Overview on Learning from Data Streams Part II

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

Outline

Data Streams

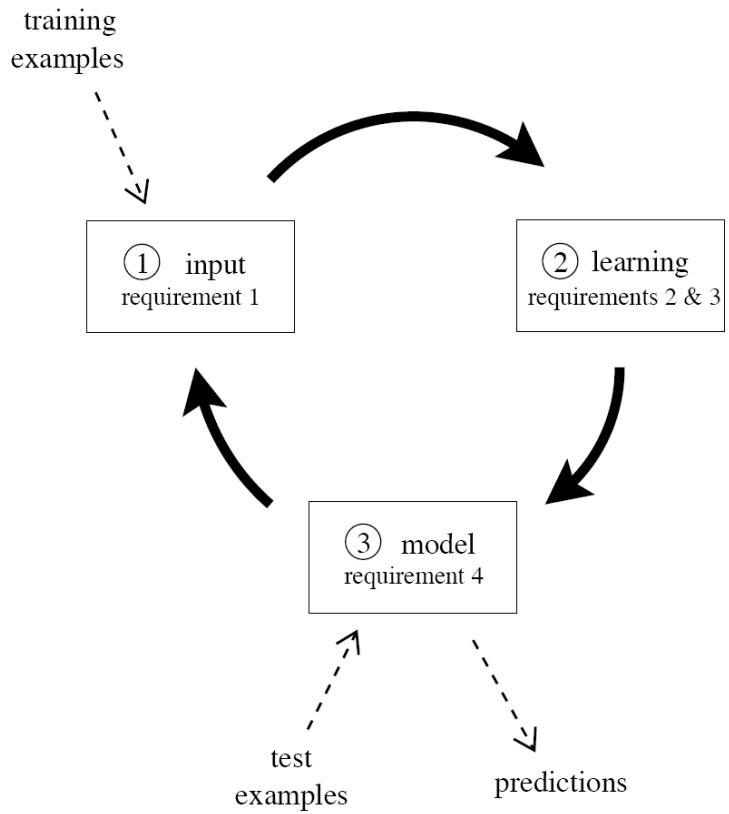
Data Streams: Continuous flow of data generated at **high-speed** in **dynamic**, **time-changing** environments.
We need to maintain **decision models** in **real time**.
Learning algorithms must be capable of:

1. **incorporating** new information at the speed data arrives;
2. **detecting** changes and **adapting** the decision models to the most recent information.
3. **forgetting** outdated information;

Unbounded training sets, dynamic models.
[Gama, 2010, Bifet et al., 2018]

Data Streams Computational Model

1. One example at a time, used at most once
2. Fixed memory
3. Limited processing time
4. Anytime prediction

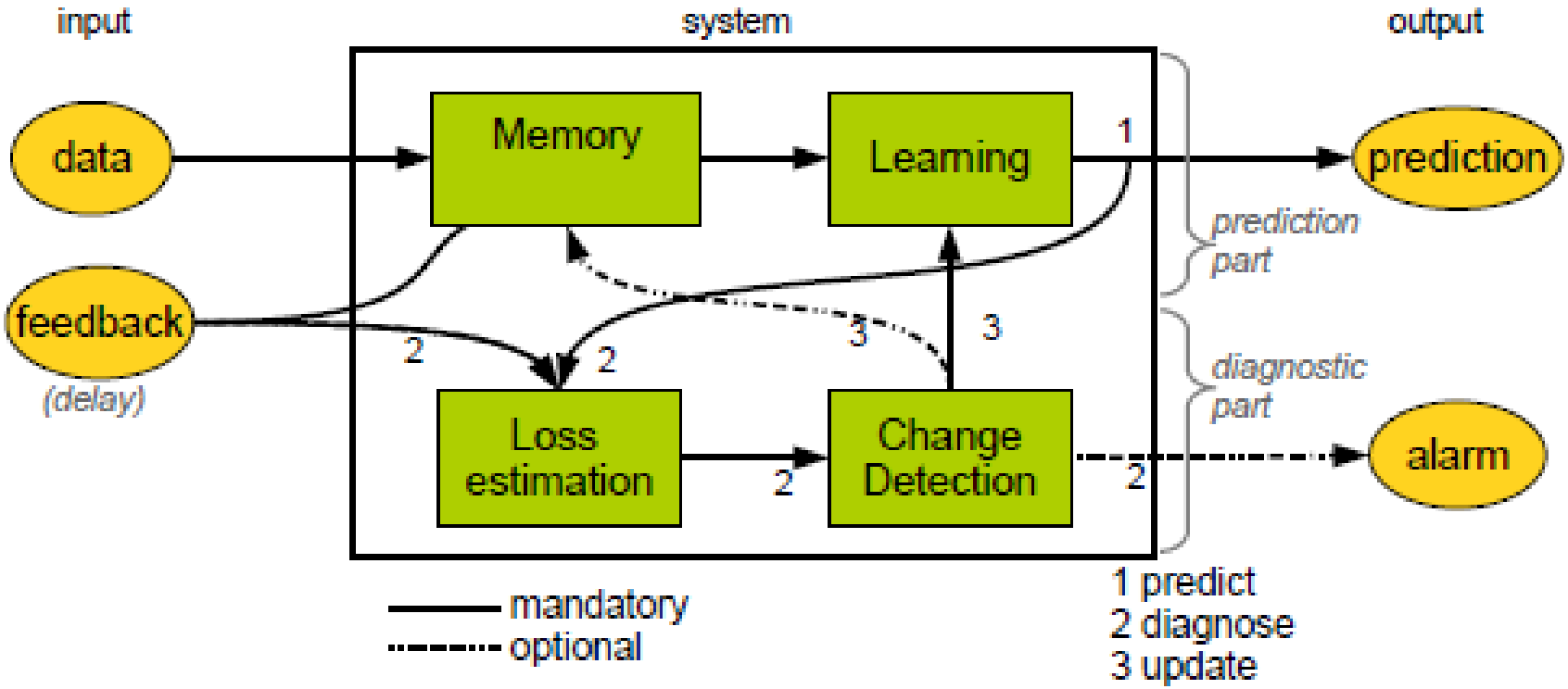


Powerful ideas

Powerful ideas

- Summarization:
  Compact and fast summaries to store sufficient statistics
- Approximation:
  How much information we need to learn an hypothesis $\hat{H}$ that is, with high probability, within small error of the true hypothesis ?
  $Pr(|H - \hat{H}| < \epsilon|H|) > 1 - \delta$
- Estimation:
  Useful for change detection

Adaptive Learning Algorithms



A generic schema for an online adaptive learning algorithm.

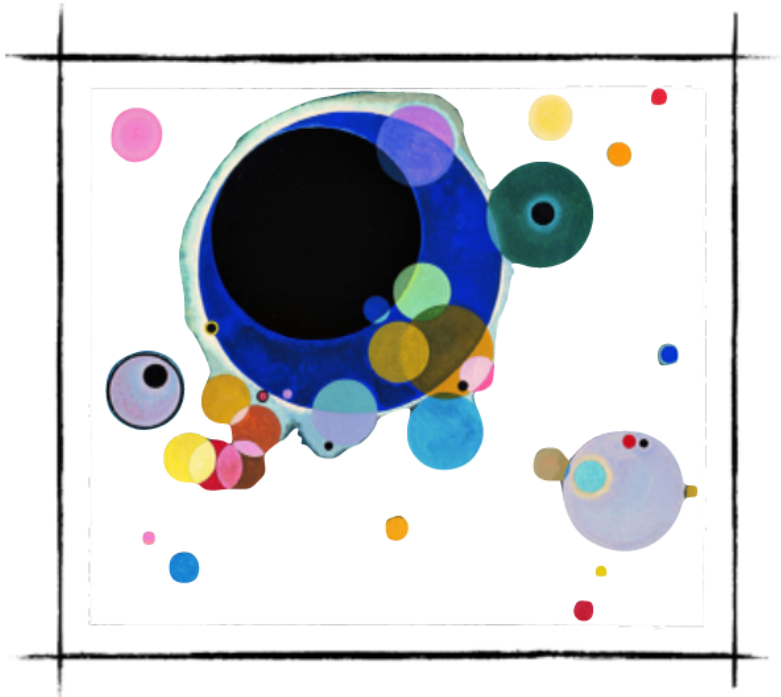
A survey on concept drift adaptation, Gama, Zliobaite, Bifet et al, ACM-CSUR 2014

Outline

Clustering

Clustering people or things into groups based on their attributes

- Costumers segmentation
- Social network communities



Major Clustering Approaches

- **Partitioning algorithms**: Construct various partitions and then evaluate them by some criterion
  - E.g., k-means, k-medoids, etc.
- **Hierarchy algorithms**: Create a hierarchical decomposition of the set of data (or objects) using some criterion.
  - Often needs to integrate with other clustering methods, e.g., BIRCH
- **Density-based**: based on connectivity and density functions
  - Finding clusters of arbitrary shapes, e.g., DBSCAN, OPTICS, etc.
- **Grid-based**: based on a multiple-level granularity structure
  - View space as grid structures, e.g., STING, CLIQUE
- **Model-based**: find the best fit of the model to all the clusters
  - Good for conceptual clustering, e.g., COBWEB, SOM

The Sequential k-Means

MacQueen, *Methods for Classification Multivariate data*, 1967

Input:

- X: A Sequence of Examples x_i
- k: Number of groups.

Output

- Centroids of the k Clusters

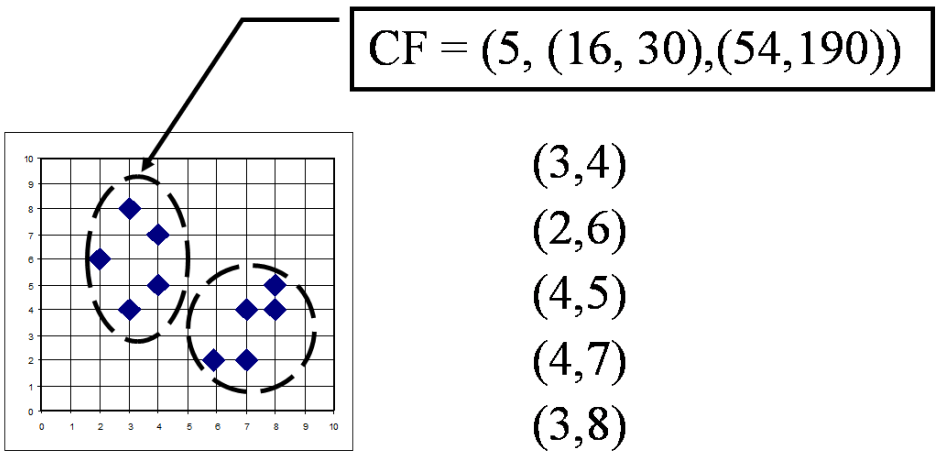
1. Initialize the set of centroids C_k with the first k observations $C_k = \{x_1, \ldots, x_k\}$
2. $n_1, \ldots, n_k = 1$
3. ForEach$(x_i \in X)$
   - Find the cluster C_j whose center is close to x_i
   - $n_j = n_j + 1$
   - $C_j = C_j + (x_i - C_j)/n_j$

Cluster Feature Vector

Birch: Balanced Iterative Reducing and Clustering using Hierarchies, by Zhang, Ramakrishnan, Livny 1996

Cluster Feature Vector: $\text{CF} = (N, LS, SS)$

- N: Number of data points
- LS: $\sum_1^N \vec{x_i}$
- SS: $\sum_1^N (\vec{x_i})^2$



Constant space irrespective to the number of examples!

Micro clusters

The sufficient statistics of a cluster A are $CF_A = (N, LS, SS)$.

- N: the number of data objects,
- LS: the linear sum of the data objects,
- SS: the sum of squared the data objects.

