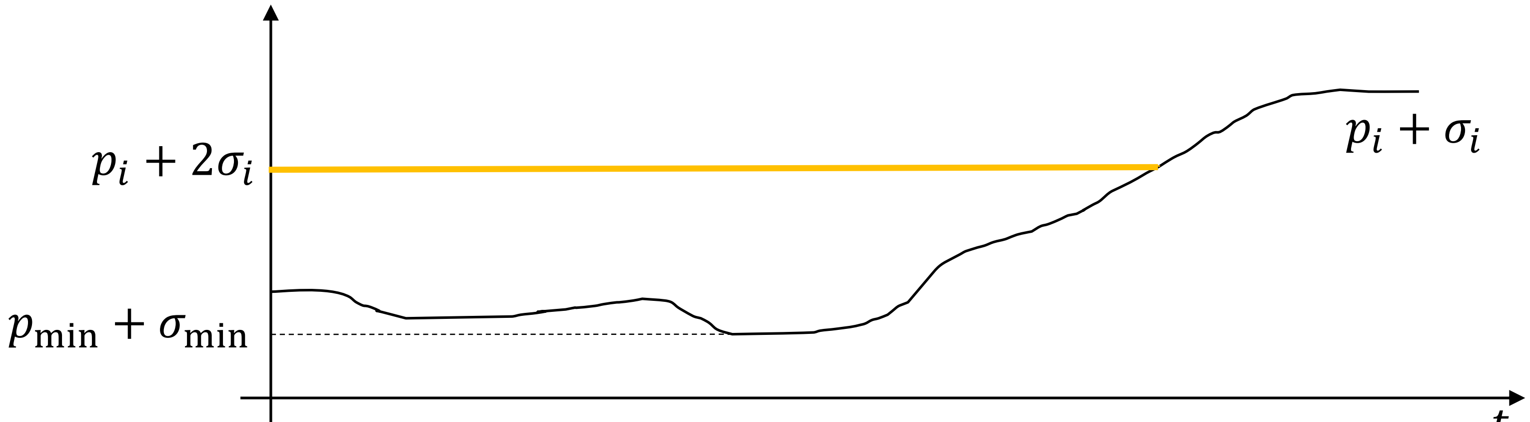
- Detect CD as **outliers** in the classification error
- Compute, over time p_i, and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$
- Let $p_{\min}$ be the minimum error, $\sigma_{\min} = \sqrt{\frac{p_{\min}(1-p_{\min})}{i}}$



MONITORING THE CLASSIFICATION ERROR: DDM

Basic idea behind Drift Detection Method (DDM):

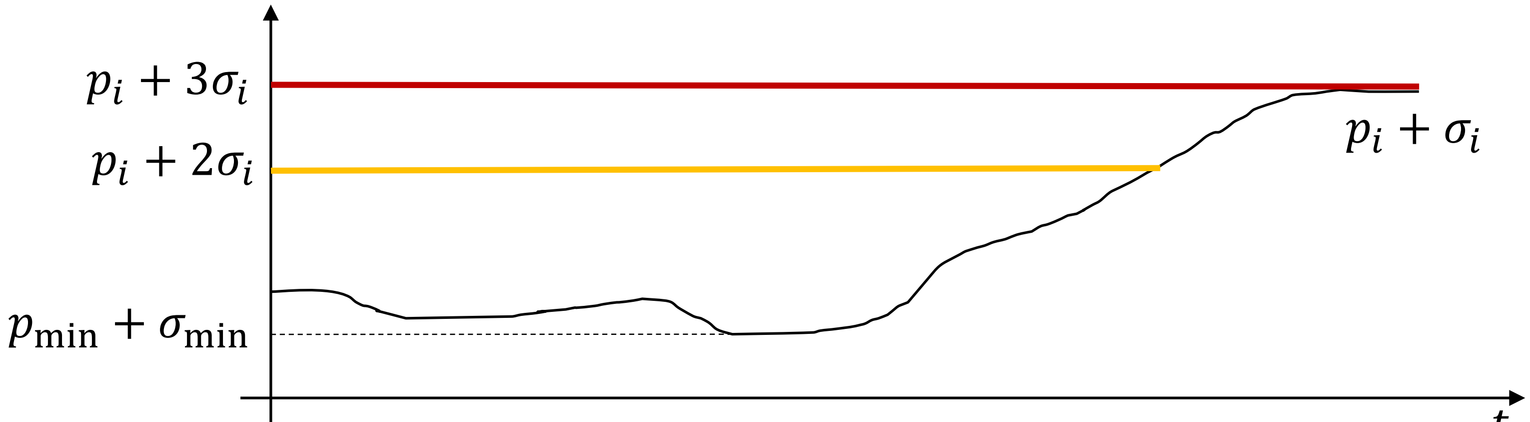
- Detect CD as **outliers** in the classification error
- Compute, over time p_i, and $\sigma_i = \sqrt{\frac{p_i(1-p_i)}{i}}$
- Let $p_{\min}$ be the minimum error, $\sigma_{\min} = \sqrt{\frac{p_{\min}(1-p_{\min})}{i}}$
- When $p_i + \sigma_i > p_{\min} + 2 * \sigma_{\min}$ raise a warning alert



MONITORING THE CLASSIFICATION ERROR: DDM

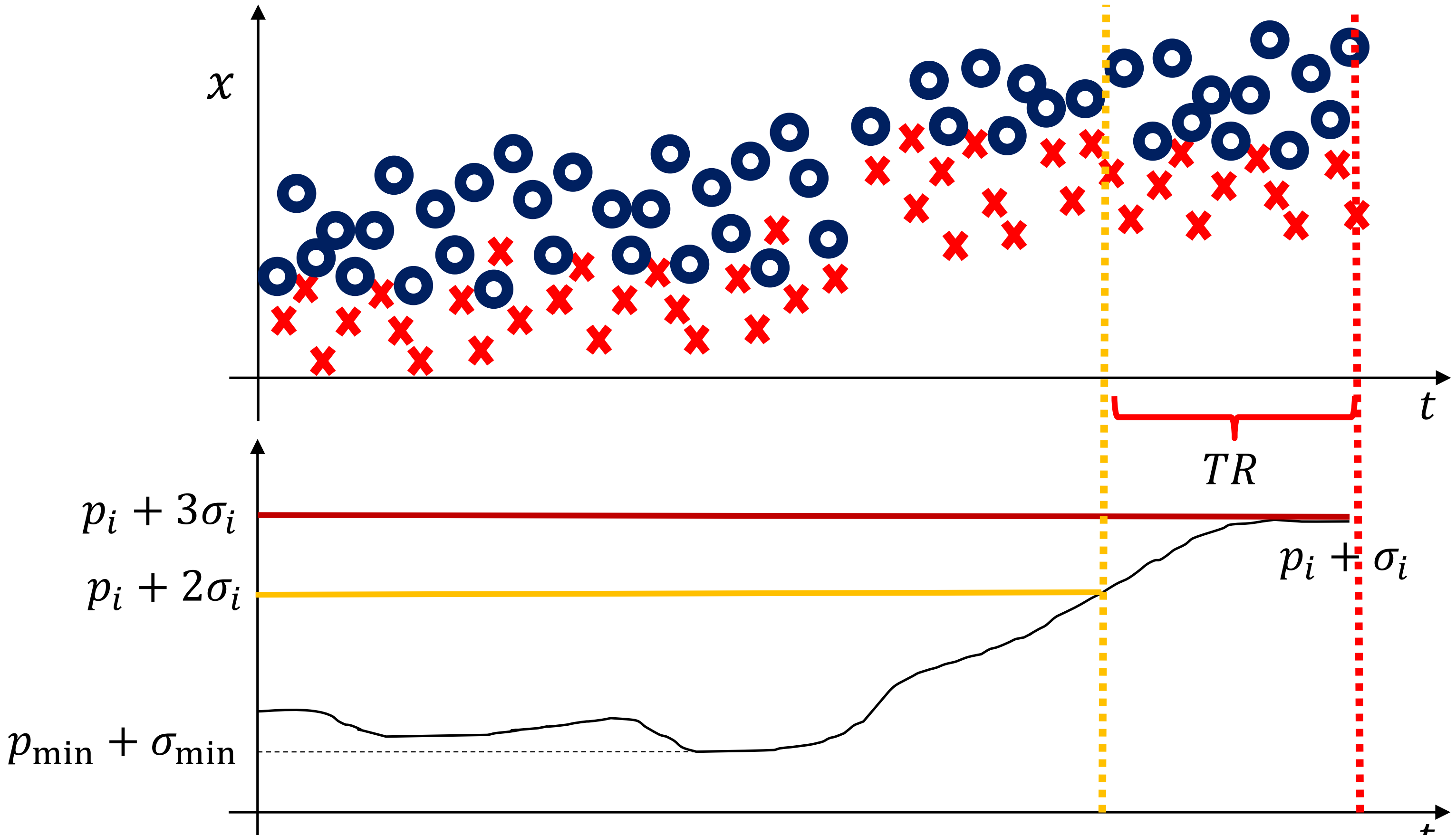
Basic idea behind Drift Detection Method (DDM):

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



POST-DETECTION RECONFIGURATION: DDM

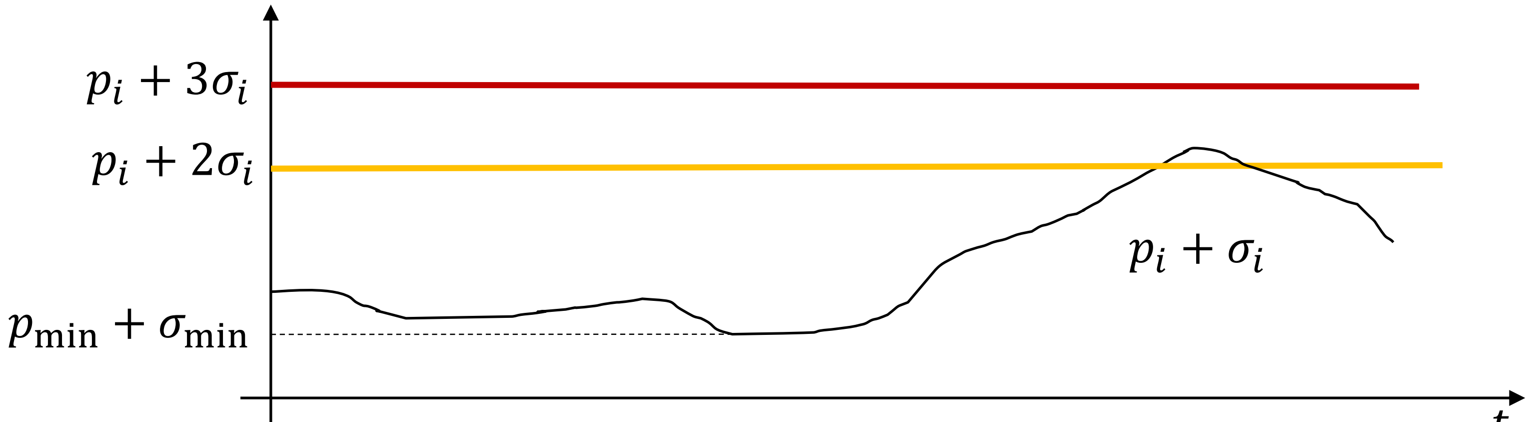
Use supervised samples in between warning and drift alert to reconfigure the classifier



POST-DETECTION RECONFIGURATION: DDM

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

Warning alerts non that are not followed by a drift alert are discarded and considered false-positive detections



MONITORING THE CLASSIFICATION ERROR: EDDM

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

- Average distance is expected to decrease under CD
- They aim at detecting gradual drifts

M. Baena-García, J. Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldá, R. Morales-Bueno. *"Early drift detection method"* In Fourth International Workshop on Knowledge Discovery from Data Streams (2006)

MONITORING THE CLASSIFICATION ERROR: EWMA

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

Compute EWMA statistic

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

Detect concept drift when

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

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

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

G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand "*Exponentially Weighted Moving Average Charts for Detecting Concept Drift*" Pattern Recogn. Lett. 33, 2 (Jan. 2012), 191–198 2012

MONITORING THE CLASSIFICATION ERROR: EWMA

Most importantly:

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

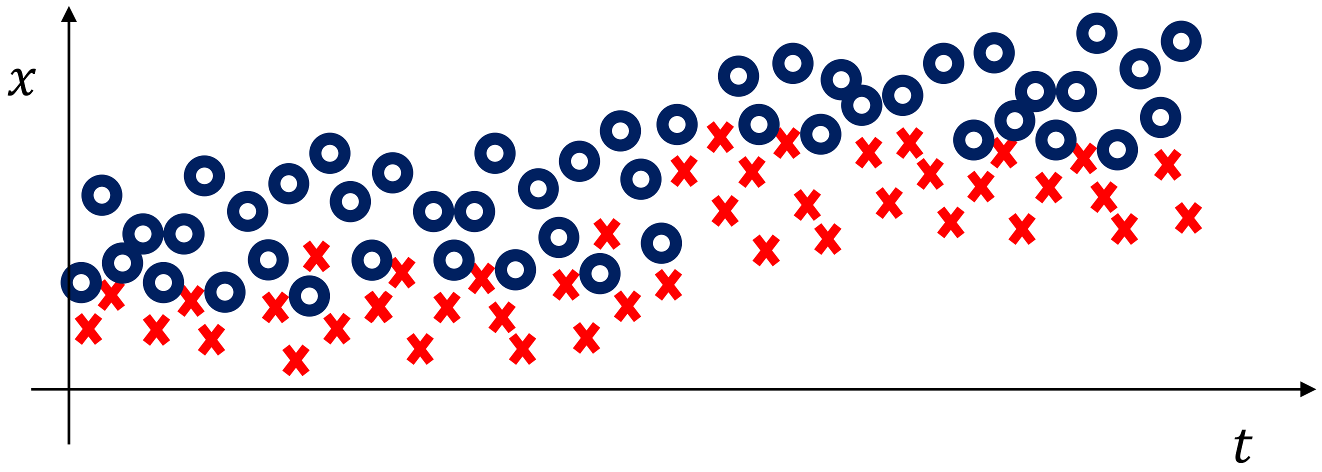
$$Z_t > p_{0,t} + 0.5\ L_t \sigma_t$$

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

G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand "*Exponentially Weighted Moving Average Charts for Detecting Concept Drift*" Pattern Recogn. Lett. 33, 2 (Jan. 2012), 191–198 2012

