# One Pass Concept Change Detection for Data Streams

Sripirakas Sakthithasan[1], Russel Pears[1], and Yun Sing Koh[2]

[1] School of Computing and Mathematical Sciences, Auckland University of Technology,
{ssakthit,rpears}@aut.ac.nz
[2] Department of Computer Science, University of Auckland
ykoh@cs.auckland.ac.nz

**Abstract.** In this research we present a novel approach to the concept change detection problem. Change detection is a fundamental issue with data stream mining as models generated need to be updated when significant changes in the underlying data distribution occur. A number of change detection approaches have been proposed but they all suffer from limitations such as high computational complexity, poor sensitivity to gradual change, or the opposite problem of high false positive rate. Our approach, termed OnePassSampler, has low computational complexity as it avoids multiple scans on its memory buffer by sequentially processing data. Extensive experimentation on a wide variety of datasets reveals that OnePassSampler has a smaller false detection rate and smaller computational overheads while maintaining a competitive true detection rate to ADWIN2.

**Keywords:** Data Stream Mining, Concept Drift Detection, Bernstein Bound.

## 1 Introduction

Data stream mining has been the subject of extensive research over the last decade or so. The well known CVFDT [1] algorithm is a good example of an early algorithm that proposed an incremental approach to building and maintaining a decision tree in the face of changes or concept drift that occur in a data stream environment. Since then there has been a multitude of refinements to CVFDT (such as [2]) and to other methods [3] [4] that perform other types of mining such as a clustering and association rule mining.

The fundamental issue with data stream mining is to manage the sheer volume of data which grows continuously over time. A standard method of coping with this issue is to use a fixed size window of width $w$, where only the most recent $w$ instances are used to update the model built [5]. While this method is conceptually appealing, the major limitation is that concept change can occur at intervals that are quite distinct from the window boundaries. If rapid changes occur within a window, then these multiple changes will be undetected by the mining algorithm thus reducing the effectiveness of the model generated. Ideally a data stream algorithm should use long periods of stability to build a more detailed model whereas in time of rapid change the window needs to be shrunk at each change, the data representing the old concept be purged and the model updated with the new concept. Concept change detection with variable-sized adaptive windows has received very little attention compared to the well established area of algorithm development for data stream mining.

J. Pei et al. (Eds.): PAKDD 2013, Part II, LNAI 7819, pp. 461–472, 2013.

The methods proposed for concept change detection with adaptive windows all suffer from limitations with respect to one or more key performance factors such as high computational complexity, poor sensitivity to gradual change or drift, or the opposite problem of high false positive rate. In this research we propose a novel concept change detection method called OnePassSampler and compare it with the state-of-the-art concept change detector, ADWIN2 [6]. Our empirical results show that OnePassSampler has a lower false positive rate and significantly lower computational overheads than ADWIN2. OnePassSampler, as its name suggests makes only a single pass through its memory buffer and employs a simple and efficient array structure to maintain data about the current window. With ADWIN2 every new data block that arrives triggers a reassessment of candidate cut points previously visited, thus making it a multi-pass algorithm with respect to its internal memory buffer.

The major contributions made by this research are: a robust one pass algorithm for concept drift detection that has low memory and run time overheads while offering a rigorous guarantee on the false positive rate. The rest of the paper is as follows. Section 2 reviews the major research relating to concept drift detection. In Section 3 we describe our novel approach to drift detection with the formulation of a model, the derivation of a test statistic and the one pass algorithmic approach that is the key to low overheads. Section 4 presents a conceptual comparison between OnePassSampler and ADWIN2. Section 5 presents our empirical results and we conclude in Section 6 with a summary of the research achievements and some thoughts on further work in the area of concept change detection.

## 2 Related Work

The concept drift detection problem has a classic statistical interpretation: given a sample of data, does this sample represent a single homogeneous distribution or is there some point in the data (i.e the concept change point) at which the data distribution has undergone a significant shift from a statistical point of view? All concept change detection approaches in the literature formulate the problem from this viewpoint but the models and the algorithms used to solve this problem differ greatly in their detail.

