Change Detection

João Gama
LIAAD-INESC Porto,
University of Porto, Portugal
jgama@fep.up.pt

Outline

Introduction

Data flows continuously over time *Dynamic Environments*.
Some characteristic properties of the problem can change over time.
Machine Learning algorithms assume:

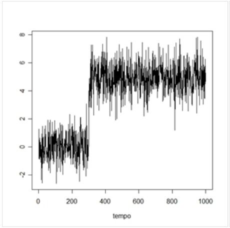
- Instances are generated at random according to some probability distribution $\mathcal{D}$.
- Instances are independent and identically distributed
- It is required that $\mathcal{D}$ is stationary

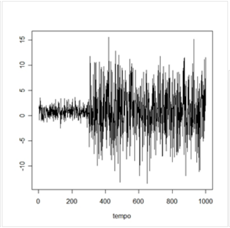
Counter Examples:

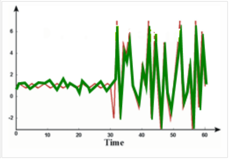
- e-commerce, user modeling
- Spam emails
- Fraud Detection, Intrusion detection

Introduction

Concept drift means that the concept about which data is obtained may shift from time to time, each time after some minimum permanence.
Any change in the distribution underlying the data
Context: a set of examples from the data stream where the underlying distribution is stationary.



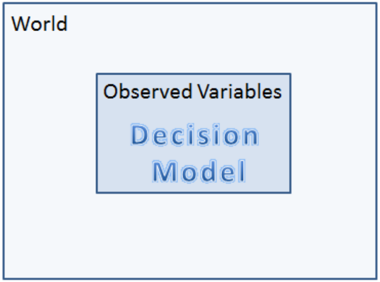




The Nature of Change

The causes of change:

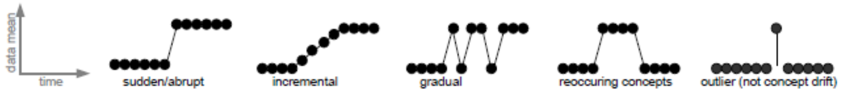
- Changes due to modifications in the context of learning due to changes in **hidden variables**.
- Changes in the characteristic properties of the observed variables.



The Nature of Change

The *rate* of change.

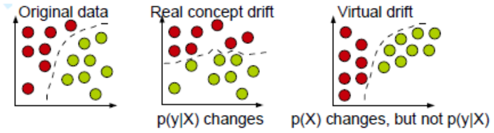
- Concept Drift
  - Abrupt changes in the target concept
- Concept Shift
  - Refers to gradual changes



Patterns of changes over time (outlier is not concept drift).

The Nature of Change

- Whenever a change in the underlying concept generating data occurs, the class-distribution of examples changes,
    - At least in some regions of the instance space.
- It is possible to observe changes in the class-distribution without concept drift.
- This is usually referred as *virtual drift*



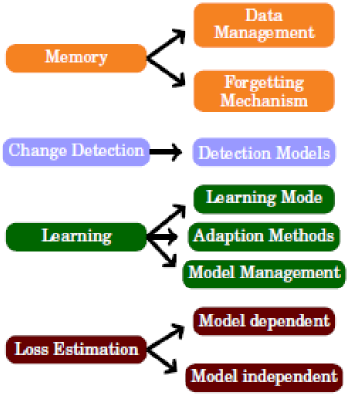
Types of drifts.

The Nature of Change

- Changes in the underlying distribution over the instance descriptions (denoted $D_{\mathcal{X}}$, the distribution over the description space $\mathcal{X}$).
  - The change affects the space of the available instances, but not the underlying regularity.
- Changes in the conditional distribution of the label w.r.t. the description (denoted $D_{\mathcal{Y}|\mathcal{X}}$, where $\mathcal{Y}$ is the label space). This is usually called *concept drift*.
  - the identified decision rule has to be adapted.
- Changes in both of the distributions.

