Handling concept drift in data stream mining

Student: Manuel Martín Salvador
Supervisors: Luis M. de Campos and Silvia Acid

SOFT COMPUTING Y SISTEMAS INTELIGENTES

Universidad de Granada

DECSAI
Universidad de Granada

Master in Soft Computing and Intelligent Systems
Department of Computer Science and Artificial Intelligence
University of Granada

Who am I?

1. Current: **PhD Student** in Bournemouth University
2. Previous:
   - **Computer Engineering** in University of Granada (2004-2009)
   - **Programmer and SCRUM Master** in Fundación I+D del Software Libre (2009-2010)
   - **Master in Soft Computing and Intelligent Systems** in University of Granada (2010-2011)
   - **Researcher** in Department of Computer Science and Artificial Intelligence of UGR (2010-2012)

Index

Data streams

1. Continous flow of instances.
   - In classification: instance = $(a_1, a_2, \ldots, a_n, c)$
2. Unlimited size
3. May have changes in the underlying distribution of the data → concept drift

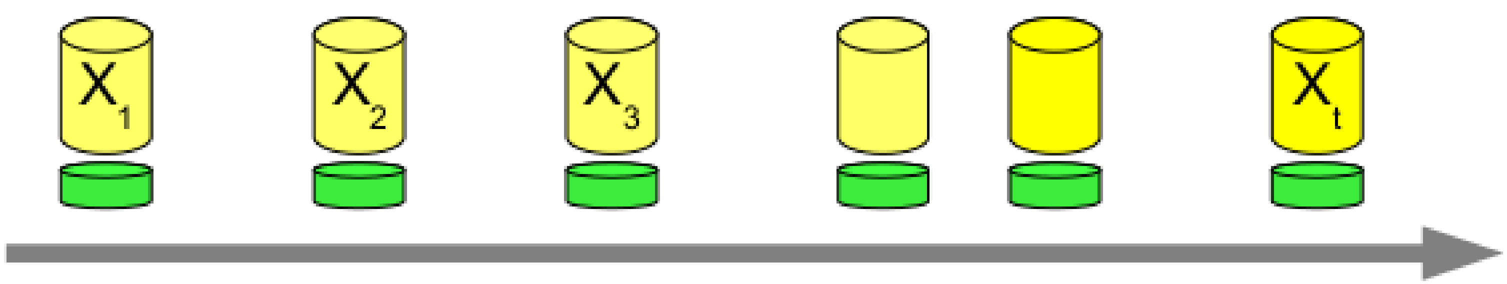


Image: I. Žliobaitė thesis

Concept drifts

- It happens when the data from a stream changes its probability distribution Π_{S1} to another Π_{S2}. Potential causes:
  - Change in P(C)
  - Change in P(X|C)
  - Change in P(C|X)
- Unpredictable
- For example: spam

