ADAPTIVE ESTIMATION OF CATEGORICAL DATA STREAMS WITH APPLICATIONS IN CHANGE DETECTION AND DENSITY ESTIMATION

A THESIS PRESENTED FOR THE DEGREE OF
Doctor of Philosophy of Imperial College London
AND THE
Diploma of Imperial College
BY
JOSHUA H. PLASSE

DEPARTMENT OF MATHEMATICS
Imperial College
180 Queen’s Gate, London SW7 2BZ

September 2019
I certify that this thesis, and the research to which it refers, are the product of my own work, and that any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Signed: ________________________________
The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.
ACKNOWLEDGMENTS

First and foremost I would like to thank my supervisor Niall Adams. His patience and guidance throughout the course of my PhD was invaluable.

I am also grateful to Imperial College London, and those who have contributed to making the President’s PhD Scholarship possible. My time in London has been one of the most rewarding experiences of my life, and this PhD would not have been possible without their financial support.

I was fortunate enough to intern at Los Alamos National Laboratory during the Summer of 2017, and would like to express gratitude to my mentor Kary Myers, as well as Aric Hagberg and Scott Backhaus for providing the data that made the internship possible.

The real-world data sets analyzed in this thesis have come from various sources, which all deserve recognition. I would like to thank Andy Thomas for the data used in Chapter 4 and Crossword Cybersecurity for the data used in Chapter 5.

I would also like to thank all of the friends that I have made while at Imperial College London. Especially for the intellectual, and entertaining conversations that we had outside of mathematics, which helped me throughout the ups and downs of the PhD.

Lastly, I would like to thank Caitlin, Kathy, Andre, Briannah, Alannah, Yvette and Lucille for the love and support they have shown me over the years.

Joshua H. Plasse
Adaptive estimation of categorical data streams with applications in change detection and density estimation

Abstract

The need for efficient tools is pressing in the era of big data, particularly in applications that generate data streams — unbounded sequences of observations, which arrive at high-frequency and are subject to unknown changes in their data generating process. These temporal changes are colloquially referred to as drift, and require methodology to dynamically reconfigure parameters whenever changes occur. This is achieved by incorporating forgetting factors into the estimation process, and the contributions of this work develop methodology for two under-researched areas of streaming inference, mainly: multiple changepoint detection for categorical data streams, and streaming density estimation using histograms.

Currently there is a dearth of literature devoted to change detection in categorical data streams, and the existing work typically introduces fixed parameters without providing insight into how to specify them. This is ill-suited to the streaming paradigm, motivating the need for approaches that introduce few parameters, which may be set without requiring prior knowledge of the stream. The first novel contribution of this thesis is a family of multinomial change detection methods (MCDMs), which assumes the observations are independent. These detectors adaptively monitor the category probabilities of a multinomial distribution, where temporal adaptivity is introduced using forgetting factors. A novel adaptive thresholding technique is also developed, which can be computed given a desired false positive rate. The observations are then assumed to satisfy a first-order Markov property, and an adaptive detection and estimation procedure for transition matrices (ADEPT-M) is developed. This detector is based on a moment matching technique and effectively monitors for multiple changepoints in a transition matrix without
making any assumptions on the number of changepoints, nor their magnitudes. The performance of the MCDMs and ADEPT-M is investigated via large simulation and real-application studies, which verifies the usefulness of our approaches.

The final contribution of this thesis are temporally adaptive streaming histograms (TASHs) which are suitable for univariate, continuous-valued, data streams. Existing methods typically construct histograms to have equal width bins, which implicitly assumes knowledge of the range of the random variables, or how the data stream evolves over time. This is an unrealistic assumption, and in this thesis the bin widths are allowed to change over time, and are computed in a data-driven manner. Novel methodology is also developed for merging and splitting existing bins, allowing for the accurate estimation of a data stream’s drifting distribution. Several other quantities that are useful in streaming inference can be adaptively estimated with little increase in computation directly from the TASHs. This includes techniques for streaming quantile estimation, non-parametric change detection, data stream comparison, ROC curves and outlier detection.
## CONTENTS

1 INTRODUCTION
   1.1 Contributions ................................................. 4
   1.2 Structure of the thesis ....................................... 6

2 BACKGROUND .................................................. 8
   2.1 Drifting data streams ......................................... 8
   2.2 Relevant distributions ........................................ 11
      2.2.1 The multinomial distribution ............................. 12
      2.2.2 The beta distribution .................................... 13
   2.3 Density estimation ............................................ 13
      2.3.1 Histograms .................................................. 14
      2.3.2 Kernel density estimates ................................. 16
   2.4 Sequential change detection .................................. 17
      2.4.1 Control charts ............................................. 18
      2.4.2 Commonly used control charts ............................. 20
      2.4.3 Assessing performance .................................... 22
      2.4.4 Burn-in and grace period regions ......................... 24
   2.5 Discussion .................................................... 25

3 ADAPTIVE PARAMETER ESTIMATION ............................. 27
   3.1 Forgetting factors ............................................. 28
   3.2 A temporally aware likelihood ............................... 29
      3.2.1 Adaptive mean for the Gaussian distribution .......... 30
   3.3 Fixed forgetting factors ..................................... 32
6.4.2 Measuring performance . . . . . . . . . . . . . . . . . . . . 126
6.4.3 Simulations in drifting data streams . . . . . . . . . . . . 127
6.5 Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . 131

7 Histogram Applications . . . . . . . . . . . . . . . . . . . . . . 133
7.1 Computing quantiles using TASHs . . . . . . . . . . . . . . . 134
7.1.1 Literature review . . . . . . . . . . . . . . . . . . . . . . 134
7.1.2 Estimating quantiles . . . . . . . . . . . . . . . . . . . . 134
7.2 A non-parametric change detector . . . . . . . . . . . . . . . 138
7.3 Two sample inference . . . . . . . . . . . . . . . . . . . . . . . 143
7.3.1 Using quantiles . . . . . . . . . . . . . . . . . . . . . . . 143
7.3.2 KS type test . . . . . . . . . . . . . . . . . . . . . . . . . 145
7.4 Dynamic ROC curves . . . . . . . . . . . . . . . . . . . . . . . 147
7.4.1 Estimating a ROC curve using TASHs . . . . . . . . . . 148
7.5 Outlier detection . . . . . . . . . . . . . . . . . . . . . . . . . 150
7.5.1 Conformal prediction . . . . . . . . . . . . . . . . . . . . 151
7.5.2 Detecting outliers using TASHs . . . . . . . . . . . . . . 151
7.6 Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 155

8 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 156

References . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 161

Appendix A Derivations and Preliminaries . . . . . . . . . . . . 176
A.1 Chapter 2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 176
A.2 Chapter 3 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 177
A.3 Chapter 4 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 178
A.4 Chapter 5 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 182
A.5 Chapter 6 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 184
Data acquisition technology is constantly being improved, allowing for 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. In this setting data arrives sequentially, and each observation has an associated time-stamp, indicating its time of arrival. Data streams are becoming commonplace, finding application in many diverse areas, such as: social media (Mathioudakis and Koudas, 2010), industrial manufacturing (Montgomery, 2007), genome sequencing (Weiβ, 2012), plastic card transactions (Dal Pozzolo et al., 2015) and sensor networks (Gaber et al., 2005). This creates a demand for efficient, flexible methodology that can perform inference while adhering to the computational demands of the stream. The focal point of this thesis is to develop such methodology, for several under-researched areas in the statistical stream mining setting.

The rapid arrival of observations from data streams provides the opportunity for real-time analysis, which is desirable, and in fact could be demanded in many applications. As an example, if a fraudulent transaction is made on a credit card the bank would like to remedy the situation as soon as possible.
to maintain favorable sentiment with its clients. However, analyzing data streams in real-time poses several challenging problems to the statistics and machine learning communities, making data stream mining fundamentally different from conventional tasks. The difficulties of learning in the presence of a large amount of data are typically characterized by the ‘many Vs’, as highlighted in Bifet et al. (2018), and include:

- **volume**: relating to the immense amount of data generated by streaming applications,
  
- **velocity**: referring to the data being continuously generated at high-frequencies, and
  
- **variability**: corresponding to the fact that such a large amount of data is likely to experience changes over time.

Volume and velocity refer to the vast amount of data continuously generated from a data stream. These two characteristics require methodology to use limited computational and storage requirements (Pavlidis et al., 2011), commonly resulting in single-pass algorithms, which remove data from memory immediately after utilizing it. Additionally, the number of floating point operations required per time-stamp should not be a function of the number of observations processed from the stream, which is an attribute common to many non-streaming statistical methods.

Due to the variability of data streams, an algorithm being single-pass will not be sufficient in effectively modeling a data stream (Aggarwal, 2007). Changes in a data stream’s generating process are colloquially referred to as *drift* (Tsymbal, 2004), and if ignored could be detrimental to the learning process. Being able to accurately capture the evolution of data streams therefore requires models to dynamically reconfigure themselves over time. This is typically achieved by allowing the parameter estimates that govern the model to change over time, i.e., to be *adaptive* and capable of responding to the unknown temporal variation that will occur. Evidently, it is clear that adaptively tuning model parameters without user supervision must be a property of any streaming methodology.
One widely used way of introducing temporal awareness into parameter estimates is to use a sliding window. Sliding windows require a parameter to be specified, which is commonly referred to as the width (or size) of the window, and dictates how many recent observations are used in estimating various quantities of interest. The choice of an appropriate window size is usually overlooked, and with few exceptions (e.g. Bifet and Gavalda (2007)), is typically taken to be constant, or chosen based on advice from a domain expert. This is not suitable for streaming data since fixed values, either subjectively chosen or prescribed by an expert, are unlikely to effectively characterize the evolution of the stream. An expert could reassess their opinion after an elapsed period of time; however, when to adjust the size of the window remains unclear.

A continuous analog to the sliding window approach, and the technique used throughout this thesis, is based on forgetting factors. Forgetting factors are commonly used in adaptive filtering (Haykin, 2008), and have found success in various statistical stream mining settings (Anagnostopoulos et al., 2012; Bodenham and Adams, 2017; Cao et al., 2006). Forgetting factors are a sequence of scalars that continuously down-weight historic data as newer data is processed. Therefore, if drift occurs older data is smoothly discarded, or ‘forgotten’, allowing the estimation to be driven by newer, more informative observations. Forgetting factors are a fundamental component of this thesis, and how they are incorporated into the estimation process is discussed in Chapter 3.

Two difficult problems associated with data streams are changepoint detection and density estimation, and this thesis is devoted to developing novel methodology in both settings. Both settings are discussed at a high-level in this chapter, and issues in the current literature are highlighted. The specific areas that this thesis improves upon are provided in Section 1.1, and relevant literature reviews appear at the beginning of the appropriate chapters.

Changepoint detection is concerned with identifying time-stamps, subsequently called changepoints, where the stream experiences drift. Much existing literature on changepoint detection typically makes assumptions that are ill-suited for streaming data, some of which include: focusing on the de-
tection of a single changepoint, assuming the distributions before and after the change are known, detecting changes using fixed thresholds, and introducing several parameters that are chosen subjectively without providing a practitioner with insight into how to specify them. The last two are particularly concerning, since fixed values for parameters are unlikely to effectively characterize a data stream that experiences drift.

Streaming density estimation maintains an estimate of the distribution that is currently generating the observations from the data stream. These techniques are typically non-parametric, meaning that no assumptions are made about the distribution generating the data. Many existing density estimation methods are not trivially extended to the streaming data setting, since they are computationally expensive, and commonly violate the single-pass assumption.

1.1 Contributions

This thesis provides novel contributions in two under-researched areas of streaming inference:

- multiple changepoint detection for categorical data streams, where the observations assume value in some (known) discrete set, and

- non-parametric density estimation for univariate, continuous-valued data streams using histograms.

The change detection methods developed for categorical data streams rely on sequentially and adaptively estimating category and transition probabilities. Therefore, a natural extension to consider is a technique for binning univariate data streams. Being able to provide a sequential binning of the stream allows for the frameworks developed for categorical data to be extended to the univariate setting, since any datum falling into a bin can be treated as a realization of a particular category. Hence, analysis of categorical data streams is at the heart of this thesis, and connects both areas of streaming inference highlighted above, which this thesis improves upon.
In the change detection literature, existing work has focused on detecting changes in continuous-valued data streams, and streams comprised of categorical observations have, for the most part, been ignored. Further, as discussed previously, many change detectors make restrictive, and unrealistic assumptions. The first contributions of this work develop categorical change detectors, which do not fall victim to the many pitfalls that are commonly found in the literature. Observations are assumed to arrive independently, or satisfy a first-order Markov property. In the former, a family of multinomial change detection methods are proposed, which extends Plasse and Adams (2019), and detects changes based on a novel adaptive thresholding technique. In the latter, an adaptive estimate for a Markov transition matrix is developed, and a moment matching technique is used to detect changes in the matrix. These contributions appear to be among the first in developing change detection strategies for categorical data streams.

The second part of this thesis is devoted to streaming density estimation, where a novel contribution is a method for constructing, and maintaining temporally adaptive streaming histograms for univariate data. In this regard, techniques for adjusting the boundary of the histogram, merging and splitting histogram bins, and reallocating the probability mass when merges or splits occur are developed. Existing literature typically constructs histograms whose bins have the same width; however, this implicitly assumes knowledge of the range of the random variables, or how the data stream experiences drift. To remedy this, the bin widths are allowed to be variable, and are computed in a data-driven manner. These histograms then allow for several other quantities of interest to be adaptively estimated from the histogram, with little increase in computation. These include novel techniques for streaming quantile estimation, non-parametric change detection, data stream comparison, receiver operating characteristic (ROC) curves and outlier detection.

Plasse and Adams (2019) is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/). Other contributions include the works Plasse and Adams (2016) and Plasse et al. (2017), both of which utilized techniques
discussed in this thesis. However, due to space limitations the main contributions of these works have been omitted.

1.2 Structure of the thesis

This thesis proceeds as follows.

- Chapter 2 introduces drifting data streams, relevant probability distributions, density estimation, and establishes notation used throughout this work. Sequential change detection procedures are also discussed, and are used as comparison methods in later chapters. A way of resetting a detector is also considered, which is necessary when monitoring a data stream that contains multiple changepoints.

- Chapter 3 introduces forgetting factors, a critical component of this thesis, which allows for temporally adaptive estimates to be constructed. This is done by optimizing a modified likelihood function, which is more suitable for drifting data streams. A method for tuning the forgetting factors online is also provided.

- Chapter 4 presents the first novel contribution of this thesis – a family of multinomial change detection methods. This chapter is based on Plasse and Adams (2019), and develops methods that utilize the behavior between adaptive and non-adaptive parameter estimates for the multinomial distribution, and detects changepoints based on an innovative adaptive thresholding technique. In addition to synthetic experiments, nearly 40 million network events collected from an Imperial College London router are monitored for changes.

- Chapter 5 assumes that the categorical observations processed from the data stream satisfy a first-order Markov property, and a change detector for transition matrices is developed. This requires an adaptive estimate for a transition matrix to be developed, which is an extension of the work in Chapter 4, and changes are detected using a moment
matching technique. Synthetic simulations are conducted, and over 9 million HTTP web requests are monitored for changes.

- Chapter 6 considers univariate, continuous-valued data streams where both the functional form of the distribution and/or its parameters are allowed to evolve over time. A novel technique for maintaining an adaptive and sequential histogram is then developed, which allows for complete non-parametric estimation of a drifting stream’s distribution.

- Chapter 7 presents several applications that are direct consequences of the histogram approach developed in Chapter 6.
2

BACKGROUND

This chapter provides a summary of prerequisites required for the thesis. Section 2.1 discusses drifting data streams, establishes notation, and introduces changepoints. Reacting to drift is done by using temporally adaptive parameter estimates, and how to construct such estimates is deferred to Chapter 3. Section 2.2 reviews relevant distributions, and Section 2.3 discusses density estimation techniques. An overview of sequential change detection is provided in Section 2.4, where performance measures, and a way of restarting a detector is addressed. Material discussed here will be essential to the novel contributions presented in Chapters 4-7.

2.1 Drifting data streams

It is assumed that the data streams of interest take the form

$$\langle x_1, x_2, \ldots, x_{t-1}, x_t, \ldots \rangle,$$

(2.1)

where the discrete time-stamp $t \in \mathbb{Z}^+$ denotes the arrival time of the observation $x_t$. The interarrival times between observations is assumed constant,
and the actual value of the time-stamp is ignored from a modeling perspective. The support of \( x_t \) is different throughout the thesis. In Chapters 4 and 5 the data streams are categorical, and observations are assumed to arrive independently, or satisfy a first-order Markov property. The support of the observations is therefore given by \( S = \{1, 2, \ldots, K\} \), for a fixed, known, positive integer \( K > 1 \). Continuous-valued univariate data streams are considered in Chapter 6, where the support of \( x_t \) may be any subset of \( \mathbb{R} \).

The observed value \( x_t \) is a realization of the random variable \( X_t \), which is assumed to follow the distribution

\[
X_t \sim \begin{cases} 
F^{(1)}(\theta^{(1)}) & 1 \leq t \leq \tau_1, \\
F^{(2)}(\theta^{(2)}) & \tau_1 < t \leq \tau_2, \\
& \vdots \\
F^{(m)}(\theta^{(m)}) & \tau_{m-1} < t \leq \tau_m, \\
F^{(m+1)}(\theta^{(m+1)}) & \tau_m < t < \infty,
\end{cases}
\]

where for each \( i \in \{1, 2, \ldots, m + 1\} \), \( F^{(i)} \) is a distribution function governed by the parameter vector \( \theta^{(i)} \). In general, each \( F^{(i)} \) is allowed to have a different parametric form, where the role of the parameter vector \( \theta^{(i)} \) would change accordingly.

A schematic representation of the data generating process is displayed in Figure 2.1. The observations from the stream are generated in segments, and in each segment the observations are generated from a distribution whose parameters remain fixed. The set \( \tau = \{\tau_k\}_{k=1}^m \) contains a sequence of unknown temporal locations where the data stream experiences a change, and any \( \tau_k \in \tau \) is referred to as a changepoint. In the data stream setting there could exist an infinite number of changepoints, and we remark that the techniques developed in subsequent chapters remain applicable when \( m = \infty \). Focal points of this thesis include: developing methodology for accurately detecting the multiple changepoints in \( \tau \), as well as maintaining accurate estimates of the distributions, and their parameters, in each segment.

Changes in the data generating process are colloquially referred to as drift,
Figure 2.1: A schematic representation of the assumed data generating process. The set of changepoints $\{\tau_k\}_{k=1}^m$ define the segments, and within the $k^{\text{th}}$ segment the stream is governed by a fixed distribution $F^{(k)}$ with parameter vector $\theta^{(k)}$ (omitted from the figure).

and can manifest in a data stream in many ways. This makes the detection of multiple changepoints a challenging task. Many common types of drift have been identified, although the naming is not consistent throughout the literature (Bifet et al., 2018). Some types of drift include:

- **abrupt** drift, where the data stream experiences a sudden change in its generating process,

- **gradual** drift, where the data stream experiences small changes over a significant period of time, resulting in a noticeable change in the generating process, and

- **seasonal** drift, where distributions that previously generated data reappear after an elapsed period of time.

Figure 2.2 depicts these types of drift in synthetic data streams – refer to Bifet et al. (2018), Tsymbal (2004) and Gama et al. (2014) for more on drift in data streams.

The distributions and parameters appearing in Equation (2.2) implicitly depend on the time-stamp $t$, since the segments they are defined over depend on the temporal locations of the changepoints. Furthermore, there is no restriction placed on the length of any segment. Due to this, Equation (2.2) encapsulates the many kinds of drift that may occur in practice. For example, suppose the distributions in Equation (2.2) have the same parametric form. Gradual drift can then be expressed by letting each segment have length one, and allowing the parameter vector to gradually change its value over time. This could be made explicit by including a time index on the
distributions and parameter vectors in Equation (2.2); however, this thesis is mainly concerned with distributions that remain static over longer segments. Consequently, the form of Equation (2.2) is sufficient for our purposes.

As in the previous example, it is common to assume the family of distributions is fixed (e.g. Gaussian), and the main concern is detecting changes in the parameter vector. This is the case in Chapters 4-5, where the data streams are assumed categorical. In Chapter 6, arbitrary univariate data streams are investigated, and drift is allowed to occur in both the distribution and its parameters.

2.2 Relevant distributions

This section reviews probability distributions that are used throughout the thesis. The multinomial distribution is reviewed first, and drives a lot of the estimation procedures in later chapters. The beta distribution is then discussed, which is used in constructing a change detector for transition matrices in Chapter 5.
2.2.1 The multinomial distribution

Consider the ‘classical setting’, where a fixed set of random variables \( \{X_k\}_{k=1}^t \) can be stored in computer memory, and are governed by a fixed probability distribution. Denote realizations from this set of random variables by \( x_{1:t} = (x_1, \ldots, x_t)^\top \).

The **multinomial distribution** generalizes the binomial distribution, and is appropriate when a trial of an experiment can result in \( K > 2 \) discrete, independent outcomes. A random vector \( C = (C^{(1)}, C^{(2)}, \ldots, C^{(K)}) \) is said to follow a multinomial distribution with parameters \( t \) and \( p \), subsequently denoted by \( C \sim \text{Multi}(p, K, t) \), if its joint probability mass function is given by

\[
\mathbb{P} \left( C^{(1)} = c^{(1)}, \ldots, C^{(K)} = c^{(K)} \mid p, t \right) = \frac{t!}{\prod_{i \in S} (c^{(i)})!} \prod_{i \in S} (p^{(i)})^{c^{(i)}},
\]

where \( S = \{1, 2, \ldots, K\} \) is the set of possible outcomes, called categories, and the \( i \)th components of \( c \) and \( p \) are denoted by \( c^{(i)} \) and \( p^{(i)} \). The random vector \( C \) represents the vector of category counts, that is, for each \( i \in S \),

\[
C^{(i)} = \sum_{k=1}^t 1(X_k = i),
\]

where \( X_k \in S \) denotes a possible outcome of the trial, and \( 1(\cdot) \) is the indicator function. The parameter vector \( p \) contains the category probabilities

\[
p^{(i)} = \mathbb{P}(X_k = i), \quad \forall \, k \in \{1, \ldots, t\}.
\]

From Equations (2.3)-(2.4) it is clear that, for any \( t \in \mathbb{Z}^+ \), the following are satisfied

\[
\sum_{i \in S} C^{(i)} = t, \quad \sum_{i \in S} p^{(i)} = 1.
\]

In Chapter 4, the maximum likelihood estimate (MLE) for \( p \) will be useful when constructing a family of change detectors for categorical data streams. These are provided in Lemma 1, which is proved in Appendix A.1.

**Lemma 1.** Suppose that \( C \sim \text{Multi}(p, K, t) \), and that \( x_{1:t} \) have been ob-
served. Then the MLE for $\mathbf{p}$ is given by $\hat{\mathbf{p}} = \hat{\mathbf{c}}/t$, where $\hat{\mathbf{c}}$ is the vector of observed category counts.

Subsequently, the standard MLE will be referred to as the static MLE, since every datum contributes equally to the estimate. Furthermore, throughout this thesis any static estimate will always be decorated with the ‘∧’ symbol.

2.2.2 The beta distribution

A random variable $X \in (0, 1)$ is said to follow a beta distribution, with shape parameters $a, b > 0$, if the probability density function of $X$ is given by

$$f_X(x \mid a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1},$$

where $\Gamma(\cdot)$ is the gamma function. Refer to Gupta and Nadarajah (2004) for an extensive survey of the beta distribution.

In Chapter 5 a change detector is developed for transition matrices, which moment matches with the mean and variance of the beta distribution. These are provided in Lemma 2, and are derived in Appendix A.1.

**Lemma 2.** Suppose that $X \sim \text{Beta}(a, b)$. Then

$$E(X) = \frac{a}{a + b}, \quad \text{Var}(X) = \frac{ab}{(a + b)^2(a + b + 1)}.$$

2.3 Density estimation

In Chapter 6 a technique is developed for constructing temporally adaptive streaming histograms, which provide density estimates for arbitrary univariate data streams. Therefore, a brief overview of offline density estimation is beneficial, and provided in this section. This includes estimation using histograms and kernel density estimates, and the section also highlights challenges involved with implementing each method. Throughout the section it
is assumed that $x_{1:t}$ are univariate, independent and identically distributed, with probability density function $f(x)$.

2.3.1 Histograms

Histograms are a non-parametric density estimation technique, whose usage can be dated back to the late 1800s (Pearson, 1895). See Ioannidis (2003) for a summary of the history related to histograms. In this section we discuss offline histograms suitable for univariate data, as well as techniques for constructing them. Streaming histograms are deferred to Chapter 6.

A histogram is defined by a set of non-overlapping bins, which partition the range of $x_{1:t}$. Each bin is comprised of endpoints, and a frequency count that keeps track of how many data points reside in the bin. Suppose a histogram contains $N$ bins. The histogram can then be completely summarized by the set

$$\mathcal{H} = \left\{ (a^{(i)}, b^{(i)}, f^{(i)}) \right\}_{i=1}^N,$$

where $a^{(i)} < b^{(i)}$ are the endpoints of the $i^{th}$ bin, and

$$f^{(i)} = \sum_{k=1}^t 1 \left( x_k \in [a^{(i)}, b^{(i)}) \right)$$

is its frequency count. In this thesis we follow convention, and assume the bins are closed at the left endpoint, and open at the right endpoint. The endpoints satisfy $a^{(i+1)} = b^{(i)}$ for $i \in \{1, \ldots, N-1\}$, cover the range of $x_{1:t}$, and the histogram estimates $f(x)$ by a constant in each bin, which is proportional to the frequency count.

To construct a histogram requires the number of bins to be specified, as well as the position of the bins endpoints. Some well-known approaches are highlighted below; refer to Poosala et al. (1996) for a summary of existing techniques.

- **Equiwidth histograms** partition the range of the data into $N$ bins that all have the same width.
• Equidepth histograms choose endpoints so that each bin has (approximately) the same frequency count.

• V-Optimal histograms (Ioannidis and Poosala, 1995; Poosala et al., 1996) minimize a weighted sum of the bin variances, i.e., the squared difference between the observations in the bin and the estimate from the histogram.

All of these approaches remove the burden of having to subjectively specify the location of the endpoints by introducing some constraint (such as the bins being equiwidth/depth). These approaches do not trivially extend to the data stream setting. For example, equiwidth histograms require the range of the data to be known. However, the range of a data stream will not be known in general, and is likely to change over time. Further, techniques that require optimization typically have computational costs that are too high for streaming data. The V-Optimal histogram is an example of this, which uses dynamic programming to minimize the sum of bin variances, and exhibits quadratic computational costs (Ioannidis, 2003).

CHOOSING THE NUMBER OF BINS

Commonly used techniques choose bin boundaries to satisfy some constraint; however, this does not alleviate the issue of having to choose $N$. Perhaps the most popular way of choosing $N$ is by using Sturges’ rule (Sturges, 1926), which chooses the number of bins as

$$N = \lceil \log_2 t \rceil + 1. \quad (2.5)$$

Sturges’ rule assumes the frequencies of the bins can be expressed as ‘ideal’ binomial coefficients, which implicitly places a Gaussian assumption on the data. Consequently, Sturges’ rule can result in poor histograms if the underlying density is highly non-Gaussian, or is severely skewed. Doane (1976) improved on Sturges’ rule by adding a term to Equation (2.5), which takes into account the (estimated) skewness of the distribution.
Scott (1979) proposed a normal reference rule for equiwidth histograms, which share a common bin width \( w \) defined by
\[
w = \frac{3.5 \sigma}{t^{1/3}},
\]  
(2.6)

where \( \sigma \) is the sample standard deviation estimated from \( x_{1:t} \). Scott’s rule is optimal for Gaussian data in the sense of mean integrated square error. Freedman and Diaconis (1981) sought to make Scott’s rule more robust to non-Gaussian data by replacing the numerator in Equation (2.6) with a multiple of the sample interquartile range, which is less sensitive to outliers.

In the streaming paradigm the amount of data being collected is unbounded. This makes many of the heuristics for choosing the number of bins ill-suited for streaming data. That is, when \( t \to \infty \) the suggested number of bins increases without bound, and the bin widths approach zero. Thus, there is a need for methodology that allows for histograms to be sequentially maintained, without requiring a large number of bins when a vast amount of data is available. Chapter 6 develops such a method.

### 2.3.2 Kernel density estimates

A continuous analog to the histogram is the **kernel density estimate** (KDE). The classical KDE for \( f(x) \) is given by
\[
\hat{f}_{\text{KDE}}(x \mid w, K) = \frac{1}{tw} \sum_{k=1}^{t} K \left( \frac{x - x_k}{w} \right),
\]

where \( K(\cdot) \) is a kernel function, and \( w \) is a bandwidth parameter, which is similar to the width of a histogram bin. At a high-level, a KDE places a ‘bump’ around each datum, and normalizes over all the bumps. The shape of the bumps depend on the choice of kernel, and their widths depend on the bandwidth parameter (Ross, 2013). To compute a KDE requires a kernel and bandwidth to be selected, and it is generally agreed upon that the choice of kernel does not greatly affect the KDE (Silverman, 1986).

Choosing the bandwidth parameter \( w \), similar to specifying the number of
bins in a histogram, is a more pressing concern. Choosing too large a bandwidth will produce an over-smoothed KDE, which may fail to capture local structure in the density; whereas, a small bandwidth will result in a KDE with many sharp peaks, which will overfit the density (Webb, 2003). Popular approaches for choosing \( w \) rely on optimizing an asymptotic mean integrated square error between the KDE and the true density. An optimal bandwidth can then be computed, and expressed as a function of \( K(\cdot) \) and \( f''(x) \) (which is unknown). Another KDE is commonly used to estimate the second derivative, and requires another bandwidth to be specified. Refer to Jones et al. (1996) for a survey of approaches for choosing the bandwidth.

Extending kernel density estimation to the data stream setting is non-trivial. This is mainly because techniques are too computationally intensive for streaming data. Additionally, offline techniques are not equipped with the ability to react to drift. Notwithstanding, online kernel density estimation methods have been considered recently. Ross (2013) develops an online KDE that can be updated by appealing to recursive updates derived from stochastic approximation theory. This allows for the KDE to be evaluated at a single data point, and is extended to several data points by repeatedly applying the same update equations, and linearly interpolating between the density estimates. Ahmed (2009) uses an online KDE to perform anomaly detection in multivariate networks; whereas, Qahtan et al. (2017) explore online kernel density estimation for spatio-temporal data. Refer to Qahtan et al. (2017) for a summary of existing techniques, which aim to lighten the computational burden of constructing a KDE. Moving forward, we favor histograms due to their simplicity, and kernel density estimation is not considered subsequently.

2.4 Sequential change detection

This section reviews sequential changepoint detectors commonly used in the literature. These methods will be revisited in later chapters, when our methodology is compared to existing methods. Performance measures used to assess online change detectors are also introduced, and used throughout this thesis. The section concludes with a discussion on how a change detector
can be initialized, and reconfigured once it detects a changepoint – something necessary when monitoring a data stream for multiple changepoints.

2.4.1 Control charts

A commonly used family of change detectors are control charts, first introduced by Shewhart (1931). Control charts are frequently used in the statistical quality control literature (Montgomery, 2007), where the charts are used to detect a change in a manufacturing process.

When an observation arrives control charts compute a statistic, and compare its value to constructed control limits. Whenever the statistic violates these limits a change in the stream is said to have occurred. Formally, suppose $s_t$ is the statistic monitored, and let $\ell < u$ be the control limits. Then the stream can exist in one of two states:

- the stream is in-control whenever $s_t \in [\ell, u]$, and
- the stream is out-of-control whenever $s_t \not\in [\ell, u]$.

The control chart developed by Shewhart (1931), referred to as the Shewhart chart, has led to an immense amount of literature in the area, and provides a constructive example on how a control chart can be used in practice. Suppose a data stream experiences a change at time $\tau$. The Shewhart chart assumes that the in-control mean and variance of the process are known, i.e.,

$$\mathbb{E}(X_t) = \mu, \quad \text{Var}(X_t) = \sigma^2 \quad \forall \ t < \tau$$

are available. The control limits are chosen to be some number of standard deviations away from $\mu$:

$$\ell = \mu - \zeta \sigma, \quad u = \mu + \zeta \sigma,$$

where $\zeta$ is a control parameter to be specified. The statistic monitored in the Shewhart chart is the observation itself, so that a change is flagged whenever
Figure 2.3: An example of a Shewhart control chart with $\zeta = 3$. The cross ($\times$) indicates the first time the chart is out-of-control, and occurs at $t = 251$, i.e., the chart was able to immediately detect the change.

$x_t \notin [\ell, u]$. An example of a Shewhart chart is shown in Figure 2.3, where

$$X_t \sim \begin{cases} 
\mathcal{N}(0, 1) & t \leq 250, \\
\mathcal{N}(4, 1) & t > 250,
\end{cases}$$

$\zeta = 3$, and $\mathcal{N}(\mu, \sigma^2)$ denotes the Gaussian distribution with mean $\mu$ and variance $\sigma^2$. In this contrived example the chart is able to correctly detect the change as soon as it occurs.

Control charts are typically constructed to detect a single changepoint. Indeed, in the statistical quality control setting once a change is detected the process can be stopped until a remedy is found. This is not possible in the streaming paradigm, since observations continuously arrive from the stream. Therefore, a way of reinitializing the chart once a change has been detected is required. One such technique is addressed in Section 2.4.4, and is used throughout the thesis.

Another issue with control charts is that they introduce control parameters that are difficult to set in practice, and typically rely on assumptions that are commonly not satisfied in streaming data. For example, the Shewhart chart required the parameter $\zeta$ to be chosen, denoting how many standard deviations away from the mean an observation needs to be before a changepoint is detected (and $\mu$ and $\sigma^2$ to be known). Therefore, $\zeta$ directly affects
the chart’s ability to detect changepoints, and in the multiple changepoint setting is unlikely to be effective in detecting the various types of drift that may occur. A central aim of this thesis is to develop change point methodology for streaming data, without introducing several control parameters that are difficult to set.

2.4.2 Commonly used control charts

Two commonly used control charts are now introduced. These were developed to detect a single change in the mean of observations generated from a stochastic process, and are used as comparison methods in later chapters. It is assumed that the observations are independent, having a common in-control mean and variance given by $\mu$, and $\sigma^2$, respectively.

CUSUM

The **CUSUM** (CUSUM) chart was introduced in Page (1954), and monitors the statistics given by

$$
C^+_t = \max\{0, C^+_t - 1 + z_t - k\},
C^-_t = \max\{0, C^-_t - 1 - z_t - k\},
$$

where $z_t = (x_t - \mu)/\sigma$ are the standardized observations from the stream and $k$ is a control parameter referred to as the reference value (Montgomery, 2007). The CUSUM control chart signals a change whenever $C^+_t$ or $C^-_t$ exceed some threshold $h$. The scalar $h$ is referred to as the decision interval (Montgomery, 2007) and is another control parameter to be set by the user.

The CUSUM chart was shown to be optimal under certain assumptions (Moustakides et al., 1986), which include the distribution before and after the changepoint being known (including their parameters). This is unlikely to be known in practice, in which case optimality is not guaranteed (Bodenham, 2014).

The tuple $(k, h)$ is difficult to choose in practice, and misspecifying its value
will affect the chart’s performance (Ross, 2013). The scalar $k$ is commonly chosen to detect a change of a specific magnitude – which implicitly assumes prior knowledge of the magnitude of the changepoint is available. The value of $h$ is typically chosen to be proportional to the in-control standard deviation $\sigma$. This assumes $\sigma$ is known, or a reliable estimate is available. These approaches for choosing the control parameters will be hindered by the presence of multiple changepoints, and restarting the algorithm once a change has been detected is certainly required. In this thesis a subset of recommendations made in Hawkins (1993) for choosing the control parameter tuple are used, and are displayed in Table 2.1. It is not obvious which pair of control parameters (which are determined heuristically) will perform well in a particular setting, and using multiple choices for $(k, h)$ is sensible, and sufficient for this thesis.

**EWMA**

The Exponentially Weighted Moving Average (EWMA) control chart was proposed in Roberts (1959), and has been investigated in Lucas and Saccucci (1990) and Ross et al. (2012). EWMA monitors the statistic $Z_t$ sequentially defined by the convex combination

$$Z_t = (1 - r)Z_{t-1} + rx_t,$$

for a user defined control parameter $r \in [0, 1]$. The parameter $r$ has a similar interpretation to a fixed forgetting factor (as discussed in Chapter 3), and affects how quickly older observations are discarded. In Bodenham (2014) connections between EWMA and the fixed forgetting case are highlighted. A change is flagged whenever

$$Z_t \notin [\mu - L\sigma_{Z_t}, \mu + L\sigma_{Z_t}],$$

for a user defined control limit $L$, and

$$\sigma_{Z_t}^2 = \sigma^2 \left\{ \frac{r}{2 - r} \left[ 1 - (1 - r)^2 \right] \right\}.$$
CUSUM \((k_1, h_1)\) \((k_2, h_2)\) \((k_3, h_3)\)

\((k, h)\) values: \((0.250, 8.010)\) \((0.500, 4.770)\) \((0.750, 3.340)\)

EWMA \((r_1, L_1)\) \((r_2, L_2)\) \((r_3, L_3)\)

\((r, L)\) values: \((1.000, 3.090)\) \((0.250, 2.998)\) \((0.03, 2.437)\)

<table>
<thead>
<tr>
<th>Table 2.1:</th>
<th>Control parameters used when implementing CUSUM and EWMA.</th>
</tr>
</thead>
</table>

The control parameters \(r\) and \(L\), as in the case of the CUSUM chart, are difficult to set in the multiple changepoint setting. Consequently, multiple choices for the parameter values, as suggested in Lucas and Saccucci (1990), are used in simulation studies of Chapter 5. These values are provided in Table 2.1.

### 2.4.3 Assessing Performance

Four performance measures, which are commonly used to assess online change detection performance, are now introduced. These performance measures will appear throughout the thesis, and provide insight into a detector’s ability to detect changepoints.

#### The Average Run Lengths

Two commonly used performance measures are the Average Run Lengths (ARLs) \(\text{ARL}_0\) and \(\text{ARL}_1\) introduced in Page (1954). \(\text{ARL}_0\) is defined as the average number of observations that arrive until a false positive is detected, whereas \(\text{ARL}_1\) is defined as the average detection delay. The ARLs are difficult to compute exactly, despite the amount of research devoted to the topic, e.g., see Lucas and Saccucci (1990), Brook and Evans (1972), Crowder (1987) and Hawkins (1992). Due to this, Monte Carlo simulations are usually conducted to approximate a detector’s \(\text{ARL}_0\) and \(\text{ARL}_1\).

Typically \(\text{ARL}_0\) is estimated by implementing a change detector over multiple streams with no changepoints present, and averaging over the first false positive flagged in each stream. To compute \(\text{ARL}_1\), a change detector is run
over multiple streams with one changepoint, and the detection delays are averaged over. Detectors exhibiting large values of ARL₀ and low values of ARL₁ are preferred, although both need to be inspected when considering the performance of a detector, since either of the ARLs can be improved at the expense of the other.

In the multiple changepoint setting the ARLs are not sufficient to assess performance. To see this, consider the case where two changepoints exist at times \( \tau_1 < \tau_2 \), and that a detector flagged a change at \( \hat{\tau} = \frac{\tau_1 + \tau_2}{2} \). It is unclear whether \( \hat{\tau} \) is a false positive for \( \tau_2 \), or a legitimate detection for \( \tau_1 \). Depending on one’s interpretation, \( \hat{\tau} \) could be used in the computation of either ARL₀ or ARL₁. This raises issues when trying to measure the performance of a detector. The ARLs also do not take into account how many changepoints a detector flags, nor the number of true changepoints a method did not detect. Because of this, as in Bodenham and Adams (2017), two other performance measures are considered.

