Adaptive Estimation and Change Detection of Correlation and Quantiles for Evolving Data Streams

a thesis presented for the degree of

Doctor of Philosophy of Imperial College London

and the

Diploma of Imperial College

by

Jordan Noble

Department of Mathematics

Imperial College

180 Queen's Gate, London, SW7 2AZ

September 2023

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: _____

Copyright

PhD advisor: Professor Niall Adams Jordan Noble

Adaptive Estimation and Change Detection of Correlation and Quantiles for Evolving Data Streams

Abstract

Streaming data processing is increasingly playing a central role in enterprise data architectures due to an abundance of available measurement data from a wide variety of sources and advances in data capture and infrastructure technology. Data streams arrive, with high frequency, as never-ending sequences of events, where the underlying data generating process always has the potential to evolve. Business operations often demand real-time processing of data streams for keeping models up-to-date and timely decision-making. For example in cybersecurity contexts, analysing streams of network data can aid the detection of potentially malicious behaviour. Many tools for statistical inference cannot meet the challenging demands of streaming data, where the computational cost of updates to models must be constant to ensure continuous processing as data scales. Moreover, these tools are often not capable of adapting to changes, or drift, in the data. Thus, new tools for modelling data streams with efficient data processing and model updating capabilities, referred to as streaming analytics, are required. Regular intervention for control parameter configuration is prohibitive to the truly continuous processing constraints of streaming data. There is a notable absence of such tools designed with both temporal-adaptivity to accommodate drift and the autonomy to not rely on control parameter tuning. Streaming analytics with these properties can be developed using an Adaptive Forgetting (AF) framework, with roots in adaptive filtering. The fundamen-

tal contributions of this thesis are to extend the streaming toolkit by using the AF framework to develop autonomous and temporally-adaptive streaming analytics.

The first contribution uses the AF framework to demonstrate the development of a model, and validation procedure, for estimating time-varying parameters of bivariate data streams from cyber-physical systems. This is accompanied by a novel continuous monitoring change detection system that compares adaptive and non-adaptive estimates. The second contribution is the development of a streaming analytic for the correlation coefficient and an associated change detector to monitor changes to correlation structures across streams. This is demonstrated on cybersecurity network data. The third contribution is a procedure for estimating time-varying binomial data with thorough exploration of the nuanced behaviour of this estimator. The final contribution is a framework to enhance extant streaming quantile estimators with autonomous, temporally-adaptive properties. In addition, a novel streaming quantile procedure is developed and demonstrated, in an extensive simulation study, to show appealing performance.

Acknowledgments

To Niall, thank you for inviting, guiding, and encouraging me on this journey. It has been immensely challenging and would not have been possible without your support.

To the Imperial College London Mathematics Department, thank you for financially supporting my PhD.

To Christoforos, thank you for originally introducing me to academic research, providing statistical consulting opportunities, and for the helpful discussions.

To Josh Neil, thank you for the opportunity to intern at Microsoft and for the advice you have given me throughout my academic career.

To Kary Myers and Aric Hagberg at Los Alamos National Laboratory, thank you for the opportunity to intern at LANL and providing the data that motivated some of this work.

To Ed Cohen, thank you for introducing me to machine learning and making statistics interesting.

To all the friends I made over the 10 years I have been at Imperial, thank you for being there and making my time at Imperial fun.

To my family, thank you for always encouraging me to pursue my own interests.

To Vessie, thank you for your unwavering patience and support.

Contents

List of Symbols

$F_X(x)$	The cumulative distribution function of the random variable X.
$(x_s)_{s\in\mathbb{N}}$	A data stream of observations $x_1, x_2, \ldots$.
w_t	The effective sample size of an adaptive forgetting procedure. Analagous to the sample size in the static case where all observations are weighted equally.
$\tilde{\theta}_t$	An adaptive estimate of θ_t that is a function of a sequence of forgetting factors, $(\lambda_i)_{i=1}^t$, as well as the data. Often corresponds to an analogous maximum likelihood estimate if θ_t is a distribution parameter.
$\hat{\theta}_t$	A static, or offline, maximum likelihood estimate of the parameter θ formed from equally weighted observations, i.e. with no forgetting.
η	The learning rate, or step size, used in stochastic gradient descent procedures, often to tune the forgetting factors. Considered fixed unless indexed by t, where it is instead an adaptive learning rate over time.
λ_t	The forgetting factor at time t. A scalar, truncated to an appropriate interval, correponding to an unnormalised weight for observation x_t. Generally tuned sequentially, but can be specified in advance.
$\mathcal{J}_t$	A next-step cost function of the current observations x_t and previous parameter estimates $\tilde{\theta}_{t-1}$. Used in stochastic gradient descent procedures, often to tune the forgetting factors.

$'$	Notation used to denote the gradient of a function, or adaptive parameter estimate, with respect to forgetting factors λ_t. For example, $\mathcal{J}_t'$ is the gradient of the next-step cost function in SGD routines, which usually depends on the parameter gradients $\tilde{\theta}_{t-1}'$.
$f_X(x)$	The probability density function of the random variable X (or probability mass function in the discrete case).
$Q_X(p)$	The quantile function of the random variable X.
$\{X_s\}_{s\in\mathbb{N}}$	An infinite stochastic process of random variables $X_1, X_2, \ldots$ generally assumed to be non-stationary unless otherwise stated.

List of Acronyms and Abbreviations

AF	Adaptive forgetting. Prefix to suggest that whatever follows uses the core methodology of adaptive forgetting as in Section 2.6.2.
AFMLE	Adaptive Forgetting Maximum Likelihood Estimator/Estimate. Obtained via maximising the adaptive forgetting likelihood as in Equation (2.4).
AFSA	Adaptive Forgetting Stochastic Approximation. A streaming single quantile estimation procedure extended from EWSA to support a data-driven dynamic rate of learning.
ARL0	Average Run Length of a changepoint detection procedure before a false positive is detected on a stationary dataset.
ARL1	Average Run Length of a changepoint detection procedure before a true positive is detected after a changepoint.
CCD	Changepoints Correctly Detected. A performance metric within the continuous monitoring context for the proportion of true changepoints correctly detected as in Bodenham and Adams (2017).
CDF	Cumulative Distribution Function.
CPS	Cyber-physical system. Combines physical devices such as sensors or actuators with cyber components to form complex systems.
CUSUM	Cumulative Sum. A statistic commonly used for control chart purposes, as originally in Page (1954).

DNF	Detections Not False. A performance metric within the continuous monitoring context for the proportion of detected changepoints that are not false as in Bodenham and Adams (2017).
DUMIQE	Dynamically Updated Multiplicative Incremental Quantile Estimator. A streaming singular quantile estimation procedure as in Yazidi and Hammer (2017).
ECDF	Empirical Cumulative Distribution Function.
EWMA	Exponentially Weighted Moving Average. A statistic commonly used for control chart purposes, as originally in Roberts (1959).
EWSA	Exponentially Weighted Stochastic Approximation. A streaming singular quantile estimation procedure as in Chen, Lambert, et al. (2000).
HVAC	Heating, Ventilation, and Air Conditioning system used in buildings to regulate temperature and ensure good air quality.
ICL	Imperial College London. Generally used in reference to the dataset collected for the purposes of the demonstration in Chapter 4.
LANL	Los Alamos National Laboratory. Generally used in reference to the computer network dataset found in Kent (2015).
MDUMIQE	Multiple Dynamically Updated Multiplicative Incremental Quantile Estimator. A streaming multiple quantile estimation procedure extended from DUMIQE that ensures monotonicity as in Hammer and Yazidi (2017).
MonotoneSA	An extension of the SA method for estimating multiple quantiles and ensuring monotonicity as in Cao et al. (2009).
NLL	Negative Log-Likelihood.

PAVA	Pool Adjacent Violators Algorithm. Used to solve the linear ordering isotonic regression problem in linear time with linear memory.
PDF	Probability Density Function. May also refer to a probability mass function in the discrete case.
ReTiNA	Real-Time Network Anomaly (Detector) as in Noble and Adams (2018).
SA	Stochastic Approximation. A family of sequential optimisation methods, generally based on Robbins and Monro (1951). Refers to the method in Tierney (1983) within the context of streaming quantile estimation.
SCAD	Streaming Correlation Anomaly Detector as in Noble and Adams (2016).
SGD	Stochastic Gradient Descent. A sequential optimisation procedure.
SL	Stochastic Learning. A streaming singular quantile estimation procedure as in Yazidi and Hammer (2016). A randomised, higher-variance version of DUMIQE.

1 Introduction

Data streams, which are composed of an unending sequence of observations arriving at high frequency, are becoming ubiquitous as advances in data collection technology continue. These advances are impacting numerous application including telecommunications, marketing, finance, weather and many more (Muthukrishnan et al. 2005). A particular, though not exclusive focus of this work is applications in cybersecurity monitoring, which manifests many of the challenges inherent to modern streaming analytics problems.

There are significant challenges to processing streaming data, and moreover, extracting statistical insight. For a variety of reasons, streaming statistical procedures are required to respect a number of constraints. First, the procedure must operate with constant storage requirement and compute demand per observation. While this can sometimes be achieved using a contiguous window of data, the approach is inelegant and leaves the problem of determining the appropriate window size. Second, the data generating mechanism is not considered to be fixed, but to evolve with unknown dynamics over time. For example, the future behaviour of a computer network may not reasonably be expected to be similar to the current behaviour. Third, the algorithmic dependence on control parameters complicates matters. It is impractical for an unending sequence of high frequency data to assume that control parameters can be fixed in perpetuity, nor could an analyst intervene to reset parameters. Hence, streaming analytics must be able to operate in an unsupervised, automatic manner.

Throughout, this research favours methods that operate sequentially on a single datum, and additionally methods from adaptive filtering (Haykin 2002) to provide some capacity to accommodate temporal variation. Notably, this class of procedure forms the basis of an adaptive forgetting framework used throughout this thesis, and makes extensive use of methods for stochastic gradient descent optimisation to provide enhanced adaptivity to streaming tools.

Another reason to favour the single datum approach arises in monitoring applications such as those in enterprise cybersecurity, where there is a need to detect malicious behaviour immediately. Indeed, a common motivation for streaming methodology is the timeliness requirement for decision-making in applications.

The extant toolkit for streaming statistical analytics is notably under-developed. While many frameworks and processing infrastructures exist for recording and processing streaming data, the statistical toolkit is incomplete. Contributing towards this toolkit is an objective of this research, by adding a variety of crucial statistical tools to existing components, such as parametric adaptive estimation (Anagnostopoulos 2010), changepoint detection (Bodenham 2014; Kifer et al. 2004), classification (Anagnostopoulos et al. 2012; Pavlidis et al. 2011) and anomaly detection (Ahmad et al. 2017; Talagala et al. 2020).

Enterprise cybersecurity provides a paradigmatic data science challenge. Vast sets of data referring to numerous types of network or computer events are routinely available within a large enterprise. Such event data is not usually recorded for the primary purpose of cybersecurity, but can be re-purposed to support anomaly detection for malicious behaviour (Rubin-Delanchy et al. 2016; Turcotte et al. 2014). Both the scale of data and the timeliness requirements for detection justify the development and deployment of streaming analytics. As such, cybersecurity examples provide excellent test cases environment for streaming analytics.

The fundamental objective of the thesis is to expand the toolkit of streaming analytics, with a focus on using the adaptive forgetting framework. The contributions of the thesis to the wider body of work are summarised in the next section.

Contributions

Much work has focussed on adaptive forgetting methods for parameter estimation with univariate data streams. Bivariate streams have received much less detailed attention. The work in Chapter 3 extends the AF framework to a specific bivariate scenario, arising from a cybersecurity problem, at the cyber-physical boundary. This work develops the AF procedure for a normal-gamma model. As an applied problem, significant attention is given to determining model adequacy, which presents specific challenges. From the AF estimation procedure, a changepoint detection method is derived. A novel feature of this method, and used more

extensively in Chapter 4, is exploiting the relationship between adaptive and non-adaptive estimators. The method is shown to perform effectively.

The correlation coefficient is a fundamental quantity of interest with bivariate data. Chapter 4 reports Noble and Adams (2018), which extends the AF framework to provide an adaptive estimator of the correlation coefficient, selecting among a number of design choices for the estimator. This estimator is again enhanced with a changepoint detector, and the trick of using both fixed and adaptive estimators is fruitfully exploited again. This work is motivated by problems of anomaly detection in enterprise cybersecurity, which further required the invention of methods to handle coincidental anomalies on multiple streams.

Among the fundamental quantities of a univariate streaming process, quantiles of the distribution have received much less attention, particularly with respect to AF methods. The remainder of the thesis is focussed on streaming quantile estimation. As noted above, the AF approach relies heavily on stochastic gradient descent. These SGD procedures are fragile, and depend on other control parameters that may be difficult to select. Moreover, when deployed in adaptive estimation settings, questions about the allowed range of parameters become relevant. Chapter 5 takes a detour into this area, to explore the behaviour of these methods, in the context of estimating a Bernoulli parameter. This choice is deliberate, since adaptive Bernoulli estimation is core to the streaming quantile estimation methods developed in Chapter 6. The investigation reveals a bias in the estimation method and explores the properties of different choices of cost function and learning rates. A novel modification to alleviate truncation bias is proposed and some loose implementation guidelines are provided.

The final contribution is the development and refinement of new streaming quantile estimation methods, including a general framework for incorporating temporal adaptivity, demonstrated on three extant streaming quantile estimators. Chapter 6 also presents the methodology and implementation for an additional novel streaming quantile estimator, `AFSQE`, before conducting an extensive simulation study. The investigation shows that there is no clear best procedure in all cases, but the adaptive methods, and `AFSQE` in particular, perform adequately across a range of different simulation scenarios, while mitigating the risk of choosing poor control parameters.

Thesis Structure

Chapter 2 reports key background material for the thesis. Chapter 3 develops an adaptive modelling framework for cyber-physical systems. Chapter 4 develops an adaptive correlation estimator and applies it in detail to enterprise cybersecurity data. Chapter 5 considers the problem of adaptive estimation for binary data and develops suitable procedures. Chapter 6 further adds to the streaming toolkit by adding highly adaptive streaming quantile estimators.

Publications

This thesis contains research published in the following papers

- The work in Chapter 3 was published in a paper titled "An Adaptive Modeling Framework for Bivariate Data Streams with Applications to Change Detection in Cyber-Physical Systems" (Plasse et al. 2017) for the 2017 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 1074–1081.

- The work in Chapter 4 was published in a paper titled "Real-Time Dynamic Network Anomaly Detection" (Noble and Adams 2018) in IEEE Intelligent Systems, Volume 33, Issue 2, 2018, pp. 5–18.

2 Background

This chapter serves as a technical introduction and summary of prerequisite material for the remaining content of this thesis. Before introducing any notion of streaming data, we begin with the existing fundamental statistical theory in a traditional *offline* setting. In this setting, observations from a data set are assumed to be instantaneously, readily available. First, we consider maximum likelihood estimation and relevant probability distributions in Section 2.1, which all the adaptive estimation procedures proposed throughout this thesis heavily rely on. Then we look at the definitions and use cases of both statistical correlation, in Section 2.2, and quantiles, in Section 2.3. The concept of statistical distance is discussed in Section 2.4, which will serve us later as a basis for assessment quantification and optimisation. Only then do we move on to introducing data streams and their challenges in Section 2.5.

Every effort has been made to ensure consistent notation throughout this thesis. Since this section covers a wide amount of prerequisite material, it also establishes elementary notation, though a comprehensive summary of symbols can be found in the List of Symbols.

2.1 Maximum Likelihood Estimation and Relevant Distributions

Parametric statistics aims to model a population with a probability distribution that has a fixed number of parameters. That is, models of the form

$$\mathcal{M} = \{f(x \mid \theta) : \theta \in \Theta\},$$

where θ is a vector of unknown parameters, existing in some parameter space $\Theta \subseteq \mathbb{R}^k$. Here, f corresponds to a Probability Density Function (PDF) in the case of continuous data, and a Probability Mass Function (PMF) in the case of discrete data.

Most data cannot be adequately modelled this way. Nevertheless, it is a convenient approach that can often serve as a reasonable approximation to the underlying data generating mechanism. Later in this thesis, we will discuss how adaptive estimation generalises the parametric modelling approach.

The goal in parametric statistics is to estimate the parameter(s) θ from some data set, or sample of observations $x_{1:N} = (x_1, \ldots, x_N)$. The most common approach to parameter estimation in this situation is maximum likelihood estimation, which, as the name implies, finds the parameter values that maximises the likelihood function of a model over the parameter space. This is akin to choosing the parameter values that correspond to the observed data being most probable under the model. The likelihood function

$$L_n(\theta \mid x_{1:N}) = f_n(x_{1:N} \mid \theta)$$

is given by the joint probability density, or mass, f_n, but as a function of the parameters θ with the data $x_{1:N}$ being held fixed. Then we define a maximum likelihood estimator as the function

$$\hat{\theta}(x_{1:N}) = \arg\max_{\theta \in \Theta} L_n(\theta \mid x_{1:N}),$$

where a specific $\hat{\theta} = \hat{\theta}(x_{1:N})$ is referred to as a maximum likelihood estimate. The acronym MLE is henceforth used for both, where context will disambiguate.

A common modelling assumption is to assume the random sample $x_{1:N}$ is independent and identically distributed (i.i.d.), with each observation modelled by the same density, or mass, f. Then the joint likelihood

$$L_n(\theta \mid x_{1:N}) = \prod_{i=1}^{N} f(x_i \mid \theta)$$

is a product of each individual likelihood function. As the logarithm function is monotonically increasing for positive values, it is common to maximise the log-likelihood $\mathcal{L}_n = \log L_n$ instead. Hence, the MLE can now be expressed in its common form as

$$\hat{\theta}(x_{1:N}) = \arg\max_{\theta \in \Theta} \sum_{i=1}^{N} \log f(x_i \mid \theta).$$

Under certain regularity conditions, there are several desirable properties of MLEs that result in them being generally useful estimators. These include consistency, functional invariance and asymptotic efficiency (Cox and Hinkley 1979).

Now we introduce some fundamental parametric probability distributions and their associated MLEs, where relevant, that form some of the core methodology later.

2.1.1 The Gaussian Distribution

The Gaussian, or normal, distribution is an example of a continuous probability distribution. Its use is ubiquitous throughout all statistical disciplines due to the nature of its convenient properties and relation to the central limit theorem. It is parameterised by a mean, or location, μ and standard deviation, or scale, σ, and its probability density function is given by

$$f\left(x \mid \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right).$$

The multivariate Gaussian distribution is a generalisation to multiple, say k, dimensions. It is parameterised by a mean vector $\boldsymbol{\mu} \in \mathbb{R}^k$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^k \times \mathbb{R}^k$ with density

$$f(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{k}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right).$$

Note that $\boldsymbol{\Sigma}$ is symmetric positive semi-definite. For our use cases, we will focus on the bivariate case, where $k = 2$. Given a multivariate set of data $\boldsymbol{x}_{1:N}$, with both mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ unknown, the MLEs are

$$\hat{\boldsymbol{\mu}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i, \tag{2.1}$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})^T. \tag{2.2}$$

The bivariate Gaussian distribution is an essential basis for the streaming correlation estimation procedure proposed in Chapter 4.

2.1.2 The Non-Standardised t-Distribution

The t-distribution is a continuous probability distribution parameterised by the number of degrees of freedom $\nu > 0$. It is a generalisation of the standard normal distribution with heavier tails that often arises when testing whether the mean of a population differs from a specified value. Specifically, let $x_{1:N}$ be an i.i.d. normal sample with mean μ and variance σ^2. The sample mean $\bar{x}$ and sample variance s^2 are given by

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2.$$

Then the sampling distribution of the pivotal quantity, or t-statistic,

$$t = \frac{\bar{x} - \mu}{\sqrt{\frac{s^2}{N}}}, \tag{2.3}$$

is the t-distribution with $N - 1$ degrees of freedom.

The t-distribution can be generalised using the location-scale transformation by introducing a location parameter a and scale parameter b. Let T be t-distributed with ν degrees of freedom. Then

$$S = aT + b,$$

follows the non-standardised t-distribution with parameters ν, a and b. Interestingly, the non-standardised t-distribution also arises when replacing the sample variance s^2 with the MLE $\hat{\sigma}^2$ in Equation (2.3). The non-standardised t-distribution has density

$$f(x \mid \nu, a, b) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)b\sqrt{\pi\nu}}\left(1 + \frac{1}{\nu}\left(\frac{x-a}{b}\right)^2\right)^{-\frac{\nu+1}{2}},$$

and is used as part of a model validation procedure in Chapter 3.

2.1.3 The Bernoulli and Binomial Distributions

The Bernoulli distribution is an example of a discrete probability distribution. It can be thought of as a model for an event with a binary outcome, such as a potentially biased coin

flip, or any trial with a chance of success or failure. It is parameterised by π, the probability of success and its probability mass function is given by

$$f(x \mid \pi) = \begin{cases} \pi & \text{if } x = 1, \\ 1 - \pi & \text{if } x = 0. \end{cases}$$
$$= \pi^x (1 - \pi)^{(1-x)}.$$

The binomial distribution is a generalisation of the Bernoulli distribution to multiple i.i.d. trials, such that a binomial random variable is a sum of finite i.i.d. Bernoulli random variables. Given the number of trials, say $m \in \mathbb{Z}^+$, its mass function is given by

$$f(x|m, \pi) = \binom{m}{x} \pi^x (1 - \pi)^{(1-x)}.$$

Hence, the Bernoulli is equivalent to the binomial distribution for $m = 1$. Given data $x_{1:N}$, the MLE for π is given by

$$\hat{\pi} = \frac{1}{Nm} \sum_{i=1}^{N} x_i.$$

Later in Chapter 6, the binomial distribution is crucial for the adaptive forgetting procedure used in streaming quantile estimation.

2.2 Correlation

It is important to understand the association between variables in any statistical modelling exercise, especially during exploratory data analysis. The extent to which random variables are related is known as the *correlation*. Of course, this relationship may not be causal, but correlation is still a useful tool for prediction purposes. Without explicitly knowing an underlying dependence structure, measuring correlation from data is challenging since no measure can perfectly capture all possible relationships. The most conventional approach is to use Pearson's correlation coefficient, which establishes the degree to which two random variables, or samples, are linearly related. Formally, Pearson's population correlation coefficient between

two random variables X and Y, with means μ_X, μ_Y and standard deviations σ_X, σ_Y, is given by

$$\rho(X,Y) = \frac{\text{Cov}\,(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathbb{E}\left[(X-\mu_X)(Y-\mu_y)\right]}{\sigma_x \sigma_Y}.$$

For the purposes of data analysis it is common to consider the correlation between two samples, say $x_{1:N}$ and $y_{1:N}$, sampled from X and Y respectively. Pearson's sample correlation coefficient is an estimate of the underlying $\rho(X,Y)$ using the sample means as plug-in estimates. That is,

$$r(x_{1:N}, y_{1:N}) = \frac{\sum_{i=1}^{N}(x_i - \bar{x}_{1:N})(y_i - \bar{y}_{1:N})}{\sqrt{\left(\sum_{i=1}^{N}(x_i - \bar{x}_{1:N})^2\right)}\sqrt{\left(\sum_{i=1}^{N}(y_i - \bar{y}_{1:N})^2\right)}},$$

where $\bar{x}_{1:N} = \sum_{i=1}^{N} x_i$ and $\bar{y}_{1:N} = \sum_{i=1}^{N} y_i$ are the sample means of $x_{1:N}$ and $y_{1:N}$. Note that $\left|r(x_{1:N}, y_{1:N})\right| \leq 1$, where values of ± 1 correspond to the data existing perfectly on a line with positive or negative slope respectively.

Later, in Chapter 4, correlation is used to detect anomalous coordinated activity in computer networks.

2.3 Quantiles

A q-quantile is a value of a distribution that corresponds to a specific proportion q of the population such that values below and including the quantile are observed with probability q. For example, the median is the 'middle' value of a distribution such that the probability of being below, and consequently above, the median is equal to 0.5. It is useful to consider the quantile function $Q(q)$ across all proportions $0 \leq q \leq 1$, defined as

$$Q(q) = \inf\{x \in R : q \leq F(x)\},$$

where $F(x)$ is the Cumulative Distribution Function (henceforth CDF) of the population. The quantile function can be seen as a generalised inverse of the CDF, since the inverse may not necessarily exist. In the case where F is continuous and strictly monotonically increasing then F is invertible and $Q = F^{-1}$. Quantile functions for the majority of parametric distributions cannot be expressed in closed-form, though some families of distributions are defined through their quantile function, most notably the Tukey lambda distribution. Applying

transformations to foundational quantile functions provides the basis for a flexible distributional modelling framework (Gilchrist 2000). Given a sample $x_{1:N}$, the Empirical Cumulative Distribution Function (henceforth ECDF) is given by

$$\hat{F}_N(x) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{x_i \leq x\}.$$

Sample quantile estimates $\hat{Q}(q)$ can then be defined by inverting the ECDF. More commonly, and equivalently, they are defined through the order statistics $x_{(1)}, \ldots, x_{(N)}$. For the case where $Nq \in \mathbb{Z}$, then the sample quantile estimate $\hat{Q}(q) = x_{(Nq)}$. This can be shown to be asymptotically unbiased normal (Cox and Hinkley 1979). Otherwise, for non-integer Nq, definitions differ between popular statistical software packages, but usually the ECDF is interpolated using a convex combination between the nearest integer-indexed order statistics (Hyndman and Fan 1996).

Later, in Chapter 6, streaming quantile estimators are developed to address the challenges of estimating quantiles in data streams.

2.4 Statistical Distances

Distance metrics play a crucial role in statistical learning theory where they commonly form the loss functions and empirical risk to be minimised. Choosing an appropriate distance metric is important in order to exploit any underlying geometrical structure of the data. Distance metrics operate on a pair of points in some space, i.e. data, but there exist statistical objects, such as probability distributions, for which it would be useful to have a notion of quantified distance. Such a distance measure would provide a means to assess model fit, or act as a cost function in a learning task for example. Throughout this thesis, the term *statistical distance* is used to refer to a measure between two probability distributions, and hence probability measures. There exist various statistical distances, each with their own properties and specific perspectives on how distributions can differ.

Statistical distances can operate on different representations of the distributions in question. For this section, consider two continuous probability distributions with PDFs $f(x), g(x)$, and CDFs $F(x), G(x)$, defined on the same probability space $\mathcal{X}$. The results generalise to discrete

distributions, which are excluded here for brevity. As is usual, the integrals would be replaced by summations, with mass functions used in place of densities.

2.4.1 Kullback-Leibler Divergence

With its solid roots in information theory, the Kullback-Leibler divergence (Kullback and Leibler 1951), or relative entropy,

$$\begin{aligned} D_{\mathrm{KL}}(F \parallel G) &= \mathbb{E}_F\left[\log \frac{f(X)}{g(X)}\right] \\ &= \int_{\mathcal{X}} f(x) \log \frac{f(x)}{g(x)} \mathrm{d}x, \end{aligned}$$

is frequently used in machine learning and statistics as it can be interpreted as the amount of information lost by choosing a model that is an approximation of the truth. Similarly, Bayesians view it as the information gain in updating a prior distribution to a posterior distribution. Of particular relevance is the relationship to maximum likelihood estimation. Assuming F represents the true data generating distribution with parameter θ, then finding the MLE $\hat{\theta}$ that parameterises G is equivalent to minimising $D_{\mathrm{KL}}(F \parallel G)$.

Importantly, the Kullback-Leibler divergence is not symmetric and violates the triangle inequality, and hence, is weaker than a metric. It is a special case of the more generalised Bregman and f-divergences and is closely related to the Fisher information metric.

2.4.2 Kolmogorov-Smirnov Distance

Typically, the Kolmogorov-Smirnov (KS) statistic is used to non-parametrically test whether a sample significantly differs from a reference distribution or a different sample. As a statistic it acts on data through the ECDF, but considered as a statistical distance it acts on the CDFs and is given by

$$d_{\mathrm{KS}}(F, G) = \sup_x \Big|F(x) - G(x)\Big|.$$

Note that for testing purposes, it is not appropriate for discrete distributions since the KS statistic then depends on the null hypothesis distribution, but it is still valid as a distance.

Intuitively, the KS distance reveals the largest amount that two distributions can differ over the probability space. It is frequently used for assessing the goodness of fit.

2.5 Evolving Data Streams

A data stream is an unending sequence of observations or measurements $(x_t)_{t\in\mathbb{N}}$ with unknown generative properties. Note, this work and most work in the streaming analytics is concerned with discrete time data and assumes the interarrival times between data observations are constant. A number of challenges arise in analysing streaming data. First, the frequency of arrival, or *velocity*, may present challenges in processing, particularly that any procedure on the data must operate in a single-pass. Second, the volume of data may be high, raising challenges related to data storage. Third, the unknown stochastic properties and possibility of future changes to the generating process — the *evolving* aspect — mean the data cannot be analysed as if it were i.i.d. Fourth, many modern applications of streaming analytics require up-to-date precision and decision-making such that the model is required to produce output that describes the current state of the stream. In building analytical procedures for data streams there is usually a need for algorithms to have fixed compute and memory requirements, otherwise the compute burden will increase as more and more data arrives.

Figure 2.1 shows an example of an evolving data stream based on the WISDM activity data set (Kwapisz et al. 2011). The grey points correspond to x-coordinate measurements from cell phone accelerometers as the user of the phone performed various activities over time. The red-dashed vertical lines correspond to points in time when the user was asked to change activity, e.g. from jogging to walking. It is clear the underlying distribution of the data changes abruptly at these points. However, even between these changepoints, it is also clear the data shows non-stationary properties. Note, the black line corresponds to an adaptive estimate of the 0.7-quantile, which can be estimated using the streaming procedures developed in Chapter 6.

To deal with the potential of data streams changing over time, any model for the data must be able to adapt over time, in a computationally feasible manner. A common approach is to model the data within sliding windows, where parameter estimates are computed within each window of data. Sliding windows are relatively simple to implement, but often require the window size to be specified in advance, which is problematic for data streams where an

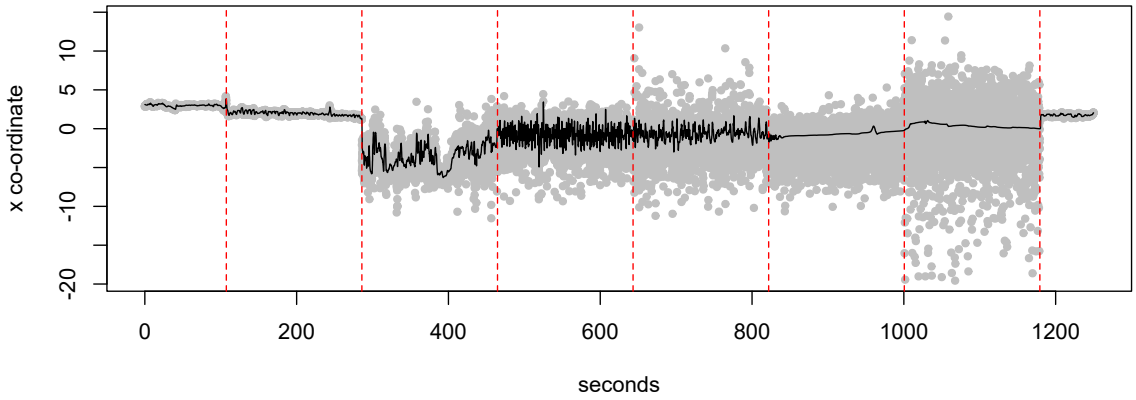

Figure 2.1: Coordinate measurements from a cell phone accelerometer as the user was asked to perform and change activity over time.

appropriate window size is difficult to determine and will likely change over time, though a notable exception is the work in Bifet and Gavalda (2007). Indeed, the concern of many methods developed in this thesis is to remove the burden of setting and fixing a parameter analogous to a window size.

A final challenge with streaming analytics is algorithm performance assessment. Like any algorithm, we may be concerned with compute and memory complexity, and as noted above, these need to be constant in time. We also should address the statistical performance of the procedure in terms of what the analytic is attempting to generate. For example, computing the mean of a time-varying process. Finally, we want to quantify the adaptability of a streaming procedure.

It is convenient to consider the observed data stream is a realisation of a stochastic process $\{X_t\}_{t\in\mathbb{N}}$, where in general, the underlying data generating mechanism is unknown and non-stationary.

2.6 Adaptive Estimation Primer

Through the use of forgetting factors (Haykin 2002), data observations can be assigned weights to give more emphasis to recent data points compared to older data. This approach can be viewed as a smooth, continuous generalisation to the discrete nature of sliding windows. Section 2.6.1 gives historical background into adaptive estimation and related work. Then

Section 2.6.2 presents an adaptive estimation framework using forgetting factors to develop temporally-adaptive estimators designed to continuously discard older data as it becomes irrelevant.

2.6.1 Background and Related Work

In 1959, an oil refinery located in Port Arthur, Texas, introduced the first integrated industrial computer control system (Nof 2009). By the early 1960s and late 1970s, the increasing prevalence and affordability of minicomputers led to a paradigm shift of factories and plants using digital computers as system controllers for industrial processes. Examples of industrial processes at the time include paper machines, ore crushers and heat exchangers.

The surge of computer control systems motivated the design of regulators for the steady state control of industrial processes. Linear stochastic control theory had already proven to be a useful tool for regulator design (Åström 1970), but required the laborious collection of experimental measurements to model system dynamics and disturbances prior to implementation. In addition, the offline computation of parameter estimates and control strategies had to be repeated if the underlying system dynamics or disturbances evolved, which was not unusual for industrial processes. By reformulating the problem with the parameters considered both *constant* and unknown, solutions could be computed using non-linear stochastic control theory obtainable, but in practice, the computational resources required to do so were not available. Instead, Åström and Wittenmark (1973) proposed *self-tuning* strategies that estimate the model parameters recursively and aim to converge to the optimal strategies in the context of a known underlying data generating mechanism. Parameter estimates updates are derived by least squares method, leading to biased estimates. While a general proof of convergence is not shown, numerical simulations are provided to demonstrate convergence in many examples. The authors do manage to prove that given convergence to the true parameters, the proposed regulator has minimal variance.

Later, Åström, Borisson, et al. (1977) considers a recursive maximum likelihood based approach for Auto-Regressive Moving Average (ARMA) models, which has proven convergence properties given the order of the ARMA process is estimated correctly. Without citing previous work, the same paper suggests exponential forgetting is "common when tracking slowly time-varying system parameters". In fact, exponential forgetting first appeared in the form of

discounted multiple regression (Brown 1963), originally motivated by earlier work on exponential smoothing (Brown 1956). Exponential forgetting relies on forgetting factors to control the extent to which past data is 'forgotten', or discounted.

Adaptive forgetting, where the forgetting factors are allowed to vary, was first considered in Fortescue et al. (1981), which developed a mechanism for self-tuning regulators. By considering a weighted sum of squares of the a posteriori errors as a measure of information content, the forgetting factors were chosen such that this information content was kept constant. The intuition behind this was to ensure that estimation is always based on the same 'amount' of information. Osorio Cordero and Mayne (1981) prove the convergence of this self-tuning regulator in an i.i.d. setting, utilising a minor modification that bounds the trace of the covariance matrix.

The first attempt at using gradient descent to choose suitable values for variable forgetting factors in the literature occurs in the context of Least Mean Squares (LMS) algorithms (Benveniste et al. 1990). The gradients used here are approximate, based on a fixed forgetting assumption. Later, the more general IDBD, or the incremental delta-bar-delta, algorithm and its close descendant, the K1 DLR algorithm were proposed to adaptively optimise the learning rate in an LMS context via gain adaptation. These approaches can be considered as *meta-learning* algorithms since they effectively learn a learning rate for the drift dynamics rather than the drift dynamics themselves.

By considering finite differences, Kushner and Yang (1995) are the first to rigorously formalise the definition of derivatives with respect to variable forgetting factors. Moreover, they prove convergence prove under weak conditions. Later in Bodenham (2014), these derivatives are developed from a first-principles approach where they are shown to agree with the approximate derivatives based on both fixed forgetting factors as in Anagnostopoulos (2010) and Benveniste et al. (1990).

2.6.2 Forgetting Factor Framework

This section introduces a forgetting factor framework that is the core basis for work in Chapters 3 to 6, where each chapter gives a specific and detailed treatment. Here, only the key aspects are reviewed.

Let $\{X_t\}_{t\in\mathbb{N}}$ be a stochastic process with time-varying, or drifting, model parameter θ_t. The evolution of θ_t can be modelled by assuming an exponential forgetting relationship on the

likelihood with *fixed* $\theta_t = \theta$ as in Anagnostopoulos et al. (2012), which places more emphasis on recent data by smoothly down-weighing past information. Given a sequence of *forgetting factors* $(\lambda_i)_{i=1}^{t-1}$ with each $\lambda_i \in [0, 1]$, the adaptive forgetting (AF) log-likelihood of a fixed set of data $(x_1, \ldots, x_t)$ is defined as

$$\mathcal{L}^{(\lambda)}(\theta \mid x_{1:t}) = \lambda_{t-1}\mathcal{L}^{(\lambda)}(\theta \mid x_{1:(t-1)}) + \mathcal{L}(\theta \mid x_t) \tag{2.4}$$

$$= \sum_{i=1}^{t} \left[\prod_{j=i}^{t-1} \lambda_j \right] \mathcal{L}(\theta \mid x_i), \tag{2.5}$$

where the empty product from $j = t$ to $j = t - 1$ is treated as equivalent to 1. Equations (2.4) and (2.5) represent the temporally-adaptive likelihood relationship in both sequential and batch form respectively. Sequential forms for parameter updates are required for memory efficient implementation, however batch forms are useful for reasoning about statistical properties. Notice this relationship generalises the log-likelihood under the i.i.d. assumption with a weighted sum, and in the case of no forgetting, where every $\lambda_i = 1$, it is equivalent to the i.i.d. case with each data point being given equal weight. This approach can be considered to be a continuous generalisation of the sliding window approach.

Using this likelihood, and maximising Equation (2.4), by setting the derivative with respect to θ equal to zero and solving, gives the adaptive forgetting maximum likelihood estimates (AFMLEs) for θ_t, as

$$\tilde{\theta}_t = \arg\min \frac{\partial}{\partial \theta} \mathcal{L}^{(\lambda)}(\theta \mid x_{1:t}). \tag{2.6}$$

Alternatively, or in the absence of a likelihood, adaptive forgetting factor estimates for the sample central moments can be estimated using a similar down-weighting scheme as in Bodenham (2014).

It remains to select the forgetting factors $(\lambda_t)_{t \in \mathbb{N}}$. One choice is to set $\lambda_t = \lambda$ for all t; the *fixed* forgetting case. The special case where $\lambda = 1$ corresponds to *no-forgetting* and results in the fixed forgetting factor mean being equivalent to a static sample mean. Interestingly the fixed forgetting factor mean update step is closely related to the update step for the EWMA (Exponentially Weighted Moving Average) chart (Bodenham 2014; Roberts 1959). The extreme

case where $\lambda = 0$ results in all past data being forgotten, where only the single, most recent observation is used.

A more flexible choice is to allow λ_t to vary; the *variable* forgetting case, which is always used in this thesis whenever the term adaptive forgetting is referenced. One way to implement adaptive forgetting is to use stochastic gradient descent (SGD) to maximise the next-step negative log-likelihood as in Anagnostopoulos et al. (2012) via the update step

$$\lambda_{t+1} = \lambda_t - \eta_t \frac{\partial}{\partial \lambda} \mathcal{L}(\theta \mid x_{t+1}), \tag{2.7}$$

where the learning rate η_t is a small quantity, often fixed. The derivative taken in Equation (2.7) is formally defined in Bodenham and Adams (2017) from a first-principles approach taking limits to determine the instantaneous rate of change. This agrees with the results in Anagnostopoulos et al. (2012) where the forgetting factor at any time t is not considered a function of previous forgetting factors, which significantly simplifies the symbolic computation of the derivative. Following Bodenham and Adams (2017), the forgetting factors are always truncated to the range $[0.6, 1]$. If this truncation is omitted, the algorithm will become unstable. This approach restricts the stochastic behaviour of the adaptive estimator. There is some subtlety around this point that is further explored in Chapter 5. Note, another choice for the cost function in the above SGD step is the next-step squared error, as covered in Bodenham (2014). Aspects related to this choice are explored in the case of Bernoulli estimation in Chapter 5.

The learning rate, or step-size, η_t controls how quickly the algorithm can accommodate changes in the underlying process. Choosing η_t is difficult; see Section 3.4.4 of Plasse (2019) for a detailed discussion, though choosing $\eta_t \in [10^{-5}, 10^{-1}]$ is generally plausible. This plausibility stems from the result of Bodenham (2014) that shows that for general i.i.d. data with mean μ and variance σ^2 that when using a squared error cost, the expected value of the derivative is of the order of the variance of the underlying process. Generally a fixed learning rate is used, however, Chapter 5 provides a more thorough treatment and comparison of both scaled and constant learning rates.

3 Adaptive Modelling and Change Detection for Data Streams in Cyber-Physical Systems

The work in this chapter is reproduced from Plasse et al. (2017) using the notation in the publication. This is concerned with applications of streaming methods for changepoint detection in cyber-physical systems.

A *cyber-physical system* (CPS) combines physical devices, such as sensors or actuators, with cyber components to form complex systems. Examples include power grids, self-driving automobiles, and office buildings. Although the integration of the physical and cyber allows for the development of advanced technology for operating the system, it also makes the system more susceptible to attacks (Lokhov et al. 2016). Issues with a CPS, whether it be faults (e.g., a broken sensor) or intrusions (e.g., unauthorised entry into the network), can have severe consequences (Cárdenas et al. 2009), especially if those issues go undetected for too long. This creates the need for a flexible, general approach for monitoring such a system for faults and intrusions *online*, or in *real-time*.

To address this need, this chapter makes three contributions. First, we introduce an *adaptive modelling framework* for the *continuous monitoring* (Bodenham and Adams 2017) of an arbitrary CPS in real-time in Section 3.2. This framework builds on a flexible statistical model called the *normal-gamma distribution*, described in Section 3.2.1, and treats the data collected from the CPS as a *data stream* to enable real-time processing. Second, we develop a novel *streaming validation procedure* in Section 3.3 to demonstrate the appropriateness of the normal-gamma assumption for cyber-physical data collected from an office building at Los Alamos National Laboratory (Figure 3.1). Third, since the ultimate goal is to support detection of faults and

intrusions in a CPS, we develop a change detection algorithm in Section 3.4, based on the adaptive modelling approach, that can be used to detect changes in a CPS in real-time. A novelty here is that this change detector exploits relationships between static and adaptive estimators. This change detection scheme is implemented on synthetic normal-gamma data in Section 3.5 and compared to a state-of-the-art offline change detector.

3.1 Related Work

The detection of cyber-attacks is well studied in the statistics and machine learning communities, e.g. (Mirkovic et al. 2002) and (Munz and Carle 2007). However, the impact of such attacks on the physical components has been less studied (Cárdenas et al. 2009), especially in a fully streaming paradigm with all model hyperparameters tuned adaptively as proposed here. In Lokhov et al. (2016) the correlations between physical time series are analysed to detect groups of anomalous behaving sensors. The survey in Zhang, Meratnia, et al. (2010) provides a thorough discussion on anomaly detection in wireless sensor networks, including Gaussian-based approaches (Wu et al. 2007), non-parametric approaches (Sheng et al. 2007), and clustering methods (Rajasegarar et al. 2006). The comprehensive discussion in Mitchell and Chen (2014) gives advantages and disadvantages for many existing detection schemes, e.g., knowledge based intrusion detection systems. Statistical based approaches can be found in Mitchell and Chen (2013), which performs intrusion detection using stochastic Petri nets, and in Linda et al. (2009), which uses neural nets.

Alternative approaches to attack detection utilise techniques from control theory to perform state estimation under various assumptions. For instance, Pasqualetti et al. (2013) considers continuous-time linear systems which take an unknown attack signal as input, while Teixeira et al. (2010) considers a weighted least squares approach to state estimation and assumes an adversary has a perturbed version of the model available.

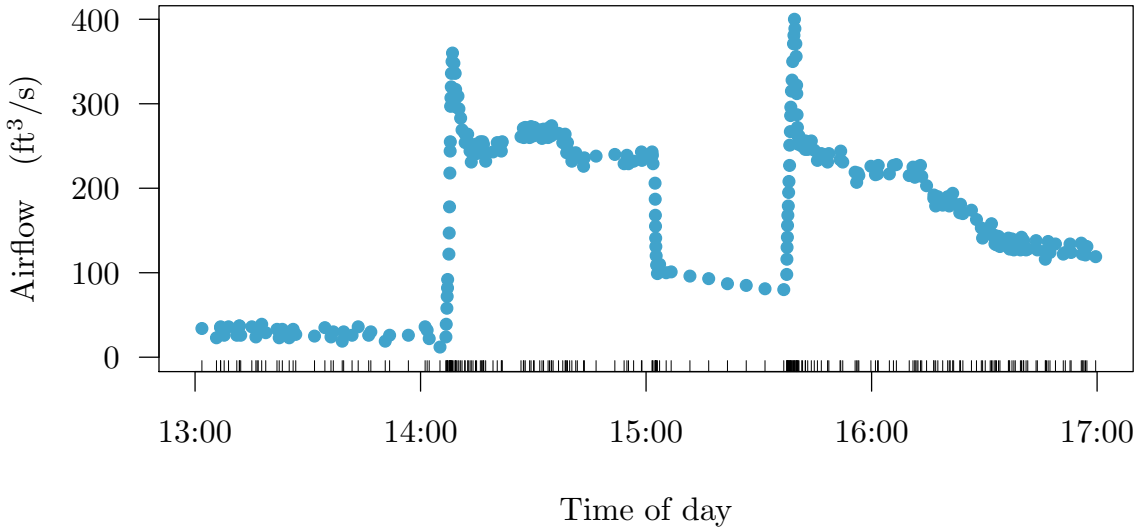

Figure 3.1: Four hours of airflow readings for one sensor in a cyber-physical system (a building) at Los Alamos National Laboratory. The ticks on the horizontal axis indicate the times at which the sensor communicated its airflow readings. Notice the irregularity of the reporting rate and how that rate increases substantially when the airflow rate experiences abrupt changes.

3.2 Adaptive modelling Framework

For the adaptive modelling framework (see Section 2.6.2) we present here, we consider the task of monitoring *bivariate* data streams $(d_t)_{t\in\mathbb{N}}$ where $\boldsymbol{d}_t = (x_t, s_t)$. The scalars x_t and s_t, respectively, represent a physical measurement recorded by the CPS (such as the airflow readings shown in Figure 3.1) and the amount of time elapsed since the last physical measurement was recorded (the distances between consecutive tick marks in Figure 3.1). Subsequently, s_t will be referred to as an *interarrival time*.

In a typical cyber-physical system, x_t and s_t will not be independent of each other for all t. For example, many systems will increase their sampling rate if a physical reading x_t changes abruptly, thus decreasing the corresponding interarrival time s_t as seen in Figure 3.1. Further, since a goal of this exposition is to monitor a CPS continuously even as the system evolves (say with time of day or season), it is unrealistic to assume that the distribution generating the data remains stationary for all time. Figure 3.2 shows an example of this non-stationary behaviour, with a substantial hour-to-hour difference in the distribution of airflow readings even within a short (two hour) time window. It is therefore necessary to provide temporally adaptive estimates for any model parameters monitored on the data stream.