Outline

Taxonomy for Adaptive Learning Algorithms



Four modules of adaptive learning systems with a taxonomy of methods.

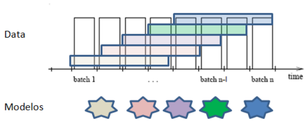
Main Characteristics in Change Detection

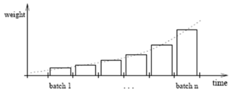
- **Data management**
  Characterizes the information about training examples stored in memory.
- **Detection methods**
  Characterizes the techniques and mechanisms for drift detection
- **Adaptation methods**
  Adaptation of the decision model to the current distribution
- **Decision model management**

Data Management

Characterize the information stored in memory to maintain a decision model consistent with the actual state of the nature.

- **Full Memory:** Store in memory sufficient statistics over all the examples.
- **Partial Memory:** Store in memory only the most recent examples.





Weighting Examples

Full Memory: Store in memory sufficient statistics over all the examples.

- Includes weighting the examples accordingly to their age.
- Oldest examples are less important.
- Suppose that at time i,
  - the stored sufficient statistics is S_{i-1} and
  - we observe an example X_i.
  - Assuming an aggregation function $G(X; S)$,
  - the new sufficient statistics is computed as; $S_i = G(X_i; \alpha S(X_{i-1}))$, where $\alpha \in (0; 1)$ is the fading factor;

Partial Memory

Partial Memory: Store in memory only the most recent examples.

- Examples are stored in a first-in first-out data structure.
- At each time step the learner induces a decision model using only the examples that are included in the window.

The key difficulty is how to select the appropriate window size:

- small window
  - can assure a fast adaptability in phases with concept changes
  - In more stable phases it can affect the learner performance
- large window
  - produce good and stable learning results in stable phases
  - can not react quickly to concept changes.

Partial Memory

- Fixed Size windows.
  - Store in memory a fixed number of the most recent examples. Whenever a new example is available, it is stored in memory and the oldest one is discarded.
  - This is the simplest method to deal with concept drift and can be used as a baseline for comparisons.
- Adaptive Size windows.
  - the set of examples in the window is variable.
  - They are used in conjunction with a detection model.
  - Decreasing the size of the window whenever the detection model signals drift and increasing otherwise.

Data Management

The data management model also indicates the forgetting mechanism.

- Weighting examples:
  - Corresponds to a gradual forgetting.
  - The relevance of old information is less and less important.
- Time windows
  - Corresponds to abrupt forgetting;
  - The examples are deleted from memory.
- Of course we can combine both forgetting mechanisms by weighting the examples in a time window.

Detection Methods

- The Detection Model characterizes the techniques and mechanisms for drift detection.
- An advantage of the detection model is they can provide:
    - meaningful descriptions:
        - indicating change-points or
        - small time-windows where the change occurs
    - quantification of the changes.

Detection Methods

- Monitoring the evolution of performance indicators.
    - Some indicators are monitored over time
      (See Klinkenberg (98) for a good overview of these indicators)
        - Performance measures,
        - Properties of the data, etc
- Monitoring distributions on two different time-windows
  (See D.Kifer, S.Ben-David, J.Gehrke; VLDB 04)
    - A reference window over past examples
    - A window over the most recent examples

Monitoring the evolution of performance indicators

Relevant work to this approach is the FLORA family of algorithms developed by Widmer and Kubat (1996).

- FLORA2 includes a window adjustment heuristic for a rule-based classifier.
- To detect concept changes the accuracy and the coverage of the current learner are monitored over time and the window size is adapted accordingly.
- FLORA3: dealing with recurring concepts.

Monitoring the evolution of performance indicators

In the context of information filtering, Klinkenberg propose monitoring the values of three performance indicators:

- Accuracy, recall and precision over time,
- and then, comparing it to a confidence interval of standard sample errors
- for a moving average value using the last M batches of each particular indicator.