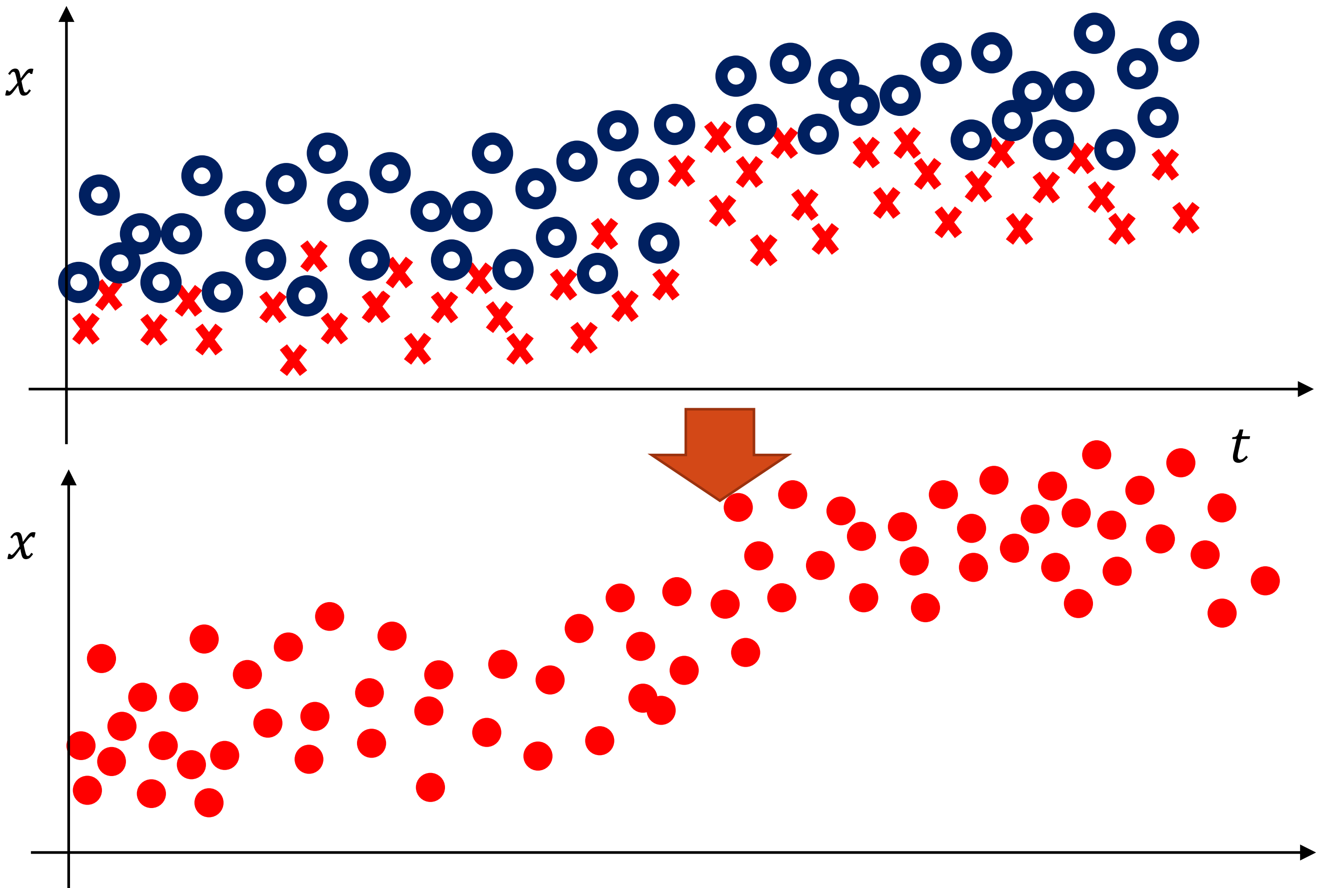
MONITORING THE RAW DATA

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

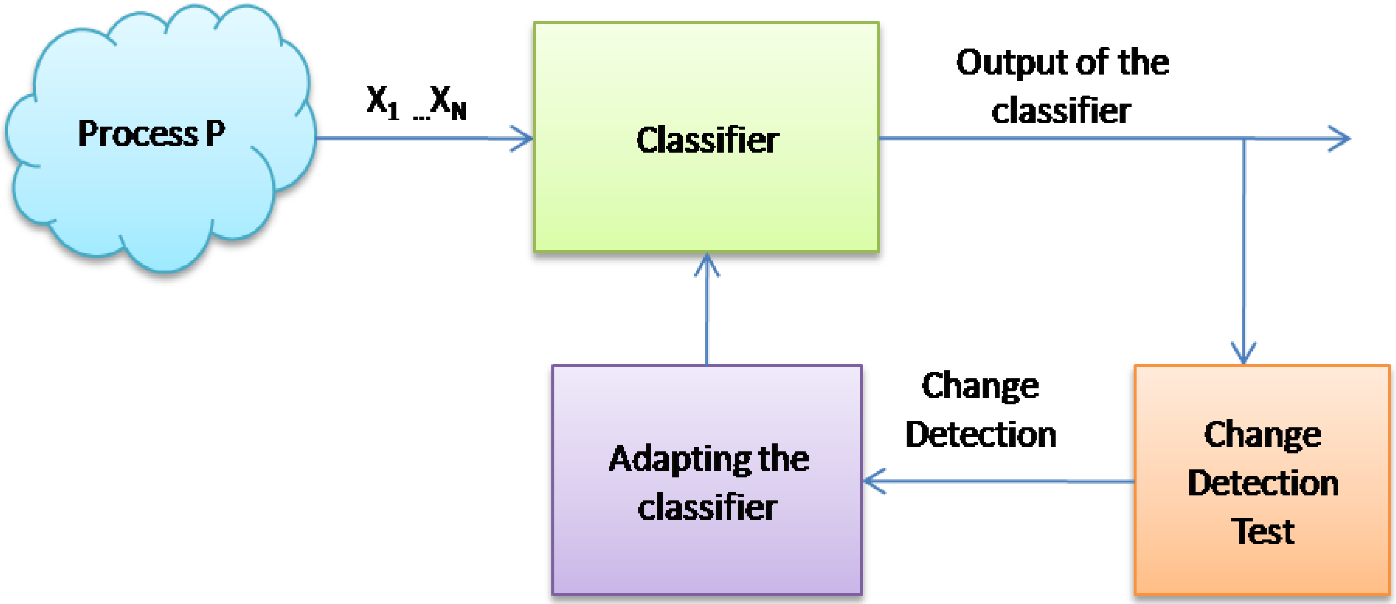


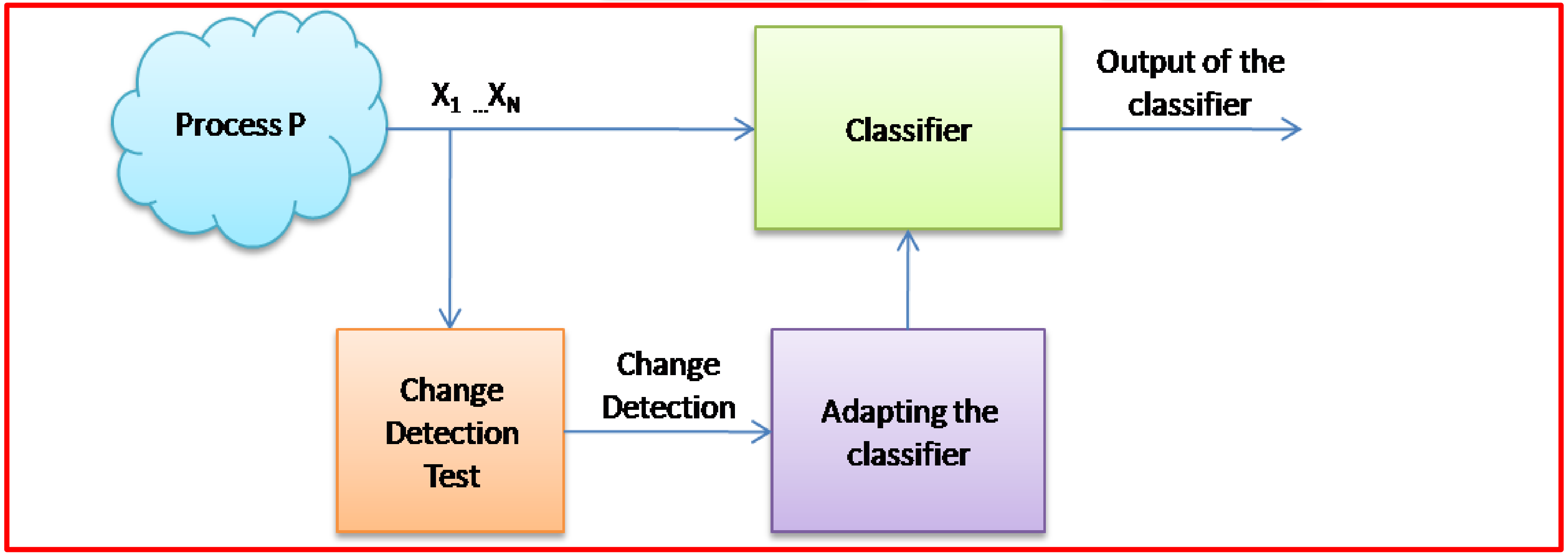
MONITORING THE RAW DATA

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



MONITORING THE RAW DATA





MONITORING THE RAW DATA

Pros:

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

Cons:

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

DETECTION TOOLS: ICI-BASED CDT

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

- the sample mean over data windows

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

- a power-law transform of the sample variance

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

$\mathrm{S}(s)$ is the sample variance over window yielding $M(s)$

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

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

DETECTION TOOLS: CI-CUSUM

Several features from non-overlapping windows including

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

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

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

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

MONITORING (MULTIVARIATE) RAW DATA

One typically resort to:

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

MONITORING THE LOG-LIKELIHOOD

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

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

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

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

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

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

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

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 and take advantage of recurrent concepts

JIT classifiers leverage:

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

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

C. Alippi, G. Boracchi, M. Roveri "*Just In Time Classifiers for Recurrent Concepts*" IEEE Transactions on Neural Networks and Learning Systems, 2013. vol. 24, no.4, pp. 620 -634 (Outstanding Paper Award 2016)

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- $\qquad \mathcal{U}(C_i, \{(x_t, y_t)\}) \rightarrow C_i$;
$\quad$ **end**
7- $\quad$ **if** ($\mathcal{D}(C_i) = 1$) **then**
8- $\qquad i = i + 1$;
9- $\qquad \Upsilon(C_{i-1}) \rightarrow (C_k, C_l)$;
10- $\qquad C_i = C_l$;
11- $\qquad C_{i-1} = C_k$;
12- $\qquad Z_{\text{rec}} = \bigcup_{\substack{\mathcal{E}(C_i, C_j)=1 \\ 0 \le j < i}} Z_j$;
$\quad$ **end**
13- $\quad$ **if** (*y_t is not available*) **then**
14- $\qquad \widehat{y}_t = K(Z_i \cup Z_{\text{rec}}, x_t)$.
$\quad$ **end**
end

Concept Representations

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

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

AN EXAMPLE OF CONCEPT REPRESENTATIONS

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

- $Z_i = \{(\boldsymbol{x}_0, y_0), \dots, (\boldsymbol{x}_n, y_n)\}$: **supervised samples** provided during the i^{th} concept
- F_i **features describing** $p(\boldsymbol{x})$ of the i^{th} concept. We take:
  - the sample mean $M(\cdot)$
  - the power-low transform of the sample variance $V(\cdot)$

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

  extracted from **non-overlapping sequences**

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

JIT CLASSIFIERS: THE ALGORITHM

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

Concept Representations

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

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

Operators for Concepts

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

JIT CLASSIFIERS: THE ALGORITHM

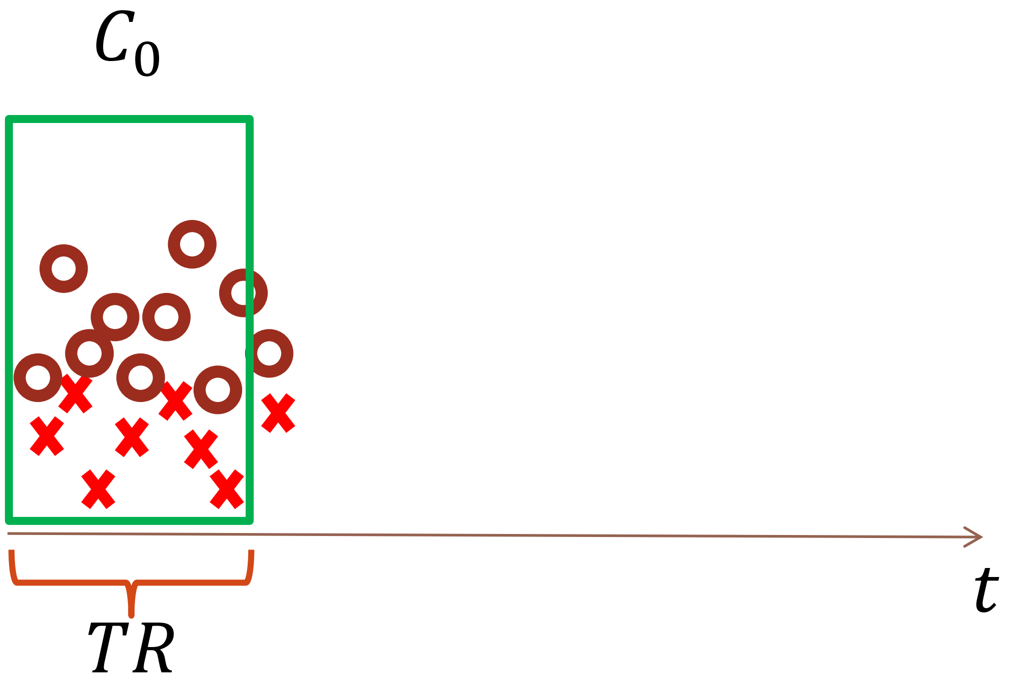
1- Build concept $C_0 = (Z_0, F_0, D_0)$ from the training sequence;
2- $Z_{\text{rec}} = \emptyset$ and $i = 0$;
3- **while** (x_t *is available*) **do**
4- $\quad \mathcal{U}(C_i, \{x_t\}) \rightarrow C_i$;
5- $\quad$ **if** (y_t *is available*) **then**
6- $\quad\quad \mathcal{U}(C_i, \{(x_t, y_t)\})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

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 $p(\boldsymbol{x})$ and $p(y|\boldsymbol{x})$ 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- $\quad \mathcal{U}(C_i, \{x_t\}) \to C_i$;
5- $\quad$ **if** *(y_t is available)* **then**
6- $\quad\quad \mathcal{U}(C_i, \{(x_t, y_t)\}) \to C_i$;
$\quad$ **end**
7- $\quad$ **if** $(\mathcal{D}(C_i) = 1)$ **then**
8- $\quad\quad i = i + 1$;
9- $\quad\quad \Upsilon(C_{i-1}) \to (C_k, C_l)$;
10- $\quad\quad C_i = C_l$;
11- $\quad\quad C_{i-1} = C_k$;
12- $\quad\quad Z_{\text{rec}} = \bigcup_{\substack{\mathcal{E}(C_i, C_j) = 1 \\ 0 \le j < i}} Z_j$;
$\quad$ **end**
13- $\quad$ **if** *(y_t is not available)* **then**
14- $\quad\quad \widehat{y}_t = K(Z_i \cup Z_{\text{rec}}, x_t)$.
$\quad$ **end**
end

During operations, each input sample is analyzed to

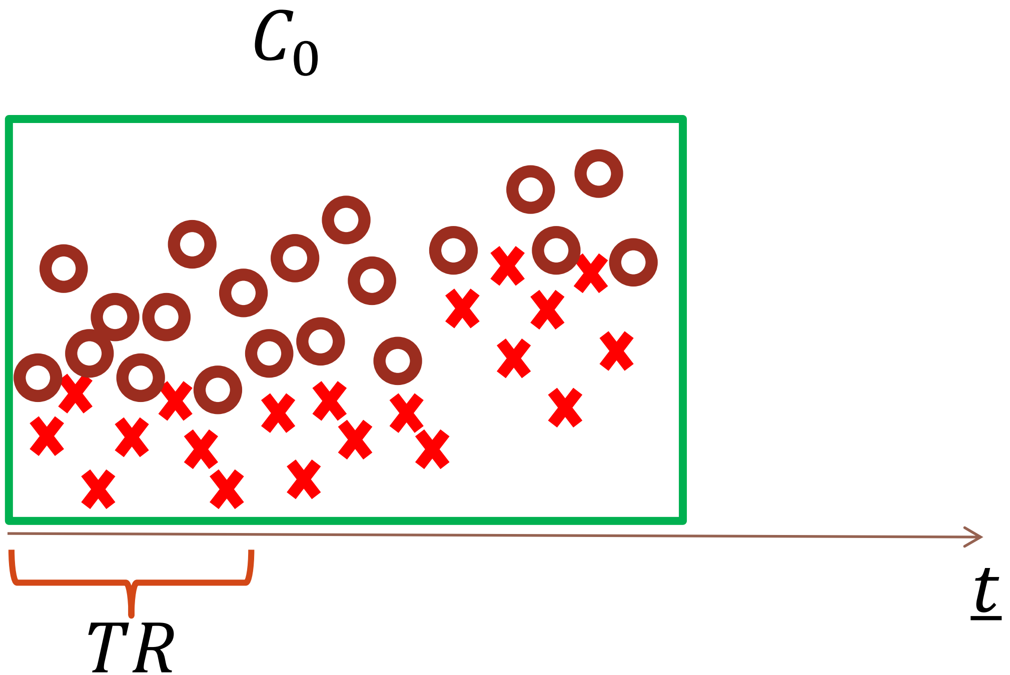
- Extract features that are appended to F_i
- Append supervised information in Z_i

thus updating the current concept representation

JIT CLASSIFIERS: CONCEPT UPDATE

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

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



JIT CLASSIFIERS: THE ALGORITHM

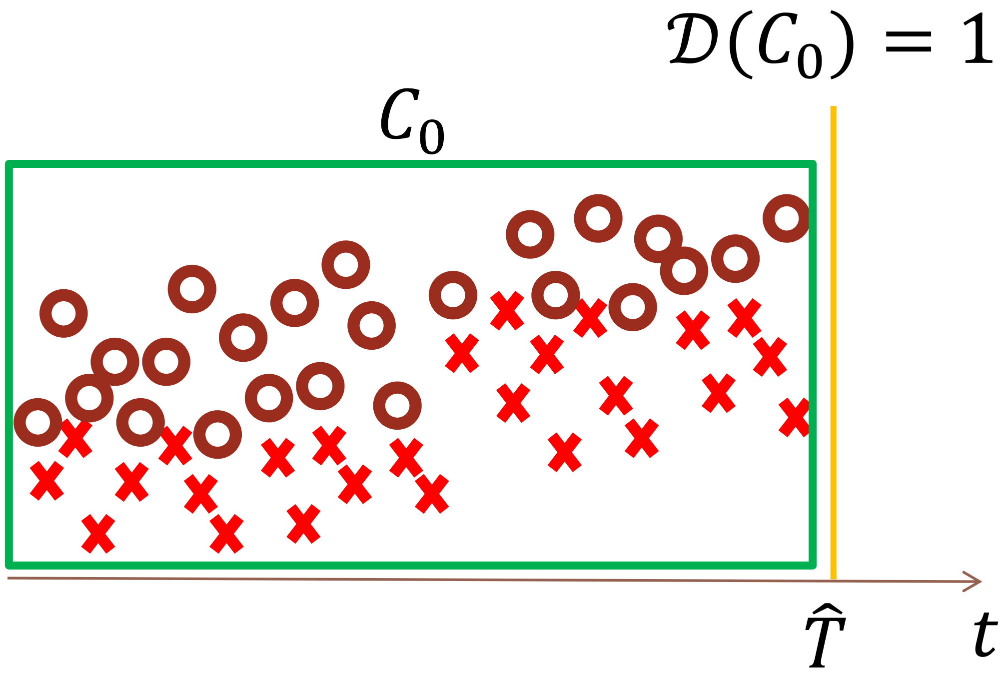
Build concept $C_0 = (Z_0, F_0, D_0)$ from the training sequence;
$Z_{\text{rec}} = \emptyset$ and $i = 0$;
while (*x_t is available*) **do**
  $\mathcal{U}(C_i, \{x_t\}) \to C_i$;
  if (*y_t is available*) **then**
    $\mathcal{U}(C_i, \{(x_t, y_t)\}) \to C_i$;
  end
  if ($\mathcal{D}(C_i) = 1$) **then**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

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

JIT CLASSIFIERS: CONCEPT DRIFT DETECTION

Determine when **features in *D*** are no more stationary

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



AN EXAMPLE OF DETECTION OPERATOR

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

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

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

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

JIT CLASSIFIERS: THE ALGORITHM

1- Build concept $C_0 = (Z_0, F_0, D_0)$ from the training sequence;

2- $Z_{\text{rec}} = \emptyset$ and $i = 0$;

3- **while** (x_t *is available*) **do**

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

5- $\quad$ **if** (y_t *is available*) **then**

6- $\quad\quad \mathcal{U}(C_i, \{(x_t, y_t)\}) \to C_i$;

$\quad$ **end**

7- $\quad$ **if** ($\mathcal{D}(C_i) = 1$) **then**

8- $\quad\quad i = i + 1$;

9- $\quad\quad \Upsilon(C_{i-1}) \to (C_k, C_l)$;

10- $\quad\quad C_i = C_l$;

11- $\quad\quad C_{i-1} = C_k$;

