Online Machine Learning

Concept Drift Detection

Antonio Carta

antonio.carta@unipi.it

Plan for Today's Lecture

- What is concept drift
- How to model concept drift
- Types of concept drift
- Concept drift estimation
- Concept drift detection

Concept Drift Example – COVID pandemic

- ML models are trained to predict the «normal» behavior
- What happens when the «normal» behavior changes?


Our weird behavior during the pandemic is messing with AI models

Machine-learning models trained on normal behavior are showing cracks —forcing humans to step in to set them straight.

By Will Douglas Heaven

May 11, 2020

Heaven, W. D. (2020). Our weird behavior during the pandemic is messing with AI models. MIT Technology Review)

What is Concept Drift (CD)?

Wha it is:

- A change in the real world
- Affects the input/output distribution
- Disrupt the model's predictions

What it's not:

- It's not noise
- It's not outliers

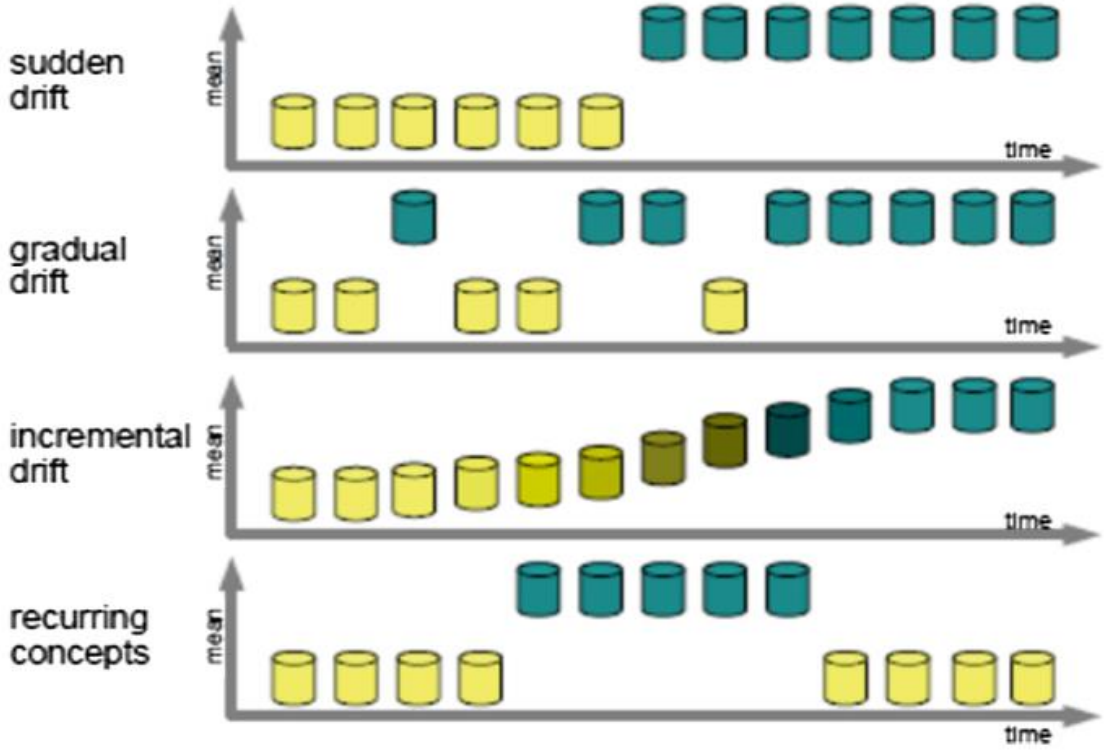


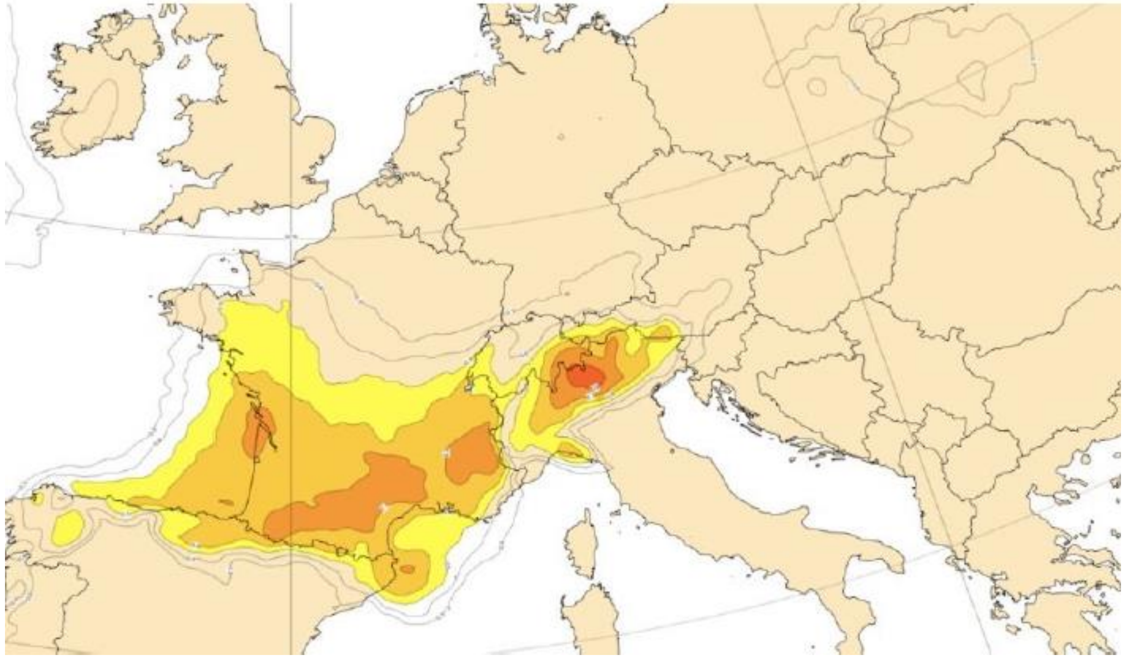
Fig. 3. Types of concept drift [99]

Figure: V. Lemaire et al. «A Survey on Supervised Classification on Data Streams"

Example – Weather Forecasts

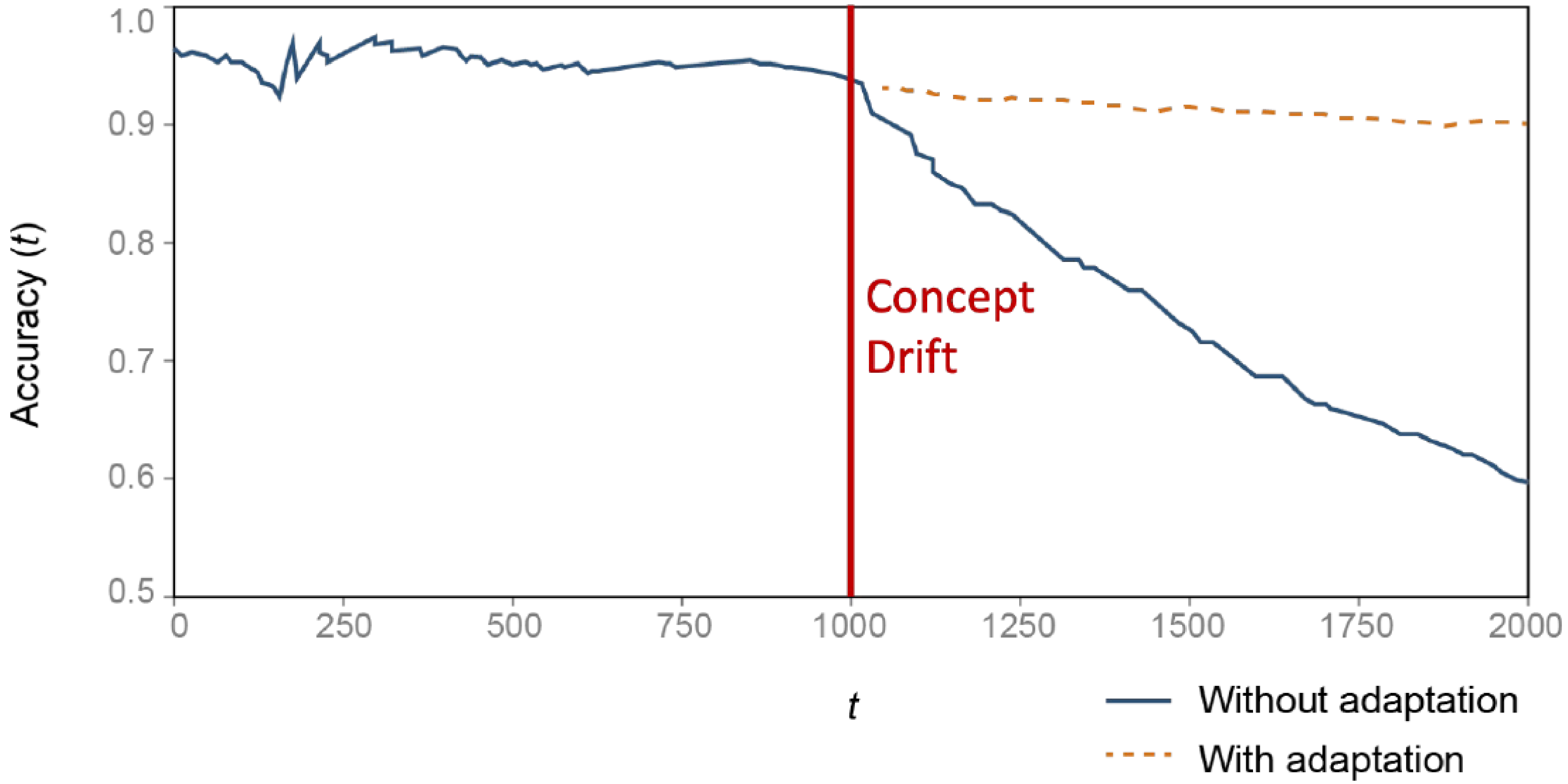
Weather forecast

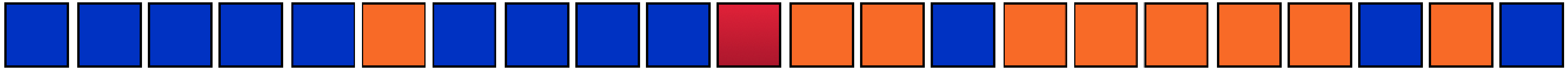
- chaotic nature of the atmosphere
- continuous and sudden weather changes (concept drifts)
- Weather forecast models must *detect these changes and adapt to them*, without be retrained from scratch



https://www.meteosvizzera.admin.ch/home/attualita/meteosvizzera-blog.subpage.html/it/data/blogs/2021/8/efi---extreme-forecast-index.html

Why do we care about CD?





