COMPSCI361: Introduction to Machine Learning

Data Stream Mining I

CS762 S1 2023

Instructor: Thomas Lacombe
Adapted from Yun Sing Koh

Overview

- Data Stream Model
- Statistics on streams; frequent elements
- Concept Drift
- Drift Detector

Traditional (static/batch) data vs streaming data

- Traditional machine learning uses static/batch datasets. The model is trained on the collected data, and once trained, can be used to make predictions.
- In a lot of real-world problems, the data is not static, but comes as a flow of instances ordered in time (e.g., sensor data, email traffic, social media feeds, …).
- Models must learn as the data instances are received and adapt to eventual changes.

Data Stream

- Data streams are an algorithmic abstraction to support real-time analytics.
- They are sequences of items, possibly infinite, each item having a timestamp, and so a temporal order.
- Data items arrive one by one, and we build and maintain models, such as patterns or predictors, of these items in real time.

$$S = \{i_1, i_2, \ldots, i_t\}$$

where S is data stream with i_n instances/items.

What is a Data Stream?

- A data stream is a real-time, continuous, ordered (implicitly by arrival time of explicitly by timestamp) sequence of items. It is impossible to control the order in which items arrive, nor it is feasible to locally store a stream in its entirety. (Golab and Ozsu, 2003).
- continuous and sequential input
- typically unpredictable input rate
- can be large amounts of data
- not error free

Algorithmic challenges when dealing with streaming data

- Data stream is large and fast – extract information in real time from it.
- Accept approximate solutions in order to use less time and memory.
- Stream may be evolving, the prediction models have to adapt when there are changes in the data.

Stream mining algorithm: predictors, clusterers, and frequent pattern miners

Algorithms working with data streams have to meet several requirements:

- R1: Process an instance at a time, and inspect it (at most) once.
- R2: Use a limited amount of time to process each instance.
- R3: Use a limited amount of memory.
- R4: Provide answer anytime (prediction, clustering, patterns).
- R5: Adapt to temporal changes (Concept drift).

Stream mining algorithm: classifier

Instances in a data stream are used by the classifier for prediction and training:

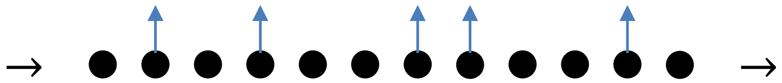
1. Receive an unlabeled example and make a prediction for it on the basis of its current model.

2. Receive the label for an example seen in the past, and use it for adjusting the model, that is, for training.

Stream mining algorithm: Approximation

- The data stream size can be very large, therefore not all instances can be stored and considered.
- Stream mining algorithms are usually working with approximations:
    - Sampling
    - Counting items and distinct items
    - Counting within a sliding window
    - Merging sketches

Sampling



- Sampling approach: for each item in the stream, process it with probability α, and ignore it with probability $1 - \alpha$.
- Reservoir sampling

Reservoir sampling

- A randomized algorithm for choosing a simple random sample without replacement of k items from a population of unknown size in a single pass over the items.
- Property that no matter how many stream elements t have been read so far, each of them is currently in the reservoir with the same probability, k/t.
- It is equally biased toward early elements and late elements.

Reservoir sampling (Vitter (1985))

- A randomized algorithm for choosing a simple random sample without replacement of k items from a population of unknown size in a single pass over the items.
- Goal: maintains a fixed-size uniform random sample
- Put the first k elements from the stream into the repository (Reservoir S)
- When the t-th element arrives where $t > k$.
- Add it to reservoir S with probability $p = k/t$.
- If added, randomly remove an element from S.

Counting items and distinct items

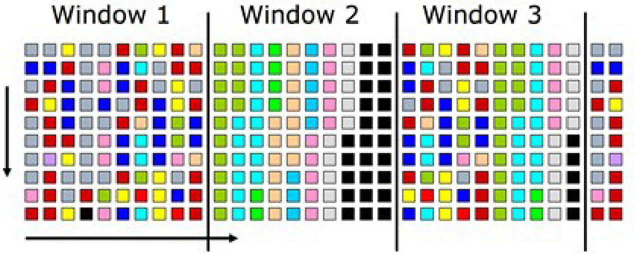
- Counting the total number of items seen so far is perhaps the most basic question one can imagine on a stream. The trivial solution keeps a standard integer counter, and has the update operation just increment the counter.
- Example: Linear Counting.
- Estimating the frequencies of all items is impossible to do in general in sublinear memory, but often it suffices to have approximate counts for the most frequent items.
- Example: Lossy Counting.