**CCD and DNF**

When there is more than one changepoint in a stream the ARLs are not complete measures of performance. Due to this, two new measures are introduced that represent the proportion of Changepoints Correctly Detected (CCD), and the proportion of Detections that are Not False (DNF).

Suppose that a data stream contains \( m \) changepoints, and a change detector makes \( \hat{m} \) detections, consisting of \( T \) true detections, and \( \hat{m} - T \) false detections. Then CCD and DNF are defined by

\[
\text{CCD} = \frac{T}{m},
\]

\[
\text{DNF} = \frac{T}{\hat{m}},
\]

where CCD and DNF take values in \([0, 1]\), with values closer to one being preferable. Indeed, a CCD and DNF both equaling one indicates that a detector correctly identified all changepoints, without raising any false alarms. Like the ARLs, CCD and DNF are estimated via Monte Carlo simulations.
In subsequent chapters we omit reporting standard errors for ARL$_0$, ARL$_1$, CCD and DNF. This is because standard errors for ARL$_0$ are notoriously high, since this run length can be interpreted as a geometric random variable (Montgomery, 2007). Consequently, the standard errors are not informative, and to avoid cluttering the presentation of the results, as well as remaining consistent, we do not report them. Refer to Bodenham (2014) for standard error estimates of the performance measures in similar scenarios.

2.4.4 Burn-in and grace period regions

To be able to accurately detect changepoints requires detectors to maintain ‘up-to-date’ estimates for parameters that are being monitored. Furthermore, before a data stream is analyzed initial parameter estimates need to be obtained, and sequentially maintained online without interrupting the continuous flow of data. This is similar to the Phase I and Phase II settings discussed in the statistical quality control literature (Montgomery, 2007).

Before analyzing a data stream a burn-in period is used to get initial parameter estimates used by a detector. This requires a parameter $B \in \mathbb{Z}^+$ to be specified, which represents the number of observations used to estimate a detector’s parameters. For example, if the in-control mean and variance for the Shewhart chart were unknown, the sample mean and variance of $B$ observations could be used to initialize the chart. Choosing a value for $B$ is not a major concern, provided that the data used is representative of the stream when it is in-control. Refer to Bodenham and Adams (2017) and Jensen et al. (2006) for a discussion on estimating parameters using a burn-in.

Once a detector has been initialized via a burn-in period it can begin monitoring a stream for changepoints. Whenever a changepoint is flagged, the detector enters a grace period, and no subsequent changepoints are monitored for during this period. This is a simple restarting mechanism, and is required to estimate the parameters of a new segment prior to a further round of monitoring. The grace period requires a parameter $G \in \mathbb{Z}^+$ to be chosen, which dictates when the grace period terminates. In the simplest setting, if a detector flagged a change at time $\hat{\tau}$, then the grace period ends.
at time ($\hat{\tau} + G$). The scalar $G$ may be considered as a control parameter; however, any methodology that is capable of detecting multiple changepoints will require some sort of restarting procedure. We prefer the grace period in this thesis, as it is simple to interpret. Furthermore, the value of $G$ implicitly defines a lower bound on ARL$_0$, since after detecting a changepoint, $G$ observations need to be processed before another changepoint can be detected.

Figure 2.4 displays the estimation process for sequential change detectors, which is repeated each time a detector flags a change. Unless otherwise specified, it is assumed that a burn-in period is used (offline) to initialize the detectors, and grace periods are used to reinitialize a detector after each detected changepoint.

2.5 Discussion

This chapter presented many of the prerequisites required for this thesis. This included a discussion on drifting data streams, which established notation, discussed many kinds of drift that can occur, and introduced the idea of multiple changepoints. An overview of the multinomial and beta distributions was then provided, as these distributions play important roles in subsequent chapters.

An overview of density estimation techniques was then provided, with focus on histograms and kernel density estimates. A main contribution of this thesis is creating a histogram suitable for data streams, which effectively estimates a drifting data stream’s distribution. The chapter concluded with a discussion on sequential change detectors, and introduced control charts that
are commonly used. Performance measures used to assess online changepoint
detectors were also introduced, and a way of restarting a detector so that it
can be used to detect multiple changepoints was mentioned.

In monitoring drifting data streams, it is not sufficient to develop change
detectors, or density estimation techniques, that are able to keep up with
the high volume of data arriving. Due to the temporal variation inherent
in data streams, methodology must also be able to autonomously adapt to
the drift that occurs. In the next section we discuss how to create tempo-
rally adaptive estimates, adopting a technique from the adaptive filtering
literature, that allows for estimates to effectively discard data from previous
segments, without affecting its estimation ability during periods where the
data is being generated by a fixed distribution.
Adaptive Parameter Estimation

It is naive to assume that unbounded data streams will not experience drift. Therefore, to be able to effectively monitor data streams requires methods to dynamically adjust how much older data contributes to current parameter estimates. If not, parameter accuracy will suffer, since the estimation will equally incorporate data generated from different processes. In this chapter, temporal adaptivity is integrated into the estimation procedure using forgetting factors – a commonly used technique in adaptive filtering (Haykin, 2008). Forgetting factors are a fundamental component of this thesis, and are used in all subsequent chapters.

The chapter proceeds as follows: forgetting factors are introduced at a high-level in Section 3.1. They are then incorporated into the estimation process, via an MLE approach, in Section 3.2. The case of fixed forgetting is discussed in Section 3.3, which reveals how the forgetting factors interact with parameter estimates. A technique for tuning the forgetting factors online is explored in Section 3.4, culminating in a way of autonomously tuning the forgetting factors without requiring any user intervention.
3.1 FORGETTING FACTORS

Forgetting factors are a sequence of scalars, taking value in \([0, 1]\), which dictate the rate in which historic data is discarded, or ‘forgotten’. Forgetting factors provide a continuous analog to many ‘hard forgetting’ approaches, such as sliding windows, where each datum either contributes equally, or not at all (Anagnostopoulos, 2010).

In this thesis, forgetting factors will be denoted either by \(\lambda\) or \(\lambda_t\). The former is referred to as a fixed forgetting factor, as its value is time independent, and is less general than the latter approach, which allows the forgetting factors to change their value over time. These are referred to as adaptive forgetting factors, and are preferred in the streaming paradigm since a fixed forgetting factor is unlikely to accurately describe a data stream indefinitely. Section 3.3 discusses fixed forgetting factors in more detail, since it is easier to interpret how they interact with parameter estimates; whereas, adaptive forgetting factors are the topic of Section 3.4.

Temporal variation in data streams has led to forgetting factors being a popular approach in the literature, spanning a range of common learning tasks. Some conventional stream mining problems, where forgetting factors have found success, include:

- **changepoint detection** (Bodenham and Adams, 2017; Plasse and Adams, 2019)
- **classification** (Anagnostopoulos et al., 2012; Pavlidis et al., 2011; Plasse and Adams, 2016),
- **clustering** (Cao et al., 2006; Filev et al., 2010),
- **density estimation** (Sebastião et al., 2014b)
- **frequent pattern mining** (Asai et al., 2002; Chen et al., 2012) and
- **model validation** (Gama et al., 2009).
Perhaps the most obvious alternative to forgetting factors are sliding windows. However, as discussed in Chapter 1, specifying the size of the window is non-trivial. As highlighted in Anagnostopoulos (2010), techniques from state-space modeling can be interpreted as an alternative approach to forgetting factors. In this scenario, a model could be placed on how a stream’s parameters drift over time. The simplest model assumes observations are Gaussian with a known variance, and whose mean follows a random walk (also with a known variance). In this scenario the optimal estimate for the mean is computed by the well-known Kalman filter equations (Kalman, 1960); however, in the streaming paradigm parameter values are unlikely to be known. Further, it is unlikely that a single state-space model will accurately characterize parameter drift indefinitely. Due to this, state-space models are not considered in the sequel.

In the next section, the mathematical details behind using forgetting factors in estimation is presented. This is done via an MLE approach, which develops a likelihood function that is more suitable for temporally evolving data streams. This likelihood can then be optimized, resulting in closed form solutions for adaptive maximum likelihood estimates.

3.2 A Temporally Aware Likelihood

Suppose that $x_{1:t}$ have been observed, and are realizations of the random variables $X_{1:t} \sim F(\theta_t)$. For the time being, only the parameter vector $\theta_t$ is allowed to evolve over time, and the functional form of $F(\cdot)$ is assumed known, and fixed. In Chapters 4-5, attention is restricted to categorical data streams; therefore, $F(\cdot)$ is known. In Chapter 6, continuous univariate streams are considered, and the distribution as well as its parameters are both allowed to drift.

Throughout this thesis, similar to Anagnostopoulos et al. (2012), adaptive parameter estimates are chosen to optimize the weighted log-likelihood func-
tion defined by

\[ \mathcal{L}_{FF}(\theta \mid x_{1:t}) = \sum_{k=1}^{t} \left[ \prod_{t=k}^{t-1} \lambda_t \right] \mathcal{L}(\theta \mid x_k) \]

(3.1)

\[ = \lambda_{t-1} \mathcal{L}_{FF}(\theta \mid x_{1:t-1}) + \mathcal{L}(\theta \mid x_t), \]

(3.2)

where \( \mathcal{L}(\cdot \mid \cdot) \) is the log-likelihood function of the data, and the time subscript has been omitted from the parameter vector \( \theta \) to reinforce that temporal adaptivity of the estimates is due to the forgetting factors. When \( k = t \), the empty product in Equation (3.1) is, by convention, assumed one.

If all the forgetting factors were equal to one, Equation (3.1) becomes the usual log-likelihood function, which can be optimized for the standard MLEs. These were referred to as static MLEs in Chapter 2, since each datum contributes equally to the likelihood. The adaptive and static MLEs play an important role in developing a family of change detectors in Chapter 4.

Now suppose that the forgetting factors are strictly less than one. The form of the sequential likelihood in Equation (3.2) highlights how the forgetting factors introduce temporal adaptivity into the estimation. The most recent data point is included in the likelihood with full-weight, and the older terms in the likelihood are smoothly down-weighted. The rate in which data is forgotten is fully determined by the values of the forgetting factors, and how to specify their values is discussed shortly.

3.2.1 Adaptive mean for the Gaussian distribution

To illustrate how to compute adaptive MLEs, we derive the adaptive mean estimate for the Gaussian distribution. Supposing that \( X_{1:t} \sim \mathcal{N}(\mu, \sigma^2) \), the weighted log-likelihood function is given by

\[ \mathcal{L}_{FF}(\mu, \sigma^2 \mid x_{1:t}) = \sum_{k=1}^{t} w_k \left( C - \frac{1}{2\sigma^2}(x_k - \mu)^2 \right), \]

(3.3)
where $C$ is a constant independent of $\mu$ and $w_k = \prod_{\ell=k}^{t-1} \lambda_\ell$. Subsequently $w_k$ will be referred to as a weight, which is a function of time, and controls how much $x_k$ contributes to the likelihood.

Differentiating Equation (3.3) with respect to $\mu$ yields

$$\frac{\partial L_{FF}(\mu, \sigma^2 | x_{1:t})}{\partial \mu} = \frac{1}{\sigma^2} \sum_{k=1}^{t} w_k(x_k - \mu).$$

Setting equal to zero, and solving for $\mu$, results in the adaptive MLE

$$\tilde{\mu}_t = \frac{1}{n_t} \sum_{k=1}^{t} w_k x_k,$$

(3.4)

where $n_t = \sum_{k=1}^{t} w_k$ is a normalization factor, commonly referred to as the effective sample size (Bodenham and Adams, 2017). Throughout this thesis all adaptive estimates will be decorated with the symbol ‘∼’.

As written, Equation (3.4) is not suitable for streaming data. This is because computing $\tilde{\mu}_t$ requires storing $x_{1:t}$ in computer memory. This issue is alleviated by rewriting Equation (3.4) as the following set of recursive update equations:

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

(3.5)

$$\tilde{\mu}_t = \left(1 - \frac{1}{n_t}\right) \tilde{\mu}_{t-1} + \frac{x_t}{n_t},$$

(3.6)

which are derived in Appendix A.2. Equations (3.5)-(3.6) do not require any historic data to be stored in memory, and can be computed using a constant number of floating point operations per time-stamp, regardless of how much data is processed from the stream. Refer to Anagnostopoulos et al. (2012) for the recursive update equations for $\sigma^2$.

Although adaptive estimation for the Gaussian distribution is not a focus of this thesis, this example encapsulates many ideas that are used in this work. For example, the driving force behind the adaptive estimates in this thesis have update equations similar to Equations (3.5)-(3.6). This is because
distributions that are members of the exponential family exhibit the same type of recursive update equations, as highlighted in Anagnostopoulos (2010).

3.3 Fixed forgetting factors

In this section we demonstrate how the forgetting factors interact with the parameter estimates. The fixed forgetting case is considered, as it helps with interpretation, and it is assumed that a time independent $\lambda \in [0, 1]$ has been chosen. All interpretation is provided using the adaptive mean $\tilde{\mu}_t$ derived in the previous section; although, the interpretation is suitable for other adaptive MLEs.

In the case of fixed forgetting, the weights are given by $w_k = \lambda^{t-k}$. Substituting in these weights, and writing out the formula for $\tilde{\mu}_t$, reveals how the fixed forgetting factor introduces temporal adaptivity into the estimate:

$$\tilde{\mu}_t = \frac{1}{n_t} \left( \lambda^{t-1} x_1 + \lambda^{t-2} x_2 + \cdots + \lambda x_{t-1} + x_t \right). \quad (3.7)$$

First consider $\lambda \in \{0, 1\}$. When $\lambda = 1$, the effective sample size is $n_t = t$ and the static sample mean is recovered. This can be considered as a ‘no-forgetting’ scenario, since all data contributes equally to the mean estimate. Conversely, when $\lambda = 0$ the adaptive MLE is given by $\tilde{\mu}_t = x_t$. This can be seen by noticing that

$$n_t = \sum_{k=1}^{t} \lambda^{t-k} = \frac{1 - \lambda^t}{1 - \lambda},$$

implying that $n_t = 1$ when $\lambda = 0$. Therefore, in this case the adaptive estimate to the mean is given by the most recent data point. This can be considered as a ‘hard-forgetting’ scenario, since all data prior to $x_t$ does not contribute to the estimate.

For any intermediary value $\lambda \in (0, 1)$, we see from Equation (3.7) that every time a data point is observed, all previous data is multiplied by a factor of $\lambda$. Therefore, as more data arrives historic data is exponentially down-weighted, and contributes less to the estimate.
Figure 3.1: An illustration of the adaptive mean’s ability to react to drift in a data stream. The solid line is the static mean estimate, which uses $\lambda = 1$, and the dashed line corresponds to $\hat{\mu}_t$ computed using $\lambda = 0.95$.

Figure 3.1 displays a data stream generated according to

$$X_t \sim \begin{cases} \mathcal{N}(0, 1) & t \in [1, 1000], \\ \mathcal{N}(4, 1) & t \in (1000, 2000], \end{cases}$$  \hspace{1cm} (3.8)$$

along with the static mean estimate (solid line), and the adaptive MLE with a fixed forgetting factor $\lambda = 0.95$ (dashed line). From the figure it is clear that the static sample mean is unable to react to the drift in the data stream; whereas, the adaptive MLE is effectively able to discard non-informative data from the previous segment.

This section has also shed light onto why $n_t$ is referred to as the effective sample size in the literature. When $\lambda \in \{0, 1\}$, the effective sample size takes value in $n_t \in \{1, t\}$, which specifies how many observations are contributing to the estimate. By extension, for any $\lambda \in (0, 1)$, the value of $n_t$ can be interpreted as ‘loosely’ characterizing how many observations are contributing to an adaptive MLE. Refer to Bodenham (2014) for a thorough explanation on the effective sample size, as well as a detailed discussion on using forgetting factors in estimation.

Although fixed forgetting factors aid in understanding, it is not obvious how to choose $\lambda$ in practice. Further, there is no reason to suspect that a fixed forgetting factor will be sensible for data streams that may experience many
different types of drift. Ideally, when the stream is not experiencing drift the forgetting factor should be near one (since all data is relevant, and should be contributing equally); whereas, when drift occurs the forgetting factor should experience a sharp decrease in value, so that older, non-informative data can be effectively ‘washed-out’. This is the basic idea behind adaptive forgetting factors, which are discussed next.

3.4 Adaptive Forgetting Factors

The rate that data is discarded is directly related to the values of the forgetting factors. Therefore, it is critical in the streaming setting to remove the burden of having to subjectively specify their values. In this section an online method for tuning the forgetting factors is presented. This method is based on a variation of stochastic gradient descent, and has been empirically investigated in the literature. Standard gradient descent is discussed in Section 3.4.1, and is extended to tune the forgetting factors in Section 3.4.2. The Gaussian example of Section 3.2.1 is then revisited in Section 3.4.3, where the behavior of $\tilde{\mu}_t$, as a function of adaptive forgetting factors, is explored. The gradient descent method introduces a step-size parameter that has to be specified, and ways of choosing its value is deferred to Section 3.4.4.

3.4.1 Gradient Descent

Standard gradient descent is an iterative scheme that aims to optimize a function, and is a popular machine learning technique. The method is widely studied (see Bottou (2003); Boyd and Vandenberghe (2004)); therefore, only a brief overview of standard gradient descent is provided here.

Gradient descent is typically used in supervised settings, where a training data set $\{(x_k, y_k)\}_{k=1}^t$ is available. The vector $x_k$ represents covariates that are used to make a prediction for the response $y_k$. Commonly, gradient descent is used to minimize the empirical risk (Rumelhart et al., 1985), which
may be expressed as

\[ w_t = w_{t-1} - \eta \nabla \left( \frac{1}{t} \sum_{k=1}^{t} J(f(x_k \mid w_{t-1}), y_k) \right). \]  (3.9)

The function \( J \) represents a loss function that measures the cost of predicting \( y_k \) by the function \( f(x_k \mid w_{t-1}) \), which is parameterized by the vector of weights \( w_{t-1} \). The gradient is taken with respect to the weights, and \( 0 < \eta \ll 1 \) is a step-size that dictates the size of the step taken in the direction of the negative gradient. An example of this is linear regression, where \( w_{t-1} \) would be the estimated coefficients for the hyperplane at time \( (t - 1) \), and the loss function would be the squared difference between the response and prediction.

When a large amount of data is available, computing the exact gradient in Equation (3.9) becomes computationally challenging. In this case, stochastic gradient descent is preferred, which imposes a simple modification that drastically reduces computation. Stochastic gradient descent replaces the average in Equation (3.9) with a randomly sampled data point in the data set. That is, the weights are updated at time \( t \) as

\[ w_t = w_{t-1} - \eta \nabla J(f(x_{t^*} \mid w_{t-1}), y_{t^*}), \]  (3.10)

where \( t^* \in \{1, 2, \ldots, t\} \) is a randomly chosen index. Therefore, the computationally expensive gradient in Equation (3.9) is replaced with a noisy approximation of the gradient in Equation (3.10), which is quicker to compute. Refer to Bottou (2010, 2012) for a discussion on using stochastic gradient descent in big data settings.

### 3.4.2 Tuning the Forgetting Factors

Suppose that we want to compute \( \lambda_t \), assuming that values for \( \lambda_{t-1} \) have already been computed. Tuning the forgetting factors requires modifications to the stochastic gradient descent method. First, Equation (3.10) uses a randomly chosen data point to approximate the gradient. This implicitly
assumes that all the data points are stored in computer memory, and can be retrieved when necessary. This is not possible in the streaming paradigm; however, this issue can be easily remedied. Instead of using a randomly chosen data point, we approximate the gradient using the most recently processed observation $x_t$. The forgetting factor is then updated as

$$\lambda_t = \lambda_{t-1} - \eta \nabla J \left( \lambda_{1:t-1} \mid x_t, \tilde{\theta}_{t-1} \right),$$

(3.11)

where $J$ is a cost function that depends on previous values of the forgetting factors $\lambda_{1:t-1}$, the current datum $x_t$, and a vector of adaptive parameters $\tilde{\theta}_{t-1}$. As before, $\eta$ represents the step-size.

Second, to make Equation (3.11) tangible assumptions must be made about the computation of the gradient. As written, the gradient in Equation (3.11) is to be taken with respect to the forgetting factors $\lambda_{1:t-1}$. However, since data streams are unbounded, it is not possible to store every forgetting factor in memory. This, along with the forgetting factors being recursively defined in terms of one another, makes an exact gradient computation challenging. Benveniste et al. (1990) proposed computing an approximate gradient by treating $\lambda_{1:t-1}$ as instances of the same variable $\lambda$, and differentiating with respect to $\lambda$. This technique has been used to update forgetting factors in Anagnostopoulos (2010), and has been shown to result in forgetting factors that behave desirably. This may seem counter-intuitive, since our goal is to develop a method for computing time-varying forgetting factors. However, this heuristic argument was shown to agree with the rigorous approach considered in Bodenham (2014), which developed a forgetting factor derivative from first-principles. Moving forward we favor the simpler approach in Anagnostopoulos (2010), and the operator ‘$\nabla$’ is used to denote a derivative with respect to the scalar variable $\lambda$.

The last assumption deals with the choice of cost function. In this thesis, as in Anagnostopoulos et al. (2012), the one-step ahead negative log-likelihood function is chosen for $J$. This is a natural way of measuring how well the
estimates at time \((t - 1)\) fit the \(t^{th}\) observation, and is given by
\[
J \left( \lambda_{t-1} \mid x_t, \tilde{\theta}_{t-1} \right) = -\mathcal{L} \left( \tilde{\theta}_{t-1} \mid x_t \right),
\]
where \(\mathcal{L}(\cdot \mid \cdot)\) is the log-likelihood function of the data, and \(\tilde{\theta}_{t-1}\) is a vector of adaptive MLEs for the parameter vector that governs the likelihood. Observe that \(\tilde{\theta}_{t-1}\) is a function of the effective sample size \(n_{t-1}\), which is a function of previous values of the forgetting factors. In the next section the behavior of the adaptive forgetting factor \(\lambda_t\), tuned via the methods proposed in this section, is demonstrated via an example.

### 3.4.3 Adaptive Forgetting for the Mean

For illustration purposes suppose that \(X_t \sim \mathcal{N}(\mu_t, 1)\), where \(\mu_t\) is unknown. Equation (3.11), for Gaussian data, is given by
\[
\lambda_t = \lambda_{t-1} + \eta \nabla \left( C - \frac{1}{2} (x_t - \tilde{\mu}_{t-1})^2 \right),
\]
where \(C\) is a constant that vanishes upon differentiation. The only term that depends on the forgetting factors is \(\tilde{\mu}_{t-1}\), and computing the gradient supposing it is a function of the single variable \(\lambda\) results in
\[
\lambda_t = \lambda_{t-1} + \eta (x_t - \tilde{\mu}_{t-1}) \nabla \tilde{\mu}_{t-1}.
\]
The gradient of the adaptive MLE can be computed via direct differentiation of Equations (3.5)-(3.6), resulting in the recursive updates
\[
\nabla n_t = \lambda_{t-1} \nabla n_{t-1} + n_{t-1}
\]
\[
\nabla \tilde{\mu}_t = \left( 1 - \frac{1}{n_t} \right) \nabla \tilde{\mu}_{t-1} - \frac{\nabla n_t}{n_t^2} (x_t - \tilde{\mu}_{t-1}).
\]
We now revisit the data stream generated according to Equation (3.8), and investigate the behavior of \(\lambda_t\) and \(\tilde{\mu}_t\). The top and bottom plots in Figure 3.2 display, respectively, the values for \(\lambda_t\) and \(\tilde{\mu}_t\), averaged over 1,000 data
Figure 3.2: Plots of the adaptive forgetting factors $\lambda_t$ (top figure) and the adaptive MLE mean estimate $\tilde{\mu}_t$ (bottom figure), averaged over 1,000 data streams generated according to Equation (3.8). In both plots the solid, dashed and dotted curves correspond to using, respectively, the step-sizes $\eta = 0.1, 0.01, 0.001$.

From the top plot in Figure 3.2, we see that $\lambda_t$ behaves similarly to the desirable property that we alluded to earlier. That is, on $[1, 1000]$ the forgetting factors remain near one, indicating that the data from the first segment are contributing to the estimate with almost full-weight. Once the data stream experiences a change in location (at time $\tau = 1000$), the forgetting factors, for each value of $\eta$, all experience an abrupt decrease in value. This allows data prior to the drift to be forgotten at a much faster rate – something that is not possible when using a fixed $\lambda$. The forgetting factors then increase, stabilizing to around one, which is desirable since all observations on $(1000, 2000]$ are being generated by the same distribution, and are informative from an
estimation perspective.

Some comments are required. First, how quickly $\lambda_t$ ‘recovers’ after drift depends on the choice of $\eta$, where larger values of the step-size allow $\lambda_t$ to stabilize more quickly. However, this does not imply that larger step-sizes should always be preferred. Indeed, larger step-sizes typically result in estimates that are much more volatile, since they allow for larger jumps in the gradient descent step. Therefore, there is a trade-off to consider when choosing $\eta$, which is discussed more Section 3.4.4. However, from an estimation perspective the different choices of $\eta$ do not, on average, greatly affect the adaptive MLE (refer to the bottom plot in Figure 3.2), indicating that choosing a value for $\eta$ is less important than choosing values for $\lambda_t$.

Lastly, the stochastic gradient descent step does not guarantee that the forgetting factors stay within the interval $[0,1]$. Therefore, truncating the value of $\lambda_t$ is necessary in practice. This is investigated in Bodenham (2014), where it is suggested that after the gradient update the forgetting factors be truncated according to

$$
\lambda_t = \max\{\min\{\lambda_t, \lambda_{\max}\}, \lambda_{\min}\}.
$$

Equation (3.12) projects the forgetting factors onto the interval $[\lambda_{\min}, \lambda_{\max}]$, where $\lambda_{\min}$ and $\lambda_{\max}$ are a minimum and maximum allowed value for $\lambda_t$. It is clear that $\lambda_{\max} = 1$ is a sensible choice; although, choosing $\lambda_{\min}$ is more subtle. It may seem obvious that $\lambda_{\min}$ should be chosen small; however, in practice this is not advantageous – as discussed in Bodenham (2014) and Anagnostopoulos et al. (2012). This is because small values for $\lambda_{\min}$ tend to increase the length of the recovery period after drift occurs, and may even cause numerical instabilities. In the literature it has been suggested that $\lambda_{\min} \in [0.6, 0.8]$ are reasonable choices, and the value of $\lambda_{\min}$ does not significantly affect the adaptive estimates. In Figure 3.2, $\lambda_{\min} = 0.8$ was used.
3.4.4 Choosing the step-size

Tuning $\lambda_t$ online introduced a step-size parameter $\eta_t$. Hence, it appears that removing the burden of subjectively choosing the forgetting factors has been replaced with having to specify the step-size. This is a caveat with adaptively tuning parameters online – in doing so other parameters are introduced whose values must be chosen. In this section we discuss some common ways for choosing a time-varying step-size $\eta_t$, and argue why a constant step-size is sensible, and sufficient for this thesis.

For offline gradient descent, common ways of choosing $\eta_t$ include line-searches (Boyd and Vandenberghe, 2004), or Robbins-Monroe type algorithms from stochastic approximation theory (Benveniste et al., 1990). However, these techniques are inappropriate for streaming data, where the function to be optimized is constantly changing. Specifically, performing a line-search at each time-stamp will either be too computationally demanding, or introduce more parameters that have to be specified. For example, the backtracking method (Boyd and Vandenberghe, 2004) introduces two parameters that need to be chosen. The standard Robbins-Monroe algorithm (Robbins and Monro, 1951) is a root finding algorithm, where the objective function is unknown, but can be expressed as an expected value of some random variable. To be able to converge, $\eta_t$ is typically chosen to satisfy

$$\sum_{t=1}^{\infty} \eta_t = \infty, \quad \sum_{t=1}^{\infty} \eta_t^2 < \infty,$$

with a common choice being $\eta_t = \frac{1}{t}$. However, for unbounded streams that experience drift, letting $\eta_t \to 0$ is not advisable. This is because we always want to retain estimates that are able to quickly react to changes in the data generating process, even after potentially observing millions of observations. In Anagnostopoulos (2010) it is suggested that the sample size $t$ be replaced with its streaming counterpart – the effective sample size $n_t$. Although a sensible suggestion, there does not appear to exist any literature connecting the theoretical results of Robbins and Monro (1951) to the drifting data stream setting.
Various other methods have also been proposed for updating $\eta$. Riedmiller and Braun (1993) introduced the $R$-$Prop$ algorithm, which gradually increases the step-size when the sign of consecutive cost function gradients are the same, and decreases its value otherwise. How much to increment (or decrement) the step-size requires a parameter to be specified, which plays a similar role to the original step-size. The $RMS$-$Prop$ method was developed by Hinton et al. (2012), and divides the current step-size by the square-root of an exponentially weighted moving average of squared cost function gradients, which is off-set by a small factor to avoid division by zero. Hence, removing the burden of choosing the step-size has essentially been replaced with choosing a fixed forgetting factor that dictates the rate in which the squared gradients are discarded. See Ruder (2016) for a comprehensive overview of the gradient descent method, and its variants.

In this thesis we favor using a fixed step-size $\eta$, as opposed to tuning a time-varying step-size that relies on a method which introduces additional parameters. This decision has been carefully made, and has considered several factors. First, the effectiveness of the methodology introduced in this work depends on the adaptive estimates, which from an estimation perspective, is not greatly affected by the choice of step-size (see Figure 3.2). Lastly, choices for the step-size have been empirically investigated in the literature (Anagnostopoulos, 2010; Bodenham, 2014), where values $\eta \in [10^{-5}, 10^{-2}]$ have been shown to be sensible. This is guidance we benefit from in subsequent chapters. Henceforth, we keep $\eta$ fixed, which is a sensible choice due to the absence of procedures that are capable of tuning the step-size without introducing additional parameters.

3.5 DISCUSSION

This chapter has introduced a framework, based on forgetting factors, that allows for temporally adaptive MLEs to be computed by optimizing a weighted log-likelihood function. The adaptive mean for the Gaussian distribution was then derived using this framework, which introduced notation and update equations that are the driving force behind adaptive estimates appearing
later in the thesis.

How the forgetting factors interact with parameter estimates was then discussed in the context of fixed forgetting. A way for computing adaptive forgetting factors was then presented, which is based on the stochastic gradient descent method. This method introduced a single parameter, and a thorough discussion on how to specify its value was provided. Adaptive forgetting factors are a fundamental component of this thesis, being used in all subsequent chapters.

At this stage, all necessary background material has been presented. The forthcoming chapters develop the novel contributions of this thesis, with Chapters 4 and 5 focusing on changepoint methodology for categorical data streams, and Chapters 6 and 7 focusing on developing an adaptive histogram estimate for univariate data streams, and exploring various applications of this technique.
This chapter presents the first major contribution of this thesis – a general family of change detection methods for categorical data streams. The proposed methodology relies on the behavior of the adaptive and static MLEs for the multinomial distribution, and detects changepoints based on a novel adaptive thresholding technique. This technique introduces a single parameter, which can be specified given constraints on the false positive rate.

Section 4.1 discusses literature directly relevant to this chapter. Section 4.2 introduces categorical data streams, and the adaptive forgetting framework discussed in Chapter 3 is modified for these streams in Section 4.3. A general framework for change detection is proposed in Section 4.4, which utilizes novel adaptive thresholds to identify changepoints. The proposed detectors require a dissimilarity measure to be specified, and two measures are explored in Section 4.5. Section 4.6 discusses comparison methods, a set of synthetic experiments are analyzed in Section 4.7, and a data stream collected from Imperial College London’s computer network is explored in Section 4.8. This chapter extends the previous work of Plasse and Adams (2019).
4.1 Literature review

Streaming changepoint detection for categorical data has received a small amount of attention in recent decades. Much existing work, although having merit in their own right, falls victim to many common pitfalls. A brief overview of existing work is provided here, where some of the current issues that are typically found in the literature are highlighted.

Wolfe and Chen (1990) appear to be one of the first papers to develop changepoint methodology for multinomial sequences. Their work focuses on detecting a single changepoint, and does so by recomputing statistics for various potential changepoint locations. This violates the single-pass requirement, and is therefore not suitable for high-frequency data streams. Similar comments can be made about Hou et al. (2013), which combines maximum likelihood estimation and the p-chart (Montgomery, 2007) to detect a single changepoint in a sequence of categorical data.

The method proposed by Ienco et al. (2014) depends on a ‘context-based’ distance; however, the choice of a good context is non-trivial. Furthermore, Chebyshev’s inequality is used to detect changes, and does so by introducing several thresholds that must be specified. A fuzzy multinomial control chart for linguistic variables is considered in Amirzadeh et al. (2008), and uses fuzzy set theory to formally address the vagueness in assigning categories (such as poor, fair and excellent) to items being monitored. Assigning weights to the linguistic variables is non-trivial, and the authors suggest that they should be specified after careful discussion with experts.

Cao and Huang (2013) develop a method based on rough set theory. Their method uses a constant-width sliding window and fixed threshold to look for changepoints, where the threshold value is suggested to be chosen based on available prior knowledge. Chen et al. (2009) present a framework for clustering drifting categorical data. Again, a constant-width sliding window is used to handle the temporal variation, and multiple thresholds are introduced that must be specified. Changepoint detectors based on moving sum statistics (Eiauer and Hackl, 1978) have been considered recently in Eichinger et al. (2018) and, similar to other methods, introduce a fixed ‘bandwidth’
parameter, which is analogous 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 it is suggested that the method be run for multiple values of the bandwidth, and an aggregation of the results be performed. However, this requires multiple passes over a single data stream and still does not guarantee that a ‘sensible’ bandwidth has been found. Weiß (2012) also considers fixed sliding windows and uses the Gini index to monitor drifting categorical data streams.

Modifications to the CUSUM control chart (see Section 2.4.2) have been proposed, which make the chart applicable to Bernoulli and multinomial sequences. Ryan et al. (2011) develop a multinomial CUSUM chart, and the authors mention that if no information is available for the direction of the drift in each category, then a Bernoulli CUSUM chart should be used separately for each category – guidance we will utilize later in Section 4.7. Another multinomial CUSUM method is developed in Höhle (2010), where the category probabilities are modeled using a logistic regression model. The method assumes that the category probabilities after drift manifest as shifts in the intercept of the regression model, which introduces many ‘scaling factors’. Both charts require prior knowledge of the shift in each category probability, which will be difficult to anticipate in the multiple changepoint setting. Subsequently, following the advice provided in Ryan et al. (2011), multiple Bernoulli CUSUM charts will be used as opposed to a multinomial CUSUM chart. An overview of the Bernoulli CUSUM chart is given in Section 4.6.1.

Due to computational complexities, Bayesian techniques are usually developed in batch settings. However, some recent research has been devoted to extending Bayesian methodology to the streaming data paradigm. Adams and MacKay (2007) compute the distribution for the length of the current run, that is, the amount of time elapsed since the last detected changepoint. The method’s complexity is linear, both in storage and time; although constant complexity per time-step can be achieved via thresholding the tails of the run length distribution. Nonetheless, the worst case complexity is linear in the length of the data, making it ill-suited for high-frequency data streams. Byrd et al. (2017) develop a lagged version of the work presented in Adams
and MacKay (2007), and it is shown through simulation that the method of Adams and MacKay (2007) has difficulties in detecting small changes. This setting is a primary focus of this chapter since the category probabilities are bounded in \([0,1]\), meaning that the largest magnitude of a changepoint can be, at most, one. The method of Byrd et al. (2017) identifies changepoints by thresholding a relative change in the maximum a posteriori estimate of the run length distribution. Introducing a lag comes at a price in terms of computational complexity, as it becomes exponential in the lag. Thus, as online Bayesian methods are not directly relevant to the focus of this chapter, they are not considered in the sequel.

4.2 Drifting categorical data streams

In this chapter it is assumed that independent realizations from a \(K\)-category multinomial distribution arrive sequentially, and at high-frequency. It is further assumed that the number of categories \(K \in \mathbb{Z}^+\) is known, fixed, and \(K > 1\). In Chapter 6, where a streaming histogram estimate is developed, these conditions will be relaxed. The univariate stream in Equation (2.1) can be reexpressed as

\[
\langle x_1, x_2, \ldots, x_{t-1}, x_t, \ldots \rangle,
\]

where \(x_t \in S = \{1, 2, \ldots, K\}\) for every time-stamp \(t\). Therefore, the stream in Equation (4.1) can be viewed as observations being generated from a \(K\)-sided die, where the die is rolled indefinitely, and whose category probabilities evolve over time.

An equivalent representation of Equation (4.1) will be necessary when detectors are used as comparison methods in Section 4.7. This representation is commonly referred to as the binarization of the categorical stream (Weiß, 2012), and can be expressed as

\[
\langle e_{x_1}, e_{x_2}, \ldots, e_{x_{t-1}}, e_{x_t}, \ldots \rangle,
\]

where \(e_{x_t} \in \mathbb{R}^K\) 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 $x_t$. As an example, if $K = 3$ and $x_t = 1$, then $e_{x_t} = (1,0,0)^\top$. This chapter is concerned with sequentially monitoring a multinomial distribution for multiple changepoints, in the streaming paradigm, where a novel framework is developed for data streams assuming the form of Equation (4.1).

Since data streams will experience drift, the discussion of the multinomial distribution provided in Section 2.2.1 needs to be revised to allow for time-varying estimation. Suppose $t$ represents our current position in the stream, and that $x_{1:t}$ have been observed. Then the multinomial distribution governing the stream may be expressed as

$$
\mathbb{P}(C_t^{(1)} = c_t^{(1)}, \ldots, C_t^{(K)} = c_t^{(K)} \mid p_t, t) = \frac{t!}{\prod_{i \in S} (c_t^{(i)})!} \prod_{i \in S} (p_t^{(i)})^{c_t^{(i)}},
$$

where

$$
p_t = \begin{cases} 
    p^{(1)} & 1 \leq t \leq \tau_1, \\
    p^{(2)} & \tau_1 < t \leq \tau_2, \\
    \vdots & \vdots \\
    p^{(m)} & \tau_{m-1} < t \leq \tau_m, \\
    p^{(m+1)} & \tau_m < t < \infty.
\end{cases}
$$

As discussed in Section 2.1, the set $\tau = \{\tau_k\}_{k=1}^m$ contains changepoints – unknown time-instances where the stream undergoes a change in the data generating process. Between changepoints it is assumed that the probability vector does not drift, that is, the set of changepoints partition the stream into $(m + 1)$ segments, and in each segment the multinomial distribution governing the data stream is assumed fixed.

Given the observed data $x_{1:t}$, the MLE for the category probabilities was shown in Section 2.2.1 to be $\hat{p}_t = \hat{c}_t / t$, $\hat{c}_t$ being the vector of observed category counts. The vector $\hat{p}_t$ represents the static MLE, and gives each datum equal weight in the computation of the estimate, regardless of how many changepoints are present. This estimate plays a vital role in the development of the family of change detection methods presented in Section 4.4.
Allowing every observation to have equal weight in the estimation process will be detrimental as more data is collected from a drifting data stream. Further, any changes in a categorical data stream will manifest in the category probabilities; therefore, a way of adaptively estimating these probabilities is crucial in the streaming paradigm. The forgetting factor framework discussed in Chapter 3 is now modified for categorical data streams, allowing for efficient and accurate estimation of the category probabilities, resulting in adaptive MLEs.