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

$\quad$ **end**

13- $\quad$ **if** (y_t *is not available*) **then**

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

$\quad$ **end**

end

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

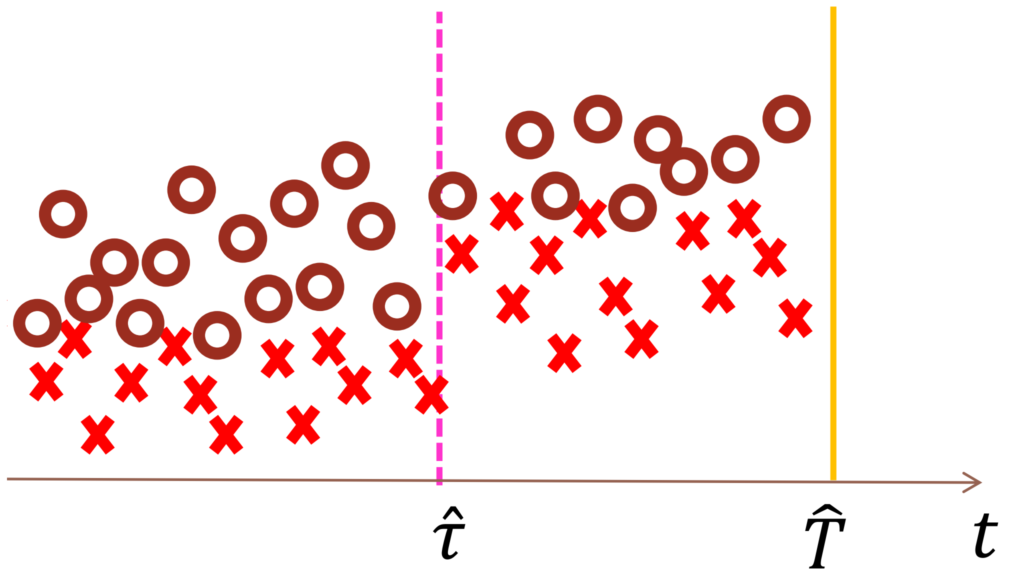
A new concept description is built

JIT CLASSIFIERS: CONCEPT SPLIT

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

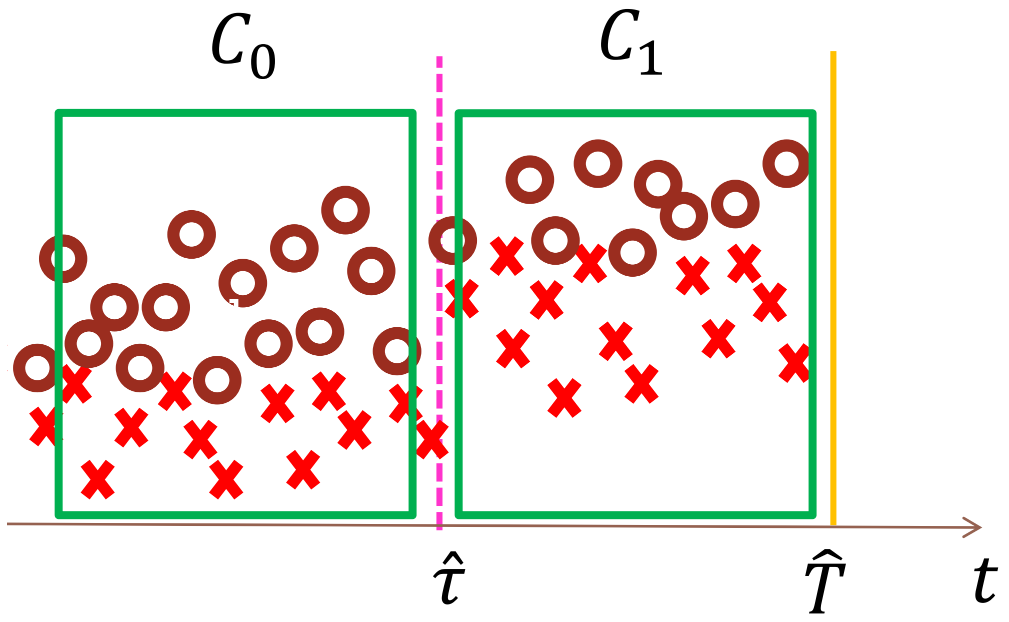
Uses statistical tools for performing an **offline** and **retrospective** analysis over the recent data, like:

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



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 $\widehat{T}$ are delayed w.r.t. the actual change point τ
- **Change-Point Methods** implement the following Hypothesis test on the feature sequence:

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

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

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

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**
        $i = i + 1$;
        $\Upsilon(C_{i-1}) \rightarrow (C_k, C_l)$;
        $C_i = C_l$;
        $C_{i-1} = C_k$;
        $\boxed{Z_{\text{rec}} = \bigcup_{\substack{\mathcal{E}(C_i, C_j)=1 \\ 0 \leq j < i}} Z_j;}$
    end
    if *(y_t is not available)* **then**
        $\widehat{y}_t = K(Z_i \cup Z_{\text{rec}}, x_t)$.
    end
end

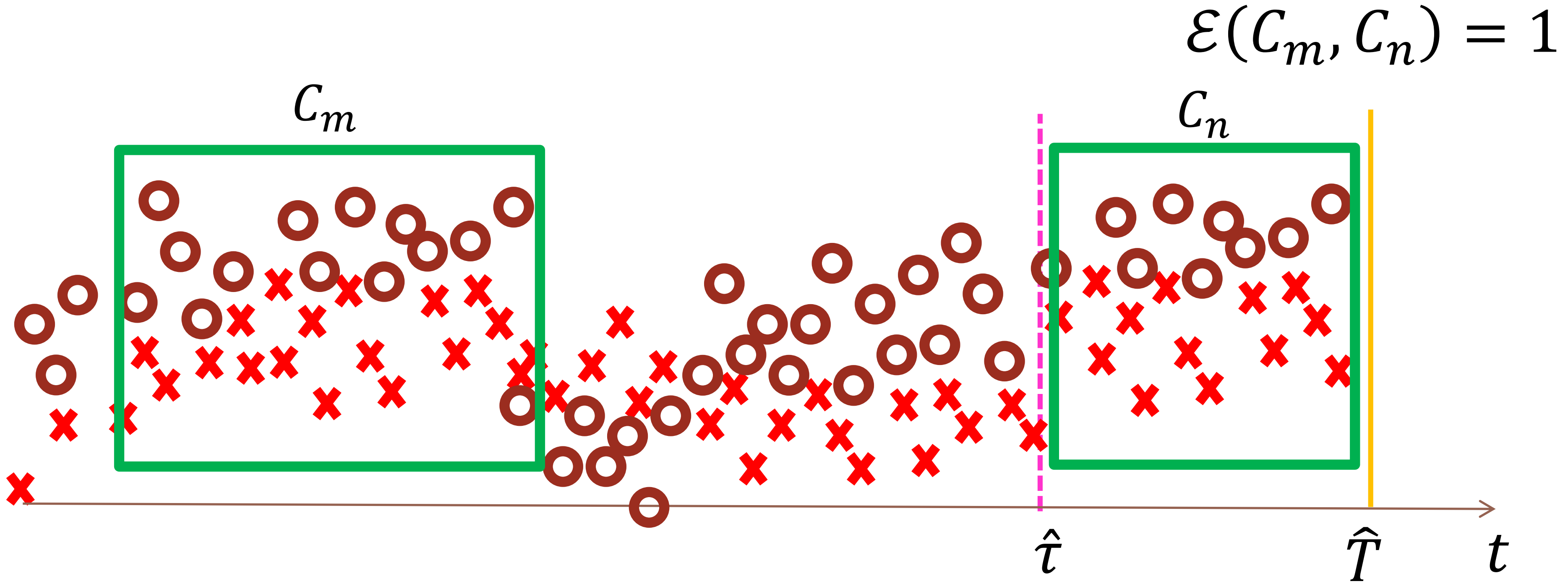
Look for concepts that are equivalent to the current one.

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



JIT CLASSIFIERS: THE ALGORITHM

1- Build concept $C_0 = (Z_0, F_0, D_0)$ 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

The classifier K is reconfigured using all the available supervised couples

COMPARING WINDOWS

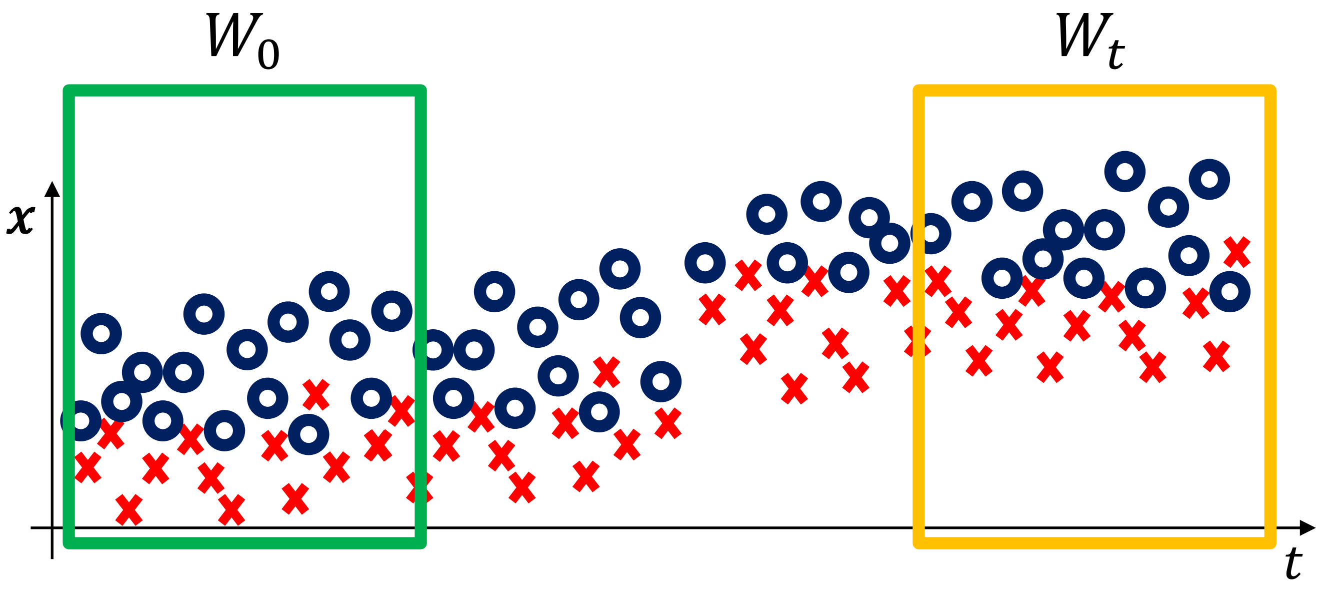
THE MOTIVATING IDEA

Detect CD at time t by comparing two different windows.

In practice, one computes:

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

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



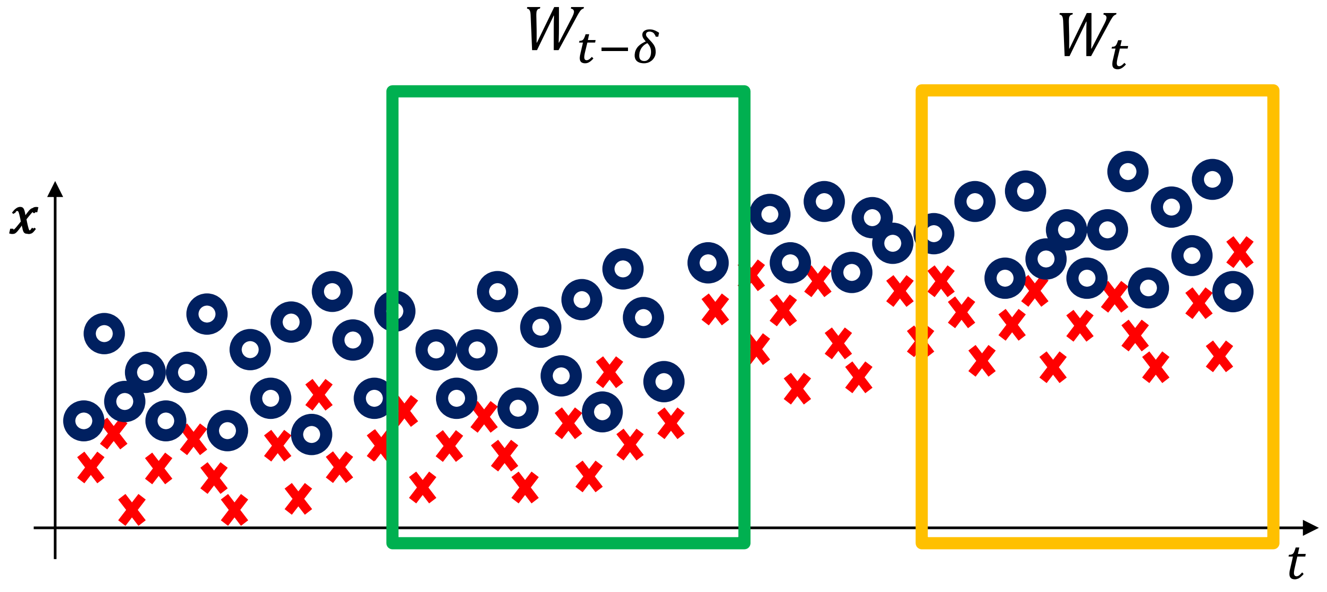
THE MOTIVATING IDEA

Detect CD at time t by comparing two different windows.

In practice, one computes:

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

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



THE MOTIVATING IDEA

Pro:

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

Cons:

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

WINDOW COMPARISON: MAIN APPROACHES

- The averages over two adjacent windows (ADWIN)

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

WINDOW COMPARISON: MAIN APPROACHES

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

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

WINDOW COMPARISON: MAIN APPROACHES

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

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

G. Ditzler and R. Polikar, *"Hellinger distance based drift detection for nonstationary environments"* in Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2011 IEEE Symposium on, April 2011, pp. 41–48.

WINDOW COMPARISON: MAIN APPROACHES

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

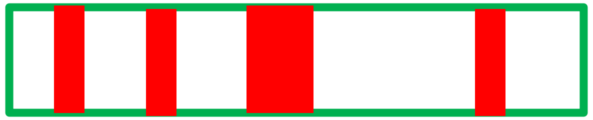
Kawahara, Y. and Sugiyama, M. "*Sequential change-point detection based on direct density-ratio estimation*". Statistical Analysis and Data Mining, 5(2):114–127, 2012.

WINDOW COMPARISON: TESTING EXCHANGABILITY

In stationary conditions, all data are i.i.d., thus if we

- Select a training set and a test set in a window

- Select another TR and TS pair after reshuffling the two



the empirical error of the two classifiers should be the same

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

WINDOW COMPARISON: PAIRED LEARNERS

Two classifiers are trained

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

During operation

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

Adaptation consists in **replacing** S by R_w

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

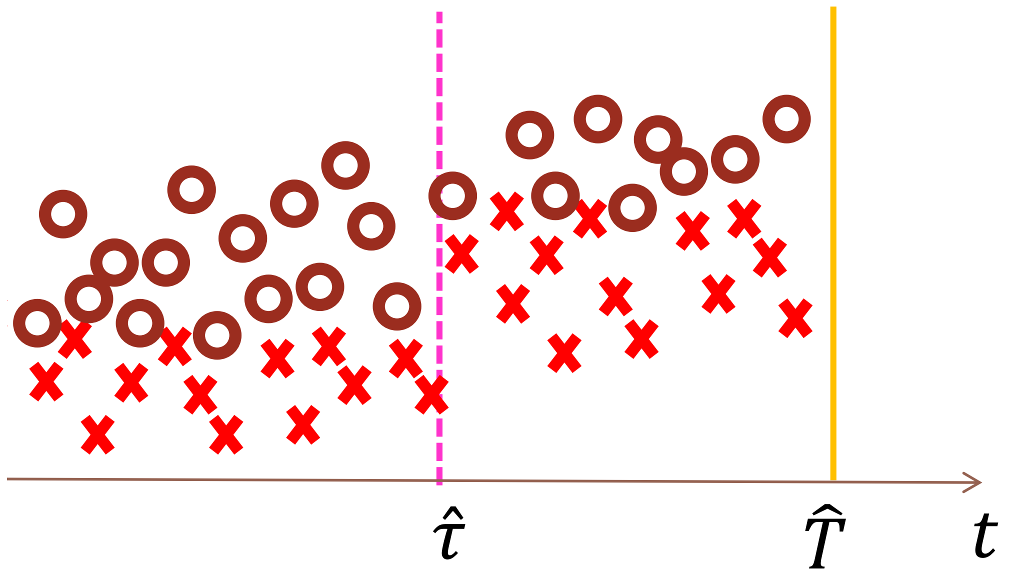
REMARKS ON ACTIVE APPROACHES

COMMENTS FROM MY PERSONAL EXPERIENCE

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

COMMENTS FROM MY PERSONAL EXPERIENCE

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



COMMENTS FROM MY PERSONAL EXPERIENCE

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

D. M. Hawkins, P. Qiu, and C. W. Kang, *"The changepoint model for statistical process control"* Journal of Quality Technology, vol. 35, No. 4, pp. 355–366, 2003.

COMMENTS FROM MY PERSONAL EXPERIENCE

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

NSL: PASSIVE APPROACHES

Giacomo Boracchi[1] and Gregory Ditzler[2]

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

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

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

NSL: A PASSIVE VIEW

(Back to)Adaptation Strategies

- Active: the learner K is combined with a tool to detect concept drift that pilots adaptation.
- Passive: assume the process undergoes continuous adaptation.