Monitoring the evolution of performance indicators

- D.Kifer, S.Ben-David, J.Gehrke (VLDB 04)
  - Propose algorithms (statistical tests based on Chernoff bound) that examine samples drawn from two probability distributions and decide whether these distributions are different.
- Gama, Fernandes, Rocha: VFDTc (IDA 2006)
  - Continuously monitoring differences between two class-distribution of the examples:
    - the distribution when a node was built and the class-distribution when a node was a leaf and
    - the weighted sum of the class-distributions in the leaves descendant of that node.

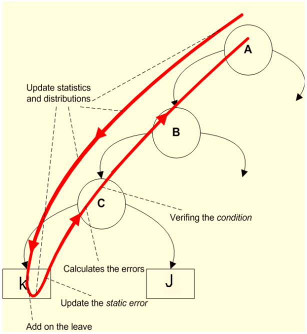
The RS Method

Implemented in the VFDTc system (IDA 2006)

- For each decision node i, two estimates of the classification error.
  - Static error (SE_i): the distribution of the error of the node i;
  - Backed up error (BUE_i): the sum of the error distributions of all the descending leaves of the node i;
- With these two distributions:
  - we can detect the concept change,
  - by verifying the condition $SE_i \leq BUE_i$

The RS Method

- Each new example traverses the tree from the root to a leaf
- Update the sufficient statistics and the class distributions of the nodes
- At the leaf update the value of SE_i
- It makes the opposite path, and update the values of SE_i and BUE_i for each decision node,
- Verify the regularization condition $SE_i \leq BUE_i$.
- If $SE_i \leq BUE_i$, then the node i is pruned to a new leaf.



Adaptation Methods

The Adaptation model characterizes the changes in the decision model do adapt to the most recent examples.

- Blind Methods:
  Methods that adapt the learner at regular intervals without considering whether changes have really occurred.
  - Examples include methods that weight the examples accordingly to their age and methods that use time-windows of fixed size.
- Informed Methods:
  Methods that only change the decision model after a change was detected. They are used in conjunction with a detection model.

Decision model management

Model management characterize the number of decision models needed to maintain in memory.
The key issue here is the assumption that data generated comes from multiple distributions,

- at least in the transition between contexts.
- Instead of maintaining a single decision model several authors propose the use of multiple decision models.

Dynamic Weighted Majority

A seminal work, is the system presented by Kolter and Maloof (ICDM03, ICML05).
The Dynamic Weighted Majority algorithm (DWM) is an ensemble method for tracking concept drift.

- Maintains an ensemble of base learners,
- Predicts using a weighted-majority vote of these *experts*.
- Dynamically creates and deletes experts in response to changes in performance.

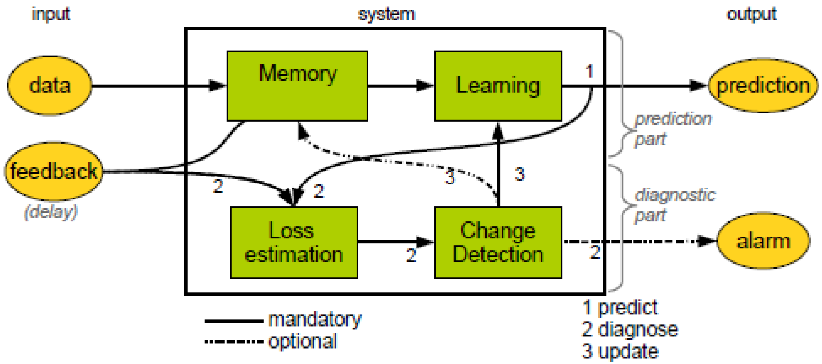
Granularity of Decision Models

Occurrences of drift can have impact in part of the instance space.