4.3 Adaptively estimating the category probabilities

In the case of the multinomial distribution, the temporally aware likelihood given in Equation 3.1 takes the form

$$L_{FF}(\mathbf{p} | x_{1:t}) = \sum_{k=1}^{t} w_k L(\mathbf{p} | x_k),$$

where $L(\cdot | \cdot)$ is the log-likelihood associated with the multinomial distribution, and the time subscript has been omitted from the probability vector to reinforce that the temporal adaptivity of the soon to be derived estimates is strictly due to the addition of the weights $w_k$. The weights $w_k$ were introduced in Chapter 3, are a function of the forgetting factors, and allow for the category probabilities to be temporally adaptive.

The weights are independent of the category probabilities; therefore, optimization of Equation (4.3) is nearly identical to the derivation of the static MLEs, which can be found in Appendix A.1. For completeness a derivation is provided in Appendix A.2. Optimization of Equation (4.3) results in, for every $i \in \mathcal{S}$, the temporally adaptive MLEs

$$\tilde{p}^{(i)}_t = \frac{1}{n_t} \sum_{k=1}^{t} w_k \mathbf{1}(x_k = i),$$

where the weights $\{w_k\}_{k=1}^{t}$, and effective sample size $n_t$ are restated below.
for convenience
\[ w_k = \prod_{\ell=k}^{t-1} \lambda_{\ell}, \quad n_t = \sum_{k=1}^{t} w_k. \] (4.5)

Henceforth \( \tilde{p}_t \in \mathbb{R}^K \) is used to denote the vector of adaptive MLEs, whose \( i^{th} \) component is given by \( \tilde{p}_t^{(i)} \). As discussed in Chapters 2-3, parameter estimates decorated with the symbols ‘\( \wedge \)’ and ‘\( \sim \)’ will refer to static and adaptive estimates respectively.

As presented, computation of the vector \( \tilde{p}_t \) is not suitable for streaming data. This is due to the fact that Equation (4.4) requires maintaining all historical data in computer memory, which is infeasible in the streaming paradigm. However, this issue is alleviated by rewriting Equation (4.4) as the following recursive update equations:

\[ n_t = \lambda_{t-1} n_{t-1} + 1, \] (4.6)
\[ \tilde{p}_t^{(i)} = \left( 1 - \frac{1}{n_t} \right) \tilde{p}_{t-1}^{(i)} + \frac{1}{n_t} \mathbb{1}(x_t = i). \] (4.7)

Derivation of Equations (4.6)-(4.7) follows directly from the derivations of Equations (3.5)-(3.6), which may be found in Appendix A.2. This formulation of the adaptive category probability estimates is similar to that of Anagnostopoulos et al. (2012) and Plasse and Adams (2016), which utilized similar updates in the development of online classification algorithms.

The stochastic gradient descent step for tuning the forgetting factors, in the case of categorical data streams, is given by

\[ \lambda_t = \lambda_{t-1} + \eta \nabla \left( \sum_{i \in S} \mathbb{1}(x_t = i) \log \left( \frac{\tilde{p}_{t-1}^{(i)}}{\tilde{p}_{t-1}^{(i)}} \right) \right) \]
\[ = \lambda_{t-1} + \eta \left( \sum_{i \in S} \mathbb{1}(x_t = i) \frac{\nabla \tilde{p}_{t-1}^{(i)}}{\tilde{p}_{t-1}^{(i)}} \right), \] (4.8)

where, as discussed in Section 3.4.2, the cost function being optimized represents the one-step-ahead log-likelihood function, which provides an empirical measure of how the estimates at time \( (t - 1) \) fit the newest observation. Additionally, the operator ‘\( \nabla \)’ denotes differentiation with respect to the scalar
variable $\lambda$, as was discussed in Section 3.4.2. The gradients appearing in
Equation (4.8) can be recursively updated by directly differentiating Equa-
tions (4.6)-(4.7), resulting in

\[
\nabla n_t = \lambda_{t-1} \nabla n_{t-1} + n_{t-1},
\]

\[\text{Equation (4.9)}\]

\[
\nabla \hat{p}_t^{(i)} = \left(1 - \frac{1}{n_t}\right) \nabla \hat{p}_{t-1}^{(i)} - \frac{\nabla n_t}{n_t^2} \left(1(x_t = i) - \hat{p}_t^{(i)}\right) .
\]

\[\text{Equation (4.10)}\]

The initial values for the adaptive and static MLEs $\hat{p}_0$ and $\hat{p}_0$ are estimated
via a burn-in period of length $B$, as discussed in Section 2.4.4. During the
burn-in, $B$ observations are used to estimate the static probabilities of seeing
each category offline; consequently, $\hat{p}_0 = \hat{p}_0$. Since $B$ observations are used
in estimating the probabilities, it also follows that $n_0 = B$. The gradients
in Equation (4.10) represent the rate of change of the adaptive MLEs with
respect to the forgetting factors. During the burn-in period the probabilities
are statically estimated using a fixed forgetting factor $\lambda = 1$; therefore the
starting values for the gradients are initialized to zero.

Equations (4.6)-(4.10) provide an entire scheme for adaptively estimating the
category probabilities, where temporal adaptivity is provided by a sequence
of forgetting factors that can be autonomously updated without any user
supervision. These recursive update equations are the driving force behind
most of the methodology proposed in this thesis and, although the functional
form appears different in other chapters, the main idea remains the same.
The recursive updates require minimal storage and can be computed using
a constant number of floating point operations per time-stamp, making the
adaptive estimation of $\hat{p}_t$ suitable for streaming data. Moreover, if all forget-
ting factors are set to one, Equations (4.4) and (4.7) become, respectively, the
non-sequential and sequential equations for the static MLE $\hat{p}_t$, as discussed
in Section 2.2.1. Therefore the static MLEs for the category probabilities can
also be computed without introducing any computational/storage overhead.

Figure 4.1 illustrates the behavior of the adaptive estimates $\hat{p}_t$ and the static
estimates $\hat{p}_t$ when data streams experience a change at time $\tau = 500$. Prior
to the drift the adaptive and static MLEs are, on average, indistinguishable
since the adaptive forgetting factors will be near one (as seen in Figure 3.2),
which is precisely the static case. Conversely, after the changepoint \( \hat{\mathbf{p}} \) quickly
adapts to the new category probabilities while \( \hat{\mathbf{p}} \) struggles to accurately
estimate the probabilities in the new segment. This is because data before
the changepoint are contributing equally to \( \hat{\mathbf{p}} \), hindering its ability to react
to the drift. The behavior of \( \hat{\mathbf{p}} \) and \( \hat{\mathbf{p}} \) during periods of drift or no-drift
forms the basis of the family of change detection methods for categorical
streams, and is discussed in the next section.

4.4 General multinomial change detectors

The intuition behind the family of Multinomial Change Detection Methods
(MCDMs) proposed in this chapter is based on the behavior of the adaptive
estimates (\( \hat{\mathbf{p}} \)) and static estimates (\( \hat{\mathbf{p}} \)) during drifting and non-drifting
periods. When drift is not occurring, the adaptive forgetting factors fluctuate
near one – corresponding to the static case. In this setting the adaptive and
static MLEs will be approximately equal. Conversely, since the vector \( \hat{\mathbf{p}} \)
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 \( \hat{\mathbf{p}} \) – resulting in the two vectors *diverging*. A change detector can then be constructed by flagging a change whenever
the dissimilarity between the two MLEs becomes ‘too large’. This forms the basis of the MCDMs proposed in this chapter, which is an extension of the work in Plasse and Adams (2019), where a single dissimilarity measure was considered, and Plasse et al. (2017), where adaptive and static estimates were compared to develop a change detector for cyber-physical systems.

To provide mathematical meaning to the vectors $\hat{p}_t$ and $\tilde{p}_t$ being ‘too far apart’ requires a dissimilarity measure to be defined. Such measures are commonplace in statistics and machine learning – refer to Webb (2003) for a survey of commonly used dissimilarity measures.

Let $A(\cdot, \cdot)$ represent an arbitrary dissimilarity measure, which satisfies

$$A(x, x) = 0, \quad (4.11)$$
$$A(x, y) > 0, \quad (4.12)$$

for every pair of vectors $x, y \in \mathbb{R}^K$. Equation (4.11) says that a vector is not dissimilar from itself, and Equation (4.12) states that distinct vectors have a strictly positive dissimilarity. The choice of dissimilarity measure will always be open to discussion (Vovk et al., 2005); however, it is intuitive to choose the measure so that the more dissimilar $x$ and $y$ are, the larger $A(x, y)$ is.

Equations (4.11)-(4.12) are the weakest assumptions that may be placed on a dissimilarity measure, since symmetry and the triangle inequality are not necessary in the definition. This is why we have carefully used ‘dissimilarity measure’ as opposed to ‘metric’; although, the MCDMs to be developed can use either dissimilarity measures or metrics. In fact, in Section 4.5.1 a dissimilarity measure is used to develop a change detector; whereas, in Section 4.5.2 a metric is used to aid in the detection of changepoints.

Suppose a dissimilarity measure has been chosen. When no drift is occurring $A(\tilde{p}_t, \hat{p}_t)$ should be close to zero, as the adaptive and static estimates should be similar. On the other hand, any changes in the data generating process will affect the category probabilities, and will result in ‘spikes’ in $A(\tilde{p}_t, \hat{p}_t)$, as the adaptive estimates will react to the change more quickly than the
static estimates. Therefore a change can be flagged, at time $t$, whenever

$$A(\hat{p}_t, \hat{p}_t) > \varepsilon_t,$$  

(4.13)

where $\varepsilon_t$ is a suitably chosen threshold.

In the sequential change detection literature it is common to choose thresholds so that the probability of the detector raising a false alarm remains fixed at each time-stamp, (e.g. see Ross et al. (2011)). This is similar to the classical hypothesis testing scenario, where the Type I error rate is to be controlled, and is analogous to choosing thresholds so that a desired value of $\text{ARL}_0$ is attained. However, choosing thresholds to obtain an exact $\text{ARL}_0$ is difficult in practice, and any guarantees typically rely on the distribution generating the data being fully specified, that is, both the distribution and its parameters are known. This is a naive assumption for streaming data, as the distribution generating the data is rarely known, and is continuously evolving. Consequently, we rely on a technique that approximately achieves a desired value for $\text{ARL}_0$, and a method is presented in the next section that results in easily computable adaptive thresholds, i.e., thresholds whose values change over time (as in Equation (4.13)).

4.4.1 Adaptive thresholds

Optimally selecting a sequence of thresholds in the streaming paradigm remains a challenging task. Consequently, much existing literature has settled on fixed thresholds to identify changepoints in a data stream. This is common in the statistical quality control literature (Montgomery, 2007), where control limits act as fixed thresholds, and are usually chosen to be some number of standard deviations away from the mean of the process being monitored.

In the streaming paradigm it is evident that the threshold used to detect changepoints is a crucial parameter for any change detection procedure. Additionally, there is no reason to suspect that a subjectively chosen threshold will efficiently detect changepoints, while simultaneously controlling the false positive rate. Hence, in the remainder of the chapter we refrain from using
fixed thresholds, and develop methodology for setting time-varying thresholds whose values, given a desired \( \text{ARL}_0 \), are completely specified.

Defining the adaptive thresholds requires an upper bound on the dissimilarity measure \( A(\cdot, \cdot) \) to be specified. That is, the thresholds require the existence of a function \( u(\cdot, \cdot) \), such that

\[
A(\tilde{p}_t, \hat{p}_t) \leq u(\tilde{p}_t, \hat{p}_t), \quad \forall \tilde{p}_t, \hat{p}_t \in S^K,
\]

\( S^K \) denoting the unit simplex in \( \mathbb{R}^K \). The adaptive threshold at time \( t \) is then defined as

\[
\varepsilon_t = \beta u(\hat{p}_t, \hat{p}_t),
\]

where \( \beta \in (0, 1) \) is a control parameter. The value of \( \beta \) can be viewed as an ‘allowance’, i.e., \( \beta \) determines how large the dissimilarity between the adaptive and static MLEs can be, in relation to the upper bound, before a changepoint is flagged. Choosing a value for \( \beta \) is reminiscent of the trade-off that must be considered when developing online change detection strategies, as discussed in Section 2.4.3. Lower values of \( \beta \) are likely to result in smaller values for \( \text{ARL}_1 \), since the dissimilarity only needs to exceed a small proportion of the upper bound. This will also coincide with smaller values for \( \text{ARL}_0 \), resulting in the detector flagging many false positives. In contrast, larger values of \( \beta \) will result in fewer false positives, but at the expense of longer detection delays. It is clear that the allowance \( \beta \) should not be subjectively chosen by an analyst, but prescribed in some way that results in a change detector approximately respecting a desired value for \( \text{ARL}_0 \). The next section proposes a novel way of tuning \( \beta \).

4.4.2 Tuning the allowance

To tune the allowance \( \beta \) we follow convention, and choose its value to result in a detector (approximately) adhering to a desired value of \( \text{ARL}_0 \). A typical way of doing this, e.g. see Ross et al. (2011), is by running Monte Carlo simulations before a stream is analyzed, to gain insight into how the value of the threshold affects the false positive rate. This section proposes a novel
way of using Monte Carlo simulations to tune the allowance $\beta$.

First, consider the experimental design for the Monte Carlo simulations. Let $\mathcal{G}$ be a discretization of the interval $[0, 1]$, which contains $\beta$ values used in the simulations. Since the simulations are conducted offline, computational issues are not a major concern; therefore, the discretization can be taken as

$$\mathcal{G} = \left\{ 0, \frac{1}{\ell}, \frac{2}{\ell}, \ldots, \frac{\ell - 1}{\ell}, 1 \right\},$$

for a large $\ell \in \mathbb{Z}^+$ (e.g. $\ell = 1000$). Another thing to consider is the length of each stream used in the simulations (denoted by $N$). Recall that we want to investigate how $\beta$, in conjunction with the upper bound $u(\cdot, \cdot)$, affects the value of $\text{ARL}_0$. It is clear that there is a positive correlation between $\beta$ and $\text{ARL}_0$, that is, as $\beta$ increases so will $\text{ARL}_0$. In the extreme case where $\beta = 1$, no changepoints will be flagged by the detector; hence, the value $N$ implicitly defines the maximum value of $\text{ARL}_0$ that a change detector can attain. Thus, from a practical stand-point the values $\ell$ and $N$ should be chosen sufficiently large so that the grid $\mathcal{G}$ is fine enough to allow for the accurate approximation of any desired value of $\text{ARL}_0 \in [0, N]$.

For each $\beta \in \mathcal{G}$ an associated value of $\text{ARL}_0$ is approximated by running the change detector over $M \in \mathbb{Z}^+$ Monte Carlo replicates, and averaging over the time-stamps where the first false positive was flagged. Since $\text{ARL}_0$ tells us, on average, how many observations are processed before the detector flags a false positive, we can restrict attention to streams with no changepoints, governed by a single probability vector. In each simulation, the probability vector is uniformly sampled from the simplex $S^K$, $K$ being the fixed number of categories used in the simulations. This is to provide an ‘overall picture’ on how the allowance $\beta$ affects the value of $\text{ARL}_0$.

The Monte Carlo simulations result in a set of tuples

$$\left\{ \left( \beta^{(k)}, \text{ARL}_0^{(k)} \right) \right\}_{k=1}^{\vert\mathcal{G}\vert},$$

where $\vert\mathcal{G}\vert$ denotes the cardinality of the discretization grid $\mathcal{G}$, and $\text{ARL}_0^{(k)}$ is the estimated $\text{ARL}_0$ using the allowance parameter $\beta^{(k)}$. These simulations
provide us with empirical evidence on how the allowance is related to the average run length. However, as we have a discrete set of tuples, it is not guaranteed that a value of $\beta \in \mathcal{G}$ results in a detector that closely approximates any desired value of $\text{ARL}_0 \in [0, N]$. To alleviate this issue, the results of the simulations are extending via a curve-fitting procedure, so that an allowance $\beta$ may be computed for any value of $\text{ARL}_0 \in [0, N]$.

There are many families of curves that could be fit to the tuples in Equation (4.14). Considering the positively correlated relationship between $\beta$ and $\text{ARL}_0$, it is clear that the fitted curve should increase as $\beta$ increases. Furthermore, in Sections 4.5.1-4.5.2 the simulation results are shown for two specific dissimilarity measures, and evidence (refer to Figures 4.4 and 4.6) suggests a sigmoidal relationship between $\beta$ and $\text{ARL}_0$. Therefore, we propose fitting the sigmoid

$$\text{ARL}_0(\beta \mid c_1, c_2, c_3) = \frac{c_1}{1 + \exp\left(\frac{c_2 - \beta}{c_3}\right)},$$

(4.15)

where the coefficients $c_1, c_2$ and $c_3$ can be estimated using the results from the Monte Carlo simulations. Equation (4.15) can then be inverted so that, given a desired value of $\text{ARL}_0 \in [0, N]$, the value of the allowance may be chosen according to

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

(4.16)

The coefficient $c_1$ represents the horizontal asymptote of the sigmoid, $c_2$ is the horizontal location of the inflection point, and $c_3$ is a scaling factor. Refer to Figure 4.2 for a graphical depiction on how the coefficients relate to the sigmoid. Subsequently, the coefficients $c_1, c_2$ and $c_3$ are computed by fitting a self-starting logistic model (Pinheiro and Bates, 2000) to the results in Equation (4.14), where optimization is done using the Gauss-Newton algorithm to obtain the nonlinear least squares estimates for the coefficients.

Some overall comments are in order. The Monte Carlo simulations require several values to be specified:

- $N$ – the length of the streams used in the simulations, which implicitly
Figure 4.2: A graphical display of how the coefficients $c_1$, $c_2$ and $c_3$ affect the sigmoid given in Equation (4.15). This figure is a modification of what appears in the R documentation for the SS1ogis function, developed by the same authors of Pinheiro and Bates (2000).

places an upper bound on the value of $\text{ARL}_0$ a detector can attain,

- $M$ – the number of Monte Carlo simulations,
- $\ell$ – the parameter that defines how fine or coarse the grid $\mathcal{G}$ is and,
- $K$ – the number of categories in each stream.

Since this is one of the few times where computational constraints are not a concern, the values for $N, M$ and $\ell$ can be chosen sufficiently large so that a practitioner’s desired value of $\text{ARL}_0$ can be accurately approximated by the choice of $\beta$. For $K$, Monte Carlo simulations should be conducted for a wide-range of values (e.g. $K \in \{3, 6, 10, 25, 50, 100\}$), so that an allowance parameter $\beta$ can be efficiently estimated in many practical applications.

Another thing to address is the fact that the upper bound $u(\cdot, \cdot)$, the allowance $\beta$, and the coefficients for the sigmoid are dependent on various things, such as, the dissimilarity measure $A(\cdot, \cdot)$, and in the case of the sigmoid coefficients, the number of categories $K$. However, to avoid messy notation we suppress this dependence, which should not cause any confusion in subsequent sections.
At the present time, a general family of MCDMs have been presented. The only requirement is a prescribed dissimilarity measure, and a corresponding upper bound. Once these are chosen, the Monte Carlo simulations discussed in this chapter can be conducted, so that an allowance $\beta$ can be computed given a desired value of $\text{ARL}_0$. In the next two sections, specific change detectors are developed based on two dissimilarity measures: the Kullback Leibler (KL) divergence, and Cohen’s $h$ score.

4.5 Specific dissimilarity measures

In this section the KL divergence and Cohen’s $h$ score are chosen as dissimilarity measures. This results in two specific MCDMs that are implemented on synthetic and real-data sets in Sections 4.7-4.8.

4.5.1 KL divergence

First, the KL divergence introduced in Kullback and Leibler (1951) is chosen for the dissimilarity measure, which is a widely used non-symmetric dissimilarity measure characterizing how one probability distribution deviates from another. Suppose the data $x_{1:t}$ has been observed, and that $\hat{p}_t$ and $\hat{p}_t$ have been recursively computed via the update equations given in Section 4.3. Then a plug-in estimate for the KL divergence, between two multinomial distributions being monitored by adaptive and static MLEs, is given by

$$A(\hat{p}_t, \hat{p}_t) = \text{KL}(\hat{p}_t \| \hat{p}_t) = \sum_{i \in S} \hat{p}_t^{(i)} \log \left( \frac{\hat{p}_t^{(i)}}{\hat{p}_t^{(i)}} \right),$$

which is zero only when $\hat{p}_t = \hat{p}_t$. For brevity, denote the KL divergence at time $t$ between the adaptive and static probability vectors by $\kappa_t$.

A fairly conservative upper bound for $\kappa_t$ is given by

$$\kappa_t \leq u(\hat{p}_t, \hat{p}_t) = K \left\| \hat{p}_t / \sqrt{\hat{p}_t} \right\| \infty^2,$$  \hspace{1cm} (4.17)

58
where all operations on the vectors $\tilde{p}_t$ and $\hat{p}_t$ appearing on the right hand side of Equation (4.17) are applied component-wise, that is, Equation (4.17) is equivalent to

$$\kappa_t \leq K \left( \max_{i \in \{1, \ldots, K\}} \frac{\tilde{P}_t^{(i)}}{\hat{P}_t^{(i)}} \right)^2. $$

The proof of the bound in Equation (4.17) is provided in Appendix A.3, and hinges on the fact that $x > \log x$ for $x > 0$. The corresponding adaptive threshold is given by

$$\varepsilon_t = \beta \left( K \left\| \frac{\tilde{P}_t}{\sqrt{\hat{P}_t}} \right\|_\infty^2 \right),$$

for the allowance parameter $\beta \in (0, 1)$.

Figure 4.3 shows a plot of the KL divergence $\kappa_t$, as well as the adaptive threshold $\varepsilon_t$ for a particular choice of $\beta$. Results obtained from the Monte Carlo simulations, as discussed in Section 4.4.2, for $\beta$ are provided shortly, and for illustration purposes a subjectively chosen $\beta$ suffices. From the figure we see that $\kappa_t$ is near zero before and after the changepoints, and when the stream experiences a change $\kappa_t$ spikes above the adaptive threshold $\varepsilon_t$. 

Figure 4.3: Plots of the KL divergence $\kappa_t$ (solid line) and the adaptive threshold $\varepsilon_t$ (dashed line) using an allowance parameter $\beta = 0.025$. **Left:** a single changepoint at time $\tau = 600$. **Right:** two changepoints at times $\tau_1 = 400$ and $\tau_2 = 800$. 

59
Figure 4.4: Plots of ARL\textsubscript{0} for various values of the allowance $\beta$ and number of categories $K$, using the KL divergence as the dissimilarity measure. Any $\beta > 0.1$ resulted in the MCDM never flagging a false positive, i.e., the detector attained the maximum ARL\textsubscript{0} = 5000. **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 $K$.

The Monte Carlo simulations discussed in Section 4.4.2 are now conducted using the KL divergence as the dissimilarity measure, and upper bound provided in Equation (4.17). For each $\beta \in \mathcal{G}$ an ARL\textsubscript{0} was estimated by averaging over 250 Monte Carlo replicates, where the streams in each simulation were comprised of 5,000 observations and no changepoints were present. The results for values of $K \in \{3, 6, 10, 25, 50, 100\}$ are shown in Figure 4.4. Any allowance parameter $\beta > 0.1$ achieved a perfect ARL\textsubscript{0} of 5,000 and are omitted from the plots. It is evident that any $\beta > 0.1$, for all $K$ considered, will result in a large ARL\textsubscript{0}. However, larger values of $\beta$ will result in the method taking longer to detect any true changepoints, or failing to detect the changepoint entirely.

From Figure 4.4, we see that the curves are similar for each value of $K$. Thus, instead of having a set of coefficients for each value of $K$, the estimated coefficients are averaged, resulting in a single set of coefficients. This results in $c_1 = 5000$, $c_2 = 0.023$ and $c_3 = 0.001$ for the fitted sigmoid, using the KL divergence as a dissimilarity measure. As an example, if an ARL\textsubscript{0} of 1,000 is desired, Equation (4.16) yields an allowance of $\beta = 0.021614.$
4.5.2 Cohen’s $h$

Cohen (1988) discusses power analysis in various applications, and specifically considers finding statistically significant differences between proportions (Cohen, 1988, Chapter 6). Given two proportions $p^{(1)}, p^{(2)} \in [0, 1]$, the non-directional Cohen’s $h$ score is defined as

$$H \left( p^{(1)}, p^{(2)} \right) = 2 \left| \arcsin \sqrt{p^{(1)}} - \arcsin \sqrt{p^{(2)}} \right|. \quad (4.18)$$

If the difference between the two proportions were used, instead of the non-linear transformation presented in Equation (4.18), then equal differences in proportions are not necessarily, what Cohen termed, ‘equally detectable’. That is, depending on the true population proportions, equal differences between them may result in statistical tests that have different powers. Although this thesis is not directly concerned with power, we still favor using Cohen’s $h$ score as a dissimilarity measure as opposed to $|p^{(1)} - p^{(2)}|$. Furthermore, the non-directional $h$ score is preferred, as it will be extended to the multinomial distribution, and we do not want any cancellation between category probabilities occurring.

As this chapter is concerned with the development of change detection strategies for categorical data streams, we extend to the multinomial setting by considering the averaged non-directional Cohen’s $h$ score, simply referred to as Cohen’s $h$ score, defined by

$$A (\hat{p}_t, \tilde{p}_t) = \frac{2}{K} \sum_{i \in S} \left| \arcsin \sqrt{\hat{p}^{(i)}_t} - \arcsin \sqrt{\tilde{p}^{(i)}_t} \right|, \quad (4.19)$$

where, as before, we are comparing the adaptive and static estimates $\tilde{p}_t$ and $\hat{p}_t$. Denote the right hand side of Equation (4.19) by $\varphi_t$. It is worth noting that Equation (4.19), unlike the KL divergence, is a valid metric. That is, in addition to Equations (4.11)-(4.12) being satisfied, the symmetry and
triangle inequality conditions

\[ A(\hat{p}_t, \hat{p}_t) = A(\hat{p}_t, \hat{p}_t), \]
\[ A(\hat{p}_t, \hat{p}_t) \leq A(\hat{p}_t, x) + A(x, \hat{p}_t), \]

for any \( x \in \mathbb{S}^K \), are also satisfied.

The MCDMs require an upper bound for \( \varphi_t \). Sun and Chen (2016) proved an inequality, which asserts that for any \( x \in (0, 1) \)

\[ \arcsin x < \frac{8x}{3\sqrt{1 - x^2 + \sqrt{25 + bx^2}}}, \quad b = \frac{256 - \pi^2}{\pi^2}, \quad (4.20) \]

where the constant \( b \) provides the sharpest upper bound. An outline of the proof provided in Sun and Chen (2016) is given in Appendix A.3. Using the inequality in Equation (4.20), as well as the triangle inequality, results in the upper bound

\[ u(\hat{p}_t, \hat{p}_t) = \frac{16}{K} \sum_{i \in S} \left[ \frac{1}{3\sqrt{1 - \hat{p}_t(i) + \sqrt{25 + b\hat{p}_t(i)}}} + \frac{\sqrt{\hat{p}_t(i)}}{3\sqrt{1 - \hat{p}_t(i) + \sqrt{25 + b\hat{p}_t(i)}}} \right], \quad (4.21) \]

which is valid for Cohen’s \( h \) score. The associated adaptive threshold is given by

\[ \varepsilon_t = \beta u(\hat{p}_t, \hat{p}_t), \]

where \( u(\cdot, \cdot) \) is now defined by Equation (4.21). Note that, unlike the upper bound for \( \kappa_t \) shown in Equation (4.17), computation of the threshold requires summing over the number of categories. However, since the number of categories does not increase over time, this should not be a concern in practical applications. Figure 4.5 plots Cohen’s \( h \) score \( \varphi_t \) and the adaptive threshold \( \varepsilon_t \) using an allowance parameter \( \beta = 0.25 \). Similar to Figure 4.3, in each segment \( \varphi_t \) is near zero; conversely, when the stream encounters a changepoint Cohen’s \( h \) score spikes above the adaptive threshold.
Figure 4.5: Plots of the Cohen’s $h$ score $\varphi_t$ (solid line) and the adaptive threshold $\epsilon_t$ (dashed line) with an allowance parameter $\beta = 0.25$. **Left:** a single changepoint at time $\tau = 600$. **Right:** two changepoints at times $\tau_1 = 400$ and $\tau_2 = 800$.

Figure 4.6 displays the results of Monte Carlo simulations that provide insight into how $\beta$ affects $\text{ARL}_0$. Similar comments that were made for Figure 4.4 apply here – with one important difference. Unlike Figure 4.4, where the curves were similar for each $K$ and it was argued that one set of coefficients will suffice, using Cohen’s $h$ score as a dissimilarity measure results in curves which are more sensitive to the value of $K$. A reason for this is because the upper bound for the KL divergence (Equation (4.17)) is far more conservative than the upper bound used for Cohen’s $h$ score (Equation (4.21)). Since the upper bound for Cohen’s $h$ score is sharper, the allowance parameter assumes a larger range of values, resulting in sigmoids whose coefficients may be more variable. Due to this, instead of averaging over the coefficients for each $K$, the coefficients provided in Table 4.1 are used when fitting the sigmoid in Equation (4.15).

A complete discussion of the MCDMs, for two specific dissimilarity measures, has now been provided. In upcoming sections the MCDMs are compared to various methods in a large simulation study. Additionally, the MCDMs are implemented on nearly 40 million network events from Imperial College London’s computer network.
Figure 4.6: Plots of ARL₀ for various values of the allowance β and number of categories K, using Cohen’s h score as the dissimilarity measure. Any β > 0.45 resulted in the MCDM never flagging a false positive, i.e., the detector attained the maximum ARL₀ = 5000. **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.

<table>
<thead>
<tr>
<th>K</th>
<th>3</th>
<th>6</th>
<th>10</th>
<th>25</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>c₁</td>
<td>4994.22</td>
<td>5000.42</td>
<td>5001.84</td>
<td>4999.65</td>
<td>4997.47</td>
<td>4997.17</td>
</tr>
<tr>
<td>c₂</td>
<td>0.07</td>
<td>0.09</td>
<td>0.11</td>
<td>0.14</td>
<td>0.18</td>
<td>0.22</td>
</tr>
<tr>
<td>c₃</td>
<td>0.03</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Table 4.1: Sigmoid coefficients used to compute the allowance parameter β, given a desired value for ARL₀, when Cohen’s h score is used as a dissimilarity measure.

4.6 Comparison methods

This section discusses four methods that the MCDMs will be compared to in the simulation study of Section 4.7. Two methods are non-sequential 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 MCDMs. The other two detectors are sequential methods suitable for streaming data, and are commonly used in applications.

4.6.1 Bernoulli CUSUM

A popular control chart, introduced in Section 2.4.2, is the CUSUM chart of Page (1954). Modifications of the CUSUM chart have been provided in Reynolds Jr and Stoumbos (1999, 2000), and makes the chart suitable for
streams of Bernoulli random variables. These modifications are summarized below, where the notation used is similar to that appearing in Reynolds Jr and Stoumbos (1999).

Suppose that when a stochastic process is in-control, i.e., behaving ‘normally’, the Bernoulli distribution is governed by $p(0) \in [0, 1]$. The Bernoulli CUSUM chart aims to detect a change (either an increase or decrease) in $p(0)$ to some out-of-control parameter value. For an increase and decrease in $p(0)$, respectively, denote the out-of-control parameters by $p^+$ and $p^−$.

In the statistical quality control literature, detecting only an increase in $p(0)$, where the terms fraction ‘conforming’ and ‘non-conforming’ are used (Montgomery, 2007), is a primary concern. This is natural in applications where $p(0)$ represents the probability of finding a defect in a manufacturing process. Clearly a decrease in $p(0)$ is not a cause for concern. However, since this chapter is concerned with monitoring a categorical data stream for multiple changepoints, increases and decreases in $p(0)$ must be monitored for.

The Bernoulli CUSUM chart sequentially monitors statistics defined by

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

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

where $\gamma^+$ and $\gamma^−$ are typically referred to as the reference values (Reynolds Jr and Stoumbos, 1999), and $x_t \in \{0, 1\}$ is a realization of the Bernoulli random variable $X_t$. In the literature, $\gamma^+$ and $\gamma^−$ are chosen by appealing to the sequential probability ratio test (Ghosh and Sen, 1991) representation of the chart. This leads to

$$
\gamma^* = -\frac{\log \left( \frac{1-p^*}{1-p(0)} \right)}{\log \left( \frac{p^*(1-p(0))}{p(0)(1-p^*)} \right)} ,
$$

(4.22)

where, for brevity, $* \in \{+,-\}$. The Bernoulli CUSUM chart flags a change in the process whenever $|B_t^*| > h^*$, where $h^*$ are suitably chosen thresholds. These thresholds remain fixed until a change is flagged by the detector, where an intervention procedure such as the grace period discussed in Section 2.4.4 is then implemented to reinitialize the chart.
Similar to tuning the allowance parameter $\beta$, the thresholds $h^*$ are commonly chosen to adhere to a desired value of $\text{ARL}_0$. For the Bernoulli CUSUM chart, as with the standard CUSUM chart, guarantees on the value of $\text{ARL}_0$ are challenging to construct. As a thorough investigation into setting the thresholds for the Bernoulli CUSUM chart is not in the scope of this thesis, 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 specify the thresholds. Subsequently the Bernoulli CUSUM chart will be referred to as the BCUSUM chart.

Notice that Equation (4.22) requires the in-control parameter $p^{(0)}$, as well as the out-of-control parameters $p^*$ to be specified. The in-control parameter can be approximated using a burn-in period; however, the out-of-control values are difficult to prescribe in the multiple changepoint setting. This is because they are typically chosen to detect a change of a certain magnitude, and in the multiple changepoint scenario multiple changepoints of varying magnitudes will exist. Furthermore, depending on the value of $p^{(0)}$ there may be issues in choosing $p^*$, as it may be too close to the boundary of $[0, 1]$. These issues are addressed in Section 4.7.1.

The BCUSUM chart is not suitable for detecting multiple changes in a multinomial distribution. Thus, when implementing the chart the binarization of the categorical process given by Equation (4.2) is used. The binarization is a stream of vectors, whereas the proposed chart is univariate. The chart is therefore implemented on each component of the vectors appearing in Equation (4.2), requiring $K$ runs of the BCUSUM method given a single stream, which increases the amount of computation needed to process the stream.

Recall that multivariate control charts, such as the multinomial charts discussed in Section 4.1, exist. See Bersimis et al. (2007) for a thorough discussion. However, much of the literature focuses on continuous-valued streams, and are not directly relevant to this chapter. Moreover, as mentioned in Section 4.1, it is not practical to specify the direction of the drift in the out-of-control parameters, for each category and every changepoint. We therefore follow the advice of Ryan et al. (2011) in Section 4.7, who recommend using $K$ BCUSUM charts to replace a multinomial CUSUM chart.
4.6.2 CPM

The R package \texttt{cpm} (Ross et al., 2015) provides various changepoint models for the detection of multiple changepoints in a univariate data stream, where either parametric, or non-parametric assumptions are placed on the stream’s distribution. Each method implemented in the \texttt{cpm} package 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 chosen test statistic $D_t$ and threshold $h_t$.

Many methods implemented in the \texttt{cpm} package could be used as comparison methods for the MCDMs. However, a complete contrast of the MCDMs to the \texttt{cpm} package is not in the scope of this thesis. Therefore, in Section 4.7 the computation of $D_t$ is based on the \textit{Mann-Whitney test} (Hawkins et al., 2003; Pettitt, 1979). This is a non-parametric test that has been extended to the streaming setting in Ross et al. (2011). Similar to our approach, in Ross et al. (2011) the sequence of thresholds are computed using Monte Carlo simulations. This method, subsequently referred to as CPM, is sufficient for the synthetic simulations analyzed in Section 4.7. Further, similar to the BCUSUM chart, the CPM method is implemented on the binarizations of the data stream, requiring $K$ passes over the categorical stream.

4.6.3 ECP

A non-sequential alternative, capable of handling multivariate data, is provided in the R package \texttt{ecp} (James and Matteson, 2013). This package implements a method that can detect multiple changepoints using a divisive or agglomerative approach. The divisive method sequentially looks for changepoints by applying a bisection technique; whereas, the agglomerative method computes an optimal partitioning. The divisive approach is used subsequently, as it has a sequential underpinning, and this method will be referred to as ECP.

The divisive methodology implemented in \texttt{ecp} is based on the work of Matteson and James (2014), and assumes that the multivariate observations are independent with a finite $\vartheta$th absolute moment for some $\vartheta \in (0, 2]$. A
dissimilarity measure is then used to detect non-stationarity in the data, and is based on the energy statistic proposed in Szekely and Rizzo (2005). The significance of each detected changepoint is assessed using a permutation test, which requires a specified number of maximum permutations $R$ to be prescribed. A significance level $\alpha^{ECP} \in (0, 1)$ for the test is also required.

The divisive approach has quadratic computational complexity, making it ill-suited for the streaming paradigm. Nonetheless, as this method utilizes multiple passes over the data, it provides a good benchmark for the MCDMs. This method, unlike CUSUM and CPM, can handle multivariate data and can be directly applied to the data stream in Equation (4.2), without having to consider each component of the binarization vectors separately.

4.6.4 PELT

The pruned exact linear time (PELT) method introduced in Killick et al. (2012) is provided in the R package `changepoint` (Killick and Eckley, 2014). PELT is able to detect multiple changepoints in a non-stationary time series, and requires several passes over the data. Therefore, similar to ECP, PELT will also not be suitable for many streaming applications, but provides a good benchmark when investigating the performance of the MCDMs.

PELT requires a cost function to be specified. Once one is provided, the method returns an optimal segmentation of the data by modifying the optimal partitioning algorithm of Jackson et al. (2005). Under certain assumptions, such as the number of changepoints increasing linearly with the number of observations, PELT has a linear computational cost. This improves on the optimal partitioning and ECP methods, which exhibit quadratic computational costs; however, if PELT’s assumptions are not satisfied the complexity is also quadratic in the length of the data. The `changepoint` package is designed for univariate time series, thus PELT is implemented over each component of the binarization of the categorical process given in Equation (4.2). If desired, a multivariate cost function could be prescribed, allowing PELT to analyze the categorical stream in Equation (4.1); however, using the binarization is sufficient for the simulations presented in this chapter.
Table 4.2: A summary of the methods and their characteristics, including: if the methods are suitable for the streaming paradigm (Streaming), if they are implemented on the binarizations or the categorical data stream (Binarization), if they have to be run once given a single data stream, or have to be run over $K$ streams, each stream being comprised of the components of binarization vectors (1-Stream), and if there is a publicly available R package that is used in the synthetic simulations (R package).

Table 4.2 displays characteristics of the MCDMs and the comparison methods, and includes: whether or not a method is suitable for streaming data, if a method is implemented on the categorical stream given in Equation (4.1) or the binarization provided in Equation (4.2), if a method is run over a single stream, or needs to be implemented over $K$ streams, which are comprised of the components of the binarization vectors, and if a publicly available R package is used to obtain the results presented in the next section.