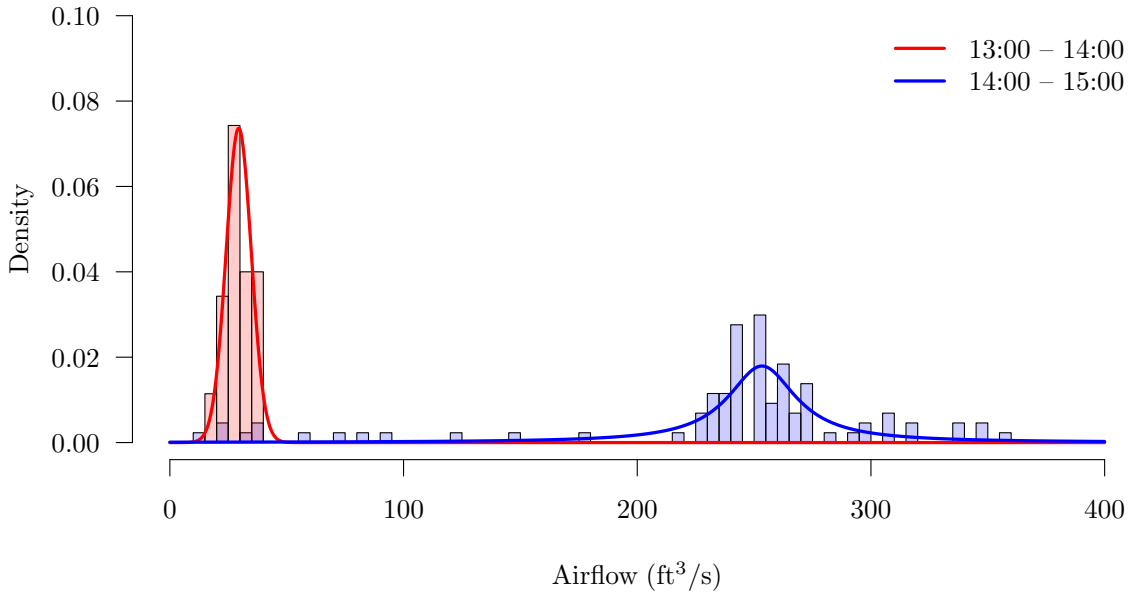

Figure 3.2: An example of how airflow distributions drift between two consecutive hourly periods for the sensor shown in Figure 3.1. The lines correspond to non-standardised t-densities (discussed in Section 3.2.1), fitted via maximum likelihood estimation. The modelling framework is able to adapt from one fitted density to the next.

Next, Section 3.2.1 introduces a flexible model, the *normal-gamma*, to address the lack of independence between x_t and s_t. Then Sections 3.2.2 and 3.2.4 develop an adaptive model estimation approach based on *forgetting factors* that addresses the lack of stationarity in the data.

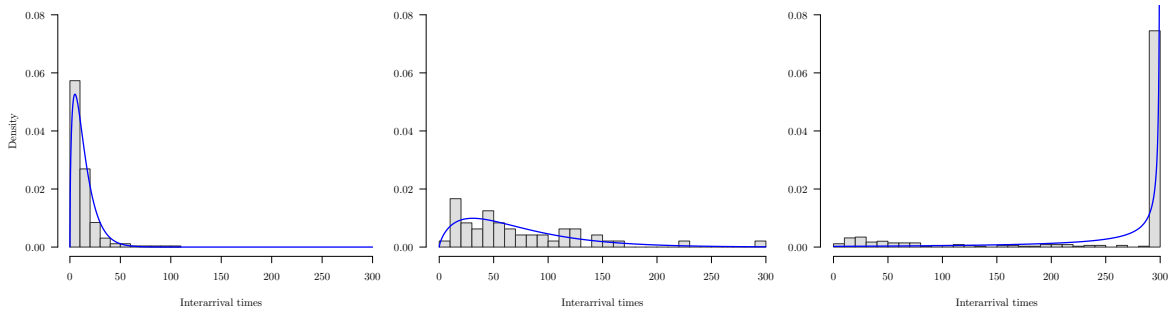

Figure 3.3: Histograms of the interarrival time distributions for three different airflow sensors over a one-hour period. The blue lines correspond to gamma densities fitted via maximum likelihood estimation to demonstrate the flexibility of the family of gamma distributions.

3.2.1 The Normal-Gamma Distribution

The normal-gamma distribution, denoted $\mathcal{NG}(\theta_t)$, is a four parameter continuous bivariate family of distributions with density given by

$$f(\boldsymbol{d}_t \mid \theta_t) = \frac{\beta_t^{\alpha_t}\gamma_t^{1/2}}{\Gamma(\alpha_t)\sqrt{2\pi}}\, s_t^{\alpha_t - 1/2} \exp\left[-\beta_t s_t - \frac{\gamma_t s_t (x_t - \mu_t)^2}{2}\right], \tag{3.1}$$

where $\theta_t = (\mu_t, \gamma_t, \alpha_t, \beta_t)$ are time evolving parameters and $\Gamma(\cdot)$ is the gamma function.

If $(X_t, S_t) \sim \mathcal{NG}(\theta_t)$ — an assumption validated on real data in Section 3.3 — we gain several desirable properties that are useful for monitoring a CPS. First, it can be shown that $X_t \mid S_t \sim \mathcal{N}(\mu_t, 1/(\gamma_t s_t))$ (the 'normal' part of the normal-gamma), which captures the dependence between the physical readings x_t and interarrival times s_t. Second, the marginal distribution of S_t is a *gamma distribution*, a flexible, continuous distribution whose support matches that of the interarrival times as seen in Figure 3.3. Interarrival times are often assumed to be exponentially distributed, a special case of the gamma distribution. Therefore, using a time evolving gamma distribution for the interarrival times, with shape and rate parameters given by α_t and β_t, encapsulates typical assumptions made in the literature. Third, the marginal distribution of the physical readings x_t can be shown to be a non-standardised t-distribution (as seen in Figure 3.2) whose parameters are a function of the components of θ_t. This will be important when developing the streaming validation procedure in Section 3.3.

Now that a model has been postulated for the tuples (X_t, S_t), a way of efficiently and adaptively estimating the parameter vector θ_t must be developed. Such a method, which uses forgetting factors, is proposed in the sections that follow.

3.2.2 Forgetting Factor Framework

Forgetting factors create temporally adaptive estimators by exponentially down-weighting older observations as new data arrives. They are commonly used in adaptive filtering (Haykin 2002) and have been studied in Anagnostopoulos et al. (2012) and Bodenham and Adams (2017) and Pavlidis et al. (2011). This framework can be thought of as a continuous analogue of a sliding window, a popular technique for streaming inference (e.g., (Bifet and Gavalda 2007)).

In what follows, a frequentist approach is adopted and a weighted version of maximum likelihood estimation is considered. The evolution of θ_t is modelled by assuming a fixed θ

and introducing forgetting factors into the parameter estimation via an *exponentially weighted log-likelihood function* of the form

$$\begin{aligned}\mathcal{L}_\lambda(\theta \mid \boldsymbol{d}_{1:t}) = \sum_{k=1}^{t} w_k \Bigg[& \alpha \log(\beta) + \frac{\log(\gamma)}{2} - \log(\Gamma(\alpha)) \\ & + \left(\alpha - \frac{1}{2}\right)\log(s_k) - \beta s_k - \frac{\gamma s_k (x_k - \mu)^2}{2} \Bigg],\end{aligned} \tag{3.2}$$

where $w_k = \prod_{p=k}^{t-1} \lambda_p$ is a function of time and controls how much *weight* is given to past data, and each $\lambda_p \in [0, 1]$ is a forgetting factor.

The values of the forgetting factors regulate how quickly (or slowly) historic data is 'forgotten'. Values of λ_p closer to zero forget data much faster than values closer to one. In Section 3.4, where a change detector is presented, the case when $\lambda_p = 1$ for all p — the *no forgetting* case — will be important.

3.2.3 Adaptive Parameter Estimation

Now that forgetting factors have been introduced into the estimation procedure, Equation (3.2) can be optimised with respect to each of the four parameters to produce temporally adaptive estimators. Since these parameter estimates need to be updated online, it is not computationally feasible to store past observations from the data stream and re-compute the estimates at each time t. Thus, *sequential* update equations need to be derived. To simplify the discussion of these updates, we introduce the notation

$$m_{f(\boldsymbol{x},\boldsymbol{s}),t} = \sum_{k=1}^{t} w_k f(x_k, s_k),$$

for $\boldsymbol{x}, \boldsymbol{s} \in \mathbb{R}^t$. In this notation, the function f is independent of the forgetting factors, x_k and s_k are the k^{th} components of $\boldsymbol{x}$ and $\boldsymbol{s}$, and w_k is as defined previously. Further, the scalar $n_t = \sum_{k=1}^{t} w_k$ will appear in the updates and is referred to as the *effective sample size* as it characterises how many observations are contributing to the estimate at any given time (Bodenham 2014). Lastly, if either $\boldsymbol{x}$ or $\boldsymbol{s}$ does not appear as an argument of f it is omitted from

the summation. Some examples of this notation, whose sequential counterparts will appear in the update equations, are:

$$m_{\log(\boldsymbol{s}),t} = \sum_{k=1}^{t} w_k \log(s_k), \qquad m_{\boldsymbol{xs},t} = \sum_{k=1}^{t} w_k x_k s_k.$$

Using this notation, as well as optimising Equation (3.2) with respect to the four parameters, results in the following set of update equations that produce the adaptive estimates $\tilde{\theta}_t = \left(\tilde{\mu}_t, \tilde{\gamma}_t, \tilde{\alpha}_t, \tilde{\beta}_t\right)$:

$$\tilde{\mu}_t = \frac{m_{\boldsymbol{xs},t}}{m_{\boldsymbol{s},t}}, \tag{3.3}$$

$$\tilde{\gamma}_t = \frac{n_t}{\xi_t}, \tag{3.4}$$

$$\tilde{\alpha}_t = \frac{(3-\zeta_t) + \sqrt{(\zeta_t - 3)^2 + 24\zeta_t}}{12\zeta_t}, \tag{3.5}$$

$$\tilde{\beta}_t = \frac{\tilde{\alpha}_t n_t}{m_{\boldsymbol{s},t}}. \tag{3.6}$$

The auxiliary variables appearing in Equations (3.3) and (3.6) can be updated recursively via:

$$n_t = \lambda_{t-1} n_{t-1} + 1, \tag{3.7}$$

$$m_{f(\boldsymbol{x},\boldsymbol{s}),t} = \lambda_{t-1} m_{f(\boldsymbol{x},\boldsymbol{s}),t-1} + f(x_t, s_t), \tag{3.8}$$

$$\xi_t = m_{\boldsymbol{x}^2\boldsymbol{s},t} - \tilde{\mu}_t m_{\boldsymbol{xs},t}, \tag{3.9}$$

$$\zeta_t = \log\left(\frac{m_{\boldsymbol{s},t}}{n_t}\right) - \frac{m_{\log(\boldsymbol{s}),t}}{n_t}. \tag{3.10}$$

The initial values at $t = 0$ in Equations (3.3) and (3.10) should be taken as zero. As the update for $\tilde{\alpha}_t$ is not available in closed form, Equation (3.5) is based on an approximate closed-form solution (see Appendix A.1). More importantly, these update equations do not require the storage of any observations prior to time t to maintain up-to-date estimates. Further, at each time t, these equations require only $\mathcal{O}(1)$ floating point operations to update the parameter estimates. Thus, the adaptive modelling framework is efficient for the data streaming from a cyber-physical system.

3.2.4 Tuning the Forgetting Factors

As mentioned in Section 3.2.2 and discussed in Section 2.6.2, the forgetting factors are parameters that control how the adaptive estimates $\tilde{\theta}_t$ incorporate older data. Thus, the forgetting factors directly affect the accuracy of the estimates. These important parameters should therefore not be chosen subjectively by a user, but rather chosen to optimise some criterion. Further, since the goal of this work is to continuously monitor a CPS that may experience several changes of unknown magnitudes, it is unlikely that a single fixed forgetting factor will suffice for an entire data stream.

As described in Section 2.6.2, the forgetting factors at time t are updated via a stochastic gradient descent (SGD) step of the form

$$\lambda_t = \lambda_{t-1} + \eta \, \nabla_\lambda \Big[\mathcal{L}\Big(\tilde{\theta}_{t-1} \mid \boldsymbol{d}_t \Big) \Big],$$

where η is a small, constant step-size. For statistical learning problems in stationary settings, it is usual to choose a vanishing $\eta_t \to 0$ to ensure asymptotic convergence. This is harmful in a non-stationary setting where the model must retain its adaptivity and so η is chosen to be non-vanishing, but still sufficiently small to ensure good parameter estimates. Given the range of the forgetting factors is $[0, 1]$ and magnitude of the gradients observed from experimentation, it is sensible to consider $0 < \eta \ll 1$ to control the size of the update as in Anagnostopoulos et al. (2012). The next-step log-likelihood

$$\mathcal{L}\Big(\tilde{\theta}_{t-1} \mid \boldsymbol{d}_t \Big) = \log f\Big(\boldsymbol{d}_t \mid \tilde{\theta}_{t-1} \Big),$$

is based on the density in Equation (3.1), whose gradient should be taken with respect to all previous forgetting factors. Following the heuristic argument given in Anagnostopoulos et al. (2012), as validated by the rigorous approach in Bodenham and Adams (2017), the gradient can

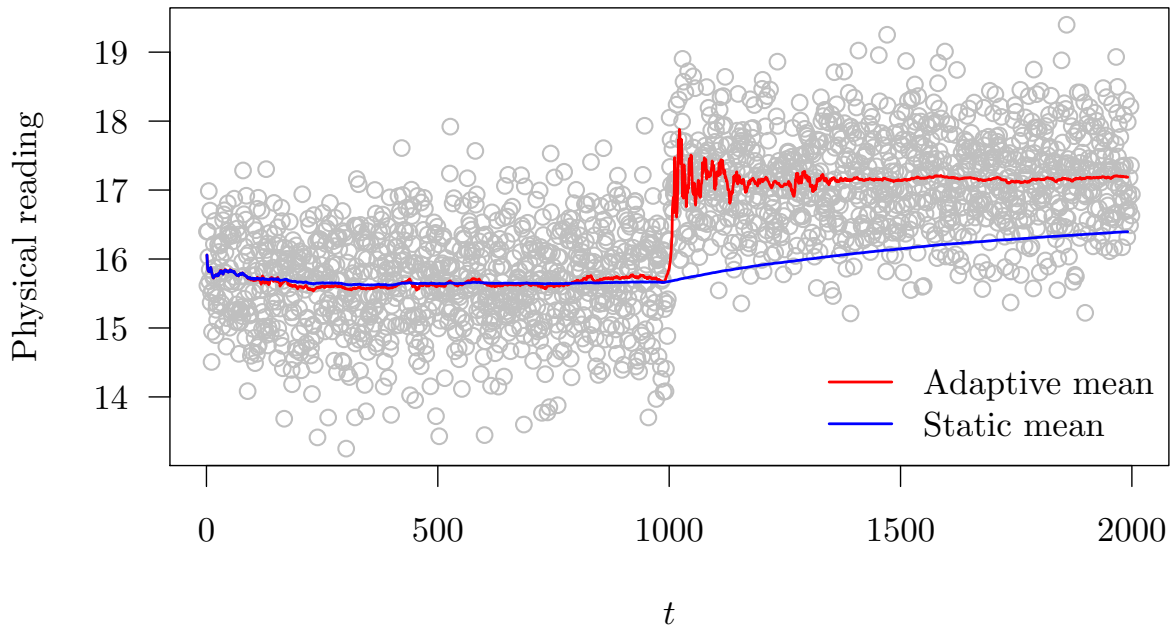

Figure 3.4: Illustration of adaptive and static estimates on a synthetic stream of physical measurements with a change in mean at $t = 1000$. In Section 3.4 we will use deviations between these two estimates to inform a change detector.

be computed by assuming that the adaptive estimate $\tilde{\theta}_{t-1}$ is a function of a single forgetting factor λ. This results in

$$\begin{aligned}\nabla_\lambda\Big[\mathcal{L}\Big(\tilde{\theta}_{t-1} \mid \boldsymbol{d}_t\Big)\Big] = \tilde{\gamma}_{t-1}s_t(x_t - \tilde{\mu}_{t-1})\nabla\tilde{\mu}_{t-1} \\ + \frac{1}{2}\left(\frac{1}{\tilde{\gamma}_{t-1}} - s_t(x_t - \tilde{\mu}_{t-1})^2\right)\nabla\tilde{\gamma}_{t-1} \\ + \Big(\log\Big(\tilde{\beta}_{t-1}s_t\Big) - \psi(\tilde{\alpha}_{t-1})\Big)\nabla\tilde{\alpha}_{t-1} \\ + \left(\frac{\tilde{\alpha}_{t-1}}{\tilde{\beta}_{t-1}} - s_t\right)\nabla\tilde{\beta}_{t-1},\end{aligned}$$

where $\psi(\cdot)$ is the digamma function and the sequential gradient updates of the parameters can be computed via direct differentiation of Equations (3.3) and (3.10). See Appendix A.2 for the full derivation.

This approach, similar to the parameter update equations, can be efficiently implemented on the data stream without storing any previous values of the forgetting factors other than λ_{t-1}. Additionally, the approach removes the burden of having to subjectively choose the forgetting factors. These *adaptive forgetting factors* will allow the parameter estimates to quickly adapt to changes in distribution; a desirable property of any streaming estimation procedure. As an

added bonus, the adaptive nature of the procedure ensures it is not sensitive to a poor choice of starting parameters since it will very quickly forget and learn sensible values based on the most recent data observed. This is why there is no initial learning period and all parameter estimates are initially set to zero. In addition, the adaptive estimates can be compared to *static* parameter estimates (i.e. estimates with no forgetting) where $\lambda_p = 1$ for every p, and deviations between these estimates are indicative of changes in the data generating process. This idea is used in the development of a change detector in Section 3.4, and is demonstrated by Figure 3.4.

At this stage, a complete adaptive modelling framework has been developed which can be used to monitor a bivariate data stream collected from an arbitrary CPS in real-time. In the next section a streaming model validation procedure is proposed and successfully implemented on real-world cyber-physical data streams (airflow readings) collected from a building at Los Alamos National Laboratory (LANL).

3.3 Model Validation for Streaming Data

To demonstrate the appropriateness of the normal-gamma model for cyber-physical data, we need to validate it against measured data. However, assessing model fit in the streaming paradigm is a challenging problem. Due to the abundance of data, and changes in the data generating process, classical model diagnostic approaches such as quantile-quantile plots and the Kolmogorov-Smirnov (KS) test are impractical. Moreover, naively using traditional measures of goodness of fit produces misleading results. In Kifer et al. (2004), the authors use the KS test on sliding windows to detect changes in data streams. The main drawback of this approach is in determining the window size since it is unknown when future changes will occur. In Lall (2015), an approximate space-efficient KS test is developed which relies on approximately inverting the output of an online estimation procedure. The storage complexity is sublinear, but the method as presented does not accommodate non-stationarity. Conversely, the incremental KS test presented in Reis et al. (2016) is valid for samples that change over time and runs in logarithmic time with respect to the sample size. However, the issue of choosing the samples, as with sliding window approaches, remains.

In this section a streaming validation procedure, which returns a score of model fit, is developed by exploiting the model presented in Section 3.2. The procedure is based on the *probability integral transform* (PIT) (Angus 1994) — a transformation of continuous random variables to standard uniform space — and returns the KS statistic (see Section 2.4.2) between the cumulative distribution functions (CDFs) of the transformed data and the standard uniform distribution. Unlike the aforementioned work, this procedure requires only constant storage and is designed for non-stationary data streams through the adaptive model. This technique is then implemented on data from a heating, ventilation, and air conditioning (HVAC) system of a large office building at LANL, demonstrating the effectiveness of the normal-gamma model fit.

3.3.1 The Validation Procedure

The validation procedure to be developed is designed to work on *univariate* data streams and is not limited to cyber-physical systems. To discuss the method generally, let $v_{1:t}$ represent observations from an arbitrary data stream whose cumulative CDF is assumed to be F_V. In the CPS setting, $v_{1:t}$ would represent either the interarrival times $s_{1:t}$ or the physical readings $x_{1:t}$. The goal of the procedure is to sequentially use the data $v_{1:t}$ to test whether the assumed CDF, F_V, is a reasonable assumption.

Under the normal-gamma model in Section 3.2, the distribution of v_t is a non-standardised t-distribution (see Section 2.1.2) with parameters $(2\alpha_t, \mu_t, \beta_t/(\gamma_t\alpha_t))$ for the physical readings. For the interarrival times, the distribution of v_t is a gamma distribution with parameters (α_t, β_t). If these distributional assumptions are sensible, then applying the PIT sequentially to $v_{1:t}$, using the appropriate CDF, will result in realisations from a standard uniform distribution. By comparing the output of the PIT to a theoretical uniform distribution, we can quantify how well the particular marginal distribution placed on $v_{1:t}$ fits the data. This idea forms the basis for the method.

Algorithm 1 presents pseudocode that describes the approach. At each time-step $t = 1, \ldots, N$, the parameter vector $\tilde{\theta}_t$ is computed via the recursive updates given in Section 3.2. The PIT

is then applied to v_t and the vector $\tilde{\varphi}_t$, which contains the estimated parameters for the distribution of v_t. In the case of monitoring a CPS via the normal-gamma model,

$$\tilde{\varphi}_t = \begin{cases} (\tilde{\alpha}_t, \tilde{\beta}_t) & \text{for interarrival times} \\ (2\tilde{\alpha}_t, \tilde{\mu}_t, \tilde{\beta}_t/(\tilde{\gamma}_t\tilde{\alpha}_t)) & \text{for physical readings.} \end{cases}$$

Let $U = \{\tilde{u}_t\}_{t=1}^N$ be the set of scalars obtained after applying the PIT to $v_{1:t}$. As stated before, if the modelling assumption is correct, these scalars would be realisations from the standard uniform distribution. Any severe departures from this distribution would be evidence that F_V is not a suitable CDF for the data. To measure 'how close' the realisations $\{\tilde{u}_t\}_{t=1}^N$ are to the standard uniform distribution, the empirical CDF of these values is computed, and is denoted by $\tilde{F}_U$. A *score* can then be computed, based on the maximum KS distance between $\tilde{F}_U$ and the standard uniform CDF. This score assumes a value in $[0, 1]$, where values closer to zero indicate that F_V is sensible for the data.

Algorithm 1 Streaming validation procedure

1: **Input:** Stream of realisations $v_{1:N}$.
2: Let $U = \emptyset$ be a set of approximate uniform mappings.
3: Initialise $\tilde{\theta}_0 = (\tilde{\mu}_0, \tilde{\gamma}_0, \tilde{\alpha}_0, \tilde{\beta}_0) = (0, 0, 0, 0)$.
4: **for** $t = 1, \ldots, N$ **do**
5: Apply Equations (3.3)-(3.10) to obtain $\tilde{\theta}_t$.
6: Let $\tilde{\varphi}_t$ be the estimates for the distribution of v_t.
7: Compute $\tilde{u}_t$ by applying the PIT to v_t and $\tilde{\varphi}_t$:

$$\tilde{u}_t \leftarrow 1 - F_V(v_t \mid \tilde{\varphi}_t).$$

8: Add $\tilde{u}_t$ to U: $U \leftarrow U \cup \{\tilde{u}_t\}$.
9: **end for**
10: Compute the empirical CDF $\tilde{F}_U$ using the elements of U.
11: **Return the KS score:** $\max_{u \in U} \left| \tilde{F}_U(u) - u \right|$.

3.3.2 Validation Results with Measured Data

The data we use for the streaming model validation procedure comes from the HVAC system of a large building at LANL. Throughout the building's eight floors, 292 sensors are considered, each reporting the airflow being delivered by the HVAC system to different parts of the building at different times. These airflow readings, along with their corresponding interarrival times,

are the two data streams that we seek to validate the normal-gamma modelling assumption against. In this exercise we tested the validity of the normal-gamma modelling assumption against 6 million bivariate tuples recorded by the 292 sensors throughout the entire month of October 2016.

The box plots in Figure 3.5 show the validation scores computed by Algorithm 1 for the 292 sensors split by building floor. Recall that lower scores indicate a better fit of the adaptive model to the data. The scores show that the fit of the non-standardised t-distribution to the airflow measurements (bottom panel) is good throughout the building. In general, we are encouraged by the good fit of the normal-gamma model for the majority of the sensors, excluding those on floors 3 and 4. Exploratory data analysis suggests that the interarrival times on these floors behave differently from the rest of the building, and a topic of future work is to determine the cause of this localised lack of fit.

In summary, the streaming validation procedure has demonstrated that the normal-gamma modelling assumption is reasonable for most of the cyber-physical data generated by LANL's HVAC system. We will therefore use the normal-gamma model in Section 3.5 to generate synthetic data to evaluate the change detector that we develop below in Section 3.4. This change detector is based on the adaptive modelling developed throughout Section 3.2 and is shown to work well for data streams where the modelling assumption is sensible.

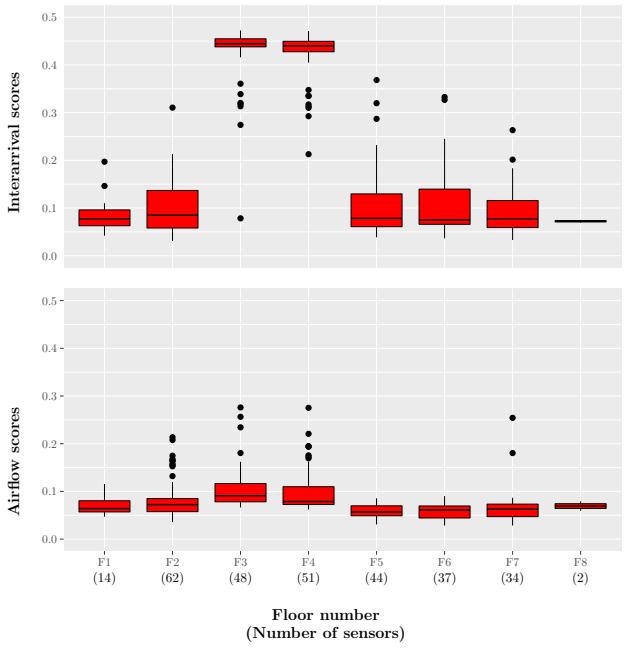

Figure 3.5: Box plots of the validation scores of the adaptive model fit for the airflow readings and interarrival times of 292 sensors in an HVAC system at Los Alamos National Laboratory during October 2016. Lower scores indicate a better fit to the model.

3.4 Change Detection

As explained earlier in the introduction to the chapter, the ultimate goal is to support detection of faults and intrusions in a cyber-physical system. Having validated the adaptive modelling framework based on the normal-gamma distribution, we will use it to construct a change detection approach, which will be demonstrated using synthetic data from the normal-gamma distribution.

Consider modelling two normal-gamma distributions, one with adaptive parameter estimates given by $\tilde{\theta}_t$, the other with the static parameter estimates $\hat{\theta}_t$. The vector $\hat{\theta}_t$ can be efficiently updated on the stream by letting the forgetting factors $\lambda_{t-1} = 1$ in Equations (3.7)-(3.8) (the no forgetting case). In this case the equations become the classical sequential update equations for the maximum likelihood estimates of the normal-gamma distribution.

In *stationary segments*, when the stream is not experiencing any changes, $\tilde{\theta}_t$ and $\hat{\theta}_t$ should be roughly equal. Now suppose the stream undergoes a change. Since $\tilde{\theta}_t$ is adaptive it will adjust to the new regime more quickly than the static $\hat{\theta}_t$ which gives equal weight to older, non-informative data. Due to this, during periods of non-stationarity, $\tilde{\theta}_t$ and $\hat{\theta}_t$ will diverge as

seen in Figure 3.4. If the divergence can be quantified, a change can be flagged whenever $\tilde{\theta}_t$ and $\hat{\theta}_t$ become 'too far' apart.

In what follows, the Kullback-Leibler (KL) divergence (see Section 2.4.1) is used to measure the dissimilarity between the normal-gamma distributions being monitored by the vectors $\tilde{\theta}_t$ and $\hat{\theta}_t$. Let $f\left(\tilde{\theta}_t\right)$ represent the normal-gamma density with estimated adaptive parameters and $f\left(\hat{\theta}_t\right)$ be the normal-gamma density with estimated static parameters. The KL-divergence, at time t, is available in closed form and is given by:

$$\begin{aligned}\kappa_t &= D_{\mathrm{KL}}\left(f(\tilde{\theta}_t) \,\middle\|\, f(\hat{\theta}_t)\right) \\ &= \frac{1}{2}\left[\log\left(\frac{\tilde{\gamma}_t}{\hat{\gamma}_t}\right) + \frac{\hat{\gamma}_t}{\tilde{\gamma}_t}\right] + \frac{\hat{\gamma}_t\tilde{\alpha}_t(\tilde{\mu}_t - \hat{\mu}_t)^2}{2\tilde{\beta}_t} + (\tilde{\alpha}_t - \hat{\alpha}_t)\psi(\tilde{\alpha}_t) - \log(\Gamma(\tilde{\alpha}_t)) + \log(\Gamma(\hat{\alpha}_t)) \\ &\quad + \hat{\alpha}_t\left[\log(\tilde{\beta}_t) - \log(\hat{\beta}_t)\right] + \tilde{\alpha}_t\left[\frac{\hat{\beta}_t - \tilde{\beta}_t}{\tilde{\beta}_t}\right] - \frac{1}{2}.\end{aligned}$$

This metric measures the dissimilarity between the adaptive and static distributions and will be zero if and only if they are identical. Therefore, κ_t should be small in stationary segments, and large in non-stationary segments where $\tilde{\theta}_t$ and $\hat{\theta}_t$ diverge. Using this scheme, a change is flagged at time t whenever $\kappa_t > \varepsilon_t$, where ε_t is an *adaptive threshold*.

Online construction of adaptive thresholds is an under researched topic, and is briefly discussed in Section 6.5. Here we follow a common practice in the literature (Carter and Streilein 2012) and let ε_t be some number of standard deviations away from the mean of the quantity being monitored. At time t, the adaptive threshold is defined as

$$\varepsilon_t = \bar{\kappa}_t + C\sigma(\kappa_t) \tag{3.11}$$

where $\bar{\kappa}_t$ and $\sigma(\kappa_t)$ are the static mean and standard deviation of the KL-divergence computed up to time t, and C is a positive constant. Both $\bar{\kappa}_t$ and $\sigma(\kappa_t)$ can be computed efficiently online, similarly to Equations (3.3)-(3.10).

Having validated the use of the normal-gamma distribution and the adaptive modelling framework, in the next section we demonstrate the changepoint detection approach through a simulation study using synthetic data generated from a normal-gamma distribution with changing parameters.

3.5 Simulation Experiments

In this section the performance of the change detector proposed in Section 3.4 is assessed and compared to a state-of-the-art offline change detector called PELT (pruned exact linear time) (Killick, Fearnhead, et al. 2012). Due to the lack of availability of ground-truth anomalies in the measured LANL cyber-physical dataset, the change detector is assessed via synthetic normal-gamma data with injected changes.

3.5.1 Performance Measures

To assess the ability of the change detector, as well as compare it to PELT, performance measures are required.

Two commonly used performance measures in the change detection literature are the *average run lengths* ARL_0 and ARL_1 (Page 1954). The scalar ARL_0 is defined as the average number of realisations that are observed until a change detector flags a false positive and is estimated via Monte Carlo simulations on a data stream with no changepoints present. ARL_1 is defined as the average number of realisations that are observed after a true changepoint until a change detector flags a true positive, and is estimated by implementing a change detector on multiple streams containing a single changepoint.

Although ARL_0 and ARL_1 are widely used for single changepoint detectors, they are not sufficient for modelling the performance of a change detector when there are multiple changepoints present (Bodenham and Adams 2017). Since the change detector proposed in Section 3.4 can report multiple changepoints it is important to define the average run length measures more formally. Suppose a change detector is run M times on a dataset of length N. For each Monte Carlo iteration, indexed by $1 \leq m \leq M$, let $\mathbb{D}_m = \left\{\hat{\tau}_1^{(m)}, \hat{\tau}_2^{(m)}, \ldots, \hat{\tau}_{D_m}^{(m)}\right\}$ be the set of time indices for all D_m detected changepoints. For computing ARL_0, the simulated data will have no changepoints, and so

$$\text{ARL}_0 = \frac{1}{M} \sum_{m=1}^{M} \min(\mathbb{D}_m \cup \{N\}),$$

where the union with $\{N\}$, ensures the measure takes into account the ideal scenario where no changepoints are detected. For computing ARL_1, the simulated data will have a single changepoint at time $\tau^{(m)}$. Hence,

$$\text{ARL}_1 = \frac{1}{M}\sum_{m=1}^{M} \min\Big(\Big\{i - \tau^{(m)} \;\Big|\; i \in \mathbb{D}^{(m)}, i > \tau^{(m)}\Big\} \cup \{N\}\Big),$$

where the union with $\{N\}$, ensures the measure takes into account the worst-case scenario where no changepoints are detected.

For situations with potentially multiple changepoints, two new measures are introduced in Bodenham and Adams (2017): the proportion of changepoints correctly detected (CCD) and the proportion of detections made that are not false detections (DNF). Again, these are measured over M Monte Carlo simulations. For each Monte Carlo iteration $1 \leq m \leq M$, let $\mathbb{D}_m = \left\{\hat{\tau}_1^{(m)}, \hat{\tau}_2^{(m)}, \ldots, \hat{\tau}_{D_m}^{(m)}\right\}$ be the set of time indices for all D_m detected changepoints, such that $T_m \leq D_m$ are *true detections*, or detections that are true positives. Then

$$\text{CCD} = \frac{1}{M}\sum_{m=1}^{M}\frac{T_m}{D_m}, \quad \text{DNF} = \frac{1}{M}\sum_{m=1}^{M}\frac{T_m}{D_m - T_m}.$$

Both CCD and DNF take their values in $[0, 1]$ with values closer to one being preferable. A value of one for both CCD and DNF means that the detector correctly detected all changepoints present without making any mistakes. It should be noted that *both* CCD and DNF should be analysed when assessing performance as either can be made optimal at the expense of the other. Note, these measures are analogous to recall and precision, respectively, in the machine learning literature.

Together, these four performance measures reveal both the speed (ARL_0 and ARL_1) and accuracy (CCD and DNF) of the detections.

3.5.2 Experimental Setup

All streams analysed in this experiment are of length 10,000. For each stream, $k \in \{0, 1, 10\}$ changepoints are simulated to evaluate the performance measures (ARL_0 and ARL_1 need to be assessed on streams with zero and one changepoint respectively, whereas the streams with ten changepoints are used to assess both CCD and DNF). This breaks up the stream into $(k + 1)$ stationary segments in which the stream does not experience any changes.

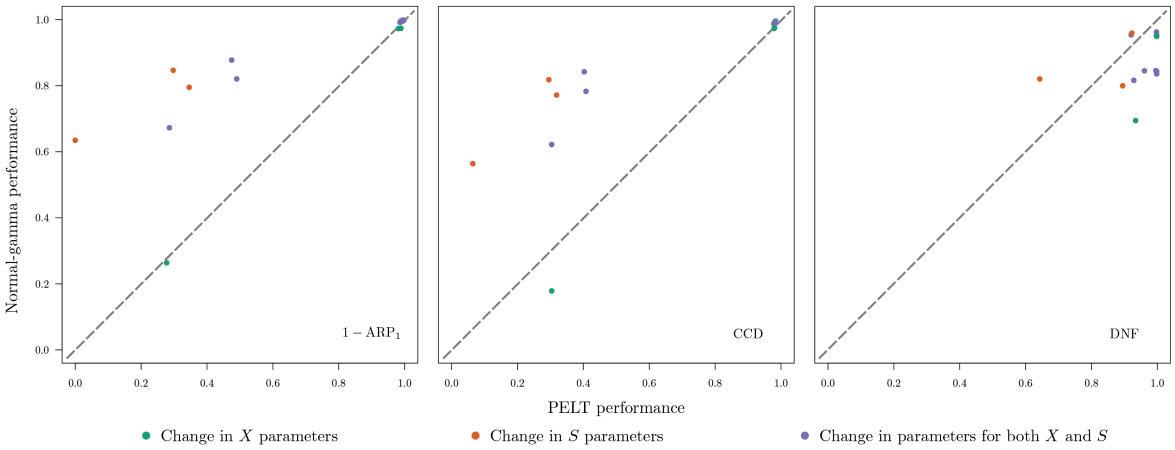

Figure 3.6: Each panel displays performance measurements for the proposed change detector and PELT, where a score of 1 is optimal. The points correspond to all fifteen possible combinations of changes in both the locations and scales of the physical readings X, and the interarrival times S. Points above the diagonal dashed line indicate where the method outperforms PELT. **Left:** a transformation of the average detection delay for a true positive (ARL_1). **Center:** the average proportion of changepoints correctly detected (CCD). **Right:** the average proportion of changepoints that are not false detections (DNF).

To simulate data in each segment, parameters $\theta_t = (\mu_t, \gamma_t, \alpha_t, \beta_t)$ of the normal-gamma distribution need to be generated. These are selected uniformly for each segment with $\mu_t \in [-25, 25]$, $\gamma_t \in \left[1, \sqrt{500}\right]$, $\alpha_t \in [100, 200]$, and $\beta_t \in \left[1, \sqrt{400}\right]$. The chosen parameter ranges ensure the simulated data covers the range of data observed across the 292 sensors. We include additional constraints based on the previous segment's parameter values such that the magnitudes of the changes are both controlled and bounded.

Accounting for potential changes in both location and scale for the distributions of X_t and S_t, there are $2^4 = 16$ possible combinations of how changes may occur: one case where no parameters change (measured by ARL_0), six combinations with changes that affect *only* one of the distributions, and nine combinations with changes to both distributions. For this experiment, a detected changepoint $\hat{\tau}$ is considered a true detection if it is the smallest detected changepoint that occurs after a true changepoint τ_i, but before any subsequent changepoints $\tau_j > \tau_i$. Then, 1,000 Monte Carlo simulations are run for each of the 16 change combinations and for each value of $m \in \{0, 1, 10\}$. We fixed the constant $C = 7$ in Equation (3.11).

To further assess the change detector's performance, it is compared with the R implementation of the PELT change detector (Killick and Eckley 2014). PELT is an offline, univariate change detection algorithm that improves the time complexity of the optimal partitioning algorithm

given in Jackson et al. (2005) while preserving the exactness of the method. Exactness here refers to the fact that, given a cost function and penalty term, PELT will return an optimal configuration of changepoints. Although reduced compared to optimal partitioning, its time complexity grows linearly with the number of data points. This makes PELT unsuitable for streaming change detection; however, it provides a good benchmark to test the online detector against.

Since the data streams we consider are bivariate, and PELT operates on univariate data, some care is required. To resolve this, PELT was implemented on two data sets using the MBIC (Zhang and Siegmund 2007) penalty. The first data set consisted of the physical readings x_t, where PELT was implemented to detect changes in both the mean and variance assuming the data was generated from a Gaussian distribution. The second data set consisted of the interarrival times s_t, where PELT was implemented to detect changes in both the mean and variance assuming the data was generated from a gamma distribution. This results in two values for each of the performance measures discussed in Section 3.5.1. For example, on any given experiment, PELT would have an $\text{ARL}_{1,X}$ corresponding to the physical readings and an $\text{ARL}_{1,S}$ for the interarrival times. To directly compare the bivariate method with PELT, the better set of the univariate performance measurements for PELT were taken, which happened to be from the physical readings for all change combinations.

3.5.3 Change Detection Results

In this section results are presented that were obtained from running the experiments discussed in Section 3.5.2 using the performance measures given in Section 3.5.1.

For the single case where no changes are present in the 1,000 simulated data streams of length 10,000, the method makes a false detection on average after 8,123 observations, whereas PELT makes no false detections; a result that is not surprising since it is an offline method.

Figure 3.6 shows the results for each of the other three performance measurements: ARP_1, CCD, and DNF. Note that ARL_1 has been rescaled such that $\text{ARP}_1 = \text{ARL}_1/N$, where $N = 10,000$. This ensures that all panels are scaled to the unit interval where 0 and 1 represent the worst and best performance attainable respectively. Each data point is a tuple of a performance measurement for both PELT and the change detector, averaged over 1,000 data streams with one of the fifteen specific change combinations described in Section 3.5.2.

In each panel, the region above the dashed line indicates where the change detector outperforms PELT. The change detector outperforms PELT with respect to the speed and accuracy of true detections, but performs marginally worse for false detections as seen in the right panel.

The left and centre panels demonstrate that PELT appears to struggle with correctly detecting changes in a timely manner when the changes are present in just the interarrival times s_t (orange points), or when there are changes in both the interarrival times s_t and the physical readings x_t, but the changes for x_t only affect the scale (the purple points clearly above the dashed line). Interestingly, both the change detector and PELT perform better for cases where any changes in the physical readings only occur in location, as seen by the cluster of points near $(1, 1)$. The single green outlier in these panels represents a change in just the scale of the physical readings where both detectors perform poorly.

It is striking that in many instances the detector outperforms PELT, an *exact* offline detector. The method achieves this by comparing both adaptive and static model parameter estimates, whereas PELT acts on the raw data.

3.6 Conclusion

By using the adaptive forgetting framework in Section 2.6.2 for Normal-Gamma data, this chapter presented a framework for monitoring a general cyber-physical system that allows for temporally adaptive parameter estimation. The difficulties of model validation in a streaming setting were discussed and addressed with a novel procedure, assessed against measurements from a real cyber-physical system. Finally, this framework was enhanced with the introduction of a novel changepoint detection method. The next chapter considers suitable methods for studying correlation structure in data streams.

4 Streaming Correlation Coefficients for Real-Time Network Anomaly Detection

The work in this chapter is reproduced from Noble and Adams (2018) using altered notation to ensure consistency throughout this document. This is concerned with applications of streaming methods for correlation estimation and changepoint detection for enterprise cybersecurity network data.

There is an emerging research effort in enterprise cybersecurity directed toward statistical anomaly detection procedures (Adams and Heard 2014; Adams and Heard 2016; Neil et al. 2013). These procedures are intended to provide complementary tools to support the range of packet-level signature-based enterprise defence systems, such as firewalls and virus scanners.

Signature-based methods are often strong, resulting in low or zero false positive rates. However, various factors are reducing the efficacy of such methods. First, ubiquitous encryption will nullify the impact of any procedure designed to query packet payload. Second, sophisticated attackers, typified by advanced persistent threat (Adams and Heard 2016, Ch. 1), are able to penetrate and traverse enterprise networks despite these defences. This capability suggests that attacker behaviour is subtle, calling for tools capable of identifying small anomalies.

Statistical anomaly detection methods are designed to detect anomalous behaviour against a putative "normal" background. As such, alerts raised from an anomaly detector are not signatures, and are not deemed to be finding malicious behaviour. Instead, such alerts are finding unusual or surprising behaviour. Careful examination of such alerts provides a means to detect sophisticated attackers, as strikingly demonstrated in Neil et al. (2013). Naturally, such alerts need combination, ranking and triage, for presentation to a network analyst.

The high volume and velocity of enterprise cybersecurity data, along with the timeliness requirements of the problem, dictate that streaming analytics are often favoured (Bodenham and Adams 2017). Such statistical analytics need to meet a number of the requirements of streaming data (see Section 2.5). Particularly, compute and memory efficient updating on data arrival, and the ability to adapt to unknown temporal variation. The latter is particularly important in cybersecurity, since network data can demonstrate significant change over time.

The main contribution of this chapter is ReTiNA (Real Time Network Anomaly-detector); an unsupervised three-stage framework for detecting anomalous network behaviour on-the-fly. Prior knowledge of the normal network behaviour is not required, and is instead learned by continuously monitoring the network traffic. The performance of ReTiNA is compared with both online and offline competitors using ideas from the changepoint detection literature. We illustrate the methodology in action on two different large computer networks. The approach is inspired by SCAD (Noble and Adams 2016), but the core methodology has been considerably enhanced with a novel online change detector, which exploits the difference in rates of convergence between adaptive and static estimators. In addition, the issue of degeneracy in covariance estimation is resolved with shrinkage.

The anomaly-based network intrusion detection literature is comprehensive; numerous papers can be found in network and information security conference proceedings such as the IEEE Symposium on Security and Privacy and the International Symposium on Research in Attacks, Intrusions and Defences. There is a wide range of proposed methods with many adapted to specific cybersecurity problems. Hence, an extensive coverage is not provided here, but reviews and surveys are thoroughly presented in García-Teodoro et al. (2009), Bhattacharyya and Kalita (2013), Bhuyan et al. (2014), Ahmed et al. (2016), Buczak and Guven (2016). Notably, the collection of methods all fall short in at least one of the following ways.

Reliance on a baseline model. Such models are constructed during an initial training phase. Both normal and anomalous network behaviour is dynamic in nature, which a static baseline model cannot hope to capture. However, some more recent methods use a sliding window approach to dynamically model behaviour, where the primary drawback is the non-trivial choice of an appropriate window size.

Not truly real-time. Methods involving some form of offline post-processing. Worse, the storage and computational complexity grow with the rate of network traffic, violating the requirements of the streaming domain.

High specificity. Methods designed for a specific type of attack do not always provide a means of generalisation to alternative attack types.

Simple heuristics. Reliance on single and simple features, ignoring the interaction effects that could be present.

Our procedure does not suffer from these shortcomings.

4.1 Overview

Netflow is a router-based protocol for assembling summaries of IP connection *events* between devices on a network. Originally, this protocol was used for accounting and resource monitoring activities. A Netflow record has a number of *fields*, including: start time, duration, source IP address, destination IP address, source port, destination port, packet count and byte volume. This summary record is assembled exactly or approximately by enumerating properties of the packets associated with the connection.

An enterprise network can be conceived of as a network graph, where *nodes* are devices and *edges* correspond to connections between devices. It is natural to distinguish *internal* nodes, those devices within the enterprise infrastructure, from *external* devices.

Connections between devices have finite temporal extent and numerous different connections can occur between devices. Thus, we have the concept of *edge activity*, the sequence of events corresponding to connections on an edge. The objective of this chapter is to explore relationships in the evolving structure of such events, to identify anomalies.

In the edge activity view of Netflow data, events on an edge can be regarded as realisations of a marked multivariate stochastic process operating on that edge, where the marks correspond to Netflow fields. While this is a convenient mathematical abstraction, we prefer to reason about the events as a sequence, where an event has extra information encoded in the Netflow fields.

We are concerned with identifying anomalies occurring in the relationship between the *inter-arrival time* of events, and Netflow fields. Given a sequence of events *on an edge*, the *correlation* between the time since the last event, and properties of the current event, such as the byte count, is a potentially valuable quantity to monitor. There is no particular reason to assume that these quantities should be correlated on a single edge. We seek to identify anomalies from the 'baseline' correlation process. Events on edges can be subject to a high degree of temporal variation and hence the methodology we deploy uses an adaptive estimation scheme, which is intended to handle such variation in a single-pass of the algorithm.

