Using Control Charts for Detecting Concept Change in Streaming Data

Ludmila I. Kuncheva *Member, IEEE*

Abstract—We address adaptive online classification in the presence of concept change. An overview of the machine learning approaches reveals a deficit of methods for explicit detection of change when the only information is the classification error of the streaming data. We look to borrow answers from the longstanding research in monitoring process quality by using control charts. Four methods for change detection are detailed and compared in the paper: two from the machine learning literature and two control charts (Shewhart and Sequential Probability Ratio Test (SPRT)). Control charts would only signal a change. To examine empirically their effect on the classification accuracy, the Shewhart and SPRT methods were equipped with a window resizing heuristic. We chose to grow the window until a change has been detected, and shrink it to a batch size upon detection. Experiments with 28 real data sets were carried out, where change was simulated by swapping class labels. Paired t-test on the classification error and paired Wilcoxon signed rank test picked SPRT as the best change detection method.

Index Terms—Concept drift, Control charts, Changing Environments, Online classifiers

1 Introduction

THE CLAIM that classifier technology has made significant progress in the past few decades has been recently challenged [15]. One of the arguments in Hand's paper is that real-life problems change with time, while most of the pattern recognition and machine learning research is focused on inventing ever more sophisticated tools for solving static problems. However, along with the strong tendency towards perfecting static models, there does exist a rich body of literature related to changing environments (most often termed "concept drift"). Part of the problem is that this literature is fairly dispersed, and related ideas are being developed independently under different terminologies. Our study was inspired by the similarity of the task of detecting a concept change (machine learning) and that of monitoring process quality (industrial engineering). The latter can offer a spectrum of change detection methods that have been the subject of substantial theoretical and empirical research over the past 50 years.

When a classifier is faced with changes in the underlying problem, it needs a mechanism to adapt to these changes. The first question here is whether an observed variation is noise or a true change that requires some form of action. Even if the classifier is being adapted with each new observation, there is a need to detect a change and subsequently "forget" or "unlearn" previous knowledge. The easiest solution is to keep a window over the streaming data and re-train the classifier on the most recent window. The window size is crucial because it determines the flexibility of the system, which needs to match the style and pace of the changes. If the window is too small, the classifier will tend to learn all the noise in the data. Conversely, large windows will make the system inert and insensitive to changes. To alleviate the severity of this plasticity-stability dilemma, a window of variable size can be applied [39]. Detecting a change will shrink the window while a stable run will increase it to a pre-specified maximum size. Managing a variable window requires an explicit change detection.

In our scenario the classifier receives streaming data and predicts a label for each incoming observation. The true label becomes available immediately after the prediction, so the classifier "knows" whether it has been right or wrong. Unlikely as it sounds, this assumption underpins almost all existing work in classification in changing environments (concept drift, concept shift, population drift) as well as incremental learning and classification of streaming data. Although labels may not be instantly available in many real life problems, scenarios like this exist. For example, in a betting exercise, the classification (prediction) is 'win' or 'loss'. After the bets are placed and the race is over, the outcome is available straight away.

Suppose that the only parameter to be used in the change detection is this correct/wrong outcome. Thus the change detection method receives as input a binary string, and raises an alarm if a "sizable" increase in the error rate has occurred. This paper proposes to use control charts (Shewhart and the Sequential Probability Ratio Test (SPRT)) to serve as change detectors in online classifiers. The rest of the paper is organised as follows. Section 2 surveys the literature on classification in changing environments in the context of change detection. As a result, two methods, called here Window Resize Algorithm for Batch Data (WRABD) [21] and Warning Window Algorithm for Instance Data (WWAID) [13], were found and further analysed. Control charts

• *L. I. Kuncheva is with the School of Computer Science, Bangor University, Bangor Gwynedd LL57 1UT, UK.*
E-mail: l.i.kuncheva@bangor.ac.uk

are introduced in Section 3. Section 4 presents a brief simulation-based comparison between the change detection methods. Experimental results on 28 real data sets with artificially introduced concept change are given in Section 5. Section 6 contains our final remarks.

2 CHANGE DETECTION: RELATED WORKS

The approaches to handling concept drift can be categorised with respect to

1) Data chunks. Depending on the organisation of the input data, we can classify the approaches into *instance-based* (streaming data) and *batch-based* (streaming batches) [36]. The choice of approach depends largely upon the way and the speed the data is collected, and the time framework for making a decision/prediction. Streaming data can be converted into streaming batches. The reverse is also possible but batch data usually comes in massive quantities, and instance-based processing may be too time-consuming to be applicable.
2) Information used. Another possible division stems from the type of information that is used to detect a change and update the classifier model.
   - *Probability distributions.* Past data is "forgotten" if the new distribution does not accommodate it well enough [9], [14], [26], [32].
   - *Feature relevance.* New distributions may render irrelevant some features (or combinations of attribute values) that were relevant in the past [7], [11], [16], [38]. Keeping track on the best combination of predictive features (clues) makes it possible to train a relevant classifier for the most recent data distribution.
   - *Model complexity.* Some classifier models are sensitive to change in the data distribution. For example, explosion of the number of rules in rule-based classifiers or the number of support vectors in SVM classifiers may signify a concept drift.
   - *Time stamp.* The time an instance or a batch has been observed can be taken as input variable in the classifier [5], [17].
   - *Classification accuracy.* This is the most widely used criterion for implicit or explicit change detection. Included in this group are most ensemble methods for changing environments: Winnow variants [25], [27], AdaBoost variants [10], [29], "replace-the-loser" approaches [22], [23], [33], [35], [37]. Many single classifier models also evaluate the accuracy either to select the window size for the next classifier [3], [13], [19]–[21], [24] or to update the current model [2], [18], [39].
3) Change detection mode. The approaches can be split into *explicit* (detecting a change and acting upon it, for example by changing the window size) and *implicit* (using a forgetting heuristic regardless of whether or not change is suspected, for example, modifying the weights of the ensemble members or selecting instances based on their recent accuracy).
4) Classifier-specific versus classifier-free. In the classifier-specific case, the forgetting mechanism can only be applied to a chosen and fixed classifier model [2], [7] or classifier ensemble [29]. In the classifier-free case any classifier can be used because the change detection and the window update depend only on the accuracy of the classification decision, and not on the model [13], [21].

This study is focused on a general set-up that operates with minimal amount of information. We assume that the only information we have access to is a streaming data of zeros and ones, indicating whether the prediction of the class label for the respective observation has been correct or wrong. Using the classification of approaches offered above, the methods we are interested in comply with the following 4-point description:

- Data chunks = any (instance-based or batch-based);
- Information used = classification accuracy;
- Change detection mode = explicit;
- Classifier-specific versus classifier-free = classifier-free.

The following studies were found to be the closest match:

• Klinkenberg and Renz [21] propose a method for determining the window size. It uses past batches of data to estimate changes in the error. A data window is maintained, on which the current classifier is trained, called here the training window. One data batch is examined at a time. Let e_c be the error rate of the current classifier estimated on the current batch. If e_c exceeds a given threshold, then a change is signalled. A further comparison with the error on the previous batch, e_{c-1}, guides the amount by which the training window is resized. If the new error is much larger, i.e., $e_c > 1 - \beta(1 - e_{c-1})$, then a rapid change is suspected (concept shift), and the training window is reduced to contain only the latest batch. If the error on the current batch is not much larger, then the detected change is gradual (concept drift) and the training window is reduced by a factor γ. If change has not been detected at all, the current batch is added to the training window. The threshold for the detection is determined using the errors of the m previous data batches, which we call the detection window. Upon change detection, we collapse the detection window. No further change detection is allowed until the detection window reaches the size of m batches. The method is named here Window Resize Algorithm for Batch Data (WRABD), and is detailed in Figure 1.

