Mach Learn (2013) 90:317–346
DOI 10.1007/s10994-012-5320-9

On evaluating stream learning algorithms

João Gama · Raquel Sebastião ·
Pedro Pereira Rodrigues

Received: 25 November 2011 / Accepted: 18 August 2012 / Published online: 24 October 2012

Abstract Most streaming decision models evolve continuously over time, run in resource-aware environments, and detect and react to changes in the environment generating data. One important issue, not yet convincingly addressed, is the design of experimental work to evaluate and compare decision models that evolve over time. This paper proposes a general framework for assessing predictive stream learning algorithms. We defend the use of prequential error with forgetting mechanisms to provide reliable error estimators. We prove that, in stationary data and for consistent learning algorithms, the holdout estimator, the prequential error and the prequential error estimated over a sliding window or using fading factors, all converge to the Bayes error. The use of prequential error with forgetting mechanisms reveals to be advantageous in assessing performance and in comparing stream learning algorithms. It is also worthwhile to use the proposed methods for hypothesis testing and for change detection. In a set of experiments in drift scenarios, we evaluate the ability of a standard change detection algorithm to detect change using three prequential error estimators. These experiments point out that the use of forgetting mechanisms (sliding windows or fading factors) are required for fast and efficient change detection. In comparison to sliding windows, fading factors are faster and memoryless, both important requirements for streaming applications. Overall, this paper is a contribution to a discussion on best practice for

Editor: Bernhard Pfahringer.

J. Gama (✉)
LIAAD – INESC TEC and Faculty of Economics, University of Porto, Rua de Ceuta, 118-6, 4050-190, Porto, Portugal
e-mail: jgama@fep.up.pt

R. Sebastião
LIAAD – INESC TEC and Faculty of Science, University of Porto, Rua de Ceuta, 118-6, 4050-190, Porto, Portugal
e-mail: raquel.sebastiao@inescporto.pt

P.P. Rodrigues
LIAAD – INESC TEC and Faculty of Medicine, University of Porto, Rua de Ceuta, 118-6, 4050-190, Porto, Portugal
e-mail: pprodrigues@med.up.pt

performance assessment when learning is a continuous process, and the decision models are dynamic and evolve over time.

Keywords Data streams · Evaluation design · Prequential analysis · Concept drift

1 Introduction

The last twenty years or so have witnessed large progress in machine learning and 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 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, for example, data fits in main memory. In that context, standard data mining techniques use finite training sets and generate static models. Nowadays we are faced with a tremendous amount of distributed data that can 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 flows, 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.

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 are interested in predicting that quantity for different time horizons. Assume that at time t our predictive model makes 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}. With a delay k we can estimate the loss of our prediction $L(\hat{y}_{t+k}, y_{t+k})$.

Recent learning algorithms such as Cormode et al. (2007), Babcock et al. (2003), Rodrigues et al. (2008) for clustering; Domingos and Hulten (2000), Hulten et al. (2001), Gama et al. (2003), Liang et al. (2010) for decision trees; Ferrer-Troyano et al. (2004), Gama and Kosina (2011) for decision rules; Li et al. (2010), Katakis et al. (2010) in change detection; Giannella et al. (2003), Chi et al. (2006) in frequent pattern mining, etc., maintain a model that continuously evolves over time, taking into account that the environment is non-stationary and computational resources are limited. Examples of publicly available software for learning from data streams include: the `VFML` toolkit (Hulten and Domingos 2003) for mining high-speed time-changing data streams, the `MOA` system (Bifet et al. 2010a) for learning from massive data sets, `Rapid-Miner` (Mierswa et al. 2006) a data mining system with plug-in for stream processing, etc.

Although there are an 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 i.i.d. examples;
– decision models evolve over time instead of being static;
– data is generated by non-stationary distributions instead of a stationary sample.

Table 1 Differences between batch and streaming learning that may affect the way evaluation is performed. While batch learners build static models from finite, static, i.i.d. data sets, stream learners need to build models that evolve over time, being therefore dependent on the order of examples, are generated from a continuous non-stationary flow of non-i.i.d data

	Batch	Stream
Data size	Finite data set	Continuous flow
Data distribution	i.i.d.	Non-i.i.d.
Data evolution	Static	Non-stationary
Model building	Batch	Incremental
Model stability	Static	Evolving
Order of observations	Independent	Dependent

The design of experimental studies is of paramount importance (Japkowicz and Shah 2011). It is a necessary condition, although not sufficient, to allow *reproducibility*, that is the ability of an experiment or study to be accurately replicated by someone else working independently. Dietterich (1996) 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 main differences in evaluating stream learning algorithms as opposed to batch learning algorithms are sketched in Table 1.

To cope with this stream evaluation scenario, 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 (Ghosh and Sen 1991). This paper is a substantial extension of a previous one (Gama et al. 2009), published in a top-level data mining conference, where these issues were firstly enunciated.[1] More than the technical contribution, this paper is a contribution to a discussion on best practice for performance assessment and differences in performance when learning dynamic models that evolve over time.

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

- We prove that, for consistent learning algorithms and an infinite number of examples generated by a stationary distribution, the holdout estimator, the prequential error and the prequential error estimated over a sliding window or using fading factors, converge to the Bayes error;
- We propose a faster and memoryless approach, using fading factors, that does not require to store all the required statistics in the window;
- We propose the Q statistic, a fast and incremental statistic to continuously compare the performance of two classifiers;

[1] Nowadays, the software MOA (Bifet et al. 2010a) implements most of the evaluation strategies discussed here.

- We show that the use of fading factors in the McNemar test provide similar results to sliding windows;
- We show that the proposed error estimates can be used for concept drift detection;
- We show that performing the Page-Hinkley test over the proposed error estimators improves the detection delay, although at the risk of increasing false alarms.

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 Sect. 3 we discuss the advantages and disadvantages of the prequential statistics in relation to the hold-out sampling strategy. We propose two new prequential error rate estimates: prequential sliding windows and prequential fading factors, and present convergence properties of these estimators. In Sect. 4 we discuss methods to compare the performance of two learning algorithms. Section 5 presents a new algorithm for concept drift detection based on sliding windows and fading factors error estimators. The last section concludes the exposition, presenting the lessons learned.

2 Learning from data streams

Hulten and Domingos (2001) identify desirable properties of learning systems for efficiently mining continuous, high-volume, open-ended data streams:

- require small constant time per data example;
- use fixed 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 these 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
- *generalization power*—how effective the model is at capturing the true underlying concept.

In this work we focus on 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 the 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 (Domingos and Hulten 2000) 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 ten million 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.

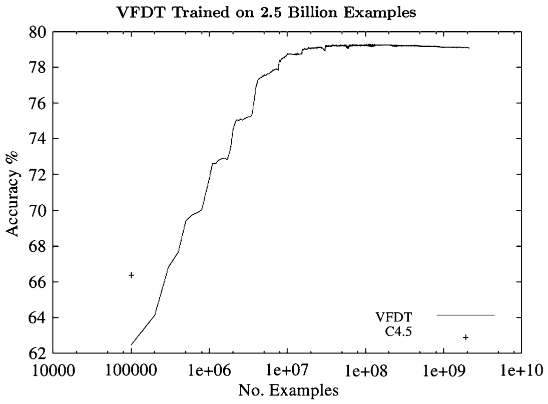

Fig. 1 Performance evolution of VFDT in a web-mining problem. The accuracy (in percentage) increases for increasing number of training examples. For illustrative purposes we present the accuracy of C4.5 using the maximum number of examples that fit in memory (100k examples)

3 Design of evaluation experiments

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?
- How to design the experiments to compare different approaches?

3.1 Experimental setup

For predictive learning tasks (classification and regression) the learning goal is to induce a function $\hat{y} = \hat{f}(\mathbf{x})$ that approximates f with arbitrary precision. 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.

One aspect in the design of experiments that has not been convincingly addressed is that learning algorithms run in computational devices that have limited computational power. For example, many existing learning algorithms assume that data fits in memory; a prohibitive 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 constraint. A notable exception is VFDT (Domingos and Hulten 2000) 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 constraints appears in Kirkby (2008). 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 types 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. Kirkby (2008) proposes that general streaming algorithms should be evaluated in the 3 mentioned scenarios. A recent work, Bifet et al. (2010b) proposes the use of RAM-Hours as an evaluation measure of the resources used by streaming algorithms. A GB of RAM deployed for 1 hour defines one RAM-Hour. It is based on rental cost options of cloud computing services.

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

Suppose a sequence of examples in the form of pairs (x_i, y_i). For each example, the actual decision model predicts $\hat{y}_i$, that can be either True ($\hat{y}_i = y_i$) or False ($\hat{y}_i \neq y_i$). For each point i in the sequence, the error-rate is the probability of observe False, p_i. For a set of examples, the error (e_i) is a random variable from Bernoulli trials. The Binomial distribution gives the general form of the probability for the random variable that represents the number of errors in a sample of examples: $e_i \sim \text{Bernoulli}(p_i) \Leftrightarrow \text{Prob}(e_i = \mathit{False}) = p_i$.