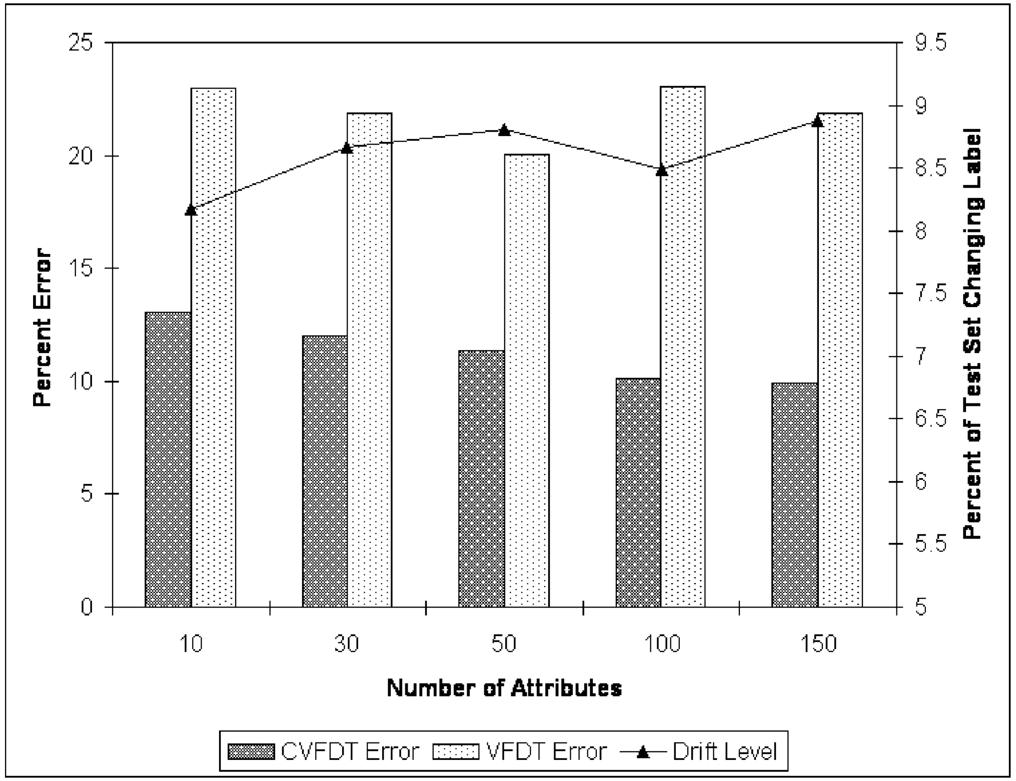
WHAT IS PASSIVE LEARNING?

- It's all about the adaptation mechanism employed to cope with the change!
  - **active approaches** rely on an explicit detection of the change in the data distribution to activate an adaptation mechanism
  - **passive approaches** continuously update the model over time (without requiring an explicit detection of the change)
    - Passively assume that some type of change is present in the data stream
- Is one approach more correct than another? No!
  - Benchmarking Dilemma – What makes an algorithm successful? Detection delay? Classification error? Can we make the comparison fair?
- Dicotomy of Passive Learning
  - Single Classifier
  - Ensemble
  - Batch versus Online

SINGLE CLASSIFIER MODELS

CONCEPT-ADAPTING VERY FAST DECISION TREE

- Decision tree models are very popular with traditional supervised learning, but how can we use them with concept drift in the data stream?
- **Concept-adapting Very Fast Decision Tree** learner (CVFDT) is an online decision tree algorithm
- CVFDT attempts to stay current by growing alternative sub-trees until an old sub-tree is accurate then it is replaced
  - The current split at a node in the tree may not be optimal throughout all of time
  - Use windows of samples to evaluate the tree to remove the low quality sub-trees
  - Time complexity of O(1)



STOCHASTIC GRADIENT DESCENT & ELM

- **Stochastic gradient descent (SGD)** is an online representation of classical "batch" gradient descent algorithm
  - Commonly used to train neural networks
  - Mini-batches are sometimes used at each timestep to achieve a smoother convergence in the function being minimized
- SGD has been implemented using a linear classifier minimizing a hinge loss function for learning in nonstationary environment
  - Massive Online Analysis (more about this later) has an implementation of this approach in their software package
- An **Extreme Learning Machine** has been also combined with a time-varying NN for learning in nonstationary environments

ONLINE INFORMATION NETWORK

- OLIN (**online information network**) is fuzzy-logic algorithm that repeatedly learns a from sliding window of examples in order to update the existing model
  - Replace it by a former model
  - Or construct a new model if a major concept drift is detected
  - Borrow from active and passive approaches
- OLIN updates the fuzzy-info network by identifying non-relevant nodes, adding new layers and replacing the output layer when new data arrive. If a new concept was presented then a new fuzzy-info network is learned.
  - A new concept could be identified by a statistically significant drop is the classification accuracy on the latest labeled data

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

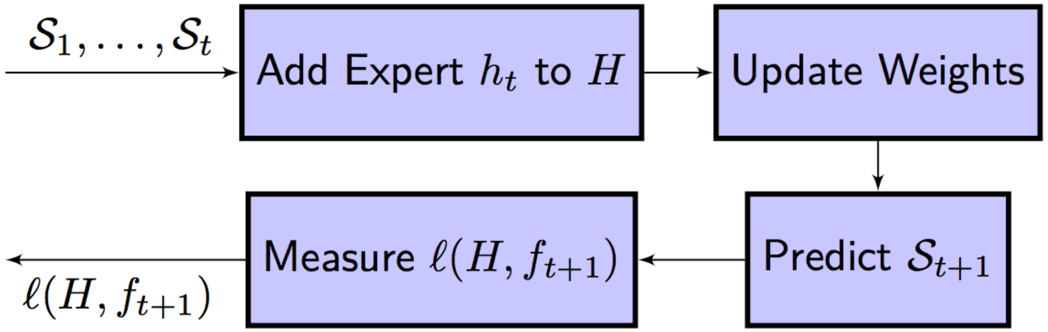
Sounds like we could have two opposing ideas!

ENSEMBLE MODELS

Balancing the stability-plasticity dilemma

ENSEMBLE MODELS

- Ensemble based approaches provide a natural fit to the problem of learning in a nonstationary setting
  - Ensembles tend to be more accurate than single classifier-based systems due to reduction in the variance of the error
  - (Stability) They have the flexibility to easily incorporate new data into a classification model when new data are presented, simply by adding new members to the ensemble
  - (Plasticity) They provide a natural mechanism to forget irrelevant knowledge, simply by removing the corresponding old classifier(s) from the ensemble



ENSEMBLE MODELS

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

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

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

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

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

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

ENSEMBLE MODELS

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

TAXONOMY OF CD ADAPTATION IN ENSEMBLE

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

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

SEA

A **fixed-size ensemble** that performs

- batch learning
- structural update to adapt to concept drift

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

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

Prune the ensmeble to improve the performance

DWM

- Littlestone's **Weighted Majority** algorithm is an ensemble **online algorithm** for combining multiple classifiers learning from a stream of data; however, keeping the same classifiers in the ensemble for the duration of the stream could be suboptimal in a nonstationary setting

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

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

- Ensemble size is not fixed

J. Zico Kolter, M.A. Maloof, "Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts," *Journal of Machine Learning Research 8*, 2007.

DIVERSITY, DIVERSITY, DIVERSITY!

- **Diversity** in an ensemble has been shown to be **benefical in ensembles** that have been learned from a static distribution, but how can diversity be used in a nonstationary stream
- 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

Add Oza reference

ONLINE BAGGING & BOOSTING

- **Central Concept**
  - Bagging and boosting works well for batches of data; however, these approaches are not equipped to handle data streams. Oza developed approaches to perform this batch to online version
  - Issue: Online bagging and boosting does not account for concept drift
  - Parallel to Active: Implement Change detection to reset classifiers

---

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathscr{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathscr{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

OnlineBoosting($\mathbf{h_M}$, $OnlineBase$, d)

- Set the example's "weight" $\lambda_d = 1$.
- For each base model h_m, ($m \in \{1, 2, \ldots, M\}$) in the ensemble,
  - 1. Set k according to $Poisson(\lambda_d)$.
  - 2. Do k times
    $h_m = OnlineBase(h_m, d)$
  - 3. If $h_m(d)$ is the correct label,
    - then
      - $\lambda_m^{sc} \longleftarrow \lambda_m^{sc} + \lambda_d$
      - $\lambda_d \longleftarrow \lambda_d \left(\frac{N}{2\lambda_m^{sc}}\right)$
    - else
      - $\lambda_m^{sw} \longleftarrow \lambda_m^{sw} + \lambda_d$
      - $\lambda_d \longleftarrow \lambda_d \left(\frac{N}{2\lambda_m^{sw}}\right)$

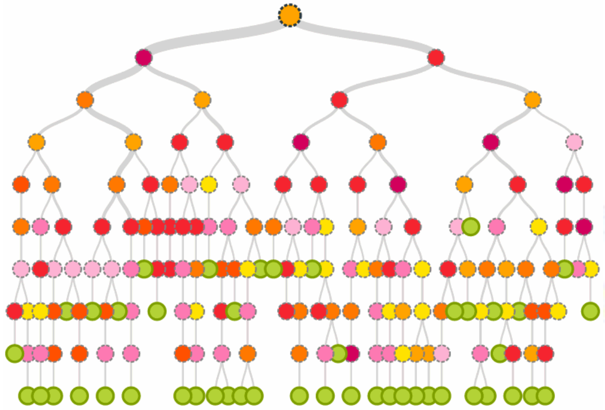
To classify new examples:

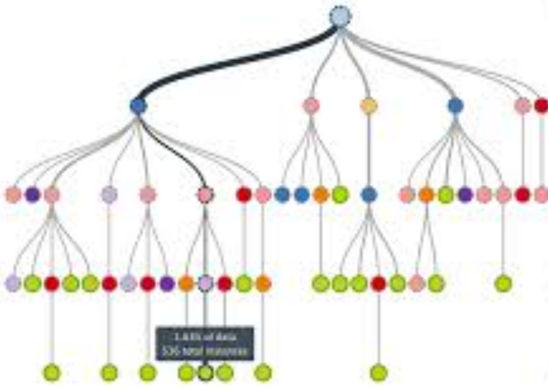
- For each $m \in \{1, 2, \ldots, M\}$
  Calculate $\epsilon_m = \frac{\lambda_m^{sw}}{\lambda_m^{sc} + \lambda_m^{sw}}$ and $\beta_m = \frac{\epsilon_m}{1 - \epsilon_m}$
- Return $h(x) = argmax_{c \in C} \sum_{m: h_m(x) = y} log \frac{1}{\beta_m}$.

Add bifet reference

ADAPTIVE SIZE HOEFFDING TREES (ASHT)

- Online bagging & boost are not designed for learning in nonstationary environments. How can we modify the base classifier to learn concepts over time?
- Thoughts on pruning trees
  - In a slowly changing environment, larger trees can better capture the concepts
  - In a quickly changing environment, smaller trees can adapt quickly to new concepts with new examples from the stream
  - Solution: Ensembles with different size Hoeffding trees
- Examples from the stream are learned by different size Hoeffding trees. These trees are "reset" at different time intervals
  - Short trees are reset more often than larger trees
  - Opportunity: Active approach to tell us when to reset a tree (small and large)





ACCURACY UPDATED ENSEMBLE

- **Motivation**
  - Build a data stream mining algorithm that is not specific to one type of drift, but rather robust on
  - The "size" of a classifier is important for many application, not only for memory limitations, but also for learning in a nonstationary environment
- The Accuracy Updated Ensemble (AUE2) learns a classifiers on the most recent data and weights them according to their accuracy; however, the base classifier is limited in its memory footprint to avoid unnecessarily large models
  - Weaker classifiers are discarded with the most recent (accurate) classifier
  - Potentially incorporates catastrophic forgetting

D. Brzezinski and J. Stefanowski, "Reacting to Different Types of Concept Drift: The Accuracy Updated Ensemble Algorithm," IEEE Transactions on Neural Networks, vol. 25, no. 1, 2014

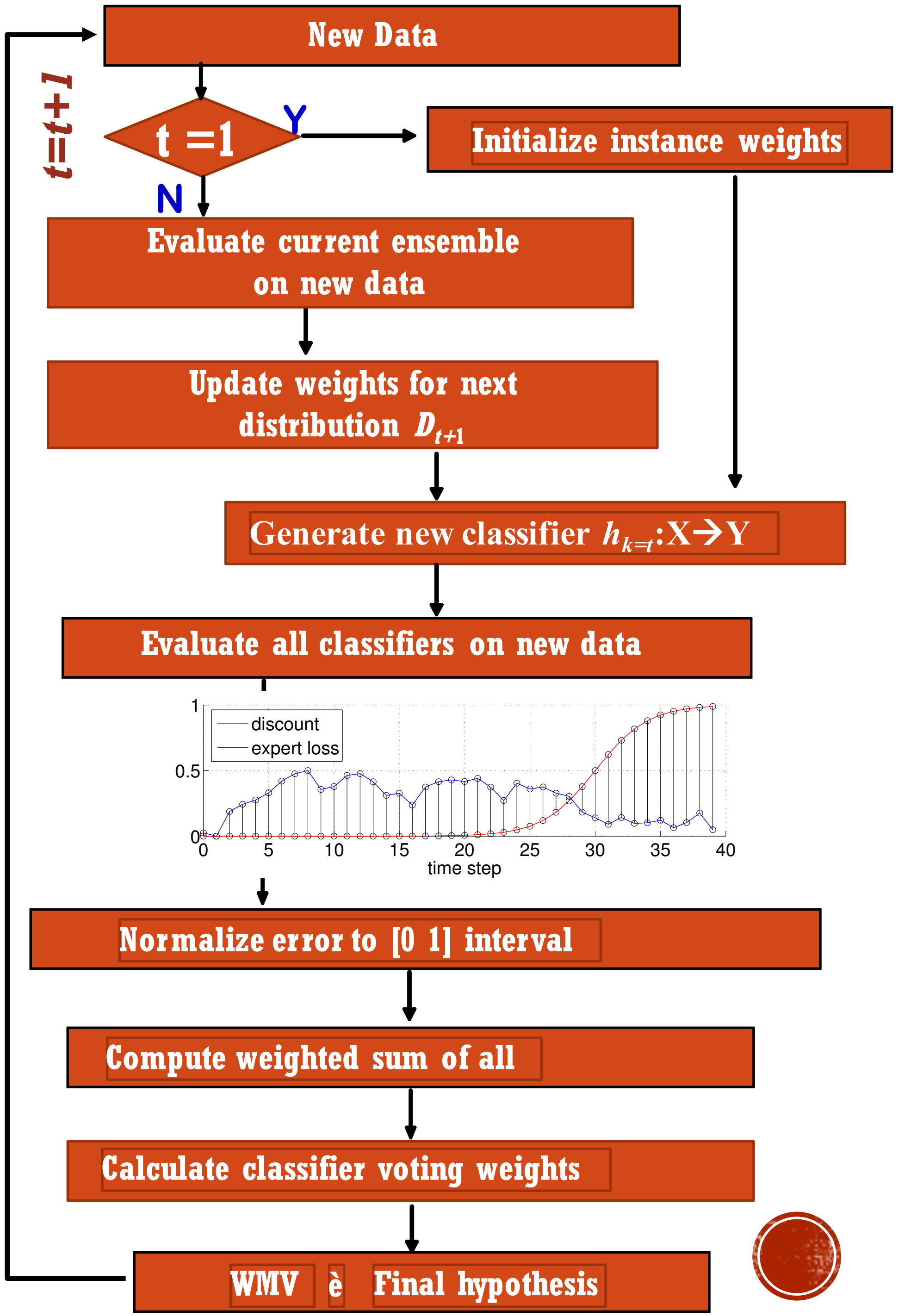
LEARN++.NSE

- **Learn++** is an incremental learning ensemble to learn classifiers from streaming batches of data
  - Think boosting, but for incremental learning
  - Batches of data are assumed to be sampled iid from a distribution
- Learn++ works well on static distributions; however, **classifier weights remain fixed**, which is an ill-advised strategy if each batch of data is not sampled iid. Especially the testing data!
- **Solution**: Learn++.NSE: Similar to DWM, Learn++.NSE extends Learn++ for learning in nonstationary environments (NSE)

Polikar R., Udpa L., Udpa, S., Honavar, V., "Learn++: An incremental learning algorithm for supervised neural networks," IEEE Transactions on System, Man and Cybernetics (C), vol. 31, no. 4, pp. 497-508, 2001

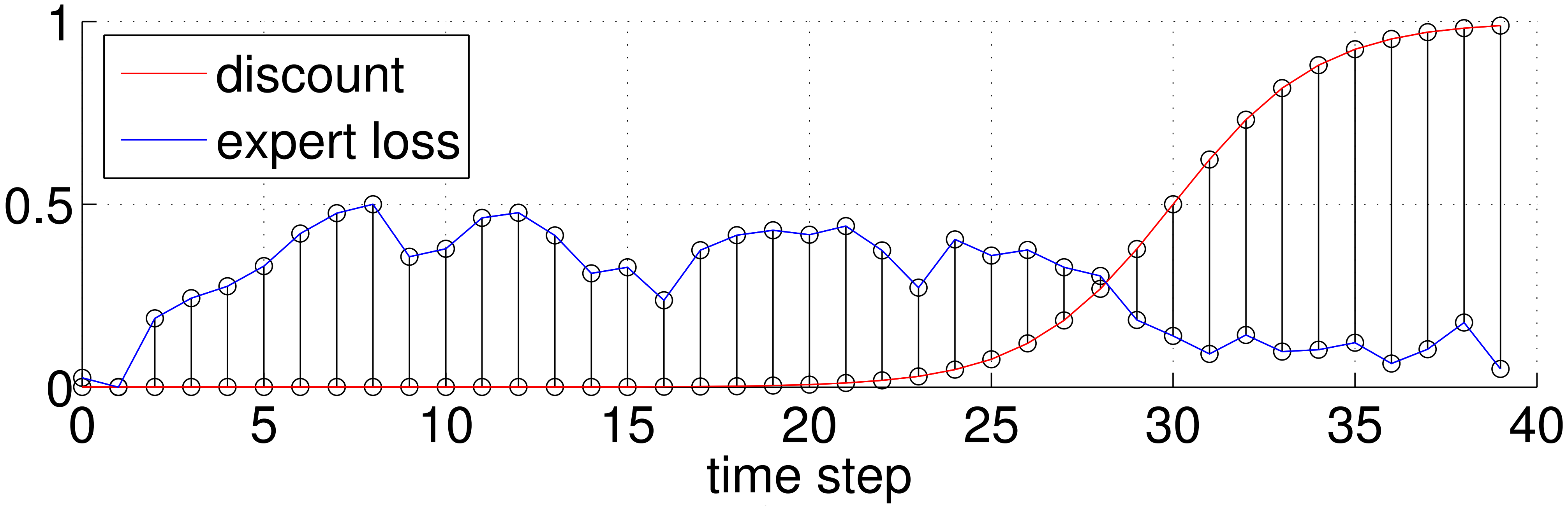
LEARN++.NSE

- **Learn^{++}.NSE**: incremental learning algorithm for concept drift
  - Generate a classifier with each new batch of data, compute a pseudo error for each classifier, apply time-adjusted weighting mechanism, and call a weighted majority vote for an ensemble decision
    - Recent pseudo errors are weighted heavier than old errors
  - Works very well on a broad range of concept drift problems