The algorithm is designed for information filtering where a stream of documents have to be classified as relevant or non-relevant. Along with the classification

WINDOW RESIZE ALGORITHM FOR BATCH DATA (WRABD)

1) Choose the parameter values[1]:
   N, the batch size;
   m, the number of batches for evaluating the old accuracy (detection window, $m = 10$);
   f, a confidence interval parameter ($f = 5$);
   β, a factor for assessing the type of the change – gradual or abrupt ($\beta = 0.5$);
   γ, the window shrinking factor ($\gamma = 0.5$).
2) Run a warm-up stage by collecting m batches that will form the initial training window and detection window.
3) For $c = m + 1, m + 2, ...$
   a) Train the chosen classifier model with the current training window and estimate its error e_c on the current batch.
   b) If change has not been detected in the past m steps, then calculate the old error, $\bar{p}$, as the error averaged across the pervious m batches, $\bar{p} = \frac{1}{m}(e_{c-m} + \ldots + e_{c-1})$. Estimate its standard deviation $\bar{\sigma}$.
      i) If $e_c > \bar{p} + f\frac{\bar{\sigma}}{\sqrt{m}}$ and $e_c > 1 - \beta(1 - e_{c-1})$, then a rapid change is suspected (concept shift). Take the last batch as the new training window. Record detection at step c (this will collapse the detection window).
      ii) Else, if $e_c > \bar{p} + f\frac{\bar{\sigma}}{\sqrt{m}}$ but $e_c \leq 1 - \beta(1 - e_{c-1})$, the suspected change is considered to be gradual (concept drift). Reduce the training window by a factor of γ (or use the last batch if the training window becomes smaller than N). Record detection at step c.
      iii) Else (no change is detected because $e_c \leq \bar{p} + f\frac{\bar{\sigma}}{\sqrt{m}}$), add the current batch to the training window.
   c) Else (change has been detected within the past m steps) add the current batch to the training window.

[1]All parameter values are quoted from [21]. These are said to be arbitrary and not a result of an optimisation.

Fig. 1. Klinkenberg and Renz [21] window resize algorithm for batch data.

error, the authors use two more performance metrics: Recall (the probability that the classifier recognises a relevant document as relevant) and Precision (the probability that the document is actually relevant given that it was classified as relevant). The window is modified if any one of the three measures signals a change.

• The drift detection method proposed by Gama et al. [13] keeps track on the probability of error for streaming instances. The training window of instances grows until a change is detected. When the error is found to exceed a certain threshold, the system enters a *warning mode* and stores the time, t_w, of the corresponding instance. If the error drops below the threshold again, the warning mode is cancelled. However, if the error exceeds a second (higher) threshold while in the warning mode, a change is recorded. The new training window is taken to be the streaming data that came after time t_w. The classifier is re-trained and the warning and detection thresholds are re-set. Figure 2 shows the details of the algorithm (with some improvisations for the parts that were not specified in the original publication).

A modification of this algorithm due to Baena-García et al. [3] is shown to be better than the original algorithm for some data sets and worse for others. The authors take a different metric in the place of the error rate – the distance between two consecutive errors. The window resize procedure is governed by the same heuristics. There is no theoretical or intuitive justification as to why the new metric will be better than taking just the error rate. For i.i.d. binary sequences with error rate p, the distance between two consecutive errors is a random variable with geometric distribution whose expected value is $1/p$. It does not make much difference whether we evaluate p or $1/p$ and compare the value with a threshold in order to detect a change. It can be argued that the reported differences in the performance of this method and WWAID are largely due to the different uses and values of the warning and detection thresholds.

Each of the algorithms comes with procedures for both change detection and modifying the data window. The detection parts can be "cut out" and analysed with respect to their detection efficiency, similar to the way control charts are compared to one another. Change detection algorithms can be compared on the number of observations between the detection and the occurrence of the change, $\Delta =<$ time of detection $> - <$ time of occurrence $>$ (provided the observations come one at a time).

The derivation of the expression for Δ_{WRABD} is given in the appendix. For WWAID, we faced a problem because the of the choice of the detection threshold $p_{\min} + s_{\min}$. Simulation runs showed that the value of this threshold is established early on in the process when both $\hat{p}$ and $\hat{s}$ vary substantially. This sets up an arbitrary reference value which leads to spurious detection patterns. This observation does not suggest that this method is not useful in practice; it only prevents us from laying out assumptions and finding a closed form expression

DRIFT DETECTION - WARNING WINDOW ALGORITHM FOR INSTANCE DATA (WWAID)

1) Choose the parameter values: T_w the factor for the warning threshold ($T_w = 2$), T_d the factor for the detection threshold ($T_d = 3$), w_0, the warm-up window size ($w_0 = 30$). Initialise the minimum classification error $p_{\min} = \infty$ and the corresponding standard deviation $s_{\min} = \infty$. Set the warning zone flag, f_w, to false.
2) At time t ($t > w_0$), classify the new instance $\mathbf{x}_t$ using the classifier trained on the current window of size w_t. Update the error rate over the current window. Let $\hat{p}$ be the updated error rate and $\hat{s} = \sqrt{\hat{p}(1-\hat{p})/w_t}$ be the standard deviation.
3) If $\hat{p} + \hat{s} < p_{\min} + s_{\min}$, then update the minimum error by $p_{\min} = \hat{p}$ and $s_{\min} = \hat{s}$.
4) If ($\hat{p} + \hat{s} > p_{\min} + T_w s_{\min}$) and f_w = false, then switch the warning zone flag f_w = true and set up the warning time $t_w = t$.
5) If ($\hat{p} + \hat{s} < p_{\min} + T_w s_{\min}$) and f_w = true, then cancel the warning zone: f_w = false, $t_w = \infty$.
6) If ($\hat{p} + \hat{s} > p_{\min} + T_d s_{\min}$), then change has been detected. Take as the new window all the observations since t_w (size $w_{t+1} = t - t_w + 1$), set $p_{\min} = \infty$, $s_{\min} = \infty$, f_w = false, and $t_w = \infty$.* Otherwise, update the window by adding $\mathbf{x}_t$ to it (size $w_{t+1} = w_t + 1$).

*It is not explicitly specified by Gama et al. [13] what action should be taken if the warning window is too small or if there has not been a warning zone at all. A possible course of action would be to have a fixed window of minimum size (w_0) for w_0 forthcoming iterations.

Fig. 2. Gama et al. [13] warning window algorithm for streaming data

for Δ_{WWAID}.

We carry forward WRABD and WWAID because they were the only two change detection algorithms we found, which comply with the scenario defined by the 4-point description in Section 2.

3 DETECTING A CHANGE USING CONTROL CHARTS

3.1 Shewhart control chart

Control charts have long been used for monitoring process quality. Consider a streaming line of objects with probability p of an object being defective. Samples of N objects (batches) are taken for inspection at regular intervals. The number of defective objects is counted and an estimate $\hat{p}$ is plotted on the chart. It is assumed that the true value of p is known (from product specification, trading standards, etc.) Using a threshold of $f\sigma$, where $\sigma = \sqrt{p(1-p)/N}$, a change is detected if $\hat{p} > p + f\sigma$. This model is known as *Shewhart* control chart, or also p-chart when binary data is being monitored. The typical value of f is 3, but many other alternative and compound criteria have been used[1].

