Online Incremental Learning

Chad Gueli, Amy Philip, Weiwei Tao

AMS 598

December 1, 2021

Table of Contents

Motivation

- Data volume
  - Impossible to mine the entire data at one time.
  - Can only afford constant memory per data sample.
- Change data characteristics
  - Previously learned models are invalid.
- Cost of learning
  - Model updates can be costly.
- **Example 1**: NASA's observation satellites generate billions of readings each per day.
- **Example 2**: London is said to have six million surveillance cameras, each producing a stream of images at intervals like one second.
- **Example 3**: Google receives hundred million search queries per day.

What is Online Learning

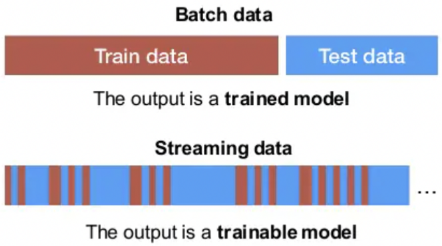

Batch data
Train data
Test data
The output is a **trained model**
Streaming data
...
The output is a **trainable model**

Figure 1: Learning from batch data and streaming data.

- Continuous model adaptation based on a constantly arriving data stream
- A single instance (or a small batch of instances) is provided to the learning algorithm at every time instant
- Very limited processing time
- Be ready to predict at any point

When to Use Online Learning

- Data is dynamic in time and rapid expanding.
- Practitioners need on-demand information in data streams.
- Model is used on stream of data WITH FEEDBACK.

Learning Styles

- Batch-wise Offline Learning
  - Deploy, track, retrain (offline), repeat
- Batch-wise Online Learning
  - Deploy, track, update (online)
- Element-wise Online Learning (Best)
  - Deploy, track and update

[1] Read et al. , 'Batch-Incremental versus Instance-Incremental Learning in Dynamic and Evolving Data'.

Applications of Online Learning

- **Fraud Detection**: Billions of financial transactions being processed every minute.
- **Internet of Things (IoT)**: Stream data analysis is able to extract useful knowledge from what is happening at real time.
- **Personalized Shopping**: Model constantly learns the real-time user behavior.

Table of Contents

Challenges of Online Learning

- Only a limited number of p training examples are allowed to be maintained $\Rightarrow$ **sampling data in a stream**
- Preservation of previously acquired knowledge and without the effect of catastrophic forgetting $\Rightarrow$ **concept drift**
- Many other challenges:
  - Continual preprocessing
  - Delayed labeling
  - $\cdots$

Challenge 1 - Sampling Data in a Stream

- Fundamental problem of data stream analysis: too much information to store or transmit.
- With limited amount of memory, one obvious approach is to store a sample.
- Two different methods:
  - Sample a fixed proportion of elements in a stream (i.e. 1/10).
  - Maintain a random sample of fixed size over all potential infinite stream.

[1] Leskovec et al. , 'Mining of Massive Datasets'.

Solution 1 - Sampling a Fixed Proportion

- **Scenario:** Search engine query stream
  - Stream of tuples: (user, query, time).
  - Question to answer: How often did a user run the same query on two different days?
  - Potential issue: Have space to store only 1/10 of query stream.
- **Solution:** Sample users using hash function
  - Pick 1/10th of users and take all their searches in the sample.
  - Use a hash function that hashes the user name or user id uniformly into 10 buckets,.

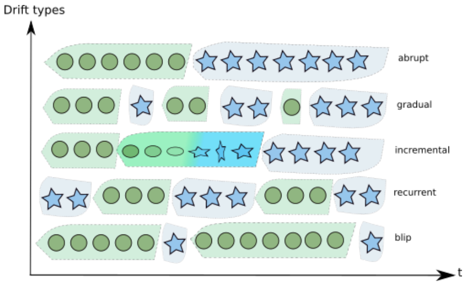
[1] Leskovec et al. , 'Mining of Massive Datasets'.

Solution 2 - Fixed-size Sampling

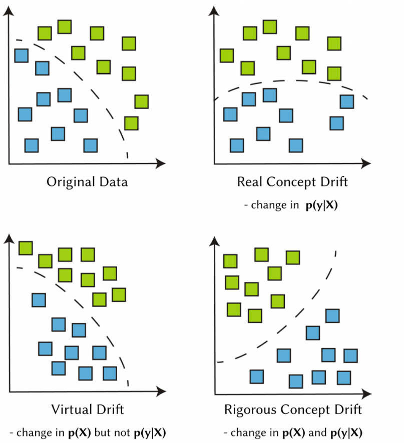
- **Scenario:** Need to main a sample S of size exactly s.
  - Don't know the length of stream in advance.
  - At any time t, each item is in the sample S with equal probability of s/n.
- **Solution:** Fixed size sample
  - Store all the first s elements of the stream to S.
  - Suppose we have seen $n-1$ elements, and now the n^{th} element arrives $(n > s)$
    - With probability s/n, pick the n^{th} element, else discard it.
    - If we picked the n^{th} element, then it replaces one of the s elements in the sample S, picked uniformly at random.