This adaptive estimation procedure is capable of detecting correlation anomalies on a single edge. More suspicion accrues when a set of edges exhibit contemporaneous anomalies, and further suspicion arises if these edges are connected. Sophisticated attack and traversal behaviour is likely to be concealed in processes on multiple edges. For example, the method in Neil et al. (2013) is concerned with short paths, or sets of connected edges. Correlation anomalies that occur over multiple edges in coincident time are thus of particular interest in cybersecurity since they may indicate coordinated activity.

The entire approach is modular. First, we use an adaptive estimation procedure to compute temporally-adaptive estimates of the correlation which dynamically model the edge activity. Second, we develop a statistical hypothesis testing procedure to detect anomalies in the correlation on an edge. Third, these edge anomalies are combined to provide an overall score for anomalous network behaviour localised in time and network space. The entire approach is represented schematically in Figure 4.1.

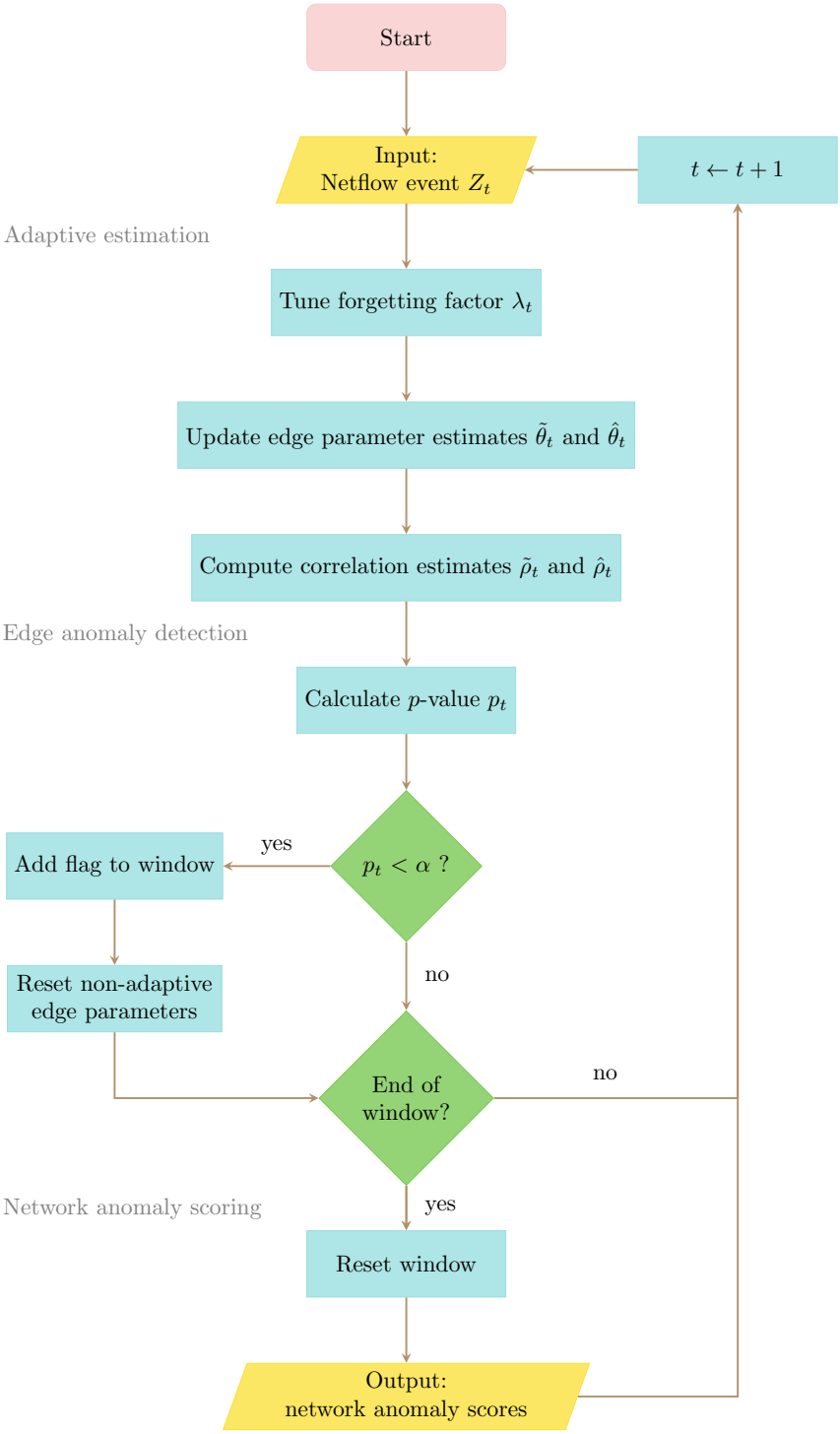

Figure 4.1: A schematic flow diagram of the three-stage ReTiNA framework.

4.2 Methodology

The core component of the approach in this section is an efficient and adaptive procedure for sequentially estimating correlation, complemented with a procedure for anomaly detection. These are the first two stages in Figure 4.1. Adaptive estimation of the correlation is reasonably

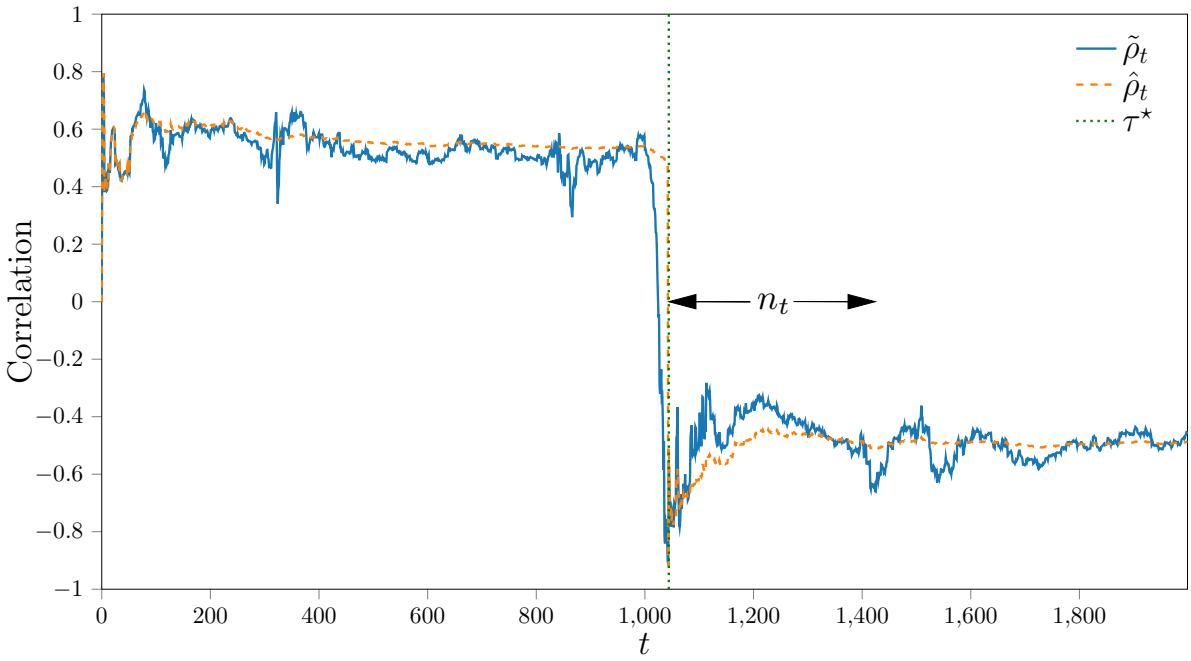

Figure 4.2: A schematic representation of the adaptive estimation and anomaly detection process for the correlation. The dashed green line represents a flagged anomaly at $t = \tau^\star$, due to the deviation between the adaptive and static estimates in solid blue and dashed orange, respectively. At this point, the static estimator is reset, by setting the static sample size $n_{\tau^\star} = 0$, where n_t in general corresponds to the number of observations seen since the most recent flagged anomaly.

straightforward, however, extending such estimation for anomaly detection requires craft. As in Section 3.4 the approach sequentially compares the adaptive estimate against its non-adaptive counterpart, as schematically illustrated in Figure 4.2.

4.2.1 Stage I: Adaptive Estimation

Let $\{Z_t\}_{t\in\mathbb{N}} = \{(X_t, Y_t)\}_{t\in\mathbb{N}}$ be a bivariate stochastic process with time-varying, or drifting, mean μ_t and covariance matrix Σ_t. The evolution of μ_t and Σ_t can be modelled by assuming an exponential forgetting relationship on the likelihood with *fixed* $\mu_t = \mu, \Sigma_t = \Sigma$ as in Anagnostopoulos et al. (2012), which places more emphasis on recent data by smoothly down-weighing past information. Specifically, given a sequence of *forgetting factors* $(\lambda_i)_{i=1}^{t-1}$ with each $\lambda_i \in [0, 1]$, the adaptive forgetting (AF) log-likelihood of $(z_1, \ldots, z_t)$ is defined as

$$\mathcal{L}^{(\lambda)}(z_{1:t} \mid \mu, \Sigma) = \lambda_{t-1}\mathcal{L}^{(\lambda)}(z_{1:(t-1)} \mid \mu, \Sigma) + \mathcal{L}(z_t \mid \mu, \Sigma). \tag{4.1}$$

Notice this generalises the log-likelihood under the i.i.d. assumption with a weighted sum, and in the case of no forgetting (where every $\lambda_i = 1$), it is equivalent to the i.i.d. case with each data point being given equal weight. Assuming $\{Z_t\}_{t\in\mathbb{N}}$ is Gaussian (see Section 2.1.1), the log-likelihood of a single realisation z_i is given by

$$\mathcal{L}(z_i \mid \mu, \Sigma) = \frac{1}{2}\Big(\log|\Sigma| + (z_i - \mu)^T\Sigma^{-1}(z_i - \mu)\Big).$$

Using this likelihood, maximising Equation (4.1), by setting the derivative (with respect to μ and Σ) equal to zero and solving, gives the adaptive forgetting maximum likelihood estimates (AFMLEs) for the moments μ_t, Σ_t and $\Pi_t = \mathbb{E}[Z_t Z_t^T]$ as

$$\begin{aligned}
\tilde{\mu}_t &= \left(1 - \frac{1}{w_t}\right)\tilde{\mu}_{t-1} + \frac{1}{w_t} z_t, \\
\tilde{\Pi}_t &= \left(1 - \frac{1}{w_t}\right)\tilde{\Pi}_{t-1} + \frac{1}{w_t} z_t z_t^T, \\
\tilde{\Sigma}_t &= \tilde{\Pi}_t - \tilde{\mu}_t\tilde{\mu}_t^T \equiv \begin{pmatrix} \tilde{\sigma}_{t,X} & \tilde{\sigma}_{t,XY} \\ \tilde{\sigma}_{t,XY} & \tilde{\sigma}_{t,Y} \end{pmatrix},
\end{aligned}$$

where

$$w_t = \lambda_{t-1} w_{t-1} + 1,$$

is the effective sample size, and is analogous to the full sample size in an i.i.d. setting. Hence, the forgetting factors control the extent of temporal adaptation of the AFMLEs. The AFMLE for the covariance, $\tilde{\Sigma}_t$, is guaranteed to be positive semi-definite, but is somewhat noisy, especially for small w_t. To reduce the variance of these estimates, and guarantee positive definiteness, the oracle approximating shrinkage (OAS) estimator in Chen, Wiesel, et al. (2010) can be used via

$$\tilde{S}_t = (1 - \gamma_t)\tilde{\Sigma}_t + \gamma_t V_t, \tag{4.2}$$

where

$$\gamma_t = \min\left(1, \frac{\mathrm{Tr}^2\left(\tilde{\Sigma}_t\right)}{n_t\left(\mathrm{Tr}\left(\tilde{\Sigma}_t^2\right) + \frac{1}{2}\mathrm{Tr}^2\left(\tilde{\Sigma}_t\right)\right)}\right),$$

$$V_t = \begin{pmatrix} \max(\varepsilon, \tilde{\sigma}_{t,X}) & 0 \\ 0 & \max(\varepsilon, \tilde{\sigma}_{t,Y}) \end{pmatrix},$$

$$n_t = n_{t-1} + 1,$$

setting $0 < \varepsilon << 1$ guarantees positive definiteness, and n_t is the number of observations since the last reset as in Figure 4.2.

The matrix $\tilde{S}_t$ is the adaptive shrinkage estimate of the covariance matrix Σ_t where

$$\begin{pmatrix} \tilde{s}_{t,X} & \tilde{s}_{t,XY} \\ \tilde{s}_{t,XY} & \tilde{s}_{t,Y} \end{pmatrix} \equiv \tilde{S}_t.$$

The Pearson correlation coefficient ρ_t can be estimated by

$$\tilde{\rho}_t = \frac{\tilde{s}_{t,XY}}{\sqrt{\tilde{s}_{t,X}\tilde{s}_{t,Y}}}.$$

The problem of determining suitable values for the forgetting factors $(\lambda_i)_{i=1}^t$ still remains, and we claim that the rate of forgetting ought to be data driven. This is made possible by using stochastic gradient descent to maximise the log-likelihood as in Anagnostopoulos et al. (2012) via the update step

$$\lambda_{t+1} = \lambda_t - \eta \frac{\partial}{\partial \lambda} \mathcal{L}\left(z_{t+1} \mid \tilde{\mu}_t, \tilde{\Sigma}_t\right), \tag{4.3}$$

where the learning rate η is a small fixed quantity and the forgetting factors are truncated to the range $[0.6, 1]$ (see Section 2.6.2). As with most procedures based on stochastic approximation, early estimates are unreliable due to the small sample size and so it is sensible to disregard the first B estimates as a burn-in phase.

Note that the derivative involved in Equation (4.3) is formally defined in Bodenham and Adams (2017) and agrees with the result in Anagnostopoulos et al. (2012) where a one-step fixed

forgetting assumption is placed on the forgetting factors. The gradient, expressed as a function of the derivatives of the AFMLEs is

$$\frac{\partial}{\partial\lambda}\mathcal{L}\Big(z_{t+1} \mid \tilde{\mu}_t, \tilde{\Sigma}_t\Big) = \frac{1}{2}(z_{t+1} - \tilde{\mu}_t)^T\Big(-2\tilde{\Sigma}_t^{-1}\tilde{\mu}_t' - \tilde{\Sigma}_t^{-1}\tilde{\Sigma}_t'\tilde{\Sigma}_t^{-1}(z_{t+1} - \tilde{\mu}_t)\Big) + \frac{1}{2}\operatorname{Tr}\Big(\tilde{\Sigma}_t^{-1}\tilde{\Sigma}_t'\Big),$$

where, as in Anagnostopoulos et al. (2012), the AFMLE *derivatives*, denoted with a $'$, are also computed sequentially by

$$\begin{aligned}
\tilde{\mu}_t' &= \Big(1 - \frac{1}{w_t}\Big)\tilde{\mu}_{t-1}' - \frac{w_t'}{w_t^2}(z_t - \tilde{\mu}_{t-1}),\\
\tilde{\Pi}_t' &= \Big(1 - \frac{1}{w_t}\Big)\tilde{\Pi}_{t-1}' - \frac{w_t'}{w_t^2}\Big(z_t z_t^T - \tilde{\Pi}_{t-1}\Big),\\
\tilde{\Sigma}_t' &= \tilde{\Pi}_t' - \tilde{\mu}_t'\tilde{\mu}_t^T - \tilde{\mu}_t(\tilde{\mu}_t')^T,
\end{aligned}$$

where

$$w_t' = \lambda_{t-1}w_{t-1}' + w_{t-1}.$$

Note, all AFMLEs, and their derivatives, are initially set to 0, with the initial forgetting factor $\lambda_0 = 1$.

This *sequential* approach ensures that the forgetting factors accommodate the evolution of the distribution of Z_t. Specifically, the forgetting factors, and hence the effective sample size w_t, will increase during periods of stability (Bodenham and Adams 2017). Similarly, they will decrease in the presence of drift or changes in the distribution of Z_t.

4.2.2 Stage II: Edge Anomaly Detection

The previous section developed an adaptive estimate for the correlation, $\tilde{\rho}_t$. This will be extended to an anomaly detector in which events on an *edge* are sequentially flagged as unusual with respect to their history. Table 4.1 shows the notation used for correlation coefficients discussed in this section.

Approximate distributional results for transformed correlation estimates are used to devise a test statistic under the assumption of no change in the correlation, in the spirit of CUSUM charts (Page 1954).

Table 4.1: Notation used for the correlation coefficients discussed in Section 4.2.

Symbol	Meaning
ρ	Pearson population correlation coefficient for the distribution used to generate an i.i.d. sample.
ρ_t	Pearson population correlation coefficient for the distribution at time t used to generate a single observation, typically evolving.
$\tilde{\rho}_t$	An adaptive estimate for the Pearson correlation coefficient ρ_t.
$\hat{\rho}_t$	The sample Pearson correlation coefficient.

Given an i.i.d. sample $(z_i)_{i=1}^N = (x_i, y_i)_{i=1}^N$ of a bivariate Gaussian random variable $Z = (X, Y)$ with population correlation coefficient ρ, the Fisher transformation (Fisher 1915) of the (*non-adaptive*) sample correlation coefficient, $\hat{\rho}_N$ is asymptotically Gaussian. Specifically,

$$\hat{r}_N = \tanh^{-1}(\hat{\rho}_N) \overset{\text{approx.}}{\sim} \mathcal{N}\left(\tanh^{-1}(\rho), \frac{1}{N-3}\right).$$

Note that the non-adaptive correlation estimates are computed sequentially using the same methodology presented in Stage I, replacing w_t with n_t, which is equivalent to setting the forgetting factors $\lambda_t = 1$ for all t.

By relaxing the i.i.d. assumption such that $(Z_t)_{t\in\mathbb{N}}$ is a bivariate Gaussian stochastic process with time evolving population correlation coefficient ρ_t, the Fisher transformation can be generalised within the sequential forgetting factor framework to give

$$\tilde{r}_t = \tanh^{-1}(\tilde{\rho}_t) \overset{\text{approx.}}{\sim} \mathcal{N}\left(\tanh^{-1}(\rho_t), \frac{1}{w_t-3}\right).$$

The assumption of no change in correlation structure can be expressed as the null hypothesis $H_0 : \rho_t = \rho$ for all t. Hence, under H_0, both the adaptive $\tilde{\rho}_t$ and non-adaptive $\hat{\rho}_t$ serve as estimates of ρ, which after the Fisher transformation, have approximately Gaussian sampling distributions as above. Then the difference between the Fisher-transformed correlation estimates is also Gaussian and can be scaled to form the test statistic

$$T_t = (\tilde{r}_t - \hat{r}_t)\left(\frac{1}{w_t-3} + \frac{1}{t-3} + k_t\right)^{-\frac{1}{2}} \overset{\text{approx.}}{\sim} \mathcal{N}(0,1), \tag{4.4}$$

where

$$k_t = 2\left(\sqrt{(w_t - 3)(t - 3)}\right)^{-\frac{1}{2}}.$$

Note, if the alternative hypothesis is true, then the adaptive correlation estimates $\tilde{\rho}_t$ converge faster to the true correlation ρ_t than the non-adaptive estimates $\hat{\rho}_t$. This property gives rise to larger values of $|T_t|$ in situations where the correlation structure changes. The corresponding two-sided p-value

$$p_t = 1 - 2(1 - \Phi(|T_t|)),$$

where Φ is the CDF of the standard normal distribution, can then be used to flag events as anomalous if $p_t < \alpha$, for some significance level α. After flagging an event as anomalous, the non-adaptive estimates need to be reset, setting $n_t = 0$.

4.2.3 Stage III: Scoring Network Anomalies

The previous section developed a sequential anomaly detector for the correlation process on a *single edge*. There is particular interest in considering anomalies across multiple edges as edge activity can be considered to be mutually independent (Neil et al. 2013). Hence, simultaneously occurring edge anomalies are indicative of overall anomalous behaviour with respect to the network. Moreover, there is little concern that false positive edge anomalies will propagate as false positive *network* anomalies since under the above independence assumption, the probability of observing a false positive across M edges is α^M.

To flag a network anomaly a windowing procedure groups together edge anomalies that occur within the same time frame T. Edge anomalies that occur between times t_1 and t_2 are added sequentially to a window W_{t_1,t_2} of fixed time length $T = t_2 - t_1$. Hence, a window corresponds to a set of flagged edges, with associated timestamps.

We propose two network anomaly scoring measures that act on these windows and report a measure of how anomalous the network behaviour is in that window. The first, ν_{t_1,t_2}, enumerates the number of unique edges amongst the flagged events in W_{t_1,t_2}. The second, κ_{t_1,t_2}, is the number of nodes in the largest connected component of the subgraph constructed from the flagged edges within W_{t_1,t_2}. It is designed to capture coordinated behaviour, such as might be observed in a sophisticated attack. Each measure is scaled by the maximum value the measure could take in the network to ensure sequential comparability.

It is particularly suspicious to observe windows where both scores agree since this implies that all flagged edges are connected, which is a strong signal of anomalous behaviour.

4.3 Evaluation

We now turn to the assessment of this anomaly detection methodology. The evaluation of network intrusion detection systems is a major challenge for in cybersecurity due to the dearth of ground-truth data. A compromise is to inject anomalies representative of attacks in real datasets (Sommer and Paxson 2010). In Noble and Adams (2016) the performance of SCAD was evaluated on real data with contrived anomalies induced by data contamination. Here, a more rigorous and complete simulation study is conducted to capture all performance aspects of ReTiNA and its competitors.

4.3.1 Performance Measures

In the classical sequential analysis literature, the focus is on the ability of a detector to accurately detect a single change. In that context, the standard performance measures are ARL_0 and ARL_1 (Page 1954). The former is the *average run length* before encountering a false positive in an environment with no change. The latter is the average number of observations between the true changepoint and the detected changepoint.

These measures by themselves are flawed when assessing algorithms that report *multiple* changepoints as they do not take into account the number of detections made. In Bodenham and Adams (2017), two complementary performance measures, CCD (proportion of changepoints correctly detected) and DNF (proportion of detections that are not false detections) are proposed for considering multiple changepoint detectors. Note that these are analogous to recall and precision in the pattern recognition literature.

4.3.2 Simulation Plan

The performance measures defined in the previous section are computed using Monte Carlo simulation. For ARL_0 a sequence of N_0 i.i.d. bivariate standard normal variates is simulated

and any flagged change is noted. To compute the other measures, data $(a_i)_{i=1}^{N_1}$ is simulated from

$$A_i \sim \begin{cases} \mathcal{N}_2(0, \Sigma_1), & \text{if } i < \tau, \\ \mathcal{N}_2(0, \Sigma_2), & \text{if } i \geq \tau, \end{cases}$$

where

$$\Sigma_1 = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix}, \quad \Sigma_2 = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix},$$

and $\tau = N_1/2$ is the changepoint. The covariance matrices Σ_1 and Σ_2 are chosen such that the correlation before the changepoint, $\rho_1 = -0.5$, is different to the correlation after the changepoint, $\rho_2 = 0.5$. A detected changepoint $\hat{\tau}$ is considered to be a true detection for the changepoint τ if it is the smallest detected changepoint such that $\tau \leq \hat{\tau}$, and is considered to be a false detection otherwise.

For each simulation, the adaptive correlation estimates are computed from the raw data a_i. This sequence provides the input for a selection of competing change detectors proposed in the literature. Figure 4.2 illustrates a run-through of ReTiNA on the above simulation environment. ReTiNA is compared with various methods. AFF (Bodenham and Adams 2017) and SCAD are both sequential anomaly detectors, whereas PELT (Killick, Fearnhead, et al. 2012) and AMOC (Killick and Eckley 2014) are non-sequential procedures intended to provide a baseline for achievable performance. AMOC is designed to detect a single changepoint and so the CCD and DNF measures are misleading, whereas the other methods allow for multiple changepoints.

ReTiNA, SCAD and AFF all assume the pre-change distribution of the data is Gaussian. PELT assumes the number of changepoints grows linearly with the size of the dataset, whereas AMOC assumes only a single changepoint is present. Both implementations for PELT and AMOC require a specified cost function, which for the purposes of this simulation was chosen to capture changes in both the mean and variance, assuming the distribution of the data is always Gaussian. For PELT, the MBIC (Zhang and Siegmund 2007) penalty term was included in the cost function to prevent overfitting.

4.3.3 Simulation Results and Discussion

Table 4.2 shows the results of 10,000 simulations with $N_0 = 10,000$ and $N_1 = 2,000$. The results clearly show that ReTiNA is comparable even with batch methods and hence shows great promise as a streaming anomaly detector. In particular, the large ARL_0 and DNF values show that ReTiNA is almost as resilient to false positives as the batch methods and much more competitive than the competing online methods. In addition, the perfect CCD indicates that changepoints are always detected.

Table 4.2: Performance measures for the considered anomaly and change detection algorithms over 10,000 simulations of AF correlation estimates. For all measures except ARL_1, higher numbers indicate better performance.

Method	Online	ARL_0	ARL_1	CCD	DNF
ReTiNA	✓	9390.52	48.06	1.0000	0.9978
SCAD	✓	197.92	124.22	0.9993	0.0970
AFF	✓	9698.59	161.46	0.9402	0.2436
AMOC	✗	10000.00	27.35	0.9997	0.9997
PELT	✗	9996.02	28.14	0.9997	0.9700

4.3.4 Implementation Considerations

The full ReTiNA procedure relies on setting five hyperparameters, which are discussed below.

The stochastic gradient descent update step in Equation (4.3) is sensitive to the choice of the learning rate η, presenting a trade-off. Larger values of η give rise to noisier estimates, but choosing too small a learning rate can significantly impact the temporal adaptation of the model. Some advice regarding the setting of this parameter is given in Bodenham and Adams (2017), where it is also shown that in the univariate Gaussian case, the magnitude of the gradient is of the same order as the underlying variance. Here, the value of $\eta = 0.001$ was used throughout, after empirical validation on the simulations and real data.

Both ARL_0 simulations for AMOC and PELT often erroneously reported false positives within the first 25 observations owing to the AF correlation estimates being initially noisy. For this reason, choosing $B = 25$ for the simulations seemed sensible. Such a short burn-in phase is promising evidence that ReTiNA does not need much data to initially learn edge

behaviour. For the demonstration, the length of the burn-in phase was increased to correspond to the first hour's worth of data.

The significance level α, used for flagging edge anomalies, controls the false positive rate. Larger values of α would lead to larger number of flagged anomalies. Note that the analysis is not particularly sensitive to the value of α since false positives are likely to be filtered out by subsequent triage in stage III. The significance level $\alpha = 0.01$ was used for both simulations and the demonstration on Netflow data as this value achieved the best results in the simulations.

The k_t term in Equation (4.4) accounts for the dependence between the adaptive and non-adaptive estimators. Practically, assuming no dependence by setting $k_t = 0$ for all t, led to improved ARL_1 results compared to fully accounting for the covariance.

In principle, the network anomaly scores are sensitive to the choice of window length T. However, in practice, choosing different values of T, ranging from 10 seconds to 15 minutes, did not substantively change the distributions of the network anomaly scores. For this reason, the window length was set to $T = 1$ minute in the demonstration.

4.3.5 Demonstration Plan

A subset (62 edges) of the Los Alamos National Laboratory (LANL) Netflow data was considered in Noble and Adams (2016) from the "Comprehensive, Multi-Source Cyber-Security Events" data set (Kent 2015). A key finding in that study was that SCAD revealed multiple network correlation anomalies at similar times with known suspicious behaviour.

Here, we deploy the improved methodology of ReTiNA on a much larger subgroup of the LANL Netflow dataset. Additionally, we analyse Netflow data collected from the busiest router on the Imperial College London (ICL) network. We expect these two networks to be different. LANL is a US government laboratory, operating under reasonably strict controls. The ICL network is subject to a much wider variety of activity. Furthermore, the LANL data contains activity belonging to a group of authorised attackers, referred to as a red team (Adams and Heard 2016, Ch. 2). The ICL data is recorded at millisecond resolution, whereas the LANL Netflow times are recorded at second resolution, which introduced bias in the adaptive estimation stage and caused degeneracy issues in SCAD. ReTiNA resolves these degeneracy issues by shrinking the adaptive covariance estimates, as in Equation (4.2).

For each network, we run the full ReTiNA procedure on three days of activity related to *internal* traffic, considering the busiest edges of each network, specifically those edges with at least 2,000 events. This corresponds to 2,359 and 2,061 edges, with over 18 and 9 million events for the ICL and LANL networks respectively.

4.3.6 Demonstration Results and Discussion

Over 20,000 edge anomalies are flagged in the LANL network with over three times that for the ICL network. The distribution of the number of anomalies on an edge also differs between the networks; both are zero-inflated, but for the ICL network there is an additional mode.

Figure 4.3 shows the resulting ν and κ scores corresponding to 1-minute windows for the ICL and LANL network respectively; localising network anomalies in time. The panels for each network should not be compared against each other, but they share the same horizontal axis since both networks have 4,320 network anomaly score values (corresponding to 3 days worth of 1-minute windows). For the ICL network, the rate of network activity is displayed alongside the ν scores, verifying that the periods of zero score occur when no activity is present. These panels share the same horizontal axis, which corresponds to the index of the 1-minute windows.

Note that the red team 'compromise' events are separate from the Netflow events (manifesting in the "authentication" data) and unknown to the anomaly detector. In fact, none of the source or destination devices associated with red team activity are present in the selected LANL network. Given this, it is striking that after the first batch of red team authentications, two anomalies are clearly revealed in the κ score near window number 2,700. Figure 4.4 shows the LANL network graph, with flagged edges in these two windows in red; localising network anomalies in network space. The large cluster of connected flagged edges centred around the green node is particularly interesting. Though any degree of certainty as to whether these anomalies are directly related to the red team activity is difficult to obtain given the somewhat independent relationship of the LANL Netflow and authentication data.

Appropriate thresholds for the scoring measures are defined by operational aspects. Figure 4.3 displays a threshold of the 0.999-quantiles for each scoring measure, leaving the four

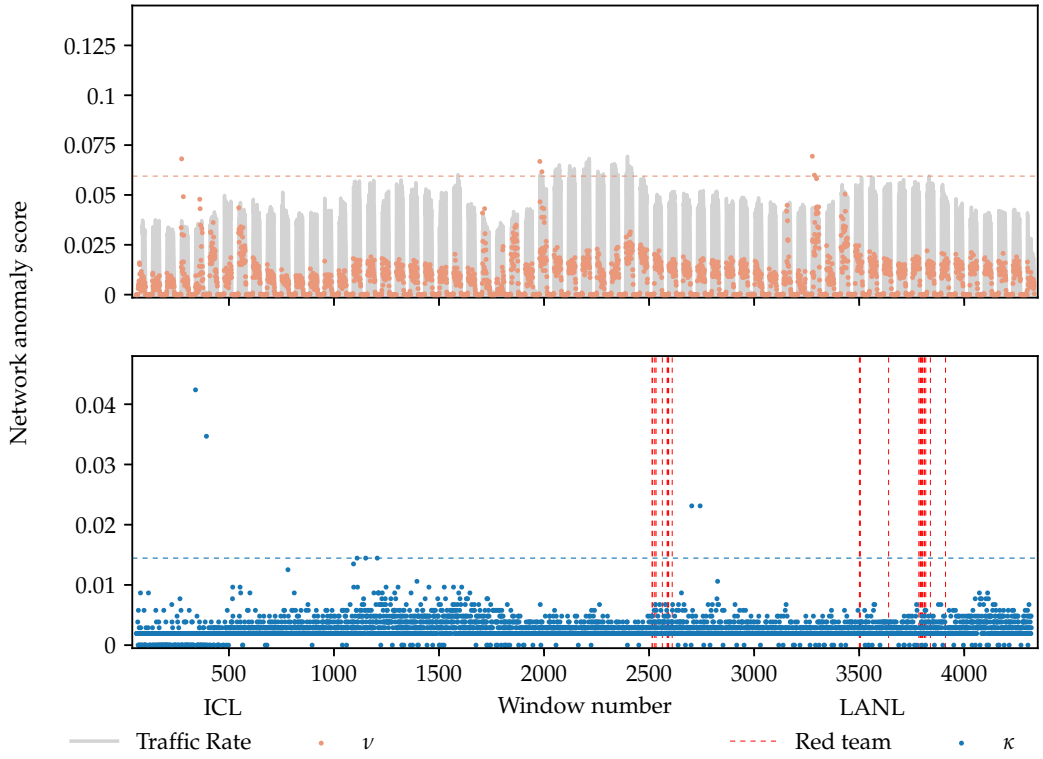

Figure 4.3: Plot of the two network anomaly scores ν and κ for each 1-minute window for the ICL (top) and LANL (bottom) networks. The horizontal dotted lines correspond to the 0.999-quantiles of the distribution for each score. The grey background (top) corresponds to a rate proportional to the number of events over all edges within the window. The vertical red dotted lines (bottom) correspond to red team authentication events.

highest scoring 1-minute windows of activity over the three days. Note that for the LANL network, this strict threshold still reveals the interesting aforementioned anomalies.

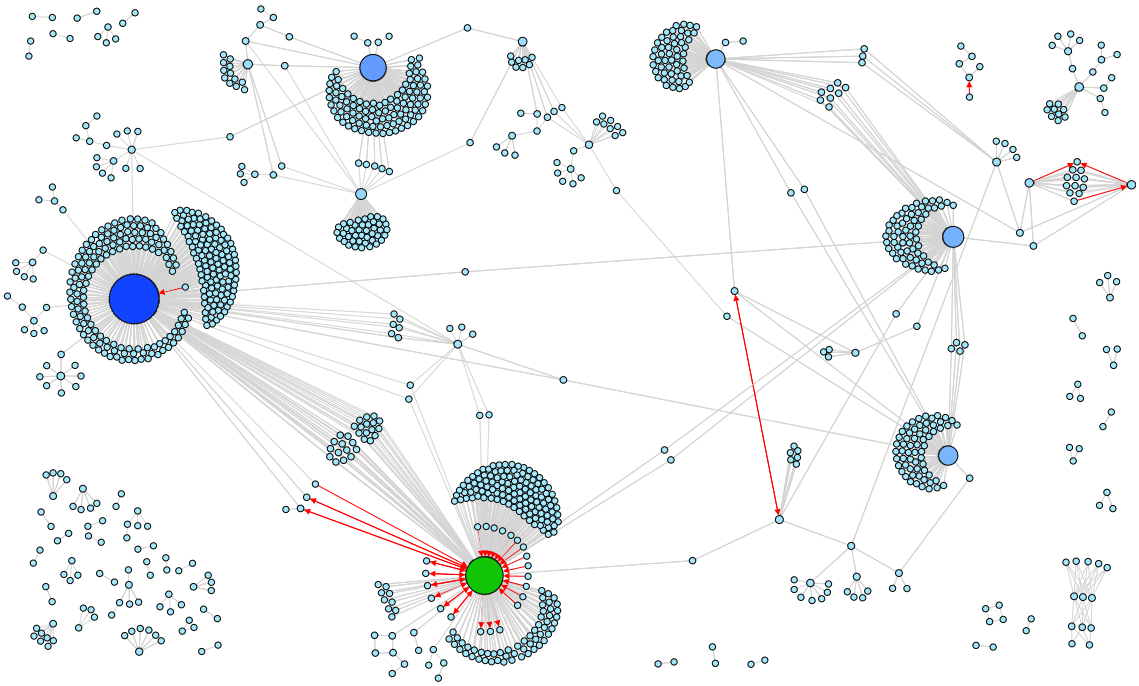

Figure 4.4: The LANL network graph for all edges with at least 2,000 events. Red edges correspond to the edges with flagged anomalies for the two highest scoring time windows after the red team authorisation events. The node coloured in green is the centre of a large connected component of flagged edges.

4.4 Conclusion

This work demonstrates a novel approach to detecting anomalies in network traffic using the ReTiNA framework to identify and aggregate changes in correlation structure on edges. The method requires constant memory, with respect to the size of the network graph, and computation time proportional to the rate of edge activity. To achieve this, the adaptive forgetting framework was extended for the online estimation of the correlation coefficient. Combined with a novel change detection scheme for correlation coefficients, this developed a framework for reasoning about anomalies in a network.

The next two chapters seek to explore another interesting aspect of data streams: quantiles. To achieve this we take a closer look at the adaptive estimation procedure for Bernoulli data in Chapter 5, which is then used as a basis for constructing streaming quantile estimators in Chapter 6.

5 Adaptive Estimation for Bernoulli Trials

The work in this chapter develops adaptive estimation procedures for binary data, which are fundamental for developing the adaptive streaming quantile procedures in Chapter 6. For a historical background of adaptive estimation and discussion of related work see Section 2.6.1. It turns out that adaptive estimation procedures for binary data are understudied and lead to peculiar and undesirable estimation behaviour. This behaviour is explored in detail.

As with many procedures dependent on stochastic gradient descent, the implementation specifics of setting control parameters are as much an art as a science. We aim to get a better understanding of how to choose the cost function and learning rate. Ultimately, these decisions depend on the situation, but we make some general recommendations.

Section 5.1 proposes methodology to develop a procedure, `AFFB`, for estimating a time-varying Bernoulli parameter. Sections 5.2 and 5.3 explore the behaviour of `AFFB` when operating on stationary and non-stationary binary data respectively, in addition to assessing performance on simulated data. Central to the tuning of adaptive forgetting factors, is a truncation step (see Section 2.6.2). Section 5.4 explores relaxing this truncation. Finally, Section 5.5 considers the extension of the approach to the case of binomial data.

5.1 Adaptive Parameter Estimation

Let $\{Y_s\}_{s\in\mathbb{N}}$ be a stochastic process of Bernoulli random variables with time-varying parameter θ_s, such that the probability mass function at any time t is given by

$$f_{Y_t}(y) = \begin{cases} \theta_t & \text{if } y = 0, \\ 1 - \theta_t & \text{if } y - 1. \end{cases}$$

Given a sequence of forgetting factors $(\lambda_s)_{s=1}^{t-1}$ with each $\lambda_s \in [0,1]$, treating $\theta_s = \theta$ as fixed allows the modelling of parameter evolution through the adaptive forgetting log-likelihood relationship in Section 2.6. Specifically,

$$\begin{aligned}\mathcal{L}^{(\lambda)}(\theta \mid y_1, \ldots, y_t) &= \mathcal{L}^{(\lambda)}(\theta \mid y_1, \ldots, y_{t-1}) + \mathcal{L}(\theta \mid y_t) \\ &= m_t \log\theta + w_t \log(1-\theta) - m_t \log(1-\theta),\end{aligned}$$

where

$$\begin{aligned}m_t &= \lambda_t m_{t-1} + y_t, \\ w_t &= \lambda_t w_{t-1} + 1.\end{aligned}$$

For a full derivation see Appendix A.3. Then differentiating with respect to θ and solving for $\frac{d}{d\theta}\mathcal{L}^{(\lambda)}(\theta \mid y_1, \ldots, y_t) = 0$ gives the adaptive forgetting maximum likelihood estimate for θ_t in closed form as

$$\tilde{\theta}_t = \frac{m_t}{w_t}$$

which is the adaptive forgetting analogue of the sample mean as in Equation (3.11) in Bodenham (2014), and equivalent to the sequential update

$$\tilde{\theta}_t = \left(1 - \frac{1}{w_t}\right)\tilde{\theta}_{t-1} + \frac{1}{w_t} y_t.$$

5.1.1 Tuning the Forgetting Factors

As discussed in Section 2.6.2, suitable values for the forgetting factors can be determined by choosing values that minimise a time-dependent cost function $\mathcal{J}_t$ suitable for capturing adaptivity, such as a next-step ahead distance between the arriving data and current model. Two sensible choices will be considered and compared in this chapter. The first is the next-step ahead negative log-likelihood (NLL)

$$\mathcal{J}_t^{(1)} = -\mathcal{L}\Big(\tilde{\theta}_{t-1} \mid y_t\Big),$$

as frequently used in Anagnostopoulos (2010) and Anagnostopoulos et al. (2012). The second is the next-step ahead squared error

$$\mathcal{J}_t^{(2)} = \Big(y_t - \tilde{\theta}_t\Big)^2,$$

as used in Bodenham (2014) and Bodenham and Adams (2017). Future references to these cost functions excludes the "next-step" term, common to both cost functions, for the purpose of brevity.

Following the nature of the forgetting factor framework, these cost functions can be sequentially optimised by using stochastic gradient descent, provided the gradient is available. In Bodenham (2014) the author derives similar gradients using first principles, which agrees with a simpler approach used in Anagnostopoulos (2010) where the derivative is taken assuming the forgetting factors are fixed one step-ahead. Using the latter approach and denoting $\mathcal{J}_t' = \frac{\partial}{\partial\lambda}\mathcal{J}_t$ for the gradient, each of the two cost function gradients can be expressed as

$$\mathcal{J}_t^{(1)\prime} = -\tilde{\theta}_{t-1}'\left(\frac{y_t}{\tilde{\theta}_{t-1}} - \frac{1-y_t}{1-\tilde{\theta}_{t-1}}\right),$$

and

$$\mathcal{J}_t^{(2)\prime} = 2\tilde{\theta}_{t-1}'\Big(y_t - \tilde{\theta}_{t-1}\Big),$$

where the following derivatives are defined sequentially

$$\begin{aligned}\tilde{\theta}_t' &= \left(1 - \frac{1}{w_t}\right)\tilde{\theta}_{t-1}' - \frac{w_t'}{w_t^2}\Big(y_t - \tilde{\theta}_{t-1}\Big),\\ w_t' &= \lambda_{t-1}w_{t-1}' + w_{t-1}.\end{aligned}$$

Note, for implementation purposes it is convenient to define and set $\mathcal{J}_s^{(1)\prime} = 0$ if $\tilde{\theta}_{t-1} \in \{0, 1\}$, which is the case in the first iteration of the procedure where a single observation is not sufficient to determine a rate of change.

Algorithm 2 presents `AFFB`; the overall Adaptive Forgetting Framework Bernoulli estimation procedure in a general form for implementation purposes. The procedure starts by initialising parameter values on line 3. Then, as in lines 4–5, as each observation y_t arrives, the parameters are sequentially updated using AFFB_UPDATE, which is defined on line 6. Subscripts denoting

the time index are intentionally excluded from parameters to emphasise the constant compute and storage complexity of the procedure. Note the order of steps within AFFB_UPDATE is crucial to ensuring the sequentially updated values respect all the sequential update equations above. Also note the bounds of the truncation range are considered variable, where the truncation step is discussed later in Section 5.4.

Algorithm 2 `AFFB`: a general adaptive forgetting estimation procedure for Bernoulli data streams.

Input: Stream of realisations $y_s \in \{0, 1\}$, for $s = 1, 2, \ldots$.
Let $\mathcal{J}(y, \theta)$ be an appropriate cost function for any $y \in \{0, 1\}, \tilde{\theta} \in [0, 1]$. Let the step-size η_s be a deterministic function of s and w. Let $[\lambda_{\min}, \lambda_{\max}]$ be the truncation range of the forgetting factors.
Initialise $(\lambda, w, w', \tilde{\theta}, \tilde{\theta}') = (1, 0, 0, 0, 0)$.
for $s = 1, 2 \ldots$ **do**
    Set $(\lambda, w, w', \tilde{\theta}, \tilde{\theta}') = \text{AFFB\_UPDATE}(y_s, \lambda, w, w', \tilde{\theta}, \tilde{\theta}')$
function AFFB_UPDATE$(y_s, \lambda, w, w', \tilde{\theta}, \tilde{\theta}')$
    Compute cost gradient: set $g = \mathcal{J}'(y, \theta)\Big|_{y=y_s, \theta=\tilde{\theta}}$
    e.g. for the squared-error cost $g = 2\tilde{\theta}'\Big(y_s - \tilde{\theta}\Big)$.
    Update forgetting factor: $\lambda = \lambda - \eta_s g$.
    Truncate forgetting factor: $\lambda = \min\big(\max(\lambda, \lambda_{\min}), \lambda_{\max}\big)$
    Update effective sample size: $w = \lambda w + 1$.
    Update parameter estimate: $\tilde{\theta} = \Big(1 - \frac{1}{w}\Big)\tilde{\theta} + \frac{y_s}{w}$.
    Update auxiliary derivatives: $w' = \lambda w' + w$, and $\tilde{\theta}' = \Big(1 - \frac{1}{w}\Big)\tilde{\theta}' - \frac{w'}{w^2}\Big(y_s - \tilde{\theta}\Big)$
    return $(\lambda, w, w', \tilde{\theta}, \tilde{\theta}')$
end function

The remainder of this chapter explores the behaviour of `AFFB` on simulated Bernoulli data with a particular focus on comparing the two cost functions. See Figure 5.3 for an example of `AFFB` in operation for each cost function. In addition, given stochastic gradient descent introduces the concern of choosing an appropriate learning rate, the simulations investigate the behaviour of the procedure under various learning rate schemes. Finally, the adaptive estimation procedure is generalised for binomial data.

5.2 Stationary Behaviour of `AFFB`

While adaptive procedures are generally developed to handle non-stationarity, it is often important that they operate effectively in stationary environments too. For example in change detection a common objective is to determine when data deviates from a stationary period.

This section demonstrates peculiar properties of the adaptive forgetting framework in the context of Bernoulli data, and so studying stationary environments provides the simplest scenario to reason about this behaviour, where theoretical stochastic optimisation convergence results may apply. Even in the absence of theoretical results, simulation can often provide counterexamples to reveal if estimators do *not* behave desirably.

Maximum likelihood estimators are known to be consistent under relatively weak assumptions in an offline setting (Cox and Hinkley 1979). For data streams, where the data is usually non-stationary, the traditional definition of consistency is less useful as model parameters can be considered as moving targets. However, in the specific case of stationary data streams it is desirable for adaptive estimators to behave as if they were consistent. That is, as the number of data samples observed in a stationary environment increases, the adaptive estimates should get 'closer' to the true parameter values. Intuition may suggest that adaptive estimators do behave in this way, but theoretical results for this do not appear to have been explored in the literature for adaptive forgetting maximum likelihood estimators. However, it is possible to demonstrate a striking phenomenon in the Bernoulli case when using the negative log-likelihood cost. To do so, first consider running simulations over stationary samples of length $n = 2,000$ for four different Bernoulli parameters $\theta_0 \in \{0.5, 0.75, 0.9, 0.99\}$, with both cost functions $\mathcal{J}^{(1)}$ and $\mathcal{J}^{(2)}$ over a range of learning rates. The effect can be more easily observed after removing the individual variation in a single simulation, and so $m = 1000$ simulations are run, where for a particular simulation run j, the adaptive forgetting maximum likelihood estimate for θ_t is denoted $\tilde{\theta}_{t,j}$, for $1 \leq j \leq m$. Then the ensemble average at time t is

$$\overline{\theta}_t^{(1:m)} = \frac{1}{m} \sum_{i=1}^{m} \tilde{\theta}_{t,m}.$$

If the adaptive estimators exhibit consistent-like behaviour, then $\tilde{\theta}_{t,j} \to \theta_0$, and by the law of large numbers, $\overline{\theta}_t^{(1:m)} \to \theta_0$, for $1 \leq j \leq n$, so the ensemble averages should get closer to the true parameter values as more data is observed.