The expected number of batches to detection is called the *Average Run Length* (ARL). Here we are interested in the number of observations to detection, Δ_{Shewhart}. Suppose that an abrupt change has occurred and the new probability of an item being defective is p^*. The number of errors (defective items) within a batch is a binomial random variable with probability of success p^*, and number of trials N. Given that N is sufficiently large, we can approximate the error rate with a normal random variable with mean p^* and standard deviation $\sigma^* = \sqrt{p^*(1-p^*)/N}$. Assuming that the change occurs before the batch is taken for inspection, the probability of detecting a change within a single batch is

$$\begin{aligned} P_d &= 1 - Pr(\hat{p} \le p + f\sigma) \\ &= 1 - \Phi\left(\frac{p + f\sqrt{p(1-p)/N} - p^*}{\sqrt{p^*(1-p^*)/N}}\right). \end{aligned}$$

The Average Run Length is a random viable with geometric distribution, whose expected value is $1/P_d$. Therefore the number of observations to detection is

$$\Delta_{\text{Shewhart}} = \frac{N}{P_d}. \quad (1)$$

If the change occurs within batch B_c, so that K observations come from the "old" distribution (p) and the next $N - K$ come from the "new" distribution (p^*), the number of observations to detection can be represented as

$$\Delta_{\text{Shewhart}} = (N-K) + (1 - Pr(\text{detected in } B_c))\frac{N}{P_d}. \quad (2)$$

The number of defective objects in batch B_c is a sum of two binomial random variables with distributions $B(K, p)$ and $B(N-K, p^*)$, respectively. Approximating each with a normal random variables (conditions permitting), the proportion of errors in B_c can be approximated with a normal random variable with mean

$$\frac{K}{N}p + \frac{N-K}{N}p^*$$

and standard deviation

$$\frac{1}{N}\sqrt{Kp(1-p) + (N-K)p^*(1-p^*)}$$

Therefore

$$Pr(\text{detected in } B_c) = 1 - Pr(\hat{p} \le p + f\sigma)$$

$$= 1 - \Phi\left(\frac{(N-K)(p-p^*) + f\sqrt{p(1-p)N}}{\sqrt{Kp(1-p) + (N-K)p^*(1-p^*)}}\right).$$

1. http://en.wikipedia.org/wiki/Control_chart

Substituting in (2), we obtain the number of observation to change, Δ_{Shewhart}. Note that (1) is subsumed by (2) for $K = 0$, as in this case $Pr(\text{detected in } B_c) = P_d$.

Avoiding the normal approximation altogether, the predicted number of observations to detection can be expressed as

$$\Delta_{\text{Shewhart}} = (N - K) + \frac{N}{1 - \mathcal{B}(h_e, N, p)} \times \sum_{j=0}^{h_e} \sum_{i=0}^{j} b(i, K, p) b(j - i, N - K, p^*) \quad (3)$$

where $b(r, M, q)$ is the binomial probability mass function for a distribution with parameters M and q. The threshold h_e is the number of errors beyond which a change is detected. h_e and the factor correction calculated as in (10) and (11) respectively ensure that the normal approximations of P_d and $Pr(\text{detected in } B_c)$, when substituted in (2), lead to the same answer as (3).

Shewhart chart is both simple and elegant, requiring only two parameters, N and f, as p can be estimated from past observations. Figure 3 details the Shewhart algorithm in the notation of change detection.

SHEWHART

1) Choose the parameter values:
   f the factor for the confidence interval ($f = 3$);
   N, the batch size.
2) At time c, evaluate the error e_c on the current batch of data B_c.
3) Declare that change has been found if e_c exceeds the f-sigma limit, $e_c > p + f\sigma$.

Fig. 3. Shewhart change detection algorithm for streaming batches of data

Adding the following 3 extra detection criteria to the original 3-sigma rule makes up a compound detection rule known as 'Western Electric rules': (i) Two out of three consecutive points exceed the 2-sigma limit; (ii) Four out of five consecutive points exceed the 1-sigma limit; (iii) Nine consecutive points are above p. Theoretical analysis of compound detection rules is not straightforward. The most appealing approach is to describe the process as an absorbing Markov chain and evaluate the Average Run Length as the number of steps to absorption [1], [4], [8], [12], [34]. The preferred empirical approach is simulation with various types of change.

3.2 CUSUM and SPRT charts

CUmulative SUM (CUSUM) charts are deemed to be more sensitive to gradual changes than Shewhart charts [28], [31]. There are two variants called "Binomial" and "Bernoulli" CUSUM charts [31]. The idea is to keep the cumulative sum of the errors and apply a change-detection test on that. In the Binomial version, data comes in batches. Let $\{x_{t,1}, x_{t,2}, \ldots, x_{t,N}\}$ be the binary sequence for batch B_t, where $x_{t,k} = 1$ means that object k from batch B_t has been misclassified and $x_{t,k} = 0$ will indicate a correct class label. Starting with a sum $S_0 = 0$, the iterative formula for calculating the control statistic is [31]. (Reynolds and Stoumbos [31] point out that there is a slight difference in the calculation of this statistic compared to the mainstream literature. The change has been adopted for convenience of the analyses, and is said to have no practical effect on the method.)

$$S_t = \max\{0, S_{t-1}\} + \left(\sum_{i=1}^{N} x_{t,i} - N\gamma \right), \quad (4)$$

where

$$\gamma = r_1/r_2, \quad r_1 = -\ln \frac{1 - p^*}{1 - p}, \quad r_2 = \ln \frac{p^*(1 - p)}{p(1 - p^*)}. \quad (5)$$

S_t is compared with a control limit h, and if $S_t > h$, then change is signalled. In a Bernoulli CUSUM chart, the observations come one at a time (streaming instances approach). The control statistic is the same as (4) with $N = 1$. The detection rule is also the same. The value of the control limit, h, is determined in regard to all other parameters of the detection scheme. In process control it is important to choose Δ_{CUSUM} and derive h from it rather than the other way around.

Reynolds and Stoumbos [30] provide a closed form approximation of the relationship between the design parameters and the control limits for another detection scheme based on Wald's *Sequential Probability Ratio Test* (SPRT). SPRT is perceived to be even more successful than CUSUM charts. The observations come one at a time and the cumulative sum of errors at step t is compared with a reference value γt. If X denotes the error rate, the statistical test being carried out is $H_0 : X = p$ against $H_1 : X = p^*$. If the difference between the cumulative error count and the reference value is greater than a control limit h, then change is detected and H_1 is accepted. If, on the other hand, this difference is smaller than another threshold g, then H_0 is accepted. If H_0 is accepted, a new SPRT is started. In our scenario, the new SPRT is started immediately after the decision. This is repeated until a change is detected. Figure 4 shows the SPRT change detection method.

The average number of observations to signal is

$$\Delta_{\text{SPRT}} = \text{Expected number of SPRT to detection} \times \text{Expected length of SPRT to decision}$$

Suppose that the change from p to p^* occurs before the first SPRT is started. Let $P_d = 1 - \beta$ be the probability that a single SPRT detects a change (accepts H_1) when the change is present. Then

$$\text{Expected number of SPRT to detection} = \frac{1}{1 - \beta}. \quad (6)$$

Reynolds and Stoumbos [30] also provide approximations for the average length of an SPRT for both cases,

SPRT

1) Choose the parameter values: α, the probability of type I error (reject H_0 when $X = p$) and β, the probability of type II error (accept H_0 when $X = p^*$). Calculate r_1, r_2 and γ from (5), and the limits h and g using the approximations

$$h = \frac{1}{r_2} \ln\left(\frac{1-\beta}{\alpha}\right) - \frac{(1-2p)}{3}, \qquad g = \frac{1}{r_2} \ln\left(\frac{\beta}{1-\alpha}\right).$$

2) At time t, start an SPRT setting its counter to $j = 1$. Let $x_{t,j}$ be the jth (0/1) observation coming after SPRT was started.
   - If $\sum_{i=1}^{j} x_{t,j} - j\gamma \leq g$, accept H_0, set $j = 1$, and start a new SPRT.
   - else, if $\sum_{i=1}^{j} x_{t,j} - j\gamma \geq h$, accept H_1, declare a change and STOP.
   - else, set $j = j + 1$ and continue the current SPRT.