- **Global models:** Require the reconstruction of all the decision model. (like naive Bayes, SVM, etc)
- **Granular decision models**: Require the reconstruction of parts of the decision model (like decision rules, decision trees)

Outline

A Generic Model for Adaptive Learning Algorithms



A generic schema for an online adaptive learning algorithm.

Change Detection in Predictive Learning

When there is a change in the class-distribution of the examples:

- The current model does not correspond any more to the current distribution.
- The error-rate increases

Base idea

Learning from data streams is a continuous process. Monitor the quality of the learning process using quality control techniques.

Main Problems:

- How to distinguish Change from Noise?
- How to React to drift?

Monitoring the Learning Process

Gama, et. al, Learning with Drift Detection, Lecture Notes in Computer Science 3171, Springer.

Suppose a sequence of examples in the form $< \vec{x}_i, y_i >$
The current decision model classifies each example in the sequence
In the 0-1 loss function, predictions are either True or False
The predictions of the learning algorithm are sequences:
$T, F, T, F, T, F, T, T, T, F, \ldots$.
The Error is a random variable from *Bernoulli* trials.
The Binomial distribution gives the general form of the probability of observing a F:
$p_i = (F/i)$ and $s_i = \sqrt{p_i(1-p_i)/i}$ where i is the number of trials.

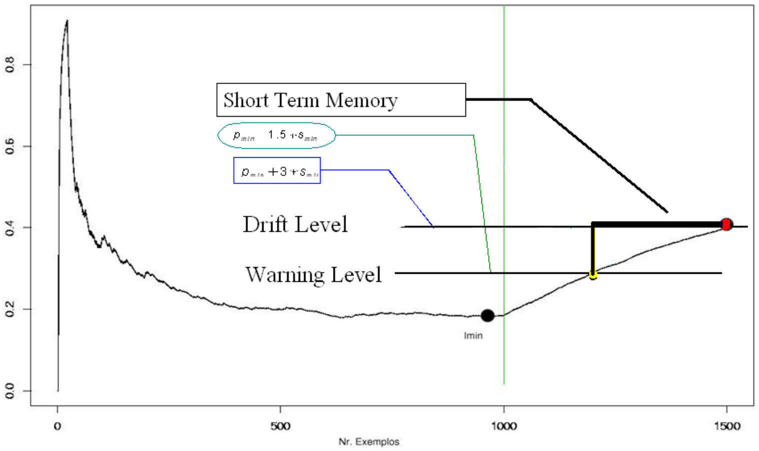
The SPC Algorithm

The *Statistical Processing Control* algorithm maintains two registers: P_{min} and S_{min} such that $P_{min} + S_{min} = min(p_i + s_i)$
Minimum of the error rate taking into account the variance of the estimator.
At example j:
The error of the learning algorithm will be

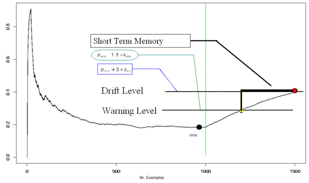
- **Out-control** if $p_j + s_j > p_{min} + \alpha * s_{min}$
- **In-control** if $p_j + s_j < p_{min} + \beta * s_{min}$
- **Warning Level**: if $p_{min} + \alpha * s_{min} > p_j + s_j > p_{min} + \beta * s_{min}$

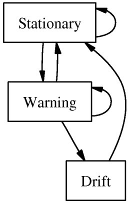
The constants α and β depend on the desired confidence level.
Admissible values are $\beta = 2$ and $\alpha = 3$.