Figure 5.1 shows the behaviour of the ensemble averages for the stationary simulation described above, with Figure 5.2 showing the corresponding ensemble-averaged forgetting factor values. As one may expect, the ensemble-averaged adaptive mean becomes less volatile over time as the learning rate η is decreased. However, initially concentrating on the first row of

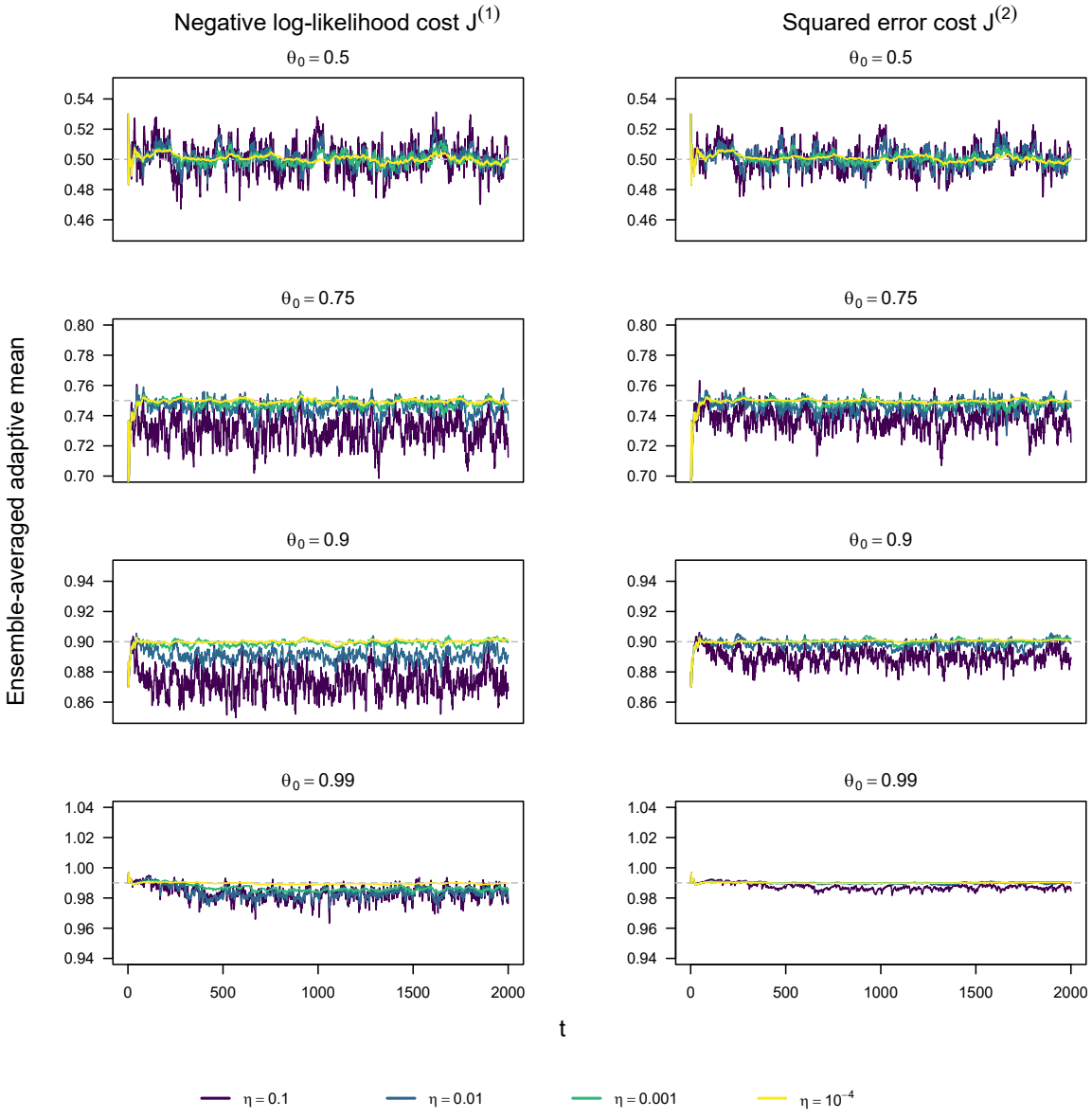

Figure 5.1: Ensemble averages of the adaptive mean estimates over $m = 1000$ stationary Bernoulli data streams for `AFFB`, written $\overline{\theta}_t^{(1:m)}$. Each panel corresponds to a different Bernoulli parameter θ_0, shown by the grey dashed line, and shows the ensemble average for $1 \leq t \leq 2000$ for four different learning rates η, shown in different colours. The left-hand side column of panels all use the negative log-likelihood cost function $\mathcal{J}^{(1)}$ to tune the forgetting factors, with the right-hand side column of panels all using the squared error cost function $\mathcal{J}^{(2)}$.

panels, where $\theta_0 = 0.5$, the ensemble-averaged adaptive estimates appear to oscillate evenly around the true value of θ_0. Whereas in the second and third rows it is evident that this *not* the case, especially for the larger learning rate values of 0.1 and 0.01. It is as if these estimators have some 'apparent bias' that does not appear to decrease over time. Furthermore, the 'apparent bias' in the third row of panels is larger than that of the second row. To quantify this further, consider defining the temporal average of these already ensemble-averaged adaptive mean after some time t^* as

$$\bar{\bar{\theta}}^{(1:m)}_{t>t^*} = \frac{1}{n - t^*} \sum_{t=t^*+1}^{n} \bar{\theta}^{(1:m)}_t.$$

Then the residual $\bar{\bar{\theta}}^{(1:m)}_{t>t^*} - \theta_0$ effectively quantifies the 'apparent bias' above, where t^* should be chosen after some initial learning period, where for $t > t^*$ the sampling distribution of the adaptive mean estimates seems stationary. Conservatively choosing $t^* = 1000$, the values of $\bar{\bar{\theta}}^{(1:m)}_{t>t^*}$ for the simulations above are displayed in Table 5.1, with corresponding 95% confidence interval estimates in Table 5.2. Notice how for $\theta_0 = 0.5$ the temporal averages for all learning rates are extremely close, leading to residuals all less than one thousandth. Particularly interesting behaviour occurs for learning rate values $\eta = 0.1$ and $\eta = 0.01$ where $\theta_0 \neq 0.5$, where it is clearly evident that the estimators are biased, with several confidence interval estimates not containing the true parameter value. While using sufficiently small learning rates to reduce the impact of this issue may seem like a sensible approach, doing so would hinder the ability of the adaptive estimators to accommodate drift. In particular this would most adversely impact abrupt drift scenarios, which benefit from relatively large learning rates, and are later demonstrated in Section 5.3.

Indeed, further experiments show that this bias grows, and the confidence interval estimate shrinks, the closer the Bernoulli parameter is to its bound. This may be due to a lower variance in the underlying data, causing updates in one direction to occur infrequently.

For the simulation setup above, the overall Monte Carlo sample size is mn, with m and n conveniently chosen for graphical display purposes. However, in general, setting $m = 1$ for a sufficiently large n simplifies notation when considering non-graphical results.

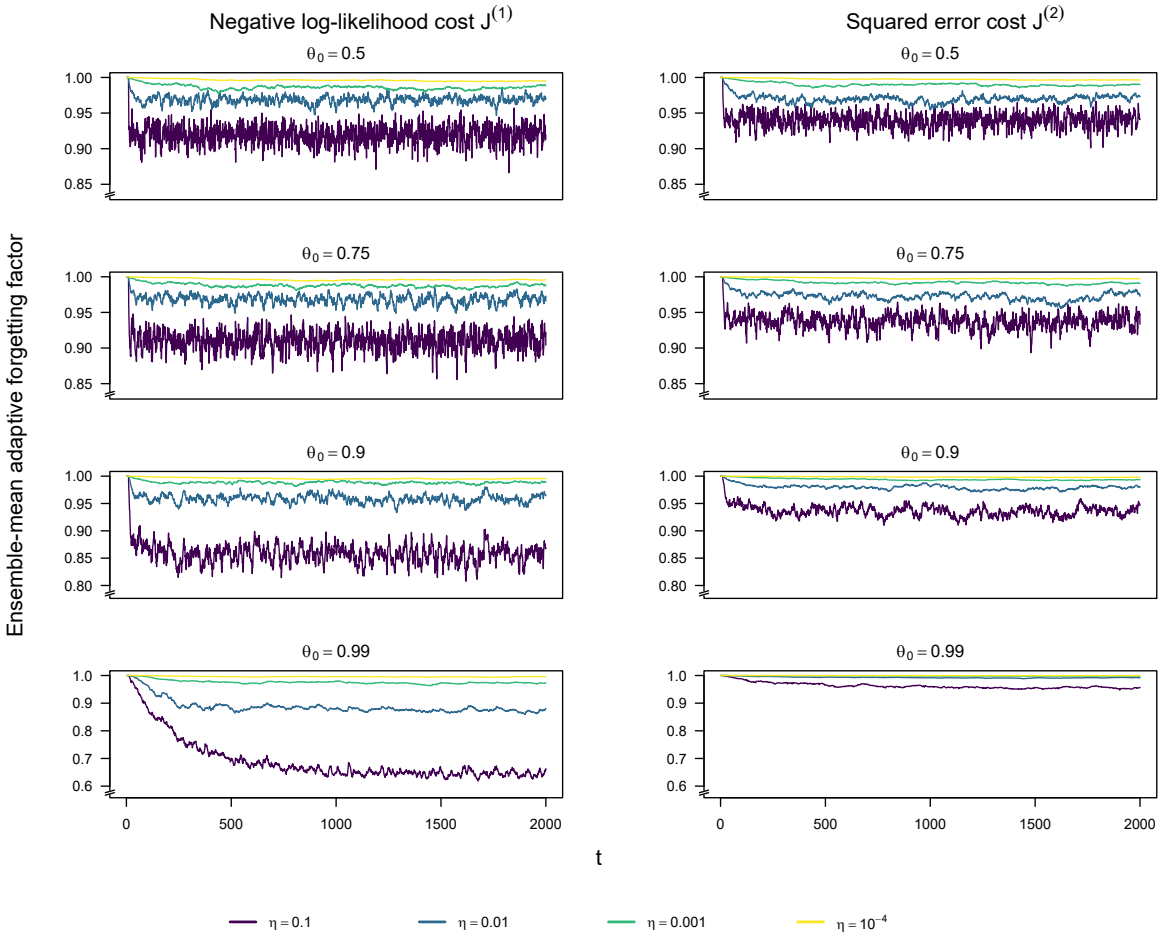

Figure 5.2: Ensemble-mean adaptive forgetting factor values running `AFFB` over $m = 1000$ stationary Bernoulli data streams. Each row of panels corresponds to a different Bernoulli parameter θ_0, and shows the ensemble-mean adaptive forgetting factor $\bar{\lambda}_t$ for $1 \leq t \leq 2000$ for four different learning rates η, shown in different colours. The left-hand side column of panels all use the negative log-likelihood cost function $\mathcal{J}^{(1)}$ to tune the forgetting factors, with the right-hand side column of panels all using the squared error cost function $\mathcal{J}^{(2)}$.

Table 5.1: Temporal-averages of already ensemble-averaged adaptive means computed with two different cost functions $\mathcal{J}^{(1)}$ and $\mathcal{J}^{(2)}$ for stationary Bernoulli simulations with parameter θ_0, using different learning rates η. Values are rounded to four decimal places.

		Cost function							
		Negative log-likelihood $\mathcal{J}^{(1)}$				Squared error $\mathcal{J}^{(2)}$			
		Learning Rate η				Learning Rate η			
		0.1	0.01	0.001	0.0001	0.1	0.01	0.001	0.0001
θ_0	0.5	0.5005	0.5010	0.5001	0.4998	0.5004	0.5009	0.5006	0.5001
	0.75	0.7298	0.7457	0.7482	0.7494	0.7366	0.7469	0.7482	0.7495
	0.9	0.8738	0.8908	0.8992	0.9002	0.8900	0.8985	0.9003	0.9008
	0.99	0.9831	0.9837	0.9857	0.9894	0.9868	0.9898	0.9902	0.9900

Table 5.2: Lower bound (LB) and upper bound (UB) estimates of the 95% confidence interval for the stationary Bernoulli parameter θ_0 based on temporally-averaging already ensemble-averaged adaptive means computed with two different cost functions $\mathcal{J}^{(1)}$ and $\mathcal{J}^{(2)}$, using different learning rates η. Values are rounded to four decimal places.

			Cost function							
			Negative log-likelihood $\mathcal{J}^{(1)}$				Squared error $\mathcal{J}^{(2)}$			
			Learning Rate η				Learning Rate η			
			0.1	0.01	0.001	0.0001	0.1	0.01	0.001	0.0001
θ_0	0.5	LB	0.4813	0.4907	0.4940	0.4963	0.4849	0.4909	0.4957	0.4973
		UB	0.5190	0.5135	0.5077	0.5041	0.5173	0.5128	0.5063	0.5032
	0.75	LB	0.7116	0.7356	0.7434	0.7474	0.7211	0.7384	0.7440	0.7481
		UB	0.7463	0.7537	0.7524	0.7512	0.7500	0.7544	0.7516	0.7517
	0.9	LB	0.8590	0.8828	0.8954	0.8973	0.8800	0.8936	0.8968	0.8997
		UB	0.8902	0.8993	0.9025	0.9019	0.9008	0.9037	0.9027	0.9015
	0.99	LB	0.9732	0.9784	0.9835	0.9887	0.9838	0.9885	0.9898	0.9897
		UB	0.9908	0.9878	0.9876	0.9900	0.9893	0.9907	0.9907	0.9903

In particular, the two cost functions can be compared directly across a set of simulations through a summary performance measure such as the root mean squared error, expressed as

$$\mathrm{RMSE}\big(\tilde{\theta}_{1:n}\big) = \left(\frac{1}{n}\sum_{t=1}^{n}\big(\tilde{\theta}_t - \theta_0\big)^2\right)^{\frac{1}{2}}.$$

Choosing $n = 10^5$, the first two rows of Table 5.3 show the RMSE for simulations with $\theta_0 = 0.5$ and $\theta_0 = 0.99$ for learning rates in decreasing order of magnitude from $\eta = 10^{-1}$ to $\eta = 10^{-4}$. As expected, decreasing the learning rate always decreases the RMSE for stationary simulations

because there is no change in distribution to learn. Interestingly, the squared error cost $\mathcal{J}^{(2)}$ always achieves a lower RMSE than the negative log-likelihood cost $\mathcal{J}^{(1)}$ in this case. However, this is not always the case for non-stationary simulations, as depicted in the subsequent rows in the table. This is explored further in the next section.

5.3 Non-stationary Behaviour of AFFB for Non-stationary Data

This section is a limited study to explore and demonstrate scientific aspects of the adaptive estimation procedure for Bernoulli data streams under different data and drift scenarios. As is common in the literature (Haykin 2002), two types of drift are covered; smooth and abrupt. Smooth drift occurs where the parameter evolution is a continuously differential (often smooth) function, and abrupt drift occurs where the parameter evolution is modelled as a generally non-continuous piecewise function. In addition to the type of drift, the *rate* of drift is an important property to explore. For smooth drift this is governed by the second derivative, or acceleration, of the parameter evolution. Damped random walks and sinusoidal functions with varying periodicity are commonly used to represent smooth drift. The latter approach is favoured here to encourage repeated patterns as suggested in Widmer and Kubat (1996). For abrupt drift this is usually controlled by changing the lengths of the continuous sub-intervals for the piecewise function. Examples using these approaches for both abrupt and smooth cases can be found in Anagnostopoulos (2010), Hammer and Yazidi (2017), Hammer, Yazidi, and Rue (2019), and Hammer, Yazidi, and Rue (2021).

This study considers these scenarios where Bernoulli data $(y_i)_{i=1}^n$ is then simulated under different definitions of the true parameter values $(\theta_i)_{i=1}^n$. The scenarios are stated formally below, with the stationary setup from the previous section included for completeness.

- **Stationary.** Set $\theta_i = \theta_0 \in (0, 1)$ for $i = 1, \ldots, n$
- **Smooth drift.** Set $\theta_i = \alpha \sin^2\left(\frac{1}{\tau} 2\pi i\right)$ for $i = 1, \ldots, n$ where $\alpha \in [0, 1]$ controls the range of θ, and $\tau \in N$ with $0 < \tau \leq n$ controls the period, and hence the rate of drift, where smaller values of τ increase the rate of drift.

- **Abrupt drift.** Set

$$\theta_i = \begin{cases} \theta^{(1)} & \text{for } i \text{ such that } (i \bmod B) < \frac{B}{2}, \\ \theta^{(2)} & \text{for } i \text{ such that } (i \bmod B) >= \frac{B}{2}, \end{cases}$$

  where B controls the rate of drift through the regime length, and $\theta^{(k)} \in [0, 1]$ for $k = 1, 2$ controls a regime-dependent Bernoulli parameter.

Two rates of drift (fast and slow) are considered for each type of drift. For smooth drift, smooth (slow) corresponds to $\tau = 10000$ and smooth (fast) corresponds to $\tau = 1000$. For abrupt drift, abrupt (slow) corresponds to $B = 10000$ and abrupt (fast) corresponds to $B = 1000$. For each drift type, two sets of simulation parameters are explored. Specifically, for the smooth case both $(\alpha, \beta) = (0.98, 0.01)$ and $(\alpha, \beta) = (0.5, 0.25)$ are tested. The former is designed to explore more extreme values than the latter. For the abrupt case, both $(\theta^{(1)} = 0.95, \theta^{(2)} = 0.05)$ and $(\theta^{(1)} = 0.75, \theta^{(2)} = 0.25)$ are tested. Again, the former is designed to cover more extreme values, as well as to experience larger parameter changes.

Figure 5.3 shows examples of individual runs of the `AFFB` procedure, with $\eta = 0.01$, when run on simulated data ($n = 2000$) from the fast non-stationary scenarios. In these instances, it is clear the adaptive estimators can track the evolving parameter, though it seems as if the estimator using the squared error cost experiences more volatility post abrupt change.

Unless otherwise specified, for all simulations in the next section the number of observations $n = 10^5$, the truncation range for the forgetting factors $[\lambda_{\min}, \lambda_{\max}] \equiv [0.6, 1]$ as discussed in Section 2.6.2, and the default constant learning rate values used are $\eta \in \{10^{-k}\}_{k=1}^{4}$.

5.3.1 Simulation Assessment

Table 5.3 shows the RMSE for each simulation scenario and learning rate where the cell with the lowest RMSE is coloured in grey. It seems surprising that the lowest learning rate value $\eta = 10^{-4}$ gives the best performance for both the fast and slow drift environments. However, this may arise from the fact that even with a high drift rate, smooth drift tends to restrict the parameter evolution more than abrupt drift. For the abrupt drift environments, the lowest RMSE results are obtained from learning rates values of $\eta = 0.01$ and $\eta = 0.001$. In particular, for the NLL cost, $\eta = 0.001$ leads to the lowest RMSE for the slow abrupt drift

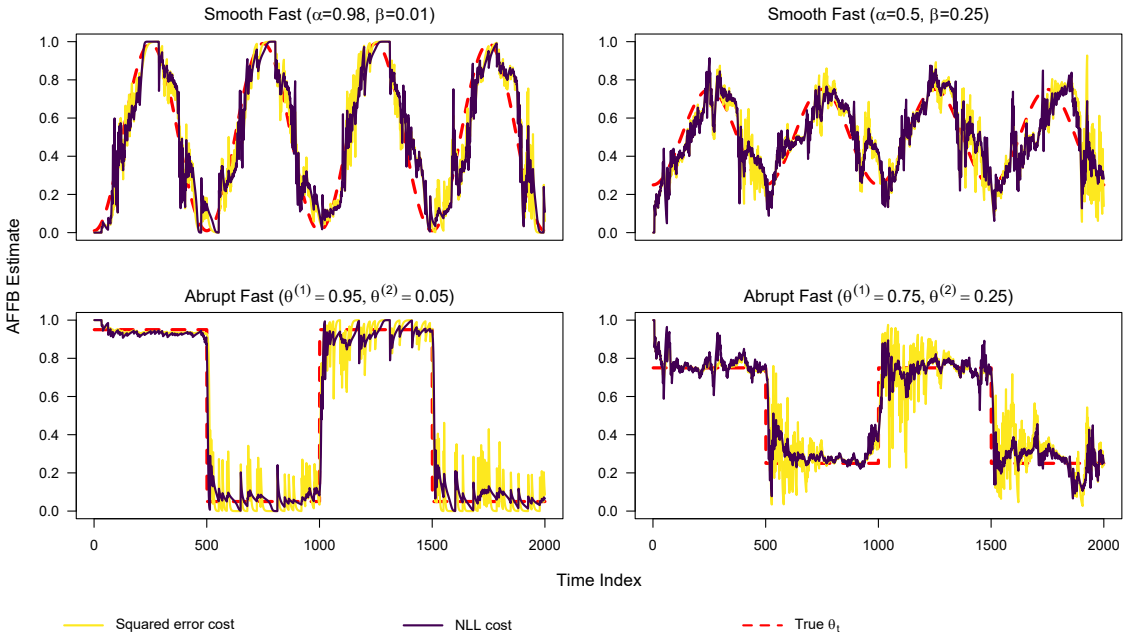

Figure 5.3: Adaptive estimates for the Bernoulli parameter for an individual run of `AFFB` for each cost function on simulated data from a selection of non-stationary environments.

and $\eta = 0.01$ leads to the lowest RMSE for fast abrupt drift case. This is to be expected since the learning rate corresponds to the rate of drift the procedure can adapt to. However, overall the RMSE results suggest that across all simulation environments, the procedure is not particularly sensitive to the choice of learning rate, for choices of $\eta < 0.1$. Using the learning rate $\eta = 0.1$ is not advisable, even in fast drift environments because this large a step-size induces too much volatility in the estimators. The RMSE for the NLL cost is larger than the RMSE for the squared error cost 75% of the time, with the mean RMSE for the NLL cost being 9.7% larger than that of the squared error cost. This seems counter-intuitive given the squared error is symmetric and does not respect the bounded parameter space $\theta_t \in [0, 1]$, whereas the NLL cost should capture the properties of the Bernoulli distribution. Regardless, if the objective is to have as accurate as possible point estimates, or the underlying Bernoulli parameter is an extreme quantile then the NLL may not be a sensible choice based on the bias observed in the previous section for `AFFB`. However, subsequent sections in this chapter present alternative procedures that exhibit less bias when using the NLL cost. Still, when using `AFFB`, these results suggest using the squared error cost. The learning rate should be chosen within $[10^{-2}, 10^{-4}]$ to match the anticipated rate of drift, but it is important to emphasise that these results show that the choosing the optimal learning rate is not crucial.

Table 5.3: Root mean squared error results for `AFFB` for a selection of stationary and non-stationary simulation environments.

		Cost function							
		Negative log-likelihood $\mathcal{J}^{(1)}$				Squared error $\mathcal{J}^{(2)}$			
Simulation		Learning Rate η				Learning Rate η			
Type	Parameters	0.1	0.01	0.001	10^{-4}	0.1	0.01	0.001	10^{-4}
Static	$\theta_0 = 0.5$	0.110	0.064	0.042	0.024	0.092	0.064	0.036	0.021
	$\theta_0 = 0.99$	0.049	0.030	0.013	0.005	0.017	0.007	0.004	0.002
Smooth (slow)	$\alpha = 0.98, \beta = 0.01$	0.092	0.059	0.057	0.047	0.069	0.061	0.051	0.043
	$\alpha = 0.5, \beta = 0.25$	0.106	0.063	0.057	0.050	0.088	0.063	0.055	0.045
Smooth (fast)	$\alpha = 0.98, \beta = 0.01$	0.108	0.110	0.103	0.093	0.111	0.103	0.090	0.090
	$\alpha = 0.5, \beta = 0.25$	0.113	0.102	0.099	0.086	0.106	0.114	0.090	0.086
Abrupt (slow)	$\theta^{(1)} = 0.95, \theta^{(2)} = 0.05$	0.085	0.050	0.049	0.081	0.049	0.056	0.073	0.064
	$\theta^{(1)} = 0.75, \theta^{(2)} = 0.25$	0.103	0.059	0.058	0.086	0.084	0.062	0.090	0.083
Abrupt (fast)	$\theta^{(1)} = 0.95, \theta^{(2)} = 0.05$	0.099	0.099	0.148	0.120	0.098	0.100	0.092	0.092
	$\theta^{(1)} = 0.75, \theta^{(2)} = 0.25$	0.112	0.093	0.128	0.102	0.102	0.129	0.109	0.100

Though not discussed in this thesis, the choice of cost function used in adaptive estimation procedures for Poisson data is thoroughly explored in Anagnostopoulos (2010). Surprisingly in the Poisson case, it is shown that the negative log-likelihood cost considerably outperforms the squared-error cost instead. The author remarks that this result is seemingly counter-intuitive, given the performance measure used is square-error based, and suggests this is likely due to the asymmetric nature of the Poisson distribution. Despite the Bernoulli distribution also being skewed and discrete, the results differ. It is possible the unbounded range of the Poisson distribution is also a contributing factor here, though this is not explored further.

5.3.2 Scaled learning rates

This section explores the use of non-constant, *scaled*, learning rates motivated by the result in Bodenham (2014) where it is shown that for general i.i.d. data with mean μ, variance σ^2, and squared error cost

$$\mathbb{E}\left[\frac{\partial}{\partial\lambda}\mathcal{J}_{t+1}^{(2)}\right] \sim O(\sigma^2).$$

In the case of i.i.d. Bernoulli data with parameter θ_0, the variance would be $\theta_0(1-\theta_0) \in [0, 0.25)$. As the range of the Bernoulli parameter θ_0 is bounded, it is possible to exhaustively compute

sampling distributions for the gradient of the squared-error cost over a grid of θ_0 values. Consider running 99 separate stationary simulations for $\theta_0 \in \{\frac{i}{100}\}_{i=1}^{9}$, with $n = 10^5$ for the usual learning rates. Figure 5.4 shows the mean and standard deviation of the cost gradients as grey points; essentially forming parabolas centred at $\theta_0 = 0.5$. It should not be surprising that the standard deviation is highest at $\theta_0 = 0.5$, as is the case for the underlying data, since the procedure is operating on stationary data. Similarly, a non-constant mean may seem somewhat surprising, but recall we have observed this behaviour in Figure 5.2 where the mean adaptive forgetting factor values fluctuate less, the closer θ_0 is to 0 or 1. To account for this, the learning rates were scaled by the reciprocal of the variance such that the new learning rates $\nu_t = \frac{\eta}{\theta_t(1-\theta_t)}$. Using the true value of θ_t to scale the learning rate is only possible in simulation, but both the true values and the current estimate $\tilde{\theta}_t$ produced similar results in experiments, so only the former is considered. Returning to Figure 5.4, the points in black correspond to simulations with a scaled learning rate of $\frac{1}{4}\nu_t$. Notice how this almost completely flattens the previously observed parabolas, struggling most at the endpoints whenever θ_t is close to 0 or 1. This is expected for the mean given the results of Bodenham (2014), but not necessarily for the standard deviation. Here the learning rate was also scaled by 0.25 for a convenient graphical contrast between the grey and black points, but in general this constant can be used to control the roughly constant value of both summary statistics.

At the end of the previous section it was remarked that the cost functions exhibit different behaviour for Poisson and Bernoulli procedures. Note that for the Bernoulli case, the summary statistics for the cost gradients are symmetric around $\theta_0 = 0.5$. However, no such symmetry exists for the Poisson case, which is likely also a factor in the poor performance of the squared-error cost function in the Poisson instance.

Similarly to the assessment in Section 5.3.1, RMSE results were obtained for scaled learning rates with different values of the constant scale factor taking values in $\{0.25, 1, 4\}$, but the RMSE for *every* non-stationary scenario is larger than for the unscaled version, though in many cases not substantially larger. It is not clear exactly why this is the case, but presumably the procedure benefits from being able to adapt at different rates, depending on the current value of θ_t. Based on this observation, scaled learning rates are **not** recommended.

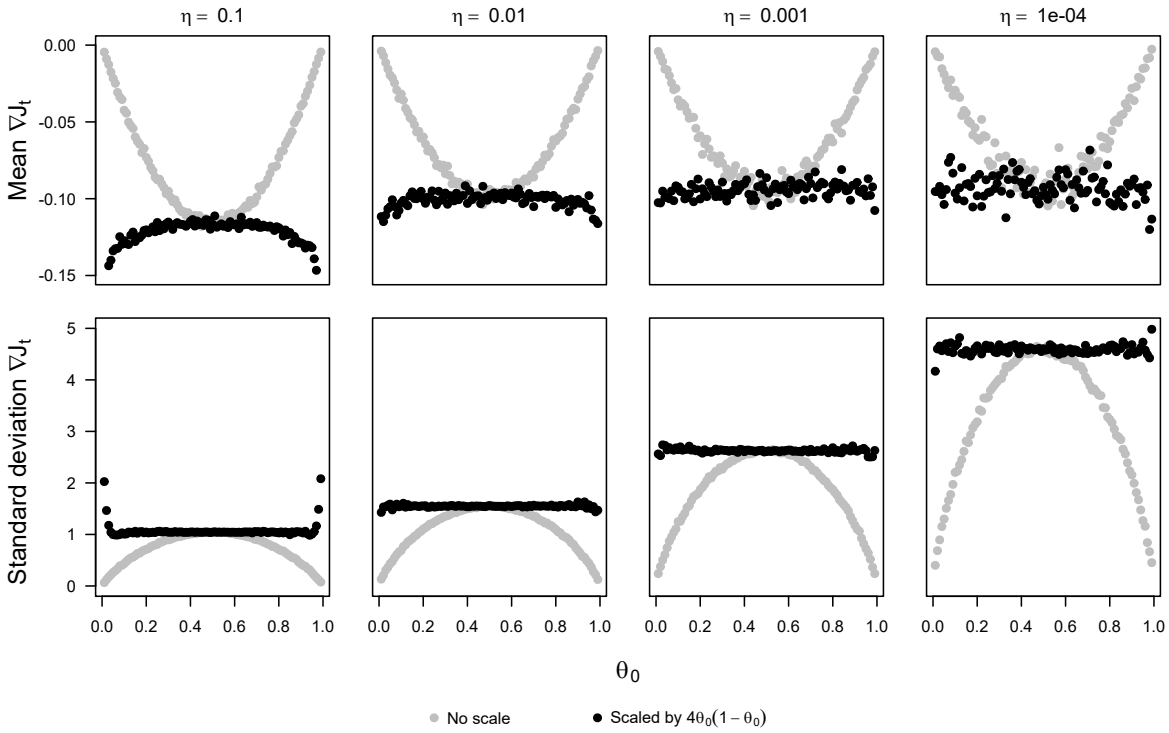

Figure 5.4: Mean (top-row) and standard deviation (bottom-row) of the gradient of the squared error cost function for $n = 100,000$ stationary Bernoulli data points. Each column of panels corresponds to a different learning rate, with the points in each panel corresponding to Bernoulli parameter values $\theta_0 \in \{0.01, 0.02, \ldots, 0.99\}$.

5.4 Truncation Bias

Recall from Section 2.6.2 that when tuning the forgetting factor λ_t, it is customary to truncate the range of the forgetting factor such that $\lambda_t \in [\lambda_{\min}, \lambda_{\max}]$ after performing the SGD update step. The truncation at the lower end is primarily in place to prevent both a delayed recovery to drift and potential numerical instability in matrix parameter estimates. The lower limit of the truncation range governs the rate of forgetting for past data and is commonly set to $\lambda_{\min} \in [0.6, 0.8]$ (Anagnostopoulos 2010; Bodenham 2014). Within this range, adaptive estimators do not appear to be particularly sensitive to the choice of $\lambda_{\min}$ for different distributions and drift dynamics. Furthermore, without prior knowledge of the future drift dynamics it is difficult to choose $\lambda_{\min}$, hence this work does not seek to optimise $\lambda_{\min}$. Instead, this section directs attention to $\lambda_{\max}$, which has avoided close study in all prior work, where it is set to 1. Here, a novel truncation step is introduced to implicitly allow values of the forgetting factor to exceed 1 in the tuning step. On the surface it may seem almost nonsensical to consider $\lambda_{\max} > 1$, especially given the close relationship between forgetting factors and the weight parameter for

updating EWMA charts (see Section 3.2.5 in Bodenham (2014)). However, the motivation arises from noticing an inherent bias during this truncation step. Recall during the SGD update step, the forgetting factors are updated according to

$$\lambda_{t+1} = \lambda_t - \eta_t \nabla \mathcal{J}_t,$$

for some cost $\mathcal{J}_t$. Clearly, the forgetting factors increase when the cost gradient is negative, and the forgetting factors decrease when the cost gradient is positive. However, the forgetting factors are also restricted to the bounds of the truncation range. The results in Table 5.1 show that for stationary binary data, the temporal mean cost gradient is always negative, which is to be expected since on average the procedure should not be forgetting. However, it is perhaps more interesting to look at how often the forgetting factors increase and decrease, which depends on $\mathbb{P}(\nabla \mathcal{J}_t > 0)$. This probability is not available in closed form due to the distribution of the cost gradient being unknown in general, but corresponding empirical frequencies can be studied instead. Formally, for $k \in \{1, 2\}$ these empirical frequencies can be written as

$$\bar{\mathcal{J}}^{(k)}_{n,+} = \sum_{i=1}^{n} \mathbb{1}\Big\{\mathcal{J}^{(k)}_i > 0\Big\},$$
$$\bar{\mathcal{J}}^{(k)}_{n,-} = \sum_{i=1}^{n} \mathbb{1}\Big\{\mathcal{J}^{(k)}_i \leq 0\Big\}.$$

Consider running a simple experiment with the `AFFB` procedure for a stationary simulation setup with $n = 10^5, \theta_0 = 0.5, \eta = 0.01$ with both cost functions $\mathcal{J}^{(1)}$ and $\mathcal{J}^{(2)}$. The results are shown in Table 5.4 which reveal an almost perfect one-to-one ratio between positive and negative values. Recall from Section 2.6.2 that setting all forgetting factors fixed to 1 ensures that adaptive estimators agree with the traditional offline MLEs. However, these empirical results suggest that even when the forgetting factors are very close to 1, they are going to decrease on average once every two observations. This is concerning because for $\lambda_{\max} = 1$, the forgetting factors *cannot* increase above 1, which would similarly be expected to happen roughly once every two observations. As a result, the forgetting factors are restricted from oscillating around 1 from above. Furthermore, given the mean cost gradient is negative (see top-row of Figure 5.4), if the truncation did not occur then these forgetting factors would be increasing on average. Simply put, this truncation introduces a bias in the forgetting

factor update step that prevents adaptive estimation procedures from performing effectively in stationary environments. This truncation bias would not be present if the truncation step

Table 5.4: Empirical frequencies of positive and negative cost gradients for both the negative log-likelihood and squared error cost functions when running the `AFFB` procedure on stationary Bernoulli data.

	Cost function	
Sign	Negative log-likelihood $\mathcal{J}^{(1)}$	Squared error $\mathcal{J}^{(2)}$
$+$	0.50214	0.50125
$-$	0.49785	0.49874

were removed or relaxed, but unfortunately the sequential parameter updates degrade for $\lambda_t > 1$, and so allowing values of $\lambda_t > 1$ for the full procedure is not viable. One approach to address this, somewhat ironically, is to introduce an additional truncation step over a relaxed forgetting factor λ_t^*, such that the SGD update step becomes

$$\lambda_{t+1}^* = \lambda_t^* - \eta_t \nabla \mathcal{J}_{t+1},$$

with λ_{t+1}^* truncated to the range $[\lambda_{\min}^*, \lambda_{\max}^*]$, where $\lambda_{\max}^* > 1$. Then as before, truncate *again* to determine λ_t via

$$\lambda_t = \min(\max(\lambda_t^*, \lambda_{\min}), 1),$$

where λ_t is used for all sequential parameter updates and λ_t^* is *only* used for the SGD update step. Here, choose $\lambda_{\min}^* = \lambda_{\min}$ for convenience since truncation at the lower bound is not relevant. The procedure when run with this two-step truncation will henceforth be referred to as `AFFB*`, and is presented in Algorithm 3.

Figure 5.5 shows trace plots for the forgetting factor values when running this procedure on the same stationary Bernoulli data simulation above with $\lambda_{\max}^* = 2$. The values of λ_t^* used in the SGD update step are shown in red, which when truncated give the values of λ_t shown in black. This is shown in contrast to the grey line which corresponds to the forgetting factor values for the usual `AFFB` procedure *without* the additional truncation step. Notice how infrequently the black line drops below 1 when compared to the grey line, demonstrating the desired behaviour of the relaxed truncation approach.

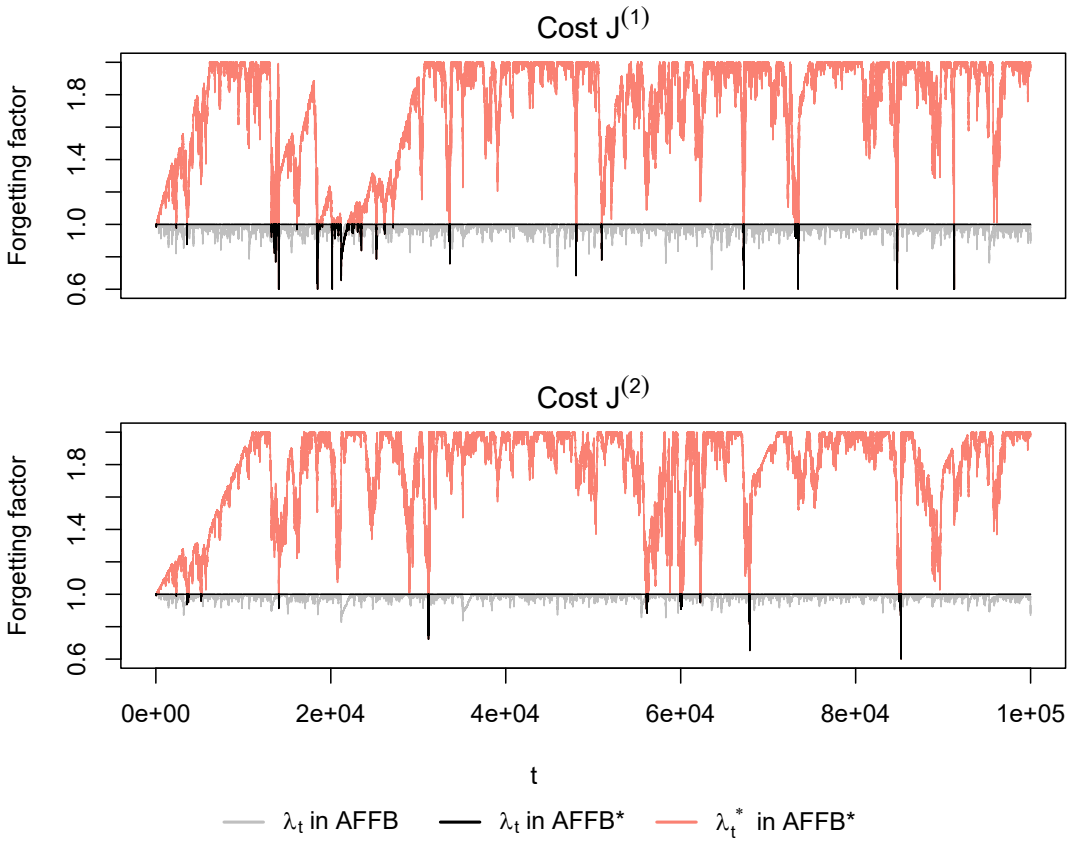

Figure 5.5: Forgetting factor trace plots for stationary Bernoulli data using the `AFFB` and `AFFB*` procedures.

Algorithm 3 `AFFB*`: a general adaptive forgetting estimation procedure for Bernoulli data streams using an additional truncation step.

Input: Stream of realisations $y_s \in \{0, 1\}$, for $s = 1, 2, \ldots$.
Let $\mathcal{J}(y, \theta)$ be an appropriate cost function for any $y \in \{0, 1\}, \tilde{\theta} \in [0, 1]$. Let the step-size η_s be a deterministic function of s and w. Let $[\lambda_{\min}, \lambda_{\max}]$ be the truncation range of the forgetting factors, and $[\lambda^*_{\min}, \lambda^*_{\max}]$ be the range of the relaxed forgetting factors.
Initialise $(\lambda^*, w, w', \tilde{\theta}, \tilde{\theta}') = (1, 0, 0, 0, 0)$.
for $s = 1, 2 \ldots$ **do**
    Set $(\lambda^*, w, w', \tilde{\theta}, \tilde{\theta}') = \text{AFFB}^*\text{\_UPDATE}(y_s, \lambda^*, w, w', \tilde{\theta}, \tilde{\theta}')$
function $\text{AFFB}^*\text{\_UPDATE}(y_s, \lambda^*, w, w', \tilde{\theta}, \tilde{\theta}')$
    Compute cost gradient: $g = \mathcal{J}'(y, \theta)\Big|_{y=y_s, \theta=\tilde{\theta}}$
    e.g. for the squared-error cost $g = 2\tilde{\theta}'\Big(y_s - \tilde{\theta}\Big)$.
    Update relaxed forgetting factor: $\lambda^* = \lambda^* - \eta_s g$.
    Truncate relaxed forgetting factor: $\lambda^* = \min\big(\max(\lambda^*, \lambda^*_{\min}), \lambda^*_{\max}\big)$
    Additional truncation to compute forgetting factor: $\lambda = \min\big(\max(\lambda^*, \lambda_{\min}), \lambda_{\max}\big)$.
    Update effective sample size: $w = \lambda w + 1$.
    Update parameter estimate: $\tilde{\theta} = \Big(1 - \frac{1}{w}\Big)\tilde{\theta} + \frac{y_s}{w}$.
    Update auxiliary derivatives: $w' = \lambda w' + w$, and $\tilde{\theta}' = \Big(1 - \frac{1}{w}\Big)\tilde{\theta}' - \frac{w'}{w^2}\Big(y_s - \tilde{\theta}\Big)$
    return $(\lambda^*, w, w', \tilde{\theta}, \tilde{\theta}')$
end function

The truncation bias pointed out in `AFFB` is not limited to Bernoulli adaptive estimation procedures, and can be observed in general for many previously considered adaptive estimation procedures for other types of data. The same relaxed truncation approach improves performance in stationary periods for these procedures too, but showing this is not relevant here. Care must be taken not to choose a value of $\lambda^*_{\max}$ too large. Doing so, restricts the rate at which the procedure can forget past information and adapt accordingly. In all simulation experiments for Bernoulli data, choosing $\lambda^*_{\max} = 2$ worked well, which is used for all future `AFFB*` simulations. However, in general it should be set relative to the expected magnitude of the cost gradients, similar to the scaling argument in Bodenham (2014). One approach that also proved effective during experimentation is to set this bound on-the-fly to be $1 + \delta_t$, where δ_t is some chosen scalar multiple of the standard deviation of the sampling distribution of cost gradients, which can be tracked simply, e.g. via EWMA.

For the same simulation setup described in Section 5.2, the adaptive forgetting factor behaviour when using the `AFFB*` scheme is shown in Figure 5.6. Notice how the ensemble mean adaptive forgetting factor paths for `AFFB*` are much closer to 1 than in Figure 5.2 for `AFFB`,

which is desirable behaviour in the stationary case. This effect is more pronounced for larger learning rates and the closer the Bernoulli parameter values θ_0 is to 0.5. Additionally, the effect is even more pronounced when using the NLL cost than for the squared error cost. Interestingly, as a consequence of improved forgetting factor behaviour, the 'apparent bias' observed and discussed in Section 5.2 is less pronounced when using `AFFB*`. This can be seen in Figure 5.7 where in each panel the ensemble adaptive mean paths oscillate much closer around the true value of θ_0, relative to the paths in Figure 5.1 for `AFFB`. Regardless of cost function, it is still particularly apparent whenever $\eta = 0.1$, but otherwise appears to be almost zero in the case of other learning rates where $\eta < 0.1$. Additionally, the volatility in the ensemble paths has drastically reduced to roughly the same extent for both cost functions. The NLL cost function benefits most from a reduction in bias, but this may only be because it originally exhibited the most bias in the `AFFB` case. Running the `AFFB*` procedure for the simulation scenarios in Section 5.3.1 produces the RMSE results shown in Table 5.5. Cells are coloured in green where the RMSE for `AFFB*` is less than or equal to the corresponding RMSE for `AFFB`, with cells being coloured in red otherwise. The cell entries with bold text correspond to the lowest RMSE across a row, which represents the lowest RMSE for a specific scenario across all tested learning rates. While the colours only give a binary indication of relative performance, in order to quantify the relative performance it is also useful to consider the *relative RMSE* obtained by dividing the `AFFB*` RMSE by the `AFFB` RMSE. The mean relative RMSE across all scenarios is 0.914, but of course that does not necessarily mean `AFFB*` is always better. For just the stationary scenarios, using `AFFB*` always produces a lower RMSE, with a mean relative RMSE of 0.55, which suggests `AFFB*` does perform substantially better for stationary environments as intended. Interestingly the mean relative RMSE across stationary scenarios for the negative log-likelihood cost is 0.49, whereas it is somewhat surprisingly larger at 0.61 for the squared-error cost. This suggests the `AFFB*` procedure substantially improves performance when using the negative log-likelihood cost function, even more-so than for the squared-error cost, however the best results are still achieved when using a squared-error cost. For the smooth drift cases, the RMSE results between the two procedures are very close, with a mean relative RMSE of 0.98. This may be the most surprising result, since in the smooth drift scenario there are no stationary periods, though regions near sinusoidal peaks and troughs could be considered locally stationary. For the abrupt drift scenarios, the data is piecewise

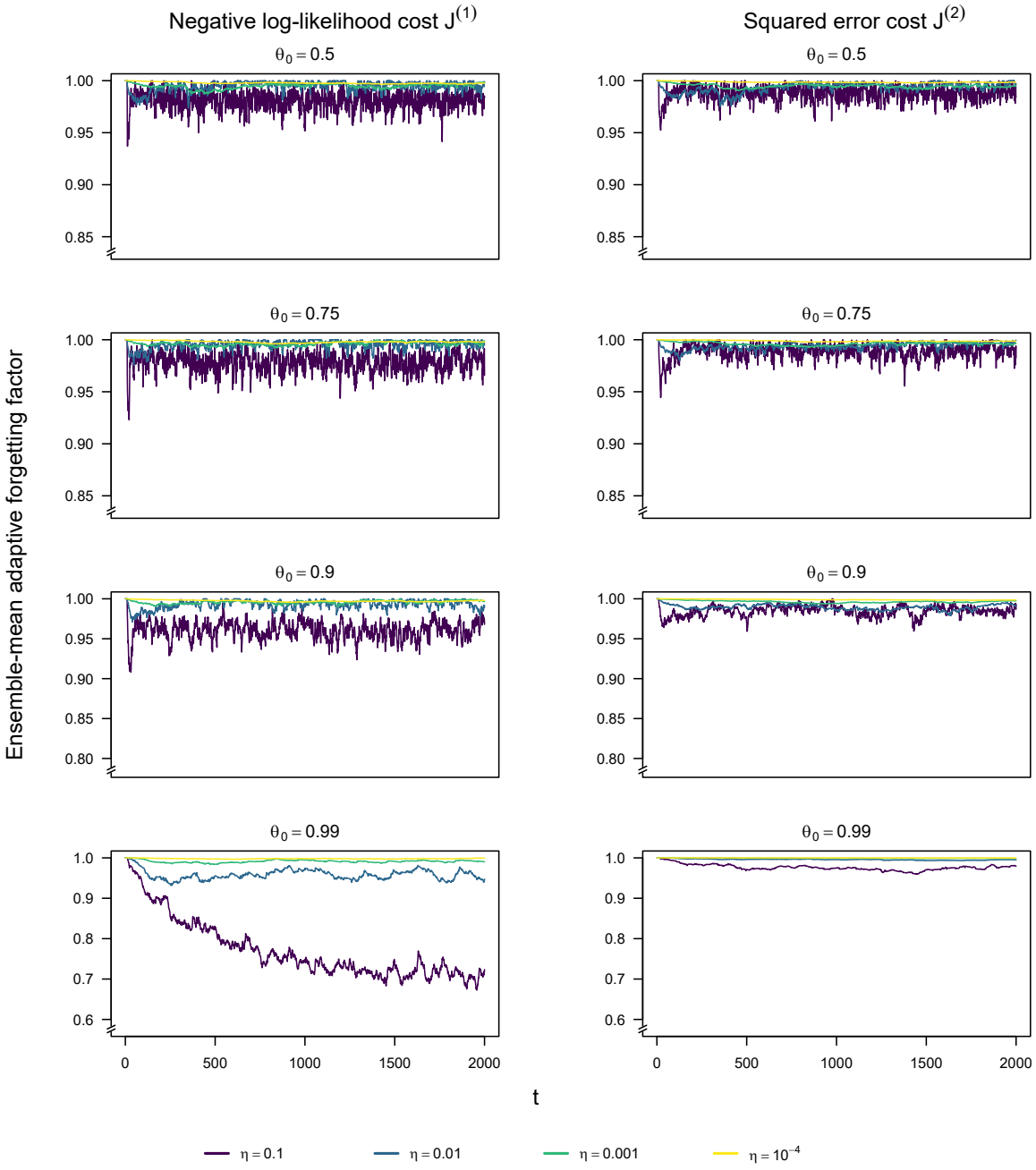

Figure 5.6: Ensemble-mean adaptive forgetting factor values running `AFFB*` over $m = 1000$ stationary Bernoulli data streams. Each panel corresponds to a different Bernoulli parameter θ_0, and shows the ensemble-mean adaptive forgetting factor $\bar{\lambda}_t$ for $1 \leq t \leq 2000$ for four different learning rates η, shown in different colours. The left-hand side column of panels all use the negative log-likelihood cost function $\mathcal{J}^{(1)}$ to tune the forgetting factors, with the right-hand side column of panels all using the squared error cost function $\mathcal{J}^{(2)}$.