Definition In the PAC learning model (Kearns and Vazirani 1994) a learner is called consistent if for a sufficient large number of independent examples generated by a stationary distribution, it outputs a hypothesis with error arbitrarily close to the Bayes error ($B + \epsilon$) with at least probability $1 - \delta$:

$$\text{Prob}(p_i - B < \epsilon) \geq 1 - \delta.$$

As a matter of fact, if the distribution of the examples is stationary and the examples are independent, the error rate of a consistent learning algorithm (p_i) will decrease when the number of training examples, i, increases. With probability equal to or greater than $1 - \delta$, for an infinite number of training examples, the error rate will tend to the Bayes error (B). Which means that for every real number $\epsilon > 0$, there exists a natural number N_1 such that for every $i > N_1$ we have, with probability equal to or greater than $1 - \delta$, $|p_i - B| < \epsilon$:

$$\forall \epsilon > 0,\ \exists N_1 \in \mathbb{N} : \ \forall i > N_1 \quad |p_i - B| < \epsilon.$$

For example, Duda and Hart (1973) prove that the k-nearest neighbor algorithm is guaranteed to yield an error rate no worse than the Bayes error rate as data approaches infin-

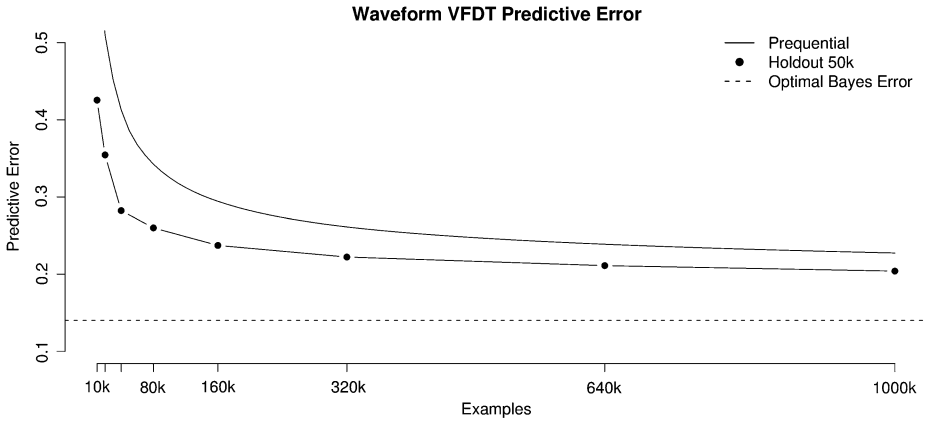

Fig. 2 Comparison of error evolution as estimated by holdout and prequential strategies, in a stationary stream (waveform data set). The learning algorithm is VFDT

ity. Bishop (1995) presents similar proofs for feed-forward neural networks, and Mitchell (1997) for decision trees. The following mathematical proofs that appear in this paper apply to consistent learners.

To evaluate a learning model in a stream context, two viable alternatives, presented in the literature, are the predictive sequential method and holdout an independent test set. Figure 2 shows a comparison of error evolution as estimated by these two strategies on the waveform data set problem, for the VFDT algorithm.

In the holdout evaluation, the current decision model is applied to a test set at regular time intervals (or set of examples). For a large enough holdout, the loss estimated in the holdout is an unbiased estimator.

Definition In a holdout test set with M examples, the 0-1 loss is computed as:

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

Theorem (Limit of the holdout error) *For consistent learning algorithms, and for large enough holdout sets, the limit of the loss estimated in the holdout is the Bayes error*: $\lim_{i\to\infty} H_e(i) = B$.

The proof is in Appendix B.1

In *predictive sequential* (or *prequential*) (Dawid 1984) 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. 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.

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

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

Theorem (Limit of the prequential error) *For consistent learning algorithms, the limit of the prequential error is the Bayes error*: $\lim_{i\to\infty} P_e(i) = B$.

The proof is in Appendix B.2.

From the Hoeffding bound (Hoeffding 1963), one can ensure that with, at least, a probability of 95 %, the prequential error estimated over all the stream converges to its average μ with an error of 1 %, for a sample of size of (at least) 18444 observations. The proof can be found in Appendix A.

3.2.1 Error estimators using a forgetting mechanism

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 Fig. 2). The prequential error estimated over all the stream might be strongly influenced by the first part of the error sequence, when few examples have been used for train the classifier. Incremental decision models evolve overtime, improving their performance. The decision model used to classify the first example is different from the one used to classify the hundredth instance. This observation leads to the following hypothesis: compute the prequential error using a forgetting mechanism. This might be achieved either using a time window of the most recent observed errors or using fading factors. Considering a sliding window of size infinite or a fading factor equal to 1, these forgetting estimators equal the prequential estimator.

Error estimators using sliding windows Sliding windows are one of the most used forgetting strategies. They are used to compute statistics over the most recent past.

Definition The prequential error is computed, at time i, over a sliding window of size w ($\{e_j \mid j \in]i-w, i]\}$) as:

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

Theorem (Limit of the prequential error computed over a sliding window) *For consistent learning algorithms, the limit of the prediction error computed over a sliding window of (large enough[2]) size w is the Bayes error*: $\lim_{i\to\infty} P_w(i) = B$.

The proof is in Appendix B.3.

[2]It is necessary to consider a window with large enough size in order to achieve an unbiased estimator of the average.

Lemma *The prequential error estimator, $P_e(i)$, is greater than or equal to the prequential error computed over a sliding window, $P_w(i)$, considering a sliding window of large enough size $w \ll i$: $P_e(i) \geq P_w(i)$.*

The proof is in Appendix B.3.

For example, using the Hoeffding bound, with a probability of 95 %, the $P_e(i)$ converges to its average μ with an error of 1 % for $i = 18444$. Taking into account the previous lemma, one can work out that at observation $i = 18444$, the $P_w(i)$ will be smaller than or equal to the $P_e(i)$. And so forth, the $P_w(i)$ will approximate the average μ with an even smaller error.

Error estimators using fading factors Another approach to discount older information across time consists of using fading factors. The *fading sum* $S_{x,\alpha}(i)$ of observations from a stream x is computed at time i, as:

$$S_\alpha(i) = x_i + \alpha \times S_\alpha(i-1)$$

where $S_\alpha(1) = x_1$ and α $(0 \ll \alpha \leq 1)$ is a constant determining the forgetting factor of the sum, which should be close to 1 (for example 0.999). This way, the *fading average* at observation i is then computed as:

$$M_\alpha(i) = \frac{S_\alpha(i)}{N_\alpha(i)} \tag{1}$$

where $N_\alpha(i) = 1 + \alpha \times N_\alpha(i-1)$ is the corresponding *fading increment*, with $N_\alpha(1) = 1$. An important feature of the fading increment is that:

$$\lim_{i\to+\infty} N_\alpha(i) = \frac{1}{1-\alpha}.$$

This way, at each observation i, $N_\alpha(i)$ gives an approximated value for the weight given to recent observations used in the fading sum.

Definition The prequential error computed at time i, with fading factor α, can be written as:

$$P_\alpha(i) = \frac{\sum_{k=1}^{i} \alpha^{i-k} L(y_k, \hat{y}_k)}{\sum_{k=1}^{i} \alpha^{i-k}} = \frac{\sum_{k=1}^{i} \alpha^{i-k} e_k}{\sum_{k=1}^{i} \alpha^{i-k}}, \quad \text{with } 0 \ll \alpha \leq 1.$$

Theorem (Limit of the prequential error computed with fading factors) *For consistent learning algorithms, the limit of the prequential error computed with fading factors is approximately the Bayes error*: $\lim_{i\to\infty} P_\alpha(i) \approx B$.

The proof is in Appendix B.4.

Furthermore, the prequential estimator computed using fading factors, $P_\alpha(i)$, will be lower than the prequential error estimator, $P_e(i)$. In Sect. 3.3 we present the pseudo-code for the three feasible alternatives to compute the prequential error previously introduced.

The proofs of the previous theorems assume the stationarity of the data and the independence of the training examples. The main lesson is that any of these estimators converge, for an infinity number of examples, to the Bayes error. All these estimators can be used in any experimental study. However, these results are even more relevant when dealing with data with concept drift. Indeed, the above theorems and the memoryless advantage support the use of prequential error estimated using fading factors to assess performance of stream learning algorithms in presence of non-stationary data. We address this topic in Sect. 5.

Illustrative example The objective of this experiment is to illustrate the demonstrated convergence properties of the error estimates using the strategies described above. The learning algorithm is VFDT as implemented in MOA (Bifet et al. 2010a). The experimental work has been done using the *Waveform* and the *LED* (Asuncion and Newman 2007) datasets, because the Bayes-error is known: 14 % and 26 %, respectively. We also use the *RandomRBF* (RBF) and *Random Tree* (RT) datasets available in MOA. The Waveform stream is a three class problem defined by 21 numerical attributes, the LED stream is a ten class problem defined by 7 boolean attributes, the RBF and the RT streams are two-class problems defined by 10 attributes.

Figure 3 plots the holdout error, the prequential error, the prequential error estimated using sliding-windows of different sizes and the prequential error estimated using different fading factors. All the plots are means from 10 runs of VFDT on datasets generated with different seeds. The most relevant fact is that the window-size and the fading factor does not matter too much: the prequential error estimate using forgetting mechanisms always converges fast to the holdout estimate.

3.3 Pseudo-code of algorithms for prequential estimation

Algorithms 1, 2 and 3 present the pseudo-code for the update rules in prequential error estimation using all history, sliding windows and fading factor respectively. The implementation of sliding windows (Algorithm 2) for prequential error estimation use circular vectors. Exponential histograms (Datar et al. 2002) might be used in the case of very large windows. All the algorithms require constant space and constant update time.

3.4 Discussion

All the error estimators discussed so far apply to stationary processes. Under this constraint, the first relevant observation is: *The prequential error estimate over all the history is pessimistic*. It is a pessimistic estimate because the decision model is not constant and evolves over time. The decision model used to classify the nth instance is not the same used to classify the first instance. Under the stationarity assumption, the performance of the decision model improves with more labeled data. The second relevant observation is that the proposed forgetting mechanism (sliding windows and fading factors) preserve convergence properties similar to the holdout error estimator. The use of these error estimators in non-stationary and evolving data is discussed in Sect. 5.

