Issues in Evaluation of Stream Learning Algorithms

João Gama
University of Porto
R. de Ceuta 118-6, 4050
Porto, Portugal
jgama@fep.up.pt

Raquel Sebastião
University of Porto
R. de Ceuta 118-6, 4050
Porto, Portugal
raquel@liaad.up.pt

Pedro Pereira Rodrigues
University of Porto
R. de Ceuta 118-6, 4050
Porto, Portugal
pprodrigues@fc.up.pt

ABSTRACT

Learning from data streams is a research area of increasing importance. Nowadays, several stream learning algorithms have been developed. Most of them learn decision models that continuously evolve over time, run in resource-aware environments, detect and react to changes in the environment generating data. One important issue, not yet conveniently addressed, is the design of experimental work to evaluate and compare decision models that evolve over time. There are no *golden standards* for assessing performance in non-stationary environments. This paper proposes a general framework for assessing predictive stream learning algorithms. We defend the use of *Predictive Sequential* methods for error estimate – the *prequential* error. The prequential error allows us to monitor the evolution of the performance of models that evolve over time. Nevertheless, it is known to be a pessimistic estimator in comparison to holdout estimates. To obtain more reliable estimators we need some forgetting mechanism. Two viable alternatives are: sliding windows and fading factors. We observe that the prequential error converges to an holdout estimator when estimated over a sliding window or using fading factors. We present illustrative examples of the use of prequential error estimators, using fading factors, for the tasks of: *i)* assessing performance of a learning algorithm; *ii)* comparing learning algorithms; *iii)* hypothesis testing using McNemar test; and *iv)* change detection using Page-Hinkley test. In these tasks, the *prequential* error estimated using fading factors provide reliable estimators. In comparison to sliding windows, fading factors are faster and memory-less, a requirement for streaming applications. This paper is a contribution to a discussion in the *good-practices* on performance assessment when learning dynamic models that evolve over time.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database applications—*data mining*; I.2.6 [**Artificial Intelligence**]: Learning—*classifiers design and evaluation*

KDD'09, June 28–July 1, 2009, Paris, France.

General Terms

Experimentation, Measurement, Performance

Keywords

Data Streams, Evaluation Design, Concept Drift

1. INTRODUCTION

The last twenty years or so have witnessed large progress in machine learning and in its capability to handle real-world applications. Nevertheless, machine learning so far has mostly centered on one-shot data analysis from homogeneous and stationary data, and on centralized algorithms. Most of Machine Learning and Data Mining approaches assume that examples are independent, identically distributed and generated from a stationary distribution. A large number of learning algorithms assume that computational resources are unlimited, e.g., data fits in main memory. In that context, standard data mining techniques use finite training sets and generates static models. Nowadays we are faced with tremendous amount of distributed data that could be generated from the ever increasing number of smart devices. In most cases, this data is transient, and may not even be stored permanently.

Our ability to collect data is changing dramatically. Nowadays, computers and small devices send data to other computers. We are faced with the presence of distributed sources of detailed data. Data continuously flow, eventually at high-speed, generated from non-stationary processes. Examples of data mining applications that are faced with this scenario include sensor networks, social networks, user modeling, radio frequency identification, web mining, scientific data, financial data, etc.

Most recent learning algorithms [6, 1, 10, 20, 15, 12] maintain a decision model that continuously evolve over time, taking into account that the environment is non-stationary and computational resources are limited. Examples of public available software for learning from data streams include: the `VFML` [19] toolkit for mining high-speed time-changing data streams, the `MOA` [21] system for learning from massive data sets, `Rapid-Miner` [24] a data mining system with plug-in for stream processing, etc. For illustrative purposes, consider a sensor network. Sensors are geographically distributed and produce high-speed distributed data streams. They measure some quantity of interest, and we can be interested in predicting that quantity for different time horizons. Suppose that at time t our predictive model made a prediction $\hat{y}_{t+k}$ for time $t+k$, where k is the desired horizon

forecast. Later on, at time $t + k$ the sensor measures the quantity of interest y_{t+k}, then we can estimate the loss of our prediction $L(\hat{y}_{t+k}, y_{t+k})$.

Although the increasing number of streaming learning algorithms, the metrics and the design of experiments for assessing the quality of learning models is still an open issue. The main difficulties are:

- we have a continuous flow of data instead of a fixed sample of *iid* examples;
- decision models evolve over time instead of being static;
- data is generated by non-stationary distributions instead of a stationary sample.

In a referenced paper, T. Diettrich [9] proposes a straightforward technique to evaluate learning algorithms when data is abundant: *"learn a classifier from a large enough training set and apply the classifier to a large enough test set."* Data streams are open-ended. This could facilitate the evaluation methodologies, because we have train and test sets as large as desired. The problem we address in this work is: *Is this sampling strategy viable in the streaming scenario?*

In this work we argue that the answer is *no*. Two aspects, in the emerging applications and learning algorithms that have strong impact in the evaluation methodologies are the continuous evolution of decision models and the non-stationary nature of data streams. The approach we propose is based on *sequential analysis*. Sequential analysis refers to the body of statistical theory and methods where the sample size may depend in a random manner on the accumulating data [16]. The paper is organized as follows. The next section presents an overview of the main lines presented in the literature, in learning from data streams and the most common strategies for performance assessment. In Section 3 we discuss the pros and cons of the prequential statistics in relation to the hold-out sampling strategy. Section 4 presents the main contributions of the paper, while the last section concludes the exposition, presenting the lessons learned.

