Statistics and Computing (2019) 29:1109–1125
https://doi.org/10.1007/s11222-019-09858-0

Multiple changepoint detection in categorical data streams

Joshua Plasse[1] · **Niall M. Adams**[2]

Received: 17 May 2018 / Accepted: 2 February 2019 / Published online: 15 February 2019

Abstract

The need for efficient tools is pressing in the era of big data, particularly in streaming data applications. As data streams are ubiquitous, the ability to accurately detect multiple changepoints, without affecting the continuous flow of data, is an important issue. Change detection for *categorical* data streams is understudied, and existing work commonly introduces fixed control parameters while providing little insight into how they may be chosen. This is ill-suited to the streaming paradigm, motivating the need for an approach that introduces few parameters which may be set without requiring any prior knowledge of the stream. This paper introduces such a method, which can accurately detect changepoints in categorical data streams with fixed storage and computational requirements. The detector relies on the ability to adaptively monitor the category probabilities of a multinomial distribution, where temporal adaptivity is introduced using forgetting factors. A novel adaptive threshold is also developed which can be computed given a desired false positive rate. This method is then compared to sequential and nonsequential change detectors in a large simulation study which verifies the usefulness of our approach. A real data set consisting of nearly 40 million events from a computer network is also investigated.

Keywords Adaptive threshold · Categorical data stream · Changepoint detection · Forgetting factor

1 Introduction

The rapid development of data acquisition technology is allowing data to be collected at an unprecedented rate. This large-scale collection of data often results in *data streams* (Aggarwal 2007; Gama 2010)—unbounded sequences of observations which arrive at high-frequency and are subject to unknown temporal variation. Data streams appear in various real-world applications, such as: industrial manufacturing (Montgomery 2007), genome sequencing (Weiß 2012), social networks and computer network traffic (Gama 2010). This creates a demand for efficient, dynamic, flexible methodology, which can perform inference while respecting the computational demands of the stream. The particular focus of this paper is on developing methodology for *categorical* data streams.

The frequency of observations arriving is typically very high. For example, Imperial College London's computer network generates roughly 1.3 billion network events per day (Heard et al. 2014), while the Large Hadron Collider at CERN generates millions of events *per second* (Ross et al. 2011). Data stream mining is therefore fundamentally different from conventional tasks which assume a complete data set is not only available, but attainable. Since data arrives continuously, this requires methodology to use limited computational and storage requirements (Pavlidis et al. 2011). Ideally, methods will be *single pass*, that is, once data arrives it is utilized and immediately discarded. The computational overhead should also be constant with respect to the length of the stream, and not increasing over time, which is a common attribute in many statistical and machine learning algorithms.

The dynamic nature of the world makes it naive to assume that data streams remain stationary. Nonstationarity in streams is typically referred to as *concept drift* (Tsymbal 2004; Widmer and Kubat 1996), and if ignored could be detrimental to the learning process. This temporal aspect gives rise to two challenges of streaming inference: the ability to *adapt* to changes in the data-generating process, as well as the capability to *detect* multiple changepoints that occur over time.

One widely used and arguably easiest way of introducing a temporal component into the estimation process is to use a

✉ Joshua Plasse
j.plasse15@imperial.ac.uk

[1] Department of Mathematics, Imperial College London, London, UK

[2] Department of Mathematics and Data Science Institute, Imperial College London, London, UK

sliding window. Choosing the width of this window is a nontrivial task, and with few exceptions (e.g., Bifet and Gavalda (2007)), it is usually taken to be constant. The choice of an appropriate window size is commonly overlooked, or is said to be chosen based on advice from a 'domain expert.' This is not suitable for the streaming paradigm as fixed values, either subjectively chosen or prescribed by an expert, will not accurately describe a data stream indefinitely. Of course, an expert could reassess their opinion after some prespecified amount of time; however, when to adjust the window remains unclear.

An alternate method for producing adaptive estimates is to use *forgetting factors*. Forgetting factors are commonly used in adaptive filtering (Haykin 2008) and have been successfully used in various statistical stream mining settings (Anagnostopoulos et al. 2012; Bodenham and Adams 2017; Pavlidis et al. 2011). Forgetting factors are a sequence of scalars, which continuously downweights historic data as new data are observed without introducing any computational burden. Forgetting factors are discussed more in Sect. 3.

The literature on sequential changepoint detection is vast, e.g., see Tartakovsky et al. (2014). However, sequential methods typically have various idiosyncrasies which make them ill-suited to the streaming paradigm. As discussed in Sect. 1.1, some of these include: (1) attention is restricted to continuous, univariate data streams that typically follow some assumed distribution (e.g., Gaussian), (2) several control parameters, such as the width of a sliding window, are introduced and are usually chosen based on information that is typically unknown (such as the distributions before and after a changepoint being completely specified), (3) the detectors liberally use fixed *thresholds* to signal when a change occurs without providing insight into how to choose their values in practice, and (4) methods focus on the detection of a single changepoint and re-initializing the detector without interrupting the flow of data is not obvious.

In this paper, attention is restricted to categorical data streams where the interarrival times between observations are on the order of milliseconds—an under-researched topic in the literature. An adaptive estimate, based on a forgetting factor framework, is developed for the category probabilities of the multinomial distribution governing the stream. These estimates can be sequentially updated with no computational/storage overhead and can be tuned online without any user supervision. Comparing the adaptive and static estimates results in a novel multinomial change detection method (MCDM) which can effectively monitor for multiple changepoints in data streams. Furthermore, adaptively maintaining estimates will make the MCDM robust to missed detections, as the estimate will quickly adjust to concept drift. This is crucial in monitoring the category probabilities, as the magnitude of the changes can be at most one, resulting in potentially small, difficult to detect drift. The detector utilizes the Kullback–Leibler divergence (Kullback and Leibler 1951) to flag changes, and does so based on an original *adaptive threshold*. This adaptive threshold introduces a single control parameter which can be specified given a desired false positive rate. This appears to be the first work which develops a streaming change detector for categorical data streams without falling into any of the previously discussed pitfalls.

This paper is structured as follows: Sect. 1.1 discusses literature directly relevant to this work. Section 2 considers categorical data streams and the multiple changepoint scenario while providing notation used throughout the paper. An adaptive estimate for the multinomial distribution, which can be recursively updated with no computational overhead, is discussed in Sect. 3. The MCDM is developed in Sect. 4, and a time-varying threshold to aid in the detection of changepoints is introduced. Section 5 describes the methods which the MCDM is compared to in the simulation study of Sect. 6. Section 7 implements the MCDM on nearly 40 million observations collected from Imperial College London's computer network.

1.1 Relevant work

Sequential change detection in categorical data streams has received little attention in recent decades. Although having merit in their own right, many existing methods fall victim to the previously mentioned pitfalls. The work of Wolfe and Chen (1990) appears to be among the first to develop changepoint methodology for multinomial sequences. However, this work focuses on the detection of a single changepoint and develops estimates which need to be recomputed for various potential changepoint locations. This violates the single pass requirement and is therefore not practical for our purposes. Similar issues occur in Hou et al. (2013), where maximum likelihood estimation and the commonly used p-chart (Montgomery 2007) are combined to detect a single change among categorical observations.

The methodology developed in Ienco et al. (2014) depends on a 'context-based' distance model; however, the choice of a good context is nontrivial. Moreover, Chebyshev's inequality is used to detect changepoints which introduces several thresholds whose values must be specified. A fuzzy multinomial control chart for linguistic variables is presented in Amirzadeh et al. (2008), the idea being that the vagueness in assigning categories (such as poor, good and excellent) to items being inspected can be made formal with techniques from fuzzy set theory. Although assigning weights to the linguistic variables is difficult, authors suggest they should be chosen after careful discussion with experts.

Cao and Huang (2013) introduce an algorithm based on rough set theory. This method uses a fixed sliding window and threshold to detect changes, and it is suggested that their values should be set based on available prior knowledge. Chen et al. (2009) develop a framework for clustering nonstationary categorical data. A fixed sliding window is used to handle the temporal variation, and several thresholds are introduced which must be specified. Detectors based on moving sum statistics (Eiauer and Hackl 1978) have been investigated recently in Eichinger et al. (2018) and, similar to other methods, introduce a fixed 'bandwidth' parameter which is similar to the width of a sliding window. The performance of moving sum detectors depends crucially on the choice of bandwidth (Eichinger et al. 2018) and is difficult to specify in practice. The work of Weiß (2012) also considers fixed sliding windows and uses the Gini index (among other measures) to monitor categorical data streams.

The cumulative sum (CUSUM) control chart introduced in Page (1954) is a popular technique used in statistical quality control. Since various modifications have been made, CUSUM is applicable to both Bernoulli and multinomial sequences. In Ryan et al. (2011), a multinomial CUSUM chart is developed. The authors mention that if the direction of the drift in *each* category cannot be specified in advance, then a Bernoulli CUSUM chart should be implemented separately for each category—guidance we will utilize later in experimental comparisons. Höhle (2010) develops a multinomial CUSUM chart by modeling the category probabilities via a multinomial logistic regression model. It is assumed that the category probabilities after a changepoint manifest as specific shifts in the intercept of their model, which introduces several scaling factors. Both multinomial CUSUM charts discussed require prior knowledge of the shift in the category probabilities, which will be difficult to anticipate in the multiple changepoint scenario. Therefore, following the advice of Ryan et al. (2011), multiple Bernoulli CUSUM charts will be implemented as opposed to a multinomial CUSUM chart. An overview of the Bernoulli CUSUM chart is provided in Sect. 5.1.

Bayesian-based techniques are mostly offline, although there has been some recent research devoted to extending Bayesian methodology to the online setting. In Adams and MacKay (2007), the distribution for the length of the current run, i.e., the amount of time elapsed since the last detected change, is computed. This method has linear storage and time complexity, although thresholding the tails of the run-length distribution allows for, on average, constant complexity per time-step. However, the worst-case complexity is still linear in the length of the data, which may be problematic for high-frequency streams. Byrd et al. (2017) present a lagged version of the work in Adams and MacKay (2007) and show via simulation that the method of Adams and MacKay (2007) has difficulties in detecting changes of small magnitude (which is a primary focus of our work). In the lagged version, changes are flagged according to thresholding a relative change in the maximum a posteriori estimate of the run-length distribution. In terms of the length of the stream, the introduction of the lag comes at a price in terms of computational complexity, as it becomes exponential in terms of the lag. As online Bayesian techniques are not directly relevant to the focus of this work, they are not considered in the sequel.