Lossy Counting (Manku, Motwani '02)

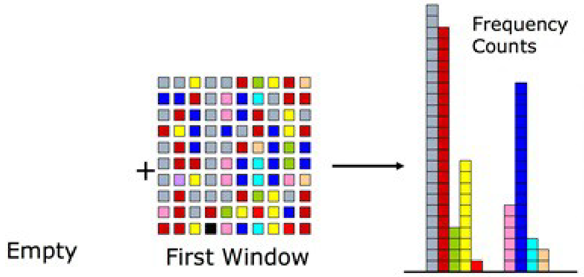
- General version:
  - Divide the incoming data stream into buckets of width $1/\epsilon$, where ϵ is mentioned by user as the error bound.
  - Increment the frequency count of each item according to the new bucket values. After each bucket, decrement all counters by 1.
  - Repeat – Update counters and after each bucket, decrement all counters by 1.

- Counts are accurate to $\epsilon \times N$.
- Analysis shows $O(1/\epsilon \log(\epsilon \times N))$ items are stored.

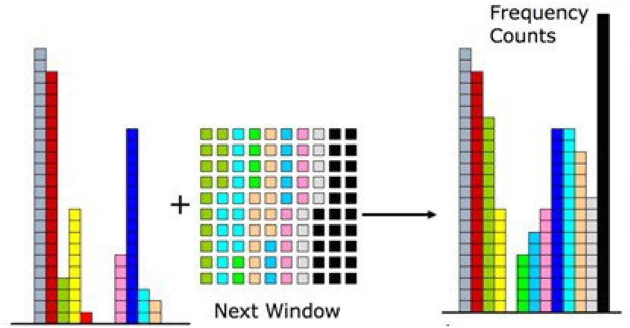
Lossy Counting (Manku, Motwani '02)



Lossy Counting (Manku, Motwani ’02)

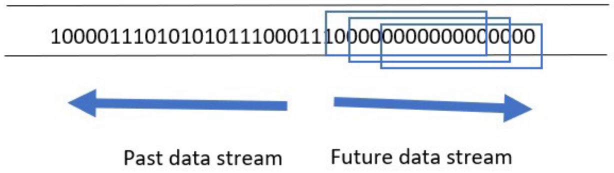


Lossy Counting (Manku, Motwani ’02)



Counting within a sliding window

- Only consider the last *n* items
- Clear way to bound memory
- Natural in applications: emphasizes most recent data
- Data that is too old does not affect our decisions



Exponential Histogram

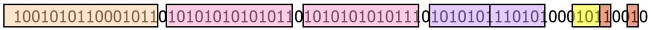
What if we want to focus on recent instances, but also keep information about older ones?

- Problem: Given a stream of 0s and 1s (Error rates).
- How many 1s are in the last k bits? where $k \leq N$?

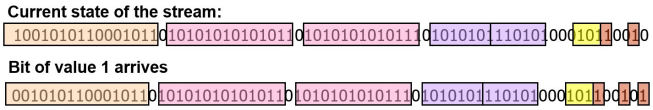
- Trick: merge buckets based on the number of bits they contain
- Constraint on buckets: Number of 1s must be a power of 2
- Buckets that are maintained:
  - Either one or two buckets with the same power-of-2 number of 1s
  - Buckets do not overlap in timestamps

Exponential Histogram

Current state of the stream:

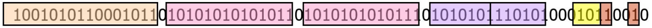


Exponential Histogram

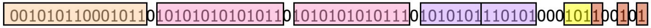


Exponential Histogram

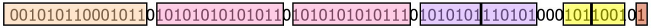
Current state of the stream:



Bit of value 1 arrives

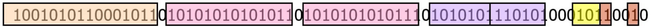


Two orange buckets get merged into a yellow bucket

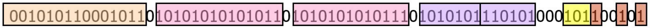


Exponential Histogram

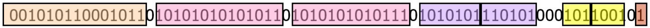
Current state of the stream:



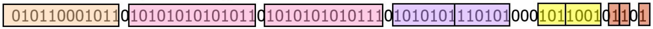
Bit of value 1 arrives



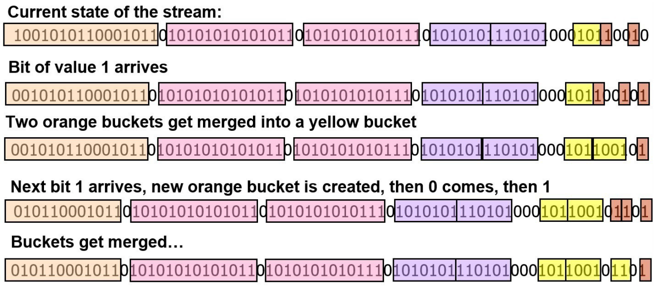
Two orange buckets get merged into a yellow bucket



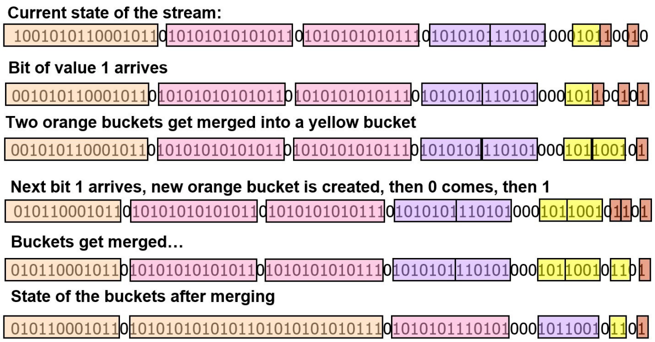
Next bit 1 arrives, new orange bucket is created, then 0 comes, then 1



Exponential Histogram



Exponential Histogram



Summary: approximation

- Data Stream Model
- Statistics on streams
- Sampling
- Lossy Counting
- Sliding Window
- Exponential Histogram

Concept Drift

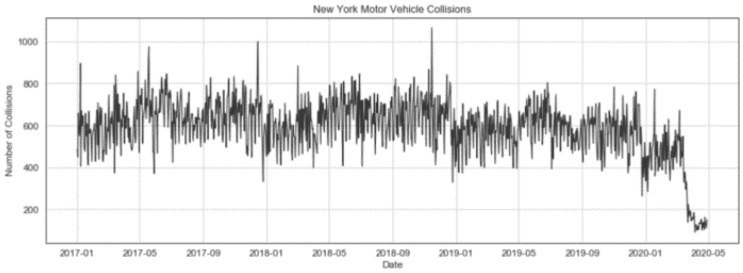


Image Credit: https://www.phdata.io/the-impact-of-covid-19-on-machine-learning-models/

Concept Drift

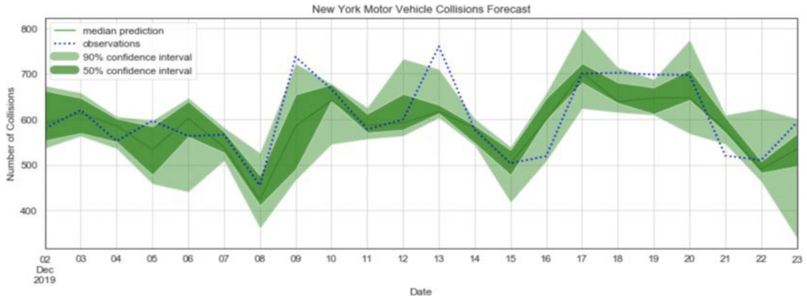


Image Credit: https://www.phdata.io/the-impact-of-covid-19-on-machine-learning-models/

Concept Drift

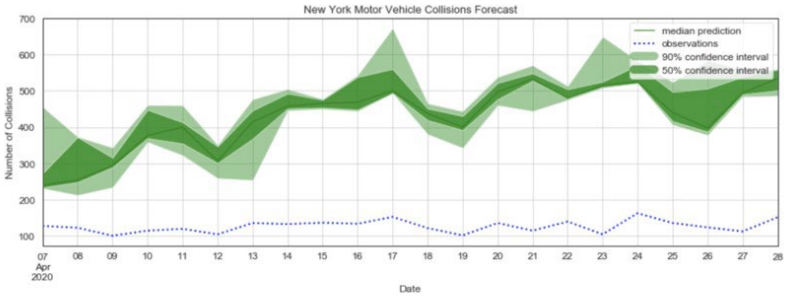
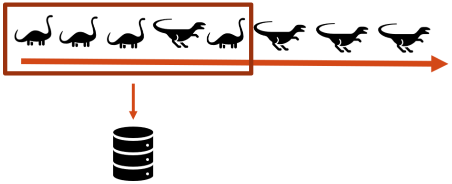
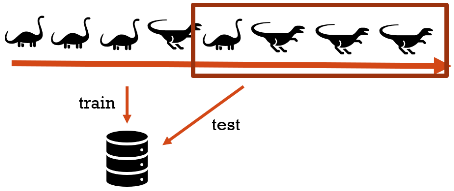


Image Credit: https://www.phdata.io/the-impact-of-covid-19-on-machine-learning-models/

Example of Change in Model.