CD Nomenclature

- **Sudden** change distribution has remained unchanged for a long time, then changes in a few steps to a significantly different one. It is often called *shift*.
- **Gradual or incremental** change occurs when, for a long time, the distribution experiences at each time step a tiny, barely noticeable change, but these accumulated changes become significant over time.
- Change may be **global** or **partial** depending on whether it affects all of the item space or just a part of it. In ML terminology, partial change might affect only instances of certain forms, or only some of the instance attributes.
- **Recurrent concepts** occur when distributions that have appeared in the past tend to reappear later. An example is seasonality, where summer distributions are similar among themselves and different from winter distributions. A different example is the distortions in city traffic and public transportation due to mass events or accidents, which happen at irregular, unpredictable times.

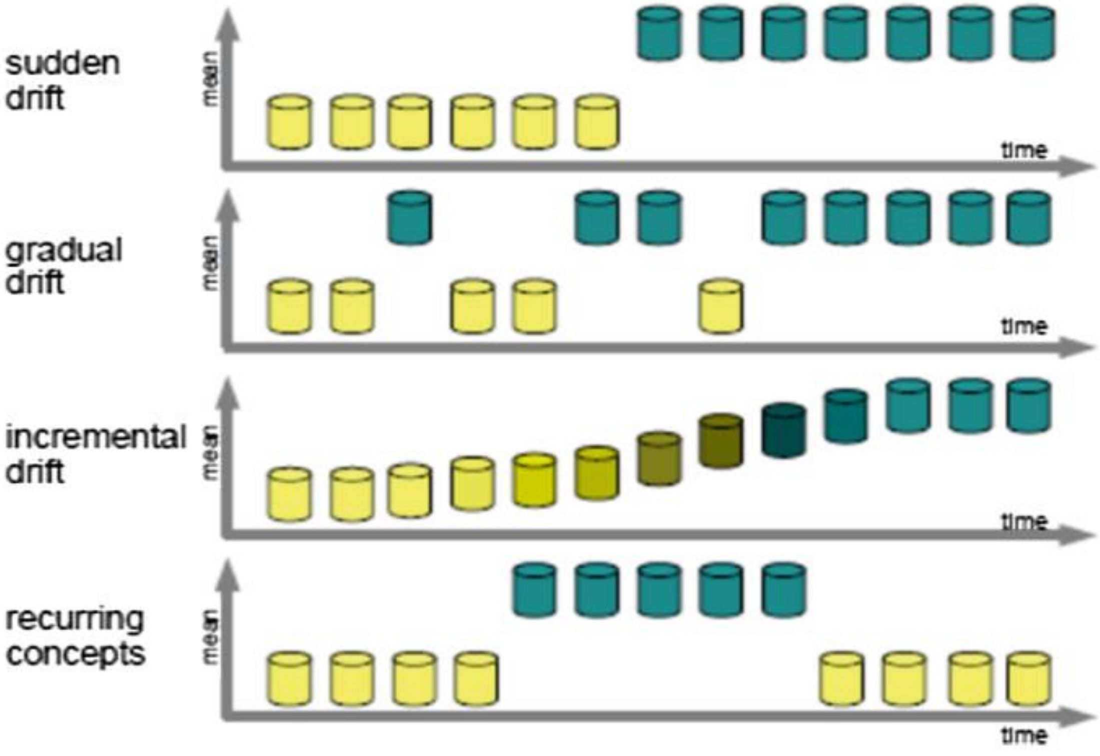
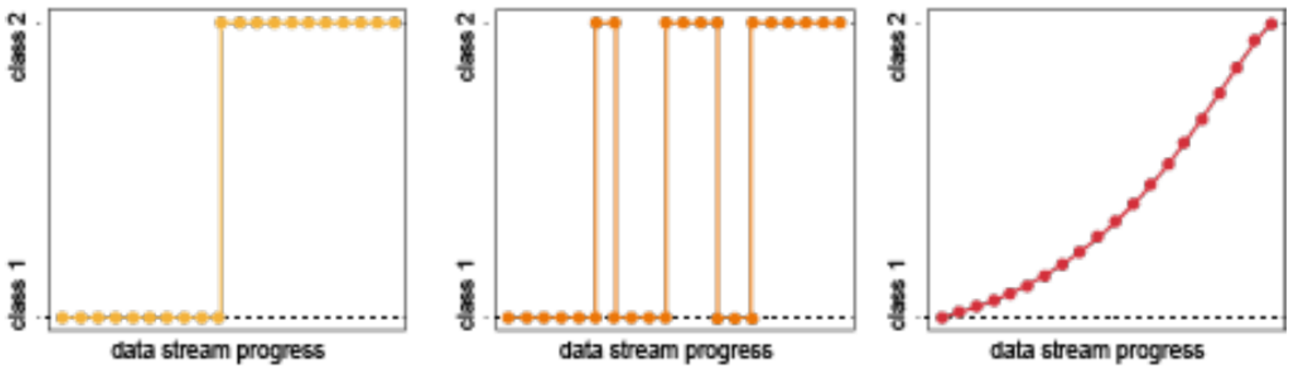


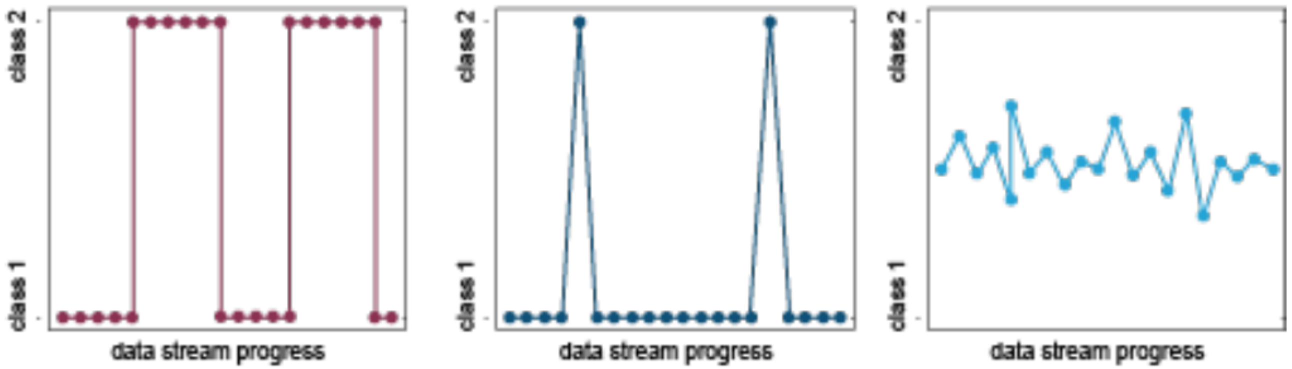
Fig. 3. Types of concept drift [99]

Def: MOA Book, figure: Figure: V. Lemaire et al. «A Survey on Supervised Classification on Data Streams"

Examples



(a) Sudden. (b) Gradual. (c) Incremental.



(d) Recurring. (e) Blips. (f) Noise.

Concept Drift vs Anomaly Detection

- **Concept Drift question:** ”Is yesterday’s model capable of explaining today’s data?”
- **Anomaly detection question:** ”Do these samples conform to the normal ones?”

- A Concept Drift is a change in the distribution that requires changing the model, while an anomaly is an example different the underlying distribution (outlier).

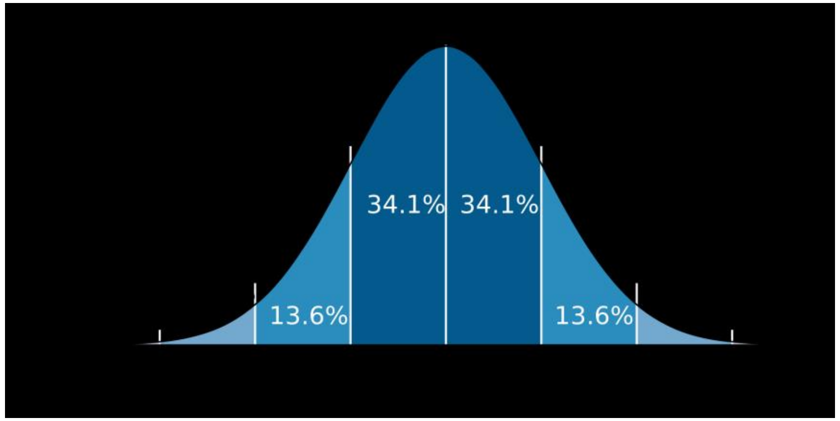
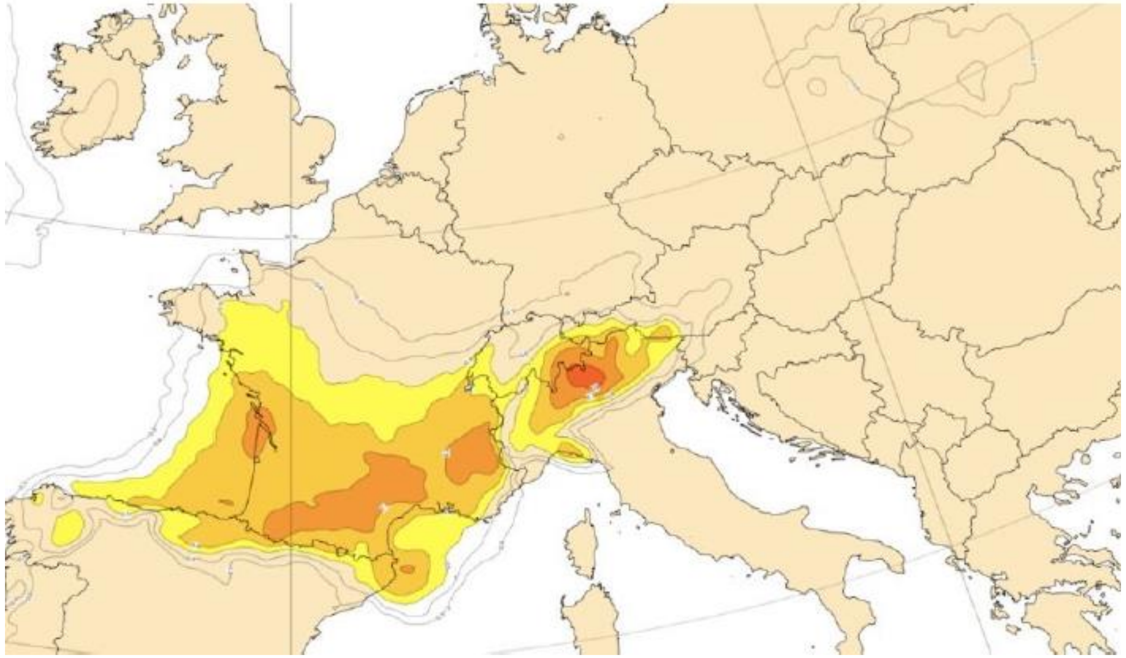