2 Nonstationary categorical streams

This section introduces notation used throughout the paper and provides a brief discussion of the multinomial distribution. The section concludes by discussing the multiple changepoint scenario and provides a simple way of restarting a detector once a detection has been made.

2.1 Categorical data streams

It is assumed that independent realizations from a K-category multinomial distribution arrive sequentially. It is further assumed that the number of categories K is known, fixed, and $K > 1$. The univariate data stream can then be expressed as

$$\langle d_1, d_2, \ldots, d_{t-1}, d_t, \ldots \rangle, \tag{1}$$

where $d_t \in \mathcal{S} \equiv \{1, 2, \ldots, K\}$. An equivalent representation of (1) will prove useful in subsequent sections where other methods are adapted to detect changes in categorical data streams. This representation is referred to as the *binarization* of the categorical process (Weiß 2012) and can be represented as

$$\langle \mathbf{e}_{d_1}, \mathbf{e}_{d_2}, \ldots, \mathbf{e}_{d_{t-1}}, \mathbf{e}_{d_t}, \ldots \rangle, \tag{2}$$

where $\mathbf{e}_{d_t} \in \mathbb{R}^K$ represents the standard basis vector in $\mathbb{R}^K$ comprised of all zeros except for a one in the index corresponding to the value of d_t. For example, if $K = 3$ and $d_t = 2$, then $\mathbf{e}_{d_t} = (0, 1, 0)^\top$. This exposition is concerned with sequentially monitoring a multinomial distribution for multiple changepoints in the streaming paradigm, where a novel method is developed for data streams assuming the form of (1).

Suppose t observations, denoted $d_{1:t} \equiv (d_1, \ldots, d_t)$, have been observed from a data stream. Then, the multinomial distribution governing the process can be expressed as

$$f(\boldsymbol{c}_t \mid \boldsymbol{p}_t) = \frac{t!}{\prod_{i\in\mathcal{S}} \left(c_t^{(i)}\right)!} \prod_{i\in\mathcal{S}} \left(p_t^{(i)}\right)^{c_t^{(i)}},$$

where $\boldsymbol{c}_t \in \mathbb{R}^K$ is the vector of category counts and $\boldsymbol{p}_t \in \mathbb{R}^K$ are the corresponding category probabilities. The ith component of each vector is denoted, respectively, by $c_t^{(i)}$ and $p_t^{(i)}$.

For each $t \in \mathbb{Z}^+$, the following are satisfied

$$\sum_{i \in \mathcal{S}} c_t^{(i)} = t, \quad \sum_{i \in \mathcal{S}} p_t^{(i)} = 1.$$

Given $d_{1:t}$, the maximum likelihood estimate for the category probabilities is given by $\hat{\boldsymbol{p}}_t = \hat{\boldsymbol{c}}_t / t$, $\hat{\boldsymbol{c}}_t$ being the vector of observed category counts. The vector $\hat{\boldsymbol{p}}_t$ will be referred to as the *static maximum likelihood estimate* as it gives each datum equal weight in the computation of the estimate. This estimate plays an important role in the development of the change detection method discussed in Sect. 4. However, allowing every observation to have equal weight in the estimation process will be detrimental as more data are collected from a drifting stream. To alleviate this issue, a way of adaptively estimating the probabilities is provided in Sect. 3.

2.2 Multiple changepoints

Any changes in a categorical data stream will manifest in the category probabilities. In this work, it is assumed that these probabilities are *locally stationary*. That is, there exists a set of changepoints $\{\tau_k\}_{k=1}^m$ that partition the stream into $(m + 1)$ segments. In each segment, the multinomial distribution governing the data stream is assumed fixed. The distribution of the stream, in the kth segment, can then be expressed as

$$\boldsymbol{C}_t \sim \text{Multi}(\boldsymbol{p}_k) \quad \forall t \in [\tau_{k-1}, \tau_k),$$

where $\text{Multi}(\boldsymbol{p}_k)$ represents the multinomial distribution with probability vector $\boldsymbol{p}_k$. Generally, $\boldsymbol{p}_t$ will be used to represent the probability vector currently generating the observations from the stream. Notice that $\boldsymbol{p}_t$ can only assume one of $(m + 1)$ vectors, and its value depends on which segment is currently generating the data. Additionally, let $\tau_0 \equiv 0$ and $\tau_{m+1} \equiv \infty$.

In various research areas, such as statistical quality control (Montgomery 2007), many methods have been developed to detect a single changepoint in a data stream. Usually this corresponds to finding a fault in some manufacturing process. Once a change is discovered, the process can be stopped until a remedy is found. However, in our context, the stream continues unabated when a changepoint is detected. Therefore, a *restarting mechanism* must be built into the method so that the continuous flow of data is unaffected by the detection of changepoints.

A simple restarting mechanism discussed in Bodenham and Adams (2017) is a *grace period*. Let $G \in \mathbb{Z}^+$ dictate the length of the grace period, and suppose that a change detector flagged a change at time $t = \hat{\tau}$. The grace period immediately begins, and during the interval $t \in [\hat{\tau}, \hat{\tau} + G]$, no subsequent changepoints are monitored for. The purpose of this period is to reestimate the new segment's probability vector prior to another round of monitoring. Although G may be considered as a control parameter, any methodology that can effectively handle multiple changepoints will require some sort of restarting procedure. Additionally, G may be chosen as a function of the minimum allowed false positive rate for a detector, as it implicitly provides a lower bound on the number of observations that are processed until a false positive is flagged. The grace period is depicted graphically in Fig. 1.

The objective of this work is to accurately estimate the location of the changepoints without any user intervention, as well as continuously maintaining an accurate estimate of the probability vector $\boldsymbol{p}_t$. Due to the nonstationarity of data streams, this will require a technique which adaptively learns to discard older data as more observations arrive from the stream. An effective way of providing temporal adaptivity is via forgetting factors—the topic of the next section.

3 Adaptive estimation

Adopting a technique commonly used in adaptive filtering (Haykin 2008); a way of introducing temporal adaptivity is to use forgetting factors. Forgetting factors are a sequence of scalars, taking values in [0, 1], that downweight historic data as newer data arrive. Forgetting factors have been previously considered (Anagnostopoulos et al. 2012; Bodenham and Adams 2017; Pavlidis et al. 2011) and can be interpreted as a continuous analog of the sliding window approach, as discussed in Sect. 1.

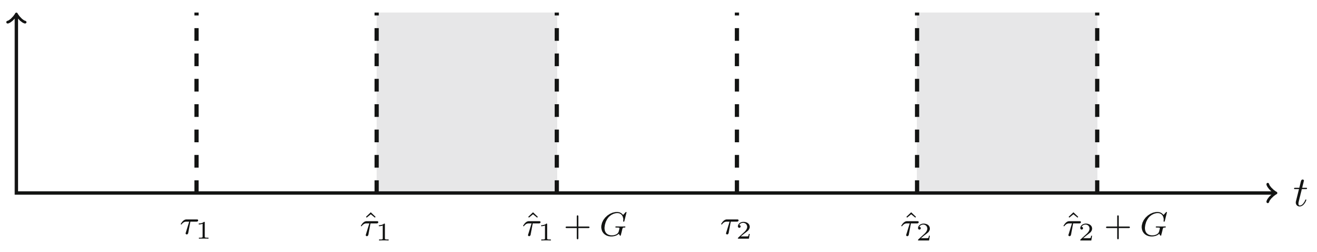

Fig. 1 An example of a stream with two changepoints at times τ_1 and τ_2, and detections made by a detector at times $\hat{\tau}_1$ and $\hat{\tau}_2$. The shaded regions indicate grace periods where the detector is not monitoring for any changepoints

In Sect. 3.1, forgetting factors are incorporated in the estimation using a frequentist approach. This leads to a modified likelihood function which is more suitable for temporally evolving data. This likelihood is then optimized, resulting in adaptive estimates for the category probabilities. Section 3.2 presents a sequential formulation of these adaptive estimates, allowing for the adaptive estimation of $\boldsymbol{p}_t$ that introduces no computational overhead nor the need to store any historical data. Section 3.3 provides a method for tuning the forgetting factors online without user supervision.

3.1 A temporally aware likelihood

Similar to Anagnostopoulos et al. (2012), consider the weighted log-likelihood function given by

$$\begin{aligned}\mathcal{L}_{\mathrm{FF}}(\boldsymbol{p} \mid d_{1:t}) &= \sum_{k=1}^{t} \left[\prod_{\ell=k}^{t-1} \lambda_\ell \right] \mathcal{L}(\boldsymbol{p} \mid d_k) &(3)\\ &= \lambda_{t-1}\mathcal{L}_{\mathrm{FF}}(\boldsymbol{p} \mid d_{1:t-1}) + \mathcal{L}(\boldsymbol{p} \mid d_t), &(4)\end{aligned}$$

where $\mathcal{L}(\cdot \mid \cdot)$ is the multinomial log-likelihood function and $\{\lambda_k\}_{k=1}^{t-1}$ are a collection of forgetting factors. When $k = t$, the empty product is assumed one. In the above formulation, the time subscript is omitted from the vector $\boldsymbol{p}_t$; this is to reinforce that the temporal adaptivity of the soon to be displayed estimates is strictly due to the forgetting factors.

The interpretation of the forgetting factors is best seen by inspecting the sequential formulation of the likelihood given by Eq. (4). The likelihood associated with the most recent observation is included in the likelihood with 'full-weight,' or weight one. Assuming that $\lambda_{t-1} < 1$, we see that the historical data $d_{1:t-1}$ are smoothly downweighted by the collection of forgetting factors. Therefore, as desired, older data are gradually 'forgotten' as newer data arrive. See Bodenham (2014) for a thorough discussion on using forgetting factors in adaptive estimation.

For every $i \in \mathcal{S}$, optimization of Eq. (3) results in the *temporally adaptive maximum likelihood estimates*

$$\tilde{p}_t^{(i)} = \frac{1}{n_t} \sum_{k=1}^{t} w_k \; I(d_k = i), \tag{5}$$

where $I(\cdot)$ is the indicator function and

$$w_k = \prod_{\ell=k}^{t-1} \lambda_\ell, \quad n_t = \sum_{k=1}^{t} w_k.$$

Subsequently, $\tilde{\boldsymbol{p}}_t \in \mathbb{R}^K$ is used to denote the vector of adaptive maximum likelihood estimates whose ith component is given by $\tilde{p}_t^{(i)}$.