Fig. 4. SPRT change detection algorithm for streaming data [30]

when $X = p$ and when $X = p^*$. Let $L(\zeta)$ be the expected length of an SPRT process until decision is made either way, given that observations come with probability $X = \zeta$. Then

$$L(p^*) \approx \frac{(1-\beta)\ln\frac{1-\beta}{\alpha} + \beta\ln\frac{\beta}{1-\alpha}}{r_2 p^* - r_1} \tag{7}$$

and

$$L(p) \approx \frac{\alpha\ln\frac{1-\beta}{\alpha} + (1-\alpha)\ln\frac{\beta}{1-\alpha}}{r_2 p - r_1} \tag{8}$$

From (6) and (7),

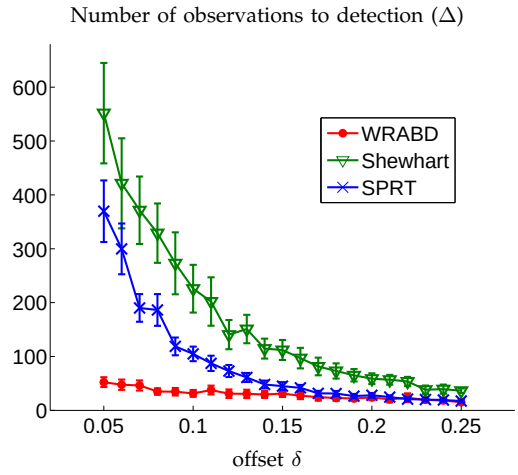
$$\Delta_{\text{SPRT}} = \frac{L(p^*)}{1-\beta}$$

The scenario where a true change is being detected is called 'out of control' whereas a false detection scenario is called 'in control'. This terminology comes from process monitoring where 'no change' means that the process is in control and an alarm should be raised if the measured quantity slips out of control.

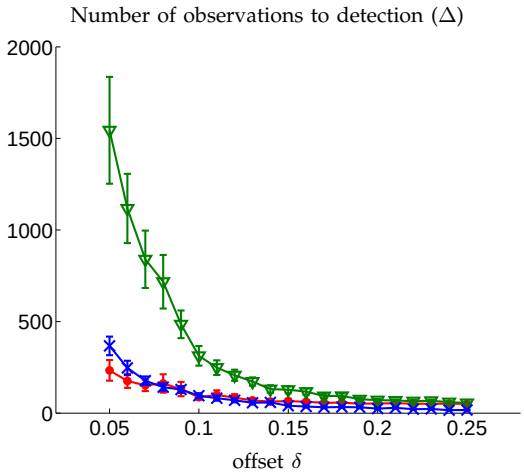
4 Comparisons between the change detection methods

Figure 5 shows the number of observations to detection (Δ) for WRABD, Shewhart and SPRT for two batch sizes, $N = 10$ and $N = 50$. 100 runs were carried out where a random i.i.d. binary sequence was generated for each run and each method. The value of p^* was varied as $p^* = p + \text{offset}$, where 'offset' took all the values in $\{0.05, 0.06, \ldots, 0.25\}$. The observations were generated and fed to the detection method one at a time, and the process was stopped as soon as change was detected. The number of observations to detection was stored for each run. The detection window length for WRABD was $m = 5$ batches and the detection factor was $f = 3$. The values of all the other parameters were fixed as in the algorithms' descriptions.

Displaying WWAID results would have been misleading because of the large percentage of failed detections. As explained earlier, if change was detected, Δ would be competitive to that of the other methods but in about half of the runs, change was not detected within the first 2000 observations. Thus averaging the existent detection lengths and ignoring the non-detection will give a false impression of the method.



(a) $N = 10$



(b) $N = 50$

Fig. 5. Average Δ and 95% confidence intervals obtained from 100 runs with $p = 0.20$ and $p^* = p + \delta$. The warm-up length for WRABD was $m = 5$ batches.

It appears that for the chosen settings, the best algorithm is WRABD followed by SPRT. However, to evaluate the methods in full, we need to consider their in-control behaviour. Table 1 provides the number of observations to false detection for the 4 methods. The estimated values are the averages of 100 simulation runs with $p^* = p = 0.2$. The 95% confidence intervals are also shown.

TABLE 1
Simulation results ($\pm$ 95% confidence interval), and the predicted number of observations to false alarm (in-control)

	$N = 10$		$N = 50$	
Algorithm	Simulation	Predicted	Simulation	Predicted
WWAID	196 $\pm$215	–	439 $\pm$288	–
WRABD	62 $\pm$10	60	372 $\pm$58	371
Shewhart	1614 $\pm$340	1570	20022 $\pm$3656	19911
SPRT	968 $\pm$162	978	1068 $\pm$212	978

The in-control Δ_{SPRT} is $L(p)/\alpha$. To calculate $L(p)$ (8), we need to specify a predicted value of p^* in order to be able to initialise the parameters of the SPRT. The results in the table are for $p^*_{\mathrm{predicted}} = 0.35$. In the case of WWAID, the detection failure, which was a severe drawback for the out-of-control scenario, can be counted as an asset now.

Taking into account both out-of-control and in-control behaviour, WRABD and SPRT stand out as the two main rivals. While WRABD is quicker to detect small changes, it is also quicker to raise a false alarm. The difference between the methods are less pronounced for larger values of the offset δ. Thus if the change in the classification problem is drastic (e.g., when some class labels are swapped), the methods may be ranked differently.

Another caveat is that the analyses of Shewhart and SPRT always assume that p and p^* are given in advance. While this is reasonable in monitoring established processes, in classification we only have online estimates (not necessarily very good ones) of these probabilities. Therefore it is imperative to evaluate the methods in a real-world scenario. The main question is still whether using one algorithm or another makes a substantial difference to the classification accuracy in changing environments.

Prediction formulas for a linear trend for the Shewhart and CUSUM algorithms have been derived elsewhere [4]. The analysis here only applies to abrupt change, and the simulations were run to reflect this.

5 Experimental results

5.1 Datasets

Twenty eight real data sets were used in the experiment. They are shown in Table 2 sorted by the total number of objects. The features in all data sets are numerical and there are no missing values.

TABLE 2
Datasets used in the experiment

Dataset	Features	Classes	Objects	Source
iris	4	3	150	UCI[1]
wine	13	3	178	UCI
sonar	60	2	208	UCI
laryngeal1	16	2	213	Collection[2]
glass	9	6	214	UCI
thyroid	5	3	215	UCI
votes	16	2	232	UCI
voice3	10	3	238	Collection
breast	9	2	277	UCI
heart	13	2	303	UCI
liver	6	2	345	UCI
spect	44	2	349	Collection
ionosphere	34	2	351	UCI
laryngeal3	16	3	353	Collection
voice9	10	9	428	Collection
wbc	30	2	569	UCI
palynomorphs	31	3	609	private[3]
laryngeal2	16	2	692	Collection
pima	8	2	768	UCI
vehicle	18	4	846	UCI
vowel	11	10	990	UCI
german	24	2	1000	UCI
image	19	7	2310	UCI
scrapie	14	2	3113	private[4]
spam	57	2	4601	UCI
phoneme	5	2	5404	UCI
satimage	36	6	6435	UCI
pendigits	16	10	10992	UCI

[1]UCI [6] http://www.ics.uci.edu/~mlearn/MLRepository.html
[2]Collection http://www.informatics.bangor.ac.uk/~kuncheva/activities/real_data_full_set.htm
[3]Images of pieces of kerogen extracted from microscope images of palynomorphs
[4]Data on scrapie disease in sheep (related to BSE in cows), provided by DEFRA, UK, http://www.defra.gov.uk/

5.2 Experimental set-up

The online linear discriminant classifier (O-LDC) was used in all experiments. Together with the four change detection methods considered hitherto, we coded the do-not-update and the fixed window methods. The size of the window was set at $N = 50$, and this was adopted as the batch size for all methods. With each data set, we first took aside a 10% stratified sample to be used for testing. Another stratified sample of $N = 50$ objects was taken from the remaining 90% of the data for training the initial classifier.[2] The remaining part of the training data was augmented to size 1000, and shuffled to simulate i.i.d streaming data.

