

INFO-H-515:

BIG DATA ANALYTICS

Streaming analytics

Gianluca Bontempi

Machine Learning Group
Boulevard de Triomphe - CP 212
<http://mlg.ulb.ac.be>

HUMAN AS DATA STREAM PROCESSOR

- Human beings perceive each instant of their life through an array of sensory observations (visual, aural, nervous, etc).
- However, over the course of their life, they manage to abstract and store only part of the observations,
- Humans may function adequately even if they can not recall every detail of each instant of their lives.

MAJOR TRENDS

1. Explosion of data driven applications with massive data streams (banking and credit transactions, satellite measurements, astronomical surveys, internet, sensor networks) due to the never ending appearance of new measurement technologies (e.g. wireless sensor networks, industry 4.0, IoT), whose common denominator is the capability of measuring in real-time.
2. Streaming nature of data: impossibility of storing all the data or of processing a sample more than once (one-pass only).
3. Need of complex functionalities: detecting outliers, extreme events, monitor complex correlations, track trends, pattern recognition (classification, forecasting)
4. Advances in hardware technology: computational resources are shifting toward parallel and distributed architectures, with multicore and cloud computing platforms providing access to hundreds or thousands of processors.

CHALLENGES

- Multiple passes on data are no more possible. One can process a data item at most once
- Traditional approach of collecting, organising, storing, and analysing data is inadequate for applications where the reaction to events must be immediate.
- Asynchronous and fast arrival of stream elements (rate of arrival not under control).
- Space less than linear in the input size: only a fixed number of observations can be stored.
- Time significantly less than the input size: amount of computation time per observation should be constant rather than increasing over time.
- Temporal component, non stationarity. Data distribution may evolve with time.
- Large dimensionality and distributed nature of streams (sensor networks).
- Ad-hoc queries: arbitrary and nonstandard query about the stream.

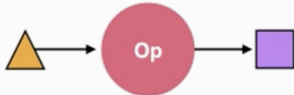
STRATEGIES FOR STREAMING ANALYTICS

Possible strategies to move from batch to streaming analytics

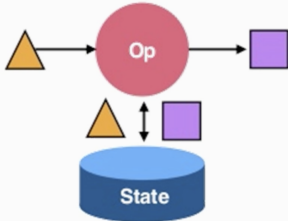
- From parallelization to sequentialization.
- From exact solution to approximation (e.g. use of randomized algorithms).
- Summarization techniques, e.g. sampling, windowing or budgeted storage.
- State estimation techniques: data are not meaningful by itself but as observations of some important hidden **state**.
- Hierarchical analysis: simple and faster analysis at low level (more data) and more sophisticated analysis at higher levels (few data), lambda architecture.
- Iterative (online) versions of batch learning algorithms.

Stateful Streaming

Stateless Stream Processing



Stateful Stream Processing



30

SUMMARIZATION SAMPLING STRATEGIES

- Rationale: select a subset of the stream such that the answer to the query using the subset is a good approximation of the answer which would have been obtained using the entire stream.
- Representative sample: subset of the population which reflects the distribution of the entire population.
- Importance of sampling strategies: in a transactional stream (e.g. credit card fraud detection), should we rather sample transactions or users?
- Use of hash functions as random number generator (with the additional feature of keeping memory of the past decision): to take a sample of size a/b , $a < b$, hash the key values to b buckets and accept the tuple if the hash value is less than a .
- Adaptation of the percentage size of sample to the stream size.

SAMPLING TECHNIQUES

- Random: Each member has equal chances of being selected. Sampling is done in a single step with each item selected independently of the other members of the population.
- Systematic: arrange the elements of the population in some order (e.g. arrival order) and then select them regularly (e.g. select a sample each 20).
- Stratified: divide the population according to some characteristics.

RANDOMIZED ALGORITHM

- A randomized algorithm is an algorithm that employs a degree of randomness as part of its logic.
- Probability of correct result can be estimated and properly tuned
- Common in streaming to solve problems like
 - Filtering: Bloom algorithm is used to test whether an element is a member of a set (without storing traces of all seen elements).
 - Counting distinct elements: Hyperloglog algorithm approximate counting of number of distinct elements in a stream.
 - Reservoir algorithm: the first n points in the data stream are added to the reservoir for initialization. Subsequently, when the $(i + 1)$ -th point from the data stream is received, it is added to the reservoir with probability $n/(i + 1)$. This point replaces a randomly chosen point in the reservoir.
 - Second moment: Alon-Matias-Szegedy Algorithm

Discussed in the first part of the course. Here we will focus more on streaming version of estimation and learning tasks.

ONLINE LEARNING

- The goal of online learning is to make a sequence of accurate predictions given knowledge of the correct answers to previous prediction tasks and additional available information.
- Given that prediction requires learning and learning is a form of estimation, there a number of links between online learning and related disciplines like numerical analysis, adaptive filtering, sequential estimation, signal processing.
- It can refer to both supervised and unsupervised learning as well as to feature selection tasks.
- In supervised learning it can refer to parametric identification (online learning of parameters) and/or structural identification (online learning of model structure).