3.2 Recursive updates

As presented, the computation of $\tilde{\boldsymbol{p}}_t$ is not suitable for streaming data. This is because Eq. (5) requires maintaining *all* historical data in computer memory. This issue is alleviated by rewriting Eq. (5) as the following recursive update equations:

$$n_t = \lambda_{t-1} n_{t-1} + 1, \tag{6}$$

$$\tilde{p}_t^{(i)} = \left(1 - \frac{1}{n_t}\right) \tilde{p}_{t-1}^{(i)} + \frac{1}{n_t} I(d_t = i). \tag{7}$$

At any time t, the current adaptive estimate $\tilde{\boldsymbol{p}}_t$ can be computed via Eqs. (6)–(7) only having to store a few parameters in computer memory. Further, this estimate can be computed using a constant number of floating point operations per time-step, making it suitable for high-frequency data.

The collection of forgetting factors are allowed to vary in time. An alternate, but less general, approach would be to fix the forgetting factors to some user specified value. This is called *fixed forgetting*, and the special case where the collection of forgetting factors are all replaced by one will prove useful in Sect. 4. In this case, Eqs. (5) and (7) represent, respectively, the nonsequential and sequential equations for the static maximum likelihood estimate $\hat{\boldsymbol{p}}_t$ discussed in Sect. 2.1. Therefore, the static maximum likelihood estimates for the category probabilities can also be computed without introducing any computational/storage overhead by setting $\lambda_{t-1} = 1$ in Eq. (6).

3.3 Tuning the forgetting factors

How quickly $\tilde{\boldsymbol{p}}_t$ reacts to drift depends entirely on the values of the forgetting factors. Larger values will continue to give a considerable amount of weight to older observations, which will hinder the ability of $\tilde{\boldsymbol{p}}_t$ to accurately estimate a new segment's category probabilities when the stream experiences drift. Conversely, smaller values will result in poorer estimates during periods of stationarity, as 'relevant' data will be quickly forgotten. It is clear that a way of tuning these parameters online without any user supervision is a necessity in the streaming paradigm.

At each time-step, the forgetting factors are updated via a stochastic gradient descent step of the form

$$\lambda_t = \lambda_{t-1} + \eta \nabla \left(\sum_{i \in \mathcal{S}} I(d_t = i) \; \log \left(\tilde{p}_{t-1}^{(i)} \right) \right), \tag{8}$$

where $0 < \eta \ll 1$ is a step-size and the gradient is taken with respect to the forgetting factors. The function that is being differentiated in Eq. (8) represents the *one-step ahead log-likelihood* and is an empirical measure of how the estimates at time $(t-1)$ fit the current data point (Anagnostopoulos et al. 2012).

Since data streams are unbounded, it is infeasible to store every forgetting factor in computer memory. This makes the gradient computation appearing in Eq. (8) challenging. These issues have been rigorously investigated in Bodenham (2014) and were shown to agree with the heuristic argument proposed in Anagnostopoulos et al. (2012). The latter supposes that the adaptive estimates are a function of a *single* forgetting factor λ and proceeds by taking the appropriate scalar derivative. Equation (8) can then be reexpressed as

$$\lambda_t = \lambda_{t-1} + \eta \left(\sum_{i \in \mathcal{S}} I(d_t = i) \, \frac{\nabla \tilde{p}_{t-1}^{(i)}}{\tilde{p}_{t-1}^{(i)}} \right),$$

where the operator '∇' represents differentiation with respect to the scalar variable λ and the gradients can be recursively updated via direct differentiation of Eqs. (6)–(7), resulting in

$$\nabla n_t = \lambda_{t-1} \nabla n_{t-1} + n_{t-1},$$
$$\nabla \tilde{p}_t^{(i)} = \left(1 - \frac{1}{n_t}\right) \nabla \tilde{p}_{t-1}^{(i)} - \frac{\nabla n_t}{n_t^2} \left(I(d_t = i) - \tilde{p}_{t-1}^{(i)} \right).$$

Similar to Eqs. (6)–(7), the gradient updates require minimal storage and can be computed using a constant number of floating point operations per time-step. These time-varying forgetting factors, recursively updated using the stochastic gradient descent method, will be subsequently referred to as *adaptive forgetting factors*.

In developing the adaptive forgetting factors, a fixed step-size η was introduced. The behavior of the forgetting factors, for various choices of η, has been investigated in Bodenham (2014), and it has been shown that a range of 'sensible' values results in favorable behavior of the forgetting factors. From experience, any value $\eta = 10^{-k}$ for $k \in [2, 5]$ results in acceptable performance. There is guidance for tuning η online; however, most methods introduce other control parameters that need to be specified. Moving forward, we keep η constant, as opposed to introducing other parameters. See Ruder (2016) for a discussion on variations of the gradient descent algorithm.

Figure 2 shows the behavior of the adaptive estimates $\tilde{\boldsymbol{p}}_t$ and the static estimates $\hat{\boldsymbol{p}}_t$. In this illustration, $K = 3$ and the estimates were averaged over 5000 data streams, all experiencing a change in distribution at time $t = 500$. Before the change, both estimates are indistinguishable, which is to be expected since during periods of stationarity the adaptive forgetting factors should be close to one (which is precisely the static case). However, after the changepoint, $\tilde{\boldsymbol{p}}_t$ quickly adapts to the probabilities in the new segment, while $\hat{\boldsymbol{p}}_t$ struggles to accurately estimate the new probabilities. This is because data prior to the changepoint are contributing equally to $\hat{\boldsymbol{p}}_t$, hindering its ability to maintain accurate estimates. We remark that there is a small 'recovery period' required for the adaptive estimates to begin accurately estimating a new segments probabilities; however, this delay is small relative to the amount of data arriving from the stream and should not be a concern in practical applications. The behavior of $\tilde{\boldsymbol{p}}_t$ and $\hat{\boldsymbol{p}}_t$ during periods of stationarity/drift forms the basis of our change detection method and is discussed in the next section.

4 A multinomial change detection method

The intuition behind the multinomial change detection method (MCDM) relies on the behavior of the adaptive ($\tilde{\boldsymbol{p}}_t$) and static ($\hat{\boldsymbol{p}}_t$) estimates during periods of stationarity and drift. The idea of comparing the behavior of adaptive and static estimates is an extension of the work in Plasse et al. (2017), where a similar concept was developed for detecting

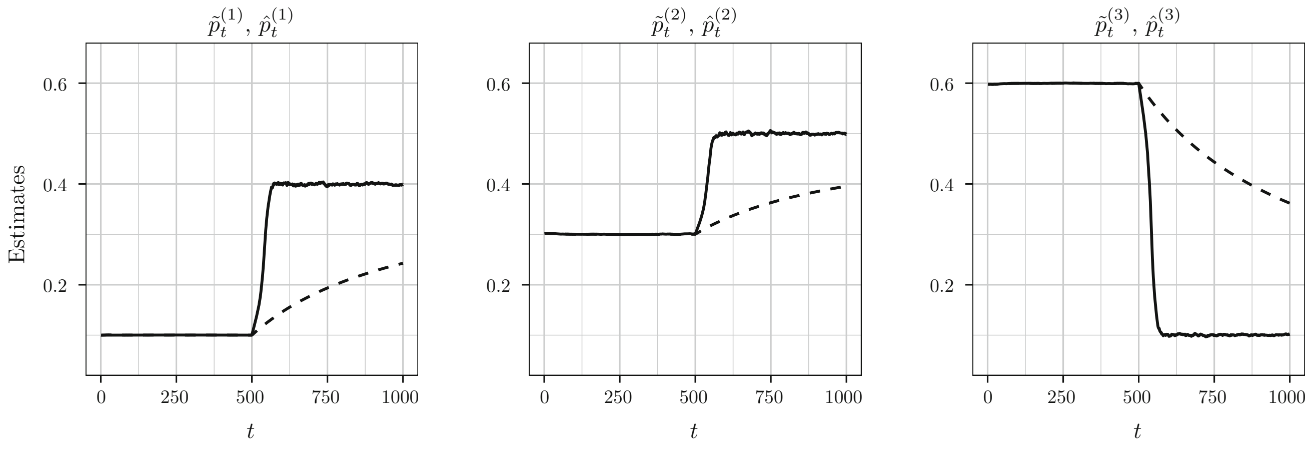

Fig. 2 An illustration showing the behavior of the adaptive estimates $\tilde{\boldsymbol{p}}_t$ (solid) and the static estimates $\hat{\boldsymbol{p}}_t$ (dashed) averaged over 5000 data streams with a change in distribution at time $t = 500$. The multinomial distributions generating the data are $\boldsymbol{p}_t = (1/10, 3/10, 6/10)^\top$ before the changepoint and $\boldsymbol{p}_t = (4/10, 5/10, 1/10)^\top$ after the changepoint

changes in a cyber-physical system. As $\tilde{\boldsymbol{p}}_t$ was developed to react quickly to changes in the data-generating process, when the stream experiences drift, the adaptive estimates will respond to the change much quicker than the static estimates, resulting in $\tilde{\boldsymbol{p}}_t$ and $\hat{\boldsymbol{p}}_t$ *diverging*. This divergence will be quantified in the next section by appealing to the commonly used Kullback–Leibler (KL) divergence.

4.1 KL divergence

The KL divergence introduced by Kullback and Leibler (1951) is a nonsymmetric measure characterizing how one probability distribution deviates from a second distribution. Let P and Q be K-category multinomial distributions with category probability vectors $\boldsymbol{p}$ and $\boldsymbol{q}$. Then, the KL divergence between P and Q is given by

$$\mathrm{KL}(P \mid\mid Q) = \sum_{i \in \mathcal{S}} p_i \log\left(\frac{p_i}{q_i}\right),$$

which is zero if and only if $p_i = q_i$ for every $i \in \mathcal{S}$.