Image: wikimedia

Example – Weather Forecasts

Weather forecast

- Sudden?
- Gradual or incremental?
- global or partial?
- Recurrent concepts?



https://www.meteosvizzera.admin.ch/home/attualita/meteosvizzera-blog.subpage.html/it/data/blogs/2021/8/efi---extreme-forecast-index.html

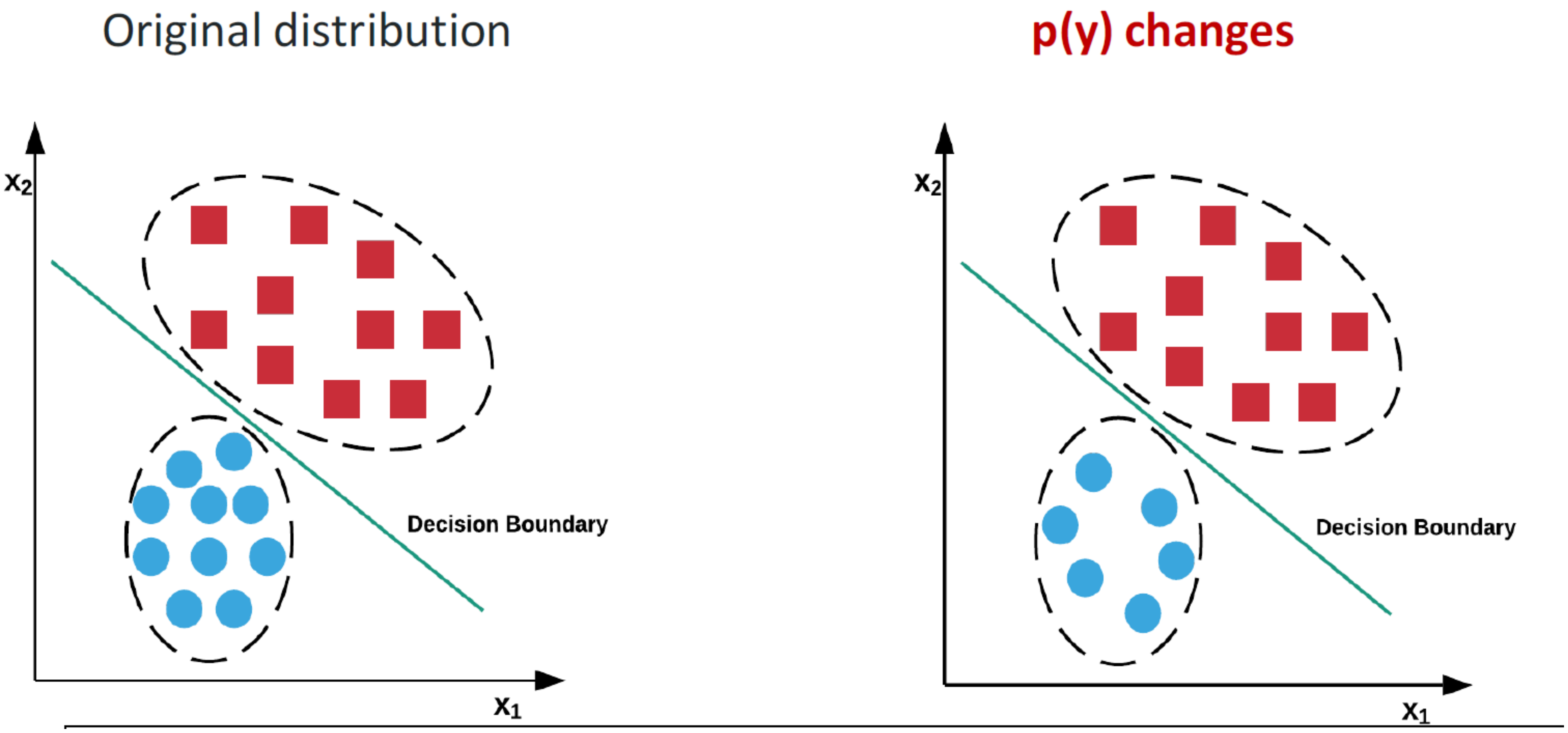
CD – A Probabilistic Definition

- Given an input $x_1, x_2, \dots, x_t$ of class y we can apply bayes theorem:

$$p(y|x_t) = \frac{p(y)p(x_t|y)}{p(x_t)}$$

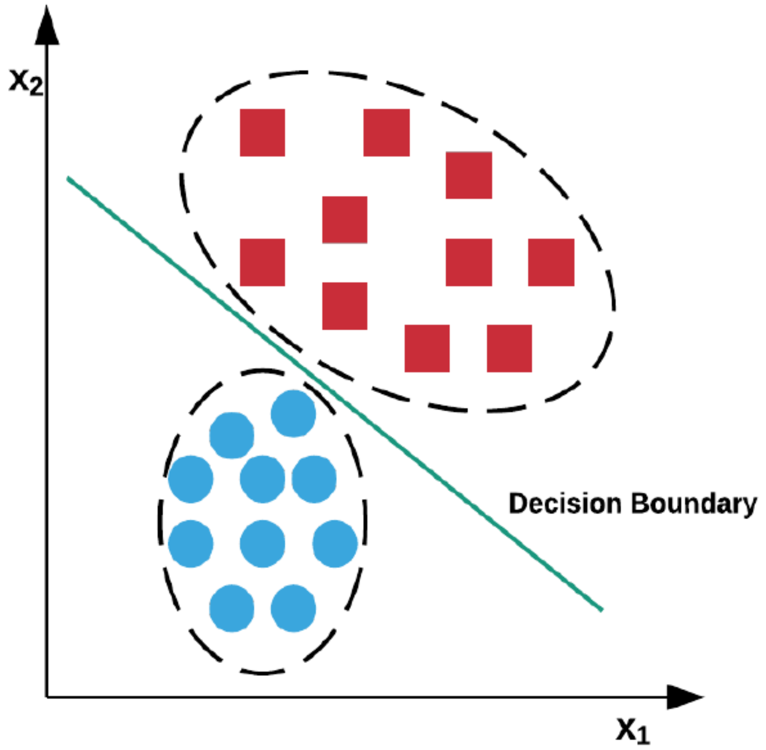
- $p(y)$ is the prior for the output class (concept)
- $p(x_t|y)$ the conditional probability
- Why do we care?
  - Different causes for changes in each term
  - Different consequences (do we need to retrain our model?)

P(y) changes

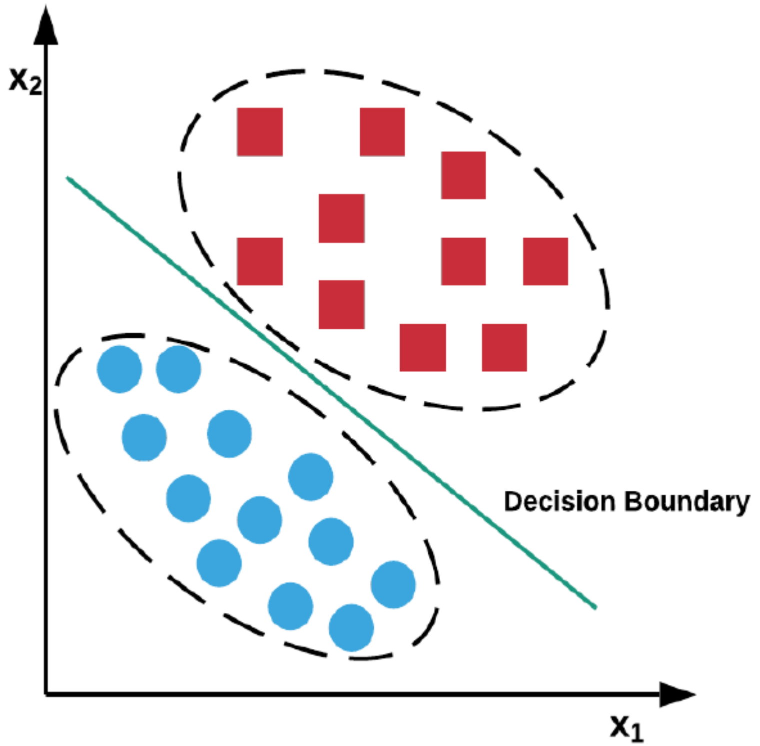

Original distribution
p(y) changes
x2
x1
Decision Boundary
x2
x1
Decision Boundary