Properties:

- Centroid, defined as the gravity center of the cluster:
  $= LS/N$
- Radius, defined as the average distance from member points to the centroid:
  $= \sqrt{SS/N - (LS/N)^2}$
- Diameter, defined as the largest distance between member points:
  $= \sqrt{\frac{2 \times N * SS - 2 \times LS^2}{N \times (N-1)}}$

Micro clusters

Given two micro-clusters CF_a and CF_b, a CF entry has sufficient information to calculate the norms:

$$L_1 = \sum_{i=1}^{n} |LS_{a_i} - LS_{b_i}|$$

and

$$L_2 = \sqrt{\sum_{i=1}^{n} (LS_{a_i} - LS_{b_i})^2}$$

Micro clusters

An Efficient Data Clustering Method for Very Large Databases, SIGMOD 1996, T Zhang, R Ramakrishnan, M Livny

Given the sufficient statistics of a cluster A, $CF_A = (N_A, LS_A, SS_A)$. Updates are:

- Incremental: a point x is added to the cluster:
  $LS_A \leftarrow LS_A + x$; $SS_A \leftarrow SS_A + x^2$; $N_A \leftarrow N_A + 1$
- Additive: merging clusters A and B:
  $LS_C \leftarrow LS_A + LS_B$; $SS_C \leftarrow SS_A + SS_B$; $N_C \leftarrow N_A + N_B$
- Subtractive:
  $CF(C_1 - C_2) = CF(C_1) - CF(C_2)$

CluStream

CluStream: A Framework for Clustering Evolving Data Streams, Aggarwal, J. Han, J. Wang, P. Yu (VLDB03)

- Divide the clustering process into online and offline components
  - Online: periodically stores summary statistics about the stream data
    - Micro-clustering: better quality than k-means
    - Incremental, online processing and maintenance
  - Offline: answers various user queries based on the stored summary statistics
    - Determine k macro-clusters on demand
    - Time-horizon queries via pyramidal snapshot mechanism
- With limited overhead to achieve high efficiency, scalability, quality of results and power of evolution/change detection

CluStream: Online Phase

Inputs:

- Maximum micro-cluster diameter D_{max}

For each x in the stream:

- Find the nearest micro-cluster M_i
  - IF the diameter of $(M_i \cup x) < D_{max}$
  - THEN assign x to that micro-cluster $M_i \leftarrow M_i \cup x$
  - ELSE Start a new micro-cluster based on x

Any Time Stream Clustering

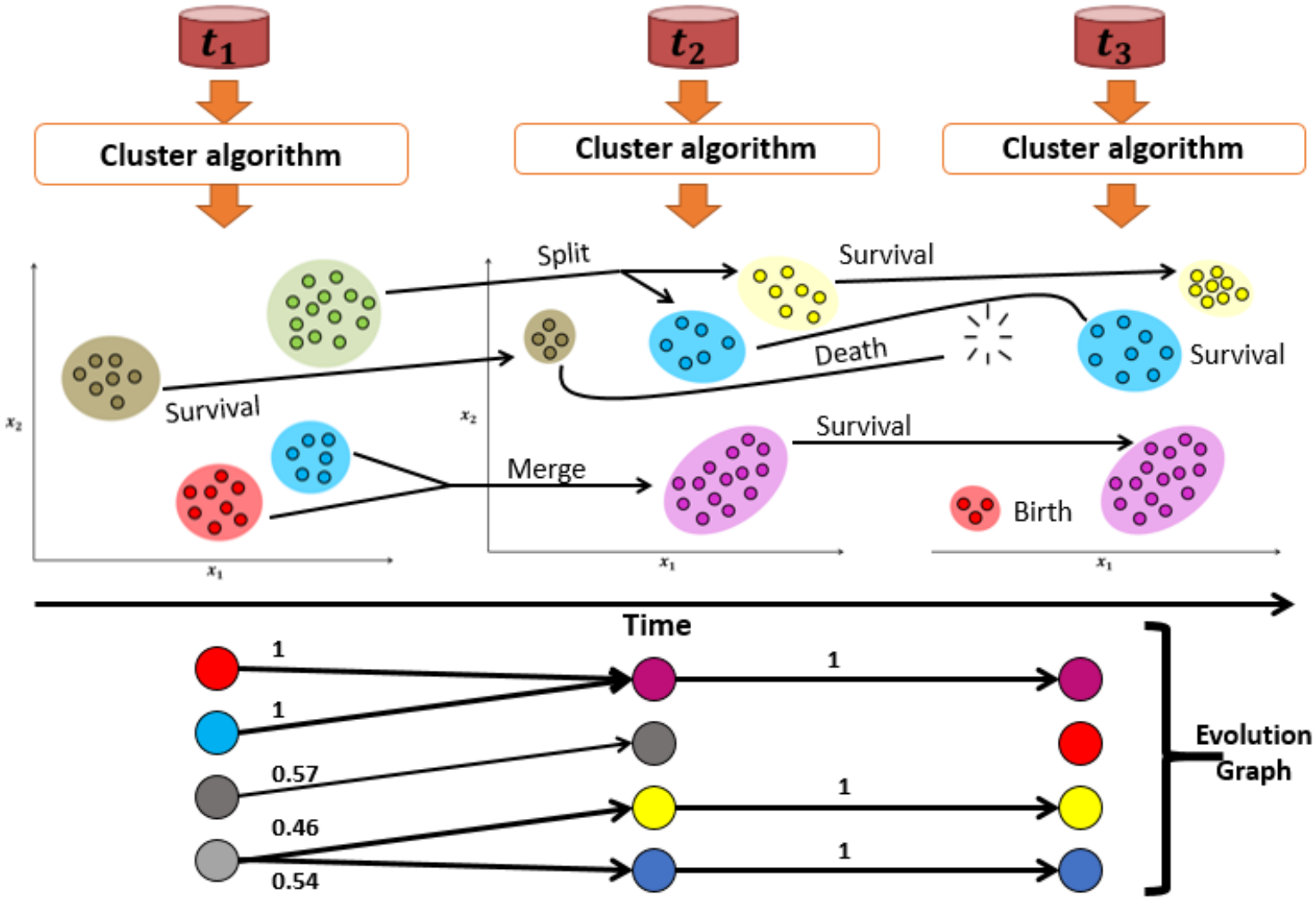
The ClusTree: indexing micro-clusters for anytime stream mining, Kranen, Assent, Baldauf, Seidl, KAIS 2011

Properties of anytime algorithms

- Deliver a model at any time
- Improve the model if more time is available
  - Model adaptation whenever an instance arrives
  - Model refinement whenever time permits
- an online component to learn micro-clusters
- Any variety of online components can be utilized
- Micro-clusters are subject to exponential aging

Cluster Evolution

M.Oliveira, J.Gama, *A framework to monitor clusters evolution applied to economy and finance problems* Intell. Data Anal. 2012



Analysis

Time-sensitive Queries:

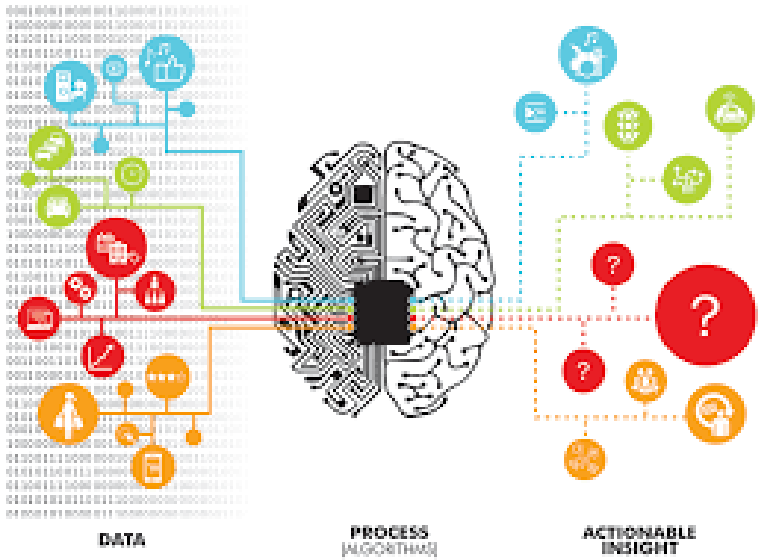
- Find the current decision structure;
- What changed in the decision structure last week?
- Which patterns disappeared / appeared last week?
- Which patterns are growing / shrinking this month?
- Mine the evolution of decision structures.

Outline

Predictive Learning

What will happen?

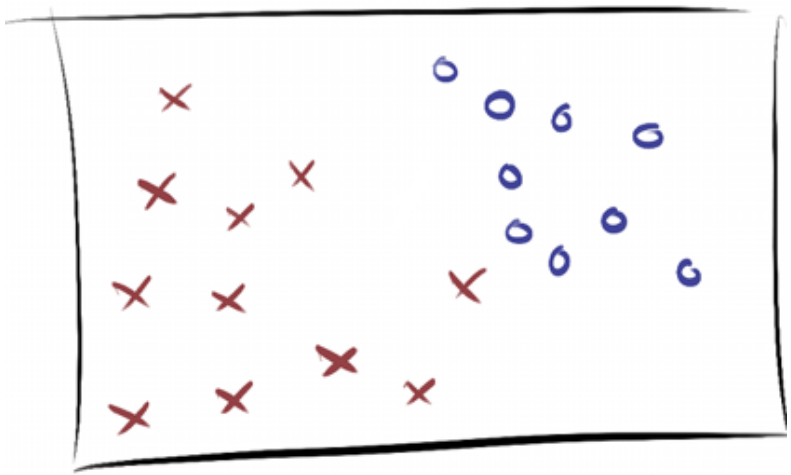
- Classification
- Regression
- Change Detection
- Evaluation



Classification

Classifying people or things into groups by recognizing patterns

- Email spam filter
- Twitter sentiment analyzer



Naive Bayes

- Based on Bayes theorem assuming attributes are independent given the class label
  - Prior class probability P(C)
  - Probability of observing feature x_i given class C
- One-pass algorithm: just counting!

$$posterior = \frac{likelihood \times prior}{evidence}$$

- Generative approach
- $P(C_k|\mathrm{x}) \propto P(C_k) \prod_i P(x_i|C_k)$
- $C = argmax_k P(C_k|\mathrm{x})$

Decision Trees

- Divide and Conquer
  - Each node tests a feature
  - Each branch represents a possible value of that feature
  - Each leaf assigns a class
- Greedy recursive induction
  - Sort all examples through the tree
  - x_i = most discriminative attribute
  - New node for x_i, new branch for each value,
  - leaf assigns majority class

Learning Decision Trees from Streams

Mining High-Speed Data Streams, P. Domingos, G. Hulten; KDD 2000

The base Idea

- Which attribute to choose at each splitting node?
- A small sample can often be enough to choose the optimal splitting attribute
  - Collect sufficient statistics from a small set of examples
  - Estimate the merit of each attribute

How large should be the sample?

- **The wrong idea:** Fixed sized, defined *apriori* without looking for the data;
- **The right idea:** Choose the sample size that allow to differentiate between the alternatives.