Suppose $d_{1:t}$ have been observed, and that $\tilde{\boldsymbol{p}}_t$ and $\hat{\boldsymbol{p}}_t$ have been recursively computed via the update equations given in Sect. 3. Then, a plug-in estimate for the KL divergence, between two multinomial distributions being monitored by adaptive and static estimates, is given by

$$\kappa_t = \sum_{i \in \mathcal{S}} \tilde{p}_t^{(i)} \log\left(\frac{\tilde{p}_t^{(i)}}{\hat{p}_t^{(i)}}\right).$$

During stationary periods κ_t should be close to zero, and any changes in the data-generating process should result in significant fluctuations in κ_t. A change can then be flagged whenever $\kappa_t > \varepsilon_t$, where ε_t is a suitably chosen threshold.

Optimally choosing a threshold in the streaming paradigm remains a challenging task. Typically, e.g., see Ross et al. (2011), the threshold is chosen so that the probability of the detector raising a false alarm remains (approximately) fixed. This can be achieved by performing Monte Carlo simulations before a stream is analyzed in hopes of gaining insight into how the value of the threshold affects the false positive rate. This usually results in a discretization of the threshold values that seem 'sensible,' which is of course subjective. This difficulty is exacerbated by the fact that $\kappa_t \in [0, \infty)$, that is, sensible choices for the threshold are not obvious. In the next section, a proposed solution to this issue is discussed, resulting in easily computable *adaptive thresholds*.

4.2 Adaptive thresholds

In this section, a choice for ε_t is developed. Our method introduces a control parameter $\beta \in (0, 1)$, which has a meaningful interpretation in the context of change detection. Furthermore, this control parameter being bounded in $(0, 1)$ allows for Monte Carlo simulations to be conducted, as an obvious discretization is available.

A fairly conservative upper bound for κ_t is given by

$$\kappa_t \leq K \left\| \tilde{\boldsymbol{p}}_t \Big/ \sqrt{\hat{\boldsymbol{p}}_t} \right\|_\infty^2, \tag{9}$$

where K is the constant number of categories and all operations on the vectors $\tilde{\boldsymbol{p}}_t$ and $\hat{\boldsymbol{p}}_t$ appearing on the right hand side of (9) are applied component wise. An adaptive threshold can then be defined as

$$\varepsilon_t = \beta \left(K \left\| \tilde{\boldsymbol{p}}_t \Big/ \sqrt{\hat{\boldsymbol{p}}_t} \right\|_\infty^2 \right).$$

The value of β can be viewed as an 'allowance,' i.e., β determines how large the KL divergence can get in relation to the upper bound before a changepoint is flagged.

As previously mentioned, thresholds are commonly chosen so that the false positive rate remains approximately fixed. This can be equivalently stated as choosing β to maintain, on average, a fixed number of observations processed until a false positive is flagged by the detector. The latter is referred to an *average run length* in the literature (Page 1954) and is denoted ARL_0. This average run length and other performance measures are discussed in Sect. 6.1.

A Monte Carlo simulation was conducted to inspect how different values of β affected ARL_0. The category probabilities in each simulation were uniformly sampled from the unit simplex in $\mathbb{R}^K$, and for every β, the MCDM was implemented on 250 data streams of length 5000 with no changepoints present. This is a common way of approximating ARL_0, and the results for several values of K are displayed in Fig. 3. Any $\beta > 0.1$ achieved a perfect ARL_0 of 5000 and are omitted from the plots. Figure 3 is reminiscent of the trade-off to be considered when developing online detection strategies. It appears that any $\beta > 0.1$, for all K considered, will result in a large ARL_0. However, larger values of β will result in the MCDM taking longer to detect any true changepoints, or failing to detect the changepoint entirely. Figure 3 also reveals that, for larger values of K, the MCDM typically performs worse for smaller values of β, but quickly begins reporting perfect values of ARL_0 when β is slightly increased. Therefore, care must be taken in choosing a value for the allowance β.

To extend the results of the simulations, consider a sigmoidal function of the form

$$\mathrm{ARL}_0\,(\beta \mid c_1, c_2, c_3) = \frac{c_1}{1 + \exp\left[\frac{c_2 - \beta}{c_3}\right]}, \tag{10}$$

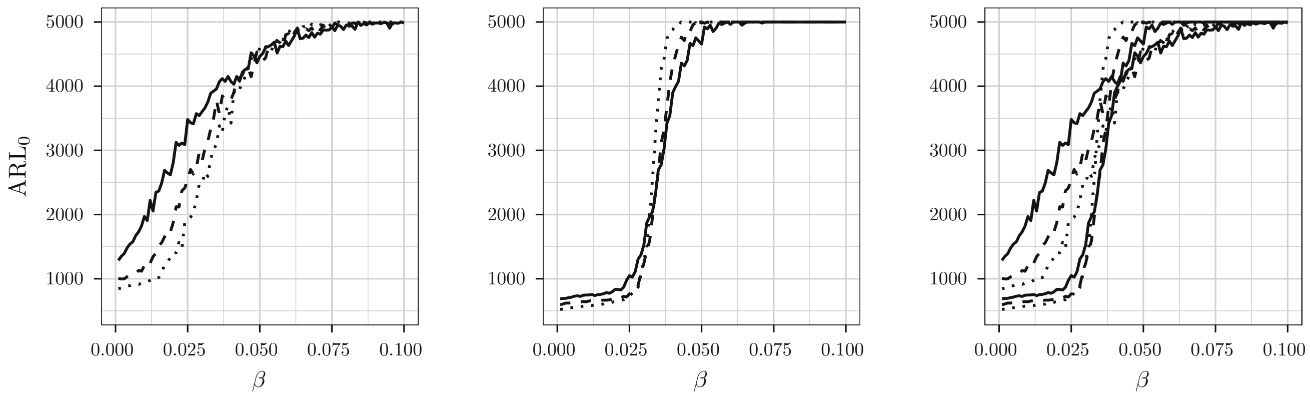

Fig. 3 Plots of ARL$_0$ for various values of the allowance β and number of categories K. Any $\beta > 0.1$ resulted in the MCDM never flagging a false positive. Left: K values 3 (solid), 6 (dashed), and 10 (dotted). Center: K values 25 (solid), 50 (dashed), and 100 (dotted). Right: A combination of the previous two graphs to show the similarity of the curves across all values of K. Observe the sigmoidal shape of each curve

whose coefficients c_1, c_2, and c_3 can be estimated using the results obtained from the Monte Carlo simulations. Equation (10) can then be inverted so that, given a desired value of ARL$_0$ (or equivalently a false positive rate), the value of the allowance may be chosen according to

$$\beta = c_2 - c_3 \log\left(\frac{c_1}{\text{ARL}_0} - 1\right) \quad \text{ARL}_0 < c_1. \tag{11}$$

The coefficients c_1, c_2, and c_3 were computed for each value of K by fitting a self-starting logistic model (Pinheiro and Bates 2000) to the results obtained from the simulations. The values for the coefficients were similar for every K, and moving forward, the values $c_1 = 5000$, $c_2 = 0.023$, and $c_3 = 0.001$ are used. As an example, if an ARL$_0$ of 1000 is desired, Eq. (11) yields an allowance of $\beta = 0.021614$.

A few comments are in order. First, observe that the estimated coefficient c_1 is the length of the streams in the simulation study. This is intuitive as c_1 in Eq. (10) represents the horizontal asymptote of the sigmoid. If larger values of ARL$_0$ are desired, these simulations can be repeated increasing the length of the data streams. Secondly, the only quantity required to compute the adaptive thresholds is a specified value for ARL$_0$ (or false positive rate). In many applications, a constraint on the false positive rate is typically available, as the amount of time to investigate the validity of detections may be costly. Lastly, we remark that other measures, such as the Hellinger distance or symmetric versions of the KL divergence, could have been used in the development of the MCDM—all that is required is an upper bound on the measure so the adaptive thresholds can be defined. Using other measures would result in different coefficients for the sigmoid function in (10), that is, β is measure dependent. However, as the allowance is tuned to approximate a desired value of ARL$_0$, the MCDM will behave similarly for various measures. Due to this, we favor the well-studied KL divergence and other measures are beyond the scope of this work.

At this stage, an entire framework for sequentially and adaptively monitoring a multinomial distribution has been developed. Based on the framework, an MCDM was constructed which requires few values to be specified. The values that must be chosen by a practitioner are: the length of the grace period, a value of ARL$_0$ so that β may be computed, and the step-size used in tuning the forgetting factors. These quantities, unlike previous work, can be chosen without having to rely on unknown characteristics of the stream. Additionally, the length of the grace period and ARL$_0$ can be chosen in terms of constraints on the false positive rate, which are typically available in practice. In Sects. 6 and 7, the effectiveness of the MCDM will be shown on synthetic and real data streams.

5 Comparison methods

This section discusses four methods that the MCDM will be compared to in the simulation study of Sect. 6. Two methods are nonsequential change detectors whose computational complexity is too demanding for streaming applications. However, in the absence of a state-of-the-art streaming detector for categorical data, these methods provide benchmarks for the MCDM. The other two detectors are suitable for streaming data and are commonly used in the literature.

5.1 Bernoulli CUSUM

In the statistical quality control literature, control charts are commonly used to detect changes in a stochastic process. One such chart, briefly mentioned in Sect. 1.1, is the CUSUM chart introduced in Page (1954). One modification of the CUSUM chart was presented in Reynolds Jr and Stoumbos (1999, 2000) and makes the chart applicable to streams of Bernoulli random variables. These modifications are briefly stated below, where the notation used follows that of Reynolds Jr and Stoumbos (1999). See Montgomery (2007) for a thorough discussion of the CUSUM chart as well as other techniques used in quality control.

Suppose $X_{1:t} \sim \text{Bernoulli}(p_0)$, where the parameter p_0 is commonly referred to as the *in-control* value of the stochastic process. The control chart aims to detect a change (either an increase or decrease) in p_0 to some *out-of-control* value. For detecting increases and decreases in p_0, respectively, the out-of-control parameters will be denoted by p_1^+ and p_1^-.

Detecting an increase in p_0, where the terms fraction conforming and nonconforming are typically used (Montgomery 2007), is frequently a primary concern. This is typical in applications where p_0 would represent the probability of finding a defect in some manufactured item. Clearly, decreases in p_0 are of no concern in this scenario. However, since this work is concerned with monitoring a multinomial distribution for multiple changes, increases *and* decreases in p_0 must be considered.