ONLINE LEARNING

- It differentiates from batch learning since the training samples are treated in a sequential manner (with or without repetition). It is also called one-pass learning.
- Historically it has been the first effective approach used to train neural networks (Hebb, 1949, Rosenblatt, 1957).
- It revives thanks to the big data setting and large scale tasks
- During the early days it was mainly based on heuristics to deal with convergence issues
- More recently theoretical analysis led to a deeper understanding of the algorithms
- Note that it is the treatment and the availability of the training set which determines the batch/online nature and not the form of the algorithm (which in both case could be iterative)

ONLINE VS BATCH LEARNING

- Online learning is typically simpler, easier to grasp since obtained as a simple function of the latest estimate (state) and the current observation.
- Batch learning typically faster for small datasets but more inefficient for large networks and large training sets
- Batch learning more prone to local minima
- Online learning more natural solution for dealing with nonstationarity
- Online learning more sensitive to the choice of training parameters (e.g. learning rate)
- From gradient descent (batch) to stochastic gradient descent (online): batch learning deals with a deterministic task while online addresses a stochastic optimization problem

Note that an iterative algorithm may be used for both types of learning.

WHEN USE ONLINE LEARNING?

- Data does not fit in memory
- You cannot store your data
- Nonstationarity issues

PERFORMANCE MEASURES

- speed of convergence
- steady state error
- tracking ability
- robustness against noise
- computational complexity
- modularity
- numerical stability and precision

Note that describing such properties in a formal manner is not easy in generic settings.

STREAMING PARAMETER IDENTIFICATION

INCREMENTAL CALCULATIONS OF MEAN AND VARIANCE

For simple statistics incremental formulations are well known

- Sample average:

- batch formulation: $\mu_{(N)} = \frac{1}{N} \sum_{i=1}^N z_i$

- sequential formulation:

$$\mu_{(N)} = \mu_{(N-1)} + \frac{1}{N}(z_N - \mu_{(N-1)}) = \mu_{(N-1)} + \alpha_{(N-1)}(z_N - \mu_{(N-1)})$$

- Note that $\alpha_{(N)} = 1/N \xrightarrow{N \rightarrow \infty} 0$

- Sample variance:

- batch formulation: $\sigma_{(N)}^2 = \frac{1}{N} \sum_{i=1}^N (z_i - \mu_{(N)})^2 = \left(\frac{1}{N} \sum_{i=1}^N z_i^2 \right) - \left(\frac{1}{N} \sum_{i=1}^N z_i \right)^2$

- sequential formulation: $\sigma_{(N)}^2 = \sqrt{S_{(N)}/N}$ where

$$S_{(N)} = S_{(N-1)} + (z_N - \mu_{(N-1)})(z_N - \mu_{(N)})$$

or

$$S_{(N)} = S_{(N-1)} + N(N-1)(\mu_{(N)} - \mu_{(N-1)})^2$$

- Recursive formulas for weighted means and variances exists too (look at linked article in the course web page)

PARAMETER IDENTIFICATION LEARNING

- It is the core step in supervised learning which requires multivariate optimization.
- Given a generic nonlinear learner with a fixed parametric structure and a number of unknown parameters (e.g. weights of a neural network), the parameter identification requires the solution of a multivariate nonlinear optimization problem
- The cost function to minimize is the empirical or training error, e.g.

$$J_N(\theta) = \frac{\sum_{i=1}^N (y_i - h(x_i, \theta))^2}{N}$$

- The outcome is a set of tuned parameters (e.g. neural network weights) $\hat{\theta}$

$$\hat{\theta} = \arg \min_{\theta} J_N(\theta)$$

such to minimize a cost function related to training error.

- Batch learning: optimization is carried out with respect to the entire training set simultaneously
- Closed form solutions vs iterative solutions

BATCH LINEAR LEAST-SQUARES

The **least-squares estimator** $\hat{\beta}$ minimizes the cost function

$$\hat{\beta} = \arg \min_b \sum_{i=1}^N (y_i - x_i^T b)^2 = \arg \min_b ((Y - Xb)^T (Y - Xb))$$

In the linear case it exists a closed form solution

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

where the $X^T X$ matrix is a symmetric $[p \times p]$ matrix.

Last time we saw how to distribute the computation of the closed form solution. In what follow, we see how to solve it in a streaming fashion.

RECURSIVE LINEAR LEAST-SQUARES

Let us consider a multiple linear regression problem and a sequence of input-output samples x_i, y_i where $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$

$$\begin{cases} V_{(t)} &= V_{(t-1)} - \frac{V_{(t-1)}x_t^T x_t V_{(t-1)}}{1+x_t V_{(t-1)}x_t^T} \\ \alpha_{(t)} &= V_{(t)}x_t^T \\ e &= y_t - x_t \hat{\beta}_{(t-1)} \\ \hat{\beta}_{(t)} &= \hat{\beta}_{(t-1)} + \alpha_{(t)}e \end{cases}$$

where $\hat{\beta}_{(t)}$ is the estimate after t observations.

Typical initialization

- $\hat{\beta}_{(0)}$ is usually put equal to a zero vector.
- $V_{(0)} = aI$, with $a > 0$ and I identity matrix since $V_{(0)}$ represents the variance of the estimator. By setting a equal to a large number the RLS algorithm will diverge very rapidly from the initialization $\hat{\beta}_{(0)}$.

RLS WITH FORGETTING FACTOR

Consider the two situations

- the phenomenon underlying the data is linear but non stationary
- the phenomenon underlying the data is stationary and nonlinear but can be approximated by a linear model locally in time. This is of large use in adaptive control problems.

In these cases it is useful not give always the same importance to all the historical data but assign higher weights to more recent data (and forget older data).

RLS techniques can deal with these situations by a formulation with forgetting factor $\nu < 1$.