Gradual concept drift

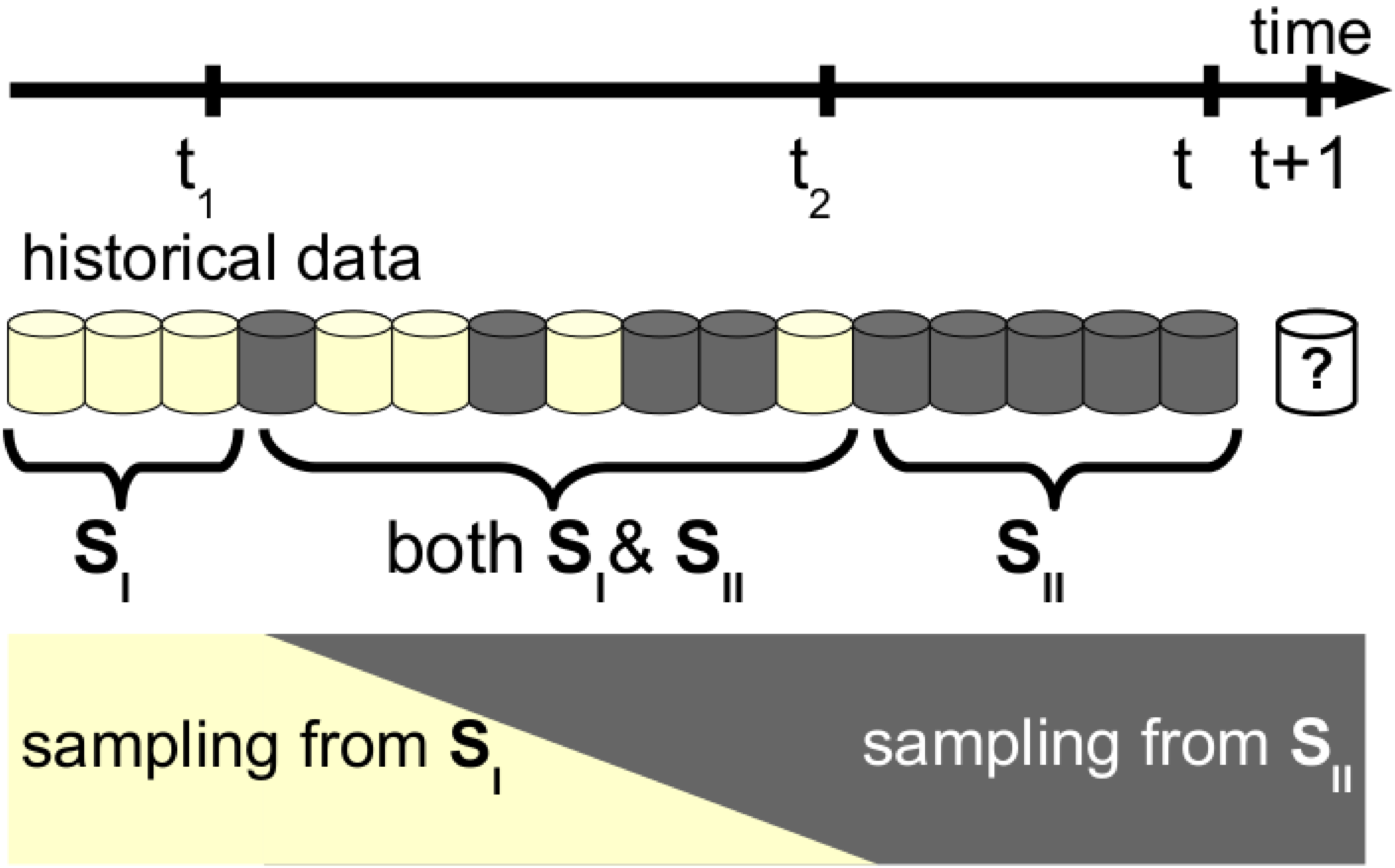


Image: I. Žliobaitė thesis

Types of concept drifts

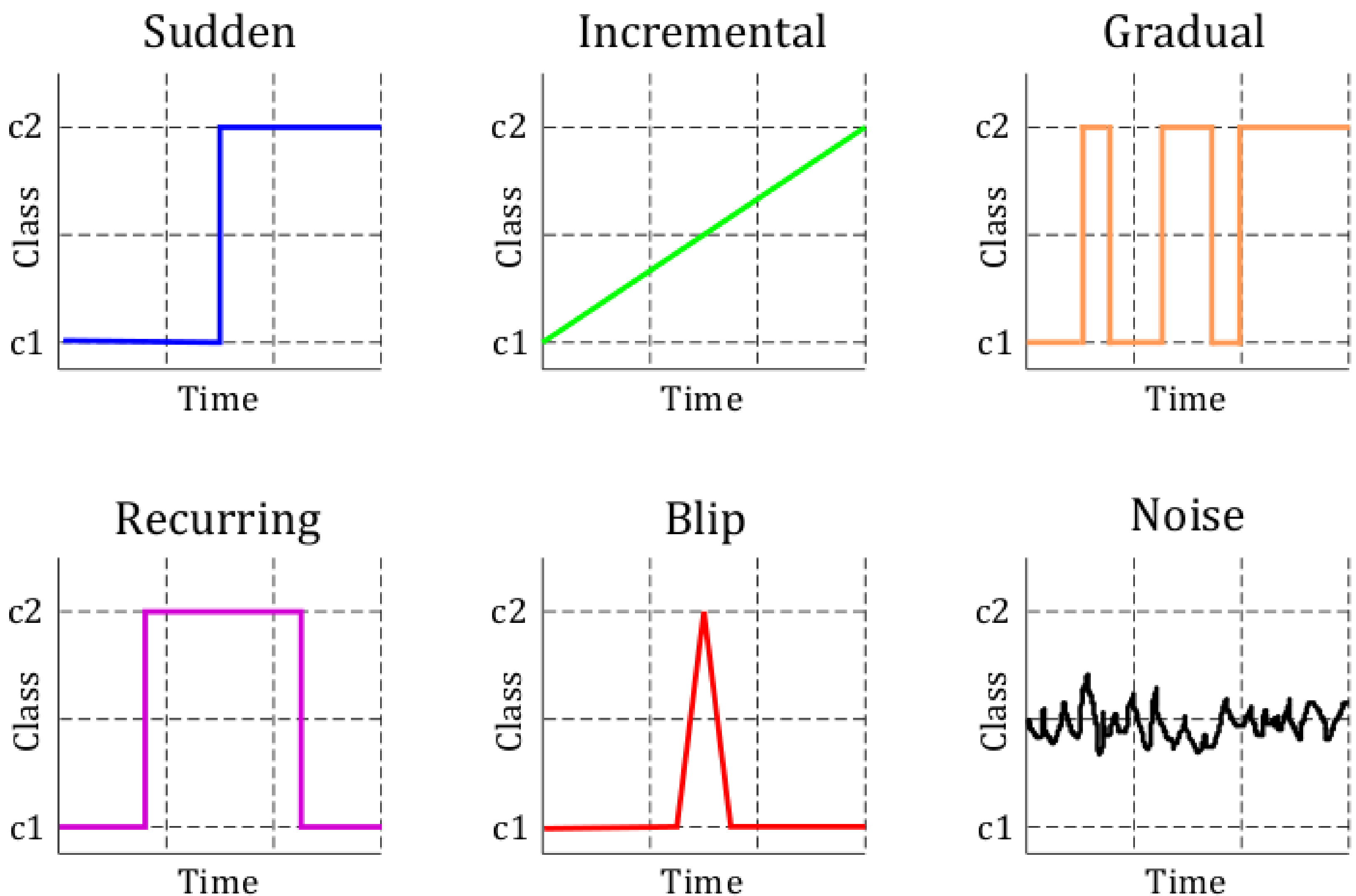


Image: D. Brzeziński thesis

Types of concept drifts

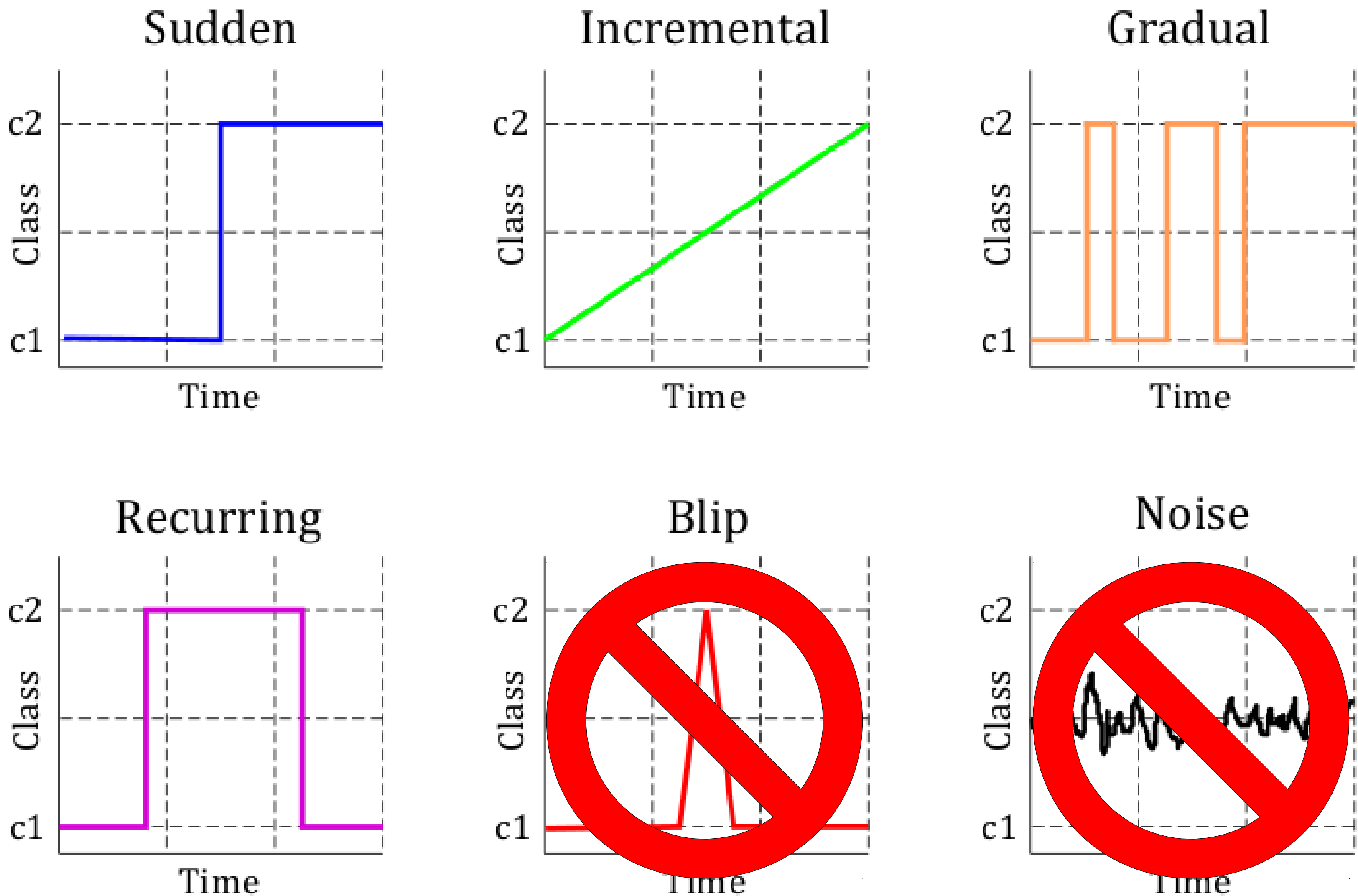
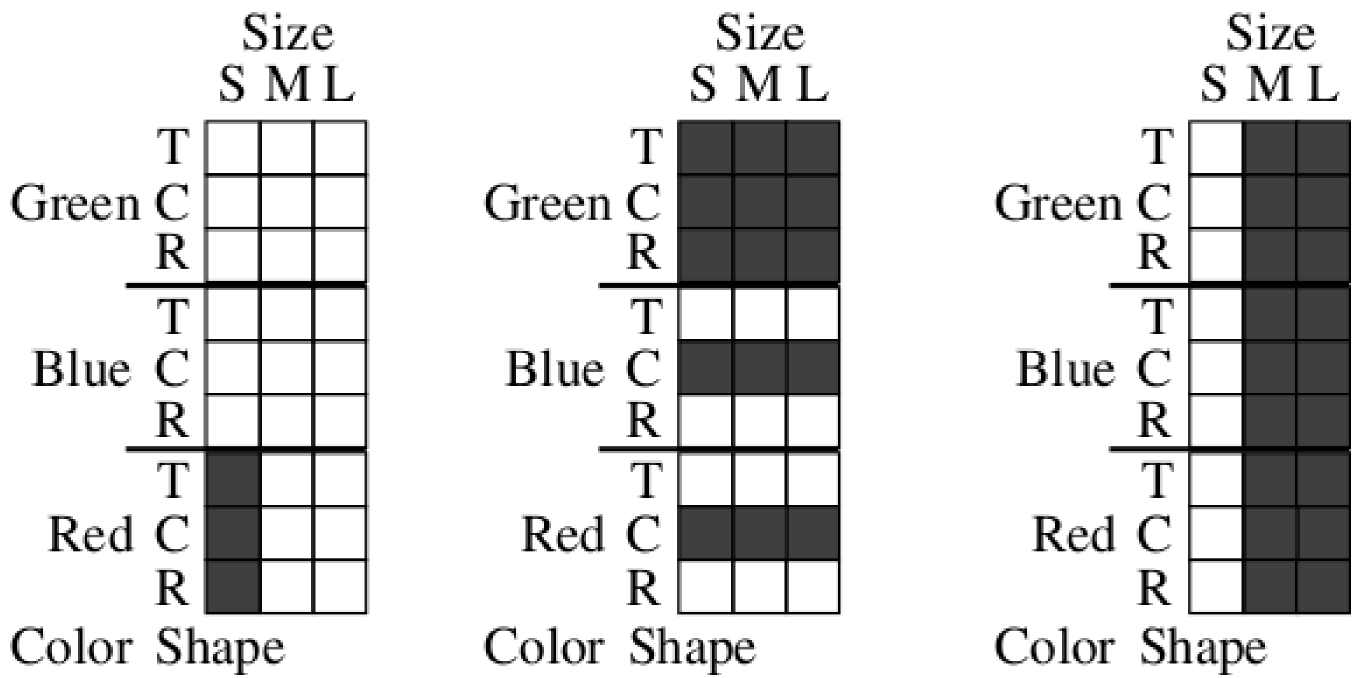


Image: D. Brzeziński thesis

Example: STAGGER

Class=true if → color=red and size=small ↯ color=green or shape=cricle ↯ size=medium or size=large



Target concept $t = 1 \ldots 40.$ Target concept $t = 41 \ldots 80.$ Target concept $t = 81 \ldots 120.$

Image: Kolter & Maloof

Online learning (incremental)

- Goal: incrementally learn a classifier at least as accurate as if it had been trained in batch
- Requirements:
  1. Incremental
  2. Single pass
  3. Limited time and memory
  4. Any-time learning: availability of the model

Online learning (incremental)

- Goal: incrementally learn a classifier at least as accurate as if it had been trained in batch
- Requirements:
  1. Incremental
  2. Single pass
  3. Limited time and memory
  4. Any-time learning: availability of the model
- Nice to have: deal with concept drift.