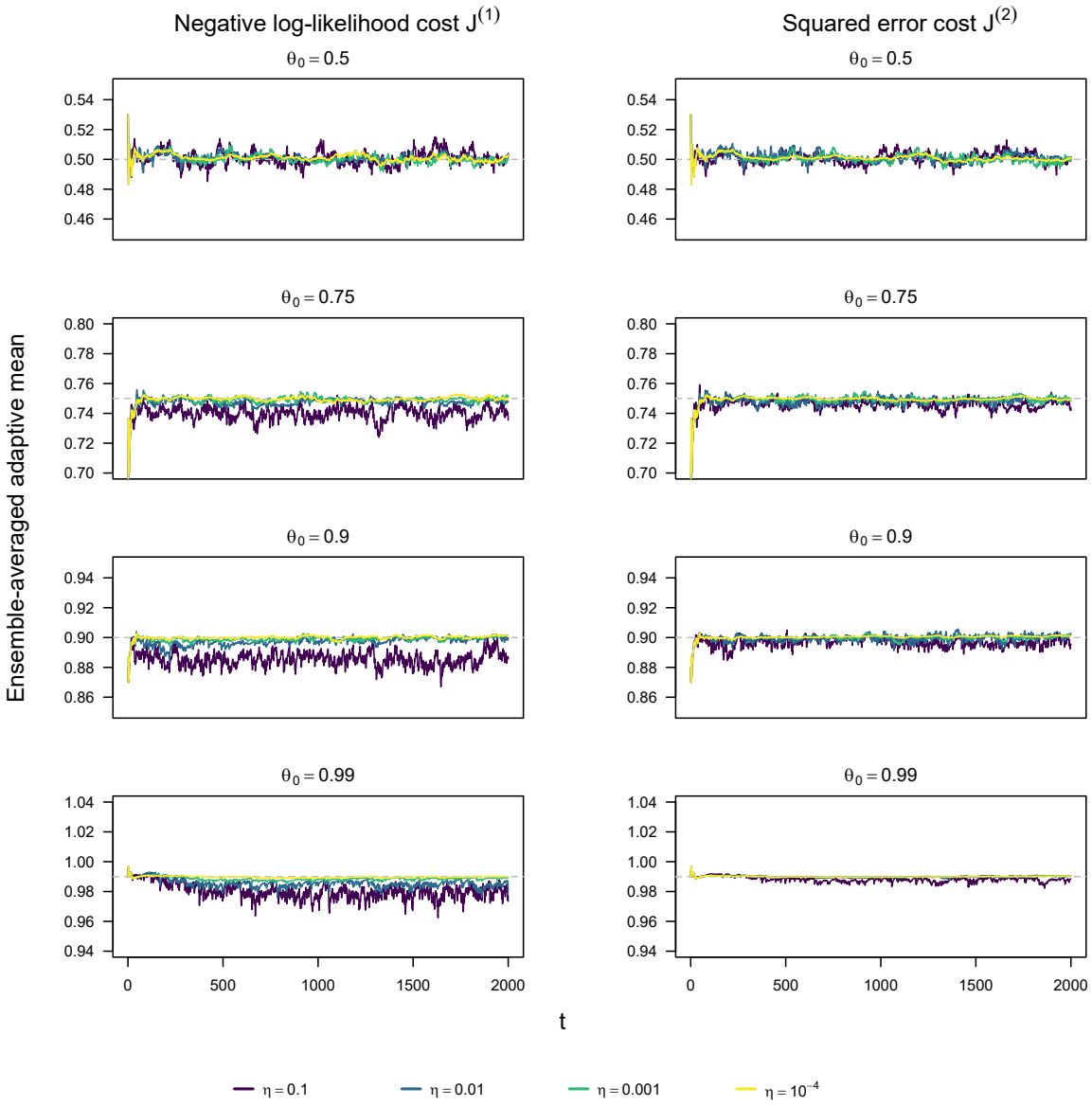

Figure 5.7: Ensemble averages of the adaptive mean estimates over $m = 1000$ stationary Bernoulli data streams for `AFFB*`, written $\overline{\theta}_t^{(1:m)}$. Each panel corresponds to a different Bernoulli parameter θ_0, shown by the grey dashed line, and shows the ensemble average for $1 \leq t \leq 2000$ for four different learning rates η, shown in different colours. The left-hand side column of panels all use the negative log-likelihood cost function $\mathcal{J}^{(1)}$ to tune the forgetting factors, with the right-hand side column of panels all using the squared error cost function $\mathcal{J}^{(2)}$.

stationary, so it is not unreasonable to expect `AFFB*` to perform better, however, that is not evident from the results. For slow abrupt scenarios the mean relative RMSE is 0.98, but for the fast abrupt scenarios the mean relative RMSE is 1.07, which is likely a consequence of `AFFB*` being marginally slower to adapt to the faster abrupt drift. Still, the difference is not substantial enough to dismiss either approach. It is evident that `AFFB*` dramatically improves RMSE when choosing a learning rate of $\eta = 0.1$. For the negative log-likelihood cost, the mean quotient across non-stationary scenarios is 0.80, compared to 0.93 for the squared-error cost. Again, this suggests the negative log-likelihood cost benefits most from the additional truncation step, but moreover, can even be considered to mitigate the risks of choosing too high a learning rate. Indeed, the RMSE results for `AFFB*` do show less sensitivity to the choice of learning rate, and hence all things considered, the use of `AFFB*` over `AFFB` is recommended for most situations where the data is expected to go through stationary periods from time to time.

Table 5.5: Root mean squared error results for `AFFB*` for a selection of stationary and non-stationary simulation environments. Cells are coloured in green where the RMSE for `AFFB*` is less than or equal to the corresponding RMSE for `AFFB`, with cells being coloured in red otherwise.

		Cost function							
		Negative log-likelihood $\mathcal{J}^{(1)}$				Squared error $\mathcal{J}^{(2)}$			
Simulation		Learning Rate η				Learning Rate η			
Type	Parameters	0.1	0.01	0.001	10^{-4}	0.1	0.01	0.001	10^{-4}
Static	$\theta_0 = 0.5$	0.110	0.064	0.042	0.024	0.092	0.064	0.036	**0.021**
	$\theta_0 = 0.99$	0.049	0.030	0.013	0.005	0.017	0.007	0.004	**0.002**
Smooth (slow)	$\alpha = 0.98, \beta = 0.01$	0.092	0.059	0.057	0.047	0.069	0.061	0.051	**0.043**
	$\alpha = 0.5, \beta = 0.25$	0.106	0.063	0.057	0.050	0.088	0.063	0.055	**0.045**
Smooth (fast)	$\alpha = 0.98, \beta = 0.01$	0.108	0.110	0.103	0.093	0.111	0.103	0.090	**0.090**
	$\alpha = 0.5, \beta = 0.25$	0.113	0.102	0.099	0.086	0.106	0.114	0.090	**0.086**
Abrupt (slow)	$\theta_0 = 0.95$	0.085	0.050	**0.049**	0.081	0.049	0.056	0.073	0.064
	$\theta_0 = 0.75$	0.103	0.059	**0.058**	0.086	0.084	0.062	0.090	0.083
Abrupt (fast)	$\theta_0 = 0.95$	0.099	0.099	0.148	0.120	0.098	0.100	**0.092**	0.092
	$\theta_0 = 0.75$	0.112	**0.093**	0.128	0.102	0.102	0.129	0.109	0.100

5.5 Generalisation to Binomial Data

It is interesting to consider the properties of adaptive estimators for generalisations of Bernoulli data, specifically data from the binomial distribution.

Let $\{Y_s\}_{s\in\mathbb{N}}$ be a data stream of binomial random variables with M trials and trial success probability θ_s, such that the probability mass function at any time t is given by

$$f_{Y_t}(y) = \binom{M}{y}\theta_t^y(1-\theta_t)^{M-y}.$$

As in Section 5.1, a similar adaptive estimation procedure can be derived by using the adaptive forgetting log-likelihood. Doing so, the adaptive estimates for θ_t can be derived sequentially as

$$\tilde{\theta}_t = \left(1 - \frac{1}{w_t}\right)\tilde{\theta}_{t-1} + \frac{1}{w_t}\frac{y_t}{M}.$$

Similarly, both cost functions used in Section 5.1 generalise intuitively such that the gradients become

$$\mathcal{J}_t^{(1)\prime} = -\tilde{\theta}'_{t-1}\left(\frac{y_t}{\tilde{\theta}_{t-1}} - \frac{M-y_t}{1-\tilde{\theta}_{t-1}}\right),$$

for the negative log-likelihood, and

$$\mathcal{J}_t^{(2)\prime} = 2\tilde{\theta}'_{t-1}\left(\frac{y_t}{M} - \tilde{\theta}_{t-1}\right),$$

for the squared error cost. Naturally, the sequential update for $\tilde{\theta}'_t$ becomes

$$\tilde{\theta}'_t = \left(1 - \frac{1}{w_t}\right)\tilde{\theta}'_{t-1} - \frac{w'_t}{w_t^2}\left(\frac{y_t}{M} - \tilde{\theta}_{t-1}\right).$$

Though not substantially different from `AFFB`, for the purposes of removing ambiguity, the above Adaptive Forgetting Framework Binomial estimation procedure is displayed in Algorithm 4 and is referred to as `AFFBinom(M)`. Note `AFFBinom(1)` is equivalent to `AFFB`.

5.5.1 Stationary Behaviour

In Section 5.2, it was shown for stationary Bernoulli data that adaptive estimates for the negative log-likelihood cost were biased, more volatile, and generally led to worse performance

Algorithm 4 `AFFBinom(M)`: a general adaptive forgetting estimation procedure for binomial data streams of fixed trial size M.

Input: Stream of realisations $y_s \in \{1, 2, \ldots, M\}$, for $s = 1, 2, \ldots$.
Let M be the fixed binomial trial size. Let $\mathcal{J}(y, \theta)$ be an appropriate cost function for any $y \in \{1, 2, \ldots, M\}, \tilde{\theta} \in [0, 1]$. Let the step-size η_s be a deterministic function of s and w. Let $[\lambda_{\min}, \lambda_{\max}]$ be the truncation range of the forgetting factors.
Initialise $(\lambda, w, w', \tilde{\theta}, \tilde{\theta}') = (1, 0, 0, 0, 0)$.
for $s = 1, 2 \ldots$ **do**
    Set $(\lambda, w, w', \tilde{\theta}, \tilde{\theta}') = \text{AFFBINOM\_UPDATE}(y_s, M, \lambda, w, w', \tilde{\theta}, \tilde{\theta}')$
function AFFBINOM_UPDATE$(y_s, M, \lambda, w, w', \tilde{\theta}, \tilde{\theta}')$
    Compute cost gradient: $g = \mathcal{J}'(y, \theta)\Big|_{y=y_s, \theta=\tilde{\theta}}$
    e.g. for the negative log-likelihood cost $g = -\tilde{\theta}'\left(\frac{y_s}{\tilde{\theta}} - \frac{M-y_s}{1-\tilde{\theta}}\right)$.
    Update forgetting factor: $\lambda = \lambda - \eta_s g$.
    Truncate forgetting factor: $\lambda = \min\left(\max(\lambda, \lambda_{\min}), \lambda_{\max}\right)$
    Update effective sample size: $w = \lambda w + 1$.
    Update parameter estimate: $\tilde{\theta} = \left(1 - \frac{1}{w}\right)\tilde{\theta} + \frac{1}{w}\frac{y_s}{M}$.
    Update auxiliary derivatives: $w' = \lambda w' + w$, and $\tilde{\theta}' = \left(1 - \frac{1}{w}\right)\tilde{\theta}' - \frac{w'}{w^2}\left(\frac{y_s}{M} - \tilde{\theta}\right)$
    return $(\lambda, w, w', \tilde{\theta}, \tilde{\theta}')$
end function

than when using the squared error cost. This section explores these phenomena for the case of stationary binomial data, before assessing performance for both stationary and non-stationary data simulations in the next section. Intriguingly, this same relationship between the cost functions does not appear to manifest with binomial data. As in, Section 5.2, consider simulating $m = 1000$ data streams of length $n = 2000$ for fourth different values of binomial trial size $M \in \{2, 5, 10, 100\}$, with fixed success parameter θ_0. Here, $\theta_0 = 0.9$ is used somewhat arbitrarily, but the behaviour observed is also evident for other fixed values of θ_0. Figure 5.8 shows the ensemble means of the adaptive mean estimates for the stationary simulation described above for various learning rates (in a similar manner to Figure 5.1 for the Bernoulli case), Each row of panels corresponds to a different binomial trial size M, with the left and right-hand columns corresponding to the negative log-likelihood and squared error cost functions respectively. Surprisingly, the phenomena mentioned above are not present in the binomial case for every trial size tested. The adaptive estimator bias when using the NLL cost appears to be almost negligible for all combinations of trial size and learning rate, excluding the case for $M = 2, \eta = 0.1$. Furthermore, it looks as if this bias is now lower when using the NLL cost compared to the squared error cost. Additionally, the ensemble averages are less volatile

for the NLL cost. For both cost functions, increasing the trial size M decreases the bias and volatility in the adaptive estimators, with any bias practically non-existent for $M = 100$.

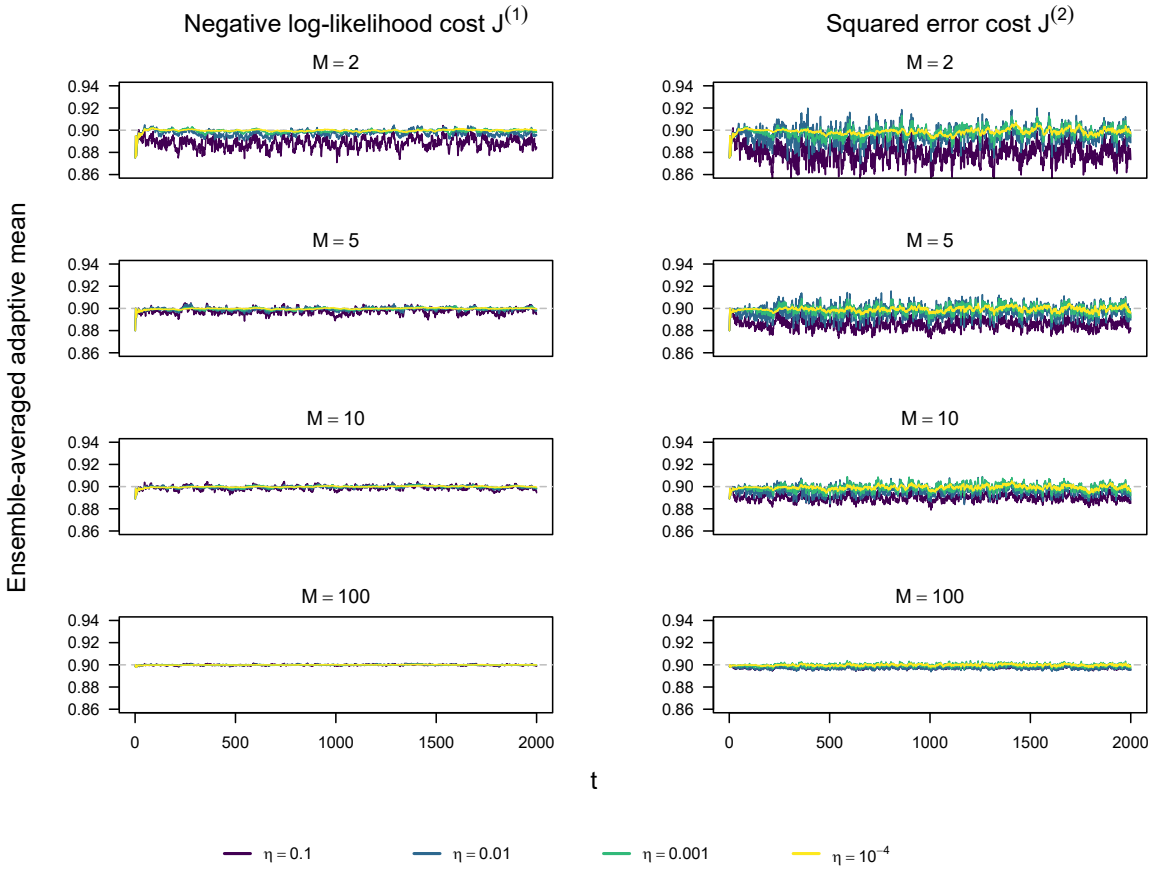

Figure 5.8: Ensemble means of the adaptive mean estimates over $m = 1000$ stationary binomial data streams, written $\overline{\theta}_t^{(1:m)}$. The success probability used to simulate the data is $\theta_0 = 0.9$ in each panel, and is shown by the horizontal grey dashed line. Each coloured line plot within a panel shows the ensemble mean for $1 \leq t \leq 2000$ for a different learning rates η. Each row of panels corresponds to a different binomial trial size $M \in \{2, 5, 10, 100\}$. The left-hand side column of panels all use the negative log-likelihood cost function $\mathcal{J}^{(1)}$ to tune the forgetting factors, with the right-hand side column of panels all using the squared error cost function $\mathcal{J}^{(2)}$.

5.5.2 Simulation Assessment

The simulation scenarios described in Section 5.3.1 are also used here to study the behaviour of `AFFBinom(M)`. However, instead of an exhaustive investigation using various binomial trial sizes, the experiment is restricted to a single trial size $M = 2$, with simulation length $n = 10^5$ as before. RMSE results for `AFFBinom(M)` may not seem perfectly comparable with `AFFB` given the underlying data processes are different. However, given the RMSE performance measure is based on the same parameter, which experiences the same drift dynamics across both sets

of simulations, the RMSE results should be on a similar scale. Though, as the binomial data can be considered as sums of Bernoulli data, in the binomial case the procedure effectively sees twice as much data, which intuitively suggests `AFFBinom(2)` may have an unfair advantage in achieving lower RMSE results.

Taking the above into consideration, the two procedures may be compared using the relative RMSE, defined to be `AFFBinom(2)` RMSE divided by the RMSE for `AFFB`, where the values for the denominator are available from Table 5.3. The values for the numerator; the RMSE results for `AFFBinom(2)`, are shown in Table 5.6, where cells coloured in green indicate a lower RMSE than for `AFFB` (or relative RMSE less than 1), with cells coloured in red indicating otherwise. The cell entries with bold text correspond to the lowest RMSE achieved by `AFFBinom(2)` for a specific simulation scenario across both cost functions and all tested learning rates.

Table 5.6: Root mean squared error results for `AFFBinom(2)` for a selection of stationary and non-stationary simulation environments. Cells coloured in green indicate a lower RMSE than for `AFFB` (see Table 5.3), with cells coloured in red otherwise.

		Cost function							
		Negative log-likelihood $\mathcal{J}^{(1)}$				Squared error $\mathcal{J}^{(2)}$			
Simulation		Learning Rate η				Learning Rate η			
Type	Parameters	0.1	0.01	0.001	10^{-4}	0.1	0.01	0.001	10^{-4}
Static	$\theta_0 = 0.5$	0.078	0.048	0.030	**0.017**	0.097	0.107	0.071	0.044
	$\theta_0 = 0.99$	0.032	0.018	0.007	**0.003**	0.037	0.025	0.017	0.011
Smooth (slow)	$\alpha = 0.98, \beta = 0.01$	0.060	0.039	0.040	**0.030**	0.075	0.074	0.050	0.043
	$\alpha = 0.5, \beta = 0.25$	0.073	0.045	0.039	**0.033**	0.092	0.098	0.065	0.058
Smooth (fast)	$\alpha = 0.98, \beta = 0.01$	0.082	0.086	0.075	**0.072**	0.094	0.094	0.078	0.076
	$\alpha = 0.5, \beta = 0.25$	0.084	0.083	0.074	**0.068**	0.097	0.110	0.082	0.073
Abrupt (slow)	$\theta_0 = 0.95$	0.050	**0.036**	0.042	0.063	0.058	0.053	0.053	0.046
	$\theta_0 = 0.75$	0.069	**0.045**	0.051	0.082	0.089	0.093	0.072	0.059
Abrupt (fast)	$\theta_0 = 0.95$	**0.075**	0.090	0.119	0.097	0.084	0.080	0.082	0.079
	$\theta_0 = 0.75$	0.081	**0.078**	0.104	0.085	0.098	0.109	0.095	0.093

There are two striking remarks regarding the RMSE results for the negative log-likelihood cost $\mathcal{J}^{(1)}$. First, the RMSE is lower for *every* simulation scenario when using `AFFBinom(2)` than `AFFB`, with a mean relative RMSE of 0.74. Second, the lowest RMSE for a given simulation scenario is *always* obtained when using the negative log-likelihood cost $\mathcal{J}^{(1)}$. Overall, the RMSE results for using the NLL cost $\mathcal{J}^{(1)}$ are substantially lower than those for the squared-

error cost $\mathcal{J}^{(2)}$. Peculiarly, using `AFFBinom(2)` has a drastic impact when using $\mathcal{J}^{(2)}$ for the stationary and slow smooth scenarios, where the mean relative RMSE is over 2. Interestingly $\mathcal{J}^{(2)}$ gives a lower mean RMSE over all fast abrupt simulations and generally performs on par, if not a little better, for most configuration combinations, despite not achieving the lowest among all combinations. Having seen in the previous section the differences in the consequences of cost function choice for `AFFBinom`, it is not unreasonable to expect the RMSE results for the NLL cost to be lower than those for the squared error cost.

Based on the results and remarks above, using the negative log-likelihood cost for `AFFBinom(2)` is generally the better choice, and never a bad choice. Further experimentation suggests these results are consistent as M increases (see the results for $M = 10$ and $M = 100$ in Tables B.1 and B.2 in the appendix). In addition, increasing M consistently lowers the RMSE achieved using the negative log-likelihood cost, but this is not the case for the squared-error cost. It is peculiar, and perhaps a little unfortunate, that the negative log-likelihood cost for the Bernoulli case, where $M = 1$, does not provide such sure-fire results.

While not perfectly comparable, the results above demonstrate that `AFFBinom(M)` using the negative log-likelihood cost leads to considerably lower RMSE results than `AFFB` for the explored simulations. When processing Bernoulli data streams, the opportunity to process the data as a Bernoulli data stream instead is generally available. If not available from the data source directly, one way to achieve this is to sum a mini-batch of size M of the Bernoulli observations and assume that the data is i.i.d. within the mini-batch, resulting in the sum being a binomial observation. Care should be taken to ensure M is not so large that the i.i.d. assumption is heavily violated due to the underlying drift. However, for many cases, even choosing $M = 2$ should benefit the adaptive estimation procedure.

A further generalisation to the multinomial case is not covered here, but a thorough treatment can be found in Plasse (2019). An interesting direction for future work would be to explore the consequences of the choice of cost function in the multinomial case.

5.6 Conclusion

This chapter developed a procedure for adaptive estimation for binary data. The estimation behaviour and performance assessment of this procedure was explored in both stationary

and non-stationary environments, with particular focus on the choice of cost function. A modification to this procedure was proposed to prevent truncation bias. Finally, the procedure was extended to operate on binomial data. Some loose recommendations are possible from this exploration. When choosing the learning rate for binary data, it is not advisable to set $\eta \geq 0.1$, and more generally, it is not advisable to scale the learning rate based on the variance of the data. Where possible, processing binomial data should lead to more reliable estimation than binary data. The NLL cost function is recommended for binomial data and the squared error cost is recommended for binary data. The next chapter builds on these tools for Bernoulli estimation to develop adaptive streaming quantile estimation.

6 Streaming Quantile Estimation

Sample quantile estimates are a central tool for many statistical methods, and for a fixed data set, are commonly obtained by inverting the empirical cumulative distribution function. This is equivalent to choosing the appropriate index of the sorted data set. Specifically, for a data set of size N, the sample q-quantile estimate is given by the order statistic $x_{(k)}$, for integers $k = Nq$ (or appropriately aggregated otherwise) as discussed in Section 2.3. Both computational restriction and temporal variation prevent the routine deployment of such procedures against streaming data. Alternative approaches are needed and this chapter reviews and extends existing approaches, and provides novel algorithms for streaming quantile estimation (SQE).

Section 6.1 explores the history of sequential quantile estimation starting from its roots in database technology. Section 6.2 builds on the fundamental methodology developed in Chapter 5 to develop a general framework for incorporating adaptive forgetting for streaming quantile estimators. Section 6.3 uses this framework to extend three modern methods in the literature, and is used in the context of a newly proposed method, `AFSQE`. Section 6.4 thoroughly explores the performance of the methods in a wide range of simulation environments. Section 6.5 takes a detour to explore deployment of these procedures on real data. Finally, Section 6.6 explores and assesses methods to estimate multiple quantiles concurrently.

6.1 Background and Related Work

Many procedures in the computer science literature have been developed to sequentially estimate quantiles without needing to sort the data beforehand. Munro and Paterson (1980) showed that any procedure able to estimate a quantile with arbitrary accuracy would require multiple passes of the data, but managed to reduce the overall storage complexity required. Later state-of-the-art work in Jain and Chlamtac (1985) and Greenwald, Khanna, et al. (2001) provide guarantees of estimation accuracy subject to how much of the stream is stored in

memory. Data sketch methods as in Cormode and Muthukrishnan (2005) propose space-efficient data structures to store approximate samples or summaries of the data. These procedures are generally designed with database efficiency in mind, rather than statistical value. Distinct from such approaches, this chapter focus on methods for which statistical accuracy is paramount. Many such methods are based on stochastic approximation (SA), as introduced in the seminal paper by Robbins and Monro (1951) which has spawned extensive applications in stochastic optimisation, especially in both big data and streaming settings. This idea was later developed into the state-of-the-art procedure in Tierney (1983), which recursively computes quantile estimates based on recursively computed density estimates. However, the SA framework is not suitable for streaming environments where the data experiences *drift* and the almost sure convergence properties of the estimator no longer hold. Similarly, the previously mentioned computer science methods are not suitable for drift because they rely on storing buffers of summary information which quickly becomes stale as the underlying distribution changes over time. The challenge of drift motivated the development of the `EWSA` (Exponentially Weighted Stochastic Approximation) procedure in Chen, Lambert, et al. (2000), which borrows from both the EWMA (Exponentially Weighted Moving Average) (Roberts 1959) and SA procedures in order to incorporate temporal adaptivity into estimators for both the density and quantiles.

The SA-based methods suffer from poor estimation in the extreme tails, sometimes even producing nonsensical estimates outside the observed range. Interestingly, the `SL` (Stochastic Learning) procedure in Yazidi and Hammer (2016) uses a multiplicative, as opposed to an additive, probabilistic-update step for quantiles and hence experiences considerably faster convergence than that of `EWSA` for both stationary and non-stationary data. This is interesting given the sequential updates in Chapters 3 to 5 all use an additive update step. `DUMIQE` is a modified version of `SL`, which introduces a control parameter to enforce a deterministic update step of controlled size (Yazidi and Hammer 2017). Both `SL` and `DUMIQE` exhibit better extreme tail performance than SA-based alternatives. Later `QEWA` (Hammer, Yazidi, and Rue 2019) generalised this multiplicative update step further to be based on the distance between arriving data and the current quantile estimates instead of a fixed size. This generalisation considerably improves the performance of the estimator to the extent that it is considered state-of-the-art (Hammer, Yazidi, and Rue 2019).

The aforementioned methods focus on the estimation of a *single* quantile, and produce contradictory results when used for multiple quantile estimation since monotonicity of the quantile estimates is not guaranteed in general. The Data Skeleton approach in McDermott et al. (2007) expands on the method in Liechty et al. (2003) by simultaneously estimating ranks of the data. This guarantees the ranks are in ascending order, but is deterministic and does not allow for non-stationarity. The `monotoneSA` method proposed in Cao et al. (2009), based on `EWSA`, involves interpolating a globally increasing density function before the standard SA update step. The `MDUMIQE` procedure proposed in Hammer, Yazidi, and Rue (2020) is a multiple quantile version of `DUMIQE`, and ensures monotonicity by limiting the rate at which the quantile estimates can adapt. While this allows the authors to prove a convergence result under stationarity, this restriction has the disadvantage of not providing sufficient temporal adaptation for a rapidly changing target distribution. More recently, the same authors presented `CondQ`; a framework for extending streaming single quantile estimation procedures for estimating multiple quantiles (Hammer, Yazidi, and Rue 2021). This framework exploits the conditional ordering of quantiles to ensure monotonicity of multiple quantile estimates, and when used to extend `QEWA` is shown to outperform `MDUMIQE` in simulation.

A particular flaw of both `EWSA`, `DUMIQE` and `QEWA` is that the rate at which data is forgotten is controlled by a user-set parameter, or weight, which assumes that the rate of change of the distribution of the data is constant. This in turn leads to a compromise between the adaptability of the procedure and the bias of its estimates. In addition, it is often the case that streaming data can exhibit periods of stability as well as both abrupt and gradual drift, for example in Figure 2.1. As such, it would be ideal to be able to vary this weight according to the degree of non-stationarity of the data. The next section introduces an AF framework for adaptively estimating ECDF values. This creates the opportunity to incorporate adaptive forgetting into these streaming quantile estimation procedures to allow for a dynamically tuned weight; removing the need for the user to set parameters that should instead be determined by the data.

6.2 Adaptive ECDF Estimation

When the data comes from a known parametric family of distributions it is typical to use the one-step ahead negative log-likelihood (NLL) as a cost function to sequentially tune the forgetting factors as discussed in Section 2.6.2. In this case the NLL is evaluated at the data for the current iteration given parameter estimates for the previous iteration. Clearly, this approach is not feasible in situations where the family of distributions for the data generating mechanism is unknown, or changes over time. However, by considering adaptive estimates of the empirical cumulative distribution function (ECDF), it is possible to learn the underlying rate of drift, and parameterise the tuning step irrespective of the data generating mechanism.

Given a random sample of realisations $(x_1, x_2, \ldots, x_N)$ of X with CDF F_X, the empirical CDF (ECDF) is given by

$$\bar{F}_N(x) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{x_i \leq x\}.$$

Hence, as a sum of i.i.d. Bernoulli random variables, $N\bar{F}_N(x) \sim \text{Binomial}(N, \theta)$, with $\theta = P(X \leq x)$.

Now consider a data stream $(x_s)_{s \in \mathbb{N}}$ where each x_s is a realisation of X_s with CDF F_{X_s}. Then, $Y_{s,x} = \mathbb{1}\{x_s \leq x\}$ is a Bernoulli random variable, whose parameter, $\theta_{s,x}$, can be estimated as an adaptive ECDF value. After re-framing the quantity of interest from the data to its corresponding ECDF value, the methodology and procedures developed in Chapter 5 can estimate $\theta_{s,x}$ sequentially via

$$\begin{aligned} \tilde{\theta}_{s,x} &= \left(1 - \frac{1}{w_{s,x}}\right) \tilde{\theta}_{s-1,x} + \frac{1}{w_{s,x}} \mathbb{1}\{x_s \leq x\}, \\ w_{s,x} &= \lambda_{s,x} w_{s-1,x} + 1. \end{aligned}$$

Then, as described in Section 5.1.1, the forgetting factors can be tuned by SGD, using either a binomial NLL or squared error cost function. Notice here that $\tilde{\theta}_{s,x}$ corresponds to an adaptive ECDF value at x, where x is some value generally in the range of X_s. Another way to view this is to consider that x is the q-quantile, where $\tilde{\theta}_{s,x}$ is an estimate of q. This strategy is employed in the next section to drive a general AF procedure that can be combined with existing streaming quantile estimation methods with the purpose of enhancing their adaptability and automating the setting of control parameters.

6.3 Adaptive Streaming Quantile Estimation

This section illustrates four adaptive streaming quantile estimation procedures. The first three approaches are based on existing fixed-weight streaming quantile estimation procedures in the literature, but have been enhanced with the AF framework. Finally, an entirely novel streaming quantile procedure is presented. These procedures and properties are summarised in Table 6.1. The next section considers comparing the performance of these approaches on simulated data.

Table 6.1: Summary of adaptive streaming quantile estimation procedures.

Method	Extension of	Novelty	Specific Conditions
`AFSA`	`EWSA` (Chen, Lambert, et al. 2000)	AF extension only	Requires simultaneous computation of the lower and upper quartiles to estimate interquartile range.
`AFMIQE`	`DUMIQE` (Hammer and Yazidi 2017)	AF extension only	Can only handle positive valued data.
`QAF`	`QEWA` (Hammer, Yazidi, and Rue 2019)	AF extension only	—
`AFSQE`	—	Entirely novel	—

6.3.1 `AFSA`

Keeping notation consistent with Chen, Lambert, et al. (2000), the `EWSA` procedure starts with an initial density estimate $f_0^{*(q)}$ at the q-quantile with corresponding initial quantile estimate $S_0^{*(q)}$. These estimates are then incrementally updated upon receiving a batch of M data points $\{x_{t,1}, \ldots, x_{t,M}\}$ at iteration t via the equations

$$
\begin{aligned}
S_t^{*(q)} &= S_{t-1}^{*(q)} + \frac{1}{w f_{t-1}^{*(q)}} \left(q - \frac{1}{M} \sum_{j=1}^{M} \mathbb{1}\left\{ x_{t,j} \leq S_{t-1}^{*(q)} \right\} \right), \\
f_t^{*(q)} &= \left(1 - \frac{1}{w}\right) f_{t-1}^{*(q)} + \frac{1}{2 w c_{t-1}^{*} M} \sum_{j=1}^{M} \mathbb{1}\left\{ \left| x_{t,j} - S_{t-1}^{*(q)} \right| \leq c_{t-1}^{*} \right\}, \\
c_t^{*} &= r_t^{*} \sum_{j=M+1}^{2M} \frac{1}{\sqrt{j}},
\end{aligned}
$$

where r_t^* is the estimated interquartile range (IQR) given by $\left(S_t^{*(0.75)} - S_t^{*(0.25)}\right)$.

The `EWSA` procedure is itself a modification of Tierney's stochastic approximation (SA) in Tierney (1983) intended to be able to handle changes in the data over time. Specifically it uses a fixed weight w to control the rate of forgetting and non-vanishing neighbourhood c_t^*. These modifications present two new challenges. First, determining a suitable choice for w. Second, in periods of stationarity, the quantile estimates require a vanishing neighbourhood for consistency, as is the case in SA, where $c_t \to 0$ as $t \to \infty$. The AF framework addresses both of these issues in that it allows the rate of forgetting to adapt according to the data. This ensures that the weight w need not be set by the user, and also that under stationarity (with all forgetting factors equal to 1) the proposed Adaptive Forgetting Stochastic Approximation (`AFSA`) procedure below will be equivalent to Tierney's SA. Notice that after replacing w with the time-varying w_t, the update equations become

$$\begin{aligned} S_t^{(q)} &= S_{t-1}^{(q)} + \frac{1}{w_t f_{t-1}^{(q)}}\left(q - \frac{1}{M}B_t^{(k)}\right), \\ f_t^{(q)} &= \left(1 - \frac{1}{w_t}\right) f_{t-1}^{(q)} + \frac{1}{w_t}\frac{1}{2c_t M}A_t^{(k)}, \end{aligned} \tag{6.1}$$

with auxiliary counts

$$\begin{aligned} A_t^{(k)} &= \sum_{j=1}^{M} \mathbb{1}\left\{\left|x_{t,j} - S_{t-1}^{(q)}\right| \leq c_t\right\}, \\ B_t^{(k)} &= \sum_{j=1}^{M} \mathbb{1}\left\{x_{t,j} \leq S_{t-1}^{(q)}\right\}, \end{aligned} \tag{6.2}$$

where $f_t^{(q)}$ is the `AFSA` density estimate at the q-quantile and $S_t^{(q)}$ is the corresponding `AFSA` quantile estimate. Note that the estimates with a $*$ correspond to `EWSA`, where estimates without correspond to `AFSA`. The fixed neighbourhood size can be relaxed such that

$$c_t = \frac{r_t}{\sqrt{w_t}},$$

where r_t is the IQR determined from the `AFSA` quantile estimates, i.e. $r_t = S_t^{(0.75)} - S_t^{(0.25)}$. Despite the claim in Chen, Lambert, et al. (2000), the quantile update in Equation (6.1) can blow up when the density estimates gets very close to zero, which usually happens if the data experiences a change of significant magnitude relative to the range of the data. For

implementation purposes this is undesirable and can be avoided by bounding the density estimate from below as in Tierney (1983), and using the following update instead

$$S_t^{(q)} = S_{t-1}^{(q)} + \frac{1}{w_t}\frac{1}{e_{t-1}^{(q)}}\left(q - \frac{1}{M}B_t\right),$$
$$e_{t-1}^{(q)} = \max\left(\frac{1}{100+w}, f_{t-1}^{(q)}\right).$$

The `AFSA` procedure is shown in Algorithm 5, with the update step represented by AFSA_UPDATE,

Algorithm 5 `AFSA`: a streaming quantile estimation procedure using adaptive forgetting based on `EWSA` (Chen, Lambert, et al. 2000)

Input: Stream of mini-batches of realisations $(x_{s,1}, x_{s,2}, \ldots, x_{s,M})$ where each $x_{s,j} \in \mathbb{R}$, for $s = 1, 2, \ldots$, and $j = 1, 2, \ldots, M$.
Let M be the fixed mini-batch size of arriving data observations. Let $\mathcal{S} = (S^q, S^{(0.25)}, S^{(0.75)})$ be a vector of adaptive quantile estimates for the q, 0.25 and 0.75 quantiles with corresponding vector of adaptive density estimates $\mathcal{F} = (f^q, f^{(0.25)}, f^{(0.75)})$ respectively.
Initialise $(\lambda, w, w', \tilde{\theta}, \tilde{\theta}') = (1, 0, 0, 0, 0)$.
Initialise $\mathcal{S} = (S^{(q)}, S^{(0.25)}, S^{(0.75)})$ using an initial sample batch $(x_{0,1}, \ldots, x_{0,M})$, and $\mathcal{F} = (f^{(q)}, f^{(0.25)}, f^{(0.75)})$ according to Equation (6.3).
for $s = 1, 2 \ldots$ **do**
    Set $y_s = \sum_{j=1}^{M} \mathbb{1}\left\{x_{s,j} < S^{(q)}\right\}$.
    Set $(\lambda, w, w', \tilde{\theta}, \tilde{\theta}') =$ AFFBINOM_UPDATE$(y_s, M, \lambda, w, w', \tilde{\theta}, \tilde{\theta}')$ (see Algorithm 4).
    Set $(\mathcal{S}, \mathcal{F}) =$ AFSA_UPDATE$((x_{s,1}, \ldots, x_{s,M}), q, \mathcal{S}, \mathcal{F}, w)$
function AFSA_UPDATE$((x_{s,1}, \ldots, x_{s,M}), q, \mathcal{S}, \mathcal{F}, w)$
    Compute IQR: $r = S^{(0.75)} - S^{(0.25)}$
    Compute neighbourhood size: $c = \frac{r}{\sqrt{w}}$
    Compute auxiliary counts: $A^{(k)} = \sum_{j=1}^{M} \mathbb{1}\left\{|x_{s,j} - S^{(k)}| \leq c\right\}$ for $k \in \{q, 0.25, 0.75\}$.
    Bound current density estimates: $e^{(k)} = \max\left(\frac{1}{100+w}, f^{(k)}\right)$
    Compute quantile estimates: $S^{(k)} = S^{(k)} + \frac{1}{w}\frac{1}{e^{(k)}}\left(k - \frac{1}{M}y_s\right)$
    Compute density estimates: $f^{(k)} = \left(1 - \frac{1}{w}\right)f^{(k)} + \frac{1}{w}\frac{1}{2cM}A^{(k)}$.
    return $(\mathcal{S}, \mathcal{F})$
end function

which is a function of the parameters with time index subscripts removed to emphasise the constant memory properties of the procedure. Initial values $S_0^{(k)}$ for $k \in \{q, 0.25, 0.75\}$ should be chosen as the relevant sample quantiles from some initial sample of data $(x_{0,1}, \ldots, x_{0,M})$. However, the procedure is not particularly sensitive to the initial quantile estimates, so long as they are reasonably within the true range. Hence, if an initial batch is not available, it

is usually sufficient to use a single data point, or sensible estimate, $x_{0,0}$, and set the initial quantile estimates as

$$S_0^{(q)} = x_{0,0}, \quad S_0^{(0.25)} = x_{0,0} - 1, \quad S_0^{(0.75)} = x_{0,0} + 1.$$

The initial density estimates should also be determined using from an initial batch using an initial neighbourhood estimate c_0, such that

$$f_0^{(k)} = \max\left(\frac{1}{2c_0 M} A_0, 1\right), \tag{6.3}$$

where

$$c_0 = \left[S_0^{(0.75)} - S_0^{(0.25)}\right] \frac{1}{M} \sum_{j=1}^{M} j^{-\frac{1}{2}},$$

as in `EWSA`, and $A_0^{(k)}$ can be computed from Equation (6.2).

For an example of this procedure in operation, see Figure 6.1.

6.3.2 `AFMIQE`

For strictly positive observations $x_t > 0$ arriving from a data stream $(x_t)_{t\in\mathbb{N}}$, the `DUMIQE` procedure (Hammer and Yazidi 2017) can be expressed in its simplest form, where quantile estimates are computed sequentially via

$$\hat{Q}_t^{(q)} = \begin{cases} (1+\omega q)\hat{Q}_{t-1}^{(q)}, & \text{if } \hat{Q}_{t-1}^{(q)} < x_t, \\ (1-\omega(1-q))\hat{Q}_{t-1}^{(q)}, & \text{if } \hat{Q}_{t-1}^{(q)} \geq x_t, \end{cases}$$

with $\omega \in (0, 1]$ fixed, but usually chosen in the range $(0, 0.2]$ to restrict the size of the update step. Note the authors use λ in place of ω to denote the weight, but this notation is avoided here to help disambiguate from forgetting factors. Negative-valued data can be handled with additional effort by using either a location shift or invertible transform, such as $g(x) = \exp(-x)$, to ensure the transformed data is positive (Yazidi and Hammer 2017). This procedure can naturally be extended using the AF framework by replacing $1-\omega$ with an adaptive forgetting factor λ_t, updated as in Section 6.2. Note in this procedure the forgetting factors should be truncated to the tighter interval with $\lambda_{\min} = 0.8$ as the update scheme is more sensitive to $\lambda_{\min}$ due

to its multiplicative update step. Then the novel `AFMIQE` (Adaptive Forgetting Multiplicative Incremental Quantile Estimation) procedure estimates individual q-quantiles recursively with

$$\widetilde{Q}_t^{(q)} = \begin{cases} (1 + (1 - \lambda_t)q)\widetilde{Q}_{t-1}^{(q)}, & \text{if } \widetilde{Q}_{t-1}^{(q)} < x_t, \\ (1 - (1 - \lambda_t)(1 - q))\widetilde{Q}_{t-1}^{(q)}, & \text{if } \widetilde{Q}_{t-1}^{(q)} \geq x_t. \end{cases}$$

Note that these updates are probabilistically weighted via q to increase or decrease the quantile estimates accordingly. Here the main difference with the original `DUMIQE` is the adaptive $1 - \lambda_t$ replaces the fixed ω, which helps regulate the volatility of the estimator, particularly improving the estimator in stationary periods. The `AFMIQE` procedure is shown in Algorithm 6 for positive-valued data streams. It is trivially extended for the full real line by either operating on a transformed data stream $(y_t)_{t\in\mathbb{N}}$ where $y_t = x_t + \Delta$, where $\Delta \geq \min_t(x_t)$ and then transforming the quantile estimates back by subtracting Δ. This is only possible if the range of values of the data stream are known in advance, which is not always possible. An alternative is to use a transformed data stream $(g(x_t))_{t\in\mathbb{N}}$ where $g(x_t) > 0$ is invertible, such as $g(x) = \exp(-x)$ and then transforming the quantile estimates back using g^{-1}.

For an example of this procedure in operation, see Figure 6.2.