[1] Leskovec et al. , 'Mining of Massive Datasets'.

Challenge 2 - Concept Drift



(a) (Log) Dividend-Price Ratio



(b) Default Spread (%)

[1]The ravages of concept drift in stream learning applications and how to deal with it.
[2]Gözüaçık (2021), Concept learning using one-class classifiers for implicit drift detection in evolving data streams

Challenge 2 - Concept Drift

- The data is drawn from a probability distribution $p(\boldsymbol{x}, y)$, which is referred to as concept.
- Concept drift is defined as the change of the joint distribution of $\boldsymbol{x}$ and y at times t_0 and t_1 where

$$p_{t0}(\boldsymbol{x}, y) \neq p_{t1}(\boldsymbol{x}, y)$$

.

- **Virtual concept drift** involves in a change in $p(\boldsymbol{x})$ only.
- **Real concept drift** refers to the changes in $p(y|\boldsymbol{x})$, but a change in $p(\boldsymbol{x})$ can also be present.

[1] Gözüaçık (2021), Concept learning using one-class classifiers for implicit drift detection in evolving data streams

Solution - Concept Drift Detector

- **Implicit (unsupervised) methods**
  - Assumption: Changes in $p(\boldsymbol{x})$ lead to changes in $p(y|\boldsymbol{x})$.
  - Example: clustering-based methods.
- **Explicit (supervised) method**.
  - Track the prediction performance of the model and signal a drift if there is a significant decline.
  - **Sequential approaches**
  - **Statistical approaches**
  - **Window-based methods**

[1]Gözüaçık (2021), Concept learning using one-class classifiers for implicit drift detection in evolving data streams

Solution - Adaptive Sliding Window (ADWIN) Method

- **Stability-Plasticity dilemma**: Trade-off between reacting quickly to changes and having few false alarms.

- Most of the sequential approaches and statistical approaches require user to guess a cutoff parameter.

- **ADWIN**
  - Window whose size is recomputed online according to the rate of change observed.
  - Pros: Resolves the trade-off by checking change at many scales simultaneously
  - Cons: Computationally more costly (in time and memory) than other simple methods.

[1] `http://khamphouj.iiens.net/ENSIIE/ENSTA/Plot1.html`

Table of Contents

Online Learning Scheme

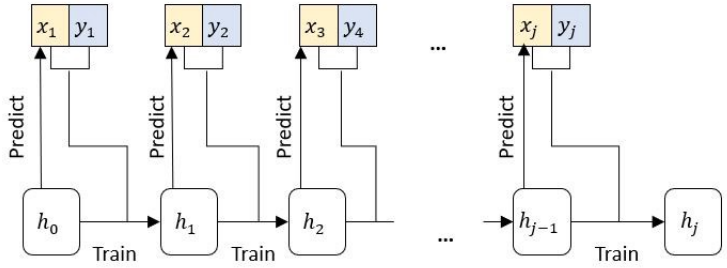


Figure 3: The online learning scheme. Data is not split into training and testing set. Instead, each model predicts subsequently one example, which is afterward used for the construction of the next model.

Online accuracy for a sequence up to the current time t is given by

$$E(S) = \frac{1}{t}\sum_{i=1}^{t}(1 - L(h_{i-1}(\boldsymbol{x_i}, y_i)))$$

[1] `https://medium.com/analytics-vidhya/incremental-online-learning-9868861db880`

Online Learning Process

- **Evaluate model**: Track the predictive performance when true labels are available.
- **Detect drift**
- **Train model**: When true labels are available or when the current model has sufficiently degraded.
- **Generate predictions**

[1]https://www.mathworks.com/help/stats/incremental-learning-overview.html

Online Linear Regression

- Goal: minimize the square loss of a linear function in an online setting.
  - Initialize $\boldsymbol{w_1} = 0$
  - Choose parameter $\alpha > 0$
  - For each round $t = 1, 2, \cdots, t$:
    - Get $\boldsymbol{x_t}$
    - Predict $\hat{y}_t = \boldsymbol{w_t} \cdot \boldsymbol{x_t}$
    - Observe y_t
    - Update $\boldsymbol{w_{t+1}} = \boldsymbol{w_t} - \alpha(\boldsymbol{w_t} \cdot \boldsymbol{x_t} - y_t)\boldsymbol{x_t}$

This algorithm is called Widrow-Hoff (WH) after its inventors, as well as Least Mean Squares (LMS).