Evaluation

Several criteria:

- Time → seconds
- Memory → RAM/hour
- Generalizability of the model → % success
- Detecting concept drift → detected drifts, false positives and false negatives

Evaluation

Several criteria:

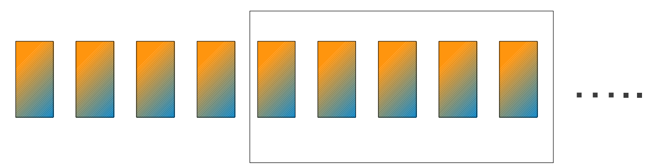
- Time → seconds
- Memory → RAM/hour
- Generalizability of the model → % success
- Detecting concept drift → detected drifts, false positives and false negatives

Problem: we can't use the traditional techniques for evaluation (i.e. cross validation). → Solution: new strategies.

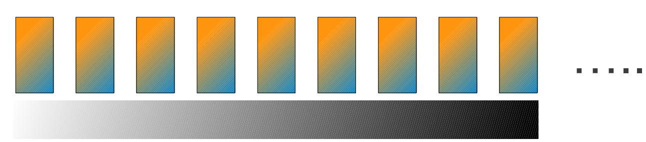
Evaluation: prequential

- Test y training each instance. $\frac{errors}{processed\ instances}$
- Is a pessimistic estimator: holds the errors since the beginning of the stream. → Solution: forgetting mechanisms (sliding window and fading factor).

Sliding window: $\frac{errors\ inside\ window}{window\ size}$

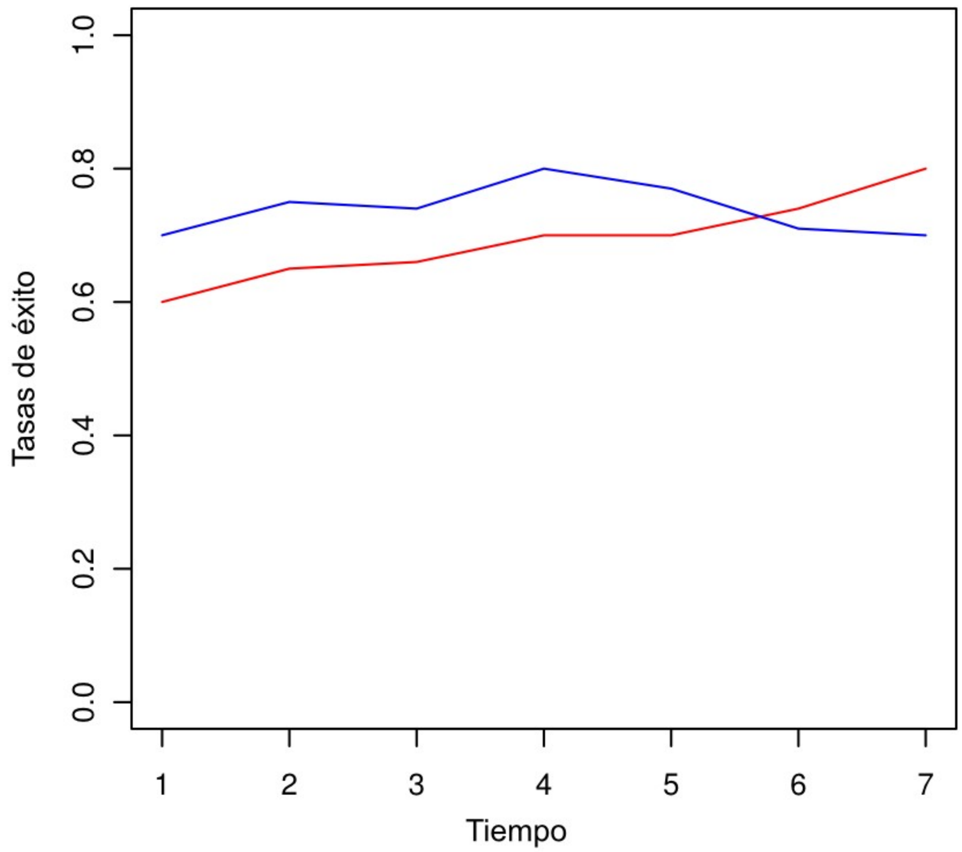


Fading factor: $\frac{currentError + \alpha \cdot errors}{1 + \alpha \cdot processed\ instances}$



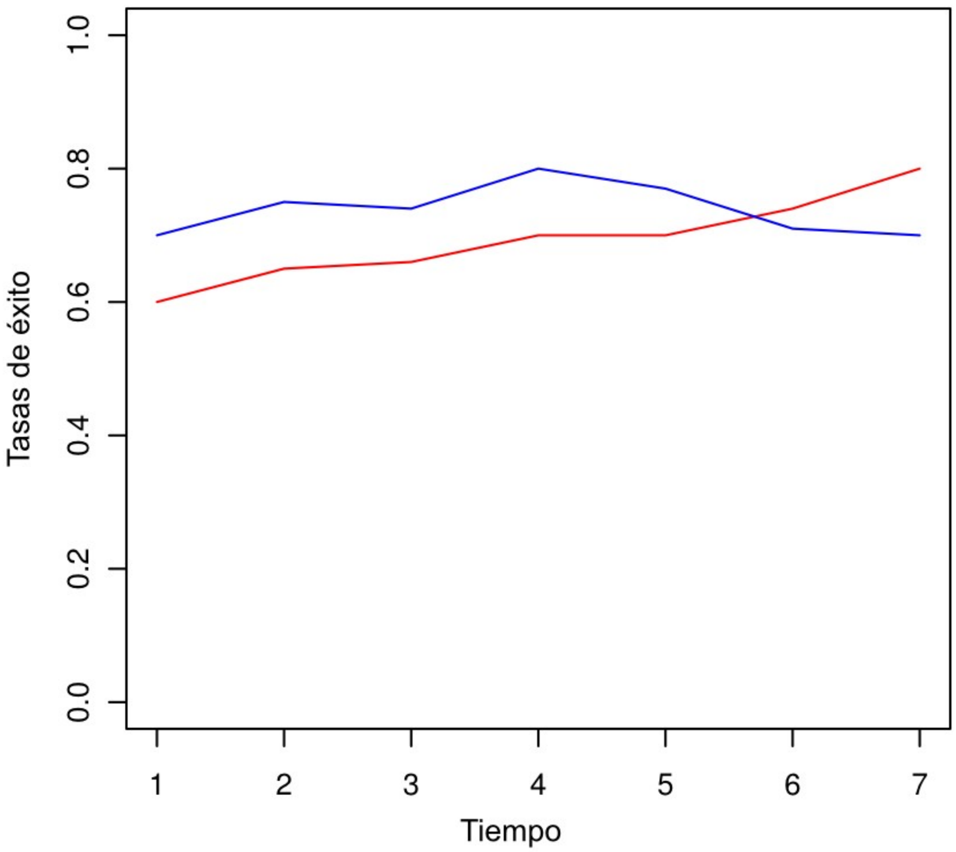
Advantages: All instances are used for training.
Useful for data streams with concept drifts.

Evaluation: comparing



Which method is better?

