Imperial College London

Temporally Adaptive Monitoring Procedures with Applications in Enterprise Cyber-Security

A thesis presented for the degree of
Doctor of Philosophy of Imperial College London
and the
Diploma of Imperial College
by

Elizabeth Riddle-Workman

Department of Mathematics
Imperial College
180 Queens Gate, London SW7 2AZ

March 2022

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.

Elizabeth Riddle-Workman
March 2022

Copyright

Acknowledgements

I would like to thank my supervisors Professor Niall Adams and Dr Marina Evangelou for guiding me throughout the PhD. It has been a great experience working with you both over the past 5 years which I will forever look back on fondly.

I am also grateful for the studentship from The Engineering and Physical Sciences Research Council (EPSRC) Centres for Doctoral Training (CDT) in Financial Computing & Analytics, for allowing me to carry out my research.

Finally I thank my friends and family for their support and encouragement especially during the challenging periods to keep me going throughout the PhD.

Abstract

Due to the perpetual threat of cyber-attacks, enterprises must employ and develop new methods of detection as attack vectors evolve and advance. Enterprise computer networks produce a large volume and variety of data including univariate data streams, time series and network graph streams. Motivated by cyber-security, this thesis develops adaptive monitoring tools for univariate and network graph data streams, however, they are not limited to this domain.

In all domains, real data streams present several challenges for monitoring including trend, periodicity and change points. Streams often also have high volume and frequency. To deal with the non-stationarity in the data, the methods applied must be adaptive. Adaptability in the proposed procedures throughout the thesis is introduced using *forgetting factors*, weighting the data accordingly to recency. Secondly, methods applied must be computationally fast with a small or fixed computation burden and fixed storage requirements for timely processing. Throughout this thesis, sequential or sliding window approaches are employed to achieve this.

The first part of the thesis is centred around univariate monitoring procedures. A sequential adaptive parameter estimator is proposed using a Bayesian framework. This procedure is then extended for multiple change point detection, where, unlike existing change point procedures, the proposed method is capable of detecting abrupt changes in the presence of trend. We additionally present a time series model which combines short-term and long-term behaviours of a series for improved anomaly detection. Unlike existing methods which primarily focus on point anomalies detection (extreme outliers), our method is capable of also detecting contextual anomalies, when the data deviates from persistent patterns of the series such as seasonality.

Finally, a novel multi-type relational clustering methodology is proposed. As multiple relations exist between the different entities within a network (computers, users and ports), multiple network graphs can be generated. We propose simultaneously clustering over all graphs to produce a single clustering for each entity using Non-Negative Matrix Tri-Factorisation. Through simplifications, the proposed procedure is fast and scalable for large network graphs. Additionally, this methodology is extended for graph streams.

This thesis provides an assortment of tools for enterprise network monitoring with a focus on adaptability and scalability making them suitable for intrusion detection and situational awareness.

Contents

Chapter 1

Introduction

The number of cyber attacks has steadily increased in the past decade [Morgan et al., 2016, Brooks, 2021] leading to a demand for new methods of intrusion detection. These cyber attacks come with large costs to companies and enterprises, where the cost of cyber-crime has increased to $13 million per organisation in 2018 [Bissell et al., 2019]. Since the start of the pandemic, the FBI reported a 300% increase in cyber-crime [Walter, 2020]. These statistics show the demand for improved intrusion detection methodologies which is the aim of this thesis.

Enterprise computer networks generate large numbers of data streams detailing the activities of computers and users on the network. By monitoring these data streams, unusual behaviour and abnormalities corresponding to adversaries may be detected. Existing intrusion detection approaches primarily rely on signature-based detection [Turcotte et al., 2019], identifying known signatures or patterns of analysed attacks. These methods are however incapable of detecting new attack vectors. As adversaries are becoming increasingly sophisticated and are continuously adapting their attack methods, it is insufficient to employ signature-based methods alone and complementary statistical based approaches should additionally be applied. Statistical approaches instead work by modelling the "normal" behaviour of the network where deviations away from the expected pattern are classified as anomalous allowing for the detection of previously unseen attack vectors. This thesis provides a variety of statistical monitoring tools for enterprise cyber-security data to assist analysts in identifying intruders through both situational awareness and detection of abnormalities in the data.

Although cyber-security is the key motivator for the procedures proposed in this thesis, they are not limited to this domain. In each chapter the methods are formulated for general application and are applied to non-cyber applications. In particular, the proposed approaches in this thesis focus on applicability to high frequency and volume data streams (unending sequences of ordered observations [Henzinger et al., 1999]) which are present in many areas of research due to the rapid advancements in data acquisition technology [Aggarwal, 2007, Gama, 2010]. The additional model challenges that are paired with high volume and frequency

data streams are the same across all domains and the methods proposed in this thesis aim to address these challenges. Due to the large volume of this data, it is impractical for the full history of a stream to be stored, and instead, it is desirable for each instance of the sequence to be processed once or passed over a small number of times using sequential or online approaches due to computational and storage limitations. In the landscape of streaming data, swift processing is imperative to support timely action. These requirements drastically reduce the number of methods suitable for data streams. In many of the research areas explored in this thesis, well established batch approaches exist however they are impractical for infinite data streams as they either process the full available data sequence or large batches. Thus in this thesis, there is a focus on online or sequential methods to allow for real-time analysis.

Another key focus of this work is adaptability. As real world data streams are never ending with unknown future behaviour, methods applied must be capable of adapting to changes in the underlying generating process of the data where the current model may not be suitable in the future. To introduce adaptability in the proposed methods, *forgetting factors* are employed throughout the thesis. These factors smoothly down-weight historical observations exponentially such that the data is weighted according to recency with the most recent data point having a weight of 1. Forgetting factors are a popular approach in adaptive filtering [Haykin, 2002] and have successfully been applied in literature for adaptability (e.g. Anagnostopoulos et al. [2012], Bodenham and Adams [2017], Pavlidis et al. [2011], Plasse and Adams [2019]). They can additionally be considered a continuous analogue of a sliding window approach, another popular mechanism for introducing a temporal aspect in a model. Due to the unlabelled nature of real world data streams, it is not possible to tune hyper-parameters on the fly, thus methods with few influential hyper-parameters are also preferable.

The thesis begins by investigating monitoring procedures for univariate data streams. For such data, various possible research problems exist. *Streaming Parameter Estimation, Forecasting, Change Point Detection* and *Anomaly Detection* are the problems investigated in this thesis. Parameter estimation and time series forecasting model the "normal" behaviour of the data whereas change point detection and anomaly detection allow for unusual behaviours in the data to be identified, possibly relating to intrusions. Here motivation and high-level descriptions of the proposed procedures are provided. Relevant literature and in-depth descriptions are provided in the corresponding chapters.

To capture the evolution of a data stream, adaptive parameter estimation is employed to estimate the current distributional parameters of the underlying data generating process. Due to the dynamic nature of the data, a novel adaptive sequential updating approach is used. It is important to note that sequential parameter estimation on data streams is a challenging problem as the methods must adapt to unknown future behaviour where only a single pass over the data is possible. Adaptability is introduced in the proposed approach using forgetting

factors within a Bayesian estimation procedure to allow the parameter estimates to adapt autonomously with minimal human input over the stream. This approach is scalable for large data streams due to the fast sequential updates, which is often a concern for Bayesian techniques.

For the detection of abnormalities in these univariate data streams, a change point procedure based on the proposed parameter estimation method is developed where change points correspond to timestamps when distributional changes occur in the generating process. Commonly data streams naturally contain periods of trend which many existing change point procedures flag as perpetual changes. In this thesis, attention is restricted to detecting abrupt change points in data streams exhibiting trend to prevent large numbers of false positives throughout these trend periods.

An alternative approach to analysing data streams is through using time series analysis as the data is naturally ordered by time. Many time series in practice often exhibit daily patterns or seasonality which is persistent throughout the series and is a non-anomalous characteristic of the data. In this thesis, a forecast procedure is proposed which incorporates this persistent daily behaviour by combining short-term forecasts (looking at the local behaviours) and long-term forecasts (focusing on the seasonal behaviour of the data) for improved forecast performance. This forecast procedure is then utilised for anomaly detection.

Anomaly detection is a highly popular intrusion detection approach in literature. A point is classified as anomalous if it deviates significantly from the "normal" behaviour of the model. In the context of forecasting, anomalies occur when the forecast and corresponding observed value differ greatly. The most analysed type of anomaly is *point anomalies* when a single data instance deviates globally from the series. These can also be regarded as outliers in the data. Alternatively, the data may deviate from the expected seasonal behaviour of the data. These types of anomalies are classified as *contextual anomalies* as, given the context, the data exhibits unusual values. In literature, time series anomaly detection methods focus primarily on point anomalies. As the proposed forecast procedure fuses information about the local (short-term) and global (long-term) behaviours, contextual anomalies can also be detected and is a benefit of the procedure.

Cyber-security data also lends itself naturally to be represented as network graphs. These graphs detail the connections made between nodes of different entity types (e.g. computers, users and ports in the case of cyber-security) where multiple relational graphs for each entity type may exist. As interactions between the nodes occur over time, graph streams can be generated capturing different snapshots of these interactions. To benefit from having multiple relations for a single entity type, information between these graphs is fused using *Multi-Type Relational* clustering. In literature, it is however more common to cluster over each graph separately, without leveraging the shared latent behaviour between the graphs. The clustering approach proposed in this thesis aims to group nodes with similar network activity. This is distinct from community detection which instead

groups highly connected nodes with few connections to nodes outside of their community. Community detection in computer networks has been more commonly explored (e.g. Moradi et al. 2014, Sanna Passino and Heard 2020). The proposed clustering is instead based on the connection activity of the nodes. Due to the large network sizes in cyber-security, existing clustering procedures are slow to compute which may result in delays in the detection of abnormalities in the network. The proposed procedure is instead fast to compute on large graph networks, allowing for timely analysis. An extension of the proposed clustering procedure to allow for dynamic clustering is provided, which has rarely been applied in the cyber domain.

1.1 Thesis Contributions

A novel Bayesian adaptive forgetting factor parameter estimation procedure is proposed in **Chapter 3**. This model utilises power priors to include forgetting factor weights within the Bayesian model and is suitable for all members of the exponential family of distributions. Sequential updating forms are calculated reducing storage requirements with fast computation making it highly suitable for application to data streams. Through the inclusion of the forgetting factor within the parameter inference, this factor is automatically estimated within the model and is adaptive. The proposed formulation only requires setting a fixed prior for the forgetting factor which can be tuned to the desired level of forgetting in the model. In comparison, competitive approaches often have hard to set hyper-parameters without intuitive meanings. The proposed parameter estimation procedure is tested and compared against the frequentist version proposed by Anagnostopoulos et al. 2012 on simulated Gaussian and Poisson data. This model forms the basis of the change point procedure proposed in the next chapter.

The parameter estimation procedure from Chapter 3 is extended in **Chapter 4** for multiple change point detection. The change point procedure, *BFF*, calculates p-values for each new observation using the estimated distribution from the previous time. Before applying a threshold approach to detect changes using the p-values, a sliding calibration window is applied to the p-values such that at the $\alpha\%$ level, approximately $\alpha\%$ of points are identified as change points. This calibration approach allows for setting the threshold based on the desired false positive rate. The performance of the procedure is compared to competitive change point approaches on both simulated and real financial data. The proposed approach benefits from being capable of detecting abrupt changes in the presence of trend, where trend is seldom addressed by existing procedures. Additionally, the proposed approach has comparable performance to high performing batch and sequential methods whilst maintaining low false positive rates across all data examples.

Two anomaly detection procedures, FDARIMA and SL Decomposition are proposed in **Chapter 5** which instead monitor the data using time series approaches. These approaches combine short-term and long-term forecasts for improved de-

tection of both point and contextual anomalies. A forgetting factor approach is utilised within the long-term model giving greater weight to more recent data, making it adaptive. Detection of contextual anomalies in literature is rare with the majority of work focusing on point anomalies. The proposed anomaly detector is compared to competitive anomaly detection approaches on simulated data and a real social media data set where the proposed procedure shows promising results in detecting both types of anomalies.

Chapter 6 introduces a novel multi-type relational clustering procedure, *Simple NMTF*, for clustering over multiple entity types and relational graphs simultaneously to produce a shared representation. Existing multi-relational clustering approaches e.g. [Wang et al., 2011b] and [Wang et al., 2019] are slow to compute, making them infeasible for application to large scale data sets. By utilising indicator matrices for the factor matrices within Non-Negative Matrix Tri-Factorisation, the computation time is significantly reduced whilst improving clustering performance. A novel extension of a popular uni-partite cluster initialisation procedure is proposed for multi-type relational clustering for improved stability. As the majority of real data sets are unlabelled, external cluster performance measures are necessary to measure the effectiveness of the clustering. The Node Cluster Similarity measure is proposed which is suitable for assessing the cluster performance at node, graph and network levels. The proposed procedure shows promising results on both simulated and real data sets with significant improvements in computation speed in comparison to existing procedures. This procedure has been published and is applied to both cyber and non-cyber applications in [Riddle-Workman et al., 2021].

The clustering procedure from Chapter 6 is extended for application to graph streams in **Chapter 7**. At each update, the previously calculated clustering is utilised to initialise the clustering procedure. Consistent performance and smooth clustering results are observed from application to simulated graph streams which are desired properties for dynamic clustering procedures.

In **Chapter 8** the proposed methods are applied to the comprehensive Unified Host and Network Data set from Los Alamos National Laboratory [Turcotte et al., 2019]. The procedures developed in this thesis provide an extensive toolkit for monitoring computer networks including monitoring both univariate and graph data streams. Performance measurement for each procedure presented in comparison to competitive approaches is provided. This chapter showcases the robustness of the proposed procedures to real network data, highlighting its suitability in practice for enterprise network monitoring, the motivator for the methods proposed.

1.2 List of Publications Based on this Work

The following paper has been accepted for publication:

Elizabeth Riddle-Workman, Marina Evangelou, and Niall M. Adams. Multi-Type Relational Clustering for Enterprise Cyber-Security Networks. Pattern Recognition Letters, 149:172-178, 2021. doi:10.1016/j.patrec.2021.05.021

The multi-type relational clustering work from Chapter 6 is based on the above paper and is extended further for dynamic clustering in Chapter 7. The following papers are due to be submitted for publication:

Riddle-Workman, E., Evangelou, M., & Adams, N. M. (2022). Combining Forecasts For Enhanced Anomaly Detection in Time Series Data. Journal of Computational and Graphical Statistics.

Riddle-Workman, E., Evangelou, M., & Adams, N. M. (2022). Adaptive Sequential Bayesian Filtering for Trend Resistant Change Point Detection. Sequential Analysis.

Additionally, all methodologies presented in this thesis have associated public Github repositories:

- elizabethriddle/BFF for Chapters 3 and 4
- elizabethriddle/tsanom for Chapter 5
- elizabethriddle/Simple-NMTF for Chapters 6 and 7.

Chapter 2

Background

This chapter details standard definitions and procedures utilised throughout the thesis. To begin, notation used in all chapters is defined. Forgetting factors, an approach central to this thesis, is described in Section 2.1 where Bayesian filtering is additionally described. Basic prerequisites for time series analysis, p-values, performance measures and network graph representation are detailed in Sections 2.2, 2.3, 2.4 and 2.5 respectively.

Throughout the subsequent chapters, let uppercase values $X_1, \ldots, X_t$ represent random variables with corresponding observations $x_1, \ldots, x_t$. Let a time series be represented using uppercase, $X_1, \ldots, X_t$ which is common in literature. Vectors are represented with bold symbols $\mathbf{x}$ where $\mathbf{x}_i$ refers to the ith entry of the vector. Matrices are represented by bold uppercase letters $\mathbf{A}$ which is indexed as $\mathbf{A}^{ij}$, referring to the $(i, j)^{th}$ position. Additionally let $\mathbf{A}^{i\cdot}$ refer to the ith row of $\mathbf{A}$ and $\mathbf{A}^{\cdot j}$ be the corresponding j^{th} column. Throughout the thesis it is noted if the notation deviates to instead follow popular notation from literature.

2.1 Adaptive Estimation

Adaptability is a key characteristic of the methods presented in this thesis as they are applied to unending sequences of data, also known as data streams. Static models are not suitable for such data as the generating process of the data often changes. Thus adaptive procedures are beneficial for improving the accuracy of the models. A temporal aspect can be introduced in a model by applying a sliding window during estimation [Plasse and Adams, 2019]. A sliding window utilises the latest s observations for estimation where s corresponds to the size of the sliding window. Setting the size of this window can however lead to further challenges where a fixed window might not be appropriate for all situations in streaming data. For example, during stationary periods, larger windows are appropriate however during trends or other non-stationary periods, smaller windows are preferred as historic data is less relevant to the current model. Throughout the thesis *forgetting factors* are instead employed to control the effective "memory" of our models and is

a popular approach in adaptive filtering [Haykin, 2002]. We now describe forgetting factors in more detail.

2.1.1 Forgetting Factors

Forgetting factors are commonly represented by λ, taking on values in $[0, 1]$. Here an exponential weighting framework is employed where historic data is discounted by λ at each time. This approach weights the data according to recency where younger samples have greater influence. Under this scheme, for a *fixed* forgetting factor, λ, which stays constant over the stream, after observing a total of t data points, the ith data point has weight given by λ^{t-i}. Thus the most recent data point has weight 1 whereas older samples have progressively less weight.

Alternatively, an *adaptive* forgetting factor approach can be employed where at time t, historic data is discounted by λ_{t-1} whose value changes over the stream. After observing a total of t data points, the ith data point has weight given by $\prod_{j=i}^{t-1} \lambda_j$. Again data point t, the most recent observation has weight 1. This adaptive forgetting approach is more favourable in practice to better adapt to the data. For example, during periods of change, smaller values are appropriate whereas, during stationary periods, values close to 1 are preferable. For both the fixed and adaptive cases, when $\lambda_t = 1 \ \forall t$, the estimates are equivalent to the unweighted case. Forgetting factors are used to improve the adaptability of the methods in Chapters 3 and 5.

2.1.2 Sequential Bayesian Updating

When dealing with data streams models need to update sequentially to prevent large computation overheads. Sequential updating forms for various models are available making it possible for models to be updated without full re-computation of the model. The simple Bayesian sequential updating filtering procedure is an example that efficiently updates a Bayesian model without requiring the full stream of data to be stored. In Chapter 3, we propose introducing forgetting factors into this sequential Bayesian updating method to allow the influence of historic data to reduce over the stream as the data changes.

Consider a data stream $x_1, \ldots, x_t, \ldots$ generated from random variables $X_1, \ldots, X_t, \ldots$ where the aim is to estimate the distribution parameter θ. Let the prior be represented by $\pi(\theta)$. The likelihood for the data at time t can be expressed as,

$$P(x_1, \ldots, x_t|\theta) = P(x_1|\theta)P(x_2|x_1, \theta) \ldots P(x_t|x_1, \ldots, x_{t-1}, \theta).$$

Let the posterior at time t be given by,

$$\pi_t(\theta) \propto \pi(\theta)P(x_1, \ldots, x_t|\theta)$$

which can be expressed in recursive terms as follows,

$$\begin{aligned}\pi_t(\theta) &\propto \pi(\theta)P(x_1,\ldots,x_t|\theta)\\ &\propto \pi(\theta)P(x_1,\ldots,x_{t-1}|\theta)P(x_t|x_1,\ldots,x_{t-1},\theta)\\ &\propto \pi_{t-1}(\theta)P(x_t|x_1,\ldots,x_{t-1},\theta)\end{aligned}$$

where $P(x_t|x_1,\ldots,x_{t-1},\theta)$ is the new likelihood for the most recent data point. Commonly $P(x_t|x_1,\ldots,x_{t-1},\theta) = P(x_t|\theta)$ through assuming independence between observations. Thus sequential forms for Bayesian inference can be formulated to allow for online implementation which we extend to include forgetting factors in Chapter 3.

2.2 Time Series Analysis

The order of the data points in cyber-security holds great importance, particularly for intrusion detection such that times of attacks can be identified. These data streams can be represented as time series and are often high frequency. A time series is a collection of data points indexed by their order. In all chapters but Chapter 6, time series data is investigated. Chapter 5 in particular applies univariate time series methods directly to the data including ARIMA and time series decomposition methods.

2.2.1 Autoregressive Integrated Moving Average (ARIMA)

An ARIMA process is a mixed model which combines Autoregressive (AR) and Moving Average (MA) processes into a single expression with the addition of differencing in the time series [Metcalfe and Cowpertwait, 2009] where this differencing allows for non-stationary time series. For a time series of data, X_t with integer index, an ARIMA(p,d,q) process can be expressed mathematically by,

$$\Theta_p(\mathbf{B})(1-\mathbf{B})^d X_t = \Phi_q(\mathbf{B})\epsilon_t, \tag{2.1}$$

where $\mathbf{B}$ is the backwards shift operator, ϵ_t is a zero mean white noise process with variance σ_ϵ. Θ_p and Φ_q are polynomials of orders p and q respectively, more specifically:

$$\begin{aligned}\Theta_p(\mathbf{B}) &= 1-\theta_1\mathbf{B}-\theta_2\mathbf{B}^2-\ldots-\theta_p\mathbf{B}^p\\ \Phi_q(\mathbf{B}) &= 1-\phi_1\mathbf{B}-\phi_2\mathbf{B}^2-\ldots-\phi_p\mathbf{B}^q.\end{aligned}$$

These coefficients are estimated using maximum likelihood estimation. For further details on calculating these coefficients see [Gardner et al., 1980]. From these models, forecasts can be made using the estimated coefficients, the past time series values and residuals. An ARIMA model is applied to model short-term aspects of

the data in Chapter 5.

2.2.2 Seasonal Trend Decomposition

Time series data is often subject to seasonality, trend and other patterns. Data of this kind is the focus of Chapter 5. For such series, time series decomposition approaches that separate these underlying patterns are often employed. The most widely used decomposition is into trend, seasonal and remainder components as follows,

$$x_t = T_t + S_t + R_t$$

where x_t is the observed value of the time series, T_t is the trend component, S_t is the seasonal component and R_t is the remainder at time t. Traditionally the trend component is first computed before the seasonal component [Hyndman and Athanasopoulos, 2018] using methods such as a moving average. The seasonal component is then computed on the detrended series $x_t - \hat{T}_t$ where $\hat{T}_t$ is the forecast of the trend at time t. The seasonal component can then be calculated by averaging over the detrended values for each value within the seasonal cycle.

2.3 *P*-Values

Intrusion detection in cyber-security data is a key motivator for the methods developed in this thesis where p-values are used to characterise the anomalousness of the data at each time point. Traditionally p-values are used for statistical hypothesis testing and express the probability of observing something at least as extreme as the observed result under the null hypothesis. These values are then used to accept or reject this hypothesis. Throughout the thesis, p-values are used less traditionally, as indicators of anomalies or abnormalities within the data. p-values are used here to quantify whether the newly observed data point is consistent with the current model of the data and take on values in $[0, 1]$ where values closer to 0 are more anomalous.

Suppose realisation x_{obs} is an observation of random variable X. Under the null it is assumed data from X has density $f(x; \theta)$. Given some test statistic $T = t(X)$ with observed value given by $t_{obs} = t(x_{obs})$ then the p-value to assess this null is calculated as,

$$p = P(t(X) > t_{obs}).$$

p-values are used as indicators of change points or anomalies in Chapter 4 and 5.

2.4 Performance Measures

The performance of the methods proposed in this thesis is assessed using several existing measures. Section 2.4.1 outlines estimation performance measures which are used for the adaptive parameter estimator in Chapter 3. The forecasts from

Chapter 5 are assessed using the forecast measures in Section 2.4.2. Finally, to assess the performance of the change point and anomaly detection procedures of Chapters 4 and 5, classification measures are applied and are described in Section 2.4.3. These measures are now described.

2.4.1 Adaptive Estimation Performance Measures

The most popular estimation error measures are Mean Square Error (MSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE). These measures look to evaluate the average estimation error over the observed data.

The MSE for true values y_i and estimates $\hat{y}_i$ over $i = 1, \ldots, n$ is calculated as,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

The MAE is calculated in a similar manner where the square is replaced by the absolute value,

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|.$$

Finally, the MAPE is similar to the MAE but scales the error by the true value as follows,

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|.$$

All of these measures are positive where smaller values are desirable. The MSE is advantageous in situations where large errors are undesirable as it gives greater weight to these. The MAE however has the benefit of being more interpretable as it is on the same scale as the data. The MAPE scales the error by the value that is being estimated. This measure is then robust to changes in the scale of the data.

2.4.2 Forecasting Performance Measures

Time series models are commonly utilised to forecast the values of future values in the series. Hyndman, 2006 discusses various measures appropriate for forecast performance in detail which include the Mean Absolute Error (MAE) and Mean Absolute Scaled Error (MASE). Their notation is adopted for this section. The MAE is calculated using the following where F_t is the forecast for observation Y_t, the observed value of the process at time t where $t = 1, \ldots, n$,

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|Y_t - F_t|.$$

A disadvantage of this measure is that it is scale dependent and cannot be easily compared across different series. To overcome this, the MASE was introduced by Hyndman and Koehler [2006] which scales the errors using the in-sample MAE of a naive one-step ahead forecaster. The scaled error for a forecast at time t is,

$$q_t = \frac{|Y_t - F_t|}{\frac{1}{n-1}\sum_{i=2}^{n}|Y_i - Y_{i-1}|}.$$

The mean absolute scaled error is calculated as,

$$MASE = \frac{1}{n-1}\sum_{t=2}^{n}(q_t),$$

where a value less than 1 signifies a forecast that performs better than the in-sample naive forecaster and similarly if the value is greater than 1, it performs worse than this naive forecaster. Both measures are utilised as they can indicate different performances for the models. The MASE can in particular allow for better comparison between methods that utilise different window sizes of data for building the model.

2.4.3 Classification Performance Measures

When determining the performance of anomaly detectors, classification performance measures are commonly employed. These measures assess how well a model correctly distinguishes between two classes which in this case is anomalous and non-anomalous. Below basic definitions for distinguishing between the proportion of anomalies correctly and incorrectly classified are detailed:
True Positive (TP) corresponds to the number of true anomalies correctly identified by the procedure
False Positive (FP) corresponds to the number of points that are incorrectly identified as anomalous by the procedure
True Negative (TN) corresponds to the number of points that are correctly identified as non-anomalous by the procedure
False Negative (FN) corresponds to the number of anomalies incorrectly classified as non-anomalous by the procedure

Precision and recall are two popular classification measures that evaluate different aspects of performance. *Precision* is the proportion of identified positives that are true positives. The *recall* instead looks at the proportion of true positives

that are correctly identified as such. More specifically,

$$\text{precision} = \frac{TP}{TP + FP}$$
$$\text{recall} = \frac{TP}{TP + FN}$$

Both of these measures take on values in $[0, 1]$ where values closer to 1 are optimal. If low false positives for a model are desirable, high precision is required. However, if detecting all positives has higher importance, higher recall is desirable. Often there is a trade off between these two measures where higher numbers of true positives often result in larger numbers of false positives. The *F1* score instead combines these measures into a single score. The *F1* measure takes the harmonic mean of the precision and recall and is calculated as,

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

The *F1* takes on values in $[0, 1]$ where again higher values are desirable where a value of 1 corresponds to perfect classification of the data with no false positives.

2.5 Graph Definitions

In Chapters 6 and 7, rather than investigating univariate data, network graphs are explored. Cyber-security data captures the relationships between entities within the network where these can naturally be represented as network graphs. Basic graph definitions used throughout the thesis are now briefly outlined. For more in depth descriptions see Gross and Yellen [2003] and Newman [2010].

A *graph* $G = (V, E)$ is a collection of **vertices**, V, also known as **nodes** that are connected by **edges**, E. An edge denoted by (i, j), connects nodes i and j where these connections can be directed or undirected. For directed graphs, the ordering of the pairs of vertices holds importance whereas in undirected graphs, it does not.

Graphs can be expressed mathematically using an *adjacency matrix*, $\mathbf{A}$, of size $|V| \times |V|$ given by,

$$\mathbf{A}^{ij} = \begin{cases} 1 & (i, j) \in E \\ 0 & \text{otherwise} \end{cases}.$$

In the case of undirected graphs, adjacency matrices are symmetric. Although weights can be given to the edges, in our work there is focus on the unweighted case where adjacency matrices are binary and primarily sparse.

Due to the complex nature of computer networks, relationships between two independent sets of vertices may exist, known as a *bi-partite* graph. Conversely,

when graphs contain a single vertex set, they are called *uni-partite*. Bi-partite graphs are denoted by $G = (V, U, E)$ where V and U are the distinct vertex sets and the edges, E, define the links between these sets. Adjacency matrices for bi-partite graphs have size $|V| \times |U|$. In bi-partite adjacency matrices, the direction of the connection cannot be recorded and thus is undirected. Finally, the *degree* of a vertex is defined as the number of edges attached to it.

Chapter 3

Adaptive Bayesian Filtering

Distribution parameter estimation for streaming data is a challenging problem as the underlying generating process of the data often changes over time. Thus adaptive parameter estimation procedures are preferable in this context. In this chapter a Bayesian adaptive parameter estimation procedure is presented, a novel extension of the frequentist filtering method proposed by Anagnostopoulos et al. [2012]. Within the proposed method, exponential weighting is employed through forgetting factors to smoothly down-weight historic data, allowing estimates to adapt at abrupt change points and during trends. Rather than estimating the parameters using maximum likelihood estimation, a Bayesian framework with sequential updating mechanisms is implemented for improved adaptability and scalability.

Besides the benefits of situational awareness and understanding of the data, adaptive filtering can be used as a basis for additional procedures such as change point detection or anomaly detection. The proposed adaptive filter is utilised for the change point procedure outlined in Chapter 4 and also within the forecast method in Chapter 5.

This chapter is structured as follows: Section 3.1 first describes the frequentist approach the proposed procedure is based on. Challenges faced by this frequentist model are outlined, motivating a Bayesian reformulation. The proposed novel Bayesian procedure is detailed in Section 3.2 including prior specification methods which help to alleviate prior parameter specification over the stream. Finally, Section 3.3 looks at the parameter estimation performance of the proposed procedure on simulated examples that include trend.

3.1 Maximum Likelihood Approach

Anagnostopoulos et al. [2012] propose an adaptive frequentist parameter estimation procedure that utilises adaptive filtering to temporally monitor the distribution parameters. Forgetting factors, a popular approach from adaptive filtering is used to weight the likelihood making it temporally aware. The distribution parameters are estimated using maximum likelihood estimation where sequen-

tial updating forms are described. This frequentist adaptive estimation approach shows great strength in filtering and is highly suitable for data streams due to its adaptive sequential nature however requires setting the forgetting factor. This procedure is now described before outlining the proposed reformulation in Section 3.2.

3.1.1 Adaptive Estimation

The procedure by Anagnostopoulos et al. [2012] adaptively estimates the parameters of a distribution by smoothly down-weighting past data points in the likelihood. As done in their paper, the case for multivariate Gaussian data is presented however Anagnostopoulos et al. [2012] additionally present the results more generally for exponential family of distributions. For multivariate Gaussian data $\boldsymbol{x}_i \in \mathbb{R}^p$ at time i with mean $\boldsymbol{\mu} \in \mathbb{R}^p$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$, at time t the exponentially weighted sum of log likelihood terms is given by,

$$\begin{aligned}
&\mathcal{L}_{FF}(\boldsymbol{x}_{1:t}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\
&= \lambda_{t-1}\mathcal{L}_{FF}(\boldsymbol{x}_{1:t-1}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) + \mathcal{L}(\boldsymbol{x}_t|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\
&= \lambda_{t-1}\sum_{i=1}^{t-1}\prod_{j=i}^{t-2}\lambda_j\mathcal{L}(\boldsymbol{x}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma}) + \mathcal{L}(\boldsymbol{x}_t|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\
&= \sum_{i=1}^{t-1} w_i^t\mathcal{L}(\boldsymbol{x}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma}) + \mathcal{L}(\boldsymbol{x}_t|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \qquad (3.1)
\end{aligned}$$

where $\mathcal{L}(\cdot|\cdot)$ is the Gaussian log likelihood, $\boldsymbol{x}_{1:t}$ refers to data from time 1 to time t, $\lambda_j \in [0, 1]$ are the forgetting factors and

$$w_i^t = \lambda_{t-1}\prod_{j=i}^{t-2}\lambda_j = \lambda_{t-1}w_i^{t-1}$$

is the unnormalised weight for observation $i = 1, \ldots, t-1$ at time t where observation t has weight $w_t^t = 1$.

This forgetting factor likelihood from Equation (3.1) has the following Maximum Likelihood Estimates (MLE),

$$\tilde{\boldsymbol{\mu}}_t = \sum_{i=1}^{t}\frac{w_i^t}{n_t}\boldsymbol{x}_i \qquad (3.2)$$

$$\tilde{\boldsymbol{\Pi}}_t = \sum_{i=1}^{t}\frac{w_i^t}{n_t}\boldsymbol{x}_i\boldsymbol{x}_i^T \qquad (3.3)$$

$$\tilde{\boldsymbol{\Sigma}}_t = \tilde{\boldsymbol{\Pi}}_t - \tilde{\boldsymbol{\mu}}_t\tilde{\boldsymbol{\mu}}_t^T \qquad (3.4)$$

where $n_t = \sum_{i=1}^{t} w_i^t$ is the effective sample size [Wang et al., 2004]. These equations have the benefit of sequential updating forms given by,

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_t &= \left(1 - \frac{1}{n_t}\right) \tilde{\boldsymbol{\mu}}_{t-1} + \frac{1}{n_t} \boldsymbol{x}_t \\
\tilde{\boldsymbol{\Pi}}_t &= \left(1 - \frac{1}{n_t}\right) \tilde{\boldsymbol{\Pi}}_{t-1} + \frac{1}{n_t} \boldsymbol{x}_t \boldsymbol{x}_t^T \\
\tilde{\boldsymbol{\Sigma}}_t &= \tilde{\boldsymbol{\Pi}}_t - \tilde{\boldsymbol{\mu}}_t \tilde{\boldsymbol{\mu}}_t^T \\
n_t &= \lambda_{t-1} n_{t-1} + 1
\end{aligned}$$

allowing for reduced storage requirements and fast computation. The only parameter that requires attention in this model is the forgetting factor λ_t that is crucial because it controls how fast historic data is forgotten.

Two cases are considered for these forgetting factors: fixed where $\lambda_i = \lambda \ \forall i$ and adaptive where this factor is updated at each iteration. In the fixed case, the data is down-weighted by a constant factor regardless of how the data has changed. Alternatively in the adaptive framework, the forgetting factor changes at each time point.

3.1.2 Tuning Forgetting Factors

The forgetting factor is responsible for determining the level of forgetting in the model. Larger values result in greater weight assigned to older samples and smaller values lead to less weight. When the data exhibits periods of change, the forgetting factor should be small to allow the estimates to adapt quickly to the new distribution, discarding irrelevant historic data. Whereas for periods when the data is stationary, a forgetting factor equal to one is optimal as all historic data has equal importance. To avoid manual specification, an online tuning procedure for the forgetting factor is necessary.

A stochastic gradient descent was used by Anagnostopoulos et al. [2012] for tuning the forgetting factor as follows,

$$\lambda_t = \lambda_{t-1} - \eta \frac{\partial}{\partial \lambda} J_t \tag{3.5}$$

where η is the learning rate or step size of the gradient descent. Various cost functions, J_t, may be used. Anagnostopoulos et al. [2012] use the one-step ahead fit for the cost function characterised by the negative log likelihood given by,

$$J_t = -\mathcal{L}(\boldsymbol{x}_{t+1} | \hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)$$

whereas Bodenham and Adams [2017] use the squared difference between the mean estimate and the true data as it is more appropriate for adaptive mean estimation

(without estimation of the variance) and is given by

$$J_t = ||\boldsymbol{x}_{t+1} - \hat{\boldsymbol{\mu}}_t||^2.$$

Bodenham, 2014 notes that under the assumption the stream is generated from a normal distribution, these two cost functions result in similar updates.

3.1.3 Criticism of Learning Rate

Although this procedure is suitable for streaming applications it has replaced the problem of choosing the forgetting factor with the learning rate. When choosing this learning rate, there is a trade-off between fast adaptability and noise control. Although a larger learning rate allows the forgetting factor to adapt quickly, this causes the estimates to be noisier, particularly if the data contains outliers. There is very little guidance in literature on setting the learning rate however values in $[0.001, 0.1]$ Bodenham and Adams, 2017 or $[10^{-8}, 10^{-6}]$ Anagnostopoulos et al. 2012 are suggested. These differences in appropriate ranges are attributed to different cost functions used within the stochastic gradient descent by the authors.

Tuning this learning rate for stochastic gradient descent is a well known challenge in literature and additional methods have been formulated to mitigate this issue (e.g. Zeiler, 2012). Additionally, it was found by Bodenham, 2014 that suitable learning rates were dependent on the scale of the data. To manage this problem, Bodenham, 2014 developed a heuristic approach that re-scales the learning rate by the variance of the data which they assume to be known or estimated.

Due to the challenges in setting the learning rate, we instead propose reformulating this model into a Bayesian framework. The adaptive Bayesian filtering approach does however require a prior for the forgetting factor to be supplied. The forgetting factor is optimised at each iteration within the inference rather than requiring additional adaptive models.

3.2 Bayesian Adaptive Filtering

The Bayesian approach to the adaptive filtering problem named the *Bayesian Forgetting Factor* (BFF) is now described. In the proposed approach the forgetting factor is estimated within the model, rather than being set manually or through additional procedures such as stochastic gradient descent.

3.2.1 Gaussian Adaptive Bayesian Forgetting Factor

The proposed novel Bayesian adaptive filtering approach is detailed for the univariate Gaussian case. Suppose a stream of data, $x_1, x_2, \ldots$ is observed which are realisations of Gaussian random variables $X_1, X_2, \ldots$ with unknown mean and variance. As before, there are two ways to include forgetting factors in the model.

A fixed forgetting factor represented by a single fixed value, λ, can be used to equally down-weight older samples as the model updates. Alternatively, an adaptive forgetting factor can be applied where historic data is down-weighted by an adapting factor λ_{t-1} at time t. Utilising an adaptive approach allows for more rapid adjustment at change points in the data.

In the following an adaptive forgetting factor is implemented. Let the weight for data point x_i at time t be expressed as,

$$\begin{aligned} w_i(t) &= \prod_{j=i}^{t-1} \lambda_j \\ &= \lambda_{t-1} \prod_{j=i}^{t-2} \lambda_j \end{aligned} \tag{3.6}$$

where x_t has weight 1 at time t. When $\lambda_j = 1 \ \forall j$, this is equivalent to the unweighted model. Similar to weighting the log likelihood in the frequentist case, the forgetting factors are incorporated within the prior using power priors as proposed by Ibrahim and Chen [2000]. Using the notation of Ibrahim and Chen [2000], the power prior for K_0 historical data sets, $D_{0k},\ k = 1 \ldots, K_0$, where the kth data set has weight a_{0k} $(0 \leq a_{0k} \leq 1)$, is given by,

$$\pi(\boldsymbol{\theta}|\mathbf{D}_0, \mathbf{a}_0) \propto \pi_0(\boldsymbol{\theta}) \prod_{k=1}^{K_0} L(\boldsymbol{\theta}|D_{0k})^{a_{0k}},$$

where $\mathbf{a}_0 = (a_{01}, \ldots, a_{0K_0})$, $\mathbf{D}_0 = (D_{01}, \ldots, D_{0K_0})$, $\pi_0(\boldsymbol{\theta})$ is the initial prior for $\boldsymbol{\theta}$ and $L(\cdot|\cdot)$ is the likelihood. As the likelihood is simply raised to a power, all asymptotic results from likelihood theory carry over when using power priors [Ibrahim et al., 2015].

The aim of this work is to estimate the current mean μ and variance σ^2 of univariate Gaussian data at time t. Historic data before time t is down-weighted by a factor λ, where this factor is to be estimated. For simplicity, the weights are rewritten in Equation 3.6 as $w_i = \lambda c_i$ where c_i is a known constant of the form $c_i = \prod_{j=i}^{t-2} \lambda_j$ for $i = 1, \ldots, t-2$ and $c_{t-1} = 1$. The factors λ_j for $j = 1, \ldots, t-2$ are the previously calculated exponential weights thus are known. For the frequentist case, Anagnostopoulos et al. [2012] perform Maximum Likelihood Estimation using the log likelihood in Equation 3.1. Bayesian Inference is instead performed using the following joint power prior for the parameters μ, σ^2 and λ at time t,

$$\begin{aligned} &P(\mu, \sigma^2, \lambda | x_1, \ldots, x_{t-1}) \\ &\propto P(\mu|\sigma^2) P(\sigma^2) P(\lambda) \prod_{i=1}^{t-1} P(x_i|\mu, \sigma^2)^{\lambda c_i} \end{aligned} \tag{3.7}$$

where

$$\begin{aligned} X_i|\mu,\sigma^2 &\sim N\left(\mu,\sigma^2\right) \text{ for } i=1,\ldots,t-1 \\ \mu|\sigma^2 &\sim N(\mu_0,\sigma_0^2\sigma^2) \\ \sigma^2 &\sim \text{Inv-Gamma}(\alpha_0,\beta_0) \\ \lambda &\sim P(\lambda). \end{aligned}$$

Here the prior for λ is left in general terms and leave specification of this prior is left for Section 3.2.3. Suppose the data at time t follows, $X_t \sim N(\mu,\sigma^2)$, the joint posterior takes the form,

$$\begin{aligned} &P(\mu,\lambda,\sigma^2|x_1,\ldots,x_t) \\ &\propto P(x_t|\mu,\sigma^2)P(\mu,\sigma^2,\lambda|x_1,\ldots,x_{t-1}) \\ &\propto \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}(x_t-\mu)^2\right)\cdot\left(\frac{1}{\sigma^2}\right)^{\alpha_0+1}\exp\left(-\frac{\beta_0}{\sigma^2}\right) \\ &\times\prod_{i=1}^{t-1}\left[\left(\frac{1}{2\pi\sigma^2}\right)^{\frac{\lambda c_i}{2}}\exp\left(-\frac{\lambda}{2\sigma^2}c_i(x_i-\mu)^2\right)\right] \\ &\times\left(\frac{1}{\sigma^2}\right)^{\frac{1}{2}}\exp\left(-\frac{1}{2\sigma_0^2\sigma^2}(\mu-\mu_0)^2\right)\cdot P(\lambda) \\ &\propto \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}\left(1+\sum_{i=1}^{t-1}\lambda c_i+\frac{1}{\sigma_0^2}\right)\left(\mu-\frac{\sum_{i=1}^{t-1}\lambda c_i x_i+x_t+\frac{\mu_0}{\sigma_0^2}}{\left(\sum_{i=1}^{t-1}\lambda c_i+1+\frac{1}{\sigma_0^2}\right)}\right)^2\right) \\ &\times\exp\left(-\frac{1}{2\sigma^2}\left(\frac{\mu_0^2}{\sigma_0^2}+\lambda\sum_{i=1}^{t-1}c_i x_i^2+x_t^2-\frac{\left(\sum_{i=1}^{t-1}\lambda c_i x_i+x_t+\frac{\mu_0}{\sigma_0^2}\right)^2}{\sum_{i=1}^{t-1}\lambda c_i+1+\frac{1}{\sigma_0^2}}\right)\right) \\ &\times\left(\frac{1}{\sigma^2}\right)^{\alpha_0+\lambda\sum_{i=1}^{t-1}\frac{c_i}{2}+\frac{1}{2}+1}\exp\left(-\frac{\beta_0}{\sigma^2}\right)\times\left(\frac{1}{2\pi}\right)^{\lambda\sum_{i=1}^{t-1}\frac{c_i}{2}}P(\lambda). \end{aligned}$$

Hence the conditional marginal posterior for μ is,

$$\begin{aligned} &\mu|\sigma^2,\lambda,x_1,\ldots,x_t\sim \\ &\quad \mathrm{N}\left(\frac{\sum_{i=1}^{t-1}\lambda c_i x_i+x_t+\frac{\mu_0}{\sigma_0^2}}{\sum_{i=1}^{t-1}\lambda c_i+1+\frac{1}{\sigma_0^2}},\frac{\sigma^2}{\sum_{i=1}^{t-1}\lambda c_i+1+\frac{1}{\sigma_0^2}}\right). \end{aligned} \tag{3.8}$$

When $\lambda_j = 1$ for all j, this is equivalent to standard Bayesian inference with no forgetting. After integrating with respect to μ, the following joint posterior expression for σ^2 and λ is obtained,

$$
\begin{aligned}
&P(\sigma^2, \lambda | x_1, \ldots, x_t) \\
&\propto \left(\frac{1}{\sigma^2}\right)^{\alpha_0 + \lambda \sum_{i=1}^{t-1} \frac{c_i}{2} + \frac{1}{2} + 1} \times \exp\left(-\frac{1}{\sigma^2}\left[\beta_0 + \frac{1}{2}\left(\frac{\mu_0^2}{\sigma_0^2} + \lambda \sum_{i=1}^{t-1} c_i x_i^2 + x_t^2 - \frac{\left(\sum_{i=1}^{t-1} \lambda c_i x_i + x_t + \frac{\mu_0}{\sigma_0^2}\right)^2}{\sum_{i=1}^{t-1} \lambda c_i + 1 + \frac{1}{\sigma_0^2}}\right)\right]\right) \\
&\times \left(\sum_{i=1}^{t-1} \lambda c_i + 1 + \frac{1}{\sigma_0^2}\right)^{-\frac{1}{2}} \times \left(\frac{1}{2\pi}\right)^{\lambda \sum_{i=1}^{t-1} \frac{c_i}{2}} P(\lambda)
\end{aligned}
$$

hence the conditional marginal posterior for the variance follows,

$$
\begin{aligned}
&\sigma^2 | \lambda, x_1, \ldots, x_t \\
&\sim \text{Inv-Gamma}\left(\lambda \sum_{i=1}^{t-1} \frac{c_i}{2} + \frac{1}{2} + \alpha_0, \beta_0 + \frac{1}{2}\left(\frac{\mu_0^2}{\sigma_0^2} + \lambda \sum_{i=1}^{t-1} c_i x_i^2 + x_t^2 - \frac{\left(\sum_{i=1}^{t-1} \lambda c_i x_i + x_t + \frac{\mu_0}{\sigma_0^2}\right)^2}{\sum_{i=1}^{t-1} \lambda c_i + 1 + \frac{1}{\sigma_0^2}}\right)\right).
\end{aligned}
$$

When $\lambda_j = 1$ for all j, this is equivalent to the unweighted case with no forgetting. The marginal posterior for the forgetting factor has the form,

$$
\begin{aligned}
P(\lambda | x_1, \ldots, x_t) \propto &\left(\sum_{i=1}^{t-1} \lambda c_i + 1 + \frac{1}{\sigma_0^2}\right)^{-\frac{1}{2}} \times \Gamma\left(\lambda \sum_{i=1}^{t-1} \frac{c_i}{2} + \frac{1}{2} + \alpha_0\right) \left(\frac{1}{2\pi}\right)^{\lambda \sum_{i=1}^{t-1} \frac{c_i}{2}} P(\lambda) \\
&\times \left(\beta_0 + \frac{1}{2}\left(\frac{\mu_0^2}{\sigma_0^2} + \lambda \sum_{i=1}^{t-1} c_i x_i^2 + x_t^2 - \frac{\left(\sum_{i=1}^{t-1} \lambda c_i x_i + x_t + \frac{\mu_0}{\sigma_0^2}\right)^2}{\sum_{i=1}^{t-1} \lambda c_i + 1 + \frac{1}{\sigma_0^2}}\right)\right)^{-\left(\lambda \sum_{i=1}^{t-1} \frac{c_i}{2} + \frac{1}{2} + \alpha_0\right)}.
\end{aligned}
$$

This does not follow a well known distribution nor can the normalising factor be found analytically. The forgetting factor estimate depends on the data and the parameters of the priors but is independent of the unknown μ and σ^2 parameters.

The Maximum a Posteriori (MAP) estimates at time t are used to estimate λ, μ and σ^2. The MAP estimates are used due to their analogue to the MLE in the frequentist formulation when uniform priors are used. First the MAP estimate for λ, denoted $\hat{\lambda}_{t-1}^{MAP}$, is found by maximising the unnormalised posterior numerically. MAP estimates for the mean and variance at time t take the following analytic form,

$$
\hat{\mu}_t^{MAP} = \frac{\hat{\lambda}_{t-1}^{MAP}\left(\sum_{i=1}^{t-2}(\prod_{j=i}^{t-2} \lambda_j) x_i + x_{t-1}\right) + x_t + \frac{\mu_0}{\sigma_0^2}}{\hat{\lambda}_{t-1}^{MAP}\left(\sum_{i=1}^{t-2}(\prod_{j=i}^{t-2} \lambda_j) + 1\right) + 1 + \frac{1}{\sigma_0^2}}, \tag{3.9}
$$

$$\hat{\sigma}_t^{2^{MAP}} = \frac{\beta_0 + \frac{1}{2}\left(\frac{\mu_0^2}{\sigma_0^2} + \hat{\lambda}_{t-1}^{MAP}\left(\sum_{i=1}^{t-2}(\prod_{j=i}^{t-2}\lambda_j)x_i^2 + x_{t-1}^2\right) + x_t^2\right)}{\frac{1}{2}\hat{\lambda}_{t-1}^{MAP}\left(\sum_{i=1}^{t-2}\prod_{j=i}^{t-2}\lambda_j + 1\right) + \frac{1}{2} + \alpha_0 + 1} - \frac{\frac{1}{2}\left(\frac{\left(\hat{\lambda}_{t-1}^{MAP}\left(\sum_{i=1}^{t-2}(\prod_{j=i}^{t-2}\lambda_j)x_i + x_{t-1}\right) + x_t + \frac{\mu_0}{\sigma_0^2}\right)^2}{\hat{\lambda}_{t-1}^{MAP}\left(\sum_{i=1}^{t-2}(\prod_{j=i}^{t-2}\lambda_j) + 1\right) + 1 + \frac{1}{\sigma_0^2}}\right)}{\frac{1}{2}\hat{\lambda}_{t-1}^{MAP}\left(\sum_{i=1}^{t-2}\prod_{j=i}^{t-2}\lambda_j + 1\right) + \frac{1}{2} + \alpha_0 + 1}. \tag{3.10}$$

3.2.2 Sequential Bayesian Updates

As new data becomes available, recalculation of the MAP estimates in Equations 3.9 and 3.10 is inefficient and requires the full stream of data to be stored. These equations can easily be simplified to allow sequential updating parameter estimates through defining additional parameters. At time t let,

$$\begin{aligned} N(t) &= \hat{\lambda}_{t-1}^{MAP}\left(\sum_{i=1}^{t-2}\left(\prod_{j=i}^{t-2}\lambda_j\right)x_i + x_{t-1}\right) + x_t \\ &= \hat{\lambda}_{t-1}^{MAP}N(t-1) + x_t, \end{aligned}$$

$$\begin{aligned} D(t) &= \hat{\lambda}_{t-1}^{MAP}\left(\sum_{i=1}^{t-2}\prod_{j=i}^{t-2}\lambda_j + 1\right) + 1 \\ &= \hat{\lambda}_{t-1}^{MAP}D(t-1) + 1 \end{aligned}$$

and

$$\begin{aligned} M(t) &= \hat{\lambda}_{t-1}^{MAP}\left(\sum_{i=1}^{t-2}\left(\prod_{j=i}^{t-2}\lambda_j\right)x_i^2 + x_{t-1}^2\right) + x_t^2 \\ &= \hat{\lambda}_{t-1}^{MAP}M(t-1) + x_t^2. \end{aligned}$$

Then the MAP estimates have the following sequential updates,

$$\hat{\mu}_t^{MAP} = \frac{N(t) + \frac{\mu_0}{\sigma_0^2}}{D(t) + \frac{1}{\sigma_0^2}}$$

$$\hat{\sigma}_t^{2^{MAP}} = \frac{\beta_0 + \frac{1}{2}\left(\frac{\mu_0^2}{\sigma_0^2} + M(t) - \frac{\left(N(t)+\frac{\mu_0}{\sigma_0^2}\right)^2}{D(t)+\frac{1}{\sigma_0^2}}\right)}{\frac{1}{2}D(t) + \alpha_0 + 1}.$$

The parameters $N(t-1)$, $D(t-1)$ and $M(t-1)$ are also used to produce a sequential updating form for the unnormalised marginal posterior density function for λ,

$$f_t(\lambda) \propto \left(\lambda D(t-1) + 1 + \frac{1}{\sigma_0^2}\right)^{-\frac{1}{2}} \times \Gamma\left(\frac{\lambda}{2}D(t-1) + \frac{1}{2} + \alpha_0\right) \times \left(\frac{1}{2\pi}\right)^{\frac{\lambda}{2}D(t-1)} P(\lambda)$$
$$\times \left(\beta_0 + \frac{1}{2}\left(\frac{\mu_0^2}{\sigma_0^2} + \lambda M(t-1) + x_t^2 - \frac{\left(\lambda N(t-1) + x_t + \frac{\mu_0}{\sigma_0^2}\right)^2}{\lambda D(t-1) + 1 + \frac{1}{\sigma_0^2}}\right)\right)^{-\left(\frac{\lambda}{2}D(t-1)+\frac{1}{2}+\alpha_0\right)}.$$

To update the MAP estimates at time t only $N(t-1)$, $D(t-1)$ and $M(t-1)$ are required from the historic data.

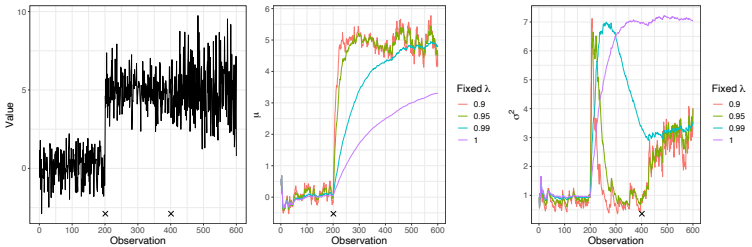
3.2.3 λ Prior Specification

Anagnostopoulos et al. [2012] state that sensible values for λ lie within $[0.5, 1]$ due to its interpretation as an exponential forgetting factor . While the stream is stationary, values of λ close to 1 are appropriate and correspond to classic Bayesian inference without forgetting. When a change point occurs, a small value for λ allows rapid forgetting of historic data points. During trend periods moderate forgetting allows more gentle forgetting of historic data as it becomes less relevant. It is also important to impose a lower bound on λ, as values below 0.5 may lead to numeric instability Anagnostopoulos et al. [2012].

As observed in Figure 3.1, for smaller forgetting values the mean estimates adapt more rapidly to the new value. For λ with values 1 and 0.99, the procedures do not reach the new mean estimates within the investigated period. It can however be seen that for smaller forgetting factors, although they adapt faster, their estimates are more noisy. Finally, it is noted that for the estimate of σ^2 for $\lambda = 1$, a large value is maintained after the variance change point unlike the other forgetting factors. This example illustrates the importance of an adaptive forgetting factor which can automatically choose appropriate levels of forgetting.

A large range of distributions are appropriate for the prior of the forgetting factor λ such as a Uniform prior over $[0.5, 1]$ or a slab and spike prior (a mixture density of two distributions where one distribution has a high mass at 1). Alternatively, a Beta distribution with a concentration around 1 is used and experimentally was most suitable.

From investigating the estimation performance for Beta priors of different



(a) **(b)** **(c)**

Figure 3.1: (a) Noisy Gaussian data stream of length 600 such that $X_1, \ldots, X_{200} \sim N(0,1)$,$X_{201}, \ldots, X_{400} \sim N(5,1)$ and $X_{401}, \ldots, X_{600} \sim N(5,3)$. (b) Corresponding mean estimate from the Bayesian adaptive estimation procedure for varying fixed forgetting factors. (c) Corresponding variance estimate from the Bayesian adaptive estimation procedure for varying fixed forgetting factors.

Table 3.1: Parameter estimation performance of the fixed Beta prior for the forgetting factor λ for various modes and variances for Gaussian data containing change points and trends as described in Section 3.3.5 running on 100 simulated data sets of length 10000. The performance is measured by the mean square error (MSE) where 20 points after a change is excluded with a burn-in of 200 points. Best prior choice presented in bold.

Mode	Var	MSE μ	MSE σ^2
0.999	1e-2	5.056(2.689)	373.724(298.603)
0.999	1e-3	1.691(0.891)	125.046(100.921)
0.999	1e-4	4.270(1.743)	824.791(667.660)
0.980	1e-2	4.981(2.644)	369.032(294.587)
0.980	1e-3	**1.622**(0.836)	**122.312** (97.709)
0.980	1e-4	4.751(2.273)	972.854(828.663)
0.950	1e-2	4.902(2.592)	364.005(290.090)
0.950	1e-3	1.687(0.860)	127.801(101.525)
0.950	1e-4	1.941(0.955)	320.215(270.976)
0.900	1e-2	4.897(2.567)	363.492(288.705)
0.900	1e-3	2.109(1.071)	156.537(123.425)
0.900	1e-4	1.733(0.858)	134.567(106.246)

modes and variances (Table 3.1), a mode of 0.98 and variance of 0.001 is optimal. This mode and variance is selected for future applications which correspond to a Beta$(39, 1.8)$ distribution. As data streams are primarily stationary, this prior suitably favours values of λ close to 1 however it still enables the parameter esti-

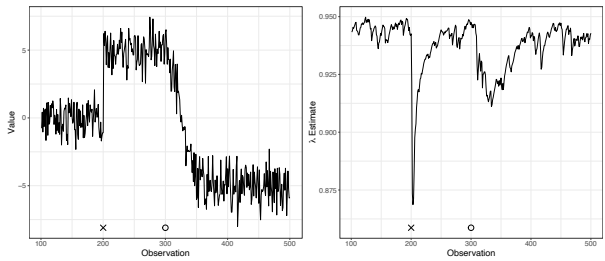
mates for λ to reduce during trends or change points. This λ prior is fixed over the stream to prevent dependence between these estimated values. This independence allows for rapid adaptation at change points and trends.

3.2.4 Behaviour of λ for Fixed Prior

The adaptive ability of the procedure is now investigated where the prior for λ is fixed as Beta(39.1.8). A small simulation case with a single change point and trend is used where the stream is sampled from:

$$\begin{aligned} X_1, \ldots, X_{200} &\sim N(0,1) \\ X_{201}, \ldots, X_{300} &\sim N(5,1) \\ X_i &\sim N(5 - 0.2(i-300), 1) \text{ for } i \in [301, 350] \\ X_{351}, \ldots, X_{500} &\sim N(-5,1). \end{aligned} \tag{3.11}$$

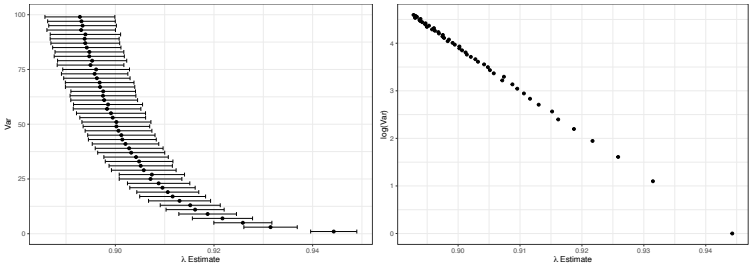
A realisation of this generation process is given in Figure 3.2 (a).



(a) **(b)**

Figure 3.2: (a) Single stream generated as described in Equation 3.11. (b) Illustration of corresponding estimated λ by BFF for the simulation in (a) with a change point at 201 illustrated with a cross and a trend period between points 301-350 with start point marked with a circle. The simulated data has a variance of 1.

For the BFF λ estimate in Figure 3.2(b), at the change point there is a significant drop in the forgetting factor before returning to the original level. For the trend region, the forgetting factor lowers. After this trend period, the forgetting factor again returns to the original value. Thus the proposed procedure appropriately adjusts the forgetting factor as expected at change points and during trends.



(a) **(b)**

Figure 3.3: (a) Mean λ estimate for stationary Gaussian data streams of varying variance. One standard deviation of the mean estimate is shown with error bars. (b) Mean λ estimate for stationary Gaussian data streams against the log variance of the data.