For methods that have to be run over each component of the binarizations, adjustments have been made in regards to multiple comparisons. For example, in each simulation the value reported for $\text{ARL}_0$ was taken as the minimum false positive flagged by the detector over the $K$ streams. This way of reporting the run length is logical in practice, as the first detection flagged would be cause for inspection. It is worth noting that these adjustments do not align with classical hypothesis testing (e.g. Bonferonni corrections). This is because multiple comparisons in the streaming context is non-trivial, as the approximations for $\text{ARL}_0$ typically rely on complex methods (such as the corrected diffusion approximations used in BCUSUM), making it difficult to apply standard corrections to guarantee the correct coverage.
4.7 Synthetic examples

In this section the MCDMs are implemented on synthetic data streams and compared to the methods discussed in Section 4.6. The detectors are compared using the performance measures ARL$_0$, ARL$_1$, CCD and DNF, which were introduced in Section 2.4.3. Section 4.7.1 discusses the experimental design of the simulations, where a way of synthetically generating changepoints so streaming and non-streaming algorithms may be fairly compared is also introduced. Section 4.7.2 analyzes the results.

4.7.1 Experimental design

All change detectors are implemented on 2,000 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 $K \in \{3, 6, 10, 25\}$ and $m \in \{0, 1, 5, 10\}$.

The generation of the changepoints in each Monte Carlo simulation warrants discussion. Since sequential and non-sequential detectors are being compared, care must be taken in what constitutes a ‘true detection’. It is evident that sequential detectors will never correctly flag a changepoint before one occurs (as they cannot revisit historical data); however, non-sequential detectors may accurately detect a change before the actual time-stamp of the changepoint. These detections should not be ignored; hence, a ‘bubble’ is defined around the time-stamp of each changepoint. Precisely, suppose $\tau$ is a changepoint location and let $\xi \in \mathbb{Z}^+$. Then for non-sequential detectors, any detection made in the interval $[\tau - \xi, \tau + \xi]$ is considered a true detection. Conversely, the true detection region for sequential detectors is taken as $[\tau, \tau + \xi]$. Any detector that does not flag a change within $\xi$ time-steps of the true changepoint location (on either side of $\tau$ for non-sequential detectors) is deemed to have missed the detection.

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

\[ \tau_1 = 2\xi + \rho + \nu_1, \]
\[ \tau_k = \tau_{k-1} + 2\xi + \rho + \nu_k \quad k = 2, \ldots, m, \]

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

Under the proposed scheme the expected length between any pair of consecutive changepoints is \( 2\xi + \rho + r \). Thus, given an average length between changepoints \( L > 2\xi + \rho \), the rate parameter of the Poisson distribution is chosen as \( r = L - 2\xi - \rho \). The length of the stream \( S \) is then taken to be the smallest integer multiple of 2,500 satisfying \( S > m(2\xi + \rho + r) \). When \( m = 0 \) or \( m = 1 \) the length of the stream is taken as \( S = 5000 \). Further, \( (\xi, \rho, L) = (50, 20, 500) \) are chosen. Note that these parameters are only used when generating changepoints, and are not control parameters for any particular method.

For the MCDMs the length of the burn-in and grace period regions are chosen as \( B = \frac{S}{10} \) and \( G = 100 \). Care was taken so that the values for the burn-in and grace period regions were used similarly in the other change detection methods. As an example, in the R packages \texttt{changepoint} and \texttt{ecp} the scalar \( G \) was used as the minimum segment length between changepoints. Moreover, the MCDMs and the CPM and BCUSUM methods allow for a desired value of ARL\(_0\) to be specified. This is chosen as 2,000, resulting in an allowance parameter for the MCDM being \( \beta = 0.022594 \) when the KL divergence is used, and \( \beta \) being a vector, which is based on the values of \( K \) when Cohen’s \( h \) score is used as the dissimilarity measure. The values for the allowance can be computed using Table 4.1. The step-size used in the gradient descent step for the MCDMs is chosen as \( \eta = 10^{-3.5} \).

The BCUSUM chart must be implemented over \( K \) data streams; hence the in-control and out-of-control parameters are actually vectors of length \( K \), and will be denoted by \( \mathbf{p}^{(0)}, \mathbf{p}^+ \) and \( \mathbf{p}^- \). The in-control vector \( \mathbf{p}^{(0)} \) is estimated
using a burn-in period of length $B = \frac{S}{10}$, and its components are truncated to $[p_{\min}, p_{\max}]$. This ensures that no component is too close to the boundary of $[0, 1]$. The out-of-control vectors are then taken as

$$p^+ = \min \{1, p^{(0)} + \zeta\}, \quad p^- = \max \{0, p^{(0)} - \zeta\},$$

where $\zeta \in \{0.01, 0.1, 0.25, 0.5\}$ determines the magnitude of the change one is looking to detect, and the minimum and maximum are applied component-wise. The truncation parameters, which ensures that no probability is too close to the boundary, are given by $(p_{\min}, p_{\max}) = (0.001, 0.999)$.

Lastly, the maximum number of allowed permutations for the ECP method was set to $R = 30$, and the significance level for the permutation test was chosen as $\alpha_{ECP} = 0.05$. For methods whose implementations have R packages, all parameters not mentioned are chosen according to the package defaults.

4.7.2 Results

The results from the synthetic simulations are summarized in Tables 4.3-4.5. Tables 4.3 and 4.4 display the estimates for $\text{ARL}_0 (m = 0)$, $\text{ARL}_1 (m = 1)$ and CCD and DNF ($m > 1$) for all the methods. In Table 4.3, KL and COHEN refer to MCDMs that used the KL divergence and Cohen’s $h$ score for dissimilarity measures. When $m > 1$ the entries in Tables 4.3 and 4.4 are tuples of the form (CCD, DNF). For BCUSUM the value appearing in parentheses in Table 4.4 is the value of $\zeta$ used in the simulations. When estimating $\text{ARL}_1$ only detections made within the detection regions contributed to the value of $\text{ARL}_1$, that is, if a detector missed a changepoint it did not worsen the estimate for $\text{ARL}_1$. Therefore, the proportion of simulations where the changepoint was correctly identified should also be investigated. These are provided in Table 4.5.

Consider the results for $\text{ARL}_0$ appearing in the upper sections of Tables 4.3 and 4.4. From the tables we see that the MCDMs reported better $\text{ARL}_0$ values only when compared to the CUSUM method (except when $K = 25$ and the KL divergence was used for dissimilarity). However, recall that we
Table 4.3: Estimated performance measures returned by the MCDMs, CPM, ECP and PELT, for every $K \in \{3, 6, 10, 25\}$ used in the simulations. When $m = 0$ and $m = 1$ the table displays, respectively, results for $\text{ARL}_0$ and $\text{ARL}_1$. For $m > 1$ the entries in the table are tuples of the form $(\text{CCD}, \text{DNF})$. Larger values for $\text{ARL}_0$, CCD and DNF and lower values of $\text{ARL}_1$ is indicative of a method performing favorably. KL and COHEN refer to the MCDM that used the KL divergence and Cohen’s $h$ score as dissimilarity measures.

<table>
<thead>
<tr>
<th>$m$</th>
<th>KL</th>
<th>COHEN</th>
<th>CPM</th>
<th>ECP</th>
<th>PELT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3144.36</td>
<td>1920.90</td>
<td>4354.73</td>
<td>4931.77</td>
<td>5000.00</td>
</tr>
<tr>
<td>6</td>
<td>2343.87</td>
<td>1959.28</td>
<td>4177.44</td>
<td>4910.33</td>
<td>5000.00</td>
</tr>
<tr>
<td>10</td>
<td>1739.37</td>
<td>1957.50</td>
<td>4350.91</td>
<td>4918.65</td>
<td>5000.00</td>
</tr>
<tr>
<td>25</td>
<td>859.30</td>
<td>2010.79</td>
<td>4790.01</td>
<td>4905.93</td>
<td>5000.00</td>
</tr>
<tr>
<td>1</td>
<td>22.78</td>
<td>23.53</td>
<td>12.20</td>
<td>2.11</td>
<td>1.23</td>
</tr>
<tr>
<td>6</td>
<td>20.85</td>
<td>23.55</td>
<td>19.33</td>
<td>2.03</td>
<td>1.45</td>
</tr>
<tr>
<td>10</td>
<td>21.94</td>
<td>24.88</td>
<td>27.03</td>
<td>1.96</td>
<td>3.05</td>
</tr>
<tr>
<td>25</td>
<td>26.44</td>
<td>30.89</td>
<td>33.05</td>
<td>2.16</td>
<td>9.10</td>
</tr>
<tr>
<td>5</td>
<td>(0.96, 0.86)</td>
<td>(0.94, 0.70)</td>
<td>(0.69, 0.79)</td>
<td>(1.00, 0.99)</td>
<td>(0.61, 0.99)</td>
</tr>
<tr>
<td>6</td>
<td>(0.97, 0.82)</td>
<td>(0.96, 0.72)</td>
<td>(0.32, 0.51)</td>
<td>(1.00, 0.99)</td>
<td>(0.20, 0.95)</td>
</tr>
<tr>
<td>10</td>
<td>(0.96, 0.75)</td>
<td>(0.95, 0.73)</td>
<td>(0.12, 0.29)</td>
<td>(1.00, 0.99)</td>
<td>(0.05, 0.86)</td>
</tr>
<tr>
<td>25</td>
<td>(0.88, 0.57)</td>
<td>(0.92, 0.75)</td>
<td>(0.01, 0.05)</td>
<td>(1.00, 0.99)</td>
<td>(0.00, 0.14)</td>
</tr>
<tr>
<td>10</td>
<td>(0.96, 0.88)</td>
<td>(0.94, 0.76)</td>
<td>(0.70, 0.80)</td>
<td>(1.00, 1.00)</td>
<td>(0.60, 1.00)</td>
</tr>
<tr>
<td>6</td>
<td>(0.97, 0.85)</td>
<td>(0.95, 0.77)</td>
<td>(0.32, 0.51)</td>
<td>(1.00, 1.00)</td>
<td>(0.18, 0.97)</td>
</tr>
<tr>
<td>10</td>
<td>(0.96, 0.81)</td>
<td>(0.96, 0.78)</td>
<td>(0.13, 0.30)</td>
<td>(1.00, 1.00)</td>
<td>(0.04, 0.91)</td>
</tr>
<tr>
<td>25</td>
<td>(0.88, 0.64)</td>
<td>(0.92, 0.79)</td>
<td>(0.01, 0.06)</td>
<td>(1.00, 1.00)</td>
<td>(0.00, 0.18)</td>
</tr>
</tbody>
</table>

specified a desired value of $\text{ARL}_0 = 2000$ for these simulations. Averaging over $K$, the MCDMs report a value of 2021.73 when using the KL divergence, and 1962.12 when using Cohen’s $h$ score as the dissimilarity measure. These are close to the desired value, and reinforces the validity of our approach for tuning the allowance parameters.

Examine the $\text{ARL}_1$ results displayed in the second sections of Tables 4.3 and 4.4. In terms of detection delays, the MCDMs reported better values only when compared with the CPM method (for $K > 6$). Although, relative to the length of the data streams, the MCDMs are reporting values similar to the other methods. Furthermore, analysis of Table 4.5 shows that the MCDMs were able to correctly detect the change in a large proportion of the simulations. Also notice that when $K = 25$ the CPM and PELT methods, respectively, only correctly detected the changepoint 9% and 23.5% of the
<table>
<thead>
<tr>
<th></th>
<th>BCUSUM(0.01)</th>
<th>BCUSUM(0.10)</th>
<th>BCUSUM(0.25)</th>
<th>BCUSUM(0.50)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(m = 0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1307.16</td>
<td>1399.68</td>
<td>1584.99</td>
<td>1653.98</td>
</tr>
<tr>
<td>6</td>
<td>1267.62</td>
<td>1340.74</td>
<td>1410.92</td>
<td>1476.76</td>
</tr>
<tr>
<td>10</td>
<td>1312.09</td>
<td>1318.60</td>
<td>1377.61</td>
<td>1500.90</td>
</tr>
<tr>
<td>25</td>
<td>1282.66</td>
<td>1300.60</td>
<td>1365.82</td>
<td>1441.71</td>
</tr>
<tr>
<td>(m = 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>18.89</td>
<td>12.45</td>
<td>9.32</td>
<td>8.38</td>
</tr>
<tr>
<td>6</td>
<td>16.14</td>
<td>11.00</td>
<td>9.16</td>
<td>9.05</td>
</tr>
<tr>
<td>10</td>
<td>14.43</td>
<td>9.91</td>
<td>8.52</td>
<td>8.86</td>
</tr>
<tr>
<td>25</td>
<td>10.49</td>
<td>7.85</td>
<td>7.10</td>
<td>7.51</td>
</tr>
<tr>
<td>(m = 5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(0.63, 0.06)</td>
<td>(0.77, 0.08)</td>
<td>(0.78, 0.08)</td>
<td>(0.76, 0.08)</td>
</tr>
<tr>
<td>6</td>
<td>(0.39, 0.04)</td>
<td>(0.57, 0.06)</td>
<td>(0.56, 0.06)</td>
<td>(0.53, 0.06)</td>
</tr>
<tr>
<td>10</td>
<td>(0.28, 0.03)</td>
<td>(0.44, 0.05)</td>
<td>(0.41, 0.05)</td>
<td>(0.38, 0.05)</td>
</tr>
<tr>
<td>25</td>
<td>(0.18, 0.03)</td>
<td>(0.24, 0.03)</td>
<td>(0.22, 0.03)</td>
<td>(0.20, 0.03)</td>
</tr>
<tr>
<td>(m = 10)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(0.64, 0.07)</td>
<td>(0.77, 0.09)</td>
<td>(0.77, 0.09)</td>
<td>(0.76, 0.08)</td>
</tr>
<tr>
<td>6</td>
<td>(0.39, 0.05)</td>
<td>(0.57, 0.07)</td>
<td>(0.56, 0.07)</td>
<td>(0.53, 0.07)</td>
</tr>
<tr>
<td>10</td>
<td>(0.28, 0.04)</td>
<td>(0.44, 0.06)</td>
<td>(0.42, 0.06)</td>
<td>(0.39, 0.05)</td>
</tr>
<tr>
<td>25</td>
<td>(0.18, 0.03)</td>
<td>(0.25, 0.04)</td>
<td>(0.23, 0.04)</td>
<td>(0.21, 0.03)</td>
</tr>
</tbody>
</table>

Table 4.4: Estimated performance measures, for every \(K \in \{3, 6, 10, 25\}\), for BCUSUM. When \(m = 0\) and \(m = 1\) the table displays, respectively, results for ARL\(_0\) and ARL\(_1\). For \(m > 1\) the entries in the table are tuples of the form (CCD, DNF). Larger values for ARL\(_0\), CCD and DNF and lower values of ARL\(_1\) is indicative of a method performing favorably. The value in parenthesis corresponds to the value of \(\zeta\) used in the simulations.

Time; whereas, the MCDMs were able to correctly identify nearly 82% (KL) and 92.4% (COHEN) of the changepoints. For both ARL\(_0\) and ARL\(_1\) the best performance is typically obtained by the non-sequential detectors, which is not surprising as these detectors utilize multiple passes over the data.

The bottom two sections of Tables 4.3 and 4.4 show results for CCD and DNF, corresponding to the multiple changepoint setting, and highlights the advantages of our methodology. As seen from the tables, the MCDMs outperform all of the sequential detectors in terms of CCD and DNF (omitting two values of DNF when \(K = 3\) and Cohen’s \(h\) score was used), and even outperforms PELT in terms of CCD (and DNF when \(K = 25\)). The only method that consistently outperforms the MCDMs is the ECP method. However, recall that this method has quadratic time complexity, making it unsuitable for streaming data, and the method has only been included to serve as a benchmark. To strengthen this argument, when \((S, K, m) = (7500, 25, 10)\) the ECP method took, on average, over 20 minutes to run per simulation;
Table 4.5: For simulations with a single changepoint, the proportion of simulations where a method detected the changepoint within its detection region are displayed. Values near one indicates that a method has correctly identified the changepoint in a large number of simulations. KL and COHEN, respectively, refer to the MCDM with KL divergence and Cohen’s h score used as dissimilarity measures. The value in parentheses for BCUSUM corresponds to the value of $\zeta$ – which defines the out-of-control parameters of the chart.

whereas, the MCDMs performed similarly, taking roughly 2 seconds per simulation.

The synthetic simulations provide several rewarding conclusions. First, the MCDMs provide comparable performance in terms of the average run lengths, and provide significant improvements in the multiple changepoint setting, which is a focal point of this thesis. Only the ECP method consistently outperforms the MCDMs, but is an unfair comparison due to its high computational demands; whereas, the MCDMs are suitable for streaming data and do not sacrifice much in terms of detection power. Second, given a prescribed $ARL_0$, our proposed solution for choosing the allowance $\beta$ leads to the MCDMs approximately adhering to the desired run length. In the next section the MCDMs are implemented on a real-world data stream from Imperial College London’s computer network.

4.8 TCP port illustration

Lewis (2008) has speculated that cyber related crimes cost approximately $600 billion per annum. The ability to dynamically monitor computer net-
work traffic in real-time for malicious activity is therefore a pressing concern. In this section a data stream consisting of nearly 40 million computer network events is monitored for changes. The data is introduced in Section 4.8.1, and the MCDMs are implemented on the stream in Section 4.8.2 to highlight that the methodology is suitable for high-frequency data streams occurring in practice. The network traffic has no ground truth labelling of malicious activity, which is frequently the case in cyber-security based applications (Anagnostopoulos, 2018). Consequently, in Section 4.8.3 the data is modified to mimic the WannaCry cyber attack (Mohurle and Patil, 2017), which took place in May 2017.

4.8.1 The data

The data stream to be analyzed consists of a sequence of network events collected from a router in Imperial College London’s 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 is assigned an integer in \([0, 1023]\), and often indicates the type of activity. Refer to Comer (2000) for a thorough treatment of TCP protocols. The purpose of this section is to highlight that the MCDMs are appropriate for large-scale, high-frequency data streams that occur in practice, and a complete investigation of the network’s activity is beyond the scope of this thesis.

A collection of 39,031,345 well-known TCP destination ports, collected over 24 hours in 2015, comprise the data stream to be analyzed. Table 4.6 displays the top ten ports present in the data, which comprise nearly 98% of the entire data stream. In consequence, the 1,024 destination ports are condensed to the top ten ports appearing in Table 4.6, and an ‘other’ category representing any TCP port not appearing in the table.

As the stream consists of over 39 million observations, previous methods that required being implemented over \(K\) streams would require processing close to 440 million observations (as there are 11 categories in the TCP stream). From a practical standpoint, any detector that monitors TCP traffic
<table>
<thead>
<tr>
<th>Port number</th>
<th>Description</th>
<th>Proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>HTTP</td>
<td>∼ 0.489</td>
</tr>
<tr>
<td>445</td>
<td>Microsoft-DS</td>
<td>∼ 0.216</td>
</tr>
<tr>
<td>443</td>
<td>HTTPS</td>
<td>∼ 0.158</td>
</tr>
<tr>
<td>88</td>
<td>Kerberos</td>
<td>∼ 0.029</td>
</tr>
<tr>
<td>389</td>
<td>LDAP</td>
<td>∼ 0.026</td>
</tr>
<tr>
<td>631</td>
<td>IPP</td>
<td>∼ 0.025</td>
</tr>
<tr>
<td>111</td>
<td>SUN remote procedure call</td>
<td>∼ 0.012</td>
</tr>
<tr>
<td>22</td>
<td>SSH</td>
<td>∼ 0.012</td>
</tr>
<tr>
<td>135</td>
<td>DCE endpoint resolution</td>
<td>∼ 0.009</td>
</tr>
<tr>
<td>139</td>
<td>NETBIOS session service</td>
<td>∼ 0.003</td>
</tr>
</tbody>
</table>

Table 4.6: The top ten well-known TCP destination ports, collected over a single router, present in Imperial College London’s computer network. These ten ports amount to roughly 98% of the observations in the data stream.

in real-time would need to be implemented directly on the router, which rules out being able to monitor each of the $K$ streams for changepoints in parallel. Further to this point, CPM took approximately 40 minutes to process one million observations, suggesting that streaming methods will have difficulty processing a day’s worth of network data if they need to be implemented over the $K$ streams. Additionally, recall that the ECP method can detect changes by analyzing a single stream; however, in Section 4.6.3 it took ECP over 20 minutes to process 7,500 observations. It is clear that ECP’s high computational complexity renders the method inapplicable to the TCP stream. Henceforth, only the MCDMs are implemented on the TCP data stream, and results are discussed in the next two sections.

4.8.2 Unmodified TCP stream

In this section the MCDMs are implemented on the TCP stream. A burn-in period of length 9,031,345 is used, resulting in 30 million destination ports being monitored for changepoints. In cyber-security applications practitioners will typically have an estimate as to how many false positives can be tolerated throughout the day, as examining detections may be costly. The value of $G$, dictating the length of the grace period, can be chosen to coin-
ciety with these concerns, as it implicitly provides a lower bound on \( ARL_0 \). Moving forward the length of the grace period is chosen as \( G = 135600 \). This results in \textit{approximate} 5 minute grace periods, as the TCP data arrive at a rate of approximately 452 per second.

Choosing the allowance parameters for the MCDMs merits discussion. Cyber analysts favor detectors that flag few false positives, or equivalently, detectors which exhibit large values of \( ARL_0 \). Since a considerable amount of data is collected daily, to meet practitioner’s demands would require an \( ARL_0 \) on the order of \textit{millions}. Indeed, to flag a false positive approximately every hour in the TCP stream would require an \( ARL_0 \) greater than 1.6 million. At the present time there appears to be no available 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, similar to those discussed in Section 4.4.2, were conducted and an allowance parameter \( \beta = 0.077 \) was computed for the KL divergence, and \( \beta = 0.158 \) for Cohen’s \( h \) score, resulting in the MCDMs having an \( ARL_0 \) roughly equal to 20,000.

Implementing the MCDMs took just over 1 hour to process the 30 million TCP destination ports. Thus, the MCDMs proposed in this chapter can process a days worth of network data in far less than 24 hours, making the family of detectors suitable for the TCP stream. Indeed, over 8,000 observations were processed per second without formally optimizing code for the MCDMs. Therefore, it is sensible to conclude that the MCDMs will be applicable to a wide-range of streaming applications, where large quantities of categorical data are generated at high-frequency.

Histograms displaying the time elapsed (in minutes) between consecutive detected changepoints are provided in Figure 4.7. The grace period \( G \) was chosen to result in approximate 5 minute grace periods, and since the data arrives irregularly from the router, we remark that detections could theoretically (and do) occur in under 5 minutes. Given our choices of \( G \) and allowance parameters we would expect the MCDMs to flag a change, on average, around every 5.74 minutes (vertical dashed lines in Figure 4.7). From the histograms we see that the detections made, more or less behave as expected, reinforc-
Figure 4.7: Histograms of the elapsed time (in minutes) between the estimated location of consecutive changepoints (denoted $\hat{\tau}_k - \hat{\tau}_{k-1}$). The dashed vertical lines is the time between detections that we would expect, on average, given the chosen values for $G$ and $\beta$ used in analyzing the TCP stream. Recall that since the data arrives irregularly, the dashed vertical lines are an approximation, and assumes that every detection is a false positive. **Left:** KL divergence. **Right:** Cohen’s $h$ score.

ing that the control parameters for the MCDMs may be chosen based on practical considerations and can be applied to real-world data streams. Note that using the KL divergence typically results in a change detector that overestimates the desired value of $\text{ARL}_0$; whereas, using Cohen’s $h$ score seems to typically underestimate the run length – agreeing with results presented in Section 4.7.2. We also remark that since the data arrives in irregular intervals, the vertical dashed line in Figure 4.7 is also an estimated value.

The data collected from the router has no ground truth accompanying it, that is, the existence and location of potential changepoints are unknown. The sole purpose of this section was to illustrate that:

- the MCDMs scale well with the size of the data, and are suitable for the sequential monitoring of streams which occur in practice, and

- choosing the MCDMs allowance parameters result in detectors that closely adhere to a constraint on $\text{ARL}_0$ (or the false positive rate).

In the next section the TCP data stream is artificially modified so that there
is a change in the sequence of destination ports.

4.8.3 WannaCry

The WannaCry cyber attack, occurring in May 2017, was estimated to have affected over 200,000 devices running the Windows’ operating system, spanning over 150 countries (Mohurle and Patil, 2017). The WannaCry attack exploited Windows’ server message block (SMB) protocol, and encrypted sensitive data on the user’s machine. The owner was then issued a ‘ransom message’, asking to be paid in Bitcoin cryptocurrency in exchange for a decryption key.

Windows’ 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 4.6. To mimic the WannaCry attack these specific ports are tampered with. Specifically, an attack begins at 11:00 and lasts until 11:10. Starting at 11:00, in minute intervals, one of the ports exhibits a burst of activity that lasts for a prespecified amount of time. This is done in such a way that the frequency of ports 445 and 139 are increased by approximately 10% in the data stream.

The values chosen for $G$ and the allowance parameters in Section 4.8.2 are also used when inspecting the infected TCP data stream. The MCDM using the KL divergence flagged two detections during the attack periods at times 11:00:29 and 11:08:10, which were not flagged when deploying the MCDM over the stream without the attack. Similarly, when using Cohen’s $h$ score, the detector flagged three changepoints during the attack period, at times 11:00:17, 11:04:10 and 11:10:24, which again were not flagged in the unmodified TCP stream. It is therefore reasonable to conclude that the MCDMs were indeed successful in detecting the attack.
4.9 Discussion

In this chapter a family of multinomial change detection methods were developed, which are applicable to data streams whose observations assume a discrete set of values. The methods rely on comparing the adaptive and static MLEs for the multinomial distribution, where temporal adaptivity was introduced in the adaptive MLEs using forgetting factors.

The MCDMs require a dissimilarity measure to be prescribed, as well as an upper bound on the dissimilarity measure to exist. The methods then detect changes based on a novel adaptive thresholding technique, which introduced an allowance parameter that can be chosen given a desired value for $\text{ARL}_0$, or equivalently, a false positive rate. We remark that other measures and upper bounds could have been considered in the construction of the MCDMs; however, the allowance parameter $\beta$ will always be tuned to approximately attain a desired value of $\text{ARL}_0$ – independent of what measure and bound were chosen. Choosing other dissimilarity measures and upper bounds would result in different coefficients for the sigmoids in Equation (4.15); however, the allowances would be tuned to approximate the same value for $\text{ARL}_0$. Hence, the allowance $\beta$ can also be considered as a ‘correction factor’ between different dissimilarity measures and upper bounds, and it is sensible to conclude that the MCDMs performance would be similar for other choices of the dissimilarity measure and upper bound.

Two specific detectors were then developed, based on the KL divergence and Cohen’s $h$ score, and the detectors were shown to perform favorably when compared to commonly used sequential and non-sequential change detectors. The MCDMs, using the KL divergence and Cohen’s $h$ score, were then deployed on nearly 40 million network events from Imperial College London’s computer network, and were successful in detecting a synthetic attack, which was similar to the WannaCry cyber attack.

This chapter assumed that the categorical observations arrived independently from the data stream. In the next chapter a first-order Markov assumption is placed on the data stream, and a change detection procedure for transition matrices is developed.
Similar to the preceding chapter, this chapter is concerned with detecting changes in categorical data streams. However, instead of assuming the categories arrive independently, the observations are assumed to obey a first-order Markov property. Consequently, a transition matrix needs to be estimated to monitor the process, and this chapter presents an innovative change detection technique for transition matrices.

The chapter proceeds as follows: Section 5.1 discusses relevant literature. Section 5.2 develops an estimator, based on forgetting factors, for transition matrices. Section 5.3 introduces an Adaptive Detection and Estimation Procedure for Transition Matrices, subsequently referred to as ADEPT-M, which is the main contribution of the chapter. Section 5.4 implements ADEPT-M on synthetic data streams, while comparing the method to the sequential change detectors discussed in Section 2.4.2. In Section 5.5 ADEPT-M is implemented on a real-world data stream comprised of web requests to a server, and is shown to be successful in detecting a common type of cyber-attack.
5.1 Literature review

Detecting changes in Markov processes has been investigated in various contexts, such as: Markov chains with finite state spaces (Yakir, 1994), hidden Markov models (Chen and Willett, 1997; Fuh, 2004), sensor networks (Raghavan and Veeravalli, 2010; Tartakovsky and Veeravalli, 2008) and in Markov-modulated time series (Dey and Marcus, 1999). However, most existing literature imposes assumptions that make the methodologies ill-suited to the streaming paradigm. For example, the references just mentioned develop detection strategies that rely on the distribution before and after the change being known. This is nonsensical in the streaming data setting, where several changepoints will occur, and prior knowledge about the post change distribution after every changepoint is unlikely to exist.

Markov transition matrices can be viewed as weighted adjacency matrices; therefore, change detection in dynamic networks is a closely related topic. See Ranshous et al. (2015) for an excellent survey. A large portion of literature in this area, similar to the references discussed in Section 4.1, introduces several parameters that are difficult to set – and have no meaningful interpretation in the multiple changepoint paradigm. For instance, Li et al. (2009) introduce several time independent control parameters when developing reward and penalty functions. In Idé and Kashima (2004), a fixed sliding window is used to compute an ‘activity vector’, and all changes in the stream are based on this vector’s departure from normal network behavior. Therefore, the detector’s ability to accurately detect changepoints relies on the length of the sliding window. As outlined in Ranshous et al. (2015), other approaches introduce a difficult to interpret threshold, and detect a change whenever some quantity monitored from the stream exceeds the threshold. Setting the threshold is a difficult task, which is exacerbated by the presence of multiple changepoints, since many techniques for specifying the threshold requires some prior knowledge of the evolution of the stream, e.g., the magnitude of the changes that will occur.
5.2 Background and estimation

This section introduces the context of the problem, followed by the details of our approach. Notation is introduced, and the adaptive MLE approach detailed in Chapter 3 is modified, resulting in a temporally adaptive estimate for a Markov transition matrix.

5.2.1 Problem definition

Similar to Chapter 4, consider the univariate data stream

\[ \{x_1, x_2, \ldots, x_{t-1}, x_t, \ldots\} , \]

where \( x_t \in \mathcal{S} = \{1, 2, \ldots, K\} \) for every time-stamp \( t \). Unlike the previous chapter, where it was assumed that observations arrived independently, this chapter places a first-order Markov property on the stream. That is, the probability of seeing a particular category at time \( t \) depends on the category that was observed at time \((t-1)\). Consequently, a \( K \times K \) transition matrix is required to describe the process, as opposed to a single probability vector. The set of categories \( \mathcal{S} \) is subsequently referred to as the state-space, which agrees with literature pertaining to Markov chains, and \( K \) is assumed fixed.

Similar to previous chapters, the stream is assumed to be generated in segments, which are governed by a set of unknown changepoints. Suppose that the locations of the changepoints are given by \( \boldsymbol{\tau} = \{\tau_k\}_{k=1}^{m} \), where \( \tau_i < \tau_j \) for every \( i < j \). The set \( \boldsymbol{\tau} \) partitions the stream into \((m+1)\) segments, and \( \mathbf{P}^{(k)} \in \mathbb{R}^{K \times K} \) is used to denote the fixed, unknown transition matrix in the \( k^{th} \) segment.

For each time-stamp let \( \mathbf{P}_t \) denote the transition matrix currently generating the data. This transition matrix can be defined piecewise as

\[ \mathbf{P}_t = \mathbf{P}^{(k)} \quad \forall t \in (\tau_{k-1}, \tau_k], \quad k \in \{1, \ldots, m+1\}, \]

where \( \tau_0 = 0 \) and \( \tau_{m+1} = \infty \). This is depicted graphically in Figure 5.1. The
$ij^{th}$ element of $P_t$ is denoted by $p_t^{(j|i)}$ and represents the Markov probability of transitioning from state $i$ to state $j$ at time $t$, that is,

$$p_t^{(j|i)} = P(X_t = j \mid X_{t-1} = i) \quad \forall i, j \in S.$$ 

When no changepoints are present in the data stream $P_t$ is independent of the time-stamp $t$. This is equivalent to the Markov chain being homogeneous – an assumption that is commonly made due to difficulties in performing inference on temporally evolving transition probabilities. However, in streaming contexts the assumption of time independent transition probabilities is unrealistic. For example, many change detection strategies have been developed for monitoring a network for abnormal activity; e.g. see Raiyn (2014) and Ten et al. (2011). Once a possible attack has been detected, model parameters must be able to dynamically reconfigure themselves without any user supervision – as activity from the network continues to arrive. A static transition matrix does not allow for this.

Consider the problem of updating an estimate for $P_t$. While data arrives in discrete-time, the opportunity to update a row $i \in S$ only arises after observing a transition from that state. Thus, two separate ‘clocks’ can be recognized in the data generating process. There is the data clock, which generates the sequence of states making up the data stream, and the transition clock that provides the opportunity to update a specific row of the estimate. The nature of such updating will require complex notation, even though the concept is relatively simple. The two different clocks also create challenges in assessing the performance of a change detector, as discussed in Section 5.4.1.
The goal of this chapter is to sequentially maintain an accurate estimate of the transition matrices in each segment and, moreover, to efficiently detect the changepoints in $\tau$. Analogous to Chapter 4, maintaining accurate estimates across segments requires intelligently discarding older data as newer data arrives. In the next section the forgetting factor framework discussed throughout Chapter 3 is modified for transition matrices.

### 5.2.2 Adaptive estimation for transition matrices

Since a matrix is being estimated, there are several ways of introducing forgetting factors into the parameter estimation. Moving forward, a forgetting factor is assigned to each row of the transition matrix estimate. Under this construction, if one row of the matrix experiences a change the appropriate forgetting factor can be tuned without affecting the others. Other formulations are possible, e.g., a single or $K^2$ forgetting factors could be assigned to monitor the transition matrix. However, a single forgetting factor would be tuned to monitor for a change in the entire matrix, and will not be able to react to subtle changes in a subset of the transition probabilities. Similarly, since each row is constrained to sum to one, $K^2$ forgetting factors would have to be aggregated across rows to preserve the constraints. Due to this, assigning a forgetting factor to each row is the most sensible, and the other formulations are not pursued further.

Suppose $x_{0:t}$ have been observed from the data stream, $x_0$ denoting the starting state of the chain. To assign a forgetting factor to each row, $x_{0:t}$ needs to be partitioned into blocks. This block decomposition is necessary since parameter estimation is driven by the transition clock instead of the data clock, as discussed in Section 5.2.1. Define these blocks by the multisets

$$B_t^{(i)} = \{x_k : x_{k-1} = i, \forall k = 1, \ldots, t\}. \quad (5.1)$$

The block $B_t^{(i)}$ represents the subset of observations from the Markov chain that were jumped to from a state $i \in S$, and is in ascending order according to the time-stamp. These blocks will be used in the updating of the parameters that are assigned to the $i^{th}$ row of the estimate. Further, $B_t^{(i)}[k]$ is used to
denote the $k^{\text{th}}$ element of $B_t^{(i)}$. This notation is essential in developing the non-sequential adaptive estimates that are presented shortly.

An example will help clarify the block decompositions. Consider the 3-state chain given in Figure 5.2. Then

$$B_t^{(1)} = \{x_1, x_2, x_7\}, \quad B_t^{(2)} = \{x_3, x_5\}, \quad B_t^{(3)} = \{x_4, x_6\},$$

where $B_t^{(1)}[2] = x_2$, $B_t^{(2)}[1] = x_3$ etc. Using this notation, the weighted log-likelihood function discussed in Chapter 3 can be presented, and optimized.

For each row $i \in S$, the weighted log-likelihood function in Equation (3.1) can be expressed as

$$L_{\text{FF}} \left( p^{(i)} \mid B_t^{(i)} \right) = \sum_{k=1}^{[B_t^{(i)}]} \left[ \prod_{\ell=k}^{[B_t^{(i)}]-1} \lambda_{\ell}^{(i)} \right] L \left( p^{(i)} \mid B_t^{(i)}[k] \right). \quad (5.2)$$

The vector $p^{(i)} \in \mathbb{R}^K$ is the parameter vector associated with the $i^{\text{th}}$ row of the transition matrix, which we are optimizing the likelihood with respect to, and $B_t^{(i)}$ was defined in Equation (5.1). Conditioning on transitioning from a state $i$, the $i^{\text{th}}$ row of the transition matrix can be viewed as a multinomial distribution; therefore, $L(\cdot | \cdot)$ is the multinomial log-likelihood function, and for every $\ell$, $\lambda^{(i)}_{\ell} \in [0, 1]$ is a forgetting factor associated with the $i^{\text{th}}$ row. The notation $[\ell]$ appearing in the subscript of the forgetting factor is analogous to that used for the block decompositions, and is used to index the multiset of forgetting factors associated with the $i^{\text{th}}$ row of the transition matrix.

Unless otherwise specified suppose that $x_t \in B_t^{(i)}$. This is equivalent to saying that an update of the $i^{\text{th}}$ row was possible at time $t$. Since no new information is available for any other row, all other blocks and forgetting factors are updated with the new time-stamp, i.e., $B_t^{(h)} = B_{t-1}^{(h)}$ and $\lambda_t^{(h)} = \lambda_{t-1}^{(h)} \forall h \neq i.$
This is a subtle issue, which allows us to remove the cumbersome multiset indexing notation \([\cdot]\) from the recursive update equations soon to be presented.

Let \(\tilde{P}_t \in \mathbb{R}^{K \times K}\) represent a matrix whose \(ij^{th}\) element is denoted by \(\tilde{p}_{ij}^{(j|i)}\). Optimization of Equation (5.2) is analogous to the derivations discussed in Chapters 3 and 4, and results in the non-sequential adaptive MLEs

\[
\tilde{p}_{ij}^{(j|i)} = \frac{1}{n_t^{(i)}} \sum_{k=1}^{|B_t^{(i)}|} w_k^{(i)} \mathbb{1}(B_t^{(i)}[k] = j) \quad \forall i, j \in S,\tag{5.3}
\]

where

\[
w_k^{(i)} = \prod_{\ell=k}^{B_t^{(i)} - 1} \lambda_{\ell}^{(i)}, \quad n_t^{(i)} = \sum_{k=1}^{|B_t^{(i)}|} w_k^{(i)}.\tag{5.4}
\]

Equations (5.3)-(5.4) are similar to Equations (4.4)-(4.5), the main difference being that we are now monitoring a transition matrix comprised of \(K\) multinomial distributions, as opposed to a single multinomial. Similar to Chapter 4, the corresponding recursive updates are

\[
n_t^{(i)} = \lambda_{t-1}^{(i)} n_{t-1}^{(i)} + 1,
\]

\[
\tilde{p}_{ij}^{(j|i)} = \left(1 - \frac{1}{n_t^{(i)}} \right) \tilde{p}_{ij}^{(j|i)} + \frac{1}{n_t^{(i)}} \mathbb{1}(x_t = j),
\]

which can be utilized whenever a transition from state \(i\) is observed. Whenever a transition from state \(i\) is not observed, similar to the blocks and forgetting factors, all parameters associated with row \(i\) are updated with the new time-stamp. Thus, the subscript \(t\) does not necessarily imply that the corresponding parameters were updated at the current time-step, but represents the most recent estimates available. Additionally, the subscript \(t\) does not mean that estimates have incorporated \(t\) observations in their estimation, since updating them relies solely on the observations in their corresponding block. Hence, the time subscript \(t\) for the parameters is appropriate and, as alluded to earlier, allows one to remove the involved multiset indexing notation on the parameter estimates. As throughout this thesis, the forgetting
factors for the $i^{th}$ row are tuned using stochastic gradient descent:

$$
\lambda_t^{(i)} = \lambda_t^{(i)} - \eta \left( \sum_{j \in S} \mathbb{1}(x_t = j) \frac{\nabla \tilde{p}_{t-1}^{(j|i)}}{\tilde{p}_{t-1}^{(j|i)}} \right),
$$

where we have assumed that $x_t \in B_t^{(i)}$, so that the forgetting factor $\lambda_t^{(i)}$ can be updated at time $t$. The gradient updates are also the same as in the previous chapter, and are given by

$$
\nabla n_t^{(i)} = \lambda_t^{(i)} - \frac{1}{n_t^{(i)}} \nabla n_{t-1}^{(i)} + n_{t-1}^{(i)},
$$

$$
\nabla \tilde{p}_{t}^{(j|i)} = \left(1 - \frac{1}{n_t^{(i)}}\right) \nabla \tilde{p}_{t-1}^{(j|i)} - \frac{\nabla n_t^{(i)}{(i)}}{n_t^{(i)}}^2 \left( \mathbb{1}(x_t = j) - \tilde{p}_{t-1}^{(j|i)} \right).
$$

Subsequently, properties of $\tilde{P}_t$ will prove useful. First, summing Equation (5.3) over $j \in S$ shows that $\tilde{P}_t$ requires no renormalization, that is, it is a stochastic matrix. Moreover, in Section 5.3 the mean and variance of each element of $\tilde{P}_t$ will be required. These are provided in Lemma 3, and are derived in Appendix A.4.