Algorithm 6 `AFMIQE`: a streaming quantile estimation procedure using adaptive forgetting based on `DUMIQE` (Hammer and Yazidi 2017)

Input: Stream of realisations $(x_1, x_2, \dots)$ where each $x_s > 0$, for $s \in \mathbb{N}$.
Let $\widetilde{Q}^{(q)}$ be the adaptive estimate for the q-quantile.
Initialise $(\lambda, w, w', \tilde{\theta}, \tilde{\theta}') = (1, 0, 0, 0, 0)$.
Initialise $\widetilde{Q}^{(q)}$ using a sample quantile from an initial data sample, or single point x_0.
for $s = 1, 2 \dots$ **do**
    Set $y_s = \mathbb{1}\left\{x_s < \widetilde{Q}^{(q)}\right\}$.
    Set $(\lambda, w, w', \tilde{\theta}, \tilde{\theta}') = \text{AFFB\_UPDATE}(y_s, \lambda, w, w', \tilde{\theta}, \tilde{\theta}')$ (see Algorithm 2).
    Set $\widetilde{Q}^{(q)} = \text{AFMIQE\_UPDATE}(x_s, \lambda, q, \widetilde{Q}^{(q)})$.
function AFMIQE_UPDATE$(x_s, \lambda, q, \widetilde{Q}^{(q)})$
    if $\widetilde{Q}^{(q)} < x_s$ **then**
        Set $\widetilde{Q}^{(q)} = (1 + (1 - \lambda)q)\widetilde{Q}^{(q)}$.
    else
        Set $\widetilde{Q}^{(q)} = (1 - (1 - \lambda)(1 - q))\widetilde{Q}^{(q)}$.
    end if
    return $\widetilde{Q}^{(q)}$.
end function

6.3.3 `QAF`

In Hammer, Yazidi, and Rue (2019), the authors remark that `DUMIQE` procedure considers a binary decision on whether to increase or decrease the quantile estimates by a fixed size, but does *not* use the observations to determine an appropriate update size. They propose `QEWA`: an incremental Quantile Estimator using a generalised Exponentially Weighted Average, in the same paper, where the intuition is to use an update step-size that is proportional to the distance between the most current quantile estimate and the most recent observation. As its name implies, the sequential update can be shown to be a convex-combination similar to the update for EWMA, but with a varying weight that encapsulates different update step-sizes. Specifically, for a data stream $(x_t)_{t\in\mathbb{N}}$, the `QEWA` update step is given by

$$\hat{Q}_t^{(q)} = \begin{cases} \hat{Q}_{t-1}^{(q)} + \omega c_{t-1} \frac{q}{\mu_{t-1}^{+} - \hat{Q}_{t-1}^{(q)}} |x_t - \hat{Q}_{t-1}^{(q)}|, & \text{if } \hat{Q}_{t-1}^{(q)} < x_t, \\ \hat{Q}_{t-1}^{(q)} - \omega c_{t-1} \frac{1-q}{\hat{Q}_{t-1}^{(q)} - \mu_{t-1}^{-}} |x_t - \hat{Q}_{t-1}^{(q)}|, & \text{if } \hat{Q}_{t-1}^{(q)} \geq x_t, \end{cases}$$

where, as in Algorithm 6, ω is a fixed weight in $(0, 1]$. Here, μ_t^+ and μ_t^- are estimates corresponding to the underlying conditional expectations of $\mathbb{E}\left[X_t \mid X_t > \hat{Q}_{t-1}^{(q)}\right]$ and $\mathbb{E}\left[X_t \mid X_t \leq \hat{Q}_{t-1}^{(q)}\right]$ respectively. The quantities

$$\frac{q}{\mu_{t-1}^{+} - \hat{Q}_{t-1}^{(q)}} \quad \text{and} \quad \frac{1-q}{\hat{Q}_{t-1}^{(q)} - \mu_{t-1}^{-}}$$

are chosen to ensure $\hat{Q}_t^{(q)}$ converges to the true quantile, assuming a stationary data generating process (Hammer, Yazidi, and Rue 2019). The authors recommend choosing the sequence c_t to normalise the quantities above. That is,

$$c_t = \left[\frac{q}{\mu_t^{+} - \hat{Q}_t^{(q)}} + \frac{1-q}{\hat{Q}_t^{(q)} - \mu_t^{-}} \right]^{-1},$$

which allows the update equation to take the simpler generalised EWMA form

$$\hat{Q}_t^{(q)} = (1 - b_{t-1})\hat{Q}_{t-1}^{(q)} + b_{t-1} x_t,$$

where

$$b_t = \omega\Big(a_t + \mathbb{1}\Big\{x_t \leq \hat{Q}_t^{(q)}\Big\}(1 - 2a_t)\Big), \tag{6.4}$$

$$a_t = \frac{1}{c_t}\left(\frac{q}{\mu_t^+ - \hat{Q}_t^{(q)}}\right). \tag{6.5}$$

Estimates for the conditional expectations are obtained using the conditional EWMA update steps

$$\mu_t^+ = \begin{cases} \hat{Q}_t^{(q)} - \hat{Q}_{t-1}^{(q)} + (1-\gamma)\mu_{t-1}^+ + \gamma x_t & \text{if } \hat{Q}_{t-1}^{(q)} < x_t, \\ \hat{Q}_t^{(q)} - \hat{Q}_{t-1}^{(q)} + \mu_{t-1}^+ & \text{if } \hat{Q}_{t-1}^{(q)} \geq x_t, \end{cases}$$

$$\mu_t^- = \begin{cases} \hat{Q}_t^{(q)} - \hat{Q}_{t-1}^{(q)} + \mu_{t-1}^- & \text{if } \hat{Q}_{t-1}^{(q)} < x_t, \\ \hat{Q}_t^{(q)} - \hat{Q}_{t-1}^{(q)} + (1-\gamma)\mu_{t-1}^- + \gamma x_t & \text{if } \hat{Q}_{t-1}^{(q)} \geq x_t, \end{cases}$$

with fixed parameter $\gamma \in [0, 1]$, where the authors recommend choosing $\gamma/\omega = 0.01$.

The `QEWA` procedure provides perhaps the most interesting comparison with AF procedures because the update step is data-driven, though the main difference is `QEWA` chooses the update step based on the magnitude of changes to data as opposed to AF procedures which attempt to (meta-)learn a learning rate for the underlying drift. Still, the `QEWA` procedure depends on two control weight parameters ω and γ. These parameters effectively control the rate at which the estimates converge under stationarity and so present a ripe opportunity for the AF framework to learn and automate the burden of choosing these values. Despite the recommended ratio, experimentation suggested replacing ω with $1-\lambda_t$ (as in Section 6.3.2) and γ with w_t^{-1} performs best. This procedure is referred to as `QAF` (Quantile estimator with Adaptive Forgetting) when the AF framework is used to determine these weights, and is shown in Algorithm 7.

This procedure performs best when given an initial sample to initialise values for the adaptive quantile and conditional expectation estimates. If this is not available, a single observation, or guess, can be used to initialise the quantile estimate, but care must be taken to ensure the conditional expectations μ_t^+ and μ_t^+ are strictly above and below the corresponding quantile estimate for all t. Lines 21–25 account for an additional practical consideration to prevent a_t blowing up and is set to zero if the denominator is close to zero during implementation.

For an example of this procedure in operation, see Figure 6.3.

Algorithm 7 `QAF`: a streaming quantile estimation procedure using adaptive forgetting based on `QEWA` (Hammer, Yazidi, and Rue 2019)

Input: Stream of realisations $(x_1, x_2, \dots)$ where each $x_s \in \mathbb{R}$, for $s \in \mathbb{N}$.
Let $\widetilde{Q}^{(q)}$ and $\widetilde{Q}^{(q)}_{-1}$ be the current and previous adaptive estimates for the q-quantile respectively. Let $\varepsilon << 1$ be some tolerance, e.g. 10^{-8}.
Initialise $(\lambda, w, w', \tilde{\theta}, \tilde{\theta}') = (1, 0, 0, 0, 0)$.
Initialise $\widetilde{Q}^{(q)}$ using a sample quantile from an initial data sample, or single point x_0.
Initialise μ^+ and μ^- based on an initial sample conditional on $\widetilde{Q}^{(q)}$.
for $s = 1, 2 \dots$ **do**
    Set $y_s = \mathbb{1}\left\{x_s < \widetilde{Q}^{(q)}\right\}$.
    Set $(\lambda, w, w', \tilde{\theta}, \tilde{\theta}') = \text{AFFB\_UPDATE}(y_s, \lambda, w, w', \tilde{\theta}, \tilde{\theta}')$ (see Algorithm 2).
    Set $(\widetilde{Q}^{(q)}, \mu^+, \mu^-) = \text{QAF\_UPDATE}(x_s, \lambda, q, \widetilde{Q}^{(q)}, \mu^+, \mu^-)$.
function QAF_UPDATE$(x_s, \lambda, q, \widetilde{Q}^{(q)}, \mu^+, \mu^-)$
    Set $\widetilde{Q}^{(q)}_{-1} = \widetilde{Q}^{(q)}$.
    Set $\widetilde{Q}^{(q)} = (1 - b)\widetilde{Q}^{(q)} + b x_s$.
    if $\widetilde{Q}^{(q)}_{-1} < x_s$ **then**
        Set $\mu^+ = \widetilde{Q}^{(q)} - \widetilde{Q}^{(q)}_{-1} + \left(1 - \frac{1}{w}\right)\mu^+ + \frac{1}{w} x_s$.
        Set $\mu^- = \widetilde{Q}^{(q)} - \widetilde{Q}^{(q)}_{-1} + \mu^-$.
    else
        Set $\mu^+ = \widetilde{Q}^{(q)} - \widetilde{Q}^{(q)}_{-1} + \mu^+$
        Set $\mu^- = \widetilde{Q}^{(q)} - \widetilde{Q}^{(q)}_{-1} + \left(1 - \frac{1}{w}\right)\mu^- + \frac{1}{w} x_s$.
    end if
    Set $a_1 = \frac{q}{\mu^+ - \widetilde{Q}^{(q)}}, \quad a_2 = \frac{1-q}{\widetilde{Q}^{(q)} - \mu^-}$.
    if $|a_1 + a_2| < \varepsilon$ **then**
        Set $a = 0$.
    else
        Set $a = \frac{a_1}{a_1 + a_2}$.
    end if
    if $\widetilde{Q}^{(q)} < x_s$ **then**
        Set $b = (1 - \lambda)a$.
    else
        Set $b = (1 - \lambda)(1 - a)$.
    end if
    return $\widetilde{Q}^{(q)}, \mu^+, \mu^-$.
end function

6.3.4 `AFSQE`

This approach is entirely novel, in that it is not an AF extension, or generalisation, of an existing streaming quantile estimation procedure in the literature. The motivation for this approach is to exploit the fundamental roughly 'monotonic' relationship between ECDF values and corresponding quantile values. That is, if the observed adaptive ECDF value increases, or decreases, with respect to a specific quantile estimate then a better estimate can generally be obtained by appropriately decreasing, or increasing, the quantile estimate respectively. In an ideal situation, values of a suitable cost function between the quantile estimates and the truth would be available. Given the true quantile values are not available, we instead consider a proxy cost function that is designed to concurrently optimise an ideal, or oracle, cost.

Scoring functions for pointwise forecasts have received vast treatment in the literature (Gneiting 2011), where for q-quantile forecasts, the following strictly consistent scoring function is commonly used:

$$S_q(x, y) = (\mathbb{1}\{x \geq y\} - q)(x - y).$$

With this scoring function as a cost between the true q-quantile $Q_t^{(q)}$ and an estimate $\widetilde{Q}_t^{(q)}$, the oracle cost can be expressed as a multiple of a proxy cost with

$$S_q\Big(Q_t^{(q)}, \widetilde{Q}_t^{(q)}\Big) \approx S_q\Big(q, \tilde{\theta}_t^{(q)}\Big)\left(\frac{Q_t^{(q)} - \widetilde{Q}_t^{(q)}}{q - \tilde{\theta}_t^{(q)}}\right). \tag{6.6}$$

This is under the assumption that $\mathbb{1}\Big\{Q_t^{(q)} \geq \widetilde{Q}_t^{(q)}\Big\} \approx \mathbb{1}\Big\{q, \tilde{\theta}_t^{(q)}\Big\}$, which is reasonable given the roughly monotonic relationship between the adaptive ECDF and quantile estimates.

While access to the piecewise-linear function $S_q\Big(q, \tilde{\theta}_t^{(q)}\Big)$ is readily available, the function is not differentiable everywhere and so it is convenient to consider a concave-up quadratic that passes through the same minimum point $(q, 0)$. As shown in Appendix A.4, this quadratic is the squared error cost

$$S_q^*\Big(q, \tilde{\theta}_t^{(q)}\Big) = \Big(q - \tilde{\theta}_t^{(q)}\Big)^2.$$

Quantile estimates can then be updated sequentially with the SGD update step

$$\begin{aligned}\widetilde{Q}_t^{(q)} &= \widetilde{Q}_{t-1}^{(q)} - \eta_t \frac{\partial}{\partial \tilde{\theta}_t^{(q)}} S_q^*\Big(q, \tilde{\theta}_t^{(q)}\Big), \\ &= \widetilde{Q}_{t-1}^{(q)} + 2\eta_t\Big(q - \tilde{\theta}_t^{(q)}\Big),\end{aligned}$$

where the step size, η_t, needs to be weighted according to the expected scalar multiple between the oracle and proxy costs, and should vanish in stationary environments. Some simple experimentation suggests

$$\eta_t = \frac{\eta_0}{w_t}\Big|x_t - \widetilde{Q}_{t-1}^{(q)}\Big|,$$

where $\eta_0 = 1$, is a sensible choice. This approach naturally extends to the situation where observations come from a data stream of mini-batches, with each mini-batch having M realisations $((x_{t,1}, \ldots, x_{t,M}))_{t\in\mathbb{N}}$, simply by using `AFFBinom(M)` to drive the AF mechanism instead of `AFFB`. In this case, the learning rate η_t should be the mean absolute deviation between the observations and current quantile estimate. The `AFSQE` (Adaptive Forgetting Streaming Quantile Estimation) procedure is displayed for a fixed M in Algorithm 8.

For an example of this procedure in operation, see Figure 6.4.

Algorithm 8 `AFSQE`: streaming quantile estimation procedure using adaptive forgetting and stochastic gradient descent.

Input: Stream of mini-batches of realisations $(x_{s,1}, x_{s,2}, \ldots, x_{s,M})$ where each $x_{s,j} \in \mathbb{R}$, for $s = 1, 2, \ldots$, and $j = 1, 2, \ldots, M$.
Let M be a fixed mini-batch size. Let $\widetilde{Q}^{(q)}$ be the adaptive estimate for the q-quantile, where $\tilde{\theta}$ is the corresponding adaptive ECDF estimate. Let η_0 be a fixed learning hyperparameter.
Initialise $(\lambda, w, w', \tilde{\theta}, \tilde{\theta}') = (1, 0, 0, 0, 0)$.
Initialise $\widetilde{Q}^{(q)}$ using a sample quantile from an initial data sample, or single point x_0.
for $s = 1, 2 \ldots$ **do**
    Set $y_s = \sum_{j=1}^{M} \mathbb{1}\Big\{x_{s,j} < \widetilde{Q}^{(q)}\Big\}$.
    Set $(\lambda, w, w', \tilde{\theta}, \tilde{\theta}') =$ AFFBINOM_UPDATE$(y_s, M, \lambda, w, w', \tilde{\theta}, \tilde{\theta}')$ (see Algorithm 4).
    Set $\widetilde{Q}^{(q)} =$ AFSQE_UPDATE$((x_{s,1}, \ldots, x_{s,M}), q, w, \widetilde{Q}^{(q)}, \tilde{\theta})$.
function AFSQE_UPDATE$((x_{s,1}, \ldots, x_{s,M}), q, w, \widetilde{Q}^{(q)}, \tilde{\theta})$
    Set $\widetilde{Q}^{(q)} = \widetilde{Q}^{(q)} + 2\frac{\eta_0}{w}\frac{1}{M}\sum_{j=1}^{M}\Big|x_{s,j} - \widetilde{Q}^{(q)}\Big|\Big(q - \tilde{\theta}\Big)$.
    return $\widetilde{Q}^{(q)}$.
end function

6.4 Simulation Study

This section provides a thorough comparison of the performance of the streaming quantile estimation procedures presented in the previous section on simulated data. Also included are the non-adaptive, or fixed-weight methods that the AF extensions are derived from.

The simulation scenarios in Hammer, Yazidi, and Rue (2019) are repeated here for the purpose of reproducibility. There are a total of eight distinct simulation scenarios, or environments, covering two different probability distributions (normal and chi-squared), with two distinct types of drift pattern (smooth or switch), with the option of two different rates of drift (fast or slow). The rate of drift is governed with a single parameter τ, where $\tau = 100$ corresponds to fast drift and $\tau = 500$ corresponds to slow drift. Then, each of these values of τ is combined with a distribution and drift model that defines the distribution parameters as a function of t, for $t = 1, 2, \ldots, n$, where n is the number of simulated realisations. Formally the four combinations of distribution and drift dynamics considered are listed below.

- **Normal Smooth**

$$\mu_t = a \sin\left(\frac{2\pi}{\tau} t\right),$$
$$\sigma_t = 1.$$

- **Normal Switch**

$$\mu_t = \begin{cases} a & \text{if } t \bmod \tau \leq \frac{\tau}{2}, \\ -a & \text{otherwise} \end{cases},$$
$$\sigma_t = 1.$$

- **Chi-squared Smooth**

$$\nu_t = a + b \sin\left(\frac{2\pi}{\tau} t\right) + b, \sigma = 1.$$

- **Chi-squared Switch**

$$\nu_t = \begin{cases} -a + b & \text{if } t \bmod \tau \leq \frac{\tau}{2}, \\ -a & \text{otherwise} \end{cases}.$$

Here $b > a$ are simulation parameters, which are chosen to be $a = 2, b = 6$ as in Hammer, Yazidi, and Rue (2019). The data is simulated from a stochastic process, where at time t, the normal location and scale parameters μ_t, σ_t respectively. In the chi-squared case, the data is simulated with degrees of freedom ν_t. Note that the switch cases are forms of recurring abrupt drift.

There are a total of 9 specific variations of procedure tested in the simulation, where each variation is tested for four different learning $\eta \in \{10^{-i}\}_{i=2}^{5}$ if the method uses the AF framework, or for four sensible weight parameters if the method is non-adaptive. Specifically, 3 of these arise from `AFSA`, `AFMIQE` and `AFSQE` using the NLL cost with `AFFB` to drive the adaptive ECDF engine. An additional variant comes from using the latter combination with the additional truncation step using `AFFB*` (see Algorithm 3), which is referred to as `AFSQE*`. A further 2 arise from using `QAF` and `AFSQE` with the squared error cost. Note that all the variations specified so far use AF procedures, but the fixed-weight procedures `EWSA(w)`, `DUMIQE(`ω`)`, `QEWA(`ω`)` are also tested, where the named parameter corresponds to the weight parameter for that procedure. The weights are chosen to span a range of sensible choices based on a combination of experimentation and recommendations in the original papers.

Figures 6.1 to 6.5 show individual runs on each of the slow drifting scenarios for all the procedures considered for comparison. Of course, individual runs do not provide sufficient evidence to make strong claims about the behaviour or performance of the procedures, but instead, serve as an illustrative example to observe and understand the basic behaviour. The simulation length used here is $n = 2000$, with the learning rates for all AF procedures set to $\eta = 0.001$ using a squared error cost function unless otherwise specified. Each figure is discussed briefly below.

Figure 6.1 shows adaptive 0.1-quantile estimates from individual runs of `AFSA` alongside runs of its fixed-weight counterpart, the `EWSA(w)` procedure, for $w \in \{0.01, 0.1\}$. In both switch cases it is evident that $w = 0.01$ is set too small for the estimates to keep up. For both smooth cases, however, the `EWSA(0.1)` estimator is the most volatile. The adaptive nature of the `AFSA`

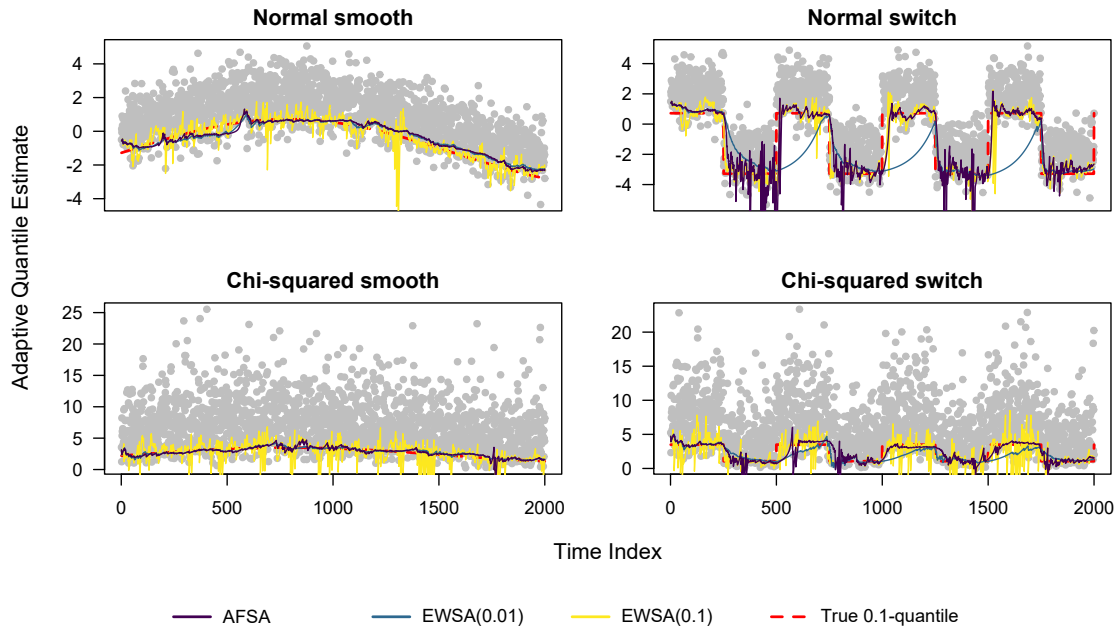

Figure 6.1: Adaptive estimates for the 0.1-quantile for an individual run of the `AFSA`, `EWSA(0.01)` and `EWSA(0.1)` streaming quantile estimators.

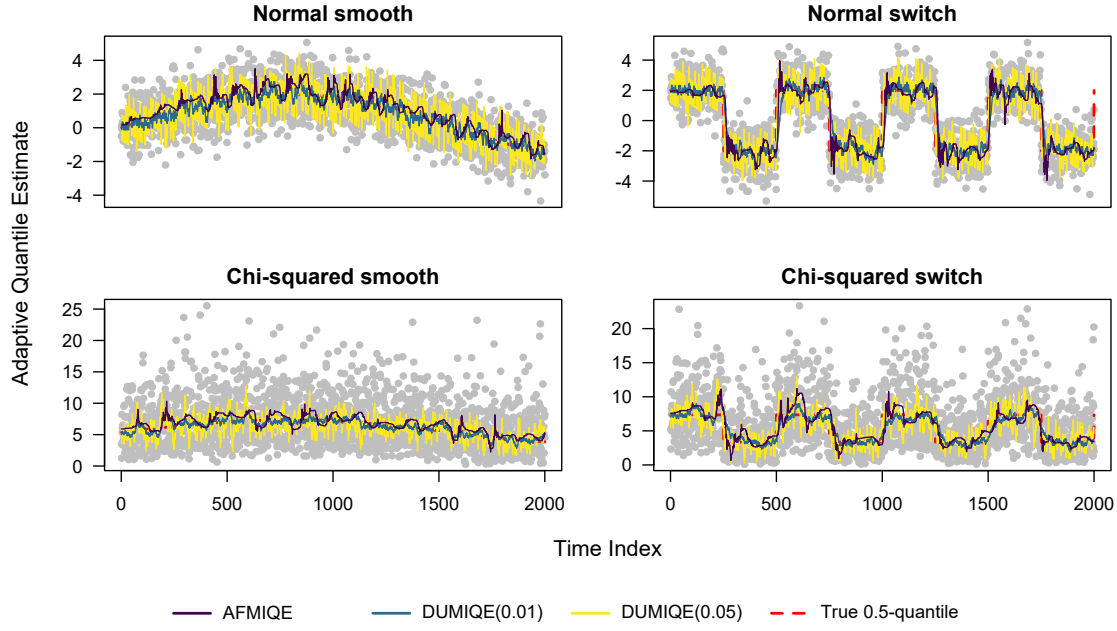

Figure 6.2: Adaptive estimates for the 0.5-quantile for an individual run of the `AFMIQE`, `DUMIQE(0.01)` and `DUMIQE(0.05)` streaming quantile estimators.

estimator is evident as it does not suffer from the same drawbacks in each of the four drifting scenarios.

Figure 6.2 shows adaptive median estimates from individual runs of `AFMIQE` alongside runs of its fixed-weight counterpart, the `DUMIQE(`ω`)` procedure, for $\omega \in \{0.01, 0.05\}$. In all four scenar-

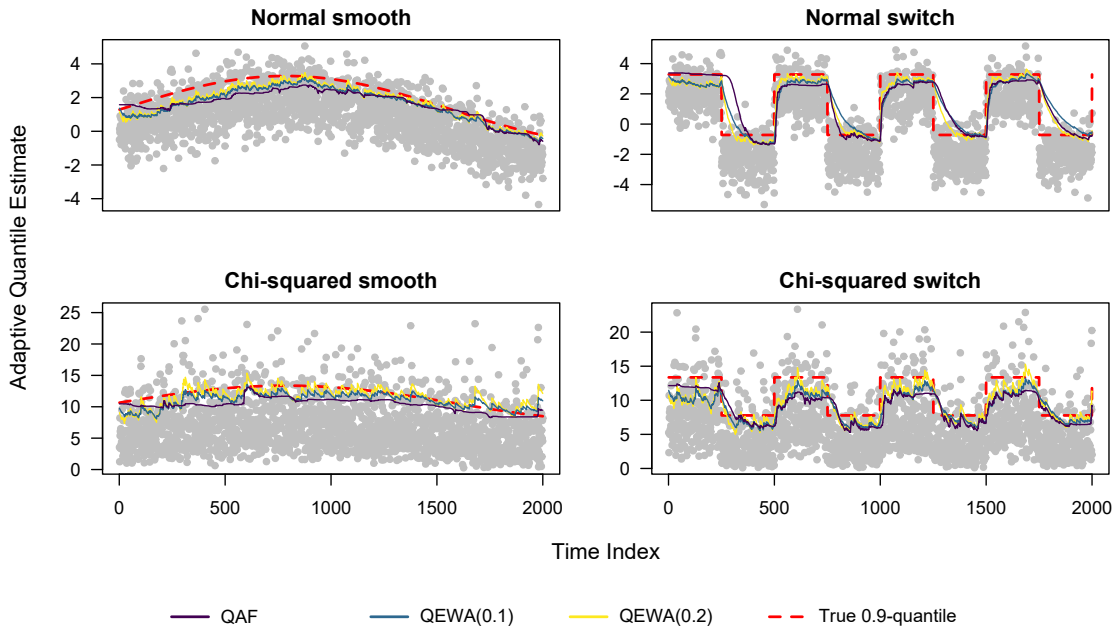

Figure 6.3: Adaptive estimates for the 0.9-quantile for an individual run of the `QAF`, `QEWA(0.1)` and `QEWA(0.2)` streaming quantile estimators.

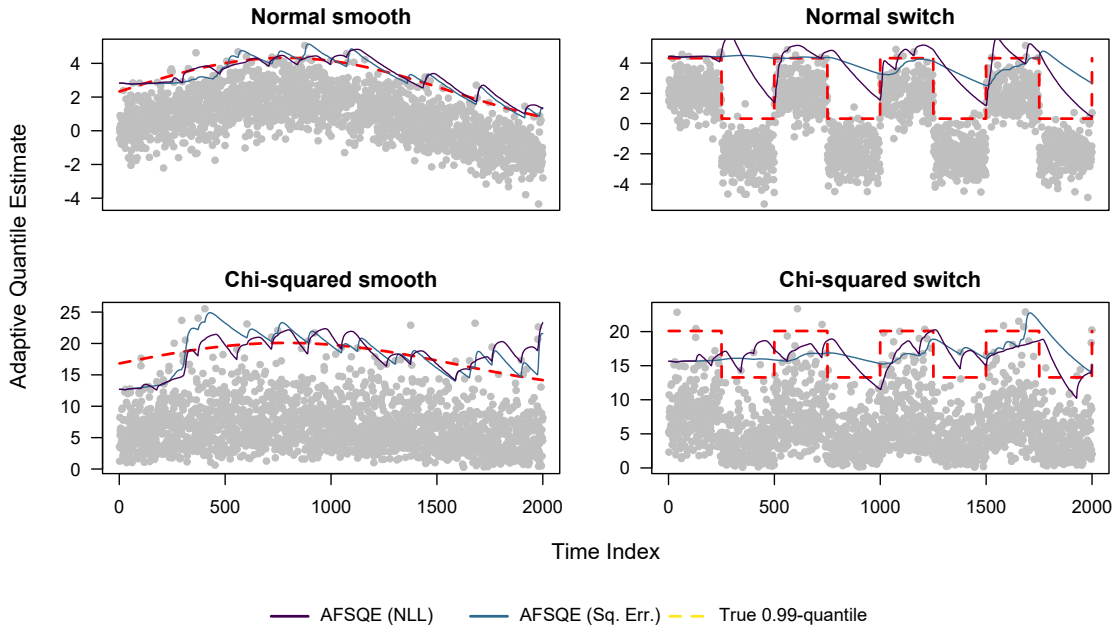

Figure 6.4: Adaptive estimates for the 0.99-quantile for an individual run of the `AFSQE` streaming quantile estimator on a selection of non-stationary simulated data environments. Here the `AFSQE` quantile estimates are shown for both a negative log-likelihood, and squared error cost function, as used in the adaptive ECDF estimation procedure.

ios, the estimates for `AFMIQE` and `DUMIQE(0.01)` are very similar, whereas the `DUMIQE(0.05)` estimates are consistently the most volatile. Even if the AF extension does not provide superior performance, it still removes the burden of choosing ω, which may be difficult in practice.

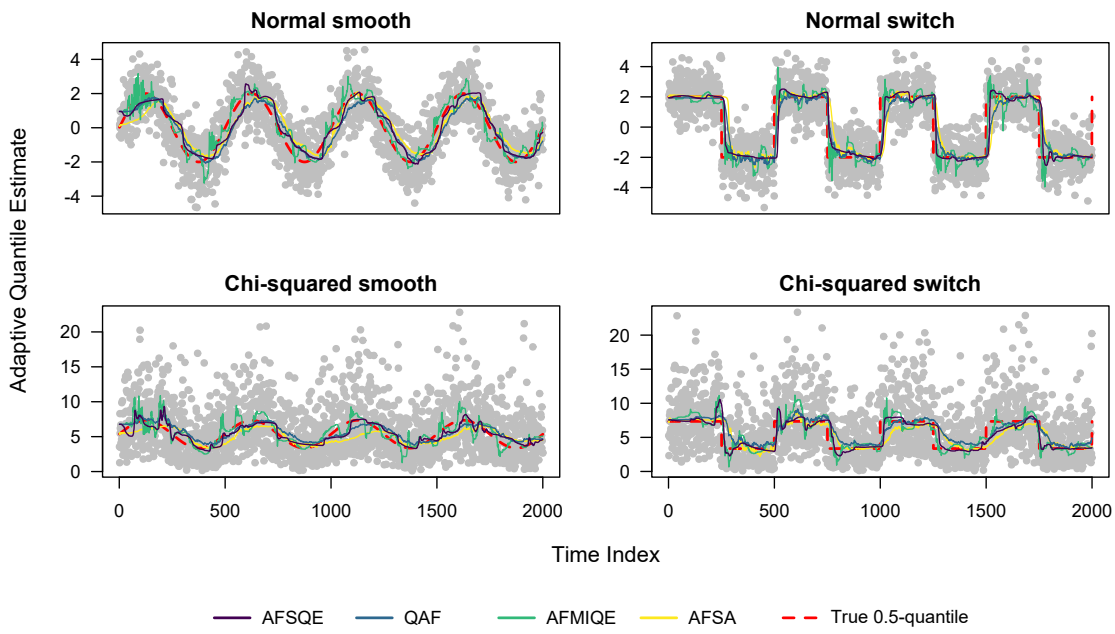

Figure 6.5: Adaptive estimates for the 0.5-quantile for individual runs of the `AFSQE`, `QAF`, `AFMIQE`, and `AFSA` streaming quantile estimators on a selection of non-stationary simulated data environments.

Figure 6.3 shows adaptive 0.9-quantile estimates from individual runs of `QAF` alongside runs of its fixed-weight counterpart, the `QEWA`(ω) procedure, for $\omega \in \{0.1, 0.2\}$. Interestingly, while the estimates for `QAF` closely resemble those of both `QEWA` runs, it appears that the QAF estimates are the further from the true quantile values. Of all the AF-extended streaming quantile estimation procedures, it does appear that `QAF` may offer the least benefit to performance. This could be due to the base `QEWA` procedure relying on two separate weights which the AF extension is not able to fully account for.

Figure 6.4 shows adaptive 0.99-quantile estimates from individual runs of `AFSQE`. As this method is novel there are is no fixed-weight counterpart to show alongside. Instead, this figure shows two distinct runs, one for each of the cost functions considered in the `AFFB` procedure that drives the forgetting factors, as discussed in Section 5.1.1. At first glance, the estimator may appear to be performing particularly poorly, though it is important to realise these are estimates for an extreme quantile, which is especially challenging, particularly for the chi-squared distribution where this quantile is the lower-density tail. In fact, all the streaming quantile estimation procedures explored struggle with extreme quantiles, which will be reflected in the performance results covered in the next section. Notice the paths are considerably less

smooth than for the figures with less extreme quantiles, which is a consequence of infrequent, staggered updates. Regardless of the cost function used, the switch cases seem to prove more challenging, though the NLL cost seems to have the edge in terms of adapting to the abrupt changes.

Figure 6.5 shows adaptive median estimates from individual runs for each of the adaptive streaming quantile estimators `AFSQE`, `QAF`, `AFMIQE`, and `AFSA`. For the same learning rate ($\eta = 0.001$), `AFMIQE` is noticeably more volatile than the others across all simulations. In the switch cases, `AFSA` appears to be the slowest to adapt to new regimes. From these simulations alone it is difficult to visually determine if the estimates from both `AFSQE` and `QAF` differ significantly.

6.4.1 Results

Tables 6.2 to 6.9 display the RMSE for each procedure tested when estimating q-quantiles for $q \in \{0.5, 0.7, 0.9, 0.99\}$, where each table corresponds to one of the specific non-stationary simulation environments described above. The lowest RMSE for a specific q-quantile among all methods and variants is emphasised in bold text. The results for each table are summarised in the paragraphs below, with overall results and recommendations summarised at the end of this section.

Looking at Table 6.2 for the Normal Smooth Fast case, the following observations are worthy of note. As is to be expected, for every method, the RMSE increases as q increases, and this holds true for all the simulation environments. In this simulation environment, while `AFSA` shows little variability with respect to the choice of learning rate, the RMSE is roughly double that of its fixed weight counterpart `EWSA` for most parameter values. This is unusual compared to most other scenarios where `AFSA` is usually as good, if not better than `EWSA`. The RMSE using the squared error variant of `AFSQE` for the 0.99-quantile is unusually low, being almost half of that of every other value. It is peculiar that, except for $\eta = 10^{-5}$, the quantile estimates for $q \neq 0.99$ are very similar, yet somehow $\eta = 0.01$ gives an exceptionally low RMSE for $q = 0.99$. Despite having the lowest RMSE for $q \in \{0.5, 0.7\}$ by a sizeable margin, `QEWA(0.2)` has almost the worst performance for $q = 0.99$.

For Table 6.3 it is not unreasonable to expect similar results to that of Table 6.2 given the scenario is very similar, just with a slower rate of drift. The lowest RMSE values in this case are roughly half of those for the Normal Smooth Fast case. Here, `QEWA` performs best for

$q \neq 0.99$, though the optimal weights for each quantile are different, and smaller than for the faster drift case, which should be expected. For $q = 0.99$, `AFSQE` is best, but this time for the NLL cost instead of the squared error. In fact, the NLL cost is generally better than the squared error across all quantiles and learning rates here. Notice that for `AFSA` with $\eta = 0.01$, and `EWSA(0.2)` the 0.99-quantile error is many magnitudes larger than for other learning rate and weight choices, suggesting these procedures may be less reliable for estimating extreme quantiles, or at least, more care needs to be taken with control parameter configuration.

The results for the Normal Switch Fast case in Table 6.4 are strikingly different from the previous two scenarios. Here `AFSQE` using the squared error cost is best, but *only* performs adequately for $\eta = 0.01$, and otherwise the `AFSQE` NLL results are lower and show almost no variation for different learning rates. This is the first scenario where the temporal-adaptivity in `QAF` outperforms `QEWA` in most cases, as long as $\eta > 10^{-5}$. Again, `AFSA` and `EWSA` show spurious extraordinarily large RMSE values for $q = 0.99$ and the largest learning rate, or weight. For $q = 0.99$, `AFMIQE` achieves the lowest RMSE, and also outperforms its fixed weight counterpart, `DUMIQE`, across the board. Notice the best RMSE is always achieved by an AF method here, using the largest learning rate $\eta = 0.01$, but despite the fast abrupt drift, results for the AF methods with $\eta < 0.01$ are still competitive.

Again, `AFSQE` with the squared error cost is best for the Normal Switch Slow case, as shown in Table 6.5. In this instance, the quantile estimates are still sensitive to the learning rate, but slightly less so than for the Normal Switch Fast case. Yet again, the NLL cost does not exhibit the same weakness, except for where it finally shows some slight sensitivity to the choice of η, when $q = 0.99$. For $q = 0.99$, `EWSA(0.05)` is a clear winner, but the results are not stable for different weights, whereas `AFSA` demonstrates stability across all learning rates. Here, choosing `QEWA(`ω`)` for $\omega \geq 0.1$ will net better results than for `QAF`, but otherwise `QAF` has a slight advantage.

Now considering the Chi-squared Smooth Fast scenario in Table 6.6, `QEWA` is best for the non-extreme quantiles, and for $w \neq 0.01$ always outperforms `QAF`. That `AFMIQE` is best for $q = 0.99$ with $\eta = 10^{-5}$ is surprising, since this is the only best result for such a small learning rate. It is evident from these results, and those of the previously discussed scenarios, that `AFMIQE` works best for smaller learning rates compared to the other AF methods, and generally unlike the others, is worse for $\eta = 0.01$. This is likely due to the multiplicative update of fixed size as

opposed to the other AF methods. The NLL cost for `AFSQE` is always better than the squared error cost here, and this may be due to the distribution being heavy tailed, which a squared error would penalise more.

The results in Table 6.7 for the Chi-squared Smooth Slow environment deserve the following remarks. Among all `EWSA` variants, it is striking that the lowest weight, $w = 0.01$, has almost the best performance across all quantiles, even considering the slower drift for this environment. However, using $w \neq 0.01$ is costly, especially when `AFSA` for $\eta > 0.01$ produces competitive and consistent estimation. For the non-extreme quantiles, `QEWA(0.05)` performs best, but notice it has the largest RMSE of all method variants for $q = 0.99$. Better extreme performance is achieved using `QAF`, which is almost as effective for $q \in \{0.5, 0.7\}$, but strangely struggles for $q = 0.9$. So far, the relationship between RMSE results for `QEWA` and `QAF` are the least predictable of all the AF and corresponding fixed weight method pairs. As in the Chi-squared Smooth Fast case, the NLL cost is best for the `AFSQE` procedure, where the RMSE is often less than half that of the RMSE for the squared error cost.

The Chi-squared Switch fast environment, with RMSE results in Table 6.8, provides the only dominant RMSE for `DUMIQE`, where $\omega = 0.2$. This choice of ω outperforms all `AFMIQE` variants, except for $q = 0.99$, where `AFMIQE` demonstrates a clear advantage. Notably, the `AFSA` and `EWSA` methods have the worst extreme quantile performance. This is likely due to the neighbourhood size being based on the interquartile range (see Algorithm 5), which may not be appropriate for such heavy tailed data. For the first time, the extreme quantile performance for `QEWA` and `QAF` are among the best. Usually `AFSQE` gives one of the lowest RMSE values for the extreme quantile, but that is not the case here, for either cost function.

Worthy of note among the results for the Chi-squared Switch Slow case in Table 6.9, is that `QEWA(0.2)` performs best for $q \neq 0.99$, which is a large weight value given the slower drift environment. Despite `QAF` having the best value for $q = 0.99$ it is clearly worse than `QEWA(`ω`)` for $\omega \neq 0.01$. Unlike the case for Chi-squared Switch Fast where both cost functions perform similarly for `AFSQE`, the NLL cost is better here. For every slow simulation environment before this, the best RMSE for $q = 0.99$ has been lower than that of the best RMSE for the corresponding fast simulation environment, but here they are within two thousandths of each other. This suggests the Chi-squared Switch case is particularly challenging for extreme quantile estimation, irrelevant of the rate of drift.

Choosing a procedure for the task of streaming quantile estimation purely based on it achieving the lowest RMSE in simulation is unwise. In streaming contexts, it is often the case that with the data evolving over time, the optimal procedure at any point in time will also change. Perhaps more important, is the property of doing a reasonably good job, most of the time. Or put another way, rarely displaying poor estimation performance. This often favours adaptive procedures as opposed to those using a fixed weight, for which the latter tend to be more vulnerable to changes in drift patterns, particularly if the rate of drift is also changing over time. Of course, if some knowledge of the underlying data and drift dynamics are known and not expected to change over time, then if possible, it may be worth using a fixed-weight method and optimising its control parameter from an initial burn-in phase. However, this is seldom the case.

Ultimately the choice of procedure should depend on which quantile is being estimated and if anything is known about the underlying data generating mechanism and drift dynamics. Assuming only the former is known, then using one of the `AF` procedures will help to mitigate the risk of incorrectly choosing control parameters. Additionally, the value of q plays an important part as some methods are particularly unreliable for extreme quantiles such as $q = 0.99$ (e.g. `AFSA` and `EWSA`). Overall, if one method must be chosen, `AFSQE` is a good choice based on it almost always demonstrating a RMSE within the best four, and never being the worst. It also demonstrates considerable resilience to the choice of learning rate, with the NLL cost being a slightly better safeguard against a poor choice of learning rate than the squared error cost. Interestingly this difference in the cost function was not apparent in the simulations results for `AFFB` in Table 5.3, where both costs appear equally resilient. There are situations where the squared error cost does manage to achieve slightly better performance overall (particularly in the Normal Switch cases), and so this decisive trade-off should be assessed based on the task. If using `AFSQE` with a squared error cost, then the best results are achieved using $\eta = 0.01$ for the lower quantiles, but do negatively impact estimation for $q = 0.99$. It also appears to perform poorly for heavy-tailed data. With the NLL cost, `AFSQE` is reliable for all quantiles, and using $\eta = 0.001$ is recommended, though the penalty of choosing a different learning rate is not substantial. It is worth remarking that the RMSE values for `AFSQE*` are generally the same, or larger, than for `AFSQE`, where both are using the NLL cost. Even though the switch cases provide temporary periods of stationarity, perhaps this is due

to the duration of the stationary periods being too short for the additional truncation step to benefit estimation. Hence, overall the NLL-based variant of `AFSQE` is recommended as a 'one-size-fits-all' procedure. However, there are some basic caveats. Specifically, `AFSQE` is rarely the best method when estimating extreme quantiles such as $q = 0.99$, and when the data is heavily tailed as in the Chi-squared cases. In the presence of heavy tailed data, the results suggest that `QEWA` performs consistently well, provided a suitable value for the control parameter can be determined.

There are several occurrences where `EWSA` performs within the best two, but this is often only for a specific choice of w, with other choices performing drastically worse. However, `AFSA` almost always achieves similar, or adequate, performance and is much more resilient to the choice of learning rate, though using $\eta < 0.01$ is advised. The trade-off of optimal performance for almost always adequate performance is consistently clear for these methods across all simulation environments. For both, there are also occasions where the RMSE for the extreme quantile estimate is significantly larger than for other methods. This often happens when the learning rate or weight is too high, and so avoiding $\eta = 0.01$ and $w = 0.2$ is recommended when estimating extreme quantiles.

Despite one occurrence where `DUMIQE(0.2)` resulted in the lowest RMSE (for Chi-squared Switch Fast with $q = 0.9$), `AFMIQE` always performs better or on-par. Hence there is little reason to justify the use of `DUMIQE` if `AFMIQE` is available. If $q \neq 0.99$, using $\eta = 0.001$ almost always gives the lowest RMSE for `AFMIQE`, otherwise a smaller learning rate should be used.

Of all AF extended procedures, `QAF` benefits the least. It does not particularly enhance the reliability of the procedure as the performance is often substantially different to the corresponding results for `QEWA`, whether higher or lower. In the Chi-squared Switch cases `QAF` does achieve the best results for the $q = 0.99$-quantile, but the corresponding `QEWA` estimates are on par. This uncertainty makes it difficult to justify the use of `QAF` over `QEWA`, especially as the latter generally performs very well. If a non-adaptive, fixed weight method must be used, then `QEWA(0.1)` strikes a good balance and often achieves a competitive RMSE among all methods.

While Tables 6.2 to 6.9 are comprehensive for the simulations covered and the RMSE results help compare methods, it is difficult to interpret how an RMSE value will affect quantile estimation performance in practice. This research does not focus on developing an alternative

measure to address this issue, but such a measure ought to take into account the spread of the data and the specific quantile being estimated.

As a reminder, all streaming quantile estimation procedures tested have constant storage complexity and the update step for each procedure has constant computational complexity.

Table 6.2: Root mean squared estimation error for several streaming quantile estimation procedures run on data simulated from the Normal Smooth Fast environment.

			Quantile			
Procedure	Cost function	Learning rate	0.5	0.7	0.9	0.99
AFSA	NLL	0.01	0.600	0.662	1.180	1.812
		0.001	0.714	0.823	1.435	1.901
		0.0001	0.746	0.866	1.462	1.902
		0.00001	0.768	0.888	1.474	1.901
EWSA(0.01)			0.974	1.091	1.562	1.884
EWSA(0.05)			0.349	0.376	0.767	1.892
EWSA(0.1)			0.322	0.334	**0.422**	1.795
EWSA(0.2)			0.449	0.450	0.448	1.479
AFMIQE	NLL	0.01	1.412	1.131	0.801	1.226
		0.001	1.391	0.922	1.096	1.609
		0.0001	1.412	1.086	1.168	1.613
		0.00001	1.413	1.120	1.325	1.888
DUMIQE(0.01)			1.414	1.453	1.661	1.909
DUMIQE(0.05)			1.413	1.126	1.070	1.875
DUMIQE(0.1)			1.412	1.053	0.752	1.725
DUMIQE(0.2)			1.412	1.053	0.599	1.415
QAF	Squared Error	0.01	0.549	0.596	0.976	1.989
		0.001	0.534	0.637	1.378	1.964
		0.0001	0.591	0.721	1.415	1.949
		0.00001	0.609	0.742	1.427	1.951
QEWA(0.01)			1.262	1.356	1.522	1.932
QEWA(0.05)			0.524	0.674	1.417	1.960
QEWA(0.1)			0.308	0.381	1.091	1.960
QEWA(0.2)			**0.264**	**0.278**	0.618	1.946
AFSQE	NLL	0.01	0.481	0.504	0.619	1.124
		0.001	0.428	0.459	0.749	1.542
		0.0001	0.513	0.547	0.791	1.779
		0.00001	0.527	0.572	0.896	1.920
AFSQE	Squared Error	0.01	0.442	0.462	0.526	**0.724**
		0.001	0.445	0.463	0.541	1.898
		0.0001	0.474	0.463	0.843	1.925
		0.00001	1.466	1.668	1.780	1.943
AFSQE*	NLL	0.01	0.498	0.553	0.850	1.426
		0.001	0.429	0.468	0.664	1.706
		0.0001	0.513	0.547	0.791	1.814
		0.00001	0.527	0.572	0.897	1.919

Table 6.3: RMSE performance for several streaming quantile estimation procedures in the Normal Smooth Slow environment.