The Bernoulli CUSUM chart sequentially monitors statistics defined by

$$B_t^+ = \max\left\{0, B_{t-1}^+\right\} + \left(x_t - \gamma^+\right),$$
$$B_t^- = \min\left\{0, B_{t-1}^-\right\} + \left(x_t - \gamma^-\right),$$

where γ^+ and γ^- are commonly called the *reference values* (Reynolds Jr and Stoumbos 1999) and $x_t \in \{0, 1\}$ is a realization of the random variable X_t. The reference values may be chosen using the sequential probability ratio test (Ghosh and Sen 1991) representation of the chart (Page 1954). This leads to $\gamma^* = r_1^*/r_2^*$, where

$$r_1^* = -\log\left(\frac{1-p_1^*}{1-p_0}\right), \quad r_2^* = \log\left(\frac{p_1^*(1-p_0)}{p_0(1-p_1^*)}\right),$$

and the asterisk symbol, for brevity, assumes a value in $\{+, -\}$. The chart flags a change whenever $|B_t^*| > h^*$ for suitably chosen control parameters h^*. These control parameters remain fixed until a change is flagged by the detector where an intervention procedure, such as the grace period in Sect. 2.2, is then implemented to reinitialize the chart.

Similar to Sect. 4.2, the values for h^* are typically chosen to approximate a desired value of ARL_0. We adopt the corrected diffusion approximation discussed thoroughly in Reynolds Jr and Stoumbos (1999), which is an extension of the work in Siegmund (1979) to choose the values of h^*. Subsequently, the CUSUM chart with Bernoulli modifications will be referred to simply as the CUSUM chart.

The values that dictate when the chart becomes out of control are parameters that must be specified by the user, and are difficult to set in the multiple changepoint setting. This is because the control parameters are typically chosen to detect a change of a certain magnitude. In the multiple changepoint scenario where changes of varying magnitudes will exist, choosing values for these parameters becomes nontrivial. Further, depending on the value of p_0, there may be issues in choosing p_1^* as it may be too close to the boundary of $[0, 1]$.

As described, the CUSUM chart is not suitable for detecting changes in a multinomial distribution. Thus, when implementing the chart, the binarization of the categorical process given by (2) is used. The binarization is a multivariate stream, whereas the proposed CUSUM chart is univariate. The chart is therefore implemented on *each component* of the vectors appearing in (2). Given a single data stream, this will require K runs of the chart, which will increase the amount of computation needed to process the stream.

It should be taken into account that multivariate control charts such as the multinomial CUSUM chart discussed in Sect. 1.1 exist. See Bersimis et al. (2007) for an excellent survey. However, most of the multivariate charts assume the observations are continuous and are not directly relevant to this work. Further, since it is impractical to specify the directions of the out-of-control parameters for every changepoint, we follow the advice of Ryan et al. (2011), who recommend using K Bernoulli charts in replace of a multinomial CUSUM chart.

5.2 CPM

The R package **cpm** (Ross et al. 2015) provides various changepoint models for the parametric or nonparametric detection of multiple changepoints in a univariate data stream. Each method assumes $X_t \sim F_t$ for some known or unknown distribution F_t, and a change is detected whenever $D_t > h_t$, for a suitably chosen test statistic D_t and threshold h_t.

There are several models implemented in the **cpm** package which may be used as a comparison method. However, a complete contrast of the MCDM to this R package is not in the scope of the paper. Therefore, in what follows the computation of D_t is based on the *Mann–Whitney test* which has been investigated in Hawkins et al. (2003) and Pettitt (1979). This test makes no assumptions on the distribution generating the observations and has been extended to the streaming setting in Ross et al. (2011). Further, the sequence of thresholds h_t is computed, similar to this work, via Monte Carlo simulation. This method, subsequently denoted by CPM, is sufficient for our purposes. Lastly, similar to the CUSUM chart, the CPM method must be implemented on the binarizations of the data stream, requiring K runs of the method to analyze a single data stream.

5.3 ECP

A *nonsequential* alternative, which can handle multivariate data, is provided in the R package **ecp** (James and Matteson 2013). This package can detect multiple changepoints via a divisive or agglomerative approach. The divisive method sequentially searches for changepoints by applying a bisection technique, whereas, the agglomerative method determines an optimal partitioning. The divisive approach is used in the sequel, and this method will be referred to as ECP.

The divisive procedure available in the **ecp** package is based on work developed in Matteson and James (2014) and assumes that the multivariate observations are independent with a finite kth absolute moment for some $k \in (0, 2]$. A divergence measure is then used to detect distributional changes, which is based on the energy statistic developed in Szekely and Rizzo (2005). The significance of an estimated changepoint is assessed via a permutation test which requires a specified number of maximum permutations M to be prescribed. A significance level α for the test is also required.

The computational complexity of the divisive approach is quadratic making it ill-suited for streaming data. However, since this is a nonsequential method which utilizes multiple passes through the data, it provides a good benchmark for the MCDM. Since this method can handle multivariate data, unlike CUSUM and CPM, it can be directly applied to the data stream in (2) without having to consider each component of the vectors separately.

5.4 PELT

The pruned exact linear time (PELT) method was introduced in Killick et al. (2012) and is implemented in the R package **changepoint** (Killick and Eckley 2014). This method is able to detect multiple changepoints in a nonstationary time series by utilizing several passes over the data. Hence, PELT will also not be suitable for a majority of streaming applications. Nonetheless, this method will provide a good benchmark when investigating the performance of the MCDM.

Given a cost function, PELT is able to return an optimal segmentation of the data by applying a modification to the optimal partitioning algorithm (Jackson et al. 2005). Provided certain assumptions hold, such as the number of changepoints increasing linearly with the size of the data, PELT has a linear computational cost. This is an improvement from optimal partitioning and ECP which have quadratic computational costs. However, if assumptions fail to hold the worst-case complexity is also quadratic in the length of the data. The **changepoint** package is currently designed for univariate time series, thus PELT is implemented over K data streams where each stream is a sequence of vector components given by (2). If desired, a multivariate cost function could be prescribed to PELT, allowing the method to analyze a stream in one run; however, using the binarization is sufficient for our purposes.

Table 1 displays the methods, whether or not they are applicable to the streaming setting, if they are implemented on the categorical stream given in (1) or the binarizations provided in (2), how many runs are required given a single data stream, and if a publicly available R package is used to obtain results in the simulation section.

Table 1 A summary of the methods and some their characteristics, including if the methods are suitable for streaming data, if they are implemented on the binarizations or the categorical data stream, if they only have to be run once given a single data stream, and if there is an available R package that is used in the simulation section

	Streaming	Binarization	1-Run	R package
MCDM	✓	✗	✓	✗
CPM	✓	✓	✗	✓
CUSUM	✓	✓	✗	✗
ECP	✗	✓	✓	✓
PELT	✗	✓	✗	✓

6 Simulation study

In this section, the MCDM is implemented on synthetic data streams and compared to the methods discussed in Sect. 5. Section 6.1 introduces performance measures used to compare the detectors, and Sect. 6.2 discusses the experimental design of the simulations. A way of synthetically generating changepoints so streaming and non-streaming algorithms may be fairly compared is also introduced. Section 6.3 analyzes the results.

6.1 Performance measures

Two commonly used performance measures for change detection algorithms are the *average run lengths* ARL_0 and ARL_1 (Page 1954). The scalar ARL_0 can be defined as the average number of observations processed until a false positive is flagged and was fundamental in developing a way for choosing the parameter β in Sect. 4.2. The other average run length, ARL_1, is defined as the average detection delay. In practice, higher values of ARL_0 *and* lower values of ARL_1 suggest that a change detector is performing favorably; however, there is a trade-off to consider as either of the run lengths can be improved at the expense of the other. To compute the average run lengths exactly would require the distributions of ARL_0 and ARL_1, which in most cases are difficult to obtain. Therefore, it is customary to approximate the run lengths via Monte Carlo simulation.

The average run lengths are not complete measures of performance in the multiple changepoint scenario as they do not take into consideration how many changepoints a detector flags nor the number of changepoints that remain undetected. Due to this, the *proportion of changepoints correctly detected* (CCD) and the *proportion of detections that are not false* (DNF) are considered. Both CCD and DNF assume values in [0, 1], with values closer to one (for both measures) indicating better performance. As an example, a CCD and DNF both equaling one indicate that a detector correctly identified all changepoints while raising no false alarms. Similar to the average run lengths, these quantities are approximated via Monte Carlo simulation; see Bodenham (2014) for more discussion.

6.2 Experimental design

All methods are implemented on 2000 Monte Carlo replicates, where the length of each stream is a function of parameters introduced in this section. The number of categories and changepoints in each stream, respectively, assume values in the sets $K \in \{3, 6, 10, 25\}$ and $m \in \{0, 1, 5, 10\}$.

How the changepoints are generated in each stream warrants discussion. As sequential and nonsequential detectors are being compared, care needs to be taken in what constitutes a 'true detection,' sequential detectors will never correctly flag a changepoint before one occurs; however, nonsequential detectors may accurately detect a change *before* the actual location of the changepoint. These detections should not be ignored; therefore, a 'bubble' is defined around the location of each changepoint. More precisely, suppose τ is a changepoint location and let $\xi \in \mathbb{Z}^+$. Then, for the nonsequential detectors, any detection made in the interval $[\tau - \xi, \tau + \xi]$ is considered a true detection; whereas, the true detection region for sequential detectors is defined as $[\tau, \tau + \xi]$. Any detector that does not flag a change within ξ time-steps of the true location (on either side of τ for nonsequential detectors) is said to have missed the detection.

How the changepoints are generated depends on the value of m. When $m = 0$, there is no changepoint to generate, and when $m = 1$, the changepoint is randomly placed near the middle of the stream. In the case where $m > 1$, a changepoint generation scheme similar to that in Bodenham and Adams (2017) is used. That is, the changepoints are generated according to

$$\begin{aligned}\tau_1 &= 2\xi + \rho + \nu_1,\\ \tau_k &= \tau_{k-1} + 2\xi + \rho + \nu_k \quad k = 2, \ldots, m,\end{aligned}$$

where $\rho \in \mathbb{Z}^+$ is a 'pad' parameter to ensure that the true detection regions around the changepoints do not overlap and $\nu_k \sim \text{Poisson}(r)$.