2. LEARNING FROM DATA STREAMS

G. Hulten and P. Domingos [18] identify desirable properties of learning systems for efficient mining continuous, high-volume, open-ended data streams:

- require small constant time per data example;
- use fix amount of main memory, irrespective to the total number of examples;
- built a decision model using a single scan over the training data;
- generate an anytime model independent from the order of the examples;
- ability to deal with concept drift. For stationary data, ability to produce decision models that are nearly identical to the ones we would obtain using a batch learner.

From this desiderata, we can identify 3 dimensions that influence the learning process:

- *space* – the available memory is fixed;
- *learning time* – process incoming examples at the rate they arrive; and

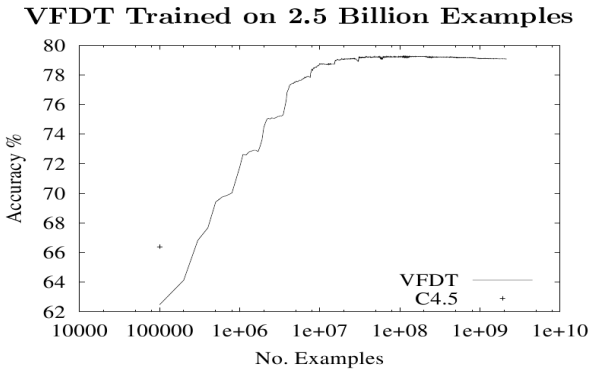


Figure 1: Performance evolution of VFDT in a web-mining problem. The accuracy (in percentage) increases for increasing number of training examples. For illustrative purposes we present the accuracy of C4.5 using the maximum number of examples that fit in memory (100k examples).

Work	Eval. Method	Mem Man	Data	Examples Train	Examples Test	Drift
VFDT	holdout	Yes	Artif	1M	50k	No
	holdout	Yes	Real	4M	267k	No
CVFDT	holdout	Yes	Artif	1M	Yes	Yes
VFDTc	holdout	No	Artif	1M	250k	No
UFFT	holdout	No	Artif	1.5M	250k	Yes
FACIL	holdout	Yes	Artif	1M	100k	Yes
MOA	holdout	Yes	Artif	1G		No
ANB	Prequen	No	Artif			Yes

Table 1: Resume of evaluation methods in stream mining literature.

- *generalization power* – how effective the model is at capturing the true underlying concept.

In this work we focus in the generalization power of the learning algorithm, although we recognize that the two first dimensions have direct impact in the generalization power of the learned model.

We are in presence of a potentially infinite number of examples. Is this fact relevant for learning? Do we need so many data points? Sampling a large training set is not enough? Figure 1 intends to provide useful information to answer these questions, showing the accuracy's evolution of VFDT [10] in a web-mining problem. One observes, in this particular problem, a rapid increase of the accuracy with the number of examples; using more than 1e+07 examples will not affect the accuracy, it will remain stable near 80%.

The fact that decision models evolve over time has strong implications in the evaluation techniques assessing the effectiveness of the learning process. Another relevant aspect is the resilience to overfitting: each example is processed once.

A brief look at the stream mining literature shows the diversity of evaluation methods. Table 1 resumes the evaluation methods found in a diverse set of well-known stream learning algorithms. The algorithms under analysis are described in [10, 20, 15, 14, 11, 21, 4] and are presented in that order.

3. EVALUATION ISSUES

A key point in any intelligent system is the evaluation methodology. Learning systems generate compact representations of what is being observed. They should be able to improve with experience and continuously self-modify their internal state. Their representation of the world is approximate. How approximate is the representation of the world? Evaluation is used in two contexts: inside the learning system to assess hypothesis, and as a wrapper over the learning system to estimate the applicability of a particular algorithm in a given problem. Three fundamental aspects are:

- What are the goals of the learning task?
- Which are the evaluation metrics?
- How to design the experiments to estimate the evaluation metrics?

For predictive learning tasks (classification and regression) the learning goal is to induce a function $\hat{y} = f(\vec{x})$. The most relevant dimension is the *generalization error*. It is an estimator of the difference between $\hat{f}$ and the unknown f, and an estimate of the loss that can be expected when applying the model to future examples.

3.1 Design of Evaluation Experiments

One aspect in the design of experiments that has not been conveniently addressed, is that learning algorithms run in computational devices that have limited computational power. For example, existing learning algorithms assume that data fits in memory; a prohibit assumption in the presence of open-ended streams. This issue becomes much more relevant when data analysis must be done *in situ*. An illustrative example is the case of sensor networks, where data flows at high-speed and computational resources are quite limited.

Very few algorithms address the bounded memory constrain. A notable exception is VFDT [10] that can save memory by freezing less promising leaves whenever memory reaches a limit. VFDT monitors the available memory and prune leaves (where sufficient statistics are stored) depending on recent accuracy. An interesting framework to evaluate learning algorithms under memory constrains appears in [21]. The author proposes 3 environments using increasing memory, for evaluating stream mining algorithms:

- Sensor environment: hundreds of Kbytes;
- Handheld computers: tens of Mbytes;
- Server computers: several Gbytes.

The memory management is more relevant for non-parametric decision models like decision trees or support vector machines because the number of free parameters evolve with the number of training examples. For other type of models, like linear models that typically depend on the number of attributes, memory management is not so problematic in the streaming context because the size of the model does not depend on the number of examples. In [21] the authors defend that general purpose streaming algorithms should be evaluated in the 3 mentioned scenarios.