Dynamic Window



The SPC Algorithm

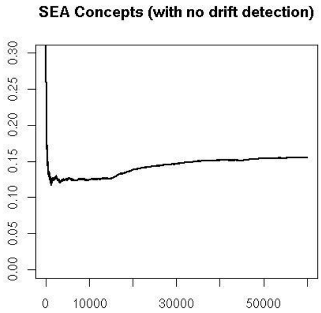


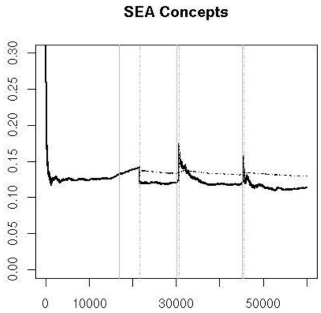


At example j the current decision model classifies the example

- Compute the error and variance: p_j and s_j
- If the error is
  - **In-control** the current model is updated Incorporate the example in the decision model
  - **Warning zone**: Maintain the current model
    First Time: the lower limit of the window is: $L_{warning} = j$
  - **Out-Control** Re-learn a new model using as training set the set of examples $[L_{warning}, j]$.

Illustrative Example





Analysis of the SPC Algorithm

- Based on statistical process control: monitor and control the learning process.
- Independent of the learning algorithm
- Resilient to false alarms
- Maintain a single decision model in memory

Outline

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

$W_0 = \boxed{1}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
1  Initialize Window $W$
2  for each $t > 0$
3      $W = W \cup \{x_t\}$ (i.e., add $x_t$ to the head of $W$)
4      repeat
5          Drop elements from the tail of $W$
6      until $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_c$ holds
7          for every split of $W$ into $W = W_0 \cdot W_1$
8      Output $\hat{\mu}_W$
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

$W_0 = \boxed{1}$ $W_1 = \boxed{01010110111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
1  Initialize Window $W$
2  for each $t > 0$
3        $W = W \cup \{x_t\}$ (i.e., add $x_t$ to the head of $W$)
4        repeat
5              Drop elements from the tail of $W$
6        until $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_c$ holds
7              for every split of $W$ into $W = W_0 \cdot W_1$
8        Output $\hat{\mu}_W$
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

$W_0 = \boxed{10}$ $W_1 = \boxed{1010110111111}$

ADWIN: Adaptive Windowing Algorithm

```
1  Initialize Window $W$
2  for each $t > 0$
3        $W = W \cup \{x_t\}$ (i.e., add $x_t$ to the head of $W$)
4        repeat
5              Drop elements from the tail of $W$
6        until $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_c$ holds
7              for every split of $W$ into $W = W_0 \cdot W_1$
8        Output $\hat{\mu}_W$
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

$W_0 = \boxed{101}$ $W_1 = \boxed{010110111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
1   Initialize Window $W$
2   **for** each $t > 0$
3       $W = W \cup \{x_t\}$ (i.e., add $x_t$ to the head of $W$)
4       **repeat**
5           Drop elements from the tail of $W$
6       **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_c$ holds
7           for every split of $W$ into $W = W_0 \cdot W_1$
8       Output $\hat{\mu}_W$
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

$W_0 = \boxed{1010}$ $W_1 = \boxed{10110111111}$

```
ADWIN: ADAPTIVE WINDOWING ALGORITHM
1   Initialize Window $W$
2   for each $t > 0$
3         $W = W \cup \{x_t\}$ (i.e., add $x_t$ to the head of $W$)
4         repeat
5               Drop elements from the tail of $W$
6         until $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_c$ holds
7               for every split of $W$ into $W = W_0 \cdot W_1$
8         Output $\hat{\mu}_W$
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

$W_0 = \boxed{10101}$ $W_1 = \boxed{0110111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
1 Initialize Window $W$
2 **for** each $t > 0$
3     $W = W \cup \{x_t\}$ (i.e., add $x_t$ to the head of $W$)
4     **repeat**
5         Drop elements from the tail of $W$
6     **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_c$ holds
7         for every split of $W$ into $W = W_0 \cdot W_1$
8     Output $\hat{\mu}_W$
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