[1] https://www.cs.princeton.edu/courses/archive/spring18/cos511/scribe_notes/0411.pdf

Online Linear Regression

- Why $\boldsymbol{w_{t+1}} = \boldsymbol{w_t} - \alpha(\boldsymbol{w_t} \cdot \boldsymbol{x_t} - y_t)\boldsymbol{x_t}$?
  - Related to performing Gradient Descent on the loss function.
  - Want $\boldsymbol{w_{t+1}}$ to achieve smaller loss on current example $(\boldsymbol{x_t}, y_t)$ and stay close to $\boldsymbol{w_t}$. Thus, to minimize their weighted sum

    $$\alpha(\boldsymbol{w_{t+1}} \cdot \boldsymbol{x_t} - y_t) + \|\boldsymbol{w_{t+1}} - \boldsymbol{w_t}\|_2^2.$$

    Solving the above optimization for w_{t+1}, we get:

    $$\boldsymbol{w_{t+1}} = \boldsymbol{w_t} - \alpha(\boldsymbol{w_t} \cdot \boldsymbol{x_t} - y_t)\boldsymbol{x_t}$$

[1]https://www.cs.princeton.edu/courses/archive/spring18/cos511/scribe_notes/0411.pdf

Online Bagging

- **Standard Bagging**
  - Bagging: random samples with replacement.
  - Each base model's training set contains each of the original training example K times, where
  $$K \sim Binomial\left(N, \frac{1}{N}\right)$$
- **Online Bagging**
  - **Issue:** Can't resample from the dataset since it is not stored
  - As $N \rightarrow$ inf, the distribution of K tends to a Poisson(1) distribution.
  - For each base model, choose the example $K \sim Poisson(1)$ times and update the base model accordingly.

[1] Oza (2001), 'Online Bagging and Boosting'

Leveraging Bagging

- Initialize base models h_m for all $m \in \{1, 2, \cdots, M\}$ and d is the latest training example to arrive.

- for $m = 1, 2, \cdots, M$ do
  - Set $w = Poisson(1)$
  - If leveraging bagging, set $w = Poisson(\lambda)$ where $\lambda > 1$.
  - Do w times $h_m = L_0(h_m, d)$
  - If ADWIN detects change in error of one of the classifier, then replace classifier with higher error with a new one.

[1] Oza (2001), 'Online Bagging and Boosting'

Table of Contents

More challenges

1. Time
2. Memory
3. Sample size

Solution - Hoeffding trees

- Pedro Domingos and Geoff Hulten of the University of Washington introduced a variant of Decision Trees called Hoeffding Trees[1]

- This is a potentially very complex tree with acceptable computational cost that solves all three challenges

- The only information needed in memory is the tree itself, and associated sufficient statistics

[1] P. Domingos and G. Hulten, Mining High-Speed Data Streams (2000), Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining

Hoeffding trees

- Key idea: Use of Hoeffding bounds[2], also known as additive Chernoff bound, which basically says - The sum (and also mean) of bounded random variables is tightly concentrated around its expected value.
- The problem of deciding exactly how many examples are necessary at each node is solved using the Hoeffding bound[3]

[2]Hoeffding, W. (1963). Probability Inequalities for Sums of Bounded Random Variables. Journal of the American Statistical Association, 58(301), 13–30. https://doi.org/10.2307/2282952

[3]O. Maron and A. Moore. Hoeffding races:Accelerating model selection search for classification and function approximation. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, Advances in Neural Information Processing Systems 6. Morgan Kaufmann, San Mateo, CA, 1994

Hoeffding bound

- The Hoeffding bound states that with probability $1 - \delta$ the true mean of a random variable of range R will not differ from the estimated mean after n independent observations by more than:

$$\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}}$$

- Useful: independent of the probability distribution generating the observations
- Drawback: conservative than distribution-dependent ones, it will take more observations to reach the same δ and ϵ)

Hoeffding tree algorithm

- The crucial decision needed to construct a decision tree is when to split a node - information gain (entropy/gini index)
- To find the best attribute to test at a given node, it may be sufficient to consider only a small subset of the training examples that pass through that node[4]
- The subset of examples considered will based on a stationary stochastic process, which means their distribution does not change over time.
- Goal: Ensure that with high probability, the attribute chosen using n examples (where n is as small as possible) is the same that would be chosen using infinite examples

[4] J. Catlett. Megainduction: Machine Learning on Very Large Databases. PhD thesis, Basser Department of Computer Science, University of Sydney, Sydney, Australia, 1991

Hoeffding tree algorithm - Pseudocode

Inputs :

- S : sequence of examples
- X : set of discrete attributes
- G(.) : heuristic measure used to choose test attributes (Gini index)
- δ : one minus the desired probability of choosing the correct attribute at any given node.