Sebastiao and Gama [7] present a concise survey on change detection methods. They point out that methods used fall into four basic categories: Statistical Process Control (SPC), Adaptive Windowing, Fixed Cumulative Windowing Schemes and finally other classic statistical change detection methods. Early Drift Detection Method (EDDM) [8] works on the same basic principle as the authors earlier work but uses different statistics to detect change. More recently Bifet et al [6] proposed an adaptive windowing scheme called ADWIN that is based on the use of the Hoeffding bound to detect concept change. The ADWIN algorithm was shown to outperform the SPC approach and has the attractive property of providing rigorous guarantees on false positive and false negative rates. ADWIN maintains a window ($W$) of instances at a given time and compares the mean difference of any two sub windows ($W_0$ of older instances and $W_1$ of recent instances) from $W$. If the mean difference is statistically significant, then ADWIN removes all instances of $W_0$ considered to represent the old concept and only carries $W_1$ forward to the next test.

An improved version of ADWIN called ADWIN2[6] was also proposed by the same author which used a variation of exponential histograms and a memory parameter, to limit the number of hypothesis tests done on a given window. ADWIN2 was shown to be superior to Gama's method and fixed size window with flushing [9] on performance measures such as the false positive rate, false negative rate and sensitivity to slow gradual changes [6]. Despite the improvements made in ADWIN2, some issues remain namely, the fact that multiple passes on data are made in the current window and an improvement in the false positive rate for noisy data environments.

## 3 The One Pass Sampler Concept Change Detector

We start by defining in formal terms the problem that we address in this research. We then describe some generic principles that govern our change detector model. A test statistic is then derived that will be used in the change detector algorithms that we propose. We present a memory management strategy that supports incremental sampling with the use of a fixed size buffer in the form of a reservoir.

### 3.1 Change Detection Problem Definition

**Concept Change Detection.** Let $S_1 = (x_1, x_2, ..., x_m)$ and $S_2 = (x_{m+1}, ..., x_n)$ with $0 < m < n$ represent two samples of instances from a stream with population means $\mu_1$ and $\mu_2$ respectively. Then the change detection problem can be expressed as testing the null hypothesis $H_0$ that $\mu_1 = \mu_2$ that the two samples are drawn from the same distribution versus the alternate hypothesis $H_1$ that they arrive from different distributions with $\mu_1 \neq \mu_2$. In practice the underlying data distribution is unknown and a test statistic based on the sample means needs to be constructed by the change detector. If the null hypothesis is accepted incorrectly when a change has occurred then a false negative is said to have taken place. On the other hand if $H_1$ is accepted when no change has occurred in the data distribution then a false positive is said to have occurred. Since the population mean of the underlying distribution is unknown, sample means need to be used to perform the above hypothesis tests. The hypothesis tests can be restated as: Accept hypothesis $H_1$ whenever $Pr(|\hat{\mu_{S_1}} - \hat{\mu_{S_2}}|) \geq \epsilon) > \delta$, where $\delta$ lies in the interval $(0, 1)$ and is a parameter that controls the maximum allowable false positive rate, while $\epsilon$ is a function of $\delta$ and the test statistic used to model the difference between the sample means.

**Detection Delay.** Due to the use of sample data to infer changes in the population, detection delay is inevitable in any concept change detector and is thus an important performance measure. Detection Delay is the distance between $(m + 1)$ and $m^{'}$, where $m^{'}$ is the instance at which change is detected. In other words, detection delay equals: $\Big(m^{'} - (m + 1)\Big)$.

### 3.2 OnePassSampler Conceptual Change Detection Model

Our change detector is designed to widen its applicability to streams with different characteristics while yielding comparable performance, accuracy and robustness to methods

such as ADWIN2. OnePassSampler has the following properties, as illustrated in our experimentation: (1) is oblivious to the underlying data distribution, and (2) is inexpensive in terms of computational cost and memory.

**Core Algorithm Overview.** We first provide a basic sketch of our algorithm before discussing details of hypothesis testing. We use a simple example to illustrate the working of the algorithm. OnePassSampler accumulates data instances into blocks of size b. When attached to a classifier that uses OnePassSampler to detect change points, input data instances consists of a binary sequence of bits where binary 1 denotes a misclassification error and binary 0 denotes a correct classification decision. We use a block of data instances as the basic unit instead of instances as it would both be very inefficient and unnecessary from a statistical point of view to test for concept changes at the arrival of every instance.