			Quantile			
Procedure	Cost function	Learning rate	0.5	0.7	0.9	0.99
AFSA	NLL	0.01	0.199	0.211	0.282	2.205
		0.001	0.250	0.257	0.313	0.788
		0.0001	0.263	0.272	0.313	0.557
		0.00001	0.288	0.298	0.340	0.504
EWSA(0.01)			0.272	0.277	0.300	0.453
EWSA(0.05)			0.219	0.235	0.333	0.998
EWSA(0.1)			0.329	0.353	0.629	1.866
EWSA(0.2)			1.507	0.928	1.726	3.766
AFMIQE	NLL	0.01	1.402	1.059	0.607	0.639
		0.001	1.404	0.938	0.634	0.983
		0.0001	1.380	0.836	0.854	1.457
		0.00001	1.406	0.999	0.848	1.765
DUMIQE(0.01)			1.411	1.114	1.063	1.838
DUMIQE(0.05)			1.405	0.985	0.499	1.157
DUMIQE(0.1)			1.405	0.988	0.451	0.690
DUMIQE(0.2)			1.405	1.017	0.475	0.450
QAF	Squared Error	0.01	0.242	0.300	0.855	1.973
		0.001	0.383	0.442	0.823	2.132
		0.0001	0.376	0.436	1.181	2.109
		0.00001	0.431	0.525	1.214	2.097
QEWA(0.01)			0.517	0.632	1.184	2.268
QEWA(0.05)			**0.154**	0.195	0.447	2.491
QEWA(0.1)			0.168	**0.179**	0.267	2.469
QEWA(0.2)			0.232	0.226	**0.225**	2.337
AFSQE	NLL	0.01	0.215	0.224	0.281	**0.427**
		0.001	0.262	0.284	0.397	0.771
		0.0001	0.300	0.301	0.457	1.391
		0.00001	0.357	0.366	0.501	1.363
AFSQE	Squared Error	0.01	0.440	0.459	0.523	0.626
		0.001	0.440	0.459	0.517	0.579
		0.0001	0.481	0.496	0.445	0.585
		0.00001	0.380	0.379	0.417	1.893
AFSQE*	NLL	0.01	0.307	0.328	0.458	0.763
		0.001	0.288	0.293	0.496	1.258
		0.0001	0.287	0.287	0.447	1.202
		0.00001	0.357	0.366	0.500	1.319

Table 6.4: RMSE performance for several streaming quantile estimation procedures in the Normal Switch Fast environment.

			Quantile			
Procedure	Cost function	Learning rate	0.5	0.7	0.9	0.99
AFSA	NLL	0.01	1.612	1.672	2.113	4.501
		0.001	1.368	1.421	1.909	2.712
		0.0001	1.358	1.416	1.885	2.627
		0.00001	1.398	1.659	2.521	2.614
EWSA(0.01)			2.054	2.295	2.492	2.605
EWSA(0.05)			1.804	1.823	1.946	2.540
EWSA(0.1)			1.366	1.397	1.628	2.551
EWSA(0.2)			1.514	1.425	2.227	4.002
AFMIQE	NLL	0.01	2.000	1.900	2.102	**2.224**
		0.001	2.000	2.000	2.387	2.444
		0.0001	2.000	1.997	2.476	2.517
		0.00001	2.000	2.028	2.524	2.567
DUMIQE(0.01)			2.008	2.331	2.523	2.613
DUMIQE(0.05)			2.000	2.238	2.472	2.637
DUMIQE(0.1)			2.000	2.083	2.346	2.657
DUMIQE(0.2)			2.000	1.887	2.083	2.709
QAF	Squared Error	0.01	1.301	1.577	2.533	2.595
		0.001	1.260	1.431	2.506	2.509
		0.0001	1.275	1.446	2.502	2.460
		0.00001	1.413	2.282	2.501	2.426
QEWA(0.01)			2.021	2.295	2.351	2.482
QEWA(0.05)			1.866	2.232	2.464	2.533
QEWA(0.1)			1.604	1.942	2.476	2.557
QEWA(0.2)			1.186	1.392	2.418	2.586
AFSQE	NLL	0.01	1.719	1.830	2.248	2.724
		0.001	1.510	1.603	2.188	2.675
		0.0001	1.506	1.606	2.181	2.640
		0.00001	1.565	1.682	2.403	2.622
AFSQE	Squared Error	0.01	**0.837**	**0.897**	**1.591**	2.371
		0.001	2.341	2.551	2.635	2.467
		0.0001	2.039	2.357	2.543	2.631
		0.00001	2.011	2.342	2.534	2.615
AFSQE*	NLL	0.01	1.626	1.648	2.179	2.903
		0.001	1.510	1.603	2.187	2.802
		0.0001	1.506	1.606	2.183	2.597
		0.00001	1.565	1.682	2.549	2.617

Table 6.5: RMSE performance for several streaming quantile estimation procedures in the Normal Switch Slow environment.

			Quantile			
Procedure	Cost function	Learning rate	0.5	0.7	0.9	0.99
AFSA	NLL	0.01	0.859	0.891	1.174	2.730
		0.001	1.005	0.994	1.583	2.405
		0.0001	0.926	0.952	1.075	2.327
		0.00001	0.937	0.966	1.114	2.503
EWSA(0.01)			1.858	1.869	1.969	2.235
EWSA(0.05)			0.913	0.917	0.984	**1.615**
EWSA(0.1)			0.686	0.698	0.910	2.202
EWSA(0.2)			1.408	1.046	1.867	3.880
AFMIQE	NLL	0.01	1.995	1.711	1.444	2.015
		0.001	1.994	1.534	1.799	2.273
		0.0001	1.999	1.720	2.029	2.468
		0.00001	2.000	1.764	2.097	2.646
DUMIQE(0.01)			2.004	2.253	2.472	2.626
DUMIQE(0.05)			1.998	1.838	1.953	2.604
DUMIQE(0.1)			1.998	1.713	1.606	2.522
DUMIQE(0.2)			1.998	1.666	1.341	2.334
QAF	Squared Error	0.01	0.807	0.889	1.910	2.908
		0.001	0.921	1.051	2.137	2.874
		0.0001	0.941	1.069	2.164	2.872
		0.00001	0.999	1.141	2.383	2.883
QEWA(0.01)			1.881	2.099	2.343	2.786
QEWA(0.05)			1.115	1.291	2.276	2.805
QEWA(0.1)			0.788	0.919	1.848	2.787
QEWA(0.2)			0.572	0.662	1.237	2.749
AFSQE	NLL	0.01	0.897	0.941	1.147	1.916
		0.001	0.987	0.996	1.137	2.439
		0.0001	0.939	1.003	1.324	2.678
		0.00001	0.966	1.028	1.384	2.645
AFSQE	Squared Error	0.01	**0.543**	**0.570**	**0.695**	2.421
		0.001	0.572	0.597	1.363	2.540
		0.0001	0.627	0.643	3.309	2.655
		0.00001	2.381	2.585	2.547	2.689
AFSQE*	NLL	0.01	1.097	1.136	1.549	2.457
		0.001	0.987	0.997	1.151	2.708
		0.0001	0.940	1.004	1.323	2.663
		0.00001	0.966	1.029	1.385	2.607

Table 6.6: RMSE performance for several streaming quantile estimation procedures in the Chi-Squared Smooth Fast environment.

			Quantile			
Procedure	Cost function	Learning rate	0.5	0.7	0.9	0.99
AFSA	NLL	0.01	0.840	1.106	1.728	4.499
		0.001	0.920	1.188	1.685	2.932
		0.0001	0.811	1.042	1.546	2.757
		0.00001	0.817	1.023	1.479	2.616
EWSA(0.01)			1.020	1.195	1.581	2.659
EWSA(0.05)			0.960	1.258	1.944	2.786
EWSA(0.1)			1.564	2.031	2.910	3.606
EWSA(0.2)			2.998	3.556	4.716	5.670
AFMIQE	NLL	0.01	0.883	1.100	1.565	3.546
		0.001	0.745	0.920	1.482	3.100
		0.0001	0.834	1.029	1.649	2.703
		0.00001	0.884	1.085	1.739	**2.555**
DUMIQE(0.01)			1.214	1.421	1.943	2.628
DUMIQE(0.05)			0.668	0.837	1.282	2.677
DUMIQE(0.1)			0.783	0.985	1.313	2.651
DUMIQE(0.2)			1.086	1.351	1.767	2.704
QAF	Squared Error	0.01	0.680	0.898	1.763	3.323
		0.001	0.789	1.031	1.875	3.198
		0.0001	0.895	1.160	1.945	2.997
		0.00001	0.927	1.193	1.964	2.982
QEWA(0.01)			1.269	1.551	2.346	3.107
QEWA(0.05)			0.623	0.846	1.761	3.220
QEWA(0.1)			**0.573**	**0.754**	1.413	3.197
QEWA(0.2)			0.737	0.938	**1.273**	3.149
AFSQE	NLL	0.01	0.842	1.046	1.615	2.887
		0.001	0.869	1.104	1.787	2.751
		0.0001	0.902	1.179	1.967	2.730
		0.00001	0.934	1.228	2.063	2.602
AFSQE	Squared Error	0.01	1.530	1.770	2.355	3.400
		0.001	1.528	1.768	2.339	3.135
		0.0001	1.435	1.650	2.183	2.649
		0.00001	1.520	1.732	2.063	2.632
AFSQE*	NLL	0.01	0.958	1.228	2.097	3.684
		0.001	0.879	1.121	1.835	3.797
		0.0001	0.903	1.178	1.967	3.154
		0.00001	0.934	1.230	2.067	2.654

Table 6.7: RMSE performance for several streaming quantile estimation procedures in the Chi-Squared Smooth Slow environment.

			Quantile			
Procedure	Cost function	Learning rate	0.5	0.7	0.9	0.99
AFSA	NLL	0.01	0.625	0.879	1.587	4.450
		0.001	0.496	0.667	0.970	2.114
		0.0001	0.529	0.683	0.956	1.894
		0.00001	0.489	0.606	0.962	2.025
EWSA(0.01)			0.417	0.551	0.894	**1.740**
EWSA(0.05)			0.934	1.246	1.933	2.367
EWSA(0.1)			1.591	2.002	2.861	3.376
EWSA(0.2)			3.018	3.597	4.767	5.592
AFMIQE	NLL	0.01	0.849	1.064	1.507	3.612
		0.001	0.490	0.618	0.858	2.453
		0.0001	0.511	0.613	1.006	2.214
		0.00001	0.563	0.695	1.223	2.574
DUMIQE(0.01)			0.471	0.569	1.008	2.543
DUMIQE(0.05)			0.545	0.695	0.868	1.779
DUMIQE(0.1)			0.761	0.968	1.208	1.761
DUMIQE(0.2)			1.089	1.351	1.735	2.278
QAF	Squared Error	0.01	0.511	0.654	1.368	3.195
		0.001	0.511	0.704	1.397	3.558
		0.0001	0.559	0.747	1.613	3.381
		0.00001	0.656	0.890	1.763	3.490
QEWA(0.01)			0.595	0.769	1.753	4.914
QEWA(0.05)			**0.398**	**0.512**	**0.799**	6.521
QEWA(0.1)			0.525	0.663	0.803	6.498
QEWA(0.2)			0.736	0.926	1.059	5.960
AFSQE	NLL	0.01	0.561	0.673	0.995	2.190
		0.001	0.517	0.650	1.046	2.257
		0.0001	0.539	0.717	1.199	2.693
		0.00001	0.567	0.741	1.247	2.645
AFSQE	Squared Error	0.01	1.542	1.792	2.385	3.370
		0.001	1.542	1.793	2.347	2.933
		0.0001	1.424	1.636	1.894	2.089
		0.00001	0.697	0.752	0.891	2.637
AFSQE*	NLL	0.01	0.645	0.828	1.512	3.311
		0.001	0.605	0.767	1.375	3.683
		0.0001	0.570	0.679	1.219	2.844
		0.00001	0.564	0.737	1.230	2.636

Table 6.8: RMSE performance for several streaming quantile estimation procedures in the Chi-Squared Switch Fast environment.

			Quantile			
Procedure	Cost function	Learning rate	0.5	0.7	0.9	0.99
AFSA	NLL	0.01	2.002	2.439	3.277	6.287
		0.001	1.815	2.220	3.019	4.608
		0.0001	1.784	2.192	2.965	4.011
		0.00001	1.950	2.342	2.948	3.895
EWSA(0.01)			1.990	2.332	2.973	4.063
EWSA(0.05)			1.801	2.155	2.927	4.682
EWSA(0.1)			1.908	2.429	3.358	5.360
EWSA(0.2)			3.076	3.653	4.929	7.132
AFMIQE	NLL	0.01	1.831	2.201	2.985	4.459
		0.001	1.772	2.129	3.036	3.652
		0.0001	1.793	2.144	2.952	3.632
		0.00001	2.004	2.348	2.937	3.577
DUMIQE(0.01)			2.000	2.345	2.958	3.841
DUMIQE(0.05)			1.913	2.240	2.964	4.045
DUMIQE(0.1)			1.728	2.043	2.886	4.259
DUMIQE(0.2)			1.578	1.892	**2.780**	4.633
QAF	Squared Error	0.01	1.663	2.071	3.121	3.575
		0.001	1.665	2.080	2.984	**3.514**
		0.0001	1.683	2.131	2.967	3.678
		0.00001	1.996	2.207	2.928	3.800
QEWA(0.01)			1.992	2.368	2.888	3.814
QEWA(0.05)			1.889	2.272	2.932	3.545
QEWA(0.1)			1.671	2.076	2.953	3.598
QEWA(0.2)			**1.371**	**1.767**	2.915	3.802
AFSQE	NLL	0.01	2.124	2.589	3.488	4.802
		0.001	2.042	2.567	3.292	4.161
		0.0001	2.051	2.593	3.261	3.935
		0.00001	2.096	2.603	2.962	3.736
AFSQE	Squared Error	0.01	1.715	2.032	2.879	5.037
		0.001	2.244	2.555	3.122	4.740
		0.0001	2.039	2.388	3.050	3.963
		0.00001	2.015	2.366	3.011	3.739
AFSQE*	NLL	0.01	2.059	2.528	3.328	5.537
		0.001	2.042	2.567	3.219	4.733
		0.0001	2.051	2.593	3.123	4.729
		0.00001	2.095	2.618	2.949	3.746

Table 6.9: RMSE performance for several streaming quantile estimation procedures in the Chi-Squared Switch Slow environment.

			Quantile			
Procedure	Cost function	Learning rate	0.5	0.7	0.9	0.99
AFSA	NLL	0.01	1.187	1.479	2.273	5.116
		0.001	1.499	1.878	2.562	3.975
		0.0001	1.258	1.531	2.189	4.055
		0.00001	1.243	1.511	2.133	3.940
EWSA(0.01)			1.669	1.946	2.425	3.895
EWSA(0.05)			1.172	1.514	2.186	3.840
EWSA(0.1)			1.647	2.085	2.965	4.302
EWSA(0.2)			3.019	3.608	4.813	5.933
AFMIQE	NLL	0.01	1.136	1.390	1.949	3.991
		0.001	1.239	1.497	2.275	3.900
		0.0001	1.311	1.594	2.446	3.552
		0.00001	1.345	1.633	2.537	3.608
DUMIQE(0.01)			1.875	2.188	2.872	3.881
DUMIQE(0.05)			1.183	1.419	2.124	3.965
DUMIQE(0.1)			1.064	1.310	1.852	3.947
DUMIQE(0.2)			1.213	1.490	2.012	3.726
QAF	Squared Error	0.01	1.004	1.318	2.354	3.630
		0.001	1.207	1.549	2.718	**3.512**
		0.0001	1.288	1.638	2.768	3.552
		0.00001	1.335	1.696	2.808	3.551
QEWA(0.01)			1.895	2.243	2.852	3.793
QEWA(0.05)			1.171	1.490	2.600	3.521
QEWA(0.1)			0.933	1.194	2.202	3.547
QEWA(0.2)			**0.907**	**1.158**	**1.832**	3.683
AFSQE	NLL	0.01	1.208	1.492	2.199	3.975
		0.001	1.346	1.687	2.467	3.899
		0.0001	1.338	1.713	2.738	4.031
		0.00001	1.363	1.740	2.824	3.746
AFSQE	Squared Error	0.01	1.581	1.844	2.443	3.891
		0.001	1.590	1.856	2.442	3.876
		0.0001	1.508	1.775	3.064	3.936
		0.00001	2.124	2.422	2.976	3.741
AFSQE*	NLL	0.01	1.380	1.722	2.765	4.651
		0.001	1.349	1.684	2.485	5.156
		0.0001	1.339	1.713	2.738	4.260
		0.00001	1.363	1.741	2.824	3.813

6.5 Experiments on Real Data

This section explores the deployment of the `AFSQE` procedure on real data. As a fundamental statistical quantity, quantiles, and their estimates, have a part to play in many applications. Here, two brief examples are considered.

6.5.1 Adaptive Thresholding

Many applications in a continuous monitoring (Bodenham and Adams 2017) context require timely decision-making, where it is common to use a rule based on thresholding a score as used in the change detector in Section 3.4, which continuously monitors the evolving mean and standard deviation of KL divergence scores. Then a change is flagged if the current score exceeds more than a fixed number of standard deviations from the mean. A thresholding rule is also used in Section 4.3.5 where the network anomaly detector uses an extreme sample quantile computed over the full time range, such that any score exceeding that quantile is considered to be sufficiently anomalous.

The motivation for thresholding based on standard deviations originally stems from the properties of the standard normal distribution, where the 95% quantile is approximately 2 standard deviations from the mean. However, in practice, many of these score samples do not look normally distributed as they can often be highly skewed and multi-modal. If the intent of the rule is to alert for some small fixed proportion of events then instead of specifying a number of standard deviations as a proxy for an extreme quantile, it may be more appropriate to use the actual quantile. If a static quantile estimate computed from some initial set of scores is used, then this quantile estimate eventually becomes unreliable. To avoid this, an adaptive quantile estimate could be used from any of the methods in Section 6.3.

Returning to the example in Section 4.3.5, network anomaly scores ν and κ were introduced for detecting anomalous activity across a computer network. Figure 4.3 shows the network anomaly scores within 1-minute bins over a 3-day period for both the LANL and ICL networks, where it is evident the distribution of network scores is changing over time. Figure 6.6 shows the same network anomaly scores, but here the black solid line corresponds to the 0.995-quantile. Notice how for both networks, initially the quantile estimates are large due to some early outliers, then gradually decrease as the thousands of observations lower than the quantile

are observed. Then when encountering an additional pair of outliers, that exceed the current quantile estimate, the subsequent quantile estimates begin to increase accordingly.

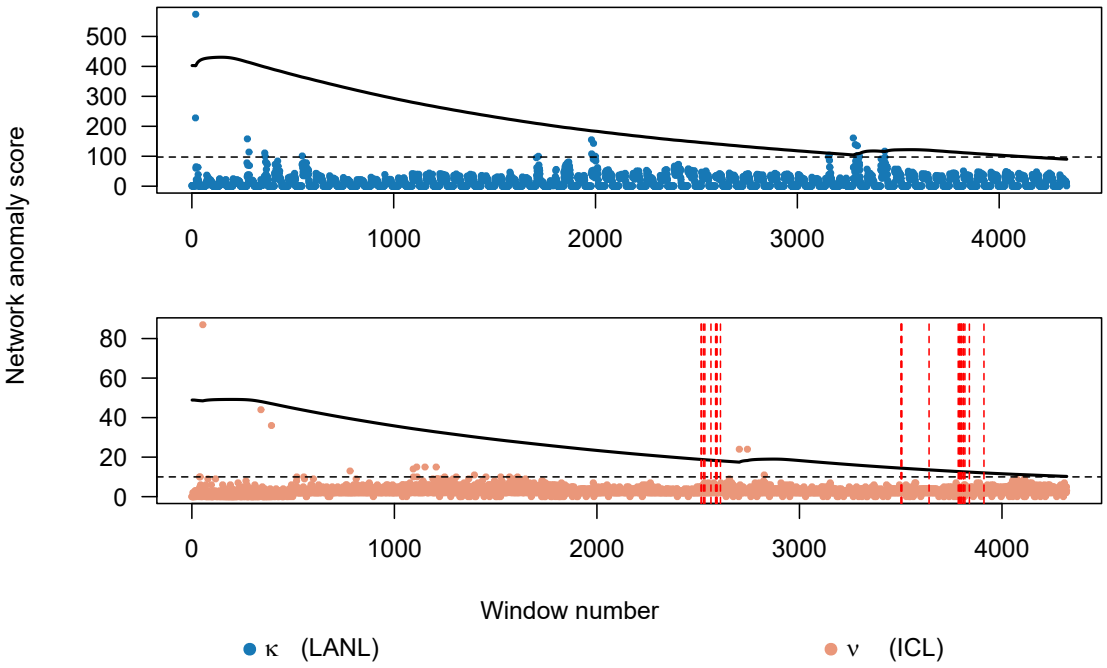

Figure 6.6: Network anomaly scores ν and κ for the ICL (top) and LANL (bottom) networks where the adaptive thresholds shown as solid black lines correspond to the adaptive 0.995-quantile estimates of the network anomaly scores. The horizontal dotted black lines correspond to the static 0.995-quantile estimates over all windows. The vertical red dotted lines (bottom) correspond to red team authentication events.

6.5.2 Concept Drift Detection

This demonstration closely follows and attempts to reproduce the experiment in Hammer, Yazidi, and Rue (2019), instead using the `AFSQE` procedure in place of `QEWA`.

As discussed in Section 2.5, the statistical properties of data streams generally change over time, for example the joint relationship between temperature and interarrival times in Section 3.3, or the correlation structures in Chapter 4. If the purpose of modelling is for prediction, or forecasting, then it is common for non-adaptive models to degrade over time. There are many strategies to overcoming this challenge (Gama et al. 2014), but one approach is to monitor the prediction error of a model and define a rule for when the model should be retrained.

The SML2020 dataset contains HVAC sensor measurements recorded every 15 minutes over 40 days, for a number of climate related indoor and outdoor variables such as temperature,

carbon dioxide levels, humidity, and precipitation (Zamora-Martínez et al. 2014). Note the LANL HVAC system data used in Section 3.3.2, was not available at the time of research into streaming quantile estimation. The focus of this demonstration is not model building, but rather to see how streaming quantile estimation can be used to deploy an automatic concept drift detection and model retraining procedure.

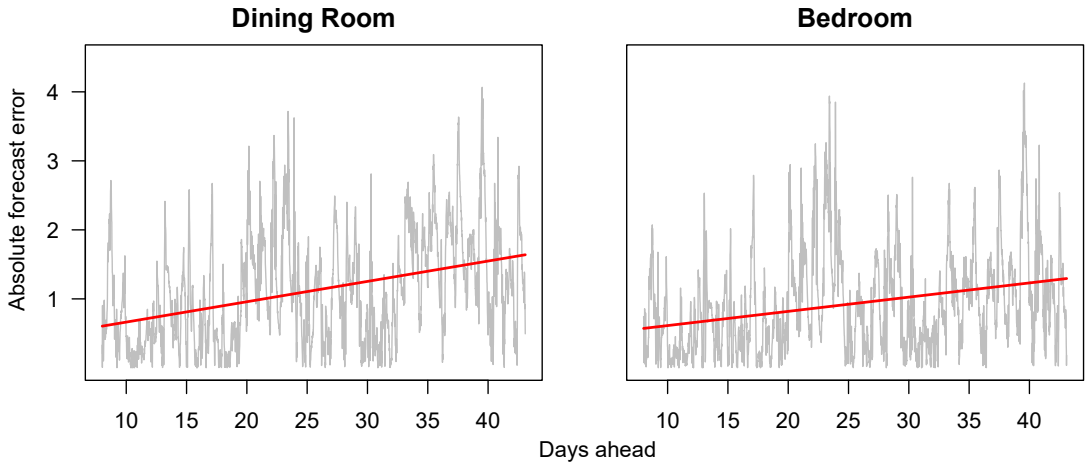

Figure 6.7: Absolute forecasting error of predicted temperature 15-minutes into the future using a model initially trained on the first 8 days of data. The left and right panels correspond to the dining room and bedroom respectively, where the red solid lines show the increasing linear trend of the absolute forecasting error over time.

Temperature readings are available for a bedroom and a dining room in the same building, governed by the same HVAC system. For each room, Hammer, Yazidi, and Rue (2021) suggests building a LASSO-regularised (Tibshirani 1996) autoregressive (AR) model of order 1 to predict the room temperature 15 minutes ahead using the current room temperature and all other current non-temperature measurements. Attempts to reproduce this model led to considerably lower forecasting errors than those reported, so perhaps the authors did not use the current room temperature readings, as these are highly autocorrelated and the most useful covariate to select. Instead, similar forecasting errors were reproduced by constructing linear LASSO-regularised Regression (LR) models for the current room temperature regressed on *only* the previous non-temperature measurements. The regularisation parameter was chosen via cross-validation to be the largest value that ensures all cross-validation prediction errors are within one standard error. Figure 6.7 shows the 15-minute ahead absolute forecasting errors using for static LR models, trained on the first 8 days, or 768 readings. These errors very closely

resemble Figure 6 in Hammer, Yazidi, and Rue (2019), and similarly, the red solid lines show the increasing linear trend in forecasting error over time. Observing such behaviour is indicative of concept drift, and ideally the model should not degrade over time. As the authors remark, it is common to retrain or update the model by controlling the mean prediction error, but they suggest a useful and interpretable strategy is to control the prediction error to rarely exceed a threshold, or some defined upper quantile.

The `AFSQE` procedure was used to continuously track the 80% quantile of the absolute forecasting errors of an LR model initially trained on 24 hours worth of data, or 96 readings. Then, whenever this quantile exceeded 2 degrees Celsius, the model was replaced with a new LR model constructed from the previous 24 hours worth of data. Figure 6.8 shows the 15-minute ahead absolute forecasting error using this retraining strategy. The black points near the horizontal axis correspond to model retraining events (notice the 80% quantile estimate shown in the solid blue line exceeds 2). With this approach, there is less linear trend for the forecasting error, though it is marginally increasing for the dining room, and in fact, slightly decreasing for the bedroom. Interestingly this approach approximately controls the mean absolute forecasting error at around 1. The clear advantage in comparison to the static model is that the resulting forecasting errors are roughly constant over time, which is indicative of consist model performance. The contrast is particularly clear after forecasting 25 days ahead, where the mean absolute forecast error for each room using the static model starts to exceed 1 degree and continuously increases, however this value is always less than 1 after 25 days using the concept drift detection retraining strategy.

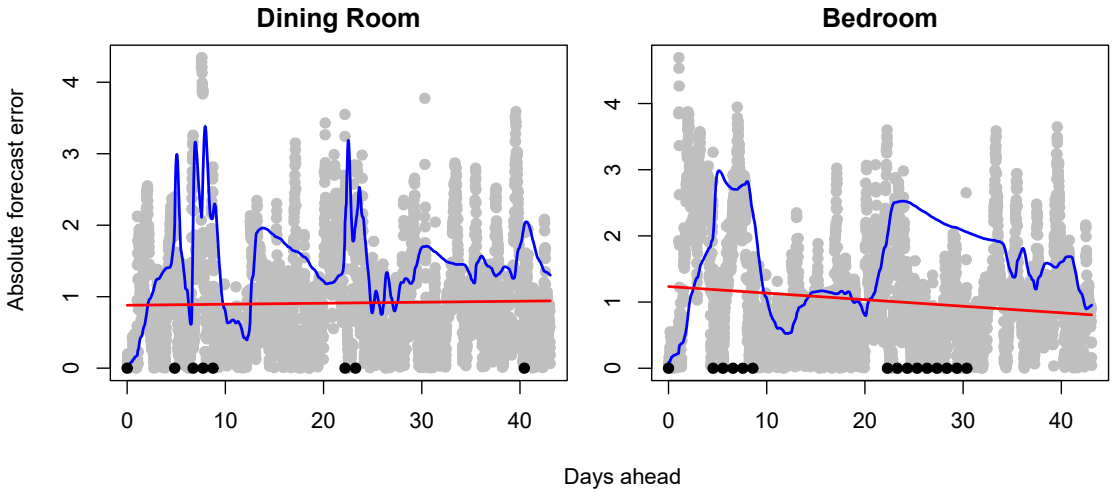

Figure 6.8: Absolute forecasting error of predicted temperature 15-minutes into the future using a model initially trained on the first 24 hours of data, and retrained whenever concept drift is detected. The left and right panels correspond to the dining room and bedroom respectively, where the red solid lines show the linear trend of the absolute forecasting error over time. The blue solid line shows the adaptive 0.8-quantile estimates using `AFSQE`. Whenever this quantile estimate exceeded 2, the model was retrained due to a detection in concept drift and is indicated by the black points along the horizontal axis.

6.6 Multiple Quantile Estimation

Previously the focus has been on the estimation of a single quantile, however, it is often the case that estimating multiple quantiles is more useful. Estimating multiple points on the quantile function allows the shape of the distribution to be captured.

Clearly, the most simple approach is to run several of the streaming quantile estimation procedures in Section 6.3 concurrently over different quantiles. These procedures would be independent of each other which raises two concerns. First, there is no guarantee that the quantile estimates will be monotone as is the case for the true quantiles, referred to as the *quantile crossover problem*. Second, the joint structure of the quantiles is not being exploited to essentially 'calibrate' the quantiles against each other.

This section explores methods for multiple streaming quantile estimation beginning with the discussion of three distinct categories of approach for addressing the above concerns. Then methodology is developed for incorporating these methods into an adaptive streaming multiple quantile estimation procedure based on `AFSQE` (see Algorithm 8). Finally, performance of both the simple and other approaches is assessed in simulation and compared to an approach in Hammer, Yazidi, and Rue (2021).

6.6.1 Post-update Ordering

The first type of approach considered orders the quantile estimates after their update steps. This is perhaps the most trivial approach, but its primary advantages are ease of implementation and applicability to any SQE procedure. The choice of ordering method for the quantile estimates implicitly reflects the joint relationship modelled between the quantiles.

Sorting Quantile Estimates

Perhaps the most obvious ordering method is to sort the quantile estimates. Specifically, given probabilities $q_1 < q_2 < \cdots < q_k$, any streaming quantile estimation procedure gives corresponding potentially unordered quantile estimates $\widetilde{Q}_t(q_1), \widetilde{Q}_t(q_2), \ldots, \widetilde{Q}_t(q_k)$ at time t. Writing the corresponding order statistics for these quantile estimates as $\widetilde{Q}_{t,(1)}, \widetilde{Q}_{t,(2)}, \ldots, \widetilde{Q}_{t,(k)}$, and setting each $\acute{Q}_t(q_j) = \widetilde{Q}_{t,(j)}$ for $j = 1, 2, \ldots, k$, gives a non-decreasing collection of quantile estimates $\acute{Q}_t(q_1), \acute{Q}_t(q_2), \ldots, \acute{Q}_t(q_k)$.

In general, most efficient sorting algorithms will result in a worst-case running time complexity of $\mathcal{O}(k \log k)$. Note that there are cases where further optimisation can be made based on the underlying streaming quantile estimation procedure. For instance, when using a shared adaptive forgetting factor (or weight) across concurrent instances of the `AFMIQE` procedure in Algorithm 6, the update step either preserves order or in the worst case splits the estimates into two concatenated ordered subcollections. This behaviour arises from the explicit nature of the update step in that quantile estimates below the most recent observation are increased, whereas quantile estimates above the most recent observation are decreased. Two concatenated ordered subcollections, $\widetilde{Q}_t(q_1) \leq \widetilde{Q}_t(q_2) \leq \cdots \leq \widetilde{Q}_t(q_s)$ and $\widetilde{Q}_t(q_{s+1}) \leq \widetilde{Q}_t(q_{s+2}) \leq \cdots \leq \widetilde{Q}_t(q_k)$, can then be sorted in $\mathcal{O}(k)$ by merging.

Monotone Optimisation

Monotone, or isotonic, optimisation enforces estimates to be non-decreasing. In the context of quantile estimation, the concern is unweighted linear ordering isotonic optimisation, which

can be expressed formally as the following quadratic program. Given potentially unordered quantile estimates $\widetilde{Q}_t(q_1), \widetilde{Q}_t(q_2), \ldots, \widetilde{Q}_t(q_k)$,

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} \Big(\overleftarrow{Q}_t(q_i) - \widetilde{Q}_t(q_i) \Big)^2, \\ \text{subject to} \quad & \overleftarrow{Q}_t(q_1) \leq \overleftarrow{Q}_t(q_2) \leq \cdots \leq \overleftarrow{Q}_t(q_k). \end{aligned}$$

Then $\overleftarrow{Q}_t(q_1), \overleftarrow{Q}_t(q_2), \ldots, \overleftarrow{Q}_t(q_k)$ is a non-decreasing fit to $\widetilde{Q}_t(q_1), \widetilde{Q}_t(q_2), \ldots, \widetilde{Q}_t(q_k)$.

The Pool Adjacent Violators Algorithm (PAVA) solves this program in linear time with linear memory, with respect to k (Barlow et al. 1972; Mair et al. 2009). PAVA sequentially combines out of order quantile estimates to their mean, resulting in equal adjacent quantile estimates. This is not necessarily ideal for reporting the raw quantile estimates, as in general a non-zero difference between adjacent estimates is expected for continuous data, but this can be viewed as an intuitive corrective step to account for the uncertainty around unordered adjacent estimates.

6.6.2 Restricted Quantile Updates

Another approach restricts the quantile update step in some manner that respects the order of quantile estimates. This is not necessarily applicable to all SQE procedures as it depends on the form of the update step, but does require tailored implementation. Two examples of this approach in the literature are `monotoneSA` (Cao et al. 2009) and `MDUMIQE` (Hammer, Yazidi, and Rue 2020). The former shows how the quantile estimates from Tierney's original SA algorithm are derived from a local linear approximation, which is then constrained to ensure the linearly interpolated estimates preserve monotonicity. The latter is a direct extension of concurrent DUMIQE instances that uses a shared, constrained fixed weight ω, across multiple quantiles. The fundamental difference is the update step is constrained to prevent updates large enough to violate monotonicity. The primary drawback of this restriction is that it caps the rate of forgetting considerably, which can hinder performance for quickly drifting data. In Hammer and Yazidi (2017) and Yazidi and Hammer (2017), the authors observe, on average, significantly better performance for `MDUMIQE` than `monotoneSA` for a sample of simple stationary and non-stationary streams. However, it is worth noting that the authors ignore the recommendation of choosing c in a self-tuned fashion, based on an estimate of the interquartile range (Cao

et al. 2009; Chen, Lambert, et al. 2000). Treating c as an adaptive self-tunable parameter, both procedures depend on a single tunable parameter that controls the rate of forgetting.

Such approaches based on restricting quantile updates are not considered due to properties that inhibit adaptability.

6.6.3 Conditional Quantile Estimation

The last approach considered ensures the quantile estimates are updated relative to each other by exploiting properties of conditional distributions of the data. This is based on a general framework in Hammer, Yazidi, and Rue (2021) which the authors use to extend `QEWA`. The framework takes advantage of two key properties. First, transformations that shift the location of a variable, shift the corresponding transformed quantile function by the same amount. That is,

$$Z = X + \delta \implies Q_Z(q) = Q_X(q) + \delta.$$

Second, for probabilities $q_1 < q_2$,

$$\begin{aligned} Z = X - Q_X(q_2) &\implies \mathbb{P}(Z < Q_Z(q_1) \mid Z < 0) = \frac{q_1}{q_2}, \\ Z = X - Q_X(q_1) &\implies \mathbb{P}(Z < Q_Z(q_2) \mid Z > 0) = \frac{q_2 - q_1}{1 - q_1}. \end{aligned}$$

The latter property means the q_1 quantile of Z is equivalent to the $\frac{q_1}{q_2}$ quantile of $Z \mid Z < 0$, and the q_2 quantile of Z is equivalent to the $\frac{q_2 - q_1}{1 - q_1}$ quantile of $Z \mid Z > 0$. Using this result, the q_1 quantile of X can be obtained from the q_2 quantile of X (or the q_1 quantile from the q_2 quantile) by first transforming to Z, estimating the conditional quantile, and then shifting back.

The framework suggests starting with an initial central quantile, e.g. the median, and then using conditional quantiles to recursively compute adjacent quantile estimates. For this process to work, it is important that the underlying streaming quantile estimation procedure ensures the conditional quantile estimates are strictly above and below zero for quantiles above and below the initial central quantile respectively. This is the case for estimators with a multiplicative update step, but otherwise this can be achieved by truncating the range of the conditional quantile estimates to prevent negative or positive values respectively. Algorithm 2 in Hammer,

Yazidi, and Rue (2021) provides an implementation for `QEWA`, which does not require this truncation due to the multiplicative update step. The same conditional quantile framework can be used to extend `AFSQE` (see Section 6.3.4) for multiple quantile estimation; as `AFSQE_CondQ` as presented in Algorithm 9. The truncation steps required are represented in lines 14 and 23, where the tolerance $\varepsilon = 10^{-8}$ proved sufficient in experimentation.

Algorithm 9 `AFSQE_CondQ`: a streaming quantile estimation procedure for multiple quantiles using adaptive forgetting and stochastic gradient descent.

1: **Input:** Stream of realisations $(x_1, x_2, \dots)$ where each $x_s \in \mathbb{R}$, for $s = 1, 2, \dots$.
2: Let $\widetilde{Q}^{(q_j)}$ be the adaptive estimate for the q_j-quantile, and $\tilde{\theta}^{(j)}$ the corresponding adaptive ECDF estimate. Let η_0 be a fixed learning hyperparameter. Let $\varepsilon << 1$ be some tolerance.
3: Initialise $(\lambda^{(j)}, w^{(j)}, w^{(j)\prime}, \tilde{\theta}^{(j)}, \tilde{\theta}^{(j)\prime}) = (1, 0, 0, 0, 0)$ for $j = 1, 2, \dots, k$.
4: Initialise $\widetilde{Q}^{(q_j)}$ using a sample quantile from an initial data sample, or single point x_0 for $j = 1, 2, \dots, k$.
5: **for** $s = 1, 2 \dots$ **do**
6: Set $\left(y_s^{(j)}\right)_{j=1}^{k} = \left(\mathbb{1}\left\{x_s < \widetilde{Q}^{(q_j)}\right\}\right)_{j=1}^{k}$.
7: Set $(\lambda^{(c)}, w^{(c)}, w^{(c)\prime}, \tilde{\theta}^{(c)}, \tilde{\theta}^{(c)\prime}) = \text{AFFB\_UPDATE}(y_s^{(c)}, \lambda^{(c)}, w^{(c)}, w^{(c)\prime}, \tilde{\theta}^{(c)}, \tilde{\theta}^{(c)\prime})$.
8: Set central quantile estimate $\widetilde{Q}^{(q_c)} = \text{AFSQE\_UPDATE}(x_s, q_c, w^{(c)}, \widetilde{Q}^{(q_c)}, \tilde{\theta}^{(c)})$.
9: **for** $j = c-1, c-2, \dots, 1$ **do**
10: **if** $x_s < \widetilde{Q}^{(q_{j+1})}$ **then**
11: Set transformed observation $z_s^{(j)} = x_s - \widetilde{Q}^{(q_{j+1})}$.
12: Set $(\lambda^{(j)}, w^{(j)}, w^{(j)\prime}, \tilde{\theta}^{(j)}, \tilde{\theta}^{(j)\prime}) = \text{AFFB\_UPDATE}(y_s^{(j)}, \lambda^{(j)}, w^{(j)}, w^{(j)\prime}, \tilde{\theta}^{(j)}, \tilde{\theta}^{(j)\prime})$
13: Update transformed quantile $\widetilde{Q}_Z^{(q_j)} = \text{AFSQE\_UPDATE}(x_s, \frac{q_j}{q_{j+1}}, w^{(j)}, \widetilde{Q}_Z^{(q_j)}, \tilde{\theta}^{(j)})$.
14: Truncate transformed quantile $\widetilde{Q}_Z^{(q_j)} = \min\left(\widetilde{Q}_Z^{(q_j)}, -\varepsilon\right)$.
15: **end if**
16: Set quantile estimate $\widetilde{Q}^{(q_j)} = \widetilde{Q}^{(q_{j+1})} + \widetilde{Q}_Z^{(q_j)}$.
17: **end for**
18: **for** $j = c+1, c+2, \dots, k$ **do**
19: **if** $x_s > \widetilde{Q}^{(q_{j-1})}$ **then**
20: Set transformed observation $z_s^{(j)} = x_s - \widetilde{Q}^{(q_{j-1})}$.
21: Set $(\lambda^{(j)}, w^{(j)}, w^{(j)\prime}, \tilde{\theta}^{(j)}, \tilde{\theta}^{(j)\prime}) = \text{AFFB\_UPDATE}(y_s^{(j)}, \lambda^{(j)}, w^{(j)}, w^{(j)\prime}, \tilde{\theta}^{(j)}, \tilde{\theta}^{(j)\prime})$
22: Update transformed quantile $\widetilde{Q}_Z^{(q_j)} = \text{AFSQE\_UPDATE}(x_s, \frac{q_j - q_{j-1}}{1 - q_{j-1}}, w^{(j)}, \widetilde{Q}_Z^{(q_j)}, \tilde{\theta}^{(j)})$.
23: Truncate transformed quantile $\widetilde{Q}_Z^{(q_j)} = \max\left(\widetilde{Q}_Z^{(q_j)}, \varepsilon\right)$.
24: **end if**
25: Set quantile estimate $\widetilde{Q}^{(q_j)} = \widetilde{Q}^{(q_{j-1})} + \widetilde{Q}_Z^{(q_j)}$.
26: **end for**

For AFFB_UPDATE and AFSQE_UPDATE see Algorithms 2 and 8.

6.6.4 Simulation Results

This section seeks to compare the approaches outlined in the previous section when used to extend the `AFSQE` procedure for multiple quantile estimation for some set of quantile probabilities $(q_j)_{j=1}^k$.

In addition to the methods described in the previous section, it is useful to consider a `Default` case, where the `AFSQE` procedure is naïvely run k times; once for each quantile with no additional methods used. Clearly it is possible for the resulting quantile estimates to violate monotonicity, but this case serves as a baseline to be compared against for the other approaches. Each instance of the procedure runs independently, with separate sets of parameters for each instance, i.e. each quantile is estimated using its own set of forgetting factors. It is worth remarking here that the option to use a shared forgetting factor scheme mechanism is available by aggregating the forgetting factors across the instances, e.g. by taking the mean, and then using the shared forgetting factor in the quantile update step. However, during the experimentation this always led to worse performance than letting each instance use its own set of forgetting factors, and so a shared forgetting factor scheme is not advocated or used here for any modification method to deal with multiple quantile estimation.

The post-update methods are implemented in the same way as the `Default` case to the `AFSQE` procedure, with the notable exception that the post-update method is applied to the quantile estimates after each update step. The `Sort` method sorts the quantile estimates as in Section 6.6.1, and the `PAVA` method uses monotone optimisation as described in Section 6.6.1.

The conditional quantile framework `CondQ` discussed in Section 6.6.3 is used to extend both `AFSQE` (as shown in Algorithm 9) and `QEWA`, where the latter is included to provide a comparison against a state-of-the-art method in the literature.

The same non-stationary simulation scenario setups as in Section 6.4 are used to evaluate the RMSE for each of the above approaches. In addition, a stationary scenario is also tested, where $n = 10^5$ data points are simulated from a standard normal distribution. The main difference here is that multiple quantiles are estimated instead of a single quantile. Specifically quantiles are estimated for the 19 quantile probabilities $(\frac{i}{20})_{i=1}^{19}$, where the RMSE is the mean RMSE across all 19 quantiles. These results are shown in Table 6.10.

Perhaps the most surprising result is that in 4 out of the 9 simulation scenarios, the `Default` approach gives the lowest RMSE. In all other cases both `Sort` and `PAVA` provide marginally better results. In 7 out of the 9 situations tested, the `CondQ` extension results in a larger RMSE than the `Default` approach, yet performs best in the Normal Switch Slow case. It is particularly poor in the stationary case where the RMSE is 3–4 times larger than the other approaches. Perhaps this poor performance is a consequence of starting with the median as a central quantile to estimate the other 18 quantiles, with estimation errors propagating all the way from the central quantile estimates to the extreme quantile estimates. Here the `AFSQE_CondQ` approach gives a noticeably lower RMSE than the `QEWA_CondQ` approach in two-thirds of the tested scenarios.

Table 6.10: Root mean squared estimation error for streaming multiple quantile estimation procedures run on across a selection of non-stationary smooth and abrupt simulated data. The bottom row shows the additional worst-case computational complexity required relative to the `Default` case for each procedure in terms of k, the number of quantiles being estimated. For all procedures, the storage complexity is $\mathcal{O}(k)$.

	Streaming Multiple-Quantile Procedure				
	AFSQE				QEWA(0.1)
Simulation Scenario	Default	Sort	PAVA	CondQ	CondQ
Stationary	0.085	0.083	**0.083**	0.237	0.344
Normal Smooth Slow	0.472	0.463	0.468	**0.387**	0.424
Normal Smooth Fast	**0.945**	0.968	0.954	0.966	1.422
Normal Switch Slow	0.782	0.781	0.781	0.909	**0.753**
Normal Switch Fast	**1.387**	1.649	1.648	1.783	2.096
Chi-Squared Smooth Slow	0.947	0.919	**0.919**	1.136	1.945
Chi-Squared Smooth Fast	**1.696**	1.703	1.702	2.228	2.011
Chi-Squared Switch Slow	1.289	1.278	**1.278**	1.510	1.635
Chi-Squared Switch Fast	**2.177**	2.219	2.217	2.778	2.343
Additional complexity	None	$\mathcal{O}(k \log k)$	$\mathcal{O}(k)$	$\mathcal{O}(k)$	$\mathcal{O}(k)$

6.7 Conclusion

This chapter has contributed to the streaming analytics toolkit by providing fully adaptive tools for streaming quantile estimation. A variety of existing tools are enhanced with the adaptive forgetting framework, and a new streaming quantile estimation method is intro-

duced. Extensive experiments show that there is no procedure that consistently outperforms all others, as if often the case with data analytic tools. In the absence of a globally optimal procedure, it is recommended to choose a model that is adequate in a wide range of scenarios. Based on this rationale, the `AFSQE` method is recommended as a preferable default. This recommendation is further supported by the ability to operate without the burden of configuring control parameters.

7 Conclusion

The objective of this research was to enhance the streaming analytic toolkit with methods based on the adaptive forgetting (AF) framework. The research contributions towards this objective are as follows.

The work in Chapter 3, focussed on bivariate data streams by extending the AF framework to a specific scenario in the context of cyber-physical systems. This work developed the AF procedure — formulation, update equations, and algorithms — for estimating the time-varying parameters of a normal-gamma model. In studying the application, significant attention is given to assessing model validity, which presents specific challenges. Based on the adaptive estimation procedure, a method for changepoint detection is constructed. This change detector exploits the relationship between adaptive and non-adaptive estimators; a feature not previously used in the literature. This strategy is also adopted more creatively in Chapter 4, where this tactic is shown to perform effectively.

Chapter 4 extended the AF framework to provide an adaptive estimator of the correlation coefficient. This estimator is again enhanced with a changepoint detector, and the tactic of using both fixed and adaptive estimators is further fruitfully exploited. Anomaly detection in cybersecurity provides the motivating application for this work, where servicing the requirements of the application dictated the invention of methods to handle coincidental anomalies across multiple streams.