$W_0 = \boxed{101010}$ $W_1 = \boxed{110111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
1  Initialize Window $W$
2  **for** each $t > 0$
3      $W = W \cup \{x_t\}$ (i.e., add $x_t$ to the head of $W$)
4      **repeat**
5          Drop elements from the tail of $W$
6      **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_c$ holds
7          for every split of $W$ into $W = W_0 \cdot W_1$
8      Output $\hat{\mu}_W$
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

$W_0 = \boxed{1010101}$ $W_1 = \boxed{10111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
1  Initialize Window $W$
2  **for** each $t > 0$
3        $W = W \cup \{x_t\}$ (i.e., add $x_t$ to the head of $W$)
4        **repeat**
5              Drop elements from the tail of $W$
6        **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_c$ holds
7              for every split of $W$ into $W = W_0 \cdot W_1$
8        Output $\hat{\mu}_W$
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

$W_0 = \boxed{10101011}$ $W_1 = \boxed{0111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
1  Initialize Window $W$
2  for each $t > 0$
3        $W = W \cup \{x_t\}$ (i.e., add $x_t$ to the head of $W$)
4        repeat
5              Drop elements from the tail of $W$
6        until $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_c$ holds
7              for every split of $W$ into $W = W_0 \cdot W_1$
8        Output $\hat{\mu}_W$
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$ $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_c$: CHANGE DET.!

$W_0 = \boxed{101010110}$ $W_1 = \boxed{111111}$

ADWIN: Adaptive Windowing Algorithm

```
1  Initialize Window $W$
2  **for** each $t > 0$
3      $W = W \cup \{x_t\}$ (i.e., add $x_t$ to the head of $W$)
4      **repeat**
5          Drop elements from the tail of $W$
6      **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_c$ holds
7          for every split of $W$ into $W = W_0 \cdot W_1$
8      Output $\hat{\mu}_W$
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{1010101101111111}$ Drop elements from the tail of W

$W_0 = \boxed{101010110}$ $W_1 = \boxed{111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
1   Initialize Window $W$
2   for each $t > 0$
3       $W = W \cup \{x_t\}$ (i.e., add $x_t$ to the head of $W$)
4       repeat
5           Drop elements from the tail of $W$
6       until $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_c$ holds
7           for every split of $W$ into $W = W_0 \cdot W_1$
8       Output $\hat{\mu}_W$
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{01010110111111}$ Drop elements from the tail of W

$W_0 = \boxed{101010110}$ $W_1 = \boxed{111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
1   Initialize Window $W$
2   **for** each $t > 0$
3       $W = W \cup \{x_t\}$ (i.e., add $x_t$ to the head of $W$)
4       **repeat**
5           Drop elements from the tail of $W$
6       **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_c$ holds
7           for every split of $W$ into $W = W_0 \cdot W_1$
8       Output $\hat{\mu}_W$
```

Algorithm ADaptive Sliding WINdow

Theorem

At every time step we have:

1. (False positive rate bound). *If μ_t remains constant within W, the probability that* ADWIN *shrinks the window at this step is at most δ.*
2. (False negative rate bound). *Suppose that for* some *partition of W in two parts $W_0 W_1$ (where W_1 contains the most recent items) we have $|\mu_{W_0} - \mu_{W_1}| > 2\epsilon_c$. Then with probability $1 - \delta$* ADWIN *shrinks W to W_1, or shorter.*

ADWIN tunes itself to the data stream at hand, with no need for the user to hardwire or precompute parameters.

Algorithm ADaptive Sliding WINdow

ADWIN using a Data Stream Sliding Window Model,

- can provide the exact counts of 1's in $O(1)$ time per point.
- tries $O(\log W)$ cutpoints
- uses $O(\frac{1}{\epsilon}\log W)$ memory words
- the processing time per example is $O(\log W)$ (amortized and worst-case).