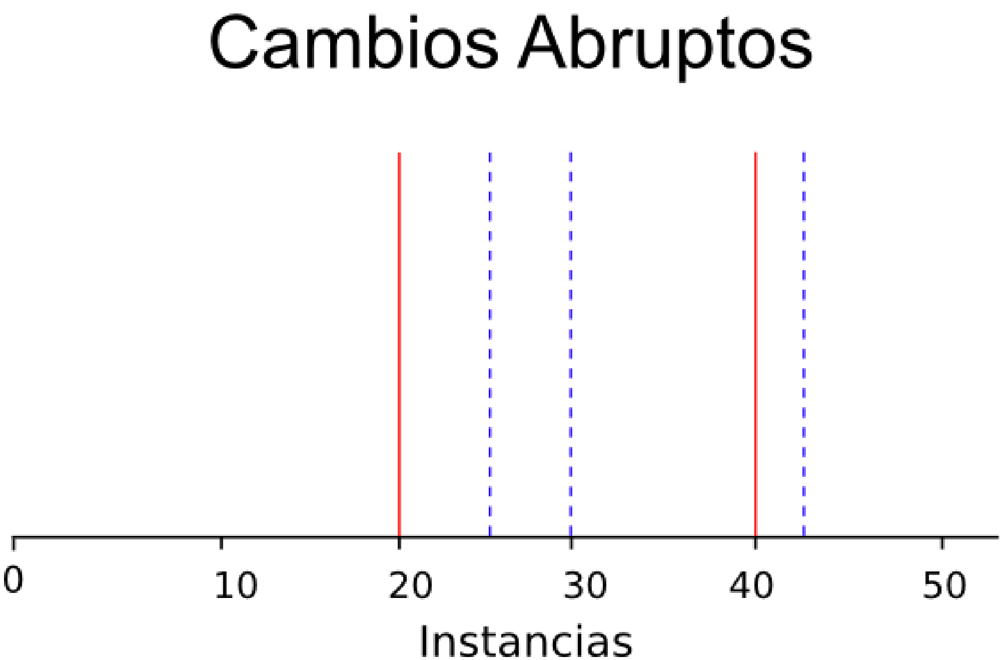
Evaluation: comparing

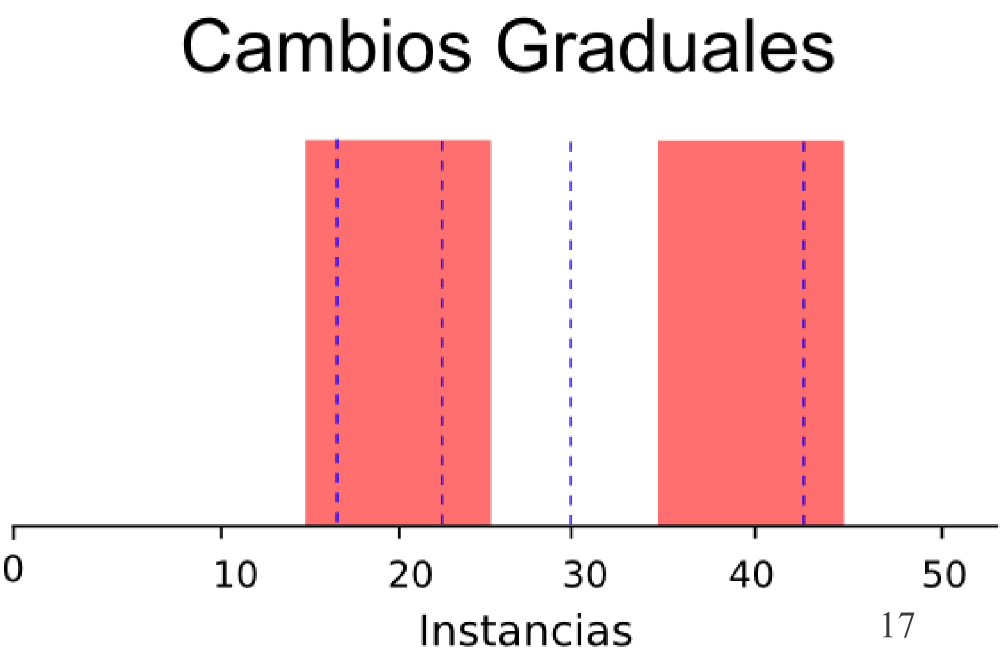


Which method is better? → AUC

Evaluation: drift detection

- First detected: correct.
- Following detected: false positives.
- Not detected: false negatives.
- Distance = correct – real.





Taxonomy of methods

Learners with triggers

- Change detectors
- Training windows
- Adaptive sampling

✔ *Advantages*: can be used by any classification algorithm.

✗ *Disadvantages*: usually, once detected a change, they discard the old model and relearn a new one.

Taxonomy of methods

Learners with triggers

- Change detectors
- Training windows
- Adaptive sampling

✔ *Advantages*: can be used by any classification algorithm.

✗ *Disadvantages*: usually, once detected a change, they discard the old model and relearn a new one.

Evolving Learners

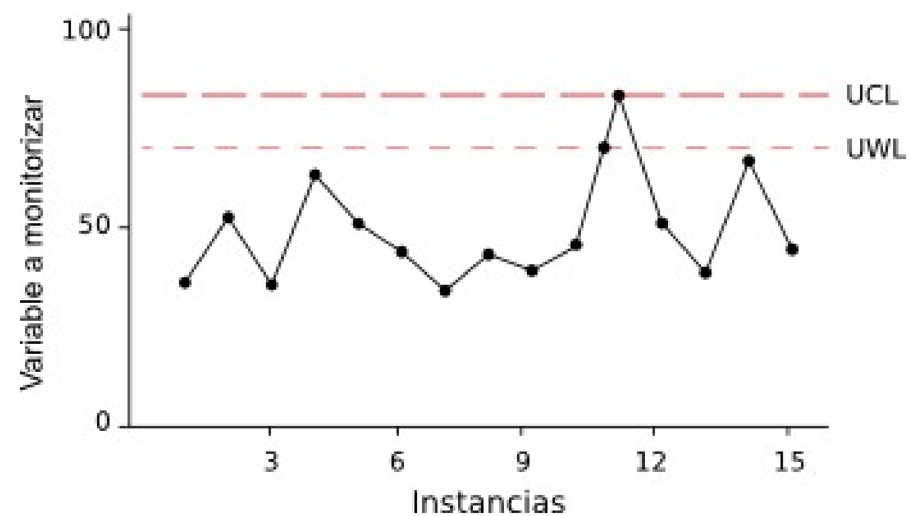
- Adaptive ensembles
- Instance weighting
- Feature space
- Base model specific

✔ *Advantages*: they continually adapt the model over time

✗ *Disadvantages*: they don't detect changes.

Contributions

- Taxonomy: triggers → change detectors
  - MoreErrorsMoving
  - MaxMoving
  - Moving Average
    - Heuristic 1
    - Heuristic 2
    - Hybrid heuristic: 1+2



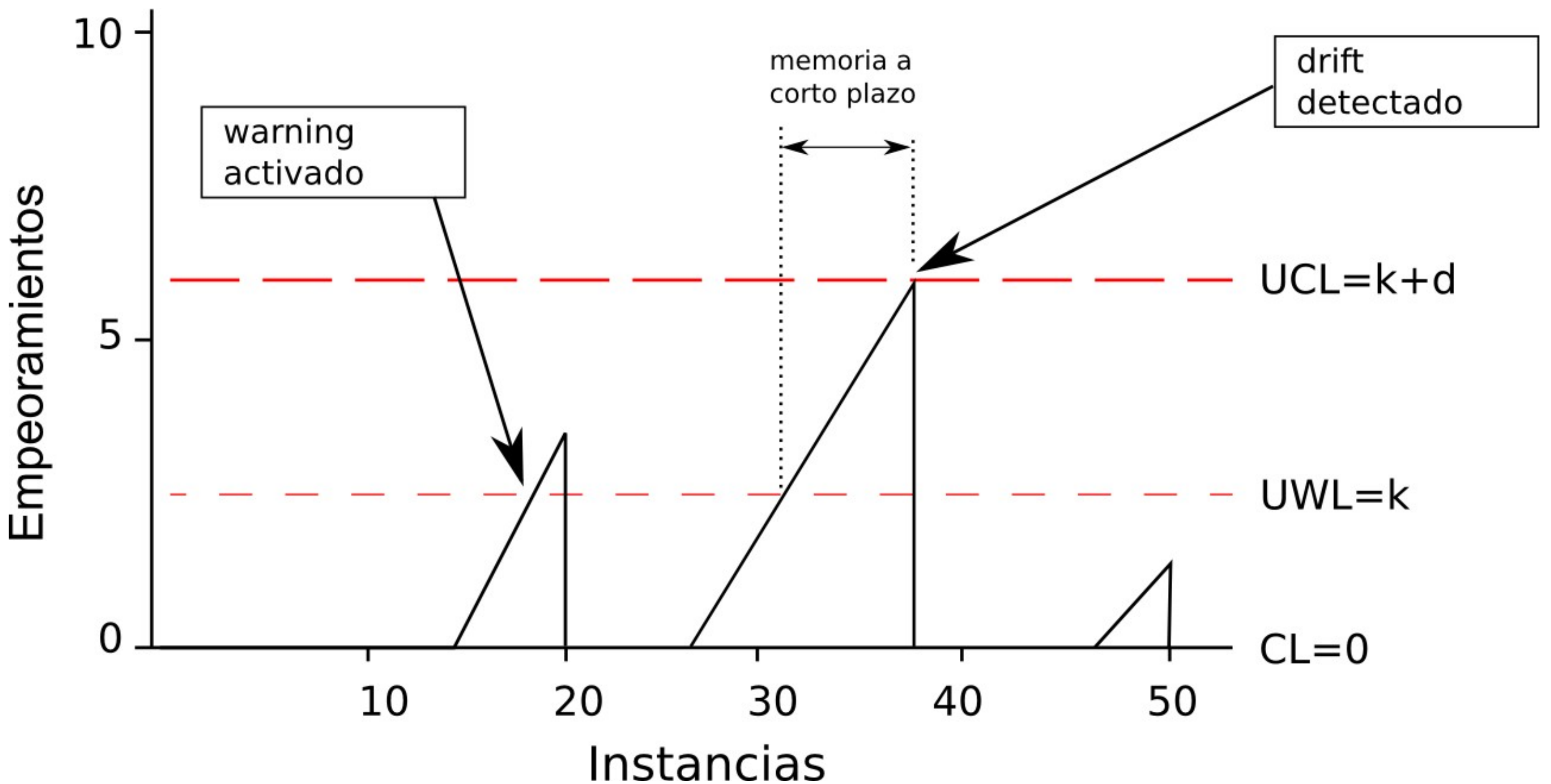
- P-chart with 3 levels: normal, warning and drift