3.2.5 Relationship between λ and σ^2

The relationship between the forgetting factor and the variance is now explored for Gaussian data. In Figure 3.3(a) it can be see that as the variance increases, the forgetting factor decreases. The optimal forgetting factor for stationary streams is however a constant value of 1. In Figure 3.3(b) a linear relationship between $log(\sigma^2)$ and the average estimated λ is observed. This is surprising as this relationship has not been noted in the literature, and provides insight into the workings of the forgetting factor. Intuitively this relationship is sensible where if there is more variance in the data, it may suggest that the data is not stable and should be forgotten more rapidly. For very small variances, the forgetting value is close to 1.

3.2.6 Adaptive Parameter Priors for μ and σ^2

At each time point, prior parameters for μ and σ^2, namely μ_0, σ_0^2, α_0 and β_0 must be specified, however over time knowledge of these parameters change. It was found empirically that fixed priors over the stream can cause poor estimates particularly if they are not well specified. Additionally for time-varying data, a fixed prior is not appropriate. In Bayesian filtering it is common for the previous posterior to be used as the prior for future inference [Sarkka, 2013]. During long periods of no distribution changes in the data, the variance of the posterior distribution is small. As the variance of the prior influences its contribution to the posterior, where smaller variances lead to larger influence, this traditional updating mechanism causes the influence of the prior to be large during periods of non-stationarity leading to slow adaptation during periods of change in the data generating process.

We instead propose keeping the influence or contribution of the prior fixed. We

motivate our approach further with an example. Figure 3.4 shows the posterior variance estimate for μ after a mean change occurs in the generating process. This posterior variance is calculated using Equation 3.8. After a change, this posterior variance increases due to the increase in variance in the data and increased uncertainty in the current estimate for μ. This is followed by a decrease in the variance until it reaches a constant small value as the process becomes stationary. If a stream is stationary before a change point, the posterior will have a small variance. If this posterior is used as the prior it will have a large bias on the parameter estimates, making them slow to adapt.

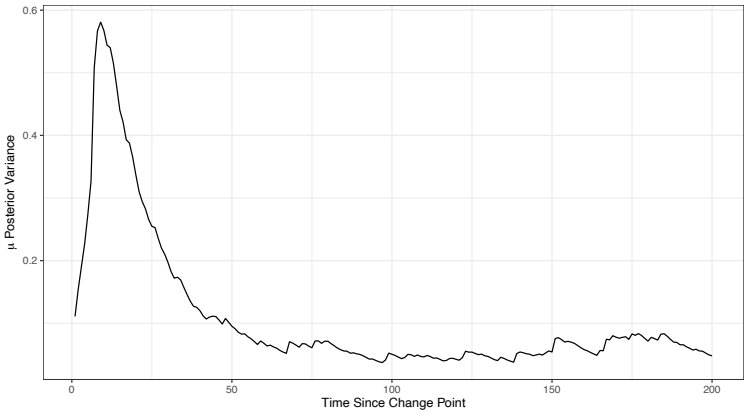


Figure 3.4: Illustration of posterior variance estimate for μ given in Equation 3.8 after a change point for a single Gaussian data stream with variance 1.

For the proposed fixed influence approach the previous parameter estimates are used to update the priors rather than keeping them fixed as done for λ. As the distribution parameters, μ and σ^2 adapt depending on the forgetting factor value, dependence on previous values is preferable to allow for smooth estimates. Constant influence of the prior ensures the estimates can adapt to the data in the presence of change points rather than assuming the data is stationary. We propose an approach that fixes the weight of the prior towards the parameter estimates where the priors take the following form,

$$\begin{aligned} \mu|\sigma^2 &\sim N(\mu_0, \sigma_0^2\sigma^2) \\ \sigma^2 &\sim \text{Inv-Gamma}(\alpha_0, \beta_0). \end{aligned}$$

At time t the mode of the priors for μ and σ^2 is updated using the MAP solutions from time $t-1$ given in Equation 3.9 and Equation 3.10 respectively. In Equation 3.9 it is clear that $\frac{1}{\sigma_0^2}$ acts as a weight for the prior mean, μ_0, towards

the estimate for μ. σ_0^2 is set to 1 resulting in the prior mean having equal weight to the most recent data point x_t. By setting $\alpha_0 = \frac{1}{2}$ for the MAP estimate of σ^2 in Equation 3.10 the prior has equal weight to the most recent data x_t. For this approach $\beta_0 = \frac{2}{3}\hat{\sigma}_{t-1}^{2^{MAP}}$, so that the prior mode is equal to the previous MAP estimate. The prior updates under this scheme are,

$$\begin{aligned} \mu|\sigma^2 &\sim N(\hat{\mu}_{t-1}^{MAP}, \sigma^2) \\ \sigma^2 &\sim \text{Inv-Gamma}\left(\frac{1}{2}, \frac{2}{3}\hat{\sigma}_{t-1}^{2^{MAP}}\right). \end{aligned} \tag{3.12}$$

This prior specification allows the priors to update autonomously with constant influence towards the estimates over the stream. Table 3.2 shows the proposed approach has superior performance to the traditional approach for data simulated as described in Section 3.3.5.

Table 3.2: Comparison in parameter estimation performance measured by the mean square error (MSE) for the BFF method with the proposed prior updates and with traditional posterior prior updates. The MSE for the mean and variance is given where results are averaged over 100 simulated Gaussian data streams of 10000 points containing change points and trend as described in Section 3.3.5. The MSE excludes the 20 points after a change point with the best performer in bold.

	MSE μ	MSE σ^2
Proposed	**0.134(1.168)**	**0.663(0.830)**
Posterior	43.435(56.808)	1371.623(865.958)

3.2.7 General Form for Exponential Family

The algorithm for the Gaussian case has been described, however, this framework can be extended more generally to the exponential family of distributions with the use of conjugate priors to produce closed form update equations.

The likelihood for members of the exponential family has the form,

$$P(x|\boldsymbol{\eta}) = h(x)\exp\left(\boldsymbol{\eta}^T\mathbf{T}(x) - A(\boldsymbol{\eta})\right) \tag{3.13}$$

where $\boldsymbol{\eta}$ is the canonical parameter with,

$$\begin{aligned} \boldsymbol{\eta} &= (\eta_1(\boldsymbol{\theta}), \ldots, \eta_p(\boldsymbol{\theta}))^T \\ \boldsymbol{\theta} &= (\theta_1, \ldots, \theta_p)^T, \end{aligned}$$

$\mathbf{T}$ is the sufficient statistic and A is the log of the normalising factor also known as the cumulant generating function. The conjugate prior of this likelihood has

the form,

$$P_\pi(\boldsymbol{\eta}|\boldsymbol{\chi}, \upsilon) = f(\boldsymbol{\chi}, \upsilon) \exp\left(\boldsymbol{\eta}^T \boldsymbol{\chi} - \upsilon A(\boldsymbol{\eta})\right) \quad \boldsymbol{\chi} \in \mathbb{R}^s$$

where $\boldsymbol{\chi}$ and υ are hyper-parameters and $f(\boldsymbol{\chi}, \upsilon)$ is the normalising constant [Wasserman, 2004].

Again the power prior is used to down-weight older observations. Using the same weighting mechanism as before, let x_i have weight given by,

$$w_i = \lambda \prod_{j=i}^{t-2} \lambda_j = \lambda c_i \quad \text{for } i = 1, \ldots, t-2$$

at time t where λ_j are known for $j = 1, \ldots, t-2$. For $i = t-1$, $w_{t-1} = \lambda = \lambda c_{t-1}$ with $c_{t-1} = 1$. Using the following joint power prior,

$$P(\boldsymbol{\eta}, \lambda|\boldsymbol{\chi}, \upsilon) \propto P_\pi(\boldsymbol{\eta}|\boldsymbol{\chi}, \upsilon) P(\lambda) \prod_{i=1}^{t-1} P(x_i|\boldsymbol{\eta})^{w_i},$$

the joint posterior is calculated as,

$$\begin{aligned}
&P(\boldsymbol{\eta}, \lambda|x_1, \ldots, x_n, \boldsymbol{\chi}, \upsilon) \\
&\propto h(x_t) \exp\left(\boldsymbol{\eta}^T \mathbf{T}(x_t) - A(\boldsymbol{\eta})\right) \\
&\times \prod_{i=1}^{t-1} \left(h(x_i) \exp\left(\boldsymbol{\eta}^T \mathbf{T}(x_i) - A(\boldsymbol{\eta})\right)\right)^{w_i} \\
&\times f(\boldsymbol{\chi}, \upsilon) \exp\left(\boldsymbol{\eta}^T \boldsymbol{\chi} - \upsilon A(\boldsymbol{\eta})\right) \times P(\lambda) \\
&\propto \exp\left(\boldsymbol{\eta}^T \left(\mathbf{T}(x_t) + \sum_{i=1}^{t-1} w_i \mathbf{T}(x_i) + \boldsymbol{\chi}\right)\right) \\
&\times \exp\left(-A(\boldsymbol{\eta}) \left[1 + \sum_{i=1}^{t-1} w_i + \upsilon\right]\right) \\
&\times P(\lambda) \left[h(x_t) \prod_{i=1}^{t-1} h(x_i)^{w_i}\right] \\
&\propto P_\pi\left(\boldsymbol{\eta} \middle| \boldsymbol{\chi} + \mathbf{T}(x_t) + \sum_{i=1}^{t-1} w_i \mathbf{T}(x_i), \upsilon + 1 + \sum_{i=1}^{t-1} w_i\right) \\
&\times \frac{P(\lambda) \left[\prod_{i=1}^{t} h(x_i)^{w_i}\right]}{f\left(\boldsymbol{\chi} + \mathbf{T}(x_t) + \sum_{i=1}^{t-1} w_i \mathbf{T}(x_i), \upsilon + 1 + \sum_{i=1}^{t-1} w_i\right)}.
\end{aligned}$$

The MAP solution for λ can be estimated by maximising the unnormalised marginal

posterior for λ,

$$\frac{P(\lambda)\left[\prod_{i=1}^{t} h(x_i)^{\lambda c_i}\right]}{f\left(\boldsymbol{\chi}+\sum_{i=1}^{t} w_i \mathbf{T}(x_i), \upsilon+\sum_{i=1}^{t} w_i\right)} \tag{3.14}$$

which is independent of the canonical parameter $\boldsymbol{\eta}$. Once λ is estimated, $\boldsymbol{\eta}$ can be calculated from its marginal posterior distribution,

$$P_\pi\left(\boldsymbol{\eta}\,\middle|\,\boldsymbol{\chi}+\mathbf{T}(x_t)+\lambda\sum_{i=1}^{t-1} c_i\mathbf{T}(x_i), \upsilon+1+\lambda\sum_{i=1}^{t-1} c_i\right). \tag{3.15}$$

The sums and products within this marginal posterior can easily be updated sequentially, making it practical for application to data streams, reducing storage requirements. Again at each time step, the prior can be updated to include more information about the current behaviour of the stream. The same approach where the mode of the prior distribution is set to the current MAP estimate whilst maintaining a fixed weight towards the estimates is used.

3.3 Synthetic Examples

3.3.1 Adaptive Estimation Performance Measures

To evaluate the parameter estimation performance of each method, several performance measures are used including MSE, MAE and MPE as described in Section 2.4.1. As no perfect measure exists, each of these is implemented to give a comprehensive view of performance.

After a change in the generating process of the data, an estimation method may require a transition period to adapt to the new distribution. The performance within this transition period is not of great interest. Therefore, after a change in the generating process of the data, a grace period of 100 points is given to allow the procedures to adapt and return to their stationary state.

Although estimation performance is important, fast adaptation is crucial. To evaluate this, the adaptation time is measured. This quantity is calculated as the number of steps after a change before the squared error is less than 1.2 times the MSE of the stationary period after the change point (but before another change occurs). The compared MSE excludes the grace period as defined above. This is further solidified mathematically below.

Consider changes in the generating process at times $\tau_1, \tau_2, \ldots$ and a grace period of length G. The time to change after the lth change is calculated as,

$$\min_i \; s.t. \; SE_{\tau_l+i} < 1.2 \times \frac{1}{\tau_{l+1}-\tau_l-G}\sum_{j=\tau_l+G}^{\tau_{l+1}-1} SE_j$$

where SE_j is the squared error in the parameter estimates at time j. This change

time signifies that the error in the estimates reduces to the stationary model state. It is important to note that this measure assumes the series returns to a stationary state after a change which may not be realistic in practice however the simulated series explored follow this assumption.

3.3.2 Comparison Parameter Estimation Methods

To assess the performance of the proposed BFF procedure, it is compared to other parameter estimation methods. Naturally, the procedure is compared to the frequentist version as detailed in Section 3.1 from which the proposed model builds. Throughout the following experiments, we name this the *Adaptive Forgetting Factor* (AFF) approach. Additionally, the fixed forgetting case is compared where the forgetting factor is kept constant whilst using the updates for the parameters in Section 3.1.1. This is termed the Fixed Forgetting Factor (FFF). Additionally, the proposed method is compared to the classic Maximum Likelihood Estimates (MLE) for the distribution which correspond to when the forgetting factor equals 1.

Another popular filter is the Exponentially Weighted Moving Average (EWMA) process where similar to the proposed approach, the weights for older data decreases exponentially. At time $t \geq 1$ the estimated mean of the data $x_1, \ldots, x_t$ using this approach is,

$$S_t = \begin{cases} x_1 & t = 1 \\ \alpha x_t + (1-\alpha) S_{t-1} & t > 1 \end{cases}$$

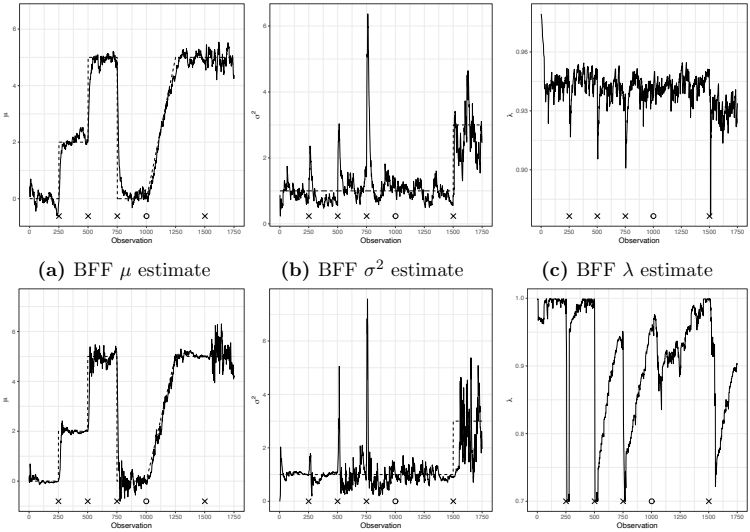
where α is the exponential weight that is specified by the analyst. Similarly, exponential weighted estimates for the variance can be calculated as,

$$V_t = \begin{cases} 0 & t = 1 \\ \alpha(x_t - S_t)(x_t - S_{t-1}) + (1-\alpha) V_{t-1} & t > 1. \end{cases}$$

Finally, the proposed method is compared to the adaptive estimation procedure of Adams and MacKay [2007] which performs Bayesian filtering whilst taking change points into account which is described further in Appendix B.1.1. Central to this procedure is the estimated run length which corresponds to the time since the last change point. This procedure additionally estimates the distribution parameters where similar to our approach, it is suitable for members of the exponential family. This procedure is named *Bayesian Online Change Point Detection* (BOCPD) and is described in more detail in Appendix B.1.1.

3.3.3 Gaussian Parameter Estimation Illustrative Example

On the simulated example in Figure 3.5 BFF and AFF have similar estimates for μ and σ^2 however the behaviour of λ differs greatly. For the mean estimates in Figures 3.5 (a) and (d), BFF and AFF are capable of adapting quickly to the mean changes and mean trend periods. AFF however experiences periods of large noise in the estimates for μ as a result of small forgetting factors as the forgetting factor is slow to return to values close to 1.



(a) BFF μ estimate **(b)** BFF σ^2 estimate **(c)** BFF λ estimate

(d) AFF μ estimate **(e)** AFF σ^2 estimate **(f)** AFF λ estimate

Figure 3.5: Illustration of Gaussian BFF and AFF adaptive estimation procedures for a single Gaussian simulation containing mean change points at 251, 501 and 751, a variance change point at 1501 and a linear trend region between 1001 and 1250. Change points are marked by crosses and the start of a trend marked by a circle. True values plotted with a dashed line for μ and σ^2.

In Figures 3.5 (b) and (e), BFF and AFF are capable of adapting to variance changes which is seen at point 1501. This variance change point increases the variance of the estimates for μ and σ^2 for both methods as the data is now noisier. For both methods, in Figures 3.5 (c) and (f), at change points the forgetting factor drops rapidly to a small value before returning to the original forgetting level as required. This is expected as after a change the model should forget the historic data whilst keeping the data from the new distribution.

3.3.4 Gaussian Performance Comparison - No Change

First the estimation performance is evaluated to comparator methods on simulated data without change points. This acts as a baseline performance for each method where it is preferable if this performance does not vary greatly between stationary and non-stationary data. For the results in Tables 3.3 and 3.4, the stream is sampled from,

$$\begin{aligned} X_1, \ldots, X_{10000} &\sim N(\mu, \sigma^2) \\ \mu &\sim U(-20, 20) \\ \sigma^2 &\sim U(0.5, 5). \end{aligned}$$

As expected, the MLE approach has optimal performance for this stationary data as it assumes the data is i.i.d. whereas the other approaches do not make this assumption (Tables 3.3 and 3.4). For methods that require setting parameters, the results vary greatly between the parameter choices chosen. This illustrates that the hyper-parameters of AFF, FFF and EWMA have a large influence on estimation performance. In practice, setting such parameters is a challenging problem, particularly in streaming cases where the behaviour of the data changes over time.

When the estimation methods are applied to data streams containing a single change point, the MLE approach has very poor performance. Additionally, for methods with hyper-parameters, the performance differs greatly depending on the parameter choice (see Appendix A.2.1). When the methods are applied to data streams containing trend, the methods that performed well on stationary data are now the worst performers (see Appendix A.2.2). The proposed method instead has consistent performance across the examples.

Table 3.3: Average parameter estimation performance for μ over 100 stationary data streams of length $10,000$. Streams generated from $N(\mu, \sigma^2)$ where $\mu \sim U(-20, 20)$ and $\sigma^2 \sim U(0.5, 5)$. Best performer in bold.

	MSE	SE SD	MAE	AE SD	MAPE	APE SD
BFF	0.093	0.132	0.231	0.174	0.048	0.036
AFF 1e-7	**0.001**	0.002	0.029	0.021	0.006	0.004
AFF 1e-4	0.006	0.011	0.057	0.048	0.011	0.009
FFF 0.9	0.139	0.196	0.286	0.216	0.058	0.044
FFF 0.98	0.027	0.038	0.125	0.094	0.026	0.020
EWMA 0.05	0.068	0.096	0.199	0.151	0.041	0.031
EWMA 0.75	1.585	2.240	0.964	0.729	0.198	0.149
MLE	**0.001**	**0.001**	**0.019**	**0.011**	**0.004**	**0.003**
BOCPD	**0.001**	0.022	**0.019**	0.017	**0.004**	0.005

Table 3.4: Average parameter estimation performance for σ^2 over 100 stationary data streams of length $10,000$. Streams generated from $N(\mu, \sigma^2)$ where $\mu \sim U(-20, 20)$ and $\sigma^2 \sim U(0.5, 5)$. Best performer in bold.

	MSE	SE SD	MAE	AE SD	MAPE	APE SD
BFF	0.648	0.757	0.590	0.383	0.216	0.142
AFF 1e-7	0.011	0.013	0.072	0.052	0.027	0.020
AFF 1e-4	0.044	0.081	0.141	0.117	0.053	0.044
FFF 0.9	0.876	1.435	0.664	0.497	0.252	0.189
FFF 0.98	0.177	0.255	0.297	0.224	0.112	0.085
EWMA 0.05	0.440	0.675	0.469	0.354	0.178	0.134
EWMA 0.75	5.010	4.413	1.834	0.757	0.696	0.287
MLE	**0.004**	**0.005**	**0.045**	**0.027**	**0.018**	**0.011**

3.3.5 Gaussian Performance Comparison - Multiple Change with Trend

The estimation methods are now compared on data streams containing multiple change points and trend periods. Ten mean changes and two trend periods are simulated uniformly over $(200, 10000)$ such that change points and trend starting points are at least 200 points apart. These simulations have the following properties:

- Constant variance across the stream where the variance is sampled as follows, $\sigma^2 \sim U(0.5, 5)$
- Change point magnitudes sampled from $U(2\sigma, 5\sigma)$
- Periods of trend have a minimum length of 50 points with gradient magnitude uniformly sampled from $\{0.005, 0.006, \ldots, 0.02\}$. Proceeding trend regions there is a constant mean of length > 30.

The estimation performance for BFF in Tables 3.5 and 3.6 is similar to that seen for the stationary data in Section 3.3.4. This highlights the robust performance of the proposed procedure on data that exhibits trends and change points, making it suitable for application to real data sets that exhibit such behaviour. The best performer on this data is the BOCPD however this method has a higher SD suggesting it does not have stable results. The MLE which was the best performer on stationary data now has the worst performance. For the estimates of the variance in Table 3.6, BFF outperforms all sequential methods but EWMA 0.05 in terms of MSE. Thus BFF has good performance for estimating both μ and σ^2.

In Table 3.5 the BFF model displays a clear improvement on AFF which our model extends. Although a fixed prior for the forgetting factor λ is required for BFF, the prior specification detailed in Section 3.2.3 has robust performance,

thus BFF is essentially hyper-parameter free. It is clear from Table 3.5 that the parameters for AFF, FFF and EWMA have a large influence on the performance. When applied to data streams, tuning these parameters is a difficult problem hence BFF offers a robust solution without requiring parameter specification.

Table 3.5: Average parameter estimation performance of μ over 100 data streams of length 10,000 with 10 mean change points and two mean trends positioned randomly throughout. Each of the change points and trends are at least 200 points apart where trends have a stationary period proceeding it. The change point magnitudes are sampled from $U(2\sigma, 5\sigma)$ where $\sigma^2 \sim U(0.5, 5)$. Best performer in bold, second best in grey bold.

	MSE	SE SD	MAE	AE SD	MAPE	APE SD	Time to Change	Time to Change SD
BFF	0.130	1.121	0.243	0.248	0.072	**0.552**	34.792	10.059
AFF 1e-7	0.480	2.486	0.301	0.599	0.091	0.902	175.573	56.706
AFF 1e-4	0.157	1.134	0.257	0.285	0.080	0.784	30.613	12.933
FFF 0.9	0.173	**1.052**	0.296	0.272	0.088	0.722	24.137	7.270
FFF 0.98	0.102	1.231	0.182	0.253	0.057	0.724	145.318	28.519
EWMA 0.05	0.107	1.153	0.213	**0.235**	0.064	0.540	55.246	13.720
EWMA 0.75	1.610	2.285	0.978	0.740	0.291	1.884	**1.292**	**2.233**
MLE	66.055	73.259	5.984	4.222	1.410	6.924	424.306	284.800
BOCPD	**0.084**	1.284	**0.110**	0.258	**0.035**	0.724	29.508	40.795

Table 3.6: Average parameter estimation performance of σ^2 over 100 data streams of length 10,000 with 10 mean change points and mean trends positioned randomly throughout. Each of the change points and trends are at least 200 points apart where trends have a stationary period proceeding it. The change point magnitudes are sampled from $U(2\sigma, 5\sigma)$ where $\sigma^2 \sim U(0.5, 5)$. Best performer in bold, second best in grey bold.

	MSE	SE SD	MAE	AE SD	MAPE	APE SD
BFF	0.655	0.826	0.602	0.394	0.218	0.144
AFF 1e-7	5.558	23.241	0.899	1.835	0.342	0.701
AFF 1e-4	0.746	1.906	0.575	0.524	0.216	0.196
FFF 0.9	0.899	1.750	0.679	0.518	0.253	0.193
FFF 0.98	0.853	3.586	0.479	0.682	0.183	0.258
EWMA 0.05	**0.446**	**0.734**	**0.478**	**0.365**	**0.178**	**0.136**
EWMA 0.75	5.090	6.197	1.870	0.797	0.697	0.297
MLE	2811.510	2876.389	31.960	21.039	12.085	7.656

The time to change in Table 3.5 is an important measure for determining the adaptability of the methods. For BFF it takes approximately 35 time steps to adapt to the new distribution which is smaller than many other procedures. It is important to note that small MSE does not necessarily correspond to fast adaptability. As the MSE calculated in these examples excludes 100 points after a

change point, it does not account for the poor estimation results during this grace period. For example, although FFF 0.98 has a small MSE, the time to change is found to be higher.

3.3.6 Poisson Parameter Estimation

BFF is now demonstrated on Poisson data to show it is suitable for other data distributions. Details of the Poisson formulation and sequential updates are given in Appendix A.1. The Poisson rate estimates of BFF and AFF in Figures 3.6 (a) and (c) respectively closely follow the true rate however the estimates from BFF are less noisy. Additionally BFF adapts to the new distribution faster than AFF and better follows the true rate during the trend period.

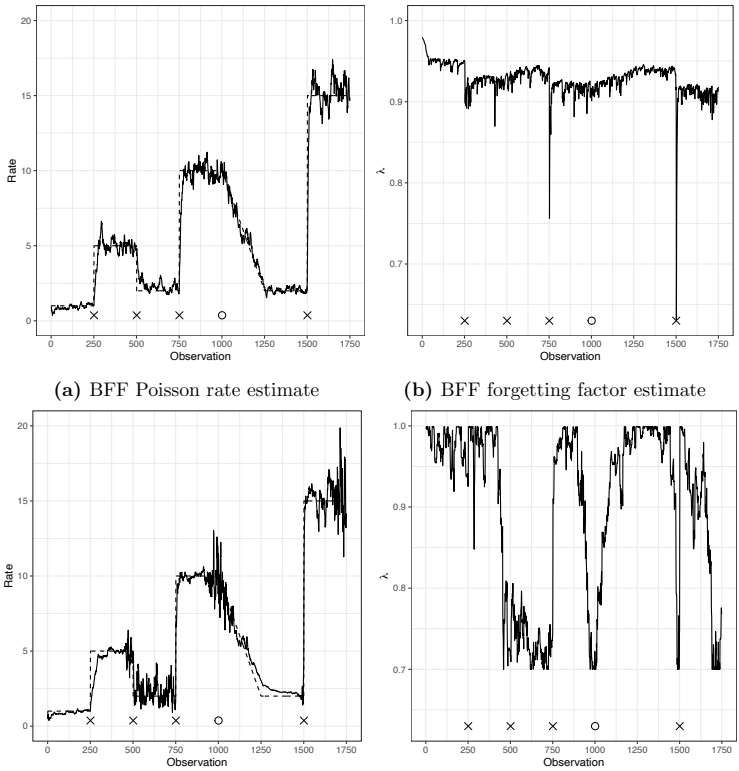
For detailed experiments similar to that done for the Gaussian case, see Appendix A.3 where comparison to the competitive approaches is performed on stationary, single change point, trend and multiple change point with trend simulations. On the Poisson data, BFF shows consistent performance across the simulated cases where AFF had a poorer performance for the non-stationary examples. BOCPD was again the best performer across the simulated examples in terms of parameter estimation however the time to adapt for this method is larger than the proposed BFF approach.

Comparing the forgetting factor estimates of the two methods in Figures 3.6 (b) and (d), the BFF forgetting factor is more similar to the behaviour expected where at change points, the forgetting factor lowers temporarily before returning to the original level. AFF instead goes between large extremes and does not return to high values of λ during stationary periods. This results in noisy estimates. A learning rate of 0.01 is used for AFF in this illustration to allow for fast adaptation of the rate estimate. Smaller learning rates kept the forgetting factor more stable however the adaptation was slower. This highlights the challenge involved in setting the learning rate for AFF which is avoided in BFF.

3.4 Discussion

In this chapter, a Bayesian adaptive estimation procedure is introduced that is capable of running autonomously on data streams with fixed storage requirements. By optimising the forgetting factor within the Bayesian inference framework, this procedure does not require setting highly influential hyper-parameters whilst automatically choosing appropriate levels of forgetting in the model. Additionally, mechanisms to update the priors are provided such that the priors are relevant to the current behaviour of the data whilst having constant influence over the stream to prevent slow adaptation during non-stationarity in the data generating process.

BFF is demonstrated on both Gaussian and Poisson data on several simulated cases with and without change points and trend. Across all simulated situations, BFF has consistent performance which highlights that it is robust for use in prac-



(a) BFF Poisson rate estimate **(b)** BFF forgetting factor estimate

(c) AFF Poisson rate estimate **(d)** AFF forgetting factor estimate

Figure 3.6: Illustration of Poisson BFF and AFF (learning rate 1e-2) adaptive estimation procedures for a single Poisson simulation containing changes in the rate at time points 251, 501, 751, and 1501 and a trend between 1001 and 1250. Change points are marked by crosses and the start of a trend is marked by a circle. True values plotted with dashed line for the Poisson rate.

tice. In comparison to competitive methods, the time for BFF to adapt after a change is very fast whilst maintaining high estimation performance.

In the next chapter, this estimation procedure is utilised to detect abrupt changes in the generating process of the stream.

Chapter 4

Detecting Abrupt Changes in the Presence of Trend for Data Streams

In the context of cyber-security, detection of adversaries is a key challenge. Attackers with a foothold on the network may cause abrupt changes to the underlying data generating process through their anomalous activities. Although adaptive estimation is capable of adjusting to distributional changes, the problem of *change point detection* aims to identify the time these changes occur and is the primary focus of this chapter. As the aim is to detect adversaries in real time, online change point detection methods for high frequency data streams are required.

In practice, data often exhibit periods of trend or local fluctuations. These behaviours could be treated as perpetual change points as the distribution of the data is continuously changing. However, only abrupt changes in the distribution are of interest rather than detecting trend. Many popular change point procedures assume the data is piecewise constant with a constant mean between change points, flagging large numbers of false positives during trend regions. These methods are generally not formulated with trend in mind. To visualise this, consider the popular well log data [Ó Ruanaidh and Fitzgerald, 1996] in Figure 4.1 investigated by many change point procedures e.g. [Fearnhead, 2006], [Adams and MacKay, 2007] and [Romano et al., 2021]. This data measures the nuclear magnetic response of underground rocks from a probe lowered into a bore-hole. The abrupt changes are marked by vertical lines corresponding to when the probe moves from one rock strata to another and these are the changes of interest. Between these changes, local fluctuations are present, thus the assumption of constant mean between changes is not held. It is thus advantageous for a change point model to be robust in the presence of trend and local fluctuations as it is a common feature of real data, to prevent large numbers of false positives.

The adaptive estimation procedure presented in Chapter 3 forms the basis of the two change point methods proposed in this chapter. Both of the proposed

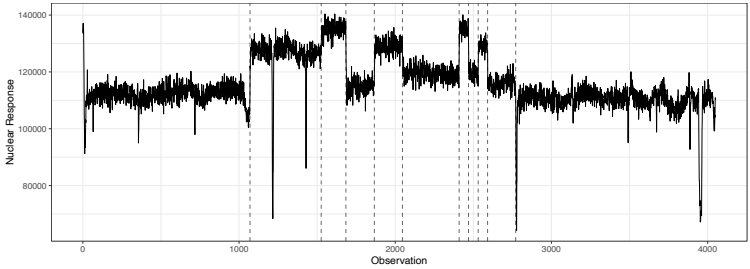


Figure 4.1: Well logs data demonstrated with black line and abrupt changes in the data identified by vertical lines.

change point approaches utilise p-values to sequentially characterise whether the new datum is consistent with the current distribution estimates. A calibration procedure with fixed storage requirements is also employed to allow the threshold to be set to control for the false positive rate.

This chapter is structured as follows; Section 4.1 outlines our change point definition which ensures trends and outliers are not falsely classified. We additionally describe relevant change point approaches and discuss existing methods that are suitable for data with trend. In Section 4.2 the change point procedure is detailed. Section 4.3 demonstrates the performance of the proposed change point approaches to competitive methods on a variety of data cases with and without change points and trends. The procedures are then applied to a real financial data set in Section 4.4 to evaluate the applicability of the procedure to real data.

4.1 Background and Literature Review

In this work the problem of multiple change point detection on indefinite data streams is the main focus where change points correspond to times in the data where its statistical properties change [Killick et al., 2012]. As highlighted earlier, it is important to distinguish between change points, which are instantaneous changes in the distribution and trends which correspond to gradual changes. In mathematical terms, consider a stream of observations $x_1, x_2, \ldots$ sampled i.i.d. from random variables $X_1, X_2, \ldots$ where change points occur at times $\tau_1, \tau_2, \ldots$ such that,

$$
\begin{aligned}
&X_1, \ldots, X_{\tau_1} \sim F_1 \\
&X_{\tau_1+1}, \ldots, X_{\tau_2} \sim F_2 \\
&X_{\tau_2+1}, \ldots, X_{\tau_3} \sim F_3 \text{ etc.}
\end{aligned}
$$

where $F_1, F_2, \ldots$ are distinct distributions i.e. $F_i \neq F_{i+1}\ \forall i$ and $\tau_{i+1} - \tau_i > 1$. The final condition is added to ensure both trends and outliers are not classed as change points however traditionally this criteria is omitted (e.g. Adams and MacKay [2007] and Aminikhanghahi and Cook [2017]). To highlight this, a trend in Gaussian data can be expressed as,

$$\begin{aligned}
&\tau = 1, 2, \ldots, \\
&F_1 \sim N(\mu, \sigma^2) \\
&F_2 \sim N(\mu + \kappa, \sigma^2) \\
&F_3 \sim N(\mu + 2\kappa, \sigma^2) \text{ etc.}
\end{aligned}$$

where κ is the fixed trend gradient, σ^2 is the constant variance and μ is the initial mean of the series. Alternatively an outlier at time t can be expressed mathematically as,

$$\begin{aligned}
&X_1, \ldots, X_{t-1} \sim F_1 \\
&X_t \sim F_2 \\
&X_{t+1}, X_{t+2}, \ldots \sim F_1.
\end{aligned}$$

Without the requirement of $\tau_{i+1} - \tau_i > 1$, both trend and outliers could be classified as change points. For clarity, in the remainder of this chapter *change points* refer to the true times of change in the data and *detections* refer to the times that are identified as change points by a change point method (both true and false positives).

Change point detection procedures can be split into two approaches, batch and sequential. Batch approaches utilise the full view of the data to determine optimal partitions often through minimising a cost function. Popular approaches include Binary Segmentation [Scott and Knott, 1974], Wild Binary Segmentation [Fryzlewicz, 2014] and Pruned Exact Linear Time (PELT) [Killick et al., 2012]. Batch methods are not suitable for application to data streams as it is infeasible to store the full stream. These procedures often require tuning of hyper-parameters and thresholds however change point labels are often not available for streaming data where Romano et al. [2021] note that default parameters often do not account for the local fluctuations in the data.

Sequential change point detection is the alternate approach where the data stream is processed as it becomes available, assessing at each iteration whether to flag a change. Sequential methods have no delay in detection whereas batch approaches have a fixed delay [Tartakovsky et al., 2006]. Sequential detection procedures are more suitable for timely detection of change points which is important in many applications in medicine [Clifford et al., 2015], finance [Plasse and Adams, 2019], earthquake tremor detection [Xie et al., 2019] and many other domains. Additionally, data streams are often high frequency thus it is ideal for procedures to

have constant memory and process observations at least as quickly as they arrive.

Much of the literature for sequential change point detection falls under statistical quality control such as the Cumulative Sum (CUSUM) [Page, 1954] and Exponentially Weighted Moving Average (EWMA) [Roberts, 1959] methods. [Lai, 1995] provides an in depth overview of classical approaches. Other methods include monitoring the likelihood ratio using both parametric (e.g. [Willsky and Jones, 1976], [Gustafsson, 2001]) and non-parametric approaches (e.g. [Kawahara and Sugiyama, 2012]). Most sequential approaches suffer from difficulties in threshold setting and large computational burdens [Gustafsson, 1996]. Kernel approaches have also been applied (e.g. [Desobry et al., 2005], [Harchaoui et al., 2009]) however these rely heavily on the choice of kernel function and parameters [Aminikhanghahi and Cook, 2017].

There have been several developments in Bayesian sequential change point detection due to the method proposed simultaneously by [Adams and MacKay, 2007] and [Fearnhead, 2006] called Bayesian Online Change Point Detection (BOCPD). Central to this method is the inference of the *run length*, corresponding to the time since the last change point. This method shows promising results, leading to its popularity and use. There have additionally been various extensions of this approach such as combining with a Gaussian process to remove the need for hyper-parameters (e.g. [Garnett et al., 2010], [Saatçi et al., 2010]) or direct hyper-parameter optimisation (e.g. [Turner et al., 2009]). Although the estimation procedure for this approach is sequential, change point detection is retrospective causing delays in detection. This issue is further discussed in Appendix B.1.1.

Adaptive filtering is another approach that can be used for change point detection. [Bodenham and Adams, 2017] utilises an adaptive filtering approach that employs forgetting factors, where change points are detected using control limits. Alternatively, [Plasse and Adams, 2019] focus on change point detection in categorical data using a similar adaptive framework but apply divergence to detect changes. Due to their success, and the success of Bayesian approaches, the Bayesian filtering approach presented in Chapter 3 is extended for change point detection.

Real data commonly contains periods of gradual trend or local fluctuations which under the traditional definition of a change point, would lead to a continual detection of changes in these regions, as the distribution is continually changing. Procedures that are built with this traditional definition raise large numbers of alerts during these periods. Change point literature that deals with trend is limited. [Gallagher et al., 2013] develop a method that considers trend, performing a similar analysis to CUSUM, however, they assume the data maintains a constant trend before and after a change point which is unrealistic in practice. Additionally, their method is applied to a single change point. [Militino et al., 2020] give an overview of methods used for trend and change points separately, but do so using batch approaches. It is also common for trend to be removed from the data before performing change point analysis (e.g. [Rebecca Killick and Idris Eckley and Philip

Jonathan (2013), however, this is more difficult in the sequential setting.

There have also been several works that focus on detecting both change points and changes in trend. Fearnhead et al. (2019) use continuous piecewise linear functions to model the data between change points rather than traditional piecewise constant functions. Their method, CPOP, allows for linear trends within the data. Baranowski et al. (2019) instead go further proposing the narrowest-over-threshold framework (NOT) which allows for higher order polynomial curves. Both approaches are however batch methods requiring the full series to be observed.

Although trend is present in most real data sets, very few change point procedures have been demonstrated on such data. Romano et al. (2021) have recently proposed DeCAFS, an approach for detecting abrupt changes in the presence of local fluctuations and autocorrelation. DeCAFS is again a batch procedure however the authors demonstrate its computational efficiency. In this chapter, there is a focus on *sequential* procedures suitable for detecting abrupt changes in the presence of trend with low false positive rates for application to streaming data. Additionally, only linear trends are investigated.

4.2 Change Point Detection Methodology

The Bayesian adaptive estimation procedure detailed in Chapter 3 allows accurate estimates for the model parameters to be evaluated over the stream. This framework has been extended for change point detection and is discussed further. To detect a change sequentially in the data stream, p-values are used to quantify whether the newly observed data point is consistent with the current estimated distribution. As outlined by Robins et al. (2000), various reference densities can be used for calculating Bayesian p-values such as the likelihood, prior predictive, posterior predictive and the partial posterior predictive distributions. Under this traditional framework predictive p-values are used as these are the most popular in literature. Additionally, an alternative approach is explored which utilises the posterior distribution directly.

4.2.1 Predictive P-Values

First the posterior predictive p-value is investigated. The posterior predictive distribution takes the form,

$$P_{PP}(x) = \int f(x|\boldsymbol{\eta}, \lambda)\pi_{post}(\boldsymbol{\eta}, \lambda|X_1, \ldots, X_t)d\boldsymbol{\eta}d\lambda \tag{4.1}$$

where $f(x|\boldsymbol{\eta})$ is the likelihood for the data and $\pi_{post}(\boldsymbol{\eta}, \lambda|X_1, \ldots, X_t)$ is the posterior derived in Section 3.2.7 given by Equation 3.13. As a closed form solution for the posterior of λ is intractable, the MAP estimate of λ is instead plugging in directly rather than marginalising with respect to this parameter. This simplification significantly improves computation speed and improves change point

performance as seen in Table 4.1. The proposed approximated posterior predictive distribution is calculated as,

$$P_{APP}(x) = \int f(x|\boldsymbol{\eta}, \hat{\lambda}^{MAP})\pi_{post}(\boldsymbol{\eta}|X_1, \ldots, X_t, \hat{\lambda}^{MAP})d\boldsymbol{\eta} \tag{4.2}$$

where $\pi_{post}(\eta|X_1, \ldots, X_t, \lambda)$ is the marginal posterior distribution for $\boldsymbol{\eta}$ given by Equation 3.15 with λ plugged in to calculate the weights w_i. In the case when the data follows a Gaussian distribution as described in Section 3.2.1, the approximated predictive posterior follows a student t-distribution with:

- Centre $\hat{\mu}^{MAP}$ from Equation 3.9
- Precision $\Lambda = \frac{\hat{\alpha}\hat{K}}{\hat{\beta}(\hat{K}+1)}$
- Degrees of freedom $2\hat{\alpha}$

where

$$\hat{\alpha} = \hat{\lambda}^{MAP}\sum_{i=1}^{t-1}\frac{C_i}{2} + \frac{1}{2} + \alpha_0$$

$$\hat{\beta} = \beta_0 + \frac{1}{2}\left(\frac{\mu_0^2}{\sigma_0^2} + \hat{\lambda}^{MAP}\sum_{i=1}^{t-1}C_iX_i^2 + X_t^2 - \frac{\left(\hat{\lambda}^{MAP}\sum_{i=1}^{t-1}C_iX_i + X_t + \frac{\mu_0}{\sigma_0^2}\right)^2}{\hat{\lambda}^{MAP}\sum_{i=1}^{t-1}C_i + 1 + \frac{1}{\sigma_0^2}}\right)$$

are the parameters of the Inverse-Gamma marginal posterior for σ^2. Finally,

$$\hat{K} = \hat{\lambda}^{MAP}\sum_{i=1}^{t-1}C_i + 1 + \frac{1}{\sigma_0^2}$$

which is equivalent to the effective sample size [Wang et al., 2004] and is the divisor in the μ posterior in Equation 3.8.

Table 4.1: Change point performance comparison of exact and approximate posterior predictive p-value methods at 0.005 level with standard deviation given in brackets. Data simulated as described in Section 4.3.3 where $n = 5000$ with 20 repetitions and $\alpha = 0.005$.

	F1	ARL0	ARL1	Recall	Precision	FP	Positives	Time
BFF Pred Exact	0.728(0.118)	641.694	0.128	0.769(0.127)	0.700(0.140)	4.100	13.550	6036.798(476.040)
BFF Pred Approx	0.840(0.065)	1005.965	0.157	0.879(0.079)	0.810(0.080)	2.803	14.813	1.891(0.130)

The p-value is calculated from the reference density as the probability of observing something at least as extreme as the new observation x^*. This is expressed mathematically as,

$$p = \int_{x|P_{APP}(x) \leq P_{APP}(x^*)} P_{APP}(x)dx$$

Depending on the distribution, and the meaning of the data, these can be either one or two sided. For the Gaussian case, two-sided p-values are used. This method is referred to as *BFF-Pred*.

4.2.2 Posterior P-Values

Alternatively at each time step, estimates for the distribution parameters are calculated which themselves adapt over time. The p-values for newly calculated estimates of $\boldsymbol{\eta}_t$ and λ_{t-1} are computed by comparing to their corresponding posterior distribution at time $t-1$, $\pi_{post}(\boldsymbol{\eta}, \lambda | X_1, \ldots, X_{t-1})$. These p-values correspond to the probability of observing parameters at least as extreme as the MAP estimate $\boldsymbol{\eta}_t$ under the posterior at time $t-1$ given in Equation 4.3. This describes how surprised we are by the new estimates given the knowledge at the previous time step.

Unlike the posterior predictive approach, p-values for each distribution parameter are calculated instead of a single value. Using the notation from Section 3.2.7 the conditional marginal posterior for parameter η_i at time $t-1$ is

$$\pi^i_{t-1}(\eta) = \pi_{post}(\eta | \eta_{i+1}, \ldots, \eta_s, \lambda, X_1, \ldots, X_{t-1}) \tag{4.3}$$

where the joint posterior is given in Equation 3.15. Posterior p-values are calculated as either one or two sided quantities. The p-value for newly estimated parameter $\hat{\eta}_i$ using the posterior at time $t-1$ is expressed mathematically as,

$$p^{\eta_i} = \int_{\eta | \pi^i_{t-1}(\eta) \leq \pi^i_{t-1}(\hat{\eta}_i)} \pi^i_{t-1}(\eta) d\eta.$$

Similarly, a one sided p-value can be calculated for the newly estimated forgetting factor $\hat{\lambda}$ using the marginal posterior for λ at time $t-1$ as,

$$p^{\lambda} = \int_0^{\hat{\lambda}} \pi_{post}(\lambda | X_1, \ldots, X_{t-1}).$$

One sided p-values are used for the forgetting factor as only when this value reduces is of concern as it signifies a change in the distribution. This p-value method is named *BFF-Post* where the parameter used is detailed but we focus only on p^{λ} as it has superior performance.

4.2.3 Calibration

It is widely known that Bayesian p-values do not typically conform to the Uniform property of frequentist p-values as discussed by Meng 1994, Robins et al. 2000 and Gelman 2013. Calibration of p-values using their empirical cumulative distribution function is commonly used in practice e.g. Davison and Hinkley 1997, Robins et al. 2000 and Gelman 2013. This empirical calibration of p-values using a sliding window approach is implemented to maintained fixed storage requirements. For a specified sliding window length s and the current stream of p-values $p_1, \ldots, p_{t-1}$ at time $t-1$, the calibrated p-value of p_t at time t is calculated as:

$$p_t^{cal} = \frac{1}{s} \sum_{i=t-s}^{t-1} I(p_i \leq p_t). \tag{4.4}$$

The sliding window size s should be large enough to allow this approximation to accurately calibrate the p-values but not so large that it causes a computational burden. To calibrate the p-values for a sliding window of size s, the data length must be at least s with p-values calculated for these time points. A change point is detected when,

$$p_t^{cal} < c$$

for some threshold value $c \in [0, 1]$. Due to the uniform nature of the calibrated p-values, for threshold value c, approximately $100c\%$ of points are identified as change points. The threshold for change point detection under this scheme can be set to control for the false positive rate and a manageable number of alerts for an analyst.

Table 4.2: Change point performance comparison measured by *F1* as described in Section 2.4.3 for different sliding windows (SW) for the proposed p-value calibration method. Data simulated as described in Section 4.3.3 where $n = 20000$ with 100 repetitions and $\alpha = 0.005$. The performance is evaluated over the final 10000 data points.

	BFF-Pred	BFF-Post-λ
Uncalibrated	0.717(0.056)	0.000(0.000)
SW 200	0.562(0.058)	0.603(0.054)
SW 500	0.670(0.057)	0.763(0.052)
SW 1000	0.720(0.056)	0.805(0.042)
SW 2000	0.746(0.055)	0.835(0.044)
SW 5000	0.761(0.062)	0.856(0.043)
SW 7000	0.764(0.059)	0.859(0.044)
SW 10000	0.764(0.062)	0.862(0.046)

Both the raw posterior predictive and posterior p-values are not uniformly distributed hence meaningful thresholds cannot be set without calibration. The performance of the proposed change point procedures with and without calibration is explored for several calibration window sizes. In Table 4.2, BFF-Post in particular has very poor calibration where without the proposed recalibration, it does not successfully identify any change points. Alternatively, BFF-Pred p-values are better calibrated where the performance for small calibration windows performs worse than uncalibrated p-values. For all methods, as the sliding window increases, the performance improves however there is little difference in performance in sliding windows greater than 2000. The computation time for different sliding window sizes is investigated in Table B.1 where it is clear, larger windows result in much longer computation times. In all future results, calibration of p-values is performed with a sliding window of 2000 to keep storage requirements small with minimal reduction of performance.

4.2.4 Grace Period

Following a detection by the change point procedure, the adaptive filter does not initially produce accurate estimates for the new data distribution as seen in Figure 3.5. This may cause additional false detections as the newly observed data does not follow the distribution of the model. As done by Bodenham and Adams [2017], to prevent such inconsequential detections, a *grace period* is implemented after a detection. Given grace period length G, if there is a detection at time τ, parameter estimation continues in the period $(\tau, \tau + G]$ but change point detection is disabled. This grace period is illustrated in Figure 4.2. After a detection, Bodenham and Adams [2017] restart the mean estimation procedure. Under the proposed Bayesian framework, estimation of all distribution parameters is performed, hence if there is a change point in a single parameter, it is not beneficial to restart the estimation procedure. Due to this reasoning, the adaptive filter is not restarted after a detection.

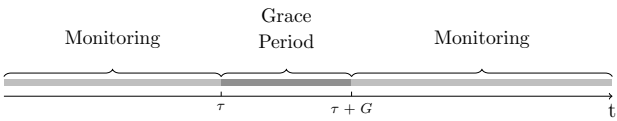


Figure 4.2: Illustration of the grace period of length G which begins after a change point is detected at time τ. Detections are not made in this grace period.

In principle, G should correspond to the length of time required for the filter to adapt to the new distribution where longer distributional shifts may require larger grace periods. The grace period can also be regarded as a mechanism for controlling the minimum false positive rate as it acts as a lower bound on the required number of observations to be seen before a false positive is flagged [Plasse

and Adams, 2019]. Given the context of the data and the desired false positive rates, an appropriate grace period for the data can be set. The grace period assumes the data returns to a stationary state after an abrupt change point and is suitable for such cases. Otherwise, detections made after this grace period cannot be trusted.

The effect of the grace period on the change point performance measured by the *F1* score is now explored. The results of these experiments are displayed in Table 4.3 where the data is simulated as described in Section 4.3.3 where $n = 10000$ with 100 repetitions. As the grace period increases, the *F1* score decreases in value for all change point procedures. However, this difference in performance is marginal. Although the performance has not improved, it is important to still implement a grace period to prevent false positives in practice.

Table 4.3: Comparison of change point performance measured by the *F1* for grace periods of different lengths. Data simulated as described in Section 4.3.3 where $n = 10000$ with 100 repetitions and $\alpha = 0.005$. Performance assessed on the final 5000 points.

Grace Period	BFF-Pred	BFF-Post-λ
1	0.744(0.051)	0.834(0.042)
5	0.742(0.052)	0.833(0.042)
10	0.741(0.052)	0.833(0.042)
20	0.735(0.056)	0.832(0.043)
50	0.712(0.060)	0.813(0.047)

Algorithm 1 outlines the general BFF change point procedure for the exponential family of distributions.

4.2.5 Ability to Detect Changes in Presence of Trend

Data streams in practice may exhibit periods of trend or gradual shifts in the data. Commonly, these changes do not signify a change point but most detection procedures regard them as such. The proposed change point procedure aims to only identify true distinct changes in the distribution of the data to prevent large numbers of false positives from being raised.

This property for the proposed approach for the posterior p-value case is now demonstrated. Suppose the data is univariate Gaussian with unknown mean and variance, $N(\mu, \sigma^2)$. Up to time $t-1$ the following data is observed,

$$X_1, \ldots, X_{t-1} \overset{iid}{\sim} N(\mu_1, \sigma_1^2),$$

where at time t, a change point occurs in the mean with magnitude change κ,

$$X_t \sim N(\mu_1 + \kappa, \sigma_2^2) \text{ where } \kappa > 0.$$

Algorithm 1 Bayesian Forgetting Factor Change Point Algorithm

Input: Threshold level $c \in [0, 1]$, grace period length G and p-value calibration sliding window length s, the fixed λ prior $P(\lambda)$

1: Initialise hyper-parameters:
$\boldsymbol{\chi}_1 = \boldsymbol{\chi}_{initial}$
$\upsilon_1 = \upsilon_{initial}$
2: Observe new data X_t.
3: Calculate p_t using either BFF-pred or BFF-post p-value methods.
4: Calibrate using Equation 4.4 to give p_t^{cal}.
5: **if** $p_t^{cal} < c$ where c is the threshold value and last change point not in $[t-G, t-1]$ **then**
6: X_t is a change point and add t to change point set.
7: **end if**
8: Update parameter estimates for λ_{t-1} and $\boldsymbol{\eta}_t$ using sequential update formulas for the MAP estimates of Equation 3.14 and Equation 3.15 respectively.
9: Update Prior Parameters
$\boldsymbol{\chi}_t = f_\chi(\boldsymbol{\eta}_t, \lambda_{t-1}, \sigma_0)$
$\upsilon_t = f_\upsilon(\boldsymbol{\eta}_t, \lambda_{t-1}, \sigma_0)$
where $f_\chi(\cdot)$ and $f_\upsilon(\cdot)$ are functions to calculate the hyper-parameters such that the modes of the priors are equal to the current MAP estimates with prior contribution weight equal to 1 as described in Section 3.2.6.
10: Return to 2.

For simplicity it is assumed that the MAP estimates of μ is equal to the true value, μ_1, before the trend. Let the MAP estimate of σ_1^2 be $\hat{\sigma}^2$. Using Equation 3.8, the posterior at time $t-1$ for μ is:

$$\mu|\hat{\sigma}^2 \sim N\left(\mu_1, \frac{\hat{\sigma}^2}{\sum_{i=1}^{t-1} w_i + \frac{1}{\sigma_0^2}}\right) \tag{4.5}$$

where σ_0^2 is the prior variance for μ. The two sided posterior p-value at time t for $\mu_2 = \mu_1 + \kappa$ is calculated as:

$$\begin{aligned} p &= 2\Phi\left(\frac{(\mu_1 - (\mu_1 + \kappa))\left(\sum_{i=1}^{t-1} w_i + \frac{1}{\sigma_0^2}\right)}{\hat{\sigma}^2}\right) \\ &= 2\Phi\left(\frac{-\kappa\left(\sum_{i=1}^{t-1} w_i + \frac{1}{\sigma_0^2}\right)}{\hat{\sigma}^2}\right). \end{aligned}$$

To be detected at the α level, the gradient, κ, of the trend must satisfy,

$$\kappa \geq \frac{\hat{\sigma}^2}{\sum_{i=1}^{t-1} w_i + \frac{1}{\sigma_0^2}} \Phi^{-1}(1 - \alpha/2). \tag{4.6}$$

Using Equation 4.6, the noisier the data (larger $\hat{\sigma}^2$), the larger the jump must be for a change to be detected. Alternatively for stationary data, as the length of the stationary period increases, $\sum_{i=1}^{t-1} w_i$ increases causing the jump size to decrease. Hence, as the time since the last change point increases, the magnitude of change in the data required to detect a change point decreases.

Trend periods can be characterised by continual gradual changes in the data. During trends, the forgetting factor reduces as historic data is less relevant which was observed in Figure 3.2 (b). Thus $\sum_{i=1}^{t-1} w_i$ decreases. The estimate of the variance during trend periods increases as the data itself is noisier (although the true variance of the data is fixed). Using Equation 4.6, with larger variance and smaller weights, much larger κ values are required for a change to be detected. This helps to explain how the procedure is capable of detecting abrupt changes in the presence of trend as seen in Sections 4.3.3 and 4.4 where the proposed method does not erroneously detect large numbers of false positives during trend periods.

4.3 Simulation Study

We designed a simulation study to evaluate the performance of the proposed methods. In our study we focused on comparison to sequential methods. Comparison to the Pruned Exact Linear Time (PELT) method [Killick et al., 2012], a well regarded batch approach is implemented to act as a benchmark to the sequential approaches. We additionally compare to DeCAFS [Romano et al., 2021], a batch procedure that is capable of detecting abrupt changes in the presence of trend and autocorrelation. Classic sequential change point procedures are also compared to including the Cumulative Sum (CUSUM) algorithm [Page, 1954] and Exponential Weighted Moving Average (EWMA) [Roberts, 1959]. The frequentist sequential adaptive estimation procedure by [Anagnostopoulos et al., 2012] which BFF is an extension of, has been used for change point detection by [Bodenham and Adams, 2017] using control limits and is called *AFF*.

Both the proposed procedure, BFF, and the frequentist version, AFF, require a threshold for detection. For both procedures, this threshold can be set based on the desired false positive rate where for threshold c approximately $100c\%$ of points are detected as change points. In the following examples a range of thresholds around the true theoretical threshold value are employed.

We also compare to the popular Bayesian sequential change point procedure called *Bayesian Online Change Point Detection* (BOCPD) proposed simultaneously by [Adams and MacKay, 2007] and [Fearnhead, 2006]. This procedure jointly and sequentially infers the distribution parameters of the data alongside the run

length, representing the time since the last change point, and is used to determine whether a change has occurred. This method is detailed further in Appendix B.1.1. Although the BOCPD estimation procedure is sequential, the change point identification procedure performs change point detection retrospectively. Through investigation of this method empirically, it was seen that although the procedure correctly identifies the time of change, it often did so with a lag. Table 4.4 displays the delay in detection for simulated data as described in Section 4.3.3 for streams of length 10000. The BOCPD method has a significant delay in detection when no threshold is used (the *Colmaxes* approach) however this reduces as larger thresholds for this probability are implemented. As this method requires probabilities for each possible run length to be stored to identify change points, the storage requirements for this method are higher than the other sequential procedures detailed. Both the lag of detection and storage requirements of the BOCPD method should be considered when compared to the other sequential methods.

Table 4.4: Mean, median and standard deviation of the change point detection lag for the BOCPD method for data simulated as described in Section 4.3.3 of length 10000.

	Mean	Median	Standard Deviation
BOCPD-Colmaxes	12.009	12.300	5.838
BOCPD-Threshold 0.5	5.916	5.286	5.311
BOCPD-Threshold 0.75	2.786	1.500	3.113
BOCPD-Threshold 0.9	3.570	2.000	4.815

A summary of the properties of the comparison methods is provided in Table 4.5 where the default parameters for each model are detailed in Appendix B for reference. For the AFF and BFF approaches a variety of thresholds are explored which are specifically detailed for each experiment.

In single change point literature, the *average run lengths*; ARL0 and ARL1, are widely applied. The ARL0 is the average number of observations until a false positive is observed whereas the ARL1 is the average time between a change point and detection. Sequential change point methods should have large ARL0 and small ARL1. A similar extension of these measures done by Bodenham and Adams [2017] is used to adapt these measures for multiple change points on data streams. The ARL0 is calculated as the average time between false detections and the ARL1 as the average time between a change point and detection.

For the multiple change situation, the average run lengths do not give a full picture of performance in terms of the number of change points correctly and incorrectly identified, hence additional performance measures are needed. The recall (proportion of change points correctly identified) and precision (proportion of detections that are not false) are used as defined in Section 2.4.3. Both recall and precision take on values in $[0, 1]$ where it is desirable for both to be close to

Table 4.5: Summary of the main characteristics of each method in terms of if the method is appropriate for data streams, the hyper-parameters required and whether it is a Bayesian or frequentist approach.

	Sequential	Hyper-Parameters	Bayesian
PELT	✗	function, penalty	✗
DeCAFS	✗	β	✗
CUSUM	✓	h, k	✗
EWMA	✓	ω, L	✗
AFF	✓	η, c, G	✗
BOCPD	✓	μ, κ, λ, α, β & threshold	✓
BFF	✓	c, G, s, data distribution	✓

1. As there is a trade off between recall and precision, the *F1* measure is used to combine these aspects. The *F1* measure takes on values in $[0, 1]$ where values closer to 1 are desirable and correspond to all changes being detected with no false detections. Finally, as the methodology is to be applied to large data streams, the methods need to be scalable, hence the run times of each algorithm is reported.

In the above measures, the definition of a true detection must be established. It is standard in change point literature to allow a window for detection. In batch approaches this window can be either side of a change however in sequential methods, this period only occurs after the change. The detection window size is set to 20 time points. The number of false positives (FP) and total detections made by each procedure are additionally displayed.

4.3.1 Performance on Change Point Only Data

Although the focus of this work is on methods appropriate for trend, it is important to measure the baseline performance of the change point procedures without this behaviour. Here 100 Gaussian data streams of length $n = 50,000$ are simulated where $m = \lceil \frac{n}{200} \rceil = 250$ change points are generated uniformly across $(30, n-1)$ such that change points are at least 30 time steps apart where jump sizes are sampled from $U(3, 6)$. The simulated Gaussian data has a fixed variance of 1 with a mean as specified by the change points.