Under this scheme, the expected length between consecutive changepoints is $2\xi + \rho + r$. Given an average length between changepoints $L > 2\xi + \rho$, the rate parameter r is chosen as $r = L - 2\xi - \rho$. The length of the stream S is then taken to be the smallest integer multiple of 2500 satisfying $S > m(2\xi + \rho + r)$. When $m = 0$ or $m = 1$, the length of the stream is taken as $S = 5000$. Additionally, $(\xi, \rho, L) = (50, 20, 500)$ are chosen. Note that these values are only used in the generation of the changepoint locations and are *not* control parameters for any particular method.

For the MCDM, the length of the burn-in and grace period regions is chosen as $B = S/10$ and $G = 100$. Burn-in periods are typical in applications and are used to obtain initial parameter estimates for the category probabilities. Care was taken so that these values were used similarly in the other change detection methods. For example, in the R packages **changepoint** and **ecp**, the scalar G was used as the minimum segment length between changepoints. Moreover, the MCDM and the CPM and CUSUM methods allow for a desired value of ARL_0 to be specified. This is chosen as 2000, resulting in an allowance parameter for the MCDM being $\beta = 0.022594$. The step-size used in the gradient descent step for the MCDM is chosen as $\eta = 10^{-3.5}$, which has been previously shown to result in favorable behavior of the forgetting factors; refer to Anagnostopoulos et al. (2012) and Bodenham and Adams (2017).

The CUSUM chart must be implemented over K data streams. Thus, the in-control and out-of-control parameters are actually vectors of length K and will be denoted by $\boldsymbol{p}_0$, $\boldsymbol{p}_1^+$, and $\boldsymbol{p}_1^-$. The in-control vector $\boldsymbol{p}_0$ is estimated via a burn-in period of length B, and the components are truncated to the interval $[p_{\min}, p_{\max}]$ to ensure no component is too close to the boundary of [0, 1]. The out-of-control vectors are then chosen as

$$\boldsymbol{p}_1^+ = \min\left\{1, \boldsymbol{p}_0 + \delta\right\}, \quad \boldsymbol{p}_1^- = \max\left\{0, \boldsymbol{p}_0 - \delta\right\},$$

where $\delta \in \{0.01, 0.1, 0.25, 0.5\}$ determines the magnitude of the change one is looking to detect, and the minimum and maximum are applied component wise. The truncation parameters in the simulations are given by $(p_{\min}, p_{\max}) = (0.001, 0.999)$.

Lastly, for ECP the maximum number of allowed permutations was set to $M = 30$, and the significance level for the permutation test was chosen as $\alpha = 0.05$. For the methods whose R implementations are being used, all other parameters not mentioned are chosen according to the package defaults.

6.3 Results

Results for the simulations are summarized in Fig. 4, Tables 2 and 4 (found in the 'Appendix' and displays all numerical results). Figure 4 displays the *differences* between the MCDM and a competing methods performance measure, so that any point appearing above the line $y = 0$ indicates that the MCDM reported better performance. The vertical dashed lines partition the plots by values of K, where from left to right the value of K *increases*. When estimating ARL_1, only detections made within the allowed detection regions contributed to the value of ARL_1, that is, if a detector missed a detection, it did not worsen the ARL_1. Thus, the proportion of simulations where the changepoint was correctly identified is also investigated; these are provided in Table 2.

Consider the ARL_0 results in the top left of Fig. 4. From the figure, we see that the MCDM reported better ARL_0 values only when compared to the CUSUM method (for $K < 25$). The desired value of ARL_0 was chosen as 2000. Averaging over K, the MCDM reports a value of 2021.73, reinforcing the validity of our approach for choosing the allowance parameter β (refer to Table 4 for the numerical values for ARL_0). Now, examine the ARL_1 results displayed in the top center plot of Fig. 4. In this case, the MCDM reported better values for ARL_1 only when compared to the CPM method (for $K > 6$). Although after rescaling this run length, even though most of the competing methods report smaller detection delays, the MCDM is reporting values close to the other methods, relative to the size of the stream. Further, inspection of Table 2 shows that the MCDM correctly detects the change in a large proportion of simulations. Notice that when $K = 25$, the CPM and PELT methods, respectively, only correctly detected the changepoint 9% and 23.5% of the time, whereas the MCDM was able to correctly identify 82% of the changes. For both average run lengths, the best performance is typically obtained by the nonsequential detectors. This is not surprising as these methods are able to make multiple passes over the data.

The top right plot, as well as the bottom row of plots in Fig. 4 display the CCD and DNF results. This corresponds to the multiple changepoint scenario and highlights the advantages of our approach. As seen from the figure, the MCDM outperforms all of the sequential detectors and even outperforms PELT in terms of CCD (and DNF when $K = 25$). The only method which seems to consistently outperform the MCDM is the ECP method. However, due to the quadratic time complexity of this method, this approach is unsuitable for streaming data and has only been included as a benchmark. To reinforce this, when $(S, K, m) = (7500, 25, 10)$, the ECP method took, on average, over 20 min to run *per simulation*, whereas the MCDM method attained similar performance in roughly 2 seconds per simulation.

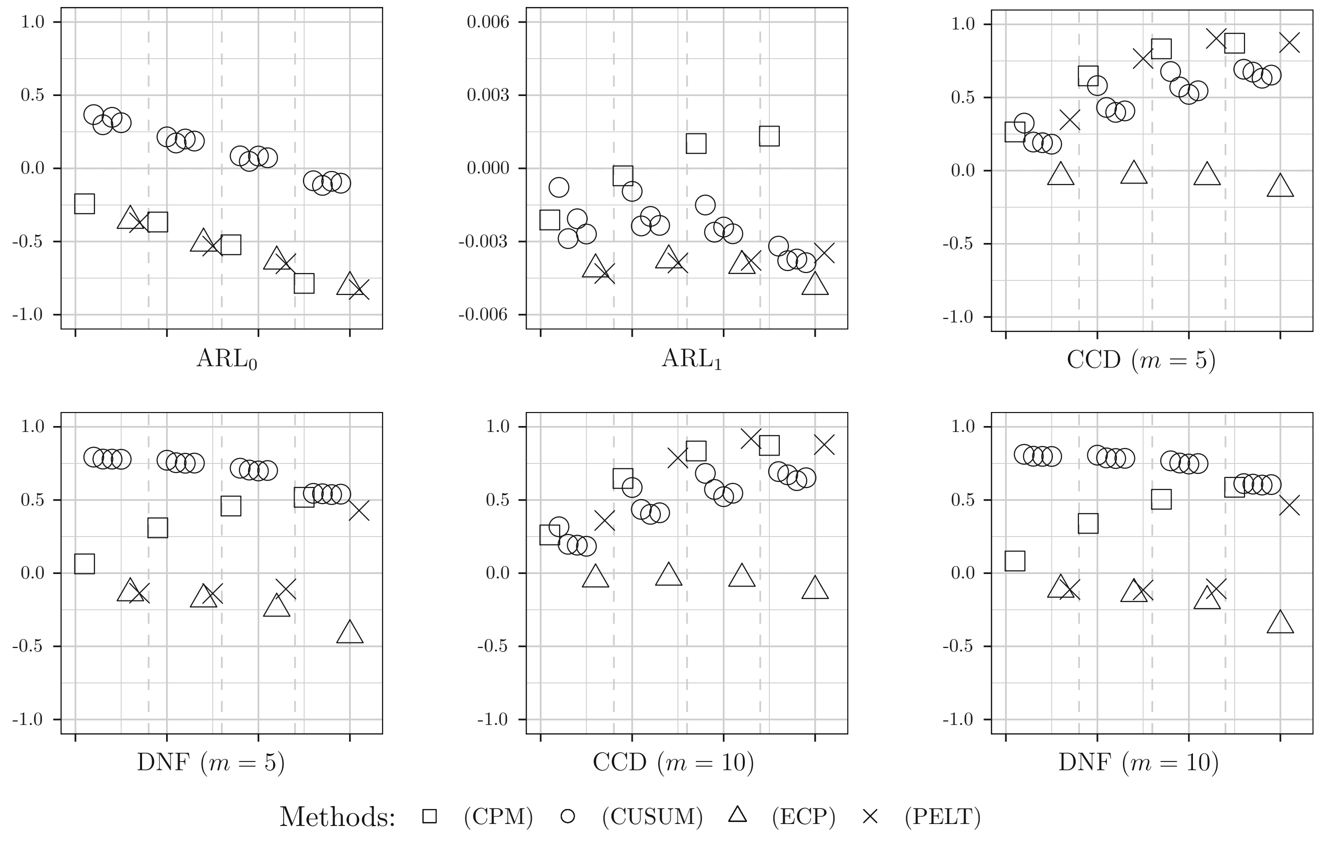

Fig. 4 For a given performance measure, each point represents the *difference* between the MCDM and a competing methods performance measure, where any point above the line $y = 0$ indicates that the MCDM reported better performance. The average run lengths have been rescaled to the unit interval so that all measures have the same range, and the vertical dashed lines partition the plots by the value of $K \in \{3, 6, 10, 25\}$, so from left to right, the value of K *increases*. Recall that for CUSUM multiple values of δ were considered, which is why the method has more points appearing in the plots

Table 2 For simulations with a single changepoint, the proportion of simulations where a particular method *detected* the changepoint within its detection region is displayed. Values closer to one indicate that a method has correctly identified the changepoint in a large proportion of simulations

	3	6	10	25
MCDM	0.945	0.953	0.937	0.820
CPM	0.991	0.967	0.770	0.090
CUSUM(0.01)	0.937	0.961	0.982	0.999
CUSUM(0.10)	0.997	0.999	0.999	1.000
CUSUM(0.25)	0.995	0.999	0.999	0.999
CUSUM(0.50)	0.993	0.998	0.997	0.996
ECP	1.000	1.000	1.000	0.999
PELT	0.999	0.997	0.957	0.236

These simulations provide several rewarding conclusions. First, the MCDM provides comparable performance in terms of average run length and provides significant improvements in the multiple changepoint scenario. Only the ECP method is able to consistently perform well, but is a rather unfair comparison due to its high computational demands, whereas the MCDM is suitable for streaming data and does not sacrifice much in terms of detection power. Secondly, given a specified ARL_0 (equivalently, a false positive rate), the choice of β leads to the MCDM approximately respecting this desired run length. Specifying a value for ARL_0 is not a burden in practice, as many practitioners have an idea of how many false positives can be tolerated for a given application. In the next section, the MCDM is implemented on a real-world data stream originating from Imperial College London's computer network.

