# Remember the Good, Forget the Bad, do it Fast Continuous Learning over Streaming Data

**Pavol Mulinka**
Czech Technical University in Prague

**Sarah Wassermann**
INRIA Paris

**Gonzalo Marín**
Universidad de la República, IIE-FING

**Pedro Casas**
Austrian Institute of Technology

## Abstract

Continuous, dynamic and short-term learning is an effective learning strategy when operating in very fast and dynamic environments, where concept drift constantly occurs. In an on-line, stream learning model, data arrives as a stream of sequentially ordered samples, and older data is no longer available to revise earlier suboptimal modeling decisions as the fresh data arrives. Learning takes place by processing a sample at a time, inspecting it only once, and as such, using a limited amount of memory; stream approaches work in a limited amount of time, and have the advantage of performing a prediction at any point in time during the stream. We focus on a particularly challenging problem, that of continually learning detection models capable to recognize cyber-attacks and system intrusions in a highly dynamic environment such as the Internet. We consider adaptive learning algorithms for the analysis of continuously evolving network data streams, using a dynamic, variable length system memory which automatically adapts to concept drifts in the underlying data. By continuously learning and detecting concept drifts to adapt memory length, we show that adaptive learning algorithms can continuously realize high detection accuracy over dynamic network data streams.

## 1 Introduction

The application of learning models to network security problems is generally addressed as an off-line learning task, where models are trained once and then applied to the incoming stream of samples. This approach is very restrictive when dealing with highly dynamic environments, where concept drift occurs often and previous knowledge rapidly becomes obsolete. We therefore propose an on-line, stream learning approach for learning-based network security; stream learning consists of processing a data instance at a time, inspecting it only once, and as such, using a limited amount of memory. To understand the adaptability properties of different stream-learning schemes, we implement and test two popular adaptive learning algorithms on the detection of multiple types of network attacks, using real network measurements: Hoeffding Adaptive Trees (HAT) and Adaptive Random Forests (ARF) models.

We implement these models using a continually adaptive memory length, window-based approach known as ADWIN (ADaptive WINdowing) [Bifet and Gavalda, 2007]. In a nutshell, ADWIN maintains a memory window of variable size containing training samples. The algorithm automatically grows the window when no change is apparent, and shrinks it when the statistical properties of the stream changes. ADWIN automatically adjusts its window size to the optimum balance point between reaction time and small variance; for the sake of completeness, we also consider fix-length memory windows, of different lengths.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

Within the data stream mining domain, the most used evaluation scheme is known as the prequential or interleaved-test-then-train evolution. The idea is very simple, using each instance first to test an initial or bootstrapped model, and then to train or update the corresponding model, if decided by a particular rule, in general a performance-based threshold based on model accuracy. Prequential evaluation can be used to measure the accuracy of a model since the start of the evaluation, by keeping in memory all previous history of instances and evaluating the model in each new instance, but it is generally applied using sliding windows or decaying factors, which forgets previously seen instances in the model update process and focuses on those instances in current sliding window. Different from more traditional fold-cross validation, generally used in the evaluation of off-line learning models, the weakness of the prequential evaluation is in general its inherent operation on single experiment runs.

We evaluate the proposed algorithms following two new strategies to evaluate stream-based algorithms: prequential k-fold cross validation [Bifet et al., 2015] and prequential Area Under the ROC Curve (AUC) [Brzezinski and Stefanowski, 2017]. Prequential k-fold cross validation is basically an adaptation of cross-validation to the streaming setting. As evaluation metric, we simply take the model detection accuracy. Prequential AUC is an extension of the AUC metric, which is very simple to understand and informative, and provides more reliable comparisons when dealing with imbalanced data. We rely on an efficient incremental algorithm that uses a sorted tree structure with a sliding window to compute the prequential AUC using constant time and memory [Brzezinski and Stefanowski, 2017].