Very Fast Decision Trees

The base Idea

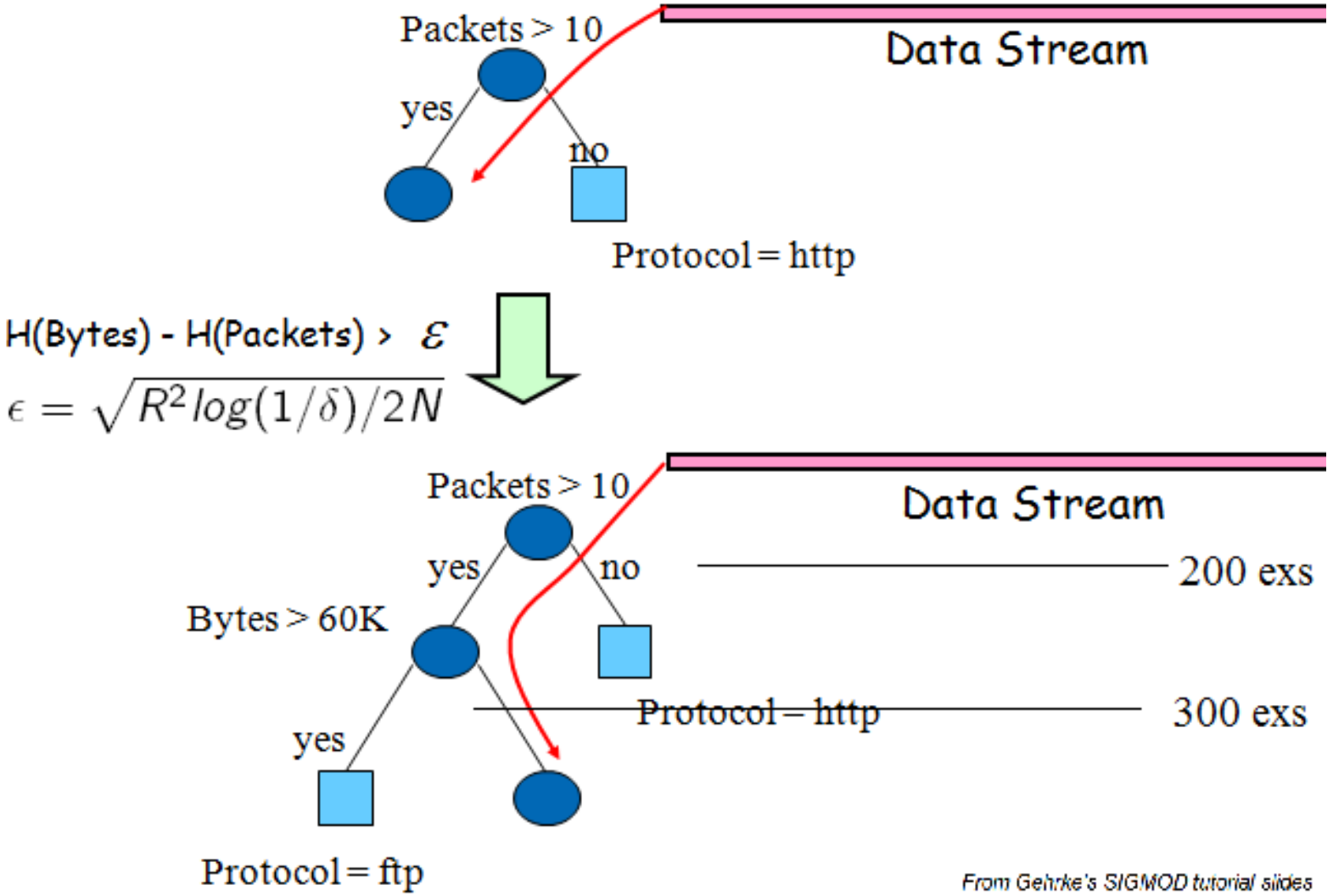
A small sample can often be enough to choose the optimal splitting attribute

- Collect sufficient statistics from a small set of examples
- Estimate the merit of each attribute
- Suppose that after seeing n examples, $G(X_a) > G(X_b) > ... > G(X_k)$
- Given a desired ϵ, the Hoeffding bound ensures that X_a is the correct choice, with probability $1 - \delta$, if $G(X_a) - G(X_b) > \epsilon$.
- If $G(X_a) - G(X_b) < \epsilon$, collect more examples

Hoeffding bound

- Suppose we have made n independent observations of a random variable r whose range is R.
  Let $\overline{r}$ be the mean computed in the sample.
- The Hoeffding bound states that:
  - With probability $1 - \delta$
  - The true mean of r is in the range $\overline{r} \pm \epsilon$ where $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}}$
- Independent of the probability distribution generating the examples.

VFDT



From Gehrke's SIGMOD tutorial slides

Concept-adapting VFDT

G. Hulten, L. Spencer, P. Domingos: *Mining Time-Changing Data Streams* KDD 2001

- Model consistent with sliding window on stream
- Keep sufficient statistics also at internal nodes
  - Recheck periodically if splits pass Hoeffding test
  - If test fails, grow alternate subtree and swap-in when accuracy of alternate is better
- Processing updates O(1), time +O(W) memory
  - Increase counters for incoming instance, decrease counters for instance going out window

Hoeffding Adaptive Tree

A. Bifet, R. Gavaldà: *Adaptive Parameter-free Learning from Evolving Data Streams*

IDA, 2009

- Replace frequency counters by estimators
  - No need for window of examples
  - Sufficient statistics kept by estimators separately
- Parameter-free change detector + estimator with theoretical guarantees for subtree swap (ADWIN)
  - Keeps sliding window consistent with the *no-change hypothesis*

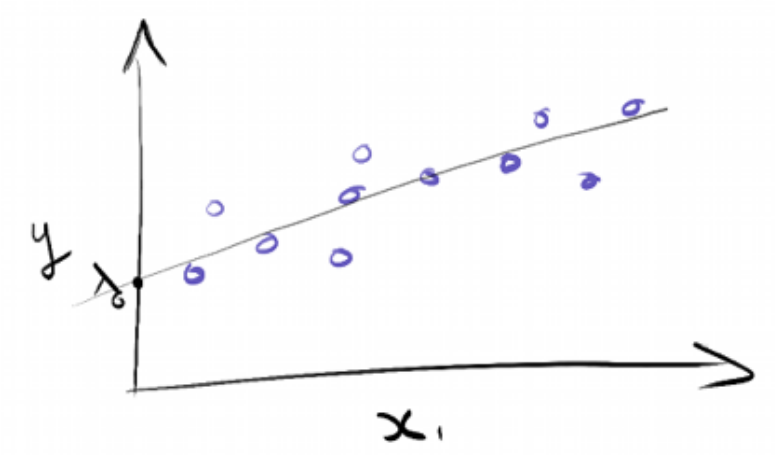
Regression

Relationship between a dependent continuous variable and one or more independent variables.

- Forecasting what may happen in the future

Applications:

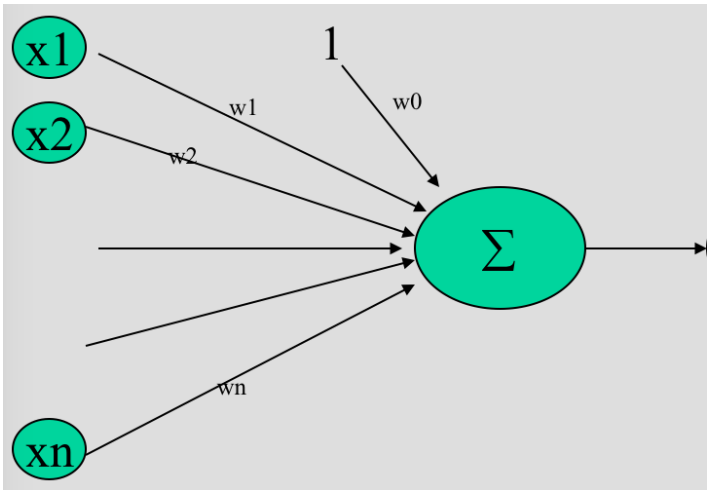
- Stocks Price
- Predicting electricity demand



$y = f(x)$

Perceptron

- Linear Regressor:
  $\hat{y} = w_0 + \sum w_i \times x_i$
- Goal:
  find the parameters w that minimize the MSE:
  $1/2 \sum (\hat{y} - y)^2$
- Using Stochastic Gradient Descent:
  $w_i(t+1) = w_i(t) + \eta(\hat{y} - y) \times x_i$



Regression Trees

- Same structure as decision trees
- Predict = average target value or linear model at leaf (vs majority)
- Gain = reduction in standard deviation (vs entropy)

$$\sigma(D) = \sqrt{\sum_{i \in D} (\bar{y} - y_i)^2 / (|D| - 1)}$$

$$Gain(Split) = \sigma(D) - \frac{|D_L|}{|D|}\sigma(D_L) - \frac{|D_R|}{|D|}\sigma(D_R)$$

Regression Trees from Streams

E.Ikonomovska, J.Gama, S.Dzeroski: Learning model trees from evolving data streams. Data Min. Knowl. Discov.2011

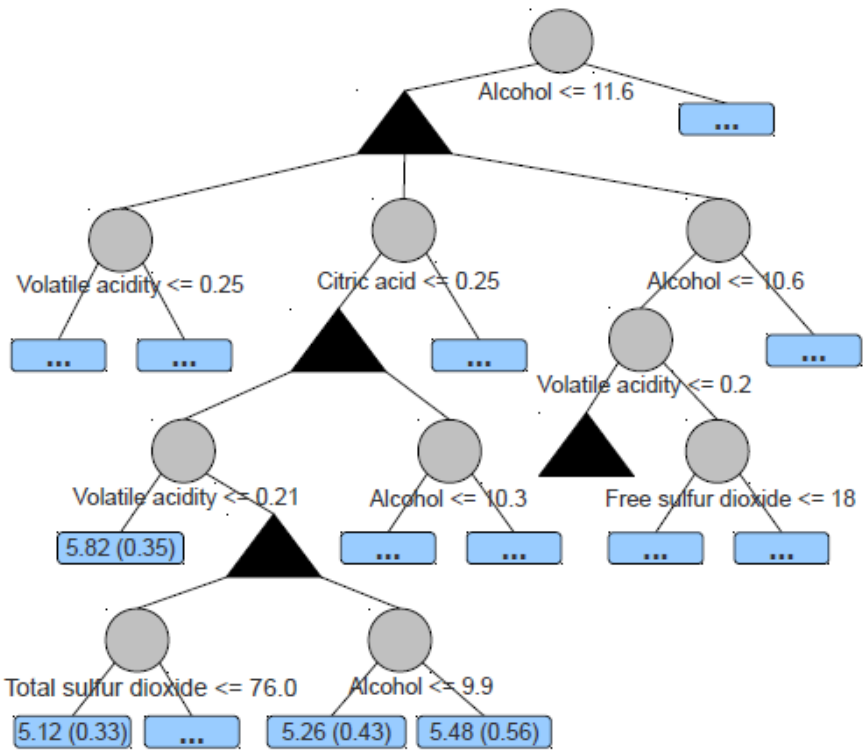
- Let s_1 be the split that most reduces variance,
  Let s_2 be the second best split:
- Is x_1 a stable option?
- Split if: $G(x_2)/G(x_1) < 1 - \epsilon = 1 - \sqrt{\frac{log(1/\delta)}{2 \times N}}$
  Statistical evidence that it is better than the second best

Option Trees

Speeding-Up Hoeffding-Based Regression Trees With Options, Ikonomovska, et al, ICML 2011

Options nodes: OR nodes to encode alternatives

Use option nodes to solve ties

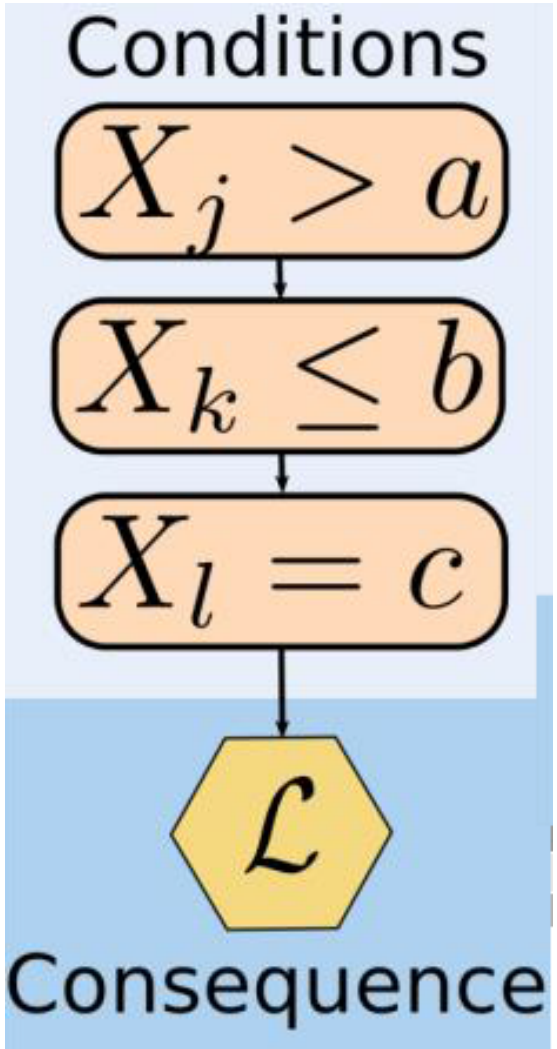


Decision and Regression Rules

Rules are one of the most expressive predictive models

- Rules are implications of the form Antecedent $\rightarrow$ Consequent
- Antecedent: conjunction of conditions
- Consequent $()\mathcal{L}$) keeps sufficient statistics to:
  make predictions
  expand the rule
  detect changes and anomalies