In batch learning using finite training sets, cross-validation and variants (leave-one-out, bootstrap) are the standard methods to evaluate learning systems. Cross-validation is appropriate for restricted size datasets, generated by stationary distributions, and assuming that examples are independent. In data streams contexts, where data is potentially infinite, the distribution generating examples and the decision models evolve over time, cross-validation and other sampling strategies are not applicable. Research communities and users need other evaluation strategies.

3.2 Evaluation Metrics

To evaluate a learning model in a stream context, two viable alternatives, presented in the literature, are:

- Holdout an independent test set. Apply the current decision model to the test set, at regular time intervals (or set of examples). The loss estimated in the holdout is an unbiased estimator.
- Predictive Sequential: *Prequential* [7], where the error of a model is computed from the sequence of examples. For each example in the stream, the actual model makes a prediction based only on the example attribute-values. The prequential-error is computed based on an accumulated sum of a loss function between the prediction and observed values:

$$S = \sum_{i=1}^{n} L(y_i, \hat{y}_i).$$

We should point out that, in the prequential framework, we do not need to know the true value y_i, for all points in the stream. The framework can be used in situations of limited feedback, by computing the loss function and S_i for points where y_i is known.

The mean loss is given by: $M = \frac{1}{n} \times S$. For any loss function, we can estimate a confidence interval for the probability of error, $M \pm \varepsilon$, using Chernoff bound [5]:

$$\varepsilon_c = \sqrt{\frac{3 \times \bar{\mu}}{n} ln(2/\delta)},$$

where δ is a user defined confidence level. In the case of bounded loss functions, like the 0-1 loss, the Hoeffding bound [17] can be used:

$$\varepsilon_h = \sqrt{\frac{R}{2n} \ln\left(\frac{2}{\delta}\right)},$$

where R is the range of the random variable. Both bounds use the sum of independent random variables and give a relative or absolute approximation of the deviation of X from its expectation. They are independent of the distribution of the random variable.

3.2.1 *Error estimators using a single algorithm and a single dataset*

Prequential evaluation provides a learning curve that monitors the evolution of learning as a process. Using holdout evaluation, we can obtain a similar curve by applying, at regular time intervals, the current model to the holdout set. Both estimates can be affected by the order of the examples. Moreover, it is known that the prequential estimator is pessimistic: under the same conditions it will report somewhat higher errors (see Figure 2). The prequential error estimated over all the stream might be strong influenced by the first part of the error sequence, when few examples have been

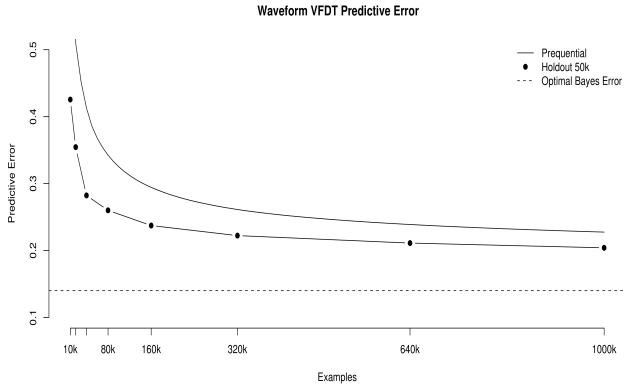


Figure 2: Comparison of error evolution as estimated by holdout and prequential strategies, in a stationary stream (Waveform data set). The learning algorithm is VFDT.

used for train the classifier. This observation leads to the following hypothesis: compute the prequential error using a forgetting mechanism, like time windows of the most recent observed errors or fading factors that weight previous errors using a decay factor. The following sections show that the error estimates using the proposed forgetting methods converge to the holdout estimator.

3.2.1.1 Illustrative Example.

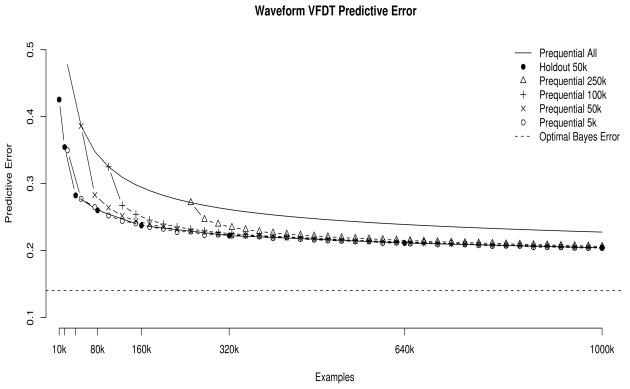
The objective of this experiment is to study convergence properties of the prequential statistics using sliding window error estimates. The learning algorithm is VFDT as implemented in VFML [19]. The experimental work has been done using the *Waveform* [3] dataset, because the Bayes-error is known: 14%. This is a three class problem defined by 21 numerical attributes.

Figure 3 (top) plots the holdout error, the prequential error, and the prequential error estimated using sliding-windows of different size. All the plots are means from 30 runs of VFDT on datasets generated with different seeds. The most relevant fact is that the window-size does not matter too much: the prequential error estimated over a sliding-window always converge fast to the holdout estimate. Figure 3 (bottom) plots the holdout error, the prequential error, the prequential error estimated using sliding-window (50k), and the prequential error estimated using fading factor (0.975). Again, the prequential error estimated using fading factor converge fast to holdout estimate.

