A Study on Change Detection Methods

Raquel Sebastião[1,2] and João Gama[1,3]

[1] LIAAD-INESC Porto L.A., University of Porto
Rua de Ceuta, 118 - 6, 4050-190 Porto, Portugal
[2] Faculty of Science, University of Porto
[3] Faculty of Economics, University of Porto
{raquel,jgama}@liaad.up.pt

Abstract. In the real word, the environment is often dynamic instead of stable. Usually the underlying data of a problem changes with time, which enhances the difficulties when learning a model from data. In this paper, different methods capable to detect changes from high-speed time changing data streams are compared. These methods are appropriated to be embedded inside learning models, allowing the adaptation to a non-stationary problem. The experimental evaluation considers different types of concept drift and data streams with different properties. Assessing measures such as: false alarm rates, number of samples until a change is detected and miss detections rates, a comparison between the algorithms' capability of consistent detection is given. The choice on the best detection algorithm relies on a trade-off between the rate of false alarms and miss detections and the delay time until detection.

Key words: change detection, data streams, machine learning, monitoring data distribution.

1 Introduction: Motivation and Challenges

The most recent developments in Science and Information Technology lead to the considerably growing of the computational capacity of small devices, which are capable to produce a large massive amount of information at a high-speed rate. Along with this, as data flows over time for large periods of time, the process generating data is not strictly stationary and evolves over time.

The motivation for studying time-changing high-speed data streams comes from the emergence of temporal applications such as signal processing, time series analysis, sensor networks, automatic control, real time-monitoring in biomedicine and industrial processes, fraud detection, user modelling, safety of complex systems and many others.

Due to the dynamic nature of data, some properties of the problem can change over time, namely the target concept on which data is obtained may shift from time to time, each time after some minimum of permanence. Learning algorithms that model time-changing underlying processes must be able to track the dynamic behaviour and adapt the decision model accordingly. For example, in a prediction task, it is reasonable to assume that a change in the underlying

data distribution will increase the error rate thus making the predictions reflect characteristics that are no longer hold. Since old observations (reflecting the behaviour of nature in the past) became irrelevant to the current state, after the occurrence of a change the prediction model need to be retrained to accurately predict the underlying actual data. These dynamic processes are challenging and need to be addressed with appropriated drift detection algorithms.

The nature of change is another important and challenging issue. Changes may occur due to modifications in the context of learning (caused by changes in hidden variables) or in the intrinsic properties of the observed variables. Usually, literature considers two types of drift. The term *Concept Shift* refers to abrupt changes (for example, the patterns of costumers' buying preferences that may change with seasons), while the term *Concept Drift* is associated to gradual changes in the target concept (for example, small faults in parts of an industrial process can modify the quality of the product). *Concept Drift* is more difficult to detect, and at least in the initial phases it may be confused with noise.

The main challenge of algorithms for change detection is the combination of *robustness* to noise with *sensibility* to concept change. Noise and outliers pose difficulties and challenges to drift detection algorithms, and may increase false alarm rates. The scope of this paper is the study of different methods for drift detection, comparing their capacities under different types of concept drift, different data streams' lengths and different scenarios.

The paper is organized as follows: next section presents the related work in the field of change detection in data streams; section 3 presents the drift detection algorithms analyzed; section 4 describes the experimental evaluation and conclusions are presented in the last section.

2 The Change Detection Problem in Data Streams

When monitoring a stream it is fundamental to know if the received data comes from the distribution observed so far. Suppose a supervised learning problem, where the data consists of sequences of pairs $(\vec{x_i}, y_i)$ where $y_i \in C_1, C_2, ..., C_k$. At each time stamp t, the task of the learner is to output the class prediction $\hat{y}_t$ of that example. After checking the class y_t the error of the algorithm is computed. According with the *Probability Approximately Correct* (PAC) learning model [1] if the distribution of examples is stationary, the error rate of the learning model will decrease when the number of examples increases. A significant increase in the error rate suggests a change in the process generating data. For large periods of time, it is reasonable to assume that the process generating data will evolve. Whenever new concepts replace old ones, the old observations become irrelevant and thus the model will become inaccurate. In such case, the learning model must be adapted in accordance with the current state of the phenomena under observation.