Quantile estimation is strikingly under-represented in the streaming analytics toolkit, particularly in terms of the AF framework. Chapters 5 and 6 both relate to this problem in different ways. The adaptive estimation of binomial parameters is core to the methods used for streaming quantile estimation, which motivated the development of an adaptive estimator for binomial data, `AFFBinom(m)`. In examining this fundamental dependence, aspects of the fragility of the stochastic gradient descent optimisation procedure required scrutiny. A col-

lection of experimental studies led to some loose conclusions for implementing `AFFBinom(m)`. Namely, that the choice of cost function should be based on the trial size; NLL for $m > 1$, squared error for $m = 1$, and that the learning rates should be unscaled and not exceed 0.1.

Finally, Chapter 6 used the procedures developed in Chapter 5 to estimate time-varying values of the empirical cumulative distribution function. This engine enhanced three streaming quantile estimators with temporal-adaptivity. Additionally, a novel streaming quantile estimation procedure called `AFSQE` is developed. An extensive simulation study assessed these four methods in a range of non-stationary environments. No method emerges as best in all cases, but `AFSQE` is recommended for its adequate performance in many situations.

Future Work

There are many avenues of further investigation suggested from this research. Three of particular interest are

- The trick of using adaptive versus non-adaptive estimates (see Sections 3.4 and 4.2.2) to construct change detectors in a continuous monitoring context was shown to be effective. Theoretical analysis of the distances between adaptive and non-adaptive estimates could lead to better methodology for sequential change detection, and perhaps even a new class of algorithm.

- The experiments in Section 5.4 reveal that there is potential promise in a deeper study of the effects of truncating the forgetting factors, not only for the use in adaptive binomial estimation, but in any adaptive estimation procedure. It was shown that an additional truncation step improved AFMLE performance in stationary periods and mitigated most of the AFMLE bias observed in Section 5.2. Further understanding of which classes of distribution family this AFMLE bias occurs for, and how truncation affects the adaptive estimation process, could lead to new AF algorithms.

- Section 6.6 explored the task of multiple quantile estimation. This is a step towards reliably estimating an entire ECDF, which could prove useful for developing non-parametric change detection procedures based on statistical distances like the KS statistic (see Section 2.4. These types of procedures would be substantial contributions to the streaming

toolkit. In fact, (Plasse 2019) derives a time-varying estimate for the ECDF through temporally-adaptive sequential histograms and constructs a change detector. Understanding how these approaches differ could lead to the development of improved change detectors.

Appendices

A Derivations

A.1 Adaptive Forgetting Maximum Likelihood Estimates for the Normal-Gamma Distribution

Recall from Equation (3.1), the bivariate pdf for a normal-gamma random variable (X, S) parameterised by $\theta = \{\mu, \gamma\, \alpha, \beta\}$ is given by

$$f(x, s \mid \theta) = \frac{\beta^{\alpha}\gamma^{1/2}}{\Gamma(\alpha)\sqrt{2\pi}}\, s^{\alpha-1/2} \exp\left[-\beta s - \frac{\gamma s(x-\mu)^2}{2}\right],$$

where $\Gamma(\cdot)$ is the gamma function and the marginal distribution of S is a gamma distribution with shape parameter α and rate parameter β. Then, as in Equation (3.2), the adaptive forgetting log-likelihood for data $\boldsymbol{d}_{1:t} = (x_i, s_i)_{i=1}^{t}$ and forgetting factors $(\lambda_i)_{i=1}^{t-1}$ is

$$\mathcal{L}_{\lambda}(\theta \mid \boldsymbol{d}_{1:t}) = \sum_{i=1}^{t}\left(\prod_{j=i}^{t-1}\lambda_j\right)\left[\alpha\log(\beta) + \frac{\log(\gamma)}{2} - \log(\Gamma(\alpha)) + \left(\alpha - \frac{1}{2}\right)\log(s_i) - \beta s_i - \frac{\gamma s_i(x_i-\mu)^2}{2}\right],$$

where the empty product when $j = t$ is defined to be 1. Then the derivatives of the AF log-likelihood with respect to each parameter are

$$\begin{aligned}
\frac{\partial}{\partial \mu}\mathcal{L}_\lambda(\theta \mid \boldsymbol{d}_{1:t}) &= \gamma\left[\sum_{i=1}^{t}\left(\prod_{j=i}^{t-1}\lambda_j\right)x_i s_i\right] - \gamma\mu\sum_{i=1}^{t}\left(\prod_{j=i}^{t-1}\lambda_j\right)s_i \\
&= \gamma(m_{\boldsymbol{xs},t} - \mu m_{\boldsymbol{s},t}), \qquad \text{(A.1)} \\
\frac{\partial}{\partial \gamma}\mathcal{L}_\lambda(\theta \mid \boldsymbol{d}_{1:t}) &= \frac{1}{2\gamma}\sum_{i=1}^{t}\left(\prod_{j=i}^{t-1}\lambda_j\right) - \frac{1}{2}\left[\sum_{i=1}^{t}\left(\prod_{j=i}^{t-1}\lambda_j\right)\left(x_i^2 s_i - 2\mu x_i s_i + \mu^2 s_i\right)\right] \\
&= \frac{1}{2}\left(\frac{n_t}{\gamma} - m_{\boldsymbol{x}^2\boldsymbol{s},t} + 2\mu m_{\boldsymbol{xs},t} - \mu^2 m_{\boldsymbol{s},t}\right), \qquad \text{(A.2)} \\
\frac{\partial}{\partial \alpha}\mathcal{L}_\lambda(\theta \mid \boldsymbol{d}_{1:t}) &= \log\beta\sum_{i=1}^{t}\left(\prod_{j=i}^{t-1}\lambda_j\right) - \psi(\alpha)\sum_{i=1}^{t}\left(\prod_{j=i}^{t-1}\lambda_j\right) + \sum_{i=1}^{t}\left(\prod_{j=i}^{t-1}\lambda_j\right)\log(s_i) \\
&= n_t(\log\beta - \psi(\alpha)) + m_{\log \boldsymbol{s},t}, \qquad \text{(A.3)} \\
\frac{\partial}{\partial \beta}\mathcal{L}_\lambda(\theta \mid \boldsymbol{d}_{1:t}) &= \frac{\alpha}{\beta}\sum_{i=1}^{t}\left(\prod_{j=i}^{t-1}\lambda_j\right) - \sum_{i=1}^{t}\left(\prod_{j=i}^{t-1}\lambda_j\right)s_i \\
&= \frac{\alpha}{\beta}n_t - m_{\boldsymbol{s},t}, \qquad \text{(A.4)}
\end{aligned}$$

where

$$\begin{aligned}
n_t &= \sum_{i=1}^{t}\left(\prod_{j=i}^{t-1}\lambda_j\right) \\
&= \lambda_{t-1}n_{t-1} + 1, \\
m_{g(\boldsymbol{x},\boldsymbol{s}),t} &= \sum_{i=1}^{t}\left(\prod_{j=i}^{t-1}\lambda_j\right)g(x_i, s_i) \\
&= \lambda_{t-1}m_{g(\boldsymbol{x},\boldsymbol{s}),t-1} + g(x_t, s_t),
\end{aligned}$$

and $\psi(\cdot) \equiv \Gamma'(\cdot)/\Gamma(\cdot)$ is the digamma function.

Since the precision $\gamma > 0$, setting Equation (A.1) equal to zero and solving gives the adaptive forgetting maximum likelihood estimate (AFMLE) for μ at time t as

$$\tilde{\mu}_t = \frac{m_{\boldsymbol{xs},t}}{m_{\boldsymbol{s},t}}. \qquad \text{(A.5)}$$

Then substituting $\mu = \tilde{\mu}_t$ from Equation (A.5) into Equation (A.2) gives

$$\frac{\partial}{\partial \gamma}\mathcal{L}_\lambda(\theta \mid \boldsymbol{d}_{1:t})\Big|_{\mu=\tilde{\mu}_t} = \frac{1}{2}\left(\frac{n_t}{\gamma} - m_{\boldsymbol{x}^2\boldsymbol{s},t} + \frac{(m_{\boldsymbol{x}\boldsymbol{s},t})^2}{m_{\boldsymbol{s},t}}\right). \tag{A.6}$$

Then setting Equation (A.6) equal to zero and solving gives the AFMLE for γ as time t as

$$\begin{aligned}\tilde{\gamma}_t &= n_t\left(m_{\boldsymbol{x}^2\boldsymbol{s},t} - \frac{(m_{\boldsymbol{x}\boldsymbol{s},t})^2}{m_{\boldsymbol{s},t}}\right)^{-1} \\ &= \frac{n_t}{m_{\boldsymbol{x}^2\boldsymbol{s},t} - \tilde{\mu}_t m_{\boldsymbol{x}\boldsymbol{s},t}}.\end{aligned} \tag{A.7}$$

Consider Equation (A.4), which is equal to zero at the AFMLE values $\alpha = \tilde{\alpha}_t, \beta = \tilde{\beta}_t$, such that

$$\tilde{\beta}_t = \tilde{\alpha}_t \frac{n_t}{m_{\boldsymbol{s},t}}. \tag{A.8}$$

Then substituting $\beta = \tilde{\beta}_t$ from Equation (A.8) into Equation (A.3) gives

$$\frac{\partial}{\partial \alpha}\mathcal{L}_\lambda(\theta \mid \boldsymbol{d}_{1:t})\Big|_{\beta=\tilde{\beta}_t} = n_t\left(\log(\tilde{\alpha}_t) - \log\left(\frac{m_{\boldsymbol{s},t}}{n_t}\right) - \psi(\alpha) + \frac{m_{\log \boldsymbol{s},t}}{n_t}\right), \tag{A.9}$$

which has no closed form solution when equal to zero. Numerical optimisation schemes such as Newton's method can be used to find an AFMLE for α, but these can be slow to converge, often requiring hundreds of iterations. The many iterative steps involved are often too computationally burdensome to meet the requirements of streaming data. Minka (2002) uses a generalised Newton scheme that approximates the offline (or no-forgetting) log-likelihood with a function of the form

$$g(x) = c_0 + c_1 x + c_2 \log(x).$$

This generalised Newton approach gives an update step that depends on evaluating digamma and trigamma functions, but impressively tends to converge in just four iterations. However, in the most stringent of settings, this approach may still be too demanding. One alternative is to approximate Equation (A.6) with a sufficiently simple function to obtain a closed form solution when set equal to zero. Grouping and writing the terms involving α as a single function $h(\alpha)$ gives

$$h(\alpha) = \log(\alpha) - \psi(\alpha) = \log\left(\frac{m_{\boldsymbol{s},t}}{n_t}\right) - \frac{m_{\log \boldsymbol{s},t}}{n_t}. \tag{A.10}$$

The Euler-Maclaurin formula can be used to derive the following asymptotic series for the digamma function (Abramowitz and Stegun 1965):

$$\psi(\alpha+1) = \log\alpha + \frac{1}{2\alpha} - \frac{1}{12\alpha^2} + \mathcal{O}(\alpha^4) \tag{A.11}$$

In addition, the digamma function has the recurrence property

$$\psi(\alpha+1) = \psi(\alpha) + \frac{1}{\alpha}. \tag{A.12}$$

Hence truncating

$$\psi(\alpha) \approx \log\alpha - \frac{1}{2\alpha} - \frac{1}{12\alpha^2}, \tag{A.13}$$

such that $h(\alpha)$ can be approximated by

$$h_1(\alpha) = \frac{1}{2\alpha} + \frac{1}{12\alpha^2}, \tag{A.14}$$

ensures Equation (A.10) is a quadratic equation in α for which the solution always exists and is cheap to compute. The approximation $h_1(\alpha)$ is generally effective, but noticeably struggles for $\alpha < 1$ where the relative absolute error

$$\frac{|h_1(\alpha) - h(\alpha)|}{h(\alpha)},$$

increases exponentially as $\alpha \to 0$, shown by the red line in Figure A.1. In contrast, the approximation originally proposed in Minka (2002)

$$h_2(\alpha) = \frac{1}{2\alpha} + \frac{1}{12\alpha^2 + 2\alpha}, \tag{A.15}$$

shown by the blue line does not suffer from this issue, where for $\alpha \in (0, 10]$, $h_2(\alpha)$ is always within 1.7% of the true value. Using $h_2(\alpha)$ as in Equation (A.15) also ensures Equation (A.10) is a quadratic, specifically of the form

$$6\zeta_t\alpha^2 + (\zeta_t - 3)\alpha + 1 = 0,$$

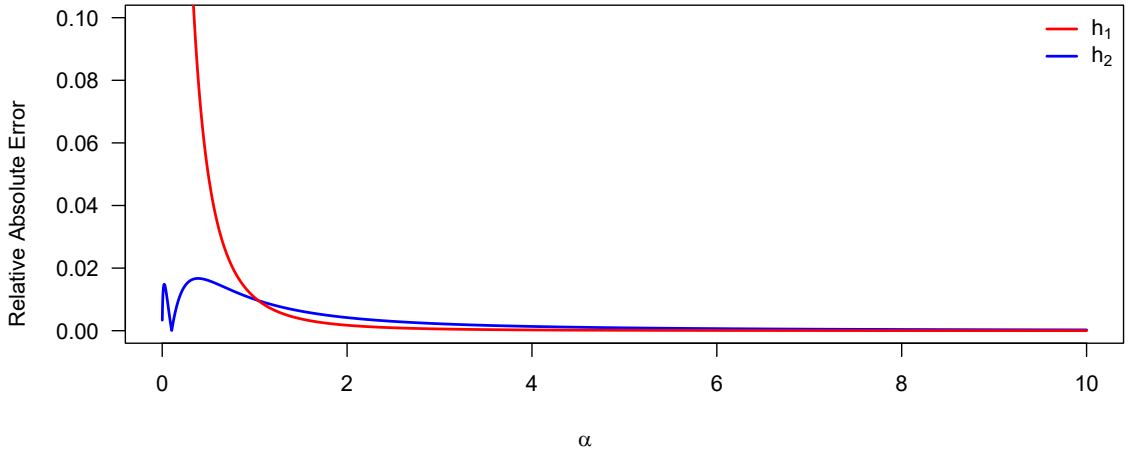

Figure A.1: Absolute relative errors of functions $h_1(\alpha)$ in red, and $h_2(\alpha)$ in blue, which approximate $h(\alpha) = \log(\alpha) - \psi(\alpha)$. Notice that h_1 is a particularly poor approximation the closer α is to 0, whereas the relative error for h_2 is bounded from above.

where

$$\zeta_t = \log\left(\frac{m_{\boldsymbol{s},t}}{n_t}\right) - \frac{m_{\log \boldsymbol{s},t}}{n_t},$$

and hence an approximation to the AFMLE for α is

$$\tilde{\alpha}_t = \frac{(3-\zeta_t) + \sqrt{(\zeta_t - 3)^2 + 24\zeta_t}}{12\zeta_t},$$

where the larger root is always chosen as this maximises the log-likelihood which is monotonically increasing with respect to α.

Finally, $\tilde{\beta}_t$ can be computed using $\tilde{\alpha}_t$ in Equation (A.8).

A.2 Gradient of the Next-Step Log-Likelihood for the Normal-Gamma Distribution

Recall from Equation (3.1), the bivariate pdf for a Normal-Gamma random variable (X, S) parameterised by $\theta = \{\mu, \gamma\, \alpha, \beta\}$ is given by

$$f(x, s \mid \theta) = \frac{\beta^\alpha \gamma^{1/2}}{\Gamma(\alpha)\sqrt{2\pi}}\, s^{\alpha - 1/2} \exp\left[-\beta s - \frac{\gamma s (x-\mu)^2}{2}\right],$$

where $\Gamma(\cdot)$ is the gamma function and the marginal distribution of S is a gamma distribution with shape parameter α and rate parameter β.

Then, the next-step log-likelihood is

$$\begin{aligned}\mathcal{L}\Big(\tilde{\theta}_{t-1} \mid x_t, s_t\Big) &= \log f\Big(x_t, s_t \mid \tilde{\theta}_{t-1}\Big) \\ &= \tilde{\alpha}_{t-1} \log \tilde{\beta}_{t-1} + \frac{1}{2}\log \tilde{\gamma}_{t-1} - \log \Gamma(\tilde{\alpha}_{t-1}) - \frac{1}{2}\log(2\pi) \\ &+ \left(\tilde{\alpha}_{t-1} - \frac{1}{2}\right)\log(s_t) - \tilde{\beta}_{t-1} s_t - \frac{1}{2}\tilde{\gamma}_{t-1} s_t (x_t - \tilde{\mu}_{t-1})^2. \end{aligned} \tag{A.16}$$

Then the gradient of the next-step log-likelihood can be computed by assuming that each component of the adaptive estimate $\tilde{\theta}_{t-1}$ is a function of a single forgetting factor λ (see Section 3.2.4). Differentiating Equation (A.16) with respect to λ gives

$$\begin{aligned}\nabla_\lambda\Big[\mathcal{L}\Big(\tilde{\theta}_{t-1} \mid x_t, s_t\Big)\Big] &= \tilde{\gamma}_{t-1} s_t (x_t - \tilde{\mu}_{t-1}) \frac{\partial}{\partial \lambda}\tilde{\mu}_{t-1} \\ &+ \frac{1}{2}\left(\frac{1}{\tilde{\gamma}_{t-1}} - s_t (x_t - \tilde{\mu}_{t-1})^2\right)\frac{\partial}{\partial \lambda}\tilde{\gamma}_{t-1} \\ &+ \Big(\log\Big(\tilde{\beta}_{t-1} s_t\Big) - \psi(\tilde{\alpha}_{t-1})\Big)\frac{\partial}{\partial \lambda}\tilde{\alpha}_{t-1} \\ &+ \left(\frac{\tilde{\alpha}_{t-1}}{\tilde{\beta}_{t-1}} - s_t\right)\frac{\partial}{\partial \lambda}\tilde{\beta}_{t-1},\end{aligned}$$

where $\psi(\cdot) \equiv \Gamma'(\cdot)/\Gamma(\cdot)$ is the digamma function.

A.3 The Adaptive Forgetting Log-Likelihood for Bernoulli Data

Recall from Section 2.1.3 the density for the Bernoulli distribution is

$$f(y \mid \theta) = \theta^y (1-\theta)^{1-y}.$$

So the likelihood for an i.i.d. Bernoulli sample $y_1, y_2, \ldots, y_t$ is

$$L(\theta \mid y_1, \ldots, y_t) = \prod_{i=1}^{t} \theta^{y_i}(1-\theta)^{1-y_i} = \theta^{t\bar{y}}(1-\theta)^{t-t\bar{y}},$$

where

$$\bar{y} = \frac{1}{t}\sum_{i=1}^{t} y_i.$$

Hence the log-likelihood is given by

$$\mathcal{L}(\theta \mid y_1, \ldots, y_t) = \log(\theta) \sum_{i=1}^{t} y_i + \log(1-\theta) \sum_{i=1}^{t} (1 - y_i),$$

or, for a single datum y_i is simply $\mathcal{L}(\theta \mid y_i) = y_i \log(\theta) + (1 - y_i) \log(1-\theta)$. Then, the *adaptive forgetting log-likelihood* is the exponentially weighted sum

$$\begin{aligned}
\mathcal{L}^{(\lambda)}(\theta \mid y_1, \ldots y_t) &= \sum_{i=1}^{t} \left(\prod_{j=i}^{t-1} \lambda_j \right) \mathcal{L}(\theta \mid y_i) \\
&= \sum_{i=1}^{t} \left(\prod_{j=i}^{t-1} \lambda_j \right) \left[y_i \log(\theta) + (1 - y_i) \log(1-\theta) \right] \\
&= \log(\theta) \sum_{i=1}^{t} \left(\prod_{j=i}^{t-1} \lambda_j \right) y_i + \log(1-\theta) \sum_{i=1}^{t} \left(\prod_{j=i}^{t-1} \lambda_j \right) (1 - y_i) \\
&= m_t \log(\theta) + (w_t - m_t) \log(1-\theta),
\end{aligned}$$

where

$$\begin{aligned}
m_t &= \sum_{i=1}^{t} \left(\prod_{j=i}^{t-1} \lambda_j \right) y_i = \lambda_t m_{t-1} + y_t, \\
w_t &= \sum_{i=1}^{t} \left(\prod_{j=i}^{t-1} \lambda_j \right) = \lambda_t w_{t-1} + 1.
\end{aligned}$$

A.4 Quadratic Approximation for the AFSQE Proxy Cost

Recall from Section 6.3.4, the strictly consistent scoring function

$$S_q(x, y) = (\mathbb{1}\{x \geq y\} - q)(x - y),$$

is chosen as a cost function to sequentially update the quantile estimate $\widetilde{Q}_t^{(q)}$ as part of the AFSQE procedure. As $Q_t^{(q)}$, the true value of the q-quantile at time t, is unknown, $S_q\left(Q_t^{(q)}, \widetilde{Q}_t^{(q)}\right)$ is not available. However, from Equation (6.6), the following proxy cost, is available

$$S_q\left(q, \tilde{\theta}_t^{(q)}\right) = \left(\mathbb{1}\left\{q \geq \tilde{\theta}_t^{(q)}\right\} - q\right)\left(q - \tilde{\theta}_t^{(q)}\right) = \begin{cases} (1-q)\left(q - \tilde{\theta}_t^{(q)}\right) & \text{if } \tilde{\theta}_t^{(q)} \leq q, \\ -q\left(q - \tilde{\theta}_t^{(q)}\right) & \text{if } \tilde{\theta}_t^{(q)} > q. \end{cases}$$

Note that $S_q\left(q, \tilde{\theta}_t^{(q)}\right)$ is a piecewise-linear function of $\tilde{\theta}_t^{(q)}$ with a global minimum of 0 when $q = \tilde{\theta}_t^{(q)}$. Optimising this function using stochastic gradient descent is non-trivial since this function is not differentiable at the global minimum. An approach to avoid this issue is to use an approximation that is differentiable everywhere. One simple approximation is to use a concave up quadratic function that passes through the same global minimum. Replacing $\tilde{\theta}_t^{(q)}$ with γ for notational convenience, let $f(\gamma) = A(\gamma - B)^2 + C$ be a quadratic that passes through $(q, 0)$. Then clearly $B = q$ and $C = 0$. Choosing $A = 1$ ensures the gradient of the quadratic is relatively close to the gradient of the piecewise-linear function. Hence, the quadratic used is of the form

$$f(\gamma) = (\gamma - q)^2,$$

which is equivalent to the squared error cost for q, and is shown alongside the proxy cost in Figure A.2 for $q = 0.4$.

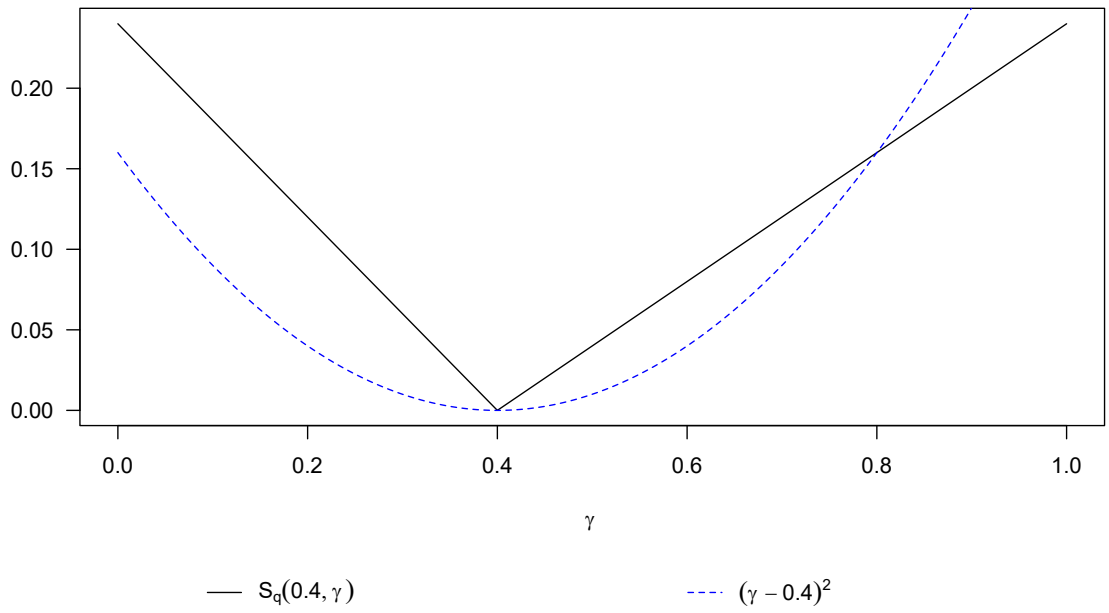

Figure A.2: The piecewise-linear proxy cost $S_q(q, \gamma)$ and the quadratic approximation used.

B Tables

Table B.1: Root mean squared error results for `AFFBinom(10)` for a selection of stationary and non-stationary simulation environments. Cells coloured in green indicate a lower RMSE than for `AFFB` (see Table 5.3), with cells coloured in red otherwise.

		Cost function							
		Negative log-likelihood $\mathcal{J}^{(1)}$				Squared error $\mathcal{J}^{(2)}$			
Simulation		Learning Rate η				Learning Rate η			
Type	Parameters	0.1	0.01	0.001	10^{-4}	0.1	0.01	0.001	10^{-4}
Static	$\theta_0 = 0.5$	0.035	0.023	0.013	**0.008**	0.052	0.069	0.060	0.039
	$\theta_0 = 0.99$	0.010	0.005	0.002	**0.001**	0.015	0.016	0.013	0.008
Smooth (slow)	$\alpha = 0.98, \beta = 0.01$	0.026	0.022	0.021	**0.016**	0.059	0.048	0.044	0.029
	$\alpha = 0.5, \beta = 0.25$	0.033	0.024	0.022	**0.017**	0.050	0.064	0.056	0.037
Smooth (fast)	$\alpha = 0.98, \beta = 0.01$	0.046	0.046	0.041	**0.041**	0.152	0.131	0.054	0.051
	$\alpha = 0.5, \beta = 0.25$	0.045	0.047	0.041	**0.040**	0.100	0.086	0.063	0.053
Abrupt (slow)	$\theta_0 = 0.95$	**0.025**	0.027	0.034	0.038	0.031	0.032	0.035	0.032
	$\theta_0 = 0.75$	0.033	**0.026**	0.032	0.048	0.046	0.058	0.059	0.044
Abrupt (fast)	$\theta_0 = 0.95$	0.066	0.082	0.067	0.058	0.063	**0.049**	0.050	0.054
	$\theta_0 = 0.75$	**0.049**	0.057	0.061	0.055	0.064	0.061	0.068	0.071

Table B.2: Root mean squared error results for `AFFBinom(100)` for a selection of stationary and non-stationary simulation environments. Cells coloured in green indicate a lower RMSE than for `AFFB` (see Table 5.3), with cells coloured in red otherwise.

		Cost function							
		Negative log-likelihood $\mathcal{J}^{(1)}$				Squared error $\mathcal{J}^{(2)}$			
Simulation		Learning Rate η				Learning Rate η			
Type	Parameters	0.1	0.01	0.001	10^{-4}	0.1	0.01	0.001	10^{-4}
Static	$\theta_0 = 0.5$	0.011	0.007	0.004	**0.002**	0.016	0.019	0.024	0.020
	$\theta_0 = 0.99$	0.002	0.001	0.001	**0.001**	0.003	0.004	0.005	0.003
Smooth (slow)	$\alpha = 0.98, \beta = 0.01$	0.009	0.009	0.008	**0.007**	0.189	0.170	0.127	0.014
	$\alpha = 0.5, \beta = 0.25$	0.011	0.009	0.008	**0.007**	0.108	0.086	0.021	0.019
Smooth (fast)	$\alpha = 0.98, \beta = 0.01$	0.020	0.019	0.018	**0.018**	0.242	0.248	0.232	0.019
	$\alpha = 0.5, \beta = 0.25$	0.020	0.019	0.018	**0.018**	0.157	0.155	0.084	0.023
Abrupt (slow)	$\theta_0 = 0.95$	0.020	0.026	0.031	0.019	0.044	0.037	**0.014**	0.015
	$\theta_0 = 0.75$	**0.015**	0.016	0.020	0.019	0.028	0.027	0.020	0.021
Abrupt (fast)	$\theta_0 = 0.95$	0.064	0.081	0.046	**0.034**	0.135	0.110	0.035	0.035
	$\theta_0 = 0.75$	0.037	0.046	0.032	**0.028**	0.077	0.071	0.029	0.029

References

M. Abramowitz and I. Stegun (1965). *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. Applied Mathematics Series. Dover Publications (cit. on p. 137).

N. Adams and N. Heard (2014). *Data Analysis for Network Cyber-Security*. Imperial College Press (cit. on p. 39).

N. Adams and N. Heard (2016). *Dynamic Networks and Cyber-Security*. Imperial College Press (cit. on pp. 39, 53).

S. Ahmad, A. Lavin, S. Purdy, and Z. Agha (2017). "Unsupervised real-time anomaly detection for streaming data". *Neurocomputing* 262, pp. 134–147 (cit. on p. 2).

M. Ahmed, A. N. Mahmood, and J. Hu (2016). "A survey of network anomaly detection techniques". *Journal of Network and Computer Applications* 60, pp. 19–31 (cit. on p. 40).

C. Anagnostopoulos (2010). "A Statistical Framework for Streaming Data Analysis". PhD thesis. London, UK: Imperial College London (cit. on pp. 2, 16, 59, 66, 69, 71).

C. Anagnostopoulos, D. K. Tasoulis, N. M. Adams, N. G. Pavlidis, and D. J. Hand (2012). "Online Linear and Quadratic Discriminant Analysis with Adaptive Forgetting for Streaming Classification". *Statistical Analysis and Data Mining* 5:2, pp. 139–166 (cit. on pp. 2, 17, 18, 23, 26, 44, 46, 47, 59).

J. E. Angus (1994). "The Probability Integral Transform and Related Results". *SIAM review* 36:4, pp. 652–654 (cit. on p. 29).

K. J. Åström and B. Wittenmark (1973). "On Self Tuning Regulators". *Automatica* 9:2, pp. 185–199 (cit. on p. 15).

K. Åström, U. Borisson, L. Ljung, and B. Wittenmark (1977). "Theory and applications of self-tuning regulators". *Automatica* 13:5, pp. 457–476 (cit. on p. 15).

K. J. Åström, ed. (1970). *Index*. Vol. 70. Mathematics in Science and Engineering. Elsevier, pp. 295–299 (cit. on p. 15).

R. E. Barlow, D. J. Bartholomew, J. M. Bremner, and H. D. Brunk (1972). *Statistical inference under order restrictions; the theory and application of isotonic regression*. Wiley, London (cit. on p. 125).

A. Benveniste, P. Priouret, and M. Métivier (1990). *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag, Berlin, Heidelberg (cit. on p. 16).

D. K. Bhattacharyya and J. K. Kalita (2013). *Network anomaly detection: A machine learning perspective*. CRC Press (cit. on p. 40).

M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita (2014). "Network Anomaly Detection: Methods, Systems and Tools". *IEEE Communications Surveys Tutorials* 16:1, pp. 303–336 (cit. on p. 40).

A. Bifet and R. Gavalda (2007). "Learning from Time-Changing Data with Adaptive Windowing". In: *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, pp. 443–448 (cit. on pp. 14, 23).

D. A. Bodenham and N. M. Adams (2017). "Continuous monitoring for changepoints in data streams using adaptive estimation". *Statistics and Computing*, pp. 1–14 (cit. on pp. xiii, xiv, 18, 19, 23, 26, 34, 35, 40, 46, 47, 50–52, 59, 119).

D. Bodenham (2014). "Adaptive estimation with change detection for streaming data". PhD thesis. London, UK: Imperial College London (cit. on pp. 2, 16–18, 24, 58, 59, 69–72, 75).

R. G. Brown (1956). "Exponential Smoothing for Predicting Demand". In: *Tenth National Meeting of the Operations Research Society of America, Florida, US, November 15–16, 1956* (cit. on p. 16).

R. G. Brown (1963). *Smoothing, Forecasting and Prediction of Discrete Time Series*. Prentice-Hall international series in management. Prentice-Hall (cit. on p. 16).

A. L. Buczak and E. Guven (2016). "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection". *IEEE Communications Surveys Tutorials* 18:2, pp. 1153–1176 (cit. on p. 40).

J. Cao, L. E. Li, A. Chen, and T. Bu (2009). "Incremental tracking of multiple quantiles for network monitoring in cellular networks". In: *Proceedings of the 1st ACM Workshop on Mobile Internet Through Cellular Networks*. ACM, pp. 7–12 (cit. on pp. xiv, 88, 125).

A. A. Cárdenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, and S. Sastry (2009). "Challenges for Securing Cyber Physical Systems". *Prevention* (cit. on pp. 19, 20).

K. M. Carter and W. W. Streilein (2012). "Probabilistic Reasoning for Streaming Anomaly Detection". In: *2012 IEEE Statistical Signal Processing Workshop (SSP)*, pp. 377–380 (cit. on p. 33).

F. Chen, D. Lambert, and J. C. Pinheiro (2000). "Incremental quantile estimation for massive tracking". In: *Proceeding of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 516–522 (cit. on pp. xiv, 87, 90–92, 126).

Y. Chen, A. Wiesel, Y. C. Eldar, and A. O. Hero (2010). "Shrinkage Algorithms for MMSE Covariance Estimation". *IEEE Transactions on Signal Processing* 58:10, pp. 5016–5029 (cit. on p. 45).

G. Cormode and S. Muthukrishnan (2005). "An improved data stream summary: the count-min sketch and its applications". *Journal of Algorithms* 55:1, pp. 58–75 (cit. on p. 87).

D. Cox and D. Hinkley (1979). *Theoretical Statistics*. Chapman & Hall/CRC (cit. on pp. 7, 11, 61).

R. A. Fisher (1915). “Frequency Distribution of the Values of the Correlation Coefficient in Samples from an Indefinitely Large Population”. *Biometrika* 10:4, pp. 507–521 (cit. on p. 48).

T. Fortescue, L. Kershenbaum, and B. Ydstie (1981). “Implementation of self-tuning regulators with variable forgetting factors”. *Automatica* 17:6, pp. 831–835 (cit. on p. 16).

J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia (2014). “A survey on concept drift adaptation”. *ACM Computing Surveys* 46:4, pp. 1–37 (cit. on p. 120).

P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez (2009). “Anomaly-based network intrusion detection: Techniques, systems and challenges”. *Computers & Security* 28:1, pp. 18–28 (cit. on p. 40).

W. Gilchrist (2000). *Statistical modelling with quantile functions*. CRC Press (cit. on p. 11).

T. Gneiting (2011). “Making and evaluating point forecasts”. *Journal of the American Statistical Association* 106:494, pp. 746–762 (cit. on p. 98).

M. Greenwald, S. Khanna, et al. (2001). “Space-efficient online computation of quantile summaries”. *ACM SIGMOD Record* 30:2, pp. 58–66 (cit. on p. 86).

H. L. Hammer and A. Yazidi (2017). “Incremental quantiles estimators for tracking multiple quantiles”. In: *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, pp. 202–210 (cit. on pp. xiv, 66, 90, 93, 94, 125).

H. L. Hammer, A. Yazidi, and H. Rue (2019). “A new quantile tracking algorithm using a generalized exponentially weighted average of observations”. *Applied Intelligence* 49:4, pp. 1406–1420 (cit. on pp. 66, 87, 90, 95, 97, 100, 101, 120, 122).

H. L. Hammer, A. Yazidi, and H. Rue (2020). “Tracking of multiple quantiles in dynamically varying data streams”. *Pattern Analysis and Applications* 23:1, pp. 225–237 (cit. on pp. 88, 125).

H. L. Hammer, A. Yazidi, and H. Rue (2021). “Joint tracking of multiple quantiles through conditional quantiles”. *Information Sciences* 563, pp. 40–58 (cit. on pp. 66, 88, 121, 123, 126).

S. Haykin (2002). *Adaptive Filter Theory*. Prentice-Hall information and system sciences series. Prentice Hall (cit. on pp. 1, 14, 23, 66).

R. J. Hyndman and Y. Fan (1996). “Sample Quantiles in Statistical Packages”. *The American Statistician* 50:4, pp. 361–365 (cit. on p. 11).

B. Jackson, J. D. Scargle, D. Barnes, S. Arabhi, A. Alt, P. Gioumousis, E. Gwin, P. Sangtrakulcharoen, L. Tan, and T. T. Tsai (2005). “An Algorithm for Optimal Partitioning of Data on an Interval”. *IEEE Signal Processing Letters* 12:2, pp. 105–108 (cit. on p. 37).

R. Jain and I. Chlamtac (1985). “The P^2 algorithm for dynamic calculation of quantiles and histograms without storing observations”. *Communications of the ACM* 28:10, pp. 1076–1085 (cit. on p. 86).

A. D. Kent (2015). *Comprehensive, Multi-Source Cyber-Security Events*. Los Alamos National Laboratory (cit. on pp. xiv, 53).

D. Kifer, S. Ben-David, and J. Gehrke (2004). “Detecting change in data streams”. In: *VLDB*. Vol. 4. Toronto, Canada, pp. 180–191 (cit. on pp. 2, 28).

R. Killick and I. Eckley (2014). “changepoint: An R Package for Changepoint Analysis”. *Journal of Statistical Software* 58:1, pp. 1–19 (cit. on pp. 36, 51).

R. Killick, P. Fearnhead, and I. A. Eckley (2012). “Optimal Detection of Changepoints with a Linear Computational Cost”. *Journal of the American Statistical Association* 107:500, pp. 1590–1598 (cit. on pp. 34, 51).

S. Kullback and R. A. Leibler (1951). “On Information and Sufficiency”. *The Annals of Mathematical Statistics* 22:1, pp. 79–86 (cit. on p. 12).

H. J. Kushner and J. Yang (1995). “Analysis of adaptive step-size SA algorithms for parameter tracking”. *IEEE Transactions on Automatic Control* 40:8, pp. 1403–1410 (cit. on p. 16).

J. R. Kwapisz, G. M. Weiss, and S. A. Moore (2011). “Activity recognition using cell phone accelerometers”. *ACM SIGKDD Explorations Newsletter* 12:2, pp. 74–82 (cit. on p. 13).

A. Lall (2015). “Data Streaming Algorithms for the Kolmogorov-Smirnov Test”. In: *2015 IEEE International Conference on Big Data*, pp. 95–104 (cit. on p. 28).

J. C. Liechty, D. K. Lin, and J. P. McDermott (2003). “Single-pass low-storage arbitrary quantile estimation for massive datasets”. *Statistics and Computing* 13:2, pp. 91–100 (cit. on p. 88).

O. Linda, T. Vollmer, and M. Manic (2009). “Neural Network based Intrusion Detection System for critical infrastructures”. In: *2009 International Joint Conference on Neural Networks*, pp. 1827–1834 (cit. on p. 20).

A. Y. Lokhov, N. Lemons, T. C. McAndrew, A. Hagberg, and S. Backhaus (2016). “Detection of Cyber-Physical Faults and Intrusions from Physical Correlations”. In: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pp. 303–310 (cit. on pp. 19, 20).

P. Mair, K. Hornik, and J. de Leeuw (2009). “Isotone optimization in R: pool-adjacent-violators algorithm (PAVA) and active set methods”. *Journal of Statistical Software* 32:5, pp. 1–24 (cit. on p. 125).

J. P. McDermott, G. J. Babu, J. C. Liechty, and D. K. Lin (2007). “Data skeletons: simultaneous estimation of multiple quantiles for massive streaming datasets with applications to density estimation”. *Statistics and Computing* 17:4, pp. 311–321 (cit. on p. 88).

T. Minka (2002). *Estimating a Gamma Distribution*. Technical report. Microsoft Research (cit. on pp. 136, 137).

J. Mirkovic, G. Prier, and P. Reiher (2002). “Attacking DDoS at the source”. In: *10th IEEE International Conference on Network Protocols, 2002. Proceedings.* Pp. 312–321 (cit. on p. 20).

R. Mitchell and I.-R. Chen (2014). “A Survey of Intrusion Detection Techniques for Cyber-Physical Systems”. *ACM Computing Surveys (CSUR)* 46:4, p. 55 (cit. on p. 20).

R. Mitchell and R. Chen (2013). “Effect of Intrusion Detection and Response on Reliability of Cyber Physical Systems”. *IEEE Transactions on Reliability* 62:1, pp. 199–210 (cit. on p. 20).

J. Munro and M. Paterson (1980). “Selection and sorting with limited storage”. *Theoretical Computer Science* 12:3, pp. 315–323 (cit. on p. 86).

G. Munz and G. Carle (2007). “Real-time Analysis of Flow Data for Network Attack Detection”. In: *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 100–108 (cit. on p. 20).

S. Muthukrishnan et al. (2005). “Data streams: Algorithms and applications”. *Foundations and Trends® in Theoretical Computer Science* 1:2, pp. 117–236 (cit. on p. 1).

J. Neil, C. Hash, A. Brugh, M. Fisk, and C. B. Storlie (2013). “Scan Statistics for the Online Detection of Locally Anomalous Subgraphs”. *Technometrics* 55:4, pp. 403–414 (cit. on pp. 39, 42, 49).

J. Noble and N. Adams (2018). “Real-Time Dynamic Network Anomaly Detection”. *IEEE Intelligent Systems* 33:2, pp. 5–18 (cit. on pp. xv, 3, 4, 39).

J. Noble and N. M. Adams (2016). “Correlation-Based Streaming Anomaly Detection in Cyber-Security”. In: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pp. 311–318 (cit. on pp. xv, 40, 50, 53).

S. Y. Nof (2009). *Springer Handbook of Automation*. 1st. Springer (cit. on p. 15).

A. Osorio Cordero and D. Q. Mayne (1981). “Deterministic convergence of a self-tuning regulator with variable forgetting factor”. *IEEE Proceedings D - Control Theory and Applications* 128:1, pp. 19–23 (cit. on p. 16).

E. S. Page (1954). “Continuous Inspection Schemes”. *Biometrika* 41:1-2, p. 100 (cit. on pp. xiii, 34, 47, 50).

F. Pasqualetti, F. Dörfler, and F. Bullo (2013). “Attack Detection and Identification in Cyber-Physical Systems”. *IEEE Transactions on Automatic Control* 58:11, pp. 2715–2729 (cit. on p. 20).

N. G. Pavlidis, D. K. Tasoulis, N. M. Adams, and D. J. Hand (2011). “λ-Perceptron: An adaptive classifier for data streams”. *Pattern Recognition* 44:1, pp. 78–96 (cit. on pp. 2, 23).

J. Plasse (2019). “Adaptive estimation of categorical data streams with applications in change detection and density estimation” (cit. on pp. 18, 84, 133).

J. Plasse, J. Noble, and K. Myers (2017). “An Adaptive Modeling Framework for Bivariate Data Streams with Applications to Change Detection in Cyber-Physical Systems”. In: *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 1074–1081 (cit. on pp. 4, 19).

S. Rajasegarar, C. Leckie, M. Palaniswami, and J. C. Bezdek (2006). “Distributed Anomaly Detection in Wireless Sensor Networks”. In: *2006 10th IEEE Singapore International Conference on Communication Systems*, pp. 1–5 (cit. on p. 20).

D. M. dos Reis, P. Flach, S. Matwin, and G. Batista (2016). "Fast Unsupervised Online Drift Detection Using Incremental Kolmogorov-Smirnov Test". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. ACM, San Francisco, California, USA, pp. 1545–1554 (cit. on p. 28).

H. Robbins and S. Monro (1951). "A stochastic approximation method". *The Annals of Mathematical Statistics*, pp. 400–407 (cit. on pp. xv, 87).

S. Roberts (1959). "Control chart tests based on geometric moving averages". *Technometrics* 1:3, pp. 239–250 (cit. on pp. xiv, 17, 87).

P. Rubin-Delanchy, D. J. Lawson, and N. A. Heard (2016). "Anomaly detection for cyber security applications". In: *Dynamic Networks and Cyber-Security*. World Scientific, pp. 137–156 (cit. on p. 2).

B. Sheng, Q. Li, W. Mao, and W. Jin (2007). "Outlier Detection in Sensor Networks". In: *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, pp. 219–228 (cit. on p. 20).

R. Sommer and V. Paxson (2010). "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection". In: *2010 IEEE Symposium on Security and Privacy*, pp. 305–316 (cit. on p. 50).

P. D. Talagala, R. J. Hyndman, K. Smith-Miles, S. Kandanaarachchi, and M. A. Muñoz (2020). "Anomaly detection in streaming nonstationary temporal data". *Journal of Computational and Graphical Statistics* 29:1, pp. 13–27 (cit. on p. 2).

A. Teixeira, S. Amin, H. Sandberg, K. H. Johansson, and S. S. Sastry (2010). "Cyber Security Analysis of State Estimators in Electric Power Systems". In: *49th IEEE Conference on Decision and Control (CDC)*, pp. 5991–5998 (cit. on p. 20).

R. Tibshirani (1996). "Regression Shrinkage and Selection via the Lasso". *Journal of the Royal Statistical Society. Series B (Methodological)* 58:1, pp. 267–288 (cit. on p. 121).

L. Tierney (1983). "A space-efficient recursive procedure for estimating a quantile of an unknown distribution". *SIAM Journal on Scientific and Statistical Computing* 4:4, pp. 706–711 (cit. on pp. xv, 87, 91, 92).

M. Turcotte, N. Heard, and J. Neil (2014). "Detecting localised anomalous behaviour in a computer network". In: *International Symposium on Intelligent Data Analysis*. Springer, pp. 321–332 (cit. on p. 2).

G. Widmer and M. Kubat (1996). "Learning in the presence of concept drift and hidden contexts". *Machine Learning* 23:1, pp. 69–101 (cit. on p. 66).

W. Wu, X. Cheng, M. Ding, K. Xing, F. Liu, and P. Deng (2007). "Localized Outlying and Boundary Data Detection in Sensor Networks". *IEEE Transactions on Knowledge and Data Engineering* 19:8, pp. 1145–1157 (cit. on p. 20).

A. Yazidi and H. Hammer (2016). "Quantile estimation in dynamic and stationary environments using the theory of stochastic learning". *ACM SIGAPP Applied Computing Review* 16:1, pp. 15–24 (cit. on pp. xv, 87).

A. Yazidi and H. Hammer (2017). "Multiplicative update methods for incremental quantile estimation". *IEEE Transactions on Cybernetics* 99, pp. 1–10 (cit. on pp. xiv, 87, 93, 125).

F. Zamora-Martínez, P. Romeu, P. Botella-Rocamora, and J. Pardo (2014). "On-line learning of indoor temperature forecasting models towards energy efficiency". *Energy and Buildings* 83, pp. 162–172 (cit. on p. 121).

N. R. Zhang and D. O. Siegmund (2007). "A modified Bayes information criterion with applications to the analysis of comparative genomic hybridization data". *Biometrics* 63:1, pp. 22–32 (cit. on pp. 37, 51).

Y. Zhang, N. Meratnia, and P. Havinga (2010). "Outlier Detection Techniques for Wireless Sensor Networks: A Survey". *IEEE Communications Surveys & Tutorials* 12:2, pp. 159–170 (cit. on p. 20).