P(x_t | y) changes

Original distribution



p(Xt | y) changes

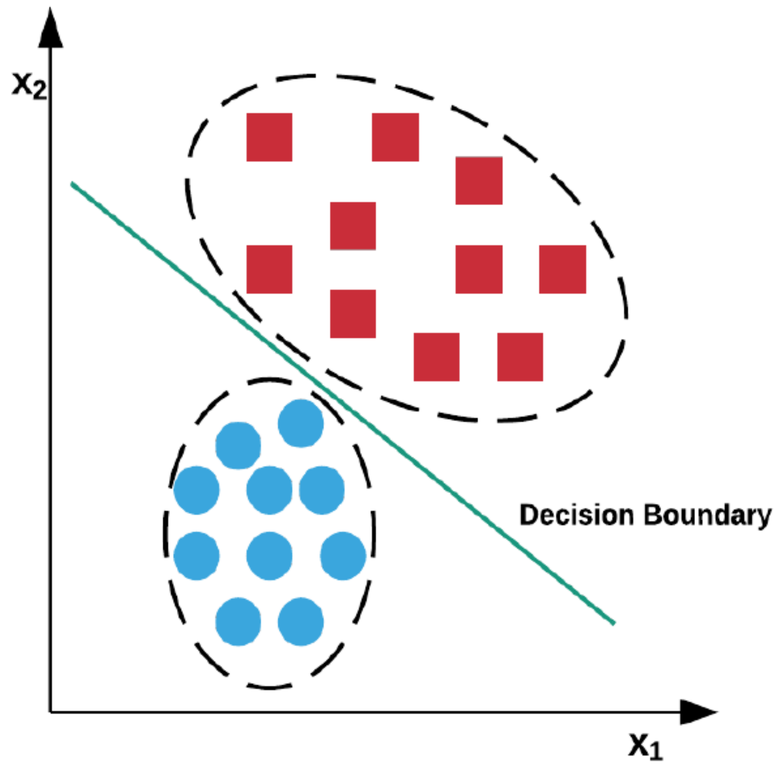


P(y|x_t) changes

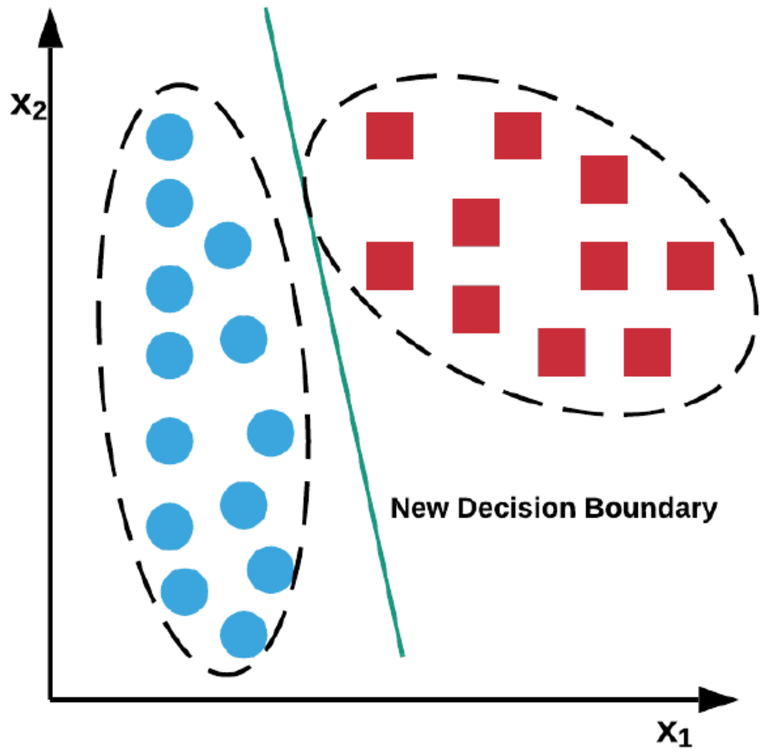
Notice that there are two decision boundaries:

- **the «True» decision boundary, i.e. the optimal solution for the current distribution**
- **The «model» decision boundary, i.e. the currently learned model**

Original distribution

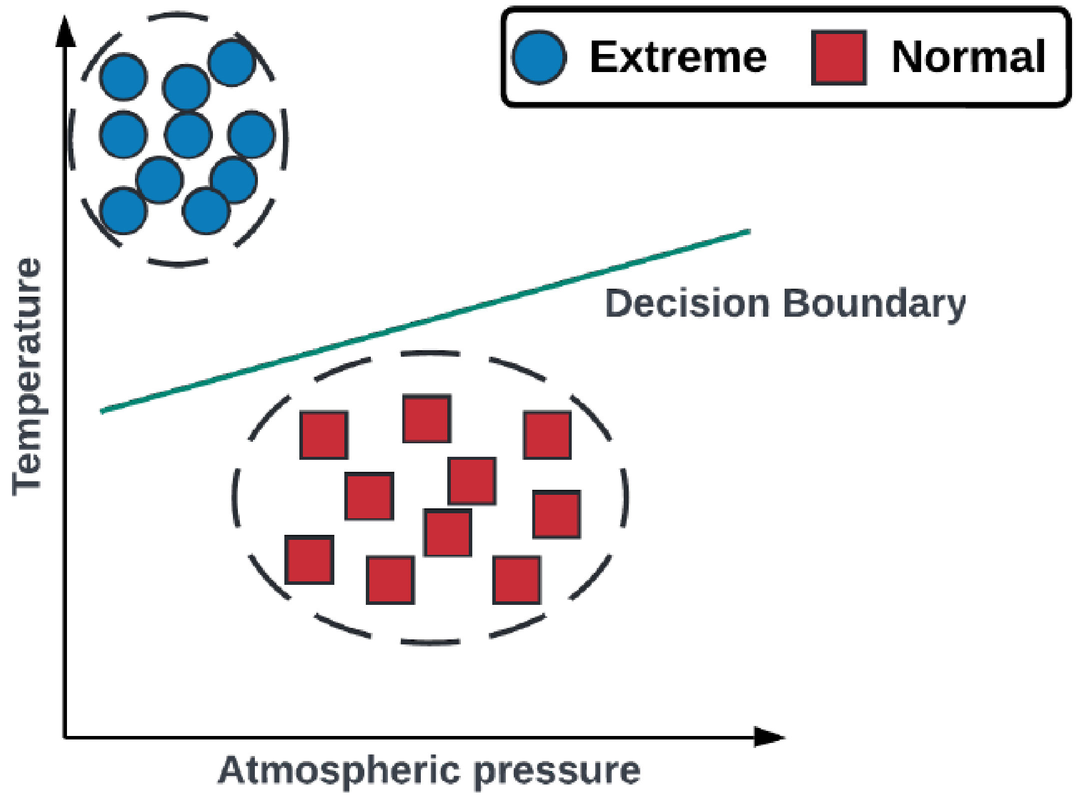


$p(y|X_t)$ changes



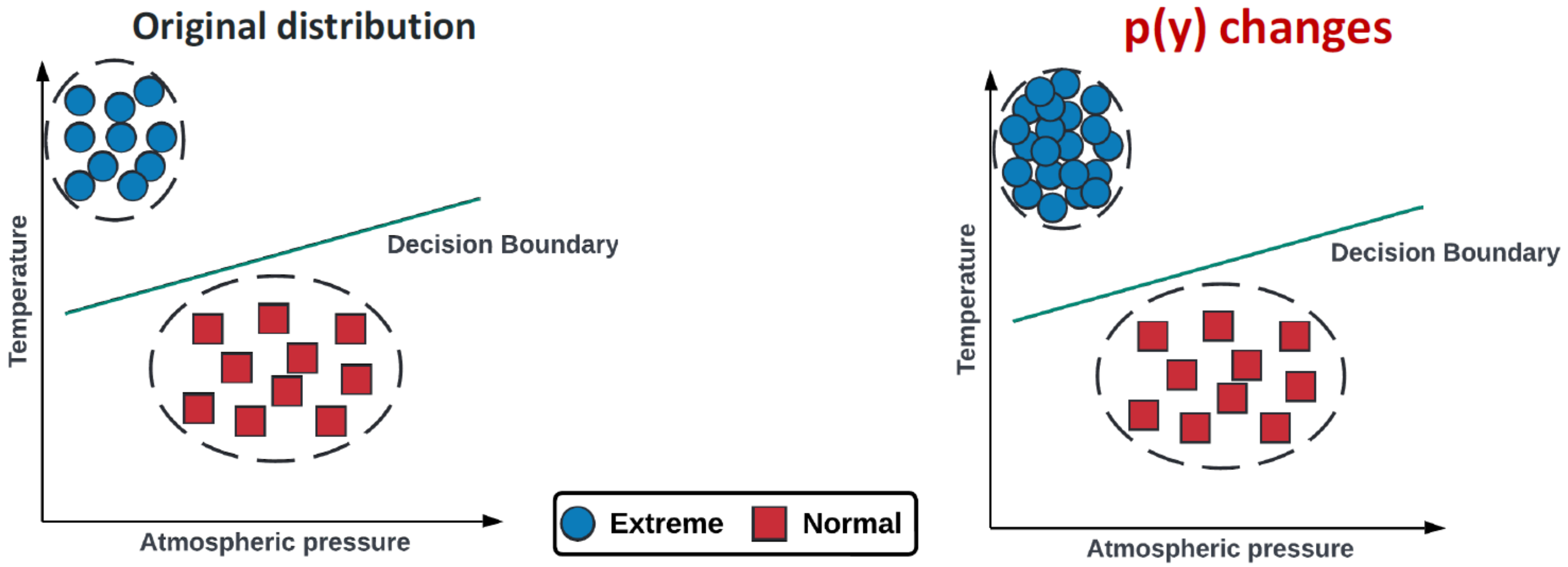
Example: Weather and Climate Change

Example: consider the case of predicting extreme weather phenomena occurrences based on atmospheric pressure and temperature. Usually, extreme weather phenomena occur in the case of low atmospheric pressure and high temperature.



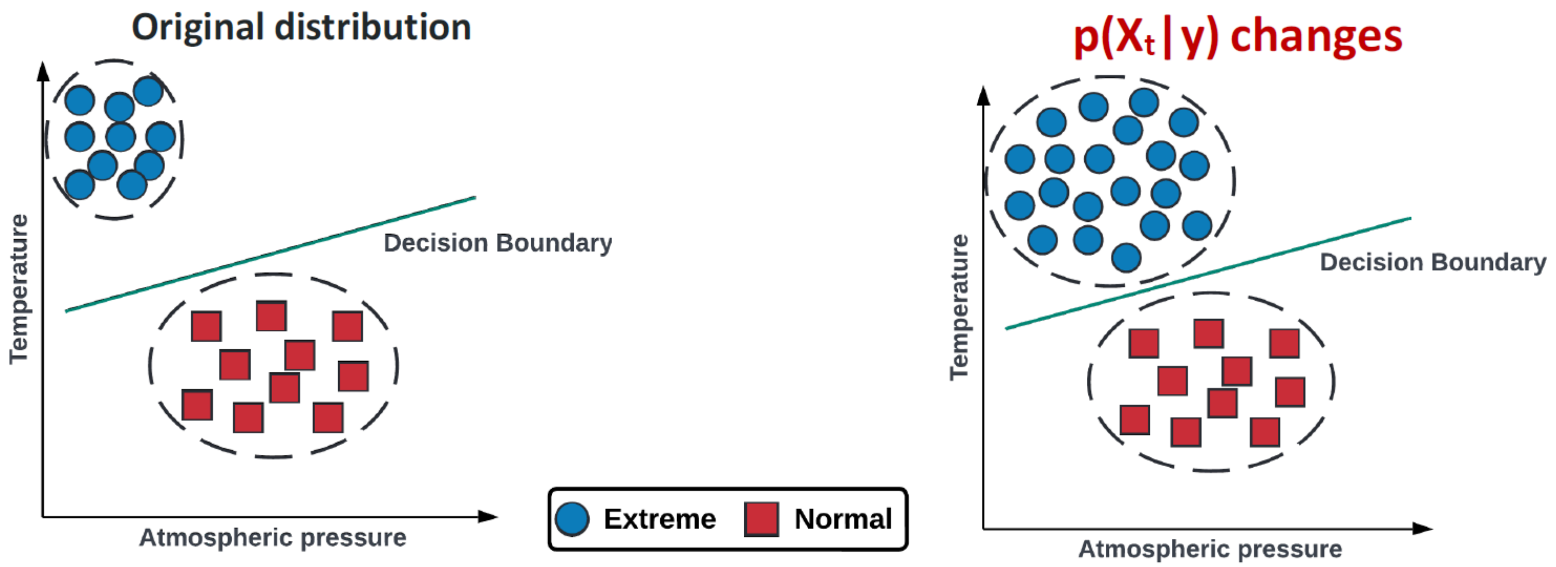
Climate Change: p(y) changes

p(y) concept drift: in the XX century, the distribution of atmospheric pressure and temperature did not change, but the extreme weather phenomena were more frequent.