7 Real data example

Lewis (2008) has estimated that cyber-crime costs approximately \$600 billion per annum. The ability to dynamically monitor network traffic in real time for malicious activity is therefore a pressing concern. In this section, a stream of nearly 40 million computer network events collected from Imperial College London is monitored for changes. The data are discussed in Sect. 7.1, and the MCDM is implemented on the stream in Sect. 7.2 to highlight that the method is suitable for high-frequency data streams occurring in practice. The network traffic has no ground truth labeling of malicious activity, as is frequently the case in cyber-security (Anagnostopoulos 2018). To provide meaning to our results, in Sect. 7.3 the data are modified to mimic the WannaCry cyber attack (Mohurle and Patil 2017), which took place in May 2017.

Table 3 The top ten TCP destination ports, collected over a single router, present in the Imperial College London network. These ten ports amount to approximately 98% of the entire data stream

Port number	Description	Proportion
80	HTTP	~ 0.489
445	Microsoft-DS	~ 0.216
443	HTTPS	~ 0.158
88	Kerberos	~ 0.029
389	LDAP	~ 0.026
631	IPP	~ 0.025
111	SUN remote procedure call	~ 0.012
22	SSH	~ 0.012
135	DCE endpoint resolution	~ 0.009
139	NETBIOS session service	~ 0.003

7.1 The data

The data consist of a sequence of network events collected from a single router in the Imperial College London computer network. Each event relates to a communication between devices, and only the 'well-known' *transmission control protocol* (TCP) events are considered. Much TCP activity is well defined by the 'destination port,' which often indicates the type of activity and is assigned an integer in [0, 1023]. Refer to Comer (2000) for more on TCP protocols. The focus of this section is to highlight that the MCDM is appropriate for *large-scale*, *high-frequency* data streams, and a complete investigation of this network is beyond the scope of the paper.

A total of 39,031,345 events produce a sequence of well-known TCP destination ports, collected over a day in 2015, and make up the data stream to be analyzed. Table 3 displays the top ten ports present in the data, which make up nearly 98% of the entire stream. In consequence, the 1024 destination ports are condensed to the ten ports appearing in Table 3, and an 'other' category representing any port not appearing in the table.

Since the stream consists of over 39 million observations, previous methods that required more than a single run to analyze the data are not suitable for the TCP stream. This would require nearly 440 million observations to be processed, as there are 11 categories present in the stream. The ECP method can detect changes in a single run over the data, but is also not applicable due to its high computational complexity. Henceforth, only the MCDM is implemented on the TCP data stream, and results are discussed in the next two sections.

7.2 Unmodified TCP stream

In this section, the MCDM is implemented on the TCP stream. A burn-in period of length 9,031,345 is used, resulting in a stream of 30 million destination ports being monitored for changes. In cyber-security, practitioners will typically have a rough estimate as to how many false positives can be tolerated throughout the day, as examining detections may be time-consuming and costly. The value of G is chosen to coincide with these concerns since it provides a lower bound on ARL_0, and moving forward the length of the grace period is chosen as $G = 135600$. This results in *approximate* 5 min grace periods as the TCP data arrive at a rate of approximately 452 per second.

Choosing the allowance parameter β merits discussion. Cyber analysts are interested in detectors which flag few false positives, or equivalently, detectors which exhibit large values of ARL_0. Since a large amount of data are collected each day, to meet practitioner's demands would require an ARL_0 on the order of *millions*. Indeed, to flag a false positive roughly every hour in the TCP stream requires an ARL_0 larger than 1.6 million. At the present time, we are unaware of any literature that can choose model parameters to accurately obtain an ARL_0 on this order of magnitude, and we remark that this provides a promising direction for future work. Moving forward, a set of Monte Carlo simulations was conducted with $S = 20{,}000$ and an allowance parameter $\beta = 0.077$ was computed, resulting in the MCDM having an ARL_0 roughly equal to 20,000.

Implementing the MCDM took roughly 9 h to run over the 30 million destination ports. The MCDM can therefore process a day's worth of network data in less than 24 h, making the MCDM a suitable method for the TCP stream. A histogram displaying the time elapsed (in minutes) between consecutive detected changepoints is displayed in Fig. 5. Recall that G was chosen to result in an approximate 5 min grace period, and since data arrives irregularly from the router, detections could theoretically occur in under 5 min. Given the choice for G and β, we would expect the MCDM to flag a detection, on average, roughly every 5.74 min (vertical dashed line in Fig. 5). From the histogram, we see that the detections made by the MCDM behave as expected, reinforcing that the control parameters can be chosen based on practical considerations, and can be successfully applied to a real-world data stream.

The TCP data have no ground truth accompanying it, that is, both the existence and location of potential changepoints are unknown. The purpose of this section was to illustrate that the MCDM scales well with the size of the data and is suitable for real-world problems. In the next section, the TCP data are tampered with so that there is a change in the sequence of destination ports.

7.3 WannaCry

The WannaCry attack took place in May 2017 and is estimated to have affected over 200,000 devices running the Windows operating system, spanning over 150 countries

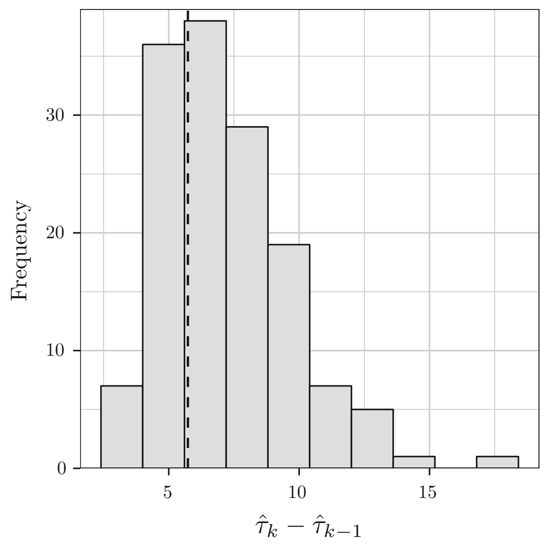

Fig. 5 A histogram of the elapsed time (in minutes) between the estimated location of consecutive changepoints (denoted $\hat{\tau}_k - \hat{\tau}_{k-1}$). The dashed vertical line represents the time between detections that we would expect, on average, given the chosen values for G and β used in analyzing the TCP stream

(Mohurle and Patil 2017). The attack exploited Windows' server message block (SMB) protocol and encrypted valuable data on the user's machine. The user was then issued a 'ransom message' asking to be paid in the Bitcoin cryptocurrency in exchange for a decryption key.

The SMB protocol can run directly over TCP port 445, or via the Netbios API, which may run over TCP port 139 (Parziale et al. 2006)—both of these ports appearing in Table 3. To mimic the WannaCry attack in the sequence of destination ports, these specific ports are tampered with. Specifically, an attack begins at 11:00:00 and lasts until 11:10:00. Starting at 11:00:00, in minute intervals, one of the ports exhibits a burst in activity which lasts for a short period of time. This is done in such a way that the frequency of the tampered ports is inflated by approximately 10% in the data stream.

The values for G and β discussed in Sect. 7.2 are also used when inspecting the altered data stream. The MCDM flagged two detections during the attack period at times 11:00:29 and 11:08:10, which were *not* flagged when deploying the MCDM over the stream without the attack. It is therefore sensible to conclude that the MCDM was indeed successful in detecting the attack.

8 Conclusions

The detection of multiple changepoints in categorical data streams is a nontrivial issue and has not received a significant amount of attention in the literature. The presence of several changepoints makes it particularly difficult to justify setting control parameters, such as fixed thresholds used to detect changepoints, or constant width sliding windows. To the best of our knowledge, this is among the first works to present a method for detecting changes in categorical data streams, while providing a practitioner with evidence on how to choose the parameters that are introduced.

To implement the MCDM requires three values to be specified: the length of the grace period, the desired value of ARL_0, and the step-size used in tuning the forgetting factors. The first two can be chosen based on a minimum, and average desired false positive rate. These values do not depend on unknown dynamics of the stream, and the ability to choose them given constraints on the number of false positives is an attractive quality. Recent literature has investigated the step-size parameter used in tuning the adaptive forgetting factors, and it has been observed that a range of values lead to desirable behaviors. Future work will develop streaming methods for this parameter. Although the number of parameters is not necessarily less than other existing methods, we remark that, unlike existing work, they can be chosen without relying on unknown characteristics of the stream.

In conclusion, the MCDM presents several novel contributions, such as: a way of comparing the behavior between adaptive and static estimates for the multinomial distribution, and an adaptive thresholding technique which can be computed requiring only a desired value for the false positive rate. The MCDM is also robust to missed detections, as the adaptiveness of the forgetting factors will result in accurate estimates regardless of how many changes the stream experiences. Through simulations, the MCDM was shown to accurately detect multiple changepoints with minimal storage/computational overhead while being compared to commonly used sequential and nonsequential detectors. The MCDM was also implemented on nearly 40 million observations, reinforcing that the method is suitable for applications which continuously generate a large amount of data.

Acknowledgements The work of Joshua Plasse was fully funded by an Imperial College London President's PhD Scholarship. The authors would like to express gratitude to Andy Thomas and Imperial College London for the data analyzed in this paper, as well as Dr. Din-Houn Lau for his helpful comments when revising the manuscript.

Appendix

See Table 4.