Suppose that at time $t_1$ blocks $B_1$ and $B_2$ have arrived. OnePassSampler then checks whether a concept change has occurred at the $B_1|B_2$ boundary by testing $H_1$ above. If $H_1$ is rejected then blocks $B_1$ and $B_2$ are concatenated into one single block $B_{12}$ and $H_1$ is next tested on the $B_{12}|B_3$ boundary. In this check the sample mean of sub-window $B_{12}$ is computed by taking the average value of a random sample of size b from the sub-window of size $2b$. This sample mean is then compared with the sample mean computed from block $B_3$, also of size b. This process continues until $H_1$ is accepted, at which point a concept change is declared; instances in the left sub-window are removed and the instances in the right sub-window are transferred to the left. At all testing points equal sized samples are used to compare the sample means from the two sides of the window. The use of random sampling accelerates the process of the computation of the sample mean while maintaining robustness. The use of the averaging function as we shall see from our experimentation helps to smooth variation in the data and makes OnePassSampler more robust to noise than ADWIN2. In essence, OnePassSampler does a single forward scan through its memory buffer without the use of expensive backtracking as employed ADWIN2. While the use of random sampling ensures that sample means can be computed efficiently, a memory management strategy is required to ensure efficient use of memory as the left sub-window has the potential to grow indefinitely during periods of long stability in the stream.

**Use of Bernstein Bound.** Our approach relies on well established bounds for the difference between the true population and sample mean. A number of such bounds exist that do not assume a particular data distribution. Among them are the Hoeffding , Chebyshev , Chernoff and Bernstein inequalities [10]. The Hoeffding inequality has been widely used in the context of machine learning but has been found to be too conservative [6], over estimating the probability of large deviations for distributions of small variance. In contrast , the Bernstein inequality provides a tighter bound and is thus adopted in our work.

The Bernstein inequality states the following:

$$Pr\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - E[X]\right| > \epsilon\right) \leq 2\exp\Big(\frac{-n\epsilon^2}{2\hat{\sigma}^2 + \frac{2}{3}\epsilon(c-a)}\Big)$$

where $X_1, ..., X_n$ are independent random variables, $E[X]$ is the expected value or population mean, $X_i \in [a, c]$ and $\hat{\sigma}$ is the sample variance.

**Memory Management in OnePassSampler.** As OnePassSampler never re-examines previous candidate cut points it does not need to maintain a history of such cut-points and thus does not need to store memory synopses in the form of exponential histograms as ADWIN2 does. Instead, OnePassSampler only requires the means of its left and right sub-windows. In order to efficiently support the computation of sample averages a random sampling strategy is employed.

In addition to improving efficiency, random sampling is also necessary to satisfy the independence requirement for data used in the computation of the Bernstein bound. In a data stream environment independence between data instances in the same locality may not always be true as changes in the underlying data causes instances arriving after such a change to have very similar data characteristics, thus violating the independence property. One simple and effective method of addressing this dependence effect is to perform random sampling.

Our memory management strategy is based on the use of arrays to store blocks of data. An array enables fast access to specific data blocks that are sampled via the use of random sampling. The array is used to capture data in OnePassSampler's memory buffer. The memory buffer is divided into a left sub-window and a right sub-window, each of which uses an array for storage. When a new data block arrives, the block is temporarily inserted into the right sub-window and the sample means from the two sub-windows are compared to check for statistically significant differences. If no such difference exists, data in the right sub-window is copied into the left sub-window and is then removed from the right sub-window. Essentially this means that the left sub-window consists of a set of largely homogeneous blocks. In this context, it is more efficient from a memory point of view to slide the oldest $\frac{w}{b}$ block from the sub-window, where w is the width of the window and b is the data block size.

In certain circumstances the right sub-window may hold more than one data block. This happens when OnePassSampler enters a warning state after which newly arriving data blocks are added to the right sub-window instead of the left sub-window. A warning state is triggered when the mean of the data block in the right sub-window is not significantly different from the mean in the left sub-window on the basis of the drift confidence value $1 - \delta_{drift}$ but is significantly different with respect to a warning confidence value $1 - \delta_{warning}$. In cases when a warning state is entered a sliding window scheme is used for the right sub-window as well.

Given the OnePassSampler's worst case memory requirements are bounded above by $2w$ as two memory buffers are allocated of size w for each of the two sub-windows. We experimented with different values of $w$ and show that the quality of change detection (false positive rate, false negative rate and detection delay) is largely insensitive to the size of $w$, provided that $w$ exceeds the block size $b$.