Assuming that examples are independent and generated at random according to an unknown distribution, to assess if a concept is shifting over time, it is necessary to perform tests in order to determine if there is a change in the

generating distribution. The null hypothesis is that the previously seen values and the current observed ones come from the same distribution. The alternative hypothesis is that they are generated from different distributions. Furthermore, such kind of change detection tests should detect only true changes with high probability, establishing a trade-off between false negatives and false positives. Specifically, these algorithms should:

- Be able to detect and react to drift.
- Not exhibit miss detections.
- Be resilience to false alarms (detect a change in stationary environments).
- Require few examples to detect a change after the occurrence of one.

2.1 Related Work

Since our environment is naturally dynamic, learning form high-speed time changing data streams is a considerably growing research field and several methods capable of dealing with concept drifts have been proposed [2–6, 11]. In general, approaches to cope with concept drift can be classified into two categories: (i) approaches that adapt a learner at regular intervals without considering whether changes have really occurred;(ii) approaches that first detect concept changes and afterwards the learner is adapted to these changes.

In the first approach, drifting concepts are often handled by time windows or weighted examples according to their age or utility. Weighted examples are based on the simple idea that the importance of an example should decrease with time (references about this approach can be found in [8–10]). When a time window is used, at each time step the learner is induced only from the examples that are included in the window. Here, the key difficulty is how to select the appropriate window's size: a small window can assure a fast adaptability in phases with concept changes. But in more stable phases it can affect the learner performance. On the other end, a large window would produce good and stable learning results in stable phases but can not react quickly to concept changes. In later approaches with the aim of detecting concept changes, some indicators (e.g. performance measures, data distribution, properties of the data, etc.) are monitored over time. A typical example is to monitor the evolution of a statistical function between two distributions: from past data in a reference window and in a current window of the most recent data points [2, 5]. If during the monitoring process a concept drift is detected, some actions to adapt the learner to these changes can be taken. When a time window of adaptive size is used these actions usually lead to adjusting the window's size according to the extent of concept drift [7]. As a general rule, if a concept drift is detected the window's size decreases; otherwise the window's size increases [2].

3 Change Detection Algorithms

This paper performs a comparison between four known algorithms taken from literature: the Statistical Process Control (SPC) presented in [3], the ADaptative WINDdowing (ADWIN) method introduced in [11], the Fixed Cumulative

Windows Model (FCWM), presented at [2] and a standard algorithm for change detection, the Page-Hinkley Test (PHT) [13].

3.1 Statistical Process Control (SPC)

This drift detection method controls online the trace of the probability of error for streaming observations. While monitoring the error, it defines a warning and a drift level. When the error exceeds the first (lower) threshold, the system enters in a warning mode and stores the time, t_w, of the corresponding observation. If the error drops below the threshold again, the warning mode is cancelled. However, if in a sequence of examples, the error increases reaching the second (higher) threshold at time t_d, a change in the distribution of the examples is declared. The classifier is retrained using only the examples since t_w and the warning and drift levels are reset.

Pseudo-code

Input:

labeled dataset x_1, . . . , x_t
warning threshold t_w (default t_w = 2)
detection threshold t_d (default t_d = 3)
warm-up window size w_0 (default w_0 = 30)

1. Initialize the minimum classification error $p_{min} = \infty$ and the corresponding standard deviation $s_{min} = \infty$. Set the warning zone flag, f_w, to false and w_1 =0.
2. For j = 1 to t - 1 (all the observations)
If $j < w_0$ then
$w_{j+1} = w_j + 1$ (warm up, only grow the window)
Else
i. Train a classifier on the current window of size w_j.
ii. Classify observation w_{j+1}.
iii. Update the error rate over the current window.
Let $\hat{p}$ be the updated error rate and $\hat{s} = \frac{\hat{p}(1-\hat{p})}{w_j}$ be the updated standard deviation.
iv. If $(\hat{p} + \hat{s}) < (p_{min} + s_{min})$ then update the minimum error by $p_{min} = \hat{p}$ and $s_{min} = \hat{s}$.
v. If $(\hat{p} + \hat{s}) > (p_{min} + t_d \times s_{min})$ and f_w = true
change has been detected, set up the
detection time $t_d = j$.
Take as the new training and detection window all the observations since t_w (size $w_{j+1} = j - t_w + 1$), set $p_{min} = \infty$, $s_{min} = \infty$ and $t_w = \infty$.
ElseIf $(\hat{p} + \hat{s}) > (p_{min} + t_w \times s_{min})$
If f_w = false
switch the warning zone flag f_w = true and set up

```
            the warning time t_w = j.
         Else
            set f_w = false and update the window by adding
            x_{j+1} to it (size w_{j+1} = w_j + 1).
3. Set DT_SPC = t_w.
```