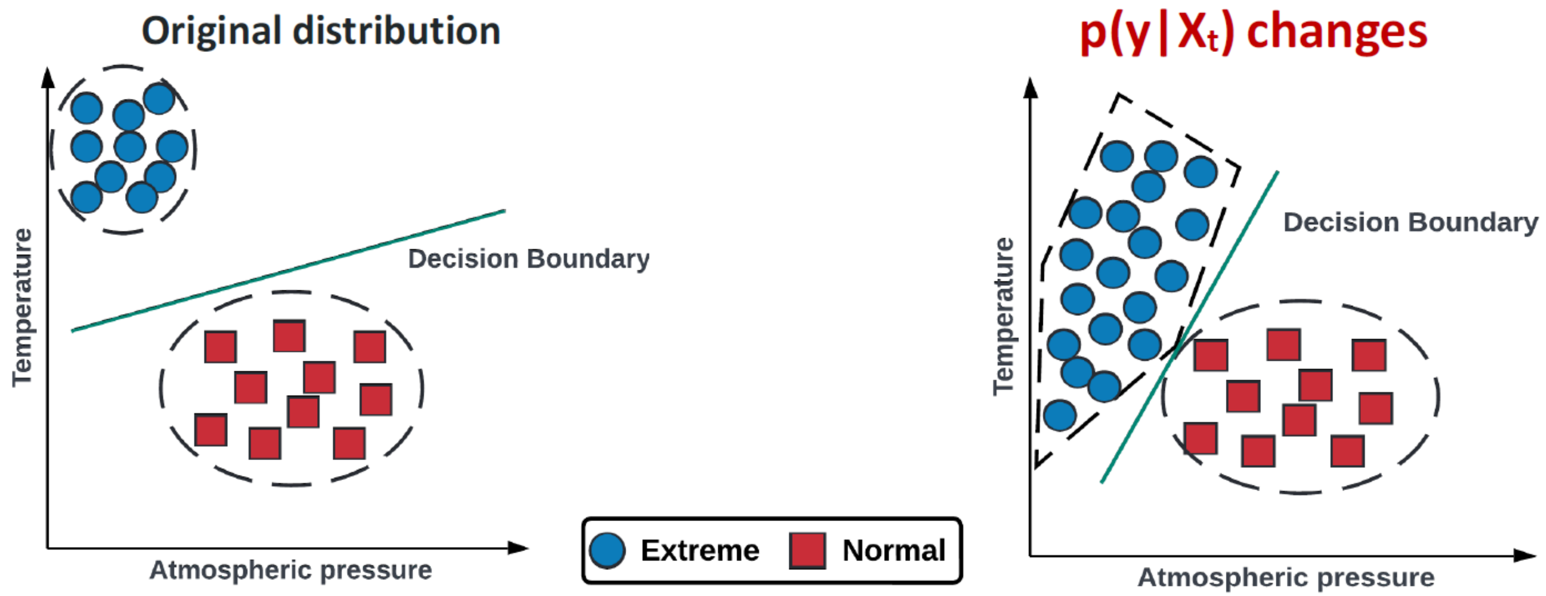
P(x|y) changes

p(X|y) concept drift: in the first two decades of XXI century, the atmospheric pressure and air temperature conditions, in which these phenomena occur, also started to change, but not so drastically to move the decision boundary we use for predicting them.



P(y|x) changes

p(y|X) concept drift: due to the on going climate change, these phenomena start occurring more frequently with higher atmospheric pressure and lower temperature. As a consequence, we have to update the decision boundary to keep an high predictive performance.



Dataset Shift Nomenclature

NOTE: We do a small detour from our sequential world into the world of offline learning with separate train and test data (and distributions)

- **Notation**:
  - x covariates/input features
  - y class/target variable
  - p(y, x) joint distribution
  - sometimes the x→ y relationship is referred with the generic term "concept
- The nomenclature is based on **causal assumptions**:
  - x→y problems: class label is causally determined by input. Example: credit card fraud detection
  - y→x problems: class label determines input. Example: medical diagnosis

Moreno-Torres, Jose G., Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. "A Unifying View on Dataset Shift in Classification." Pattern Recognition 45, no. 1 (January 2012): 521–30. https://doi.org/10.1016/j.patcog.2011.06.019.

Dataset Shift Nomenclature

KEEP IN MIND: we are talking about train/test distributions here, but in OML we will have past/present subsequences

Dataset Shift: $p_{tra}(x, y) \neq p_{tst}(x, y)$

- Informally: any change in the distribution is a shift

Covariate shift: happen in X→Y problems when

- $p_{tra}(y|x) = p_{tst}(y|x)$ and $p_{tra}(x) \neq p_{tst}(x)$
- informally: the input distribution changes, the input->output relationship does not

Prior probability shift: happen in Y→X problems when

- $p_{tra}(x|y) = p_{tst}(x|y)$ and $p_{tra}(y) \neq p_{tst}(y)$
- Informally: output->input relationship is the same but the probability of each class is changed

Concept shift:

- $p_{tra}(y|x) \neq p_{tst}(y|x)$ and $p_{tra}(x) = p_{tst}(x)$ in X→Y problems.
- $p_{tra}(x|y) \neq p_{tst}(x|y)$ and $p_{tra}(y) = p_{tst}(y)$ in Y→X problems.
- Informally: the «concept» (i.e. the class)

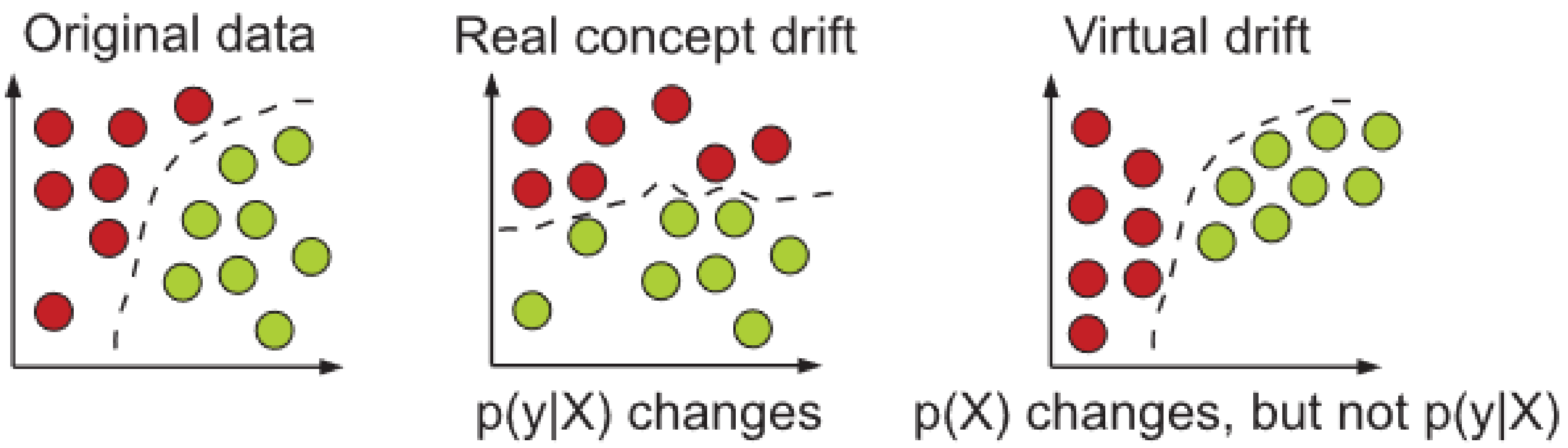
Moreno-Torres, Jose G., Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. "A Unifying View on Dataset Shift in Classification." Pattern Recognition 45, no. 1 (January 2012): 521–30. https://doi.org/10.1016/j.patcog.2011.06.019.

Causes of Shifts

- **Sampling bias**:
  - The world is fixed but we only see a part of it
  - The «visible part» changes over time, causing a shift
  - We will also call it virtual drift
  - Examples: bias in polls, limited observability of environments, change of domain…
- **Non-stationary environments**:
  - The world is continuously changing
  - We will also call it real drift
  - Examples: weather, financial markets, …

> We will see different forms of dataset shifts in all the modules of this course. It’s important to be aware of the types of shifts and its underlying causes for each setting.

Real vs Virtual Drift



Gama, João, et al. A survey on concept drift adaptation. ACM computing surveys (CSUR) 46.4 (2014): 1-37.

Offline ML Objective

- Loss without drift (same as in the offline setting):

Definition 11. Static Risk. If we assume virtual drift, then there exists a true data distribution $p(\mathbf{x}, y)$, which is static even though the data stream distribution is changing. We call $p(\mathbf{x}, y)$ the *static distribution*. Given a model h and a loss function $\mathcal{L}$, we can compute its *Static Risk* as the risk $R^S(h) = E_{p(\mathbf{x},y)}[\mathcal{L}(h, (\mathbf{x}), y)] = \int \mathcal{L}(h(\mathbf{x}), y) dp(\mathbf{x}, y)$. Again, notice that the static distribution $p(\mathbf{x}, y)$ is unknown, which means that this quantity will always be estimated empirically. This evaluation measure focuses on the whole task that the model is trying to solve. However, it must be pointed out that the actual data is going to come from another distribution $p_t(\mathbf{x}, y \mid d)$ conditioned on some domain d.

OML Objective with Drift and Prequential Eval

- Loss in the online «prequential» setting with drifts:

Definition 13. Next-Step Risk. The two previous measures assume virtual drift and evaluate the model on the underlying true distribution. However, the actual distribution at the next step $p_{t+1}(\mathbf{x}, y)$ can be very different from the true distribution $p(\mathbf{x}, y)$. Furthermore, in a real drift setting there is no true distribution. Given a model h_t at step t and a loss function $\mathcal{L}$, the *Next-Step Risk* is $R_t^N(h_t) = E_{p_{t+1}(\mathbf{x},y)}[\mathcal{L}(h_t, (\mathbf{x}), y)] = \int \mathcal{L}(h_t(\mathbf{x}), y) dp_{t+1}(\mathbf{x}, y)$. Similar to the previous cases, we have introduced the next-step distribution $p_{t+1}(\mathbf{x}, y)$. The next-step risk evaluates the performance of the current model h_t on the next-step $(t+1)$ distribution.