3.3 Comparative Assessment

In this section we discuss methods to compare the performance of two algorithms (A and B) in a stream. Our goal is to distinguish between random and non-random differences in experimental results.

Let S_i^A and S_i^B be the sequences of the prequential accumulated loss for each algorithm. A useful statistic that can be used with almost any loss function is: $Q_i(A,B) = log(\frac{S_i^A}{S_i^B})$. The signal of Q_i is informative about the relative performance of both models, while its value shows the strength of the differences. In a experimental study using real data from a electrical load-demand forecast problem,



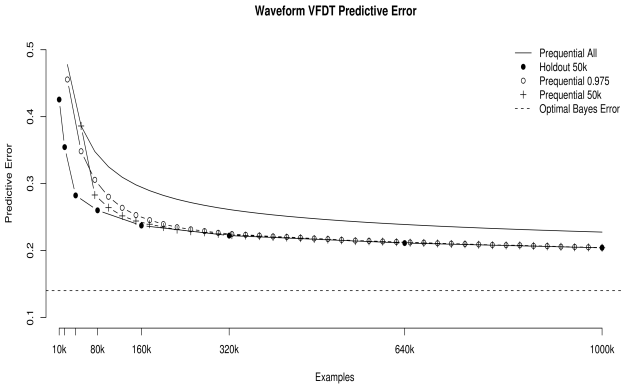


Figure 3: Comparison of error evolution between holdout, prequential and prequential over sliding windows of different sizes (top). The bottom figure present similar results using also fading factors.

plotted in Figure 4, Q_i reflects the overall tendency but exhibit long term influences and is not able to fast capture when a model is in a recovering phase. Two feasible alternatives are sliding windows, with the known problems of deciding the window-size, and fading-factors. Both methods have been used for blind adaptation, e.g. without explicit change detection, of decision models in drift scenarios [22, 23]. The formula for using fading factors with the Q_i statistic is:

$$Q_i^\alpha(A,B) = log(\frac{L_i(A) + \alpha \times S_{i-1}^A}{L_i(B) + \alpha \times S_{i-1}^B}).$$

It is interesting to observe that these two alternatives exhibit similar plots (see Figure 5).

The fading factors are multiplicative, corresponding to an exponential forgetting. At time-stamp t the weight of example $t-k$ is α^k. For example, using $\alpha = 0.995$ the weight associated with the first term after 3000 examples is $2.9E{-}7$. In general, assuming that we can ignore the examples with weights less than ϵ, an upper bound for k (e.g. the set of "important" examples) is $log(\epsilon)/log(\alpha)$. The fading factors are memoryless, an important property in streaming scenarios. This is a strong advantage over sliding-windows that require maintaining in memory all the observations inside the window.

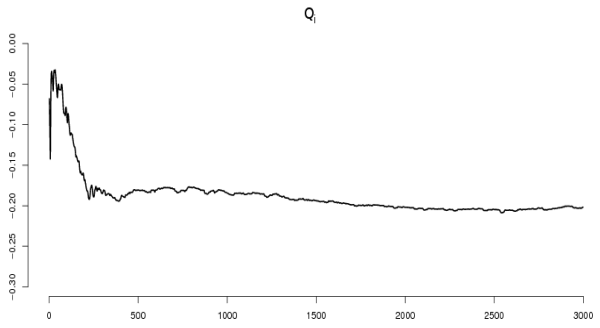


Figure 4: Comparison between two different neural-networks topologies in a electrical load-demand problem. The loss function is the *mean-squared error*. The figure plots the evolution of the Q_i statistic. The sign of Q_i is always negative, illustrating the overall advantage of one method over the other.

3.3.1 The 0-1 loss function

For classification problems, one of the most used tests is the McNemar test[1]. To be able to apply this test we only need to compute two quantities $n_{i,j}$: $n_{0,1}$ denotes the number of examples misclassified by A and not by B, whereas $n_{1,0}$ denotes the number of examples misclassified by B and not by A. The contingency table can be updated on the fly, which is a desirable property in mining high-speed data streams. The statistic $M = sign(n_{0,1} - n_{1,0}) \times \frac{(n_{0,1}-n_{1,0})^2}{n_{0,1}+n_{1,0}}$ has a χ^2 distribution with 1 degree of freedom. For a confidence level of 0.99, the null hypothesis is rejected if the statistic is greater than 6.635 [9].

3.3.1.1 Illustrative Example.

We have used the dataset SEA concepts [27], a benchmark problem for concept drift. Figure 6 (top panel) shows the evolution of the error rate of two naive-Bayes variants: a standard one and a variant that detects and relearn a new decision model whenever drift is detected. The McNemar test was performed to compare both algorithms. The bottom panel shows the evolution of the statistic test computed over all the stream. As it can be observed, once this statistic overcomes the threshold value (6.635), it never decreases below it, which is not informative about the dynamics of the process under study. Again, the problem is the long term influences verified with the Q_i statistic. It is well known, that the power of statistical tests, the probability of signaling differences where they do not exist, are highly affected by data length. Data streams are potentially unbounded, which might increase the number of Type II errors.

To overthrow this drawback, and since the fading factors are memoryless and proves to exhibit similar behaviors to sliding windows, we compute this statistic test using different windows size and fading factors. Figure 7 illustrates a comparison on the evolution of a signed McNemar statistic between the two algorithms, computed over a sliding window

[1]We do not argue that this is the most appropriate test for comparing classifiers. An in depth analysis on statistical tests to compare classifiers in batch scenario appears in [8].