As an example, consider a data set of 400 objects. A random subsample of 40 objects is removed to be used for testing, leaving 360 objects for training. Another random stratified sample of 50 objects will train the initial classifier. The remaining 310 objects will be pooled with a random sample of 690 taken from that set, to make up the online data of 1000 objects.

One object from the online data set is submitted at

2. When N was less than the number of features, the initial covariance matrix was taken to be the identity matrix, reducing the linear discriminant classifier to the nearest mean classifier with Euclidean distance.

a time and labelled by the current classifier. As it is assumed that the true label becomes available immediately, we know whether the object has been correctly labelled. This 0 or 1 is fed to the respective detection method. The batch-based methods (Fixed Window, WRBAD and Shewhart) will only issue a signal after they have seen a whole batch of 50 objects. The instance-based methods (WWAID and SPRT) may signal at any observation.

The parameter choices for the methods were not optimised. They are given in Table 3.

TABLE 3
Parameter values used in the experiments. (The notations are as in the method description)

Fixed window	WWAID	WRABD	Shewhart	SPRT
$N = 50$	$w_0 = 50$	$N = 50$	$N = 50$	$\alpha = 0.05$
–	$T_w = 2$	$f = 3$	$f = 3$	$\beta = 0.05$
–	$T_d = 3$	$m = 5$	–	–
–	–	$\beta = 0.5$	–	–
–	–	$\gamma = 0.5$	–	–

If change is detected, the window is resized. The resizing heuristics for WRABD and WWAID are specified within the method descriptions. For Shewhart and SPRT we adopted the following simple rule. While there is no change, keep the window growing. Here we use the fact that O-LDC can be updated in constant time with each new observation, and so the training window can be practically unlimited. As soon as change is detected, the window is shrunk to the last batch of 50 objects and a new linear classifier is trained on that.

The analyses and the simulations in Section 4 assumed a "sterile" environment where the error of the classifier did not change along with the constant updates of the classifier. In real-life situations, the classifier may change with each new observation or batch, so the thresholds for the Shewhart and SPRT methods are not constant. What is more, there is no pre-assigned values for p and p^*, so the initial thresholds must be estimated from data. The strategy adopted here was as follows. The initial thresholds were estimated using the training error of the initial classifier, e_{training}. The thresholds were updated with each new observation. The value p needed for the update was calculated as the proportion of wrong predictions since last change detection or since the start of the experiment, if there has been no detection. The predicted value of p^* needed for initialising SPRT was set at $p^*_{\text{predicted}} = e_{\text{training}} + 0.3$, and was updated at each step, together with p, using the same offset of 0.3. We put a lower limit N for the size of the detection window for the instance-based methods SPRT and WWAID. In case the last detection was sooner than N steps back, the classifier kept being updated but change detection was withheld, until the detection window reached size N.

The classification error was evaluated on the testing set after each new observation, giving a sequence of 1000 values. This experiment was run 100 times for each data set and each method. The runs were synchronised for all 6 detection methods. This means that, in each run, all methods received exactly the same training, testing and online data sets, and the 1000 objects in the online data set were submitted in the same order.

We simulated abrupt changes at step 400 and then at step 800. Following Klinkenberg's model, the change was implemented by picking randomly two classes in the data set and swapping their labels. If there were only two classes originally, the labels would be swapped at step 400 and returned to the initial labels at step 800.

5.3 Results

Figure 6 displays the results with the 6 methods. Each plot shows the classification error as a function of "time", i.e., the number of the incoming observations. The plotted error values are the averages across the data sets and across the 100 runs for each data set. Clearly data sets have very different errors and an average estimate of the error would not make sense as a value. However, we observed a compelling similarity among the patterns of the error across the data sets, and this justifies the choice of display. As a reference, in each subplot we show also the error pattern with the Fixed window method.

The change detection patterns of WWAID, WRABD, Shewhart and SPRT are displayed in Figure 7. The x-axis corresponds again to the incoming observations. The y-axis represents the average number of times (out of 100 runs), a change has been detected at observation x. The ideal pattern is also shown.

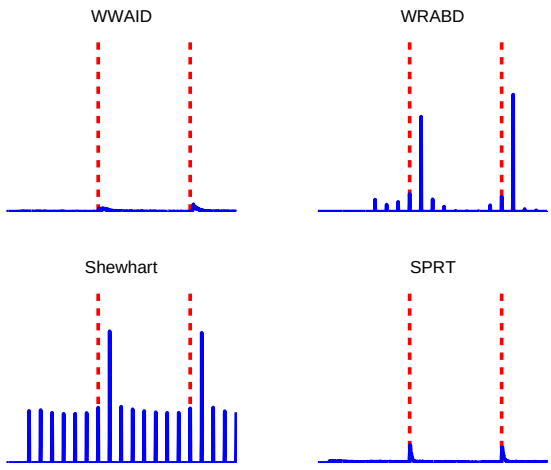


Fig. 7. Average number of detections out of 100 runs. The ideal detection pattern with singleton peaks at observations 400 and 800 (height 100) is also plotted.

To measure the overall quality of a method we chose the average error across the 1000 incoming observations. Let $E_{i,j,k}$ be the error of the k-th run for method i and data set j, where $k = 1, \ldots, 100$, $i = 1, \ldots, 6$, $j = 1, \ldots, 28$. The methods were ranked on each data set from 1 (best) to 6 (worst) using $\bar{E}_{i,j,\bullet}$, where index $\bullet$ means average across the possible values of the index. Table 4 shows the 6 methods, sorted by their average

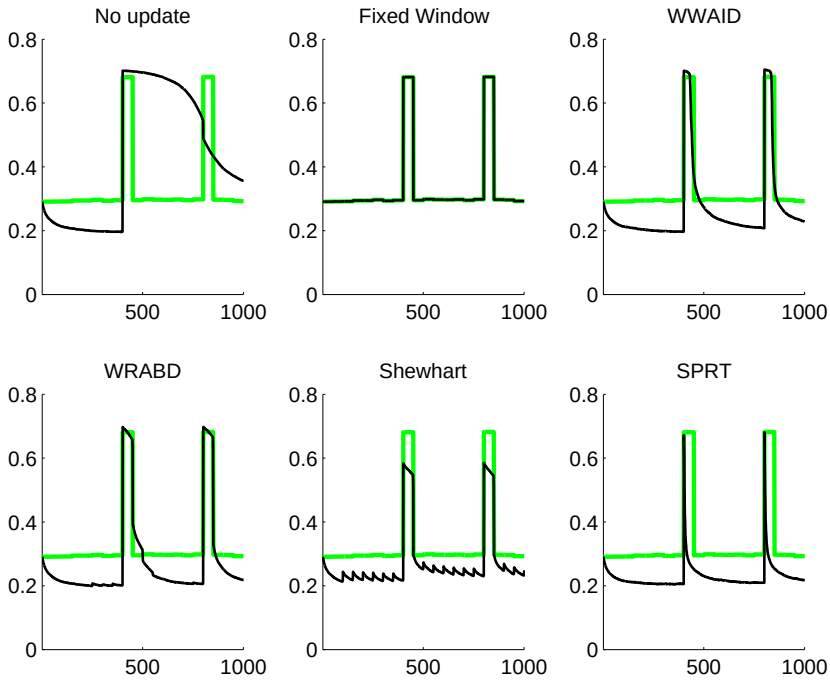


Fig. 6. Classification errors for the 6 window updating methods. The Fixed window pattern is shown in each plot for reference.

ranks. Friedman's ANOVA on the ranks shows that there are significant differences between the 6 methods ($\chi^2 = 108.43$, df = 5, $p \approx 0$). The entries in Table 4 show the p-values for the paired Wilcoxon signed rank test. The input for each test was the paired data for methods i_1 and i_2 and data set j, for example, $E_{i_1,j,\bullet}$ and $E_{i_2,j,\bullet}$ for $j = 1, \ldots, 28$. Values below 0.01 indicate a significant difference between the methods.