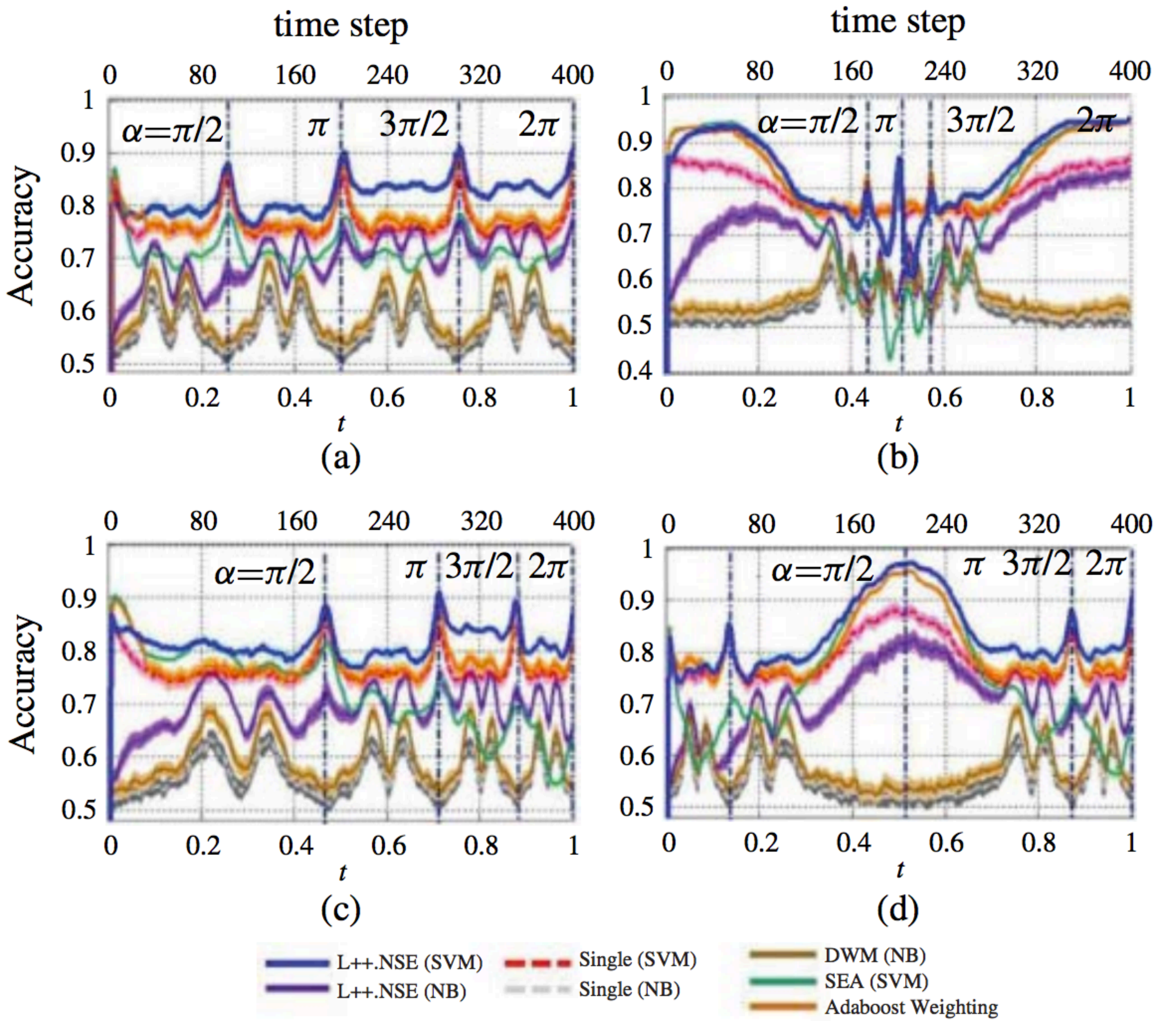
Elwell R. and Polikar R., "Incremental Learning of Concept Drift in Nonstationary Environments" IEEE Transactions on Neural Networks, vol. 22, no. 10, pp. 1517-1531, 2011.

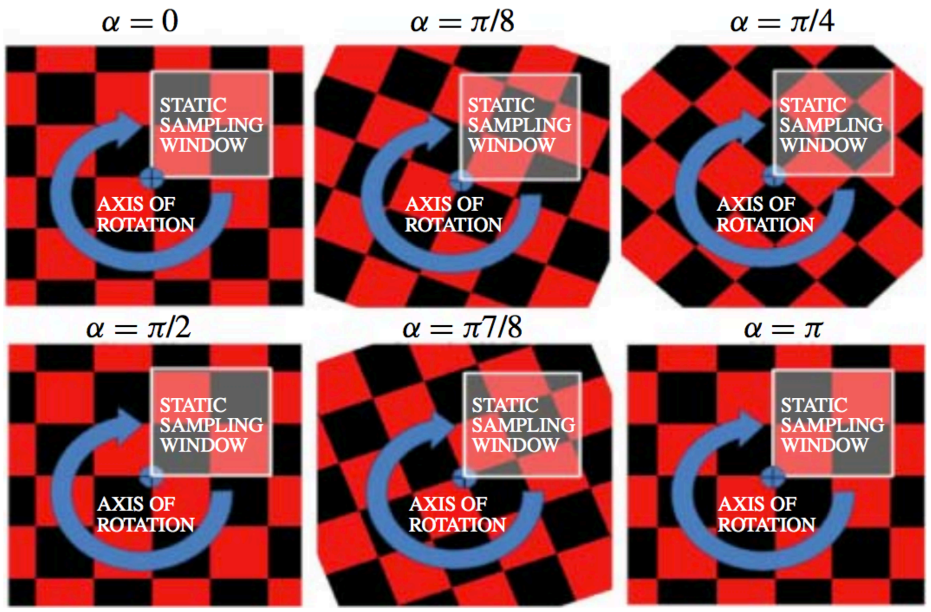
DISCOUNTED LOSS (ERROR)



$$\hat{\ell}_t(h_j) = \sum_{k=1}^{t} \rho_{t-k} \ell_k(h_j)$$

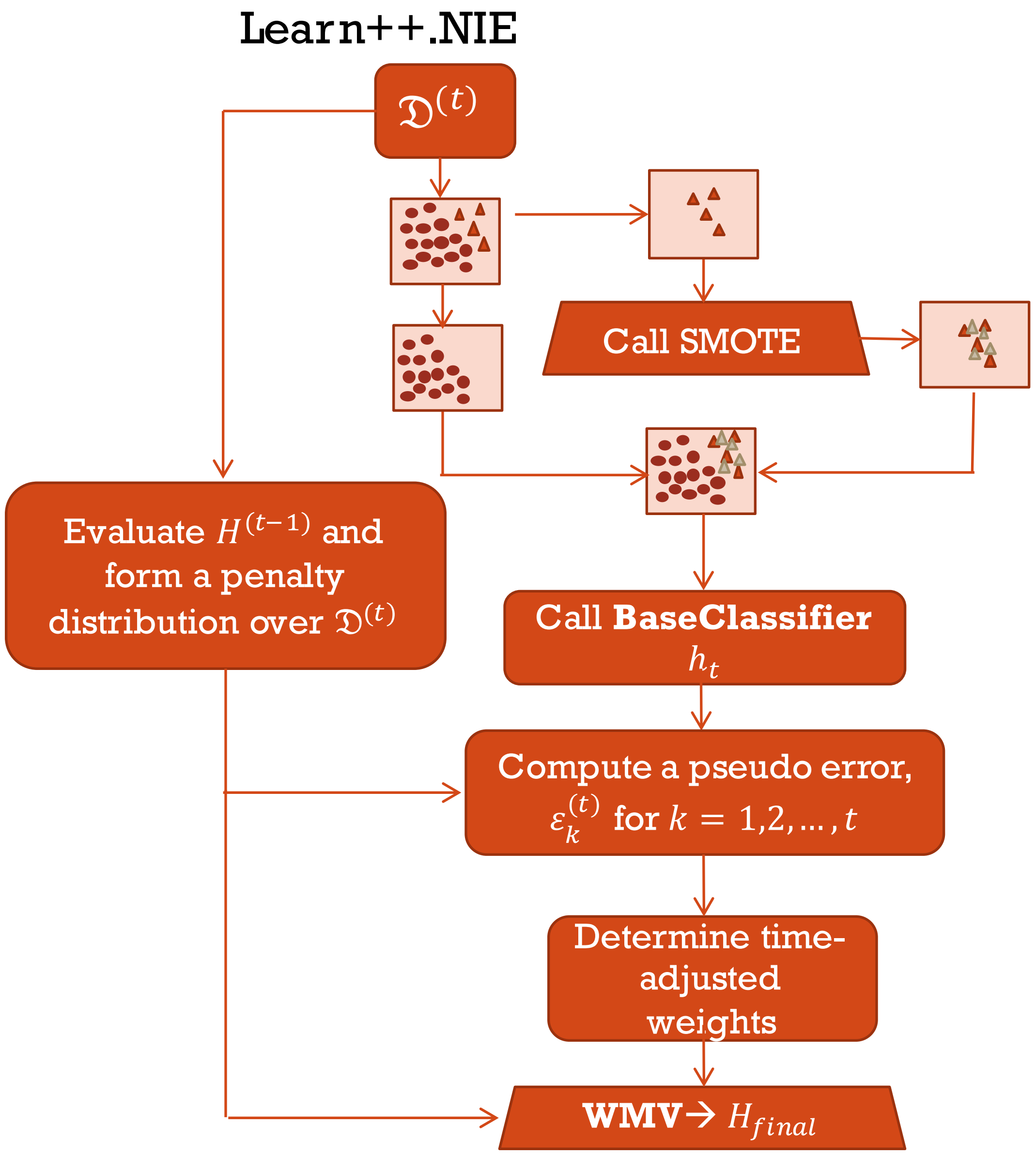
LEARN++.NSE IN ACTION



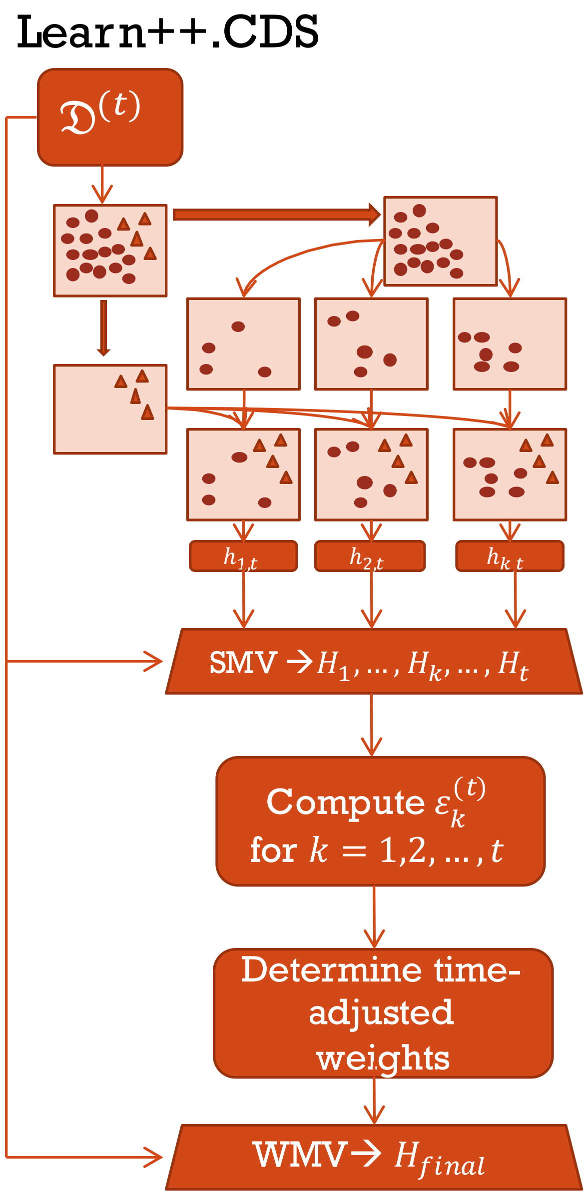


IMBALANCED & NSE

- **Class Imbalance**
  - Learners tend to bias themselves towards the majority class
    - Minority class is typically of great importance
  - Many concept drift algorithms tend to use **error or a figure of merit derived from error** to adapt to a nonstationary environment
- **Learn^{++}.CDS** {*Concept Drift with SMOTE*}
  - Apply SMOTE to Learn^{++}.NSE
    - Learn^{++}.NSE works well on problems involving concept drift
    - SMOTE works well at increasing the recall of a minority class
- **Learn^{++}.NIE** {*Nonstationary and Imbalanced Environments*}
  - Classifiers are replaced with sub-ensembles
    - Sub-ensemble is applied to learn a minority class
  - Voting weights are assigned based on figures of merit besides a class independent error
- Other Approaches: SERA, UCB, muSERA, REA



Learn++.NIE
$\mathfrak{D}^{(t)}$
Call SMOTE
Evaluate $H^{(t-1)}$ and form a penalty distribution over $\mathfrak{D}^{(t)}$
Call **BaseClassifier** h_t
Compute a pseudo error, $\varepsilon_k^{(t)}$ for $k = 1,2,\dots,t$
Determine time-adjusted weights
WMV→ H_{final}



Learn++.CDS
$\mathfrak{D}^{(t)}$
$h_{1,t}$
$h_{2,t}$
$h_{k\ t}$
SMV → $H_1, \dots, H_k, \dots, H_t$
Compute $\varepsilon_k^{(t)}$ for $k = 1,2,\dots,t$
Determine time-adjusted weights
WMV→ H_{final}

A GLIMPSE AT ENSEMBLE ERROR

- Assuming the loss is convex in its first argument, the ensemble's expected loss becomes

$$\mathbb{E}_T[\ell(H, f_T)] \leq \sum_{k=1}^{t} w_{k,t}\mathbb{E}_T[\ell(h_k, f_T)]$$

  which is a weighted average each expert's loss
  - Generally not computable and is difficult to interpret

- Decomposing the loss using existing domain adaptation theory with prior work leads to the bound

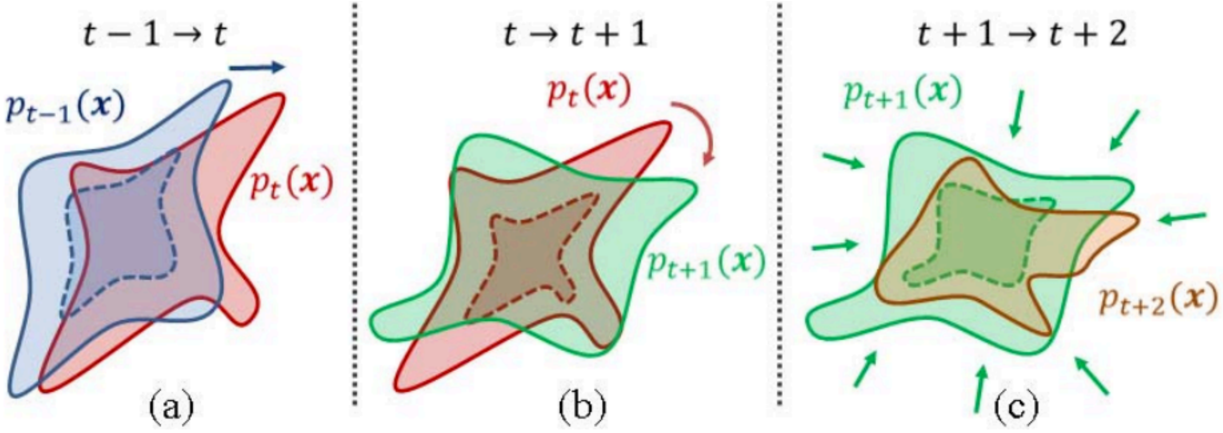
$$\begin{aligned}\mathbb{E}_T[\ell(H, f_T)] &\leq \sum_{k=1}^{t} w_{k,t}\mathbb{E}_T[\ell(h_k, f_T)] \\ &\leq \sum_{k=1}^{t} w_{k,t}\left(\mathbb{E}_k[\ell(h_k, f_k)] + \mathbb{E}_T[\ell(f_k, f_T)] + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}\left(\mathcal{U}_k, \mathcal{U}_T\right)\right)\end{aligned}$$

- Looks bad! But interpretable!
  Loss(ensemble) < weightedSum(training loss + labeling disagreement + divergence of unlabeled data)
  - Real & virtual drifts have been defined in literature, but now related to loss!