RLS WITH FORGETTING FACTOR (II)

$$\begin{cases} V_{(t)} &= \frac{1}{\nu} \left(V_{(t-1)} - \frac{V_{(t-1)} x_t^T x_t V_{(t-1)}}{1 + x_t V_{(t-1)} x_t^T} \right) \\ \alpha_{(t)} &= V_{(t)} x_t^T \\ e &= y_t - x_t \hat{\beta}_{(t-1)} \\ \hat{\beta}_{(t)} &= \hat{\beta}_{(t-1)} + \alpha_{(t)} e \end{cases}$$

- The smaller ν , the higher the forgetting.
- Note that for $\nu = 1$ we go back to the conventional RLS formulation.

ITERATIVE ALGORITHMS: FIRST ORDER APPROXIMATION

- Let us consider a generic nonlinear learner for which no closed form solution of the cost function $J_N(\theta)$ exists.
- Let $\hat{\theta}_N = \arg \min_{\theta} J_N(\theta)$ be the optimal solution, i.e. $J(\hat{\theta}_N) \leq J(\theta)$
- The rationale of an iterative algorithm that searches for the optimal solution is to build a sequence $\hat{\theta}_{(t)}$ such that $J_N(\hat{\theta}_{(t)}) \leq J_N(\hat{\theta}_{(t-1)})$
- Let us consider the iteration

$$\hat{\theta}_{(t)} = \hat{\theta}_{(t-1)} - \alpha_{(t-1)} A \nabla_{\hat{\theta}} J_N(\hat{\theta}_{(t-1)})$$

where $\nabla_{\hat{\theta}} J_N(\hat{\theta})$ is the gradient of the cost function and A is a positive definite matrix

FIRST ORDER APPROXIMATION

- Since

$$\begin{aligned} J_N(\hat{\theta}_{(t)}) &= J_N(\hat{\theta}_{(t-1)} - \alpha_{(t-1)} A \nabla_{\hat{\theta}} J_N(\hat{\theta}_{(t-1)})) = \\ &J_N(\hat{\theta}_{(t-1)}) - \alpha_{(t-1)} \nabla_{\hat{\theta}}^T J_N(\hat{\theta}_{(t-1)}) A \nabla_{\hat{\theta}} J_N(\hat{\theta}_{(t-1)}) \end{aligned}$$

it follows $J_N(\hat{\theta}_{(t)}) \leq J_N(\hat{\theta}_{(t-1)})$ since

$$\alpha_{(t-1)} \nabla_{\hat{\theta}}^T J_N(\hat{\theta}_{(t-1)}) A \nabla_{\hat{\theta}} J_N(\hat{\theta}_{(t-1)}) \geq 0$$

- When the recursion

$$\hat{\theta}_{(t)} = \hat{\theta}_{(t-1)} - \alpha_{(t-1)} A \nabla_{\hat{\theta}} J_N(\hat{\theta}_{(t-1)})$$

converges, the gradient will be null so it will eventually converge to a minimum of the cost function (possibly a local minimum)

- If $A = I_n$ the recursion is the Steepest Descent algorithm

BATCH PARAMETRIC IDENTIFICATION

Let

$$\hat{\theta}_N = \hat{\theta}(D_N) = \arg \min_{\theta \in \Lambda} \widehat{\text{MISE}}_{\text{emp}}(\theta) = \arg \min_{\theta \in \Lambda} J_N(\theta)$$

be the set of parameters which minimize the **empirical risk or training error**

$$J(\theta) = \frac{\sum_{i=1}^N Q(z_i, \theta)}{N} = \frac{\sum_{i=1}^N (y_i - h(x_i, \theta))^2}{N}$$

BATCH GRADIENT DESCENT ALGORITHM

$$\hat{\theta}_{(t)} = \hat{\theta}_{(t-1)} - \alpha_{(t)} \nabla_{\hat{\theta}} J_N(\hat{\theta}) = \hat{\theta}_{(t-1)} - \alpha_{(t)} \frac{1}{N} \sum_{i=1}^N \nabla_{\hat{\theta}} Q(z_i, \hat{\theta}_{(t-1)})$$

where $\alpha_{(t)}$ stands for the learning rate.

- this iterative algorithm demands the computation at each step of the average of the gradients of the loss function over the entire training set.
- if the learning rate is small enough, the algorithm converges towards a local minimum of $J_N(\cdot)$
- Convergence speed-up is achieved by replacing the learning rate by a suitable definite positive matrix
- Note that this could be easily distributed in a Map Reduce form because of the summation form.

ONLINE GRADIENT DESCENT ALGORITHM

This is a simple modification of the batch version obtained by dropping the averaging operation and taking at random a sample $z_t \in D_N$

$$\hat{\theta}_{(t)} = \hat{\theta}_{(t-1)} - \alpha_{(t)} \nabla_{\hat{\theta}} Q(z_t, \hat{\theta}_{(t-1)})$$

- The underlying assumption is that the estimation error obtained by replacing the average with a single term will not perturb the average behaviour of the algorithm.
- This algorithm can be easily used in an adaptive online setting where no training set needs to be stored and observations are processed immediately to improve performance

ONLINE ALGORITHMS AND STOCHASTIC APPROXIMATION

- Most online algorithms are instances of stochastic approximation whose basic paradigm is

$$\hat{\theta}_{(t)} = \hat{\theta}_{(t-1)} + \alpha_{(t)} z_t$$

where $\hat{\theta}_{(t-1)}$ is the estimation of an unknown parameter θ at step $t - 1$ and z_t is some (function of) observation associated to $\hat{\theta}_{(t)}$.

- Sequential small correction so that the goal is met asymptotically.
- From stochastic approximation theory a sufficient condition for convergence is

$$\sum_{t=1}^{\infty} \alpha_{(t)} = \infty, \quad \sum_{t=1}^{\infty} \alpha_{(t)}^2 < \infty$$

thanks to the implicit averaging of the noise.