Output: detection time DT_{SPC}.

3.2 ADaptive WINdowing (ADWIN)

The ADaptative WINDdowing method keeps a sliding window W (with length n) with the most recently received examples and compares the distribution on two sub-windows of W. Whenever two *large enough* sub-windows, W_0 and W_1, exhibit *distinct enough* averages, the older sub-window is dropped and a change in the distribution of examples is assigned. The window cut threshold is computed as follows:

$\epsilon_{cut} = \sqrt{\frac{1}{2m} ln \frac{4}{D}}$, with $m = \frac{1}{1/n_0 + 1/n_1}$, where n_0 and n_1 denote the lengths of W_0 and W_1.

A confidence value D is used within the algorithm, which establishes a bound on the false positive rate. However, as this first version was computationally expensive, the authors propose to use a data structure (a variation of exponential histograms), where the information on the number of 1's is kept as a series of buckets (in the Boolean case). It keeps at most M buckets of each size 2^i, where M is a design user-defined parameter. For each bucket, two (integer) elements are recorded: *capacity* and *content* (*size* or the number of 1s it contains). The method is detailed in [11].

Pseudo-code

Input:

```
labeled dataset x_1, . . . , x_t
confidence value D ∈ (0, 1)
bucket's parameter M

1. Initialize W as an empty list of buckets
2. Initialize ε_cut
3. for each t > 0
      do SetInput(X_t, W)
         Output u_W and ChangeAlarme

SetInput (item e, List W)
      1. InsertElement(e, W)
      2. repeat DeleteElement(W)
         until |u_{W_0} − u_{W_1}| < ε_cut holds
               for every split of W into W = W_0 · W_1

InsertElement (item e, List W)
```

```
      1. Create a new bucket b with content e and capacity 1
      2. W ← W ∪ b (add e to the head of W)
      3. Update ε_cut
      4. CompressBuckets(W)

DeleteElement(List W)
      1. Remove a bucket from the tail of list W
      2. Update ε_cut
      3. ChangeAlarm ← true

CompressBuckets(List W)
      1. Transverse the list of buckets in increasing order
         do If there are more than M buckets of the same capacity
            do merge buckets
               CompressBuckets(sublist of W not transversed)
```

3.3 Fixed Cumulative Windows Model (FCWM)

In a previous work [2] the FCWM was presented as a method to detect changes in data streams. To summarize data, it first constructs histograms using the two layer structure of the Partition Incremental Discretization (PiD) algorithm, which was designed to learn histograms from high-speed data streams [12]. The change detection problem is addressed by monitoring distributions in two different time windows: a reference window (RW), reflecting the distribution observed in the past; and a current window (CW) which receives the most recent data. In order to assess drifts, both distributions are compared using the Kullback-Leibler divergence (KLD), defining a threshold for change detection decision based on the asymmetry of this measure.

Pseudo-code

Input:

```
labeled dataset x_1, . . . , x_t
number of bins nBins
number of observations in the current window InitialObs
evaluation interval EvalInterval
drift threshold λ

1. for t > 0
      1.1 If t > InitialObs then
          Compute the probability distribution p for the RW
      else
          Compute the probability distribution q for the CW
          From EvalInterval to EvalInterval compute
                A = |KLD(p||q) − KLD(q||p)|
          If A > λ ⇒ a drift is detected
```

```
            Return and report a change at time t_FCWM
      else
            Return to 1.1
```

Output: detection time t_{FCWM}.