**Lemma 3.** Consider the $i^{th}$ row of $\tilde{P}_t$, and let $B_t^{(i)}$ be defined as in Equation (5.1). Assuming that the true transition probabilities are $\{p_t^{(j|i)}\}_{j=1}^K$, and that $\tilde{p}_t^{(j|i)}$ is a random variable, the mean and variance of $\tilde{p}_t^{(j|i)}$ are given by

$$
\mathbb{E} \left( \tilde{p}_t^{(j|i)} \right) = p_t^{(j|i)}, \quad \text{Var} \left( \tilde{p}_t^{(j|i)} \right) = u_t^{(i)} \tilde{p}_t^{(j|i)} \left(1 - p_t^{(j|i)} \right),
$$

where

$$
u_t^{(i)} = m_t^{(i)} \left[ n_t^{(i)} \right]^{-2}, \quad m_t^{(i)} = \sum_{k=1}^{\lfloor B_t^{(i)} \rfloor} \left[ w_k^{(i)} \right]^2.
$$

The quantity $m_t^{(i)}$ can be computed recursively on the stream according to

$$
m_t^{(i)} = \left( \lambda_t^{(i)} \right)^2 m_{t-1}^{(i)} + 1,
$$
and is derived in Appendix A.4.

In the next section a change detection method is developed, which monitors for changes in each component of the transition matrix by constructing control charts for the adaptive transition probabilities. We remark that the MCDMs of Chapter 4 could be independently run over each row of the transition matrix; however, this would only tell us when a particular row has experienced a change. As matrices have $K(K-1)$ more elements than a $K$ category multinomial distribution, the matrix could undergo many different types of structural changes, making the monitoring of each component appealing.

5.3 ADEPT-M

The adaptive estimate developed in Section 5.2 is now used in the formulation of an online change detection method. This adaptive detection and estimation procedure for transition matrices (ADEPT-M) sequentially monitors for multiple changepoints in each element of $P_t$, as well as maintains an adaptive estimate for the transition matrix currently generating the data. This method introduces control parameters that neither depend on how many changepoints are in the stream, nor the magnitudes of the unknown changes. This appears to be the first attempt at developing an online, adaptive, multiple changepoint detector for transition matrices.

Consider monitoring for a change in $p_{t}^{(j|i)}$. At every time-step, ADEPT-M is either in a grace period, or is monitoring the stream for changepoints. These ideas were discussed in Section 2.4.4 and were depicted in Figure 2.4. When a changepoint is detected, the $i^{th}$ row enters a grace period, and ADEPT-M does not monitor for subsequent changes while in this period. Every other row of the transition matrix, which is currently not in a grace period, continues monitoring for changepoints. Control limits are then computed via a function of the parameters estimated at the end of the grace period. The left panel of Figure 5.3 illustrates this procedure for a single detection (the right panel shows the mathematical notation introduced in this section). The
Figure 5.3: An illustration of how ADEPT-M resets after a change is detected. The \( \times \)'s denote if a transition occurred at a particular time-step or not. The left panel is given to help understand the mathematical notation appearing in the right panel.

Scalars \( L^{(1)} \) and \( U^{(1)} \) are the control limits before the detection, and \( L^{(2)} \) and \( U^{(2)} \) are the control limits computed after the grace period terminates. This process is repeated every time ADEPT-M detects a changepoint.

Suppose at time \( t = \hat{\tau} \) a change is detected in \( p_{t|i}^{(j|i)} \). ADEPT-M immediately enters a grace period, which terminates based on the value \( G \in \mathbb{Z}^+ \) (as discussed in Section 2.4.4). Unlike the grace period used in Chapter 4, the value of \( G \) has several interpretations when monitoring a transition matrix on the stream. One interpretation is that the grace period will end at time \( (\hat{\tau} + G) \), which was the case in the preceding chapter. However, if \( p_{t|i}^{(j|i)} \) is small after a changepoint, the value of \( G \) may not be large enough to observe any transitions from state \( i \) to state \( j \). To mitigate this problem, the grace period is terminated only when \( G \) transitions from state \( i \) to state \( j \) are observed, that is, a data-adaptive grace period is used. Since \( G \) is defined this way, ADEPT-M will have the opportunity to estimate the transition probabilities associated with the new segment.

As discussed in the previous chapter, a common way of choosing control limits is to provide guarantees on the false positive rate. Given a significance level \( \alpha \in (0, 1) \) the control limits, denoted \( L_t^{(j|i)} \) and \( U_t^{(j|i)} \), are typically chosen so
that

\[ \mathbb{P} \left( \tilde{p}_t^{(j|i)} \in \left[ L_t^{(j|i)}, U_t^{(j|i)} \right] \right) \approx 1 - \alpha. \]

Exact 100(1 − \(\alpha\))% control limits would require the distribution of \(\tilde{p}_t^{(j|i)}\). This quantity is a weighted sum of Bernoulli random variables whose weights may assume any value in [0, 1], and under this formulation, a known distribution does not appear to exist. There exists some work in learning the distribution of a weighted sum of Bernoulli random variables, e.g. Daskalakis et al. (2015) proposed an algorithm that can learn the distribution in polynomial time, for a discrete set of weights. However, this high computational complexity is not suitable for streaming data. Raghavan (1986) provided bounds on weighted sums of Bernoulli random variables; however, in practice these bounds are too conservative, frequently not respecting the range of the transition probability estimates.

Moving forward a moment matching technique is used to construct approximate control limits. Such methods have found success in various areas, such as: developing adaptive thresholds for network counts (Lambert and Liu, 2006), visual scanning in particle physics (Sanathanan, 1972) and in estimating parameters for generalized extreme value distributions (Hosking et al., 1985). Since \(\tilde{p}_t^{(j|i)} \in [0, 1]\), a natural distribution to match with is a Beta\((a, b)\), as it respects the range of the estimate. The beta distribution was discussed in Section 2.2.2, where the mean and variance of a random variable \(Y \sim \text{Beta}(a, b)\) were shown to be

\[ \mathbb{E}(Y) = \frac{a}{a + b}, \quad \text{Var}(Y) = \frac{ab}{(a + b)^2(a + b + 1)}, \quad a, b > 0. \]

Equating with the mean and variance of \(\tilde{p}_t^{(j|i)}\) provided in Lemma 3, and using the adaptive transition probabilities as plug-in estimates, the following system of equations are obtained

\[ \tilde{p}_t^{(j|i)} = \frac{a}{a + b}, \quad u_t^{(i)} \tilde{p}_t^{(j|i)} \left( 1 - \tilde{p}_t^{(j|i)} \right) = \frac{ab}{(a + b)^2(a + b + 1)}. \]
Solving for \( a \) and \( b \), including appropriate notation, results in

\[
a_t^{(j|i)} = \left( \frac{1}{u_t^{(i)}} - 1 \right) \tilde{p}_t^{(j|i)}, \quad b_t^{(j|i)} = \left( \frac{1}{u_t^{(i)}} - 1 \right) \left( 1 - \tilde{p}_t^{(j|i)} \right).
\]

Only the *most recent detection* made is required to construct the control limits for a new segment. Let \( \hat{\tau}^{(j|i)} \) be the most recent detection made for \( p_t^{(j|i)} \), and let \( \hat{\gamma}^{(j|i)} \) denote the length of the grace period associated with this detection. That is, \( \hat{\gamma}^{(j|i)} \) is the amount of time taken to observe \( G \) transitions from state \( i \) to state \( j \), following the detection \( \hat{\tau}^{(j|i)} \). Using this notation the grace period will terminate at time \( \hat{\tau}^{(j|i)} + \hat{\gamma}^{(j|i)} \) (refer to the right panel of Figure 5.3). For an \( \alpha \in (0, 1) \), when ADEPT-M is leaving a grace period the control limits are computed as

\[
L_t^{(j|i)} = Q\left( \frac{\alpha}{2} \mid a_t^{(j|i)}, b_t^{(j|i)} \right) \bigg|_{t=\hat{\tau}^{(j|i)} + \hat{\gamma}^{(j|i)}},
\]

\[
U_t^{(j|i)} = Q\left( 1 - \frac{\alpha}{2} \mid a_t^{(j|i)}, b_t^{(j|i)} \right) \bigg|_{t=\hat{\tau}^{(j|i)} + \hat{\gamma}^{(j|i)}},
\]

where \( Q(\cdot \mid a, b) \) is the quantile function associated with a Beta(\( a, b \)) distribution. A change is said to occur in the \( ij \)th component whenever the adaptive transition probability falls outside the control limits, i.e., when

\[
\tilde{p}_t^{(j|i)} \notin \left[ L_t^{(j|i)}, U_t^{(j|i)} \right].
\]

The right hand side of Equations (5.5)-(5.6) use the estimate of \( \tilde{p}_t^{(j|i)} \) obtained immediately after the grace period ends, and these control limits will remain *constant* until the algorithm detects another change – in which case ADEPT-M will enter a new grace period, and the values of \( \hat{\tau}^{(j|i)} \) and \( \hat{\gamma}^{(j|i)} \) would be updated accordingly.

5.4 Synthetic examples

Synthetic simulations are now considered, where ADEPT-M is compared to the CUSUM and EWMA control charts discussed in Section 2.4.2. Since
these methods were not designed to detect changes in transition matrices, both methods are run over the *binarization* of the data streams, similar to what was discussed in Section 4.6. However, in the current setting the control charts need to be run over $K^2$ streams (as opposed to $K$), where the streams indicate what transition occurred at each time-stamp.

Section 5.4.1 discusses performance measures, and modifications that are required to use them in the current setting. Section 5.4.2 describes the synthetic simulations, and Section 5.4.3 presents the results.

5.4.1 Performance measures

The performance measures $\text{ARL}_0$, $\text{ARL}_1$, CCD and DNF introduced in Section 2.4.3 are used to gauge ADEPT-M’s ability to detect multiple change-points. However, modifications are required to use the average run lengths for performance measures, which are discussed below.

Suppose a detector, in two simulations, flagged its first false positive at times 100 and 200 respectively. The standard definition of $\text{ARL}_0$ states that, on average, a false positive is flagged by the detector every 150 time-stamps. However, since a transition from state $i$ to state $j$ is not observed at every time-stamp, this value of $\text{ARL}_0$ is an over-estimate for the true $\text{ARL}_0$. To see this, suppose that a transition from state $i$ to state $j$ occurred, respectively, 30 and 40 times on the intervals $[0, 100]$ and $[0, 200]$. Since 30 and 40 transitions occurred over the two simulations, a more accurate $\text{ARL}_0$ to report is 35. Any time a transition from state $i$ to state $j$ did not occur should not contribute to the value of $\text{ARL}_0$. The same reasoning can be applied to $\text{ARL}_1$.

This example makes it clear that the ARLs should be computed with respect to the transition clock, as opposed to the data clock. That is, the number of state $i$ to state $j$ transitions that occur in certain time intervals should be used as opposed to the estimated changepoint locations. Henceforth, it is understood that these adjustments have been made when computing the ARLs.

Since each performance measure is estimated for every component of $P_t$, they
can be represented as $K \times K$ matrices. In the next section a large number of Monte Carlo simulations are analyzed, and reporting four $K \times K$ matrices for every simulation would be too much information to interpret. Therefore, these ‘performance measure matrices’ are averaged over their elements to return a single value that reflects how well a change detector did overall. These estimates are still denoted by $\text{ARL}_0$, $\text{ARL}_1$, CCD and DNF.

5.4.2 Experimental design

To estimate performance measures, each change detector is implemented on 200 data streams of length $10^5$. In each simulation the number of changepoints assumes a value $m \in \{0, 1, 10, 50, 100\}$, and the changepoints are randomly generated according to a set of rules. If $m = 0$ there is no changepoint to generate, and if $m = 1$ the changepoint is randomly placed near the middle of the stream. When $m > 1$, a generation scheme similar to Chapter 4 is used. This results in the changepoint locations being generated as:

$$
\tau_1 = F + \nu_1,
\tau_k = \tau_{k-1} + (D + F) + \nu_k \quad k = 2, 3, \ldots, m,
$$

where each $\nu_k \sim \text{Poisson}(r)$ and $r, D, F \in \mathbb{Z}^+$. The rate parameter $r$ dictates the average length between consecutive changepoints, and $D$ and $F$ are used to ‘pad’ the intervals to give the detectors time to detect a change and undergo a grace period. The scalars $r$, $D$ and $F$ are not control parameters for any of the change detectors, and are only used when generating changepoints.

In each segment the stream is governed by a fixed probability transition matrix, and $(m + 1)$ matrices need to be generated. Consider the task of generating $P^{(k)}$. To simplify the estimation of performance measures, across segments an abrupt change is induced in every element of the matrix. To ensure there has been a noticeable change in the matrices across consecutive changepoints, a scheme is required to construct $P^{(k)}$ from $P^{(k-1)}$. This is itself an interesting problem, although not a focal point of the chapter. Succinctly, for each row of $P^{(k-1)}$ five ‘candidate vectors’ are uniformly sampled...
from the unit simplex. Each row of $P^{(k)}$ is then given by the candidate vector which differs most from the corresponding row of $P^{(k-1)}$ in terms of Euclidean distance. We remark that a minimum value for the Euclidean distances between the rows of $P^{(k-1)}$ and the sampled candidate vectors was not defined. This is because we want to investigate detector performance for a wide range of magnitudes between changepoints, and not restrict to situations where the magnitudes are larger than a predefined threshold.

The scalar that determines the end of the grace period assumes a value $G \in \{25, 50, 75, 100\}$, and for changepoint generation, $D = 50$, $F = 20$ and $r = \lceil 10^5 / m \rceil$ are chosen. All of these parameters are shared among the three change detectors. The control parameters used for CUSUM and EWMA were given in Table 2.1, and for ADEPT-M several choices for the control parameter tuple $(\alpha, \eta)$ are considered. Specifically, $\alpha = 10^{-i}$ and $\eta = 10^{-j}$ where $i \in \{2, 3, 4\}$ and $j \in \{4, 5, 6\}$. Due to space limitations, only the case when $K = 3$ is presented.

5.4.3 Results

Results are summarized in Tables 5.1-5.3. ADEPT-M’s performance is displayed in Tables 5.1 and 5.3, and results for CUSUM and EWMA appear in Table 5.2. In Table 5.2 a subscript for CUSUM and EWMA corresponds to which control parameter tuple is being used. For example, CUSUM$_1$ corresponds to results obtained using $(k_1, h_1)$, as given in Table 2.1.

Consider the ARL results in Table 5.1, and in the upper sections of Table 5.2. Since $\text{ARL}_0$ is computed by averaging over the first false positive flagged, its value is invariant with respect to $G$. The value of $\text{ARL}_1$ does depend on $G$, as any false positive flagged before the changepoint will affect the monitoring periods for ADEPT-M. Each value of $\eta$ results in ARLs that are comparable to the results obtained from CUSUM and EWMA. As expected, there are combinations of their parameters that report better values. The point is, a priori, the choice of the control parameters for CUSUM and EWMA typically depend on information that is unavailable to the analyst. Further, no single set of control parameters for CUSUM or EWMA seems to do well across all
columns in Table 5.2; whereas, ADEPT-M seems to consistently balance the trade-off between the ARLs, resulting in favorable performance.

Results for CCD and DNF are provided in Table 5.3 for ADEPT-M, and in the lower sections of Table 5.2 for CUSUM and EWMA. These tables show the capability of ADEPT-M, and for competing methods, highlight issues in having control parameters whose values should be chosen based on characteristics of the changepoints. Table 5.2 has combinations of control parameters that result in $CCD \geq 0.90$; however, the corresponding DNF is very low. This means that CUSUM and EWMA are flagging many changes, resulting in a low proportion of detections that are not false. In many applications, such as monitoring computer traffic for malicious activity, a high number of false positives would be unacceptable. There are some cases where CCD and DNF are reasonable; however, as previously stated, the analyst is given little guidance into when a certain set of control parameters will yield adequate performance. Furthermore, many entries in Table 5.2 have both performance
Table 5.2: Performance measures for CUSUM and EWMA when $K = 3$. The subscript on the methods correspond to which control parameter tuple is being used in the simulations. The control parameters were provided in Table 2.1.

<table>
<thead>
<tr>
<th>$m$</th>
<th>EWMA$_1$ ARL</th>
<th>EWMA$_2$ ARL</th>
<th>EWMA$_3$ ARL</th>
<th>CUSUM$_1$ ARL</th>
<th>CUSUM$_2$ ARL</th>
<th>CUSUM$_3$ ARL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8795.49</td>
<td>4153.88</td>
<td>739.97</td>
<td>981.33</td>
<td>2518.27</td>
<td>3481.03</td>
</tr>
<tr>
<td>1</td>
<td>2838.82</td>
<td>942.42</td>
<td>83.76</td>
<td>89.24</td>
<td>434.20</td>
<td>700.17</td>
</tr>
<tr>
<td>10</td>
<td>2791.25</td>
<td>801.72</td>
<td>73.14</td>
<td>103.01</td>
<td>407.70</td>
<td>660.62</td>
</tr>
<tr>
<td>50</td>
<td>2466.87</td>
<td>753.27</td>
<td>93.15</td>
<td>99.88</td>
<td>428.81</td>
<td>612.61</td>
</tr>
<tr>
<td>75</td>
<td>2499.27</td>
<td>749.55</td>
<td>98.25</td>
<td>98.25</td>
<td>372.02</td>
<td>588.76</td>
</tr>
<tr>
<td>100</td>
<td>2791.25</td>
<td>801.72</td>
<td>73.14</td>
<td>103.01</td>
<td>407.70</td>
<td>660.62</td>
</tr>
</tbody>
</table>

measures being low, indicating that the methods fail to successfully detect multiple changepoints in the streaming data setting.

Table 5.3 displays ADEPT-M's results for the multiple changepoint setting, where each element in the table is a tuple (CCD, DNF). The table highlights that when $\alpha$ decreases, CCD decreases and DNF increases. This agrees with intuition, as lower values of $\alpha$ result in wider control limits.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>(10^{-4})</th>
<th>(10^{-5})</th>
<th>(10^{-6})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(G = 25)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(10^{-2})</td>
<td>(0.84, 0.51)</td>
<td>(0.89, 0.27)</td>
<td>(0.92, 0.21)</td>
</tr>
<tr>
<td></td>
<td>(10^{-3})</td>
<td>(0.73, 0.75)</td>
<td>(0.83, 0.57)</td>
<td>(0.88, 0.48)</td>
</tr>
<tr>
<td></td>
<td>(10^{-4})</td>
<td>(0.63, 0.83)</td>
<td>(0.77, 0.73)</td>
<td>(0.84, 0.67)</td>
</tr>
<tr>
<td>(m = 10)</td>
<td>(G = 50)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(10^{-2})</td>
<td>(0.84, 0.53)</td>
<td>(0.88, 0.30)</td>
<td>(0.89, 0.24)</td>
</tr>
<tr>
<td></td>
<td>(10^{-3})</td>
<td>(0.73, 0.76)</td>
<td>(0.83, 0.64)</td>
<td>(0.86, 0.54)</td>
</tr>
<tr>
<td></td>
<td>(10^{-4})</td>
<td>(0.63, 0.84)</td>
<td>(0.76, 0.81)</td>
<td>(0.82, 0.75)</td>
</tr>
<tr>
<td></td>
<td>(G = 75)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(10^{-2})</td>
<td>(0.83, 0.54)</td>
<td>(0.86, 0.33)</td>
<td>(0.88, 0.26)</td>
</tr>
<tr>
<td></td>
<td>(10^{-3})</td>
<td>(0.72, 0.76)</td>
<td>(0.81, 0.67)</td>
<td>(0.84, 0.56)</td>
</tr>
<tr>
<td></td>
<td>(10^{-4})</td>
<td>(0.71, 0.44)</td>
<td>(0.74, 0.84)</td>
<td>(0.80, 0.78)</td>
</tr>
<tr>
<td></td>
<td>(G = 100)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(10^{-2})</td>
<td>(0.81, 0.54)</td>
<td>(0.84, 0.35)</td>
<td>(0.86, 0.28)</td>
</tr>
<tr>
<td></td>
<td>(10^{-3})</td>
<td>(0.70, 0.76)</td>
<td>(0.79, 0.69)</td>
<td>(0.83, 0.58)</td>
</tr>
<tr>
<td></td>
<td>(10^{-4})</td>
<td>(0.61, 0.84)</td>
<td>(0.72, 0.86)</td>
<td>(0.78, 0.80)</td>
</tr>
<tr>
<td></td>
<td>(G = 25)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(10^{-2})</td>
<td>(0.77, 0.63)</td>
<td>(0.83, 0.53)</td>
<td>(0.85, 0.51)</td>
</tr>
<tr>
<td></td>
<td>(10^{-3})</td>
<td>(0.67, 0.83)</td>
<td>(0.78, 0.74)</td>
<td>(0.81, 0.70)</td>
</tr>
<tr>
<td></td>
<td>(10^{-4})</td>
<td>(0.56, 0.92)</td>
<td>(0.73, 0.84)</td>
<td>(0.77, 0.80)</td>
</tr>
<tr>
<td>(m = 50)</td>
<td>(G = 50)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(10^{-2})</td>
<td>(0.71, 0.70)</td>
<td>(0.76, 0.62)</td>
<td>(0.77, 0.60)</td>
</tr>
<tr>
<td></td>
<td>(10^{-3})</td>
<td>(0.61, 0.89)</td>
<td>(0.70, 0.83)</td>
<td>(0.73, 0.80)</td>
</tr>
<tr>
<td></td>
<td>(10^{-4})</td>
<td>(0.52, 0.96)</td>
<td>(0.65, 0.92)</td>
<td>(0.69, 0.89)</td>
</tr>
<tr>
<td></td>
<td>(G = 75)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(10^{-2})</td>
<td>(0.66, 0.74)</td>
<td>(0.69, 0.68)</td>
<td>(0.71, 0.66)</td>
</tr>
<tr>
<td></td>
<td>(10^{-3})</td>
<td>(0.57, 0.91)</td>
<td>(0.64, 0.87)</td>
<td>(0.66, 0.85)</td>
</tr>
<tr>
<td></td>
<td>(10^{-4})</td>
<td>(0.48, 0.96)</td>
<td>(0.59, 0.95)</td>
<td>(0.62, 0.93)</td>
</tr>
<tr>
<td></td>
<td>(G = 100)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(10^{-2})</td>
<td>(0.61, 0.78)</td>
<td>(0.64, 0.71)</td>
<td>(0.65, 0.70)</td>
</tr>
<tr>
<td></td>
<td>(10^{-3})</td>
<td>(0.53, 0.92)</td>
<td>(0.58, 0.89)</td>
<td>(0.60, 0.87)</td>
</tr>
<tr>
<td></td>
<td>(10^{-4})</td>
<td>(0.46, 0.97)</td>
<td>(0.54, 0.96)</td>
<td>(0.56, 0.94)</td>
</tr>
<tr>
<td></td>
<td>(G = 25)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(10^{-2})</td>
<td>(0.72, 0.72)</td>
<td>(0.74, 0.68)</td>
<td>(0.76, 0.66)</td>
</tr>
<tr>
<td></td>
<td>(10^{-3})</td>
<td>(0.64, 0.88)</td>
<td>(0.69, 0.84)</td>
<td>(0.72, 0.80)</td>
</tr>
<tr>
<td></td>
<td>(10^{-4})</td>
<td>(0.57, 0.94)</td>
<td>(0.64, 0.92)</td>
<td>(0.68, 0.87)</td>
</tr>
<tr>
<td>(m = 100)</td>
<td>(G = 50)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(10^{-2})</td>
<td>(0.62, 0.82)</td>
<td>(0.63, 0.77)</td>
<td>(0.65, 0.76)</td>
</tr>
<tr>
<td></td>
<td>(10^{-3})</td>
<td>(0.55, 0.94)</td>
<td>(0.57, 0.91)</td>
<td>(0.60, 0.89)</td>
</tr>
<tr>
<td></td>
<td>(10^{-4})</td>
<td>(0.49, 0.98)</td>
<td>(0.53, 0.96)</td>
<td>(0.57, 0.95)</td>
</tr>
<tr>
<td></td>
<td>(G = 75)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(10^{-2})</td>
<td>(0.54, 0.86)</td>
<td>(0.55, 0.82)</td>
<td>(0.56, 0.82)</td>
</tr>
<tr>
<td></td>
<td>(10^{-3})</td>
<td>(0.48, 0.95)</td>
<td>(0.50, 0.93)</td>
<td>(0.52, 0.93)</td>
</tr>
<tr>
<td></td>
<td>(10^{-4})</td>
<td>(0.43, 0.98)</td>
<td>(0.46, 0.97)</td>
<td>(0.48, 0.97)</td>
</tr>
<tr>
<td></td>
<td>(G = 100)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(10^{-2})</td>
<td>(0.48, 0.89)</td>
<td>(0.49, 0.85)</td>
<td>(0.49, 0.85)</td>
</tr>
<tr>
<td></td>
<td>(10^{-3})</td>
<td>(0.43, 0.96)</td>
<td>(0.44, 0.95)</td>
<td>(0.46, 0.94)</td>
</tr>
<tr>
<td></td>
<td>(10^{-4})</td>
<td>(0.39, 0.99)</td>
<td>(0.42, 0.98)</td>
<td>(0.43, 0.98)</td>
</tr>
</tbody>
</table>

*Table 5.3:* ADEPT-M’s results for CCD and DNF. Each entry in the table is a tuple \((\alpha, \eta)\), where values closer to one indicate better performance.
Choosing larger values for $\eta$ typically results in a smaller value for DNF, which is also intuitive since we are taking larger steps in each iteration of stochastic gradient descent, resulting in more volatile forgetting factors. Smaller values of $\eta$, as in the ARL case, provides a good balance between both performance measures, and the results show considerable improvements when compared to CUSUM and EWMA.

The development of ADEPT-M introduced the control parameters $G$, $\alpha$ and $\eta$. Practically, in the presence of multiple changepoints any detector will need to be equipped with a restarting procedure, which will typically involve a parameter similar to $G$. We favor the grace period as it is easy to understand, and $G$ can be chosen given a lower bound on the false positive rate. The scalar $\alpha$ results in approximate $100(1 - \alpha)\%$ control limits, and from empirical results, decreasing $\alpha$ generally results in smaller values of CCD and larger values for DNF. The step-size $\eta$, to an extent, affects the behavior of the forgetting factors, and empirical evidence suggests that values $\eta < 10^{-3}$ result in robust values for the performance measures. In this regard, Table 5.3 could be used to suggest values for $(\alpha, \eta)$, which can help balance the trade-off between the performance measures.

### 5.5 HTTP Request Data

Morgan (2017) has estimated that cyber-crime will cost the world $6$ trillion annually by the year 2021. With cyber-crime continuing to be a major threat, the development of methods to detect attacks in an efficient manner is a pressing concern. A Distributed Denial of Service (DDoS) attack is a common type of cyber-attack, where a malicious agent overwhelms a service to prevent legitimate access (Mirkovic and Reiher, 2004). In this section a sequence of HTTP web requests being managed by a scheduler is considered. Section 5.5.1 introduces the data and discusses a special type of DDoS attack – an HTTP flood attack. Section 5.5.2 manipulates the request data to mimic an HTTP flood, and ADEPT-M is shown to successfully detect the attack.
5.5.1 The data

A sequence of 9,241,302 HTTP web requests collected on September 7, 2016 from 9am to 3pm, corresponding to approximately 428 observations per second, is considered. Web requests include a request type field, and the six appearing in the data stream are

\[ S = \{ \text{DELETE, GET, HEAD, OPTIONS, POST, PUT} \}. \]

Briefly, a GET request pulls information from a server, a PUT request pushes information to a server, and a POST request sends a client’s data to a server for processing. The POST request can be manipulated in an HTTP flood attack, as discussed below. See Gourley and Totty (2002) for more details on HTTP web requests.

The data has no ground-truth associated with it, that is, the existence and location of changepoints are unknown. This is typically the case in many real-world data streams, and to circumvent this issue the data is manipulated to mimic an HTTP flood attack. An HTTP flood attack is a DDoS attack where an attacker exploits either POST or GET requests, typically using ‘botnets’ to generate a high volume of malicious activity (Zargar et al., 2013). These malicious request packets are used to attack a web server by over-allocating its resources, resulting in legitimate users being denied access. Furthermore, the tampered packets are hard to distinguish from genuine traffic since they have legitimate HTTP payloads (Yatagai et al., 2007). For illustration purposes we restrict to HTTP POST floods in the next section.

5.5.2 Results

The simplest way of mimicking an HTTP flood is to choose a time for the attack, and then flood the stream with POST requests that lasts for a prespecified amount of time. Let \( \tau \) be the start time of the attack, and let \( \delta \in \{0.25, 0.50, 1.00, 2.00\} \) represent the duration of the attack (in minutes). Since approximately 428 observations arrive per second, the number of POST requests in the flood is taken as \( t_5 = \delta (428 \times 60) \). Although uncomplicated,
this set-up provides a practical way of determining if ADEPT-M can detect such an attack, and is a sensible simulation in the absence of ground-truth labeling of malicious activity.

In the cyber-security setting it is typical to require a detector to raise few false alarms (Turcotte et al., 2017). Therefore, given prior domain knowledge the user could choose the grace period $G$ to provide a lower bound on the number of detections. Moving forward we choose $G = 15 \times 428$, resulting in roughly 15 second grace periods (in accordance with the data clock). Since the length of the data stream depends on $\delta$, the burn-in length $B$ is chosen so ADEPT-M is run on a stream of length 9 million. The significance level $\alpha$ used to compute the control limits, and the step-size $\eta$ used in the gradient descent step, take values in the sets

$$\alpha \in \{10^{-3}, 10^{-4}, 10^{-5}\}, \quad \eta \in \{10^{-4}, 10^{-5}, 10^{-6}\}.$$  

Table 5.4 shows the earliest detection made by ADEPT-M, for each $(\delta, \alpha, \eta)$, in the row corresponding to POST requests during the attack period $[\tau, \tau + t_\delta]$. These values have been scaled to give approximately the amount of time (in seconds) that it took ADEPT-M to detect the attack. ADEPT-M was also run over the request data without the HTTP flood present (using the same parameters), and the earliest detection made was roughly 423 seconds after $\tau$, that is, it is reasonable to conclude that ADEPT-M genuinely detected the attack. Further inspection of Table 5.4 shows that the detection of the HTTP flood is fairly robust to the choice of $(\alpha, \eta)$. Additionally, ADEPT-M took roughly 5 minutes to process the 9 million out of burn-in observations – making ADEPT-M an attractive, and practical choice for real-world data streams that require adaptively estimating a transition matrix.

5.6 Discussion

In this chapter the forgetting factor and multinomial frameworks discussed throughout Chapters 3 and 4 were extended to Markov transition matrices. This required carefully reformulating the estimation procedure, which
\[
\begin{array}{cccc}
(\alpha, \eta) & 10^{-4} & 10^{-5} & 10^{-6} \\
\hline
\delta = 0.25 & 10^{-3} & 1.83 & 1.83 & 1.83 \\
& 10^{-4} & 1.91 & 1.92 & 1.92 \\
& 10^{-5} & 2.89 & 2.90 & 2.91 \\
\hline
\delta = 0.50 & 10^{-3} & 1.82 & 1.83 & 1.83 \\
& 10^{-4} & 1.88 & 1.90 & 1.91 \\
& 10^{-5} & 1.90 & 1.92 & 3.22 \\
\hline
\delta = 1.00 & 10^{-3} & 1.86 & 1.86 & 1.86 \\
& 10^{-4} & 1.91 & 1.92 & 1.96 \\
& 10^{-5} & 1.96 & 3.22 & 3.22 \\
\hline
\delta = 2.00 & 10^{-3} & 1.87 & 1.88 & 1.89 \\
& 10^{-4} & 1.90 & 1.92 & 1.93 \\
& 10^{-5} & 1.90 & 1.92 & 3.25 \\
\end{array}
\]

**Table 5.4:** The earliest detection made by ADEPT-M during the interval \([\tau, \tau + t_5]\), in the row of the adaptive transition matrix corresponding to POST requests. Each detection is scaled to give a rough estimate of the amount of time (in seconds) ADEPT-M took to detect the attack, for each combination of the tuple \((\delta, \alpha, \eta)\).

required introducing non-trivial notation to accurately describe an adaptive estimate for a Markov transition matrix.

This adaptive transition matrix was then used in developing ADEPT-M, an adaptive detection and estimation procedure for transition matrices, which can detect multiple changepoints in a drifting data stream, and is the main contribution of the chapter. ADEPT-M monitors each component of the matrix for changes, and does so based on matching moments with a beta distribution, resulting in approximate 100(1 − \(\alpha\))% control limits.

Through a simulation study ADEPT-M was shown to outperform commonly used control charts in the literature. Performance was assessed using the measures discussed in Section 2.4.3; however, adjustments to the average run lengths were necessary in the transition matrix setting, and were addressed in this chapter. ADEPT-M was then implemented on a real-world data stream consisting of HTTP web requests, and was shown to be successful in detecting a synthetic HTTP flood attack.

In the next chapter we steer away from categorical data streams, and consider
continuous-valued, univariate data streams. A technique for maintaining a sequential histogram is developed, where temporal adaptivity is introduced using the probability that an observation falls inside a particular histogram bin. Several interesting consequences of temporally adaptive streaming histograms are explored in Chapter 7.
6

Temporally Adaptive Streaming Histograms

It is infeasible to store data streams in computer memory, requiring the need for computationally efficient methods that can summarize drifting data streams. Summarization methods are broadly referred to as synopsis techniques, which includes estimation via histograms – the main focus of this chapter. Histograms can be dated back to the late 1800s (Pearson, 1895), and are favored due to their simplicity, and the fact that they have been a ‘go-to statistical tool’ for experts and non-experts alike, for over 100 years.

The focal point of this chapter is inherently different than the previous two, which were concerned with changepoint detection in categorical streams. In this chapter we move away from changepoint detection, and focus on developing a framework for constructing temporally evolving histograms for univariate, continuous-valued data streams, where no distributional assumptions are placed on the data. The histograms are capable of sequentially discretizing the univariate data stream, where many of the categorical techniques developed throughout this thesis are then applicable. This chapter presents several novel contributions, namely: (1) using adaptive bin widths,
as opposed to the popular equiwidth strategy, to build the histograms, (2) the development of new techniques to merge and split histogram bins and (3) modifications to the adaptive forgetting framework are made so that the histograms can react to drift. These contributions culminate in Temporally Adaptive Streaming Histograms (TASHs), which are able to provide accurate density estimates in drifting and non-drifting data streams.

The chapter proceeds as follows: Section 6.1 provides an overview of the literature, while Section 6.2 presents notation relevant to the chapter. Section 6.3 provides the machinery necessary to sequentially maintain histogram estimates. Synthetic simulations are then conducted, which verifies that the proposed approach works well in non-drifting data streams. Section 6.4 modifies the forgetting factor framework for categorical data, resulting in temporally dynamic histograms that provide accurate density estimates in the presence of drift, as well as in non-drifting environments. Simulations in drifting streams are then presented, and it is shown that TASHs outperform various approaches commonly used in practice.

6.1 Literature review

A review of histograms in the classical setting has been provided in Section 2.3.1; consequently, the current overview focuses solely on histograms developed for streaming data. The database management literature appears to be among the first to consider histogram summaries for data streams. As data streams are unbounded, the amount of storage to provide an exact answer to a database query may also increase without bound (Babcock et al., 2002). This issue makes approximate query answers not only desirable, but necessary. Guha et al. (2001) develop an (arbitrarily close) approximation to the V-Optimal histogram (see Section 2.3.1), whose complexity is linear in time and polylogarithmic in space. However, they restrict attention to streams of non-negative integers, and extending their method to continuous-valued observations is not obvious. Buragohain et al. (2007) develop algorithms that aim to minimize the $L_\infty$ error of the histogram, and considers fixed width sliding windows as well as piecewise linear histograms. It is assumed that the
stream is comprised of integer valued observations, and choosing the number of bins is not formally addressed. In Donjerkovic et al. (1999) alternatives to the V-Optimal and dynamic histograms are presented and, in the latter, histograms are repartitioned whenever a chi-square test statistic falls below a specified significance level.

The fading histogram method presented in Sebastião et al. (2014b) is similar to the techniques developed in this chapter, and has been extended to the change detection and distributional comparison settings (Sebastião et al., 2014a, 2017). It is similar in the sense that their approach uses a fading factor to down-weight historical data, which is similar to the concept of a fixed forgetting factor. The fading histograms are constructed using equi-width bins, which implicitly assumes the range of the observations is known. Nonetheless, the fading histogram approach of Sebastião et al. (2014b) is a good comparison method, and is compared to TASHs in the simulations considered in Section 6.4.

The Partition Incremental Discretization (PiD) method was introduced in Gama and Pinto (2006), and is further discussed in Gama (2010, Chapter 4). The PiD algorithm consists of two layers, where the first layer simplifies and summarizes the stream, and the second layer constructs a histogram using results from the first layer. The algorithm is therefore greedy in the sense that the constructed histogram is entirely dependent on results from the first layer. Similar to Sebastião et al. (2014b), the first layer requires specifying the range of the observations and, similar to TASHs, utilizes split and merge operations. In PiD, histogram bins are split whenever the count exceeds a user specified tolerance, and mergers are triggered whenever it is deemed ‘necessary’, e.g., by a specialist seeing a need for reconstruction.

Other synopsis techniques exist in the univariate and multivariate settings, such as sketching (Yang et al., 2017), reservoir sampling (Vitter, 1985) and wavelets (Gilbert et al., 2003). Due to the vast amount of literature devoted to synopsis techniques, we do not provide a complete treatment. Instead, refer to Cormode et al. (2011) and the references therein for an excellent survey. We favor histograms due to their simplicity and their wide-spread applicability, and other synopsis structures are not considered subsequently.
6.2  NOMENCLATURE

This section establishes notation, without providing technical details on how to sequentially update a streaming histogram. The technicalities are deferred to Section 6.3. Unlike Chapters 4 and 5, where the streams were comprised of categorical observations, here we consider a continuous-valued, univariate stream given by

$$<x_1, x_2, \ldots, x_{t-1}, x_t, \ldots>.$$  

The observation $x_t \in \mathbb{R}$ is a realization of the random variable $X_t \sim F_t(\theta_t)$, where $F_t$ is an unknown distribution governed by the unknown parameter vector $\theta_t$. Observe that the distribution generating the data, as well as the parameter vector, are both allowed to vary in time, and no assumptions are placed on the parametric form of $F_t$. We remark that changes in $F_t$ would necessarily result in changes in $\theta_t$; however, drift could occur in $\theta_t$ without affecting the parametric form of $F_t$.