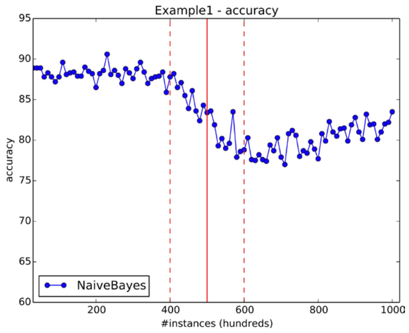
Example of Change



Example of Change

- Change detection on evaluation of model
- Training error should decrease with more examples
- Change in distribution of prediction error (Input into the change detector can be the prediction error, i.e., 0 represents a right prediction and 1 represents a wrong prediction.)
- Input = stream of real/binary (prediction error)

Example of Change



What is change?

- Nonstationary distributions of data may appear in batch data analysis.
- Data in batch datasets may also be timestamped and vary statistically over time.
- Algorithms may take this possibility into account when drawing conclusions from the data, but otherwise can perform several passes and examine data from before and after any given recorded time.
- In streaming, we cannot explicitly store all past data to detect or quantify change, we cannot use data from the future to make decisions in the present.

What is change?

- In time series analysis, the emphasis is often on forecasting the future evolution of the data.
- For example, if the average increase in sales has been 1% per month in the last six months, it is reasonable to predict that sales next month are going to be 1% higher than this month.
- Most of the work in streaming does not necessarily assume that change occurs in predictable ways, or has trends.
- The task is to build models describing how the world behaves right now, given what we are observing right now.

What is change?

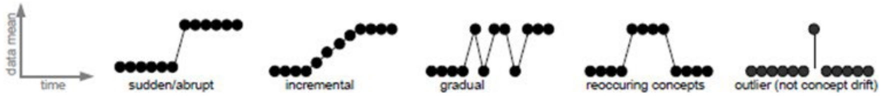
- Dataset shift occurs whenever training and testing datasets come from different distributions.
- Streaming is one instance of this setting; for example, the predictive model we have built so far is going to be used to predict data that will arrive later, which may follow a different distribution.

- But it also applies to other situations, for instance where a classifier trained with batch data from a geographical region is applied to predict data coming from another region

Types of change

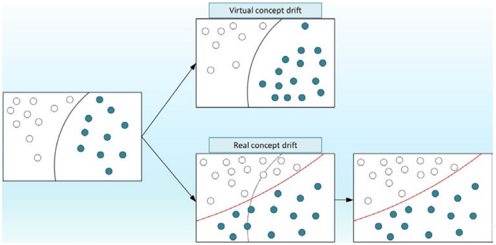
- Change in general is often called **concept drift**.
- **Sudden** or **abrupt** change occurs when the distribution has remained unchanged for a long time, then changes in a few steps to a significantly different one. It is often called shift.
- **Gradual** or **incremental** change occurs when, for 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/local** depending on whether it affects all of the item space or just a part of it.
- **Recurrent concepts** occur when distributions that have appeared in the past tend to reappear later.

Concept Drift Type



Concept Drift vs Virtual Drift

- In prediction, we are expected to predict some outcome feature Y of an item given the values of input features X observed in the item.
- **Real change** occurs when $Pr[Y|X]$ changes, with or without changes in $Pr[X]$.
- **Virtual change** occurs when $Pr[X]$ changes but $Pr[Y|X]$ remains unchanged.



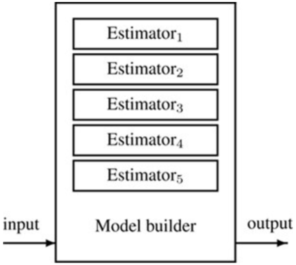
Dealing with Concept Drift

- Detect change in the stream (and adapt the models, if needed) as soon as possible.
- At the same time, be robust to noise and outliers.
- Operate in less than instance arrival time and sublinear memory (some fixed, preset amount of memory).