### 3.3 Computation of Cut Point Threshold $\epsilon$

We now establish the value of the cut threshold against a null hypothesis that the data in the left and right sub-windows are drawn from the same population. Our null

hypothesis is expressed as: $H_0$ is $H_0 : \mu_l = \mu_r = \mu$ and the alternate hypothesis as $H_1 : \mu_l \neq \mu_r$. Let $S_l =$ a random sample $\{z_1, z_2, \ldots, z_l\}$ of size $l$ from $\{x_1, x_2, ..., x_m\}$ which comprise the m blocks in the left sub-window and let $S_r =$ a random sample $\{z_1, z_2, \ldots, z_r\}$ of size $r$ from $\{x_{m+1}, x_{m+2}, \ldots, x_n\}$ which comprises the (n-m) blocks in the right sub-window. With the application of the union bound on expression (1), we derive the following for every real number $k \in (0, 1)$:

$$Pr\left[|\hat{\mu_l} - \hat{\mu_r}| \geq \epsilon\right] \leq Pr\left[|\hat{\mu_l} - \mu| \geq k\epsilon\right] + Pr\left[|\mu - \hat{\mu_r}| \geq (1-k)\,\epsilon\right] \tag{1}$$

Applying the Bernstein inequality on the R.H.S of Equation 1, we get:

$$\begin{aligned} Pr\left[|\hat{\mu_l} - \hat{\mu_r}| \geq \epsilon\right] \quad &\leq 2\exp\Big(\tfrac{-b(k\epsilon)^2}{2\sigma_s^2 \frac{2}{3}k\epsilon(c-a)}\Big) \\ &+2\exp\Big(\tfrac{-b((1-k)\epsilon)^2}{2\sigma_s^2+\frac{2}{3}(1-k)\epsilon(c-a)}\Big) \end{aligned} \tag{2}$$

In the classification context, the bounds $a$ and $c$ for the Bernstein bound take values $a = 0, c = 1$. Substituting this in (2) we get:

$$Pr\left[|\hat{\mu_l} - \hat{\mu_r}| \geq \epsilon\right] \leq 2\exp\Big(\frac{-b(k\epsilon)^2}{2\sigma_s^2 + \frac{2}{3}k\epsilon}\Big) + 2\exp\Big(\frac{-b((1-k)\epsilon)^2}{2\sigma_s^2 + \frac{2}{3}(1-k)\epsilon}\Big) \tag{3}$$

The probability $Pr\left[|\hat{\mu_l} - \hat{\mu_r}| \geq \epsilon\right]$ represents the false positive rate $\delta$ and hence we have:

$$\delta = Pr\left[|\hat{\mu_l} - \hat{\mu_r}| \geq \epsilon\right] \leq 2\exp\Big(\frac{-b(k\epsilon)^2}{2\sigma_s^2 + \frac{2}{3}k\epsilon}\Big) + 2\exp\Big(\frac{-b((1-k)\epsilon)^2}{2\sigma_s^2 + \frac{2}{3}(1-k)\epsilon}\Big) \tag{4}$$

We now need to minimize the RHS of (4) in order to minimize the upper bound $\delta$ for the false positive rate. Given the two exponential terms, the RHS of (4) can be minimized when:

$$\frac{-b(k\epsilon)^2}{2\sigma_s^2 + \frac{2}{3}k\epsilon} = \frac{-b((1-k)\epsilon)^2}{2\sigma_s^2 + \frac{2}{3}(1-k)\epsilon} \tag{5}$$

The variable k above represents the proportion of instances among the left and right sub-windows. OnePassSampler uses equal sized samples across the sub-windows, giving $k = \frac{1}{2}$. We note that $k = \frac{1}{2}$ satisfies (5) above. Substituting $k = \frac{1}{2}$ in (4) gives:

$$\delta \leq 2\exp\Big(\frac{-b\frac{1}{4}\epsilon^2}{2\sigma_s^2 + \frac{2}{3}\cdot\frac{1}{2}\epsilon}\Big) + 2\exp\Big(\frac{-b\frac{1}{4}\epsilon^2}{2\sigma_s^2 + \frac{2}{3}\cdot\frac{1}{2}\epsilon}\Big) \tag{6}$$

Solving (6) to find $\epsilon$ gives:

$$\epsilon = \frac{2}{3b}\left\{p + \sqrt{p^2 + 18\sigma_s^2 bp}\right\} \tag{7}$$

where $p = ln\left(\frac{4}{\delta}\right)$. If $|\hat{\mu_l} - \hat{\mu_r}| \geq \epsilon$, concept change is declared at instance $(m+1)$ and $S_l$, $S_r$ can be considered to be from different distributions with probability $(1-\delta)$,

otherwise, hypothesis $H_0$ is accepted that there is no concept change in the window of instances $S_n$.