- Note however that approaches where $\alpha_{(t)}$ does not converge to zero are used to deal with non stationary configurations, e.g. to allow tracking

ONLINE LEAST-MEAN SQUARES: ADALINE

- Adaline is one of the earliest learning algorithm (Widrow and Hoff, 1960).
- It has been designed to tune the parameters of a threshold classifier

$$\hat{y} = \text{sgn}(\mathbf{x}^T \boldsymbol{\theta})$$

- It is a modification of the general algorithm where $Q(\mathbf{z}, \boldsymbol{\theta}) = (\mathbf{y} - \mathbf{x}^T \boldsymbol{\theta})^2$ instead of $Q(\mathbf{z}, \boldsymbol{\theta}) = (\mathbf{y} - \hat{y})^2$. In that way, it avoids the discontinuities of the cost function (gradient zero almost everywhere).
- The parameter set $\boldsymbol{\theta}$ is then adjusted using the **delta rule**

$$\hat{\boldsymbol{\theta}}_{(t)} = \hat{\boldsymbol{\theta}}_{(t-1)} + \alpha_{(t)}(y_t - \mathbf{x}_t^T \hat{\boldsymbol{\theta}}_{(t-1)})\mathbf{x}_t$$

- In the signal processing community this recursion is also known as Least Mean Square (LMS).

LMS

- Milestone in the history of online learning and adaptive filtering
- Simple in terms of memory requirements
- Low computational complexity $O(n)$: $2n + 1$ multiplications and $2n$ additions
- Stable when implemented with finite-precision arithmetic
- Unbiased convergence
- If the learning rate decreases, the behavior of LMS becomes closer to the batch one.
- It exists a variable step size LMS

$$\hat{\theta}_{(t)} = \hat{\theta}_{(t-1)} + (y_t - x_t^T \hat{\theta}_{(t-1)}) \frac{x_t}{\|x_t\|^2}$$

where α is time varying. It can be shown that this learning rate minimizes the a posteriori error (i.e. the error after update).

PERCEPTRON RULE

- Let us consider a binary classification task where the target can take either the value 1 or -1.
- The perceptron adapts the parameters of the threshold classifier

$$\hat{y} = \text{sgn}(\mathbf{x}^T \boldsymbol{\theta})$$

only when a misclassification occurs and according to the rule

$$\hat{\boldsymbol{\theta}}_{(t)} = \hat{\boldsymbol{\theta}}_{(t-1)} - \alpha_{(t)}(\text{sgn}(\mathbf{x}_t^T \hat{\boldsymbol{\theta}}_{(t-1)}) - y_t)\mathbf{x}_t$$

- So if $\hat{y}_t = y_t$ no update takes place. Otherwise, if a misclassification occurs, the following rule applies

$$\hat{\boldsymbol{\theta}}_{(t)} = \hat{\boldsymbol{\theta}}_{(t-1)} + 2\alpha_{(t)}y_t\mathbf{x}_t$$

- Cost function reaches its minimal value zero when all examples are properly recognized or when the vector $\hat{\boldsymbol{\theta}}$ is null. If the training set is linearly separable (i.e. a threshold element can achieve zero misclassification) the perceptron algorithm finds a linear separation with probability one.

MULTI-LAYER NETWORKS

- Stack of several layers of threshold elements, each layer using the output of the previous layer as input
- Adaline not useful in this context because by ignoring the threshold it would transform the network in a linear model
- Key idea: replace the threshold by a smooth nonlinear approximation returned by the sigmoid function

$$\text{sign}(x^T \theta) \approx \tanh(x^T \theta)$$

- The sigmoid can be made arbitrarily close to a step function by playing with the parameter
- Back-propagation algorithms used to compute the gradient to be used in the online gradient descent approach.

NEWTON METHOD: SECOND ORDER APPROXIMATION

- Iterative algorithm which uses at the t^{th} step a local quadratic approximation in the neighborhood of the current solution $\hat{\theta}_{(t-1)}$

$$\hat{J}(\theta) = J\left(\hat{\theta}_{(t-1)}\right) + \left(\theta - \hat{\theta}_{(t-1)}\right)^T \nabla J\left(\hat{\theta}_{(t-1)}\right) + \frac{1}{2} \left(\theta - \hat{\theta}_{(t-1)}\right)^T H\left(\hat{\theta}_{(t-1)}\right) \left(\theta - \hat{\theta}_{(t-1)}\right)$$

where H is the Hessian matrix of $J(\theta)$ computed in $\hat{\theta}_{(t-1)}$.

- The minimum of the approximation satisfies

$$\theta^{\min} = \hat{\theta}_{(t-1)} - H^{-1} \nabla J\left(\hat{\theta}_{(t-1)}\right)$$

where the vector $H^{-1} \nabla J\left(\hat{\theta}_{(t-1)}\right)$ is denoted as the Newton direction or the Newton step and forms the basis for the iterative strategy

$$\hat{\theta}_{(t)} = \hat{\theta}_{(t-1)} - H^{-1} \nabla J\left(\hat{\theta}_{(t-1)}\right) \quad (1)$$

ALTERNATIVES TO NEWTON METHOD

- Batch gradient-descent converges much faster when the scalar learning rate is replaced by the positive definite matrix H
- This is effective also for online learning though in a less impressive manner
- However there are several difficulties with such an approach, mainly related to the prohibitive computational demand.
- Alternative approaches, known as quasi-Newton or variable metric methods, instead of calculating the Hessian directly, and then evaluating its inverse, build up an approximation to the inverse Hessian.