Figure 5: Plot of the Q_i statistic over a sliding window of 250 examples (top). The bottom figure plots the Q_i statistic using a fading factor of $\alpha = 0.995$.

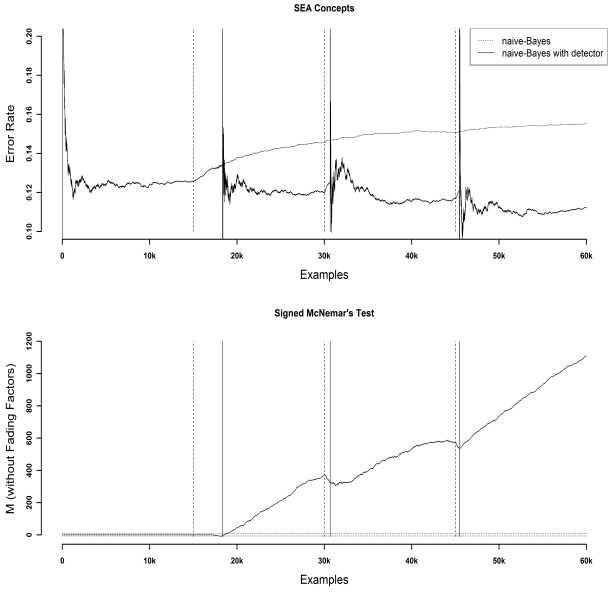


Figure 6: The evolution of signed McNemar statistic between two algorithms. Vertical dashed lines indicates drift in data, and vertical solid lines indicates when drift was detected. The top panel shows the evolution of the error rate of two naive-Bayes variants: a standard one and a variant that detect and relearn a new model whenever drift is detected. The bottom panel shows the evolution of the signed McNemar statistic computed for these two algorithms.

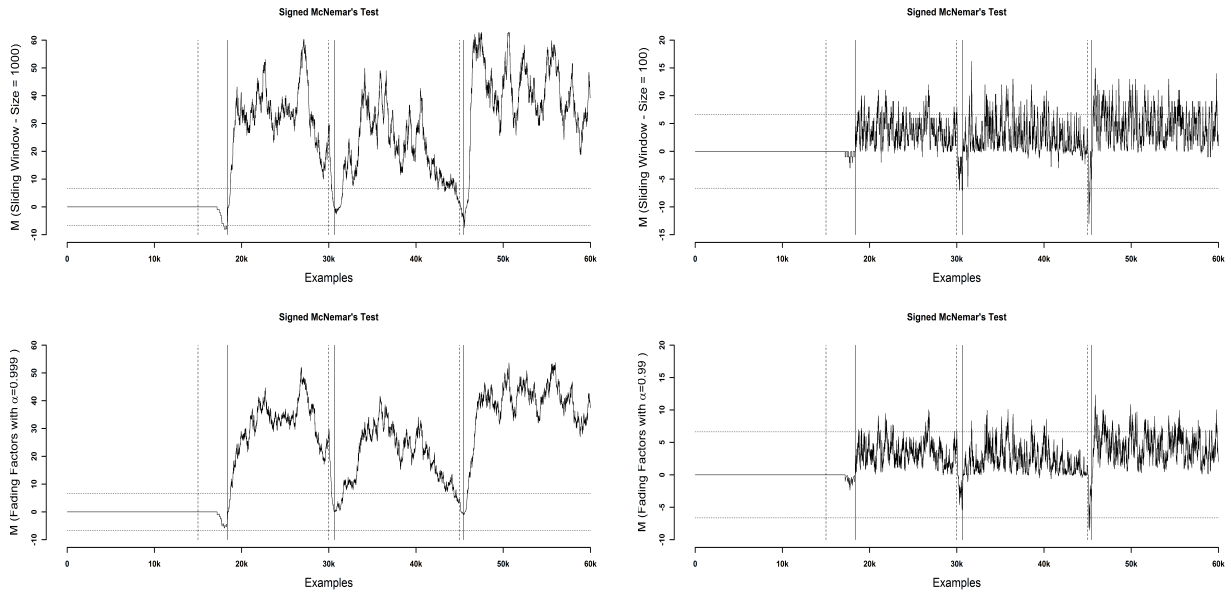


Figure 7: The evolution of signed McNemar statistic between two naive-Bayes variants. The top panel shows the evolution of the signed McNemar statistic computed over a sliding window of 1000 and 100 examples and the bottom panel shows the evolution of the signed McNemar statistic computed using a Fading Factor with $\alpha = 0.999$ and $\alpha = 0.99$, respectively. The dotted line is the threshold for a significance level of 99%. For different forgetting factors we got different results about the significance of the differences.

of 1000 and 100 examples (on the top panel) and computed using a fading factor with $\alpha = 0.999$ and $\alpha = 0.99$ (on the bottom panel). It can be observed that the statistics reject the null hypothesis almost at the same point. The use of this statistical test to compare stream-learning algorithms now shows itself feasible by applying sliding-window or fading-factors techniques. Nevertheless, these experiments points out that for different forgetting factors we got different results about the significance of the differences.