As seen in Table 4.6, the best performers for the change point only data are PELT, DeCAFS and BOCPD followed by the BFF methods. Both batch methods, PELT and DeCAFS perform well on this data as there are clear distinct change points. In comparison to *sequential* methods, BFF has high-performance. In particular, the precision is very high for BFF with a small ARL1. Thus in practice BFF has very few false positives making it favourable. These results provide a

Table 4.6: Change point performance over 100 simulated Gaussian data streams of length 50,000 containing only change points and no trend as described in Section 4.3.1. Standard deviation given in brackets. Best performer in bold, second best in grey bold.

	F1	ARL0	ARL1	Recall	Precision	FP	Detections	Time (sec)
PELT	**0.980**(0.009)	7970.875	0.025	0.980(0.009)	0.979(0.009)	4.940	240.430	0.158(0.020)
DeCAFS	0.973(0.010)	9064.632	0.023	0.967(0.012)	0.979(0.009)	5.050	237.380	4.494(0.341)
CUSUM	0.647(0.027)	283.093	2.546	0.830(0.026)	0.530(0.027)	177.330	376.820	**0.002**(0.001)
EWMA	0.738(0.023)	420.905	1.734	0.875(0.022)	0.638(0.026)	119.460	329.890	**0.002**(0.001)
AFF 0.008	0.682(0.025)	293.232	2.996	0.870(0.025)	0.561(0.026)	163.930	373.100	0.003(0.002)
AFF 0.005	0.731(0.025)	370.537	2.837	0.887(0.023)	0.622(0.028)	129.750	343.000	**0.002**(0.001)
AFF 0.003	0.783(0.022)	485.998	2.780	0.909(0.018)	0.688(0.026)	99.180	317.550	0.003(0.002)
BOCPD-Colmaxes	0.965(0.011)	3514.038	**0.014**	**0.984**(0.009)	0.946(0.016)	13.550	250.050	530.833(33.812)
BOCPD-Threshold	0.970(0.011)	5083.008	0.017	0.976(0.014)	0.964(0.013)	8.790	243.350	530.833(33.812)
BFF-Pred 0.008	0.729(0.020)	493.474	0.141	0.805(0.023)	0.666(0.021)	96.930	290.420	26.137(2.529)
BFF-Pred 0.005	0.764(0.020)	1420.588	0.122	0.703(0.022)	0.836(0.023)	33.250	202.280	26.137(2.529)
BFF-Pred 0.003	0.668(0.019)	5603.144	0.086	0.519(0.019)	0.938(0.023)	8.340	133.070	26.137(2.529)
BFF-Post-λ 0.008	0.872(0.016)	2728.559	0.986	0.828(0.019)	0.920(0.020)	17.290	216.270	127.714(12.986)
BFF-Post-λ 0.005	0.796(0.015)	9335.263	1.016	0.672(0.019)	0.977(0.014)	3.820	165.310	127.714(12.986)
BFF-Post-λ 0.003	0.653(0.018)	**11795.341**	1.010	0.487(0.020)	**0.994**(0.009)	**0.750**	117.690	127.714(12.986)

benchmark for the performance results in Section 4.3.3 where if similar performance is observed, the methods demonstrate robustness when subject to trend. When applied to data with trend alone it is clear that for DeCAFS, BOCPD and BFF, the number of detections is not affected by the presence of trend in the data (see Appendix B.3).

4.3.2 Illustrative Example

To begin the change point performance of the comparison methods is showcased on a simulated example of length $n = 10,000$, however, only the final 5000 points are displayed to clearly visualise the results and allow for a burn-in period. The data is generated as described in Section 4.3.3 where change points and trends are added throughout. For this example, the parameters of each of the methods are optimised to display their best efforts on the problem. It is noted however that this is not feasible in application as locations of the change points are unknown. A grid search is performed for each model where the parameters that optimise the *F1* score are chosen. The list of parameters for which this search is performed can be found in Appendix B where the optimal parameters are detailed.

Figure 4.3 displays the best efforts for each method. DeCAFS perfectly identifies the change points and is a benchmark for sequential methods as it is unfair to compare to batch methods. Many of the sequential procedures do not perform well during trend periods with large numbers of false positives identified during these regions. BOCPD does however show strong performance where there is a single false detection and all change points are identified in this period. It is important to note that for this method, the *F1* value is heavily reliant on the choice of prior parameters and threshold. In addition, it is significantly slower than all other methods. Both the competitive sequential methods (AFF, CUSUM

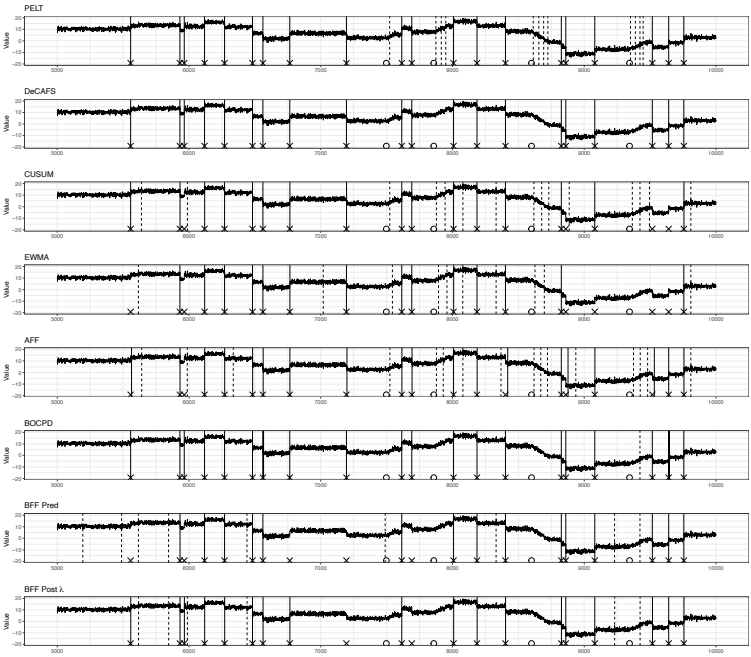


Figure 4.3: Single simulation illustration as described in Section 4.3.2 containing both change points and trend. For each change point procedure, solid and dotted vertical lines represent correct and incorrect change point detections respectively. A cross at the bottom of the figure indicates the location of true change points and a circle defines the start of a trend period.

and EWMA) and PELT are not robust to trend, unlike the proposed procedures. These competitive methods do however detect a large proportion of the change points.

As seen in Figure 4.3, the proposed change point procedures, BFF-Pred and BFF-Post-λ, both have many false detections throughout the region. Due to its formulation, our method is susceptible to detecting both change points and anomalies or outliers within the data. Alternatively, it does not have false detections during trend periods. BFF-Post-λ detects a higher number of the changes compared to BFF-Pred with fewer false positives making it more favourable.

Figure 4.4 demonstrates the parameter estimation of BFF for λ for the simulated data as described in Section 4.3.2. This estimate reflects the desired prop-

erties given in Section 3.2.3. When a change point occurs, marked by a cross in the figure, there is an abrupt drop in the λ value before returning to the baseline value. This behaviour is not seen for trends where instead, the forgetting factor reduces slightly.

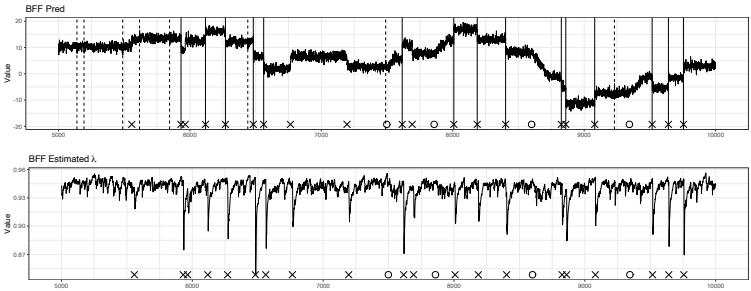


Figure 4.4: Single simulation illustration. First plot shows the change points detected by BFF-Pred on the raw signal where the second plot shows the estimated λ value for the BFF model with corresponding change points and trend positions detailed by crosses and circles respectively.

As these results are demonstrated on a single data set, they do not give a representative view of the performance of these methods. Performance comparison of these procedures is now performed using Monte Carlo replication on larger data sets.

4.3.3 Large Scale Simulation with Change Points and Trend

The robustness of the proposed procedure is now evaluated for large scale change point detection on data with trend. Data is simulated from a Gaussian distribution using a similar approach to that found in the literature (e.g. Killick et al. 2012, Bodenham and Adams 2017). As data streams often include periods of trend, periods of such behaviour are included which is seldom addressed in change point literature. For a data stream of length n, $m = \lceil \frac{n}{250} \rceil$ change points and $p = \lceil \frac{n}{1000} \rceil$ trend regions are generated uniformly across $(30, n-1)$ such that both change points and trend periods are at least 30 time steps apart with the following properties,

- Change points have mean jump magnitude sampled from $U(3, 6)$
- Periods of trend have a minimum length of 50 points with gradient magnitude uniformly sampled from $\{0.05, 0.06, 0.07, 0.08\}$

and the simulated Gaussian data has fixed variance 1 with a mean as specified by the change point and trend periods. As the methodology is intended to be applied

to data streams indefinitely, the simulated data are sizeable to test this, hence $n = 250,000$. For the results in Table 4.7, the experiments are replicated 100 times using the defaults of each of the change point procedures which are specified in Appendix B. For both AFF and BFF methods in Table 4.7, the threshold values are varied where by construction, 0.4% of the data points are change points. Thus a threshold close to this value is sensible.

Table 4.7: Simulated change point performance results over 100 streams of length 250,000 containing both change points and trend as described in Section 4.3.3. Standard deviation given in brackets. Best performer in bold, second best in grey bold.

	F1	ARL0	ARL1	Recall	Precision	FP	Detections	Time (sec)
PELT	0.757(0.007)	408.268	0.025	**0.980**(0.005)	0.617(0.008)	604.546	1576.814	0.852(0.101)
DeCAFS	**0.965**(0.005)	13285.22	0.022	0.951(0.007)	**0.980**(0.005)	18.990	962.470	99.724(6.642)
CUSUM	0.507(0.011)	188.985	2.837	0.792(0.014)	0.373(0.009)	1322.557	2108.155	0.013(0.005)
EWMA	0.584(0.009)	239.769	2.039	0.846(0.012)	0.446(0.008)	1042.010	1880.948	**0.012**(0.006)
AFF 0.008	0.529(0.009)	184.981	3.301	0.845(0.012)	0.385(0.007)	1339.526	2177.711	0.013(0.008)
AFF 0.005	0.569(0.009)	211.621	3.167	0.866(0.011)	0.423(0.008)	1170.804	2030.268	0.014(0.009)
AFF 0.003	0.606(0.010)	241.469	3.057	0.884(0.011)	0.461(0.008)	1025.814	1902.918	0.014(0.010)
BOCPD-Colmaxes	0.609(0.022)	213.112	0.018	0.957(0.054)	0.448(0.025)	1177.617	2127.287	4213.459(404.630)
BOCPD-Threshold	0.871(0.034)	1496.274	**0.017**	0.925(0.081)	0.831(0.047)	192.936	1110.170	4213.459(404.630)
BFF-Pred 0.008	0.676(0.010)	403.567	0.162	0.826(0.012)	0.572(0.010)	613.872	1433.723	150.098(18.974)
BFF-Pred 0.005	0.749(0.011)	1038.357	0.151	0.743(0.012)	0.756(0.012)	238.468	975.245	150.098(18.974)
BFF-Pred 0.003	0.703(0.010)	3502.145	0.125	0.580(0.010)	0.891(0.012)	70.415	646.064	150.098(18.974)
BFF-Post-λ 0.008	0.839(0.009)	1172.669	1.038	0.875(0.009)	0.805(0.010)	210.713	1079.213	769.904(101.217)
BFF-Post-λ 0.005	0.833(0.008)	4263.764	1.083	0.755(0.010)	0.928(0.009)	57.777	806.574	769.904(101.217)
BFF-Post-λ 0.003	0.722(0.009)	**20919.567**	1.084	0.572(0.010)	**0.980**(0.007)	**11.415**	578.745	769.904(101.217)

As the *F1* score combines both recall and precision, this measure is capable of summarising both aspects of performance. DeCAFS is used as a gold standard for sequential approaches which exhibits a very high F1 score. From Table 4.7, BOCPD-Threshold is the best sequential performer with BFF-Post-λ following closely. In comparison to purely sequential methods, BFF has superior performance with much higher precision. Many of the methods have large numbers of false positives due to the trend periods within the data. As trend is a natural feature of many real data sets, these methods may not have accurate results in practice.

It is important to compare the performance of this experiment to that without trend in Table 4.6. The performance of PELT and BOCPD reduce significantly when trend is introduced. In particular, the precision of these methods is much lower. Sequential methods (CUSUM, EWMA and AFF) also have a reduction in performance which is attributed to both recall and precision reductions. The BFF procedures, particularly the BFF-Post-λ approach still exhibit high precision with similar recall. This suggests the BFF methods are robust to trend.

For a threshold of 0.004, a procedure should detect approximately 0.4% of points in the stream (equivalent to 1000 for length 250,000) if it is well calibrated. AFF however detects many more suggesting it is not well calibrated. The BFF methods do approximately make this number of detections suggesting it is well calibrated. AFF alternatively detects many more changes at the same levels.

Figure 4.5: IBM open stock price between January 2, 1998 to January 21 2021.

Calibration issues can cause the threshold to be hard to set in practice when labels are unavailable. However, at each threshold, the BFF procedures detect approximately the correct number of changes thus can be set to control for the false positive rate.

In Table 4.7, between the two proposed p-value methods, the BFF-Post approach has better results, particularly for λ. BFF-Post-λ however suffers from longer computation times and larger ARL1 in comparison to BFF-Pred. The ARL1 of the BFF-Post-λ approach is still smaller than that of the other sequential approaches. Thus depending on the requirements of the application (faster computation or higher performance), either of the predictive or posterior approaches could be chosen as both have favourable performance.

Although the change point performance of BOCPD is very high, this is coupled with a large computational load and slow implementation. This procedure updates sequentially however it requires a long stream of run length probabilities to be stored where change points are determined retrospectively, potentially with a lag as it uses extra information to determine whether a change point has occurred. The BFF procedures are significantly faster with similar performance without large storage requirements.

4.4 Real Data Example

To demonstrate the robust performance of the proposed change point procedure, it is compared to competitive approaches on financial data, for which such monitoring is heavily required. In this context, the aim is to identify when the price of a stock has an abrupt change to trigger trading actions. This financial data is collected continuously during trading hours and can exhibit various changes to its structure such as trend, thus change point procedures must be suitable to prevent unwanted trading actions from being performed.

As the majority of real financial data is notoriously non-stationary, this section looks to gauge the suitability of the proposed methods for such data. It is common

in the literature to perform change point analysis on the log returns for financial data to deal with the non-stationarity. The robust performance of BFF is instead showcased on raw data. Here *one minute* intraday stock prices for IBM collected between January 2, 1998 and January 21, 2021[1] are used. Unlike the majority of experiments performed in literature, to test the suitability of BFF all methods are run over this full large scale data stream consisting of approximately 2.4 million points rather than on a smaller subset. The full data set is plotted in Figure 4.5 where at this scale, it is hard to visualise and locate the change points however there are clear trends in the data. Several real world events cause abrupt changes in the data that can provide explanations for the predicted change points.

Table 4.8: Number of detections for each change point method over the full IBM Open Stock price data for default implementations.

PELT	6411
DeCAFS	35820
AFF	27915
BOCPD-Colmaxes	19314
BOCPD-Threshold	737
BFF-Pred	4557
BFF-Post-λ	3084

Similar to the simulated case, default parameters are used for each model as it is unrealistic to tune these parameters when labels are not available. Additionally, EWMA and CUSUM are omitted from the results as these have the poorest performance on simulated data. Table 4.8 outlines the number of detections made by each of the procedures. There is a large discrepancy between the number of detections BOCPD makes for the Colmaxes and Threshold procedures. In the simulated example in Section 4.3.3, the threshold method was more favourable however it is very conservative on this data with very few detections made over a 20 year period. In the remaining discussion, the BOCPD-Colmaxes approach is used.

For illustration purposes the methods have been applied to the data points of October 2014, the results of which are presented in Figure 4.6. This period contains approximately 10,000 data points. In the figure, common change points between all comparison methods except AFF (due to its less favourable performance) are recorded with solid lines while the non-common changes are illustrated with dashed lines. During this period it is clear that AFF does not handle trend periods well where there are a large number of changes alerted throughout the first 18 days of the month and the final few days where a slight trend can be seen in the data.

There are two highly apparent change points in this period, one at the start of the 20th and another on the 21st. The change point on the 20th of October

[1] http://www.kibot.com/free_historical_data.aspx

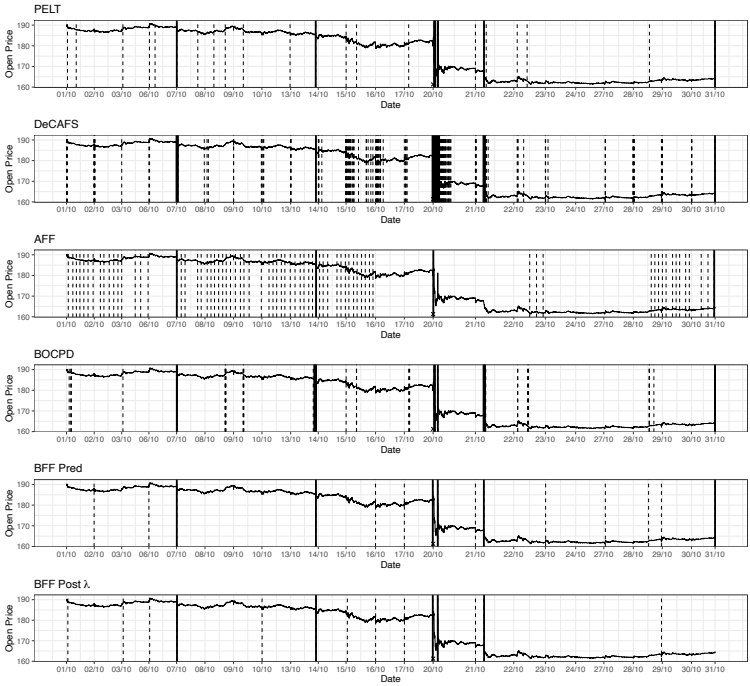


Figure 4.6: IBM open stock price throughout October 2014. Change points for each method are indicated as vertical lines where common change points between PELT, DeCAFS, BOCPD, BFF-Pred and BFF-Post are indicated with a solid vertical line (while dashed lines corresponds to change points that are not shared among all methods).

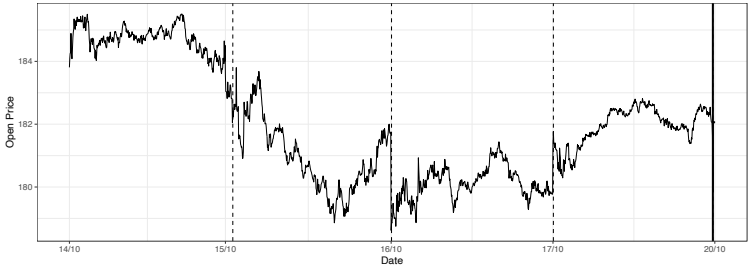


Figure 4.7: IBM open stock price between October 14 2014 to October 17 2014 where BFF-Post-λ change points are identified with vertical lines for detections in common between all methods and dashed lines for change points that are not shared among all methods.

corresponds to IBM dumping their chip unit and posting disappointing earnings[2] resulting in a drop in the price. All methods except AFF detect both of these changes. It can be seen that PELT, DeCAFS and BOCPD may additionally be detecting trend rather than only change points such as the point in the middle of October 15th which does not correspond to an apparent change in the data.

In Figure 4.6 both BFF methods perform well where once again they are detecting change points and extreme points within the data. Interestingly these methods detect different points where these differences primarily correspond to a difference in anomalies detected. BFF-Post-λ has more detections in the first half of the data whilst BFF-Pred detects a larger proportion during the second half. Figure 4.7 zooms in on the period between October 14th and October 17th 2014 to better visualise the change points detected by BFF-Post-λ during this period. At this scale, it is evident the detected change points correspond to abrupt changes in the data and highlight the importance of scale when investigating these change points as these are undetectable in Figure 4.6.

Another important detail when running these methods over large data streams is the computational speed. For this data the computation of BOCPD ran over 2 days and DeCAFS had a run time of 196 minutes. The BFF models ran for 25 (Pred) and 135 (Post) minutes with comparable performance. Additionally, these computation times are suitably small such that when run over a stream, updates can be computed at least as fast as the arrival of the data (0.003 seconds per update on average). Thus the proposed models show robust performance on real data in addition to the simulated data presented. Finally, it is noted that the BFF methods have similar detections to PELT and DeCAFS which are both strictly not sequential.

[2]https://money.cnn.com/2014/10/20/investing/ibm-sale-earnings/

4.5 Discussion

Sequential change point detection in the presence of trend is a challenging problem and in this chapter, it is demonstrated that existing methods are unable to operate in such cases. In streaming data, as the future structure is unknown, change point procedures must be capable of adapting. This work shows the capabilities of the proposed BFF change point procedure in the presence of trend in comparison to existing procedures where BFF particularly benefits from low numbers of false positives. The proposed default parameters are robust to a wide variety of cases where the detection threshold can be set based on the desired false positive rate.

In this chapter criticism of the popular BOCPD procedure has been provided. Although the parameter inference of this method is sequential, the change point aspect is retrospective. Change point locations are correctly identified but with a lag. Thus a direct comparison to this model is unfair for truly sequential models which do not flag changes in the past. Additionally, BOCPD requires a large number of values to be stored, increasing its computational burden and requires setting many influential prior parameters. When applied to the IBM data set, the computation time of BOCPD was excessive making it less suitable in practice.

Both of the proposed change point procedures based on the BFF method detailed in Chapter 3, have superior performance to other sequential approaches however the posterior approach has more favourable *F1* values. The predictive posterior change point procedure boasts a faster computation time than the posterior approach, whilst both are faster than BOCPD. From the simulated and real data examples explored, it is evident that the BFF change point procedure is capable of detecting abrupt mean changes in the presence of trend. The suitability of BFF for change point detection on cyber-security data is investigated in Chapter 8.

Chapter 5

Combined Forecasts for Improved Anomaly Detection

Anomaly detection in time series data is a challenging problem in many domains including medical, financial and computer networks. Anomaly detection aims to identify when the series deviates away from its "normal" behaviour, where this normal behaviour can itself be a challenge to model. Time series data streams in practice often exhibit trend, seasonality and irregularities, making the problem of anomaly detection more difficult. For enterprise cyber-security data, a daily pattern of activity is often observed as a result of habitual human interaction on the network during working hours.

Chandola et al. 2009 describe three types of anomalies: point, contextual and collective anomalies. *Point* anomalies are the most analysed type for time series data e.g. Kato and Klyuev 2014 and Ding et al. 2018 and refer to instances when single data points deviate globally from the data. Point anomaly detection in cyber-security can alert to extreme outliers in the data and may correspond to attacks such as Distributed Denial of Service (DDoS) or zero-day attacks which are characterised by high volumes of activity. Data instances that are unusual given their context (e.g. time of day) are classified as *contextual* anomalies and can be single points or collections of points. There are very few procedures that detect contextual anomalies. Monitoring of cyber-security time series for contextual anomalies can be used for detecting intruders with a foothold on the network as they often create many novel connections Metelli and Heard 2019 and do not conform to the usual patterns of the data. Finally, a collection of related data points are classified as a *collective* anomaly if individually these points may weakly be classified as point or contextual anomalies but their occurrence together makes them strongly anomalous.

This chapter is focused on the detection of point and contextual anomalies which are demonstrated in Figure 5.1. As most existing anomaly detectors focus on global outliers such as the highlighted point anomaly, non-extreme deviations from the usual periodic behaviour in the data such as the highlighted contextual

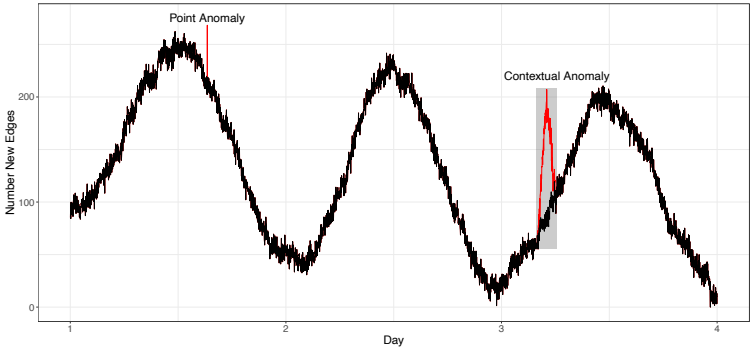


Figure 5.1: Simulated data with example of a point and contextual anomaly within the data where these anomalies are indicated in red.

anomaly would not be identified by most existing anomaly detection procedures. In this chapter methods capable of detecting both types of anomalies are proposed.

Although the proposed methods are motivated by cyber-security, they are not limited to this domain and can be applied to other settings such as monitoring social network data. The proposed models focus on identifying anomalies within univariate time series with regular temporal patterns. To achieve this two aspects of the data are modelled, the *long-term* structure and the *short-term* structure. The long-term model describes periodic behaviours and is well suited to detecting contextual anomalies. Functional Data Analysis (FDA) is implemented for this component. The short-term model instead captures the local behaviour of the data and is more appropriate for detecting point anomalies. A simple ARIMA model is applied for this.

We propose two anomaly detection approaches, both of which are capable of detecting point and contextual anomalies based on their forecasts. The first approach uses a novel time-series decomposition and the second combines short-term and long-term forecasts using weighted linear regression. The proposed methods are intended to be applied to time series data streams which are often non-stationary and never ending. The procedures presented in this chapter are thus adaptive with fixed computation requirements. The majority of existing popular time series anomaly detection procedures are however batch approaches, which are not suitable for application to streaming data due to their high computational storage and time requirements and their lack of adaptability.

This chapter is structured as follows. Section 5.1 provides a review of relevant literature for the described problem. To begin the adaptive short and long-term models are outlined in Section 5.2. These models are combined in Section 5.3 to produce a single forecast capturing both short and long-term structures where

details of the decomposition approach are described. The proposed anomaly detection methods based on the forecast models proposed are given in Section 5.4. The performance of the procedures is compared against competitive approaches on simulated data in Section 5.5 and on real labelled non-cyber data in Section 5.6.

5.1 Background and Relevant Literature

A number of methods have been applied for anomaly detection within network traffic data however these primarily focus on point anomaly detection. Bernacki and Kołaczek, 2015, propose an anomaly detection procedure for computer network time series using a modified Exponential Moving Average with subjective logic opinions which is well suited to periodic data. They however only demonstrate the detection performance on extreme outliers.

An Autoregressive Integrated Moving Average (ARIMA) process is a popular time series model that has been applied for network intrusion detection. Some examples of such work include Zhang et al., 2009, who apply a fixed ARIMA(1,1,1) model to available service rates of a server to predict DDoS attacks and Yaacob et al., 2010, who use an ARIMA model to predict future traffic rates to detect anomalies. For both papers, good point anomaly detection results are presented for application to small time series without seasonality suggesting ARIMA is a suitable model for capturing short-term aspects of the data for detecting point anomalies and is the method applied.

To handle the seasonality within cyber-security time series it is common to decompose the series into seasonal, trend and noise components as described in Section 2.2.2. The Seasonal Trend Loess (STL) Cleveland et al., 1990, decomposition is a very popular method used for seasonal adjustments of time series. STL uses a Loess smoothing technique Cleveland and Devlin, 1988, to iteratively solve for the seasonal and trend components. STL has been widely applied for anomaly detection. Verbesselt et al., 2010, utilise STL to decompose satellite image time series to find abrupt and gradual anomalies within the trend and seasonal components respectively. Although this approach is successful in detecting abrupt changes in the trend component, demonstration of detecting changes in the seasonal component was not presented. Additionally, this approach looks to find change points rather than anomalies.

Whilst seasonal trend decompositions are a popular approach, a wide range of other decompositions are available. Marcjasz et al., 2019, decompose electricity prices into long-term and short-term seasonal components using a Seasonal Component Autoregressive (SCAR) model. Taylor and Letham, 2018, decompose their series into trend, seasonality and holiday components, however, unlike STL they utilise regressions models to allow for interpretable parameters. Their method in particular focuses on scalability. Alternatively, it is common in vehicle traffic prediction to decompose the series into historic (long-term) and short-term activity

e.g. Chrobok et al. 2004 and Li et al. 2015, and is the approach taken. Both Chrobok et al. 2004 and Li et al. 2015 utilise a simple average over previous days to model the historic patterns or *long-term* structure in the data. This average approach for the long-term model is applied due to its simplicity and wide application for this problem.

There is a limited number of methods available for contextual anomaly detection. The method proposed by Munir et al. 2018 named deep learning-based anomaly detection approach (DeepAnT) is one of the few methods suitable for detecting both point and contextual anomalies in time series data. DeepAnT consists of two modules, a time series predictor which uses deep convolutional neural networks and an anomaly detector module that detects anomalies via the euclidean distance between the forecast and true value. This approach however requires a threshold to be set to identify anomalous euclidean distances, with no clear guidance for setting this value. Despite the promising results presented by this work, comparison to this approach is not explored as code is not publicly available. Twitter Inc. have also released a robust anomaly detection procedure appropriate for detecting both local (contextual) and global (point) anomalies in time series data [Kejariwal, 2015, Hochenbaum et al., 2017] combining a modified STL decomposition and ESD with robust statistics to detect both point and contextual anomalies. A downfall of this method however is that it does not perform well when the time series trend is changing [Munir et al., 2018] and is an offline approach.

5.2 Short and Long Term Models

The short and long-term aspects of the data are modelled to build anomaly detection procedures capable of detecting both point and contextual anomalies. In the literature described in Section 5.1, the majority of time series anomaly detection procedures focus on point anomalies, more specifically global outliers. Contextual anomalies are equally important, especially in cyber-security where to detect these types of anomalies, the regular patterns of the data must be modelled. Time series data is often not stationary thus the methods we present are adaptive and capable of running autonomously on data streams. We now outline the methods used to model the short and long-term aspects of the data separately. These methods are then combined in Section 5.3 for improved forecasting and Section 5.4 for anomaly detection.

Throughout this chapter, a time series random variable is denoted using upper case X_i with corresponding observed values given by lower case x_i and forecast value given by $\hat{x}_i$. Here the subscript i denotes the time of the observation. For example, x_i is the observation of the series at time i.

5.2.1 Short-Term Model

The short-term model describes the local trends and deviations of the time series rather than regular behaviours of the data such as periodicity or seasonality. Due to its popularity and success for detecting point anomalies in the literature detailed, an ARIMA process is implemented to model the short-term structure. See Section 2.2.1 for a detailed description of this approach. The ARIMA model is fit over a short time window of data consisting of the latest s points to only model the local behaviour of the series. To best represent the local behaviour, the model updates at each time point where the optimal orders (p and q) and differencing (d) for the ARIMA model to minimise the Akaike information criterion (AIC) are selected. Appropriate ranges for the parameters of the ARIMA model are discussed in Section 5.5.2. Due to its simplicity and fast computational time (for models built using relatively small amounts of data) this method is favourable for implementation in a streaming context.

5.2.2 Long-Term Model

The long-term structure describes the regular persistent patterns of the data such as seasonality and periodicity. In the cyber-security setting, long-term behaviours correspond to the habitual network activity pattern of the working day. Hochenbaum et al. 2017 also note the presence of seasonal patterns in social network data. In the literature, seasonal components are regularly approximated using smoothing such as Loess in STL [Cleveland et al., 1990] or by simply averaging over sub-cycles [Hyndman and Athanasopoulos, 2018]. These ideas are combined where a smooth adaptive curve is fit to describe the long-term structure using Functional Data Analysis (FDA). FDA is chosen over Loess smoothing as regression functions can be calculated and the data can easily be weighted for improved adaptability. Additionally, FDA is less computationally intensive than Loess which is important for real time monitoring.

FDA has scarcely been applied in cyber-security despite the periodic nature of the data. FDA additionally benefits from not enforcing parametric assumptions about time effects [Ullah and Finch, 2013]. Similar to the proposed procedure, Millán-Roures et al. 2018 use FDA to detect anomalous urban water flows in water networks, particularly for contextual anomalies with promising results. This suggests FDA is a suitable model candidate for the long-term structure.

Functional Data Analysis is used to find a smooth curve $x(\tau)$ to describe discrete observations $x_t, t = 1, \ldots, n$ at discrete integer times using the following model,

$$x_t = x(t) + \epsilon_t$$

where ϵ_t models the noise. We focus on functions defined using a basis system where the curve describing the data, $x(\tau)$, is a linear combination of basis functions

expressed mathematically as:

$$x(\tau) = \sum_{k=1}^{K} a_k \phi_k(\tau) = \mathbf{a}^T \boldsymbol{\phi}(\tau), \tag{5.1}$$

where $\phi_k(\tau)$ are the K basis functions, a_k are their corresponding coefficients and τ is real values in $[0, n]$. Multiple types of basis functions could be used such as polynomial, Fourier or splines. B-splines, short for basis splines, are used as it allows for greater flexibility in the curve [Ramsay et al., 2009]. The R package *fda* [Ramsay et al., 2020] is applied to calculate the coefficients in Equation 5.1 and to produce a smooth curve.

These B-splines are defined by their range, knots and order [Ramsay et al., 2009] and are piecewise polynomials. The range defines the set of values for which the basis should be defined, the knots detail the positions for the breaks at which the polynomial curves connect and the order is one larger than the degree of the polynomials used. If there are r knots at a single point, the first $order - r$ derivatives of the spline at that knot must be continuous. The number of splines for the given knots and order is *order + number interior knots* where interior knots refers to the knots that are not at the beginning or end of the range.

To calculate the coefficients for the basis functions, Ordinary Least Squares minimisation is performed where the following is minimised:

$$\sum_{t=1}^{n} [x_t - \sum_{k=1}^{K} a_k \phi_k(t)]^2,$$

with optimal coefficients calculated as,

$$\hat{\mathbf{a}} = (\Phi^T \Phi)^{-1} \Phi \mathbf{x},$$

where $\Phi_{tk} = \phi_k(t)$ and $\mathbf{x} = (x_1, \ldots, x_n)^T$. Hence a function for the data is defined by Eq. 5.1 with $\hat{\mathbf{a}}$.

The long-term model fits a spline over the periodic signal of the data to produce a smooth curve to represent this pattern and is used as a forecast. Consider the real valued series $x_1, x_2, \ldots, x_i, \ldots$ with integer indices. This data is assumed to have seasonal frequency of length n and the series is rewritten as,

$$y_1^1, y_2^1, \ldots, y_t^j, \ldots$$

where $t = 1, \ldots, n$ is the time within a single seasonal cycle and $j = 1, 2, \ldots$ is the seasonal cycle number (the number of cycles observed). This splits the series into the respective seasonal cycles. More concretely,

$$x_i = y_{(i-1)(\mod n)+1}^{\lceil \frac{i}{n} \rceil}$$

To forecast the long-term pattern of the data, a smooth curve is fit over the weighted sum of historical data. The weighted average using all previously observed cycles up until the $c-1^{th}$ cycle is:

$$\hat{y}_t^c = \frac{\sum_{j=1}^{c-1} w_j y_t^j}{\sum_{j=1}^{c-1} w_j}, \tag{5.2}$$

where w_j is the weight for the jth cycle and $t = 1, \ldots, n$. A cubic B-spline is fit over the values $\{\hat{y}_t^c\}_{t=1\ldots n}$ to produce the smooth function $x_c(\tau)$ using Equation 5.1. This function is used to forecast the series for cycle c given the data up to cycle $c-1$. This formulation can be shown to be equivalent to fitting a spline to each cycle $\{y_t^j\}_{t=1\ldots n}$ for $j = 1, \ldots, c-1$ separately before computing a weighted average of these FDA curves.

As more recent cycles describe the current behaviour of the data better than older cycles it is sensible to weight the data accordingly. To achieve this exponential weighting is employed to the data where older samples are down-weighted by a factor λ (termed a forgetting factor) at each update of the model. The proposed approach for applying exponential weights to calculate the weighted average over the series is now detailed.

5.2.3 Long-Term Adaptive Forgetting Approach

An adaptive forgetting factor method is applied as done in Chapter 3 to implement forgetting factors. To calculate the weighted average for data up to cycle $c-1$, data for cycles $c-2$ and older are down-weighted by factor λ_c. Using this weighting scheme, the weight for data from cycle j with a total of $c-1$ cycles observed is expressed as

$$w_j^{c-1} = \lambda_c w_j^{c-2} = \prod_{l=j+2}^{c} \lambda_l \qquad j = 1, \ldots, c-2 \tag{5.3}$$

where $w_j^j = 1 \forall j$. These λ_i are estimated at the end of cycle $i-1$. Thus at cycle $c-1$, all $\lambda_1, \ldots, \lambda_c$ are known. The weighted average using data from cycles up to cycle $c-1$ is,

$$\hat{y}_t^c = \frac{\sum_{j=1}^{c-1} w_j^{c-1} y_t^j}{\sum_{j=1}^{c-1} w_j^{c-1}} \tag{5.4}$$

$$= \frac{\lambda_c \left(\sum_{j=1}^{c-2} w_j^{c-2} y_t^j \right) + y_t^{c-1}}{\lambda_c \left(\sum_{j=1}^{c-2} w_j^{c-2} \right) + 1}$$

$$= \frac{\lambda_c \hat{y}_t^{c-1} \left(\sum_{j=1}^{c-2} w_j^{c-2} \right) + y_t^{c-1}}{\lambda_c \left(\sum_{j=1}^{c-2} w_j^{c-2} \right) + 1}$$

$$= \frac{\lambda_c \hat{y}_t^{c-1} w(c-2) + y_t^{c-1}}{\lambda_c w(c-2) + 1} \tag{5.5}$$

where $w(c-2) = \sum_{j=1}^{c-2} w_j^{c-2}$, the sum of the cycle weights at cycle $c-2$. The simplification to calculate Equation 5.5 utilises $\sum_{j=1}^{c-2} w_j^{c-2} y_t^j = \hat{y}_t^{c-1} \left(\sum_{j=1}^{c-2} w_j^{c-2} \right)$, the weighted average for observations up to cycle $c-2$. It is clear from Equation 5.3 that $w(c-1) = \lambda_c w(c-2) + 1$. Thus the weighted average using this forgetting factor mechanism can be updated sequentially, only requiring the previous sum of the weights $w(c-2)$, the weighted average of historical data up to cycle $c-2$, $\hat{y}_t^{c-1}$, the forgetting factor λ_c, and the observed value from cycle $c-1$, y_t^{c-1}.

To automatically update the forgetting factor λ we use the same approach as Bodenham and Adams 2017 and use stochastic gradient descent Haykin 2002 as follows,

$$\lambda_{c+1} = \lambda_c - \alpha \frac{\partial C_c(\lambda)}{\partial \lambda} \bigg|_{\lambda = \lambda_c} \tag{5.6}$$

where c refers to the cycle number and α is the learning rate that controls how quickly the forgetting factor adjusts. Implementation of stochastic gradient descent removes the need for human input in the model after initialisation, where the data itself determines the level of forgetting performed.

Within the literature there is a lack of guidance on setting the learning rate and instead, this parameter is generally set empirically Bodenham and Adams 2017 on training data and is the approach chosen. The cost function, $C_c(\lambda)$, is to be minimised with respect to λ where the following function is used,

$$C_c(\lambda) = \frac{1}{n} \sum_{t=1}^{n} |y_t^c - \hat{y}_t^c| \tag{5.7}$$

corresponding to the mean absolute error between the true value of the series for the cth cycle and the weighted average over historical cycles up to cycle $c-1$. For the FDA model a spline is fit over this weighted average and is utilised as

a forecast for cycle c, however, in Equation 5.7, rather than utilising this spline forecast we utilise the weighted average as it results in simple sequential formula updates. Here the mean absolute error is utilised for the cost function instead of the popular MSE as it is less sensitive to outliers and is on the same scale as the data Hyndman and Koehler, 2006. The derivative of the cost function $C_c(\lambda)$ with respect to λ is,

$$\frac{\partial C_c}{\partial \lambda} = -\frac{1}{n}\sum_{t=1}^{n} \operatorname{sgn}(y_t^c - \hat{y}_t^c)\frac{\partial \hat{y}_t^c}{\partial \lambda},$$

where the derivative of $\hat{y}_t^c$ is given by,

$$\begin{aligned}\frac{\partial \hat{y}_t^c}{\partial \lambda} &= \frac{\left(\sum_{j=1}^{c-2} w_j^{c-2} y_t^j\right)}{\lambda\left(\sum_{j=1}^{c-2} w_j^{c-2}\right)+1} - \frac{\lambda\left(\sum_{j=1}^{c-2} w_j^{c-2} y_t^j\right) + y_t^{c-1}}{\left(\lambda\left(\sum_{j=1}^{c-2} w_j^{c-2}\right)+1\right)^2} \times \left(\sum_{j=1}^{c-2} w_j^{c-2}\right) \\ &= \frac{\sum_{j=1}^{c-2} w_j^{c-2} y_t^j - y_t^{c-1}\sum_{j=1}^{c-2} w_j^{c-2}}{\left(\lambda\left(\sum_{j=1}^{c-2} w_j^{c-2}\right)+1\right)^2}.\end{aligned}$$

Thus the update rule for the adaptive forgetting factor is,

$$\begin{aligned}\lambda_{c+1} &= \lambda_c + \frac{\alpha}{n}\sum_{t=1}^{n} \operatorname{sgn}(y_t^c - \hat{y}_t^c)\frac{\sum_{j=1}^{c-2} w_j^{c-2} y_t^j - y_t^{c-1}\sum_{j=1}^{c-2} w_j^{c-2}}{\left(\lambda_c\left(\sum_{j=1}^{c-2} w_j^{c-2}\right)+1\right)^2} \\ &= \lambda_c + \frac{\alpha}{n}\sum_{t=1}^{n} \operatorname{sgn}(y_t^c - \hat{y}_t^c)\frac{\hat{y}_t^{c-1} w(c-2) - y_t^{c-1} w(c-2)}{(\lambda_c w(c-2)+1)^2}. \qquad (5.8)\end{aligned}$$

An expanding window can be used for this method as sequential updating forms can be computed resulting in fixed storage requirements and computation time when run over data streams indefinitely. To update the forgetting factor only a fixed amount of information is required, namely the previous forgetting factor value, the sum of the weights $w(c-2)$ up to cycle $c-2$, and the true value of the series for cycles $c-1$ and c.

5.2.4 Applying the Long-Term Model

We now summarise how this long-term model can be implemented in practice. As we require full seasonal periods to be observed to calculate the weighted average in Equation 5.2, when applied to data streams, it is calculated at the end of each fully observed seasonal period rather than at each time resolution. Algorithm 2 summarises the update scheme of the FDA algorithm. To begin, initial forgetting factors are required for the exponential weights. In practice, we suggest setting

Algorithm 2 Long-Term Model

Input: Time series $x_1, \ldots, x_i, \ldots$ with initial length i, length of seasonal period n, learning rate α for updating the forgetting factor, initial forgetting factors $\lambda_3, \ldots, \lambda_{\lfloor \frac{i}{n} \rfloor}$

Output: Forecast of long-term structure in data

1: Calculate the weighted average over the data up to cycle $\lfloor \frac{i}{n} \rfloor - 1$ using Equation 5.2 namely $\left\{ \hat{y}_t^{\lfloor \frac{i}{n} \rfloor}, t = 1, \ldots, n \right\}$

2: Calculate the updated forgetting factor $\lambda_{\lfloor \frac{i}{n} \rfloor + 1}$ using learning rate α and Equation 5.8

3: **for** $j = i+1, i+2, \ldots$ **do**

4: **if** A full cycle of the series has been observed corresponding to $j \mod n \equiv 0$ **then**

5: Calculate the weighted average over the data up to cycle $\frac{j}{n}$ using Equation 5.2 namely $\left\{ \hat{y}_t^{\frac{j}{n}+1}, t = 1, \ldots, n \right\}$ using the sequential update from Equation 5.5

6: Fit a spline to the weighted average values, $\left\{ \hat{y}_t^{\frac{j}{n}+1}, t = 1, \ldots, n \right\}$, to produce smooth curve $x_{\frac{j}{n}}(\tau)$. The values $\left\{ x_{\frac{j}{n}}(k), k = 1, \ldots, n \right\}$ are the FDA forecast values for $\{x_{j+k}, k = 1, \ldots, n\}$.

7: Update the forgetting factor $\lambda_{\frac{j}{n}+2}$ using learning rate α with Equation 5.8

8: **end if**

9: **end for**

these values close to 1 (equivalent to unweighted). For all future applications these are set initially to 0.99 however this initial value is not of great concern as the estimate for λ adapts over the series.

To allow for sequential updates, initially the weighted average over the penultimate fully observed cycle is calculated using Equation 5.4 with weights calculated using Equation 5.3. After this initial calculation, sequential formulas can be utilised to update the weighted average over previous cycles using Equation 5.5 and update the forgetting factor using Equation 5.8. To produce FDA forecasts, a spline is fit to this average to produce smooth estimates over the cycle.

5.3 Combining Short and Long Forecasts

The short-term and long-term models have now been detailed. These two models capture different aspects of the data where the short-term model focuses on local behaviours whereas the long-term model captures the global features of the data such as seasonality within the data. These forecasts are combined as it is

known to reduce misspecification bias while increasing forecast accuracy [De Gooijer and Hyndman, 2006, Theodosiou, 2011]. Two approaches are proposed: the first approach is a decomposition method which decomposes the series into short and long-term components whereas the other approach does a simple regression combination.

5.3.1 Short-Long Decomposition (SL Decomposition)

We first propose a novel time series decomposition method suitable for data that exhibits a periodic structure. Similar to conventional time series decompositions, the series is split into two components, one representing the periodic *long-term* structure or historic structure, while the other represents the behaviour on a local scale called the *short-term* structure. This short-term behaviour also contains the local trends of the series and the deviations away from the long-term pattern. Let the time series random variables be represented by X_i for discrete time i with corresponding observed value x_i, which is decomposed into the following terms,

$$x_i = L_i + S_i + v_i,$$

where L_i is the long-term structure, S_i is the short-term structure and v_i is the random noise.

This long-term structure, L_i is modelled using the FDA model described in Section 5.2.2 and is applied to the raw series x_i to produce the forecasts of the long-term structure $\hat{L}_i$ utilising data up to the last full cycle observed and forecasts a full cycle ahead.

Once this historic view of the data has been calculated, the short-term structure is recovered through modelling the residual,

$$\tilde{S}_i = x_i - \hat{L}_i.$$

The short-term component, $\tilde{S}_i$, is modelled using an ARIMA process as detailed in Section 5.2.1 to forecast one-step ahead and is updated at every timestamp, i. This procedure is similar to forecasts made using the popular STL decomposition (discussed in Section 2.2.2) where forecasts are made on the seasonally adjusted time series using methods such as ARIMA.

The final combined forecast of the model at time i given the data up to time $i-1$ is,

$$\hat{x}_i = \hat{L}_i + \hat{S}_i, \tag{5.9}$$

where $\hat{S}_i$ is the ARIMA forecast for $\tilde{S}_i$. An advantage of this model over the standard seasonal decomposition methods is that the long-term structure component is still directly comparable to the original time series and can be used for identifying anomalies directly from the data. Both contextual and point anomalies

can be found by identifying when the data deviates from the long-term pattern L_i whereas deviations from the short-term component S_i only refer to global outliers. The order the short and long-term models are applied is important. If the short-term model is applied first, the long-term component is not directly comparable to the original series hence the detection of contextual anomalies is not possible. This proposed decomposition model is termed the *Short-Long Decomposition (SL Decomposition)*.

5.3.2 Regression Combination

We additionally propose a simple forecast combination approach. To produce more accurate forecasts the short-term (modelled by the ARIMA process) and long-term (modelled by FDA curve) forecasts are combined using linear regression which is a popular forecast combination approach [De Gooijer and Hyndman, 2006]. For this model, both FDA and ARIMA are applied directly to the raw series x_i. Let the FDA forecast for time i be $\hat{x}_i^{FDA}$ and the ARIMA forecast be $\hat{x}_i^{ARIMA}$ for x_i.

The fused forecast of the FDA and ARIMA forecasts for time i is,

$$x_i = \alpha \hat{x}_i^{ARIMA} + \beta \hat{x}_i^{FDA} + v_i$$

where v_i is the Gaussian error and α & β are the coefficients. As this method is applied in an online manner, it is desirable for the model to update over time. As recent data better exhibits the current behaviour of the data, exponential weighting is again applied. The following weighted least squares problem is solved:

$$\left(\mathbf{y} - \mathbf{X}\begin{pmatrix}\alpha\\ \beta\end{pmatrix}\right)^T \mathbf{W} \left(\mathbf{y} - \mathbf{X}\begin{pmatrix}\alpha\\ \beta\end{pmatrix}\right)$$

where the current length of the series is m, $\mathbf{y} = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}$, $\mathbf{X} = \begin{pmatrix} \hat{x}_1^{ARIMA} & \hat{x}_1^{FDA} \\ & \vdots \\ \hat{x}_m^{ARIMA} & \hat{x}_m^{FDA} \end{pmatrix}$

and

$$\mathbf{W} = \begin{pmatrix} \gamma^{n-1} & & & & \\ & \gamma^{n-2} & & & \\ & & \ddots & & \\ & & & \gamma & \\ & & & & 1 \end{pmatrix}, \tag{5.10}$$

for a given fixed forgetting factor, $\gamma \in (0, 1]$. This forgetting factor can be set to to maximise the forecast performance during a tuning period in the data. Values within $[0.8, 0.99]$ are suggested. Sequential regression formulas for this weighted regression have been derived by [Riddle-Workman et al., 2018] and are applied

here to maintain fixed computational requirements. Forecasts from this model are termed *regression combination*. The estimated coefficients of the regression are investigated further in Appendix C.2.

5.4 Combined Anomaly Detection Procedures

The primary aim of this chapter is to develop procedures for identifying both point and contextual anomalies. Non-extreme shifts in behaviour from what is "normal" such as the highlighted contextual anomaly in Figure 5.1, would not be identified by point anomaly procedures, which is the focus of most existing approaches. Point anomaly procedures can only detect extreme global outliers such as the point anomaly highlighted in Figure 5.1. Both types of anomalies are important hence we propose anomaly detection procedures to reflect this. To achieve this aim, results are combined from the short-term and long-term models to ensure both types of anomalies are detected. For clarity, the general proposed model framework is illustrated in Figure 5.2.

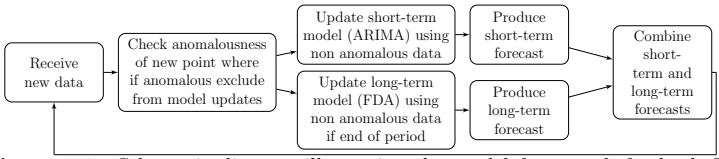


Figure 5.2: Schematic diagram illustrating the model framework for both SL Decomposition and FDARIMA.

In this section two anomaly detection procedures are detailed. The first approach, SLD Anomaly Detection, utilises the long-term and full series forecasts from SL Decomposition to identify anomalies in the series. FDARIMA Anomaly Detection is additionally proposed which detects anomalies from both the FDA model and the ARIMA model applied to the raw series. Both procedures utilise conformal prediction to detect anomalies.

5.4.1 Conformal Prediction p-values

Rather than using crude measures for anomaly detection as often done in the literature, prediction intervals are used instead. Prediction intervals are a popular approach for anomaly detection e.g. Oliveira and Meira [2006] and Hill and Minsker [2010] to measure how anomalous a newly observed value is compared to its model forecast. Prediction intervals are commonly calculated using normality assumptions which are often violated in network connections data [Ullah et al. 2021] and other domains. Due to these unrealistic assumptions, the Gaussian prediction intervals produced are often very large making them non informative

for anomaly detection. A non-parametric approach can instead be implemented to avoid making untenable distributional assumptions using Conformal Prediction (CP). Implementation of CP for anomaly detection is not a new concept e.g. Laxhammar and Falkman [2011] and Ishimtsev et al. [2017], where similarly Dashevskiy and Luo [2008] apply CP to time series data for network traffic demand prediction.

Conformal Prediction (CP) prediction intervals are implemented to characterise the uncertainty of the forecast. This approach does not make unrealistic distributional assumptions about the data and can be applied to both the short and long-term forecasts due to its flexibility. From these prediction intervals, p-values can be calculated to characterise the anomaly scores of newly observed points and anomalies are detected at the ϵ level if these p-values fall below this value.

Following the notation of Vovk et al. [2005], let a sequence of examples be represented as $z_1, z_2, \ldots, z_{n-1}$ and a new example by z_n where each example z_i is formed of an object-label pair (x_i, y_i). CP produces a p-value detailing the probability of observing something at least as extreme as z_n using a *nonconformity* measure A, a real valued function that describes the dissimilarity between the examples [Shafer and Vovk, 2007]. The results for CP are valid for any nonconformity measure however the size of the prediction interval depends on this nonconformity measure [Vovk et al., 2005]. A small p-value ($\leq \epsilon$) suggests example z_n is statistically different from the *bag* or *multi-set* of examples $\wr z_1, \ldots, z_{n-1} \int$ (a mathematical object which uses the same concepts as a set but allows for repetition of the elements).

The p-value of a new example z_n given a bag of observations $\wr z_1, \ldots, z_{n-1} \int$ is computed using the empirical distribution of the nonconformity measures in the following way,

$$p(z_n) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\alpha_i \geq \alpha_n}, \tag{5.11}$$

where $\alpha_i = A(\wr z_1, \ldots, z_n \int \setminus \wr z_i \int, z_i)$. As the produced p-values are within $[\frac{1}{n}, 1]$, to identify anomalies at an ϵ significance level, at least $\frac{1}{\epsilon}$ points are needed for comparison.

For the prescribed problem, let each example be represented as,

$$z_i = (\hat{x}_i, x_i)$$

where x_i is the observed value of the time series and $\hat{x}_i$ is its corresponding forecast. The absolute forecast error is used as the nonconformity measure given by,

$$\alpha_i = |x_i - \hat{x}_i|. \tag{5.12}$$

As the procedure is intended to operate indefinitely, it is necessary to use a sliding window to maintain fixed storage requirements. An initial burn-in period is required for a sliding window which for application to data streams does not present

a problem, however as the real data examples are of a fixed size, an expanding window is implemented.

5.4.2 SLD Anomaly Detection

Algorithm 3 SL Decomposition Anomaly Detection Algorithm

Input: Initial series $X_1, \ldots, X_i$, number of data points in each cycle n, ARIMA window size s, learning rate α, significance level ϵ, initial forgetting factors $\lambda_3, \ldots, \lambda_{\lfloor \frac{i}{n} \rfloor}$

Output: Sequence of anomalous time points A

1: Initialise FDA weighted average and forgetting factors using data up to time $l = i - (i \mod n)$ and produce forecasts for times $l+1, l+2, \ldots, l+n$ namely $\hat{L}_{l+1}, \ldots, \hat{L}_{l+n}$ as described in Algorithm 2.
2: **for** $j = i+1, i+2, \ldots$ **do**
3: Fit ARIMA model to long-term forecast errors $x_{j-1-s} - \hat{L}_{j-1-s}, \ldots, x_{j-1} - \hat{L}_{j-1}$ to produce forecast for time j, $\hat{S}_j$ where anomalous times A are excluded.
4: Combine predictions to get reconstructed series $\hat{x}_j = \hat{L}_j + \hat{S}_j$.
5: Calculate p-values using CP on absolute residuals of $\hat{L}_j$ and $\hat{x}_j$ from x_j to give p_j^{Long} and $p_j^{Reconstruct}$ respectively.
6: Combine p-values using Fisher Product Test Statistic $S_F = -2(log(p_j^{Long}) + log(p_j^{Reconstruct}))$.
7: **if** $S_F > \beta$ where β is the $(1-\epsilon)$th quantile of χ_4^2 **then**
8: X_j is anomalous and add j to A.
9: **end if**
10: **if** $j \mod n \equiv 0$ **then**
11: Calculate the weighted average over previous cycles using data up to and including time j excluding anomalous times A (these points get given weight of 0).
12: Fit a smooth spline over the weighted average values to produce the long-term forecast for the future cycle $\hat{L}_{j+1}, \ldots, \hat{L}_{j+n}$.
13: Update FDA forgetting factor λ using learning rate α and Equation 5.8.
14: **end if**
15: **end for**

To detect anomalies using the SL Decomposition method two instances of the proposed conformal prediction framework are implemented to detect both point and contextual anomalies. CP is applied to the long-term structure $\hat{L}_i$ forecast and additionally on the combined forecast $\hat{x}_i$ detailed in Equation 5.9. By calculating p-values for the long-term structure, L_i, contextual anomalies can be identified as departures from the historical pattern. To create a shared anomaly signal, these p-values are combined. A range of p-value combination techniques are available

as explored by Heard and Rubin-Delanchy [2018] where here the Fisher's Product Test Statistic [Fisher, 1925], a popular p-value combination technique, is used. As noted by Heard and Rubin-Delanchy [2018], the Fisher's statistic is appropriate here as it is well suited to positive-valued data where the data is larger under the alternative hypothesis. Given k p-values, $p_1, \ldots, p_k$, this method produces a score,

$$S_F = -2\sum_{i=1}^{k} log(p_i). \tag{5.13}$$

Under the assumption the p-values are independent, this follows a chi-squared distribution with $2k$ degrees of freedom. When applied to the data, empirically the p-values produced from CP are approximately uniform and additionally, the Fisher product test statistic follows this chi-squared distribution with 4 degrees of freedom which is demonstrated in Appendix C.3.1. Anomalies are detected at the ϵ level when the p-value falls below this value.

As anomalous points are not representative of the true underlying patterns, when included within the model, empirically they lead to poorer forecasts and anomaly detection performance. To combat this, if a point is anomalous with respect to the combined anomaly score at a certain significance level, ϵ, this point is not included in the short and long-term models when updating. The improvement in forecast performance is explored when anomalies are removed in Appendix C.4. This anomaly detection algorithm is termed *SL Decomposition* or *SLD* and is detailed in Algorithm 3.

5.4.3 FDARIMA Anomaly Detection

We additionally propose a more simple approach for detecting anomalies. CP is applied to the forecasts from FDA and ARIMA calculated directly on the raw series x_i to produce two p-values. Let the FDA forecast for time i be $\hat{x}_i^{FDA}$ and the ARIMA forecast be $\hat{x}_i^{ARIMA}$ using data up to time $i-1$. To calculate the p-values, the same nonconformity measure given in Equation 5.12 is used and is applied to $\hat{x}_i^{FDA}$ and $\hat{x}_i^{ARIMA}$ to give p_i^{FDA} and p_i^{ARIMA} respectively. The uniformity of the p-values is explored in Appendix C.3.2 and it is again seen that they are uniform. These p-values are combined using the Fishers Product test statistic in Equation 5.13 to give a single anomaly score as follows,

$$S_F = -2(log(p_i^{ARIMA}) + log(p_i^{FDA})).$$

If this combined anomaly score is anomalous at a certain significance level, ϵ, this point is not included in the model when it is updated as it is not representative of the usual expected behaviour. This anomaly detection procedure is termed *FDARIMA*.

The key difference between the FDARIMA and SLD anomaly detectors is how the short-term p-values are detected. For FDARIMA, conformal prediction is

applied to the ARIMA forecast for the original series x_i, whereas for the SLD approach, p-values are calculated using the reconstructed series. The FDARIMA approach is implemented in the same way as SLD in Algorithm 3 however the ARIMA model is applied to the input series directly with conformal p-values calculated for these forecasts.

5.4.4 Scalability for Online Settings

The methods proposed here can suitably run over data streams with fixed storage requirements and computation time through the usage of sliding windows and sequential updates. A sliding window for the conformal predictor can be applied where the size of this window can be set based on computational restrictions. A small window for the ARIMA model can allow for updates in real time and is more suitable for modelling short-term behaviours of the series. From empirical experiments, small windows for the ARIMA model led to better forecasts as shown in Appendix C.1. Thus the methods proposed are robust to high frequency data where each update can be computed rapidly for real time analysis.