## 2 Related Work

The data stream learning domain has a long standing tradition and many interesting references are worth mentioning when considering the application and evaluation of stream learning models; these cover general problems related to the learning properties for stream-based algorithms [Domingos and Hulten, 2001, Stonebraker et al., 2005], the mining and evaluation processes when dealing with massive datasets [Hulten et al., 2005], the identification of model evaluation issues [Gama et al., 2009] as well as propositions of general frameworks for data streaming [Gama et al., 2013]. Of particular relevance for stream learning model evaluation are the problems of *imbalanced classes* and *concept drift*, which are extensively addressed in [Hoens et al., 2012].

## 3 Adaptive Stream Learning

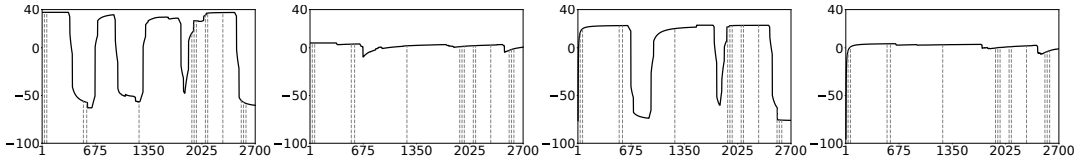
The analysis of stream-based learning models on the detection of different types of network attacks is performed on real network traffic measurements collected at the WIDE backbone network, using the MAWILab dataset for attacks labeling [Fontugne et al., 2010]. MAWILab uses a combination of four traditional anomaly detectors to partially label the collected traffic. Attacks include DDoS attacks (DDoS), Flooding attacks (Flooding), distributed network scans (Scan) and flashcrowds (Flash).

A commonly applied approach to evaluate the performance of stream-based algorithms is to benchmark them against their corresponding off-line, batch implementations. Therefore, we use as baseline the results corresponding to batch-based algorithms, including Hoeffding Tree (HT-batch) and Random Forest (RF-batch). We subtract the batch metric results from the prequential results to show the performance of stream algorithms in comparison to their batch versions.

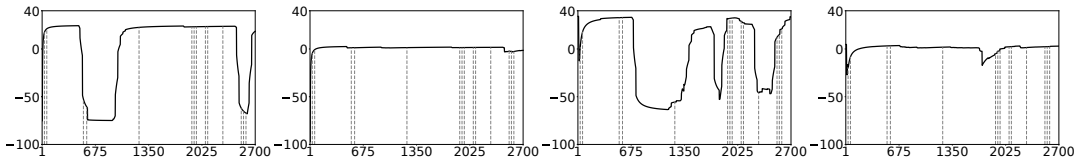
To perform the analysis in a stream-based manner, we split the traffic traces in consecutive time slots of five seconds each, and compute a set of features describing the traffic in each of these slots. In addition, each slot $i$ is assigned a label $l_i$, consisting of a binary vector $l_i \in \mathbb{R}^{4\times 1}$ which indicates at each position if attack of type $j = 1..4$ is present or not in current time slot. We compute a large number $n = 245$ of features describing a time slot $i = 1..m$, using traditional packet measurements.

### 3.1 Dynamic Memory Length with ADWIN

One of they key features of a stream-based learning model is that of continuously adapting to the changes on the underlying statistics describing current data under analysis, by periodically re-training or re-calibration. To deal with such changing data, one needs to define strategies for: firstly, detecting when changes occur; secondly, decide which data to use for a subsequent model




(a) HAT DDoS. (b) ARF DDoS. (c) HAT Flooding. (d) ARF Flooding.




(e) HAT Scan. (f) ARF Scan. (g) HAT Flash. (h) ARF Flash.

Figure 3: Prequential 10-fold cross evaluation, using detection accuracy as performance metric. Values above zero show when the stream algorithm performance exceeds the batch result. Concept drifts are marked with dashed lines.

re-calibration (or, more in general, keeping updated sufficient statistics); and finally, re-train or re-calibrate the model when significant change has been detected. Most strategies use variations of a simple sliding window approach, where a memory window containing the most recent measurements is kept and constantly updated, removing older measurements based on some specific criteria.