The table shows that SPRT outperforms all the other methods, followed by the group of the three change-detection methods. Fixed window is better than No update but worse than WRABD. Schematically,

$$\text{SPRT} \; > \; \{\text{Shewhart, WWAID, WRABD}\} \; > \; \text{Fixed window} \; > \; \text{No update}$$

To detail the comparison even further, paired t-test was carried out for each pair of methods and each data set. The paired data for methods i_1 and i_2 and data set j, for example, were $E_{i_1,j,k}$ and $E_{i_2,j,k}$ for $k = 1, \ldots, 100$. Each method participates in 5×28 = 140 comparisons. Table 5 shows the win/loss scores. Column 'Wins' is the total number of pairwise comparisons where the method was found significantly better ($\alpha = 0.05$) than the competitor and column 'Losses' is the number of comparisons lost against the other method. The number of draws is therefore 140−Wins−Losses. The last column shows a performance index (PI) calculated as Wins−Losses. Note the significant gap in PI between SPRT and the next best method (Shewhart) and the one between WRABD and the Fixed window method.

Further comments on the individual methods are given below

TABLE 5
Win/Loss results and
Performance Index (PI) = Wins – Losses

Method	Wins	Losses	PI
SPRT	133	3	130
Shewhart	82	49	33
WWAID	80	47	33
WRABD	69	57	12
Fixed window	25	112	–87
No update	8	132	–124

1) *No-update.* The No-update method is expectedly the worst. It shows a seemingly strange drop in the error from step 800 onwards. The reason for this is that a large proportion of the data sets consist of two classes only. In this case, from step 800 to step 1000 the class assignment will be exactly as the original one, to which the classifier has been tuned for the first 400 steps. The classifier "still remembers" the first part of the data set, and this puts it in a privileged position. The initial dip in the error is not surprising either. Since we use O-LDC with all methods, the difference in the errors will only come from insufficient or inadequate training data. In the case of the No-update method, the training set grows without interruption, leading to a stable lowest error, until the first change.
2) *Fixed window.* The behaviour of the Fixed Window classifier is also consistent with the expectation. The classifier shows stable performance on all the batches except on the ones where the old classifier

TABLE 4
Average ranks and Wilcoxon signed rank test results

	Method	Average rank	2	3	4	5	6
1	SPRT	1.11	0.000004	0.000004	0.000011	0.000004	0.000004
2	Shewhart	2.86		0.909351	0.412347	0.000004	0.000005
3	WWAID	3.04			0.284504	0.000004	0.000004
4	WRABD	3.29				0.000046	0.000004
5	Fixed window	5.00					0.001129
6	No update	5.71					

is completely inadequate. The error is high for one batch only, and drops to the basic value with the following batch. Apparently, training O-LDC with a larger sample than $N = 50$ is beneficial, as demonstrated by WWAID, Shewhart, SPRT and even No-update.

3) *WWAID.* On the simulations, WWAID was found to have quite a large variability in the number of observations to detection Δ. Nevertheless, this method seems to detect the changes quickly enough, at the same time keeping the false detection rate to minimum. The crucial factor for this good behaviour is that the changes in the error are drastic, larger than the offset δ used in the previous simulations.
4) *WRABD.* WRABD is a close competitor to the control chart methods. It does not rely on pre-assigned threshold values. In fact, its detection pattern is more accurate than that of Shewhart (Figure 7). The batch update, however, is slower to follow the detection than an instant update as in SPRT, which comes as a disadvantage in terms of overall accuracy.
5) *Shewhart.* The "chainsaw effect" seen in Figure 6 comes from the multiple false detections as indicated in Figure 7. This is inconsistent with the simulation results in Section 4 which rank Shewhart as the method with the best in-control behaviour, i.e., largest Δ to false alarm. The reason for this discrepancy is the way we chose to update the threshold. In the original method the threshold is set in advance to the correct *known* value of $p + 3\sigma$. Here the threshold was updated at each step as the average error since the last detection. This choice may have put the Shewhart method at disadvantage. The same strategy was applied to SPRT but it seemed to have coped with the variable threshold much better.
6) *SPRT.* The abrupt changes simulated by swapping class labels lead to a drastic increase in error, which is expected to be picked within the first batch by both Shewhart and WRABD methods. However, the batch detection methods can only signal a change after a whole batch while SPRT can do that at any time. This explains the rapid dip in the error shortly after the change occurs.

The overall results from the experimental study favour SPRT followed by Shewhart, WWAID and WRABD.

6 Conclusions

We investigated the suitability of control charts, specifically Shewhart and SPRT, as change detection methods for classification in changing environments. Two methods found in the machine learning literature, termed here WWAID and WRABD, were also studied. The analysis and the simulations of abrupt change did not pick a clear winner among the methods, but favoured SPRT and WRABD followed by Shewhart.

Since the acid test for all four methods is their classification performance, we simulated abrupt change on real data sets by swapping class labels. We augmented the set of methods with two straw men: the No update classifier, where no changes were being detected, and the Fixed window classifier, where a new classifier was trained on each new batch of the data. The same Online Linear Discriminant Classifier (O-LDC) was applied for all 6 change-detection methods. The results demonstrate the potential of the two control charts. As a by-product, these methods come with ready made ARL analyses and closed form solutions for specific types of change, e.g., abrupt and linear. This will enable theoretical comparisons between different detection methods prior to embedding one of them into an online classifier.

We should be cautious and not read too much in the dominance of SPRT that came as the result of the experiments. First, the change introduced by swapping labels was so drastic, that all methods were likely to detect it within one batch. Being an instance-based method, SPRT was capable of detecting a change earlier than a full batch size. Second, the success of SPRT and Shewhart can be attributed, to some extent, to the serendipitous choice of a window resizing heuristic and parameter values. This comes to highlight the importance of all the choices that accompany the design of any online classification system. Third, this study looks at abrupt changes while there are many other possible change simulation scenarios.

Our future investigation plans include analysis and design of ensembles of control chart detectors. It would be interesting to derive expressions for Δ (number of

observations to detection) for Shewhart and SPRT when the thresholds are not fixed in advance but depend on estimates from the streaming data.

APPENDIX

Analysis of the WRABD

Here we derive an expression for the detection time Δ_{WRABD} of the WRABD algorithm (Figure 1). To be able to do so, consider the following scenario. Assume that the data has been coming from a stationary distribution, long enough so that the classifier does not benefit from expanding the training data window any further. Denote by p the true error rate of the classifier. At some time point, a rapid change of the underlying distribution occurs so that the error rate increases to p^* on the new distribution. To ease the analysis we shall assume that the change occurs after batch B_{c-1} and before batch B_c. Let t_d be the batch number at which a change is detected.

The previous errors $e_{c-m}, \ldots, e_{c-1}$ needed for calculating $\bar{p}$ are estimated using the respective current classifiers at batches $c-m, \ldots, c-1$. As we assumed that the classifier model is already fully trained, the variability in $\bar{p}$ will only be due to the different testing batches. Since they all come from the distribution before the change, the total number of errors in m batches is a binomial random variable with parameters mN and p. For large enough mN, the average error across the past m batches can be approximated with a normal random variable X with mean p and standard deviation $\sigma = \sqrt{p(1-p)/(mN)}$.