A change detection algorithm by its very nature needs to test multiple cut points before an actual change point is detected. Each test involves a hypothesis test applied at a certain confidence level. The effect of multiple tests is to reduce the confidence from $\delta$ to $\delta^{'}$ which represents the effective (overall) confidence after $n$ successive hypothesis tests have been carried. However, we note that the hypothesis tests in the change detection scenario are not independent of each other as the probability of a false positive (i.e incorrectly accepting hypothesis H1 that the means across the left and and right sub-windows are different) at a particular test has an influence on whether a false positive occurs at subsequent tests and hence methods such as Bonferroni do not apply. We use our own error correction factor, $\delta^{'} = 2\frac{\delta}{(1-\frac{1}{2}^{n})}$. The derivation of $\delta^{'}$ is omitted due to space constraints. Thus, in our model the change and warning significance levels, $\delta_{Change}$ and $\delta_{Warning}$ are set to $\delta^{'}_{Change}$ and $\delta^{'}_{Warning}$ respectively to control the false positive probability. We observe that the the correction factor above converges to $2\delta$ for large values of $n$.

### 3.4 OnePassSampler Change Detection Algorithms

This section presents the core algorithms used in our change detector system. $S_r$ and $S_l$ denote the right and left sub windows. Algorithm (1) decides the change type given the mean values $\hat{\mu_r}$ and $\hat{\mu_l}$ of $S_r$ and $S_l$ respectively, $\epsilon_{change}$ (the threshold mean difference for $\delta_{change}$) and $\epsilon_{warning}$ (the warning threshold mean difference for $\delta_{warning}$). $\epsilon_{change}$ and $\epsilon_{warning}$ are calculated using the equation (7). Though OnePassSampler detects drifts in any variation in the mean, algorithm (1) only reports the change when mean increases ($\hat{\mu_r} > \hat{\mu_l}$). In the event of a concept change Algorithm (2) transfers the contents of the right sub-window into the left. When a warning state is triggered it increases the sample size, in expectation of a subsequent concept change. This increase has the effect of increasing precision in sampling and the algorithm may become more sensitive to slow gradual change.

```
Input: μ̂_l, μ̂_r, ε_Change, ε_Warning
Output: Change || Warning || Internal
if ε_Warning ≤ |μ̂_l − μ̂_r| then
    if ε_Change ≤ |μ̂_l − μ̂_r| then
        if μ̂_r > μ̂_l then
            return Change;
        end
        return Internal;
    end
    return Warning;
end
return None;
```

**Algorithm 1.** GetDriftType()

```
Input: An instance(Ins), BlockSize, S_l, S_r
Output: True/False
Increment the instance counter;
S_l = S_r ∪ {Ins};
if At the block boundary then
    ChangeType = GetDriftType();
    if (DriftType is Change or Internal) then
        Remove all elements from S_l;
        Copy all elements of S_r to S_l;
        Remove all elements from S_r;
        Set SampleSize to BlockSize;
        if (DriftType is Change) then
            return True;
        end
        return False;
    end
    else if (DriftType is Warning) then
        Double the sample size;
        return False;
    end
    Copy all elements of S_r to S_l;
    SampleSize = BlockSize;
    return False;
end
```

**Algorithm 2.** IsDrift()

## 4 OnePassSampler versus ADWIN2: Similarities and Differences

Two major design differences exist between the two change detectors. The first lies in the policy used in determining cuts. When new data arrives, ADWIN2 creates a new bucket and adds it to its memory buffer. It then searches through all buckets currently stored in its memory buffer for a possible cut point. A cut point in ADWIN2 lies on the boundary between buckets. With $N$ buckets currently in storage, ADWIN2 will examine a total of $(N-1)$ possible cut points. Furthermore, as each new bucket arrives previous bucket boundaries that were examined before will be re-examined for possible cuts. Effectively, ADWIN2 makes multiple passes through its memory buffer. In contrast, OnePassSampler never re-examines previous block (equivalent of ADWIN2's bucket) boundaries for cuts and only examines the boundary between the newly arrived block and the collection of blocks that arrived previously for a possible cut. In this sense, OnePassSampler can be said to do a single pass through its memory buffer when searching for cuts, and hence its name.