3.4 Page Hinkley Test (PHT)

The Page-Hinkley test (PHT) is a sequential analysis technique typically used for monitoring change detection [13]. It allows efficient detection of changes in the normal behaviour of a process which is established by a model. The PHT was designed to detect a change in the average of a Gaussian signal [14]. This test considers a cumulative variable U_T defined as the cumulated difference between the observed values and their mean till the current moment:

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

where $\bar{x_T} = 1/T \sum_{t=1}^{t} x_t$ and δ corresponds to the magnitude of changes that are allowed. To detect increases, it computes the minimum value of U_t: $m_T = min(U_t, t = 1 \ldots T)$ and monitors the difference between U_T and m_T: $PH_T = U_T - m_T$. When the difference PH_T is greater than a given threshold (λ) a change in the distribution is assigned. The threshold λ depends on the admissible false alarm rate. Increasing λ will entail fewer false alarms, but might miss or delay some changes. Controlling this detection threshold parameter makes it possible to establish a trade-off between the false alarms and the miss detections.

Pseudo-code

Input:

```
labeled dataset x_1, . . . , x_t
magnitude threshold δ
detection threshold λ
```

```
1. for t > 0
      1.1 Computes
                  x̄_T = 1/T Σ_{t=1}^{t} x_t
                  U_T = Σ_{t=1}^{T} (x_t − x̄_T − δ)
                  m_T = min(U_t, t = 1 . . . T)
      If PH_T = U_T − m_T > λ
         return and report a change at time t_PH
      else
         return to 1.1
```

Output: detection time t_{PH}.

4 Experimental Evaluations

This section describes the evaluation of the mentioned methods. To assess the performance of change detection algorithms under different scenarios, different kinds of experiments with distinct purposes were implemented. The rate of false alarms, miss detections and delay time until detection were evaluated using: (i) data underlying a Bernoulli distribution, (ii) artificial datasets with distinct characteristics and (iii) a public dataset.

4.1 Artificial Data

The first set of experiments uses data streams of lengths $L = 2.000$, 5.000 and 10.000, underlying a stationary Bernoulli distribution of parameter $\mu = 0.2$ during the first $L - 1.000$ observations. During the last 1.000 samples the parameter is linearly increased using different slopes: 0(no change), 10^{-4}, 2.10^{-4}, 3.10^{-4} and 4.10^{-4}. These experiments also allow analyzing the influence (in the delay time until detections) of the length of the stationary part (the first $L - 1.000$ samples).

Table 1. Mean delay time until drift detection (DT), false alarms rates (FA) and the miss detections rates (MD), for the four methods, using the data streams with lengths 2.000, 5.000 and 10.000 and with different slopes in the Bernoulli parameter distribution. For $slope = 0$ (no change) the measurements DT and MD are not applicable.

Length	**Slope**	**ADWIN**			**SPC**			**FCWM**			**PHT**		
		DT	FA	MD	DT	FA	MD	DT	FA	MD	DT	FA	MD
2.000	0	(n.a.)	0.05	(n.a.)	(n.a.)	0	(n.a.)	(n.a.)	0.02	(n.a.)	(n.a.)	0.04	(n.a.)
	1.10^{-4}	581.6	0	0.03	626.6	0	0.02	853.8	0	0.37	573.0	0	0.03
	2.10^{-4}	577.6	0	0	687.2	0	0.16	894.8	0	0.59	522.9	0	0
	3.10^{-4}	428.4	0	0	536.9	0	0	647.0	0	0	397.3	0	0
	4.10^{-4}	358.6	0	0	534.4	0	0	616.3	0	0	331.3	0	0
5.000	0	(n.a.)	0.17	(n.a.)	(n.a.)	0.17	(n.a.)	(n.a.)	0.35	(n.a.)	(n.a.)	0.41	(n.a.)
	1.10^{-4}	721.9	0.16	0.30	866.4	0.21	0.77	705.5	0.27	0.64	649.6	0.23	0.13
	2.10^{-4}	512.0	0.12	0.13	732.3	0.19	0.37	674.7	0.18	0.14	462.9	0.25	0
	3.10^{-4}	382.6	0.14	0.14	667.9	0.20	0.17	572.8	0.31	0.04	337.0	0.32	0
	4.10^{-4}	320.1	0.10	0.10	587.1	0.09	0.12	501.8	0.35	0.06	279.2	0.29	0
10.000	0	(n.a.)	0.15	(n.a.)	(n.a.)	0.44	(n.a.)	(n.a.)	0.22	(n.a.)	(n.a.)	0.68	(n.a.)
	1.10^{-4}	721.6	0.19	0.35	829.3	0.39	0.94	723.7	0.27	0.69	649.6	0.60	0.10
	2.10^{-4}	505.0	0.19	0.19	842.7	0.56	0.57	687.4	0.31	0.36	466.0	0.71	0
	3.10^{-4}	401.3	0.17	0.17	719.5	0.29	0.53	597.4	0.29	0.29	343.9	0.68	0
	4.10^{-4}	327.2	0.23	0.23	642.2	0.52	0.42	481.0	0.24	0.24	280.2	0.66	0