CD Detection and Estimation

CD Algorithms

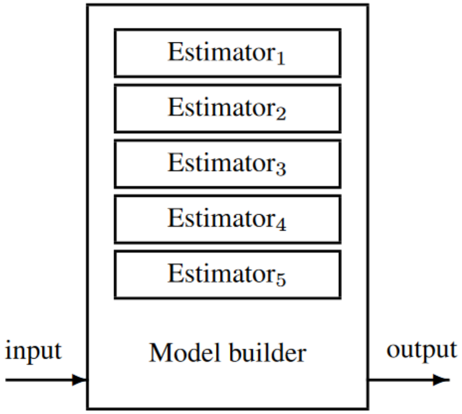
- **Requirements**:
  - fast detection of change
  - robustness to noise and outliers
  - low computational overhead
- **Families**:
  - *estimators-based*: track stream statistics → algorithms update model's statistics
  - *detector-based*: detect CD → update model
  - *ensembling*: dynamic population of models
- **Limitations**: all of these methods are for single-dimensional / low-dimensional data

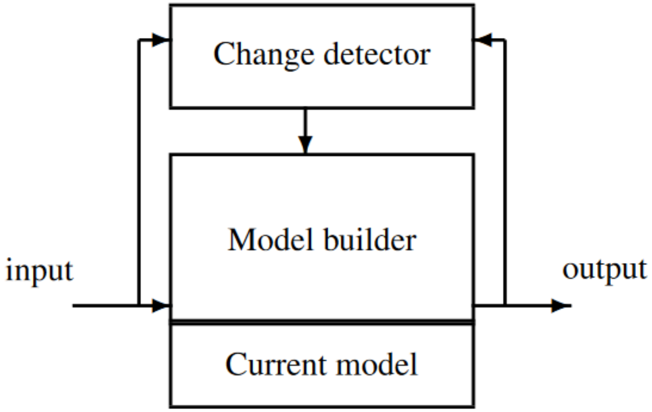
Families

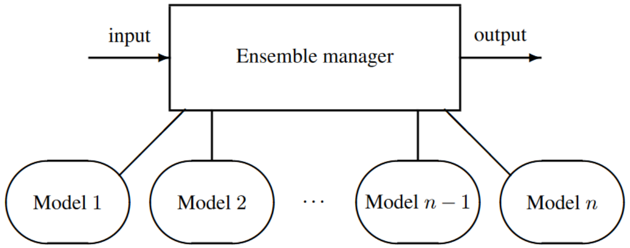
estimators-based:
- track stream statistics
- algorithms update model's statistics

detector-based:
- detect CD
- update model

ensembling:
- dynamic population of models
- Policy to add new model
- Policy to select model







Figures: *MOA book*

CD Estimation

- **compute statistics needed by the model**
  - These will be some expected value $\theta_t = E_{p_t(x)}[f(x)]$ (possibly drifting)
  - We want to update the estimate of θ_t
- **How do we find outdated element and discard them from the computation?**
  - store memory of samples (e.g. a window/buffer):
    - linear estimator over sliding window
  - memoryless estimators:
    - EWMA - exponentially weighted moving average
    - Kalman Filter
    - Autoregressive Models (AR, ARMA)

Window-based: Linear Estimator Over a Sliding Window

- IDEA: let's use only a recent window to estimate θ_t

$$\theta_t = E_{p_t(x)}[f(x)] \approx \frac{1}{k}\sum_{i=1}^{k} f(x_{t-i})$$

Properties:

- Approximate statistic with value computed in the window
- ignore older elements
- window size k is a fixed param

Memoryless – EWMA

- IDEA: memoryless estimator-based concept drift detection with exponential averaging

EWMA - exponentially weighted moving average

$$A_t = \alpha x_t + (1-\alpha)A_{t-1}, \quad A_1 = x_1$$

- α decay factor

Properties

- Does not need a window size
- α controls the forgetting. It's an exponential factor

CD Detection

CD Detection Methods

IDEA: Provide an alarm when a change is detected

Methods:

- Statistical tests monitoring the input distribution:
  - CUSUM
  - Page-Hinkley
- Monitoring the model's accuracy:
  - DDM - Drift Detection Method
  - EDDM - Early Drift Detection Method
  - ADWIN - ADaptive sliding WINdow

Evaluation

tradeoff between detection and false positives

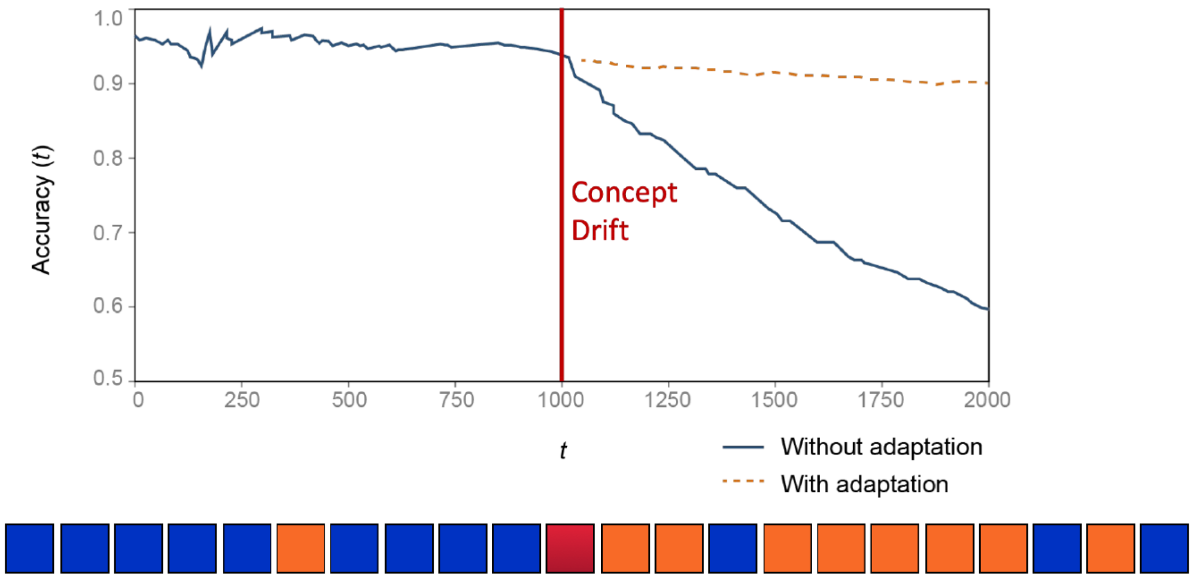
- Detect too early and you get false positives, wasting time for retraining
- Detect too late and the model is going to make a lot of errors

Many metrics depend on minimum magnitude θ of the changes we want to detect.

- **Mean time between false alarms (MTFA)**: how often we get false alarms when there is no change.
  - The *false alarm rate* $FAR = \frac{1}{MTFA}$
- **Mean time to detection (MTD(θ))**: capacity of the system to detect and react to change when it occurs.
- **Missed detection rate (MDR(θ))**: probability of not generating an alarm when there has been change.
- **Average run length (ARL(θ))**: generalizes MTFA and MTD, indicates how long we have to wait before detecting a change after it occurs.
  - MTFA = ARL(0)
  - MTD(θ) = ARL(θ) for θ > 0

Evaluation

- **Mean time between false alarms (MTFA)**: how often we get false alarms when there is no change.
    - The *false alarm rate* $FAR = \frac{1}{MTFA}$
- **Mean time to detection (MTD(θ))**: capacity of the system to detect and react to change when it occurs.
- **Missed detection rate (MDR(θ))**: probability of not generating an alarm when there has been change.

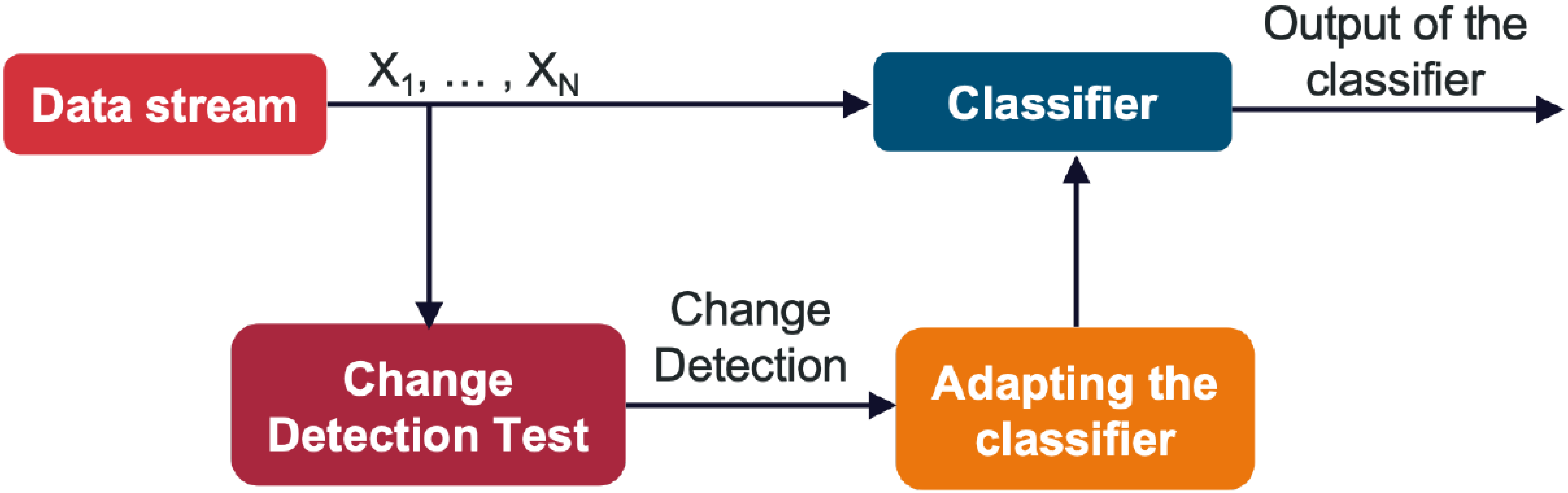


Monitoring the Input Distribution

A class of CD detectors works by monitoring only the input distribution

- **Advantage**:
  - Does not require supervised samples
  - We can use simple statistical tests to detect changes
- **Disadvantages**:
  - Difficult to design detectors for multivariate streams or where the underlying distribution is unknown
  - It does not detect changes that do not affect the distribution of observations

CD Architecture Monitoring the Input Distribution



CUSUM Test

- *give an alarm when the mean of the input data significantly deviates from its previous value*
- **CUSUM** = cumulative sum control chart
- x_t input sequence
- $z_t = (x_t - \mu)/\sigma$ standardized input
- $g_t = \max(0, g_{t-1} + z_t - k)$ relative residual error
- mean and std μ, σ are given a priori or estimated from the input sequence

CUSUM (k, h hyperparameters):

- $g_0 = 0$
- $g_t = g_{t-1} + z_t - k$
- $g_t > h$, declare change and reset g_t= 0, and μ and σ.