1: initialize window $W$
2: **for** each $t > 0$ **do**
3: $W \leftarrow W \cup \{x_t\}$ (add $x_t$ to the head of $W$)
4: **repeat** drop instances from the tail of $W$
5: **until** $\|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}\| \geq \epsilon$ holds for every
6: split of $W = W_0 \cdot W_1$
7: return $\hat{\mu}_W$
8: **end for**

Figure 1: The ADWIN algorithm.

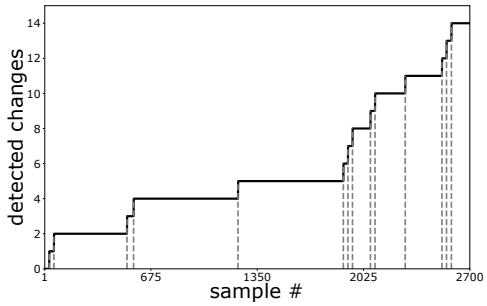
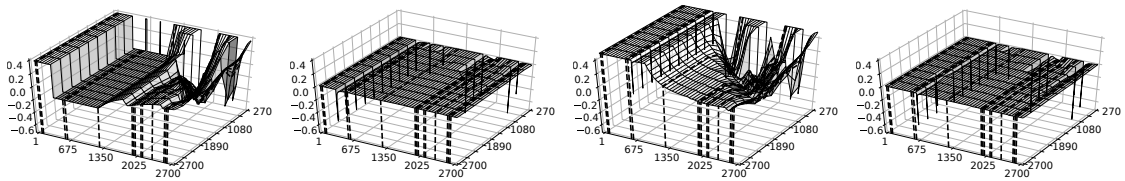



Figure 2: Page-Hinkley Concept Drift Detection.

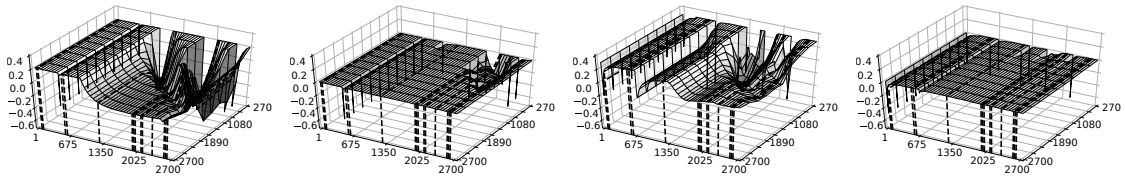
The ADWIN algorithm keeps a sliding window $W$ with the most recently observed measurements $x_i$. At each time $t$, instance $x_t$ is generated according to an unknown probability distribution $D_t$. Let $\mu_t$ be the expected value of $x_t$ when it is generated according to $D_t$. Naturally, $\mu_t$ is unknown for all $t$. Let $n$ be the length of $W$, $\hat{\mu}_W$ the (computed) average of the measurements in $W$, and $\mu_W$ the (unknown) average of $\mu_t$ for $t \in W$. The idea of ADWIN is straightforward: whenever two *large enough* sub-windows of $W$ exhibit *distinct enough* averages, we conclude that the corresponding expected values are different, and the older portion of the window is dropped. More specifically, as depicted in Figure 1, $\hat{\mu}_{W_0}$ and $\hat{\mu}_{W_1}$ are the averages of the instances in $W_0$ and $W_1$ respectively. ADWIN is therefore a simple statistical test for different distributions in $W_0$ and $W_1$, which checks whether the observed average in both sub-windows differs by more than the threshold $\epsilon$. This threshold is defined based on the global error $\delta$ of the statistical test and the lengths $n_0$ and $n_1$ of the corresponding sub-windows.

### 3.2 Concept Drift Detection

An important element to deal with when working with stream learning is the so-called concept drift; it happens when the statistical properties of the analyzed dataset abruptly shift in time. To detect concept drifts, we apply a standard statistical test for change detection known as the Page-Hinkley test, commonly used for change detection in time series analysis. Figure 2 depicts the cumulative number of changes observed in the dataset, as well as the times when those changes are detected. Concept drifts permit to better understand when and why the window size of the ADWIN algorithm changes along time.