As it is presented in table 1, rows are indexed by the value of L and corresponding slope. So, these rows present the delay time (DT) until the detection of the change that occurs at time stamp $L - 1.000$ (averaged over all runs), the miss

detections rates (MD) and the false alarms rates (FA) for each algorithm studied. In general, the increase of the number of samples leads to an increase of the number of false alarms and miss detections. It can also be observed that as the number of samples increases, the percentage of changes detected decreases (with exception of the Page Hinkley Test). As it is reasonable, for all the algorithms, the increase on the slope of the Bernoulli's parameter contributes to a decrease of the time until the change is detected. As a final remark, the PHT maintains a good rate of miss detections, independent of the number of stationary past samples. This method presents smaller mean delay times, however these results are compromised with false alarms.

In the following set of experiments were used the following artificial datasets previously used in concept drift detection [15]:

- **SINE1**: Abrupt concept drift, noise-free examples.
- **SINIRREL1** Abrupt concept drift, noise-free examples, presence of irrelevant attributes (the same classification function of SINE1 but the examples have two more random attributes with no influence on the classification function).
- **CIRCLES**: Gradual concept drift, noise-free examples.
- **GAUSS**: Abrupt concept drift, noisy examples.

These artificial datasets have several different characteristics that allow assessing the methods' performance in various conditions: abrupt and gradual drift, presence and absence of noise and presence of irrelevant attributes. All the problems have two classes and each class is represented by 50% of the examples in each context. To ensure a stable learning environment within each context, the positive and negative examples in the training set are alternated. The number of examples in each concept is 1.000. To compute the error rate in these classification problems, the detection algorithms were embedded inside a decision tree classifier. Figure 1 presents the mean delay time for the four methods. In spite of the smaller delay for PHT, this method closely competes with ADWIN and SPC. On the other extreme, FCWM exhibits the highest delays. With exception of the ADWIN that presents a false alarm when using the SINE1 dataset, in all of the other cases there are no false alarms to report.

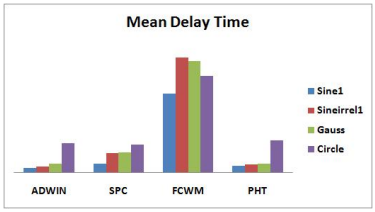

Fig. 1. Mean delay time until drift detection (DT) for the four methods, using the datasets SINE1, SINEIRREL1, GAUSS and CIRCLES.

4.2 Evaluation on a Public Dataset

In the previous experiments, the datasets did not allow checking the algorithms' performance in large problems, which is important since concept drift mostly occurs in huge amounts of data arriving in the form of stream. To overcome this drawback, an evaluation of the change detection algorithms was performed using the dataset SEA concepts [16], a benchmark problem for concept drift. Figure 2 shows the error rate (computed using a naive-Bayes classifier), which presents three drifts. The drifts and the corresponding detections are also represented by vertical lines. Table 2 presents the delay time in detection of concept drifts in this dataset (where the number of false alarms is indicated in parenthesis). One can observe that all the algorithms require too much examples to detect the first drift. However, with the ADWIN, the FCWM and PHT, the others are detected within a reasonable delay time. The resilience to false alarms and the ability to reveal changes without miss detections must be stressed out.