Table 4 A summary of performance measures for every method and for every $K \in \{3, 6, 10, 25\}$ used in the synthetic simulations

	MCDM	CPM	CUSUM(0.01)	CUSUM(0.10)	CUSUM(0.25)	CUSUM(0.50)	ECP	PELT
$m = 0$								
3	3144.36	4354.73	1307.16	1399.68	1584.99	1653.98	4931.77	5000.00
6	2343.87	4177.44	1267.62	1340.74	1410.92	1476.76	4910.33	5000.00
10	1739.37	4350.91	1312.09	1318.60	1377.61	1500.90	4918.65	5000.00
25	859.30	4790.01	1282.66	1300.60	1365.82	1441.71	4905.93	5000.00
$m = 1$								
3	22.78	12.20	18.89	12.45	9.32	8.38	2.11	1.23
6	20.85	19.33	16.14	11.00	9.16	9.05	2.03	1.45
10	21.94	27.03	14.43	9.91	8.52	8.86	1.96	3.05
25	26.44	33.05	10.49	7.85	7.10	7.51	2.16	9.10
$m = 5$								
3	(0.96, 0.86)	(0.69, 0.79)	(0.63, 0.06)	(0.77, 0.08)	(0.78, 0.08)	(0.76, 0.08)	(1.00, 0.99)	(0.61, 0.99)
6	(0.97, 0.82)	(0.32, 0.51)	(0.39, 0.04)	(0.57, 0.06)	(0.56, 0.06)	(0.53, 0.06)	(1.00, 0.99)	(0.20, 0.95)
10	(0.96, 0.75)	(0.12, 0.29)	(0.28, 0.03)	(0.44, 0.05)	(0.41, 0.05)	(0.38, 0.05)	(1.00, 0.99)	(0.05, 0.86)
25	(0.88, 0.57)	(0.01, 0.05)	(0.18, 0.03)	(0.24, 0.03)	(0.22, 0.03)	(0.20, 0.03)	(1.00, 0.99)	(0.00, 0.14)
$m = 10$								
3	(0.96, 0.88)	(0.70, 0.80)	(0.64, 0.07)	(0.77, 0.09)	(0.77, 0.09)	(0.76, 0.08)	(1.00, 1.00)	(0.60, 1.00)
6	(0.97, 0.85)	(0.32, 0.51)	(0.39, 0.05)	(0.57, 0.07)	(0.56, 0.07)	(0.53, 0.07)	(1.00, 1.00)	(0.18, 0.97)
10	(0.96, 0.81)	(0.13, 0.30)	(0.28, 0.04)	(0.44, 0.06)	(0.42, 0.06)	(0.39, 0.05)	(1.00, 1.00)	(0.04, 0.91)
25	(0.88, 0.64)	(0.01, 0.06)	(0.18, 0.03)	(0.25, 0.04)	(0.23, 0.04)	(0.21, 0.03)	(1.00, 1.00)	(0.00, 0.18)

When $m = 0$ and $m = 1$, respectively, the table shows results for ARL_0 and ARL_1. For $m > 1$, the entries in the table are tuples of the form (CCD, DNF) corresponding to the multiple changepoint setting—the focal point of the paper. Larger values for ARL_0, CCD and DNF and lower values of ARL_1 are indicative of a method performing favorably. For CUSUM, the value in parenthesis corresponds to the value of δ used in the simulations

References

Adams, R.P., MacKay, D.J.: Bayesian online changepoint detection. arXiv:0710.3742 (2007)

Aggarwal, C.C.: Data Streams: Models and Algorithms. Springer, Berlin (2007)

Amirzadeh, V., Mashinchi, M., Yaghoobi, M.: Construction of control charts using fuzzy multinomial quality. J. Math. Stat. **4**(1), 26–31 (2008)

Anagnostopoulos, C.: Weakly supervised learning: how to engineer labels in cyber security. In: Data science for cyber security (2018, Forthcoming)

Anagnostopoulos, C., Tasoulis, D.K., Adams, N.M., Pavlidis, N.G., Hand, D.J.: Online linear and quadratic discriminant analysis with adaptive forgetting for streaming classification. Stat. Anal. Data Min. **5**(2), 139–166 (2012)

Bersimis, S., Psarakis, S., Panaretos, J.: Multivariate statistical process control charts: an overview. Qual. Reliab. Eng. Int. **23**(5), 517–543 (2007)

Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM International Conference on Data Mining, pp. 443–448. SIAM, Philadelphia (2007)

Bodenham, D.A., Adams, N.M.: Continuous monitoring for changepoints in data streams using adaptive estimation. Stat. Comput. **27**(5), 1257–1270 (2017)

Bodenham, D.A.: Adaptive estimation with change detection for streaming data. Ph.D. Thesis, Imperial College London (2014)

Byrd, M., Nghiem, L., Cao, J.: Lagged exact Bayesian online changepoint detection. arXiv:1710.03276 (2017)

Cao, F., Huang, J.Z.: A concept-drifting detection algorithm for categorical evolving data. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 485–496. Springer, Berlin (2013)

Chen, H.L., Chen, M.S., Lin, S.C.: Catching the trend: a framework for clustering concept-drifting categorical data. IEEE Trans. Knowl. Data Eng. **21**(5), 652–665 (2009)

Comer, D.E.: Internetworking with TCP/IP: Principles, Protocols, and Architecture, vol. 1. Prentice-Hall, Upper Saddle River, NJ (2000)

Eiauer, P., Hackl, P.: The use of MOSUMS for quality control. Technometrics **20**(4), 431–436 (1978)

Eichinger, B., Kirch, C., et al.: A MOSUM procedure for the estimation of multiple random change points. Bernoulli **24**(1), 526–564 (2018)

Gama, J.: Knowledge Discovery from Data Streams. CRC Press, Boca Raton (2010)

Ghosh, B.K., Sen, P.K.: Handbook of Sequential Analysis. CRC Press, Boca Raton (1991)

Hawkins, D.M., Qiu, P., Kang, C.W.: The changepoint model for statistical process control. J. Qual. Technol. **35**(4), 355–366 (2003)

Haykin, S.S.: Adaptive Filter Theory. Pearson Education India, New Delhi (2008)

Heard, N., Rubin-Delanchy, P., Lawson, D.J.: Filtering automated polling traffic in computer network flow data. In: IEEE Joint Intelligence and Security Informatics Conference. pp. 268–271. IEEE, Washington (2014)

Höhle, M.: Online change-point detection in categorical time series. In: Kneib, T., Tutz, G. (eds.) Statistical Modelling and Regression Structures, pp. 377–397. Springer, Berlin (2010)

Hou, C.D., Shao, Y.E., Huang, S.: A combined MLE and generalized P chart approach to estimate the change point of a multinomial process. Appl. Math. Inf. Sci. **7**(4), 1487–1493 (2013)

Ienco, D., Bifet, A., Pfahringer, B., Poncelet, P.: Change detection in categorical evolving data streams. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, pp. 792–797. ACM, New York (2014)

Jackson, B., Scargle, J.D., Barnes, D., Arabhi, S., Alt, A., Gioumousis, P., Gwin, E., Sangtrakulcharoen, P., Tan, L., Tsai, T.T.: An algorithm for optimal partitioning of data on an interval. IEEE Signal Process. Lett. **12**(2), 105–108 (2005)

James, N.A., Matteson, D.S.: ecp: an R package for nonparametric multiple change point analysis of multivariate data. arXiv:1309.3295 (2013)

Killick, R., Eckley, I.: changepoint: an R package for changepoint analysis. J. Stat. Softw. **58**(3), 1–19 (2014)

Killick, R., Fearnhead, P., Eckley, I.A.: Optimal detection of changepoints with a linear computational cost. J. Am. Stat. Assoc. **107**(500), 1590–1598 (2012)

Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Stat. **22**(1), 79–86 (1951)

Lewis, J.: Economic impact of cybercrime—no slowing down. Technical report, McAfee (2008)

Matteson, D.S., James, N.A.: A nonparametric approach for multiple change point analysis of multivariate data. J. Am. Stat. Assoc. **109**(505), 334–345 (2014)

Mohurle, S., Patil, M.: A brief study of Wannacry threat: ransomware attack 2017. Int. J. Adv. Res. Comput. Sci. Softw. Eng. **8**(5), 2016–2018 (2017)

Montgomery, D.C.: Introduction to Statistical Quality Control. Wiley, New York (2007)

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

Parziale, L., Liu, W., Matthews, C., Rosselot, N., Davis, C., Forrester, J., Britt, D.T., et al.: TCP/IP Tutorial and Technical Overview. IBM Redbooks, New York (2006)

Pavlidis, N.G., Tasoulis, D.K., Adams, N.M., Hand, D.J.: λ-perceptron: an adaptive classifier for data streams. Pattern Recognit. **44**(1), 78–96 (2011)

Pettitt, A.: A non-parametric approach to the change-point problem. J. R. Stat. Soc. Ser. C Appl. Stat. **28**, 126–135 (1979)

Pinheiro, J.C., Bates, D.M.: Mixed-Effects Models in S and S-PLUS. Springer, Berlin (2000)

Plasse, J., Noble, J., Myers, K.: An adaptive modeling framework for bivariate data streams with applications to change detection in cyber-physical systems. In: IEEE International Conference on Data Mining Workshops, pp. 1074–1081. IEEE, Washington (2017)

Reynolds Jr., M.R., Stoumbos, Z.G.: A CUSUM chart for monitoring a proportion when inspecting continuously. J. Qual. Technol. **31**(1), 87 (1999)

Reynolds Jr., M.R., Stoumbos, Z.G.: A general approach to modeling CUSUM charts for a proportion. IIE Trans. Qual. Reliab. Eng. **32**(6), 515–535 (2000)

Ross, G.J., Tasoulis, D.K., Adams, N.M.: Nonparametric monitoring of data streams for changes in location and scale. Technometrics **53**(4), 379–389 (2011)

Ross, G.J., et al.: Parametric and nonparametric sequential change detection in R: the cpm package. J. Stat. Softw. **66**(3), 1–20 (2015)

Ruder, S.: An overview of gradient descent optimization algorithms. arXiv:1609.04747 (2016)

Ryan, A.G., Wells, L.J., Woodall, W.H.: Methods for monitoring multiple proportions when inspecting continuously. J. Qual. Technol. **43**(3), 237–248 (2011)

Siegmund, D.: Corrected diffusion approximations in certain random walk problems. Adv. Appl. Probab. **11**(4), 701–719 (1979)

Szekely, G.J., Rizzo, M.L.: Hierarchical clustering via joint between-within distances: extending Ward's minimum variance method. J. Classif. **22**(2), 151–183 (2005)

Tartakovsky, A., Nikiforov, I., Basseville, M.: Sequential Analysis: Hypothesis Testing and Changepoint Detection. Chapman and Hall/CRC, Boca Raton (2014)

Tsymbal, A.: The problem of concept drift: definitions and related work. Technical report, Computer Science Department, Trinity College Dublin (2004)

Weiß, C.H.: Continuously monitoring categorical processes. Qual. Technol. Quant. Manag. **9**(2), 171–188 (2012)

Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Mach. Learn. **23**(1), 69–101 (1996)

Wolfe, D.A., Chen, Y.S.: The changepoint problem in a multinomial sequence. Commun. Stat. Simul. Comput. **19**(2), 603–618 (1990)