Error estimators using sliding windows are additive; the contribution of a term is constant, while it is inside the window and the forgetting mechanism (when the term is out of the window) is abrupt. The key difficulty is how to select the appropriate window size. A small window can assure fast adaptability in phases with concept changes; while large

Algorithm 1 Update Prequential Error Estimator

```
Require: $e_i$ {/* Loss at example $i$ */}
Ensure: Error estimator $P_e(i)$
  $P_e(0) \leftarrow 0$ {/* Initialize the error estimate */}
  ...
  $P_e(i) \leftarrow \frac{e_i+(i-1)*P_e(i-1)}{i}$ {/* Update the error estimate */}
  ...
```

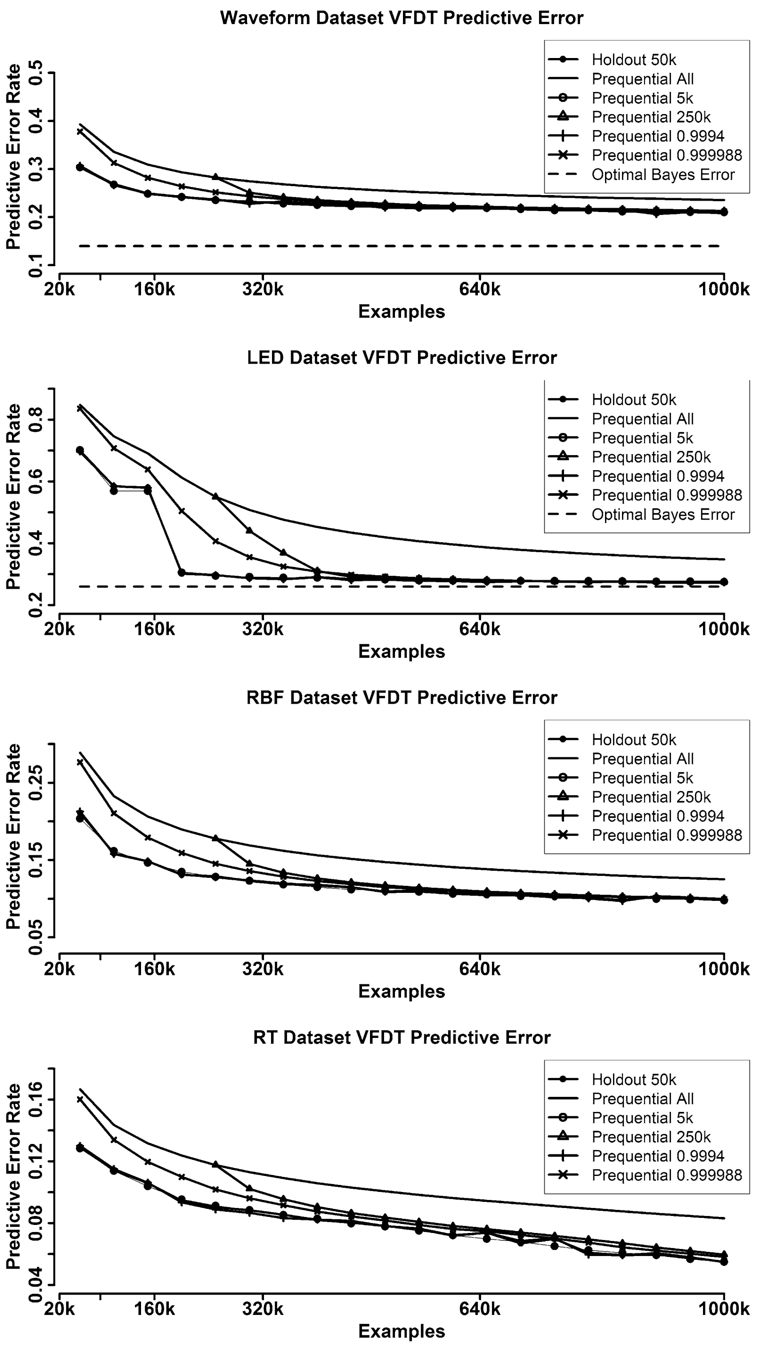

Fig. 3 Comparison of error evolution between holdout, prequential, prequential over sliding windows of different sizes and prequential using fading factors

Algorithm 2 Update Prequential Error Estimator in sliding window

```
Require: Window size: $w$
Require: $e_i$ {/* Loss at example $i$ */}
Ensure: Window error estimator $P_w(i)$ ($i > w$)
  {/* Initialization */}
  $S \leftarrow 0$ {/* Initialize the error estimate */}
  $E[1:w] \leftarrow 0$ {/* Initialize the vector of size $w$ */}
  ...
  {/* Update the error estimate */}
  $p \leftarrow ((i-1) \bmod w) + 1$
  $S \leftarrow S - E[p] + e_i$
  $E[p] \leftarrow e_i$
  $P_w(i) = \frac{S}{\min(w,i)}$
  ...
```

Algorithm 3 Update rule for Prequential error estimator using fading factors

```
Require: Fading factor $\alpha$ ($0 \ll \alpha \leq 1$)
Require: $e_i$ {/* Loss at example $i$ */}
Ensure: Fading error estimator $P_\alpha(i)$
  $S_\alpha(0) \leftarrow 0$; $N_\alpha(0) \leftarrow 0$ {/* Initialize the error estimate */}
  ...
  {/* Update the error estimate */}
  $S_\alpha(i) \leftarrow e_i + \alpha * S_\alpha(i-1)$
  $N_\alpha(i) \leftarrow 1 + \alpha * N_\alpha(i-1)$
  $P_\alpha(i) = \frac{S_\alpha(i)}{N_\alpha(i)}$
  ...
```

windows produce lower variance estimators in stable phases, but cannot react quickly to concept changes. The fading factors are multiplicative, corresponding to exponential forgetting. A term always contributes to the error, smoothly decreasing has more terms are available.

Error estimates using sliding windows and fading factors require user defined parameters: the window size and the fading factor. It is known that estimating unknown parameters over larger sample sizes (and stationary environments) generally leads to increased precision. Nevertheless, after some point in sample size, the increase in precision is minimal, or even non-existent. The first question we need to discuss is: *What are the admissible values for the window size?* The estimator of the error rate, a proportion is given by $\hat{p} = \frac{\#e}{n}$, where $\#e$ is the number of errors. When the observations are independent, this estimator has a (scaled) binomial distribution. The maximum variance of this distribution is $0.25/n$, which occurs when the true parameter is $\hat{p} = 0.5$. For large n, the distribution of $\hat{p}$ can be approximated by a normal distribution. For a confidence interval of 95 %, the true p is inside of the interval $\{\hat{p} - 2\sqrt{0.25/n}; \hat{p} + 2\sqrt{0.25/n}\}$. If this interval cannot be larger than ϵ, we can solve the equation $4\sqrt{0.25/n}$ for n. For example, a window of $n = 1000$ implies $\epsilon = 3$ %, while for $\epsilon = 1$ % a sample size of $n = 10000$ is required. With respect to fading factors, the equivalent question is: *What are the admissible values for the fading factor?* At time-stamp t the weight of example $t - k$ is α^{i-k}. For example, using $\alpha = 0.995$ the weight associated with the first term after 3000 examples is $2.9E{-}7$. Assuming that we can ignore the examples with weights less than ϵ, an upper bound for k (the set of "important" examples) is

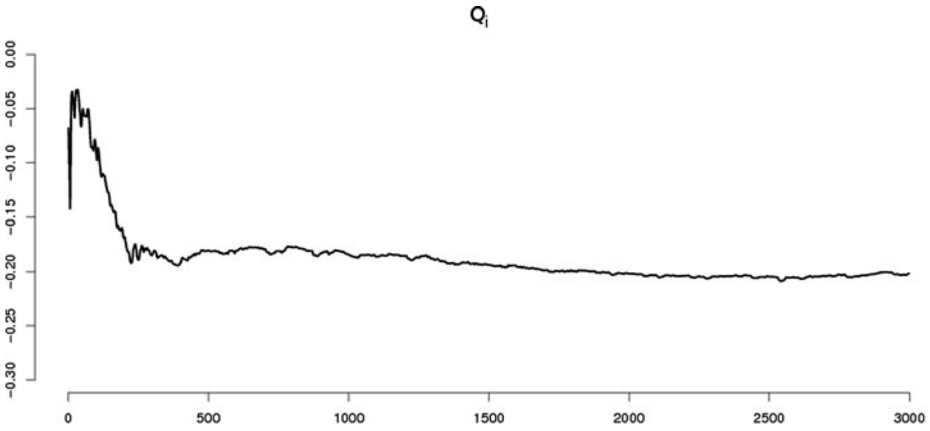

Fig. 4 Comparison between two different neural-networks topologies in a electrical load-demand problem. The loss function is the *mean-squared error*. The figure plots the evolution of the $Q_i = \log(A/B)$ statistic. The sign of Q_i is always negative, illustrating the overall advantage of method B over method A

$\log(\epsilon)/\log(\alpha)$. 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.

4 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. Q_i is symmetric, given that $\log(A/B) = -\log(B/A)$. In an experimental study using real data from an electrical load-demand forecast problem, plotted in Fig. 4, Q_i reflects the overall tendency but exhibits 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 model adaptation without explicit change detection, in drift scenarios (Klinkenberg 2004; Koychev 2000). The formula for using fading factors with the Q_i statistic is:

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

Figure 5 plots the evolution of the Q_i statistic computed using a sliding window and a fading factor. It is interesting to observe that the Q_i statistic captures changes in the relative performance of the two learning algorithms.

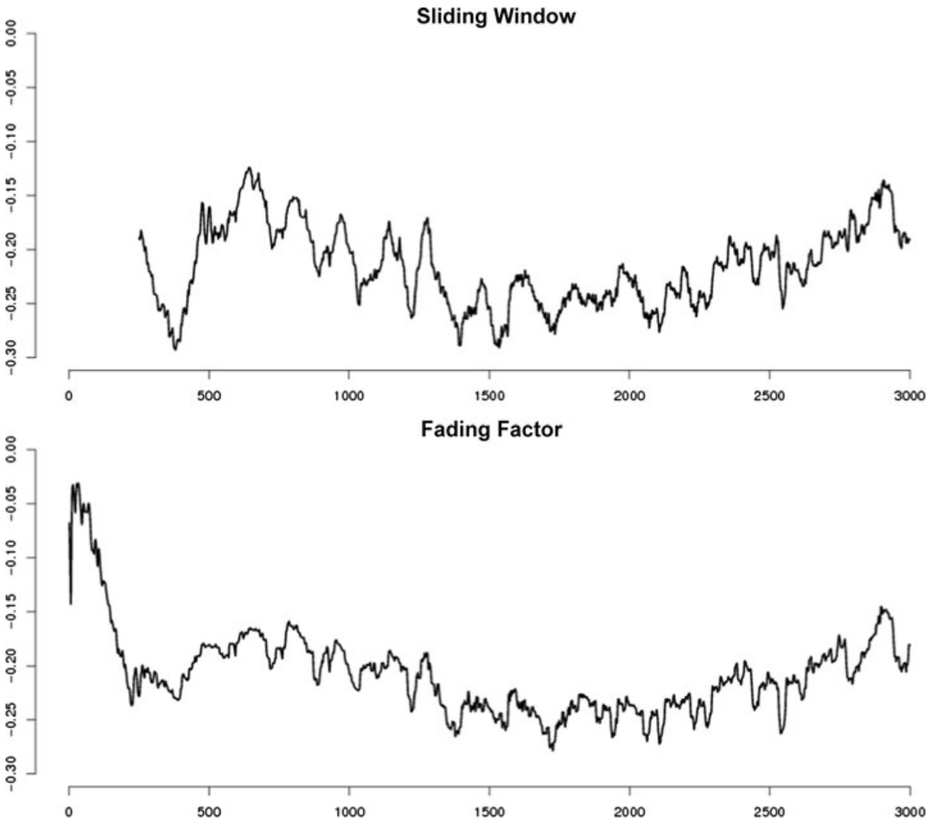

Fig. 5 Plot of the Q_i statistic over a sliding window of 250 examples (*top*). The *bottom figure* plots the Q_i statistic using a fading factor of $\alpha = 0.995$

4.1 The 0–1 loss function

For classification problems, one of the most used tests is the McNemar test.[3] The McNemar's test is a non-parametric method used on nominal data. It assesses the significance of the difference between two correlated proportions, where the two proportions are based on the same sample. It has been observed that this test has acceptable type I error (Dietterich 1996; Japkowicz and Shah 2011).