5.5 Simulated Data Results

5.5.1 Data Generation Procedure

Real data sets are often unlabelled or are small, making it challenging to assess anomaly detection performance in practice, particularly in cyber-security. A time series that exhibits a periodic structure with point and contextual anomalies added throughout is simulated. The series is split into a short-term and a long-term process with the addition of a noise component and is illustrated using the equation below:

$$X_i = L_i + S_i + \epsilon_i, \quad \text{for } i = 1, \ldots, N, \tag{5.14}$$

where L_i is the long-term process, S_i is the short-term process and ϵ_i is the noise. Here $\epsilon_i \sim N(0, 5)$ and the long-term structure is generated using the following equation,

$$L_i = 100 \left(\sin \left(2\pi \frac{i-1}{1440} - \frac{\pi}{2} \right) + 1 \right)$$

corresponding to a series with a period of 1440 points (analogous to minute data with a daily seasonal period). The short-term process is simulated using an ARMA(1,1) process with both coefficients set to 0.05. An ARMA process is used here instead of ARIMA to prevent large trends in the data.

Point anomalies are generated at random throughout the time period and the magnitude of the anomalies are randomly sampled from a $N(200, 20)$ distribution. This value is then added or subtracted from X_i at the anomalous time point.

Contextual anomalies are inserted throughout the time period and are generated again using a sine curve. For a contextual anomaly with a period of p points starting at time b with amplitude a, the process between this period is:

$$X_i^{(anomalous)} = X_i \pm a\left(\sin\left(2\pi\frac{i-b}{p} - \frac{\pi}{2}\right) + 1\right) + r_i, \quad i \in [b, b+p-1]$$

where $r_i \sim N(0, 5)$. To ensure X_i is positive (as the data corresponds to counts), if the generated process X_i in Equation 5.14 is not positive for all i it is transformed by,

$$\tilde{X}_i = X_i - \min_{i=1,\ldots,N}(X_i).$$

For future experiments, 100 data sets have been generated each over a 20 cycle period with a seasonal frequency of 1440 points where the first 10 cycles are used as an initialisation period for the models. This frequency is selected as it corresponds to minute data with a periodic frequency of 1 day. Anomalies have been added to the final 10 cycles of the data. More specifically, 80 point anomalies have been included at random. Additionally, 4 contextual anomalies have been included with different durations.

5.5.2 Model Specification

There are several parameters and formulation choices in the methods proposed in Section 5.2 that must be specified. To begin the short-term model which is modelled by an ARIMA process is investigated. The *autoARIMA* function in R [Hyndman and Khandakar, 2008] is implemented which automatically chooses the optimal parameters to model the data, however ranges for the orders (p and q) and differencing (d) parameters must be selected. In all applications, a maximum order of 10 for both p and q are applied. For the differencing parameter, d, a maximum value of 5 is applied. A small differencing value is used as the short-term model does not need to capture long-term seasonal patterns. Increasing the orders and differencing parameters causes longer computation times, where the values utilised give good forecasting performance which is explored further in Section 5.5.3. Finally, we proposed implementing ARIMA using a sliding window for a fixed computational burden at each update. The size of this sliding window has been investigated further in Appendix C.1 on the simulated data described in Section 5.5.1 where the optimal window size is 30 points however the difference in performance between sliding window sizes is marginal. In all future implementations a window of 60 points is used.

For the long-term model, cubic b-splines are fit to the weighted average over the past cycles. To fit these splines the number of basis functions, K used must be selected. Alternatively, break points can be set throughout the cycle period $t = 1, \ldots, n$. Here the number of basis functions is specified where these break points are evenly spread throughout the period. Smaller K results in less computation

time, however, less flexibility in the model. Alternatively larger K may result in overfitting. In all future implementations $K = 102$ is utilised corresponding to 100 break points ($nbreak = K - order + 2$ where for cubic splines the order is 4).

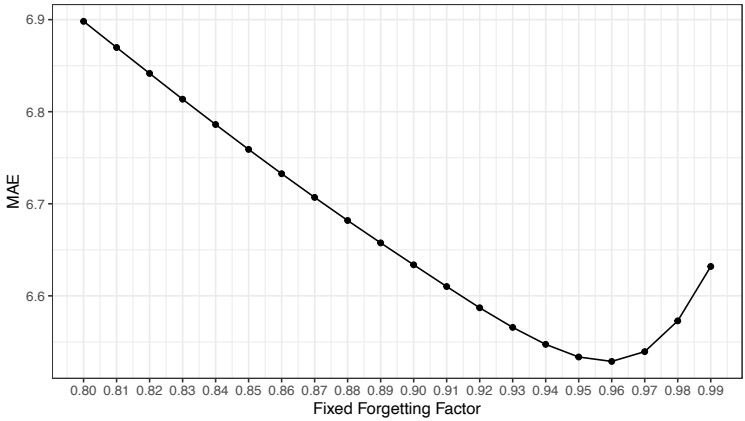


Figure 5.3: MAE of regression combination forecast for varying fixed forgetting factors over 100 simulations.

The regression combination forecast approach in Section 5.3.2 requires a fixed forgetting factor to be set which can be set empirically to maximise the forecast performance ideally on an initial tuning period in the data. For the simulated data described in Section 5.5.1, Figure 5.3 displays the MAE for the regression combination forecast for varying forgetting factors over 100 simulations with 20 cycles. It can be seen that there is a minimum at 0.96 and is the value implemented in future experiments.

5.5.3 Forecast Performance

Before assessing the anomaly detection performance, the forecast ability of the proposed approaches is investigated using the measures described in Section 2.4.2. Table 5.1 summarises the forecast performance where the ARIMA and FDA forecasts correspond to the results from application to the raw time series individually.

In Table 5.1 the naive one-step ahead forecast approach has the smallest MAE and MASE followed by the ARIMA and combination approaches. It is important to note that the FDA procedure forecasts a full cycle ahead rather than a single step, hence it is expected to have poorer forecast performance. Although the forecast combination approaches have poorer performance than ARIMA alone,

these additionally model the long-term behaviours which is useful for contextual anomaly detection.

Table 5.1: Forecast performance comparison of proposed methods over 100 simulated data sets generated as described in Section 5.5.1

	MAE	AE SD	MASE	ASE SD
ARIMA	6.512	14.933	1.179	2.695
FDA	14.821	21.974	2.769	3.980
Regression Combination	6.633	14.283	1.243	2.613
SL Decomposition	7.315	17.978	1.334	3.298
Naive	5.949	15.033	1.000	0.000

5.5.4 Comparison Anomaly Detection Methods

Only methods that have existing R packages or Python libraries are implemented. In addition comparison to well established approaches is performed. To that end, we compare the proposed method to ARIMA alone using the *tsoutlier* R Package [López-de Lacalle, 2019]. We also compare the proposed procedures to the classic STL decomposition using the *anomalize* R package [Dancho and Vaughan, 2020]. Other comparison methods include the Twitter anomaly detector [Kejariwal, 2015, Hochenbaum et al., 2017], SmartSifter [Yamanishi et al., 2004] and TSSD-EWMA [Raza et al., 2015]. For each method, default parameters are used. Table 5.2 provides a summary of the key features of the comparison methods to the proposed anomaly detection approaches FDARIMA and SL Decomposition. For a full description of these methods see Appendix C.5

Table 5.2: Summary of the main characteristics of each anomaly detection method in terms of being online and their ability to detect point and contextual anomalies.

	Online	Point	Contextual
Twitter	✗	✓	✓
Anomalize	✗	✓	✗
tsoutliers	✗	✓	✗
TSSD-EWMA	✓	✓	✗
Smart Sifter	✓	✓	✗
FDARIMA	✓	✓	✓
SL Decomposition	✓	✓	✓

5.5.5 Anomaly Detection Performance

To evaluate the anomaly detection performance of the proposed approaches, SLD and FDARIMA, the simulated data sets described in Section 5.5.1 are utilised. The performance of the anomaly detection procedures is evaluated using the F1 measure, recall and precision where these measures are additionally split into the two types of anomalies. Only the performance over the last 10 cycles is assessed which contains both point and contextual anomalies. A significance level of $\epsilon = 0.005$ is used, corresponding to approximately 7.2 alerts per cycle. As the p-values are well calibrated, this value is set by the analyst based on appropriate numbers of anomalies per cycle. The results of these experiments are displayed in Table 5.3.

Table 5.3: Anomaly detection performance comparison with competitive procedures on 100 simulated data sets with 20 cycles generated as described in Section 5.5.1. Best performer bold, second best bold dark grey and third in bold light grey. As well as having the overall performance, it additionally measures the performance of detecting point and contextual anomalies separately.

		Batch			Online			
	Measure	Twitter	Anomalize	tsoutliers	TSSD-EWMA	Smart Sifter	FDARIMA	SLD
Overall	F1	**0.730**	0.262	0.166	0.024	0.153	**0.589**	**0.609**
	Recall	**0.576**	0.156	0.092	0.012	0.089	**0.435**	**0.462**
	Precision	**1.000**	**0.990**	0.838	0.571	0.555	**0.937**	0.909
Point	F1	**1.000**	**0.992**	**0.905**	0.222	0.714	0.863	0.803
	Recall	**1.000**	**1.000**	0.995	0.139	0.999	**1.000**	**1.000**
	Precision	**0.999**	**0.985**	**0.833**	0.568	0.555	0.767	0.682
Context	F1	**0.696**	0.129	0.008	0.000	0.000	**0.533**	**0.557**
	Recall	**0.534**	0.074	0.004	0.000	0.000	**0.380**	**0.410**
	Precision	**1.000**	0.807	0.146	0.013	0.000	**0.922**	**0.890**

It is clear the Twitter algorithm has the best performance overall, particularly for precision, and performs well for both point and contextual anomalies. The Twitter algorithm is however a **batch** method and is not suitable for online analysis, requiring all data to be stored. It is important to note that batch methods are expected to have superior performance to online approaches as they can have multiple passes over the data, hence it is not a fair or direct comparison to online methods and is instead used as a benchmark.

The proposed approaches are superior to the comparable online methods (TSSD-EWMA and Smart Sifter) in all aspects, particularly contextual anomaly detection. Very few of the competitive approaches have good performance detecting contextual anomalies.

The two proposed detection procedures have very similar detection performance, however, SLD has higher recall whereas FDARIMA has higher precision. Poorer recall for contextual anomalies is attributed to the set of true anomalies including the build-up and build-down of the contextual periods. These build-up

and down periods may not significantly differ from the usual pattern, hence are undetected.

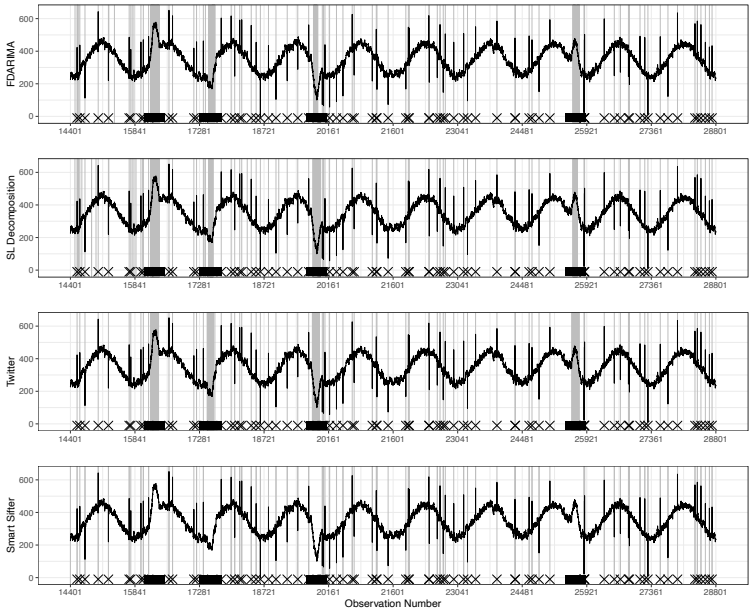


Figure 5.4: A single simulated anomalous time series as described in 5.5.1. Detected anomalies by FDARIMA, SL Decomposition, Twitter and Smart Sifter models identified with vertical grey lines and are compared to the positions of the true anomalous points marked by crosses for point anomalies and solid rectangles for contextual anomaly periods. The series shows the final 10 cycles of the simulated series.

Figure 5.4 compares the detection of FDARIMA and SL Decomposition, to the best batch and online methods Twitter and Smart Sifter. Similar figures for Anomalize, tsoutliers and TSSD-EWMA are displayed in Figure C.4 in Appendix C. From Figure 5.4 it is clear that only SL Decomposition, FDARIMA and the Twitter algorithms can detect contextual anomalies whereas Smart Sifter is better suited to only finding point anomalies.

From Table 5.3 and Figure 5.4 it is clear the proposed procedures are capable of detecting both point and contextual anomalies with similar performance to the Twitter method, a batch approach that is used as a benchmark for the online approaches. We now investigate the robustness of the proposed methods on real

data.

The Bayesian adaptive estimation procedure in Chapter 3 is another suitable method for monitoring the local behaviours of a time series. As this approach aims to estimate the current distribution parameters of the data, forecasts can be calculated using the predictive posterior distribution for data from exponential families of distributions. In Appendix C.6 the forecast and anomaly detection performance when BFF is used instead of ARIMA for the short-term model is performed, however, ARIMA showed more favourable performance.

5.6 Real Data Example

The Numenta Anomaly Benchmark (NAB)[1] [Lavin and Ahmad, 2015] contains 58 data streams of varying length with human identified anomalies. The data comes from a variety of domains including Amazon Web Services, advertising, road traffic data and Twitter mentions.

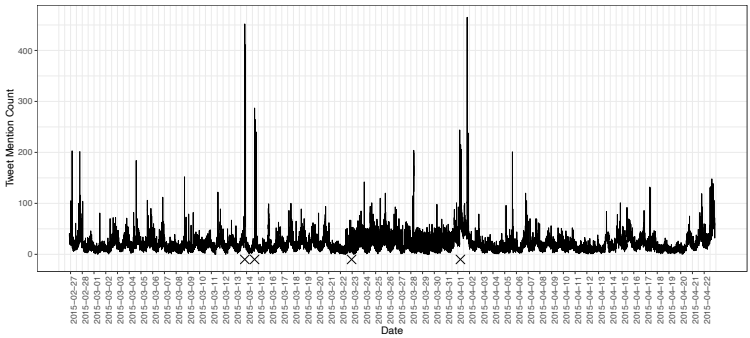


Figure 5.5: Full Twitter mention count for Google between 27th February 2015 and 22 April 2015 at 5 minute intervals. The anomaly label locations are displayed with crosses.

Here attention is restricted to the Twitter mention volume for Google. This data is collected at 5 minute intervals over 55 days and counts the number of tweets that mention Google. Figure 5.5 displays the full data set. From this figure, it is clear there exists a daily periodic nature with a peak towards the end of the day. Thus this data is suitable for the proposed approach as a persistent periodic pattern exists. Unfortunately only point anomaly labels are identified in the data hence contextual anomaly detection performance cannot be evaluated on this data set.

[1] https://github.com/numenta/NAB

Table 5.4: Anomaly detection performance comparison to competitive approaches on tweet mention data for Google. Best performer in bold, second best bold dark grey and third in bold light grey.

	Batch			Online			
Measure	Twitter	Anomalize	tsoutliers	TSSD-EWMA	Smart Sifter	FDARIMA	SL Decomposition
F1	0.014	0.015	0.016	0.000	**0.222**	**0.020**	0.018
Recall	0.500	**0.750**	**0.750**	0.000	0.500	**0.750**	**0.750**
Precision	0.007	0.008	0.008	0.000	**0.143**	**0.010**	0.009

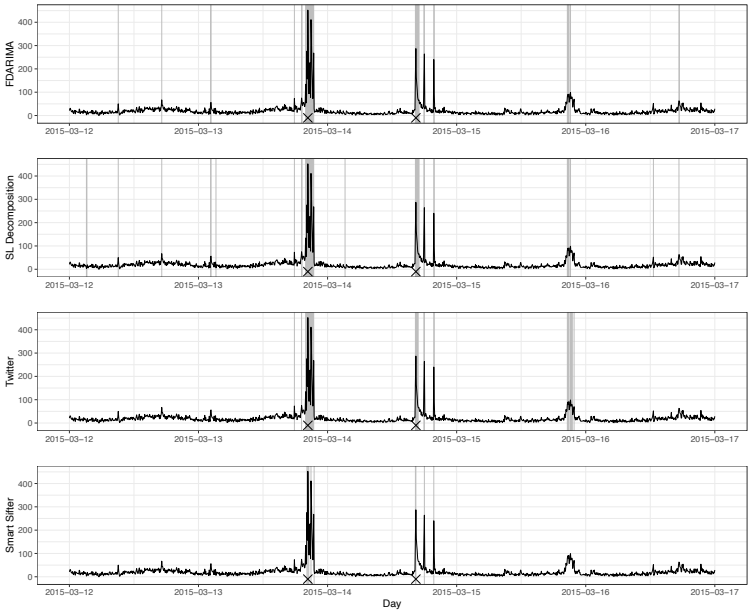


Figure 5.6: Comparison of detected anomaly locations for FDARIMA, SL Decomposition, Twitter and Smart Sifter methods where true anomalous locations are marked with a cross for the Google Twitter mention data between 12th March 2015-16th March 2015. Anomalies detected by the algorithms are identified with grey vertical lines.

There exist four true anomalies in the data which is a small number given the length of the series. This results in low precision for many of the procedures as they flag many more anomalies than this (Table 5.4). The majority of methods have a recall of 0.75 with varying precision. Surprisingly for this data, Smart Sifter

has the best performance according to F1 followed by the proposed procedures. Smart Sifter did not have as favourable performance on simulated data as by formulation, it is only capable of detecting point anomalies. As the proposed procedure outperforms the batch approaches, this example showcases the robust performance of our methods on numerous data sets.

Figure 5.6 displays the anomalies identified by FDARIMA, SL Decomposition, Twitter and Smart Sifter algorithms between 12th March 2015 to 16th March 2015. Similar plots for Anomalize, tsoutliers and TSSD-EWMA are displayed in Figure C.5 in Appendix C for reference however from Table 5.4 it is clear these methods do not have favourable performance. All algorithms in Figure 5.6 identify the unusual activity on the 13th and 14th. As Smart Sifter is a point anomaly detector, it is unable to detect the contextual anomaly on the 15th which is not extreme globally. Although only two anomalous labels are given in this period, it is clear that numerous unusual activities exist which are not labelled. Visualising the results is highly important for assessing the performance, particularly for unlabelled cases as the anomalies identified by the procedures seem sensible.

5.7 Discussion

In this chapter two anomaly detection algorithms are proposed, namely FDARIMA and SLD, for identifying both point and contextual anomalies in time series data that exhibit a regular pattern. These algorithms combine information about the long-term structure, modelling the periodic nature of the data and the short-term structure describing the local behaviour. For the long-term structure, an adaptive Functional Data Analysis model is proposed which utilises adaptive weights. The short-term structure is modelled using an ARIMA process. The models implemented in this chapter use an adaptive framework and can run autonomously after initialisation.

From both simulated and real data results, it is evident that the two proposed detection procedures, FDARIMA and SLD, can identify both point and contextual anomalies and both outperform the competitive online algorithms with comparable performance to the best batch methods. On simulated data, FDARIMA boasts higher precision whereas SLD has improved recall. Due to the similarity in anomalies produced by these methods, both are equally suitable for application, however, the choice of algorithm should be based on the aim of the application (higher recall or precision). Additionally, the suitability of these methods for real data has been shown. Further application of the proposed anomaly detection procedure to cyber-security data is explored in Chapter 8.

Chapter 6

Multi-Type Relational Clustering

Cyber-security data collects information about the connections of nodes in the network (e.g. computers and users). These connections can be represented by binary adjacency matrices describing the relations between these entities. Additionally, each entity may have several relations relating to it. The focus in this chapter is on clustering network data to group entities with similar behaviour for situational awareness and understanding of the relationships within the network rather than finding intruders or abnormalities in the data. This work gives further details of the work published in Riddle-Workman et al. 2021.

Clustering, a type of unsupervised learning, aims to partition data from a single data type into groups. Many data sets such as document-word matrices benefit from clustering both the rows and columns of a matrix, known as bi-clustering. In the case where there are multiple views of these matrices, multi-view clustering aims to find a single clustering over these matrices (all matrices have the same dimensions referring to the same entities). Multi-type relational clustering goes a step further clustering over multiple data matrices that either have common rows or columns, additionally incorporating the interrelationships between the rows and the columns of different matrices and is an extension of multi-view clustering. This clustering problem is the focus of this chapter. We are concerned with relational type data where inter-type relationships correspond to the relations between data objects of different types and intra-type relationships refer to relations between data objects of the same type. By jointly clustering over multiple relational matrices, the clustering accuracy can be improved over clustering a single data matrix Kumar et al. 2011.

Cyber-security data lends itself naturally to be represented using relational matrices to describe the interactions between users, computers and network ports within an enterprise computer network. Clustering these different node types into groups of nodes performing similar activities is useful for improved situational awareness for network administrators and could be used for the detection of intruders within the network. A challenge often associated with cyber-security data is the size of the networks and the high volume of events, requiring the methods

to be scalable. Additionally, as the network is continuously changing, methods applied must be computationally fast to allow for cluster updates. A large majority of network graph clustering falls under *community detection* where nodes are grouped such that there is a high density of edges within each cluster with few between clusters. Alternatively, in this chapter the aim is to cluster nodes based on similar activity within the graph, where a high density of edges between clusters may exist. This distinction is particularly important when considering appropriate comparison methods and performance measures.

Matrix factorisation is a popular approach for dealing with multi-relational clustering where data matrices are decomposed into a number of components. This method aims to minimise the difference between the input matrix and the lower rank decomposition of the data. The proposed clustering method, Simple NMTF, is based on this approach.

This chapter is structured as follows. Section 6.1 provides a review of relevant literature for the described problem. In Section 6.2 the proposed multi-type clustering approach, Simple NMTF, is outlined including a weighted extension for improved interpretability. As multi-type relational clustering has not been extensively studied, methods for data generation, initialisation and cluster validation are lacking in this area. We propose methods to fill these gaps. A method for generating multi-relational graph simulations is described in Section 6.3, the novel cluster validation measure in Section 6.4 and the cluster initialisation procedure in Section 6.5. The performance of the proposed clustering methods are compared against competitive approaches on simulated and a real data set in Section 6.6.

6.1 Background and Relevant Literature

Non-Negative Matrix Factorisation (NMF) is used extensively for decomposing non-negative matrices into two lower rank non-negative matrices where under orthogonality constraints, it is equivalent to k-means clustering [Ding et al., 2005] with the following interpretation; the first matrix represents the cluster centroids and the second acts as a cluster indicator matrix. NMF has been applied to a variety of areas such as text mining [Pauca et al., 2004] and recommender systems [Xin Luo et al., 2014] and was popularised by [Lee and Seung, 2001] who developed a multiplicative update rule.

NMF was extended by [Long et al., 2005] to Non-Negative Matrix Tri-Factorisation (NMTF) for bi-clustering. They utilise the duality of the rows and columns, clustering both simultaneously. In the following chapter, $||.||_F^2$ refers to the squared Frobenius norm and $||.||^2$ refers to the squared L^2 norm. NMTF factorises input non-negative matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ into three non-negative matrices by minimising the objective,

$$J = ||\mathbf{X} - \mathbf{F}\mathbf{S}\mathbf{G}^T||_F^2 \; s.t. \; \mathbf{F} \geq 0, \; \mathbf{S} \geq 0 \; \& \; \mathbf{G} \geq 0 \tag{6.1}$$

where $\mathbf{F} \in \mathbb{R}^{n\times k}$, $\mathbf{S} \in \mathbb{R}^{k\times l}$ and $\mathbf{G} \in \mathbb{R}^{m\times l}$. Generally $k \ll n$ and $l \ll m$. The matrices $\mathbf{F}$ and $\mathbf{G}$ contain the row and column association to their clusters respectively and $\mathbf{S}$ is a compact representation of $\mathbf{X}$ [Long et al., 2005]. This $\mathbf{S}$ matrix is needed to absorb the different scales of $\mathbf{X}$, $\mathbf{F}$ and $\mathbf{G}$ [Ding et al., 2006]. These matrices are found through performing either: multiplicative, alternating least squares, projected gradients or coordinate descent updates, the details of which are given in [Čopar et al., 2019]. The following multiplicative updates are the most popular in literature,

$$\mathbf{F}^{ij} = \mathbf{F}^{ij}\left[\frac{\left(\mathbf{X}\mathbf{G}\mathbf{S}^T\right)^{ij}}{\left(\mathbf{F}\mathbf{S}\mathbf{G}^T\mathbf{G}\mathbf{S}^T\right)^{ij}}\right]$$
$$\mathbf{G}^{ij} = \mathbf{G}^{ij}\left[\frac{\left(\mathbf{X}^T\mathbf{F}\mathbf{S}\right)^{ij}}{\left(\mathbf{G}\mathbf{S}^T\mathbf{F}^T\mathbf{F}\mathbf{S}\right)^{ij}}\right]$$
$$\mathbf{S}^{ij} = \mathbf{S}^{ij}\left[\frac{\left(\mathbf{F}^T\mathbf{X}\mathbf{G}\right)^{ij}}{\left(\mathbf{F}^T\mathbf{F}\mathbf{S}\mathbf{G}^T\mathbf{G}\right)^{ij}}\right].$$

The majority of updating mechanisms for NMTF are often slow due to these large matrix multiplication updates, rendering them infeasible for application to large data matrices. This procedure also requires an additional post-processing step to obtain the cluster assignment of the rows and columns that may lead to non-unique solutions [Wang et al., 2011c].

As suggested by [Wang et al., 2011c], to avoid the slow computation of NMTF using multiplicative updates and their lack of interpretability, the matrices $\mathbf{F}$ and $\mathbf{G}$ are instead indicator matrices. By enforcing this constraint, they are orthogonal allowing for clustering interpretations where $\mathbf{G}$ and $\mathbf{F}$ contain the cluster assignments of the rows and columns of $\mathbf{X}$ respectively. [Wang et al., 2011c] propose *Fast NMTF* which minimises the following objective function:

$$J = ||\mathbf{X} - \mathbf{F}\mathbf{S}\mathbf{G}^T||_F^2 \;\; s.t. \; \mathbf{F} \in \Psi^{n\times k}, \; \mathbf{S} \in \mathbb{R}^{k\times l} \;\&\; \mathbf{G} \in \Psi^{m\times l}, \tag{6.2}$$

where Ψ is the set of all cluster indicator matrices. The solution is found by alternating between solving for each matrix. First keeping both $\mathbf{F}$ and $\mathbf{G}$ fixed and minimise J w.r.t. $\mathbf{S}$ (by taking derivative and setting to 0) giving,

$$\mathbf{S} = (\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{X}\mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1}. \tag{6.3}$$

Keeping $\mathbf{F}$ and $\mathbf{S}$ fixed, J is minimised by $\mathbf{G}$, an indicator matrix where $\mathbf{G}$ is such that,

$$\mathbf{G}^{ij} = \begin{cases} 1 & if \; j = \text{argmin}_k ||\mathbf{X}^{.i} - (\mathbf{F}\mathbf{S})^{.k}||^2 \\ 0 & \text{otherwise} \end{cases}. \tag{6.4}$$

A similar expression can be calculated for $\mathbf{F}$. As the updates do not require large

matrix multiplications, the clustering is fast to compute. Additionally, the algorithm converges to a local minimum due to the nature of alternating optimisation. An additional benefit of this method is its interpretability. $\mathbf{SG}^T$ and $\mathbf{FS}$ contain the row and column clusters centres respectively (represented by their mean).

Recently there has been an influx of research performed on Multi-View Clustering where clustering is performed over multiple "views" of the data (all with the same dimensions referring to the same entities). Liu et al. 2013 applied Non-Negative Matrix Factorisation to address this problem where the clustering for each view is regularised towards a consensus clustering. Alternatively Yan et al. 2020 apply an adaptive k-means approach for multi-view clustering. Common methods applied for multi-view clustering include k-means, spectral clustering, matrix factorisation, tensor decomposition and deep learning [Ye et al. 2018].

In contrast, this chapter focuses on the problem of multi-type relational clustering. Multi-view clustering is a subset of multi-type clustering where the latter clusters over matrices that either have common rows and/or columns (i.e. the entities that the rows and columns correspond to do not need to be the same in all matrices unlike the case for multi-view clustering). Banerjee et al. 2007 propose using a relational graph model to jointly cluster over multiple entity classes. A probabilistic approach has also been implemented that allows for both hard and soft clustering of the data by Long et al., 2007. Non-Negative Matrix Factorisation (NMF) is also a popular approach for multi-type clustering e.g. Liu and Wang 2018, Ni et al. 2015, Pei et al. 2015 and is the method used. Zitnik and Zupan 2015 propose Data Fusion Matrix Factorisation (DFMF) for gene prediction which uses a penalised matrix tri-factorisation to cluster both intra and inter-type relations. Despite the success of these NMF methods, performing such decompositions is slow due to large matrix computations. In many applications, particularly cyber-security, fast computation of clustering for situational awareness of the network is desirable for timely decision making.

6.1.1 Competitive Approaches

Three competitive multi-type clustering approaches are now introduced. Pei et al. 2015 simultaneously cluster over multiple networks using a method called *R-NMTF* for clustering in social media networks imposing orthogonality constraints on the cluster matrices and manifold regularisation on intra-type relations. The objective function for R-NMTF is given by,

$$\begin{aligned} J =& ||\mathbf{A}_{uu} - \mathbf{U}\mathbf{H}_1\mathbf{U}^T||_F^2 + ||\mathbf{A}_{tf} - \mathbf{V}\mathbf{H}_2\mathbf{W}^T||_F^2 + ||\mathbf{A}_{uf} - \mathbf{U}\mathbf{H}_3\mathbf{W}^T||_F^2 \\ &+ \alpha \cdot tr(\mathbf{U}^T\mathbf{L}^u\mathbf{U}) + \beta \cdot tr(\mathbf{V}^T\mathbf{L}^t\mathbf{V}) + \gamma tr(\mathbf{U}^T\mathbf{L}^r\mathbf{U}) \ s.t. \ \mathbf{U}\mathbf{U}^T = I, \ \mathbf{V}\mathbf{V}^T = I \end{aligned}$$

which clusters over user-user relations ($\mathbf{A}_{uu}$), message-word ($\mathbf{A}_{tf}$) and user-word ($\mathbf{A}_{uf}$) matrices. The matrices $\mathbf{L}^u$, $\mathbf{L}^t$ and $\mathbf{L}^r$ correspond to the Laplacian matrices for users, messages and interactions respectively. R-NMTF however suffers from

long computation times due to slow multiplicative updates.

Wang et al. 2011b propose Fast Non-Negative Matrix Tri-Factorisation (F-NMTF), where both inter-type and intra-type relationship matrices are clustered simultaneously using NMTF. Similar to our work, they perform hard clustering via cluster indicator restrictions to improve computational speed and use manifold regularisation on the intra-type relationships. The authors propose the following objective function,

$$J = ||\mathbf{R} - \mathbf{G}\mathbf{S}\mathbf{G}^T||_F^2 + \lambda tr(\mathbf{G}^T\mathbf{L}\mathbf{G}) \; s.t. \; \mathbf{G} \in \mathbf{\Psi}$$

where $\mathbf{R}$ is a block matrix that concatenates the inter relationships, $\mathbf{G}$ is a diagonal block matrix made up of the individual cluster indicator matrices for each entity type, $\mathbf{S}$ is a block matrix and $\mathbf{L}$ is the normalised graph Laplacian of $\mathbf{W}$ which is a block diagonal matrix concatenating the intra relationship matrices. The first term performs Fast NMTF on the inter relationship matrix while the second term is a manifold regularisation of $\mathbf{W}$. To improve computational speed and scalability, the authors propose an algorithm based on low rank eigendecompositions however these are applied to the extremely large block matrices, $\mathbf{R}$, obtained by combining intra and inter matrices. Although their algorithm does present marginal improvements in computational speed, it is still slow.

Finally, Wang et al. 2019 presents Selective Matrix Factorisation for Multi-Relational Data Fusion (SelDFMF) which uses NMTF to cluster over multi-type relational data where each relational matrix is weighted and is an extension of DFMF. Weighting each relational matrix prevents sparse matrices from having large effects on the clustering Wang et al. 2019. Similar to the proposed approach, SelDFMF does not use a manifold regularisation for the inclusion of intra relations and instead includes these matrices using the standard NMTF approach. SelDFMF has the following objective function,

$$\begin{aligned} J = &\sum_{\mathbf{R}_{ij} \in \mathcal{R}} \mathbf{W}_{ij}^r ||\mathbf{R}_{ij} - \mathbf{G}_i\mathbf{S}_{ij}\mathbf{G}_j^T||_F^2 + \sum_{i=1}^{m}\sum_{t=1}^{\tau} \mathbf{W}_{it}^h ||\mathbf{R}_{ii}^{(t)} - \mathbf{G}_i\mathbf{S}_{ii}\mathbf{G}_i^T||_F^2 \\ &+ \alpha||vec(\mathbf{W}^r)||_F^2 + \beta||vec(\mathbf{W}^h)||_F^2 \\ &s.t. \; \mathbf{W}^r \geq 0, \; \mathbf{W}^h \geq 0, \sum vec(\mathbf{W}^r) = 1 \; \& \; \sum vec(\mathbf{W}^h) = 1 \end{aligned}$$

where $\mathcal{R}$ contains the set of inter relational matrices, $\mathbf{R}_{ij}$ is the inter relational matrix between node types i and j, $\mathbf{R}_{ii}^{(t)}$ is the tth intra relational matrix for entity type i and matrices $\mathbf{W}^r$ and $\mathbf{W}^h$ contain the weights for the inter and intra relational matrices respectively. The function $vec(\cdot)$ performs row concatenation and parameters α and β control the regularisation performed. The weights are regularised to improve parameter selection where irrelevant relational matrices are removed. The weights are calculated automatically however these calculations involve further hyper-parameters. This procedure is very slow in comparison to

the preceding competitive approaches detailed and is not scalable.

All three competitive approaches require hyper-parameters to be set empirically using a known clustering which in practice is not feasible as labelled data often does not exist. Additionally, these methods are not scalable and have long computation times. In the following section, the proposed approach is outlined which improves computational speed and clustering performance.

6.2 Simple Non-Negative Matrix Tri-Factorisation (Simple NMTF) for Multi-Type Bi-clustering

In this section, the novel multi-type clustering procedure is detailed in a general framework that simultaneously clusters over multiple entities whose behaviour is described using a number of adjacency matrices.

6.2.1 Objective Function

In many settings, particularly enterprise networks, high dimensional data sets are generated hence clustering procedures must be computationally fast and scalable. Interpretability is highly important in many fields including cyber-security. Similar to F-NMTF, the factor matrices within NMTF are restricted to be cluster indicator matrices to produce hard clustering results for faster computational speeds and better interpretability.

The aim is to cluster simultaneously over multiple relation matrices consisting of both intra and inter-type relations for multiple entity types where we let:

- $A^{(i)}_{k(i)k(i)}$ be the ith intra-type relational adjacency matrix involving entity type $k(i)$,
- $A^{(i)}_{l(i)m(i)}$ be the ith inter-type relational adjacency matrix between entity types $l(i)$ and $m(i)$,
- Z be the set of entity types.

From these matrices, we seek a single clustering for each of the entity types that incorporates information from the different relations. By incorporating multiple matrices, the latent representation of the entities shared between the data sets can be better realised and can help reduce the noise present in each. To find this shared clustering structure, multiple objective functions are minimised simultaneously in the following manner:

$$J = \sum_{i=1}^{q} ||\mathbf{A}^{(i)}_{k(i)k(i)} - \mathbf{G}_{k(i)}\mathbf{S}^{(i)}_{k(i)k(i)}||^2_F + \sum_{i=1}^{r} ||\mathbf{A}^{(i)}_{l(i)m(i)} - \mathbf{G}_{l(i)}\mathbf{S}^{(i)}_{l(i)m(i)}\mathbf{G}^T_{m(i)}||^2_F$$
$$\text{s.t. } \mathbf{G}_z \in \Psi^{n_z \times c_z},\ \mathbf{S}_{zw} \in \mathbb{R}^{c_z \times c_w}\ \&\ z, w \in Z \tag{6.5}$$

where n_z and c_z are the numbers of nodes and clusters respectively for entity type z. Here the intra relationship matrices $\mathbf{A}^{(i)}_{k(i)k(i)}$ are not bi-clustered as it is a uni-partite matrix. Additionally when the objective for $\mathbf{A}^{(i)}_{k(i)k(i)}$ is minimised in isolation, this is equivalent to k-means clustering. Unlike both F-NMTF and R-NMTF, we do not use manifold regularisation for the inclusion of intra relationships as empirically doing so gave less weight to intra relationships when these are equally as important. Our proposed procedure is simpler and faster. The inter relationship matrices are however bi-clustered so that information about both node types are incorporated during the clustering. This function is minimised with respect to $\mathbf{G}_z$ and $\mathbf{S}_{zw}$ for $z, w \in Z$. This formulation also allows for multiple views of the same relation.

6.2.2 Optimisation Strategy

As the optimisation function is not convex over all matrices, the solution is instead found by iteratively solving for each matrix separately until convergence. First, by fixing all matrices but $\mathbf{S}^{(i)}_{k(i)k(i)}$ and setting the derivative with respect to $\mathbf{S}^{(i)}_{k(i)k(i)}$ of the full objective to zero gives,

$$\mathbf{S}^{(i)}_{k(i)k(i)} = (\mathbf{G}^T_{k(i)}\mathbf{G}_{k(i)})^{-1}\mathbf{G}^T_{k(i)}\mathbf{A}^{(i)}_{k(i)k(i)}. \tag{6.6}$$

Similarly consider when all matrices but the inter $\mathbf{S}^{(i)}_{l(i)m(i)}$ matrix are held fixed. By setting the derivative of the objective function with respect to $\mathbf{S}^{(i)}_{l(i)m(i)}$ to zero gives,

$$\mathbf{S}^{(i)}_{l(i)m(i)} = (\mathbf{G}^T_{l(i)}\mathbf{G}_{l(i)})^{-1}\mathbf{G}^T_{l(i)}\mathbf{A}^{(i)}_{l(i)m(i)}\mathbf{G}_{m(i)}(\mathbf{G}^T_{m(i)}\mathbf{G}_{m(i)})^{-1}. \tag{6.7}$$

Suppose $\mathbf{G}_z$ for $z \in Z$ is associated with multiple norms, when optimising for this matrix, all related terms are considered. Keeping all other matrices but $\mathbf{G}_z$ fixed, the problem is decoupled to the following problem for each node i ($1 \leq i \leq n_z$),

$$\begin{aligned}\min_{\mathbf{G}_z \in \Psi} & \sum_{j \in \{t \in \{1,\ldots,q\} | k(t)=z\}} ||\mathbf{A}^{(j)^{i\cdot}}_{k(j)k(j)} - \mathbf{G}^{i\cdot}_z \mathbf{S}^{(j)}_{k(j)k(j)}||^2_F \\ & + \sum_{j \in \{t \in \{1,\ldots,r\} | l(t)=z\}} ||\mathbf{A}^{(j)^{i\cdot}}_{l(j)m(j)} - \mathbf{G}^{i\cdot}_z \mathbf{S}^{(j)}_{l(j)m(j)}\mathbf{G}^T_{m(j)}||^2_F \\ & + \sum_{j \in \{t \in \{1,\ldots,r\} | m(t)=z\}} ||\mathbf{A}^{(j)^{\cdot i}}_{l(j)m(j)} - \mathbf{G}_{l(j)}\mathbf{S}^{(j)}_{l(j)m(j)}(\mathbf{G}^T_z)^{\cdot i}||^2_F.\end{aligned} \tag{6.8}$$

Algorithm 4 Simple NMTF

Input: Intra-type relational matrices $\mathbf{A}^{(i)}_{k(i)k(i)}$, inter-type relational matrices $\mathbf{A}^{(i)}_{l(i)m(i)}$ and cluster sizes c_z for each entity type $z \in Z$.
Initialisation: Initialise $\mathbf{G}_z$ for $z \in Z$ using the NNDSVD initialisation procedure detailed in Section 6.5
repeat
  Update intra $\mathbf{S}$ matrices using Equation 6.6
  Update inter $\mathbf{S}$ matrices using Equation 6.7
  Update G_z for $z \in Z$ using Equation 6.9
until *Converges*
Output: $\mathbf{G}_z$ for $z \in Z$ indicating cluster membership for each entity type.

As $\mathbf{G}_z$ is a cluster indicator matrix, the optimal matrix is given by,

$$\mathbf{G}_z^{ij} = \begin{cases} 1 & j = \text{argmin}_s \sum_{p \in \{t \in \{1,\ldots,q\} | k(t)=z\}} ||\mathbf{A}^{(p)^{i\cdot}}_{k(p)k(p)} - \mathbf{S}^{(p)^{s\cdot}}_{k(p)k(p)}||^2 \\ & \quad + \sum_{p \in \{t \in \{1,\ldots,r\} | l(t)=z\}} ||\mathbf{A}^{(p)^{i\cdot}}_{l(p)m(p)} - \left(\mathbf{S}^{(p)}_{l(p)m(p)} \mathbf{G}^T_{m(p)}\right)^{s\cdot} ||^2 \\ & \quad + \sum_{p \in \{t \in \{1,\ldots,r\} | m(t)=z\}} ||\mathbf{A}^{(p)^{\cdot i}}_{l(p)m(p)} - \left(\mathbf{G}_{l(p)} \mathbf{S}^{(p)}_{l(p)m(p)}\right)^{\cdot s} ||^2 \\ 0 & \text{otherwise} \end{cases}. \tag{6.9}$$

In the simulated examples in Section 6.6.3 Simple NMTF is applied over a single intra matrix $\mathbf{A}_{CC}$ and two inter-type relation matrices $\mathbf{A}_{CU}$ and $\mathbf{A}_{CP}$. Unlike the three multi-type clustering procedures detailed in Section 6.1.1, the proposed clustering algorithm does not require any hyper-parameters to be set. The proposed optimisation strategy for Simple NMTF is summarised in Algorithm 4. For full derivations see Appendix D.1.

6.2.3 Weighted Extension

To better understand how each relational matrix contributes towards the clustering, weighted multi-type relational clustering can be performed where the weights of each matrix in the objective are optimised and correspond to their contribution to the clustering. This is useful to better understand which relational matrices are more important to the final clustering. Similar to SelDFMF [Wang et al. 2019] described in Section 6.1.1, the inclusion of weights within the optimisation is investigated but we do so using a multi-view approach as there are very few weighted multi-type relational clustering procedures.

Within the multi-relational problem, for entity types with multiple relational matrices, these different *views* can contribute differently to the final clustering to

give greater weight to the representation with a clearer clustering structure. Alternatively, greater weight may be given to relational matrices that have poorer clustering results to allow the clustering to incorporate the structure of these matrices more, such as that done in boosting algorithms. Additionally, it is preferable for the proposed method to autonomously decide these weights.

Many of the weighted multi-view approaches e.g. Tzortzis and Likas 2012 and Liu et al. 2013 utilise a weighted sum of the individual cluster factor matrices as the consensus however as the cluster factor matrices in Simple NMTF are indicator matrices, this will not lead to an appropriate clustering. For example if the idea presented in the Kernel approach of Tzortzis and Likas 2012 is used, the consensus clustering for node type z where i references each of the individual input cluster matrices matrices $G_z^{(i)}$ and I_z is the index set for all data matrices related to z, is given by,

$$\mathbf{G}_z^* = \frac{\sum_{i \in I_z} \left(\lambda_z^{(i)}\right)^\gamma \mathbf{G}_z^{(i)}}{\sum_i \left(\lambda_z^{(i)}\right)^\gamma}$$

where the weights $\lambda_z^{(i)}$ are updated at each iteration using,

$$\lambda_z^{(i)} = \frac{1}{\sum_{v \in I_z} \left(\frac{||\mathbf{G}_z^{(i)} - \mathbf{G}_z^*||_F^2}{||\mathbf{G}_z^{(v)} - \mathbf{G}_z^*||_F^2}\right)^{\frac{1}{\gamma-1}}}.$$

This approach gives the greatest weight to the individual clustering matrix $\mathbf{G}_z^{(i)}$ that is most similar to the consensus, thus the consensus has a greater preference to this clustering. Although the clusterings for each graph could be identical with different permutations, the consensus clustering will favour the most similar clustering even though they all display the same clustering. If no shared cluster assignments exist between the different views, the consensus will be the same as the most similar clustering, thus will not incorporate the different graphs.

Greene and Cunningham 2009 instead cluster each view separately before combining using NMF where the two components represent the contributions and combined clustering respectively. Their approach aims to deal with several multi-view clustering challenges: incomplete views, missing patterns and disagreement between views and is the approach we choose such that all clusterings can be incorporated appropriately.

Weighted Simple NMTF

Weighted Simple NMTF is now proposed using a similar framework as the weighted multi-view clustering of Greene and Cunningham 2009. Rather than calculating a single clustering for each entity type $\mathbf{G}_z$ as done in Simple NMTF, each rela-

tional matrix has a separate clustering which is combined using NMF to give a simple clustering for each entity type. Greene and Cunningham [2009] apply their method retrospectively to pre-calculated clusterings however we instead propose incorporating the combination of the clusterings into the objective as done by other multi-view approaches as follows,

$$
\begin{aligned}
J = &\sum_{i=1}^{q} ||\mathbf{A}^{(i)}_{k(i)k(i)} - \mathbf{G}^{(i)}_{k(i)}\mathbf{S}^{(i)}_{k(i)k(i)}||^2_F + \sum_{i=q+1}^{q+r} ||\mathbf{A}^{(i)}_{l(i)m(i)} - \mathbf{G}^{(i)}_{l(i)}\mathbf{S}^{(i)}_{l(i)m(i)}\mathbf{G}^{(i)^T}_{m(i)}||^2_F \\
&+ \sum_{z \in Z} \lambda_z ||\mathbf{G}^*_z - \mathbf{P}_z\mathbf{M}_z||^2_F \text{ s.t. } \lambda_z \geq 0,\ \mathbf{G}^{(i)}_z \in \Psi^{n_z \times c_z},\ \mathbf{S}^{(i)}_{zw} \in \mathbb{R}^{c_z \times c_w}, \\
&\mathbf{G}^*_z \in \mathbb{R}^{|I_z|c_z \times n_z},\ \mathbf{P}_z \in \mathbb{R}^{|I_z|c_z \times c_z},\ \mathbf{M}_z \in \mathbb{R}^{c_z \times n_z} \text{ for } z, w \in Z
\end{aligned} \tag{6.10}
$$

where the matrix $\mathbf{G}^*_z$ row concatenates all $\mathbf{G}^{(i)^T}_z$ $\forall i \in I_z$ where I_z contains the index set of matrices related to z. Using this approach the contributions or weights of each individual clustering of each graph are contained in $\mathbf{P}_z$ whereas the combined clustering is contained in $\mathbf{M}_z$. More formally the contribution matrix of each view h to the clustering is denoted by $\mathbf{T}_z$ and is calculated as,

$$
\mathbf{T}^{hf}_z = \frac{\sum_{j \in \mathcal{C}^h_z} \mathbf{P}^{jf}_z}{\sum_{g=1}^{|I_z|c_z} \mathbf{P}^{gf}_z} \tag{6.11}
$$

where $\mathcal{C}^h_z$ is the index set of clusterings from the hth view towards the combined cluster matrix $\mathbf{G}^*_z$.

This objective function is solved using a combination of alternating optimisation where each matrix is iteratively solved separately until convergence and multiplicative updates. The optimal solution for the matrices $\mathbf{S}^{(i)}_{k(i)k(i)}$ and $\mathbf{S}^{(i)}_{l(i)m(i)}$ remain unchanged and are solved using Equation 6.6 & Equation 6.7. The intra cluster indicator matrices are updated using,

$$
\mathbf{G}^{(i)^{jt}}_{k(i)} = \begin{cases} 1 & t = \text{argmin}_s\, ||\mathbf{A}^{(i)^{j\cdot}}_{k(i)k(i)} - \mathbf{S}^{(i)^{s\cdot}}_{k(i)k(i)}||^2 - 2\lambda_{k(i)} \left(\mathbf{P}_{k(i)}\mathbf{M}_{k(i)}\right)^{I^{(i)}_{k(i)}(s)j} \\ 0 & \text{otherwise} \end{cases}
$$

and similarly for the inter-type relational matrices,

$$
\mathbf{G}^{(i)^{jt}}_{l(i)} = \begin{cases} 1 & t = \text{argmin}_s\, ||\mathbf{A}^{(i)^{j\cdot}}_{l(i)m(i)} - \left(\mathbf{S}^{(i)}_{l(i)m(i)}\mathbf{G}^{(i)^T}_{m(i)}\right)^{s\cdot} ||^2 - 2\lambda_{l(i)} \left(\mathbf{P}_{l(i)}\mathbf{M}_{l(i)}\right)^{I^{(i)}_{l(i)}(s)j} \\ 0 & \text{otherwise} \end{cases}
$$

$$\mathbf{G}_{m(i)}^{(i)jt} = \begin{cases} 1 & t = \text{argmin}_s \, ||\mathbf{A}_{l(i)m(i)}^{(i)\cdot j} - \left(\mathbf{G}_{l(i)}^{(i)} \mathbf{S}_{l(i)m(i)}^{(i)}\right)^{\cdot s} ||^2 - 2\lambda_{m(i)} \left(\mathbf{P}_{m(i)} \mathbf{M}_{m(i)}\right)^{I_{m(i)}^{(i)}(s)j} \\ 0 & \text{otherwise} \end{cases}$$

where $I_z^{(i)}(s)$ is the index of cluster s for entity type z from input matrix i in the combined concatenated matrix $\mathbf{G}_z^*$. For full derivation of these indicator matrices see Appendix D.2. For $\mathbf{P}_z$ and $\mathbf{M}_z$, as they are dependent on the final term of the objective function in Equation 6.10, the problem can be decoupled to solve standard NMF. Hence standard multiplicative updates are used for these,

$$\mathbf{P}_z^{ij} \leftarrow \mathbf{P}_z^{ij} \frac{\left(\mathbf{G}_z^* \mathbf{M}_z^T\right)^{ij}}{\left(\mathbf{P}_z \mathbf{M}_z \mathbf{M}_z^T\right)^{ij}} \qquad \mathbf{M}_z^{ij} \leftarrow \mathbf{M}_z^{ij} \frac{\left(\mathbf{P}_z^T \mathbf{G}_z^*\right)^{ij}}{\left(\mathbf{P}_z^T \mathbf{P}_z \mathbf{M}_z\right)^{ij}}$$

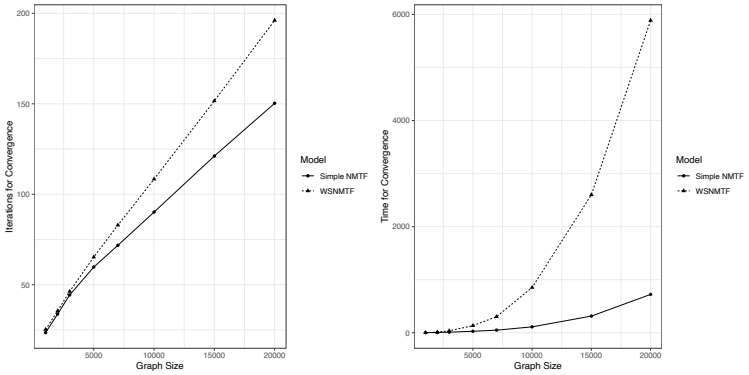
Here for each entity type z a single regularisation parameter must be set to control the influence of the combined clustering on each individual clustering. As a default $\lambda = 0.1$ to allow a suitable contribution to be realised by the algorithm. Unlike Simple NMTF where each graph contributes equally, noisy or irrelevant matrices can be down-weighted to prevent poor clusterings overall. In Appendix D.3 the affect of this regularisation parameter on the cluster performance is investigated where suitable values range in $(0, 0.5]$. Additionally, a comparison between this weighted method and the unweighted Simple NMTF approach are provided in Appendix D.3 where the weighted version has similar performance for entities that have a single view, and improved performance on entities with multiple views or representations for the simulation case investigated. This method is termed the *Weighted Simple NMTF (WSNMTF)*.

6.2.4 Convergence and Complexity

Alternating optimisation is used in Simple NMTF where convergence to a local minimum is guaranteed [Wang et al. 2011c]. By replacing the non-negative matrices $\mathbf{G}$ and $\mathbf{F}$ with cluster indicator matrices, the complexity is reduced as these matrices are sparse. For dense matrices $\mathbf{A}$ $(m \times n)$ and $\mathbf{B}$ $(n \times p)$, their multiplication has complexity $\mathcal{O}(mnp)$. Consider the case where matrix $\mathbf{A}$ is sparse with d non-zero elements. The complexity of the multiplication is reduced to $\mathcal{O}(dp)$ [Yuster and Zwick 2005]. This is a considerable reduction in complexity. For the proposed algorithm with a total of t iterations for convergence, the complexity is $\mathcal{O}(t(\sum_{i=1}^{q} n_{k(i)}^2 + \sum_{i=1}^{r} n_{l(i)} n_{m(i)}))$. In comparison, if indicator matrices are replaced by standard non-negative matrices and a standard multiplicative update rule was implemented such as that in R-NMTF, the complexity is $\mathcal{O}(t(\sum_{i=1}^{q} c_{k(i)} n_{k(i)}^2 + \sum_{i=1}^{r} (c_{l(i)} + c_{m(i)}) n_{l(i)} n_{m(i)}))$.

On simulated graph networks it was found that as the graph size increases,

the number of iterations increases approximately linearly are seen in Figure 6.1. The computation time instead increases exponentially with WSNMTF being much slower at larger graph sizes (Figure 6.1). In Section 6.6.3, the time complexity is compared using computation time for each algorithm.



(a) **(b)**

Figure 6.1: The number of iterations and time (in seconds) for convergence of Simple NMTF and WSNMTF are illustrated for different graph sizes. Three entity types are simulated with a single intra relational matrix and two inter relational matrices where the number of clusters found is a proportion of the graph size. Results are averaged over multiple runs on 20 graphs for each size.

6.3 Stochastic Block Model Generation

To generate the adjacency matrices, a stochastic block model [Holland et al., 1983] is employed where connections between clusters occur with a certain probability and each cluster connects to a specified number of clusters. Stochastic block models lend themselves well to the type of clustering performed where nodes in each cluster connect to similar nodes or clusters.

More specifically, for the simple uni-partite graph case let N nodes be divided into K clusters, $\{C_1, C_2, \ldots, C_K\}$. Let $\mathbf{C} \in \mathbb{R}^{N \times N}$ contain the probability of an edge between the clusters where $\mathbf{C}^{ij}$ is the probability of a connection between nodes in clusters i and j. The connections within the graph are sampled at random where for any two nodes $x \in C_i$ and $y \in C_j$ a connection occurs with probability $\mathbf{C}^{ij}$ where $i, j \in \{1, \ldots, K\}$.

Bi-partite graphs are simulated in a similar manner where instead let there be N_1 nodes of type 1 and N_2 nodes of type 2 where these nodes are split into K_1

and K_2 clusters respectively with the following cluster sets: $\{C_1^1, C_2^1, \ldots, C_{K_1}^1\}$ and $\{C_1^2, C_2^2, \ldots, C_{K_2}^2\}$.

Let matrix $\mathbf{C} \in \mathbb{R}^{N_1 \times N_2}$ contain the probability of a connection between the clusters of the two node types. To generate these cluster connection probability matrices, $\mathbf{C}$, the number of connections for each cluster is restricted to a finite number of clusters, where their probability of connection is sampled over a restrictive range of probabilities to give a stronger or weaker clustering structure for each graph. The following experiments have omitted random noise in the graphs (in the form of random edges) to simplify the problem.

6.4 Cluster Validation Measures

Clustering performance techniques are divided into two categories: external and internal performance measures [Jain and Dubes, 1988]. External performance measures evaluate how similar the clustering is to a known true clustering of the data. These measures are useful for simulated examples where the true clustering is known. Internal measures alternatively utilise the data to assess the clustering performance where a desired property of the clustering is measured such as having compact clusters. To evaluate the performance of the proposed methods and other competitive methods both external and internal measures are used.

6.4.1 External Measures

External cluster validation measures utilise a priori knowledge about the data such as the true intrinsic clustering structure to assess the validity of the calculated clustering. Normalised Mutual Information and the Adjusted Rand Index, two popular measures, are used to assess the external performance. However there is a wealth of such measures, see [Liu et al., 2019] for further choices.

Adjusted Rand Index

The *Adjusted Rand Index* (ARI) [Hubert and Arabie, 1985] measures the similarity between clusterings, accounting for chance. Let n_{ij} represent the number of objects in common between cluster i of the algorithm's clustering and true cluster j, then the ARI is given by,

$$\text{ARI} = \frac{\sum_{i,j} \binom{n_{ij}}{2} - \sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2} / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}\right] - \sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2} / \binom{n}{2}}, \tag{6.12}$$

where $n = \sum_{i,j} n_{ij}$. The ARI has values between -1 and 1 where 0 corresponds to random labelling independent of the numbers of clusters and 1 signifies that the clusterings are identical (up to permutation). The ARI is negative when there is

less agreement between the clusterings than expected for a random labelling. The ARI however favours large clusters [Liu et al., 2019].

Normalised Mutual Information

The *Normalised Mutual Information* (NMI) [Strehl and Ghosh, 2003] is a normalised version of the mutual information between two clusterings. The NMI takes on values between 0 and 1 where 1 corresponds to perfect correlation between clusterings and 0 to no mutual information. Suppose clustering X contains K clusters, $X = \{x_1, \ldots, x_K\}$, where x_i is the set of nodes in cluster i. The NMI between clusterings X and Y is calculated from the mutual information by,

$$NMI(X,Y) = \frac{2I(X,Y)}{H(X) + H(Y)}$$

where $H(X) = -\sum_{x \in X} P(x) \log P(x)$ is the entropy and the mutual information is calculated as,

$$I(X,Y) = \sum_{x \in X, y \in Y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)}$$

with $P(x,y) = \frac{|x \cap y|}{N}$, $P(x) = \frac{|x|}{N}$ and N is the total number of nodes that have been clustered. The NMI is one of the most popular cluster evaluation metrics used in literature. Unlike ARI, the NMI does give importance to smaller clusters however favours larger numbers of partitions [Liu et al., 2019].

It is common to assess the performance of multi-relational clustering using external measures however generally the true clustering is often only known for one entity type e.g. [Del Buono and Pio, 2015] and [Wang and Huang, 2017], hence the performance over all vertices is not truly known. Although one can assess the performance by concatenating the clusterings of the different vertex types, it is important to note that this overall performance will be biased towards larger vertex groups.

Both measures described here are normalised, hence can be used for performance comparison between different clusterings however as these are external measures, they require true cluster labels.

6.4.2 Internal

As labels are often not available in practice, measures that assess the clustering internally should be used. The majority of existing internal measures seek to optimise two criteria: compactness within clusters and separation between clusters [Tan et al., 2018]. Popular measures such as Davies-Bouldin (DB) Index [Davies and Bouldin, 1979], Dunn's Index [Dunn, 1974], Modularity [Newman and Girvan, 2004] and Silhouette Index [Rousseeuw, 1987] focus on these criteria but are devised for the evaluation of community detection which is a different problem. The

clustering method in this work aims to instead group nodes within the network that exhibit similar behaviours, thus the performance measure shouldn't discount clusterings where there are large numbers of edges between clusters. Thus the majority of existing measures are not appropriate.

Bipartite Measures

There are a number of measures that extend existing uni-partite measures to bi-clustering. In particular, there are several extensions of Modularity for bipartite clustering devised by Barber [2007], Guimerà et al. [2007] and Murata [2009], however, many of these extensions are very restrictive. Barber [2007] assume equal numbers of clusters in both node sets with a one-to-one correspondence between the clusters. These conditions are unrealistic in practice. Guimerà et al. [2007] only calculated their measure over a single entity type hence can not give an overall measure of performance of the biclustering. The bipartite Modularity proposed by Murata [2009] is however much more flexible and is described further in Appendix D.4. To our knowledge, multi-type relational clustering performance measures do not exist that internally assess the clustering performance over all matrices and vertex types. To address this we now propose our novel similarity based clustering measure for assessing the performance of the generated clustering for multi-type relational data.