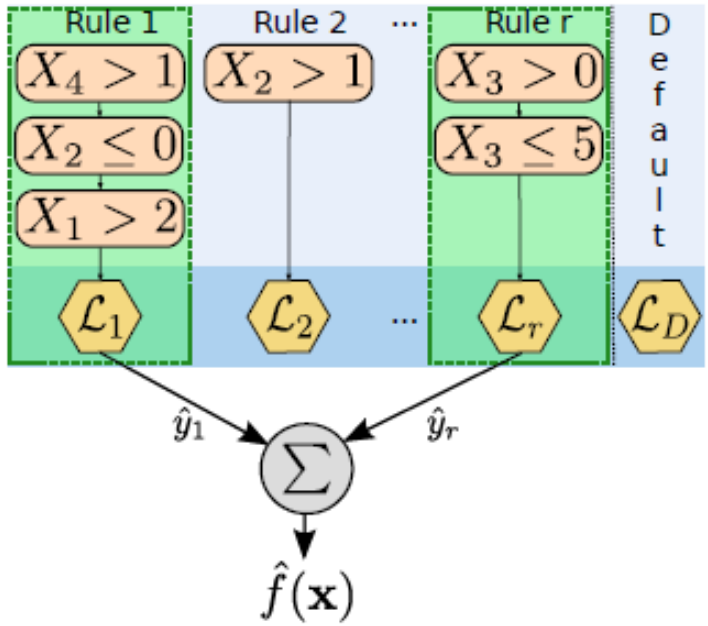
Rules are self-contained, modular, easier to interpret, no need to cover the universe



Adaptive Model Rules from Streams

Adaptive Model Rules from Data Streams, Almeida, Ferreira, Gama; ECML/PKDD 2013

- Ruleset: ensemble of rules
- Rule prediction: mean, linear model
- Ruleset prediction:
  - Ordered: only first rule covers instance
  - Unordered: weighted avg. of predictions of rules covering instance x
  - Weights inversely proportional to error



E.g: $\mathbf{x} = [4, -1, 1, 2]$

$$\hat{f}(\mathbf{x}) = \sum_{R_l \in S(\mathbf{x}_i)} \theta_l \hat{y}_l,$$

AMRules Induction

- Rule creation: default rule expansion
- Rule expansion: split on attribute maximizing σ reduction
  - Hoeffding bound $\epsilon = \sqrt{R^2 ln(1/\delta)/(2n)}$
  - Expand when $\sigma_{1st}/\sigma_{2nd} < 1 - \epsilon$
- Evict rule when P-H signals an alarm
- Detect and explain local anomalies

Algorithm 1: Training AMRules

Input: S: Stream of examples

begin
- $\mathcal{R} \leftarrow \{\}$, $D \leftarrow 0$
- **foreach** $(\mathbf{x}, y) \in S$ **do**
  - **foreach** *Rule* $r \in S(\mathbf{x})$ **do**
    - **if** $\neg$*IsAnomaly*$(\mathbf{x}, r)$ **then**
      - **if** *PHTest*$(error_r, \lambda)$ **then**
        - Remove the rule from $\mathcal{R}$
      - **else**
        - Update sufficient statistics $\mathcal{L}_r$
        - *ExpandRule*(r)
  - **if** $S(\mathbf{x}) = \emptyset$ **then**
    - Update $\mathcal{L}_D$
    - *ExpandRule*(D)
    - **if** *D expanded* **then**
      - $\mathcal{R} \leftarrow \mathcal{R} \cup D$
      - $D \leftarrow 0$
- **return** $(\mathcal{R}, \mathcal{L}_D)$

Hoeffding Algorithms

- Classification:
  Mining high-speed data streams, P. Domingos, G. Hulten, KDD, 2000
- Regression:
  Learning model trees from evolving data streams; Ikonomovska, Gama, Dzeroski; Data Min. Knowl. Discov. 2011
- Decision Rules:
  Learning Decision Rules from Data Streams, J. Gama, P. Kosina; IJCAI 2011
- Regression Rules
  E. Almeida, C. Ferreira, J. Gama: Adaptive Model Rules from Data Streams. ECML/PKDD 2013
- Clustering:
  Hierarchical Clustering of Time-Series Data Streams. Rodrigues, Gama, IEEE TKDE 20(5): 615-627 (2008)
- Multiple Models:
  Ensembles of Restricted Hoeffding Trees. Bifet, Frank, Holmes, Pfahringer; ACM TIST; 2012
  J. Duarte, J. Gama, Ensembles of Adaptive Model Rules from High-Speed Data Streams. BigMine 2014.
- ...

Hoeffding Algorithms: Analysis

The number of examples required to expand a node only depends on the Hoeffding bound: ϵ decreases with $\sqrt{N}$.

- Low variance models:
  Stable decisions with statistical support.
- Low overfiting:
  Examples are processed only once.
- No need for pruning;
  Decisions with statistical support;
- **Convergence**: Hoeffding Algorithms becomes asymptotically close to that of a batch learner. The expected disagreement is δ/p; where p is the probability that an example fall into a leaf.

Concept Drift

Detecting Changes in the process generating data

- Signaling Alarms
- Detecting Faults, Anomalies

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.

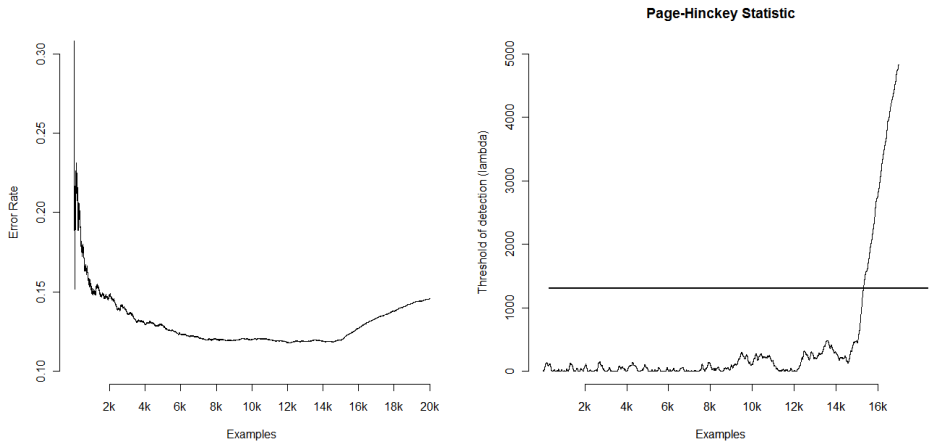
The Page-Hinckley Test

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

- 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.



The left figure plots the on-line error rate of a learning algorithm. The right plot presents the evolution of the *PH* statistic.

Concept Drift

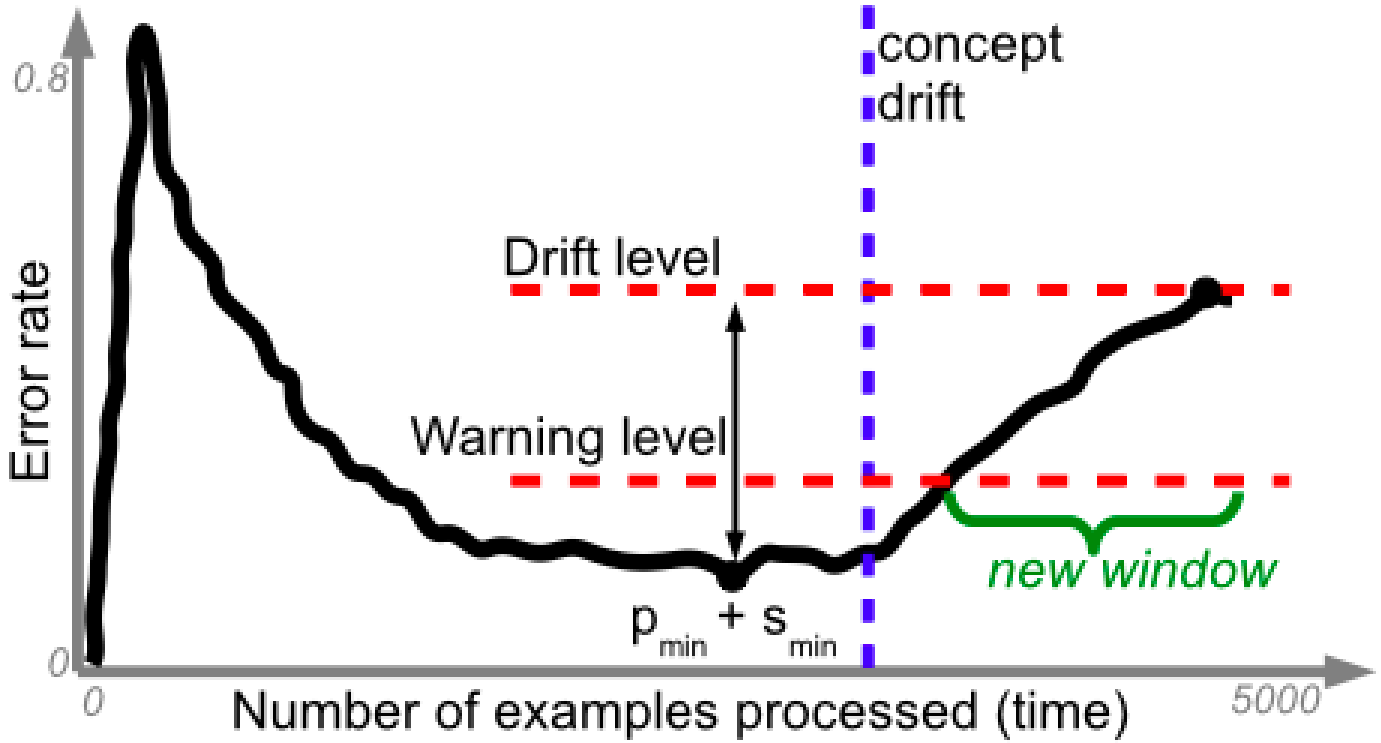
Gama, et. al, Learning with Drift Detection, SBIA 2004, Springer.

Learning from data streams is a continuous process. Monitor the quality of the learning process using quality control techniques. The online error (e) of a learning algorithm is:

- **In-control** if $e < e_{min} + 2 \times s_{min}$
- **Out-control** if $e > e_{min} + 3 \times s_{min}$
- **Warning Level**: otherwise

Concept Drift

Statistical process control: monitor and control the learning process.



Algorithm ADaptive Sliding WINdow

A. Bifet, R Gavalda: Learning from Time-Changing Data with Adaptive Windowing. SDM 2007

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

ADWIN using a Exponential Histogram 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

Evaluation

Assessing the learned models

- Error estimation
- Model Selection

Metrics for Evaluation in Data Streams

- **Loss**: measuring how appropriate is the current model to the actual status of the nature.
- **Memory used**: Learning algorithms run in fixed memory. We need to evaluate the memory usage over time, and the impact in accuracy when using the available memory.
- **Speed of Processing examples**: Algorithms must process the examples as fast if not faster than they arrive.

Evaluation Methods

You cannot touch the same water twice.

Cross Validation and variants does not apply.

Two alternatives:

- Holdout if data is stationary.
- Sequential Sampling

What if the distribution is non-stationary ?

- The *prequential* approach.
  - For each example:
    - First: make a prediction
    - Second: update the model, whenever the target is available.
- Evaluation over time-windows?