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 $M = \operatorname{sign}(n_{0,1} - n_{1,0}) \times \frac{(n_{0,1}-n_{1,0})^2}{n_{0,1}+n_{1,0}}$ 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 (Dietterich 1996).

[3]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 Demsar (2006).

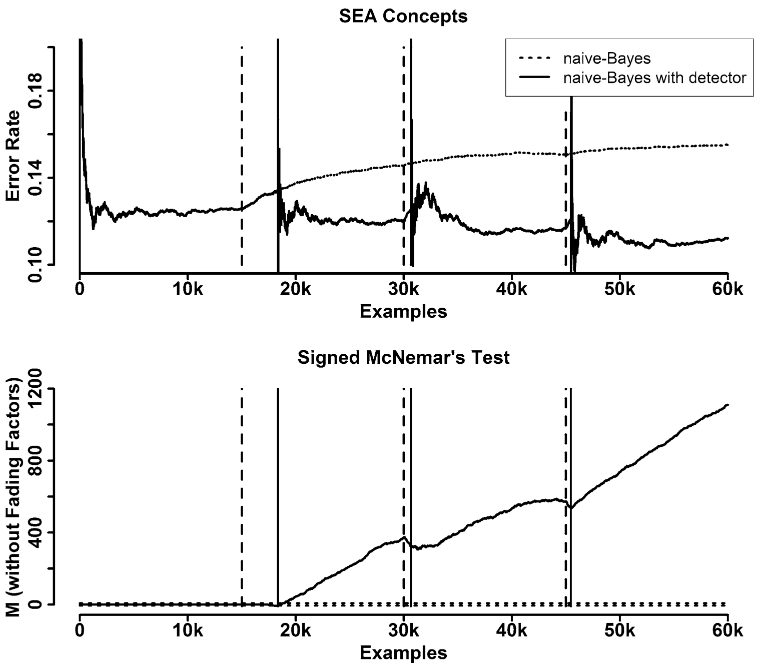

Fig. 6 The evolution of signed McNemar statistic between two algorithms. *Vertical dashed lines* indicate drift in data, and *vertical solid lines* indicate when drift was detected. The *top panel* shows the evolution of the error rate of two naive-Bayes variants: a standard one and a variant that detects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

4.2 Illustrative example

We have used the SEA concepts dataset (Street and Kim 2001) 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 the entire 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 prove to exhibit similar behaviors to sliding windows, we compute this statistical 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 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

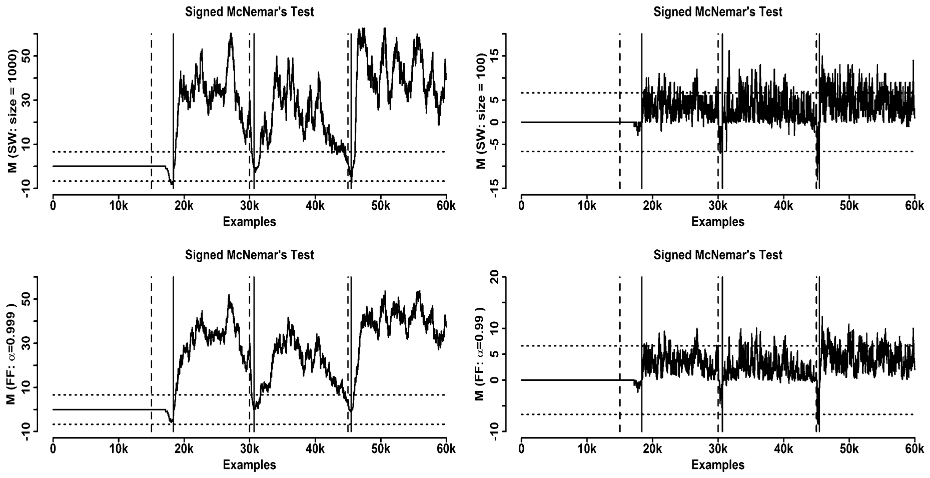

Fig. 7 The evolution of signed McNemar statistic between two naive-Bayes variants. The *two top panels* show the evolution of the signed McNemar statistic computed over a sliding window of 1000 and 100 examples, respectively, and the *two bottom panels* show the evolution of the signed McNemar statistic 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 fading factors we got different results about the significance of the differences

techniques. Nevertheless, these experiments point out that for different fading factors we got different results about the significance of the differences.

5 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. Learning time-evolving concepts is infeasible, if no restrictions are imposed on the type of admissible changes. For example, Kuh et al. (1990) determine a maximal rate of drift that is acceptable for any learner. We restrict this work to methods for explicit change detection because they are informative about the dynamics of the process generating data. We focus on change detectors that monitor the evolution of learning performance, a common strategy for drift detection metrics (Widmer and Kubat 1996; Street and Kim 2001; Klinkenberg 2004). In this section we study the use of forgetting mechanisms, based on sliding windows or fading factors, in error estimates for drift detection.

Learning from high-speed time changing data streams is a considerably growing research field and several methods capable of dealing with change detection have been presented in the literature (Hulten et al. 2001; Basseville and Nikiforov 1993; Kifer et al. 2004; Widmer and Kubat 1996; Klinkenberg 2004; Koychev 2000; Gama et al. 2004; Bifet and Gavaldà 2007). In evolving streams, some useful evaluation metrics for assessing change detection methods used in this work, include:

– Probability of True detection: capacity to detect drift when it occurs;

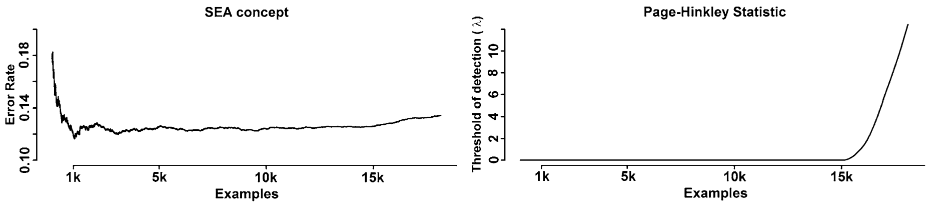

Fig. 8 Experiments in SEA dataset illustrating the first drift at point 15000. The *left panel* shows the evolution of the naive-Bayes prequential error. The *right panel* represents the evolution of the Page-Hinkley test statistic and the detection threshold λ

- Probability of False alarms: resilience to false alarms when there is no drift; that is not detect drift when there is no change in the target concept;
- Delay in detection: the number of examples required to detect a change after the occurrence of a change.

5.1 The Page-Hinkley algorithm

The Page-Hinkley (PH) test (Page 1954) is a sequential analysis technique typically used for monitoring change detection in signal processing (Mouss et al. 2004; Hartl et al. 2007). It allows efficient detection of changes in the normal behavior of a process which is established by a model. The PH test is designed to detect a change in the average of a Gaussian signal (Mouss et al. 2004). 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: $M_T = \min(m_t, t = 1 \dots T)$. 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 signal 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 change detection.

Figure 8 illustrates how PH test works. The left figure plots the trace of the prequential error of a naive-Bayes classifier (using data from the first concept of the SEA dataset). A concept drift occurs at point 15000 which leads to an error increment. The PH test allows detecting the significant increase of the error. The right 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 threshold parameter we establish a tradeoff between the false positive alarms and the miss detections.

5.2 Monitoring drift using prequential error estimates

To assess the proposed error estimates in drift scenarios, we monitor the evolution of the different prequential error estimates using the PH test. The data stream generators are Waveform, LED, Random Tree (RT) and RBF as implemented in MOA. The data was generated by emulating an abrupt concept drift event as a combination of two distributions. For the LED, Waveform and RBF datasets the first distribution was generated with the LEDGenerator, the WaveformGenerator and the RandomRBFGenerator (respectively) and the second distribution was generated with the LEDGeneratorDrift, the WaveformGeneratorDrift and the RandomRBFGeneratorDrift (respectively). For second stream of LED and Waveform datasets we set to 7 and 21 the number of attributes with drift (respectively). The second RBF stream was generated setting the seed for the random generation of the model to 10 and adding speed drift to the centroids of the model (0.01). For the RT dataset, both distributions were generated with the RandomTreeGenerator, varying the seed of the second concept. For the LED data stream the change occurs at example 128k and for the other streams the change occurs at example 32k.

The learning algorithms are VFDT-MC and VFDT-NBAdaptive as implemented in MOA. The PH test parameters are $\delta = 0.1$ and $\lambda = 100$. For the prequential error estimates we use sliding windows of size 1k, 2k, 3k, 4k and 5k and fading factors of 0.9970, 0.9985, 0.9990, 0.9993 and 0.9994.