6.4.3 Proposed Node Cluster Similarity (NCS) Measure

As the majority of internal measures are formulated for community detection, they are not suitable to assess the calculated clusterings. The proposed performance measure seeks to evaluate the similarity between nodes in the same cluster in terms of similarity in node connections and similarity in the clusters it has connections to, where higher similarity is desirable. Unlike community detection, edges between clusters are not penalised. The measure however does not take dissimilarity between clusters into consideration where having multiple similar clusters is not penalised hence it may be prone to favouring larger numbers of clusters.

More formally let $\mathbf{a}_i$ be a binary vector detailing the vertices node i connects to and $\mathbf{c}_i$ details the number of edges node i has to each cluster. The *Node Cluster Similarity* (NCS) is proposed to measure the similarity between nodes i and j given by the average of their edge and cluster similarities:

$$s_{ij} = \frac{sim(\mathbf{a}_i, \mathbf{a}_j) + sim(\mathbf{c}_i, \mathbf{c}_j)}{2} \tag{6.13}$$

which under a similarity measure that has values in $[0, 1]$, takes values between 0 and 1 where 1 signifies both nodes have identical connections. The cosine similarity

is used which is defined as,

$$\text{cosine}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{x}||\mathbf{y}|}.$$

For a single node set of size N divided into K clusters C_k, the performance over all nodes is calculated as:

$$\text{NSC}_{\text{single entity}} = \frac{2}{N} \sum_{k=1}^{K} \frac{1}{2} \mathbb{1}_{|C_k|=1} + \sum_{i,j \in C_k, i<j} \frac{s_{ij}}{|C_k| - 1} \mathbb{1}_{|C_k| \neq 1}. \tag{6.14}$$

Equation 6.14 can be extended for bipartite graphs. If $\{C_k, k = 1, .., K_1\}$ and $\{C_k, k = K_1+1, ..., K_1+K_2\}$ are the sets of clusters for node set 1 and 2 respectively and N_1 and N_2 are the number of nodes in these sets respectively, then the NCS for bi-clustering is:

$$\text{NCS}_{\text{bi-clustering}} = \frac{2}{N_1 + N_2} \sum_{k=1}^{K_1+K_2} \frac{1}{2} \mathbb{1}_{|C_k|=1} + \sum_{i,j \in C_k, i<j} \frac{s_{ij}}{|C_k| - 1} \mathbb{1}_{|C_k| \neq 1} \tag{6.15}$$

For multi-type data, the clustering performance can be evaluated using either Equation 6.14 or 6.15 on a matrix with the relevant matrices concatenated along the shared axis. If the bipartite formulation is used on this matrix, a single score detailing the performance of the clustering algorithm over both the rows and columns is produced.

6.5 Cluster Initialisation Strategy

In the majority of papers that utilise NMTF, a random initialisation procedure is used however the clustering solutions for these algorithms are not consistent between runs. Additionally, certain initialisations may prevent a global minima from being reached with slow convergence. Žitnik and Zupan [2015] show for single matrix approaches that initialisation results in smaller final objective function values over random initialisation.

A single clustering is desirable for the interpretability of the data and to ensure the global minima is reached. Non-Negative Double Singular Value Decomposition (NNDSVD) proposed by Boutsidis and Gallopoulos [2008] is widely used for initialising the two components within standard NMF. This popular method utilises the positive section of the k leading singular triplets of the matrix. Alternate SVD based initialisation procedures for NMF have also been proposed e.g. Atif et al. [2019] and Qiao [2015], however, due to its popularity and effectiveness, NNDSVD is extended for multi-type clustering.

Algorithm 5 NNDSVD intialisation procedure.

Input: Matrix $\mathbf{A} \in \mathbb{R}_+^{m \times n}$, integer $k < \min(m, n)$.

1: Compute largest k singular triplets $(\mathbf{U}, \mathbf{S}, \mathbf{V})$ for $\mathbf{A}$
2: Initialise $\mathbf{W}^{*\cdot^1} = \sqrt{\mathbf{S}^{11}}\mathbf{U}^{\cdot 1}$ and $\mathbf{H}^{*\cdot^1} = \sqrt{\mathbf{S}^{11}}\mathbf{V}^{1\cdot}$
3: **for** $j = 1 : k$ **do**
4: $\quad \mathbf{x} = \mathbf{U}^{\cdot j}$ and $\mathbf{y} = \mathbf{V}^{j\cdot}$
5: $\quad$ Let $\mathbf{x}_+,\ \mathbf{x}_-,\ \mathbf{y}_+,\ \mathbf{y}_-$ be the positive and negative parts of $\mathbf{x}$ and $\mathbf{y}$
6: $\quad$ Let $\hat{x}_+ = ||\mathbf{x}_+||$ be the normalised value of $\mathbf{x}_+$ where $\hat{x}_-$, $\hat{y}_+$ and $\hat{y}_-$ are the normalised values for $\mathbf{x}_-$, $\mathbf{y}_+$ and $\mathbf{y}_-$ respectively
7: $\quad m_p = \hat{x}_+\hat{y}_+$ and $m_n = \hat{x}_-\hat{y}_-$
8: $\quad$ **if** $m_p > m_n$ **then**
9: $\quad\quad \mathbf{u} = \frac{\mathbf{x}_+}{\hat{x}_+}$, $\mathbf{v} = \frac{\mathbf{y}_+}{\hat{y}_+}$, $\sigma = m_p$
10: $\quad$ **else**
11: $\quad\quad \mathbf{u} = \frac{\mathbf{x}_-}{\hat{x}_-}$, $\mathbf{v} = \frac{\mathbf{y}_-}{\hat{y}_-}$, $\sigma = m_n$
12: $\quad$ **end if**
13: $\quad \mathbf{W}^{*\cdot j} = \sqrt{\mathbf{S}^{jj}\sigma}\mathbf{u}$ and $\mathbf{H}^{*j\cdot} = \sqrt{\mathbf{S}^{jj}\sigma}\mathbf{v}^T$
14: **end for**

Output: Rank-k non-negative initialisation factors $\mathbf{W}^*$ and $\mathbf{H}^*$ for $\mathbf{A}$.

For matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ of rank $r \leq \min(m, n)$, the optimal rank-k approximation, $\mathbf{A}^{(k)}$ $(k \leq r)$, with respect to the Frobenius norm is given by the SVD rank-k approximation [Eckart and Young, 1936],

$$\mathbf{A}^{(k)} = \sum_{j=1}^{k} \sigma_j \mathbf{u}_j \mathbf{v}_j^T = \sum_{j=1}^{k} \sigma_j \mathbf{C}^{(j)}, \tag{6.16}$$

where σ_j are the non-negative singular values in decreasing order and $\mathbf{u}_j$ and $\mathbf{v}_j$ are the corresponding left and right singular vectors respectively where $\mathbf{C}^{(j)} = \mathbf{u}_j \mathbf{v}_j^T$. The best non-negative approximation of $\mathbf{C}^{(j)}$ with regards to the Frobenius norm is given by its non-negative part $\mathbf{C}_+^{(j)}$ [Boutsidis and Gallopoulos, 2008]. For standard NMF, NNDSVD initialises the two factors using the left and right singular vectors of $\mathbf{C}_+^{(j)}$ respectively multiplied by the square root of the singular value to give $\mathbf{W}^*$ and $\mathbf{H}^*$, the initialised values for standard NMF from NNDSVD. See Algorithm 5 for further details for calculating the initial values using NNDSVD.

6.5.1 Multi-Relational Extension

NNDSVD is designed for single entity clustering. This methodology is now extended to initialise hard bi-clustering. The NNDSVD initialisation procedure is first applied to the bipartite adjacency matrices to produce initial cluster indicator

matrices as follows:

$$\mathbf{F}^{ij}_{\text{initial}} = \begin{cases} 1 & j = \text{argmax}\, \mathbf{W}^{*^{ij}} \\ 0 & \text{otherwise} \end{cases} \qquad \mathbf{G}^{ij}_{\text{initial}} = \begin{cases} 1 & j = \text{argmax}\, \mathbf{H}^{*^{ij}T} \\ 0 & \text{otherwise} \end{cases}. \tag{6.17}$$

This is equivalent to the initial clustering being chosen as the singular vector with the largest value for each node and initially setting $\mathbf{S}$ as the identity matrix.

If the rows and columns are to be clustered into c_r and c_c clusters respectively, NNDSVD is first performed using $\max(c_r, c_c)$ as the rank for the decomposition. When calculating $\mathbf{W}^*$ and $\mathbf{H}^*$, only the leading c_r and c_c singular value/vector pairs are kept respectively before using Equation 6.17 to obtain the initial cluster indicator matrices. For initialisation of multi-type clustering in Simple NMTF, averaging the relevant $\mathbf{W}^*$ and $\mathbf{H}^*$ referring to each entity type before applying Equation 6.17 is performed. For Weighted Simple NMTF, the cluster indicator matrices for each graph are initialised separately by applying NNDSVD to each matrix directly.

6.5.2 Performance Improvement

Žitnik and Zupan [2015] have explored the effect of initialisation on their method however they simply apply single matrix approaches without extending these further for multi-relational cases. Using an approach similar to the authors, the error of the objective function is compared at initialisation and after 20 iterations. To assist in comparison between graphs of different sizes, the standardised error is used given by,

$$\text{Err}(v) = \frac{\sum_{i=1}^{q} ||\mathbf{A}^{(i)}_{k(i)k(i)} - \mathbf{G}^{(v)}_{k(i)} \mathbf{S}^{(i)^{(v)}}_{k(i)k(i)}||_F^2 + \sum_{i=1}^{r} ||\mathbf{A}^{(i)}_{l(i)m(i)} - \mathbf{G}^{(v)}_{l(i)} \mathbf{S}^{(i)^{(v)}}_{l(i)m(i)} \mathbf{G}^{(v)^T}_{m(i)}||_F^2}{\sum_{i=1}^{q} ||\mathbf{A}^{(i)}_{k(i)k(i)}||_F^2 + \sum_{i=1}^{r} ||\mathbf{A}^{(i)}_{l(i)m(i)}||_F^2}$$

where $\mathbf{G}^{(v)}_z$ and $\mathbf{S}^{(i)^{(v)}}_{zw}$ are the cluster indicator matrices and compact representation matrices after v iterations for Simple NMTF. A similar expression for the error of the weighted formulation can be calculated. From this error, we can assess whether the initialisation assists the algorithm in reaching the global minima. Additionally the improvement in performance in terms of the number of iterations for convergence, NMI and ARI is assessed on simulated data of varying sizes. In terms of stability, the standard deviation of the results over 20 runs on each graph is presented where a total of 50 graphs are generated.

The initial error of both Simple NMTF and Weighted Simple NMTF is smaller when the methods are initialised using the proposed NNDSVD procedure however for Simple NMTF, the error after 20 iterations and after convergence is larger (Table 6.1). In terms of clustering performance, the results are very similar however initialising WSNMTF significantly improves the computer clustering which utilises multiple representations of the data. The standard deviation in brackets

corresponds to the variation of the algorithm over multiple runs of the same graph. For both algorithms, initialisation dramatically reduces the standard deviation between clustering results on the same graph, which is important in cyber-security and other domains, to produce stable and meaningful clusters. This is a key benefit of initialisation.

Table 6.1: Effectiveness of initialisation procedure in comparison to random initialisation (Rand) in terms of scaled error, ARI, NMI, number of iterations for convergence and computation time. The standard deviation between runs on the same data set are in brackets. Results are averaged over 50 randomly generated stochastic block models.

	Simple NMTF Rand	Simple NMTF Initial	WSNMTF Rand	WSNMTF Initial
Err(0)	777919.357(137.259)	**590994.131**(28.635)	777963.843(166.398)	**466178.104**(4760.995)
Err(20)	**359935.951**(10094.146)	425467.867(1239.284)	677292.252(3489.669)	**394128.522**(4275.028)
Err Opt	**358989.909**(10042.100)	423826.898(1422.850)	651803.669(5263.182)	**390921.974**(4497.168)
ARI Comp	**0.899**(0.026)	0.778(0.004)	0.003(0.001)	**0.800**(0.014)
ARI User	0.766(0.027)	**0.773**(0.006)	**0.770**(0.022)	0.763(0.009)
ARI Port	**0.858**(0.048)	0.831(0.001)	**0.840**(0.050)	0.818(0.003)
NMI Comp	**0.973**(0.004)	0.953(0.001)	0.446(0.008)	**0.949**(0.003)
NMI User	**0.938**(0.005)	**0.938**(0.001)	**0.939**(0.004)	0.937(0.002)
NMI Port	**0.958**(0.010)	0.955(0.000)	**0.955**(0.010)	0.951(0.001)
Iterations	48.874(7.088)	**44.660**(4.102)	54.840(6.867)	**45.805**(5.116)
Time (seconds)	**44.980**(11.739)	87.044(11.253)	806.638(95.259)	**697.808**(66.998)

In terms of efficiency, initialisation for matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ using the NNDSVD procedure has complexity $\mathcal{O}(mnk)$ for a rank k decomposition. The initialisation procedure does increase the complexity of the algorithm however it generally results in fewer iterations for convergence to a minima and is only performed once on each relational matrix. In the case of WSNMTF, the initialisation improves computational speed due to the significant improvement in iterations for convergence. Finally, producing a unique cluster solution is highly beneficial in practice for interpretability. In the following experiments, the proposed NNDSVD initialisation procedure is applied before clustering.

6.6 Experiments and Results

The comparison in performance of the proposed clustering procedures to competitive and traditional approaches on both simulated and real data sets is now presented. Where labels are available, both internal and external measures are utilised otherwise only internal measures are used.

6.6.1 Simulated Data

In the context of cyber-security, the focus is on monitoring the actions of different node types, specifically computers, users and ports and their interactions. Ports

Table 6.2: Description of parameters used for the graph simulations.

Number Computers / Clusters	5000 / 200
Number Users / Clusters	5000 / 200
Number Ports / Clusters	1000 / 50
CC Number Cluster Connections	{1,2,3}
CU Number Cluster Connections	{1,2}
CP Number Cluster Connections	{1,2,...,7}
Probability of Connection CC	{0.1,0.15,0.2}
Probability of Connection CU	{0.01,0.05,0.1}
Probability of Connection CP	{0.3,0.4,0.5,0.6}

are communication endpoints that can help to identify the type of network connection (e.g. port 80 is used for HTTP traffic). Within network data, various interactions between and within these node types exist where in particular the computer-computer, computer-user and computer-port relations are of interest.

Let $\mathbf{A}_{CC}$ be the adjacency matrix detailing the connections between the computers which is constructed as an undirected graph ($\mathbf{A}_{CC} = \mathbf{A}_{CC}^T$). $\mathbf{A}_{CU}$ details the relationship between computers and users. Finally $\mathbf{A}_{CP}$ details the relationship between computers and ports. The aim is to produce a single computer, user and port clustering that incorporates information from these three graphs simultaneously.

For each simulation, a single computer, user and port clustering is generated by random assignment for the desired cluster numbers. This clustering is used for the generation of all three relational graphs. The following restrictions are implemented to aid in generating graphs that exhibit similar properties to observed network data. For each cluster, the number of connecting clusters is restricted where this number is chosen at random from a finite set of possible values. The probability of connections between clusters is also chosen at random from a restricted set of probabilities. Table 6.2 summarises the parameters used to generate the data. Both the CC and CU graphs are very sparse where for these graphs the model parameters reflect this. The CP graph is much denser where the probability of connection between clusters is higher. Due to these different parameters, the clustering structure in the CC and CU graphs may be less evident than that in the CP graph hence the scale of the performance will differ between these graphs in the results. As initialisation affects the performance for a number of the competitor's methods, each method was run 20 times on 20 simulated networks with results averaged over the runs and networks. All experiments are performed on a four 16-core AMD Opteron CPU at 3.0GHz and 512G bytes memory.

6.6.2 Comparison Methods

Simple NMTF is compared to several competitive methods on real and simulated data sets for the specified problem. The competitive clustering methods applied are divided into the following three classes:

Single Entity Clustering Methods:

Uni-partite clustering is applied to each entity type separately. For the computer clustering, all three data matrices are concatenated to produce a single computer clustering. The methods applied are the standard NMF algorithm [Lee and Seung, 2001] and SVD+K-Means [Zha et al., 2001].

Bi-Clustering Methods:

Orthogonal Non-Negative Matrix Tri-Factorisation (ONMTF) [Ding et al., 2006] is applied to each adjacency matrix separately. Similar to the single entity methods, for the computer entities, clustering is performed over the concatenated matrix to produce a single clustering for this entity type. Utilising ONMTF instead of standard NMTF allows for clustering interpretations as discussed earlier.

Multi-Type Clustering Methods:

F-NMTF [Wang et al., 2011b] and SelDFMF [Wang et al., 2019] detailed in Section 6.1 are used. Additionally R-NMTF proposed by [Pei et al., 2015] is implemented where their method is translated to fit the problem. Each of the methods require hyper-parameter setting where author recommendations are used. For the hyper-parameter λ in F-NMTF, $\lambda = 0.01$. For SelDFMF, the regularisation parameters are set as $\alpha = 10^7$ and $\beta = 10^4$. For R-NMTF $\alpha,\ \beta,\ \gamma = 0.5$.

For all competitive approaches except F-NMTF, R-NMTF and SelDFMF, an initialisation procedure has either been proposed by the cited author or if it is a common approach such as NMF, a standard initialisation procedure is employed. Table 6.3 describes the initialisation procedure used by each competitive method. For further details see original publications for these methods.

Table 6.3: Initialisation procedure for each competitive approach.

Method	Initialisation Procedure
NMF	NNDSVD [Boutsidis and Gallopoulos, 2008]
KMeans	K-Means++ [Arthur and Vassilvitskii, 2007]
ONMTF	K-Mean clustering on rows and columns separately
F-NMTF	Random
R-NMTF	Random
SelDFMF	Random

Table 6.4: Average ARI, NMI, NCS and time (in minutes) comparison for the different methods for the 20 simulated networks. Best performer for each measure in red, second best in blue with standard deviation given in brackets.

	Entity	True	NMF	KMeans	ONMTF	F-NMTF	R-NMTF	SelDFMF	Simple NMTF	WSNMTF
ARI	Computer		0.280(0.030)	0.367(0.061)	0.157(0.060)	0.496(0.029)	0.496(0.030)	0.261(0.018)	**0.705**(0.053)	**0.523**(0.033)
	User		0.284(0.053)	0.008(0.002)	0.322(0.173)	0.119(0.018)	**0.340**(0.081)	0.332(0.031)	**0.393**(0.042)	0.254(0.053)
	Port		**0.928**(0.031)	0.402(0.071)	0.643(0.134)	0.803(0.019)	0.692(0.098)	0.449(0.049)	**0.870**(0.059)	0.861(0.060)
	All		0.299(0.031)	0.052(0.005)	0.169(0.088)	0.232(0.021)	**0.383**(0.079)	0.280(0.013)	**0.515**(0.037)	0.373(0.036)
NMI	Computer		0.707(0.014)	0.801(0.015)	0.632(0.092)	**0.820**(0.013)	0.810(0.044)	0.671(0.011)	**0.925**(0.011)	0.818(0.012)
	User		**0.767**(0.022)	0.507(0.032)	0.669(0.212)	0.699(0.014)	0.758(0.052)	0.703(0.015)	**0.826**(0.012)	0.698(0.025)
	Port		**0.975**(0.006)	0.793(0.029)	0.848(0.054)	0.895(0.016)	0.885(0.041)	0.737(0.020)	**0.962**(0.012)	0.961(0.011)
	All		0.774(0.011)	0.717(0.012)	0.720(0.058)	0.788(0.009)	**0.807**(0.032)	0.718(0.008)	**0.882**(0.008)	0.790(0.012)
NCS	CC-C	0.534(0.005)	0.165(0.014)	0.291(0.017)	0.165(0.069)	0.266(0.015)	0.267(0.042)	0.177(0.007)	**0.405**(0.015)	**0.321**(0.016)
	CU-C	0.547(0.012)	0.109(0.010)	0.308(0.014)	0.125(0.134)	0.234(0.018)	0.292(0.037)	0.126(0.008)	**0.413**(0.018)	**0.329**(0.019)
	CU-U	0.547(0.014)	0.337(0.031)	0.302(0.028)	0.348(0.129)	0.336(0.022)	0.318(0.060)	0.211(0.020)	**0.460**(0.014)	**0.343**(0.029)
	CP-C	0.722(0.004)	0.461(0.021)	0.519(0.022)	0.386(0.053)	**0.593**(0.010)	0.547(0.040)	0.312(0.009)	**0.664**(0.011)	0.455(0.012)
	CP-P	0.727(0.004)	**0.706**(0.006)	0.492(0.030)	0.581(0.041)	0.593(0.021)	0.625(0.034)	0.429(0.016)	**0.641**(0.030)	0.631(0.031)
	Computer	0.653(0.005)	0.383(0.017)	0.460(0.015)	0.337(0.033)	**0.497**(0.015)	0.469(0.038)	0.287(0.008)	**0.586**(0.011)	0.410(0.011)
	Overall	0.611(0.006)	0.391(0.019)	0.391(0.015)	0.364(0.048)	**0.433**(0.015)	0.414(0.033)	0.265(0.009)	**0.533**(0.008)	0.399(0.015)
Time			12.940(5.782)	**2.458**(1.311)	33.218(22.575)	25.024(17.149)	118.822(53.898)	272.685(120.399)	**0.877**(0.631)	8.150(1.200)

Table 6.5: Average ARI, NMI, NCS and time (in minutes) comparison of the different methods for the newsgroup data. Best performer for each measure in red, second best in blue with standard deviation given in brackets.

	Entity	True	NMF	KMeans	F-NMTF	R-NMTF	SelDFMF	Simple NMTF	WSNMTF
ARI	Article		0.102(0.003)	0.102(0.003)	**0.241**(0.058)	0.17(0.04)	0.045(0.002)	**0.267**(0.013)	0.006(0.005)
	Newsgroup		0.039(0.014)	0.030(0.000)	0.0240(0.060)	**0.423**(0.165)	**0.097**(0.040)	0.031(0.073)	0.003(0.062)
	All		0.102(0.003)	0.102(0.003)	**0.241**(0.058)	0.171(0.040)	0.046(0.002)	**0.267**(0.013)	0.006(0.005)
NMI	Article		0.253(0.003)	0.253(0.003)	**0.550**(0.061)	0.351(0.032)	0.104(0.001)	**0.564**(0.014)	0.019(0.012)
	Newsgroup		0.435(0.008)	0.359(0.000)	0.413(0.035)	**0.707**(0.075)	**0.471**(0.054)	0.391(0.061)	0.379(0.081)
	All		0.255(0.003)	0.255(0.003)	**0.551**(0.061)	0.353(0.032)	0.107(0.001)	**0.565**(0.014)	0.022(0.012)
NCS	AW-A	-	**0.303**(0.007)	**0.283**(0.016)	0.188(0.006)	0.253(0.009)	0.190(0.013)	0.184(0.002)	0.178(0.008)
	AW-W	-	0.315(0.003)	0.209(0.006)	**0.340**(0.007)	0.304(0.003)	0.207(0.002)	**0.343**(0.005)	0.231(0.011)
	AN-A	1.000(0.000)	0.257(0.009)	0.438(0.001)	**0.534**(0.066)	0.394(0.024)	0.164(0.014)	**0.622**(0.008)	0.174(0.046)
	AN-N	0.050(0.000)	0.270(0.031)	0.449(0.004)	0.309(0.081)	0.469(0.050)	0.320(0.005)	**0.495**(0.012)	**0.512**(0.021)
	Article	-	**0.307**(0.007)	**0.296**(0.015)	0.203(0.006)	0.267(0.008)	0.197(0.014)	0.204(0.004)	0.181(0.008)
	Overall	-	**0.312**(0.002)	0.236(0.004)	0.299(0.006)	0.293(0.004)	0.204(0.006)	**0.301**(0.003)	0.216(0.007)
Time			42.859(8.456)	**8.957**(1.643)	26.514(5.115)	568.639(26.653)	733.538(7.435)	9.067(3.074)	**8.062**(0.248)

6.6.3 Results

Simple NMTF achieves the highest ARI and NMI for computer, user and overall whereas NMF performs well for identifying the port clustering only (Table 6.4). Similar results are observed for NCS over all matrices and node types, highlighting its consistent clustering performance. It is interesting to note that the NCS of Simple NMTF is close to the NCS value for the true clustering, suggesting a meaningful clustering has been found by the algorithm. From Table 6.4 it is clear that Simple NMTF has either the best or second best performance in comparison to the other methods where the other multi-type clustering approaches F-NMTF and R-NMTF, have competitive performance. This highlights the improvement in clustering using multi-type clustering approaches over single entity methods for this problem. Surprisingly SelDFMF, the most recent competitive advancement in multi-relational clustering, has poor performance with long computation times hence it is less suitable for this clustering problem.

Although on simulated data Weighted Simple NMTF showed promising results, outperforming Simple NMTF (Appendix D.3), under the graph simulation in this example, Simple NMTF has higher performance. WSNMTF however has the benefit of understanding the contribution of each relational matrix to the joint clustering to improve situational awareness.

The poor user clustering performance for all algorithms is due to the weak clustering structure present for this node type by construction. Additionally, although the clustering structure within the CC and CU graphs is much less prominent than that of CP, the clustering algorithm can utilise information from all matrices to uncover the true computer clustering with high similarity to the truth. Simple NMTF is over 4 times faster than the second fastest approach, KMeans, with significantly better performance than all other algorithms.

6.6.4 20-Newsgroup Data

Table 6.6: 20-Newsgroup newsgroup clustering by topic.

comp.graphics	rec.autos	sci.crypt
comp.os.ms-windows.misc	rec.motorcycles	sci.electronics
comp.sys.ibm.pc.hardware	rec.sport.baseball	sci.med
comp.sys.mac.hardware	rec.sport.hockey	sci.space
comp.windows.x		
	talk.politics.misc	talk.religion.misc
misc.forsale	talk.politics.guns	alt.atheism
	talk.politics.mideast	soc.religion.christian

Although this clustering mechanism has cyber motivations, it is not limited to this domain. Additionally, the previous experiments simply illustrate the potential

of this method where the data is not particularly large scale. To demonstrate the scalability and robustness of Simple NMTF its performance is compared to competitors on a well known labelled data set. As we are concerned with the problem of multi-type relational clustering, the data set used must include multiple matrices relating to the different entities. The 20-Newsgroup data set [Lang, 1995] is a collection of 18,846 articles from 20 newsgroups and is a labelled data set. To process these articles we remove stop words, headers, footers and quotes and require a word to have a minimum frequency of 2. This reduces the number of words from 134,410 to 42,837. Binary article-word (AW), and article-newsgroup (AN) relational matrices are constructed. The articles are clustered by their newsgroup where the 20 newsgroups are clustered further into 6 groups described in Table 6.6. The words do not have a known clustering hence only NCS can be used to evaluate the clustering performance on this entity.

The performance score for each algorithm is averaged over 20 runs in Table 6.5. Due to the high computational burden for RNMTF and SelDFMF, these algorithms are only repeated 4 times. Additionally, when running ONMTF on this data, memory requirements are exceeded hence it is omitted from the following results. The computation time of Simple NMTF and Weighted Simple NMTF outperforms other multi-type clustering procedures showing its scalability. Weighted Simple NMTF shows very poor performance on this data hence this method may not be robust in a wide range of applications. Simple NMTF shows high similarity to the true clustering for the articles in particular, which is the harder clustering problem of this data due to a large number of articles. Again consistently high NCS is demonstrated for Simple NMTF over the different entities. Interestingly Simple NMTF also has the highest NCS for the words which do not have a known clustering. These results highlight the robustness and scalability of Simple NMTF.

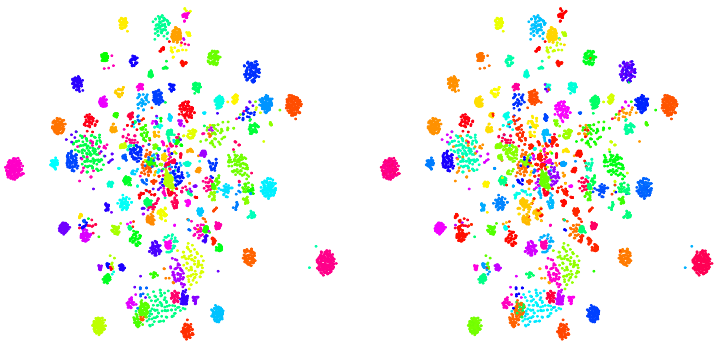
6.7 Cluster Visualisation

To visualise the effectiveness of the clustering procedure, t-distributed Stochastic Neighbour Embedding (t-SNE) [Van Der Maaten and Hinton, 2008, Van Der Maaten, 2014] a popular approach for visualising high dimensional data is applied. Visualising the cluster performance allows the process underlying the data to be uncovered [Keim et al., 2010]. Additionally in the case of real unlabelled data, visualising the results may help determine how well the data has been clustered. t-SNE maps high dimensional data to two or three dimensions where the divergence between the distribution of the pairwise similarity of the input and the pairwise similarity of the low dimensional embedding is minimised. It is recommended by [Van Der Maaten, 2014] to perform a dimensionality reduction technique such as PCA before applying t-SNE to reduce the effect of noise and to speed up calculations. Here PCA with 600 components is applied. The perplexity parameter in t-SNE controls the number of nearest neighbours used in manifold learning and is a significant parameter within the model [Cao and Wang, 2017]. For large data sets,

this parameter value should be large, where throughout the following examples, this value is set to 100.

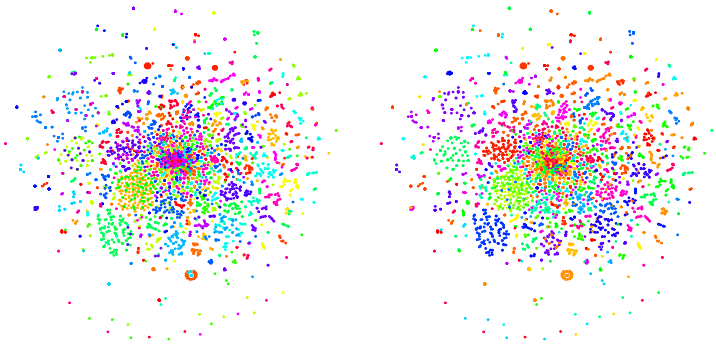
To visualise the performance for each of the computer, user and port clusterings on the simulated data, the t-SNE representation on the relevant data matrices is shown where for the computers, $\mathbf{A}_{CC}$, $\mathbf{A}_{CU}$ and $\mathbf{A}_{CP}$ are concatenated. In the following scatter plots of this section, each point represents a node (of a specific type) where they are coloured according to their assigned cluster. The different clusters should be well separated whilst each cluster has a compact representation. A single simulated graph is used for Figures 6.2, 6.3 and 6.4 where the true clustering and the clustering produced from Simple NMTF, is compared. Comparison for Weighted Simple NMTF, F-NMTF and R-NMTF are found in Appendix D.5.

The computer clustering in Figure 6.2 is very similar between the true and Simple NMTF clusterings suggesting the proposed procedure is effective in clustering such data. Well separated groups of points are generally identified as a single cluster however Simple NMTF sometimes struggles if these clusters are not well separate or are noisy. Interestingly in comparison to the computer and port representations, the user relation representation in Figure 6.3 visually does not have a clear structure which was constructed by design. Additionally, this may suggest t-SNE is not suitable for this data to effectively visualise the clustering results. The port representation very clearly identifies the clusters in Figure 6.4. The clusterings discovered by Simple NMTF are visually very similar to the truth where each distinct grouping of nodes is coloured accordingly. These visualisations suggest Simple NMTF does correctly cluster the data.



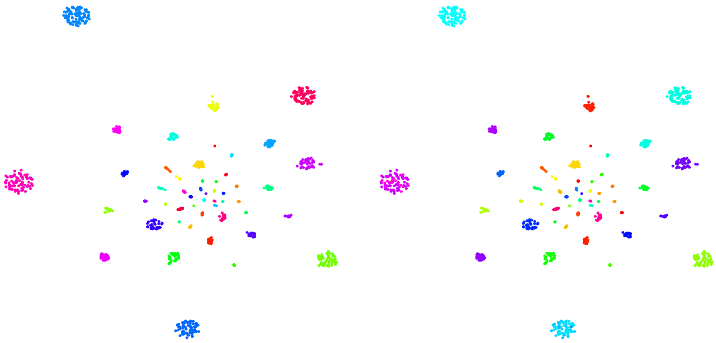
(a) True Clustering **(b)** Simple NMTF Clustering

Figure 6.2: Computer t-SNE representation for true and Simple NMTF clustering respectively. Each colour represents a different cluster.



(a) True Clustering **(b)** Simple NMTF Clustering

Figure 6.3: User t-SNE representation for true and Simple NMTF clustering respectively. Each colour represents a different cluster.



(a) True Clustering **(b)** Simple NMTF Clustering

Figure 6.4: Port t-SNE representation for true and Simple NMTF clustering respectively. Each colour represents a different cluster.

6.8 Discussion

This chapter proposes a multi-type clustering procedure, Simple NMTF, that significantly improves the computational speed and clustering performance on existing approaches making it suitable for large scale data sets. Many of the existing

procedures in this field suffer from slow multiplicative updates where by replacing the non-negative cluster factor matrices with binary cluster indicator matrices, fast updates are achieved. This procedure allows for greater situational awareness of the network in identifying nodes with similar behaviour within each entity type based on their behaviour over the entire network. The ability of this method to be applied to alternate applications is demonstrated.

A weighted version of the clustering procedure is also proposed which allows each graph to contribute with different weights towards the combined clustering. These weights are useful for understanding the signal presented in each graph. Finally, the performance of Simple NMTF outperforms competitive methods on both simulated and real data. This clustering procedure is applied further to a real cyber-security network in Chapter 8.

A number of novel contributions have additionally been made in the field of multi-type clustering which has not yet been extensively studied. Our contributions include extending a univariate cluster initialisation procedure to both the bi-partite and multi-type clustering problems. This initialisation procedure improves the stability of the clustering in comparison to random initialisation with good performance. The second contribution of this chapter is the proposed internal clustering performance measure which is suitable for multi-type clustering. This measure shows consistent results to well established external cluster performance measures.

In the next chapter, the clustering procedure is extended to a streaming implementation such that the clustering adapts over the network stream. This adaptive clustering procedure can then be monitored to detect intruders.

Chapter 7

Dynamic Multi-Relational Clustering

In the previous chapter Simple NMTF was proposed that groups nodes with similar activities over the network to better understand the structure of the data. This method showed capabilities of clustering with high accuracy on static networks. However in practice, the network structure changes thus clustering procedures need to be adaptive. In this chapter we extend Simple NMTF to the more challenging dynamic case where the clustering updates as edges within the relational graphs are added and removed. Additionally, the procedure allows nodes to be added and removed from the graph.

Whilst static network analysis has been heavily studied, the dynamic case is comparably less investigated [Matias and Miele, 2017]. Commonly in dynamic clustering literature (e.g. [Matias and Miele, 2017] and [Ma and Dong, 2017]) there is an emphasis on balancing the smoothness between consecutive clusterings of the data and clustering performance. This balance is necessary as most nodes seldom change their cluster assignment between time points. In the context of cyber-security, this is especially relevant as it is expected that each user performs actions related to their job where these actions should not vary greatly over time. Alternatively, the adaptation between clusterings may be more gradual as their job role shifts.

Much of the existing literature focuses on univariate dynamic clustering and in particular for community detection which as discussed in Chapter 6 is a different problem to the proposed approach. The literature can be split into non-evolutionary and evolutionary models. Non-evolutionary models such as that presented by [Kumar et al., 2006], [Asur et al., 2009] and [Sallaberry et al., 2013], apply static clustering approaches to each instance of the graph separately and find relations between these consecutive clusterings, mapping clusterings between time stamps based on their similarity. However, these approaches do not leverage the clustering from the previous time. Evolutionary clustering approaches instead apply temporal smoothness such that consecutive clusterings are related. There are a

wide variety of approaches that have been applied such as stochastic block models [Matias and Miele, 2017, Ludkin et al., 2018], modularity based clustering [Görke et al., 2013] and matrix factorisation [Cao et al., 2007, Ma and Dong, 2017, Xiong et al., 2010, Wang et al., 2011a] including NMF. [Cao et al., 2007] extend NMF to allow for additional data instances to be added. This approach does not however allow for the activity of previous data instances to change. [Ma and Dong, 2017] instead extend NMF to minimise a convex combination of the snapshot cost (i.e. how well the current graph is clustered) and the temporal smoothness to ensure cluster membership is maintained. [Wang et al., 2011a] instead utilise projected gradients to incrementally update the NMF factor matrices using the change in the graph between time points to prevent a full recalculation of the clustering.

The clustering procedures described however primarily focus on fixed graph sizes and univariate graphs. In the case of cyber-security, multi-type relationships exist where capitalising on this joint information allows for improved clustering. To our knowledge, dynamic multi-type relational clustering approaches do not exist. A similar approach to [Câmpan and Åerban, 2006] is adopted where the calculated clustering from the previous time step is used to initialise the clustering procedure at the current time. Additionally, the graph size is allowed to expand and retract over time as nodes become active and inactive respectively.

This brief chapter is structured as follows, Section 7.1 presents the dynamic clustering extension of Simple NMTF proposed in Chapter 6. Section 7.2 contains the experimental results. First, a novel dynamic multi-type relational graph generation model is detailed in Section 7.2.1 using a stochastic block model which allows for changes in cluster membership. Experimental performance results are presented in Section 7.2.3 to assess the robust performance of the procedure in comparison to the static version in the presence of cluster changes.

7.1 Adaptive Graph Monitoring

Consider the sequence of graphs, $G_1, G_2, \ldots, G_t, \ldots$ where G_t is the network at time t with clustering c_t. The aim is to cluster the graph at time $t+1$, G_{t+1}, leveraging the clustering of time t. Hard clustering procedures such as Simple NMTF and k-means clustering lend themselves well for adaptive clustering as initial clusterings are required for each algorithm where these initial clusterings influence the local minima of the optimisation. An approach similar to [Câmpan and Åerban, 2006] is employed where we initialise the algorithm with the clustering calculated at the previous time to maintain smoothness between consecutive clusterings whilst additionally improving computational efficiency.

7.1.1 Cluster Updates

Between consecutive time steps, it is assumed the graph structure is similar, with very few nodes changing their cluster assignment. As only the cluster indicator

matrices $\mathbf{G}_z$ are required to initialise the Simple NMTF algorithm presented in Chapter 6, the previous clustering is utilised as a starting point. For time $t = 1$ the proposed multi-type NNDSVD initialisation procedure is used instead. This initialisation approach ensures the clustering between consecutive times is maintained if no change in assignment occurs. However, if a change has occurred, the algorithm is not restricted to maintain this clustering and a new minima will instead be found.

7.1.2 Addition and Removal of Nodes

In many domains including cyber-security, the size of the graph may adapt over time as nodes become active or inactive. For example, for Netflix movie recommendations, the movies offered on the service may increase as new films are released however these may also be removed by Netflix as they discontinue showing the film. Thus it is important to allow the graph size to vary.

For a new node i^* at time t, it's graph clustering is initialised using the results from time $t-1$. Using the notation from Chapter 6, let the cluster indicator matrices from time $t-1$ be $\mathbf{G}_z$ for $z \in Z$ and the compact representation matrices be $\mathbf{S}^{(i)}_{k(i)k(i)}$ and $\mathbf{S}^{(i)}_{l(i)m(i)}$. The initial cluster assignment for new node i^* using the results from time $t-1$ is,

$$\begin{aligned} j = \underset{s}{\operatorname{argmin}} & \sum_{p \in \{t \in \{1,\ldots,q\} | k(t)=z\}} ||\mathbf{a}^{(p)}_{k(p)k(p)} - \mathbf{S}^{(p)^{s.}}_{k(p)k(p)}||^2 \\ & + \sum_{p \in \{t \in \{1,\ldots,r\} | l(t)=z\}} ||\mathbf{a}^{(p)}_{l(p)m(p)} - \left(\mathbf{S}^{(p)}_{l(p)m(p)} \mathbf{G}^T_{m(p)}\right)^{s.} ||^2 \\ & + \sum_{p \in \{t \in \{1,\ldots,r\} | m(t)=z\}} ||\mathbf{a}^{(p)}_{l(p)m(p)} - \left(\mathbf{G}_{l(p)} \mathbf{S}^{(p)}_{l(p)m(p)}\right)^{.s} ||^2 \end{aligned} \tag{7.1}$$

where $\mathbf{a}^{(p)}_{k(p)k(p)}$, $\mathbf{a}^{(p)}_{l(p)m(p)}$ and $\mathbf{a}^{(p)}_{l(p)m(p)}$ are the connection vectors for node i^* at time t for each of the intra and inter relational matrices and the $\mathbf{G}$ and $\mathbf{S}$ matrices are that calculated from time $t-1$. Thus the initial clustering for this node is the optimal clustering at time $t-1$.

Alternatively, if a node is removed, its corresponding rows in the cluster indicator matrices are removed. If this node is in its own cluster, the empty cluster should additionally be removed from the cluster indicator matrix. This procedure is named *Dynamic SNMTF* and is summarised by Algorithm 6.

7.1.3 Relation Between Clustering

Between each consecutive iteration, clusters may remain, split, merge, emerge or vanish. To determine whether a node stays in the same cluster over time, the

Algorithm 6 Dynamic SNMTF

Input: Initial intra-type relational matrices $\mathbf{A}^{(i)}_{k(i)k(i)}$, inter-type relational matrices $\mathbf{A}^{(i)}_{l(i)m(i)}$ and cluster sizes c_z for each entity type $z \in Z$.
Initialisation: Apply Simple NMTF to initial network to calculate starting clustering at time $t = 1$.
Output: Sequence of clusterings.

1: **for** $t = 2, 3, \ldots$ **do**
2: For nodes active in previous period, initialise clustering with previously calculated clusters.
3: For new nodes, assign clustering using Equation 7.1.
4: Remove empty clusters.
5: Apply Simple NMTF with new initialisation.
6: **end for**

transition of the clusters is monitored. To measure the smoothness between clusterings, a similar cluster tracking mechanism as Landauer et al. 2018 is employed whereby the overlap between clusters from time $t-1$ and t is calculated. If the overlap between these clusters is greater than some threshold then these clusterings can be mapped between times $t-1$ and t. More formally, let the nodes in cluster i at time $t-1$ be represented by the set c_i and nodes in cluster j from time t be represented by c'_j. The overlap between clusters i and j is defined as,

$$\text{overlap}(c_i, c'_j) = \frac{|c_i \cap c'_j|}{|(c_i \cup c'_j) \cap n \cap n'|} \tag{7.2}$$

where n is the set of active nodes at time $t-1$ and n' is the set of active nodes at time t. This quantity has a value of 1 if the nodes that are active within both time intervals within clusters c_i and c'_j are the same. The overlap measure is based on the Jaccard coefficient where the similarity between sets C and C' is calculated as,

$$\frac{|C \cap C'|}{|C \cup C'|}$$

and is used by Greene et al. 2010 for tracking the overlap between clusters. In the proposed formulation in Equation 7.2, the additional restriction of nodes on the denominator to those within both time intervals allows the overlap to be equal to 1 in cases where nodes become active/inactive between the two times.

A cluster c'_j at time t originates from cluster c_i at time $t-1$ if $\text{overlap}(c_i, c'_j) > \theta$ for $\theta \in [0, 1]$. This quantity is used to determine whether the cluster assignment of a node has changed. For each new cluster c', the clusters from time $t-1$ such that the overlap is greater than some threshold are detailed. This approach allows for both merging and splits provided the threshold is small enough. Here $\theta = 0.5$. For each node, if the previous cluster assignment matches the mapped clustering,

then the cluster assignment has not changed.

The primary focus of this chapter is on the cluster performance of the proposed dynamic clustering procedure rather than monitoring the cluster changes. For further details on cluster transition characteristics see Spiliopoulou et al. [2006].

7.2 Simulation Study

The performance of the proposed dynamic clustering procedure is investigated to assess whether consistent clustering performance is achieved with smoothness between consecutive clusterings.

7.2.1 Data Generation

The graph generation procedure from Section 6.3 is extended to allow for cluster changes and dynamic node activity. First, the cluster allocation process is described. Similar to Ludkin et al. [2018] the cluster membership of each node is modelled using a continuous time Markov chain [Norris, 1998]. Let $C_i(t)$ be the cluster assignment of node i at time t. It is assumed that the distribution for the time node i spends in each cluster is $\text{Exp}(\lambda_i)$, independently of the current cluster assignment. This new cluster assignment is chosen uniformly over the cluster choices. Thus the number of cluster changes, M_i, for node i is generated by, $M_i \sim \text{Po}(\lambda)$ with times of change, $\boldsymbol{\tau}_i = (\tau_i^1, \ldots, \tau_i^{M_i})$, which are sampled uniformly over the time interval $[2, T-1]$ where T is the maximum length of the series.

At each time step, the current cluster assignment for each node is established by the above procedure. Cluster assignments are simulated for each node type. Using these cluster assignments, graphs can be simulated using the stochastic block model as described in Section 6.3 at each time t. To allow for a dynamic graph, a sliding window approach is employed where s consecutive simulated graphs are combined. This sliding window allows the graphs to change smoothly. As multiple graphs are combined, the probability of connection between clusters is reduced approximately by a factor of the number of bins in the sliding window to achieve a similar number of edges to the full graphs in the static case in Chapter 6. This sliding window approach is used in practice in cyber-security to produce graph streams where at each time step new connections are added to the graph while old expired edges are removed.

The generated graphs in the proceeding experiments are simulated with the parameters detailed in Table 7.1. In comparison to the parameters for the static graph in Table 6.2, the probability of connection is reduced and additionally smaller graph sizes are investigated. A sliding window of size 144 is employed where it is assumed each graph represents 5 minutes of data resulting in each window containing 12 hours of data. A total of 1440 graphs are simulated, equivalent to five days. Specification of λ_i, the expected number of cluster changes for each node, is described in each simulation example.

Table 7.1: Description of simulated data parameters for the dynamic graph generation.

Number Computers / Clusters	1000 / 20
Number Users / Clusters	1000 / 20
Number Ports / Clusters	500 / 10
CC Number Cluster Connections	{1,2,3,4}
CU Number Cluster Connections	{1,2,3,4}
CP Number Cluster Connections	{1,2}
Probability of Connection CC	{0.001,0.002,...,0.008,0.009}
Probability of Connection CU	{0.001,0.002,...,0.008,0.009}
Probability of Connection CP	{0.001,0.002,...,0.008,0.009}

7.2.2 Performance Measures

As mentioned previously, two key aims when performing dynamic clustering are balancing the current cluster performance and the smoothness between clusters. To achieve this both the cluster performance and the similarity between clusterings is measured. For the cluster performance, the same measures as in Chapter 6 are employed, more specifically ARI, NMI and the proposed measure NCS. For full details of these measures see Section 6.4. As ARI and NMI show very similar behaviour, NMI is omitted from the proceeding analysis. In the following experiments, it is desirable for NCS, the proposed internal measure, to show similar patterns to ARI so that this measure can be used in practice when labels are not available.

To determine the smoothness between consecutive clusterings the similarity between clusterings is measured using the ARI. For graphs with no cluster changes, high values are desired. However, for graphs with many cluster changes, this may be lower. Finally, for each node type, the number of cluster changes is reported, where a cluster change is determined by the overlap between consecutive clusterings as described in Section 7.1.3.

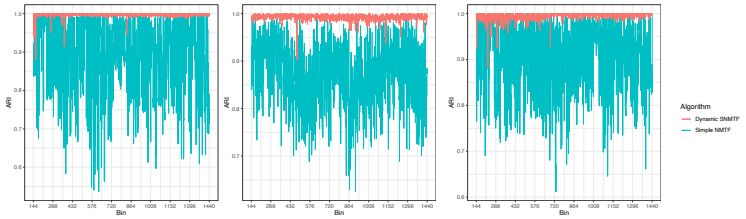
To our knowledge, there does not exist any competitive dynamic multi-type clustering methods that can be used for comparison. In literature, dynamic clustering procedures are primarily univariate or bivariate. Instead, we choose to compare to Simple NMTF where at each update, the full clustering procedure is implemented with NNDSVD initialisation. By comparing to the restart model, the degree of performance degradation caused by enforcing cluster smoothness in Dynamic SNMTF is assessed. Additionally, each clustering approach is timed to determine the speedup provided by initialising with the previous clustering.

7.2.3 Results

First, the dynamic clustering is applied to static graph streams in Appendix E. For the static clustering example (Appendix E.1), Dynamic SNMTF provides a good

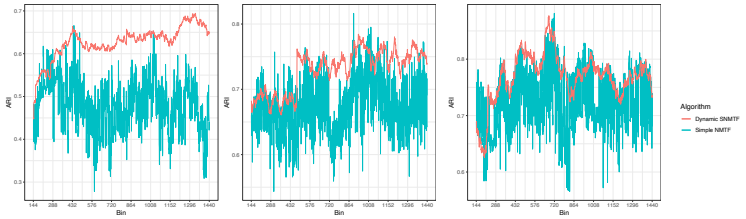
balance between the smoothness of the clustering and the performance. For the single node cluster change example (Appendix E.2) it is clear Dynamic SNMTF is robust to cluster changes within the data.

To mimic real cyber-security networks all nodes from each node type are allowed to change cluster assignment. We set $\lambda_i = 2\ \forall i$, for all node types as in practice, nodes rarely change cluster. For this rate value some nodes do not have any cluster changes, reflecting what is observed in practice. This simulation example should help to determine whether the proposed Dynamic SNMTF is robust to graph streams which exhibit many cluster changes.



(a) Computer **(b)** User **(c)** Port

Figure 7.1: ARI of consecutive clusterings over a graph stream as described in Section 7.2.1 where $\lambda = 2$ in the cluster membership generation process resulting in many cluster changes over the stream.

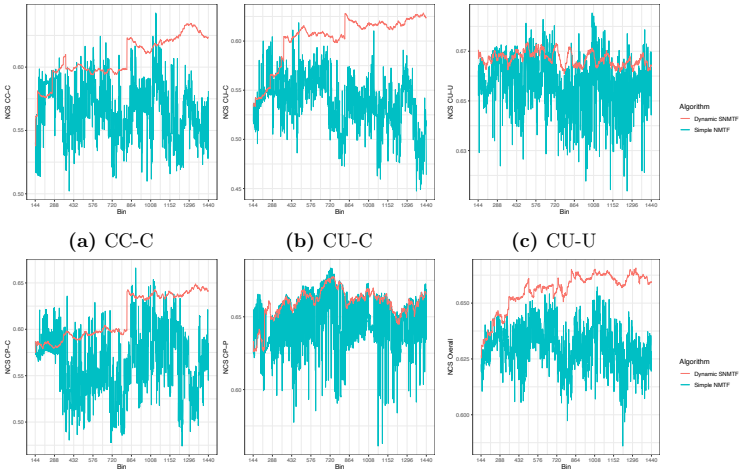


(a) Computer **(b)** User **(c)** Port

Figure 7.2: ARI to true clusterings over a graph stream as described in Section 7.2.1 where $\lambda = 2$ in the cluster membership generation process resulting in many cluster changes over the stream.

Figure 7.1 displays the similarity between consecutive clusterings. The ARI for Dynamic SNMTF seen in Figure 7.1 is more noisy in comparison to the more static examples in Appendix E but is still very high. Additionally, over the investigated period, the ARI does not reduce in value suggesting the process does not degrade over time. For Simple NMTF, there is less similarity between consecutive clusters. Thus the proposed procedures drastically improve smoothness.

For the ARI to the true clustering in Figure 7.2, the ARI improves initially for the Dynamic SNMTF model before reaching a constant level. Dynamic SNMTF has more noisy results for this simulation in comparison to the static cases, but the values are still relatively smooth. It is important to note that the ARI for the port clustering is lower than seen for the static cases as the problem is now much harder. As the graph contains a sliding window of edges, edges from two distinct clusterings may be present making the clustering harder to identify.



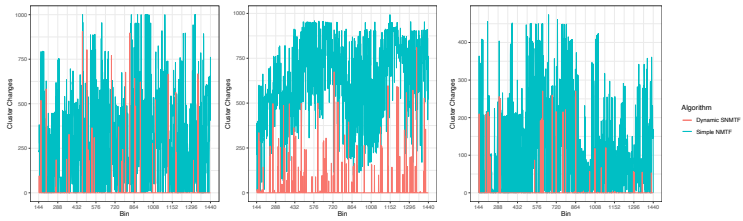
(a) CC-C **(b)** CU-C **(c)** CU-U

(d) CP-C **(e)** CP-P **(f)** Overall

Figure 7.3: NCS for calculated clusterings for a simulated graph stream as described in Section 7.2.1 where $\lambda = 2$ in the cluster membership generation process resulting in many cluster changes over the stream.

Similar to the ARI, the NCS for Dynamic SNMTF is higher than Simple NMTF in Figure 7.3. Additionally, overall, the internal cluster performance is higher. These results suggest Dynamic SNMTF is robust to cluster changes, with superior performance to restarting the clustering at each update as done by Simple NMTF. Dynamic SNMTF is also 5 times faster than Simple NMTF.

Finally for the number of cluster changes for each node type in Figure 7.4, in comparison to the static simulations, there are larger numbers of cluster changes however Dynamic SNMTF only has spikes of change rather than continuous large change as seen by Simple NMTF. Thus from this experiment, it is clear that Dynamic SNMTF has a good balance between smoothness between clusterings and high cluster performance making it suitable for application to real graph streams.



(a) Computer **(b)** User **(c)** Port

Figure 7.4: Number of cluster changes for each node type for a graph stream as described in Section 7.2.1 where $\lambda = 2$ in the cluster membership generation process resulting in many cluster changes over the stream.

7.3 Discussion

This chapter has presented an extension to the proposed static multi-type clustering procedure from Chapter 6 to allow for dynamic clustering. By utilising the clustering calculated at the previous time as an initialisation for the procedure, smooth clusterings can be calculated without compromising cluster performance. The dynamic method has been demonstrated on several simulated cases with and without cluster changes.

A dynamic multi-type graph generation procedure is also presented to help compare the performance of the proposed dynamic and original static version of Simple NMTF. The dynamic graph generator allows for varying rates of cluster change for each node and a sliding window approach is employed. To our knowledge, no competitive dynamic multi-type relational clustering procedures have been proposed in the literature. The dynamic clustering procedure is demonstrated on a real cyber data set in the next chapter.

Chapter 8

Cyber-Security Applications

In the previous chapters, the proposed procedures are detailed for general application where non-cyber applications are used to showcase their wide applicability. However, the key motivator for this thesis is constructing monitoring procedures for enterprise cyber-security data. Attention is now restricted to the *Unified Host and Network Data Set* [Turcotte et al., 2019] from Los Alamos National Laboratory (LANL) [1] which provides a wealth of information about LANL's enterprise network. This data set is unusual in that it contains both Netflow data, detailing the connections between computers and Windows Logging Service Data, recording authentication and process events for each host, making it comprehensive in comparison to competitive data sets.

This chapter first describes the LANL data generally in Section 8.1 before applying each of the proposed methods in the proceeding sections. For each separate method, further details about the extracted data used from the LANL data logs are given. Each of the proposed methods monitors a different aspect of the data allowing for different activities to be tracked.

8.1 LANL Data Description

The LANL data set consists of 90 consecutive days for the *Windows Logging Service* (WLS) data and 89 days for the Netflow data (due to missing data on the first day). Similar to LANL's previously released data set [Kent, 2015], the data does not include any records of external connections and has been anonymised for security purposes. In comparison to other publicly available data sets, the LANL data set benefits from containing events from a real enterprise network rather than a simulated network.

[1] The data is publicly available online upon request at `https://csr.lanl.gov/data/2017/` (last accessed on 8th March 2022)

UNIX epoch time is used, with a start epoch of 1 and a 1 second time resolution. From analysing the activity levels, it is evident that the data starts on a Monday (day 1) and is assumed throughout the subsequent analysis. The computer names match across both Netflow and WLS data sets and have been de-identified however, the Netflow data also contains many non-Windows computers and other devices that do not exist in the WLS data. Well-known network ports, system-level usernames, processes and core enterprise hosts are not de-identified.

8.1.1 Netflow

The Netflow data contains logs of connections between computers within the network collected by the core network routers. As raw Netflow data is uni-flowed (no relationship between the direction of the record and the initiator of the connection) and contains duplicates, the data has been transformed into a Biflow which aggregates duplicates and combines opposing uniflows of connections into a single directed Biflow record. More details of the data transformation can be found in Turcotte et al. 2019.

Over the 90 days, approximately 58,000 computers are tracked across the data. There are around 200 million events per day, hence the data sets used here are extremely large. As a result, the methods and models applied must be scalable to meet the requirements of such sizeable data.

Time, Duration, SrcDevice, DstDevice, Protocol, SrcPort, DstPort, SrcPackets, DstPackets, SrcBytes, DstBytes

Figure 8.1: Example of Netflow data structure.

The Netflow records detail the time of connection, the duration, the source (src) and destination (dst) devices that the connection is initiated and terminated at respectively, the protocol (limited to Protocols 1, Internet Control Message Protocol (ICMP), 6, Transmission Control Protocol (TCP), and 17, User Datagram Protocol (UDP)), source and destination ports, source and destination packets and source and destination bytes with an example of the field names in Figure 8.1.

8.1.2 Windows Host Log Data

The Windows Logging Service events detail the authentication and process activity on computers running the Microsoft Windows operating system. Each record contains an *EventID* (which uniquely identifies an event type) as well as a *LogHost* (computer identifier where the event is recorded) and *Time*.

Depending on the type of event, each record may have 18 other possible attributes. Other important attributes include the *UserName*, which details the user initiating the event and will be termed the *source* user and the *SubjectUserName* which refers to the *destination* user for mapped authentication events but

this field is often the same as the *UserName*. Both of these usernames are specific to the domain which is detailed in *DomainName* and *SubjectDomainName* for *UserName* and *SubjectUserName* respectively. For the subsequent work, these will be appended together with an "@" to ensure unique users are identified in the network.

"UserName": "Comp505809$", "EventID": 4624, "LogHost": "Comp505809", "LogonID": "0x1b9dba9f", "DomainName": "Domain001", "LogonTypeDescription": "Network", "AuthenticationPackage": "Kerberos", "Time": 1296000, "LogonType": 3

{"UserName": "User959346", "EventID": 4688, "LogHost": "Comp031357", "LogonID": "0x12535e2", "DomainName": "Domain001", "ParentProcessName": "Proc143120", "ParentProcessID": "0xf8c", "ProcessName": "Proc027182.exe", "Time": 1296000, "ProcessID": "0xe00"}

Figure 8.2: Two examples of Windows Host Logs.

Two accounts with different event types from the raw log data are displayed in Figure 8.2. The first example details a local network logon authentication event on *Comp505809*. Logons of this kind are the most common event type for EventID 4624 and represent the user accessing a Windows resource (e.g. shared folders or printers) but could also be logons to Internet Information Services (IIS). The second example details a process event on *Comp031357* by user *User959346@Domain001* and gives details of the parent process and the process ID. These logs record each program that has been executed and the process that initiated it.

As the events for the WLS data are restricted to Windows computers only, the number of computers in this data set is much smaller than the Netflow data. For the WLS data set, there are approximately 56 million events, 9000 users and 9000 host computers per day. Similar to Netflow, this data set is substantially large.

8.2 Research Problem

The key motivator for this thesis is to develop monitoring procedures suitable for cyber-security data streams. Through monitoring the data, network analysts have the potential to identify network intrusions or other unusual network activity. There are a large number of attack vectors taken by adversaries such as distributed denial-of-service (DDoS), probe and user to root [Ahmed et al., 2016] each leaving a different trace on the network data. Thus multiple statistical methods should be applied for a comprehensive tool set for intrusion detection. For most attack vectors, the attack data differs from the normal behaviour of the network, allowing for detection. However, these differences are often slight, making them hard to detect. Intrusion detection approaches can be split into three approaches: node-based ([Passino and Heard, 2019]), edge-based ([Noble and Adams, 2016], [Metelli and Heard, 2019]) and global models ([Akoglu et al., 2015], [Heard and Rubin-Delanchy, 2016]) where the applications in this chapter focus on the latter two.