INITIALLY LABELED ENVIRONMENTS

INITIALLY LABELED ENVIRONMENTS

- All previous learning scenarios assumed a supervised learning setting
  - Transductive and Semi-Supervised was discussed
  - 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



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

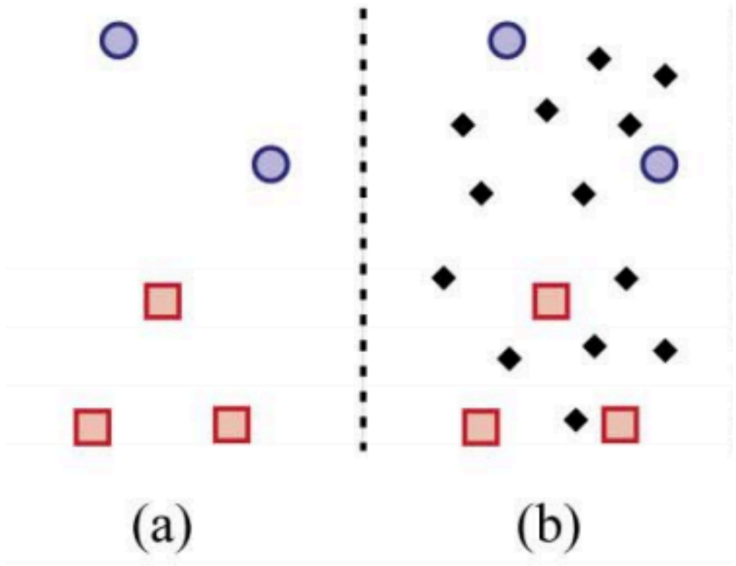
COMPACTED OBJECT SAMPLE EXTRACTION

Initially Labeled Data

Compose

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, vol. 25, no. 1, pp. 12-26, 2014

COMPACTED OBJECT SAMPLE EXTRACTION



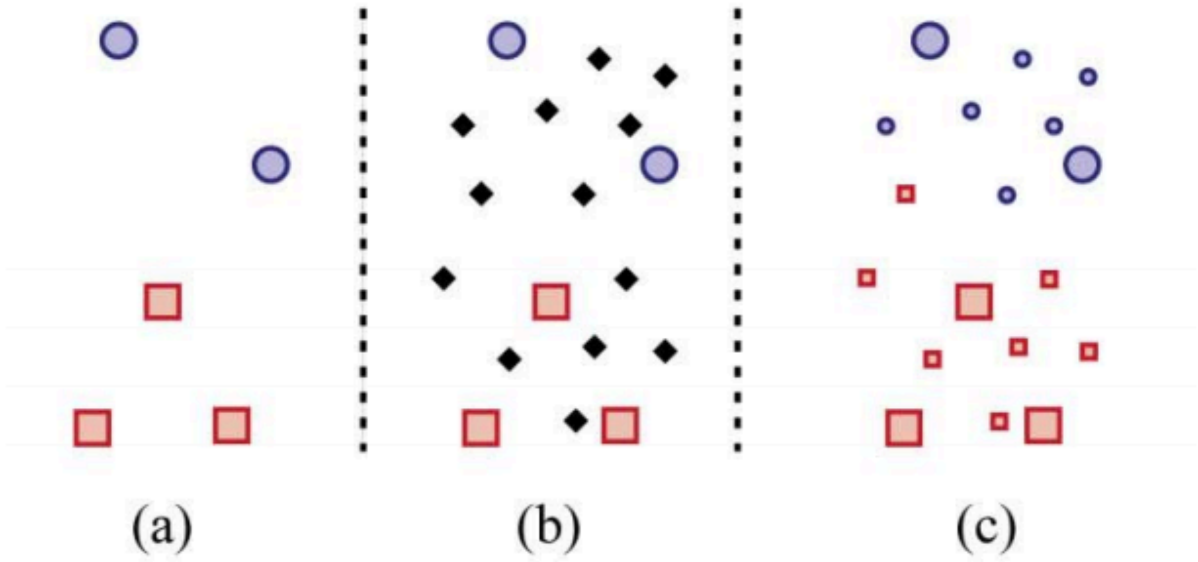
Initially Labeled Data

Receive Unlabeled Data

Compose

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, vol. 25, no. 1, pp. 12-26, 2014

COMPACTED OBJECT SAMPLE EXTRACTION



Initially Labeled Data

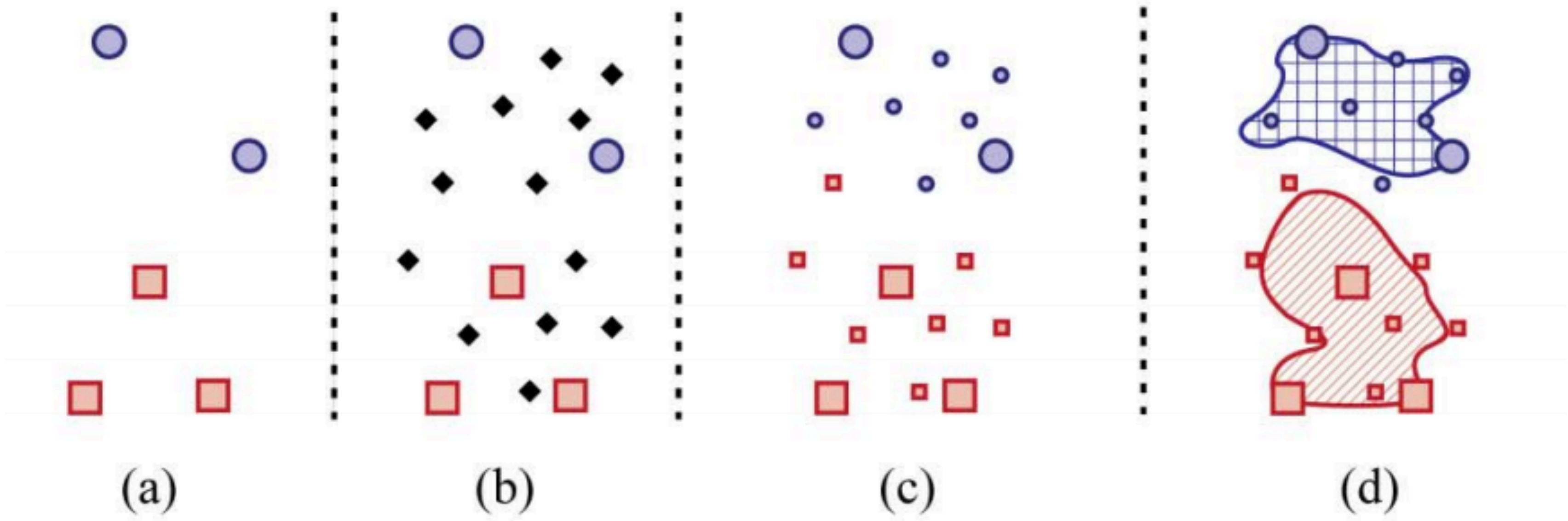
Receive Unlabeled Data

Classify Using SSL

Compose

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, vol. 25, no. 1, pp. 12-26, 2014

COMPACTED OBJECT SAMPLE EXTRACTION



Initially Labeled Data

Receive Unlabeled Data

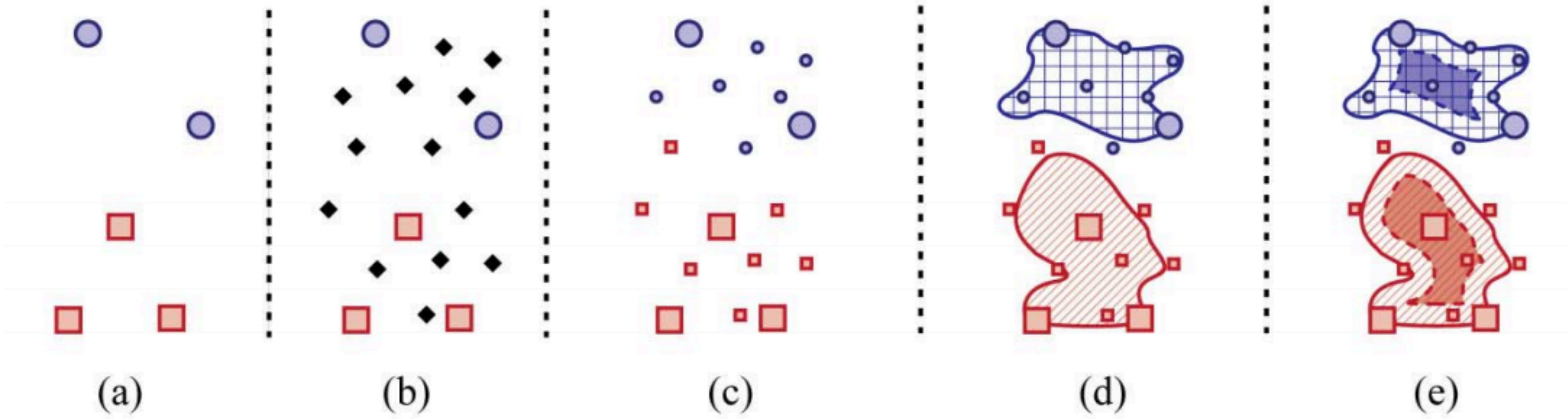
Classify Using SSL

Construct a Boundary

Compose

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, vol. 25, no. 1, pp. 12-26, 2014

COMPACTED OBJECT SAMPLE EXTRACTION



Initially Labeled Data

Receive Unlabeled Data

Classify Using SSL

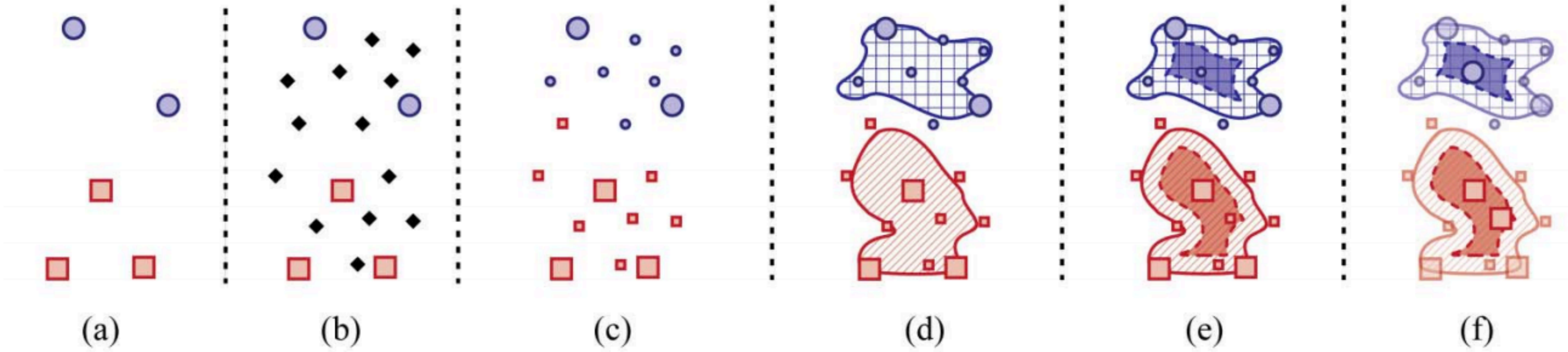
Construct a Boundary

Compact the Boundary

Compose

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, vol. 25, no. 1, pp. 12-26, 2014

COMPACTED OBJECT SAMPLE EXTRACTION

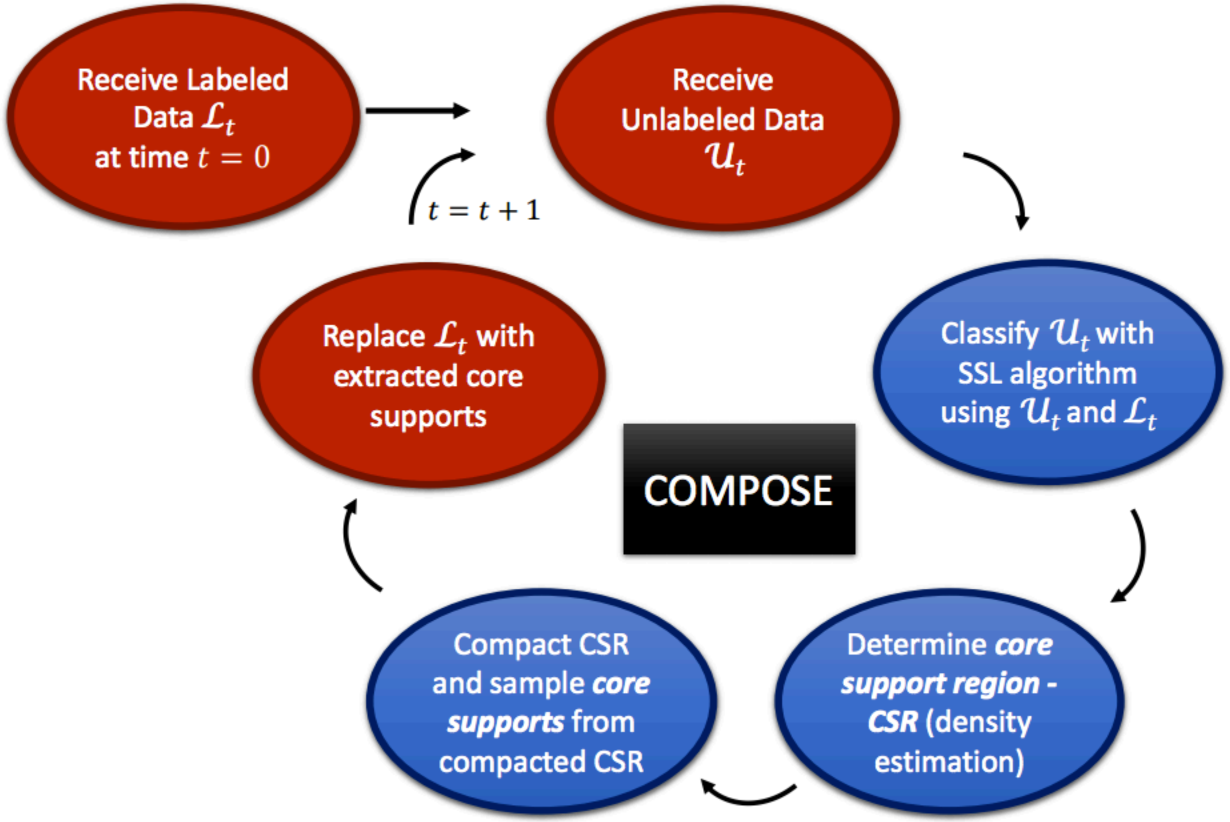


Initially Labeled Data | Receive Unlabeled Data | Classify Using SSL | Construct a Boundary | Compact the Boundary | Extract Core Set

Compose

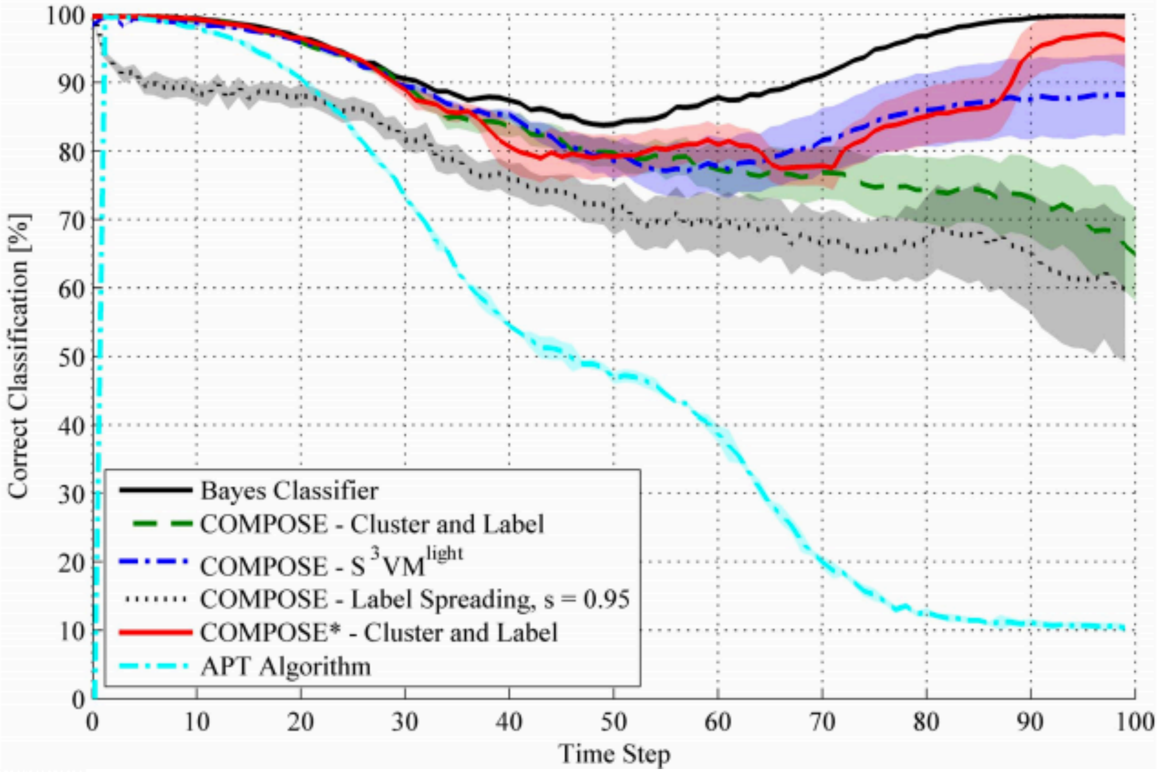
Dyer K., Capo R., Polikar R., "COMPOSE: A Semi-Supervised Learning Framework for Initially Labeled Non-Stationary Streaming Data" IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 1, pp. 12-26, 2014

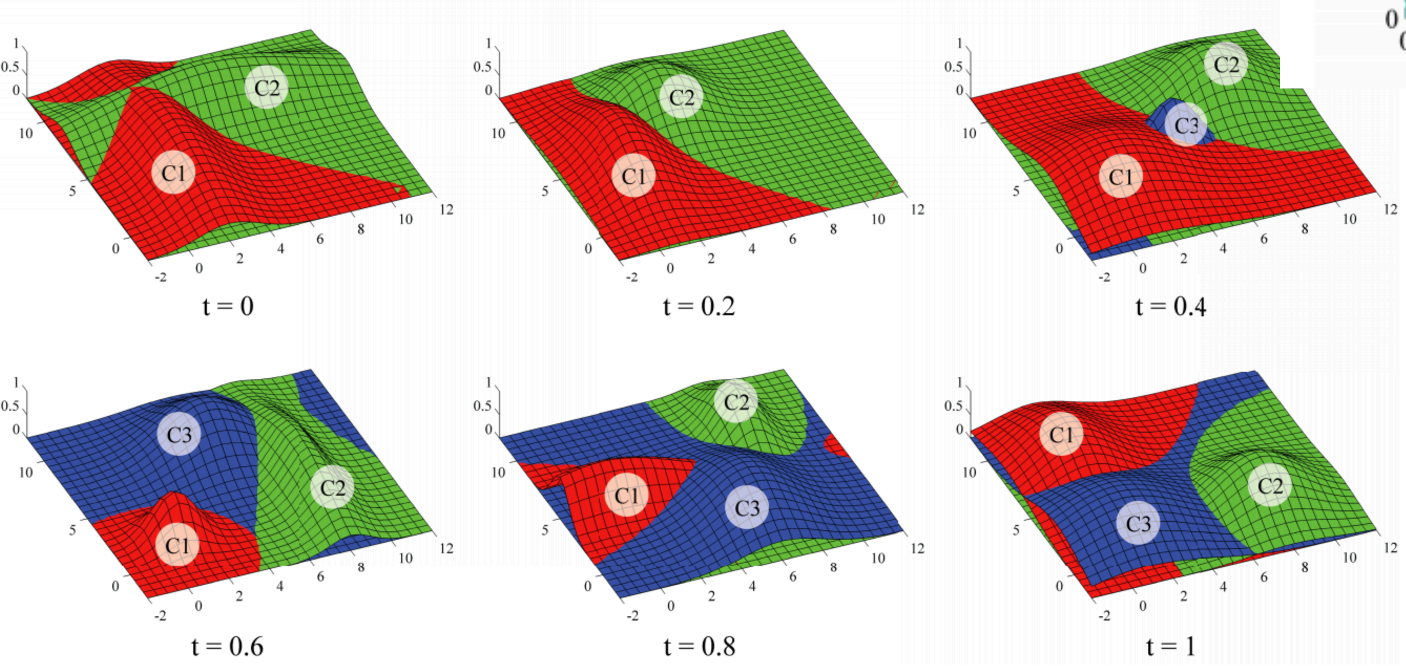
COMPOSE



Sarnelle J., Sanchez A., Capo R., Haas J. and Polikar R., "Quantifying the Limited and Gradual Concept Drift Assumption," Int. Joint Conf. on Neural Networks (IJCNN), 2015.