Prequential Evaluation

On evaluating stream learning algorithms Gama, Sebastião, Rodrigues, Machine Learning 2013

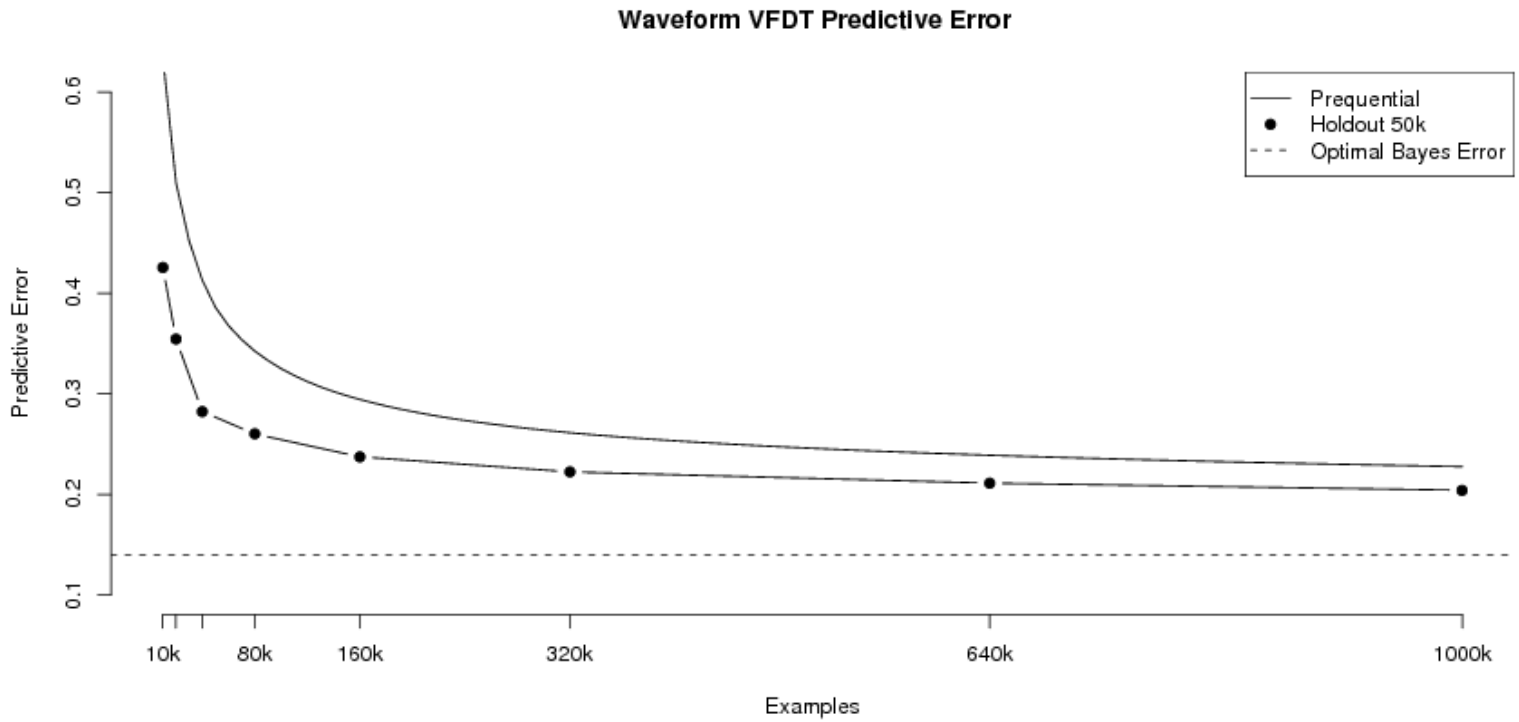
Definition: *The prequential error, computed at time i, is based on an accumulated sum of a loss function between the prediction and observed values:*

$$P_e(i) = \frac{1}{i} \sum_{k=1}^{i} L(y_k, \hat{y}_k) = \frac{1}{i} \sum_{k=1}^{i} e_k.$$

1. Provides a single number **at each time stamp**: a learning curve.
2. Pessimist estimator of accuracy.
3. Problematic to apply with algorithms with large testing time (k-NN).

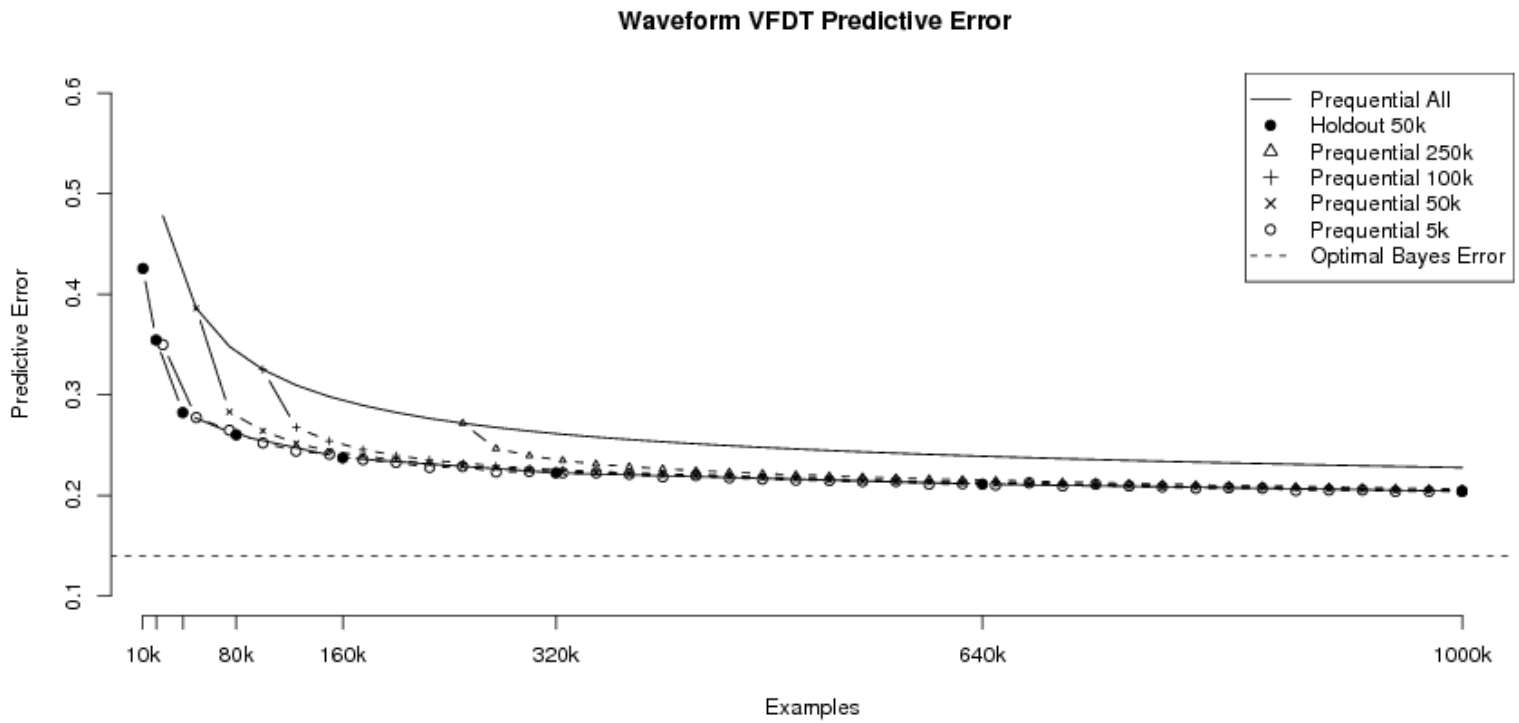
Prequential versus Holdout

Prequential is a pessimistic estimator.



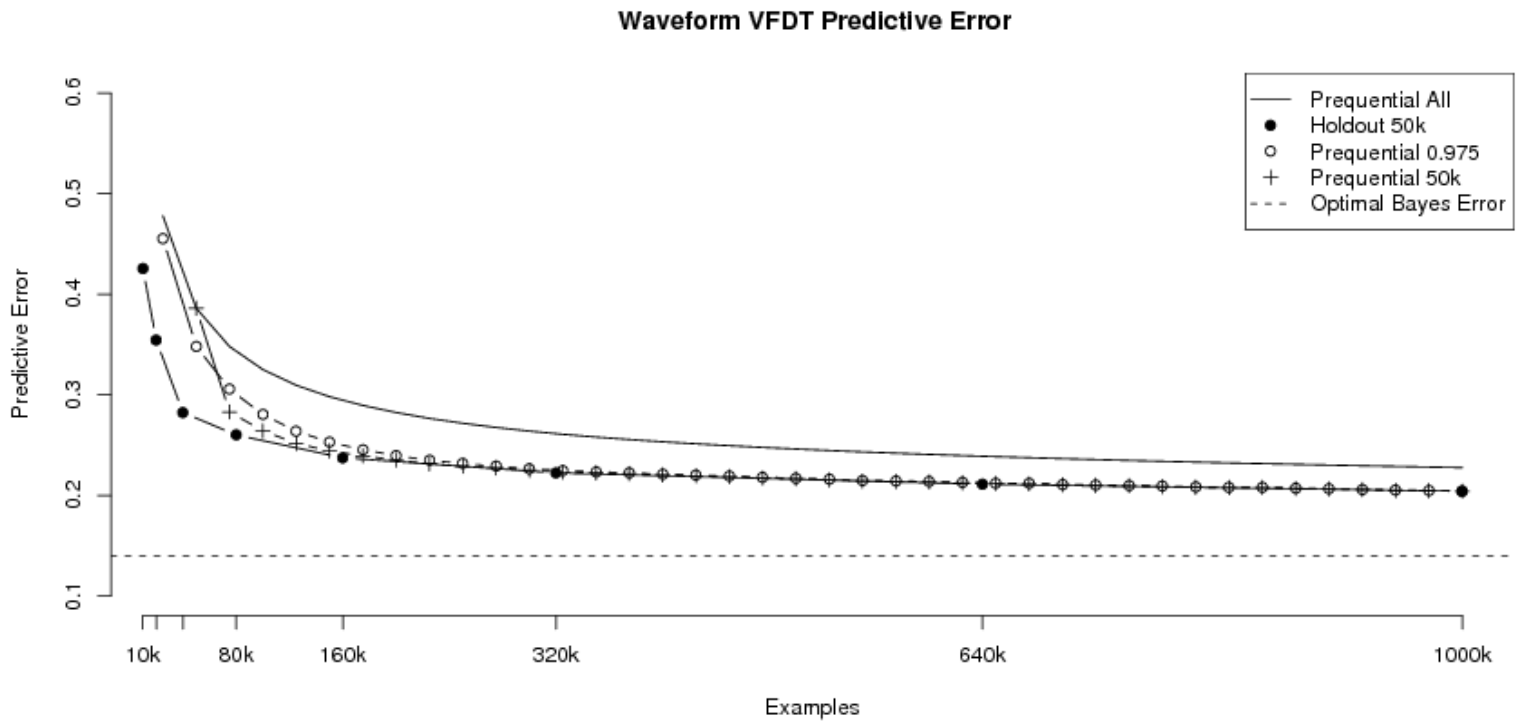
Prequential (sliding window) versus Holdout

Prequential over a sliding window converges to the holdout estimator.



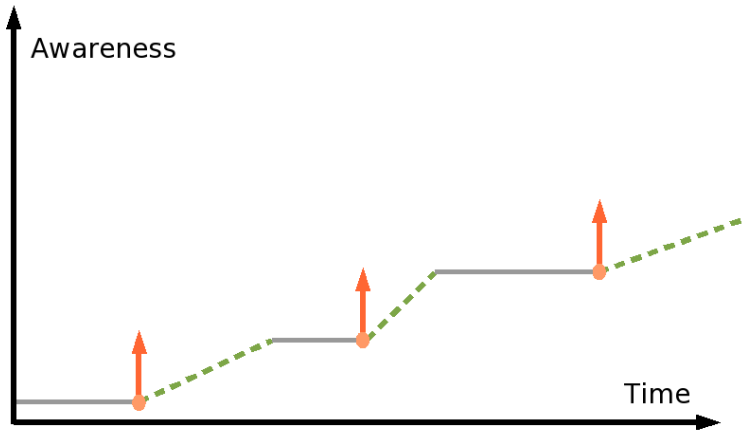
Prequential (fading factor) versus Holdout

Prequential using fading factors converges to the holdout estimator.