Contributions: MoreErrorsMoving

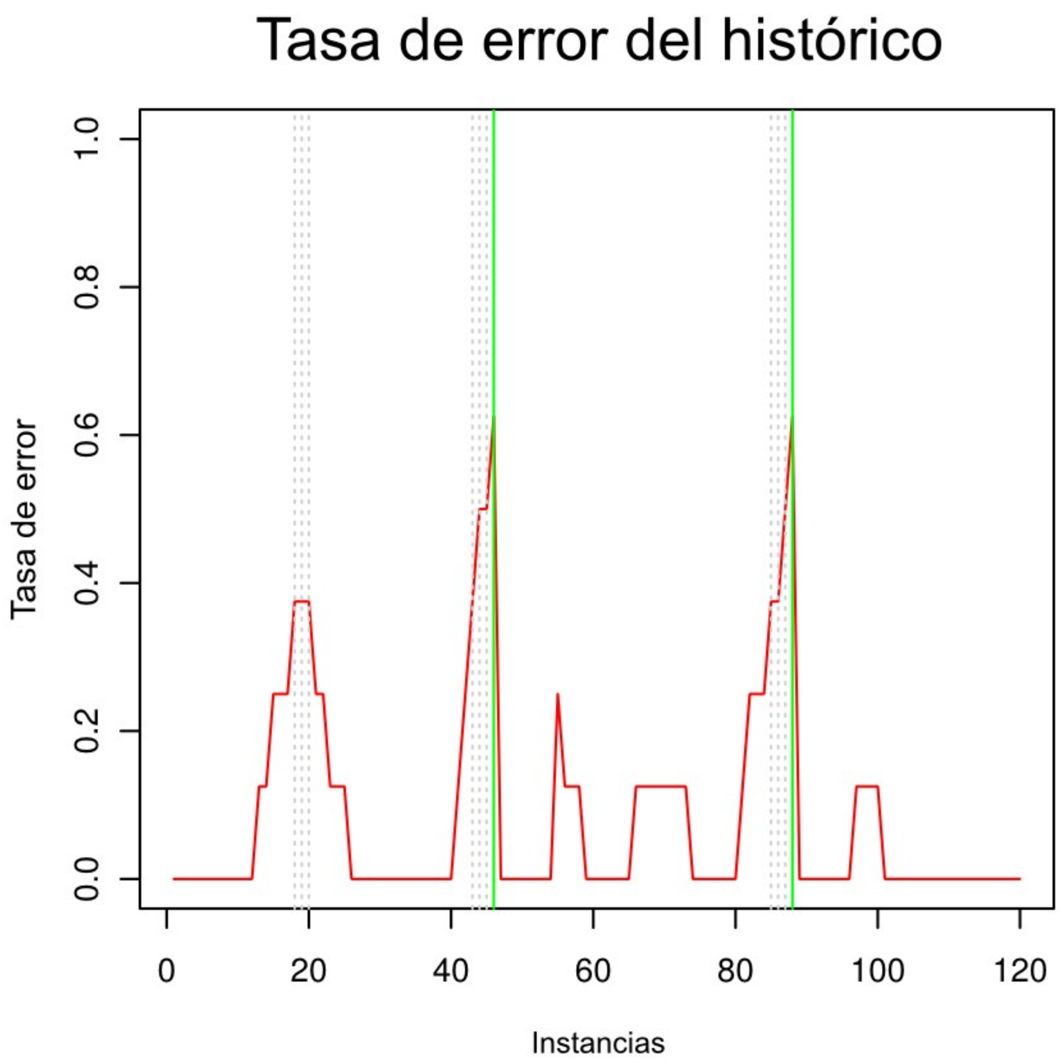
- *n* latest results of classification are monitored → History = {e_i, e_{i+1}, …, e_{i+n}} (i.e. 0,0,1,1)
- History error rate: $$c_i = \frac{\sum_{j=0}^{n} e_j}{n} | e_j \in H_i$$
- The consecutive declines are controlled
- At each time step:
  - If $c_{i-1} < c_i$ (more errors) → declines++
  - If $c_{i-1} > c_i$ (less errors) → declines=0
  - If $c_{i-1} = c_i$ (same) → declines don't change

Contributions: MoreErrorsMoving

- If consecutive declines > k → enable Warning
- If consecutive declines > k+d → enable Drift
- Otherwise → enable Normality