COMPOSE IN ACTION





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, vol. 25, no. 1, pp. 12-26, 2014 .

COMMENTS FROM MY EXPERIENCE

- Ensemble classifier approaches have had more success that single classifier implementations for nonstationary environments*
- Hybrid approaches (active & passive) can be beneficial! There is no single best strategy
  - Sometimes we lump these approach in the the active category
- In practice, a weighted majority vote is a better strategy as long as we have a reliable estimate of a classifiers error

* That is not to say there is not single classifier solutions that do not work well.

NSL: PASSIVE APPROACHES

Giacomo Boracchi[1] and Gregory Ditzler[2]

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

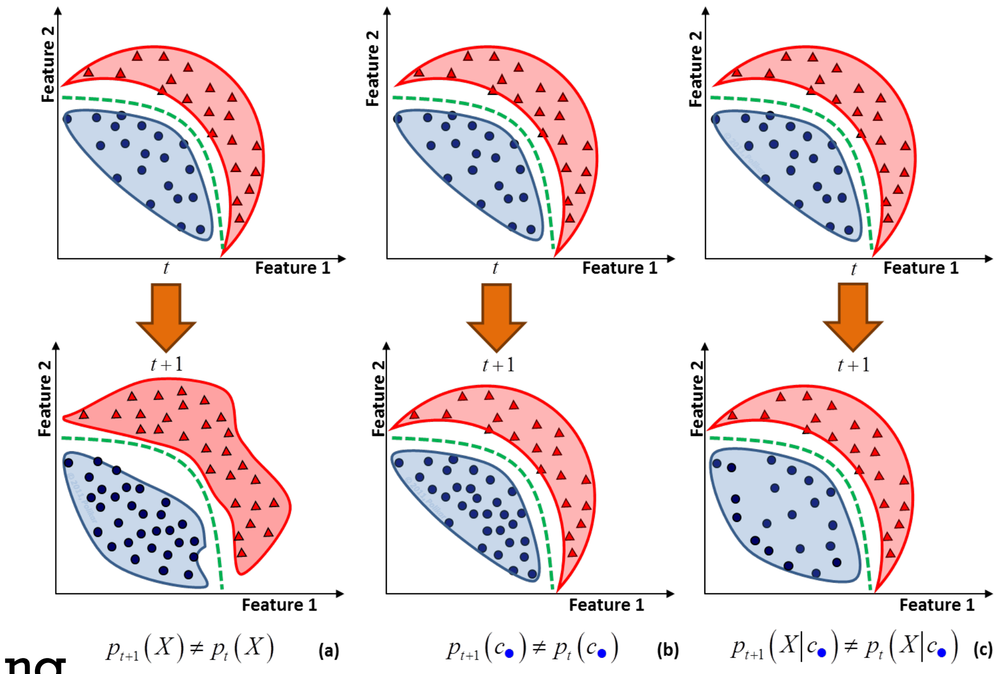
[2] The University of Arizona
Department of Electrical & Computer Engineering
Tucson, AZ USA

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

DATA SETS & GENERATORS

BUILDING A NONSTATIONARY DATA STREAM

- **Recall**: A **concept drift** occurs at time t if
$$\phi_t(\boldsymbol{x}, y) \neq \phi_{t+1}(\boldsymbol{x}, y)$$
(we also say $\mathcal{X}$ becomes **nonstationary**)
- Drift might affect $\phi_t(y|\boldsymbol{x})$ and/or $\phi_t(\boldsymbol{x})$
  - Real and virtual drifts
  - Abrupt, Gradual, Fast



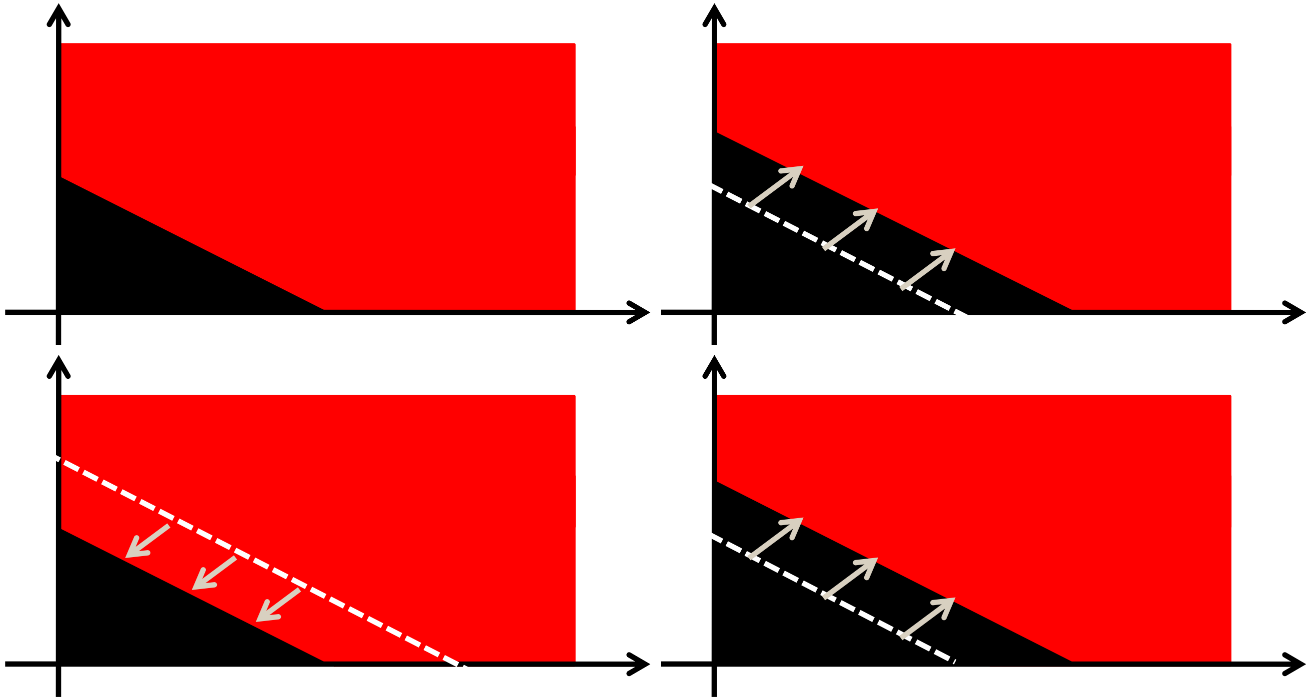
- Synthetic data streams can be generated by sampling data from a distribution that simulates the changes in the probabilities
  - E.g., Data could be sampled from a Gaussian distribution with changing parameters or classes abruptly changed/swap

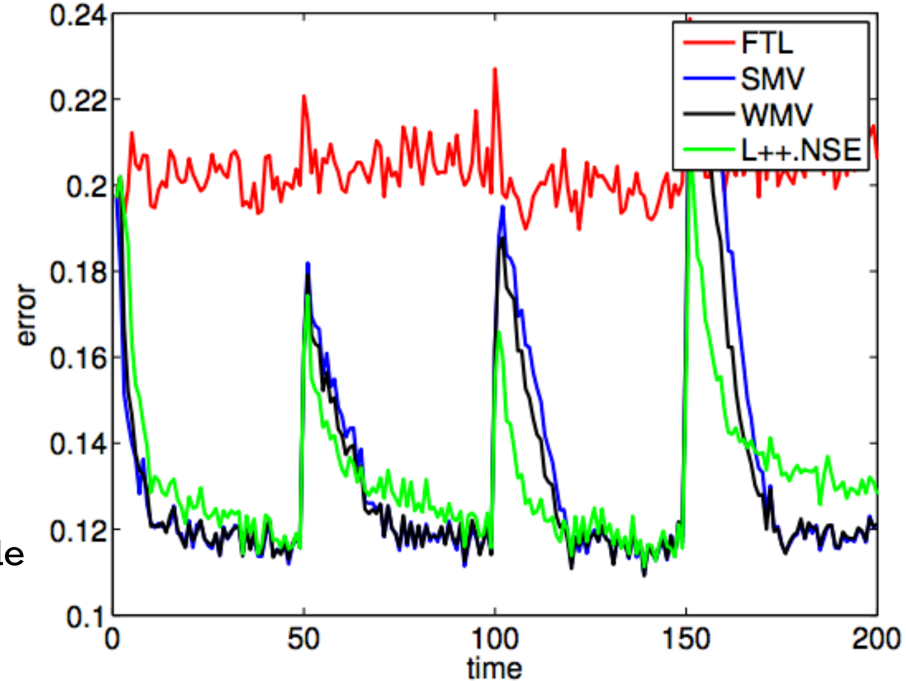
THE VALUE OF SYNTHETIC & REAL DATA

- Concept drift algorithms have been benchmarked against a vast pool of synthetic and real-world data sets, both of which are of great importance to appropriately benchmarking
  - **Synthetic data** allow us to carefully design experiments to evaluate the limitations of an approach
  - **Real world** data serve as the ultimate benchmark about how we should expect an algorithm to perform when it is deployed

SEA DATA STREAM

- Hyperplane changes location at three points in time
  - Three features only two of which are relevant. One feature is noise with 10% noise in the labels
  - Class imbalance changes as the plane shift. Thus, change in $\phi(\boldsymbol{x}|y)$ and $\phi(y)$ changes.
    - Dual change



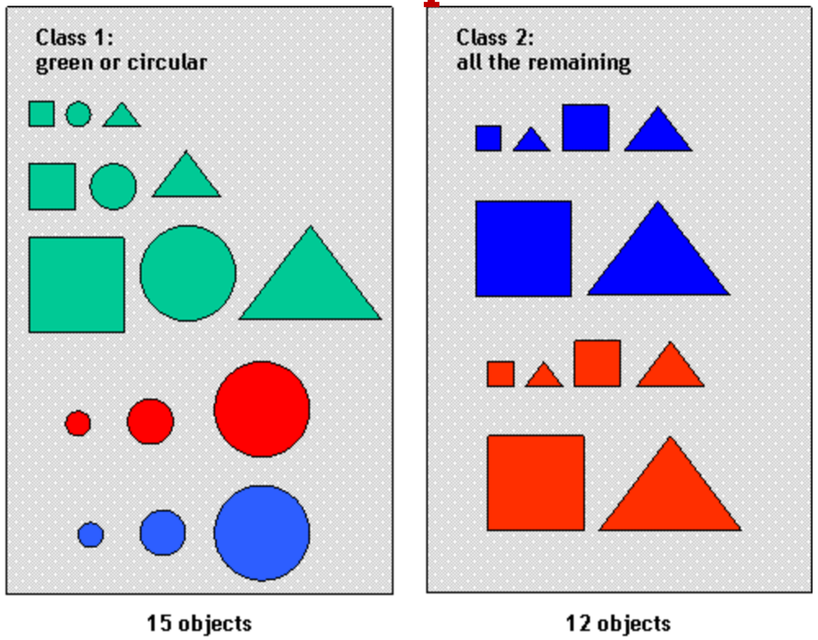


G. Ditzler, G. Rosen, and R. Polikar, "Domain Adaptation Bounds for Multiple Expert Systems Under Concept Drift," International Joint Conference on Neural Networks, 2014. (best student paper)

KUNCHEVA'S DRIFT GENERATOR

- Narasimhamurthy & Kuncheva (2007) developed a Matlab package for simulating nonstationary data streams. Feature include:
  - STAGGER: The feature space is described by three features: size, color and shape. There are three data sources
    - Target Concept 1 : size = small AND color = red
    - Target Concept 2 : color = green OR shape = circular
    - Target Concept 3 : size = medium OR size = large
  - Drifting Hyperplanes: Similar to SEA with a plane
  - Drifting Multi-Modal Gaussian Distributions:
  - Stationary to Nonstationary Data Stream: The above data streams can be sampled from a static distribution; however, to add in nonstationarities, drift can be simulated in the stream by sampling from different concepts

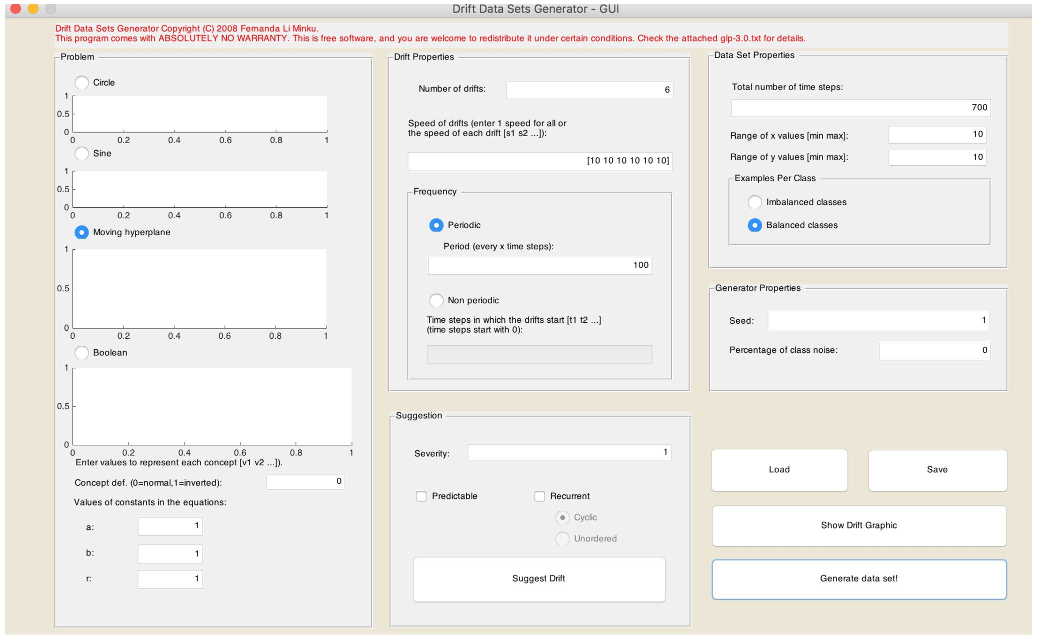
STAGGER: Concept 2



Narasimhamurthy A., L.I. Kuncheva, A framework for generating data to simulate changing environments, *Proc. IASTED, Artificial Intelligence and Applications,* Innsbruck, Austria, 2007, 384-389

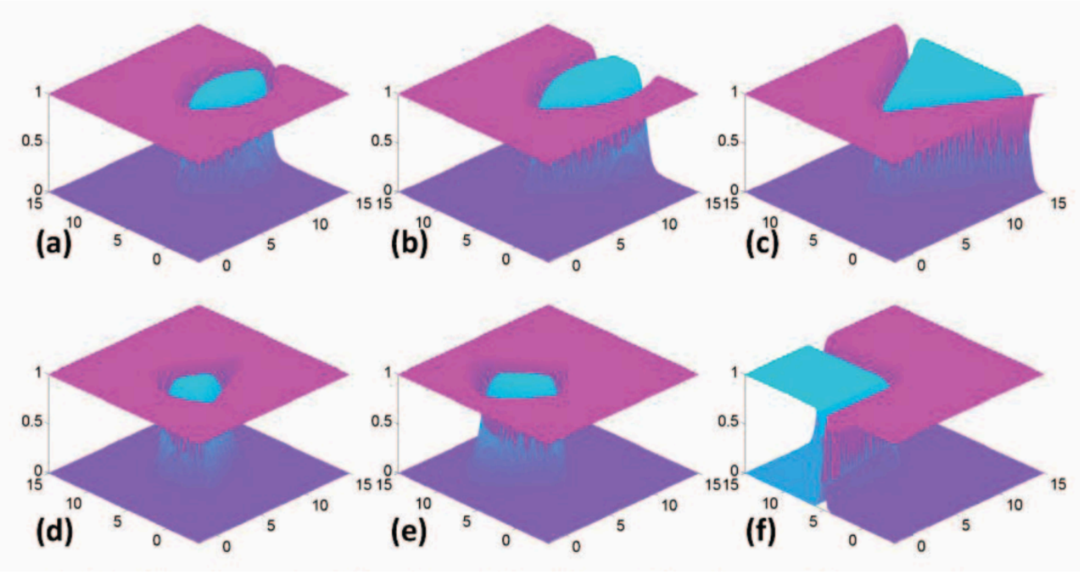
DDD GENERATOR

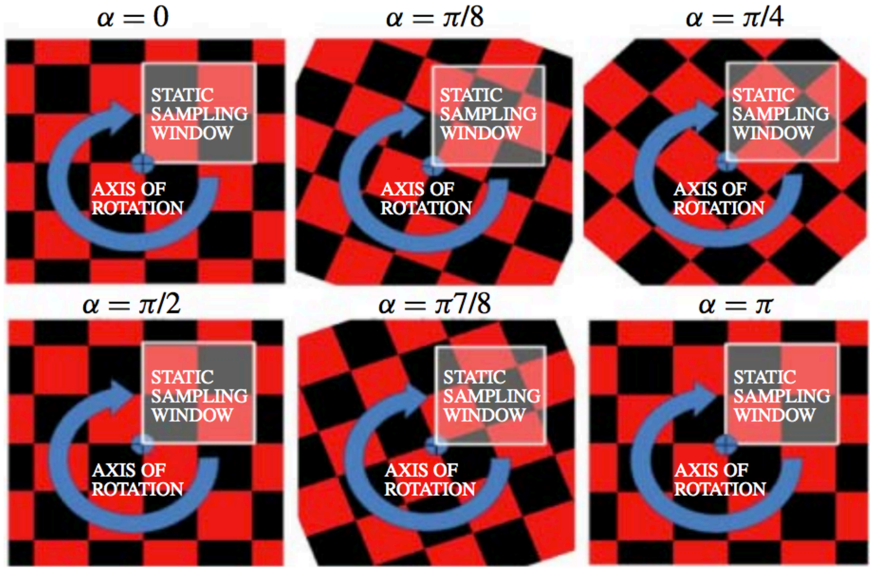
- Minku et al.'s concept drift generator for Matlab
  - **Circle**: Given two variables and a point, do samples fall in or out of a circle with radius r? Let r change to simulate drift
  - **Sine**: $y > a\sin(b\ x + c) + d$? Changle the parameters a, b, c, and d.
  - **Moving Hyperplane**: Similar to SEA with a 1D line
  - **Boolean**: Modification of a STAGGER themed data set
- You can simulate a lot of different data streams with non-stationarities using the GUI