Change management strategies

1. Adaptive estimators (implicit change adaptation)

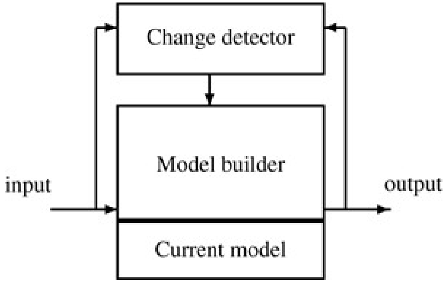
   - Adaptive estimators monitor relevant statistics, and model maintained in synchrony with these statistics.
   - E.g., Naive Bayes keeps counts of co-occurrences of attribute values and class values, or perceptron algorithm updates weights considering agreement between attributes and the outcome to be predicted.



Change management strategies

2. Explicit change detection (change/drift detector)

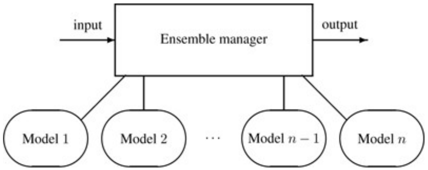
- Monitor statistics (e.g., error rate, accuracy) with an explicit change/drift detector.
- Create models that are adapted or rebuilt when a change detector indicates that change has occurred.



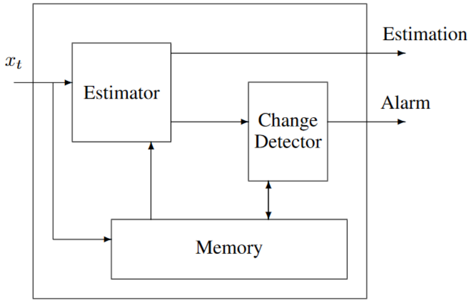
Change management strategies

3. Model ensembles

- A single or several model-building algorithms are called at different times, perhaps on different subsets of the data stream.
- An ensemble manager algorithm contains rules for creating, erasing, and revising the models in its ensemble, as well as for combining the predictions of the models into a single prediction.
- It is mainly the responsibility of the ensemble manager to detect and react to change



Drift Detector



Types of Drift Detectors

- Sequential analysis - CUSUM
- Statistical Process Control - DDM
- Monitoring two distributions - ADWIN

Cumulative Sum (CUSUM)

- Alarm when mean of input data differs from zero
- Parameters: threshold h, drift speed v

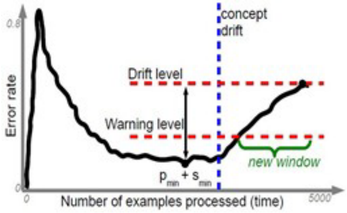
$$g_0 = 0, g_t = \max(0, g_{t-1} + z_t - v)$$

- if $g_t > h$ then alarm; $g_t = 0$
- where $z_t = (x_t - \mu)/\sigma$, where μ is the expected value of the x_t and σ is their standard deviation in "normal" conditions; if μ and σ are not known a priori, they are estimated from the sequence itself.

Drift Detection Method (DDM) (Gama et al.)

- DDM monitors the number of errors produced by a model learned on the previous stream items.
- Generally, the error of the model should decrease or remain stable as more data is used, assuming that the learning method controls overfitting and that the data and label distribution is stationary.
- If DDM observes that the prediction error increases, it takes this as evidence that change has occurred.
- Let p_t denote the error rate of the predictor at time t. Given the number of errors in a sample of t examples, its standard deviation at time t is given by: $S_t = \sqrt{(p_t(1 - p_t)/t}$. DDM stores the smallest value p_{min} of the error rates observed up to time t, and the standard deviation s_{min} at that point.

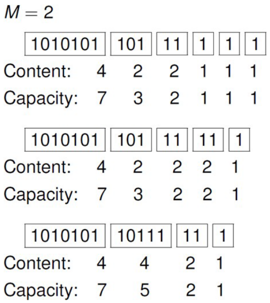
Drift Detection Method (DDM) (Gama et al.)



- If $p_t + s_t \geq p_{min} + \alpha s_{min}$, a warning is declared. Typically $\alpha = 2$. From this point on, new examples are stored in anticipation of a possible declaration of change.
- If $p_t + s_t \geq p_{min} + \beta s_{min}$, change is declared. Typically suggest that the drift is "out-of-control". Typically $\beta = 3$. The model induced by the learning method is discarded and a new model is built using the examples stored since the warning occurred. The values for p_{min} and s_{min} are reset as well.