Table 2 presents a summary of the delay in the drift detection on the aforementioned datasets, and varying the parameters of the different prequential error estimates. The results refer to the mean and standard deviation from 10 runs on streams generated with different seeds. These experiments point out the advantage of using forgetting error estimators. The PH test using the prequential error computed over all the stream only detects the change in the Waveform stream and VFDT-NBAdaptive learner (in all the runs), and misses the detection in all the other cases. The PH test computed over prequential error estimates using sliding windows or fading factors always detects the drift without any false alarm. The exception is on the RBF stream and VFDT-MC learner where, for a small number of runs (reported in parenthesis), the PH test was not able to detect the change. These results point out that the delay in detection increases by increasing the window size or by increasing the fading factor. The PH test detects much faster change points with VFDT-NBAdaptive than using VFDT-MC. Moreover, there is some evidence that error estimates based on fading factors allow fast detection rates. Figure 9 presents, for each stream, the 3 error estimates: the prequential error using all the history, the prequential error over a sliding window of size 1k and the prequential error using a fading factor of 0.997. Each plot shows the evolution of the different prequential estimates and the point where a change is detect. The learning algorithm is VFDT-NBAdaptive.

5.3 Monitoring drift with the ratio of prequential error estimates

Learning in time-evolving streams requires a tradeoff between memory and forgetting. A common approach to detect changes (Bach and Maloof 2008; Bifet and Gavaldà 2007) consists of using two sliding windows: a short window containing the most recent information and a large window, used as reference, containing a larger set of recent data including the data in the short window. The rationale behind this approach is that the short window is more reactive while the large window is more conservative. When a change occurs, statistics computed in the short window will capture the event faster than using the statistics in the larger window. Similarly, using fading factors, a smooth forgetting mechanism, a smaller fading factor will detect drifts earlier than larger ones.

Based on this assumption, we propose a new online approach to detect concept drift. We propose to perform the PH test with the ratio between two error estimates: a long term error estimate (using a large window or a fading factor close to one) and a short term error estimate (using a short window or a fading factor smaller than the first one). If the short term error

Table 2 Detection delay time using PH test over different error estimates. The learning algorithms are VFDT majority class (MC) and VFDT Naive-Bayes adaptive (NBa). We report the mean and standard deviation of 10 runs. In parenthesis the number of runs, if any, where PH test miss the detection. The PH test using the prequential error over the entire stream misses the detection in all streams, except in waveform and VFDT-NBa. In these experiments the delay increases by increasing the window size and the fading factor

Prequential	LED		RBF			RT		Waveform	
	MC	NBa		MC	NBa	MC	NBa	MC	NBa
	Miss	Miss		Miss	Miss	Miss	Miss	Miss	19698 ± 3935
Fading factors									
$\alpha = 0.9970$	2155 ± 370	486 ± 10		3632 ± 4413	1416 ± 342	911 ± 132	800 ± 84	1456 ± 326	836 ± 490
$\alpha = 0.9985$	3391 ± 797	693 ± 15		4288 ± 4413	1992 ± 430	1317 ± 177	1163 ± 121	2072 ± 437	1129 ± 565
$\alpha = 0.9990$	4279 ± 947	872 ± 19	(1)	6474 ± 4995	2454 ± 486	1676 ± 212	1481 ± 153	2678 ± 541	1420 ± 740
$\alpha = 0.9993$	5433 ± 1130	1076 ± 24	(1)	9241 ± 7769	3023 ± 571	2122 ± 255	1861 ± 190	3452 ± 656	1766 ± 952
$\alpha = 0.9994$	6103 ± 1331	1183 ± 26	(1)	9818 ± 7587	3369 ± 644	2365 ± 279	2068 ± 208	3874 ± 718	1931 ± 1002
Sliding windows									
$w = 1000$	2542 ± 512	725 ± 14		3765 ± 4444	1617 ± 321	1157 ± 120	1054 ± 79	1637 ± 290	1069 ± 444
$w = 2000$	3484 ± 619	1146 ± 30		4631 ± 4430	2397 ± 338	1866 ± 156	1718 ± 121	2491 ± 353	1622 ± 523
$w = 3000$	4454 ± 619	1503 ± 36	(1)	7085 ± 5205	3151 ± 344	2549 ± 206	2331 ± 164	3366 ± 400	2158 ± 713
$w = 4000$	5436 ± 804	1836 ± 49	(1)	8315 ± 5320	3987 ± 384	3209 ± 249	2925 ± 203	4246 ± 433	2654 ± 824
$w = 5000$	6419 ± 860	2162 ± 55	(2)	9198 ± 5683	4899 ± 466	3859 ± 295	3507 ± 237	5152 ± 463	3133 ± 918

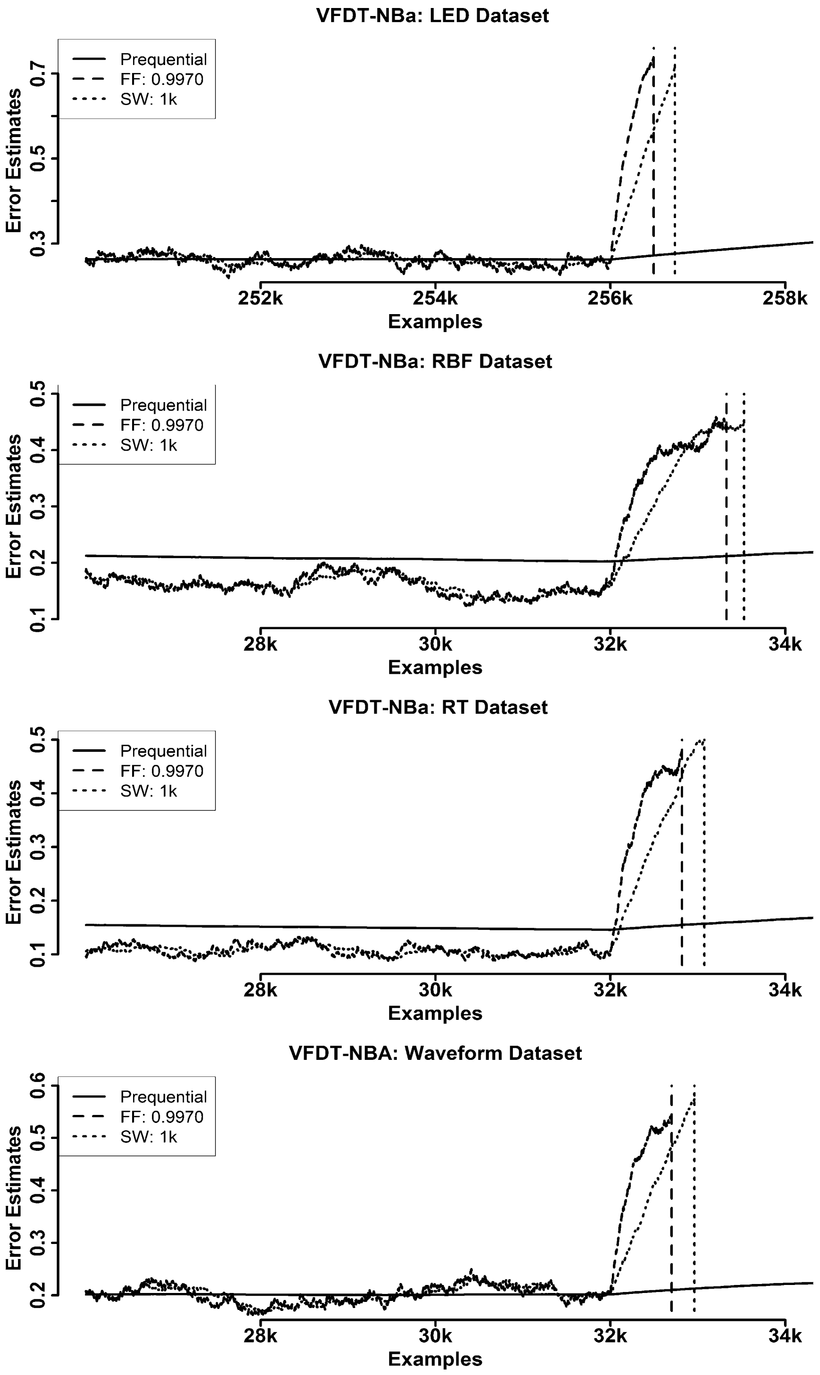

Fig. 9 Change detection using PH test from prequential error estimates. The learning algorithm is VFDT-NBAdaptive. Each plot corresponds to a data stream and presents the evolution of prequential error estimates. The *vertical lines* identify the point where change was detected. In these experiments, the fastest drift detection method is the prequential estimate based on fading factors

estimator is significantly greater than the long term error estimator, we signal a drift alarm. In the case of fading factors, we consider α_1 and α_2 (with $0 \ll \alpha_2 < \alpha_1 < 1$) and compute the *fading average* w.r.t. both: $M_{\alpha_1}(i)$ and $M_{\alpha_2}(i)$ at observation i using Algorithm 3. The ratio between them is referred as: $R(i) = M_{\alpha_2}(i)/M_{\alpha_1}(i)$. The PH test monitors the evolution of $R(i)$ and signals a drift when a significant increase of this variable is observed. The pseudo-code is presented in Algorithm 4. With sliding windows, the procedure is similar. Considering two sliding windows of different sizes $SW_1 = \{e_j | j \in]i - w_1, i]\}$ and $SW_2 = \{e_j | j \in]i - w_2, i]\}$ (with $w_2 < w_1$) and computing the moving average w.r.t. both: $M_{w_1}(i) = 1/w_1 \sum_{j=i-w_1}^{i} e_j$ and $M_{w_2}(i) = 1/w_2 \sum_{j=i-w_2}^{i} e_j$, at observation i using Algorithm 2. The PH test monitors the ratio between both moving averages.