Novelty

- Open Set recognition
- Emerging Classes



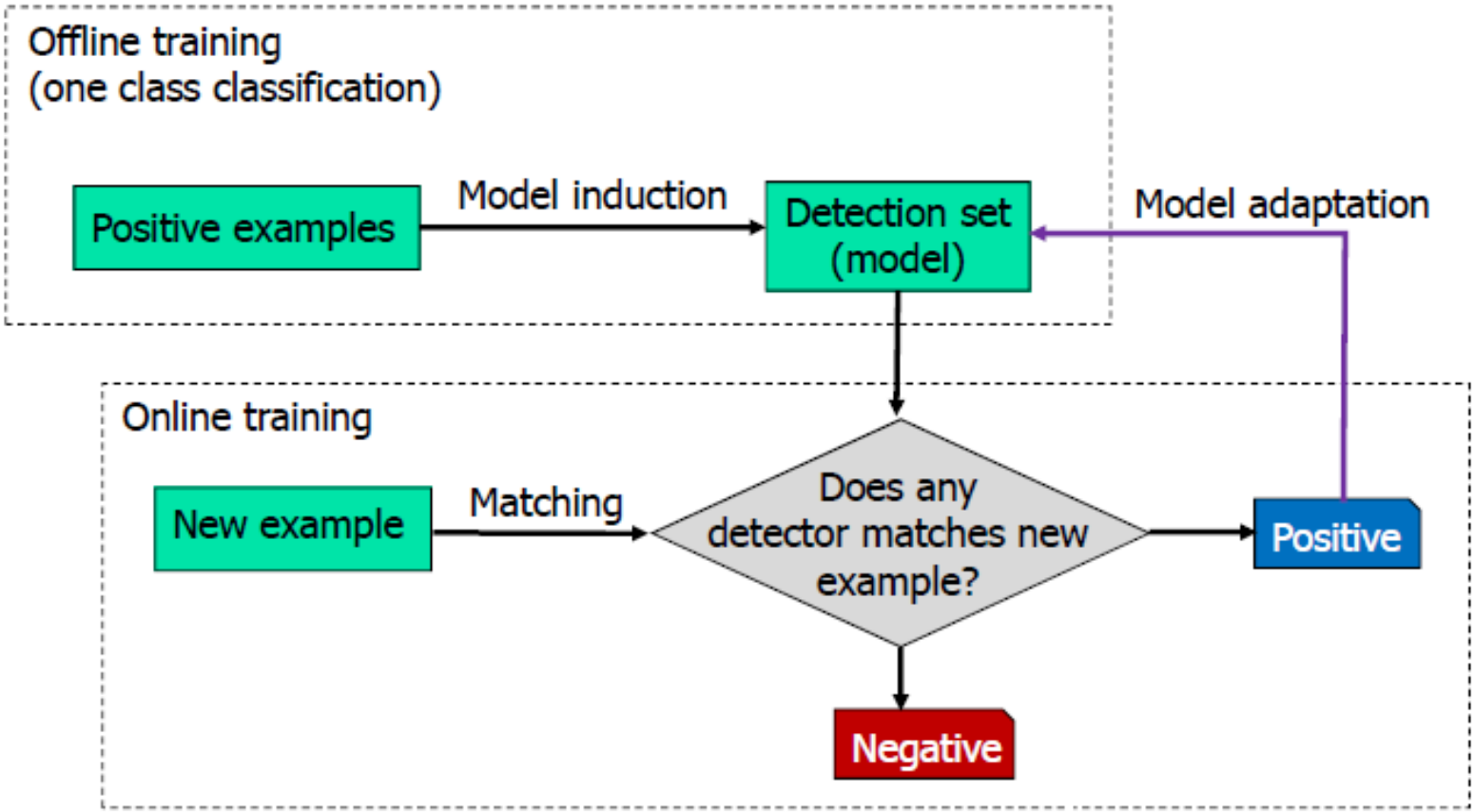
Definition

- Novelty Detection refers to the automatic identification of unforeseen phenomena embedded in a large amount of normal data.
- *Novelty* is a relative concept with regard to our current knowledge:
  - It must be defined in the context of a representation of our current knowledge.
- Specially useful when novel concepts represent abnormal or unexpected conditions
  - Expensive to obtain abnormal examples
  - Probably impossible to simulate all possible abnormal conditions

Context

- In real problems, as time goes by
  - The distribution of known concepts may change
  - New concepts may appear
- By monitoring the data stream, emerging concepts may be discovered
- Emerging concepts may represent
  - An extension to a known concept (Extension)
  - A novel concept (Novelty)
- Several interesting applications: Early Detection of Fault in Jet Engines, Intrusion Detection in computer networks, Breaking News in a flow of text documents (news articles), Burst of Gamma-ray (astronomical data),

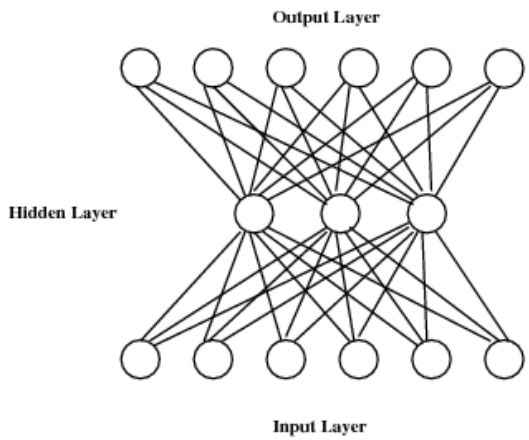
One-Class Classification



Autoassociator Networks

Concept-learning in the absence of counter-examples: an autoassociaton-based approach Nathalie Japcowicz, 1999

- Three layer network
- The nr. of neurons in the output layer is equal to the input layer
- Train the network such that $\vec{y}$ is equal to the $\vec{x}$
- The network is trained to reproduce the input at the output layer



Autoassociator Networks

To classify a test example $\vec{x}$

- Propagate $\vec{x}$ through the network and let $\vec{y}$ be the corresponding output;
- If $\sum_i^k (x_i - y_i)^2 < \textit{Threshold}$ Then the example is considered from class *normal*;
- Otherwise, $\vec{x}$ is a counter-example of the *normal* class.

Novelty detection

- Training set (Offline Phase)
  - $D_{tr} = (X_1, y_1), (X_2, y_2), \ldots, (X_m, y_m)$
  - X_i: vector of input attributes for the ith example
    y_i: target attribute
  - $y_i \in Y_{tr}$ where $Y_{tr} = c_1, c_2, \ldots, c_L$
- When new data arrive (Online Phase)
  - Given a sequence of unlabelled examples X_{new}
    Goal: Classify X_{new} in Y_{all} where $Y_{all} = c_1, c_2, \ldots, c_L, \ldots, c_K$ and $K > L$

Novelty Detection Systems

- ECSMiner: Assume that the class label of new examples is known
- OLINDDA: unsupervised, but restricted to binary classification problems
- MINAS (MultI-class learNing Algorithm for data Streams)
  - Does not use the class labels of new examples
  - Can deal with novelty detection in data streams multi-class problem

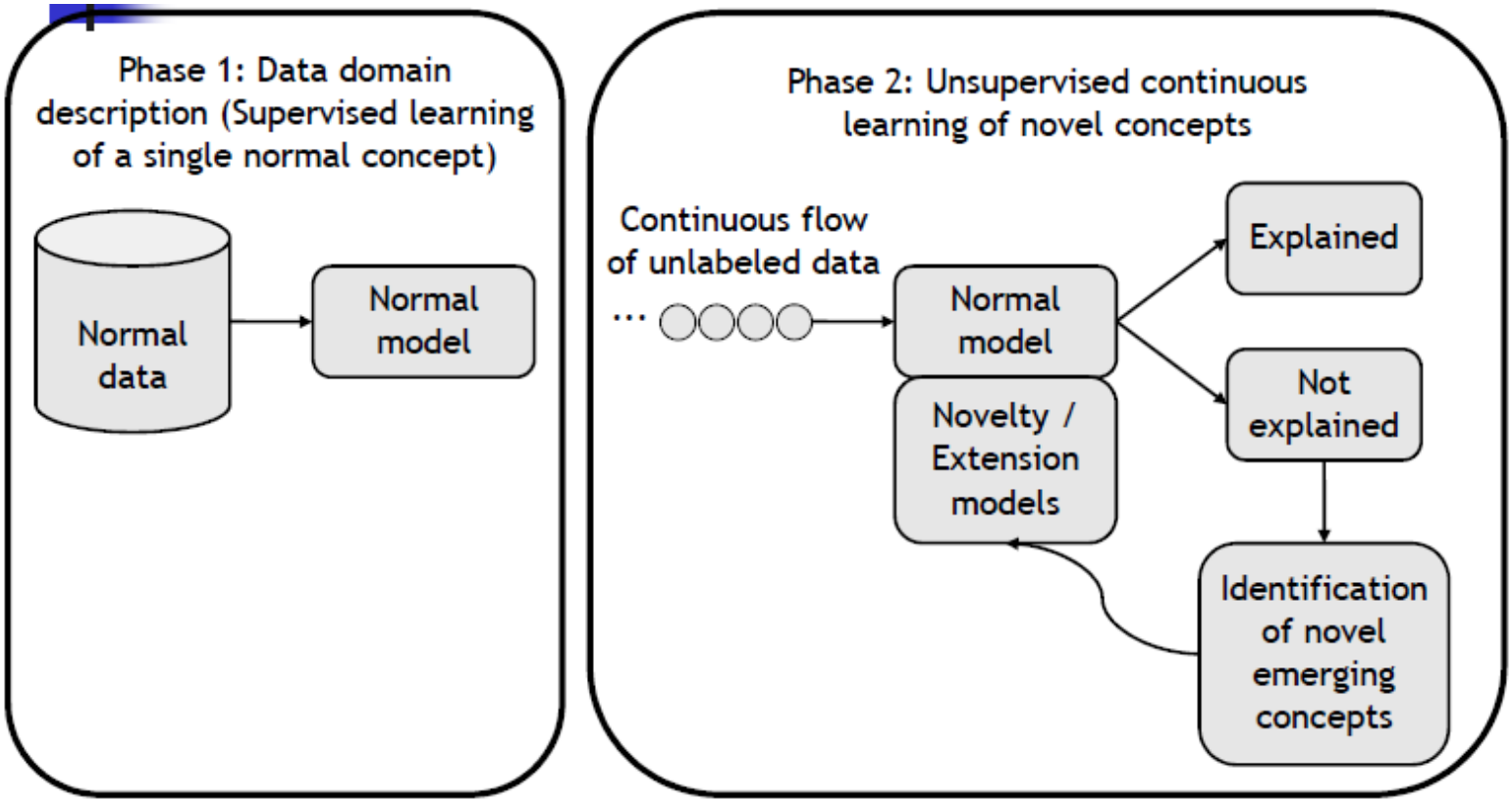
OLINDDA algorithm

OnLIne Novelty and Drift Detection Algorithm

Spinosa, Carvalho, Gama: *OLINDDA: a cluster-based approach for detecting novelty and concept drift in data streams* SAC 2007

- Offline and Online phases
- Models: normal, extension and novelty
- Each model is represented by a set of clusters
- Not suitable for multi-class problem

OLLINDA


Phase 1: Data domain description (Supervised learning of a single normal concept)
Normal data
Normal model
Phase 2: Unsupervised continuous learning of novel concepts
Continuous flow of unlabeled data
Normal model
Explained
Not explained
Novelty / Extension models
Identification of novel emerging concepts

ECSMiner algorithm

Masud, Gao, Khan, Han, and Thuraisingham, *Classification and novel class detection in concept-drifting data streams under time constraints*, TKDE 2011

Supervised algorithm integrating novel concepts and concept drift

- Ensemble of classifiers
- Creates a new model when all examples in a chunk are labeled
  - Supposes that all examples in the stream will be labeled (after a delay of TI time units)
  - An instance will be classified in until Tc time units of its arrival

Minas algorithm

MINAS: Multiclass Learning Algorithm for Novelty Detection in Data Streams, E. Faria, J. Gama, A. Carvalho, DAMI (to appear)

- Unsupervised algorithm for novelty detection in data streams multi-class problems
  Represents each known class by a set of hyperspheres
- Use of offline (training) and online phases
  In each phase learns one or more classes
- Cohesive set of examples is necessary to learn new concepts or extensions
  Isolated examples are not considered as novelty