3.4 Evaluation Methodology in Non-Stationary Environments

An additional problem of the holdout method comes from the non-stationary properties of data streams. Non-stationarity or **concept drift** means that the concept about which data is obtained may shift from time to time, each time after some minimum permanence. The permanence of a concept is designated as **context** and is defined as a set of consecutive examples from the stream where the underlying distribution is stationary. Without loss of generality, we restrict this work to methods for explicit change detection because they are informative about the dynamics of the process generating data. In that case, some useful evaluation metrics include: *i)* probability of false alarms; *ii)* probability of true alarms; *ii)* delay in detection.

3.4.1 The Page-Hinkley Algorithm.

Several tests for change detection have been presented in the literature [2, 28, 22, 23, 13]. One of the most referred is the Page-Hinkley test (PHT), a sequential analysis technique typically used for monitoring change detection [26] in signal processing. It allows efficient detection of changes in the normal behavior of a process which is established by a model. The PHT is designed to detect a change in the average of a Gaussian signal [25]. This test considers a cumulative variable m_T, defined as the cumulated difference between the observed values and their mean till the current moment:

$$m_T = \sum_{t=1}^{T} (x_t - \bar{x}_T - \delta)$$

where $\bar{x_T} = 1/T \sum_{t=1}^{t} x_t$ and δ corresponds to the magnitude of changes that are allowed.

The minimum value of this variable is also computed: $M_T = min(m_t, t = 1 \ldots T)$. As a final step, 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. The threshold λ depends on the admissible false alarm rate. Increasing λ will entail fewer false alarms, but might miss or delay some changes.

3.4.1.1 Illustrative Example.

Figure 8 illustrates how PHT works. The top figure plots the trace of the prequential error of a naive-Bayes classifier (using data from the first concept of the SEA dataset [27]). A concept drift occurs at point 15000 which leads to an error increment. The PHT allows us to detect the significant increase of the error. The bottom figure represents the evolution of the statistic test PH_t and the detection threshold (λ). As it can be observed, the PH statistic test follows the increase of the error rate. The λ parameter should guarantee that the algorithm, while being resilient to false alarms, can detect and react to changes as soon as they occur, decreasing the detection delay time. Controlling this detection

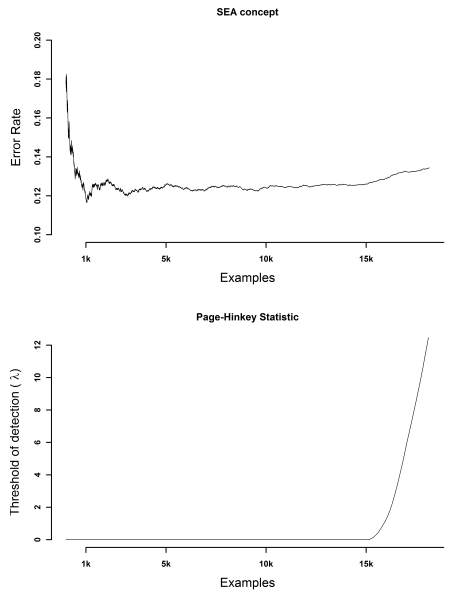


Figure 8: Experiments in SEA dataset illustrating the first drift at point 15000. The top figure shows the evolution of the naive-Bayes prequential error. The bottom figure represents the evolution of the Page-Hinkley test statistic and the detection threshold λ.

threshold parameter we establish a tradeoff between the false positive alarms and the miss detections.

As described before in this paper, the use of fading factors, as a smooth forgetting mechanism, may be an advantage in change detection algorithms. In a drift scenario, as new data is available, older observations are less useful. Using fading factors, e.g. attributing less weight to past observations, the change detection algorithm will focus in the most recent data, which in a drift scenario may lead to fast detections. For detection purposes, we monitor the evolution of the error rate of a naive-Bayes classifier (using again the SEA concepts dataset). The formula we use to embed fading factors in the Page-Hinkley test is: $m_T = \alpha \times m_{T-1} + (x_t - \hat{x}_T - \delta)$. To detect increases in the error rate (due to drifts in data) we compute the PHT, setting δ and λ parameters to 10^{-3} and 2.5, respectively. Figure 9 shows the delay time for this test: (a) without fading factors, (b) and (c) using different fading factors. The advantage of the use of fading factors in the PHT can be easily observed in this figure. The exponential forgetting results in small delay times without compromise miss detections.

We can control the rate of forgetting using different fading factors; as close to one is the α value of the fading factor the less it will forget data. We evaluate the PHT using different fading factors to assess the delay time in detection. Figure 9 (b) and c)) shows the delay time in detection of concept drifts. We have used the PHT and different fading factors ($\alpha = 1 - 10^{-5}$ and $\alpha = 1 - 10^{-4}$, respectively) to detect these changes. Table 2 presents the delay time in detection of concept drifts in the same dataset used in figure 9. As the fading factor increases, one can observe that the delay time also increases, which is consistent with the design of experiments. As close to one is the α value of the fading factor the greater is the weight of the old data, which will lead to higher delay times. The feasible values for α are between $1-10^{-4}$ and $1-10^{-8}$ (with $\alpha = 1-10^{-8}$ the delay times are not decreased and with $\alpha = 1 - 10^{-4}$ the delay time decreases dramatically but with false alarms). We may focus on the resilience of this test to false alarms and on its ability to reveal changes without miss detections. The results obtained with this dataset were very consistent and precise, supporting that the use of fading factors improves the accuracy of the Page-Hinkley test.