KALMAN ALGORITHM

- Consider a linear model and the online iteration

$$\hat{\beta}_{(t)} = \hat{\beta}_{(t-1)} - H_{t-1}^{-1} \nabla Q(z_t, \hat{\beta}_{(t-1)})$$

- H_t is the Hessian of the online empirical cost¹

$$J_t(\hat{\beta}) = \frac{1}{2} \sum_{i=1}^t (y - x_i \hat{\beta})^2$$

- Then

$$H_t = \nabla_{\hat{\beta}}^2 J_t(\hat{\beta}) = \sum_{i=1}^t x_i^T x_i$$

$$\text{and } H_t = H_{t-1} + x_t^T x_t$$

¹In order to keep the consistency with the RLS derivation in INFOF422 we consider here x_i as a row vector of dimensionality $[1, p]$

KALMAN ALGORITHM

- If we set $V_{(t)} = H_t^{-1}$, thanks to a well known property of inverse matrices (see RLS in INFOF422 class) we boils down to the RLS algorithm
-

$$V_{(t)} = V_{(t-1)} - \frac{V_{(t-1)}x_t^T x_t V_{(t-1)}^T}{1 + x_t V_{(t-1)} x_t^T}$$
$$\hat{\beta}_{(t)} = \hat{\beta}_{(t-1)} + V_{(t)}(y_t - x_t \hat{\beta}_{(t-1)})x_t^T$$

- This algorithm converges much faster than the Delta rule and is quite easy to implement

GAUSS NEWTON ALGORITHMS

- The Kalman filter can be used only in case of linear models
- In case of nonlinear model it is possible to use the Gauss-Newton approximation

$$\begin{aligned}\frac{1}{2}\nabla_{\theta}^2(y - h(x, \theta))^2 &= \\ &= \nabla_{\theta} h(x, \theta) \nabla_{\theta}^T h(x, \theta) - (y - h(x, \theta)) \nabla^2 h(x, \theta) \\ &\approx \nabla_{\theta} h(x, \theta) \nabla_{\theta}^T h(x, \theta)\end{aligned}$$

- The Hessian of the empirical online cost is then approximated by using only the gradient

$$H_t(\theta_t) \approx \nabla_{\theta_t} h(x_t, \theta_t) \nabla_{\theta_t}^T h(x_t, \theta_t)$$

- Its inverse can be obtained using formulas similar to the ones for the linear case.

ONLINE K-MEANS

- Let us consider a set of N points $x_i \in \mathbb{R}^n, i = 1, \dots, N$
- K-means is a clustering algorithm which for a fixed $K > 0$ returns the coordinates $\theta_k \in \mathbb{R}^n$ of K centroids such that the cost function $\sum_{i=1}^N Q(x_i, \theta)$ is minimized where

$$Q(x, \theta) = \min_{k=1}^K (x - \theta_k)^2 = (x - \theta^-)^2$$

and θ^- the closest centroid to x .

- We can approximate the derivative of the loss with the derivative of the distance to the closest centroid
- Each time a new sample arrives, the centroid which is the closest to the new sample has its coordinates updated

$$\hat{\theta}_{(t)}^- = \hat{\theta}_{(t-1)}^- + \alpha_{(t)}(x_t - \hat{\theta}_{(t-1)}^-)$$

STREAMING STRUCTURAL IDENTIFICATION

STREAMING STRUCTURAL IDENTIFICATION

- Though parameter identification is crucial in machine learning, a still more relevant aspect is the choice of the learner structure.
- Model selection and validation is typically performed in a batch mode by use of cross-validation techniques
- A brute-force approach to model selection consists in estimating the generalization accuracy of the alternative models by means of a sufficiently large number of training and test phases. The candidate configuration with the smallest estimated error is then selected.
- The same computational resources are allocated to each model configuration: manifestly poor configurations are thoroughly tested to the same extent as the best ones are.

RACING ALGORITHMS

- Racing algorithms have been proposed in [8].
- The essential idea of racing methods is to evaluate a given set of candidate configurations iteratively on a stream of samples.
- As soon as enough statistical evidence is gathered against some candidate configurations, these are eliminated and the race continues only with the surviving ones.
- A racing algorithm tackles the model selection problem by using one training sample at the time and generating a sequence of nested sets of candidates
- Use of a statistical test to discard alternatives that are significantly worse
- Use of Hoeffding test (non parametric and non blocking) and Friedman test (nonparametric and blocking) ([4])

HOEFFDING FORMULA

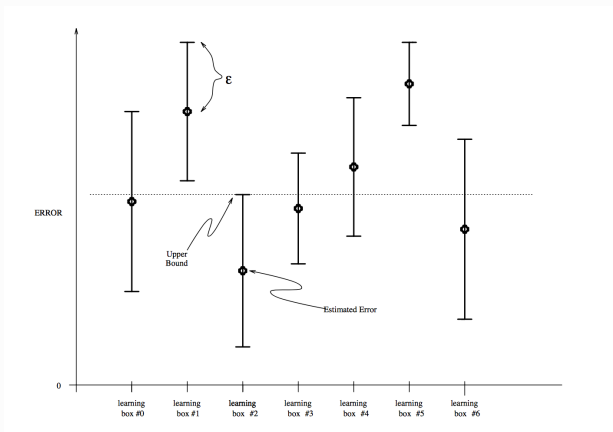
Let us consider a continuous random variable $\mathbf{z} \in Z$ such that the length of the interval is smaller than B . Given the expectation $\mu = E[\mathbf{z}]$ and the sample mean $\hat{\mu} = \sum_{i=1}^N z_i$ it can be shown that

$$\text{Prob} \{ |\hat{\mu} - \mu| > \epsilon \} < 2 \exp^{-2N\epsilon^2/B^2}$$

In other terms with confidence $1 - \delta$

$$|\hat{\mu} - \mu| < \sqrt{\frac{B^2 \log(2/\delta)}{2N}}$$

HOEFFDING RACE



From [8]