The histogram summary at time $t$, denoted by $\mathcal{H}_t$, is a collection of non-overlapping bins that span the range of the random variables, along with the estimated probability of an observation falling inside a particular bin. Mathematically this may be represented as a matrix whose rows represent the current histogram bins:

$$\mathcal{H}_t = \begin{pmatrix} a^{(1)}_t & b^{(1)}_t & \hat{p}^{(1)}_t \\ a^{(2)}_t & b^{(2)}_t & \hat{p}^{(2)}_t \\ \vdots & \vdots & \vdots \\ a^{(K_t)}_t & b^{(K_t)}_t & \hat{p}^{(K_t)}_t \end{pmatrix} \in \mathbb{R}^{K_t \times 3}. \quad (6.1)$$

The scalars $a^{(i)}_t, b^{(i)}_t \in \mathbb{R}$ represent the endpoints of the $i$th bin, $\hat{p}^{(i)}_t \in [0, 1]$ is the estimated probability of an observation falling inside the bin, and $K_t \in \mathbb{Z}^+$ is the number of bins at the current time-stamp. Note that the number of bins is allowed to vary in time and, as seen shortly, the bin widths are not constrained to be equiwidth. For brevity let

$$\mathcal{B}^{(i)}_t = \left( a^{(i)}_t, b^{(i)}_t, \hat{p}^{(i)}_t \right)$$

108
denote the $i^{th}$ bin (or row) of $H_t$.

The bins are defined by the half-closed interval $\left[a_t^{(i)}, b_t^{(i)}\right)$, which follows convention, and the indexing of the bins are always enumerated to satisfy

$$a_t^{(1)} < b_t^{(1)} = a_t^{(2)} < b_t^{(2)} = a_t^{(3)} < \cdots < a_t^{(K_t)} < b_t^{(K_t)}.$$  (6.2)

The enumeration in Equation (6.2) allows the histogram summary to be ordered in the sense that, for any $i < j$, the bin $B_t^{(i)}$ appears ‘above’ the bin $B_t^{(j)}$ in the matrix given in Equation (6.1). A bin $B_t^{(i)}$ is referred to as a boundary bin if $i \in \{1, K_t\}$, and an interior bin otherwise. Furthermore, we remark that our notation for the number of bins $K_t$, and the estimated bin probabilities $\hat{p}_t^{(i)}$, does not conflict with notation presented in previous chapters. This is because the number of bins in the histogram summary can be viewed as a multinomial distribution, whose number of categories change over time, and the estimated probabilities play an analogous role to the static MLEs for the multinomial distribution. In Section 6.4 the static estimates are replaced with adaptive estimates in the histogram summary, so that the histograms are capable of reacting to drift.

Since data streams experience changes in the data generating process, a way of incrementally and adaptively maintaining the histogram summary is an important, non-trivial issue. Moving forward, we suppose that $x_{t+1}$ has just arrived from the stream and a current histogram summary $H_t$ is available. In the next section a novel way of constructing $H_{t+1}$ from $H_t$ is proposed. This is done by incorporating the newest datum $x_{t+1}$ into the summary, updating the bin probabilities, and applying various bin operations.

The order in which the bin operations are applied is an important, subtle issue, that helps clarify the time subscripts on certain estimates. Upon the arrival of $x_{t+1}$, the bin probabilities are updated before any histogram operations are conducted. Hence, in upcoming sections the probabilities may have a time subscript that is one ‘step-ahead’ of other estimates. This was not done in error, and is a direct consequence of the order in which the operations are applied to the histogram summary.
6.3 Histogram construction

This section provides the technical details on sequentially computing $H_{t+1}$ from $H_t$. First, the initial histogram summary $H_0$ is constructed via a burn-in period, where $B$ observations are used to define the initial histogram bin boundaries. Once $H_0$ has been constructed, accurately maintaining the histogram summary of the data stream relies on three crucial operations: (1) boundary adjustment, (2) bin merging and (3) bin splitting. These are discussed in the next three sections.

6.3.1 Boundary adjustment

Whenever $x_{t+1}$ falls outside the boundary bins, i.e., when

$$x_{t+1} \notin \left[ a_t^{(1)}, b_t^{(K_t)} \right],$$

an adjustment to $H_t$ is required. This is likely to occur when outliers are observed, or if the data stream experiences drift. For the sake of demonstration, suppose that $x_{t+1} < a_t^{(1)}$. The most straightforward approach is to create a new leftmost boundary bin, and append it to the beginning of the histogram summary. That is, define

$$H_{t+1} = \begin{pmatrix} B_{t+1}^* \\ H_t \end{pmatrix} \in \mathbb{R}^{(K_t+1) \times 3}, \quad \text{where } B_{t+1}^* = \left( x_{t+1}, a_t^{(1)}, 0 \right).$$

Since a new bin is added $K_{t+1} = K_t + 1$, and the bin subscripts and superscripts are, respectively, reindexed with the new time-stamp and its new bin number in the updated summary – this ensures that the relation in Equation (6.2) is satisfied. An analogous update is performed when $x_{t+1} > b_t^{(K_t)}$. In this case the summary is updated as:

$$H_{t+1} = \begin{pmatrix} H_t \\ B_{t+1}^* \end{pmatrix} \in \mathbb{R}^{(K_t+1) \times 3}, \quad \text{where } B_{t+1}^* = \left( b_t^{(K_t)}, x_{t+1}, 0 \right),$$

110
and similar bookkeeping is conducted. We remark that the temporary bin $B_{t+1}^*$ has only been used to clean up the presentation of the updating of the summary, and will be used throughout the chapter whenever a temporary bin helps with clarification.

Whenever an observation lies outside the boundary bins, appending a new boundary bin to the histogram summary is a straightforward approach. However, when $x_{t+1}$ falls inside an existing bin matters become more challenging. In the next two sections methodology is developed for **merging** adjacent bins, as well as **splitting** an existing bin into two new bins. This provides histograms with the ability to reconfigure themselves in a data-driven manner, so that they can more accurately approximate a data stream’s distribution.

### 6.3.2 **Merging bins**

Whenever $x_{t+1}$ falls inside an existing bin, the bin probabilities are sequentially updated and a **merge condition** is checked. If this condition is satisfied the bin is merged with an adjacent bin. Consider the case when $x_{t+1}$ falls inside an interior bin $B_t^{(i)}$; the case when $x_{t+1}$ falls in a boundary bin is addressed at the end of the section.

The idea behind merging is that a bin will be merged with a neighboring bin if the width of the bin is below some prespecified tolerance. This requires a minimum allowed bin width to be specified. Denote this width by $\delta_t$, which is defined by

$$
\delta_t = R_t \frac{W}{W},
$$

$R_t$ being an estimate of the range of the observations, and $W > 0$ being a user defined parameter. The time subscript $t$ is appropriate, as opposed to $t + 1$, since when $x_{t+1}$ arrives these parameters have not used $x_{t+1}$ in their updating yet. Since the stream will experience drift, the range estimate $R_t$ should be computed over a sliding window. That is,

$$
R_t = \max_{t-\omega+1 \leq k \leq t} x_k - \min_{t-\omega+1 \leq k \leq t} x_k,
$$
where $\omega \in \mathbb{Z}^+$ denotes how many recent observations are used in estimating the range. Computing $R_t$ requires storing the past $\omega$ observations in computer memory, and from a practical standpoint, $\omega$ does not need to be large. This should therefore not introduce any storage issues. In fact, estimating a whole distribution is a more difficult problem than simply estimating parameters; correspondingly, more information must be stored in memory.

The bin $B_t^{(i)}$ is merged with a neighboring bin whenever the merge condition

$$b_t^{(i)} - a_t^{(i)} < \delta_t$$

(6.3)

is satisfied, that is, whenever $x_{t+1}$ falls inside a bin whose width falls below the specified tolerance. The two candidate bins for the merger are the neighbors $B_t^{(i-1)}$ and $B_t^{(i+1)}$ – since we have assumed $x_{t+1}$ fell inside an interior bin, both these neighbors exist. The $i^{th}$ bin is then merged with the neighbor that results in the smallest overall gain in width. That is, the $i^{th}$ bin is merged with $B_t^{(k^*)}$ where

$$k^* = \arg\min_{k \in \{i-1,i+1\}} \left( b_t^{(i)} - a_t^{(i)} + b_t^{(k)} - a_t^{(k)} \right).$$

Once a bin is merged, similar to the boundary adjustment case, some bookkeeping is required. For the sake of demonstration suppose that $B_t^{(i)}$ was merged with its left neighbor, i.e., $k^* = i - 1$. The merged bin is chosen to have elements

$$B_{t+1}^* = \left( a_t^{(i-1)}, b_t^{(i)}, \hat{p}_{t+1}^{(i-1)} + \hat{p}_{t+1}^{(i)} \right),$$

(6.4)

and the updated histogram is given by

$$H_{t+1} = \begin{pmatrix} B_t^{(1:i-2)} \\ B_{t+1}^{*} \\ B_t^{(i+1:K_t)} \end{pmatrix} \in \mathbb{R}^{(K_t-1) \times 3}.$$
(6.5) is short-hand for the matrix

\[ B_t^{(i:j)} = \begin{pmatrix} B_t^{(i)} \\ B_t^{(i+1)} \\ \vdots \\ B_t^{(j)} \end{pmatrix}, \]

and recall that the bin probabilities are updated before implementing the merger of a bin; therefore, the time-stamp \((t + 1)\) appearing in Equation (6.4) is appropriate. As before, all indices are reenumerated so the superscripts coincide with the corresponding bin number, the number of bins is decreased by one, and all subscripts are updated with the new time-stamp.

If an interior bin is to be merged, it merges with its neighbor that results in the smallest overall gain in bin width, takes the appropriate endpoints of the two bins, and sums the corresponding bin probabilities to create the merged bin. An analogous procedure occurs if a bin merges with its right neighbor. Selecting the neighbor for the merger may be done in various ways, and the approach taken here was chosen to avoid creating bins with a large width that would be split at a future time-stamp. In the case of a boundary bin, which only has one unique neighbor, the merge condition in Equation (6.3) is checked, and if satisfied, the boundary bin is merged with its only neighbor.

### 6.3.3 Splitting bins

Suppose that \(x_{t+1} \in B_t^{(i)}\) where, unlike Section 6.3.2, we do not need to distinguish between interior/boundary bins. As a complement to merging, we split a bin into two ‘sub-bins’ whenever its width exceeds a tolerance. The maximum allowed bin width is denoted \(\Delta_t\), and is defined as

\[ \Delta_t = \frac{R_t}{w}, \]
where \( w \in (0, W) \) is a parameter, and \( R_t \) and \( W \) were discussed in Section 6.3.2. The bin \( B_t^{(i)} \) is then split whenever the split condition

\[
b_t^{(i)} - a_t^{(i)} > \Delta_t,
\]

is satisfied. Unlike merging, where the endpoints of the merged bin were already existing endpoints in the histogram summary, splitting a bin requires more care, as there are numerous ways of constructing the sub-bins. Additionally, deciding how to divide the probability mass among the sub-bins is challenging. Both of these issues are addressed below.

Perhaps the most intuitive way of constructing the sub-bins is to split \( B_t^{(i)} \) at its midpoint, resulting in two sub-bins of equal width. However, this implicitly assumes that observations are uniformly distributed within a bin, i.e., the midpoint is the best place for a split to occur. This may not always be the case; therefore, we propose using the bin averages to aid in the construction of the sub-bins. Let \( \mu_t^{(i)} \) represent the average of all the data that fell into the \( i \)th bin, which has incorporated the newest datum \( x_{t+1} \) in its calculation. These averages can be efficiently computed on the stream using Equations (3.5)-(3.6) with a fixed forgetting factor \( \lambda = 1 \).

The idea of splitting is to create one sub-bin that has \( \mu_{t+1}^{(i)} \) as its midpoint, and another sub-bin that accommodates for the rest of the interval. To determine where to split requires a break point to be defined, which depends on the location of \( \mu_{t+1}^{(i)} \) in relation to the midpoint of the bin to be split. Since the bin average is to be the center of one of the sub-bins, the break point, denoted \( \Xi_{t+1} \), is defined piecewise as

\[
\Xi_{t+1} = \begin{cases} 
2\mu_{t+1}^{(i)} - a_t^{(i)} & \text{if } \mu_{t+1}^{(i)} < \frac{b_t^{(i)} + a_t^{(i)}}{2}, \\
2\mu_{t+1}^{(i)} - b_t^{(i)} & \text{otherwise}.
\end{cases}
\]

Choosing the break point in this fashion guarantees that \( \mu_{t+1}^{(i)} \) is the midpoint of one of the sub-bins. Thus, the original bin can be represented as the decomposition

\[
B_t^{(i)} = \left[a_t^{(i)}, \Xi_{t+1} \right] \cup \left[\Xi_{t+1}, b_t^{(i)} \right].
\]

114
The last thing to discuss is how to allocate the probability mass to the sub-bins. Unlike merging, where the bin probabilities were simply added, sharing the mass requires more thought. The simplest choice would be to evenly distribute the mass; however, this does not take into account the widths of the new sub-bins (nor is it based on any mathematical reasoning). Therefore, a criterion must be developed which yields a ‘sensible’ proportion of the mass to allocate to each sub-bin. That is, we find a $\psi_{t+1} \in (0, 1)$ such that

- $(1 - \psi_{t+1}) \hat{p}_t^{(i)}$ is allocated to $[a_t^{(i)}, \Xi_{t+1}]$,
- and $\psi_{t+1} \hat{p}_t^{(i)}$ is allocated to $[\Xi_{t+1}, b_t^{(i)}]$. 

A popular criterion used in histogram construction is that of entropy, where much research has focused on maximum entropy histograms (e.g., Collins and Wragg (1977)). Indeed, in Jaynes (1957) it was remarked that “in making inferences on the basis of partial information we must use that probability distribution which has maximum entropy subject to whatever is known. This is the only unbiased assignment we can make; to use any other would amount to arbitrary assumption of information which by hypothesis we do not have.” A goal of this chapter is to provide completely non-parametric inference for data streams using TASHs; therefore, we follow the guidance of Jaynes (1957) and choose $\psi_{t+1}$ to maximize the difference in entropy of the histogram before and after the split, as a function of $\psi_{t+1}$.

The change in entropy, denoted $\Delta E(\psi_{t+1})$, is given by

$$\Delta E(\psi_{t+1}) = \hat{p}_t^{(i)} \log \left( \frac{\hat{p}_t^{(i)}}{b_t^{(i)} - a_t^{(i)}} \right) - (1 - \psi_{t+1}) \hat{p}_t^{(i)} \log \left( \frac{(1 - \psi_{t+1}) \hat{p}_t^{(i)}}{\Xi_{t+1} - a_t^{(i)}} \right)$$

$$- \psi_{t+1} \hat{p}_t^{(i)} \log \left( \frac{\psi_{t+1} \hat{p}_t^{(i)}}{b_t^{(i)} - \Xi_{t+1}} \right),$$

where the first term corresponds to no split, and the last two terms correspond to splitting the $i^{th}$ bin. All other bins remain unchanged, and their corresponding terms vanish when computing the difference in entropy. Dif-
ferentiating Equation (6.6) with respect to $\psi_{t+1}$ yields
\[
\frac{d}{d\psi_{t+1}} (\Delta E(\psi_{t+1})) = \hat{p}_{t+1}^{(i)} \left\{ \log \left( \frac{1 - \psi_{t+1}}{\psi_{t+1}} \right) + \log \left( \frac{b_t^{(i)} - \Xi_{t+1}}{\Xi_{t+1} - a_t^{(i)}} \right) \right\},
\]
resulting in the maximizer
\[
\psi_{t+1}^* = \frac{b_t^{(i)} - \Xi_{t+1}}{b_t^{(i)} - a_t^{(i)}}.
\tag{6.7}
\]
The details of the above calculation can be found in Appendix A.5, and it is interesting to note that the amount of mass allocated to each sub-bin is the ratio of the widths of the sub-bins to the original width of the interval. This can be immediately seen from Equation (6.7) and by noticing that
\[
1 - \psi_{t+1}^* = \frac{\Xi_{t+1} - a_t^{(i)}}{b_t^{(i)} - a_t^{(i)}}.
\]

Having derived $\psi_{t+1}^*$, the sub-bins can now be expressed as
\[
\mathcal{L}_{t+1}^{(i)} = \left\{ \left( a_t^{(i)}, \Xi_{t+1}, (1 - \psi_{t+1}^*) \hat{p}_{t+1}^{(i)} \right) \right\}, \quad \mathcal{R}_{t+1}^{(i)} = \left\{ \left( \Xi_{t+1}, b_t^{(i)}, \psi_{t+1}^* \hat{p}_{t+1}^{(i)} \right) \right\},
\]
and the updated histogram summary is taken as
\[
\mathcal{H}_{t+1} = \left( \begin{array}{c} \mathcal{B}_{t+1}^{(1:i-1)} \\ \mathcal{L}_{t+1}^{(i)} \\ \mathcal{R}_{t+1}^{(i)} \\ \mathcal{B}_{t+1}^{(i+1:K_t)} \end{array} \right) \in \mathbb{R}^{(K_t+1) \times 3},
\]
where, as before, the indices are reenumerated, the bin count is increased by one and the subscripts are labelled with the new time-stamp. Figure 6.1 provides a graphical representation of what happens when a bin $\mathcal{B}_t^{(i)}$ is either merged (with its left neighbor) or split, and helps clarify notation.

At this stage, the necessary machinery has been provided to sequentially update histograms in the streaming paradigm. We remark that there are
various alternatives for merging and splitting bins. For example, when merging bins the ‘time elapsed since last occupancy’ for each bin, i.e., the amount of time passed since a datum fell in each of the bins could be monitored. Bins that have not been occupied for a sufficiently long time could then be merged with neighboring bins; however, how long to wait until mergers are triggered is not obvious. When bins are split, a maximum entropy argument is used to allocate the probability mass to the sub-bins. Alternatively, recall that the last $\omega$ observations are stored in memory to recompute the range $R_t$. The data residing in this window could be used to empirically estimate the probability mass that should be allocated to each sub-bin; however, this is more computationally expensive than the entropy approach, and the complexity will depend on the value of $\omega$ and how often splits occur. Henceforth, we favor the approaches developed in this chapter and alternative methods for merging and splitting are not considered. Furthermore, the proposed merging and splitting procedures remove the burden of having to specify the

Figure 6.1: A graphical representation of what happens when a bin $B_{t}^{(i)}$ is either merged (with its left neighbor) or split, and highlights the notation introduced throughout this chapter.
number of bins in the histogram. This is because after specifying \((w, W)\), any bin width in the interval \([\delta_t, \Delta_t]\) is allowed, and the number of bins is entirely data-driven. This is a more relaxed setting than enforcing an equiwidth strategy, which requires a constant number of bins to be specified.

6.3.4 Examples in non-drifting data streams

A simulation study is now conducted, which highlights the effectiveness of the sequential histogram approach just described, when applied to non-drifting data streams. This methodology is compared to batch histograms, which use Sturges’ rule (refer to Section 2.3.1) to choose the number of bins. This is the default behavior when using R’s \texttt{hist} function. As the focal point of this chapter is to develop TASHs that can monitor drifting streams, we defer comparing to streaming methods until Section 6.4.3.

To gauge performance, Monte Carlo replicates are implemented on streams generated from well-known distributions, whose governing parameters are randomly generated between simulations. The performance measure chosen is the average area between the true density and the histogram estimate. This measure, which is a function of a histogram estimate, is defined by

\[
A \left( \hat{f} \mid f \right) = \frac{1}{2M} \sum_{i=1}^{M} \left[ \int_{-\infty}^{\infty} \left| f(x) - \hat{f}^{(i)}(x) \right| \, dx \right],
\]

(6.8)

where \(M\) is the number of Monte Carlo replicates, \(f(\cdot)\) is the true density, and \(\hat{f}^{(i)}(\cdot)\) is the batch or sequential histogram estimate obtained after processing the observations in the stream during the \(i^{th}\) simulation. The factor of \(\frac{1}{2}\) is included so that the performance measure is rescaled to the interval \([0, 1]\).

Results, averaged over \(M = 10,000\) Monte Carlo replicates for various distributions, are given in Table 6.1. In each simulation, 2,500 observations were considered, and the average area between the curves was estimated for the batch and sequential histogram estimates. For each distribution, the average area between the true density and the sequential estimate is low, and are similar to the results obtained using the equiwidth strategy that used Sturges’
Table 6.1: The area between the true density and the sequential and batch histogram estimates, averaged over 10,000 Monte Carlo simulations for several distributions. Each simulation was run over streams of length 2,500, and each element in the table is tuple of the form $(A, \text{SE}(A))$, SE$(A)$ representing the standard error of $A$.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Sequential</th>
<th>Batch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beta</td>
<td>(0.054, 0.005)</td>
<td>(0.046, 0.007)</td>
</tr>
<tr>
<td>Exponential</td>
<td>(0.077, 0.009)</td>
<td>(0.075, 0.012)</td>
</tr>
<tr>
<td>Gamma</td>
<td>(0.083, 0.015)</td>
<td>(0.096, 0.014)</td>
</tr>
<tr>
<td>Gaussian</td>
<td>(0.065, 0.006)</td>
<td>(0.077, 0.008)</td>
</tr>
<tr>
<td>Uniform</td>
<td>(0.029, 0.006)</td>
<td>(0.062, 0.006)</td>
</tr>
<tr>
<td>Weibull</td>
<td>(0.058, 0.005)</td>
<td>(0.055, 0.013)</td>
</tr>
</tbody>
</table>

Figure 6.2: Plots of the sequential histogram introduced in Section 6.3 along with the true density (dashed line) for two distributions. Both plots represent the estimated histogram after processing the entire stream of 2,500 observations. **Left:** Beta$(2, 5)$. **Right:** $\mathcal{N}(0, 1)$.

rule to choose the number of bins. This reinforces the validity of our approach and, for illustration purposes, Figure 6.2 displays the sequential histogram estimate and the true density for a beta and Gaussian distribution.

This section has provided evidence that the sequential histogram developed in Section 6.3 provides acceptable density estimates when implemented on non-drifting data streams. However, it is impractical to assume that data streams do not exhibit any drift, reinforcing the need for an efficient way of discarding older data as newer data is observed. Since the sequential
histogram density estimates are proportional to the probabilities of a particular datum falling inside the bins, the forgetting factor framework established throughout Chapters 3-5 is applicable, and can be utilized to induce temporal adaptivity into the histograms. The next section reformulates the adaptive forgetting framework, culminating in TASHs.

6.4 INTRODUCING ADAPTIVITY IN HISTOGRAM ESTIMATION

A histogram that places equal weight on all observations will not be appropriate for data streams that experience drift. As an example, consider a stream of random variables generated according to:

\[
X_t \sim \begin{cases} 
\mathcal{N}(0,1) & \text{for } 1 \leq t \leq 2500 \\
\mathcal{N}(5,1) & \text{for } 2500 < t \leq 5000.
\end{cases} \tag{6.9}
\]

The estimated histogram displayed in Figure 6.3 incorrectly depicts a bi-modal distribution, resembling a mixture of the two Gaussians. Ideally, a histogram would efficiently estimate the \(\mathcal{N}(0,1)\) distribution for the first 2,500 observations, and autonomously learn to discard older data as realizations from the \(\mathcal{N}(5,1)\) distribution begin arriving from the stream.

The shortcomings of this toy example are due to the fact that observations are contributing equally in the estimation of the bin probabilities, resulting in a misleading (and incorrect) representation of the stream’s distribution. Hence, for the histogram to react quickly to drift requires a way of adaptively estimating the bin probabilities. Creating temporally adaptive estimates for probabilities has been a central point of this thesis, and can be achieved by incorporating forgetting factors into the estimation of the probabilities, which is the topic of the next section.

6.4.1 ADAPTIVE ESTIMATION OF BIN PROBABILITIES

Suppose that \(\mathcal{H}_t\) has been sequentially updated via the method proposed in Section 6.3, is comprised of \(K_t\) bins, and that \(x_{t+1}\) has just arrived from the
Figure 6.3: The estimated sequential histogram obtained after analyzing the stream in Equation (6.9). The first 2,500 observations were generated from a \(\mathcal{N}(0, 1)\) distribution (solid line), and the last 2,500 observations come from a \(\mathcal{N}(5, 1)\) distribution (dashed line).

stream. The collection of bin probabilities may be treated as a multinomial distribution with \(K_t\) categories, and techniques introduced throughout this thesis for adaptively estimating category probabilities can be directly implemented here. The discussion of adaptively estimating the category probabilities of a multinomial distribution has been thoroughly discussed throughout the thesis; consequently, we briefly discuss the modifications required to adjust the framework to the histogram setting.

Let \(\tilde{\mathbf{p}}_t \in \mathbb{S}^{K_t}\) represent the vector of adaptive bin probabilities at time \(t\), whose dimension is now allowed to vary in time. Suppose further that \(x_{t+1}\) falls inside an existing bin; if not, the boundary adjustments discussed in Section 6.3.1 are carried out and \(\tilde{\mathbf{p}}_{t+1}\) differs from \(\tilde{\mathbf{p}}_t\) only in dimension, where a zero is appended to either the first or last entry of the vector (depending on the value of \(x_{t+1}\)). The components of \(\tilde{\mathbf{p}}_{t+1}\), denoted by \(\tilde{p}_{t+1}^{(i)}\), are sequentially updated according to

\[
\begin{align*}
  n_{t+1} &= \lambda_t n_t + 1, \\
  \tilde{p}_{t+1}^{(i)} &= \left(1 - \frac{1}{n_{t+1}}\right) \tilde{p}_t^{(i)} + \frac{1}{n_{t+1}} \mathbb{1}\left(x_{t+1} \in \mathcal{B}_t^{(i)}\right),
\end{align*}
\]

where \(\mathbb{1}\left(x_{t+1} \in \mathcal{B}_t^{(i)}\right) = 1\) if and only if \(a_t^{(i)} \leq x_{t+1} < b_t^{(i)}\), and \(\lambda_t \in (0, 1)\) is an adaptive forgetting factor. Equations (6.10)-(6.11) are similar to the recursive
update equations presented in Chapter 4-5, the only difference being the indicator function. The indicator functions in Chapters 4 and 5, respectively, depended on which category or state transition was currently observed. In the present context, the indicator function now depends on which bin the most recent datum resides.

Applying the stochastic gradient descent method to tune the forgetting factors for histograms is less obvious than in previous chapters. This is because previous chapters considered categorical data streams, where the observations were either assumed independent, or satisfied a first-order Markov property. Thus, likelihood functions were known and could be used to define the cost function that the stochastic gradient descent method sought to optimize. In the current chapter we are concerned with non-parametric inference, where no distributional assumptions are made. Thus, the likelihood of the data is unknown.

As the distribution of the data is unknown, a natural way of estimating its likelihood is via the sequential histogram developed in Section 6.3. At time $t$, and for any $x \in \mathbb{R}$ inside the current histogram’s range, the density estimate is given by

$$\tilde{f}(x \mid \tilde{p}_t) = \sum_{i=1}^{K_t} \left( \frac{\tilde{p}^{(i)}_t}{b^{(i)}_t - a^{(i)}_t} \right) \mathbb{1}(x \in \mathcal{B}^{(i)}_t).$$

The forgetting factors are then updated according to

$$\lambda_{t+1} = \lambda_t + \eta \nabla \left( \log \left[ \tilde{f}(x_{t+1} \mid \tilde{p}_t) \right] \right), \quad (6.12)$$

where $\log \left[ \tilde{f}(x_{t+1} \mid \tilde{p}_t) \right]$ is now an estimate of the one-step ahead log-likelihood function, and $\eta$ is the step-size used in the stochastic gradient descent step. If a datum falls outside the range of the histogram, the boundary adjustments discussed in Section 6.3.1 are implemented, and the forgetting factor is not updated since the new boundary bin’s probability is initialized to zero.

Taking the scalar derivative of Equation (6.12) with respect to $\lambda$, as discussed in Section 3.4.2, results in
\[
\lambda_{t+1} = \lambda_t + \eta \left( \sum_{i=1}^{K_t} \frac{p^{(i)}_t}{b^{(i)}_t - a^{(i)}_t} \mathbb{I} \left( x_{t+1} \in B^{(i)}_t \right) \right), \tag{6.13}
\]
where the gradients can be recursively updated by appealing to
\[
\nabla n_{t+1} = \lambda_t \nabla n_t + n_t, \tag{6.14}
\]
\[
\nabla \tilde{p}^{(i)}_{t+1} = \left( 1 - \frac{1}{n_{t+1}} \right) \nabla \tilde{p}^{(i)}_t - \frac{\nabla n_{t+1}}{n_{t+1}^2} \left( \mathbb{I} \left( x_{t+1} \in B^{(i)}_t \right) - \tilde{p}^{(i)}_t \right). \tag{6.15}
\]

Although Equation (6.13) looks computationally expensive, knowing which bin \(x_{t+1}\) belongs to drastically simplifies the updating of the forgetting factor. For example, if \(x_{t+1} \in B^{(i)}_t\) then the gradient of the cost function is simply \(\nabla \tilde{p}^{(i)}_t / \tilde{p}^{(i)}_t\), which agrees with the update equations given in previous chapters.

The final point to discuss is how the gradients in Equations (6.14)-(6.15) are adjusted when merging or splitting a bin occurs. Since differentiation is a linear operation, and merging or splitting bins results in either summing, or appropriately rescaling bin probabilities (refer to Sections 6.3.2-6.3.3), the same operations may be applied to the gradients to obtain the gradient of the merged bin, or the gradients of the two sub-bins. That is, supposing that the bin \(B^{(i)}_t\) was merged with its left neighbor, the resulting bin would have a gradient given by
\[
\nabla \left( p^{(i-1)}_{t+1} + \tilde{p}^{(i)}_{t+1} \right) = \nabla \tilde{p}^{(i-1)}_{t+1} + \nabla \tilde{p}^{(i)}_{t+1},
\]
which requires no extra computation (besides one addition), as the right hand side was computed once \(x_{t+1}\) was observed to fall into the \(i^{th}\) bin. Similarly, if \(B^{(i)}_t\) were split, the gradients of the sub-bins require no extra computation, and are given by
\[
\nabla \left( (1 - \psi_{t+1}^*) p^{(i)}_{t+1} \right) = (1 - \psi_{t+1}^*) \nabla \tilde{p}^{(i)}_{t+1} \quad \text{(Left sub-bin)},
\]
\[
\nabla \left( \psi_{t+1}^* \tilde{p}^{(i)}_{t+1} \right) = \psi_{t+1}^* \nabla \tilde{p}^{(i)}_{t+1} \quad \text{(Right sub-bin)},
\]
where $\psi_{t+1}$ was defined in Equation (6.7). With these modifications, the static bin probabilities used in Section 6.3 are now replaced with the adaptive bin probabilities developed in this section. Everything else presented in Section 6.3 remains unchanged.

The toy example governed by Equation (6.9) is now revisited using adaptive bin probabilities. The histogram estimates for the true density are displayed in Figure 6.4 for various time-stamps. At the moment of the change (top-left plot) the $\mathcal{N}(0,1)$ distribution is estimated well by the histogram. After the changepoint, unlike in Figure 6.3, the adaptive forgetting factors allow for the quick discarding of previous, non-informative data points. This allows for the $\mathcal{N}(5,1)$ distribution to be efficiently estimated after the change occurs.

Additionally, the simulations whose results were provided in Table 6.1 have been rerun using TASHs. These results are provided in Table 6.2, where the batch results obtained using R’s default histogram option are redisplayed for convenience. From Table 6.2 we see that there has been a slight increase in the area between the true density and the TASHs. This is expected, since introducing adaptivity essentially induces volatility into the estimation procedure. However, as seen from Figure 6.4, this slight modification allows for the histogram to quickly react to drift, and is able to provide accurate density estimates in drifting, and non-drifting data streams.

<table>
<thead>
<tr>
<th></th>
<th>TASHs</th>
<th></th>
<th>Batch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beta</td>
<td>(0.090, 0.017)</td>
<td></td>
<td>(0.046, 0.007)</td>
</tr>
<tr>
<td>Exponential</td>
<td>(0.099, 0.015)</td>
<td></td>
<td>(0.075, 0.012)</td>
</tr>
<tr>
<td>Gamma</td>
<td>(0.097, 0.013)</td>
<td></td>
<td>(0.096, 0.014)</td>
</tr>
<tr>
<td>Gaussian</td>
<td>(0.093, 0.015)</td>
<td></td>
<td>(0.077, 0.008)</td>
</tr>
<tr>
<td>Uniform</td>
<td>(0.078, 0.020)</td>
<td></td>
<td>(0.062, 0.006)</td>
</tr>
<tr>
<td>Weibull</td>
<td>(0.090, 0.016)</td>
<td></td>
<td>(0.055, 0.013)</td>
</tr>
</tbody>
</table>

**Table 6.2:** The area between the true density and the TASHs and batch histograms, averaged over 10,000 Monte Carlo simulations for several distributions. Each simulation was run over streams of length 2,500, and each element in the table is tuple of the form $(A, SE(A))$, $SE(A)$ representing the standard error of $A$.

Due to space limitations, and the difficulties in adequately displaying a time-varying histogram in figures, an R Shiny app has been developed and is
Figure 6.4: The data stream governed by Equation (6.9) is revisited using the adaptive forgetting factor scheme developed in this section. The top-left plot is a snapshot taken at time $t = 2500$ (at the time of the change), and the remaining plots on the top row are were taken at times $t = 2550, 2600$. The bottom row shows plots associated with time-stamps $t = 2700, 3500, 4000$ and highlight that the data from the pre-change distribution is effectively discarded, allowing for the post-change distribution to be accurately estimated.

publicly available at:


Appendix B presents the GUI for the R Shiny app, and discusses how it may be used to produce videos of TASHs.

An entire framework has now been presented, allowing for the adaptive estimation of a data stream’s drifting density using TASHs. In the next section, a performance measure is introduced, which is used to compare the TASHs with other methods in the simulation study of Section 6.4.3.
6.4.2 Measuring performance

Unlike Chapters 4 and 5, where detecting changepoints was the primary concern, this chapter’s primary focus is on TASHs. The performance measures $\text{ARL}_0$, $\text{ARL}_1$, CCD and DNF are therefore not suitable for the current setting, and a new measure must be considered.

Much of the existing literature focuses on constructing histograms to (approximately) minimize some quantity. Perhaps the most popular approach is to minimize a variant of the mean square error (MSE) of the histogram. For example, the V-Optimal histogram discussed in Section 2.3.1 chooses bin endpoints to exactly minimize the MSE (using a computationally expensive method), and the approach in Sebastião et al. (2014b) chooses the number of bins to adhere to a bound on the MSE. Therefore, there is bias in using the MSE as a performance measure (which is arguably the most obvious choice), as many competing methods were designed specifically to control this quantity. With this in mind, we continue to favor the area between the true density and the histogram estimate, as discussed in Section 6.3.4, to measure performance. However, slight adjustments are required.

It has been observed that TASHs can accurately estimate non-drifting distributions using static or adaptive bin probabilities. Thus, we propose computing $\mathcal{A}(\mathcal{H})$, as defined in Equation (6.8) for some histogram estimate $\mathcal{H}$, immediately after a change occurs. This will quantify how long the histograms take to adapt to changes in the data generating process. This is done by computing the performance measure at various ‘snapshots’ after a change occurs. Specifically, suppose a data stream experiences a change at time $t = \tau$. Then $\mathcal{A}(\mathcal{H})$ is computed at the time-stamps

$$\{\tau + 50, \tau + 100, \ldots, \tau + 1000\},$$

giving us an indication as to how quickly a streaming histogram can adapt to the drift, and begin accurately estimating the post-change distribution.
6.4.3 Simulations in drifting data streams

This section presents results from a set of Monte Carlo simulations, where the data streams experience a change in the data generating process, and highlights the capability of TASHs. Specifically, \( M = 2500 \) simulations are considered, and each stream is comprised of 5,000 observations – the first 2,500 being generated from a pre-change distribution, and the last 2,500 being sampled from a post-change distribution. Since the purpose of this section is to inspect how quickly the histograms react to drift, longer streams need not be considered, and Table 6.3 provides the specific distributions used. Simulations (1)-(3) consider streams of Gaussian random variables where both changes in location and scale are considered. Simulations (4)-(5) consider, respectively, drift in the beta and uniform distributions; whereas, simulation (6) experiences a complete distributional change (switching from uniform to gamma).

<table>
<thead>
<tr>
<th>Simulation number</th>
<th>Pre-change</th>
<th>Post-change</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>( \mathcal{N}(0, 1) )</td>
<td>( \mathcal{N}(2, 1) )</td>
</tr>
<tr>
<td>(2)</td>
<td>( \mathcal{N}(0, 1) )</td>
<td>( \mathcal{N}(10, 1) )</td>
</tr>
<tr>
<td>(3)</td>
<td>( \mathcal{N}(0, 3) )</td>
<td>( \mathcal{N}(0, 0.5) )</td>
</tr>
<tr>
<td>(4)</td>
<td>Beta(0.5, 1)</td>
<td>Beta(2, 5)</td>
</tr>
<tr>
<td>(5)</td>
<td>Unif((-1, 1))</td>
<td>Unif((1, 10))</td>
</tr>
<tr>
<td>(6)</td>
<td>Unif((0, 1))</td>
<td>Gamma((2, 0.5))</td>
</tr>
</tbody>
</table>

Table 6.3: A table outlining the distributions considered in the simulations. Streams of length 5,000 are considered, where the first 2,500 observations are sampled from the pre-change distribution, and the last 2,500 observations are sampled from the post-change distribution. The simulation number is also provided, and matches the number appearing in the top left of the plots in Figure 6.5.

There is currently no state-of-the-art method for maintaining an accurate histogram estimate in the streaming setting; consequently, we compare our approach to two natural choices: (1) a sliding window based histogram and (2) the fading histogram method developed in Sebastião et al. (2014b). These methods are discussed next.
Arguably the most straightforward approach is to compare TASHs with histograms that are constructed using a fixed-width sliding window. Similar to Section 6.3.4, the histograms are constructed using an equiwidth strategy, and Sturges’ rule is used to choose the number of bins. How often to rebuild the histogram is not obvious, and depends on how quickly observations are generated by the stream; further, choosing the width of the sliding window is non-trivial. However, as this section is for illustration purposes, the sliding histograms are constructed using a window of size 200, and it is rebuilt at the ‘snapshots’ where the performance measures are computed.

The fading histogram method presented in Sebastião et al. (2014b) is similar to our approach, and was discussed in Section 6.1. This method requires two parameters to be specified: the range of the observations, and an admissible error for the MSE of the histogram. Once chosen, the number of bins can be computed as a function of these parameters, and equiwidth bins are used to partition the range. Their construction guarantees that the resulting histogram will have an MSE at most the chosen admissible error. The authors have suggested that a fixed fading factor 0.997, and an admissible error that results in 20 equiwidth bins, are sensible choices – these recommendations being used in the simulations in this section.

Assuming the range is known is problematic in streaming data applications, since it implicitly assumes knowledge of the evolution of the stream. Consequently, in the simulations we provide the fading histogram method with the minimum/maximum observations over the entire stream, allowing for the correct partitioning of the range into equiwidth bins. It should be noted that this is unrealistic, as the minimum and maximum observations will not be known in advance; however, allowing access to the range is simpler from an implementation perspective, as there is no guidance into how to handle an unknown, or dynamic range in Sebastião et al. (2014b).
We remark that if the bin operations developed in Sections 6.3.1-6.3.3 were replaced with an equiwidth bin strategy, and a fixed forgetting factor is used, TASHs are essentially the same as the fading histogram approach. Thus, if TASHs are able to out perform fading histograms, this provides evidence that: (1) adaptive forgetting factors are useful and (2) merging/splitting bins based on our techniques has merit in streaming data settings.