Our change point and anomaly detectors from Chapters 4 and 5 respectively are appropriate for detecting abnormalities in univariate monitoring statistics from the network. Ideally these statistics contain possible indicators of attacks in the data. Alternatively, the data can naturally be represented by multiple network graphs describing the multi-type relations between the computers, users and ports. Through clustering this data, better situational awareness about how the nodes interact can be formed. The data processing performed to produce the univariate and graph data streams used in the subsequent analysis are now described.

8.2.1 Data Processing

The primary concern is with computer to computer (CC), computer to port (CP) and computer to user (CU) connections or edges within the network where graphs and data streams are produced from these. To ensure the nodes investigated do not correspond to automated processes or activities which may dilute the signal of unusual activities in the data, the following restrictions are implemented:

- Only the 58,000 computers whose activity has successfully been tracked across the full period of the data is investigated;
- Computers must be active in both Netflow and WLS data;
- Computers and users must be contained in the de-identified characteristic data set provided by LANL.

Although the data has second resolution, it is common practice in cyber-security to bin the data. A variety of bin sizes are used in literature including minute [Nezhad et al., 2016], 5 minute [Whitehouse et al., 2016] and 15 minute [Riddle-Workman et al., 2018] bins. Minute and 15 minute intervals are utilised in this chapter. For further details on optimal bin sizes see Neil et al. [2013] who suggest 30 minute bins to adequately capture the behaviour of the network without being buried in non-intrusion data. In general, the optimal bin size depends on the problem at hand where there is an additional computational trade-off for the window size selected.

In the following, only the weekdays are investigated as there is less activity on weekends, where the activity patterns of these periods are distinct. As true anomalous times are unknown in the LANL data set, the following results are exploratory and showcase the applicability of the proposed methods to real cyber data.

8.2.2 Univariate Data Processing

During an attack, novel connections or *edges* may be made between entities within the network (computers, users and ports) and can be used as an indicator of intrusion [Metelli and Heard, 2019]. Thus investigation of such quantities is of

interest. To classify an edge as new, a sliding window approach is utilised over the bins where if an edge has not been seen in the past s time bins, it is classified as new. The number of new edges is investigated for both the change point procedure in Section 8.3 and the forecast procedure in Section 8.4. In the proceeding analysis a sliding window of 12 hours is utilised. This window of data helps to ensure only novel events are collected rather than regularly occurring automated processes.

8.2.3 Graph Data Processing

To produce graphs for the clustering procedure in Sections 8.5 and 8.6, a 24 hour and 12 hour window are utilised respectively. In the case of the dynamic graph clustering, this window shifts in 15 minute bin increments resulting in 96 cluster updates per day.

8.3 Change Point Detection

During cyber attacks such as DDoS attacks, there is an abrupt influx of network traffic. These attacks attempt to disrupt the normal traffic by overwhelming the network. Provided the network data seen prior to an attack is stationary, change point detection methods could be appropriate for identifying DDoS attacks resulting in abrupt high volume changes in network activity. The suitability of the change point procedure, BFF, presented in Chapter 4 for such data is now investigated. BFF is applied to the number of new computer port edges in the Netflow data using a 12 hour sliding window with updates every 15 minutes. By monitoring such data, abrupt changes in these counts can be detected, possibly referring to anomalous network activity. The change point procedures are applied to two weeks of the data (weekdays only), but only the results for the final week of the data from day 22 (Monday) to day 26 (Friday) are displayed. As stated previously, the weekends are excluded as the behaviour on these days differs greatly from weekdays. As there are only 96 time points per day, the threshold for AFF and BFF is increased to 0.05, equivalent to approximately 5 detections per day for a well calibrated model.

Table 8.1: Number of detections for each change point method over the investigate week for the number of new computer port edges.

PELT	13
DeCAFS	5
AFF	7
BOCPD-Colmaxes	23
BOCPD-Threshold	11
BFF-Pred	13
BFF-Post-λ	14

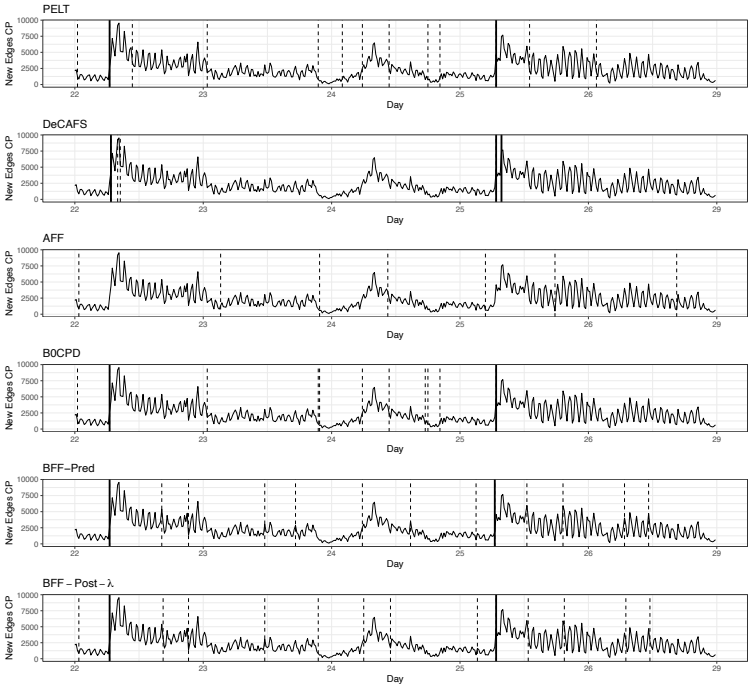


Figure 8.3: Number of new edges in CP graph over the weekday starting on day 22 of the data until day 26. Change points for each method are indicated as vertical lines where common change points (within 5 steps) between PELT, DeCAFS, BOCPD, BFF-Pred and BFF-Post-λ are indicated with a solid vertical line (while dashed lines correspond to change points that are not shared among all methods).

All methods but DeCAFS, AFF and BOCPD-Colmaxes have similar numbers of detections in Table 8.1. In agreement with the findings of the simulation study of Section 4.3, BOCPD-Colmaxes detects higher numbers of points than other procedures. AFF however detects fewer than expected. The BFF approaches have similar numbers of detections to PELT and BOCPD-Threshold which had optimal performance in previous experiments in Chapter 4.

On this data, the BOCPD-Threshold approach is utilised as it has a more favourable performance. This is in disagreement with what we have seen in the IBM analysis. Thus selection between colmaxes and threshold approaches for

BOCPD presents an additional challenge in practice.

In Figure 8.3 there are a number of clear change points in the data. In particular, the change points on days 22, 24 and 25 have been detected by the procedures (with a solid line). The detections made by the two BFF procedures are very similar however from comparing to the results of PELT, DeCAFS and BOCPD, BFF-Post-λ has more similar detections, making it more favourable. BFF-Post-λ also showed more optimal performance in previous experiments. AFF makes very few detections at the same threshold as BFF where several extreme change points in the data are not identified.

As mentioned previously in Chapter 4, the BFF methods detect anomalies in addition to change points, hence a number of the detections made differ from that of PELT, DeCAFS and BOCPD. However, BFF-Post-λ has a high similarity to PELT, a batch approach. The robust performance of the BFF procedures is demonstrated on cyber data, showcasing its suitability in this domain with similar performance to batch approaches.

8.4 Time Series Anomaly Detection

Adversaries with a foothold on the network often go unnoticed as their activity may not be extreme globally e.g. attack methodology used by APT1 [Mcwhorter 2013]. However, their activity may still be unusual given the context. Examples of such behaviour include unusual behaviour for the given user they are exploiting or attacks generating moderate volumes of traffic at unusual times of the day such as that seen from user to root attacks. During these attacks, attackers aim to gain super user access starting from normal user accounts by traversing the network by exploiting network vulnerabilities. These attacks may not result in abrupt changes in the data or have extreme volumes, thus may be undetected by change point procedures or point anomaly detectors. Detection of these non-extreme and potentially non-abrupt attacks is the aim of this section. For such attacks, contextual anomaly detection can be applied to identify when the data deviates from the usual patterns of the data. However, contextual anomaly detection assumes the network data investigated exhibits seasonal behaviour where the attacker behaviour causes deviations in the expected data patterns. Here our anomaly detection procedures proposed in Chapter 5 are applied.

The WLS data is utilised, particularly for the number of novel edges between the computers and users. Such data is expected to have a daily pattern as the behaviour of the users should follow the natural habitual working pattern of the employees and be highly similar from day to day. A 12 hour sliding window is used with minute updates, where edges not previously observed in the past 12 hours are recorded. Additionally, attention is again restricted to the weekdays. Due to incomplete data and an initialisation period for the LANL data collection process, the first and last days of the data set are excluded, leaving 64 workdays.

A persistent daily curve due to human interaction in the network is often seen

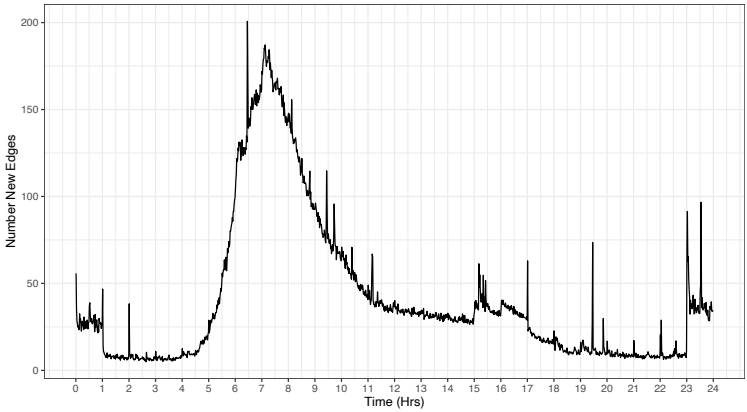


Figure 8.4: The mean number of new edges in the LANL WLS computer-user 12 hour graph that updates every minute over weekdays.

in cyber-security data such as that seen in Figure 8.4 for the mean number of new edges. As expected, this data has a daily pattern with an increase in the number of new edges when the workday begins. There is a peak in the data at around 7.15am before a slow decrease to its baseline value. Another interesting feature is the unusually high number of new edges between 11pm and 1am which is presumably due to automated system updates and processes that are carried out at the night. There is also an unexplained small peak in the number of new edges between 3-4pm and 4-5pm. The combined forecast anomaly detector is well suited to this data as a daily seasonal behaviour exists and the data itself is not particularly noisy.

The two anomaly detection procedures, SL Decomposition and FDARIMA described in Section 5.4 are applied to this LANL data and the various implementations described in 5.5.4 are compared and contrasted. Similar to the simulation from Chapter 5, only the previous $s = 60$ data points in the series are used for the ARIMA model in both algorithms which corresponds to an hour of data. This value has been chosen to ensure the computation of the ARIMA model at each minute update can be computed in real time and to allow for only the local behaviours to be modelled. The diagram in Figure 8.5 demonstrates the data split for initialising, tuning and testing the proposed anomaly detection models.

8.4.1 Forecast Performance

It can be seen in Figure 8.6 that all model forecasts except the FDA model fit the data very well. As the FDA model only uses data from the previous days to

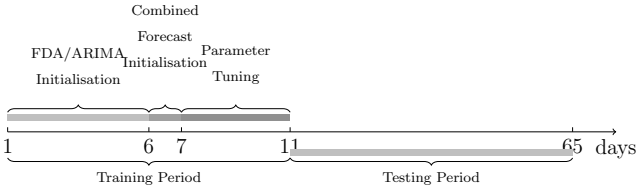


Figure 8.5: Training and testing periods for the anomaly detection procedures.

forecast the next day's values, this causes poorer forecasts than the other models which only perform one-step ahead forecasts. As noted previously, the FDA model is useful for identifying when the data deviates away from its usual behaviour. Day 23 is an example of such a day where this day experiences much less activity than any other day and is thought to correspond to a public holiday. Additionally, day 22 has an unusual peak in the afternoon between 3-6pm which is not seen in the other days however the FDA forecast can model what should be seen for that day.

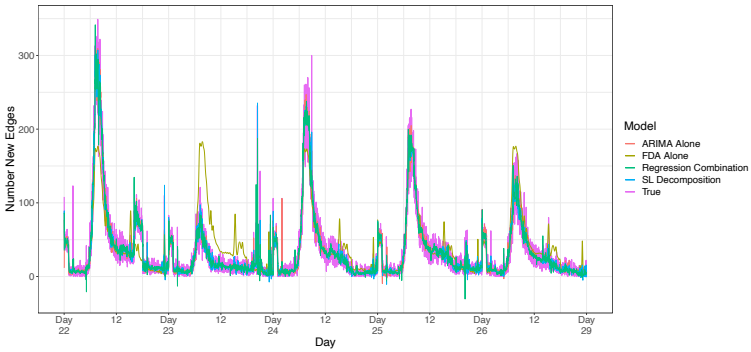


Figure 8.6: Comparison of each model to the data for days 22-29.

Over the test period, the ARIMA model is the best performer among the models and has the smallest standard deviation for both the absolute error and absolute scaled error (Table 8.2). Combining the ARIMA and FDA predictions using the proposed approaches does not improve the performance over the ARIMA alone model however they are comparable. As expected, the FDA model has poor forecast performance in comparison to the other approaches as it forecasts a full day ahead. As anomaly detection is the main goal, the FDA model has the advantage of being able to identify contextual anomalies, unlike the ARIMA model. This highlights the discrepancy between forecast ability and anomaly detection performance.

Table 8.2: Forecast performance of each model on LANL new edge count data with the standard deviation of these forecast errors over days 11-64.

	MAE	AE SD	MASE	ASE SD
ARIMA	9.42	34.04	0.86	1.74
FDA	17.65	43.33	1.61	3.94
Regression Combination	9.93	35.46	0.90	3.20
SL Decomposition	10.73	37.52	0.98	3.40
Naive One-Step Ahead Forecaster	11.44	38.33	1.00	0.00

8.4.2 Anomaly Detection Performance

Anomalous numbers of new edges have been detected using the two anomaly detection procedures, SL Decomposition and FDARIMA, after the initial 10 weekday training period. The significance level for both procedures has been set to $\epsilon = 0.005$. This significance level can be tuned by network administrators to control the number of alerts where at the chosen level 7 alerts are expected per day. In addition, the proposed procedures are compared to the Twitter algorithm which was the best performer amongst the competitive methods on the simulated data in Section 5.5 and the only other method able to detect contextual anomalies. Between FDARIMA and SL Decomposition 1652 anomalies over the full period are shared, which is more than 85% of detected anomalies by each method.

Unlike the simulated data, this real data is not labelled hence to verify the validity of the anomalies, identified anomalies will be further explored. In practice, labels often do not exist therefore procedures that can help administrators identify the source of the alert are beneficial. To better understand the anomalies detected, the computers and users related to these new edges are investigated. Here day 39 is explored as this is a highly anomalous day for the procedures and additionally day 85, a less anomalous day, is investigated. The Twitter algorithm appears to be much more sensitive than the proposed approaches with double the number of detections as seen in Table 8.3.

Table 8.3: Number of detections made by FDARIMA, SL Decomposition and Twitter anomaly detectors on the LANL new edges count for days 39 and 85. All represents the detections over the full test period (day 11-65).

Day	Method	Anomaly Count
39	FDARIMA	148
	SL Decomposition	142
	Twitter	268
85	FDARIMA	38
	SL Decomposition	36
	Twitter	96
All	FDARIMA	1935
	SL Decomposition	1789
	Twitter	7530

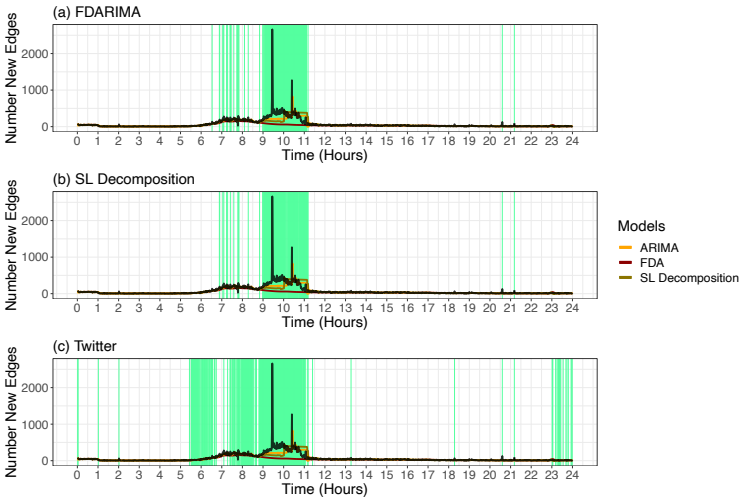


Figure 8.7: The LANL new edge count values for day 39 plotted with the forecasts of the ARIMA, FDA and SL Decomposition models where anomalous positions have been highlighted with vertical lines for the FDARIMA, SL Decomposition and Twitter algorithms separately.

The three algorithms share 126 points on day 39. An unusual secondary peak in activity can be seen in Figure 8.7 where the data normally follows the pattern of that in Figure 8.4, hence the models are correctly identifying this contextual

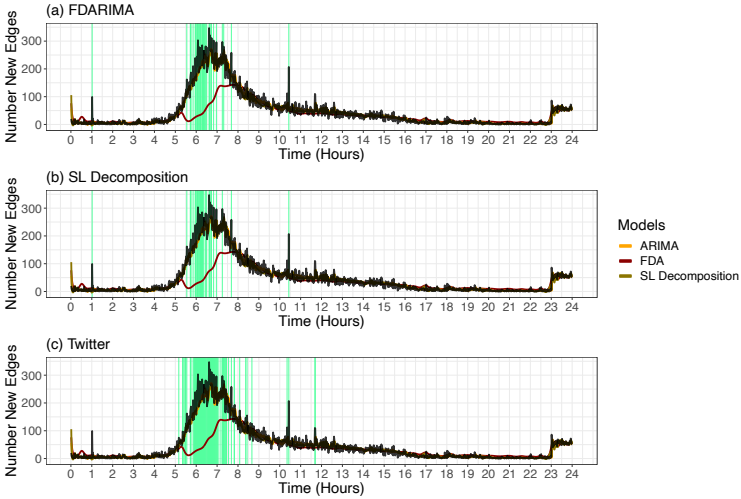


Figure 8.8: The data plotted with the predictions of the ARIMA, FDA and SL Decomposition forecasts that are used for identifying anomalies where anomalous positions have been highlighted with vertical lines for day 85 for the FDARIMA, SL Decomposition and Twitter algorithms.

anomaly. Both FDARIMA and SL Decomposition detect a number of anomalies from 6.30am onward. From further exploration of the data, the secondary peak in new edges was primarily a result of a large number of network logons made by a few select computers. The extreme point that is seen at 9.30am is caused by large numbers of network logons made by a single computer. Both instances are highly unusual therefore the anomalies detected here are of interest. It is interesting to note that the Twitter procedure detects a large number of points during the first usual peak which is not anomalous in terms of the patterns previously seen in the data. From the investigation of less anomalous days, it was seen that the Twitter method often incorrectly flags this usual daily peak as anomalous, potentially raising many false positives in practice as the data exhibits large variation in the seasonal component between the days. This highlights that although one method may perform well on simulated data, it will not necessarily have the same performance on real data which exhibits unusual patterns.

There are several anomalous points and periods throughout day 85 in Figure 8.8. Firstly the daily peak starts an hour earlier and has a larger magnitude than that of the FDA curve which represents the historic pattern of the data. All models have successfully identified this contextual anomaly. There are also a

number of point anomalies that occur such as that at 1am, 6.45am and 10.30am. All three algorithms detect the anomaly at 10.30am caused by a single computer having many network logoffs. The anomaly at 1am which is only identified by the proposed procedures, resulted from a large number of process start events in the data. As both of these activities are anomalous, the method successfully identifies point anomalies within the data with comparable performance to the Twitter algorithm, a batch approach.

8.5 Static Graph Clustering

Our attention now moves from univariate data to network graphs. As cyber-security data sets contain information about the connections of computers, ports and users, the data can naturally be expressed as network graphs. Additionally, as each entity can have multiple connections to multiple different entities, a multi-type network consisting of many graphs is formed. This section aims to investigate these network graphs to gain information about the network and its structure. To achieve this, we apply the clustering procedure developed in Chapter 6 to group nodes with similar activity together across the network.

From the LANL data three network graphs are constructed: $\mathbf{A}_{CC}$ detailing the undirected connections between the source and destination computers from the Netflow data, $\mathbf{A}_{CU}$ containing the edges between source computers (host computer) and source user (*UserName* field) in the Windows Logging Service data and $\mathbf{A}_{CP}$ details the connections between the source computers and destination ports from the Netflow data. Here the destination ports are used as they provide information about the type of interaction such as HTTP (port 80) or SSH (port 22) activity. As there are 65,535 ports, many of which are not informative of the activity, only the top 2000 ports (with the highest degrees) within the window of data explored are investigated.

Table 8.4: Summarising 24 hour window graph features of LANL data for day 22.

Number Computers	9321
Number Users	9413
Number Ports	2000
Number CC Edges	109019
Number CU Edges	44487
Number CP Edges	435684
Number Computer Clusters	300
Number User Clusters	300
Number Port Clusters	100

Table 8.5: Summarising 24 hour window graph features of LANL data for day 33.

Number Computers	7702
Number Users	7316
Number Ports	2000
Number CC Edges	63037
Number CU Edges	26633
Number CP Edges	242329
Number Computer Clusters	300
Number User Clusters	300
Number Port Clusters	100

The static multi-type clustering procedure from Chapter 6 is illustrated on a 24 hour window of data from day 22 (Monday) and day 33 (Friday). The size of the generated network graphs are described in Table 8.4 and Table 8.5 for days 22 and 33 respectively along with the input cluster sizes utilised for all algorithms. There are approximately equal numbers of computers and users within the graphs for both days however the number of edges in the CU graph is much smaller than the other graphs. The CP graphs in particular have many edges as each computer has many types of Netflow activity during the period resulting in a higher degree. It is important to note that due to the difference in the number of edges in each graph, CP may have a large influence on the clustering.

8.5.1 Day 22 Results

As the LANL data set is unlabelled, unlike the simulated example in Chapter 6, external performance measures (e.g. ARI and NMI) cannot be used. In terms of NCS, Simple NMTF outperforms the other algorithms when clustering on the CC matrix for the computers on day 22 (Table 8.6). Additionally, it consistently performs well and is generally either the second or third best method across the

Table 8.6: NCS and time (in minutes) comparison for the different methods for day 22 of the LANL data with results averaged over 50 repetitions. Best performer in red, second best in blue with standard deviation given in brackets.

	Entity	NMF	KMeans	ONMTF	F-NMTF	R-NMTF	SelDFMF	Simple NMTF
NCS	CC	0.452(0.009)	0.513(0.008)	**0.613**(0.000)	0.542(0.014)	0.587(0.028)	0.534(0.017)	**0.746**(0.002)
	CU-C	0.043(0.003)	**0.190**(0.046)	0.131(0.000)	0.042(0.003)	0.054(0.009)	0.009(0.001)	**0.176**(0.006)
	CU-U	0.579(0.011)	0.482(0.032)	**0.642**(0.000)	**0.634**(0.011)	0.529(0.016)	0.469(0.013)	0.631(0.004)
	CP-C	0.655(0.016)	0.660(0.011)	0.696(0.000)	**0.762**(0.012)	0.716(0.017)	0.618(0.025)	**0.761**(0.001)
	CP-P	0.507(0.020)	0.462(0.011)	**0.550**(0.000)	0.287(0.015)	**0.548**(0.005)	0.479(0.017)	0.399(0.002)
	Comp	0.592(0.014)	0.603(0.010)	0.654(0.000)	**0.696**(0.014)	0.668(0.015)	0.576(0.021)	**0.730**(0.001)
	Overall	0.578(0.004)	0.534(0.015)	**0.639**(0.000)	0.628(0.008)	0.593(0.009)	0.518(0.009)	**0.653**(0.002)
Time		46.349(4.741)	**5.627**(1.176)	186.042(44.595)	47.907(8.193)	101.778(0.916)	998.882(85.282)	**2.562**(0.665)

entities. In contrast, none of the other algorithm's performance have consistent behaviour between the matrices and nodes. For example, the computer clustering for F-NMTF performs very well on the CP matrix with poorer performance for the computers on the CC and CU matrices.

The cluster performance for the computers in the CU graph is poorer than the other graphs due to the nature of the data (Table 8.6). For each computer, there is a computer specific user as well as a small number of other users. Hence the similarity in this graph may be smaller than other graphs as there is less signal present in the CU graph for the computers. Additionally, the clustering patterns for the computers in this graph may differ from that in $\mathbf{A}_{CC}$ and $\mathbf{A}_{CP}$. The user clustering performance across the methods is higher suggesting there exists a signal for the user's activity in the $\mathbf{A}_{CU}$ graph.

The clear advantage of Simple NMTF is the computational speed which is significantly faster than any other algorithm, irrespective of the large magnitude of the data. For example, Simple NMTF is over 72 times faster than ONMTF, the overall second best algorithm. The computational speed and overall consistent performance of Simple NMTF on all matrices makes it more suitable for real time monitoring compared to the other methods.

8.5.2 Day 33 Results

The proposed Simple NMTF is often the best or second best performer amongst the methods for day 33 in Table 8.7. In comparison to all other methods, the procedure again has consistently high performance over all graphs. ONMTF here also has very high performance however the computation time of this approach is 25 times longer, making it less feasible in practice.

In comparison to the other multi-type clustering procedures (F-NMTF, R-NMTF and SelDFMF), Simple NMTF has superior performance and is significantly faster. Thus the proposed procedure outperforms existing multi-type methods. The results between day 33 (Table 8.7) and day 22 (Table 8.6) for Simple NMTF are very similar which suggests this procedure is robust on the LANL graphs.

Table 8.7: NCS and time (in minutes) comparison for the different methods for day 33 of the LANL data with results averaged over 20 simulations. Best performer in red, second best in blue with standard deviation given in brackets.

	Entity	NMF	KMeans	ONMTF	F-NMTF	R-NMTF	SelDFMF	Simple NMTF
NCS	CC	0.509(0.007)	0.588(0.012)	**0.646**(0.000)	0.568(0.010)	0.636(0.016)	0.605(0.032)	**0.797**(0.002)
	CU-C	0.066(0.002)	**0.202**(0.033)	0.159(0.001)	0.082(0.005)	0.090(0.016)	0.010(0.001)	**0.194**(0.005)
	CU-U	0.532(0.007)	0.416(0.019)	**0.563**(0.001)	0.538(0.006)	0.469(0.012)	0.381(0.018)	**0.558**(0.004)
	CP-C	0.792(0.011)	0.730(0.009)	0.781(0.000)	**0.833**(0.007)	0.748(0.009)	0.676(0.014)	**0.795**(0.004)
	CP-P	0.452(0.024)	0.403(0.017)	**0.553**(0.000)	0.228(0.016)	**0.529**(0.011)	0.415(0.022)	0.400(0.008)
	Comp	0.723(0.011)	0.662(0.008)	0.729(0.000)	**0.760**(0.011)	0.689(0.009)	0.622(0.013)	**0.764**(0.003)
	Overall	0.609(0.006)	0.526(0.008)	**0.637**(0.000)	0.602(0.004)	0.576(0.006)	0.494(0.012)	**0.632**(0.003)
Time		30.799(2.053)	**2.147**(0.221)	25.022(1.481)	14.868(2.611)	174.366(4.820)	648.494(13.378)	**0.973**(0.144)

This section has demonstrated the high performance of the proposed procedure, Simple NMTF, on a real cyber-security data set showing its applicability in practice. As this method is fast to compute, it is scalable for larger networks than demonstrated here.

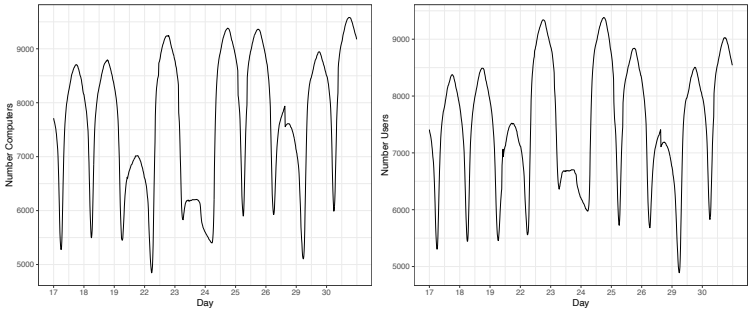
8.6 Dynamic Graph Clustering

The static clustering in the previous section can be applied to multiple snapshots of the network over time however there is no smoothness between the generated clusterings. This section aims to monitor how this clustering adapts over time, where a more dynamic approach is required as computer networks are constantly changing. The proposed dynamic clustering approach from Chapter 7 is a suitable method for this. From the dynamic clustering produced, nodes whose cluster assignments have changed can be identified, potentially corresponding to anomalous behaviour.

From the LANL data, graph streams can be generated that capture the changing behaviour of the network using a sliding window approach, similar to the simulated example in Chapter 7. A 12 hour window is utilised to represent the current activity of the network which updates every 15 minutes. More specifically, if the data is binned into 15 minute increments, each 12 hour graph contains the most recent 48 consecutive 15 minute bins. At each update, the edges from the oldest bin are removed whilst the new edges in the most recent bin are added. In the following, only the weekdays are investigated as previously discussed.

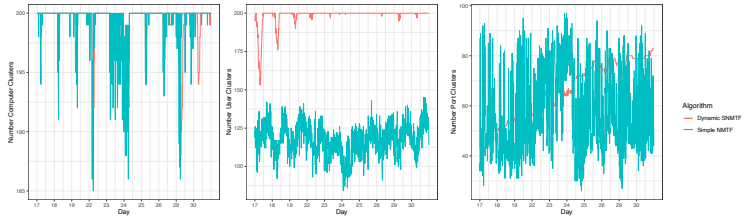
As in the static clustering case, attention is restricted to the most popular 2000 ports (those with the highest degrees) hence the number of ports within the CP graph is constant at 2000. However, it should be noted that this list of top ports may change over time. In the following, two weeks of the data is investigated from day 17 to day 30 where a 12 hour burn-in period for the dynamic clustering is used. A daily pattern can be seen in Figure 8.9 for the number of active computers and users within the 12 hour windows used to build the graphs, where there is a peak in the number of nodes around 6pm. The peak at this time is expected as it contains all activity from the working hours when the network is most busy. In both plots, day 23, a Tuesday has unusual activity. The reason for this is unknown but suggests fewer employees are using the network on this day and is possibly a public holiday (see the previous remark about the public holiday). Thus the performance on this day may differ from the others. Additionally, on Fridays (day 19 and 26) there is less activity. This is as the employees at LANL often take every other Friday off.

Although the number of clusters in the algorithms is set to 200 for the computers and users and 100 for the ports, fewer clusters are often detected by Simple NMTF and Dynamic SNMTF as seen in Figure 8.10. For the computer clustering, both methods detect similar numbers of clusters, at the allowed upper limit. This may suggest that there are more clusters than allowed. Alternatively, the number



(a) Computer **(b)** User

Figure 8.9: Number active computers and users common across the LANL data sets over the graph stream using a 12 hour sliding window over the data.



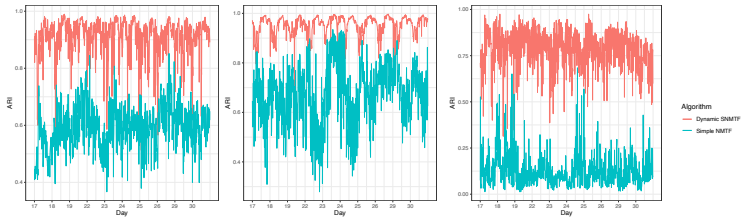
(a) Computer **(b)** User **(c)** Port

Figure 8.10: Number of clusters identified by the dynamic and static clustering approaches for the LANL network stream.

of clusters for the users differs greatly between the algorithms. The dynamic clustering finds approximately 200 clusters across the whole period whereas Simple NMTF which restarts at each update has on average 115 clusters. For the ports, both procedures have similar numbers of clusters however the number of clusters for Simple NMTF is much more noisy. This is additionally seen across all node types where the number of clusters for Dynamic SNMTF is smoother.

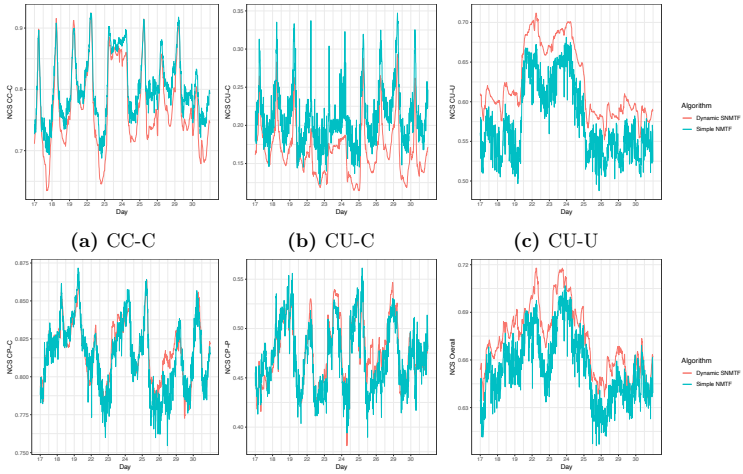
It can be seen in Figure 8.11 that Dynamic SNMTF has a higher similarity between consecutive clusterings in terms of ARI than the static version. This is ideal in practice to enable monitoring of the clustering. In particular, the similarity in consecutive port clusterings for Simple NMTF is very low, suggesting there is no smoothness between clusterings.

To assess the performance of the procedures, the proposed internal measure, the Node Cluster Similarity (NCS) is used as defined in Section 6.4.3. In general, the performance between the static and dynamic clustering methods is similar, with



(a) Computer **(b)** User **(c)** Port

Figure 8.11: ARI of consecutive clusterings for the dynamic and static clustering approaches over the LANL network stream.



(a) CC-C **(b)** CU-C **(c)** CU-U

(d) CP-C **(e)** CP-P **(f)** Overall

Figure 8.12: NCS for the calculated clusterings for the dynamic and static clustering approaches over the LANL network stream.

similar patterns (Figure 8.12). The performance for CC-C and CU-C has daily peaks just after midnight. This suggests there is a greater signal in the graphs at this time which the proposed procedure can detect. For these graphs, Simple NMTF has better performance, but the difference is marginal. For the users, Dynamic SNMTF has superior performance to Simple NMTF. This difference in performance may however be attributed to the difference in the number of clusters found, where having more clusters may result in more similarity between the nodes in each cluster.

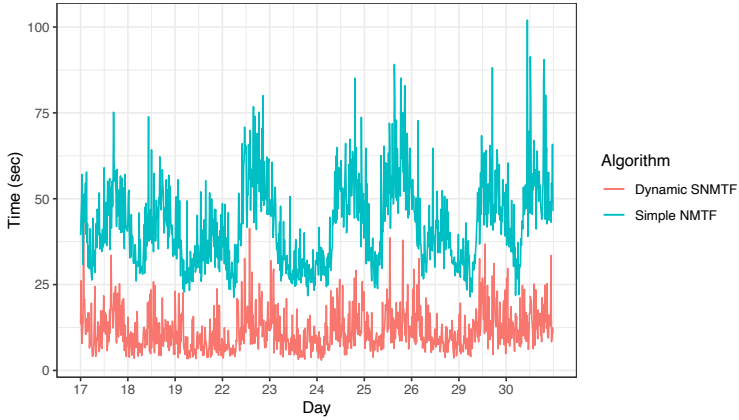


Figure 8.13: Computation time at each update for the dynamic and static clustering approaches on the LANL network stream.

The performance for the computers in the CP graph (CP-C) does not follow the daily pattern seen for CC-C and CU-C in Figure 8.12. This may suggest that this graph is less affected by the human interactions on the network. As noted previously, day 23 has an unusual activity that has influenced the results in Figure 8.12. Generally, over all nodes and graphs, the NCS performance is higher on this day. This may suggest the clustering signal is clearer when there are fewer user related events on the network. For the overall NCS performance in Figure 8.12 (f), Dynamic SNMTF has higher performance. This suggests utilising the previous clustering improves the clustering performance. In addition, the results for Dynamic SNMTF are much smoother than Simple NMTF for all entity types which is desirable in practice.

Finally, the computation time of the two approaches is compared. In practice, more regular updates (e.g. every minute) could be utilised provided the clustering algorithm can appropriately update in that time frame. In Figure 8.13 it is clear that Dynamic SNMTF is very fast with less variability throughout the investigated period. This makes it more suitable than Simple NMTF, the static clustering procedure.

Overall, this section has demonstrated that the proposed Dynamic SNMTF can be applied to real cyber graph data streams to find the clustering of the network in close to real time. To update the clustering, only the cluster assignment from the previous time is required with a small computation storage overhead. It has additionally been shown that utilising Dynamic SNMTF over Simple NMTF does not cause a reduction in performance, where overall, the performance is improved

in terms of smoothness between clusters, computation time and NCS.

8.7 Discussion

This chapter has demonstrated the applicability of the proposed monitoring procedures on a real enterprise data set. Although labels of anomalous activity are not available, the performance of the procedures using alternate measures are performed which primarily investigate the model fit. In many applications, the primary focus is on monitoring the number of new edges, however other indicators of intrusion could be utilised such as monitoring the volume of activity for each entity (computer, user or port) individually.

From applying the proposed change point procedure to the LANL data it is evident BFF has similar performance to batch approaches. Additionally, for this data, the threshold approach for BOCPD had more favourable results over colmaxes which contrasts with the results from the IBM analysis in Chapter 4. In the case of anomaly detection, both the proposed FDARIMA and SL Decomposition methods detect similar points to the Twitter method, a batch approach, showing their suitability in practice. The Twitter method however flags large numbers of alerts in the data due to the variation in the seasonal component between the days.

On static graphs, Simple NMTF showed high clustering performance in terms of the proposed NCS measure where for day 22, Simple NMTF had the best clustering performance overall. In addition, the computation time for the proposed procedure is significantly faster than any competitive approach. The dynamic version of this procedure produced smooth clustering on the LANL network with minimal reduction in clustering performance in comparison to the static method. Additionally, Dynamic SNMTF provides an improvement in the computational speed over the static version. Further investigation into the cluster evolution of the clustering could be performed to assist in identifying intruders in the network but is beyond the scope of the thesis.

In summary, it has been shown that all of the proposed procedures presented are fast and scalable, making them suitable for application to large scale enterprise cyber-security networks with high performance. Additionally, all approaches are adaptable, making them suitable for data streams. The methods presented provide a wide range of tools that can be used to monitor the network for improved situational awareness and detection of intruders.

Chapter 9

Conclusion

Computer networks produce vast numbers of high frequency data streams that can be monitored for the detection of intruders. As the number and sophistication of attacks increases, new methods of detection are required to complement existing signature-based procedures. In this thesis, statistical monitoring and intrusion detection procedures are proposed to assist analysts to identify abnormal activities in the network. Due to the high volume, frequency and ever changing nature of cyber data, the procedures proposed are adaptive, autonomous, sequential/online and computationally fast for real time analysis.

A comprehensive set of monitoring tools are provided for both situational awareness and intrusion detection in enterprise computer networks. The first set of tools are for the analysis of univariate ordered data streams. A clustering tool is additionally provided to group nodes based on their activity across all network data sets. Each of the methods have been validated on real enterprise data and the contributions for each piece of work are now described in more detail.

To monitor the changing behaviour of a data stream, the novel Bayesian Forgetting Factor (BFF) adaptive estimator with sequential updating forms is proposed in Chapter 3 which is suitable for applications to high frequency data streams. Forgetting factors are incorporated in the model to introduce a temporal aspect where, unlike the frequentist counterpart, estimation of this forgetting factor does not require hard to set hyper-parameters. A prior updating mechanism is additionally proposed such that prior estimates have a constant influence over the stream to improve the adaptability of the method. From the experiments performed, the proposed BFF procedure has consistent performance, unlike competitive approaches which are highly sensitive to the choice of hyper-parameter. Additionally, the estimates of the procedure adapt swiftly when changes in the data generating process occur.

Abrupt changes may exist in the data stream, potentially corresponding to adversaries, where detection of such abnormalities is the focus of Chapter 4. The proposed novel change point method utilises the estimated posterior distribution from the adaptive estimator in Chapter 3 to calculate p-values for newly observed

data points. Through calibration of these p-values, setting the change point threshold becomes intuitive unlike competitive approaches such as AFF and BOCPD. An unequivocal contribution of this method is its ability to detect abrupt changes in the presence of trend with a low false positive rate. These qualities are desirable to prevent an unproductive and costly investigation by human analysts on falsely identified change points. BFF shows comparable performance to well regarded benchmark procedures in the field on simulated and real data. When applied to real IBM stock open prices, BFF shows high similarity in the detections to DeCAFS, a batch procedure, with small computation times in comparison to BOCPD, the highest performing sequential procedure on simulated examples. Criticisms of BOCPD are also provided to highlight its practical flaws, namely long computation times, detection lags and hard to set hyper-parameters that BFF does not face.

Data streams are often subject to regular patterns or seasonality. The proposed combined forecast methods and associated anomaly detection procedures in Chapter 5 are well suited for such data. Unlike the majority of existing approaches, the proposed methods, FDARIMA and SLD are capable of detecting both point and contextual anomalies through fusing the short-term and long-term behaviours of the series using an adaptive framework. Additionally, the proposed methods can update sequentially with fixed storage requirements. From application to simulated data, the proposed procedures, FDARIMA and SLD, outperform existing approaches with high similarity to the best *batch* approach, the Twitter procedure. However, during application to enterprise network data in Chapter 8 it was evident the Twitter method was not suitable as it was unable to capture the daily pattern of the data, labelling it as anomalous. The proposed procedures show promising results for the online detection of anomalies with low false positive rates.

Computer networks can also naturally be expressed as multiple network graphs where the work in Chapter 6 looks to find the underlying relationships between the network entities using clustering. Simple NMTF, a Non-Negative Matrix Tri-Factorisation is proposed fusing information from multiple network graphs to produce a single clustering for each node type. Non-negative factor matrices within the factorisation are replaced with binary cluster indicator matrices to improve both computational speed and interpretability, the key benefit of this method. This simplification significantly reduces the computational load, making it scalable for large computer networks. A novel weighted version of the procedure is additionally explored which allows for a deeper understanding of the contribution of each graph towards the final clustering. As multi-type clustering is a relatively unexplored approach, initialisation and cluster validation methods are not well established in this field. We propose an extension of the popular uni-partite clustering initialisation procedure, NNDSVD, for multi-type relational clustering. From simulations, it is shown to improve the stability of the clustering in comparison to random initialisation without compromising cluster performance. As internal

clustering performance measures for multi-type relational clustering do not currently exist, NCS is proposed which can assess the clustering at the node, graph and network levels showing similar performance rankings to established external performance measures. The clustering procedure is applied to both simulated and real data. The proposed procedure shows consistently high performance and notably fast computation times, outperforming the existing multi-type relational clustering procedures.

To our knowledge, dynamic multi-type relational clustering methods have not yet been developed for clustering on graph streams. The proposed clustering, Simple NMTF, is extended for application to graph streams in Chapter 7 to fill this research gap. The proposed dynamic clustering procedure, Dynamic SNMTF, utilises the calculated clustering from the previous time to initialise Simple NMTF rather than using the proposed NNDSVD initialisation procedure at each update. From application to simulated graph streams using a stochastic block model, the proposed dynamic clustering method improves computation speed and smoothness between consecutive clusterings without significantly compromising the clustering performance in comparison to the static version of the procedure.

Finally, in Chapter 8 all proposed approaches are applied to a real enterprise network data set to test the suitability of the methods for such application. Although the data does not contain labels for intrusions or the underlying clustering, the performance is assessed via comparison to highly regarded methods or through the use of internal measures. This chapter shows the proposed procedures are suitable for enterprise network monitoring with high similarity to batch and well regarded methods. In addition, the procedures are quick to compute with fixed storage requirements, making them practical.

The proposed statistical network monitoring and intrusion detection procedures presented in this thesis are important in practice for identifying intruders with a foothold on the network. The tools provided give analysts a comprehensive tool set for identifying a number of attack vectors such as large volume attacks which can be identified using the proposed change point procedure, contextual anomalies within the data can be detected using the proposed anomaly detection procedure and finally understanding about the underlying grouping of the nodes based on their activity is given by the Simple NMTF procedure. As adversaries are constantly changing their attack vectors it is important to monitor the data using such a wide range of approaches to prevent adversaries from avoiding detection in the future.

Future Work

To conclude, possible extensions and future work directions are now presented for the methods proposed in this thesis. Within the Bayesian adaptive estimation procedure in Chapter 3, a clear relationship between the estimated forgetting factor and the variance of the data is observed. This relationship gives insight into the workings of the forgetting factor within the model. Further exploration of

this relationship to better understand this adaptive forgetting factor could be investigated. From the analysis performed, this relationship suggests that for the given variance of the data, an appropriate forgetting factor can be assigned, which is a novel view of this factor.

The change point method in Chapter 4 is presented for univariate data streams however this can readily be extended to the multivariate Gaussian case. Change point detection on multivariate data streams is an exceptionally challenging research problem where exploration of this issue is beyond the scope of this thesis. Whilst initial experiments for this problem have been conducted, several challenges were presented which include identifying change points when only small subsets of the dimensions observe a change and determining whether a universal forgetting factor over all dimensions or an individual factor for each is more appropriate.

Finally, although Simple NMTF provides a tool for improved awareness of node activities and groupings of these nodes, this work can be extended for anomaly detection. In practice, further understanding of the calculated clusterings may be available from obtaining additional features for the computers and users (such as job descriptions/computer functionality) and can be used to categorise the clusters found and potentially validate them. There is a wide range of possible avenues for identifying anomalies from the clustering results at the node, edge, graph and network level. Firstly cluster changes can be monitored. For example, investigation of nodes that move between low degree to high degree clusters or clusters with very little similarity can be identified. The anomalousness of new edges can also be monitored. From the clustering, the probability of the edge between the respective clusters (that the nodes are assigned to) can be calculated where unusual edges are flagged. Cluster evolution analysis as done by Landauer et al. [2018] can additionally be performed to identify anomalies in the data. Thus there is a large potential for intrusion detection from the network clustering performed.

Bibliography

Ryan Prescott Adams and David J. C. MacKay. Bayesian Online Changepoint Detection. 2007. URL https://arxiv.org/abs/0710.3742. arXiv: 0710.3742.

Charu C. Aggarwal. *Data Streams*, volume 31 of *Advances in Database Systems*. Springer US, Boston, MA, 2007. doi:10.1007/978-0-387-47534-9.

Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016. doi:10.1016/j.jnca.2015.11.016.

Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph Based Anomaly Detection and Description: A Survey. *Data Mining and Knowledge Discovery*, 29 (3):626–688, 2015. doi:10.1007/s10618-014-0365-y.

Samaneh Aminikhanghahi and Diane J. Cook. A Survey of Methods for Time Series Change Point Detection. *Knowledge and Information Systems*, 51(2): 339–367, 2017. doi:10.1007/s10115-016-0987-z.

Christoforos Anagnostopoulos, Dimitris K. Tasoulis, Niall M. Adams, Nicos G. Pavlidis, and David J. Hand. Online Linear and Quadratic Discriminant Analysis with Adaptive Forgetting for Streaming Classification. *Statistical Analysis and Data Mining*, 5(2):139–166, 2012. doi:10.1002/sam.10151.

David Arthur and Sergei Vassilvitskii. K-means++: The Advantages of Careful Seeding. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, 2007. ISBN 9780898716245.

Sitaram Asur, Srinivasan Parthasarathy, and Duygu Ucar. An Event-Based Framework for Characterizing the Evolutionary Behavior of Interaction Graphs. *ACM Transactions on Knowledge Discovery from Data*, 3(4):1–36, 2009. doi:10.1145/1631162.1631164.

Syed Muhammad Atif, Sameer Qazi, and Nicolas Gillis. Improved SVD-based Initialization for Nonnegative Matrix Factorization using Low-Rank Correction. *Pattern Recognition Letters*, 122:53–59, 2019. doi:10.1016/j.patrec.2019.02.018.

Arindam Banerjee, Sugato Basu, and Srujana Merugu. Multi-Way Clustering on Relation Graphs. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 145–156. Society for Industrial and Applied Mathematics, 2007. doi:10.1137/1.9781611972771.14.

Rafal Baranowski, Yining Chen, and Piotr Fryzlewicz. Narrowest-Over-Threshold Detection of Multiple Change Points and Change-Point-Like Features. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 81(3):649–672, 2019. doi:10.1111/rssb.12322.

Michael J. Barber. Modularity and Community Detection in Bipartite Networks. *Physical Review E*, 76(6):066102, 2007. doi:10.1103/PhysRevE.76.066102.

Jarosław Bernacki and Grzegorz Kołaczek. Anomaly Detection in Network Traffic Using Selected Methods of Time Series Analysis. *International Journal of Computer Network and Information Security*, 7(9):10–18, 2015. doi:10.5815/ijcnis.2015.09.02.

Kelly Bissell, Ryan M. Lasalle, and Paolo Dal Cin. The Cost of Cybercrime. Technical report, Accenture Security and Ponemon Institue LLC, 2019. URL `https://www.accenture.com/us-en/insights/security/cost-cybercrime-study`.

Dean Bodenham. *ffstream: Forgetting Factor Methods for Change Detection in Streaming Data*, 2018. URL `https://CRAN.R-project.org/package=ffstream`. R package version 0.1.6.

Dean A. Bodenham. *Adaptive Estimation with Change Detection for Streaming Data*. PhD thesis, Imperial College London, Department of Mathematics, 2014.

Dean A. Bodenham and Niall M. Adams. Continuous Monitoring for Changepoints in Data Streams using Adaptive Estimation. *Statistics and Computing*, 27(5): 1257–1270, 2017. doi:10.1007/s11222-016-9684-8.

Christos Boutsidis and Efstratios Gallopoulos. SVD Based Initialization: A Head Start for Nonnegative Matrix Factorization. *Pattern Recognition*, 41(4):1350–1362, 2008. doi:10.1016/j.patcog.2007.09.010.

Chuck Brooks. Alarming Cybersecurity Stats: What You Need To Know For 2021, 2021. URL `https://www.forbes.com/sites/chuckbrooks/2021/03/02/alarming-cybersecurity-stats-------what-you-need-to-know-for-2021/?sh=132a7e8658d3`.

Alina Câmpan and Gabriela Åerban. Adaptive Clustering Algorithms. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4013 LNAI, pages 407–418. Springer Verlag, 2006. doi:10.1007/11766247_35.

Bin Cao, Dou Shen, Jian Tao Sun, Xuanhui Wang, Qiang Yang, and Zheng Chen. Detect and Track Latent Factors with Online Nonnegative Matrix Factorization. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 2689–2694, 2007. doi:10.5555/1625275.1625708.

Yanshuai Cao and Luyu Wang. Automatic Selection of t-SNE Perplexity. 2017. URL `https://arxiv.org/abs/1708.03229`. arXiv: 1708.03229.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly Detection: A Survey. In *ACM Computing Surveys*, volume 41, pages 1–58. 2009. doi:10.1145/1541880.1541882.

Chung Chen and Lon-Mu Liu. Joint Estimation of Model Parameters and Outlier Effects in Time Series. *Journal of the American Statistical Association*, 88(421): 284, 1993. doi:10.2307/2290724.

Roland Chrobok, Oliver Kaumann, Joachim Wahle, and Michael Schreckenberg. Different Methods of Traffic Forecast Based on Real Data. In *European Journal of Operational Research*, volume 155, pages 558–568. North-Holland, 2004. doi:10.1016/j.ejor.2003.08.005.

Robert B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma Terpenning. STL: A Seasonal-Trend Decomposition Procedure Based on Loess. *Journal of Official Statistics*, 6:3–73, 1990.

William S Cleveland and Susan J Devlin. Locally-Weighted Fitting: An Approach to Fitting Analysis by Local Fitting. *Journal of the American Statistical Association*, 83(403):596–610, 1988.

Gari D. Clifford, Ikaro Silva, Benjamin Moody, Qiao Li, Danesh Kella, Abdullah Shahin, Tristan Kooistra, Diane Perry, and Roger G. Mark. The PhysioNet/Computing in Cardiology Challenge 2015: Reducing False Arrhythmia Alarms in the ICU. In *2015 Computing in Cardiology Conference (CinC)*, pages 273–276. IEEE, 2015. doi:10.1109/CIC.2015.7408639.

Andrej Čopar, Blaž Zupan, and Marinka Zitnik. Fast Optimization of Non-Negative Matrix Tri-Factorization. *PLOS ONE*, 14(6), 2019. doi:10.1371/journal.pone.0217994.

Matt Dancho and Davis Vaughan. *anomalize: Tidy Anomaly Detection*, 2020. URL `https://cran.r-project.org/package=anomalize`. R package version 0.2.2.

Mikhail Dashevskiy and Zhiyuan Luo. Network Traffic Demand Prediction with Confidence. In *2008 IEEE Global Telecommunications Conference*, pages 1–5. IEEE, 2008. doi:10.1109/GLOCOM.2008.ECP.284.

David L. Davies and Donald W. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979. doi:10.1109/TPAMI.1979.4766909.

Anthony C. Davison and David V. Hinkley. *Bootstrap Methods and their Application*. Cambridge University Press, 1997. doi:10.1017/CBO9780511802843.

Jan G. De Gooijer and Rob J. Hyndman. 25 Years of Time Series Forecasting. *International Journal of Forecasting*, 22(3):443–473, 2006. doi:10.1016/j.ijforecast.2006.01.001.

Nicoletta Del Buono and Gianvito Pio. Non-Negative Matrix Tri-Factorization for Co-Clustering: An Analysis of the Block Matrix. *Information Sciences*, 301: 13–26, 2015. doi:10.1016/j.ins.2014.12.058.

Frédéric Desobry, Manuel Davy, and Christian Doncarli. An Online Kernel Change Detection Algorithm. *IEEE Transactions on Signal Processing*, 53(8):2961–2974, 2005. doi:10.1109/TSP.2005.851098.

Chris Ding, Xiaofeng He, and Horst D. Simon. On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 606–610. Society forIndustrial and Applied Mathematics, 2005. doi:10.1137/1.9781611972757.70.

Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal Nonnegative Matrix Tri-Factorizations for Clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '06*, page 126, 2006. doi:10.1145/1150402.1150420.

Nan Ding, Huanbo Gao, Hongyu Bu, Haoxuan Ma, and Huaiwei Si. Multivariate-Time-Series-Driven Real-time Anomaly Detection Based on Bayesian Network. *Sensors*, 18(10):3367, 2018. doi:10.3390/s18103367.

J. C. Dunn. Well-Separated Clusters and Optimal Fuzzy Partitions. *Journal of Cybernetics*, 4(1):95–104, 1974. doi:10.1080/01969727408546059.

Carl Eckart and Gale Young. The Approximation of One Matrix by Another of Lower Rank. *Psychometrika*, 1(3):211–218, 1936. doi:10.1007/BF02288367.

Paul Fearnhead. Exact and Efficient Bayesian Inference for Multiple Changepoint Problems. *Statistics and Computing*, 16(2):203–213, 2006. doi:10.1007/s11222-006-8450-8.

Paul Fearnhead, Robert Maidstone, and Adam Letchford. Detecting Changes in Slope With an L0 Penalty. *Journal of Computational and Graphical Statistics*, 28(2):265–275, 2019. doi:10.1080/10618600.2018.1512868.

Ronald A. Fisher. *Statistical Methods for Research Workers*. Number V. 1925. doi:10.1056/NEJMc061160.

Piotr Fryzlewicz. Wild Binary Segmentation for Multiple Change-Point Detection. *The Annals of Statistics*, 42(6):2243–2281, 2014. doi:10.1214/14-AOS1245.

Colin Gallagher, Robert Lund, and Michael Robbins. Changepoint Detection in Climate Time Series with Long-Term Trends. *Journal of Climate*, 26(14):4994–5006, 2013. doi:10.1175/JCLI-D-12-00704.1.

João Gama. *Knowledge Discovery from Data Streams*. Chapman and Hall, Boca Raton, 2010. ISBN 978-1-4398-2611-9.

G Gardner, Andrew C. Harvey, and Garry D. Phillips. Algorithm AS 154: An Algorithm for Exact Maximum Likelihood Estimation of Autoregressive-Moving Average Models by Means of Kalman Filtering. *Applied Statistics*, 29(3):311–322, 1980.

Ryan Garnett, Michael A. Osborne, Steven Reece, Alex Rogers, and Stephen J. Roberts. Sequential Bayesian Prediction in the Presence of Changepoints and Faults. *The Computer Journal*, 53(9):1430–1446, 2010. doi:10.1093/comjnl/bxq003.

Andrew Gelman. Two Simple Examples for Understanding Posterior P-Values whose Distributions are Far from Uniform. *Electronic Journal of Statistics*, 7 (1):2595–2602, 2013. doi:10.1214/13-EJS854.

Robert Görke, Pascal Maillard, Andrea Schumm, Christian Staudt, and Dorothea Wagner. Dynamic Graph Clustering Combining Modularity and Smoothness. *ACM Journal of Experimental Algorithmics*, 18(1), 2013. doi:10.1145/2444016.2444021.

Derek Greene and Pádraig Cunningham. A Matrix Factorization Approach for Integrating Multiple Data Views. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 423–438. Springer, Berlin, Heidelberg, 2009. doi:10.1007/978-3-642-04180-8_45.

Derek Greene, Dónal Doyle, and Pádraig Cunningham. Tracking the Evolution of Communities in Dynamic Social Networks. In *2010 International Conference on Advances in Social Networks Analysis and Mining*, pages 176–183. IEEE, 2010. doi:10.1109/ASONAM.2010.17.

Jonathan L. Gross and Jay Yellen. *Handbook of Graph Theory*. CRC Press, 2003. ISBN 978-1439880180.

Roger Guimerà, Marta Sales-Pardo, and Luís A. Nunes Amaral. Module Identification in Bipartite and Directed Networks. *Physical Review E*, 76(3):036102, 2007. doi:10.1103/PhysRevE.76.036102.

Fredrik Gustafsson. The Marginalized Likelihood Ratio Test for Detecting Abrupt Changes. *IEEE Transactions on Automatic Control*, 41(1):66–78, 1996. doi:10.1109/9.481608.

Fredrik Gustafsson. *Adaptive Filtering and Change Detection*. John Wiley & Sons, Ltd, Chichester, UK, 2001. ISBN 9780470841617. doi:10.1002/0470841613.

Zaïd Harchaoui, Eric Moulines, and Francis Bach. Kernel Change-point Analysis. In D Koller, D Schuurmans, Y Bengio, and L Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21, pages 609–616. Curran Associates, Inc., 2009.

Douglas M. Hawkins. Cumulative Sum Control Charting: An Underutilised SPC Tool. *Quality Engineering*, 5(3):463–477, 1993. doi:10.1080/08982119308918986.

Simon S. Haykin. *Adaptive Filter Theory*. Prentice Hall, 2002. ISBN 9780273764106.

Nicholas A. Heard and Patrick Rubin-Delanchy. Network-wide anomaly detection via the dirichlet process. In *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, pages 220–224, 2016. doi:10.1109/ISI.2016.7745478.

Nicholas A. Heard and Patrick Rubin-Delanchy. Choosing between methods of combining p-values. *Biometrika*, 105(1):239–246, 2018. doi:10.1093/BIOMET/ASX076.

Monika R. Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. *Computing on Data Streams*, pages 107–118. American Mathematical Society, USA, 1999. ISBN 0821811843.

David J. Hill and Barbara S. Minsker. Anomaly Detection in Streaming Environmental Sensor Data: A Data-Driven Modeling Approach. *Environmental Modelling & Software*, 25(9):1014–1022, 2010. doi:10.1016/j.envsoft.2009.08.010.

Jordan Hochenbaum, Owen S. Vallis, and Arun Kejariwal. Automatic Anomaly Detection in the Cloud Via Statistical Learning. 2017. URL `https://arxiv.org/abs/1704.07706`. arXiv: 1704.07706.

Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic Blockmodels: First Steps. *Social Networks*, 5(2):109–137, 1983. doi:10.1016/0378-8733(83)90021-7.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985. doi:10.1007/BF01908075.

Rob J. Hyndman. Another Look At Forecast-Accuracy Metrics for Intermittent Demand. *Foresight: The International Journal of Applied Forecasting*, (4):43–46, 2006. doi:10.1.1.218.7816.

Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Melbourne, Australia, 2nd edition, 2018. ISBN 9780987507112.

Rob J. Hyndman and Yeasmin Khandakar. Automatic Time Series Forecasting: The forecast Package for R. *Journal of Statistical Software*, 26(3), 2008. doi:10.18637/jss.v027.i03.

Rob J. Hyndman and Anne B. Koehler. Another Look at Measures of Forecast Accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006. doi:10.1016/j.ijforecast.2006.03.001.

Joseph G. Ibrahim and Ming-Hui Chen. Power Prior Distributions for Regression Models. *Statistical Science*, 15(1):46–60, 2000. doi:10.1214/ss/1009212673.

Joseph G. Ibrahim, Ming-Hui Chen, Yeongjin Gwon, and Fang Chen. The Power Prior: Theory and Applications. *Statistics in Medicine*, 34(28):3724–3749, 2015. doi:10.1002/sim.6728.

Vladislav Ishimtsev, Ivan Nazarov, Alexander Bernstein, and Evgeny Burnaev. Conformal k-NN Anomaly Detector for Univariate Data Streams. *Proceedings of Machine Learning Research*, 60:1–15, 2017.

Alaiñe Iturria, Jacinto Carrasco, Santi Charramendieta, Angel Conde, and Francisco Herrera. otsad: A package for Online Time-Series Anomaly Detectors. *Neurocomputing*, 374:49–53, 2020. doi:10.1016/j.neucom.2019.09.032.

Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, 1988. ISBN 978-0-13-022278-7.

Keisuke Kato and Vitaly Klyuev. Large-scale Network Packet Analysis for Intelligent DDoS Attack Detection Development. In *The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014)*, pages 360–365. IEEE, 2014. doi:10.1109/ICITST.2014.7038838.

Yoshinobu Kawahara and Masashi Sugiyama. Sequential Change-Point Detection based on Direct Density-Ratio Estimation. *Statistical Analysis and Data Mining*, 5(2):114–127, 2012. doi:10.1002/sam.10124.

Daniel Keim, Jörn Kohlhammer, Geoffrey Ellis, and Florian Mansmann. *Mastering the Information Age: Solving Problems with Visual Analytics*. Goslar: Eurographics Association, Germany, 2010. doi:10.2312/14803.

Arun Kejariwal. Introducing Practical and Robust Anomaly Detection in a Time Series, 2015. URL `https://blog.twitter.com/engineering/en_us/a/2015/introducing-practical-and-robust-anomaly-detection-in-a-time-series.html`.

Alexander D. Kent. Comprehensive, Multi-Source Cyber-Security Events. Los Alamos National Laboratory, 2015. URL http://dx.doi.org/10.17021/1179829.

Rebecca Killick, Paul Fearnhead, and Idris A. Eckley. Optimal Detection of Changepoints With a Linear Computational Cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012. doi 10.1080/01621459.2012.737745.

Rebecca Killick, Kaylea Haynes, and Idris A. Eckley. *changepoint: An R package for changepoint analysis*, 2016. URL https://CRAN.R-project.org/package=changepoint. R package version 2.2.2.

Abhishek Kumar, Piyush Rai, and Hal Daumé III. Co-regularized Multi-view Spectral Clustering. In *NIPS'11 Proceedings of the 24th International Conference on Neural Information Processing Systems*, pages 1413–1421, 2011.

Ravi Kumar, Jasmine Novak, and Andrew Tomkins. Structure and Evolution of Online Social Networks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '06*, page 611. ACM Press, 2006. doi 10.1145/1150402.1150476.

Tze Leung Lai. Sequential Changepoint Detection in Quality Control and Dynamical Systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(4):613–644, 1995. doi:10.1111/j.2517-6161.1995.tb02052.x.

Max Landauer, Markus Wurzenberger, Florian Skopik, Giuseppe Settanni, and Peter Filzmoser. Dynamic Log File Analysis: An Unsupervised Cluster Evolution Approach for Anomaly Detection. *Computers & Security*, 79:94–116, 2018. doi 10.1016/j.cose.2018.08.009.

Ken Lang. NewsWeeder: Learning to Filter Netnews. In *Machine Learning Proceedings 1995*, pages 331–339. 1995.

Alexander Lavin and Subutai Ahmad. Evaluating Real-Time Anomaly Detection Algorithms - The Numenta Anomaly Benchmark. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 38–44, 2015. doi:10.1109/ICMLA.2015.141.

Rikard Laxhammar and Göran Falkman. Sequential Conformal Anomaly Detection in Trajectories based on Hausdorff Distance. In *14th International Conference on Information Fusion*, pages 1–8, 2011.

Daniel D Lee and H Sebastian Seung. Algorithms for Non-Negative Matrix Factorization. In *Advances in Neural Information Processing Systems*, pages 556–562, 2001. doi:10.1109/IJCNN.2008.4634046.

Li Li, Xiaonan Su, Yi Zhang, Yuetong Lin, and Zhiheng Li. Trend Modeling for Traffic Time Series Analysis: An Integrated Study. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3430–3439, 2015. doi:10.1109/TITS.2015.2457240.

Jialu Liu, Chi Wang, Jing Gao, and Jiawei Han. Multi-View Clustering via Joint Nonnegative Matrix Factorization. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 252–260. 2013. doi:10.1137/1.9781611972832.28.

Kai Liu and Hua Wang. High-Order Co-Clustering via Strictly Orthogonal and Symmetric L1-Norm Nonnegative Matrix Tri-Factorization. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 2454–2460, 2018. doi:10.24963/ijcai.2018/340.

Xin Liu, Hui-Min Cheng, and Zhong-Yuan Zhang. Evaluation of Community Detection Methods. *IEEE Transactions on Knowledge and Data Engineering*, pages 1736–1746, 2019. doi:10.1109/TKDE.2019.2911943.

Bo Long, Zhongfei (Mark) Zhang, and Philip S Yu. Co-Clustering by Block Value Decomposition. In *Proceeding of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining - KDD '05*, page 635. ACM Press, 2005. doi:10.1145/1081870.1081949.

Bo Long, Zhongfei Mark Zhang, and Philip S. Yu. A Probabilistic Framework for Relational Clustering. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '07*, page 470. ACM Press, 2007. doi:10.1145/1281192.1281244.

Javier López-de Lacalle. *tsoutliers: Detection of Outliers in Time Series*, 2019. URL `https://cran.r-project.org/package=tsoutliers`. R package version 0.6-8.

James M. Lucas and Michael S. Saccucci. Exponentially Weighted Moving Average Control Schemes: Properties and Enhancements. *Technometrics*, 32(1):1–12, 1990. doi:10.1080/00401706.1990.10484583.

Matthew Ludkin, Idris Eckley, and Peter Neal. Dynamic Stochastic Block Models: Parameter Estimation and Detection of Changes in Community Structure. *Statistics and Computing*, 28(6):1201–1213, 2018. doi:10.1007/s11222-017-9788-9.

Xiaoke Ma and Di Dong. Evolutionary Nonnegative Matrix Factorization Algorithms for Community Detection in Dynamic Networks. *IEEE Transactions on Knowledge and Data Engineering*, 29(5):1045–1058, 2017. doi:10.1109/TKDE.2017.2657752.

Grzegorz Marcjasz, Bartosz Uniejewski, and Rafał Weron. On the Importance of the Long-Term Seasonal Component in Day-Ahead Electricity Price Forecasting with NARX Neural Networks. *International Journal of Forecasting*, 35(4):1520–1532, 2019. doi:10.1016/j.ijforecast.2017.11.009.

Catherine Matias and Vincent Miele. Statistical Clustering of Temporal Networks Through a Dynamic Stochastic Block Model. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 79(4):1119–1141, 2017. doi:10.1111/rssb.12200.

Dan Mcwhorter. APT1: Exposing One of China's Cyber Espionage Units. Technical report, Mandiant, 2013. URL `https://www.mandiant.com/resources/apt1-exposing-one-of-chinas-cyber-espionage-units`.

Xiao-Li Meng. Posterior Predictive p-Values. *The Annals of Statistics*, 22(3): 1142–1160, 1994. doi:10.1214/aos/1176325622.

Andrew V. Metcalfe and Paul S. P. Cowpertwait. *Introductory Time Series with R*. Springer New York, New York, 2009. doi:10.1007/978-0-387-88698-5.

Silvia Metelli and Nicholas A. Heard. On Bayesian New Edge Prediction and Anomaly Detection in Computer Networks. *The Annals of Applied Statistics*, 13(4):2586–2610, 2019. doi:10.1214/19-AOAS1286.

Ana Militino, Mehdi Moradi, and M. Ugarte. On the Performances of Trend and Change-Point Detection Methods for Remote Sensing Data. *Remote Sensing*, 12(6):1008, 2020. doi:10.3390/rs12061008.

Laura Millán-Roures, Irene Epifanio, and Vicente Martínez. Detection of Anomalies in Water Networks by Functional Data Analysis. *Mathematical Problems in Engineering*, 2018:1–13, 2018. doi:10.1155/2018/5129735.

Farnaz Moradi, Tomas Olovsson, and Philippas Tsigas. Overlapping Communities for Identifying Misbehavior in Network Communications. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8443 LNAI, pages 398–409. 2014. doi:10.1007/978-3-319-06608-0_33.

Matthew Morgan, Joseph Sexton, Joshua Neil, Aleta Ricciardi, and Joshua Theimer. Network Attacks and the Data they Affect. In *Dynamic Networks and Cyber-Security*, volume 1, pages 1–36. World Scientific (Europe), 2016. doi:10.1142/9781786340757_0001.

Mohsin Munir, Shoaib Ahmed Siddiqui, Andreas Dengel, and Sheraz Ahmed. DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. *IEEE Access*, 7:1991–2005, 2018. doi:10.1109/ACCESS.2018.2886457.

Tsuyoshi Murata. Detecting Communities from Bipartite Networks Based on Bipartite Modularities. In *2009 International Conference on Computational Science and Engineering*, pages 50–57. IEEE, 2009. doi:10.1109/CSE.2009.81.

Joshua Neil, Curtis Hash, Alexander Brugh, Mike Fisk, and Curtis B. Storlie. Scan Statistics for the Online Detection of Locally Anomalous Subgraphs. *Technometrics*, 55(4):403–414, 2013. doi:10.1080/00401706.2013.822830.

Mark Newman. *Networks: An Introduction*. Oxford University Press, 2010. doi:10.1093/acprof:oso/9780199206650.001.0001.

Mark E. J. Newman and Michelle Girvan. Finding and Evaluating Community Structure in Networks. *Physical Review E*, 69(2), 2004. doi:10.1103/PhysRevE.69.026113.

Seyyed M. T. Nezhad, Mahboubeh Nazari, and Ebrahim A. Gharavol. A Novel DoS and DDoS Attacks Detection Algorithm Using ARIMA Time Series Model and Chaotic System in Computer Networks. *IEEE Communications Letters*, 20 (4):700–703, 2016. doi:10.1109/LCOMM.2016.2517622.

Jingchao Ni, Hanghang Tong, Wei Fan, and Xiang Zhang. Flexible and Robust Multi-Network Clustering. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15*, volume 1, pages 835–844, 2015. doi:10.1145/2783258.2783262.

Jordan Noble and Niall M. Adams. Correlation-based streaming anomaly detection in cyber-security. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 311–318, 2016. doi:10.1109/ICDMW.2016.0051.

James R. Norris. *Markov Chains*. Cambridge University Press, 1998. ISBN 9780521633963.

Joseph J. K. Ó Ruanaidh and William J. Fitzgerald. *Numerical Bayesian Methods Applied to Signal Processing*. Statistics and Computing. Springer New York, 1996. doi:10.1007/978-1-4612-0717-7.

Adriano L.I. Oliveira and Silvio R.L. Meira. Detecting Novelties in Time Series Through Neural Networks Forecasting with Robust Confidence Intervals. *Neurocomputing*, 70(1-3):79–92, 2006. doi:10.1016/j.neucom.2006.05.008.

E. S. Page. Continuous Inspection Schemes. *Biometrika*, 41(1/2):100, 1954. doi:10.2307/2333009.

Andrea Pagotto. *ocp: Bayesian Online Changepoint Detection*, 2019. URL `https://CRAN.R-project.org/package=ocp`. R package version 0.1.1.

Francesco Sanna Passino and Nicholas A. Heard. Modelling dynamic network evolution as a pitman-yor process. *Foundations of Data Science*, 1(3):293–306, 2019.

V. Paul Pauca, Farial Shahnaz, Michael W. Berry, and Robert J. Plemmons. Text Mining using Non-Negative Matrix Factorizations. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 452–456. Society for Industrial and Applied Mathematics, 2004. doi:10.1137/1.9781611972740.45.

Nicos G. Pavlidis, Dimitris K. Tasoulis, Niall M. Adams, and David J. Hand. λ-Perceptron: An Adaptive Classifier for Data Streams. *Pattern Recognition*, 44 (1):78–96, 2011. doi:10.1016/j.patcog.2010.07.026.

Yulong Pei, Nilanjan Chakraborty, and Katia Sycara. Nonnegative Matrix Tri-Factorization with Graph Regularization for Community Detection in Social Networks. *Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 2083–2089, 2015.

Joshua Plasse and Niall M. Adams. Multiple Changepoint Detection in Categorical Data Streams. *Statistics and Computing*, 29(5):1109–1125, 2019. doi:10.1007/s11222-019-09858-0.

Hanli Qiao. New SVD based Initialization Strategy for Non-Negative Matrix Factorization. *Pattern Recognition Letters*, 63:71–77, 2015. doi:10.1016/j.patrec.2015.05.019.

James Ramsay, Giles. Hooker, and Spencer. Graves. *Functional Data Analysis with R and MATLAB*. Springer New York, New York, 2009. doi:10.1007/978-0-387-98185-7.

James O. Ramsay, Spencer Graves, and Giles Hooker. *fda: Functional Data Analysis*, 2020. URL `https://CRAN.R-project.org/package=fda`. R package version 5.1.5.1.

Haider Raza, Girijesh Prasad, and Yuhua Li. EWMA Model based Shift-Detection Methods for Detecting Covariate Shifts in Non-Stationary Environments. *Pattern Recognition*, 48(3):659–669, 2015. doi:10.1016/j.patcog.2014.07.028.

Rebecca Killick and Idris Eckley and Philip Jonathan. A Wavelet-Based Approach for Detecting Changes in Second order Structure within Nonstationary Time Series. *Electronic Journal of Statistics*, 7:1167–1183, 2013. doi:10.1214/13-EJS799.

Elizabeth Riddle-Workman, Marina Evangelou, and Niall M. Adams. Adaptive Anomaly Detection on Network Data Streams. In *2018 IEEE International Conference on Intelligence and Security Informatics, ISI 2018*, pages 19–24. IEEE, 2018. doi:10.1109/ISI.2018.8587401.

Elizabeth Riddle-Workman, Marina Evangelou, and Niall M. Adams. Multi-Type Relational Clustering for Enterprise Cyber-Security Networks. *Pattern Recognition Letters*, 149:172–178, 2021. doi:10.1016/j.patrec.2021.05.021.

S. W. Roberts. Control Chart Tests Based on Geometric Moving Averages. *Technometrics*, 1(3):239–250, 1959. doi:10.1080/00401706.1959.10489860.

James M. Robins, Aad van der Vaart, and Valérie Ventura. Asymptotic Distribution of P Values in Composite Null Models. *Journal of the American Statistical Association*, 95(452):1143–1156, 2000. doi:10.1080/01621459.2000.10474310.

Gaetano Romano, Guillem Rigaill, Vincent Runge, and Paul Fearnhead. Detecting Abrupt Changes in the Presence of Local Fluctuations and Autocorrelated Noise. *Journal of the American Statistical Association*, 0:1–16, 2021. doi:10.1080/01621459.2021.1909598.

Gaetano Romano, Guillem Rigaill, Vincent Runge, and Paul Fearnhead. *DeCAFS: Detecting Changes in Autocorrelated and Fluctuating Signals*, 2022. URL `https://CRAN.R-project.org/package=DeCAFS`. R package version 3.3.1.

Bernard Rosner. Percentage Points for a Generalized ESD Many-Outlier Procedure. *Technometrics*, 25(2):165, 1983. doi:10.2307/1268549.

Peter J. Rousseeuw. Silhouettes: A Graphical Aid to The Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. doi:10.1016/0377-0427(87)90125-7.

Yunus Saatçi, Ryan Turner, and Carl Edward Rasmussen. Gaussian Process Change Point Models. In *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, pages 927–934, 2010. ISBN 9781605589077.

Arnaud Sallaberry, Chris Muelder, and Kwan-Liu Ma. Clustering, Visualizing, and Navigating for Large Dynamic Graphs. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7704 LNCS, pages 487–498. Springer, Berlin, Heidelberg, 2013. doi:10.1007/978-3-642-36763-2_43.

Francesco Sanna Passino and Nicholas A. Heard. Bayesian Estimation of the Latent Dimension and Communities in Stochastic Blockmodels. *Statistics and Computing*, 30(5):1291–1307, 2020. doi:10.1007/s11222-020-09946-6.

Simo Sarkka. *Bayesian Filtering and Smoothing*. Cambridge University Press, Cambridge, 2013. doi:10.1017/CBO9781139344203.

A. J. Scott and M. Knott. A Cluster Analysis Method for Grouping Means in the Analysis of Variance. *Biometrics*, 30(3):507, 1974. doi:10.2307/2529204.

Glenn Shafer and Vladimir Vovk. A Tutorial on Conformal Prediction. *Journal of Machine Learning Research*, 9:371–421, 2007. doi:1390681.1390693.

Myra Spiliopoulou, Irene Ntoutsi, Yannis Theodoridis, and Rene Schult. MONIC: Modeling and Monitoring Cluster Transitions. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06*, page 706, New York, New York, USA, 2006. ACM Press. doi:10.1145/1150402.1150491.

Alexander Strehl and Joydeep Ghosh. Cluster Ensembles — a Knowledge Reuse Framework for Combining Multiple Partitions. *Journal of Machine Learning Research*, 3:583–617, 2003. doi:10.1162/153244303321897735.

Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. *Introduction to Data Mining*. Pearson, 2nd edition, 2018. ISBN 0133128903, 9780133128901. doi:10.5555/3208440.

Alexander G. Tartakovsky, Boris L. Rozovskii, Rudolf B Blažek, and Hongjoong Kim. A Novel Approach to Detection of Intrusions in Computer Networks via Adaptive Sequential and Batch-Sequential Change-Point Detection Methods. *IEEE Transactions on Signal Processing*, 54(9):3372–3381, 2006. doi:10.1109/TSP.2006.879308.

Sean J. Taylor and Benjamin Letham. Forecasting at Scale. *The American Statistician*, 72(1):37–45, 2018. doi:10.1080/00031305.2017.1380080.

Marina Theodosiou. Forecasting Monthly and Quarterly Time Series using STL Decomposition. *International Journal of Forecasting*, 27(4):1178–1195, 2011. doi:10.1016/j.ijforecast.2010.11.002.

Melissa J.M. Turcotte, Alexander D. Kent, and Curtis Hash. Unified Host and Network Data Set. In *Data Science for Cyber-Security*, pages 1–22. World Scientific, 2019. doi:10.1142/9781786345646_001.

Ryan Turner, Yunus Saatci, and Carl Edward Rasmussen. Adaptive Sequential Bayesian Change Point Detection. In *Temporal Segmentation Workshop at NIPS*, 2009.

Grigorios F. Tzortzis and Aristidis C. Likas. Kernel-Based Weighted Multi-view Clustering. In *2012 IEEE 12th International Conference on Data Mining*, pages 675–684. IEEE, 2012. doi:10.1109/ICDM.2012.43.

Insha Ullah, Kerrie Mengersen, Rob J. Hyndman, and James Mcgree. Detection of Cybersecurity Attacks through Analysis of Web Browsing Activities using Principal Component Analysis. 2021. URL https://arxiv.org/abs/2107.12592. arXiv: 2107.12592.

Shahid Ullah and Caroline F. Finch. Applications of functional data analysis: A systematic review. *BMC Medical Research Methodology*, 13(1):43, 2013. doi:10.1186/1471-2288-13-43.

Laurens Van Der Maaten. Accelerating T-SNE Using Tree-Based Algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014. doi:10.5555/2627435.2697068.

Laurens Van Der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Jan Verbesselt, Rob Hyndman, Glenn Newnham, and Darius Culvenor. Detecting Trend and Seasonal Changes in Satellite Image Time Series. *Remote Sensing of Environment*, 114(1):106–115, 2010. doi:10.1016/j.rse.2009.08.014.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. Conformal Prediction. In *Algorithmic Learning in a Random World*, pages 17–51. Springer-Verlag, New York, 2005. doi:10.1007/0-387-25061-1_2.

Jenna Walter. COVID-19 News: FBI Reports 300% Increase in Reported Cybercrimes, 2020. URL https://www.imcgrupo.com/covid-19-news-fbi-reports-300-increase-in-reported-cybercrimes/.

Fei Wang, Hanghang Tong, and Ching Yung Lin. Towards Evolutionary Nonnegative Matrix Factorization. In *Proceedings of the National Conference on Artificial Intelligence*, pages 501–506, 2011a. ISBN 9781577355083.

Hua Wang, Feiping Nie, Heng Huang, and Chris Ding. Nonnegative Matrix Tri-Factorization based High-Order Co-Clustering and its Fast Implementation. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 774–783. IEEE, 2011b. doi:10.1109/ICDM.2011.109.

Hua Wang, Feiping Nie, Heng Huang, and Fillia Makedon. Fast Nonnegative Matrix Tri-Factorization for Large-Scale Data Co-Clustering. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 1553–1558, 2011c. doi:10.5591/978-1-57735-516-8/IJCAI11-261.

Shiping Wang and Aiping Huang. Penalized Nonnegative Matrix Tri-Factorization for Co-Clustering. *Expert Systems with Applications*, 78:64–73, 2017. doi:10.1016/j.eswa.2017.01.019.

Xiaogang Wang, Constance van Eeden, and James V. Zidek. Asymptotic Properties of Maximum Weighted Likelihood Estimators. *Journal of Statistical Planning and Inference*, 119(1):37–54, 2004. doi:10.1016/S0378-3758(02)00410-X.

Yuehui Wang, Guoxian Yu, Carlotta Domeniconi, Jun Wang, Xiangliang Zhang, and Maozu Guo. Selective Matrix Factorization for Multi-relational Data Fusion. In *Database Systems for Advanced Applications*, pages 313–329. Springer International Publishing, 2019. doi:10.1007/978-3-030-18576-3_19.

Larry A. Wasserman. *All of Statistics: a Concise Course in Statistical Inference*. Springer, 2nd edition, 2004. ISBN 9780387217369.

Mark Whitehouse, Marina Evangelou, and Niall M. Adams. Activity-based temporal anomaly detection in enterprise-cyber security. In *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, pages 248–250. IEEE, 2016. doi:10.1109/ISI.2016.7745483.

A. Willsky and H. Jones. A Generalized Likelihood Ratio Approach to the Detection and Estimation of Jumps in Linear Systems. *IEEE Transactions on Automatic Control*, 21(1):108–112, 1976. doi:10.1109/TAC.1976.1101146.

Liyan Xie, Yao Xie, and George V. Moustakides. Asynchronous Multi-Sensor Change-Point Detection for Seismic Tremors. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 787–791. IEEE, 2019. doi:10.1109/ISIT.2019.8849413.

Xin Luo, Mengchu Zhou, Yunni Xia, and Qingsheng Zhu. An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems. *IEEE Transactions on Industrial Informatics*, 10(2): 1273–1284, 2014. doi:10.1109/TII.2014.2308433.

Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G Carbonell. Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 211–222, 2010. doi:10.1137/1.9781611972801.19.

Asrul H. Yaacob, Ian K.T. Tan, Su Fong Chien, and Hon Khi Tan. ARIMA Based Network Anomaly Detection. In *2nd International Conference on Communication Software and Networks, ICCSN 2010*, pages 205–209. IEEE, 2010. doi:10.1109/ICCSN.2010.55.

Kenji Yamanishi, Jun-ichi Takeuchi, Graham Williams, and Peter Milne. On-Line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms. *Data Mining and Knowledge Discovery*, 8(3):275–300, 2004. doi:10.1023/B:DAMI.0000023676.72185.7c.

Fei Yan, Xiao-dong Wang, Zhi-qiang Zeng, and Chao-qun Hong. Adaptive Multi-View Subspace Clustering for High-Dimensional Data. *Pattern Recognition Letters*, 130:299–305, 2020. doi:10.1016/j.patrec.2019.01.016.

Fanghua Ye, Zitai Chen, Hui Qian, Rui Li, Chuan Chen, and Zibin Zheng. New Approaches in Multi-View Clustering. In *Recent Applications in Data Clustering*. InTech, 2018. doi:10.5772/intechopen.75598.

Raphael Yuster and Uri Zwick. Fast Sparse Matrix Multiplication. *ACM Transactions on Algorithms*, 1(1):2–13, 2005. doi:10.1145/1077464.1077466.

Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method, 2012. URL `https://arxiv.org/abs/1212.5701`. arXiv: 1212.5701.

Hongyuan Zha, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu. Spectral Relaxation for K-means Clustering. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, pages 1057–1064. MIT Press, 2001.

Guoxing Zhang, Shengming Jiang, Gang Wei, and Quansheng Guan. A Prediction-Based Detection Algorithm Against Distributed Denial-of-Service Attacks. In *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing Connecting the World Wirelessly - IWCMC '09*, page 106, New York, 2009. ACM. doi:10.1145/1582379.1582403.

Marinka Žitnik and Blaž Zupan. Data Fusion by Matrix Factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015. doi:10.1109/TPAMI.2014.2343973.

Appendix A

Adaptive Estimation Appendix

A.1 Poisson Formulation

Suppose a stream of data, $x_1, x_2, \ldots$ is observed. We weight the data using adaptive forgetting factors such that the weight for data point x_i at time t is expressed as,

$$\begin{aligned} w_i(t) &= \prod_{j=i}^{t-1} \lambda_j \\ &= \lambda_{t-1} \prod_{j=i}^{t-2} \lambda_j \end{aligned}$$

where where x_t has weight 1. Following the Exponential family formulation in Section 3.2.7, these weights are included in the prior using a power prior.

Assuming the data follows a Poisson distribution, we aim to estimate the rate γ at time t where historic data before time t is down-weighted by factor λ. Again for simplicity we rewrite the weights as $w_i = \lambda c_i$ where c_i is a known constant of the form $c_i = \prod_{j=i}^{t-2} \lambda_j$ for $i = 1, \ldots, t-2$ and $c_{t-1} = 1$ calculated at previous time steps. At time t the joint power prior for the parameters γ and λ is,

$$\begin{aligned} &P(\gamma, \lambda | x_1, \ldots, x_{t-1}) \\ &\propto P(\gamma) P(\lambda) \prod_{i=1}^{t-1} P(x_i|\gamma)^{\lambda c_i} \end{aligned} \tag{A.1}$$

where

$$\begin{aligned} x_i|\gamma &\sim \text{Pois}\,(\gamma) \;\text{ for } i = 1, \ldots, t-1 \\ \gamma &\sim \text{Gamma}(\alpha_0, \beta_0) \\ \lambda &\sim P(\lambda). \end{aligned}$$

Again the λ prior is left in general terms where the same Beta prior in the Gaus-

sian case given in Section 3.2.3 is used. The joint posterior takes the form,

$$
\begin{aligned}
&P(\gamma, \lambda | x_1, \ldots, x_t) \\
&\propto P(x_t|\gamma) P(\gamma, \lambda | x_1, \ldots, x_{t-1}) \\
&\propto \left(\frac{\gamma^{x_t} e^{-\gamma}}{x_t!} \right) \prod_{i=1}^{t-1} \left(\frac{\gamma^{x_i} e^{-\gamma}}{x_i!} \right)^{\lambda c_i} \times \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)} \gamma^{\alpha_0 - 1} \exp\left(-\gamma \beta_0\right) \times P(\lambda) \\
&\propto \gamma^{\alpha_0 + x_t + \lambda \sum_{i=1}^{t-1} c_i x_i - 1} e^{-\gamma \left(\beta_0 + 1 + \lambda \sum_{i=1}^{t-1} c_i\right)} \times \prod_{i=1}^{t-1} \left(\frac{1}{x_i!} \right)^{\lambda c_i} P(\lambda)
\end{aligned}
$$

Hence the conditional marginal posterior for γ is,

$$
\begin{aligned}
&\gamma | \lambda, x_1, \ldots, x_t \sim \\
&\quad \text{Gamma}\left(\alpha_0 + x_t + \lambda \sum_{i=1}^{t-1} c_i x_i, \beta_0 + 1 + \lambda \sum_{i=1}^{t-1} c_i \right). \qquad \text{(A.2)}
\end{aligned}
$$

The marginal posterior for the forgetting factor has the form,

$$
P(\lambda | x_1, \ldots, x_t) \propto \left(\prod_{i=1}^{t-1} \left(\frac{1}{x_i!} \right)^{\lambda c_i} \right) \frac{\Gamma(\alpha_0 + x_t + \lambda \sum_{i=1}^{t-1} c_i x_i)}{\left(\beta_0 + 1 + \lambda \sum_{i=1}^{t-1} c_i\right)^{\alpha_0 + x_t + \lambda \sum_{i=1}^{t-1} c_i x_i}} P(\lambda)
$$

This does not follow a well known distribution nor can the normalising factor be found analytically. The forgetting factor estimate then depends only on the data and the prior parameters but is independent of the unknown γ parameter.

Similar to the Gaussian case, the Maximum a Posteriori (MAP) estimates at time t are used to estimate λ and γ. First the MAP estimate for λ denoted $\hat{\lambda}_{t-1}^{MAP}$, is found by maximising the unnormalised posterior numerically. The MAP estimate for the Poisson rate at time t takes the following analytic form,

$$
\hat{\gamma}_t^{MAP} = \frac{\alpha_0 + \hat{\lambda}^{MAP} \sum_{i=1}^{t-1} c_i x_i + x_t}{\beta_0 + \hat{\lambda}^{MAP} \sum_{i=1}^{t-1} c_i + 1} \qquad \text{(A.3)}
$$

$$
= \frac{\alpha_0 + \hat{\lambda}_{t-1}^{MAP} \left(\sum_{i=1}^{t-2} (\prod_{j=i}^{t-2} \lambda_j) x_i + x_{t-1} \right) + x_t}{\beta_0 + \hat{\lambda}_{t-1}^{MAP} \left(\sum_{i=1}^{t-2} (\prod_{j=i}^{t-2} \lambda_j) + 1 \right) + 1} \qquad \text{(A.4)}
$$

Similar to the Gaussian case, sequential updating forms can be calculated by defining the following parameters at time t,

$$
\begin{aligned}
N(t) &= \hat{\lambda}_{t-1}^{MAP} \left(\sum_{i=1}^{t-2} \left(\prod_{j=i}^{t-2} \lambda_j \right) x_i + x_{t-1} \right) + x_t \\
&= \hat{\lambda}_{t-1}^{MAP} N(t-1) + x_t,
\end{aligned}
$$

$$
\begin{aligned}
D(t) &= \hat{\lambda}_{t-1}^{MAP}\left(\sum_{i=1}^{t-2}\prod_{j=i}^{t-2}\lambda_j + 1\right) + 1 \\
&= \hat{\lambda}_{t-1}^{MAP} D(t-1) + 1
\end{aligned}
$$

$$
\begin{aligned}
F(t) &= \hat{\lambda}_{t-1}^{MAP}\left(\sum_{i=1}^{t-2}\left(\prod_{j=i}^{t-2}\right)\lambda_j \log(x_i!) + \log(x_{t-1}!)\right) + \log(x_t!) \\
&= \hat{\lambda}_{t-1}^{MAP} F(t-1) + \log(x_t!)
\end{aligned}
$$

The MAP estimate for the rate has the following sequential updates,

$$
\hat{\gamma}_t^{MAP} = \frac{\alpha_0 + N(t)}{\beta_0 + D(t)}
$$

and again the parameters $N(t-1)$, $D(t-1)$ and $F(t-1)$ are used to produce sequentially updating forms for the unnormalised marginal posterior density function for λ,

$$
P(\lambda|x_1,\ldots,x_t) \propto \exp\left(\lambda F(t-1)\right)\frac{\Gamma(\alpha_0 + x_t + \lambda N(t-1))}{(\beta_0 + 1 + \lambda D(t-1))^{\alpha_0 + x_t + \lambda N(t-1)}} P(\lambda).
$$

Thus to update the parameters of this Poisson model only $N(t-1)$, $D(t-1)$ and $F(t-1)$ are required from the historic data.

To update the parameter priors when implementing this method sequentially, the following prior for the update at time t is used,

$$
\gamma \sim \text{Gamma}(\hat{\gamma}_{t-1} + 1, 1)
$$

which corresponds to this distribution having a mode equal to $\hat{\gamma}_{t-1}$ with equal influence on the estimates as x_t (weight of 1).

A.2 Gaussian Synthetic Simulations

A.2.1 Gaussian Performance Comparison Single Change

We now evaluate the performance of the methods when the stream contains a single mean change point. The series used in Tables A.1 and A.2 are sampled

from,

$$
\begin{aligned}
X_1, \ldots, X_{5000} &\sim N(\mu, \sigma^2) \\
X_{5001}, \ldots, X_{10000} &\sim N(\mu \pm \kappa, \sigma^2) \\
\mu &\sim U(-20, 20) \\
\sigma^2 &\sim U(0.5, 5) \\
\kappa &\sim U(2\sigma, 5\sigma).
\end{aligned}
$$

In Tables A.1 and A.2 the BFF procedure has notably more comparable performance to competitive approaches for both μ and σ^2 in comparison to the stationary results in Section 3.3.4. For BFF, no degradation in performance is observed which suggests BFF is suitable for non-stationary data monitoring unlike many competitive procedures whose performance worsens, particularly AFF 1e-7 and MLE. As the MLE assumes the data is i.i.d. it has poor estimation performance as it is unable to adapt quickly to the data.

Table A.1 additionally records the average time for the models to adapt to the new data distribution. It can be seen that for models that are built to adapt slowly i.e. AFF 1e-7, FFF 0.98, EWMA 0.05 and MLE, the time to change is high. Alternatively, our approach has similar performance to procedures that can adapt quickly such as AFF 1e-4, FFF 0.9 and EWMA 0.75. Finally, it can be seen that although BOCPD has small errors, its' time to change to the new distribution is high.

Table A.1: Average parameter estimation performance of μ over 100 data streams with a single change point of length $10,000$. The stream is generated such that $X_1, \ldots, X_{5000} \sim N(\mu, \sigma^2)$ where $\mu \sim U(-20, 20)$ and $\sigma^2 \sim U(0.5, 5)$ and $X_{5001}, \ldots, X_{10000} \sim N(\mu \pm \kappa, \sigma^2)$ where $\kappa \sim U(2\sigma, 5\sigma)$. The change point sign is chosen at random.

	MSE	SE SD	MAE	AE SD	MAPE	APE SD	Time to Change
BFF	0.096	0.356	0.233	0.184	0.057	0.069	35.250
AFF 1e-7	0.268	2.154	0.154	0.475	0.045	0.146	316.040
AFF 1e-4	0.056	0.410	0.130	0.189	0.035	0.060	30.770
FFF 0.9	0.142	0.370	0.289	0.224	0.071	0.086	**24.420**
FFF 0.98	**0.032**	**0.359**	**0.129**	**0.114**	**0.032**	**0.043**	158.270
EWMA 0.05	0.071	**0.352**	0.202	0.161	0.050	0.061	58.600
EWMA 0.75	1.594	2.255	0.976	0.738	0.240	0.286	**0.910**
MLE	9.050	10.278	1.976	2.002	0.601	0.614	2140.800
BOCPD	**0.006**	0.373	**0.031**	**0.066**	**0.007**	**0.021**	83.790

Table A.2: Average parameter estimation performance of σ^2 over 100 data streams with a single change point of length $10,000$. The stream is generated such that $X_1, \ldots, X_{5000} \sim N(\mu, \sigma^2)$ where $\mu \sim U(-20, 20)$ and $\sigma^2 \sim U(0.5, 5)$ and $X_{5001}, \ldots, X_{10000} \sim N(\mu \pm \kappa, \sigma^2)$ where $\kappa \sim U(2\sigma, 5\sigma)$. The change point sign is chosen at random.

	MSE	SE SD	MAE	AE SD	MAPE	APE SD
BFF	0.625	**0.726**	0.590	0.384	0.217	0.142
AFF 1e-7	1.568	11.504	0.356	0.993	0.133	0.364
AFF 1e-4	**0.310**	1.116	**0.300**	0.401	**0.113**	0.151
FFF 0.9	0.848	1.382	0.666	0.498	0.252	0.188
FFF 0.98	**0.239**	1.033	**0.313**	**0.302**	**0.118**	**0.112**
EWMA 0.05	0.424	**0.643**	0.470	**0.354**	0.178	**0.134**
EWMA 0.75	4.876	4.308	1.843	0.758	0.695	0.286
MLE	44.141	52.364	3.678	3.904	1.336	1.418

A.2.2 Gaussian Performance Comparison Trend

We now investigate the performance of the comparison approaches on data that exhibits trend, a common feature of real data sets. For the experiments in Tables A.3 and A.4, the stream is generated from,

$$\begin{aligned} X_1, \ldots, X_{5000} &\sim N(\mu, \sigma^2) \\ X_i &\sim N(\mu \pm (5000 - i)\kappa, \sigma^2) \quad \text{for } i \in \{5001, 5002, \ldots, 7500\} \\ X_{7501}, \ldots, X_{10000} &\sim N(\mu \pm (2500 \times \kappa), \sigma^2) \\ \mu &\sim U(-20, 20) \\ \sigma^2 &\sim U(0.5, 5). \end{aligned}$$

The gradient sign is chosen at random within the simulations. In Table A.3 BFF has superior performance to many methods including BOCPD. BFF performs well here as it can adapt to trend regions and stationary periods automatically. AFF 1e-4, which can adapt quickly, has superior performance to 1e-7. However, we note that in comparison to the stationary example in Table 3.3, AFF 1e-7 has much poorer performance. This highlights the difficulty in setting this learning rate appropriately for the data. For trend data, BOCPD, which assumes the data has stationary sections within the data rather than trends, has poorer performance than previously seen. Finally, we note that the MLE approach has very poor performance for this data, particularly for the estimation of σ^2.

Table A.3: Average parameter estimation performance of μ over 100 data streams with a trend between 5001 and 7500. The initial stream is generated such that $X_1, \ldots, X_{5000} \sim N(\mu, \sigma^2)$ where $\mu \sim U(-20, 20)$ and $\sigma^2 \sim U(0.5, 5)$. The gradient of the trend is sampled from $\kappa \in \{0.005, 0.006, \ldots, 0.019, 0.02\}$. Finally $X_{7501}, \ldots, X_{10000} \sim N(\mu \pm (2500 \times \kappa), \sigma^2)$. Again the sign of the gradient is chosen randomly.

	MSE	SE SD	MAE	AE SD	MAPE	APE SD
BFF	0.110	0.156	0.257	0.195	0.116	**1.129**
AFF 1e-7	0.499	1.987	0.307	0.613	0.146	3.872
AFF 1e-4	**0.052**	**0.136**	**0.140**	**0.173**	**0.061**	1.326
FFF 0.9	0.152	0.214	0.302	0.228	0.136	1.288
FFF 0.98	0.145	0.224	0.263	0.252	0.103	1.724
EWMA 0.05	**0.090**	**0.127**	0.233	**0.178**	0.103	**1.083**
EWMA 0.75	1.680	2.380	1.001	0.757	0.446	4.197
MLE	238.761	275.926	9.996	10.311	1.202	25.382
BOCPD	0.179	0.379	**0.228**	0.344	**0.086**	1.926

Table A.4: Average parameter estimation performance of σ^2 over 100 data streams with a trend between 5001 and 7500. The initial streams is generated such that $X_1, \ldots, X_{5000} \sim N(\mu, \sigma^2)$ where $\mu \sim U(-20, 20)$ and $\sigma^2 \sim U(0.5, 5)$. The gradient of the trend is sampled from $\kappa \in \{0.005, 0.006, \ldots, 0.019, 0.02\}$. Finally $X_{7501}, \ldots, X_{10000} \sim N(\mu \pm (2500 \times \kappa), \sigma^2)$. Again the sign of the gradient is chosen randomly.

	MSE	SE SD	MAE	AE SD	MAPE	APE SD
BFF	0.687	0.819	0.625	0.408	0.218	0.143
AFF 1e-7	1.140	6.524	0.386	0.879	**0.151**	0.369
AFF 1e-4	**0.212**	**0.604**	**0.276**	**0.317**	**0.102**	**0.119**
FFF 0.9	0.942	1.571	0.706	0.531	0.252	0.190
FFF 0.98	**0.270**	**0.431**	**0.376**	**0.307**	0.152	**0.133**
EWMA 0.05	0.479	0.754	0.501	0.382	0.180	0.138
EWMA 0.75	5.309	4.700	1.943	0.801	0.696	0.287
MLE	12646.882	21030.832	53.955	75.129	28.381	39.530

A.3 Poisson Synthetic Simulations

A.3.1 Poisson Performance Comparison No Change

We now evaluate the estimation performance to comparison methods on stationary Poisson data. Similar to the Gaussian case, these estimates can be used as a baseline of performance for each procedure when compared to non-stationary

results. For the results in Table A.5, the stream is sampled from,

$$X_1, \ldots, X_{10000} \sim Pois(\gamma)$$
$$\gamma \sim U(0.5, 20).$$

Table A.5 measures the average estimation performance for the Poisson rate over 100 simulations as explained above. Similar to the Gaussian case, for this data, BFF does not have the best performance where AFF, MLE and BOCPD are optimal. Once again the importance of hyper-parameter setting is noted. For methods which forget slowly (AFF 1e-7, FFF 0.98 and EWMA 0.05), their errors are very low however in non-stationary examples their performance is less optimal. Once again the MLE has optimal performance for this example due to its i.i.d assumption.

Table A.5: Average parameter estimation performance for Poisson rate over 100 stationary data streams of length $10,000$. The stream is generated from Pois(γ) where $\gamma \sim U(0.5, 20)$. Best performer in bold, second best in grey bold.

	MSE	SE SD	MAE	AE SD	MAPE	APE SD
BFF	0.419	0.594	0.480	0.362	0.063	0.048
AFF 1e-7	**0.005**	**0.007**	0.055	**0.040**	0.008	**0.006**
AFF 1e-4	0.038	0.063	0.136	0.112	0.019	0.016
FFF 0.9	0.512	0.725	0.538	0.406	0.077	0.058
FFF 0.98	0.100	0.140	0.237	0.179	0.034	0.025
EWMA 0.05	0.251	0.353	0.376	0.284	0.054	0.040
EWMA 0.75	5.854	8.439	1.819	1.374	0.258	0.194
MLE	**0.002**	**0.002**	**0.035**	**0.020**	**0.005**	**0.003**
BOCPD	0.008	0.204	**0.037**	0.064	**0.006**	0.016

A.3.2 Poisson Performance Comparison Single Change

The performance of the methods for Poisson streams with a single rate change point is compared. The stream used in Table A.6 is sampled from:

$$\begin{aligned} X_1, \ldots, X_{5000} &\sim Pois(\gamma) \\ X_{5001}, \ldots, X_{10000} &\sim Pois(\gamma \pm \kappa) \\ \gamma &\sim U(0.5, 20) \\ \kappa &\sim U(2\sqrt{\gamma}, 5\sqrt{\gamma}). \end{aligned}$$

Table A.6 averages over 100 data streams of length 10,000 where there is a single change point. In comparison to the results in Table A.5, the errors are now larger than that in the stationary case, particularly for MLE and AFF. Here BFF outperforms AFF in terms of MSE, MAPE and time to change. Although FFF 0.98, EWMA 0.03 and BOCPD have low errors, the time to change is high for these approaches. As the calculated errors exclude the grace period after a change, they do not take this poor performance during these regions into account, hence the time to change measure is highly important to fully assess the performance. BFF has a good balance between the error and the time to change, making it appropriate in practice.

Table A.6: Average parameter estimation performance of Poisson rate over 100 data streams of length $10,000$ with a single change point. The stream is generated such that $X_1, \ldots, X_{5000} \sim \text{Pois}(\gamma)$ where $\gamma \sim U(0.5, 20)$ and $X_{5001}, \ldots, X_{10000} \sim \text{Pois}(\gamma \pm \kappa)$ where $\kappa \sim U(2\sqrt{\gamma}, 5\sqrt{\gamma})$. The change point sign is chosen at random.

	MSE	SE SD	MAE	AE SD	MAPE	APE SD	Time to Change
BFF	0.591	1.495	0.554	0.474	0.111	4.023	25.800
AFF 1e-7	5.345	15.811	1.002	1.904	43.647	86.636	1237.200
AFF 1e-4	1.201	5.073	0.405	0.838	8.254	24.257	781.130
FFF 0.9	0.689	1.559	0.610	0.510	0.121	**3.936**	**22.450**
FFF 0.98	**0.147**	**1.266**	**0.271**	**0.247**	0.382	5.364	152.670
EWMA 0.05	0.341	**1.306**	0.426	0.363	**0.107**	4.248	54.030
EWMA 0.75	7.778	12.551	2.064	1.699	0.257	**1.170**	**0.960**
MLE	31.676	35.974	3.681	3.726	165.991	170.539	2140.030
BOCPD	**0.029**	1.358	**0.069**	**0.145**	**0.061**	4.865	73.090

A.3.3 Poisson Performance Comparison Trend

The performance of the comparison approaches is now investigated on data that exhibits trend, a common feature of real data sets. In Table A.7 the stream is generated from,

$$\begin{aligned} X_1, \ldots, X_{5000} &\sim \text{Pois}(\gamma) \\ X_i &\sim \text{Pois}(\gamma \pm (5000 - i)\kappa) \text{ for } i \in \{5001, 5002, \ldots, 7500\} \\ X_{7501}, \ldots, X_{10000} &\sim \text{Pois}(\gamma \pm (2500 \times \kappa)) \\ \gamma &\sim U(0.5, 20). \end{aligned}$$

The gradient sign is chosen at random in the simulations and κ is uniformly sampled from $\{0.005, 0.006, \ldots, 0.019, 0.02\}$. Table A.7 displays the average estimation performance of the Poisson rate for the comparison methods when the data exhibits a trend. Again MLE has poor performance on this non-stationary data due to the violation of assumptions for this model. AFF also has poor performance on this data. As the gradients are gradual, a large forgetting factor of 0.98 is suitable for this example (see FFF 0.98), and has one of the smallest errors. BOCPD again has favourable performance for this data. BFF also has good performance showing it is robust to different situations.

Table A.7: Average parameter estimation performance of Poisson rate over 100 data streams with a trend between 5001 and 7500. The initial stream is generated such that $X_1, \ldots, X_{5000} \sim \text{Pois}(\gamma)$ where $\gamma \sim U(0.5, 20)$. The gradient of the trend is sampled from $\kappa \in \{0.005, 0.006, \ldots, 0.019, 0.02\}$. Finally $X_{7501}, \ldots, X_{10000} \sim \text{Pois}(\gamma \pm (2500 \times \kappa))$. The sign of the gradient is chosen randomly.

	MSE	SE SD	MAE	AE SD	MAPE	APE SD
BFF	1.123	2.038	0.770	0.693	0.046	0.040
AFF 1e-7	31.416	47.206	3.397	3.997	0.119	0.138
AFF 1e-4	18.355	29.166	2.438	2.932	0.086	0.094
FFF 0.9	1.219	2.141	0.816	0.713	0.053	0.048
FFF 0.98	**0.349**	**0.568**	**0.434**	**0.377**	**0.027**	**0.024**
EWMA 0.05	**0.607**	**1.053**	**0.576**	0.502	0.037	0.034
EWMA 0.75	13.932	24.921	2.757	2.419	0.178	0.163
MLE	240.691	278.159	10.115	10.419	0.294	0.272
BOCPD	0.683	1.511	**0.457**	0.659	**0.021**	**0.033**

A.3.4 Poisson Performance Comparison Multiple Change with Trend

Similar to the Gaussian experiments, we now compare the estimation results on Poisson data streams with both change points and trend periods. Ten rate change points and two trend periods are generated uniformly over $(200, 10000)$ such that change points and trend starting points are at least 200 points apart. These simulations have the following properties:

- The initial rate of the series is sampled from $U(0.5, 20)$
- Change point magnitudes are sampled from $U(2\sqrt{\gamma}, 5\sqrt{\gamma})$ where γ is the most recent Poisson rate of the series
- Periods of trend have a minimum length of 50 points with gradient magnitude uniformly sampled from $\{0.005, 0.006, \ldots, 0.02\}$. Proceeding this trend, there is a constant mean of length > 30.

The results in Table A.8 displays the average performance results over 100 Poisson simulations that contain both change points and trends. Again AFF has poor performance for Poisson data due to the unusual behaviour of the forgetting factor which was seen in Figure 3.6 (d) making it less suitable for Poisson data. Again although FFF 0.98, EWMA 0.05 and BOCPD have small errors, the time to change is higher than BFF for these models. Similarly, FFF 0.9 and EWMA 0.75 are fast to change however have higher errors than BFF. Thus BFF provides a balance between low estimation errors and fast adaptation. In the context of cyber-security, this fast adaptation is important however accuracy of the results is also a priority.

Table A.8: Average parameter estimation performance of Poisson rate over 100 data streams of length 10,000 with 10 rate change points and two trends positioned randomly throughout. Each of the change points and trends are at least 200 points apart where trends have a stationary period proceeding it. Initially the rate is sampled from $U(0.5, 20)$. The change point magnitudes are sampled from $U(2\sqrt{\gamma}, 5\sqrt{\gamma})$ where γ is the current rate. Trend gradients are sampled from $\{0.005, 0.006, \ldots, 0.019, 0.02\}$. Best performer in bold, second best in bold grey.

	MSE	SE SD	MAE	AE SD	MAPE	APE SD	Time to Change	Time to Change SD
BFF	1.257	9.669	0.715	0.800	0.061	0.343	25.685	11.063
AFF 1e-7	78.113	128.687	6.146	5.553	2.315	6.582	390.999	209.135
AFF 1e-4	68.525	116.139	5.676	5.264	2.160	6.243	382.340	199.591
FFF 0.9	1.338	**9.467**	0.762	0.813	0.075	0.344	23.883	9.354
FFF 0.98	0.635	10.878	0.420	0.643	0.065	0.420	142.641	31.986
EWMA 0.05	0.807	10.267	0.541	0.674	0.055	0.348	54.515	17.902
EWMA 0.75	12.399	23.124	2.514	2.279	0.228	**0.338**	**1.453**	**2.658**
MLE	230.100	356.861	10.274	8.926	4.123	10.318	440.396	291.396
BOCPD	**0.473**	11.228	**0.254**	**0.614**	**0.026**	0.367	26.754	37.210

Appendix B

Change Point Appendix

B.1 Recalibration Window Size Timings

Table B.1: Average timing in seconds for proposed p-value calibration for different sliding window sizes. Data simulated as described in Section 4.3.3 where $n = 20,000$ with 100 repetitions and $\alpha = 0.005$.

	BFF-Pred	BFF-Post
Uncalibrated	4.955(0.266)	36.718(2.077)
SW 200	4.955(0.266)	36.718(2.077)
SW 500	5.717(0.307)	38.174(2.64)
SW 1000	7.022(0.321)	40.729(3.087)
SW 2000	9.539(0.477)	45.25(2.649)
SW 5000	18.38(0.963)	61.149(3.78)
SW 7000	21.311(1.204)	67.549(4.692)
SW 10000	24.755(1.582)	74.26(4.268)

B.1.1 Change Point Comparison Methods

The Pruned Exact Linear Time (PELT) method [Killick et al., 2012] is a well regarded batch approach and acts as a benchmark to the sequential approaches. PELT builds upon optimal partitioning approaches, by pruning to make it more optimal. We implement PELT using the **changepoint** R package [Killick et al. 2016] with a SIC penalty as suggested in their paper.

We additionally compare to DeCAFS [Romano et al., 2021], a batch procedure that is capable of detecting abrupt changes in the presence of trend and autocorrelation. DeCAFS models the local fluctuations (or trends) as a random walk process and autocorrelated noise via an AR(1) process. The number and location of the change points are found by minimising a penalised negative log likelihood based on their proposed data model and is solved using dynamic programming. The only

parameter required is the penalty which they suggest be $\beta = 2 \log n$ where n is the length of the data. This method is implemented using the R package **DeCAFS** [Romano et al., 2022].

The Cumulative Sum (CUSUM) algorithm was first proposed by Page [1954] and is a popular method in statistical quality control for sequential change point detection. Given the current Gaussian parameters of the stream, a change point in the mean is detected if either of the two CUSUM statistics exceeds a specified threshold. There are two control parameters, h and k required for this model. Setting these parameters is not intuitive hence they present a problem in streaming contexts where setting these values empirically cannot be done. Unless specified, we use the recommendations of Hawkins [1993] and implement using the R package **ffstream** [Bodenham, 2018].

Another popular sequential change point procedure is the Exponential Weighted Moving Average (EWMA) method proposed by Roberts [1959] for statistical quality control. A statistic calculated as the convex combination of the newly observed data value and the previous value of the statistic is monitored. This procedure requires two parameters, one for the combination weight, ω whose value is in $[0, 1]$, and another for determining the control limit, L. A change is detected when this statistic exceeds the control limits. Lucas and Saccucci [1990] suggests $\omega \in [0.05, 1.0]$ and $L \in [2.4, 3]$. Again we implement this method using the R package **ffstream**.

The frequentist sequential adaptive estimation procedure by Anagnostopoulos et al. [2012] has been extended by Bodenham and Adams [2017] for change point detection using control limits and is called *AFF*. Bodenham and Adams [2017] assume the adaptive mean estimate follows a Gaussian distribution, producing control limits for a specified significance level α. After each change point, the estimation procedure is restarted, requiring a grace period to relearn the distribution of the data. As described in Section 3.1.2, a step size, η, for updating the forgetting factor is also required. Additionally, similar to our approach, this procedure requires a threshold value c. We implemented this using **ffstream** where unless stated, we use the default parameter $\eta = 0.001$ for the step size, however, various thresholds are implemented.

Finally a very popular and high performing sequential change point procedure called *Bayesian Online Change Point Detection* (BOCPD) proposed simultaneously by Adams and MacKay [2007] and Fearnhead [2006] is implemented. This procedure jointly and sequentially infers the distribution parameters of the data alongside the run length and has been detailed for exponential family distributions. The run length represents the time since the last change point and is used to determine whether a change has occurred. Priors for the parameters are required and after a change, the estimation procedure is restarted. Default parameters for the priors are used with an exponential prior on the run length distribution. This procedure is implemented using the R package **ocp** [Pagotto, 2019]. Although neither Adams and MacKay [2007] or Fearnhead [2006] specifically details how to

classify detections using this model, within the package, there are two approaches implemented. The first is called *colmaxes* where at each point the run length, r, with the highest probability is chosen which corresponds to a change r points prior. The second method goes a step further called the *threshold* method and uses the same methodology as colmaxes however only considers cases when this probability exceeds a threshold whose default value is 0.5. Setting this threshold value is an additional challenge with no guidance given.

B.2 Change Point Model Parameter Specification

Below we outline the parameters used within the grid search for each model for Section 4.3.2 and their default parameters. The range of parameters used here is based on author recommendations or suitable values in appropriate ranges. The parameter combination that optimises the *F1* score for the example in Section 4.3.2 has been documented in bold. Default parameters are boxed.

B.2.1 PELT

Function: `cpt.mean*`, `cpt.var`, `cpt.meanvar`
Penalty: None, SIC, BIC, **MBIC**, AIC, Hannan-Quinn

B.2.2 DeCAFS

L_0 penalty, β: $2log(n)$
For the optimisation in Section 4.3.2 we did a grid search over 0.5 to 100 with a step size of 0.5. The optimal penalty was 10.5 whereas the default penalty value for this example is 18.421 (data has length 10000).

B.2.3 CUSUM

h: 0.25, 0.5, **1**
k: **2.71**, 5.14, 8.76

B.2.4 EWMA

ω : 0.05, 0.25, 0.5, **1**
L : 2.4, 2.5, 2.8, **3**

B.2.5 AFF

η: **0.001**, 0.01, 0.1
c: **0.001**, 0.005, 0.01, 0.05, 0.08, 0.1

B.2.6 BOCPD

In this work we assume the data follows a Gaussian distribution where we use a Normal prior for the mean with mean zero and variance κ, we use a Gamma prior on the inverse variance with parameters α and β, and an exponential prior with parameter λ for the gap distribution as done by the authors Adams and MacKay 2007. This setup is default within the **ocp** package in R.
λ: 50, $\boxed{100}$, **200**
α: $\boxed{0.01}$, 1, **10**
β: $\boxed{\text{1e-4}}$, **0.01**, 1, 100
κ: $\boxed{\mathbf{0.01}}$, 1, 100
Threshold: $\boxed{0.5}$, 0.75, **0.9**

B.2.7 BFF

Threshold: 0.001, 0.003, $\boxed{0.005}$, **0.008**, 0.01, 0.015, 0.02
Grace Period: **10**, $\boxed{20}$, 30, 50
Parameter choice (for post method): μ, $\boxed{\boldsymbol{\lambda}}$
Fixed prior Beta$(39, 1.8)$ prior for λ which has a mode of 0.98 and variance 0.001. This prior is robust for a wide variety of simulations.

B.3 Additional Simulation Study

B.3.1 Performance on Stationary Data

We first begin by exploring how each change point procedure handles stationary data. For such data, no change points exist hence the number of detections should be minimal. Alternatively, where thresholds are used, there exists an expected number of detections regardless of the number of changes in the data. We simulate stationary Gaussian data streams as follows,

$$X_1, \ldots, X_{50000} \sim N(0, \sigma^2)$$
$$\sigma^2 \sim U(0.5, 20)$$

where we record the total number of detections made by each of the algorithms detailed in Section B.1.1. We repeat this experiment 100 times where the standard deviation between each stream is given in brackets.

PELT, DeCAFS and BOCPD-Threshold have the fewest detections. As PELT and DeCAFS are batch approaches, they benefit from viewing the full stream. It is important to note the extreme difference in BOCPD for the colmaxes and threshold methods where colmaxes has labelled $\frac{1}{5}^{th}$ of the series as a change point. Additionally, this method has large computation times.

Table B.2: Number of detections and computation time in seconds over 100 stationary data streams of length 50,000 generated as described in Appendix B.3.1. Standard deviation between streams given in brackets.

	Detections	Time (sec)
PELT	0.000(0.000)	3.525(1.211)
DeCAFS	0.000(0.000)	3.249 (1.016)
CUSUM	143.770(15.504)	0.003(0.001)
EWMA	72.720(16.956)	0.004(0.001)
AFF 0.008	67.050(19.225)	0.003(0.001)
AFF 0.005	45.060(17.789)	0.003(0.000)
AFF 0.003	27.890(12.857)	0.003(0.000)
BOCPD Colmaxes	10146.510(1857.670)	416.437(21.106)
BOCPD Threshold	9.300(3.043)	416.437(21.106)
BFF-Pred 0.008	349.260(6.683)	25.387(1.431)
BFF-Pred 0.005	227.060(5.680)	25.387(1.431)
BFF-Pred 0.003	139.070(5.161)	25.387(1.431)
BFF-Post-λ 0.008	342.690(7.689)	119.835(8.371)
BFF-Post-λ 0.005	222.890(6.153)	119.835(8.371)
BFF-Post-λ 0.003	137.530(4.980)	119.835(8.371)

While our method does have false detections, the number of detections is as expected. At a threshold level of 0.005, a well calibrated procedure should detect 250 changes. Thus our procedures detect similar numbers of points to what is expected. Similarly at the 0.008 and 0.003 levels, our procedures make the expected number of detections.

B.3.2 Performance on Trend Only Data

We now investigate the effect trend has on change point performance in comparison to the stationary case. Ideally there should be little difference in performance to the results in Section B.3.1 for methods that are resistant to trend. The results in Table B.3 average over 100 Gaussian data streams of length $n = 50,000$ which contain $p = \lceil \frac{n}{500} \rceil = 100$ trend regions each with a minimum length of 50 points. Additionally, after a trend region, there is a stationary period with a minimum length of 30 points. The gradient magnitude of these trend regions is sampled from $\{0.05, 0.06, 0.07, 0.08\}$ where the data has a fixed variance of 1.