ADWIN (Bifet et al.)

- The ADWIN algorithm (for ADaptive sliding WINdow) is a change detector and estimation algorithm based on the exponential histograms.



$M = 2$

	1010101	101	11	1	1	1
Content:	4	2	2	1	1	1
Capacity:	7	3	2	1	1	1

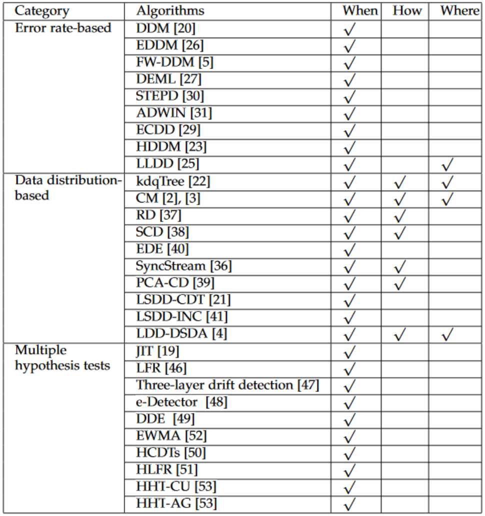
	1010101	101	11	11	1
Content:	4	2	2	2	1
Capacity:	7	3	2	2	1

	1010101	10111	11	1
Content:	4	4	2	1
Capacity:	7	5	2	1

ADWIN (Bifet et al.)

- The inputs to ADWIN are a confidence value $\delta \in (0, 1)$ and a (possibly infinite) sequence of real values.
- The algorithm is parameterized by a test $T(W0, W1, \delta)$ that compares the averages of two windows W0 and W1 and decides whether they are likely to come from the same distribution:
  - If W_0 and W_1 were generated from the same distribution (no change), then with probability at least $1 - \delta$ the test says "no change."
  - If W_0 and W_1 were generated from two different distributions whose average differs by more than some quantity $\in(W_0, W_1, \delta)$, then with probability at least $1 - \delta$ the test says "change."
  - Statistical test normally used is Hoeffding Bound (δ error tolerance).

Drift Detector Survey (Lu et al.)



Category	Algorithms	When	How	Where
Error rate-based	DDM [20]	√		
	EDDM [26]	√		
	FW-DDM [5]	√		
	DEML [27]	√		
	STEPD [30]	√		
	ADWIN [31]	√		
	ECDD [29]	√		
	HDDM [23]	√		
	LLDD [25]	√		√
Data distribution-based	kdqTree [22]	√	√	√
	CM [2], [3]	√	√	√
	RD [37]	√	√	
	SCD [38]	√	√	
	EDE [40]	√		
	SyncStream [36]	√	√	
	PCA-CD [39]	√	√	
	LSDD-CDT [21]	√		
	LSDD-INC [41]	√		
	LDD-DSDA [4]	√	√	√
Multiple hypothesis tests	JIT [19]	√		
	LFR [46]	√		
	Three-layer drift detection [47]	√		
	e-Detector [48]	√		
	DDE [49]	√		
	EWMA [52]	√		
	HCDTs [50]	√		
	HLFR [51]	√		
	HHT-CU [53]	√		
	HHT-AG [53]	√		

https://arxiv.org/pdf/2004.05785.pdf

Evaluating Change Detection

- Mean Time between False Alarms, MTFA: Measures how often we get false alarms when there is no change. The False Alarm Rate (FAR) is defined as 1/MTFA.
- Mean Time to Detection, MTD: Measures the capacity of the learning system to detect and react to change when it occurs.
- Missed detection rate: Measures the probability of not generating an alarm when there has been change.
- Average run length: This measure, which generalizes MTFA and MTD, indicates how long we have to wait before detecting a change after it occurs, s, relative to the FAR.

Resources

Packages

- MOA Java
- Scikit-Multiflow Python

Theory

- Machine Learning for Data Streams with Practical Examples in MOA (https://moa.cms.waikato.ac.nz/book-html/) – Chapter 3 and 5
- Rajaraman, Anand, and Jeffrey David Ullman. Mining of massive datasets. Cambridge University Press, 2011. - Chapter 4
- Part of the slides are from Machine Learning for Data Streams with Practical Examples
- Part of the slides are from Rajaraman, Anand, and Jeffrey David Ullman. Mining of massive datasets. Cambridge University Press, 2011. - Chapter 4