NONSTATIONARY AND EVOLVING ENVIRONMENTS

NONSTATIONARITY

A data generating process is stationary if the probabilistic process underlying the generation of data does not depend on time. In practice, a time-varying generating process is common in many settings like

- finance
- energy
- climate
- web
- environmental monitoring: sensor networks
- malware/spam/frauds
- recommendation systems

Causes:

- seasonality, periodicity effects
- changes in users' habits or preferences, changes in operating regimes
- faults
- natural evolution, drifts
- aging effects
- hidden state

TIME-SERIES VS. TIME VARYING

- A time-series is not necessarily time-varying
- For fixed values of a and b the process

$$y(t + 1) = ay(t) + by(t - 1) + \epsilon$$

generates a time-series which is time-invariant.

- If a and b change with time the series becomes non-stationary

NONSTATIONARITY

- Assumption of identical distribution in training and test is then unrealistic.
- Existence of drifts. Probability distribution changes with time
- In such setting a nonadaptive model is suboptimal and bound to become obsolete.
- Evident importance of learning in nonstationary environments
- Two main approaches [5]
 - Active methods rely on an explicit detection of the change in the data distribution
 - Passive: continuously update the model over time (without explicit detection)
- Active methods preferable in abrupt changes
- Active methods performance sensitive to the false positive (false alarm) ratio
- Passive methods more suitable in gradual drifts and recurring concepts settings.

DRIFTS

- Let us consider a classification task and a data generating process $\mathbf{z} = \langle \mathbf{x}, \mathbf{y} \rangle$ characterized by the joint distribution $p(\mathbf{x}, \mathbf{y})$ where $\mathbf{x} \in \mathbb{R}^n$ and \mathbf{y} is categorical.
- In supervised learning we are interested in modeling the conditional distribution $p(\mathbf{y}|\mathbf{x} = \mathbf{x})$ where the covariate \mathbf{x} is distributed according to $p(\mathbf{x})$.
- Simple covariate (also called virtual) shift: only the distribution $p(\mathbf{x})$ changes
- Prior probability shift: only the distribution over \mathbf{y} changes
- Real drift: $p(\mathbf{y}|\mathbf{x} = \mathbf{x})$ varies over time

Drifts can also be characterized in terms of its rate: fast (abrupt drift or concept change) or slow (gradual). Another distinction is related to the fact if it is permanent, transient (after a certain amount of time, the effect of the drift disappears) or recurrent (e.g. periodical) [5].

PASSIVE APPROACHES

- Continuous updating the model without explicitly knowing whether a concept drift occurred or not
- Neither a priori nor derived information available about concept drift
- Strategies:
 - Single learner: online learning, forgetting strategies
 - Ensemble learning: for instance $\sum_{m=1}^M w_m h_m(x)$ where the weights of the aggregation are inversely proportional to the error of the component For instance if E_m is validation error of the m th model we could set

$$w_m = \frac{\frac{1}{E_m}}{\sum_{m=1}^M \frac{1}{E_m}}$$

They manage the drift by adding or removing classifiers [7].

- Racing strategies

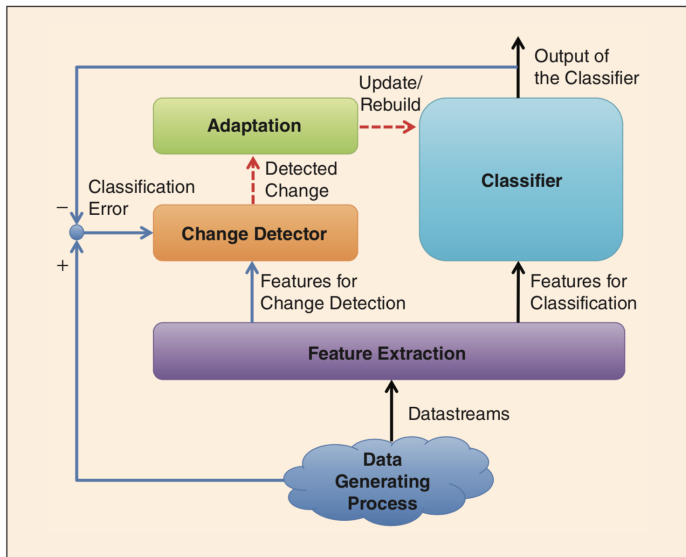
PASSIVE APPROACHES: ISSUES

- (+): no risk of false alarms (like in active)
- (-): stability-plasticity dilemma: trade off between stability (i.e. retaining existing and still relevant knowledge) and plasticity (i.e. learning new knowledge)

ACTIVE APPROACHES

- They take advantage of a priori information or automatic systems able to detect changes in either input or conditional distributions
- Use of features (marginal and conditional) assumed to be stationary in stationary configurations.
- Detect and react approach: once an explicit detection of the change is done, obsolete knowledge is discarded and an adaptation mechanism is activated.
- Change point: time instant the data-generating process changes its statistical behavior.
- Risk of false positives
- Main approaches:
 1. Hypothesis Tests
 2. Sequential Hypothesis Tests
 3. Change-Point Methods
 4. Change Detection Tests.