Experimental evaluation In these experiments we use the same streams, learning algorithms and parameters of the PH test as in the previous section. In this study, we vary the values of the first fading factor and the length of the larger window to compare results. Table 3 present the delay time for the experiments in the context described above. We can control the rate of forgetting using different fading factors or different windows lengths. With respect to the ratio using different *fading factors*, the value of the second *fading factor* is set to 0.9970 and the value of the first one varies from 0.9994 to 0.9990. For the ratio using different sliding windows, the length of the second window is set to 1000 and the length of the first one varies from 5k to 3k. Greater differences between the fading factors values (or the sliding windows' lengths) will reinforce the weight of most recent data, enhancing the capacity to forget old data and leading to earlier detections. Figure 10 illustrates the delay time for both methods. With respect to fading factors we use $\alpha_1 = 0.9994$ and $\alpha_2 = 0.9970$; for sliding windows we use windows of size 1k and 5k. As stated in the previous sections, the fading factors, besides less memory consumption, get advantage over sliding windows, allowing fast concept drift detections. It is also possible to notice that an increase in length of the larger window and an increase in the first fading factor produce similar results in the delay time: the ratio of error rates computed with a fading factor close to one presents smaller delay times in drift detection, alike the ratio of error rates computed with a large window's length (larger windows present a comparable behavior to higher fading factors).

The order of magnitude in detection delay time of the results presented in Table 2 is thousands of examples, while in Table 3 is hundreds of examples. Nevertheless, while monitoring the ratio of error estimates allow much faster detection, it is more risky. We observed false alarms and miss detections, mostly with VFDT-MC. In a set of experiments not reported here, we observed that the choice of α in fading factors and the window size is critical. Once more, in these experiments drift detection based on the ratio of fading estimates is somewhat faster that with sliding windows.

6 Conclusions

The design of scientific experiments is in the core of the scientific method. The definition of the metrics and procedures used in any experimental study is a necessary condition, albeit not sufficient, for *reproducibility*, that is the ability of an experiment or study to be accurately replicated by someone else working independently. 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 sliding windows or fading factors to assess the performance of stream learning algorithms in presence of non-stationary data. The prequential method is a general

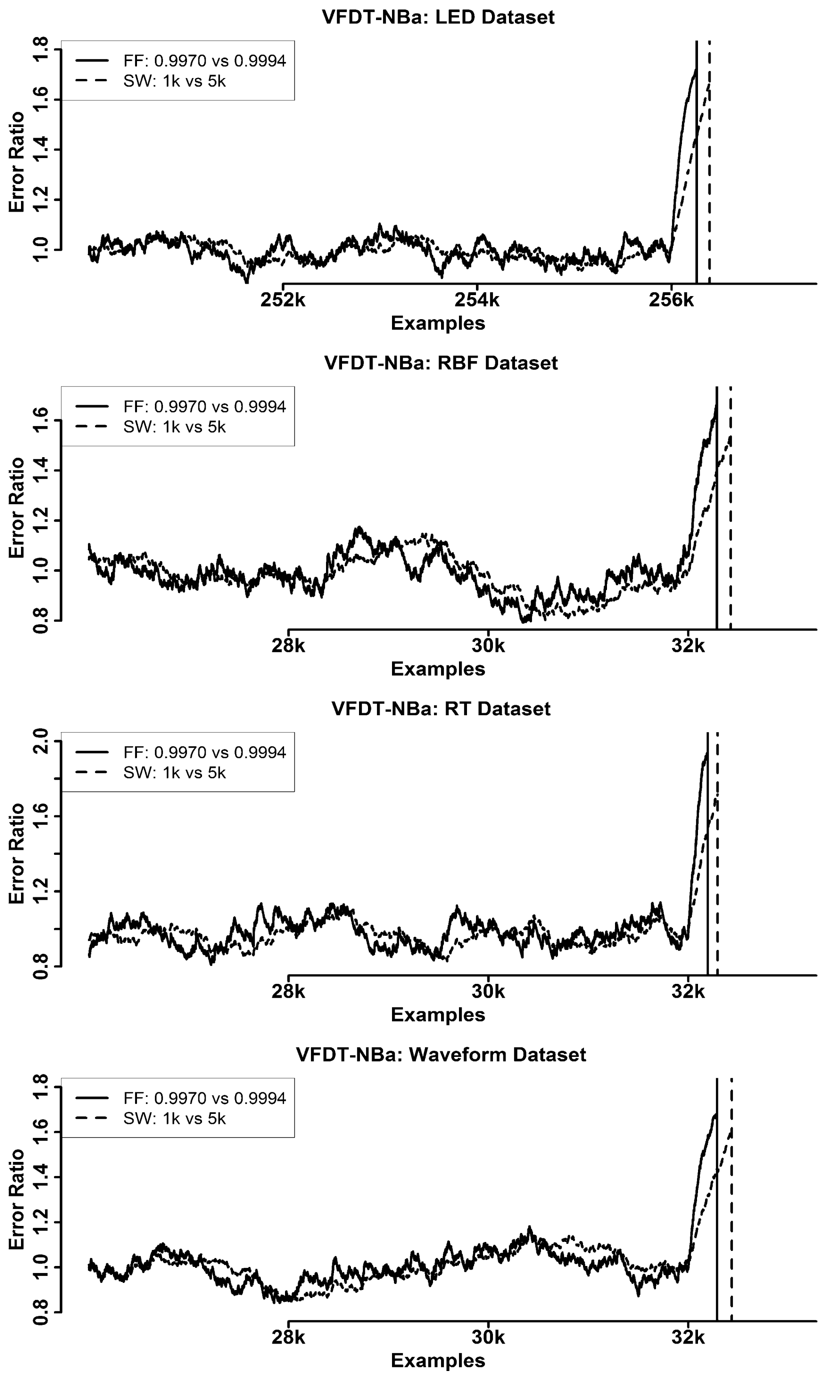

Fig. 10 The evolution of the ratio of error rate estimates and the delay times in drift detection. The learning algorithm is VFDT-NBAdaptive. Each figure corresponds to a data stream and plots the ratio of error estimates using two different fading factors and two different sliding windows. The *vertical lines* identify the point where change was detected

Table 3 Detection delay time, average and standard deviation, using PH test monitoring the ratio of error estimates. The learning algorithms are VFDT majority class (MC) and VFDT Naive-Bayes adaptive (NBa). We report the mean of 10 runs. In parenthesis the number of runs, if any, where PH test miss the detection or signal a false alarm. They are in the form (Miss; False Alarm). With respect to the ratio using different *fading factors*, the value of the second *fading factor* is set to 0.9970 and the value of the first one varies from 0.9994 to 0.9990. For the ratio using different sliding windows, the length of the second window is set to 1000 and the length of the first one varies from 5k to 3k

	LED		RBF		RT		Waveform	
	MC	NBa	MC	NBa	MC	NBa	MC	NBa
Fading factors								
$\alpha = 0.9990$	Miss	349 ± 11	787 ± 190	384 ± 38	325 ± 39	262 ± 27	780 ± 171	457 ± 161
$\alpha = 0.9993$	969 ± 255 (4; 0)	281 ± 8	563 ± 100	309 ± 29	261 ± 33	211 ± 25	541 ± 65	348 ± 89
$\alpha = 0.9994$	856 ± 186 (3; 0)	264 ± 7	522 ± 92	291 ± 28	244 ± 32	198 ± 24	496 ± 57 (2; 0)	324 ± 77 (1; 0)
Sliding windows								
$w = 3000$	1254 ± 355 (1; 0)	473 ± 27	779 ± 94	489 ± 48	429 ± 93 (0; 1)	350 ± 51 (0; 1)	762 ± 62	551 ± 102
$w = 4000$	1089 ± 261	423 ± 26	710 ± 100 (0; 1)	465 ± 41 (0; 1)	397 ± 79 (0; 2)	312 ± 52 (0; 1)	673 ± 57	501 ± 102
$w = 5000$	1005 ± 216	397 ± 25	679 ± 98 (0; 2)	440 ± 45 (0; 2)	384 ± 67 (0; 3)	299 ± 37 (0; 3)	662 ± 60 (0; 5)	468 ± 93

Algorithm 4 Drift detector based on the ratio of two fading factors

```
Require: Fading factor $\alpha_1$ ($0 \ll \alpha_1 \leq 1$)
Require: Fading factor $\alpha_2$ ($0 \ll \alpha_2 < \alpha_1$)
Require: Admissible change $\delta$
Require: Drift threshold $\lambda$
Require: $e_i$ {/* Loss at example $i$ */}
Ensure: $drift \in \{TRUE, FALSE\}$
  ...
  {/* Initialize the error estimators */}
  $S_{\alpha_1}(0) \leftarrow 0$; $S_{\alpha_2}(0) \leftarrow 0$; $SR(0) \leftarrow 0$; $m_T(0) \leftarrow 0$; $M_T \leftarrow 1$
  {/* Update the error estimator */}
  $S_{\alpha_1}(i) \leftarrow e_i + \alpha_1 * S_{\alpha_1}(i-1)$
  $N_{\alpha_1}(i) \leftarrow 1 + \alpha_1 * N_{\alpha_1}(i-1)$
  $M_{\alpha_1} \leftarrow \frac{S_{\alpha_1}(i)}{N_{\alpha_1}(i)}$
  $S_{\alpha_2}(i) \leftarrow e_i + \alpha_2 * S_{\alpha_2}(i-1)$
  $N_{\alpha_2}(i) \leftarrow 1 + \alpha_2 * N_{\alpha_2}(i-1)$
  $M_{\alpha_2} \leftarrow \frac{S_{\alpha_2}(i)}{N_{\alpha_2}(i)}$
  $R(i) = \frac{M_{\alpha_2}}{M_{\alpha_1}}$
  {/* Page Hinkley test */}
  $SR(i) \leftarrow SR(i-1) + R(i)$
  $m_T(i) \leftarrow m_T(i-1) + R(i) - \frac{SR(i)}{i} - \delta$
  $M_T \leftarrow \min(M_T, m_T(i))$
  if $m_T(i) - M_T \geq \lambda$ then
    $drift \leftarrow TRUE$
  else
    $drift \leftarrow FALSE$
  end if
  ...
```

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 the learning algorithm. For stationary data and consistent learners, we proved that the prequential error estimated with memoryless forgetting mechanisms is a good estimate of the error for stream learning algorithms. The convergence properties of the forgetting mechanisms in error estimation can be applied in concept drift detection. Thus, the use of a forgetting prequential error is recommended for continuously assess the quality of stream learning algorithms. We present applications of the proposed method in performance comparison, hypothesis testing and drift detection. We observe that drift detection is much more efficient using prequential error estimates with forgetting mechanisms.