CUSUM Properties

Properties

- Doesn't need a fixed window
- O(1) cost per element
- *One-sided test*: only detects changes in the positive direction (use min to detect negative changes)

Guidelines:

- set k to half the value of the change to be detected (in std)
- Set h to $\ln\frac{1}{\delta}$, where δ is an acceptable False Alarm Rate

CUSUM (k, h hyperparameters):

- $g_0 = 0$
- $g_t = g_{t-1} + z_t - k$
- $g_t > h$, declare change and reset g_t= 0, and μ and σ.

Page-Hinkley

- *Similar to CUSUM. Give an alarm when the mean of the input data significantly deviates from its previous value*
- PAGE-HINKLEY
- $g_0 = 0$
- $g_t = g_{t-1} + z_t - k$
- $G_t = \min\{g_t, G_{t-1}\}$
- If $g_t - G_t > h$, declare change and reset $g_t = 0$

Monitoring the Classification Error

- Monitoring the input distribution is simple but somewhat limited
- Instead, we can monitor the classification error

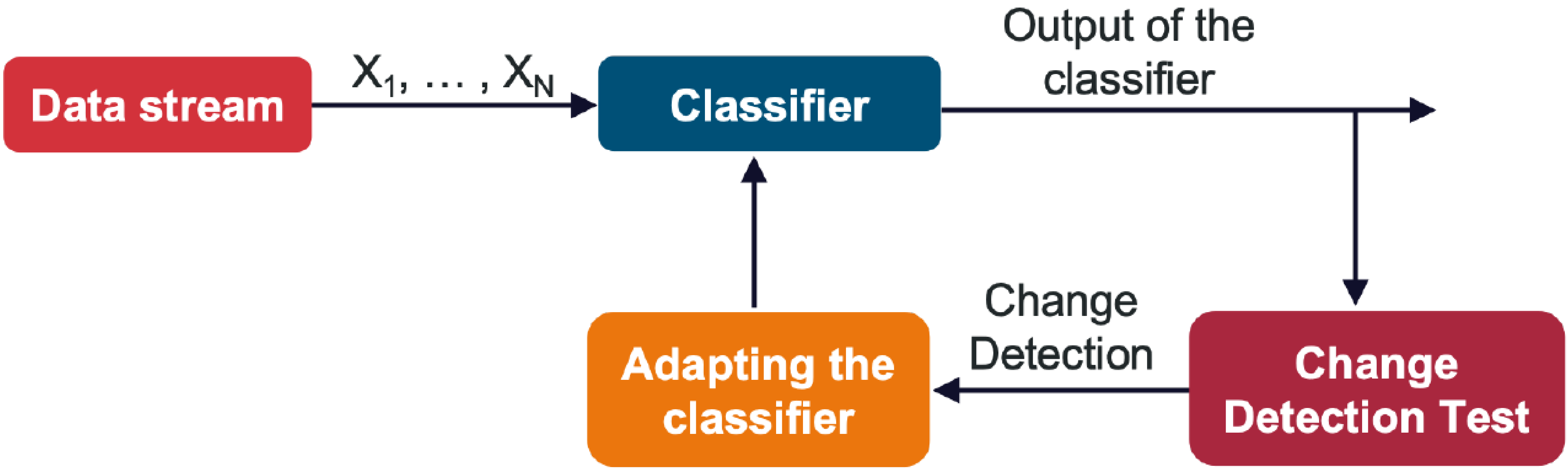
Advantages:

- Straightforward: if the accuracy is low, we know we need to retrain
- Simple: the classification error is a one-dimensional time series even if the input is very complex

Disadvantages:

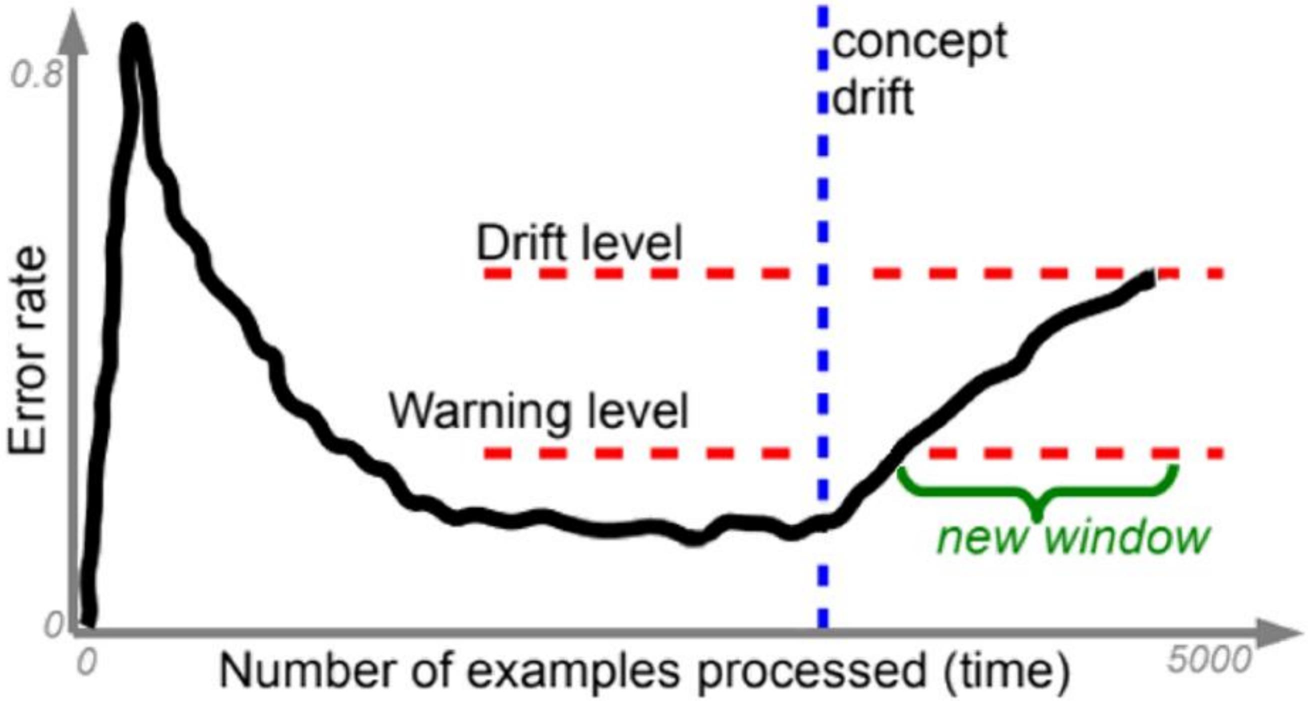
- We can do it only if we have supervised signals to compute the error.

CD Architecture Monitoring Classification Error



Warning and Drift Level

Warning level: change is detected but is not high enough to be considered a drift
Drift level: the change is big enough to be considered a drift
The difference between the two levels causes a latency between the start of the concept drift and the detection time



Gama, et. al, Learning with Drift Detection, SBIA 2004, Springer.

CD Detection with Prediction Error

Problem

- Given an input sequence $X_1, X_2, \dots, X_t$ we want to output an alarm at time t if there is a distribution change.
- We may use the prediction error $|\hat{X}_{t+1} - X_{t+1}|$

Outputs

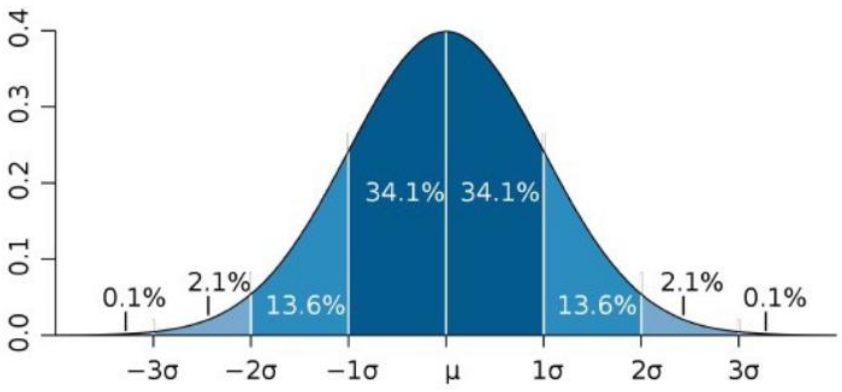
- An estimate of some parameters of the distributions (e.g. the mean of the recent inputs)
- An alarm that signals the distribution change

DDM – Drift Detection Method

- DDM is a drift detection based on model's accuracy

IDEA:

- without drifts, error should decrease over time as more data is used
- If the errors increase, the we have a drift
- If we can model the distribution of errors over time, we can detect the drift by detecting unexpected values in the error distribution



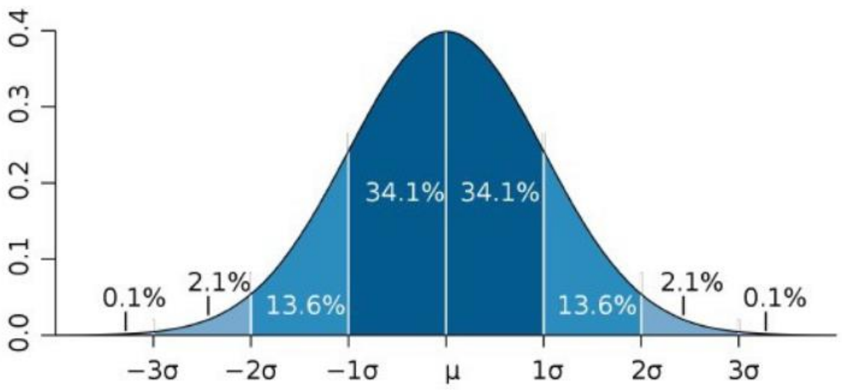
Gama, et. al, Learning with Drift Detection, SBIA 2004, Springer.

DDM – Drift Detection Method

- DDM is a drift detection based on model's accuracy

How to model the error distribution:

- Given p_t error rate at time t
  - *This is the value we expect not to decrease!*
- number of errors in a binomial distribution of t examples has std $s_t = \sqrt{p_t(1 - p_t)/t}$



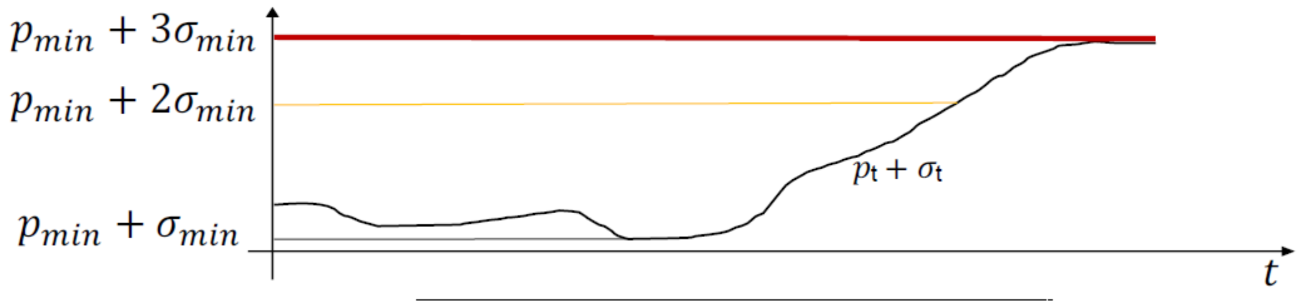
Gama, et. al, Learning with Drift Detection, SBIA 2004, Springer.