Table 2. Delay time in three drift scenarios using different change detection methods. The number of false alarms is indicated in parenthesis.

$\#Drift$	Delay Time			
	ADWIN	SPC	FCWM	PHT
1	826	3314	817	1404
2	115	607	273	118
3	242	489	249 (1)	256

5 Conclusions and Further Research

Regarding the delay time until detection and the miss detections, the Page Hinkley Test and the ADaptive WINdowing revealed to be the more appropriated algorithms to detect changes in the evaluated drift scenarios. However, in the first set of experiments the PHT presents a high rate of false alarms. Nevertheless, since learning algorithms run in fixed memory, time and memory consumption are also important constrains. It should be pointed out that PHT and Statistical Process Control are less time and memory consuming than the ADWIN and the Fixed Cumulative Windows Model, since they do not require any data structure to evaluate drifts.

Acknowledgements

The work of Raquel Sebastião is supported by the Portuguese Foundation for Science and Technology (FCT) under the PhD Grant SFRH/BD/41569/2007.

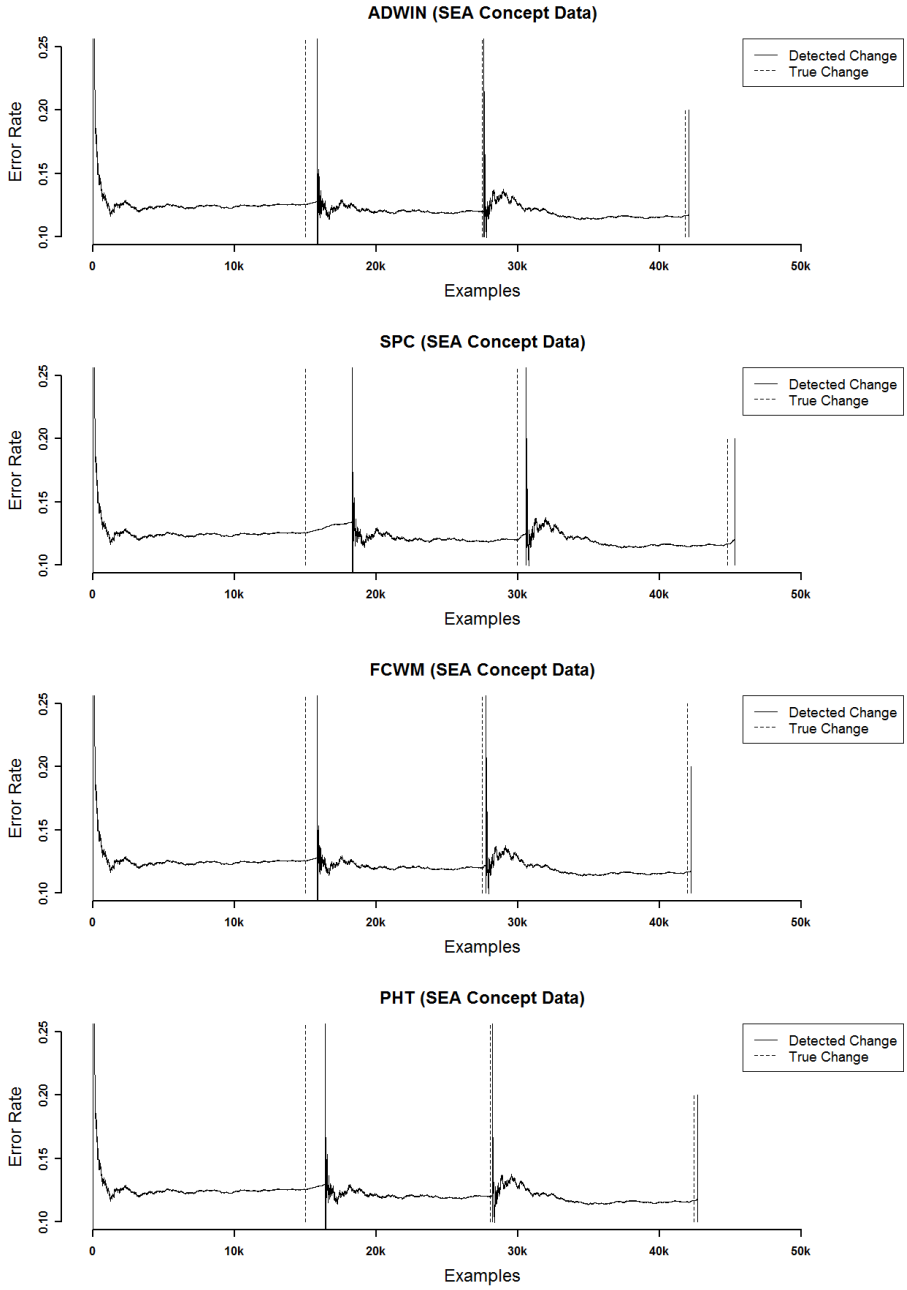

Fig. 2. Evolution of the error rate and the delay times in drift detection using the four presented methods. Vertical dashed lines indicate drift in data, and vertical lines indicate when drift was detected.

References

1. Tom Mitchell. Machine Learning. McGraw Hill, 1997.
2. R. Sebastião and J. Gama. Monitoring Incremental Histogram Distribution for Change Detection in Data Streams. In Ranga Raju Vatsavai, Olufemi Omitaomu, João Gama, Nitesh V. Chawla, Mohamed Medhat Gaber and Auroop Ganguly, editors. Lecture Notes in Computer Science (LNCS). Springer Verlag (to appear).
3. J. Gama, P. Medas, G. Castillo and P. P. Rodrigues. Learning with Drift Detection. *Advances in Artificial Intelligence - SBIA 2004*, Vol.3171 of Lecture Notes in Computer Science, pages:286-295, So Luiz, Maranhão, Brazil, October 2004. Springer Verlag.