Contributions: MoreErrorsMoving



History = 8
Warning = 2
Drift = 4

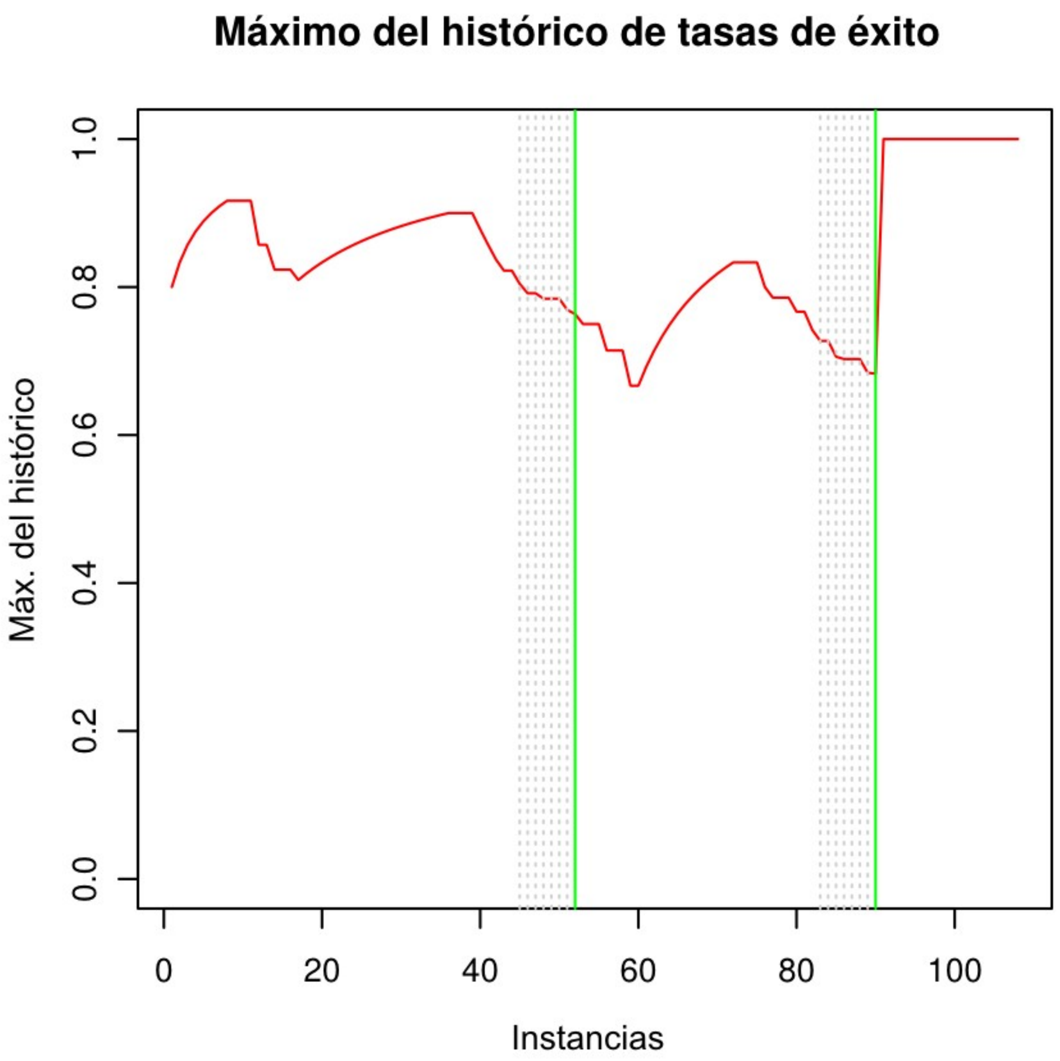
Detected drifts:
46 y 88

Distance to real drifts:
46-40 = 6
88-80 = 8

Contributions: MaxMoving

- *n* latest success accumulated rates are monitored since the last change
  - History={a_i , a_{i+1} , …, a_{i+n}} (i.e. H={2/5, 3/6, 4/7, 4/8})
- History maximum: $m_i = \max\{a_j | a_j \in H_i\}$
- The consecutive declines are controlled
- At each time step:
  - If $m_i < m_{i-1}$ → declines++
  - If $m_i > m_{i-1}$ → declines=0
  - If $m_i = m_{i-1}$ → declines don't change

Contributions: MaxMoving



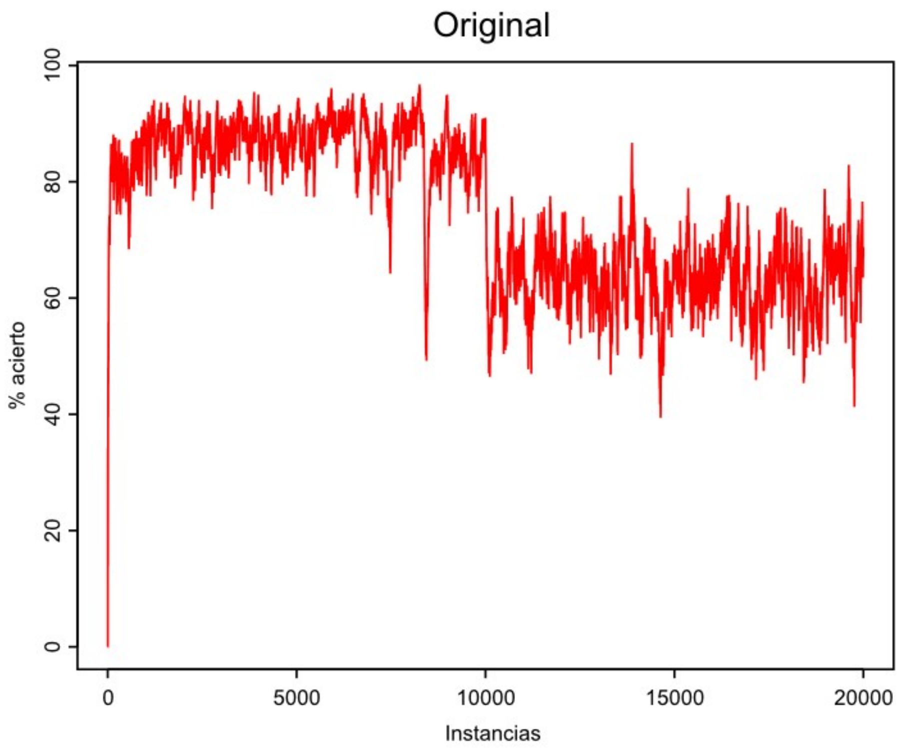
History = 4
Warning = 4
Drift = 8

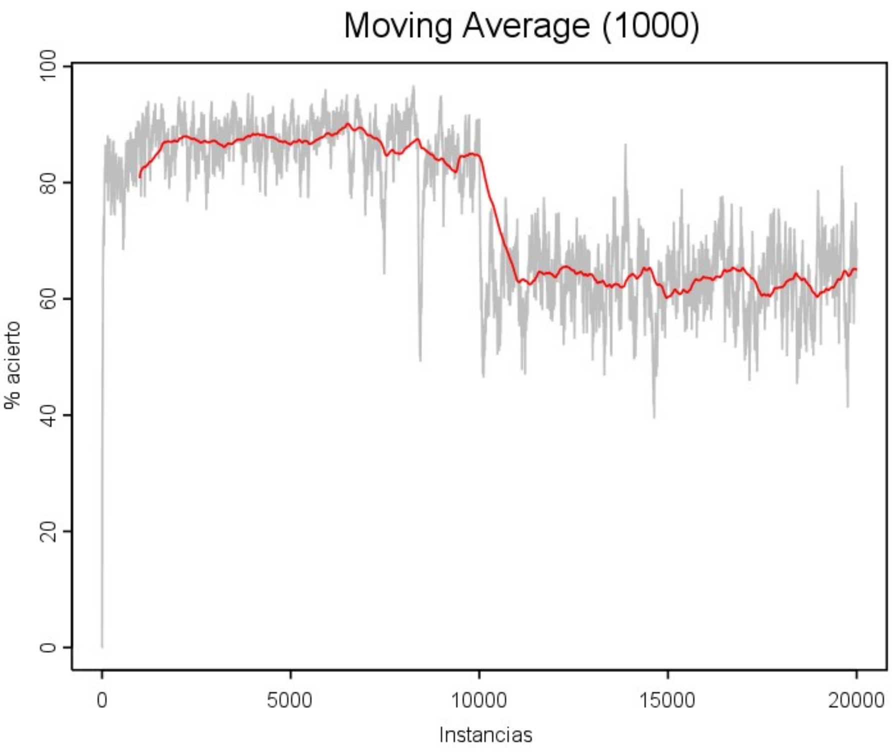
Detected drifts:
52 y 90

Distance to real drifts:
52-40 = 12
90-80 = 10

Contributions: Moving Average

Goal: to smooth accuracy rates for better detection.