	Fading Factors $(1-\alpha)$					
Drifts	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	0
1st drift	1045 (1)	1609	2039	2089	2094	2095
2nd drift	654 (0)	2129	2464	2507	2511	2511
3rd drift	856 (1)	1357	1609	1637	2511	1641

Table 2: Delay times in drift scenarios using different *fading factors*. We observe false alarms only for $1-\alpha = 10^{-4}$. The number of false alarms is indicated in parenthesis.

4. MAIN CONTRIBUTIONS

The prequential method is a general methodology to evaluate learning algorithms in streaming scenarios. The contributions of this paper are:

- We show that the prequential error computed over a sliding window converges to the holdout error;
- We propose a faster and memory less approach, using fading factors, that do not require to store in memory all the required statistics in the window, and show the convergence towards the holdout estimate.
- We propose the Q statistic, a fast and incremental statistic to continuously compare the performance of two classifiers.
- We show that the use of fading factors in the McNemar test provide similar results to sliding windows.
- We show that the use of fading factors in the Page-Hinkley test achieve faster detection rates, maintaining the capacity of being resilient to false alarms when there are no drifts.

More than the technical contribution, this paper is a contribution to a discussion in the *good-practices* on performance assessment and differences in performance when learning dynamic models that evolves over time.

5. CONCLUSIONS

The main problem in evaluation methods when learning from dynamic and time-changing data streams consists of monitoring the evolution of the learning process. In this work we defend the use of Predictive Sequential error estimates using fading factors to assess performance of stream learning algorithms in presence of non-stationary data. The prequential method is a general methodology to evaluate learning algorithms in streaming scenarios. In those applications where the observed target value is available later in time, the prequential estimator can be implemented inside

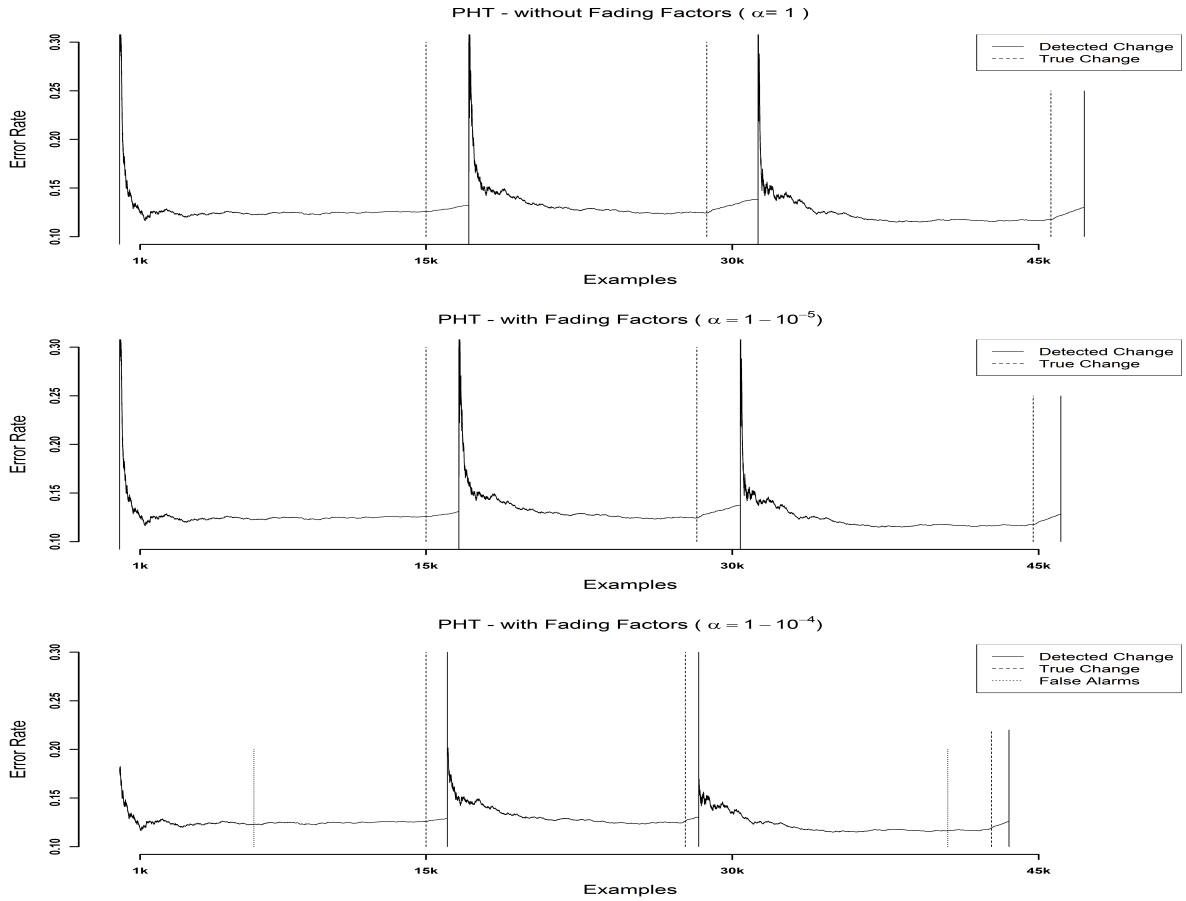


Figure 9: The evolution of the error rate and the delay times in drift detection using the Page-Hinkley test and different fading factors. The top panel shows the delay times using the PH test without fading factors. The middle and bottom panels show the delay times using fading factors with $\alpha = 1 - 10^{-5}$ and $\alpha = 1 - 10^{-4}$, respectively.