DETECT AND REACT



HYPOTHESIS TESTS

- Null hypothesis: no change, i.e. same mean or same distribution
- Parametric and nonparametric , univariate and multivariate approaches
- They operate on fixed length sequences: no sequential analysis
- Kolmogorov-Smirnov distance measuring the differences between cumulative density functions estimated on training samples and a window of recent data

$$KS = \max_z |F(z) - \hat{F}_N(z)|$$

where $F(z)$ is the reference cumulative distribution and $\hat{F}_N(z)$ is the empirical distribution computed on the basis of a sliding window of length N .

- Signal an alarm when the KS-test reject the null hypothesis that F and \hat{F}_N are identical.

MULTIPLE TESTING ISSUES

Multiple testing issues: if $P(\text{FP}_i) = p_i > 0$ is the probability of a false positive for the i th test then $P(\text{TP}_i) = 1 - p_i$

$$P(\cup_{i=1}^N \text{FP}_i) = 1 - P(\cap_{i=1}^N \text{TP}_i) = 1 - \prod_{i=1}^N (1 - p_i) = 1 - (1 - p_i)^N$$

the probability of at least a false positive converge to 1 for $N \rightarrow \infty$

SEQUENTIAL HYPOTHESIS TESTS

- They analyze the stream of data until they have enough statistical confidence to decide either "change" or "nonchange"
- Essential feature: number of observations is not predetermined
- Drawback: Once a decision is taken, the process is terminated.
- Examples: sequential probability ratio test (SPRT) and repeated significance test

SEQUENTIAL PROBABILITY RATIO TEST (FROM WIKIPEDIA)

- Two parametric hypothesis are given and are represented by two different distributions f_0 (null) and f_1 (alternative).
- Every time a sample is added it computes a cumulative sum of the log-likelihood ratio $S_i = S_{i-1} + \log L_i$ where

$$\log L_i = \log \frac{f_1(z_i)}{f_0(z_i)}$$

- SPRT and applies a stopping rule

$$\begin{cases} a < S_i < b : \text{continue monitoring} \\ S_i \geq b : \text{accept the alternative hypothesis} \\ S_i \leq a : \text{accept the null hypothesis} \end{cases}$$

where a and b ($a < 0 < b$) depend on the desired Type I (False positive rate) and Type II (False negative rate) errors.

CHANGE POINT METHODS

- Statistical process control techniques: they operate on a fixed set of data to take a decision about the presence of a change point τ

$$z(t) \sim \begin{cases} F_0(z) & t < \tau \\ F_1(z) & t \geq \tau \end{cases}$$

i.e. the time instant the data-generating process changes its statistical behavior

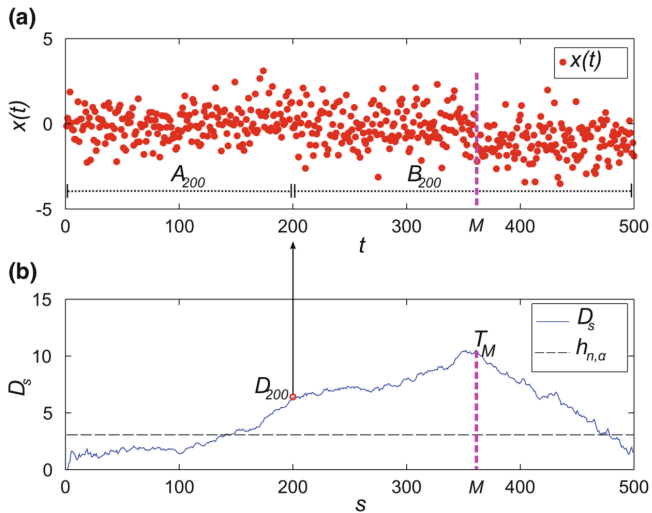
- They aim at verifying whether the sequence contains a change-point or not by analyzing **all possible partitions of the data sequence**.
- They jointly address the detection of the presence of a change and the estimation of the change time
- High computational complexity that is required to analyse all the partitions of the data sequence, which makes their use in a streaming scenario costly

CHANGE POINT METHODS

- Definition of a test statistics \mathcal{D}
- Given a time sequence \mathcal{S} of length N , for all feasible time locations $t \in \{2, \dots, N-1\}$ the sequence is partitioned in two non overlapping subsequences \mathcal{S}_1 and \mathcal{S}_2 and their dissimilarity is computed by the statistic $\mathcal{D}_t(\mathcal{S}_1, \mathcal{S}_2)$.
- The most likely change point is the one that maximises the statistics

$$t^* = \arg \max_{t \in \{2, \dots, N-1\}} \mathcal{D}_t$$

CHANGE POINT METHODS



CHANGE DETECTION TESTS

- Designed to operate in a fully sequential manner
- Reduced computational complexity
- No control of the false positive rate
- Typically relying on thresholds (Hoeffding bound, validation error).
- Difficult to set the threshold at design time; too low values induce many False Positives while too large values induce False Negatives.