Sliding Window Model

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

Outline

The Cumulative Sum Algorithm

Page, *Continuous Inspection Scheme*. Biometrika 41 1954

- A sequential analysis technique due to E. Page of the University of Cambridge
- The CUSUM test is as follows:

$$g_0 = 0$$

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

- The decision rule is: **if** $g_t > h$ **then alarm and** $g_t = 0$.

The CUSUM Test

- The CUSUM test is memoryless, and its accuracy depends on the choice of parameters v and h.
- Both parameters are relevant to control the trade-off between earlier detecting true alarms by allowing some false alarms.

The Page-Hinckley Test

- The PH test is a sequential adaptation of the detection of an abrupt change in the average of a Gaussian signal.
- It considers a cumulative variable m_T, defined as the cumulated difference between the observed values and their mean till the current moment:

$$m_{t+1} = \sum_{1}^{t} (x_t - \bar{x}_t + \alpha)$$

  - where $\bar{x} = 1/t \sum_{l=1}^{t} x_l$ and
  - α corresponds to the magnitude of changes that are allowed.

The Page-Hinckley Test

$$m_{t+1} = \sum_{1}^{t}(x_t - \bar{x}_t + \alpha)$$

- The minimum value of this variable is also computed with the following formula: $M_T = min(m_t, t = 1 \ldots T)$.
- The test monitors the difference between M_T and m_T: $PH_T = m_T - M_T$.
- When this difference is greater than a given threshold (λ) we alarm a change in the distribution.

Analysis

The threshold λ depends on the admissible false alarm rate. Increasing λ will entail fewer false alarms, but might miss some changes.

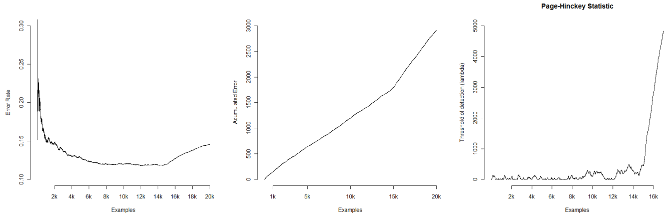


Figure: Illustrative example of the Page-Hinckey test. The left figure plots the on-line error rate of a learning algorithm. The center plot is the accumulated on-line error. The slope increases after the occurrence of a change. The right plot presents the evolution of the *PH* statistic.

Outline

Bibliography on Change Detection

- A survey on concept drift adaptation. J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, A. Bouchachia: ACM Comput. Surv. 46(4): 44 (2014)
- Learning from Time-Changing Data with Adaptive Windowing. A. Bifet, R. Gavalda: SDM 443-448; 2007
- *Detection of Abrupt Changes - Theory and Application*, M. Basseville, I. Nikiforov, Prentice-Hall, Inc. 1993.
- *Statistical Quality Control*, E. Grant, R. Leavenworth, McGraw-Hill, 1996.
- *Continuous Inspection Scheme*, E. Page. Biometrika 41 1954
- *Learning in the Presence of Concept Drift and Hidden Contexts*, G. Widmer, M. Kubat: Machine Learning 23(1): 69-101 (1996)
- *Learning drifting concepts: Example selection vs. example weighting*, R. Klinkenberg, IDA 2004
- *Learning with Drift Detection*; J. Gama, P. Medas, G. Castillo, P. Rodrigues; SBIA 2004, Springer.

Bibliography on Change Detection

- *Mining Time-Changing Data Streams*, by Geoff Hulten, Laurie Spencer, Pedro Domingos, in the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD) 2001.
- *Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift*, by Jeremy Z. Kolter, Marcus A. Maloof, in the IEEE International Conf. Data Mining (ICDM) 2003.
- *Mining Concept Drifting Data Streams using Ensemble Classifiers*, by Haixun Wang, Wei Fan, Philip S. Yu, Jiawei Han, in the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD) 2003.
- *Decision Trees for Mining Data Streams*; J. Gama, R. Fernandes, R. Rocha. Intelligent Data Analysis 10(1):23-45 (2006)