The second major difference lies in the estimation strategy for assessing means of data segments. ADWIN2 relies on exponential histograms for estimating mean values, whereas OnePassSampler uses random sampling base on an efficient array structure to estimate means. The problem with exponential histograms is that some of the buckets, typically the more recent ones may be too small in size to yield accurate estimations for mean values. This is due to the fact that in ADWIN2 a bucket is created whenever a 1 appears in the stream, and when data has high variation bucket size will vary widely. For buckets that are too small in size to support accurate estimation, ADWIN2 will end up overestimating the true mean and false positives may then result.

## 5 Empirical Study

Our empirical study had two broad objectives. Firstly, we conducted a comparative study of OnePassSampler with ADWIN2 on key performance criteria such as the true positive rate, the false positive rate, the time delay in detecting changes and the execution time overheads involved in change detection. We used Bernoulli distribution in all experiments to simulate classifier outputs though OnePassSampler is a general drift detector for any distribution.

In the second part of our experimentation we conducted a sensitivity analysis of the effects of block size, warning level and sliding window size on the delay detection time for OnePassSampler.

### 5.1 Comparative Performance Study

One first experiment was designed to test OnePassSampler's false positive rate vis-a-vis ADWIN2. We used a stationary Bernoulli distribution for this and tested the effect of various combinations of mean values ($\mu$) and confidence values ($\delta$) as shown in Table 1. For this experiment the block size for OnePassSampler was set to its default value of 100 and ADWIN2's internal parameter M was also set to its default value. We conducted a total of 100 trials for each combination of $\mu$ and $\delta$ and the average false positive rate for each combination was recorded.

**Table 1.** False Positive Rate for stationary Bernoulli distribution

| | One Pass Sampler | | | ADWIN2 | | |
|---|---|---|---|---|---|---|
| $\mu$ | $\delta$ =0.05 | $\delta$ =0.1 | $\delta$ =0.3 | $\delta$ =0.05 | $\delta$ =0.1 | $\delta$ =0.3 |
| 0.01 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 0.1 | 0.0000 | 0.0000 | 0.0000 | 0.0001 | 0.0002 | 0.0018 |
| 0.3 | 0.0000 | 0.0000 | 0.0001 | 0.0008 | 0.0017 | 0.0100 |
| 0.5 | 0.0000 | 0.0000 | 0.0001 | 0.0012 | 0.0030 | 0.0128 |

Table 1 shows that both OnePassSampler and ADWIN2 have good false positive rates that are substantially lower than the confidence level set. However, we observe that as the variance in the data increases with the increase in the $\mu$ value (for a Bernoulli distribution, the variance is $\mu \times (1-\mu)$) that ADWIN2 starts to register false positives. The ADWIN2 false positive rate increases progressively with the increase in the variance as well as the lowering of confidence (ie higher $\delta$ values). On the other hand OnePassSampler retains a virtually zero false positive rate except when the confidence is low at 0.3 when it registers a rate of $0.01\%$, compared to the ADWIN2 rate of $1.28\%$ at $\mu = 0.5$ and $\delta = 0.3$. As the confidence becomes lower the $\epsilon$ value decreases and this results in an increase in the false positive rate for ADWIN2. However, OnePassSampler is virtually insensitive to the decrease in $\epsilon$ due to the fact that the mean value can be estimated more accurately through the combined use of random sampling and the use of the aggregated running average mechanism.

The second experiment was designed to test the true positive (detection) rates of OnePassSampler and ADWIN2 over data that was also generated from a Bernoulli distribution. We generated four different data streams of length $L = 10,000$, 50,000, 100,000 and 1,000,000 bits from a Bernoulli distribution. The data generated was stationary with mean $0.01$ in the first $L - 2300$ time steps and we then varied the distribution in a linear fashion with different gradients in the last 2300 time steps. A total of 100 trials were conducted for each combination of data length and slope values. We tracked key performance indicators such as the true detection rate, average execution time and the detection delay time. Both OnePassSampler and ADWIN2 managed to achieve a true detection rate of $100\%$ for all combinations of data length and change gradients.