(a) HAT DDoS. (b) ARF DDoS. (c) HAT Flooding. (d) ARF Flooding.



(e) HAT Scan. (f) ARF Scan. (g) HAT Flash. (h) ARF Flash.

Figure 4: Prequential AUC evaluation. Diagrams show the dependency of the Prequential AUC results on the evaluation window size, for each attack type. Concept drifts are marked with dashed lines.

## 4 Experiments

We evaluate the performance and behavior of the algorithms in four binary classification variants - one for each attack type, resulting in four sets of results for each tested algorithm. Figure 3 reports the performance results for each attack type, considering detection accuracy as performance metric, and using the batch-algorithms accuracy as baseline. Here we take the ADWIN version of the algorithms, using an adaptive memory size. The prequential 10-fold cross-validation shows that the Adaptive Random Forests (ARF) model rapidly converges to the batch-based accuracy results, with minimum performance variations under concept drifts. On the other hand, the HAT model does not show any apparent convergence and results tend to oscillate around the batch algorithm baseline. In the case of HAT we can appreciate the correlation between the detected concept drifts and the performance variations of the model.

Figure 4 shows the prequential AUC values along time - number of samples, considering the batch-based AUC values as baseline. Differently from ADWIN, now we take a fixed-size window length; in particular, we test the dependency of the results on the sliding window size, by setting it to a fraction of the total dataset size: {10%,20%,...,40%,...,100%}, i.e. {270,540,...,1080,...,2700}, i.e., considering 10 independent runs. As expected, we can first observe how increasing the window size smooths performance variations along time. However, as observed before for the case of ADWIN, using smaller or more fine-tuned window sizes can translate into better performance; for example, the HAT model achieves a performance gain of up to 40% (w.r.t. the baseline) when using 10%-dataset window size. Indeed, the window size allows to track long or short-term changes better, depending on the tuning.

## 5 Concluding Remarks

Dynamic and adaptive-memory-based learning looks a promising learning strategy to adapt to very dynamic environments, where concept drift occurs often. We have compared and analyzed the results from the most recent evaluation approaches for sequential data on commonly used batch-based learning algorithms and their corresponding on-line, stream-based extensions, for the specific problem of on-line network security. Prequential 10-fold cross-validation using ADWIN for dynamic window memory sizing makes a good choice for stream-based algorithm benchmarking in these applications, as it can better track transient changes and concept drifts. Similar results can be obtained by fine-tuning the memory window size, but requires additional calibration efforts. Adaptive learning algorithms can continuously realize high detection accuracy over dynamic network data streams, when dynamically adapting their learning pace and memory to changes in the underlying samples' statistics.

## References


Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM Conference*, pages 443–448, 2007.

Albert Bifet, Gianmarco de Francisci Morales, Jesse Read, Geoff Holmes, and Bernhard Pfahringer. Efficient online evaluation of big data stream classifiers. In *Proceedings of the 21th ACM SIGKDD Conference*, pages 59–68. ACM, 2015.

Dariusz Brzezinski and Jerzy Stefanowski. Prequential auc: properties of the area under the roc curve for data streams with concept drift. *Knowledge and Information Systems*, 52(2):531–562, 2017.

Pedro Domingos and Geoff Hulten. Catching up with the data: Research issues in mining data streams. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)*, 2001.

Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proceedings of the 6th ACM CoNEXT Conference*, 2010.

João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD Conference*, pages 329–338. ACM, 2009.

João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine learning*, 90(3):317–346, 2013.

Ryan Hoens, Robi Polikar, and Nitesh Chawla. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1(1):89–101, 2012.

Geoffrey Hulten, Pedro Domingos, and Laurie Spencer. *Mining massive data streams*. University of Washington, 2005.

Michael Stonebraker, Uğur Çetintemel, and Stan Zdonik. The 8 requirements of real-time stream processing. *ACM Sigmod Record*, 34(4):42–47, 2005.