The research presented in this work opens interesting opportunities: the system would be capable of monitoring the evolution of the learning process itself and self-diagnosis the evolution of it. We are currently implementing change detection mechanisms and corresponding adaptation strategies inside VFDT like algorithms. Other direct applications are tracking the best expert algorithms (Herbster and Warmuth 1998) and dynamic weighted majority algorithms (Kolter and Maloof 2007) that continuously track the evolution of the performance of its components.

In this work we focus on loss as performance criteria. Nevertheless, other criteria, imposed by data streams characteristics, must be taken into account. Learning algorithms run in devices with 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, some sort of sampling is required. The number of examples processed per second and the impact of sampling on 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.

Acknowledgements This work is funded by the ERDF through the Programme COMPETE and by the Portuguese Government through FCT – Foundation for Science and Technology, project refs. PEst-C/SAU/UI0753/2011 and PTDC/EIA/098355/2008, *Knowledge Discovery from Ubiquitous Data Streams*, and through POPH and FCT by a PhD grant ref. SFRH/BD/41569/2007.

Appendix A: Error estimation and sample size

For any bounded loss function, we can estimate a confidence interval for the sum of random variables to deviate from its expected value, using Hoeffding bounds (Hoeffding 1963):

$$\operatorname{Prob}\big(|\bar{X}-\mu| \geq \ 1-\varepsilon\big) \leq 2\exp\left(\frac{-2\varepsilon^2 n}{R}\right),$$
$$\text{where } R \text{ is the range of the random variable.} \tag{2}$$

For example, with a probability of 95 %, the error estimated over a sample size of 18444 randomly drawn examples, approximate the expected value over all the stream with an error less than 1 %. A simple algebraic manipulation of (2) transforms it into

$$\operatorname{Prob}\big(|\bar{X}-\mu| < \varepsilon\big) \geq 1 - 2\exp\left(\frac{-2\varepsilon^2 n}{R}\right).$$

Setting $\delta = 5\ \%$ and $\varepsilon = 1\ \%$, we obtain:

$$2\exp\left(\frac{-2\times 0.01^2\times n}{1}\right) = 0.05 \quad \Leftrightarrow \quad n = 18444.$$

Appendix B: Convergence proofs

B.1 Limit of the holdout error

Theorem (Limit of the holdout error) *For consistent learning algorithms, and for large enough holdout sets, the limit of the loss estimated in the holdout is the Bayes error.*

Proof Assuming that at time i the probability of observing a false is p_i, the errors in the holdout test are independent and identically distributed (i.i.d.) random variables, all Bernoulli distributed with success probability p_i: $e_k \sim \operatorname{Bernoulli}(p_i)$, $\forall k = 1, \dots, M$, where the expected value of e_k is p_i: $E(e_k) = p_i$. Then, from the *Law of Large Numbers*, the average obtained from a large number of trials ($H_e(i)$) converges to the expected value:

$$\forall\epsilon > 0,\ \exists N_1 \in \mathbb{N}: \ \forall i > N_1 \quad \left|\frac{1}{M}\sum_{k=1}^{M} e_k - E(e_k)\right| < \epsilon$$
$$\Leftrightarrow \quad \left|H_e(i) - E(e_k)\right| < \epsilon \quad \Leftrightarrow \quad \left|H_e(i) - p_i\right| < \epsilon.$$

Since for an infinite number of examples, the error rate of the learning algorithm (p_i) will tend to the Bayes error (B), we obtain:

$$\forall\epsilon > 0,\ \exists N_1 \in \mathbb{N}: \ \forall i > N_1 \quad \left|H_e(i) - B\right| < \epsilon \quad \Leftrightarrow \quad \lim_{i\to\infty} H_e(i) = B. \qquad \square$$

B.2 Limit of the prequential error

Theorem (Limit of the prequential error) *For consistent learning algorithms, the limit of the prequential error is the Bayes error.*

Proof Using simple algebraic manipulation:

$$\begin{aligned}\left|P_e(i) - B\right| &= \left|\frac{1}{i}\sum_{k=1}^{i} e_k - B\right| = \left|\frac{1}{i}\sum_{k=1}^{i}(e_k - B)\right| \\ &= \left|\frac{1}{i}\sum_{k=1}^{N_1}(e_k - B) + \frac{1}{i}\sum_{k=N_1+1}^{i}(e_k - B)\right| \\ &\le \left|\frac{1}{i}\sum_{k=1}^{N_1}(e_k - B)\right| + \left|\frac{1}{i}\sum_{k=N_1+1}^{i}(e_k - B)\right|.\end{aligned}$$

Let $\epsilon > 0$. Then, there exists $N_1 \in \mathbb{N}$, such that for any $i > N_1$, we have $\text{Prob}(e_i \sim \text{Ber}(B)) \ge 1 - \epsilon$. Therefore, from the *Law of Large Numbers*, the average obtained from a large number of trials converges to the expected value (B):

$$\left|\frac{\sum_{k=N_1+1}^{i} e_k}{i - N_1} - B\right| < \frac{\epsilon}{2}.$$

Hence, for $i > N_1$, we have:

$$\begin{aligned}&\left|\frac{1}{i}\sum_{k=1}^{N_1}(e_k - B)\right| + \left|\frac{1}{i}\sum_{k=N_1+1}^{i}(e_k - B)\right| \\ &\quad < \left|\frac{1}{i}\sum_{k=1}^{N_1}(e_k - B)\right| + \frac{(i - N_1)\epsilon}{2i} < \left|\frac{1}{i}\sum_{k=1}^{N_1}(e_k - B)\right| + \frac{\epsilon}{2}.\end{aligned}$$

Hence, considering $N_2 > N_1 \in \mathbb{N}$ such that $\forall i > N_2$, $\frac{1}{i}\sum_{k=1}^{N_1} |e_k - B| < \frac{\epsilon}{2}$, $\forall\epsilon > 0$, we obtain that:

$$\exists\{N_1, N_2\} \in \mathbb{N}: \forall i > N_2: \quad \left|\frac{1}{i}\sum_{k=1}^{i} e_k - B\right| < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon, \quad \forall \epsilon > 0$$

$$\Leftrightarrow \quad \lim_{i\to\infty} P_e(i) = B. \qquad \square$$

B.3 Limit of the prequential error using a sliding window

Theorem (Limit of the prequential error computed over a sliding window) *For consistent learning algorithms, the limit of the prediction error computed over a sliding window of (large enough*[4]*) size w is the Bayes error.*

Proof Using simple algebraic manipulation:

$$\begin{aligned}\left|P_w(i) - B\right| &= \left|\frac{1}{w}\sum_{k=i-w+1}^{i} e_k - B\right| \\ &= \left|\frac{1}{w}\sum_{k=i-w+1}^{N_1}(e_k - B) + \frac{1}{w}\sum_{k=N_1+1}^{i}(e_k - B)\right| \\ &\le \left|\frac{1}{w}\sum_{k=i-w+1}^{N_1}(e_k - B)\right| + \left|\frac{1}{w}\sum_{k=N_1+1}^{i}(e_k - B)\right|.\end{aligned}$$

From the *Law of Large Numbers*, we obtain that:

$$\left|\frac{\sum_{k=N_1+1}^{i} e_k}{i - N_1} - B\right| < \frac{\epsilon}{2}.$$

Hence, for $i > N_1$, we have:

$$\begin{aligned}&\le \left|\frac{1}{w}\sum_{k=i-w+1}^{N_1}(e_k - B)\right| + \left|\frac{1}{w}\sum_{k=N_1+1}^{i}(e_k - B)\right| \\ &< \left|\frac{1}{w}\sum_{k=i-w+1}^{N_1}(e_k - B)\right| + \frac{(i - N_1)\epsilon}{2w} \\ &< \left|\frac{1}{w}\sum_{k=i-w+1}^{N_1}(e_k - B)\right| + \frac{\epsilon}{2}.\end{aligned}$$

Hence, considering $N_2 \in \mathbb{N}$ such that $\forall w > N_2$:

$$\frac{1}{w}\sum_{k=i-w+1}^{N_1}|e_k - B| < \frac{\epsilon}{2}, \quad \forall \epsilon > 0,$$

[4] It is necessary to consider a window with large enough size in order to achieve a feasible computation of an average.

we obtain that:

$$\exists\{N_1, N_2\} \in \mathbb{N} : \forall i > N_1, \forall w > N_2 : \quad \left| \frac{1}{w} \sum_{k=i-w+1}^{i} e_k - B \right| < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon, \quad \forall \epsilon > 0$$

$$\Leftrightarrow \quad \lim_{i\to\infty} P_w(i) = B. \qquad \square$$

Lemma *The prequential error estimator, $P_e(i)$, is greater than or equal to the prequential error computed over a sliding window, $P_w(i)$, considering a sliding window of large enough size $w \ll i$.*

Proof [5]

$$P_e(i) = \frac{1}{i}\sum_{k=1}^{i} e_k \quad \Leftrightarrow \quad P_e(i) = \frac{\sum_{k=1}^{i-w} e_k + \sum_{k=i-w+1}^{i} e_k}{i}$$

$$\Leftrightarrow \quad P_e(i) = \frac{(i-w)\frac{\sum_{k=1}^{i-w} e_k}{i-w} + w\frac{\sum_{k=i-w+1}^{i} e_k}{w}}{i}$$

$$\Leftrightarrow \quad P_e(i) = \frac{(i-w)\bar{e}_{i-w} + wP_w(i)}{i} \quad \Rightarrow \quad P_e(i) \geq \frac{(i-w)\bar{e}_i + wP_w(i)}{i}$$

$$\Leftrightarrow \quad P_e(i) \geq \frac{(i-w)P_e(i) + wP_w(i)}{i} \quad \Leftrightarrow \quad i * P_e(i) - (i-w)P_e(i) \geq wP_w(i)$$

$$\Leftrightarrow \quad \big(i - (i-w)\big) * P_e(i) \geq wP_w(i) \quad \Leftrightarrow \quad P_e(i) \geq P_w(i). \qquad \square$$

B.4 Limit of the prequential error using fading factors

Theorem (Limit of the prequential error computed with fading factors) *For consistent learning algorithms, the limit of the prequential error computed with fading factors is approximately the Bayes error.*