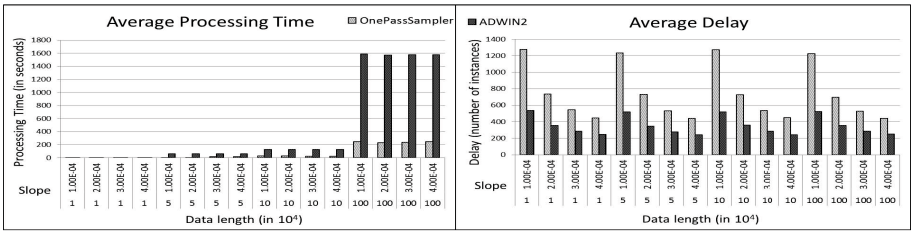


**Fig. 1.** Comparative Change Detection Performance of OnePassSampler and ADWIN2

Figure 1 also illustrates that ADWIN2 was much slower in stream processing than OnePassSampler. Furthermore, the gap between the two processing times becomes wider as the length of the stable segment of the stream becomes longer. This was expected as ADWIN2 spends much time doing repeated scans through the histogram and

examines every possible combination of cuts defined by the buckets. OnePassSampler, on the other hand does a single pass through the window segment and at each block of data it assesses whether the newly arrived block is sufficiently different from the previous blocks in its memory buffer.

However, it is clear from Figure 1 that ADWIN2 has better mean detection delay when compared to OnePassSampler. OnePassSampler needs a relatively larger window segment before it can decide whether a newly arrived block is sufficiently different due to the sampling strategy that it uses. As expected, the delay times reduced with increasing gradient of change, although we observe that OnePassSampler reduces at a faster rate than ADWIN2 with the gap between the two closing for higher gradients of change. Section 5.2 shows that OnePassSampler's detection delay can be reduced with proper use of warning level and particularly block size on which it is most sensitive with respect to delay.

The final part of our experimentation involved an investigation of the sensitivity of OnePassSampler's key parameters on detection delay time. From previous experimentation with Bernoulli data it was observed that OnePassSampler had a higher detection delay time than ADWIN2 and thus the motivation was to determine parameter settings that minimize OnePassSampler's detection delay time.

### 5.2 Sensitivity Analysis on OnePassSampler

In the first experiment we investigated the effect of block size on Bernoulli data streams with different gradients. Section 5.1. Figure 2 shows that as block size increases, delay time initially decreases, reaches a minimum value and then starts to rise once again. In order to detect changes in data distribution a sample of sufficient size is required, which in turn is determined by the block size. If the size of the block (sample size) is too low,

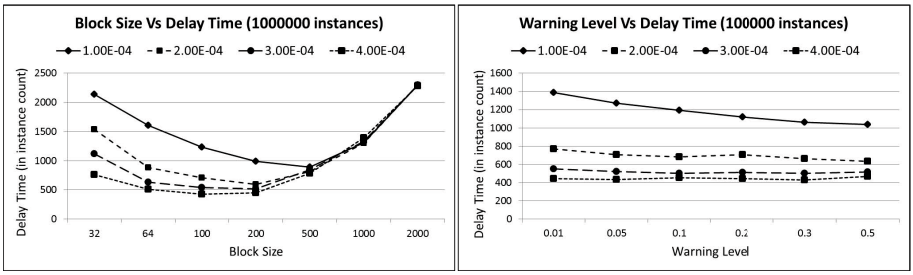


**Fig. 2.** Effects of Block Size and Warning Level on Detection Delay Time for OnePassSampler

then in common with other statistical tests of significance, a statistical difference cannot be determined until a greater change occurs with time, thus delaying the detection. On the other hand, if the block size is too large then the probability increases that a change occurs too late within a given block for the change to be detected and so the change will go undetected until at least a new block arrives, thus giving rise to an increased detection delay. A block size of 200 appears to be optimal across a range of different change gradients, except when the change is very gradual , in which case 500 gives a slightly lower delay.

We next checked the effect of warning level on delay. Figure 2 shows that warning level has a much smaller effect on delay than block size. With a slope of $1.00E-04$ the warning level setting has a negligible effect on delay and thus a pragmatic setting that is twice the significance level should suffice in most cases to reduce the delay.

Next, we assessed the effect of sample size increment. Whenever the warning level is triggered the sample size is incremented in the hope of trapping an impending change earlier. We investigated a range of increments and as Figure 3 shows, a doubling of sample size produces optimal results across the entire spectrum. As with the warning level, too large an increase results in an increase in the detection delay.