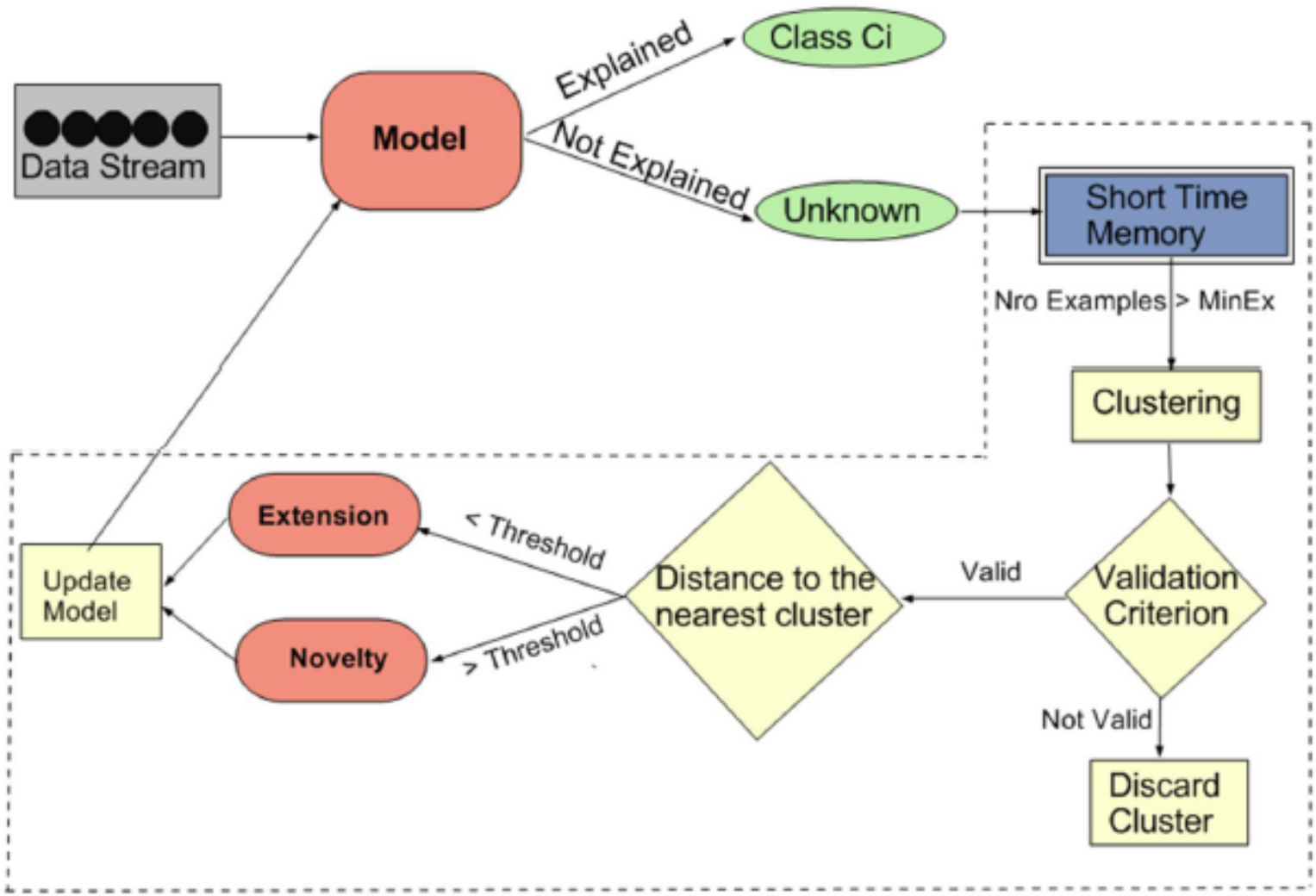
MINAS - Offline phase

- Learns a decision model based on the known concept about the problem
  KMeans or Clustream
- Run only once
- Each class is represent by a set of clusters (hyperspheres)

MINAS - Online phase

- Receives new examples from the stream
- Classify each new example
  - In one of the known classes or
  - As unknown
- Cohesive group of unknown examples are used to detect new classes or extensions

Minas


Data Stream
Model
Explained
Class Ci
Not Explained
Unknown
Short Time Memory
Nro Examples > MinEx
Clustering
Validation Criterion
Valid
Not Valid
Discard Cluster
Distance to the nearest cluster
< Threshold
> Threshold
Extension
Novelty
Update Model

Novelty Detection Bibliography

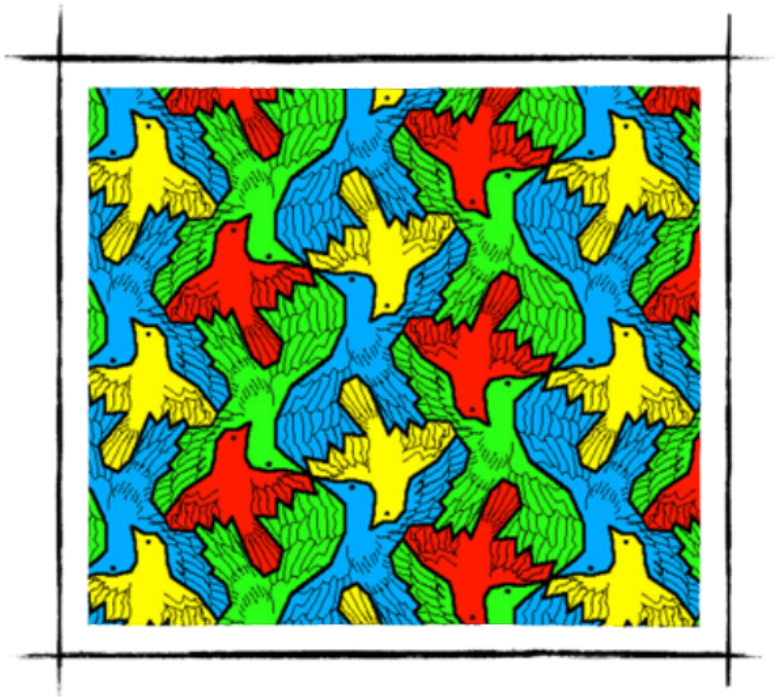
- Masud, Gao, Khan, Han, and Thuraisingham, *Classification and novel class detection in concept-drifting data streams under time constraints*, TKDE 2011
- Spinosa, Carvalho, Gama: *OLINDDA: a cluster-based approach for detecting novelty and concept drift in data streams* SAC 2007
- MINAS: Multiclass Learning Algorithm for Novelty Detection in Data Streams, E. Faria, J. Gama, A. Carvalho, DAMI (to appear)
- P. Angelov and X. Zhou, *Evolving fuzzy-rule-based classifiers from data streams* Trans. Fuz Syst. 2008.
- D. Tax and R. Duin, *Growing a multi-class classifier with a reject option* Pattern Recognit. Lett., 2008.
- F. Denis, R. Gilleron, and F. Letouzey, *Learning from positive and unlabeled examples*, Theoretical Comput. Sci., 2005.
- D. Cardoso and F. França *A Bounded Neural Network for Open Set Recognition*, IJCNN 2015

Outline

Frequent Pattern Mining

Given a collection of sets of items, find all the subsets that occur frequently

- Market basket mining
- Item recommendation



Frequent Itemsets

- Frequent pattern mining refers to finding patterns that occur greater than a pre-specified threshold value.
- Patterns refer to items, itemsets, or sequences.
- *Support:* the percentage of the pattern occurrences to the total number of transactions.

Frequent Pattern Mining in Data Streams

The process of frequent pattern mining over data streams differs from the conventional one as follows:

- The technique should be linear or sublinear: **You Have Only One Look**.
- top-k items, *heavy hitters*, sketch-based techniques
- frequent itemsets.

Mining Patterns over Data Streams

Requirements: fast, use small amount of memory and adaptive

- Type:
  - Exact
  - Approximate
- Per batch, per transaction
- Incremental, Sliding Window, Adaptive
- Frequent, Closed, Maximal patterns

Approximate Counting

Given a stream S of m items $\langle e_1, e_2, \ldots e_m \rangle$ the frequency of an item $e \in S$ is $f(e) = |\{e_j \in S : e_j = e\}|$.

- The exact ϕ-frequent items are those with $f(e) > \phi \times m$, with $\phi \leq 1$
- The ϵ-approximate frequent items those with $f(e) > (\phi - \epsilon) \times m$, with $\phi \leq 1$

Tasks

Main tasks:

- Representing sets
- **Frequency estimates for all elements in the stream**:
  Sketch-based techniques: linear projection of the input
  - Count-min sketch
  - Distinct elements: FM sketch
- **Top-k items**:
  Counter-based techniques: monitor a subset of items
  - The *Frequent* Algorithm
  - The *Space-Saving* Algorithm
  - Frequent items
  - Sticky Sampling
  - Lossy Counting

Frequent Items (Heavy Hitters) in Data Streams

Manku and Motwani have two master algorithms in this area:

- Sticky Sampling
- Lossy Counting

G. S. Manku and R. Motwani. *Approximate Frequency Counts over Data Streams*, in Proceedings of the 28th International Conference on Very Large Data Bases (VLDB), 2002.

Lossy Counting

- Lossy counting is a deterministic technique.
- The user inputs two parameters
  - Minimum Support (s)
  - Admissible Error (ϵ)
- The data structure has entries of data elements, their associated frequencies (e, f, $\triangle$) where $\triangle$ is the maximum possible error in f.
- The stream is conceptually divided into buckets with a width $w = 1/\epsilon$.
- Each bucket is labeled by a value of N/w, where N starts from 1 and increases by 1.

Lossy Counting

- For a new incoming element, the data structure is checked
  - If an entry exists, then increment the frequency
  - Otherwise, add a new entry with $\triangle = b_{current} - 1$ where $b_{current}$ is the current bucket label.
- When switching to a new bucket, all entries with $f + \triangle < b_{current}$ are deleted.

Error Analysis

Output:

- Elements with counter values exceeding $s \times N - \epsilon \times N$

How much do we undercount?

- If the current size of stream is N and window-size $= 1/\epsilon$ then *frequency error* $\leq$ *#window* $= \epsilon \times N$

Approximation guarantees:

- Frequencies underestimated by at most $\epsilon \times N$
- No false negatives
- False positives have true frequency at least $s \times N - \epsilon \times N$

How many counters do we need?

- Worst case: $1/\epsilon log(\epsilon N)$ counters

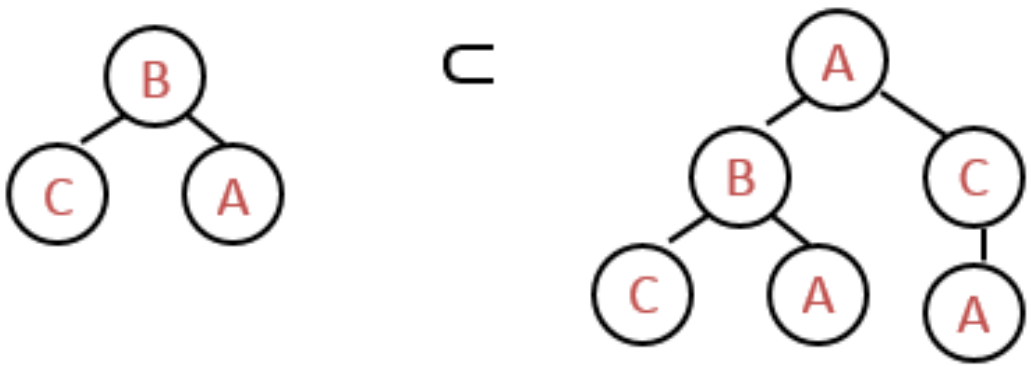
Frequent Pattern mining

Patterns: sets with a *subpattern* relation $\subset$

Sets: $\{cheese, milk\} \subset \{milk, peanuts, cheese, butter\}$

Sequences: $(search?buy) \subset (home?search?cart?buy?exit)$

Graphs:

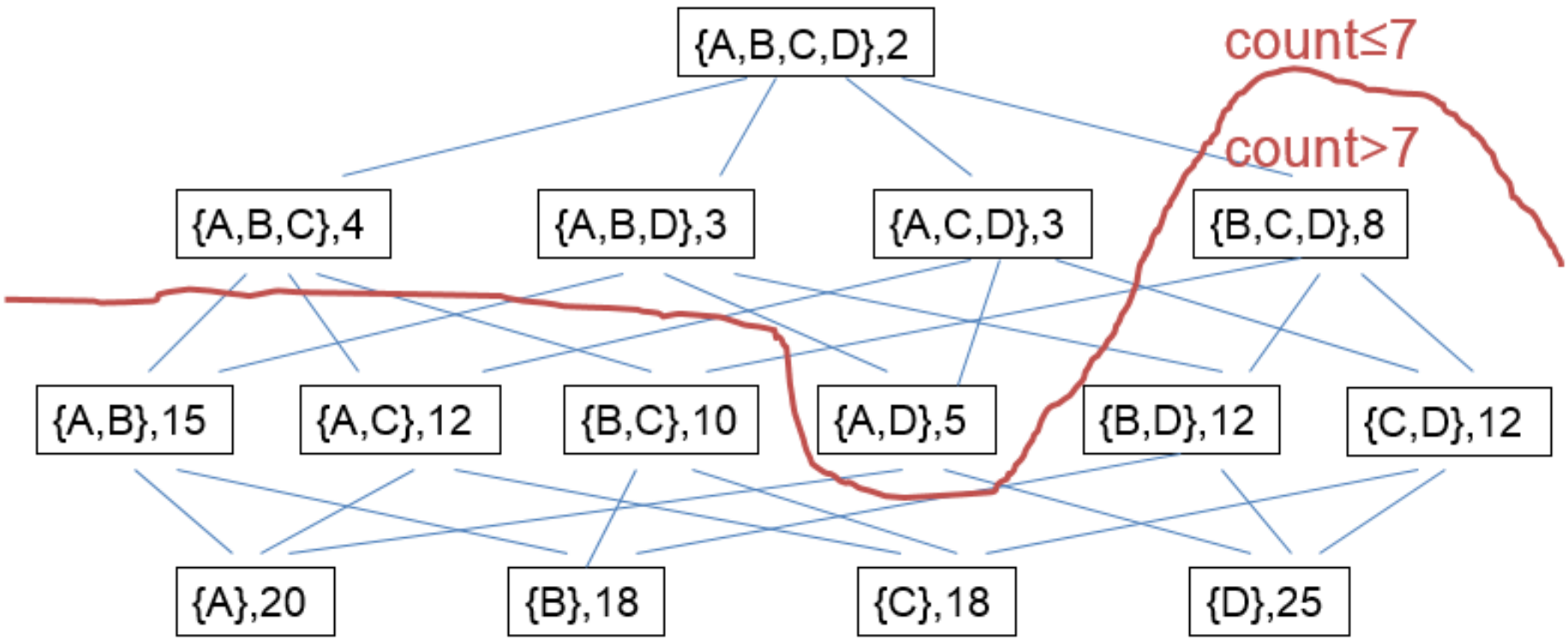


Applications: market basket analysis, intrusion detection, churn prediction, feature selection, XML query analysis, query and clickstream analysis, anomaly detection ...

Pattern mining in streams: definitions

- The support of a pattern T in a stream S at time t is the probability that a pattern T' drawn from S's distribution at time t is such that $T \subset T'$
- **Typical task**: Given access to S, at all times t, produce the set of patterns T with support at least ϵ at time t
- A pattern is closed if no superpattern has the same support.
- No information is lost if we focus only on closed patterns.

Key data structure: Lattice of patterns, with counts



Fundamentals

- *A priori* property: $t \subseteq t' \Rightarrow support(t) \geq support(t')$
- Closed: none of its supersets has the same support
  Can generate all freq. itemsets and their support
- Maximal: none of its supersets is frequent
  Can generate all freq. itemsets (without support)
- Maximal $\subseteq$ Closed $\subseteq$ Frequent $\subseteq$ D

FP-Stream

C. Giannella, J. Han, J. Pei, X. Yan, P. S. Yu: *Mining frequent patterns in data streams at multiple time granularities*. NGDM (2003)

- Multiple time granularities
- Based on FP-Growth (depth-first search over itemset lattice)
- Pattern-tree with Tilted-time window
  Tilted-time window: logarithmically aggregated time slots (log number of levels, aggregate when the level is full, push the aggregate one level up)
- Time sensitive queries, emphasis on recent history
- High time and memory complexity

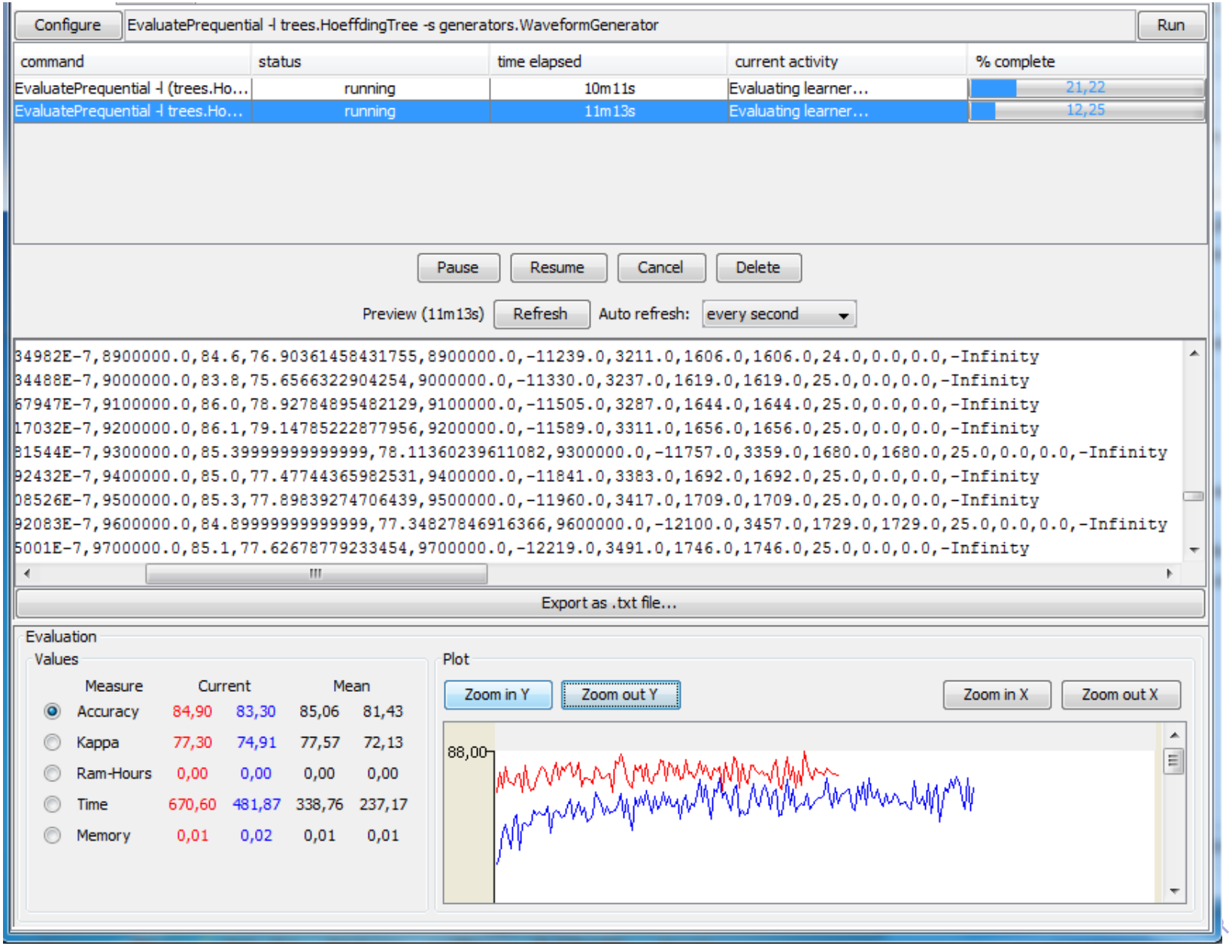
Moment

Y. Chi , H. Wang, P. Yu , R. Muntz: *Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window*. ICDM 2004

- Keeps track of boundary below frequent itemsets
- Closed Enumeration Tree (CET) ($\approx$ prefix tree)
  - Infrequent gateway nodes (infrequent)
  - Unpromising gateway nodes (infrequent, dominated)
  - Intermediate nodes (frequent, dominated)
  - Closed nodes (frequent)
- By adding/removing transactions closed/infreq. do not change

Outline

Massive Online Analysis



Open Challenges

Open Challenges

- Structured input and output
- semi-supervised learning
- Multi-target, multi-task and transfer learning
- Millions of classes
- Visualization
- Distributed Streams
- Representation learning
- Ease of use

Lessons Learned

Learning from data streams:

- A new mind-set for machine learning!
- Learning is not *one-shot*: is an evolving process;
- We need to monitor the learning process;
- Opens the possibility to reasoning about the learning

Reasoning about the Learning Process

Intelligent systems must:

- be able to adapt continuously to **changing environmental conditions** and evolving user habits and needs.
- be capable of **predictive self-diagnosis**.

The development of such self-configuring, self-optimizing, and self-repairing systems is a major scientific and engineering challenge.

References I

Aggarwal, C. C., Han, J., Wang, J., e Yu, P. S. (2003).
A framework for clustering evolving data streams.
In *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany*, pages 81–92.

Bifet, A., Gavaldà, R., Holmes, G., e Pfahringer, B. (2018).
Machine Learning for Data Streams with Practical Examples in MOA.
MIT Press, Cambridge, MA.

Domingos, P. M. e Hulten, G. (2000).
Mining high-speed data streams.
In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000*, pages 71–80.

Duarte, J., Gama, J., e Bifet, A. (2016).
Adaptive model rules from high-speed data streams.
TKDD, 10(3):30:1–30:22.

Fritsch, S., Guenther, F., e following earlier work by Marc Suling (2012).
neuralnet: Training of neural networks.
R package version 1.32.

Gama, J. (2010).
Knowledge Discovery from Data Streams.
Chapman and Hall / CRC Data Mining and Knowledge Discovery Series. CRC Press.

References II

Gama, J., Rocha, R., e Medas, P. (2003).

Accurate decision trees for mining high-speed data streams.
In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 523–528.

Gama, J., Sebastião, R., e Rodrigues, P. P. (2013).

On evaluating stream learning algorithms.
Machine Learning, 90(3):317–346.

Hempstalk, K., Frank, E., e Witten, I. H. (2008).

One-class classification by combining density and class probability estimation.
In *ECML/PKDD (1)*, pages 505–519.

Hornik, K., Buchta, C., e Zeileis, A. (2009).

Open-source machine learning: R meets Weka.
Computational Statistics, 24(2):225–232.

Hulten, G., Spencer, L., e Domingos, P. M. (2001).

Mining time-changing data streams.
In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA, August 26-29, 2001*, pages 97–106.

Ikonomovska, E., Gama, J., e Dzeroski, S. (2011).

Learning model trees from evolving data streams.
Data Min. Knowl. Discov., 23(1):128–168.

References III

Japkowicz, N., Myers, C., e Gluck, M. A. (1995).
A novelty detection approach to classification.
In *IJCAI*, pages 518–523. Morgan Kaufmann.

Kosina, P. e Gama, J. (2015).
Very fast decision rules for classification in data streams.
Data Min. Knowl. Discov., 29(1):168–202.

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., e Wozniak, M. (2017).
Ensemble learning for data stream analysis: A survey.
Information Fusion, 37:132–156.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., e Leisch, F. (2014).
e1071: Misc Functions of the Department of Statistics (e1071), TU Wien.
R package version 1.6-4.

Pereira, P., Ribeiro, R. P., e Gama, J. (2014).
Failure prediction - an application in the railway industry.
In *Discovery Science - 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014. Proceedings*, pages 264–275.

R Core Team (2014).
R: A Language and Environment for Statistical Computing.
R Foundation for Statistical Computing, Vienna, Austria.

Ribeiro, R. P., Pereira, P. M., e Gama, J. (2016).
Sequential anomalies: a study in the railway industry.
Machine Learning, 105(1):127–153.

References IV

Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., Carvalho, A. C. P. L. F. d., e Gama, J. a. (2013).
Data stream clustering: A survey.
ACM Comput. Surv., 46(1):13:1–13:31.

Tax, D. M. J. e Duin, R. P. W. (2004).
Support vector data description.
Machine Learning, 54(1):45–66.

Wang, Z. (2015).
Predictive maintenance (from a machine learning perspective).
IEEE BigData Tutorial.

Witten, I. H. e Frank, E. (2005).
Data Mining: Practical Machine Learning Tools and Techniques.
Morgan Kaufmann, San Francisco, 2nd edition.

Zhang, T., Ramakrishnan, R., e Livny, M. (1996).
Birch: an efficient data clustering method for very large databases.
In *In Proc. of the ACM SIGMOD Intl. Conference on Management of Data (SIGMOD*, pages 103–114.