Results

All the results analyzed in this section, as well as throughout the chapter, have used \((\eta, \omega, w, W) = (10^{-4}, 50, 5, 20)\). The results of the simulations are displayed in Figure 6.5, where the number appearing at the top left of each plot corresponds to the simulation number provided in Table 6.3. The values displayed on the \(x\)-axis correspond to \(\tau + 50x\) for \(x \in \{1, 2, \ldots, 20\}\), representing the set of snapshots where the performance measures were computed. The \(y\)-axis is the performance measures computed at each snapshot, averaged over the Monte Carlo simulations.

In each simulation the TASHs react to the drift quicker than the fading histograms. This is to be expected, since the fading histograms forget data at a fixed rate, and cannot adaptively modify the histograms bins. This coincides with an earlier discussion, where it was argued that the forgetting factors, and the number of bins and their widths, should be chosen in a data-driven manner. In some cases the sliding histograms react quicker to the drift when compared with the TASHs. However, recall that these histograms force a hard decision rule on which observations are currently contributing to the histogram. In the present context, once 200 data points are observed after \(\tau\), the sliding histograms are constructed using only data from the post-change distribution. It is interesting to note, however, that the TASHs attain smaller values for \(A(H)\) after 500 observations (or fewer) have been observed from the post-change distribution.

These results provide us with valuable conclusions. First, adaptive forgetting factors result in histograms that can react to drift quicker than fading histograms, which disregard data at a fixed rate. Second, in conjunction with
adaptive forgetting, the techniques for merging and splitting result in histograms that are able to provide accurate density estimates in the presence of drift. This provides evidence that adaptive forgetting, as well as our proposed merging/splitting techniques are valuable in the drifting data setting. Further, it can be speculated that the TASHs performing better than fading/sliding histograms can be attributed to using adaptive bin widths – an understudied topic in the literature – since the histogram is not constrained to introduce many equiwidth bins to accommodate the drift. Finally, although sliding histograms in some cases react to drift quicker, it does so based on heuristics, and in practical applications choosing the width of the
window, and when to rebuild the histogram, will not be obvious.

6.5 Discussion

This chapter developed sequential histograms that can be maintained in the streaming paradigm. This required novel techniques for adjusting the histogram boundary, as well as merging/splitting existing bins. This technique was compared to a histogram that used Sturges’ rule to choose the number of bins, and was shown to perform favorably in non-drifting environments.

Temporal adaptivity was then introduced by including adaptive forgetting factors in the estimation of the bin probabilities. This required using the histogram estimate as an approximate likelihood in the stochastic gradient descent step, and culminated in TASHs – the main contribution of the chapter. A simulation study verified the effectiveness of TASHs, where the histograms were shown to react to changes quicker than sliding and fading histograms. Furthermore, implementing TASHs (non-optimally) in the R programming language resulted in roughly 5500 observations being processed per second, reinforcing that TASHs are suitable for streaming data. This timing estimate was obtained by averaging run-times over 500 data streams, and the machine used was a 2017 MacBook Pro running Mojave version 10.14.6, with a 2.5 GHz Intel Core i7 processor.

From a practical stand-point, TASHs require the specification of:

- $0 < \eta \ll 1$ – the step-size used in stochastic gradient descent,
- $\omega \in \mathbb{Z}^+$ – the number of past observations stored in a sliding window to estimate the current range of the observations,
- $W > 0$ – a parameter that specifies the minimum allowed bin width,
- $0 < w < W$ – a parameter specifying the maximum allowed bin width.

Choosing the step-size $\eta$ has been discussed in this thesis, and empirical evidence has been provided for choosing its value in practice. The parameter
\( \omega \) may be chosen liberally since \( \omega \) does not define a hard window in which the histogram summary is constructed from, but rather, a sliding window used to estimate the range. This is an important distinction, as many existing methods rely on fixed width windows to construct their histograms, which will rarely be adequate for continuously evolving data streams. Choosing \((w,W)\) is similar to choosing the number of bins in a histogram. However, we argue that choosing \((w,W)\) is less critical than choosing the number of bins in a histogram. This is because after specifying these parameters, any bin width in the interval \([\delta, \Delta]\) is admissible. This is a more relaxed setting than an equiwidth approach. Further, the number of bins do not need to be specified by the user since the number of bins is entirely data-driven.
Several applications are now presented, which are consequences of the TASHs developed in Chapter 6. Our intent is to demonstrate that TASHs can be applied across a wide-range of problems, such as: quantile estimation, change detection, two-sample inference, ROC curves and outlier detection, as opposed to investigating several real-world applications. Further, we focus solely on the applications, and exhaustively comparing the methodology to existing methods is not in the scope of the chapter.

This chapter proceeds as follows: Section 7.1 discusses how to estimate quantiles using TASHs. Estimating quantiles from TASHs is a novel contribution, and other sections utilize these estimates. In Section 7.2 the change detection problem is revisited, and a non-parametric change detection method is presented. Section 7.3 develops a framework for two sample inference, which can be used to determine if two data streams are being generated by different processes. This has promise in various statistical stream mining settings; e.g., the data stream could be comprised of residuals generated by some model, and the methodology could be used to assess degradation of modeling assumptions. Section 7.4 develops dynamic ROC curves, while Section 7.5 considers the detection of outliers in the streaming setting.
7.1 Computing quantiles using TASHs

This section discusses how to sequentially estimate quantiles – values that partition the support of a random variable into regions that contain pre-specified proportions of the data. An overview of the literature is discussed in Section 7.1.1, and mathematical details for estimating the quantiles are provided in Section 7.1.2.

7.1.1 Literature review

Estimating quantiles in streaming environments has received a fair amount of attention in the computer science literature. For example, Greenwald et al. (2001) provide error guarantees on estimates, which is based on how much data needs to be stored in memory. Refer to Wang et al. (2013), who compare existing quantile methods in the computer science literature.

Tierney (1983) proposed a sequential quantile estimate that is based on stochastic approximation, and in non-drifting streams, Tierney’s estimate and the sample quantile are shown to behave similarly. However, this behavior need not hold in drifting data streams. Chen et al. (2000) modify the work of Tierney (1983), and combine stochastic approximation with EWMA (see Section 2.4.2) to develop a quantile estimate. Their estimate discards data at a fixed rate, and can be used to estimate a single quantile. Other approaches include Lin et al. (2004) who use sliding windows to approximate quantiles, and Zhang and Wang (2007) who partition the stream into sub-streams, and use merge and sample operations to approximate quantiles.

7.1.2 Estimating quantiles

Let $\phi \in [0, 1]$ and $F_t$ be the distribution function currently generating the data. In this section details are provided for estimating the $\phi$-quantile of $F_t$, denoted $q_{t, \phi}$, which satisfies the equation

$$F_t(q_{t, \phi}) = P(X_t \leq q_{t, \phi}) = \phi.$$
If the distribution were known, $q_{t, \phi}$ is given by

$$q_{t, \phi} = \inf \{ x \in \mathbb{R} : \phi \leq F_t(x) \},$$

which is generally not available in closed form and requires approximation. Furthermore, in the streaming paradigm estimates cannot be obtained via sorting, since the stream continuously generates a large amount of data.

In Chapter 6 the histogram bins were enumerated to satisfy the inequalities given in Equation (6.2). Due to this, quantile estimates can be obtained by \textit{cumulatively summing the bin probabilities}. Let

$$\tilde{F}_t(k) = \sum_{i=1}^{k} \tilde{p}_t^{(i)}, \quad k \in \{1, 2, \ldots, K_t\},$$

represent the partial, cumulative summations, and note that $\tilde{F}_t(k)$ is an approximation of the proportion of data that lies to the left of the $k$th bin’s right endpoint, i.e.,

$$\tilde{F}_t(k) \approx \mathbb{P}(X_t \leq b_t^{(k)}).$$

Given the target proportion $\phi$, a straightforward estimate of $q_{t, \phi}$, denoted $\tilde{q}_{t, \phi}$, is given by

$$\tilde{q}_{t, \phi} = \frac{a_t^{(k^*)} + b_t^{(k^*)}}{2}, \quad (7.1)$$

where

$$k^* = \arg\min_{k \in \{1, 2, \ldots, K_t\}} \left| \phi - \tilde{F}_t(k) \right|. \quad (7.2)$$

The estimate of the quantile is therefore given by the midpoint of the bin whose partial cumulative sum is closest to the proportion $\phi$. We remark that there are various other ways of choosing the estimate; however, Equation (7.1) is sufficient for our purposes.

The number of bins will always be significantly smaller than the amount of data being processed, thus the computation of Equation (7.2) should not be a burden in practice. If required, the equation could be modified so we do not optimize over all bin indices. This can be done by realizing that we only need to cumulatively sum until we find the first bin index whose sum exceeds
\( \phi \). That is, sum until we find the unique bin index \( k \) such that

\[
\tilde{F}_t(k - 1) < \phi, \quad \tilde{F}_t(k) \geq \phi,
\]

and choose the bin index (either \( k - 1 \) or \( k \)) whose partial cumulative sum is closer to \( \phi \). This could be further modified given the quantile that is being approximated. That is, if \( \phi \leq 0.5 \) we begin summing from the ‘left’ until we find the first bin’s cumulative sum that exceeds \( \phi \). A similar argument can be made when \( \phi > 0.5 \).

Figure 7.1 displays the median estimates \( \tilde{q}_{t,0.5} \), averaged over 100 Monte Carlo simulations for gamma distributions in drifting and non-drifting settings. The solid lines correspond to \( \tilde{q}_{t,0.5} \), and the dotted lines are the averaged sample medians computed over a window that used all previous observations in its calculation. The asterisks in the plots correspond to the median estimates of the gamma distribution provided in Banneheka and Ekanayake (2009), which assumes the mean of the distribution is known, and are included for reference. The top plot of Figure 7.1 shows that the adaptive quantile \( \tilde{q}_{t,0.5} \) accurately approximates the target quantile in non-drifting streams. Similarly, the bottom plot of Figure 7.1 highlights the ability of the adaptive quantile to react to changes in the data generating process.

This methodology can be extended to estimating multiple quantiles. That is, given a vector of proportions \( \phi \), a vector of adaptive quantiles \( \tilde{q}_{t,\phi} \) may be computed. This is done by computing Equations (7.1)-(7.2) for each component of \( \phi \), and the details are provided in Section 7.2. Estimating multiple quantiles has several rewarding consequences, e.g.:

- given an \( \alpha \in [0,1] \), choosing \( \phi = (\alpha/2, 1 - \alpha/2)^\top \) allows for an adaptive ‘quantile interval’ to be constructed, which encompasses approximately 100(1 - \( \alpha \))% of the data and,

- letting \( \phi = (0.25, 0.50, 0.75)^\top \) results in estimates for the median and interquartile range – all the ingredients required to construct temporally evolving box-plots, which could be used as control charts, or aid in the detection of outliers.
Figure 7.1: Figures showing $\tilde{\theta}_{0.5}$ in drifting and non-drifting environments. In both plots, the estimates were averaged over 100 Monte Carlo simulations, and the solid lines display $\tilde{\theta}_{0.5}$; whereas the dashed lines correspond to the averaged sample median computed over a sliding window. Top: a Gamma(9, 1/4) distribution. Bottom: a drifting gamma distribution that changes from a Gamma(9, 1/4) to a Gamma(5, 2) distribution at time $\tau = 500$. The asterisks in the plots represent the median estimates provided in Banneheka and Ekanayake (2009) for the gamma distribution, and are included for reference.

In the statistical quality control literature (Montgomery, 2007) many control charts use distributional assumptions to construct control limits. Similarly, in Chapter 5 a moment matching technique was used when developing ADEPT-M. This was because the distribution of the adaptive transition probabilities is unknown, and constructing probabilistic control limits required simplifying assumptions. However, now we may use TASHs to estimate quantiles, allowing for the computation of adaptive control limits, which do not place any assumptions on the data stream’s distribution. Furthermore, computing the quantiles comes at a small increase in computation, since we only need to compute the bin whose cumulative sum is closest to a particular quantile.
of interest, and compute the bin’s midpoint.

As an example, Figure 7.2 shows an estimated 95% adaptive quantile interval for a drifting data stream, which is generated according to

\[
X_t \sim \begin{cases} 
\mathcal{N}(0, 1) & t \in [1, 1250], \\
\mathcal{N}(3, 1) & t \in (1250, 2500], \\
\mathcal{N}(0, 1) & t \in (2500, 3750], \\
\mathcal{N}(-3, 1) & t \in (3750, 5000].
\end{cases}
\]

The quantile estimates were averaged over 100 Monte Carlo simulations, and the average coverage, i.e., the proportion of observations residing inside the limits, was 0.9478. Even for data streams that experience several changes in the data generating process, the adaptive quantile interval is able to closely approximate the desired coverage of 0.95. A similar illustration is provided in Figure 7.3, where the data stream experiences changes in the parametric form of the distribution, switching between Gaussian and uniform distributions with changes in location and scale. In this case, the adaptive quantiles provide an average coverage of 0.9402, which is again close to the desired coverage, and highlights the effectiveness of the novel streaming quantiles proposed in this section. In the next section the adaptive quantiles are used in the development of a non-parametric change detector.

7.2 A non-parametric change detector

In this section a change detector is developed, which is similar to the MCDMs presented in Chapter 4, and uses the quantile estimates of Section 7.1. The idea is to compare the vector of estimated quantiles at the current time-stamp to the quantiles estimated at a previous time-stamp. This is analogous to comparing vectors of adaptive quantiles obtained at the start and end of a sliding window.

Assume that a vector of proportions \( \phi \) has been specified, and let \( \tilde{q}_{t,\phi} \) represent the most recent estimate of the quantiles. The \( i^{th} \) component of these
Adaptive quantile estimates $\tilde{q}_{t, \phi}$, averaged over 100 Monte Carlo simulations, for the quantiles $\phi = (0.025, 0.975)^\top$, resulting in an approximate 95% quantile interval for $X_t$. Averaging over the Monte Carlo simulations resulted in a coverage of 0.9478, which despite the drift, is close to the desired coverage.

Similar to Figure 7.2, but now the data stream switches between Gaussian and uniform distributions with changes in both location and scale. The coverage, averaged over 100 Monte Carlo simulations, is 0.9402.

Vectors are denoted by $\phi^{(i)}$ and $\tilde{q}_{t, \phi}^{(i)}$, where we have omitted the dependence of the quantile on $\phi^{(i)}$ as it will be clear from context, and simplifies notation. Let $I_t$ be an index set, which has the same cardinality as $\phi$, and contains the indices corresponding to which bin’s midpoint was used in the estimation of the quantiles. That is, the components of $I_t$ are given by

$$I_t^{(i)} = \arg \min_{k \in \{1, \ldots, K_t\}} |\phi^{(i)} - \tilde{F}_t(k)| \quad \forall i = 1, 2, \ldots, |\phi|, \quad (7.3)$$
\(|\phi|\) being the number of elements in the vector \(\phi\). Although notationally cumbersome, this is just the vectorized version of Equation (7.2), and is necessary since we are estimating multiple quantiles. The quantile estimate \(\tilde{q}_t^{(i)}\) is then given by the midpoint of the bin whose index is given by \(\mathcal{I}_t^{(i)}\). In addition to the quantile estimates, the widths of the bins that were used in estimating the quantiles are also required for the detector. Let \(\ell_t \in \mathbb{R}^{|\phi|}\) denote the vector containing these widths, whose \(i^{th}\) component is

\[
\ell_t^{(i)} = b_t^{(x_t^{(i)})} - a_t^{(x_t^{(i)})}, \quad \forall i = 1, 2, \ldots, |\phi|.
\]

From a computational standpoint, the vector \(\mathcal{I}_t\) does not need to be stored in memory, and only helps with notation. In practice, you simply need to find the bins that satisfy Equation (7.3), and store the corresponding midpoints and widths. Lastly, let \(\omega \in \mathbb{Z}^+\) define the width of the sliding window, so that we are comparing how similar the current vector of quantiles \((\tilde{q}_t, \phi)\) is to a historical vector of quantiles \((\tilde{q}_{t-\omega}, \phi)\).

The detectors in Chapter 4 required a dissimilarity measure \(A(\cdot, \cdot)\), which could be bounded above by a function \(u(\cdot, \cdot)\). In the current context, this requires defining a dissimilarity measure, and upper bound, so that

\[
A(\tilde{q}_{t, \phi}, \tilde{q}_{t-\omega, \phi}) \leq u(\tilde{q}_{t, \phi}, \tilde{q}_{t-\omega, \phi}).
\]

Euclidean distance is used for the dissimilarity measure, so that

\[
A(\tilde{q}_{t, \phi}, \tilde{q}_{t-\omega, \phi}) = \|\tilde{q}_{t, \phi} - \tilde{q}_{t-\omega, \phi}\|_2.
\]

Computing the upper bound relies on a two assumptions; mainly,

- the bins satisfying Equation (7.3) ‘trap’ the quantiles, i.e.,

\[
q_t^{(i)} \in \left[ b_t^{(x_t^{(i)})}, a_t^{(x_t^{(i)})} \right], \quad \forall i = 1, 2, \ldots, |\phi|,
\]

- and the quantiles have not changed in the last \(\omega\) time-stamps, i.e.,

\[
q_t^{(i)} = q_{t-\omega}^{(i)}.
\]
Under these assumptions, an upper bound is given by

\[ A(\check{q}_{t,\phi}, \check{q}_{t-\omega,\phi}) \leq \frac{1}{2} \| \ell_t + \ell_{t-\omega} \|_2, \]  

which is derived in Appendix A.6. A change is flagged at time \( t \) whenever

\[ A(\check{q}_{t,\phi}, \check{q}_{t-\omega,\phi}) > \frac{\beta}{2} \| \ell_t + \ell_{t-\omega} \|_2, \]

where \( \beta \) is an allowance parameter, analogous to those used in Chapter 4. Once a changepoint is flagged a grace period of length \( G \in \mathbb{Z}^+ \) begins, and no subsequent changepoints are monitored for during this period.

There is a subtle difference between the change detector developed here and the MCDMs. In Chapter 4 legitimate upper bounds were computed; whereas, the bound in Equation (7.4) is based on two assumptions. If these assumptions are not satisfied, the upper bound may fail to hold. This may occur when: the quantiles do change in the sliding window, which will likely be the result of a distributional change, or when the bins do not consistently trap a majority of the true quantiles, which may be a good indicator for when to rebuild the histogram. Since the upper bound in Equation (7.4) is based on assumptions, it is advantageous to choose the allowance \( \beta \geq 1 \), as opposed to Chapter 4, where \( \beta \in (0,1) \).

Figure 7.4 provides an example of the quantile detector, where the stream consists of Gaussian observations that experience a change in variance at time \( \tau = 1000 \). The top figure plots the data stream, and the bottom figure plots the dissimilarity measure (solid line), as well as the adaptive threshold with \( \beta = 1.5 \) (dashed line). When no drift is present the dissimilarity measure remains below the adaptive threshold, and the threshold is exceeded shortly after the occurrence of the changepoint. We remark that the dissimilarity measure remains above the threshold for a brief period of time; however, due to the grace period, this should not raise several false detections. A simulation study was also conducted, and the estimated performance measures are displayed in Table 7.1. An analysis of the results is omitted, since remarks similar to those in Chapters 4-5 apply to Table 7.1.
Figure 7.4: An illustration of the streaming quantile detector developed in this section. Top: a plot of the data stream, which is sampled from a Gaussian distribution that experiences a change in variance. Bottom: the dissimilarity measure (solid line) and the adaptive threshold (dashed line) used for detecting changepoints.

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>ARL$_0$</th>
<th>ARL$_1$</th>
<th>CCD$_5$</th>
<th>DNF$_5$</th>
<th>CCD$_{10}$</th>
<th>DNF$_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.25</td>
<td>3411.92</td>
<td>16.69</td>
<td>0.99</td>
<td>0.56</td>
<td>0.99</td>
<td>0.52</td>
</tr>
<tr>
<td>1.50</td>
<td>4323.16</td>
<td>17.29</td>
<td>0.98</td>
<td>0.63</td>
<td>0.98</td>
<td>0.57</td>
</tr>
<tr>
<td>2.00</td>
<td>4724.21</td>
<td>21.25</td>
<td>0.94</td>
<td>0.68</td>
<td>0.94</td>
<td>0.62</td>
</tr>
<tr>
<td>2.50</td>
<td>4834.87</td>
<td>26.45</td>
<td>0.89</td>
<td>0.72</td>
<td>0.89</td>
<td>0.65</td>
</tr>
</tbody>
</table>

Table 7.1: Results, averaged over 1000 simulations, for the streaming quantile change detector. The upper section displays results for Gaussian streams whose mean and variance both drift, and the bottom portion displays results from various beta streams.
In this thesis methodology has been developed for analyzing a single data stream. We now consider situations where two data streams are observed, and develop procedures to determine whether the streams are being generated by different stochastic processes.

It is assumed that an observation arrives from each data stream at each time-stamp, and

\[ (x_1, x_2, \ldots, x_{t-1}, x_t, \ldots), \]  
\[ (y_1, y_2, \ldots, y_{t-1}, y_t, \ldots), \]  

for \( x_t, y_t \in \mathbb{R} \), denote the two streams of interest. We are interested in investigating the hypothesis

\[ H_0 : F_t = G_t, \quad \text{vs.} \quad H_1 : F_t \neq G_t, \]  

where \( F_t \) and \( G_t \) are, respectively, the distributions generating the streams in Equation (7.5) and Equation (7.6) at time \( t \), i.e., \( X_t \sim F_t \) and \( Y_t \sim G_t \).

There has been some work in streaming two sample inference, e.g. Cormode et al. (2003) propose using Hamming norms to compare streams; whereas, Sebastião et al. (2014a) use fading histograms, and the KL divergence, to compare distributions. Similar to the latter, we develop two procedures for testing the hypothesis in Equation (7.7) by modifying the TASH framework. Section 7.3.1 utilizes quantiles from both streams, and is similar to the detector discussed in Section 7.2; whereas Section 7.3.2 develops an approach based on a Kolmogorov-Smirnov (KS) type test (Massey Jr, 1951).

7.3.1 Using quantiles

The first approach is a modification of the quantile change detector developed in Section 7.2. As before, let \( \phi \) be a vector of proportions, and let \( \hat{X}_{t,\phi} \) and \( \hat{Y}_{t,\phi} \) be, respectively, the adaptive quantile estimates for the streams...
in Equations (7.5) and (7.6), estimated using the techniques discussed in Section 7.1.2.

Instead of comparing a single stream’s quantiles estimated over a sliding window, we can compare the two vectors of quantile estimates \( \tilde{X}_{t,\phi} \) and \( \tilde{Y}_{t,\phi} \) directly. That is, we say the data streams are being generated by different distributions at time \( t \) (reject \( H_0 \)) whenever

\[
\left\| \tilde{X}_{t,\phi} - \tilde{Y}_{t,\phi} \right\|_2 > \frac{\beta}{2} \| \ell_{t,X} + \ell_{t,Y} \|_2,
\]

and conclude that the data streams are being generated by the same distribution (fail to reject \( H_0 \)) whenever Equation (7.8) does not hold.

As an example, consider two data streams generated by

\[
X_t \sim \mathcal{N}(0, 1), \quad t \in [1, 3000],
\]

\[
Y_t \sim \begin{cases} 
\mathcal{N}(0, 1) & t \in [1, 1000], \\
\mathcal{N}(1.5, 1) & t \in (1000, 2000], \\
\mathcal{N}(0, 1) & t \in (2000, 3000].
\end{cases}
\]

An ideal detector will reject \( H_0 \) on the interval \( t \in (1000, 2000] \), and fail to reject \( H_0 \) otherwise. Figure 7.5 displays the Euclidean distance between \( \tilde{X}_{t,\phi} \) and \( \tilde{Y}_{t,\phi} \) (solid line), the corresponding threshold with \( \beta = 2 \) (dashed line), and \( \phi = 1/100(1, 5, 10, 25, 50, 75, 90, 95, 99)^\top \) was used for the target proportions. The threshold is violated for the first time at \( t = 1022 \), i.e., \( H_0 \) was not rejected during the first regime, where the distributions were the same. Further, it took 22 data points before the method correctly rejected \( H_0 \) in favor of \( H_1 \), and over the interval \( [1040, 2000) \) the threshold remains violated (on the interval \( [1022, 1040] \) there were a few instances where the threshold was not violated). For \( t \geq 2000 \) the distributions are again equivalent, and the method should fail to reject \( H_0 \). The Euclidean distance remains below the threshold for \( t > 2161 \), implying that the method correctly failed to reject \( H_0 \), and it was realized that the data streams were again being generated by the same process.
Small modifications to the TASH framework and quantile detector resulted in a streaming method for determining whether two data streams are being generated by different processes. The only additional requirement is that a TASH be maintained for two data streams, as opposed to one, which can easily be done in parallel. In the next section we present a KS type test, which is arguably the most popular test for comparing two samples.

7.3.2 KS Type test

The (two sample) KS test is a commonly used method to test the hypothesis in Equation (7.7). Given two samples \( x_{1:t}, y_{1:t} \), the KS test rejects the null hypothesis, at level \( \alpha \), whenever

\[
\sup_x \left| \hat{F}_t(x) - \hat{G}_t(x) \right| > \sqrt{\frac{-\log(\alpha)}{t}},
\]

where \( \hat{F}_t \) and \( \hat{G}_t \) are, respectively, the empirical distribution functions for the samples \( x_{1:t} \) and \( y_{1:t} \). Historically, KS tests have been applied for small \( t \). Indeed, tables of critical values for the test statistic rarely go above \( t > 50 \), and are not suitable for streaming data.

Streaming variants of the KS test have been considered in the literature.
In Kifer et al. (2004) the KS test is implemented over sliding windows to detect changes. Lall (2015) develops a space-efficient KS test, which approximately inverts the output of a sequential outlier procedure that uses sketches. However, as presented, the method does not accommodate for changes in the data generating process. The incremental KS test proposed in dos Reis et al. (2016) is able to handle drift; although, the method hinges on inserting/removing observations from the two samples, which is non-trivial.

The KS test proposed in this section replaces the empirical distribution functions $\hat{F}_t$ and $\hat{G}_t$ with their temporally adaptive counterparts $\tilde{F}_t$ and $\tilde{G}_t$, which are computed from the TASHs. The test statistic is then the maximum absolute difference between the adaptive distribution functions.

Computing the adaptive KS test statistic requires a discretization of the combined range of the two histograms, which requires complex notation, although the concept is relatively simple. Succinctly, discretizing the combined range is needed to align the two TASHs so the adaptive distribution functions are computed at the same ‘grid points’. This results in the test statistic

$$K_t = \max_{(i,j) \in I_t} \left| \tilde{F}_t(i) - \tilde{G}_t(j) \right|,$$

where $I_t$ is an index set containing relevant bin indices computed from the discretization of the combined range. The mathematical details behind the derivation of the adaptive KS test statistic, as well as the index set $I_t$, are provided in Appendix A.6.

Larger values of $K_t$ provides evidence against the null hypothesis; however, determining an adaptive threshold $\varepsilon_t$ for the KS test is challenging. This is partly because: (1) an upper bound for $K_t$ is not readily available, making previous methods explored in this thesis not possible, and (2) to the best of our knowledge, a limiting distribution for an adaptive KS test statistic does not appear to exist. This makes it difficult to choose a sequence of thresholds that adhere to a specified significance level for the test, and we remark that this is a promising area for future work.

Figure 7.6 plots $K_t$, computed from the data streams considered in Section
7.3.1, and uses a fixed threshold \( \varepsilon = 0.25 \). The null hypothesis is rejected for the first time at \( t = 1056 \), i.e., it took 56 observations to realize that the two streams were being generated by different processes. The null hypothesis continues to be rejected over \([1056, 2193]\), where the KS statistic falls below the threshold. For \( t > 2193 \) the test correctly fails to reject the null hypothesis.

### 7.4 Dynamic ROC curves

Receiver operating characteristic (ROC) curves are commonly used to visualize, compare, and select binary classifiers based on their performance (Fawcett, 2006). The term ‘ROC curve’ was first used in the signal detection literature, where the goal was to accurately determine the existence of a signal, while simultaneously minimizing the false positive rate; e.g., see Green et al. (1966) or Spackman (1989). Since their discovery, ROC curves have been used in various applications, such as: finance, medicine, and speech recognition – refer to Krzanowski and Hand (2009) for a thorough treatment. The goal of this section is to modify the TASH framework to allow for adaptive and sequential ROC curve estimation. For an overview of ROC curves and associated terminology, such as the true and false positive rates of a classifier and the score of a feature vector, refer to Appendix A.6.
The TASH framework is now modified to allow for adaptive ROC curve estimation. To generate a ROC curve, only the true and false positive rates, computed at various values of a threshold, are required. These can be acquired from TASHs since the true and false positive rates can be computed from cumulative distribution functions.

Assume that the score of a feature vector has distribution $F_t$ when the vector comes from the ‘positive’ class, and the distribution $G_t$ when the vector is generated by the ‘negative’ class. Further, suppose that a threshold $T$ has been chosen. The true and false positive rates are then given by

\[ T_t(T) = 1 - F_t(T) \quad \text{(True positive rate),} \quad (7.9) \]
\[ F_t(T) = 1 - G_t(T) \quad \text{(False positive rate).} \quad (7.10) \]

To see why Equations (7.9)-(7.10) are valid refer to Figure 7.7, where $f_t$ and $g_t$ denote the associated probability density functions. The true positive rate is the proportion of observations from the positive class that were correctly classified, which is precisely the area under $f_t$, to the right of $T$. The dark gray region to the left of $T$ is the probability that an observation from the positive class is incorrectly predicted to the negative class. Similarly, the false positive rate is the proportion of negative class observations that were wrongly predicted to the positive class, corresponding to the area under $g_t$, to the right of $T$. This is the light gray region shaded in the figure.

In Section 7.3.2 the temporally adaptive distribution functions $\tilde{F}_t$ and $\tilde{G}_t$ were evaluated over a discretized grid $\mathcal{G}_t$ (defined in Appendix A.6), resulting in an adaptive KS test statistic. Since the true and false positive rates can be computed using distribution functions, the methodology proposed in the previous section can be applied to produce adaptive ROC curves. That is, the dynamic ROC curves can be defined by the plug-in estimates

\[ \tilde{T}_t(T) = 1 - \tilde{F}_t(T), \]
\[ \tilde{F}_t(T) = 1 - \tilde{G}_t(T), \]
Figure 7.7: An illustration that helps in understanding Equations (7.9)-(7.10). The dark shaded region represents the probability of wrongly predicting a positive class observation to the negative class; whereas, the light gray shaded area is the probability of incorrectly assigning a negative class observation to the positive class.

where $T \in G_t$ are similar to the grid points discussed previously.

Consider a toy example, where two streams are generated according to

$$s(X_t) \sim \mathcal{N}(0, 1), \quad t \in [1, 4000], \quad (7.11)$$

$$s(Y_t) \sim \begin{cases} 
\mathcal{N}(0, 1) & t \in [1, 1000], \\
\mathcal{N}(-3, 1) & t \in (1000, 2000], \\
\mathcal{N}(3, 1) & t \in (2000, 3000], \\
\mathcal{N}(0, 1) & t \in (3000, 4000],
\end{cases} \quad (7.12)$$

where $X_t$ and $Y_t$ are the positive and negative class observations, and $s(X_t)$ and $s(Y_t)$ are their scores. Figure 7.8 plots the dynamic ROC curves at various time-stamps. In the first and last segments there is no difference between the distributions; consequently, the best a classifier could do would be to randomly guess the classes, resulting in ROC curves behaving similarly to the line $y = x$ (refer to the top left and bottom right plots). Over $t \in (1000, 2000]$, the mean of the negative class distribution shifts to the left. In this case we would expect the ROC curves to migrate towards the top left of the unit square (think about sliding $g_t$ to the left in Figure 7.7 while
Figure 7.8: Plots of the dynamic ROC curves, taken at various time-stamps, for the data streams given in Equations (7.11) and (7.12).

keeping $f_t$ fixed). This behavior is seen in the top center and top right plots. Conversely, over $t \in (2000, 3000]$ the mean of the negative class distribution shifts to the right, and we would expect the ROC curves to fall below the line $y = x$, which is precisely the behavior shown in the bottom left, and bottom center plots. We remark that this is an artifact of the labeling, since switching the labels would result in ROC curves above the identity line.

7.5 Outlier detection

Lastly, TASHs are used to detect outliers, where Hawkins (1980) defined an outlier as: “an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism”. Outlier detection in streaming data is an important issue, since their presence could affect parameter accuracy, or be indicative of issues in the stochastic process being monitored. For example, malicious activity in a network could be deemed outliers when compared to the ‘normal’ behavior of the network.
7.5.1 Conformal prediction

The outlier detection method developed in Section 7.5.2 is based on conformal prediction, which was introduced in Vovk et al. (2005), and can be modified to determine outliers in streaming data. We briefly discuss conformal prediction here; however, a thorough summary is beyond the scope of this section.

Given observations $x_{1:t}$, conformal prediction uses a dissimilarity measure to assess how different each datum is, relative to the other observations. Consider an arbitrary dissimilarity measure

$$A_i = A(\{x_1, x_2, \ldots, x_i, \ldots, x_t\}, x_i),$$

where $A_i$ is a measure of how dissimilar $x_i$ is to $x_{1:t}$. Due to scaling, the value of $A_i$ does not, by itself, reveal how different $x_i$ is to the other data (Vovk et al., 2005). To alleviate this issue, conformal prediction compares $A_i$ to other dissimilarity scores, resulting in a $p$-value associated with $A_i$. This conformal $p$-value is defined by

$$\rho_i = \frac{1}{t} \left| \left\{ j = 1, \ldots, t : A_j \geq A_i \right\} \right|,$$

which represents the proportion of dissimilarity scores $A_{1:t}$ that exceed $A_i$. The $p$-value is discrete, taking values in

$$\rho_i \in \left\{ \frac{1}{t}, \frac{2}{t}, \ldots, \frac{t - 1}{t}, 1 \right\},$$

where values closer to $1/t$ provides evidence that $x_i$ is very dissimilar to the other observations, and could potentially be an outlier. Conversely, values closer to 1 indicates that $x_i$ is very similar to other observations.

7.5.2 Detecting outliers using TASHs

In this section the conformal prediction framework is modified for streaming data. This requires reformulating the $p$-value, and choosing a dissimilarity measure that can be efficiently computed online.
It is clear that the $p$-value in Equation (7.13) cannot be efficiently computed in the streaming paradigm. To remedy this issue, we compute the $p$-value over a sliding window, which contains the most recent $\omega$ observations. Suppose $x_t$ has just arrived from the stream; the conformal $p$-value for $x_t$ is taken as

$$\rho_t = \frac{\left| \{ j = t - (\omega - 1), \ldots, t : A_j \geq A_t \} \right|}{\omega},$$

for a dissimilarity measure to be discussed shortly. Then, given a significance level $\alpha \in [0, 1]$, we say $x_t$ is an outlier whenever $\rho_t < \alpha$. The $p$-value now assumes value in the discrete set

$$\rho_t \in \left\{ \frac{1}{\omega}, \frac{2}{\omega}, \ldots, \frac{\omega - 1}{\omega}, 1 \right\},$$

and due to this, given a significance level $\alpha$ the size of the sliding window must be chosen to satisfy $\omega > 1/\alpha$. If not, the $p$-values will never be less than the significance level, resulting in the method never detecting any outliers.

For the dissimilarity measure we propose using the absolute difference between $x_t$ and the adaptive median estimated from the TASHs at time $t - 1$. That is,

$$A_t = |x_t - \tilde{q}_{t-1,0.5}|,$$  \hspace{1cm} (7.14)

where $\tilde{q}_{t-1,0.5}$ was defined in Equation (7.1). Choosing a dissimilarity measure will always be a topic of discussion (Vovk et al., 2005), and others could be chosen to replace Equation (7.14). A thorough investigation of dissimilarity measures is not in the scope of this chapter, since the focal point is highlighting applications of TASHs. However, we remark that using TASHs allow for a wide-range of dissimilarity measures to be used. For example, instead of the median the interquartile range could be estimated, and used to construct a box plot. An outlier can then be flagged whenever a datum falls outside the corresponding ‘whiskers’ of the box plot.

Consider a synthetic example, where a data stream of length 3,000 is generated according to

$$x_t = \sin(g_t) + \epsilon,$$  \hspace{1cm} (7.15)

where $g_t \in \mathcal{G}$. The grid $\mathcal{G}$ equally partitions the interval $[-6\pi, 6\pi]$ into 3,000
Figure 7.9: Top: The stream generated according to Equation (7.15). Bottom: the same stream, with the data points that were flagged as outliers removed.

grid points, and \( \epsilon \sim \mathcal{N}(0, 0.25^2) \). Every 200 data points (excluding 3,000), the stream experiences an outlier, which is not generated according to Equation (7.15), but is normally distributed with mean 2.5. For sake of demonstration, we choose \( \alpha = 0.01 \) and \( \omega = 1/\alpha + 50 \). The top plot in Figure 7.9 depicts the data stream with the outliers present; whereas, the bottom figure plots the same data stream with the data points flagged as outliers removed.

We chose a stream that exhibits periodicity to test the outlier detector because it is interesting to see if the detector was able to correctly distinguish between outliers and the recurring pattern. From Figure 7.9 we see that the method was successfully able to detect the outliers.

As a final example, we synthetically generate a data stream that not only contains outliers, but also drift. The goal of this example is to see if the
Figure 7.10: The data stream generated according to Equation (7.16), which contains both outliers and drift. **Top:** the data stream with outliers present. **Bottom:** the same data stream, where the detected outliers have been removed.

detector can effectively distinguish between the two. Consider the stream

\[ X_t \sim \begin{cases} 
\mathcal{N}(0, 1), & t \in [1, 1500] \\
\text{Unif}(6, 7), & t \in (1500, 3000], 
\end{cases} \tag{7.16} \]

where Unif\((a, b)\) denotes the continuous uniform distribution on \([a, b]\). This is quite a drastic example; however, it is beneficial since we want to investigate how the outlier detector reacts to such a large change in the data generating process. Again, we take \(\alpha = 0.01\) and \(\omega = 1/\alpha + 50\), and several outliers were added to both segments. The data stream is displayed in the top plot of Figure 7.10, and the bottom plot of Figure 7.10 shows the same data stream, where detected outliers have been removed. Similar to the previous
example, we see that the outlier detector based on conformal prediction can successfully detect outliers, without the drift affecting its performance.

It is worth commenting on the role of $\alpha$. Outliers are detected when $\rho_t < \alpha$, where the discrete $p$-value $\rho_t$ represents the proportion of dissimilarity measures more dissimilar than $A_t$. Therefore $\alpha$ can be interpreted as a threshold, which dictates how dissimilar an observation must be to be deemed an outlier. If there were a prior expectation on the proportion of outliers in a data stream, the value of $\alpha$ can be chosen to coincide with this expectation. Additionally, when a stream contains no outliers the value of $\alpha$ will dictate the proportion of extreme observations that are stripped from the true distribution. In this regard, the value of $\alpha$ could be considered as a smoothing parameter, which trims extreme observations from the stream’s distribution.