The number of errors e_c in batch B_c, coming from the new distribution, is itself a binomial random variable. Provided that $p^*N \geq 5$ and $(1-p^*)N \geq 5$, the error rate can be approximated by a normal random variable Y with mean p^* and standard deviation $\sigma^* = \sqrt{p^*(1-p^*)/N}$. Form a new random variable $\zeta = Y - X$. It will have normal distribution with mean $p^* - p$ and standard deviation $\sqrt{(\sigma^*)^2 + (\sigma)^2}$.

The detection inequality in WRABD can be reformatted as $e_c - \bar{p} > f\frac{\bar{\sigma}}{m}$. Note that $\frac{\bar{\sigma}}{m}$ is an estimate of σ. To make the analysis feasible, we assume that σ is given and fixed, so the detection threshold does not depend on data. The detection inequality becomes $e_c - \bar{p} > f\sigma$. Restrictive, as it may be, without this assumption the the analysis becomes more complicated, and is hardly justified for the purposes of the current study.

The probability that the change is detected within a single batch (B_c) of size N is

$$\begin{aligned} P_d(0) &= Pr(\zeta > f\sigma) \\ &= 1 - Pr(\zeta \leq f\sigma) \\ &= 1 - \Phi\left(\frac{f\sigma - p^* + p}{\sqrt{(\sigma^*)^2 + (\sigma)^2}}\right), \end{aligned}$$

where Φ is the cumulative distribution function for the normal distribution.

The probability $P_d(k)$ that the change is detected at batch B_{c+k}, where $1 \leq k < m$, needs a further elaboration. In this case $m-k$ batches from the old distribution and k batches from the new distribution will be used in the calculation of $\bar{p}$. The number of errors in the m batches will be a sum of two binomial variables with distributions $B((m-k)N, p)$ and $B(kN, p^*)$. Taking the normal approximations of both, the error rate variable Y, associated with $\bar{p}$, will be a normal random variable with mean $\frac{m-k}{m}p + \frac{k}{m}p^*$ and standard deviation

$$\sigma = \sqrt{\frac{(m-k)p(1-p) + kp^*(1-p^*)}{m^2 N}}.$$

Recall the assumption in this scenario: adding new training data from the original distribution will not change the classifier. However, until detection occurs, the training data will be augmented with batches from the new distribution. We need to assume in addition that adding batches from the new distribution is not going to "dilute" too much the training set and disturb the learned classifier. In other words, the classifier is supposed to have the same probability of error p^* on all unseen batches coming from the new distribution.

Using again $\zeta = Y - X$, the probability that the change occurring at c is detected at batch B_{c+k}, where $1 < k < m$ is

$$P_d(k) = Pr(\zeta > f\sigma) = 1 - \Phi\left(\frac{f\sigma + \frac{m-k}{m}(p - p^*)}{\sqrt{(\sigma^*)^2 + (\sigma)^2}}\right).$$

For any $k \geq m$, we will have a static distribution with probability of error p^*. Then the mean of ζ is zero, and $\sigma = \sqrt{p^*(1-p^*)/mN} = \sigma^*/\sqrt{m}$. Hence

$$P_d(m) = Pr(\zeta > f\sigma) = 1 - \Phi\left(f\sqrt{\frac{m}{m+1}}\right)$$

Regarding $P_d(m)$ as probability of "success", the number of batches to detection is a random variable with a geometric distribution with parameter $P_d(m)$ and expected value $1/P_d(m)$ (batches). Therefore, if we fall in the "non-detection zone" where $k \geq m$, then we can expect

$$\Delta_{\text{WRABD}}^{\text{non-detection}} = N\left((m-1) + \frac{1}{P_d(m)}\right)$$

observations to detecting a change.

To find the total number of observations to detecting a change for WRABD, we create a discrete random variable taking values in the set $V = \{0, 1, 2, \ldots, m-1\}$. These values correspond to the possible number of batches to detection starting with B_c. The probability mass function is

$$\begin{aligned} P(i) &= Pr(\text{detection in batch } B_{c+i}) \\ &= \frac{1}{Z}P_d(i)\prod_{k=0}^{i-1}(1 - P_d(k)), \quad 1 \leq i \leq m-1, \end{aligned}$$

where Z is a normalising constant which amounts to the probability of detecting a change within m batches

including B_c.

$$Z = 1 - \prod_{k=0}^{m-1}(1 - P_d(k)).$$

For $i = 0$, $P(0) = P_d(0)$. The expected number of observation according to this random variable is

$$\Delta_{\text{WRABD}}^{\text{detection}} = N\left(1 + \frac{1}{Z}\sum_{i=1}^{m-1} i\, P_d(i) \prod_{k=0}^{i-1}(1 - P_d(k))\right).$$

The 1 in the brackets reflects the fact that B_c needs to be counted as well. The total number of observation to change is

$$\begin{aligned}\Delta_{\text{WRABD}} &= Z\Delta_{\text{WRABD}}^{\text{detection}} + (1-Z)\Delta_{\text{WRABD}}^{\text{non-detection}} \\ &= N\left(Z + \sum_{i=1}^{m-1} i\, P_d(i) \prod_{k=0}^{i-1}(1 - P_d(k))\right) \\ &+ (1-Z)N\left((m-1) + \frac{1}{P_d(m)}\right)\end{aligned}$$

Continuity correction. Due to the discrete nature of the binomial distribution the theoretical calculation of Δ using normal distribution is inaccurate, especially for small N. Consider the following example. Suppose that we are investigating the case where batches of size N are taken and the number of errors is compared to a threshold h_e. Alarm is triggered if the number of errors within the batch *strictly exceeds* h_e. Let $N = 10$ and $p = 0.2$. According to the WRABD detection step (with factor $f = 3$),

$$h_e = \left\lfloor N \times \left(p + 3\sqrt{p(1-p)/N}\right)\right\rfloor = \lfloor 10{\times}0.5795 \rfloor = 5 \text{ errors.}$$

This means that an alarm will be raised if strictly more than 5 observations from the batch of 10 are misclassified. The 3σ-confidence interval, assuming normal distribution, gives probability of false alarm (type I error) $\alpha = 0.00135$. The number of observations to alarm, predicted using the normal approximation, is $N/\alpha \approx 7408$. Using the binomial distribution, the probability of type I error will be $\alpha_B = 1 - \mathcal{B}(h_e, N, p) \approx 0.00637$, where $\mathcal{B}$ is the cumulative binomial distribution function. Hence the true number of observations to alarm is $N/\alpha_B \approx 1570$.

To correct for the error in the theoretical prediction, we adjust the factor for the confidence interval, f, and still use the normal approximation of the binomial distribution. The adjusted factor f' is

$$f' = \Phi^{-1}(1 - \alpha_B) = \Phi^{-1}(\mathcal{B}(h_e, N, p)). \quad (9)$$

where

$$h_e = \left\lfloor N \times \left(p + f\sqrt{p(1-p)/N}\right)\right\rfloor. \quad (10)$$

For the example above, $f' = \Phi^{-1}(\mathcal{B}(5, 10, 0.2)) = 2.4910$. With this f', the adjusted type I error is $\alpha' = \alpha_B$, and the predicted number of observations to detection is the correct one.

Consider the more general case where we have a binomial random variable X with parameters N and p^* and a threshold h_e calculated through (10) for a fixed p. The adjusted factor, f'', is obtained in the following way. The probability for $X > h_e$ is $1 - \mathcal{B}(h_e, N, p^*)$. Using the normal approximation, $Pr(X > h_e) = 1 - \Phi((p + f''\sigma - p^*)/\sigma^*)$, where σ and σ^* are the respective standard deviations. As both expressions amount to the same probability, $Pr(X > h_e)$, we can take $\mathcal{B}(h_e, N, p^*) = \Phi((p + f''\sigma - p^*)/\sigma^*)$ and solve for f''

$$f'' = \frac{1}{\sigma}(\sigma^*\Phi^{-1}(\mathcal{B}(h_e, N, p^*)) + p^* - p). \quad (11)$$

Notice that f'' reduces to f' for $p^* = p$. ■

We carried out simulation runs to illustrate the match between the empirical and the theoretical Δ_{WRABD}. With the continuity correction in place, Figure 8 shows Δ_{WRABD} as a function of p^* along with two simulation curves. The parameter values were: batch size $N = 10$ and $N = 50$, past batches $m = 5$, confidence interval factor $f = 3$ and fixed $p = 0.2$. Each of the empirical curves is the average of 100 simulation runs. The 95% confidence intervals are also shown.