Output :

- HT : decision tree

Hoeffding tree algorithm - Pseudocode

```
HoeffdingTree($S, X, G, \delta$) :
1.  Let HT be a tree with a single leaf (the root)
2.  for all training examples do
3.      Sort example into leaf $l$ using HT
4.      Update sufficient statistics in $l$
5.      Increment $n_l$ , the number of examples seen at $l$
6.      If $n_l \ mod \ n_{min} = 0$ and examples seen at $l$ not all of same class then
7.          Compute $\overline{G_l}(Xi)$ for each attribute
8.          Let $X_a$ be attribute with highest $\overline{G_l}$
9.          Let $X_b$ be attribute with second-highest $\overline{G_l}$
10.         Compute Hoeffding bound $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}}$
11.         if $X_a \neq X_\emptyset$ and $(\overline{G_l}(X_a) - \overline{G_l}(X_b) > \epsilon$ or $\epsilon < \tau)$ then
12.             Replace $l$ with an internal node that splits on $X_a$
13.             for all branches of the split do
14.                 Add a new leaf with initialized sufficient statistics
15.             end for
16.         end if
17.     end if
```

Hoeffding tree algorithm - Pseudocode

```
HoeffdingTree($S, X, G, \delta$) :
7.      Compute $\overline{G_l}(Xi)$ for each attribute
8.      Let $X_a$ be attribute with highest $\overline{G_l}$
9.      Let $X_b$ be attribute with second-highest $\overline{G_l}$
```

- $G(X_i)$ is the splitting criterion function (information gain) and $\overline{G(X_i)}$ is its estimated value
- Let X_a be the attribute with highest observed G after seeing n examples, and X_b be the second-best attribute.
- Goal: Ensure that with high probability, the attribute chosen using n examples (where n is as small as possible) is the same that would be chosen using infinite examples

Hoeffding tree algorithm - Pseudocode

```
10.         Compute Hoeffding bound $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}}$
11.         if $X_a \neq X_\emptyset$ and $(\overline{G_l}(X_a) - \overline{G_l}(X_b) > \epsilon$ or $\epsilon < \tau)$ then
```

- The test involving τ is used for tie-breaking

Table of Contents

Adaptive Random Forest

- Adaptive Random Forest (ARF) - a new Random forests algorithm for evolving data stream classification
- Gomes et al. introduced ARF[5] in 2017
- High learning performance and low demands with respect to input preparation and hyper-parameter tuning
- Effective resampling method and adaptive operators that can cope with different types of concept drifts without complex optimizations

[5] Gomes, H.M., Bifet, A., Read, J. et al. Adaptive random forests for evolving data stream classification. Mach Learn 106, 1469–1495 (2017). https://doi.org/10.1007/s10994-017-5642-8

Adaptive Random Forest

- Dealing with evolving data streams: Instead of resetting trees as soon as drifts are detected, in ARF we use a more permissive threshold to detect warnings

- Drift/warning detection methods, weighted voting and training trees in the background before replacing existing trees

Adaptive Random Forest Algorithm

- Drift adaptation : ADWIN and Page Hinkley Test
    - If a drift is detected for the tree that originated the warning signal it is then replaced by its respective background tree.

- Parallelizing the adaptive random forests algorithm: Training a tree includes updates to the underlying drift detector, incrementing its estimate test-then-train accuracy, and, if a warning is signalled, starting a new background tree. This can be executed independently for each tree.
    - Parallel version ARF[M] is around 3 times faster than the serial version ARF[S]

- Online Bagging: Leveraging bagging. This “leverages” resampling, and has the practical effect of increasing the probability of assigning higher weights to instances while training the base models.

Table of Contents

In Python

- You shouldn't use Python
  - Use whichever language your app is developed in
- Maybe train in Python then use ONNX to transfer model to other language
- Use River
  - It is a collaboration by the creators of, and intended to replace Scikit-Multiflow and Creme
  - Still in early development (would recommend sk-multiflow) but it is not under active development
- In Java: use MOA

Serving Data

- Sometimes this is simple and you get a reasonable input every so often
- Sometimes you have BIG streaming data
  - Imagine you had 1,000,000 sensors, each transmitting a Float64 per second, then you are receiving 4MB of data per second and 3.5TB per day.[6]
  - We recommend apache AirFlow, but this is beyond the scope of the presentation

[6] Mining of Massive Datasets Ch. 4

Thank you!