the learning algorithm. This opens interesting opportunities: the system would be capable of monitoring the evolution of the learning process itself and self-diagnosis the evolution of it. In this work we focus on loss as performance criteria. Nevertheless, other criteria, imposed by data streams characteristics, must be taken into account. Memory is one of the most important constrains. Learning algorithms run in fixed memory. They need to manage the available memory, eventually discarding parts of the required statistics or parts of the decision model. We need to evaluate the memory usage over time, and the impact in accuracy when using the available memory. Another aspect is that algorithms must process the examples as fast as (if not faster than) they arrive. Whenever the rate of arrival is faster than the processing speed, same sort of sampling is required. The number of examples processed per second and the impact of sampling in performance are other evaluation criteria. Overall, with this work, a new step forward is given in the discussion of good-practices on performance assessment of stream learning algorithms.

Acknowledgments

Thanks to the financial support given by the FEDER, the Plurianual support attributed to LIAAD and the project *Knowledge Discovery from Ubiquitous Data Streams*. The work of R. Sebastião and P. Rodrigues is supported by the Portuguese Foundation for Science and Technology (FCT) under the PhD Grants SFRH/BD/41569/2007 and SFRH/BD/29219/2006, respectively.

6. REFERENCES

[1] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Proc. of the 22nd Symposium on Principles of Database Systems*, pages 234–243. ACM Press, 2003.

[2] Michele Basseville and Igor Nikiforov. *Detection of Abrupt Changes: Theory and Applications*. Prentice-Hall Inc, 1987.

[3] C. Blake, E. Keogh, and C.J. Merz. UCI repository of Machine Learning Databases, 1999.

[4] Gladys Castillo and João Gama. Bias management of bayesian network classifiers. In A. Hoffmann, H. Motoda, and T. Scheffer, editors, *Discovery*

Science, Proceedings of 8th International Conference, pages 70–83. LNAI 3735, Springer Verlag, 2005.

[5] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sums of observations. *Annals of Mathematical Statistics*, 23:493– 507, 1952.

[6] Graham Cormode, S. Muthukrishnan, and Wei Zhuang. Conquering the divide: Continuous clustering of distributed data streams. In *ICDE*, pages 1036–1045, 2007.

[7] A. P. Dawid. Statistical theory: The prequential approach. *Journal of the Royal Statistical Society-A*, 147:278–292, 1984.

[8] Janez Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[9] T. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. Corvallis, technical report nr. 97.331, Oregon State University, 1996.

[10] Pedro Domingos and Geoff Hulten. Mining High-Speed Data Streams. In Ismail Parsa, Raghu Ramakrishnan, and Sal Stolfo, editors, *Proceedings of the ACM Sixth International Conference on Knowledge Discovery and Data Mining*, pages 71–80. ACM Press, 2000.

[11] F.J. Ferrer-Troyano, J.S. Aguilar-Ruiz, and J.C. Riquelme. Incremental rule learning and border examples selection from numerical data streams. *Journal of Universal Computer Science*, 11(8):1426–1439, 2005.

[12] Francisco Ferrer-Troyano, Jesus S. Aguilar-Ruiz, and Jose C. Riquelme. Discovering decision rules from numerical data streams. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 649–653. ACM Press, 2004.

[13] João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In Ana L. C. Bazzan and Sofiane Labidi, editors, *Advances in Artificial Intelligence - SBIA 2004*, volume 3171 of *Lecture Notes in Computer Science*, pages 286–295. Springer Verlag, October 2004.

[14] João Gama, Pedro Medas, and Ricardo Rocha. Forest trees for on-line data. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 632–636. ACM Press, 2004.

[15] João Gama, Ricardo Rocha, and Pedro Medas. Accurate decision trees for mining high-speed data streams. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 523–528. ACM Press, 2003.

[16] B. Ghosh and P. Sen. *Handbook of Sequential Analysis*. Narcel Dekker, 1991.

[17] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[18] Geoff Hulten and Pedro Domingos. Catching up with the data: research issues in mining data streams. In *Proc. of Workshop on Research issues in Data Mining and Knowledge Discovery*, 2001.

[19] Geoff Hulten and Pedro Domingos. VFML – a toolkit for mining high-speed time-changing data streams. `http://www.cs.washington.edu/dm/vfml/`. 2003.

[20] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the 7th ACM SIGKDD International conference on Knowledge discovery and data mining*, pages 97–106. ACM Press, 2001.

[21] Richard Kirkby. *Improving Hoeffding Trees*. PhD thesis, University of Waikato - New Zealand, 2008.

[22] Ralf Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281–300, 2004.

[23] I. Koychev. Gradual forgetting for adaptation to concept drift. In *Proceedings of ECAI 2000 Workshop Current Issues in Spatio-Temporal Reasoning. Berlin, Germany*, pages 101–106, 2000.

[24] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks. In *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 935–940. ACM Press, 2006.

[25] H. Mouss, D. Mouss, N. Mouss, and L. Sefouhi. Test of Page-Hinkley, an approach for fault detection in an agro-alimentary production system. In *Proceedings of the 5th Asian Control Conference*, volume 2, pages 815–818, 2004.

[26] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[27] W. Nick Street and YongSeog Kim. A streaming ensemble algorithm SEA for large-scale classification. In *Proc. seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–382. ACM Press, 2001.

[28] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23:69–101, 1996.