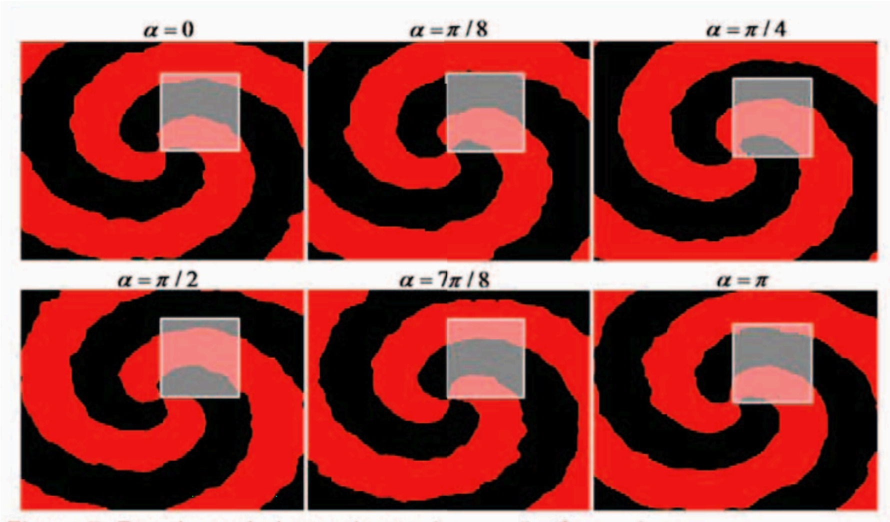


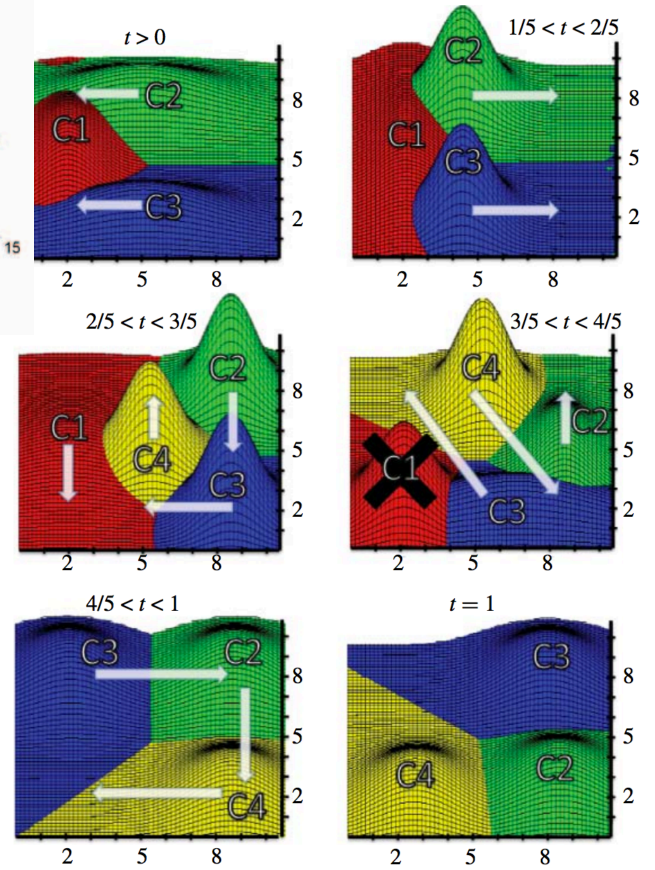
MINKU, L. L.; WHITE, A. P.; YAO, X. . "The Impact of Diversity on On-line Ensemble Learning in the Presence of Concept Drift.", *IEEE Transactions on Knowledge and Data Engineering,* IEEE, v. 22, n. 5, p. 730-742, May 2010

OTHER APPROACHES









REAL-WORLD DATA

REAL-WORLD DATA

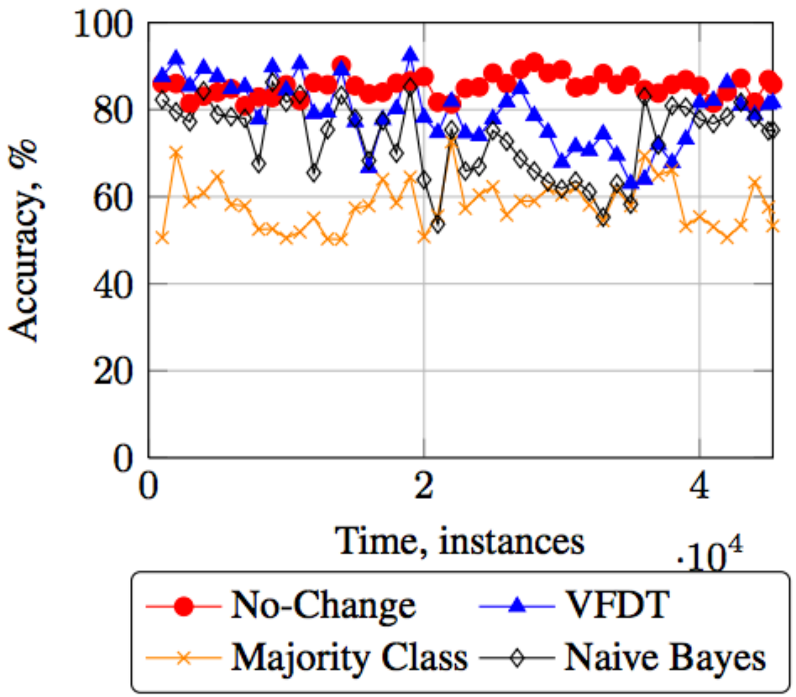
- On **real data** it is sometimes difficult to obtain statistically significant results,
  - How can we quantify delays in change-detection applications on time-dependent data
    - For synthetic data we will know the location of the change; however, this is a bit more ambiguous with real-world data
  - Sometimes time-dependant are data correlated, e.g., the New South Whales electricity data (elec2)
  - Data may not be evolving through a sequence of stationary states
    - This is a problem for active methods
  - Difficult to estimate what could be the performance in real-world because sometimes supervised samples are provided depending on your previous performance
  - Labeled data are not always available to tell us what the current error is for the system to be able to update classifier paremeters (e.g., classifier weights)

REAL-WORLD DATA

- **Airlines Data**: 100M+ instances contain flight arrival and departure records. The goal is to predict if a flight is delayed.
- **Chess.com**: Game records for a player over approximately three years
- elec2:
- **KDD Cup 1999 Data**: Collection of network intrusion detection data.
- **Luxembourg**: Predict a users internet usage European Social Survey data
- **NOAA**: ~27 years of daily weather measurements from Nebraska. The goal is to predict rainfall.
- **POLIMI Rock Collapse/Landslide Forecasting**: Sensor measurements coming from monitoring systems for rock collapse and landslide forecasting deployed on the Italian Alps.
- **Spam**: Collection of spam & ham emails collected over two years

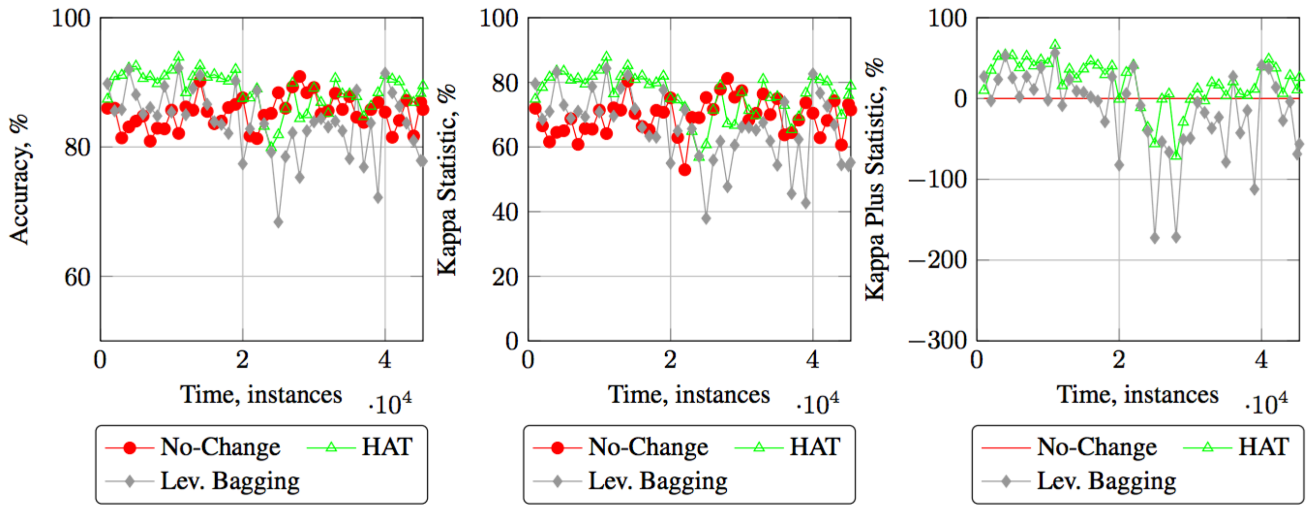
https://github.com/gditzler/ConceptDriftResources

NOTE ON ELEC2



- Data streams could have a temporal component that is not considered in the evaluation of a classifier(s)
  - Assumption is that data are not sampled iid, but still sampled independently
  - This is an issue if the data are auto-correlated
- Bifet et al. pointed out this flaw in the elec2 data set and presented a new statistic for benchmarking such data sets that have temporal dependence

$$\kappa^+ = \frac{n}{n-1} \cdot p_0 - \frac{1}{n-1}$$

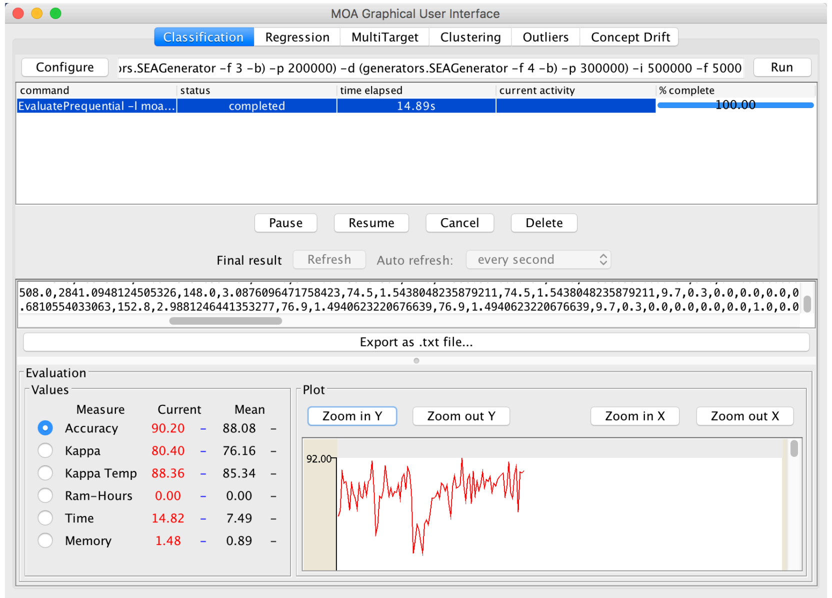


Albert Bifet, Indrė Žliobaitė, Jesse Read, Bernhard Pfahringer and Geoff Holmes: Pitfalls in benchmarking data stream classification and how to avoid them ECML-PKDD 2013

SOFTWARE

MASSIVE ONLINE ANALYSIS

- **Massive Online Analysis (MOA)** is a Java software package for developing and benchmarking data stream algorithms
  - **Active approaches**
    - DDM
    - EDDM
    - ADWIN
  - **Passive approaches**
    - ASHT
    - SGD
    - AUE
    - AUE2



- **Extensions**: Scalable MOA (Apache Storm), and StreamDM (Apache Streaming Spark)

INCREMENTAL LEARNING WITH ENSEMBLES

- Matlab-based toolbox (collection) of scripts that implement several ensemble-based algorithms for learning in nonstationary environments
  - Weighted Majority Ensemble
  - Simple Majority Ensemble
  - Follow the Leader
  - Learn++/Learn++.NSE/Learn++.CDS
  - Example scripts are included

	Data set									
	rain	*elec2*	*german*	*ringworm*	*splice*	*twonorm*	*wave*	*chess*	*lux*	rank
					1–0 loss					
nse	0.218 (3)	0.339 (1)	0.224 (1)	0.232 (2)	0.18 (1)	0.0234 (2)	0.139 (1)	0.329 (3)	0.102 (1)	1.67
avg	0.216 (2)	0.365 (4)	0.259 (4)	0.242 (3)	0.206 (3)	0.0245 (3)	0.152 (3)	0.419 (4)	0.499 (4)	3.33
exp	0.214 (1)	0.354 (3)	0.229 (2)	0.232 (1)	0.184 (2)	0.0231 (1)	0.144 (2)	0.31 (2)	0.485 (3)	1.89
ftl	0.294 (4)	0.341 (2)	0.243 (3)	0.249 (4)	0.271 (4)	0.0387 (4)	0.162 (4)	0.288 (1)	0.13 (2)	3.11
					logistic loss					
nse	0.781 (3)	0.897 (2.5)	0.69 (1.5)	0.747 (1)	0.756 (1.5)	0.489 (1)	0.637 (1)	0.879 (3)	0.992 (2)	1.83
avg	0.781 (2)	0.897 (2.5)	0.69 (1.5)	0.747 (2)	0.756 (1.5)	0.491 (3)	0.637 (2)	0.879 (4)	1.03 (4)	2.50
exp	0.779 (1)	0.892 (1)	0.691 (3)	0.748 (3)	0.756 (3)	0.491 (2)	0.637 (3)	0.873 (2)	0.998 (3)	2.33
ftl	0.87 (4)	0.938 (4)	0.738 (4)	0.795 (4)	0.824 (4)	0.502 (4)	0.665 (4)	0.83 (1)	0.627 (1)	3.33
					mse loss					
nse	0.652 (3)	0.896 (3)	0.716 (1.5)	0.705 (1.5)	0.569 (1.5)	0.0759 (1)	0.454 (3)	0.991 (3)	0.377 (1)	2.06
avg	0.625 (1)	0.892 (2)	0.716 (1.5)	0.705 (1.5)	0.569 (1.5)	0.0768 (3)	0.41 (2)	0.933 (2)	1.47 (4)	2.06
exp	0.63 (2)	0.872 (1)	0.719 (3)	0.707 (3)	0.572 (3)	0.0766 (2)	0.409 (1)	0.918 (1)	1.1 (3)	2.11
ftl	1.18 (4)	1.37 (4)	0.971 (4)	0.997 (4)	1.09 (4)	0.155 (4)	0.646 (4)	1.15 (4)	0.52 (2)	3.78

G. Ditzler, G. Rosen and R. Polikar, "Discounted expert weighting for concept drift," International Symposium on Computational Intelligence in Dynamic and Uncertain Environments, 2013.

COMPARING MULTIPLE CLASSIFIERS

- Comparing multiple classifiers on multiple datasets is not a trivial problem
  - Confidence intervals will only allow for the comparison of multiple classifiers on a single dataset
- The rank based Friedman test can determine if classifiers are performing equally across multiple dataset
  - Apply ranks to the average of each measure on a dataset
  - Standard deviation of the measure is not used in the Friedman test

$$\chi_F^2 = \frac{12N}{k(k+1)}\left(\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4}\right)$$

$$F_f = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}$$

- z-scores can be computed from the ranks in the Friedman test
  - The α-level or critical value must be adjusted based on the multiple comparisons being made
  - Bonferroni-Dunn procedure adjusts α to $\alpha\,/(k-1)$

$$z_m(i,j) = \frac{R_m(i) - R_m(j)}{\sqrt{\frac{k(k+1)}{6N}}}$$

J. Demsar, "Statistical Comparisons of Classifiers over Multiple Data Sets," J. Machine Learning Research, vol. 7, pp. 1-30, 2006.

CHALLENGES AND PERSPECTIVES

CHALLENGES

- Learning in nonstaionary environments is becoming a more mature field; however, there are many sub-problems in the field that still need to be addressed more rigorously
  - **Unbalanced environment**: Data from each of the classes are extremely imbalanced, which is a serious problem if the algorithm is using error to track the environment.
  - **Semisupervised**: How can we best incorporate data that are unlabeled into our model if we cannot assume the data are sampled iid?
    - Consensus Maximization: Train supervised and unsupervised models
    - Semi-supervised vs. Transductive?
  - **Latency verification**: What if we cannot assume that the classifier will receive immediate feedback?
    - A study of extreme latency verification and how to perform benchmarks
  - **Error estimation**: How can error be accurately estimated in the precense of non-stationary data streams when an concept is abruptly re-introduced

CHALLENGES

- A **theoretical framework** is lacking that incorporates drift information in the labeled and unlabeled data
  - Less heuristics more statistics!
- Integration of expert-driven knowledge with data-driven knowledge