Proof Using simple algebraic manipulation:

$$\begin{aligned} \left|P_\alpha(i) - B\right| &= \left| \frac{\sum_{k=1}^{i} \alpha^{i-k} e_k}{\sum_{k=1}^{i} \alpha^{i-k}} - B \right| \\ &= \left| \frac{\sum_{k=1}^{N_1} \alpha^{i-k} e_k}{\sum_{k=1}^{i} \alpha^{i-k}} + \frac{\sum_{k=N_1+1}^{i} \alpha^{i-k} e_k}{\sum_{k=1}^{i} \alpha^{i-k}} - B \right|. \end{aligned}$$

From the proof of the limit of the prequential error, we obtain that:

$$\frac{\sum_{k=N_1+1}^{i} e_k}{i - N_1} = B.$$

Hence, $\exists N_1 \in \mathbb{N} : \forall i > N_1$, we have:

[5] From the PAC learning theory the error rate of the learning algorithm will decrease, so the average of errors can be seen as a decreasing function and so if $N_1 < N_2 \Rightarrow \bar{e}_{N_1} > \bar{e}_{N_2}$.

$$\begin{aligned}
&\left|\frac{\sum_{k=1}^{N_1}\alpha^{i-k}e_k}{\sum_{k=1}^{i}\alpha^{i-k}} + \frac{\sum_{k=N_1+1}^{i}\alpha^{i-k}e_k}{\sum_{k=1}^{i}\alpha^{i-k}} - B\right| \\
&\approx \left|\frac{\sum_{k=1}^{N_1}\alpha^{i-k}e_k}{\sum_{k=1}^{i}\alpha^{i-k}} + \frac{\sum_{k=N_1+1}^{i}\alpha^{i-k}\bar{e}_k}{\sum_{k=1}^{i}\alpha^{i-k}} - B\right| \\
&= \left|\frac{\sum_{k=1}^{N_1}\alpha^{i-k}e_k}{\sum_{k=1}^{i}\alpha^{i-k}} + \frac{\sum_{k=N_1+1}^{i}\alpha^{i-k}B}{\sum_{k=1}^{i}\alpha^{i-k}} - B\right| \\
&= \left|\frac{\sum_{k=1}^{N_1}\alpha^{i-k}e_k}{\sum_{k=1}^{i}\alpha^{i-k}} + B\left(\frac{\sum_{k=N_1+1}^{i}\alpha^{i-k}}{\sum_{k=1}^{i}\alpha^{i-k}} - 1\right)\right| \le \epsilon, \quad \forall\epsilon > 0 \\
&\Leftrightarrow \quad \lim_{i\to\infty} P_\alpha(i) \approx B. \qquad \square
\end{aligned}$$

References

Asuncion, A., & Newman, D. (2007). UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Babcock, B., Datar, M., Motwani, R., & O'Callaghan, L. (2003). Maintaining variance and k-medians over data stream windows. In T. Milo (Ed.), *Proceedings of the 22nd symposium on principles of database systems*, San Diego, USA (pp. 234–243). New York: ACM.

Bach, S. H., & Maloof, M. A. (2008). Paired learners for concept drift. In *ICDM* (pp. 23–32). Los Alamitos: IEEE Comput. Soc.

Basseville, M., & Nikiforov, I. (1993). *Detection of abrupt changes: theory and applications*. New York: Prentice Hall

Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings SIAM international conference on data mining*, Minneapolis, USA (pp. 443–448). Philadelphia: SIAM.

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010a). MOA: massive online analysis. *Journal of Machine Learning Research*, *11*, 1601–1604.

Bifet, A., Holmes, G., Pfahringer, B., & Frank, E. (2010b). Fast perceptron decision tree learning from evolving data streams. In *Advances in knowledge discovery and data mining*, 14th Pacific-Asia conference (pp. 299–310).

Bishop, C. (1995). *Neural networks for pattern recognition*. London: Oxford University Press.

Chi, Y., Wang, H., Yu, P. S., & Muntz, R. R. (2006). Catch the moment: maintaining closed frequent itemsets over a data stream sliding window. *Knowledge and Information Systems*, *10*(3), 265–294.

Cormode, G., Muthukrishnan, S., & Zhuang, W. (2007). Conquering the divide: continuous clustering of distributed data streams. In *ICDE: proceedings of the international conference on data engineering*, Istanbul, Turkey (pp. 1036–1045).

Datar, M., Gionis, A., Indyk, P., & Motwani, R. (2002). Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, *31*(6), 1794–1813.

Dawid, A. P. (1984). Statistical theory: the prequential approach. *Journal of the Royal Statistical Society. Series A*, *147*, 278–292.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Dietterich, T. (1996). *Approximate statistical tests for comparing supervised classification learning algorithms*. Corvallis, technical report nr. 97.331, Oregon State University.

Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. In I. Parsa, R. Ramakrishnan, & S. Stolfo (Eds.), *Proceedings of the ACM sixth international conference on knowledge discovery and data mining*, Boston, USA (pp. 71–80). New York: ACM.

Duda, R., & Hart, P. (1973). *Pattern classification and scene analysis*. New York: Willey.

Ferrer-Troyano, F., Aguilar-Ruiz, J. S., & Riquelme, J. C. (2004). Discovering decision rules from numerical data streams. In *Proceedings of the ACM symposium on applied computing*, Nicosia, Cyprus (pp. 649–653). New York: ACM Press.

Gama, J., & Kosina, P. (2011). Learning decision rules from data streams. In *Proceedings of the 22nd international joint conference on artificial intelligence, IJCAI* (pp. 1255–1260).

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. In A. L. C. Bazzan & S. Labidi (Eds.), *Lecture notes in computer science: Vol. 3171. Advances in artificial intelligence—SBIA 2004*, São Luis, Brasil (pp. 286–295). Berlin: Springer.

Gama, J., Rocha, R., & Medas, P. (2003). Accurate decision trees for mining high-speed data streams. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining*, Washington, DC, USA (pp. 523–528). New York: ACM.

Gama, J., Sebastião, R., & Rodrigues, P. P. (2009). Issues in evaluation of stream learning algorithms. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining*, Paris, France (pp. 329–338). New York: ACM.

Ghosh, B., & Sen, P. (1991). *Handbook of sequential analysis*. New York: Dekker.

Giannella, C., Han, J., Pei, J., Yan, X., & Yu, P. (2003). Mining frequent patterns in data streams at multiple time granularities. In H. Kargupta, A. Joshi, K. Sivakumar, & Y. Yesha (Eds.), *Next generation data mining*. Menlo Park/Cambridge: AAAI Press/MIT Press.

Hartl, C., Baskiotis, N., Gelly, S., & Sebag, M. (2007). Change point detection and meta-bandits for online learning in dynamic environments. In *Conférence Francophone sur l'apprentissage automatique*, Cepadues (pp. 237–250).

Herbster, M., & Warmuth, M. (1998). Tracking the best expert. *Machine Learning*, *32*(2), 151–178.

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

Hulten, G., & Domingos, P. (2001). Catching up with the data: research issues in mining data streams. In *Proc. of workshop on research issues in data mining and knowledge discovery*, Santa Barbara, USA.

Hulten, G., & Domingos, P. (2003). *VFML—a toolkit for mining high-speed time-changing data streams*. Technical report, University of Washington. http://www.cs.washington.edu/dm/vfml/

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the 7th ACM SIGKDD international conference on knowledge discovery and data mining*, San Francisco, California (pp. 97–106). New York: ACM.

Japkowicz, N. & Shah, M. (Eds.) (2011). *Evaluating learning algorithms: a classification perspective*. Cambridge: Cambridge University Press.

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2010). Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems*, *22*, 371–391.

Kearns, M., & Vazirani, U. (1994). *An introduction to computational learning theory*. Cambridge: MIT Press.

Kifer, D., Ben-David, S., & Gehrke, J. (2004). Detecting change in data streams. In *Proceedings of the international conference on very large data bases*, Toronto, Canada (pp. 180–191). San Mateo: Morgan Kaufmann.

Kirkby, R. (2008). *Improving Hoeffding trees*. Ph.D. thesis, University of Waikato, New Zealand.

Klinkenberg, R. (2004). Learning drifting concepts: example selection vs. example weighting. *Intelligent Data Analysis*, *8*(3), 281–300.

Kolter, J. Z., & Maloof, M. A. (2007). Dynamic weighted majority: an ensemble method for drifting concepts. *Journal of Machine Learning Research*, *8*, 2755–2790.

Koychev, I. (2000). Gradual forgetting for adaptation to concept drift. In *Proceedings of ECAI workshop current issues in spatio-temporal reasoning*, Berlin, Germany (pp. 101–106). Leipzig: ECAI Press.

Kuh, A., Petsche, T., & Rivest, R. (1990). Learning time-varying concepts. In *Proceedings advances in neural information processing* (pp. 183–189). San Mateo: Morgan Kaufmann.

Li, P., Wu, X., & Hu, X. (2010). Mining recurring concept drifts with limited labeled streaming data. *Journal of Machine Learning Research—Proceedings Track*, *13*, 241–252.

Liang, C., Zhang, Y., & Song, Q. (2010). Decision tree for dynamic and uncertain data streams. *Journal of Machine Learning Research—Proceedings Track*, *13*, 209–224.

Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., & Euler, T. (2006). Yale: rapid prototyping for complex data mining tasks. In *ACM SIGKDD int. conf. on knowledge discovery and data mining* (pp. 935–940). New York: ACM Press.

Mitchell, T. (1997). *Machine learning*. New York: McGraw-Hill

Mouss, H., Mouss, D., Mouss, N., & Sefouhi, L. (2004). Test of Page-Hinkley, an approach for fault detection in an agro-alimentary production system. In *Proceedings of the Asian control conference* (Vol. 2, pp. 815–818).

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

Rodrigues, P. P., Gama, J., & Pedroso, J. P. (2008). Hierarchical clustering of time series data streams. *IEEE Transactions on Knowledge and Data Engineering*, *20*(5), 615–627.

Street, W. N., & Kim, Y. (2001). A streaming ensemble algorithm SEA for large-scale classification. In *Proceedings 7th ACM SIGKDD international conference on knowledge discovery and data mining*, San Francisco, California (pp. 377–382). New York: ACM Press.

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, *23*, 69–101.