DDM – Drift Detection Method

- p_{min} = minimum error rate measured up to time t
- s_{min} = minimum standard deviation measured at time t

Algorithm (DDM):

- If $p_t + s_t \geq p_{\min} + 2 \cdot s_{\min} \rightarrow$ *warning*.
  - From now on, start storing examples in the buffer to prepare for retraining.
- If $p_t + s_t \geq p_{\min} + 3 \cdot s_{\min} \rightarrow$ *drift*.
  - Discard the previous model
  - Train a new model using the buffer collected from the warning time.
  - Reset p_{min} and s_{min}

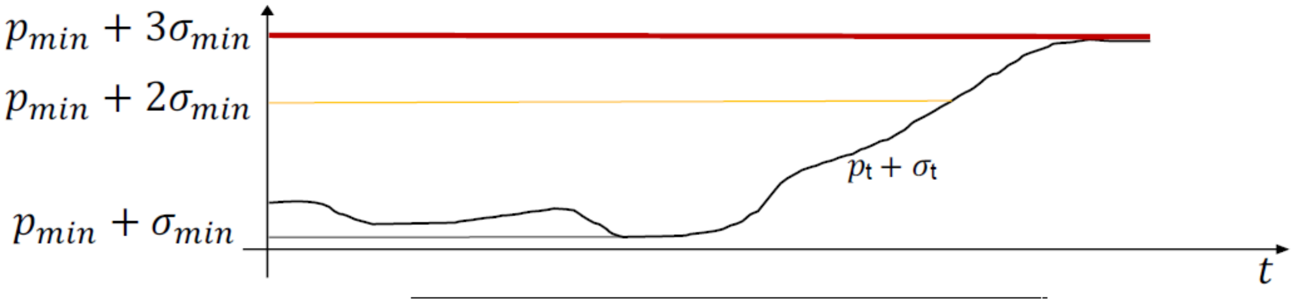


Gama, et. al, Learning with Drift Detection, SBIA 2004, Springer.

DDM - Properties

1. Simple and general method
2. May be slow to changes since p_t is computed over all the examples since the last drift
3. Memory occupation depends on the distance between warning and drift

- We can use EWMA to estimate errors
  - Partially mitigates (2)



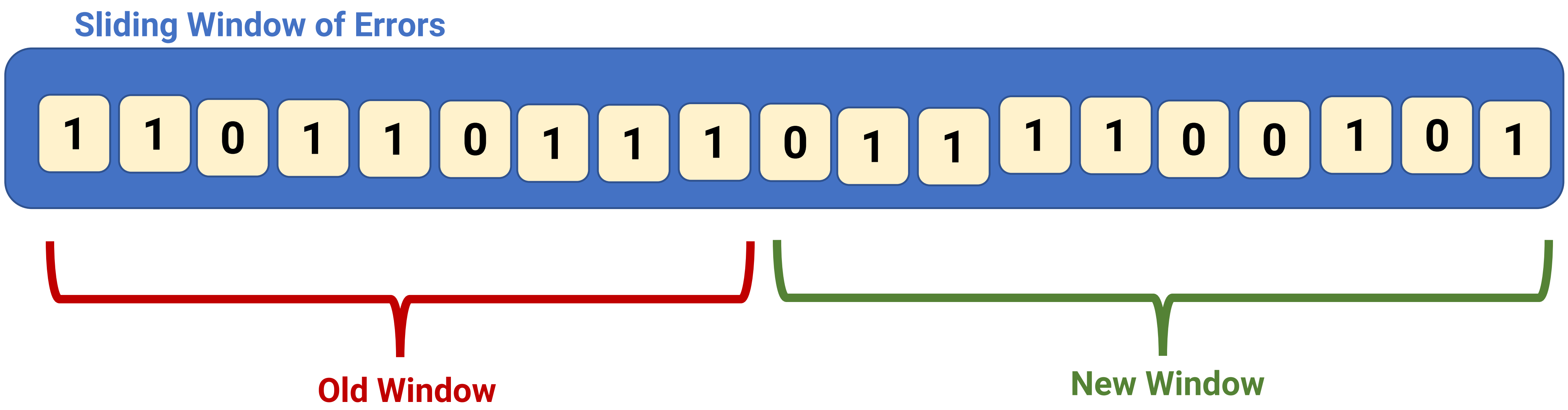
EDDM – Early DDM

- It considers the distance between two errors classification instead of considering only the number of errors.
- While the learning method is learning, it will improve the predictions and the distance between two errors will increase.
- When a drift occurs, the distance between two errors will decrease.
- Compute the average distance between 2 errors and its std, and look for outliers in the tails.

ADWIN

1=correct
0=error
Sum=number of errors
Drift=large difference in #errors

Q: How do we keep a sum of error for each window?
Q: How can we compare multiple windows?
The trivial solution is O(W) in memory and time.

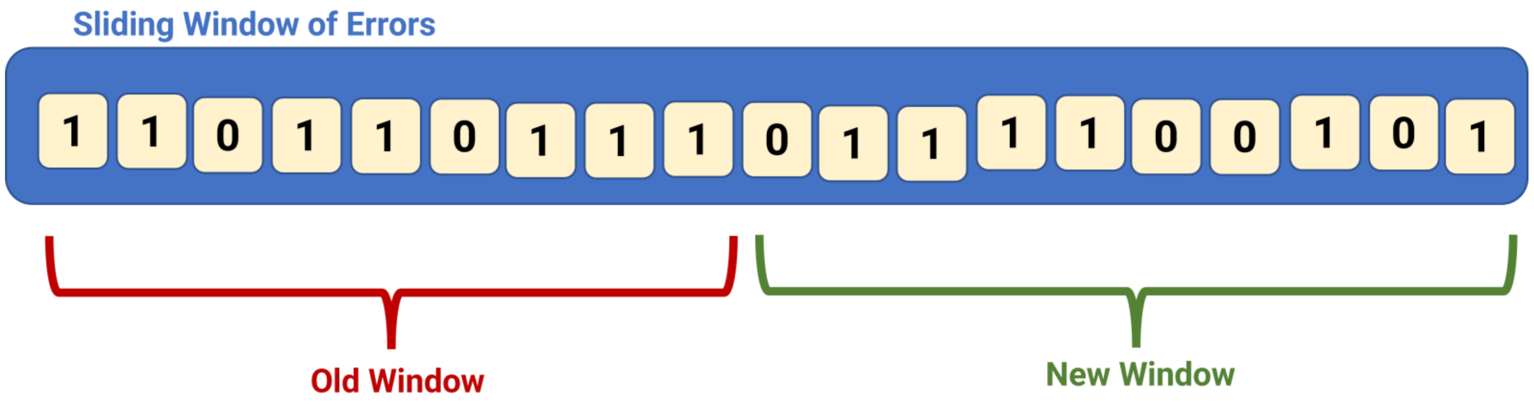


ADWIN – ADaptive sliding WINdow

- **IDEA**: we want a method that compares multiple sliding windows of different lengths

ADWIN:

- **Exponential Histograms** for efficient computation of sums of sliding windows of different sizes
- Statistical test to detect differences in error distribution
- Apply the test to all the possible sliding windows



A. Bifet, R Gavalda: Learning from Time-Changing Data with Adaptive Windowing. SDM2007

Exponential Histograms

- Example of sketching algorithm: Computes an approximation of statistics over a stream using a fixed memory
- Group the stream into buckets. We have the sum for each bucket
- Last bucket may have errors
- We won't see the algorithm in detail (it's a pure algorithmic problem).

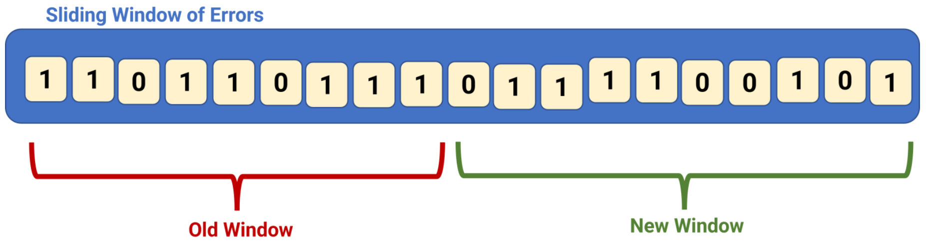
Exponential Sketch

Bucket:	1011100	10100101	100010	11	10	1000
Capacity:	4	4	2	2	1	1
Timestamp:	$t-24$	$t-14$	$t-9$	$t-6$	$t-5$	$t-3$

Figure 4.10

Partitioning a stream of 29 bits into buckets, with $k=2$. Most recent bits are to the right.

Original Sliding Window (values are different)



ADWIN Algorithm

ADWIN Statistical Test

- $\delta \in (0,1)$ confidence value,
- A statistical test $T(W_0, W_1, \delta)$ that compares averages of two windows and decides whether they come from the same distribution.
  - If W_0 and W_1 were generated from the same distribution (no change), then with probability at least 1 − δ the test says "no change."
  - If W0 and W1 were generated from two different distributions whose average differs by more than some quantity $\epsilon(W_0, W_1, \delta)$ then with probability at least 1 − δ the test says "change."

Algorithm

for each new element

  update the exponential histogram

  runs b-1 the statistical test T using

    W_0 oldest i buckets of the exponential histogram

    W_1 newest $b - i$ buckets of the exponential histogram

  drop old buckets when a change is detected

Conclusion and Take-home Messages

- **Concept drift** is a fundamental problem in OML
- First, you need to **identify the types of drifts** in your domain
- Then, you have to deal with the drift:
  - **Estimate** parameters that drift over time
  - **Detect** drift and retrain
- **Retraining** can be expensive. Next lecture we see OML training methods

References

- Streaming Data Analytics Course - Emanuele Della Valle and Alessio Bernardo @ POLIMI
- MOA Book

Next Lecture

Online Classification Models

- Basics
- Naturally online methods: SGD, Naive Bayes
- From offline to online methods:
  - kNN -> online kNN
  - Decision Tree -> VFDT