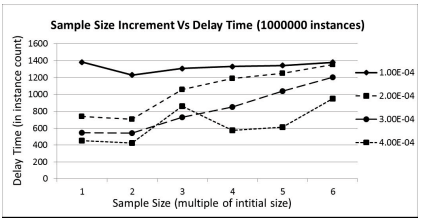


**Fig. 3.** Effects of Sample Size Increment on Detection Delay Time for OnePassSampler

Overall, it appears that block size is of prime importance in minimizing delay time; a block size of 200 works well for a range of datasets with different change dynamics. The other two parameters have a much smaller effect in general but can also contribute to smaller delay times with settings that we discussed above, especially in the case of slowly varying data.

Finally, we assessed the effects of the sliding window size on true positive rate, false positive rate and delay time. We varied the sliding window size in the range 500 to 10,000. For each window size, 30 trials were conducted on data from a Bernoulli distribution and the average for each of the performance measures were recorded. Due to space constraints we show the detection delay for the smallest change gradient of $1.00E-4$; the results for the other change gradients followed very similar trends. As Table 2 shows, the detection delay is largely insensitive to window size. In addition, all window sizes recorded a true positive rate of $100\%$. The false positive rate was in line with the other two measures, virtually no change in rate was observed across the entire range of window sizes used. Once again space constrains prevent us from showing the entire set of results; we only show the case with mean value $0.3$ and delta $0.3$. All other combinations of mean and delta returned virtually identical results. These results indicate that window size when set at a reasonable multiple of block size has no significant effect on key factors such as the true positive rate and delay time. These results are to be expected as data that is slid out of the window consists of a set of homogeneous instances from OnePassSampler's left sub-window.

**Table 2.** Detection delay for varying window sizes

| Sliding Window Size | 500 | 1000 | 2000 | 4000 | 6000 | 8000 | 10000 |
|---|---|---|---|---|---|---|---|
| True Positive Rate | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Delay Time | 1300 | 1330 | 1350 | 1320 | 1350 | 1370 | 1330 |
| False Positive Rate | 0.00000 | 0.00001 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |

## 6 Conclusions and Future Work

This research has shown that a concept change detector based on a sequential hypothesis testing strategy based on use of the Bernstein bound as a test statistic yields excellent performance in terms of false positive rate, true positive rate and processing time. Our comparative study with ADWIN2 clearly shows that a single pass strategy can produce competitive false positive and true positive rates to ADWIN2, with much lower computational overheads.

The use of sequential hypothesis testing combined with an efficient incremental strategy that updates statistics on the memory buffer were the two major factors behind the greatly reduced computational overheads over ADWIN2. Despite lower computational overheads, OnePassSampler has a higher detection delay time in certain cases and our future work will focus on improving this aspect. By means of a mechanism that monitors change in the running average of data arriving in the window an alternate candidate cut point can be defined at a point further downstream than the current block boundary. The system would then check both the current block boundary as well as the alternate point. This modification would result in trading off computational overhead with an improvement in detection delay for datasets with small gradients of change.

## References

1. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proc. of the 2001 ACM SIGKDD, pp. 97–106 (2001)
2. Hoeglinger, S., Pears, R., Koh, Y.: Cbdt: A concept based approach to data stream mining. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 1006–1012. Springer, Heidelberg (2009)
3. Koh, Y.S., Pears, R., Yeap, W.: Valency based weighted association rule mining. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010, Part I. LNCS, vol. 6118, pp. 274–285. Springer, Heidelberg (2010)
4. Widiputra, H., Pears, R., Serguieva, A., Kasabov, N.: Dynamic interaction networks in modelling and predicting the behaviour of multiple interactive stock markets. Int. J. Intell. Syst. Account. Financ. Manage. 16, 189–205 (2009)
5. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the 9th ACM SIGKDD, KDD 2003, pp. 226–235 (2003)
6. Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing. In: SDM. SIAM (2007)
7. Sebastiao, R., Gama, J.: A study on change detection methods. In: 4th Portuguese Conf. on Artificial Intelligence (2009)
8. Jose, M.B., Campo-Ávila, J.D., Fidalgo, R., Bifet, A., Gavaldà, R., Morales-bueno, R.: Early Drift Detection Method. In: Proc. of the 4th ECML PKDD Int. Workshop on Knowledge Discovery from Data Streams, pp. 77–86 (2006)
9. Kifer, D., Ben-David, S., Gehrke, J.: Detecting change in data streams. In: Proceedings of the Thirtieth International Conference on VLDB, vol. 30, pp. 180–191. VLDB Endowment (2004)
10. Peel, T., Anthoine, S., Ralaivola, L.: Empirical bernstein inequalities for U-statistics. In: NIPS, pp. 1903–1911 (2010)