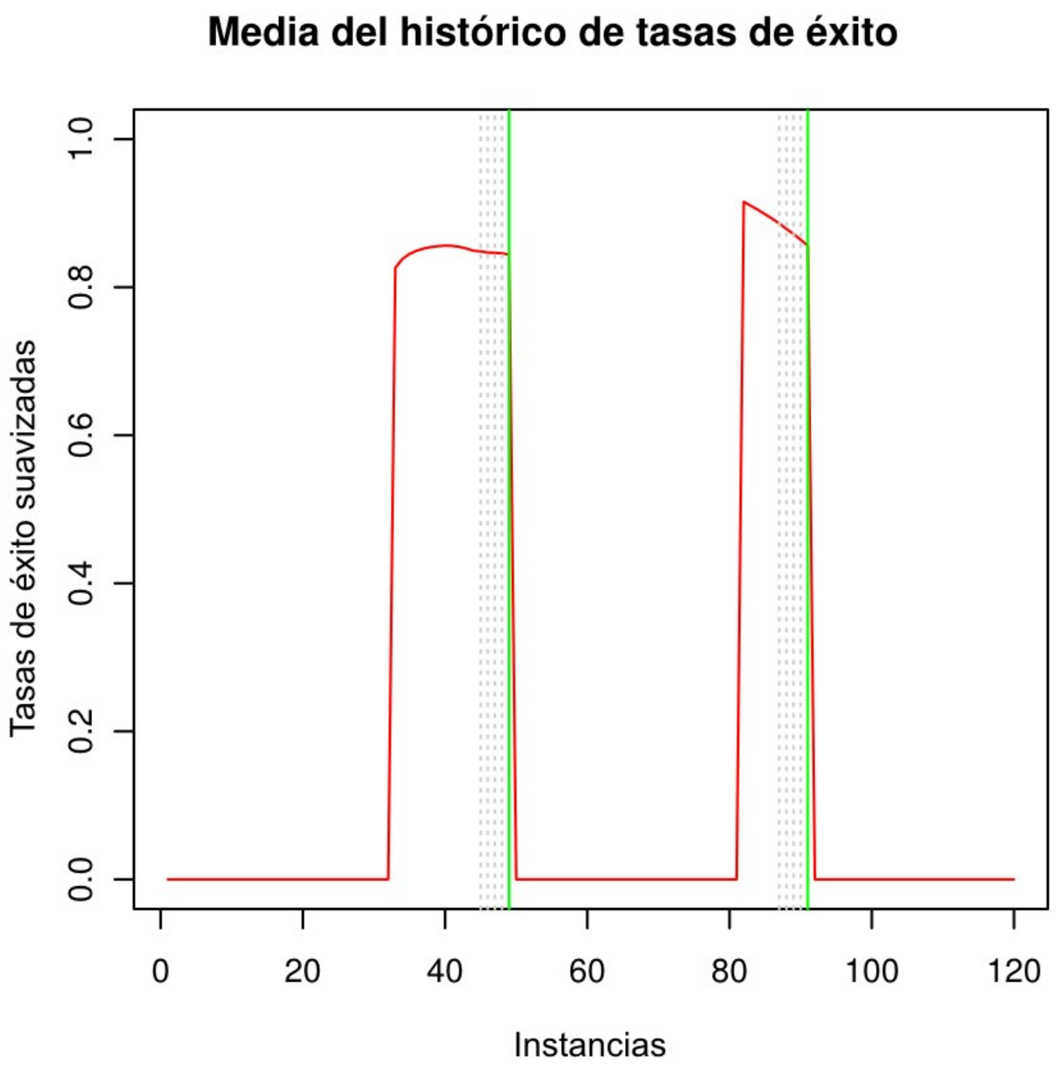
Contributions: Moving Average 1

- *m* latest success accumulated rates are smoothed → Simple moving average (unweighted mean)

$$s_t = \frac{1}{m}\sum_{n=0}^{m-1} x_{t-n} = \frac{x_t + x_{t-1} + x_{t-2} + \cdots + x_{t-(m-1)}}{m}$$

- The consecutive declines are controlled
- At each time step:
  - If $s_t < s_{t-1}$ → declines++
  - If $s_t > s_{t-1}$ → declines = 0
  - If $s_t = s_{t-1}$ → declines don't change

Contributions: Moving Average 1



Smooth = 32
Warning = 4
Drift = 8

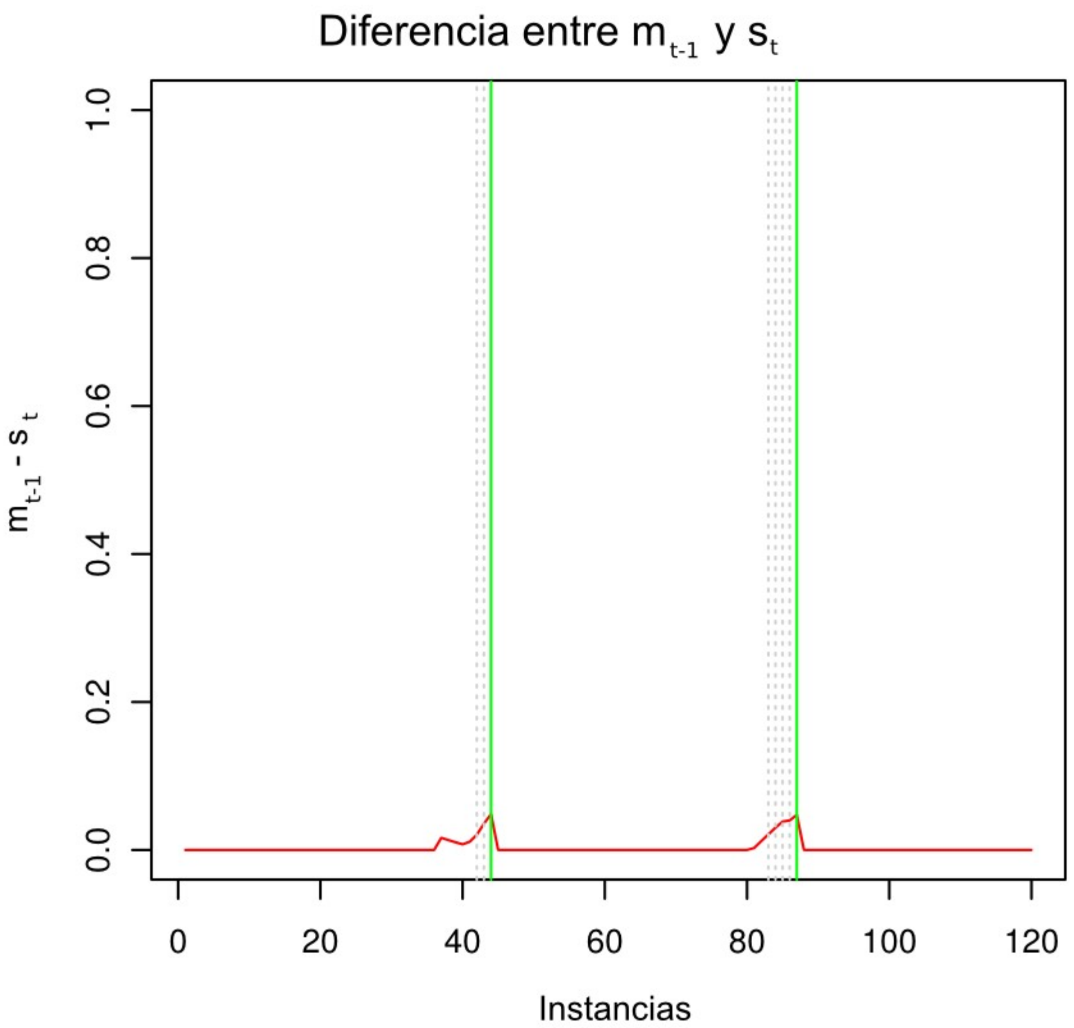
Detected
drifts:
49 y 91

Distance to real drifts:
49-40 = 9
91-80 = 11

Contributions: Moving Average 2

- History of size *n* with the smoothed success rates → History={s_i, s_{i+1}, …, s_{i+n}}
- History maximum: $m_i = \max\{s_j | s_j \in H_i\}$
- Difference between s_t and m_{t-1} is monitored
- At each time step:
  - If $m_{t-1} - s_t > u$ → enable Warning
  - If $m_{t-1} - s_t > v$ → enable Drift
  - Otherwise → enable Normality
- Suitable for abrupt changes

Contributions: Moving Average 2



Smooth = 4
History = 32
Warning = 2%
Drift = 4%

Detected drifts:
44 y 87

Distance to real drifts:
44-40 = 4
87-80 = 7

Contributions: Moving Average Hybrid

- Heuristics 1 and 2 are combined:
  - If Warning_1 or Warning_2 → enable Warning
  - If Drift_1 or Drift_2 → enable Drift
  - Otherwise → enable Normality

MOA: Massive Online Analysis

- Framework for data stream mining. Algorithms for classification, regression and clustering.
- University of Waikato → WEKA integration.
- Graphical user interface and command line.
- Data stream generators.
- Evaluation methods (holdout and prequential).
- Open source and free.

http://moa.cs.waikato.ac.nz

Experimentation

- Our data streams:
  - 5 synthetic with abrupt changes
  - 2 synthetic with gradual changes
  - 1 synthetic with noise
  - 3 with real data

Experimentation

- Our data streams:
  - 5 synthetic with abrupt changes
  - 2 synthetic with gradual changes
  - 1 synthetic with noise
  - 3 with real data
- Classification algorithm: Naive Bayes

Experimentation

- Our data streams:
  - 5 synthetic with abrupt changes
  - 2 synthetic with gradual changes
  - 1 synthetic with noise
  - 3 with real data
- Classification algorithm: Naive Bayes
- Detection methods:

No detection	MovingAverage1
MoreErrorsMoving	MovingAverage2
MaxMoving	MovingAverageH
DDM	EDDM

Experimentation

- Parameters tuning:
    - 4 streams y 5 methods → 288 experiments

Experimentation

- Parameters tuning:
  - 4 streams y 5 methods → 288 experiments
- Comparative study:
  - 11 streams y 8+1 methods → 99 experiments