7.6 Discussion

This chapter explored several applications of the TASH methodology, which was developed in Chapter 6. This included techniques for estimating quantiles, non-parametric change detection, two sample inference, adaptive ROC curve estimation and outlier detection. Adaptive quantiles were computed from TASHs by cumulatively summing the bin probabilities, which requires little increase in computation. These estimates were then used to create a change detection method for univariate streams by comparing the adaptive quantiles over a sliding window. Tests to determine whether two data streams are being generated by different processes were then developed, which were based on the quantiles and a modified KS test. The latter required computing adaptive distribution functions, which are easily computed from TASHs. These distribution functions also allowed for dynamic ROC curves to be estimated, and the chapter concluded with an outlier detector based on techniques from conformal prediction. Together, these applications culminated in an adaptive histogram ‘tool-kit’, which is applicable in a diverse range of data stream mining settings.
Conclusions

Data stream inference is fundamentally different than conventional learning tasks, since streaming applications require methodology to learn sequentially with fixed computational and storage requirements. Algorithms must also be capable of reacting to drift, as data streams will experience several changes over time. These difficulties provide rich opportunities for research, and this thesis has presented novel contributions for detecting multiple change-points in categorical data streams, and in estimating histograms for univariate, continuous-valued data streams.

The first contributions of this thesis focused on developing change detectors for categorical data streams, which is a non-trivial issue that has not received a significant amount of attention in the literature. In Chapter 4 the categorical observations were assumed independent, and a family of MCDMs were developed. These methods utilized an adaptive MLE for the probability vector of a multinomial distribution, where temporal adaptivity was introduced using forgetting factors. The MCDMs monitor the differences between the adaptive and static MLEs, which is quantified using a dissimilarity measure. Changepoints were then detected using a novel adaptive thresholding technique, which can be tuned given a value for $\text{ARL}_0$. The MCDMs require
three values to be specified: the length of the grace period, the desired value of ARL₀, 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, which is an attractive feature of the detectors. The KL divergence and Cohen’s h score were then used as dissimilarity measures, resulting in MCDMs that performed favorably when compared with other methods. The MCDMs were also implemented on nearly 40 million observations collected from a computer network, reinforcing that the method is suitable for applications that continuously generate a large amount of data.

Chapter 5 assumed the categorical observations satisfied a first-order Markov property, requiring an adaptive estimate for a transition matrix to be developed. Monitoring a matrix on the stream presented new challenges, such as the two different ‘clocks’ that drive the estimation, and the interpretation of the grace period. ADEPT-M was then introduced, which monitors for changepoints in each transition probability using a moment matching technique and, similar to the MCDMs, requires a grace period and step-size to be chosen. A significance level that is related to the false positive rate is also required. This allows for approximate control limits to be constructed for each transition probability, which are used to detect changepoints. ADEPT-M was then shown to be effective in detecting multiple changepoints in a large simulation study, and was capable of detecting a cyber-attack in a real-world data stream consisting of over 9 million HTTP web requests.

A sequential histogram estimate was developed in Chapter 6 to monitor univariate, continuous-valued data streams. This required innovative techniques for updating the boundary of the histogram, as well as merging and splitting bins to maintain an accurate density estimate. Discretizing a continuous-valued data stream ‘bridged the gap’ between the chapters in this thesis, and allowed for the adaptive MLEs for categorical data to be applicable to univariate data streams. Non-drifting streams were analyzed, and the sequential histogram was shown to provide similar accuracy when compared with a batch histogram. Temporal adaptivity was then introduced using forgetting factors, which appealed to the techniques developed throughout this thesis. A new way for tuning the forgetting factors was explored, since
the histogram estimate itself could be used as an approximate likelihood for the data. This is critical when providing non-parametric inference for streaming data, since the distribution of the data is unknown. This resulted in TASHs, which were able to provide accurate density estimates in drifting data streams, and simulation studies highlighted their ability to react to changes quicker than fading/sliding histograms.

Lastly, the TASH framework was extended to allow for streaming quantile estimation, non-parametric change detection, streaming two sample inference, dynamic ROC curves, and outlier detection. This inference is suitable for the streaming paradigm, as the estimates inherit adaptivity from the TASHs and come with little increase in computation. This culminated in a streaming histogram ‘tool-kit’, where adaptive histograms can be maintained in the streaming paradigm, and used to compute several statistics of interest. This makes the TASHs an appealing method in a variety of data stream mining problems.

**Future work**

To conclude, we provide several directions for future work. Forgetting factors were the driving force behind adaptive estimation in this thesis, and tuning them required a step-size parameter $\eta$ to be specified. In Section 3.4.4 it was argued that a constant step-size was sufficient, and it was shown to not drastically affect the adaptive parameter estimates (see Figure 3.2). However, smaller values of $\eta$ do affect how quickly the forgetting factors recover after drift. A possible area of research would be investigating ways of tuning $\eta$ to expedite this recovery. Specifically, higher order optimization methods such as Newton’s method could be investigated, or the accelerated gradient based methods discussed in Ruder (2016).

Regarding change detection, it is typical to construct detectors that control the false positive rate. However, streaming applications require few false alarms to be raised since investigating them may be costly and time consuming. This would require detectors to have an ARL$_0$ on the order of millions, which is a difficult task currently not being investigated. This raises the
question as to whether or not ARL$_0$ is the correct metric to control when developing change detectors. Therefore, a possible direction for research is in developing change detectors to control for other metrics, which are more suitable for streaming data. Indeed, there is no ‘golden standard’ in assessing streaming methodology (Gama et al., 2009), which is itself a promising, closely related, direction for research.

The allowance parameters for the MCDMs were chosen to approximate a desired value of ARL$_0$, and it was shown that the parameters resulted in detectors that closely attained an ARL$_0$ of 2,000 for synthetic data, and 20,000 for real-data. However, the sigmoid approximation technique for choosing the allowance parameters is not directly applicable for an ARL$_0$ in the millions. This is because the sigmoid is ‘pinned down’ at its endpoints, making the allowance parameters very sensitive to small perturbations in its value. Thus, an area of research is in developing new methods for choosing the allowances, so that larger values of ARL$_0$ could be obtained by the MCDMs. The number of categories K could also be incorporated into the estimation of the allowances, so that a sigmoid does not need to be fit for each K.

The moment matching technique used in ADEPT-M was required since a known distribution does not appear to exist for a weighted sum of Bernoulli random variables. An avenue for future work would be in developing approximation techniques for the distribution of the adaptive transition probabilities. Another interesting topic would be modifying ADEPT-M to allow for a state space whose cardinality changes over time. This would require methodology that could handle new states emerging in the Markov chain, as well as determining when to expire older states that are no longer visited.

For the TASHs developed in Chapter 6, research could be done into developing methods for setting the parameters $w$ and $W$, which dictate the range of allowable bin widths. An interesting application to explore would be using the TASHs in conjunction with the naive Bayes classifier. Since naive Bayes supposes that covariates are independent from one another, a TASH could be constructed for each covariate and used to compute the probabilities required for classification.
Finally, it is worth briefly mentioning the limitations of the methodology developed in this thesis. A large portion of this work focused on detecting changes in a single data stream. If many data streams were available, methods could be explored for aggregating data from multiple sources, so that several streams could simultaneously be monitored for changepoints. The MCDMs and ADEPT-M assumed the observations from the stream were independent or satisfied a first-order Markov property. If a more complicated dependence structure were present, these methods may not be entirely appropriate, and a direction for future work would be developing changepoint detectors that take into account more involved dependencies between observations. The TASHs are applicable for univariate data streams, or for multivariate streams under the assumption of independence between the elements of the multivariate vector (since a TASH can be run independently for each component). Future work could explore a multivariate extension of the TASHs, where the assumption of independence among the features may be inappropriate.
References


Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Computing Surveys 46(4), 44:1–44:37 (2014)


Pearson, K.: Contributions to the mathematical theory of evolution II. Skew variation in homogeneous material. Philosophical Transactions of the Royal Society of London (A) (186), 343–414 (1895)


Roberts, S.: Control chart tests based on geometric moving averages. Technometrics 1(3), 239–250 (1959)


Shewhart, W.A.: Economic control of quality of manufactured product. ASQ Quality Press (1931)


Yakir, B.: Optimal detection of a change in distribution when the observations form a Markov chain with a finite state space. Lecture Notes-Monograph Series 23, 346–358 (1994)


Derivations and Preliminaries

A.1 Chapter 2

Proof of Lemma 1 (multinomial MLE)

Given the data \( x_{1:t} \), which determine the observed category counts, the log-likelihood is given by

\[
\mathcal{L}(p | x_{1:t}) = \log(t!) - \sum_{i=1}^{K} \log(c^{(i)}!) + \sum_{i=1}^{K} c^{(i)} \log(p^{(i)}).
\]

Since the category probabilities are constrained to sum to one, optimization must be done via the method of Lagrange multipliers. That is, we optimize the Lagrangian, which takes into account the constraint, given by

\[
\Lambda(p, \gamma | x_{1:t}) = \mathcal{L}(p | x_{1:t}) + \gamma \left(1 - \sum_{i=1}^{K} p^{(i)}\right), \tag{A.1}
\]

where \( \gamma \) is the Lagrange multiplier. Differentiating (A.1) with respect to \( p^{(i)} \) yields

\[
\frac{\partial \Lambda(p, \gamma | x_{1:t})}{\partial p^{(i)}} = \frac{c^{(i)}}{p^{(i)}} - \gamma,
\]

and setting equal to zero results in \( \hat{p}^{(i)} = \hat{c}^{(i)}/\gamma \), where the hats have been included to stress that \( \hat{p}^{(i)} \) is an estimate, as opposed to an estimator. Summing over \( i \in \{1, \ldots, K\} \) reveals that \( \gamma = t \), completing the proof. \( \square \)
Proof of Lemma 2 (Beta mean/variance)

Let $X \sim \text{Beta}(a,b)$. Basic calculations reveal that

$$E(X) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} \int_0^1 x^{(a+1)-1}(1-x)^{b-1}dx$$

$$= \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} \frac{\Gamma(a + 1)\Gamma(b)}{\Gamma(a + 1 + b)},$$

as the integrand is the kernel of a Beta($a + 1, b$) distribution. Recalling that the gamma function satisfies the equation $\Gamma(x + 1) = x\Gamma(x)$, for any $x > 0$, it immediately follows that $E(X) = a/(a + b)$. A similar calculation shows that $E(X^2) = \frac{a(a + 1)}{(a + b + 1)(a + b)}$, resulting in the variance displayed in the lemma.

A.2 Chapter 3

Recursive updates for the adaptive mean

Both equations can be derived from basic algebra. For the effective sample size $n_t$ observe that

$$n_t = \sum_{k=1}^{t} w_k = \sum_{k=1}^{t} \left( \prod_{\ell=k}^{t-1}\lambda_\ell \right)$$

(Definition of $w_k$)

$$= \sum_{k=1}^{t-1} \left( \prod_{\ell=k}^{t-1}\lambda_\ell \right) + 1$$

(Empty product is one)

$$= \lambda_{t-1} \sum_{k=1}^{t-1} \left( \prod_{\ell=k}^{t-2}\lambda_\ell \right) + 1$$

(Factor out $\lambda_{t-1}$)

$$= \lambda_{t-1} n_{t-1} + 1$$

(Definition of $n_{t-1}$).

For $\hat{\mu}_t$ observe that

$$n_t \hat{\mu}_t = \sum_{k=1}^{t} \left( \prod_{\ell=k}^{t-1}\lambda_\ell \right) x_k = \lambda_{t-1} \sum_{k=1}^{t-1} \left( \prod_{\ell=k}^{t-2}\lambda_\ell \right) x_k + x_t.$$
where we have taken the $t$\textsuperscript{th} term out of the summation and have factored out $\lambda_{t-1}$ (since it is common to every term). From our definition of $\tilde{\mu}_{t-1}$ we see that

$$n_{t-1} \tilde{\mu}_{t-1} = \sum_{k=1}^{t-1} \left( \prod_{\ell=k}^{t-2} \lambda_{\ell} \right) x_k.$$  

Plugging this into what was just derived yields

$$n_t \tilde{\mu}_t = (\lambda_{t-1} n_{t-1}) \tilde{\mu}_{t-1} + x_t = (n_t - 1) \tilde{\mu}_{t-1} + x_t,$$

where we have used the recursive update for $n_t$. Dividing through by $n_t$ completes the derivation. \hfill \Box

A.3 Chapter 4

Adaptive MLE

We must optimize

$$L_{\text{FF}}(p \mid x_{1:t}) = \sum_{k=1}^{t} w_k L(p \mid x_k),$$

where $L(p \mid x_k)$ is the multinomial log-likelihood function evaluated at the single datum $x_k$. Consequently, the vector of category counts is the standard basis vector in $\mathbb{R}^K$ with a one in the position corresponding to the value of $x_k$, i.e., $c^{(i)} = \mathbb{1}(x_k = i)$. The Lagrangian can then be expressed as (omitting constants independent of the category probabilities)

$$\Lambda_{\text{FF}}(p, \gamma \mid x_{1:t}) = \sum_{k=1}^{t} w_k \left[ \sum_{i \in S} \mathbb{1}(x_k = i) \log(p^{(i)}) \right] - \gamma \left( 1 - \sum_{i \in S} p^{(i)} \right),$$

and optimization is identical to the derivation of the multinomial maximum likelihood estimates given in Appendix A.1. \hfill \Box
KL UPPER BOUND

Since the category probabilities are positive, we may appeal to the bound \( \log x < x \), resulting in

\[
\sum_{i \in S} p_i^{(t)} \log \left( \frac{p_i^{(t)}}{\tilde{p}_t^{(i)}} \right) < \sum_{i \in S} \left( \frac{p_i^{(t)}}{\tilde{p}_t^{(i)}} \right)^2
\]

\[
= \sum_{i \in S} \left( \frac{p_i^{(t)}}{\sqrt{\tilde{p}_t^{(i)}}} \right)^2
\]

\[
\leq K \max_{i \in \{1, \ldots, K\}} \left( \frac{p_i^{(t)}}{\sqrt{\tilde{p}_t^{(i)}}} \right)^2
\]

\[
= K \left\| \tilde{p}_t \right\|_\infty^2.
\]

In the last equality we interchanged the maximum and square, which is justified as the square is monotonically increasing. This completes the proof. \(\square\)

ARCSINE UPPER BOUND

Here we fill in the details for the proof provided in Sun and Chen (2016), which asserts that for any \( x \in (0, 1) \)

\[
\frac{8x}{3\sqrt{1 - x^2} + \sqrt{25 + ax^2}} < \arcsin x < \frac{8x}{3\sqrt{1 - x^2} + \sqrt{25 + bx^2}},
\]

with the best possible constants \( a = 5/3 \) and \( b = 1/\pi^2(256 - 25\pi^2) \). Recall that only the upper bound was needed in Chapter 4; however, for completeness we prove both inequalities.

As a first step, we rewrite the inequalities in Equation (A.2) to isolate \( a \) and \( b \) on both sides. Doing so results in

\[
b < \left( \frac{8x}{\arcsin x} - \frac{3\sqrt{1 - x^2}}{x^2} \right)^2 - 25 < a.
\]
Proving the lemma is then equivalent to showing $b < F(x) < a$, where $F(x)$ denotes the function appearing in between the inequalities in Equation (A.3).

Making the change of variable $x = \sin t$ allows us to rewrite $F(\cdot)$ as

$$G(t) = F(\sin t) = \frac{\left(\frac{8 \sin t}{t} - 3 \cos t\right)^2 - 25}{\sin^2(t)}, \quad (A.4)$$

where $t \in (0, \pi/2)$, since $x \in (0, 1)$. The goal is to show that $F(x)$ is strictly decreasing on $x \in (0, 1)$ by showing $G(t)$ is strictly decreasing on $t \in (0, \pi/2)$.

Differentiating Equation (A.4) with respect to $t$ yields

$$\frac{t^3 \sin^3 t}{16} G'(t) = 2t^3 \cos t + 3t \cos t \sin t - 8 \sin^3 t$$

$$= -\left(8 \sin^3 t - 3t \sin t - (2t^3 + 3t) \cos t + 3t \cos^3 t\right).$$

Using the identities

$$\sin^3 t = \frac{1}{4}(3 \sin t - \sin 3t), \quad \cos^3 t = \frac{1}{4}(3 \cos t + \cos 3t),$$

multiplying by minus one, and cleaning up the equation results in

$$-\frac{t^3 \sin^3 t}{16} G'(t) = (6 - 3t^2) \sin t - 2 \sin 3t + \frac{3}{4} t \cos 3t - \left(\frac{3}{4} t + 2t^3\right) \cos t.$$ 

Taylor expanding (about $t = 0$) the right hand side of the above yields

$$-\frac{t^3 \sin^3 t}{16} G'(t) = \frac{16}{945} t^9 - \frac{16}{4725} t^{11} + \sum_{n=6}^{\infty} (-1)^n u_n(t), \quad (A.5)$$

where

$$u_n(t) = \frac{21 + 2n + 48n^2 + 64n^3 + (6n - 21)9^n}{4(2n + 1)!} t^{2n+1}.$$ 

Then for $t \in (0, \pi/2)$, and $n \geq 6$ the ratio of the coefficients can be bounded
by
\[
\frac{u_{n+1}(t)}{u_n(t)} = \frac{64n^3 + 240n^2 + 290n + 135 + (54n - 135)9^n}{2(n + 1)(2n + 3)(21 + 2n + 48n^2 + 64n^3 + 9^n(6n - 21))} t^2 < \frac{135 + 290n + 240n^2 + 64n^3 + 9^n(54n - 135)}{2(2n + 3)(21 + 2n + 48n^2 + 64n^3 + 9^n(6n - 21))} \left(\frac{\pi^2}{4}\right) \frac{9^n}{n + 1} < \frac{135 + 290n + 240n^2 + 64n^3 + 9^n(54n - 135)}{2(2n + 3)(21 + 2n + 48n^2 + 64n^3 + 9^n(6n - 21))} t^2
\]
where the first inequality used that \( t \in (0, \pi/2) \) and the second inequality used that \( (\pi^2/4)/(n + 1) < 1 \) for \( n \geq 6 \).

Let \( N_n \) and \( D_n \) denote the numerator and denominator in Equation (A.6) respectively. Then
\[
D_n - N_n = (24n^2 - 102n + 9)9^n + 256n^4 + 512n^3 + 56n^2 - 194n - 9,
\]
which is positive for all \( n \geq 1 \), and since \( n \geq 6 \), it shows that the fraction in Equation (A.6) is always less than one. Therefore,
\[
\frac{u_{n+1}(t)}{u_n(t)} < 1 \iff u_{n+1}(t) < u_n(t),
\]
implying that \( \{u_n(t)\}_{n=6}^{\infty} \) is a strictly decreasing sequence in \( n \).

Returning to Equation (A.5) we have that
\[
-\frac{t^3 \sin^3 t}{16} G'(t) = \frac{16}{945} t^9 - \frac{16}{4725} t^{11} + \sum_{n=6}^{\infty} (-1)^n u_n(t) \geq t^9 \left(\frac{16}{945} - \frac{16}{4725} t^2\right),
\]
since the infinite series is positive (as the terms form a strictly decreasing sequence). Manipulation of the inequality reveals that \( G'(t) < 0 \) for all \( t \in (0, \pi/2) \) or, equivalently, \( F(x) \) is decreasing for all \( x \in (0,1) \). Hence
\[
\lim_{x \to 1} F(x) < F(x) < \lim_{x \to 0} F(x),
\]
and computing the limits results in the constants \( a \) and \( b \) stated in the lemma and completes the proof. \( \Box \)
A.4 Chapter 5

Proof of Lemma 3

Taking the expectation of the adaptive transition probability yields

\[
\mathbb{E} \left( \hat{p}_t^{(ji)} \right) = \frac{1}{n_t^{(i)}} \sum_{k=1}^{B_t^{(i)}} w_k^{(i)} \mathbb{E} [ \mathbb{I} (B_t^{(i)}[k] = j) ] = \frac{1}{n_t^{(i)}} \sum_{k=1}^{B_t^{(i)}} w_k^{(i)} \mathbb{P} (B_t^{(i)}[k] = j)
\]

\[
= \frac{1}{n_t^{(i)}} \sum_{k=1}^{B_t^{(i)}} w_k^{(i)} p_t^{(ji)} = p_t^{(ji)},
\]

where we have used the fact that the expectation of an indicator function is just the probability of the particular event occurring.

To derive the variance we will need to know the variance associated with an indicator function; this is given by

\[
\text{Var} \left( \mathbb{I} (B_t^{(i)}[k] = j) \right) = \mathbb{P} (B_t^{(i)}[k] = j) \left( 1 - \mathbb{P} (B_t^{(i)}[k] = j) \right)
\]

\[
= p_t^{(ji)} \left( 1 - p_t^{(ji)} \right).
\]

Since we have focused on only the transitions out of a particular state \(i\), there is no covariance term. That is, transitioning to other states, given that you are in state \(i\), are independent from another. Then

\[
\text{Var} \left( \hat{p}_t^{(ji)} \right) = \frac{1}{\left[ n_t^{(i)} \right]^2} \text{Var} \left\{ \sum_{k=1}^{B_t^{(i)}} w_k^{(i)} \mathbb{I} (B_t^{(i)}[k] = j) \right\}
\]

\[
= \frac{1}{\left[ n_t^{(i)} \right]^2} \left\{ \sum_{k=1}^{B_t^{(i)}} w_k^{(i)} \right\}^2 \text{Var} \left[ \mathbb{I} (B_t^{(i)}[k] = j) \right]
\]

\[
= \frac{1}{\left[ n_t^{(i)} \right]^2} \left\{ \sum_{k=1}^{B_t^{(i)}} w_k^{(i)} \right\}^2 \mathbb{P} (B_t^{(i)}[k] = j) \left( 1 - \mathbb{P} (B_t^{(i)}[k] = j) \right)
\]

\[
= u_t^{(i)} p_t^{(ji)} \left( 1 - p_t^{(ji)} \right),
\]

which completes the proof. \(\square\)
Comments on the assumptions made in proving Lemma 3 are warranted. First, it has been assumed that the observations are identically distributed, that is, the transition probabilities do not drift. Second, the weights and effective sample size have been treated as constants, allowing them to be ‘pulled through’ the expectation and variance calculations. This assumes that the forgetting factors are fixed, i.e., they do not depend on observations from the stream. These assumptions were made to allow for expressions for the mean and variance to be obtained, and without them the derivations become intractable. Hence, when adaptive forgetting factors are used the mean and variance provided in the lemma are understood to be approximations.

**Recursive update for** \( m_t^{(i)} \)

The update for \( m_t^{(i)} \) can be computed algebraically as

\[
m_t^{(i)} = \sum_{k=1}^{B_t^{(i)}} w_k^{(i)} \]

\[
= \sum_{k=1}^{B_t^{(i)}} \left( B_t^{(i)} - 1 \right) \prod_{\ell=k}^{B_t^{(i)}-1} \lambda_{[\ell]}^{(i)} \quad \text{(Definition of } w_k^{(i)})
\]

\[
= \sum_{k=1}^{B_t^{(i)}-1} \left( \prod_{\ell=k}^{B_t^{(i)}-1} \lambda_{[\ell]}^{(i)} \right) ^2
\]

\[
= \prod_{\ell=B_t^{(i)}}^{B_t^{(i)}-1} \left( \lambda_{[\ell]}^{(i)} \right) ^2 + \sum_{k=1}^{B_t^{(i)}-1} \left[ \prod_{\ell=k}^{B_t^{(i)}-1} \left( \lambda_{[\ell]}^{(i)} \right) ^2 \right] \quad \text{(Empty product)}
\]

\[
= 1 + \left( \lambda_{[\ell=B_t^{(i)}-1]}^{(i)} \right) ^2 \cdot \sum_{k=1}^{B_t^{(i)}-1} \left[ \prod_{\ell=k}^{B_t^{(i)}-1} \left( \lambda_{[\ell]}^{(i)} \right) ^2 \right]
\]

\[
= 1 + \left[ \lambda_{[\ell=B_t^{(i)}-1]}^{(i)} \right] ^2 \cdot m_t^{(i)}
\]

\[
= 1 + \lambda_{t-1}^{(i)} \cdot m_t^{(i)}.
\]

Notice that in the last equality we have replaced the multiset indexing notation \([\cdot]\) with the time-stamp \( t \), which is allowed since anytime a transition
from row $i$ does not occur, the subscript is updated with the new time-stamp. That is, due to the updating of the subscript, $\lambda^{(i)}_{t-1}$ and $\lambda^{(i)}_{[B_t^{(i)}]-1}$ represent the same forgetting factor \( \text{(similarly for } m^{(i)}_{[B_t^{(i)}]-1} \text{).} \)

\[ \sum \]

A.5 Chapter 6

Maximizing the change in entropy

First we derive an approximation for a continuous random variable’s entropy using histograms. This derivation is similar to what appears in Wallis (2006). Suppose $X$ is a continuous random variable with density $f_X$, and that $\mathcal{H}_t$ is a current histogram estimate. Then the entropy of $X$ is defined by

\[
E(X) = - \int_{\mathcal{X}} f_X(x) \log(f_X(x)) \, dx,
\]

which can be approximated via the histogram as

\[
E(X) \approx - \sum_{i=1}^{K_t} \int_{a_t^{(i)}}^{b_t^{(i)}} f_X(x) \log(f_X(x)) \, dx.
\]

The density in the $i$th bin can be estimated by the constant $p_t^{(i)}/(b_t^{(i)} - a_t^{(i)})$ since

\[
p_t^{(i)} = \int_{a_t^{(i)}}^{b_t^{(i)}} f_X(x) \, dx \approx \left( b_t^{(i)} - a_t^{(i)} \right) f_X(x_k)
\]

for some test-point $x_k$. Therefore

\[
E(X) \approx - \sum_{i=1}^{K_t} \int_{a_t^{(i)}}^{b_t^{(i)}} f_X(x) \log(f_X(x)) \, dx
\]

\[
= - \sum_{i=1}^{K_t} p_t^{(i)} \log \left( \frac{p_t^{(i)}}{b_t^{(i)} - a_t^{(i)}} \right), \quad (A.7)
\]

184
where the unknown $p_t^{(i)}$ can be replaced with the static, or adaptive bin probabilities developed in Chapter 6. Since the maximization of the entropy used in splitting bins was discussed before the adaptive probabilities were developed, we use the static estimates denoted by $\hat{p}_t^{(i)}$ moving forward.

Suppose that we are interested in splitting the $i$th bin, based on the ideas presented in Section 6.3.3. When comparing the difference in entropy, the only term that will remain in Equation (A.7) is the $i$th term, since all other bins remain unchanged. The change in entropy is then given by

$$\Delta E(\psi_{t+1}) = \hat{p}_{t+1}^{(i)} \log \left( \frac{\hat{p}_{t+1}^{(i)}}{b_t^{(i)} - a_t^{(i)}} \right) - \psi_{t+1} \hat{p}_{t+1}^{(i)} \log \left( \frac{1 - \psi_{t+1}}{\Xi_{t+1} - a_t^{(i)}} \right)$$

$$- \psi_{t+1} \hat{p}_{t+1}^{(i)} \log \left( \frac{\psi_{t+1} \hat{p}_{t+1}^{(i)}}{b_t^{(i)} - \Xi_{t+1}} \right)$$

$$= C - \hat{p}_{t+1}^{(i)} \log (1 - \psi_{t+1}) + \psi_{t+1} \hat{p}_{t+1}^{(i)} \log (1 - \psi_{t+1})$$

$$- \psi_{t+1} \hat{p}_{t+1}^{(i)} \log \left( \Xi_{t+1} - a_t^{(i)} \right) - \psi_{t+1} \hat{p}_{t+1}^{(i)} \log (\psi_{t+1})$$

$$+ \psi_{t+1} \hat{p}_{t+1}^{(i)} \log \left( b_t^{(i)} - \Xi_{t+1} \right),$$

where $C$ is a constant independent of $\psi_{t+1}$. Differentiating the above with respect to $\psi_{t+1}$ results precisely in Equation (6.7), which can be readily solved for the maximizer $\psi_{t+1}^*$. \hfill \Box

### A.6 Chapter 7

#### Proof of Equation (7.4) (quantile detector upper bound)

Consider the vectors of quantile estimates $\tilde{q}_{t,\phi}$ and $\tilde{q}_{t-\omega,\phi}$. Restricting to the $i$th components of these vectors, it follows that

$$\left| \tilde{q}_t^{(i)} - \tilde{q}_{t-\omega}^{(i)} \right| = \left| \tilde{q}_t^{(i)} - q_t^{(i)} + q_t^{(i)} - \tilde{q}_{t-\omega}^{(i)} \right|$$

$$\leq \left| \tilde{q}_t^{(i)} - q_t^{(i)} \right| + \left| q_t^{(i)} - \tilde{q}_{t-\omega}^{(i)} \right|$$

$$= \left| \tilde{q}_t^{(i)} - q_t^{(i)} \right| + \left| q_{t-\omega}^{(i)} - \tilde{q}_{t-\omega}^{(i)} \right|,$$

where we have used the triangle inequality, and the assumption that the true quantile has not changed in the last $\omega$ time-steps, that is, $\tilde{q}_t^{(i)} = \tilde{q}_{t-\omega}^{(i)}$. Now, under the assumption that each bin traps the actual value of the quantile, the
difference between the estimates and the true quantile is bounded above by half the width of the bin whose midpoint was used for the quantile estimate. Hence,
\[
\left| \tilde{q}_t^{(i)} - \tilde{q}_{t-\omega}^{(i)} \right| \leq \frac{1}{2} \left( \ell_t^{(i)} + \ell_{t-\omega}^{(i)} \right),
\]
and squaring each side, as well as summing over all the quantile estimates, yields
\[
\sum_{i=1}^{\phi} \left( \tilde{q}_t^{(i)} - \tilde{q}_{t-\omega}^{(i)} \right)^2 \leq \frac{1}{4} \sum_{i=1}^{\phi} \left( \ell_t^{(i)} + \ell_{t-\omega}^{(i)} \right)^2. \tag{A.8}
\]
Taking the square root of both sides reveals that
\[
A(\tilde{q}_{t,\phi}, \tilde{q}_{t-\omega,\phi}) = \| \tilde{q}_{t,\phi} - \tilde{q}_{t-\omega,\phi} \|_2 \\
\leq \frac{1}{2} \left\| \ell_t + \ell_{t-\omega} \right\|_2,
\]
completing the proof. \(\square\)

**Mathematical details for adaptive KS test statistic**

For each time-stamp \(t\) the histogram summaries estimated from the data streams in Equations (7.5)-(7.6) are expressed as
\[
\mathcal{H}_{t,X} = \left\{ \left( a_t^{(i)}, b_t^{(i)}, p_t^{(i)} \right) \right\}_{i=1}^{K_{t,X}}, \quad \mathcal{H}_{t,Y} = \left\{ \left( c_t^{(i)}, d_t^{(i)}, \tilde{v}_t^{(i)} \right) \right\}_{i=1}^{K_{t,Y}}, \tag{A.9}
\]
where the subscripts \(X\) and \(Y\) have been added to distinguish what stream the TASH has been estimated from. The temporally adaptive distribution functions are then defined as
\[
\tilde{F}_t(k) = \sum_{i=1}^{k} p_t^{(i)}, \quad k \in \{1, \ldots, K_{t,X}\}, \tag{A.10}
\]
\[
\tilde{G}_t(k) = \sum_{i=1}^{k} \tilde{v}_t^{(i)}, \quad k \in \{1, \ldots, K_{t,Y}\}. \tag{A.11}
\]
Maximizing the absolute difference between the adaptive distribution functions requires a discretization of the combined range of the two histograms. That is, we consider a grid \(\mathcal{G}_t\), which contains grid points that partitions the interval
\[
\left[ \min\left\{ a_t^{(i)}, c_t^{(i)} \right\}, \max\left\{ b_t^{(K_{t,X})}, d_t^{(K_{t,Y})} \right\} \right],
\]
186
into $\ell$ equiwidth sub-intervals. Since any $T \in \mathcal{G}_t$ is unlikely to a positive integer, we extend $\tilde{F}_t$ and $\tilde{G}_t$ to allow for non-integer arguments by defining

$$
\tilde{F}_t(T) = \sum_{i=1}^{\pi(T|\mathcal{H}_{t,X})} \tilde{p}_t^{(i)},
$$

(A.12)

$$
\tilde{G}_t(T) = \sum_{i=1}^{\pi(T|\mathcal{H}_{t,Y})} \tilde{v}_t^{(i)},
$$

(A.13)

where the function $\pi(T|\cdot)$, given a histogram $\mathcal{H}$ with a collection of right endpoints $\{b^{(i)}\}$, is defined as

$$
\pi(T|\mathcal{H}) = \arg\min_{\{i: T \geq b^{(i)}\}} (T - b^{(i)}).
$$

(A.14)

Thus, given a $T \in \mathcal{G}_t$, the function $\pi(T|\cdot)$ returns the index of the bin residing immediately to the left of $T$ in the summaries $\mathcal{H}_{t,X}$ and $\mathcal{H}_{t,Y}$. The adaptive distribution functions evaluated at $T$ are then the cumulative sums of the bin probabilities up until these bin indices. We remark that in the case when $T \in \mathbb{Z}^+$, Equations (A.12)-(A.13) are equivalent to (A.10)-(A.11). Evaluating (A.12)-(A.13) for every $T \in \mathcal{G}_t$ results in the index set of tuples

$$
I_t = \{\pi(T|\mathcal{H}_{t,X}), \pi(T|\mathcal{H}_{t,Y})\}_{\forall T \in \mathcal{G}_t},
$$

(A.15)

resulting in an adaptive KS test statistic given by

$$
\kappa_t = \max_{(i,j) \in I_t} \left| \tilde{F}_t(i) - \tilde{G}_t(j) \right|,
$$

(A.16)

which is the statistic appearing in Section 7.3.2.

**Classical ROC curves**

We present ROC curves in terms of a non-streaming classification problem, as it is easier to understand. A binary classifier is a mapping $c : \mathbb{R}^d \rightarrow \{+, -\}$, which takes a feature vector $\mathbf{x}$ as input, and predicts a corresponding class for the feature vector. Following convention, we use a plus sign to denote the ‘positive class’, and a minus sign to denote the ‘negative class’. The goal of any classification task is to learn an effective mapping, which assigns many feature vectors to the correct class, while making few misclassifications.

In practice, classifiers will inevitably make misclassifications, that is, certain
feature vectors will be incorrectly assigned to the wrong class. For binary classifiers there are only four possible outcomes for each classification: a feature vector from the positive class can be correctly/incorrectly classified, or a feature vector from the negative class can be correctly/incorrectly classified. These outcomes are typically summarized by a confusion matrix, which is depicted in Figure A.1.

\[
\begin{array}{c|c|c}
\text{True class} & + & - \\
\hline
+ & \text{True positive (TP)} & \text{False positive (FP)} \\
- & \text{False negative (FN)} & \text{True negative (TN)} \\
\end{array}
\]

Figure A.1: A confusion matrix highlighting the possible outcomes of a binary classifier. The notation in parentheses represents the total number that a classifier makes. For example, \(TP\) represents the total number of true positives a binary classifier makes over a given data set.

Various performance measures can be computed directly from the confusion matrix. Perhaps the most obvious measure is the accuracy of a classifier:

\[
\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \quad (A.17)
\]

which tells us the proportion of feature vectors that were correctly classified. Although intuitive, using classification accuracy as a performance measure has been scrutinized for various reasons (e.g., see Provost et al. (1998)). This has led to an increase in the use of ROC curves (Fawcett, 2006), which utilizes two other measures of performance: the true positive rate and the false positive rate. From a confusion matrix these are, respectively, given by

\[
\mathcal{T} = \frac{TP}{TP + FN},
\]

\[
\mathcal{F} = \frac{FP}{FP + TN}.
\]

The true positive rate is the proportion of feature vectors belonging to the
positive class that were correctly classified, and the false positive rate is the proportion of feature vectors belonging to the negative class which were incorrectly classified.

To formally discuss ROC curves, the score of a feature vector \( \mathbf{x} \), and a threshold \( T \), which is used to make the classification, needs to be addressed. The score of a feature vector, denoted \( s(\mathbf{x}) \), is a numeric value that represents the degree to which \( \mathbf{x} \) belongs to a particular class (Fawcett, 2006) – usually the positive class. Loosely speaking, the larger the score, the more likely \( \mathbf{x} \) belongs to the positive class. How large the score needs to be for \( \mathbf{x} \) to be classified to the positive class depends on the threshold \( T \), that is,

\[
c(\mathbf{x}) = \begin{cases} +, & \text{if } s(\mathbf{x}) > T, \\ - , & \text{otherwise}. \end{cases}
\] (A.18)

Once a scoring function and threshold is specified, a classification can be made for each feature vector. The true and false positive rates can then be computed, which results in a single point that can be plotted in ROC space – the unit square, where the false positive rate is plotted on the \( x \)-axis, and the true positive rate on the \( y \)-axis. A ROC curve is then created by connecting several points of the form \( (\mathcal{F}(T), \mathcal{T}(T)) \), for various values of the threshold \( T \), where we have explicitly expressed the dependence of the true and false positive rates on the threshold value. Hence, the ROC curve may be thought of as a curve parameterized by the threshold values.
Examples of ROC curves are provided in Figure A.2. The identity line $y = x$ is provided for reference, as it represents a ROC curve computed from a classifier that randomly makes predictions. To see this, suppose that a classifier randomly predicts the positive class half the time. The classifier will then correctly predict roughly half the feature vectors belonging to the positive and negative class – this is represented as the point $(0.5, 0.5)$ in ROC space (Fawcett, 2006). Similar arguments can be made for any point lying on the line $y = x$. Any curve appearing below $y = x$ performs worse than a classifier that randomly predicts classes. In this case the classifier can be ‘negated’, that is, all the predictions are swapped with the other class. In general, ROC curves which ‘hug’ the top left corner of the unit square are indicative of a classifier performing favorably – indeed, the point $(0, 1)$ represents a classifier with perfect performance. Hence, in Figure A.2 the classifier which produced the solid curve can be thought of as superior to the classifiers that produced the dashed and dotted curves (the dotted curve being the worst of the three).
The time-varying histogram approach discussed in Chapter 6 is accompanied with an R shiny app as supplementary material, which may be found at the address:


Figure B.1 displays the graphical user interface for the app. The Adaptive Forgetting tab allows the user to specify whether adaptive bin probabilities (Yes), or static bin probabilities (No) are used in the construction of the histograms. The user is also allowed to specify a pre and post change distribution. The stream is then sampled according to

\[
X_t \sim \begin{cases} 
\text{Pre-change distribution} & \text{if } 1 \leq t \leq 2500 \\
\text{Post-change distribution} & \text{if } 2500 < t \leq 5000
\end{cases}
\]

resulting in a data stream, governed by the parameters specified on the app sliders, which has an abrupt change at time $t = 2500$. The current supported pre and post change distributions are the beta, exponential, gamma, Gaussian, uniform and Weibull distributions, and once submit is pressed a video of the time-varying histogram is displayed for the user. In the video the pre-change distribution is plotted with dotted-lines, and the post-change distribution is plotted with a solid-line. Further, the parameter tuple $(w, W, \omega)$ is taken as $(5, 25, 50)$, whose values were shown to yield favorable performance in Chapter 6.
Figure B.1: The GUI for the R shiny app, which displays videos of the proposed time-varying histogram approach. The user can specify whether adaptive forgetting factors, or no forgetting factors are used – that is, the histogram uses static bin probabilities in its construction. Additionally, a pre/post change distribution, along with its parameters, may be specified. This results in the histogram being constructed using 5,000 observations – the first 2,500 being sampled from the pre-change distribution, and the last 2,500 sampled from the post-change distribution.