CHANGE DETECTION TESTS

- CUMulative SUM control chart (CUSUM) is a sequential analysis technique designed for change detection that requires a priori parametric information about concept drift and the process generating the data.
- Given a time sequence \mathcal{S} of length N , suppose we know the parametric data distributions before (θ_0) and after (θ_1) the change.
- We compute the log-likelihood ratio

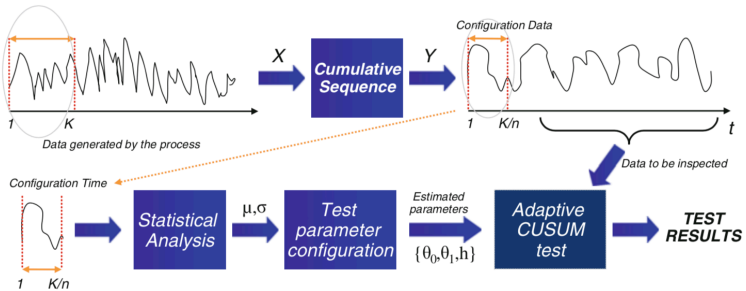
$$s(t) = \ln \frac{f(z(t), \theta_0)}{f(z(t), \theta_1)}, \quad t = 1, \dots, N$$

the cumulative sum $S(t) = \sum_{\tau=1}^t s(\tau)$ and the minimum $m(t) = \min_{\tau=1, \dots, t} S(\tau)$

- CUSUM identifies a change when the difference between the cumulative sum and its current minimum value exceeds a given threshold.
- For a threshold h the return change time is the earliest time when $g(t) = S(t) - m(t) \geq h$

ADAPTIVE CUSUM

- Transformation of the original sequence in a moving average signal Y . The averaging authorizes the Gaussian hypothesis about the distribution f .
- Use of a training set and computation of parameter θ_0 (mean and variance) under Gaussian assumption
- Definition of parameter θ_1 in the neighborhood of θ_0
- Application of conventional CUSUM



From [3]

ADVANCED TECHNIQUES

- CI-CUSUM: use of a large number of descriptive features (e.g. kurtosis, skew, information) and adoption of PCA for dimensionality reduction
- ICI-CDT: use of confidence intervals related to disjoint subsequences. As soon as the intersection of all the confidence intervals up to the current one results in an empty set, ICI-CDT detects a change.
- H-CDT: hierarchical version of ICI-CDT. Once ICI-CDT returns a change, it activates a second layer implementing a statistical test (based on additional points) to validate the choice of the first level and then reduce the risk of false positives.

ADAPTATION AFTER THE DETECTION

Three main strategies:

- Windowing: samples within a recent window are used to retrain the classifier whereas older ones are discarded
- Weighting: all samples are taken into consideration but suitably weighted according to recency or relevancy
- Biased reservoir sampling: e.g. reservoir sampling guaranteeing a uniform sampling also for very long streams. However a completely unbiased sample is undesirable since the data stream may evolve, and the vast majority of the points in the sample may represent the old history of the stream. Biased reservoir sampling techniques have been proposed in [2].

OTHER STREAMING ANALYTICS TOPICS

From [1]

- Data Stream Clustering (dynamic aspects, microclustering, snapshots)
- Data Stream Classification (ensemble of classifiers, Very Fast Decision Trees, ANNCAD, LWClass)
- Frequent Pattern Mining
- Loadshedding (adjustment of the incoming stream)
- Dimensionality reduction
- Forecasting
- Distributed mining of data streams


VFDT


- Classical decision trees assume that all training samples may be accessed simultaneously in main memory.
- DT are built by recursively replacing leaves by test nodes: the choice of the split attribute is made by comparing all the attributes and choosing the best according to some heuristics G (to be maximized).
- VFDT [6] is a decision tree for extremely large datasets, when each sample is read at most once
- Idea: store in each leaf l the sufficient statistics $n_{ijk}(l)$ representing the number of observations of class k , attribute i and value j .
- The statistics are updated each time a new sample arrive and allow to compute for each attribute the heuristic G .
- A statistical test (Hoeffding test) guarantees with a certain probability that the best split is statistically better than the second best.
- it is possible to guarantee under realistic assumptions that a VFDT trees is asymptotically arbitrarily close to the one produced by a batch learner (i.e., a learner that uses all the examples to choose a test at each node).

CLUSTREAM ALGORITHM


- The algorithm maintains a set of q micro-clusters $\mathcal{M}_1, \dots, \mathcal{M}_q$. The size q depend on memory size.
- Each microcluster (snapshot) stores data and time statistics (sum, sum of squares, number of samples).
- Whenever a new point arrives, the distance to the centroids of existing clusters is computed.
- If the point appears to be an outlier a new microcluster is created (if needed older microclusters are either deleted or merged), otherwise the closest microcluster is updated.
- If a new cluster is created, the cluster to be removed is the one with the least recent time stamp.

Issues: outlier definition, deletion criterion

 Charu C. Aggarwal.
Data Streams: Models and Algorithms (Advances in Database Systems).
Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

 Charu C. Aggarwal.
On biased reservoir sampling in the presence of stream evolution.
In Proceedings of the 32Nd International Conference on Very Large Data Bases, VLDB '06, pages 607–618. VLDB Endowment, 2006.

 C. Alippi.
Intelligence for Embedded Systems.
Springer, 2014.

 Mauro Birattari, Thomas Stützle, Luis Paquete, and Klaus Varrentrapp.
A racing algorithm for configuring metaheuristics.
In Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '02, pages 11–18, San Francisco, CA, USA, 2002.
Morgan Kaufmann Publishers Inc.

 G. Ditzler, M. Roveri, C. Alippi, and R. Polikar.

Learning in nonstationary environments: A survey.

IEEE Computational Intelligence Magazine, 10(4):12–25, Nov 2015.



Pedro Domingos and Geoff Hulten.

Mining high-speed data streams.

In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '00, pages 71–80, New York, NY, USA, 2000. ACM.



R. Elwell and R. Polikar.

Incremental learning of concept drift in nonstationary environments.

IEEE Transactions on Neural Networks, 22(10):1517–1531, Oct 2011.



O. Maron and A. Moore.

The racing algorithm: Model selection for lazy learners.

Artificial Intelligence Review, 11(1–5):193–225, 1997.