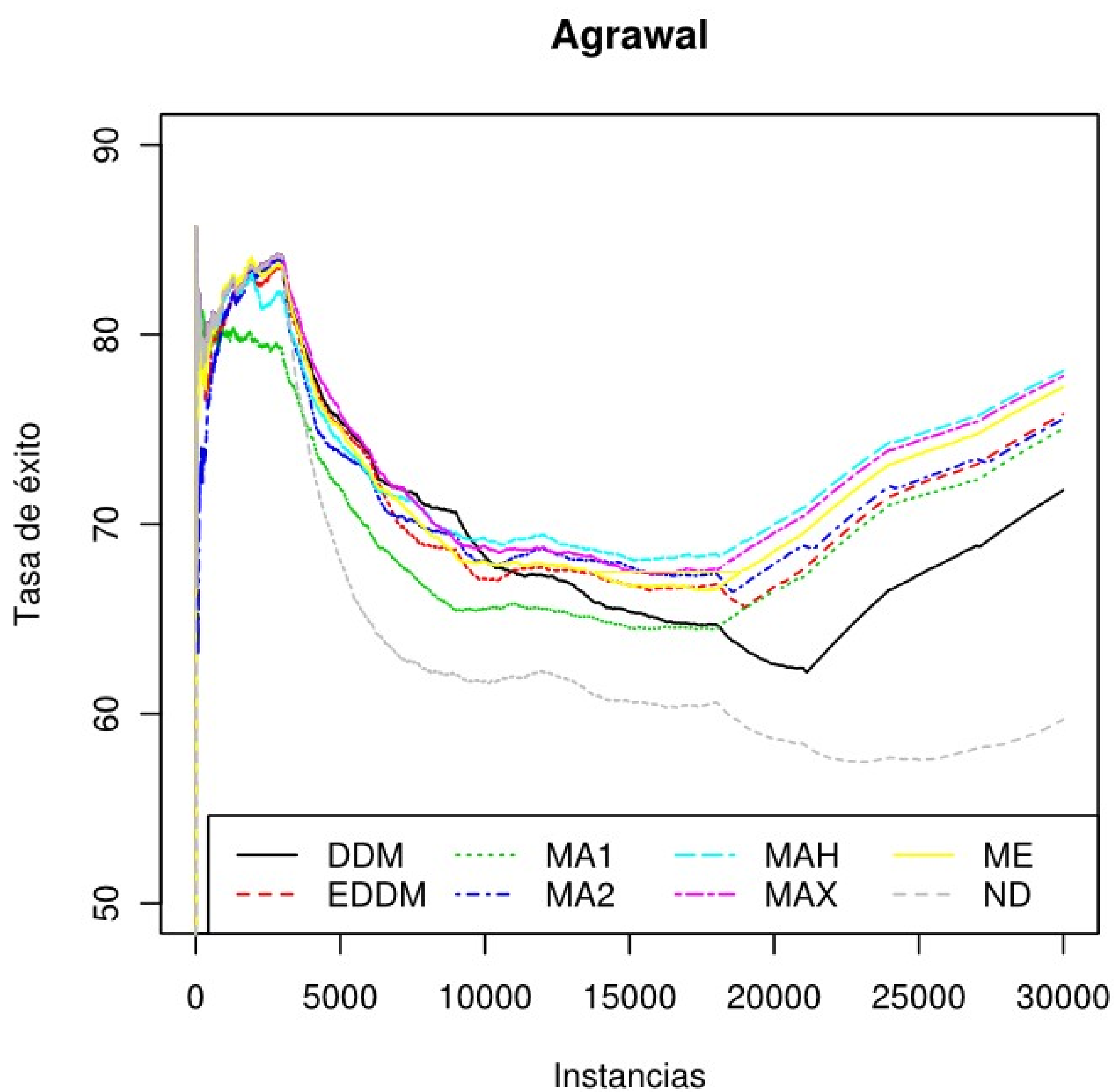
Experimentation

- Parameters tuning:
  - 4 streams y 5 methods → 288 experiments
- Comparative study:
  - 11 streams y 8+1 methods → 99 experiments
- Evaluation: prequential

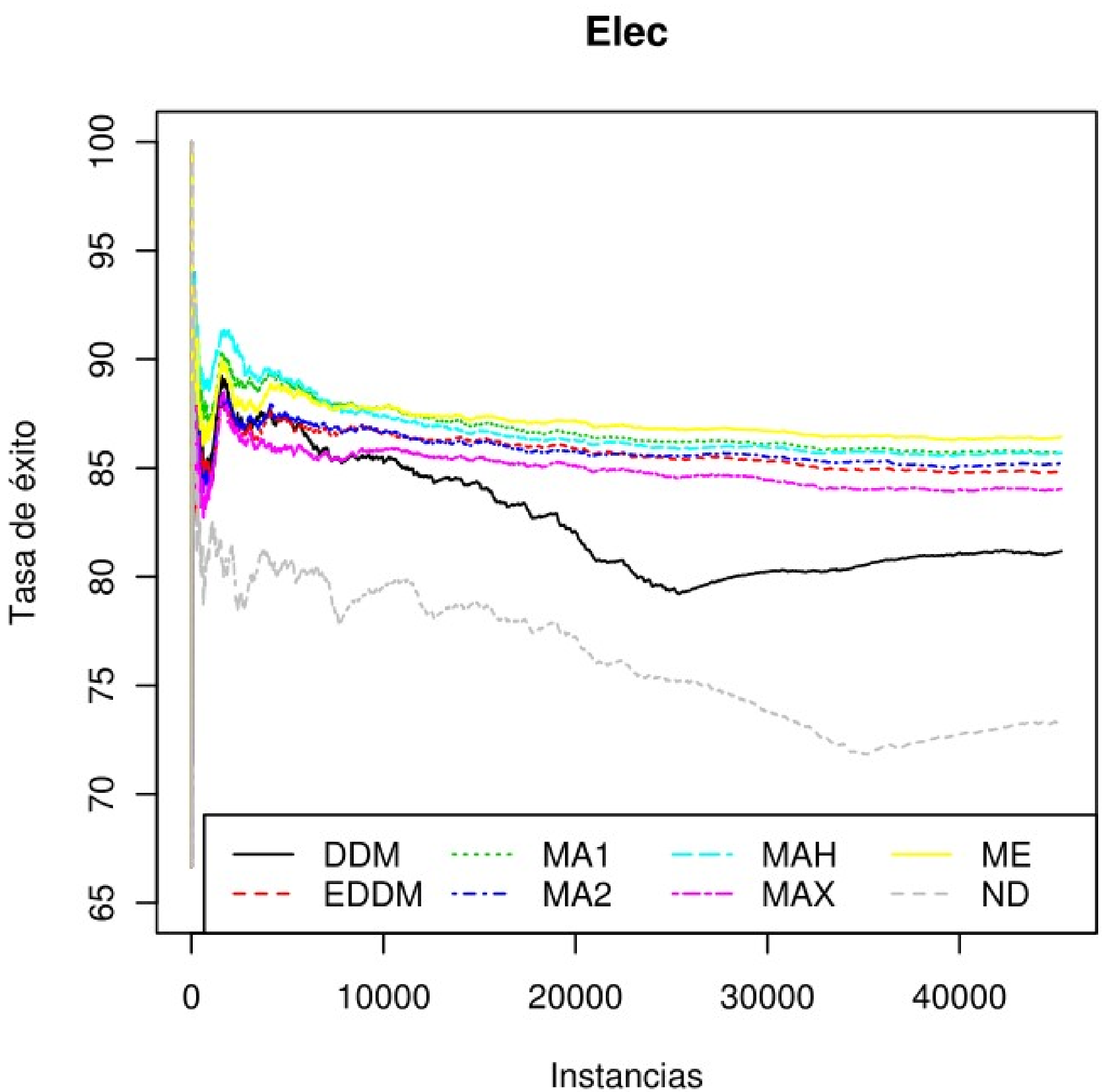
Experimentation

- Parameters tuning:
  - 4 streams y 5 methods → 288 experiments
- Comparative study:
  - 11 streams y 8+1 methods → 99 experiments
- Evaluation: prequential
- Measurements:
  - AUC: area under the curve of accumulated success rates
  - Number of correct drifts
  - Distance to drifts
  - False positives and false negatives

Experimentation: Agrawal



Experimentation: Electricity



Conclussions of experimentation

1. **With abrupt changes:**
   - More victories: DDM and MovingAverageH
   - Best in mean: MoreErrorsMoving → very responsive
2. **With gradual changes:**
   - Best: DDM and EDDM
   - Problem: many false positives → parameter tunning only with abrupt changes
3. **With noise:**
   - Only winner: DDM
   - Problem: noise sensitive → parameter tunning only with no-noise data
4. **Real data:**
   - Best: MovingAverage1 and MovingAverageH

Conclussions of this work

1. Our methods are competitive, although sensitive to the parameters → Dynamic fit
2. Evaluation is not trivial → Standardization is needed
3. Large field of application in industry
4. Hot topic: last papers from 2011 + conferences

Future work

1. Dynamic adjustment of parameters.
2. Measuring the abruptness of change for:
    - Using differents forgetting mechanisms.
    - Setting the degree of change of the model.
3. Develop an incremental learning algorithm which allows partial changes of the model when a drift is detected.

Thank you