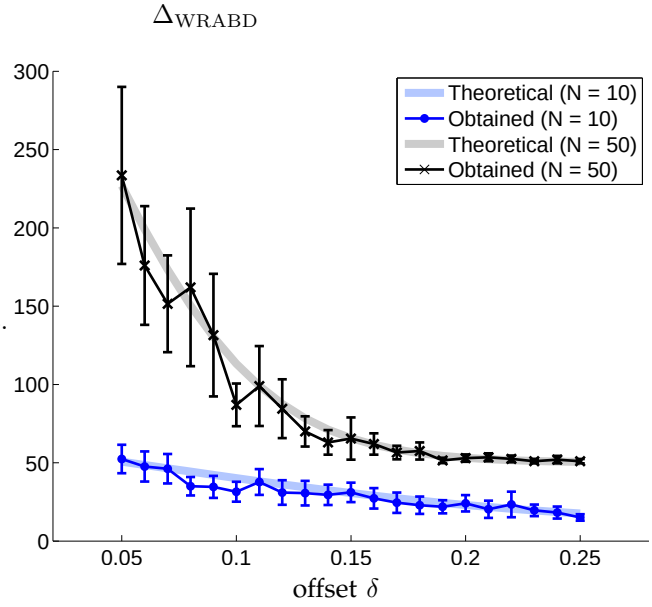


Fig. 8. Number of observations to detection of change for the WRABD algorithm.

ACKNOWLEDGEMENTS

I am grateful to Chris Whitaker and Juan Rodríguez for the insightful discussions and for pointing to me crucial reference sources which fuelled the inspiration for this study. This work was sponsored by EPSRC grant #EP/D04040X/1.

REFERENCES

[1] C. A. Acosta-Mejia. Improved p charts to monitor process quality. *IIE Transactions*, 31:509–516, 1999.

[2] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[3] M. Baena-García, J. Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno. Early drift detection method. In *Fourth International Workshop on Knowledge Discovery from Data Streams*, pages 77–86, 2006.

[4] A. F. Bissel. The performance of control charts and cusums under linear trend. *Applied Statistics*, 33:145–151, 1984.

[5] M.M Black and R.J. Hickey. Maintaining the performance of a learned classifier under concept drift. *Intelligent Data Analysis*, 3(6):453–474, 1999.

[6] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[7] A. Blum. Empirical support for Winnow and weighted-majority based algorithms: results on a calendar scheduling domain. In *Proc. 12th International Conference on Machine Learning*, pages 64–72, San Francisco, CA., 1995. Morgan Kaufmann.

[8] D. Brook and D.A. Evans. An approach to the probability distribution of cusum run length. *Biometrika*, 59:539–549, 1972.

[9] S. J. Delany, P. Cunningham, and A. Tsymbal. A comparison of ensemble and case-based maintenance techniques for handling concept drift in spam filtering. Technical Report TCD-CS-2005-19, Trinity College Dublin, 2005.

[10] W. Fan, S. J. Stolfo, and J. Zhang. Application of adaboost for distributed, scalable and on-line learning. In *Proc KDD-99*, San Diego, CA, 1999. ACM Press.

[11] G. Forman. Tackling concept drift by temporal inductive transfer. Technical Report HPL-2006-20(R.1), HP Laboratories Palo Alto, June 2006.

[12] J. C. Fu, G. Shmueli, and Y. M. Chang. A unified Markov chain approach for computing the run length distribution in control charts with simple or compound rules. *Statistics & Probability Letters*, 65:457–466, 2003.

[13] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence*, volume 3171 of *Lecture Notes in Computer Science*, pages 286–295. Springer Verlag, 2004.

[14] V. Ganti, J. Gehrke, and R. Ramakrishnan. Mining data streams under block evolution. *ACM SIGKDD Explorations Newsletter*, 3:1–10, 2002.

[15] D.J. Hand. Classifier technology and the illusion of progress (with discussion). *Statistical Science*, 21:1–34, 2006.

[16] M. Harries and K. Horn. Detecting concept drift in financial time series prediction using symbolic machine learning. In *Eighth Australian Joint Conference on Artificial Intelligence*, pages 91–98, Singapore, 1995. World Scientific Publishing.

[17] R.J. Hickey and M. M. Black. Refined time stamps for concept drift detection during mining for classification rules. In *Proceedings of the International Workshop on Temporal, Spatial and Spatio-Temporal Data Mining (TSDM2000), Lecture Notes in Artificial Intelligence*, volume 2007, pages 20–30. Springer-Verlag: Berlin, 2000.

[18] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *In Proc. 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–106. ACM Press, 2001.

[19] R. Klinkenberg. Using labeled and unlabeled data to learn drifting concepts. In *Workshop notes of the IJCAI-01 Workshop on Learning from Temporal and Spatial Data*, pages 16–24, Menlo Park, CA, USA, 2001. IJCAI, AAAI Press.

[20] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 487–494, San Francisco, CA, USA, 2000. Morgan Kaufmann.

[21] R. Klinkenberg and I. Renz. Adaptive information filtering: Learning in the presence of concept drifts. In *AAAI-98/ICML-98 workshop Learning for Text Categorization*, Menlo Park,CA, 1998.

[22] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Proc 3rd International IEEE Conference on Data Mining*, pages 123–130, Los Alamitos, CA, 2003. IEEE Press.

[23] L. I. Kuncheva. *Combining Pattern Classifiers. Methods and Algorithms*. John Wiley and Sons, N.Y., 2004.

[24] M.M. Lazarescu and S. Venkatesh. Using selective memory to track concept drift effectively. In *Intelligent Systems and Control*, volume 388, Salzburg, Austria, 2003. ACTA Press.

[25] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[26] M. Markou and S. Singh. Novelty detection: A review, Part I: Statistical approaches. *Signal Processing*, 83(12):2481– 2521, 2003.

[27] C. Mesterharm. Tracking linear-threshold concepts with winnow. *Journal of Machine Learning Research*, 4:819–838, 2003.

[28] E. S. Page. Continuous inspection schemes. *Biometrika*, 41:100–114, 1954.

[29] R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, 31(4):497–508, 2001.

[30] M. R. Reynolds Jr and Z. G. Stoumbos. The SPRT chart for monitoring a proportion. *IIE Transactions*, 30:545–561, 1998.

[31] M. R. Reynolds Jr and Z. G. Stoumbos. A general approach to modeling CUSUM charts for a proportion. *IIE Transactions*, 32:515–535, 2000.

[32] M. Salganicoff. Density-adaptive learning and forgetting. In *Proceedings of the 10th International Conference on Machine Learning*, pages 276–283, 1993.

[33] K. O. Stanley. Learning concept drift with a committee of decision trees. Technical Report AI-03-302, Computer Science Department, University of Texas-Austin., 2003.

[34] S. H. Steiner. Grouped data exponentially weighted moving average control charts. *Applied Statistics*, 47:203–216, 1998.

[35] W. N. Street and Y. S. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 377–382. ACM Press, 2001.

[36] A. Tsymbal. The problem of concept drift: definitions and related work. Technical Report TCD-CS-2004-15, Trinity College Dublin, April 2004.

[37] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept drifting data streams using ensemble classifiers. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–235. ACM Press, 2003.

[38] G. Widmer. Tracking context changes through meta-learning. *Machine Learning*, 27(3):259–286, 1997.

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