4. G. Hulten, L. Spencer and P. Domingos. Mining Time-Changing Data Streams. ACM SIGKDD 2001. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages:97-106, 2001. ACM Press.
5. D. Kifer, S. Ben-David and J.Gehrke. Detecting change in data streams. In *VLDB 04: Proceedings of the 30th International Conference on Very Large Data Bases*, pages:180-191, 2004. Morgan Kaufmann Publishers Inc.
6. G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23:69-101, 1996.
7. R. Klinkenberg and I. Renz. Adaptive information filtering: Learning in the presence of concept drifts. In *Learning for Text Categorization*, pages:33-40, 1998. AAAI Press.
8. R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 487–494, Stanford, US, 2000. Morgan Kaufmann Publishers.
9. R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281-300, 2004.
10. M. Maloof and R. Michalski. Selecting examples for partial memory learning. *Machine Learning*, 41:27–52, 2000.
11. A. Bifet and R. Gavaldà. Learning from Time-Changing Data with Adaptive Windowing. *Proceedings of the 7th SIAM International Conference on Data Mining*, Minneapolis, Minnesota, USA, 2007.
12. J. Gama and C. Pinto. Discretization from Data Streams: applications to Histograms and Data Mining. *Proceedings of the 2006 ACM Symposium on Applied Computing*, pages:662-667, 2006.
13. E. S. Page. Continuous Inspection Schemes. *Biometrika*, Vol. 41:100-115, 1954.
14. H. Mouss, D. Mouss, N. Mouss and L. Sefouhi. Test of Page-Hinkley, an Approach for Fault Detection in an Agro-Alimentary Production System. *Proceedings of the 5th Asian Control Conference*, Vol.2, pages: 815-818, 2004.
15. M. Kubat and G. Widmer. Adapting to drift in continuous domain. *Proceedings of the 8th European Conference on Machine Learning*, pages 307-310. Springer Verlag, 1995.
16. W. N. Street and Y. Kim. A streaming ensemble algorithm SEA for large-scale classification. *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages: 377-382, 2001. ACM Press.