Table B.3 displays the number of detections made by each of the change point approaches on the simulated trend data. In comparison to the stationary results in Table B.2, all methods but DeCAFS, BOCPD and BFF have a significant increase in the number of detections. PELT has gone from 0 detections in the stationary case to over 600 for the trend data suggesting it is highly affected by trend. DeCAFS still has zero detections, suggesting it is robust to trend. The BFF de-

tections are however very similar to that of the stationary case, meaning it is not affected by trend, unlike the competitive approaches. Strangely BOCPD-Colmaxes has fewer detections for data with trend than the stationary case. BOCPD threshold has a smaller increase in the number of detections hence this method may be affected less by trend than the other competitive approaches.

Table B.3: Simulated change point performance results over 100 streams of Gaussian data of length 50,000 containing only trend regions as described in Appendix B.3.2. Standard deviation given in brackets.

	Detections	Time (sec)
PELT	655.150(43.822)	0.207(0.073)
DeCAFS	0.000(0.000)	4.771(0.303)
CUSUM	557.620(23.697)	0.002(0.001)
EWMA	536.880(26.614)	0.002(0.001)
AFF 0.008	602.420(29.386)	0.002(0.001)
AFF 0.005	582.450(29.503)	0.002(0.001)
AFF 0.003	564.110(30.054)	0.002(0.001)
BOCPD Colmaxes	1435.930(116.079)	559.658(36.944)
BOCPD Threshold	126.370(62.076)	559.658(36.944)
BFF-Pred 0.008	340.420(7.696)	28.301(2.627)
BFF-Pred 0.005	223.170(6.739)	28.301(2.627)
BFF-Pred 0.003	138.020(5.207)	28.301(2.627)
BFF-Post-λ 0.008	333.230(7.391)	140.564(12.084)
BFF-Post-λ 0.005	218.830(5.754)	140.564(12.084)
BFF-Post-λ 0.003	136.280(5.689)	140.564(12.084)

B.3.3 Poisson Data Streams

To showcase the robustness of our proposed procedure, we additionally perform change point detection on Poisson data with rate γ. As the variance of the data changes for different rates, at change points, the rate has jumps of magnitude within one to five standard deviations of the current rate. This ensures the change points are detectable. PELT, BOCPD and BFF have different versions for the Poisson case which we implement here. We note that DeCAFS assumes the errors of the data are Gaussian, so is not suitable for this example. To assess the performance, 100 streams of length $n = 10,000$ are generated where at random 0.4% of points are selected as change points. We also include trend periods within the data where the trend is scaled depending on the current rate. Table B.4 displays the performance of the algorithms on Poisson data.

In Table B.4, PELT has the best performance followed by BOCPD threshold in terms of *F1*. For our proposed approaches, BFF-Pred with a threshold of 0.008 has the highest *F1*. The BFF procedures in general have large ARL0 and high

Table B.4: Simulated change point performance results over 100 streams of Poisson data of length 10,000 as described in Appendix B.3.3. Standard deviation given in brackets.

	F1	ARL0	ARL1	Recall	Precision	FP	Detections	Time (sec)
PELT	0.872(0.076)	1568.633	0.124	0.934(0.068)	0.820(0.094)	3.098	16.629	0.064(0.039)
DeCAFS	0.339(0.233)	185.352	0.597	0.853(0.113)	0.245(0.213)	104.4	116.8	0.152(0.041)
CUSUM	0.388(0.058)	264.587	2.838	0.863(0.091)	0.252(0.045)	37.720	50.220	0.001(0.001)
EWMA	0.478(0.068)	373.534	1.986	0.891(0.094)	0.329(0.059)	26.932	39.833	0.001(0.001)
AFF 0.008	0.601(0.071)	476.146	3.980	0.918(0.078)	0.451(0.074)	16.742	30.045	0.001(0.001)
AFF 0.005	0.631(0.072)	537.788	4.019	0.917(0.076)	0.485(0.075)	14.553	27.833	0.001(0.000)
AFF 0.003	0.706(0.070)	719.190	4.066	0.941(0.062)	0.569(0.079)	10.606	24.250	0.001(0.000)
BOCPD Colmaxes	0.690(0.137)	704.540	0.099	0.958(0.058)	0.554(0.160)	13.235	27.106	136.688(5.850)
BOCPD Threshold	0.813(0.124)	1056.595	0.102	0.911(0.076)	0.747(0.167)	5.311	18.508	136.688(5.850)
BFF-Pred 0.008	0.678(0.119)	1149.037	0.861	0.651(0.139)	0.728(0.151)	2.386	8.376	5.746(0.424)
BFF-Pred 0.005	0.653(0.134)	1194.540	0.827	0.545(0.151)	0.857(0.148)	0.911	5.931	5.746(0.424)
BFF-Pred 0.003	0.499(0.155)	756.250	0.732	0.353(0.141)	0.963(0.095)	0.178	3.406	5.746(0.424)
BFF-Post-γ 0.008	0.677(0.130)	1107.174	1.266	0.620(0.161)	0.795(0.165)	1.762	7.465	20.559(0.909)
BFF-Post-γ 0.005	0.627(0.139)	1568.743	1.299	0.509(0.152)	0.883(0.149)	0.792	5.475	20.559(0.909)
BFF-Post-γ 0.003	0.486(0.171)	972.500	1.184	0.345(0.148)	0.959(0.103)	0.188	3.366	20.559(0.909)
BFF-Post-λ 0.008	0.655(0.125)	765.312	1.951	0.728(0.133)	0.618(0.164)	4.683	11.396	20.559(0.909)
BFF-Post-λ 0.005	0.676(0.123)	1121.151	1.862	0.629(0.148)	0.765(0.164)	2.040	7.822	20.559(0.909)
BFF-Post-λ 0.003	0.568(0.152)	490.600	1.724	0.429(0.151)	0.926(0.125)	0.386	4.347	20.559(0.909)

precision. Additionally in comparison to sequential methods; CUSUM, EWMA and AFF, the ARL1 is lower, particularly for BFF-Pred. BOCPD has the longest computation time but shows strong change point performance on Poisson data. This example demonstrates that the proposed procedure is appropriate for other distributions of data.

Appendix C

Forecast Appendix

C.1 ARIMA Sliding Window Experiment

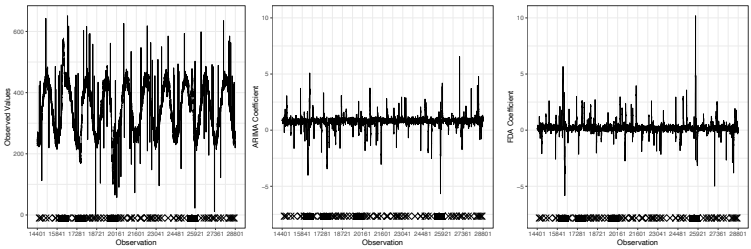
To investigate appropriate sliding window sizes for the ARIMA model, we investigate the forecast performance for varying sizes. Table C.1 displays the ARIMA forecast performance for varying sliding windows over 100 simulated series, as specified in Section 5.5.1. It can be seen that the difference in MASE is marginal between the window sizes. Here it is important to look at MASE as each model uses a different amount of data hence this scaling factor ensures fair comparison. These results suggest using small windows does not cause poorer performance over larger window sizes.

Table C.1: ARIMA forecast performance for varying sizes of sliding window over 100 simulated series generated as described in Section 5.5.1

	MAE	AE SD	MASE	ASE SD
30 Minutes	6.783	14.879	1.174	2.531
1 Hour	7.352	16.735	1.187	2.579
2 Hours	7.458	16.841	1.191	2.612
6 Hours	7.268	14.466	1.154	2.242
12 Hours	7.216	13.736	1.141	2.063

C.2 Regression Combination Coefficient Investigation

We now investigate the behaviour of the estimated coefficients of the linear regression model from Section 5.3.2 for a single simulation example. The single simulation is generated as described in Section 5.5.1 with results shown for the final 10 cycles of the data. Here the fixed forgetting factor has been set to $\gamma = 0.96$. The coefficients of the regression determine the influence of the short and long-term



(a) Raw Data **(b)** ARIMA Coefficient **(c)** FDA Coefficient

Figure C.1: Illustration of coefficient estimates for a single simulated series as described in Section 5.5.1. The simulation runs over 20 cycles where the final 10 cycles are shown. Point anomalies marked by crosses, contextual anomalies by solid rectangles.

forecasts on the regression combination forecast. Figure C.1 displays the coefficient values where the point and contextual anomalies are highlighted. Generally, ARIMA has larger coefficient values in comparison to FDA. Both have relatively small coefficient values with a number of outliers. For the contextual anomaly around 25921, it can be seen that FDA has large coefficients whilst ARIMA has negative coefficients, somewhat mirroring one another. This can be seen for other contextual anomalies where FDA has larger coefficient values during these periods. The other spikes in the values correspond to anomalies within the data where otherwise the coefficients are stable.

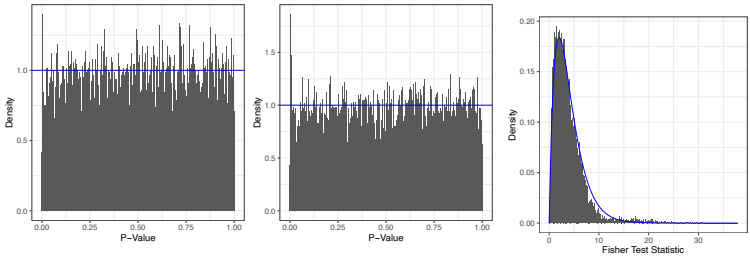
C.3 *P*-Value Investigation

C.3.1 SLD *P*-Values

Figure C.2 displays the distribution of the p-values and Fishers product test statistic for a single simulation generated as described in Section 5.5.1. The density of the reconstructed p-values and the long-term p-values are displayed in Figure C.2 (a) and (b) respectively. Both of these densities display a uniform distribution, hence these p-values are approximately uniform. Additionally, the density of the combined Fisher Test Statistic in Figure C.2 (c) approximately follows the prescribed chi-squared distribution. Due to these empirical results, this chi-squared distribution is used to calculate p-values for the Fisher combined score.

C.3.2 FDARIMA *P*-Values

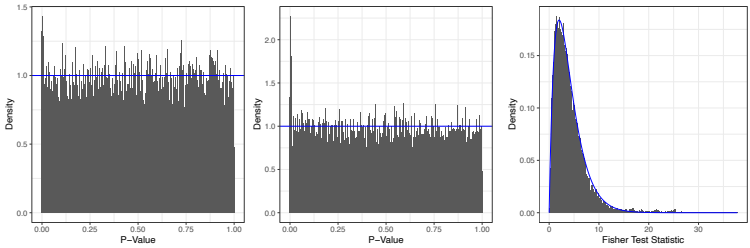
Figure C.3 displays histograms of the p-values and Fisher Test Statistic for FDARIMA for a single simulation generated as described in Section 5.5.1. The desired distri-



(a) Reconstructed series p-value **(b)** Long-term component p-value **(c)** Fisher Test Statistic

Figure C.2: Histograms showing distribution of the p-values and Fisher's Test Statistic for a single simulation as described in Section 5.5.1 for the SL Decomposition method.

butions for these quantities are plotted in blue. It can be seen that the p-values are approximately uniform and that the Fisher Test Statistic does follow the desired chi-squared distribution. Thus the p-values are well calibrated.



(a) ARIMA p-value **(b)** FDA p-value **(c)** Fisher Test Statistic

Figure C.3: Histograms showing distribution of the p-values and Fisher's Test Statistic for a single simulation as described in Section 5.5.1 for the FDARIMA approach.

C.4 Forecast Improvement Remove Anomalies

As anomalies are not representative of the "normal" behaviour of the series, these points should be removed. Table C.2 displays the comparison in forecast performance with and without anomalies in the model fitting for 100 simulated series as generated in Section 5.5.1. It can be seen that removing anomalies improves forecast performance, particularly for ARIMA and the forecast combination results.

Table C.2: Forecast performance with and without anomalous points within model fitting over 100 simulated data sets generated as described in Section 5.5.1

	MAE	AE SD	MASE	ASE SD
ARIMA Without Anom	6.513	14.926	1.179	2.694
ARIMA With Anom	7.343	16.720	1.186	2.569
FDA Without Anom	14.951	22.071	2.792	3.995
FDA With Anom	14.841	21.976	2.772	3.980
Regression Combination Without Anom	6.634	14.303	1.370	2.955
Regression Combination With Anom	7.634	28.152	1.577	5.815
SL Decomposition Without Anom	7.318	17.985	1.334	3.298
SL Decomposition With Anom	8.376	19.763	1.523	3.604
Naive	5.949	15.033	1.000	0.000

C.5 Comparison Method Description

We compare our method to ARIMA alone using the *tsoutlier* R Package López-de Lacalle, 2019. In this package, anomalies are found using the popular time series outlier procedure proposed by Chen and Liu, 1993, through iterative outlier detection and adjustment to produce joint estimates of the ARIMA model parameters and outlier effects. This procedure can identify multiple types of anomalies including additive outliers and temporary changes. However, this procedure is a **batch** approach. This method however does not take long-term patterns into account and is unsuitable for detecting contextual anomalies.

We also compare our procedures to the classic STL decomposition using the *anomalize* R package Dancho and Vaughan, 2020. First, the series is decomposed into trend and seasonal components using Loess before anomalies are detected using the inter quantile range on the residuals. Again this implementation is done in **batch**. This procedure differs from ours in that the trend is extracted before the seasonal behaviour, making the scale of the seasonal component different from the raw data. Additionally, contextual anomalies are not detected by this procedure.

Twitter Inc. have released an open source robust anomaly detection procedure appropriate for detecting both local (contextual) and global (point) anomalies in time series data Kejariwal, 2015, Hochenbaum et al., 2017. Their method named Seasonal Hybrid ESD (S-H-ESD) builds upon the Generalized ESD test Rosner, 1983. They combine a modified STL decomposition and ESD with robust statistics to detect both point and contextual anomalies. A downfall of this method however is that it does not perform well when the time series trend is changing Munir et al., 2018, and again is a **batch** approach. This method differs from traditional STL by utilising the median of the series to model the trend component before finding the seasonal component as done in STL using Loess. By replacing the mean and standard deviation by the median and MAD within the ESD test

to detect anomalies, their approach is more robust in the presence of large numbers of anomalies in the data. Unlike our method which additionally monitors the residuals of the long-term component, the Twitter method only analyses the decomposed residual.

SmartSifter [Yamanishi et al., 2004] is an online statistical learning outlier detector that uses a finite mixture model to represent the underlying data generating process. As new data arrives, the model is updated where past examples are gradually discounted. A score representing the change in the model after updating is used to assess the probability of the data being an outlier. We implement this model using the *smartsifter* Python library [1]. This model is however not specifically designed for contextual anomaly detection. Additionally, unlike our proposed methodology, the long-term behaviours or periodicity are not directly modelled.

Finally, a new R package specifically targeted at providing a range of methods for online time series outlier detection called *otsad* [Iturria et al., 2020] has been developed. A number of methods are implemented including TSSD-EWMA [Raza et al., 2015]. TSSD-EWMA works by detecting covariate shifts using an exponentially weighted moving average (EWMA) based control chart. These shifts are then validated using a Kolmogorov-Smirnov statistical hypothesis test. Seasonal behaviours are again not considered by this model thus similar to tsoutliers, anomalize and smartsifter, this method is not specifically designed for contextual anomaly detection.

C.6 BFF as Short-Term Model Results

We now present the forecast and anomaly detection results when BFF, the adaptive estimation procedure described in Chapter 3, is used instead of ARIMA as the short-term model. As BFF is sequential and provides up to date estimates, this method is very fast with very small computation requirements. Table C.3 displays the forecast performance when BFF is used instead of ARIMA. In comparison with the results in Table 5.1, BFF has much higher errors where the errors for all combined forecasts are additionally higher. This suggests that the Poisson BFF may not be suitable for the simulated data.

Table C.4 displays the anomaly detection performance for FDABFF (replacing ARIMA with BFF in FDARIMA anomaly detection procedure) and SLD (again replacing ARIMA with BFF in this procedure). The results are averaged over the same 100 simulations used for Table 5.3 hence the results are comparable. The overall F1 for SLD is higher using BFF however it is lower for FDABFF. This increase in F1 is attributed to a higher recall value. The point anomaly performance is lower when using BFF and the contextual performance is only higher for SL Decomposition. These results suggest it is preferable to utilise ARIMA for the short-term model.

[1] https://pypi.org/project/smartsifter/

Table C.3: Forecast performance when BFF used as short-term model over 100 simulated data sets generated as described in Section 5.5.1.

	MAE	AE SD	MASE	ASE SD
BFF	66.042	36.230	12.118	7.631
FDA	14.952	22.081	2.792	3.997
Regression Combination	9.011	14.029	1.861	2.898
SLD	16.189	25.896	2.943	4.716
Naive	5.949	15.033	1.000	0.000

Table C.4: Anomaly detection performance when replacing ARIMA with BFF in FDARIMA and SLD on 100 simulated data sets generated as described in Section 5.5.1. The performance is calculated overall (for both types of anomalies) and additionally splits the performance for point and contextual anomalies individually.

		FDABFF	SLD
Overall	F1	0.531	0.717
	Recall	0.399	0.635
	Precision	0.830	0.836
Point	F1	0.692	0.608
	Recall	0.938	1.000
	Precision	0.561	0.457
Context	F1	0.471	0.686
	Recall	0.347	0.600
	Precision	0.783	0.815

C.7 Additional Figures for Anomalize, tsoutliers and TSSD-EWMA

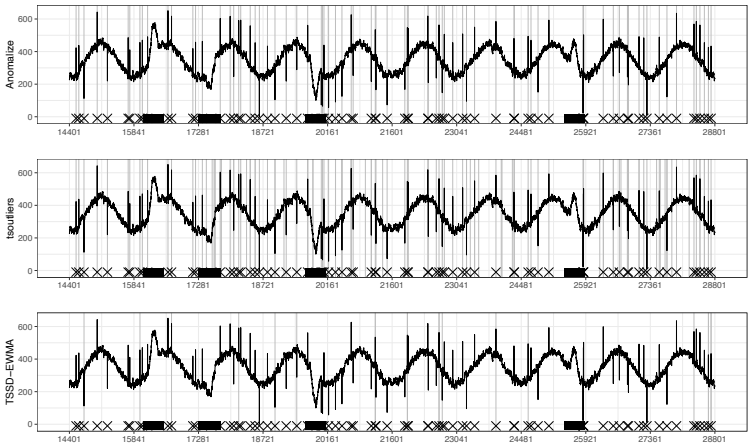


Figure C.4: Locations of identified anomalies for Anomalize, tsoutliers and TSSD-EWMA models identified with vertical grey lines and are compared to the positions of the true anomalous points marked by crosses for point anomalies and solid rectangles for contextual anomaly periods. The series shows the final 10 days of the series.

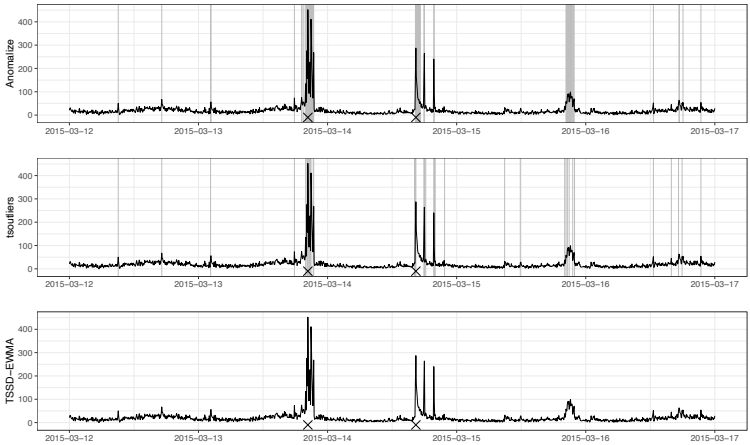


Figure C.5: Comparison of detected anomaly locations for Anomalize, tsoutliers and TSSD-EWMA methods where true anomalous locations marked with a cross for the Google Twitter mention data between 12th March 2015-16th March 2015. Detected anomalies by the algorithms are identified by grey vertical lines.

Appendix D

Multi-Type Clustering Appendix

D.1 Simple NMTF Derivations

In the following the derivations for the updates of the alternating optimisation approach used for Simple NMTF is detailed in full. Recall the objective function for Simple NMTF

$$J = \sum_{i=1}^{q} ||\mathbf{A}^{(i)}_{k(i)k(i)} - \mathbf{G}_{k(i)}\mathbf{S}^{(i)}_{k(i)k(i)}||_F^2 + \sum_{i=1}^{r} ||\mathbf{A}^{(i)}_{l(i)m(i)} - \mathbf{G}_{l(i)}\mathbf{S}^{(i)}_{l(i)m(i)}\mathbf{G}^T_{m(i)}||_F^2$$
$$\text{s.t. } \mathbf{G}_z \in \Psi^{n_z \times c_z},\ \mathbf{S}_{zw} \in \mathbb{R}^{c_z \times c_w}\ \&\ z, w \in Z$$

As this function is not convex over all matrices, the solution is found by iteratively solving for each matrix separately until convergence. First, by fixing all matrices but $\mathbf{S}^{(i)}_{k(i)k(i)}$ we take the derivative of the objective with respect to $\mathbf{S}^{(i)}_{k(i)k(i)}$,

$$\begin{aligned}
\frac{\partial J}{\partial \mathbf{S}^{(i)}_{k(i)k(i)}} &= \frac{\partial}{\partial \mathbf{S}^{(i)}_{k(i)k(i)}} \left|\left|\mathbf{A}^{(i)}_{k(i)k(i)} - \mathbf{G}_{k(i)}\mathbf{S}^{(i)}_{k(i)k(i)}\right|\right|_F^2 \\
&= \frac{\partial}{\partial \mathbf{S}^{(i)}_{k(i)k(i)}} tr\left[\left(\mathbf{A}^{(i)}_{k(i)k(i)} - \mathbf{G}_{k(i)}\mathbf{S}^{(i)}_{k(i)k(i)}\right)^T \left(\mathbf{A}^{(i)}_{k(i)k(i)} - \mathbf{G}_{k(i)}\mathbf{S}^{(i)}_{k(i)k(i)}\right)\right] \\
&= \frac{\partial}{\partial \mathbf{S}^{(i)}_{k(i)k(i)}} tr\left[\mathbf{A}^{(i)^T}_{k(i)k(i)}\mathbf{A}^{(i)}_{k(i)k(i)} - 2\mathbf{A}^{(i)^T}_{k(i)k(i)}\mathbf{G}_{k(i)}\mathbf{S}^{(i)}_{k(i)k(i)} + \mathbf{S}^{(i)^T}_{k(i)k(i)}\mathbf{G}^T_{k(i)}\mathbf{G}_{k(i)}\mathbf{S}^{(i)}_{k(i)k(i)}\right] \\
&= tr\left[-2\mathbf{A}^{(i)^T}_{k(i)k(i)}\mathbf{G}_{k(i)} + 2\mathbf{S}^{(i)^T}_{k(i)k(i)}\mathbf{G}^T_{k(i)}\mathbf{G}_{k(i)}\right]
\end{aligned}$$

and setting the derivative this to zero gives,

$$\mathbf{S}^{(i)}_{k(i)k(i)} = (\mathbf{G}^T_{k(i)}\mathbf{G}_{k(i)})^{-1}\mathbf{G}^T_{k(i)}\mathbf{A}^{(i)}_{k(i)k(i)}.$$

Similarly, consider when all matrices but the inter $\mathbf{S}^{(i)}_{l(i)m(i)}$ matrix are held fixed,

the derivative of the objective with respect to this matrix is,

$$\begin{aligned}
\frac{\partial J}{\partial \mathbf{S}^{(i)}_{l(i)m(i)}} &= \frac{\partial}{\partial \mathbf{S}^{(i)}_{l(i)m(i)}} \left|\left| \mathbf{A}^{(i)}_{l(i)m(i)} - \mathbf{G}_{l(i)}\mathbf{S}^{(i)}_{l(i)m(i)}\mathbf{G}^T_{m(i)} \right|\right|^2_F \\
&= \frac{\partial}{\partial \mathbf{S}^{(i)}_{l(i)m(i)}} tr\left[\left(\mathbf{A}^{(i)}_{l(i)m(i)} - \mathbf{G}_{l(i)}\mathbf{S}^{(i)}_{l(i)m(i)}\mathbf{G}^T_{m(i)} \right)^T \left(\mathbf{A}^{(i)}_{l(i)m(i)} - \mathbf{G}_{l(i)}\mathbf{S}^{(i)}_{l(i)m(i)}\mathbf{G}^T_{m(i)} \right) \right] \\
&= \frac{\partial}{\partial \mathbf{S}^{(i)}_{k(i)k(i)}} tr\left[\mathbf{A}^{(i)^T}_{l(i)m(i)}\mathbf{A}^{(i)}_{l(i)m(i)} - 2\mathbf{G}^T_{l(i)}\mathbf{A}^{(i)}_{l(i)m(i)}\mathbf{G}_{m(i)}\mathbf{S}^{(i)}_{l(i)m(i)} \right. \\
&\qquad \left. + \mathbf{G}^T_{m(i)}\mathbf{G}_{m(i)}\mathbf{S}^{(i)^T}_{l(i)m(i)}\mathbf{G}^T_{l(i)}\mathbf{G}_{l(i)}\mathbf{S}^{(i)}_{l(i)m(i)} \right] \\
&= tr\left[-2\mathbf{G}^T_{l(i)}\mathbf{A}^{(i)}_{l(i)m(i)}\mathbf{G}_{m(i)} + 2\mathbf{G}^T_{l(i)}\mathbf{G}_{l(i)}\mathbf{S}^{(i)}_{l(i)m(i)}\mathbf{G}^T_{m(i)}\mathbf{G}_{m(i)} \right]
\end{aligned}$$

and setting this to zero gives,

$$\mathbf{S}^{(i)}_{l(i)m(i)} = (\mathbf{G}^T_{l(i)}\mathbf{G}_{l(i)})^{-1}\mathbf{G}^T_{l(i)}\mathbf{A}^{(i)}_{l(i)m(i)}\mathbf{G}_{m(i)}(\mathbf{G}^T_{m(i)}\mathbf{G}_{m(i)})^{-1}.$$

To calculate the minima of the objective with respect to $\mathbf{G}_z$ for $z \in Z$ whilst keeping all other matrices fixed, the problem is decoupled to the following for each node i $(1 \leq i \leq n_z)$,

$$\begin{aligned}
\min_{\mathbf{G}_z \in \Psi} & \sum_{j \in \{t \in \{1,\ldots,q\} | k(t)=z\}} ||\mathbf{A}^{(j)^{i\cdot}}_{k(j)k(j)} - \mathbf{G}^{i\cdot}_z \mathbf{S}^{(j)}_{k(j)k(j)}||^2_F \\
& + \sum_{j \in \{t \in \{1,\ldots,r\} | l(t)=z\}} ||\mathbf{A}^{(j)^{i\cdot}}_{l(j)m(j)} - \mathbf{G}^{i\cdot}_z \mathbf{S}^{(j)}_{l(j)m(j)}\mathbf{G}^T_{m(j)}||^2_F \\
& + \sum_{j \in \{t \in \{1,\ldots,r\} | m(t)=z\}} ||\mathbf{A}^{(j)^{\cdot i}}_{l(j)m(j)} - \mathbf{G}_{l(j)}\mathbf{S}^{(j)}_{l(j)m(j)}(\mathbf{G}^T_z)^{\cdot i}||^2_F
\end{aligned}$$

as $\mathbf{G}_z$ is a cluster indicator matrix. Thus the best cluster allocation j for node i is such that it minimises this objective function hence,

$$\mathbf{G}^{ij}_z = \begin{cases} 1 & j = \text{argmin}_s \sum_{p \in \{t \in \{1,\ldots,q\} | k(t)=z\}} ||\mathbf{A}^{(p)^{i\cdot}}_{k(p)k(p)} - \mathbf{S}^{(p)^{s\cdot}}_{k(p)k(p)}||^2 \\ & \qquad + \sum_{p \in \{t \in \{1,\ldots,r\} | l(t)=z\}} ||\mathbf{A}^{(p)^{i\cdot}}_{l(p)m(p)} - \left(\mathbf{S}^{(p)}_{l(p)m(p)}\mathbf{G}^T_{m(p)}\right)^{s\cdot}||^2 \\ & \qquad + \sum_{p \in \{t \in \{1,\ldots,r\} | m(t)=z\}} ||\mathbf{A}^{(p)^{\cdot i}}_{l(p)m(p)} - \left(\mathbf{G}_{l(p)}\mathbf{S}^{(p)}_{l(p)m(p)}\right)^{\cdot s}||^2 \\ 0 & \text{otherwise} \end{cases}.$$

D.2 Weighted Simple NMTF Derivations

Alternating optimisation is again used to find a solution for the Weighted Simple NMTF non convex objective function,

$$
\begin{aligned}
J = &\sum_{i=1}^{q} ||\mathbf{A}^{(i)}_{k(i)k(i)} - \mathbf{G}^{(i)}_{k(i)}\mathbf{S}^{(i)}_{k(i)k(i)}||_F^2 + \sum_{i=q+1}^{q+r} ||\mathbf{A}^{(i)}_{l(i)m(i)} - \mathbf{G}^{(i)}_{l(i)}\mathbf{S}^{(i)}_{l(i)m(i)}\mathbf{G}^{(i)^T}_{m(i)}||_F^2 \\
&+ \sum_{z \in Z} \lambda_z \, ||\mathbf{G}^*_z - \mathbf{P}_z\mathbf{M}_z||_F^2 \text{ s.t. } \lambda_z \geq 0, \; \mathbf{G}^{(i)}_z \in \Psi^{n_z \times c_z}, \; \mathbf{S}^{(i)}_{zw} \in \mathbb{R}^{c_z \times c_w}, \\
&\mathbf{G}^*_z \in \mathbb{R}^{|I_z|c_z \times n_z}, \; \mathbf{P}_z \in \mathbb{R}^{|I_z|c_z \times c_z}, \; \mathbf{M}_z \in \mathbb{R}^{c_z \times n_z} \text{ for } z, w \in Z
\end{aligned}
$$

where recall the matrix $\mathbf{G}^*_z$ row concatenates all $\mathbf{G}^{(i)^T}_z$ $\forall i \in I_z$ where I_z contains the index set of matrices related to z.

Similarly to the Simple NMTF algorithm in Appendix D.1, the optimal solution for the matrix $\mathbf{S}^{(i)}_{k(i)k(i)}$ and $\mathbf{S}_{l(i}m(i)^{(i)}$ is calculated by taking the derivative of the objective function and setting to zero. First we take the derivative of the objective function with respect to $\mathbf{S}^{(i)}_{k(i)k(i)}$ while keeping all other matrices fixed,

$$
\begin{aligned}
\frac{\partial J}{\partial \mathbf{S}^{(i)}_{k(i)k(i)}} &= \frac{\partial}{\partial \mathbf{S}^{(i)}_{k(i)k(i)}} \left|\left| \mathbf{A}^{(i)}_{k(i)k(i)} - \mathbf{G}^{(i)}_{k(i)}\mathbf{S}^{(i)}_{k(i)k(i)} \right|\right|_F^2 \\
&= \frac{\partial}{\partial \mathbf{S}^{(i)}_{k(i)k(i)}} tr\left[\left(\mathbf{A}^{(i)}_{k(i)k(i)} - \mathbf{G}^{(i)}_{k(i)}\mathbf{S}^{(i)}_{k(i)k(i)} \right)^T \left(\mathbf{A}^{(i)}_{k(i)k(i)} - \mathbf{G}^{(i)}_{k(i)}\mathbf{S}^{(i)}_{k(i)k(i)} \right) \right] \\
&= \frac{\partial}{\partial \mathbf{S}^{(i)}_{k(i)k(i)}} tr\left[\mathbf{A}^{(i)^T}_{k(i)k(i)}\mathbf{A}^{(i)}_{k(i)k(i)} - 2\mathbf{A}^{(i)^T}_{k(i)k(i)}\mathbf{G}^{(i)}_{k(i)}\mathbf{S}^{(i)}_{k(i)k(i)} + \mathbf{S}^{(i)^T}_{k(i)k(i)}\mathbf{G}^{(i)^T}_{k(i)}\mathbf{G}^{(i)}_{k(i)}\mathbf{S}^{(i)}_{k(i)k(i)} \right] \\
&= tr\left[-2\mathbf{A}^{(i)^T}_{k(i)k(i)}\mathbf{G}^{(i)}_{k(i)} + 2\mathbf{S}^{(i)^T}_{k(i)k(i)}\mathbf{G}^{(i)^T}_{k(i)}\mathbf{G}^{(i)}_{k(i)} \right].
\end{aligned}
$$

Setting this to zero gives,

$$
\mathbf{S}^{(i)}_{k(i)k(i)} = (\mathbf{G}^{(i)^T}_{k(i)}\mathbf{G}^{(i)}_{k(i)})^{-1}\mathbf{G}^{(i)^T}_{k(i)}\mathbf{A}^{(i)}_{k(i)k(i)}.
$$

which is similar to the solution for Simple NMTF however instead of a single clustering for entity type $k(i)$ the individual clustering for graph $\mathbf{A}^{(i)}_{k(i)k(i)}$ is used.

Similarly for $\mathbf{S}^{(i)}_{l(i)m(i)}$ the derivative of the objective with respect to this matrix is,

$$\begin{aligned}
\frac{\partial J}{\partial \mathbf{S}^{(i)}_{l(i)m(i)}} &= \frac{\partial}{\partial \mathbf{S}^{(i)}_{l(i)m(i)}} \left|\left|\mathbf{A}^{(i)}_{l(i)m(i)} - \mathbf{G}^{(i)}_{l(i)}\mathbf{S}^{(i)}_{l(i)m(i)}\mathbf{G}^{(i)^T}_{m(i)}\right|\right|^2_F \\
&= \frac{\partial}{\partial \mathbf{S}^{(i)}_{l(i)m(i)}} tr\left[\left(\mathbf{A}^{(i)}_{l(i)m(i)} - \mathbf{G}^{(i)}_{l(i)}\mathbf{S}^{(i)}_{l(i)m(i)}\mathbf{G}^{(i)^T}_{m(i)}\right)^T \left(\mathbf{A}^{(i)}_{l(i)m(i)} - \mathbf{G}^{(i)}_{l(i)}\mathbf{S}^{(i)}_{l(i)m(i)}\mathbf{G}^{(i)^T}_{m(i)}\right)\right] \\
&= \frac{\partial}{\partial \mathbf{S}^{(i)}_{k(i)k(i)}} tr\left[\mathbf{A}^{(i)^T}_{l(i)m(i)}\mathbf{A}^{(i)}_{l(i)m(i)} - 2\mathbf{G}^{(i)^T}_{l(i)}\mathbf{A}^{(i)}_{l(i)m(i)}\mathbf{G}^{(i)}_{m(i)}\mathbf{S}^{(i)}_{l(i)m(i)}\right. \\
&\qquad \left. + \mathbf{G}^{(i)^T}_{m(i)}\mathbf{G}^{(i)}_{m(i)}\mathbf{S}^{(i)^T}_{l(i)m(i)}\mathbf{G}^{(i)^T}_{l(i)}\mathbf{G}^{(i)}_{l(i)}\mathbf{S}^{(i)}_{l(i)m(i)}\right] \\
&= tr\left[-2\mathbf{G}^{(i)^T}_{l(i)}\mathbf{A}^{(i)}_{l(i)m(i)}\mathbf{G}^{(i)}_{m(i)} + 2\mathbf{G}^{(i)^T}_{l(i)}\mathbf{G}^{(i)}_{l(i)}\mathbf{S}^{(i)}_{l(i)m(i)}\mathbf{G}^{(i)^T}_{m(i)}\mathbf{G}^{(i)}_{m(i)}\right]
\end{aligned}$$

and setting this to zero gives,

$$\mathbf{S}^{(i)}_{l(i)m(i)} = (\mathbf{G}^{(i)^T}_{l(i)}\mathbf{G}^{(i)}_{l(i)})^{-1}\mathbf{G}^{(i)^T}_{l(i)}\mathbf{A}^{(i)}_{l(i)m(i)}\mathbf{G}^{(i)}_{m(i)}(\mathbf{G}^{(i)^T}_{m(i)}\mathbf{G}^{(i)}_{m(i)})^{-1}.$$

To solve for the optimal cluster indicator matrix for each graph, first consider the intra relationships. The problem for $\mathbf{G}^{(i)}_{k(i)}$ is decoupled to,

$$\min_{\mathbf{G}\in\Psi} ||\mathbf{A}^{(i)}_{k(i)k(i)} - \mathbf{G}\mathbf{S}^{(i)}_{k(i)k(i)}||^2_F + \lambda_{k_{(i)}} \left|\left|\mathbf{G}^T - (\mathbf{P}_{k(i)}\mathbf{M}_{k(i)})^{I^{(i)}_{k(i)}\cdot}\right|\right|^2_F$$

where $I^{(i)}_z$ is the set of indices for graph i in the combined concatenated matrix $\mathbf{G}^*_z$. Let $\mathbf{g}(k)$ be a cluster indicator vector of length n where where the kth entry is 1 and all other entries are zero. Let $\mathbf{b}$ be a real vector. Consider the following minimisation problem,

$$\begin{aligned}
\min_k ||\mathbf{g}(k) - \mathbf{b}||^2 &= \min_k (1 - \mathbf{b}^k)^2 + \sum_{j=1, j\neq k}^{n} \mathbf{b}^{j^2} \\
&= \min_k 1 - 2\mathbf{b}^k + \mathbf{b}^{k^2} + \sum_{j=1, j\neq k}^{n} \mathbf{b}^{j^2} \\
&= \min_k 1 - 2\mathbf{b}^k + ||\mathbf{b}||^2 \\
&= -2\mathbf{b}^k.
\end{aligned}$$

Thus as $\mathbf{G}$ is a cluster indicator matrix, for each node j, choose the cluster s that minimises the objective function,

$$\mathbf{G}^{(i)^{jt}}_{k(i)} = \begin{cases} 1 & t = \text{argmin}_s ||\mathbf{A}^{(i)^{j\cdot}}_{k(i)k(i)} - \mathbf{S}^{(i)^{s\cdot}}_{k(i)k(i)}||^2 - 2\lambda_{k(i)} \left(\mathbf{P}_{k(i)}\mathbf{M}_{k(i)}\right)^{I^{(i)}_{k(i)}(s)j} \\ 0 & \text{otherwise} \end{cases}$$

where $I_z^{(i)}(s)$ is the index of cluster s for entity type z from input matrix i in combined concatenated matrix $\mathbf{G}_z^*$. Similarly for the inter-type relational matrices, the optimal cluster indicator matrix for $\mathbf{G}_{l(i)}^{(i)}$ solves,

$$\min_{\mathbf{G}\in\Psi} ||\mathbf{A}_{l(i)m(i)}^{(i)} - \mathbf{G}\mathbf{S}_{l(i)m(i)}^{(i)}\mathbf{G}_{m(i)}^{(i)^T}||_F^2 + \lambda_{l_{(i)}} \left|\left|\mathbf{G}^T - (\mathbf{P}_{l(i)}\mathbf{M}_{l(i)})^{I_{l(i)}^{(i)}\cdot}\right|\right|_F^2$$

thus similarly to the intra case, for the jth node find cluster s to minimise this which is given by,

$$\mathbf{G}_{l(i)}^{(i)^{jt}} = \begin{cases} 1 & t = \text{argmin}_s \, ||\mathbf{A}_{l(i)m(i)}^{(i)^{j\cdot}} - \left(\mathbf{S}_{l(i)m(i)}^{(i)}\mathbf{G}_{m(i)}^{(i)^T}\right)^{s\cdot} ||^2 - 2\lambda_{l(i)} \left(\mathbf{P}_{l(i)}\mathbf{M}_{l(i)}\right)^{I_{l(i)}^{(i)}(s)j} \\ 0 & \text{otherwise} \end{cases}$$

Similarly for $\mathbf{G}_{m(i)}^{(i)}$ the problem is decoupled to,

$$\min_{\mathbf{G}\in\Psi} ||\mathbf{A}_{l(i)m(i)}^{(i)} - \mathbf{G}_{l(i)}^{(i)}\mathbf{S}_{l(i)m(i)}^{(i)}\mathbf{G}^T||_F^2 + \lambda_{m_{(i)}} \left|\left|\mathbf{G}^T - (\mathbf{P}_{m(i)}\mathbf{M}_{m(i)})^{I_{m(i)}^{(i)}\cdot}\right|\right|_F^2$$

which is solved by,

$$\mathbf{G}_{m(i)}^{(i)^{jt}} = \begin{cases} 1 & t = \text{argmin}_s \, ||\mathbf{A}_{l(i)m(i)}^{(i)\cdot j} - \left(\mathbf{G}_{l(i)}^{(i)}\mathbf{S}_{l(i)m(i)}^{(i)}\right)^{\cdot s} ||^2 - 2\lambda_{m(i)} \left(\mathbf{P}_{m(i)}\mathbf{M}_{m(i)}\right)^{I_{m(i)}^{(i)}(s)j} \\ 0 & \text{otherwise} \end{cases}$$

where $I_z^{(i)}(s)$ is the index of cluster s for entity type z from input matrix i in combined concatenated matrix $\mathbf{G}_z^*$.

D.3 Weighted Simple NMTF Regularisation Parameter

To investigate the influence of the regularisation parameter on the performance of the Weighted Simple NMTF clustering procedure, we calculate the performance in terms of similarity to the true clustering via the ARI. The graphs used for this method are generated using the stochastic block model procedure detailed in Section 6.3 where we vary the sizes of the graphs and the probability of connection between the nodes. As our methods are motivated by cyber-security, the graphs reflect this. We generate computer-computer, computer-user and computer-port graphs where there is a shared computer clustering between these relational matrices. Table D.1 outlines the parameters sampled from to generate the random graphs. For each regularisation parameter λ the results average over 50 random graphs where the methods utilise the initialisation procedure outlined in Section 6.5.

As the Weighted Simple NMTF calculates both individual computer clusterings for each graph alongside the combined clustering, we can determine how well each

Table D.1: Description simulation graph parameters for the Weighted Simple NMTF parameter investigation.

Number Computers / Clusters	{3000,5000,7000} / {100,200,300}
Number Users / Clusters	{3000,5000,7000} / {100,200,300}
Number Ports / Clusters	{1000,2000} / {50,100}
CC Number Cluster Connections	{1,2,. . . ,5}
CU Number Cluster Connections	{1,2,. . . ,5}
CP Number Cluster Connections	{1,2,. . . ,10}
Probability of Connections Between Clusters	{0.1,. . . ,0.9}

individual clustering compares to the combined. Additionally using Equation 6.11 the contribution of the individual clusterings from each graph can be calculated. Table D.3 shows the ARI between the Weighted Simple NMTF clustering and the true clustering for each individual graph for the computers (CC, CU and CP) and the combined clustering, Comp, plus the performance for the users and ports. Finally, it displays the weights each graph contributes to the final clustering. For comparison Table D.3 contains the similarity of Simple NMTF to the true clustering calculated over the same graphs split for each of the different λ values although Simple NMTF does not utilise this parameter.

Table D.2: ARI of clustering from Weighted Simple NMTF to true clustering calculated over 50 random graphs for each parameter.

λ	CC	CU	CP	Comp	User	Port	CC Weight	CU Weight	CP Weight
0.0001	0.6116	0.4743	0.6566	0.6405	0.6825	0.7607	0.3799	0.3695	0.2506
0.0005	0.6268	0.5140	0.6680	0.6284	0.6711	0.7579	0.3779	0.3724	0.2496
0.001	0.6144	0.4931	0.6649	0.6520	0.7143	0.7520	0.3702	0.3738	0.2560
0.005	0.6230	0.5014	0.6653	0.6454	0.7085	0.7641	0.3711	0.3694	0.2595
0.01	0.6073	0.4910	0.6495	0.6461	0.7062	0.7597	0.3702	0.3729	0.2569
0.05	0.5941	0.4840	0.6283	0.6600	0.7314	0.7673	0.3587	0.3692	0.2721
0.1	0.6014	0.5069	0.5901	0.6579	0.6777	0.7717	0.3435	0.3764	0.2801
0.5	0.5886	0.6017	0.4012	0.6273	0.7315	0.7865	0.3344	0.3766	0.2890
1.0	0.5534	0.5993	0.3332	0.5881	0.7084	0.7763	0.3338	0.3659	0.3003
5.0	0.5189	0.5837	0.3219	0.5553	0.7269	0.7787	0.3360	0.3680	0.2961
10.0	0.5131	0.5783	0.3205	0.5755	0.6635	0.7819	0.3333	0.3638	0.3030

From Table D.2 it is clear that for all values of λ below 0.5, the combined computer clustering of WSNMTF generally outperforms Simple NMTF while the user and port clusterings have very similar performance. This suggests appropriate values for λ lie in $(0, 0.5)$. It is interesting to note that the CU clustering for WSNMTF is poorer than that of CC and CP although this graph has been given large weight for small parameter values. Surprisingly CP has the smallest weight although it has the best performance for small parameter values. Thus similar to boosting, greater weight is given to the representation that performs weakly. Table D.4 displays the similarity between the clusterings from the weighted and

unweighted Simple NMTF algorithms. It is clear from this table that the computer clusterings differ greatly however the user and port clusterings are the same. Interestingly, these two approaches have similar ARI values to the truth yet are different from one another.

Table D.3: ARI of clustering from Simple NMTF to true clustering calculated over 50 random graphs for each parameter for the Weighted NMTF for comparability.

λ	Comp	User	Port
0.0001	0.5829	0.6961	0.7628
0.0005	0.6308	0.6849	0.7561
0.001	0.5981	0.7127	0.7532
0.005	0.6094	0.7106	0.7602
0.01	0.5851	0.715	0.7609
0.05	0.5283	0.7347	0.7641
0.1	0.5029	0.6804	0.7637
0.5	0.6323	0.7618	0.7629
1.0	0.5988	0.7324	0.7591
5.0	0.6123	0.7545	0.7624
10.0	0.6112	0.7102	0.7643

Table D.4: ARI between Simple NMTF and Weighted Simple NMTF calculated over 50 random graphs for each parameter.

Gamma	CC	CU	CP	Comp	User	Port
0.0001	0.4575	0.4571	0.4558	0.4458	0.7945	0.9225
0.0005	0.4894	0.4784	0.4886	0.4690	0.7940	0.9266
0.001	0.4653	0.4713	0.4659	0.4597	0.8075	0.9159
0.005	0.4776	0.4738	0.4757	0.4651	0.8058	0.9174
0.01	0.4631	0.4730	0.4521	0.4611	0.7919	0.9220
0.05	0.4209	0.4591	0.4073	0.4312	0.7947	0.9025
0.1	0.3953	0.4471	0.3675	0.4180	0.7761	0.8955
0.5	0.4428	0.5090	0.3089	0.4642	0.7964	0.8860
1.0	0.4002	0.4722	0.2530	0.4262	0.7823	0.8804
5.0	0.3914	0.4672	0.2490	0.4093	0.7849	0.8802
10.0	0.3898	0.4670	0.2498	0.4191	0.7389	0.8851

D.4 Bipartite Modularity

Consider a bipartite graph $G = (V, U, E)$ with adjacency matrix $\mathbf{A}$ where $|E| = M$. Let C^V and C^U be the clusterings for the two nodes sets V and U respectively. Let e_{lm} be the fraction of edges that link nodes in the lth cluster of C^V (denoted C_l^V) to nodes in the mth cluster of C^U (denoted C_m^U) calculated as,

$$e_{lm} = \frac{1}{2M} \sum_{i \in C_l^V} \sum_{j \in C_m^U} \mathbf{A}^{ij}.$$

Let a_l be the row sums of e,

$$a_l = \sum_m e_{lm} = \frac{1}{2M} \sum_{i \in C_l^V} \sum_{j \in U} \mathbf{A}^{ij}.$$

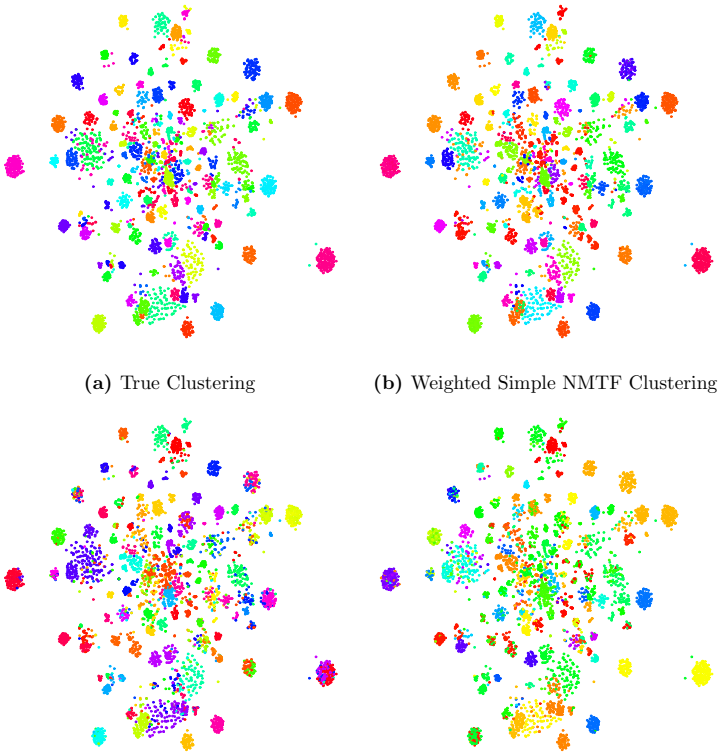
The bipartite modularity is then given by,

$$Q_B = \sum_l Q_{B_l} = \sum_l (e_{lm} - a_l a_m), \quad m = \operatorname*{argmax}_k e_{lk}.$$

This bipartite modularity measures the proportions of edges in the network that connect between communities minus the expected value of the same quantity in a network with the same community division but random connections. However as mentioned, as this measure is for community detection it is not a suitable internal measure for our problem which focus on clustering based on similar edges rather than dividing the network into communities.

D.5 Additional t-SNE Representations

This section displays the t-SNE comparison plots for the Weighted Simple NMTF, F-NMTF and R-NMTF multi-type clustering models to the true clustering for each node type. These visualisations use the same graph investigated in Section 6.7. Figure D.1 displays the computer clustering for the algorithms. It is clear that the competitive approaches, particularly F-NMTF, do not cluster the data as well as the proposed approaches where there is more mixing of colours. Again for the users in Figure D.2, there is less similarity in colouring to the truth for F-NMTF and R-NMTF which is validated by the poorer user performance of these clustering approaches in Table 6.4. Again F-NMTF has poor port performance in Figure D.3 with mixing of cluster assignments whereas the other algorithms better distinguish these clusters.



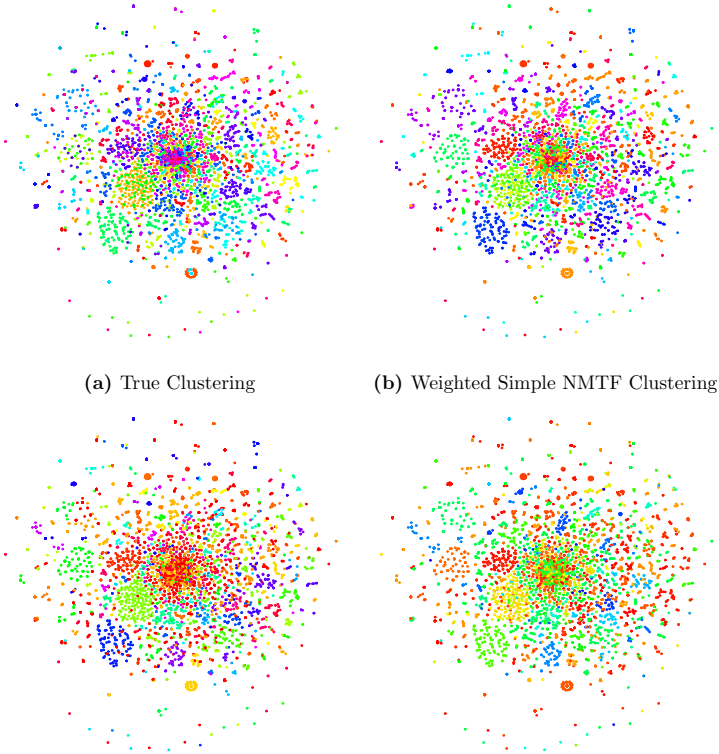
(a) True Clustering

(b) Weighted Simple NMTF Clustering

(c) F-NMTF

(d) R-NMTF

Figure D.1: Computer t-SNE representation for true, Weighted Simple NMTF, F-NMTF and R-NMTF clustering procedures respectively. Each colour represents a different cluster.



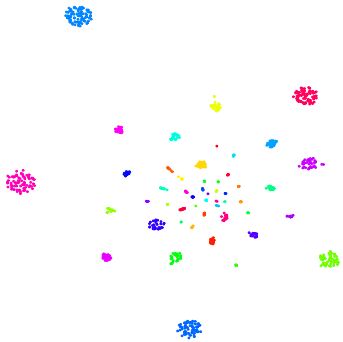
(a) True Clustering

(b) Weighted Simple NMTF Clustering

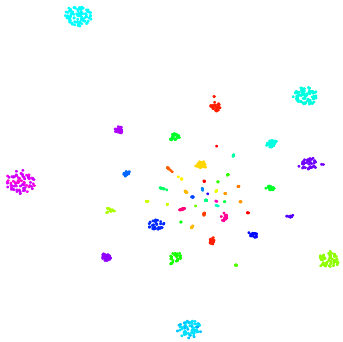
(c) F-NMTF

(d) R-NMTF

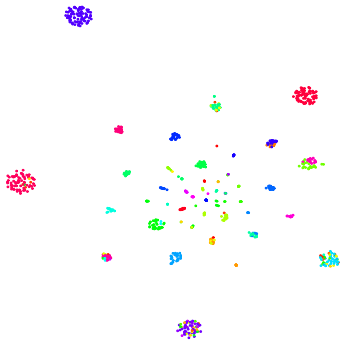
Figure D.2: User t-SNE representation for true, Weighted Simple NMTF, F-NMTF and R-NMTF clustering procedures respectively. Each colour represents a different cluster.



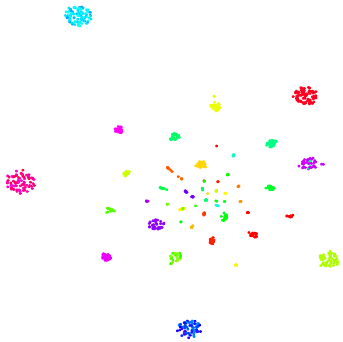
(a) True Clustering



(b) Weighted Simple NMTF Clustering



(c) F-NMTF



(d) R-NMTF

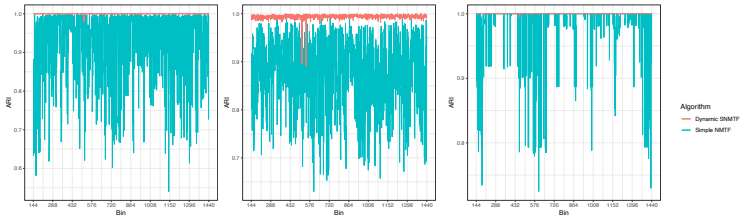
Figure D.3: Port t-SNE representation for true, Weighted Simple NMTF, F-NMTF and R-NMTF clustering procedures respectively. Each colour represents a different cluster.

Appendix E

Dynamic Clustering Appendix

E.1 No Cluster Change

We begin by assessing the clustering performance over a graph stream with no cluster changes as a baseline. The cluster performance should be consistent over the stream with high similarity between consecutive clusterings and very few cluster changes. Figure E.1 shows the ARI between consecutive clusterings for each node type for our two approaches. It can be seen that Dynamic SNMTF which utilises the previous clustering when updating, has very high similarity. Simple NMTF which performs NNDSVD initialisation at each update has less similarity between consecutive clusterings. Thus our proposed approach produces smooth clusterings.

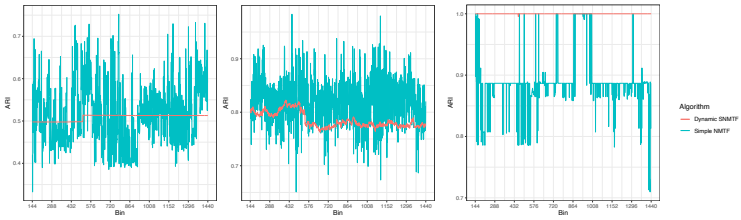


(a) Computer **(b)** User **(c)** Port

Figure E.1: ARI of consecutive clusterings over network stream as described in Section 7.2.1 where there exist no cluster changes.

We now explore the performance of our approaches. Figure E.2 displays the similarity (measured by ARI) to the true clustering. For the computer clustering, the performance of Dynamic SNMTF averages between the extremes of Simple NMTF. Additionally, the clustering performance for Dynamic SNMTF is much more stable for all node types. Surprisingly the port clustering has an ARI very close to 1 for Dynamic SNMTF, suggesting it has found the correct clustering.

Interestingly, both algorithms have lower ARI for the computer clusterings in comparison to the user and port clusterings. Poorer computer clustering was also seen in the simulations in Chapter 6. Thus Dynamic SNMTF provides a good balance between the smoothness of clustering and performance.

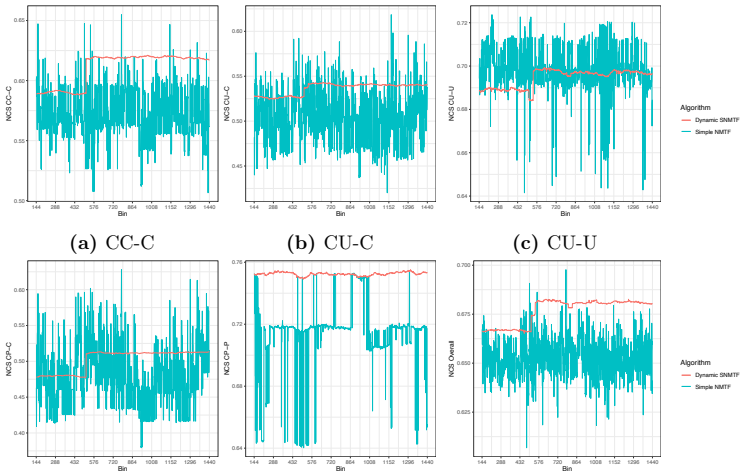


(a) Computer **(b)** User **(c)** Port

Figure E.2: ARI of consecutive clusterings over network stream as described in Section 7.2.1 where there exist no cluster changes.

As labels are not available in practice it is also important to utilise internal performance measures. Figure E.3 shows the NCS measure for each of the graphs and node types as well as an overall score. The computer clusterings for each of the graphs show similar NCS behaviour where there is an increase around 504 due to cluster changes in the computer clustering. After this change, the NCS for Dynamic SNMTF improves for the computers. This change additionally results in an improvement in the CU-U plot and overall performance. For the port clustering, Dynamic SNMTF has superior performance. Additionally, Dynamic SNMTF has much more stable results than Simple NMTF.

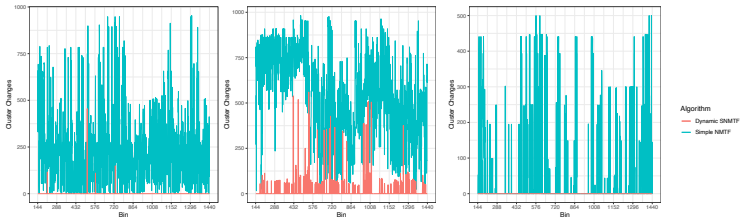
Figure E.4 shows the number of cluster changes for each entity type. It can be seen for the computer and port clusterings that Dynamic SNMTF does not have many changes whereas there are more changes for the users. Notably, Dynamic SNMTF has a large number of changes (approximately 500) at 504, which resulted in the improvement in clustering results for NCS in Figure E.3. This shows that utilising the previous cluster to initialise Simple NMTF does not cause the algorithm to be static, and still allows the cluster assignment to change.



(a) CC-C **(b)** CU-C **(c)** CU-U

(d) CP-C **(e)** CP-P **(f)** Overall

Figure E.3: NCS for calculated clusterings for simulated network as described in Section 7.2.1 where there exist no cluster changes.

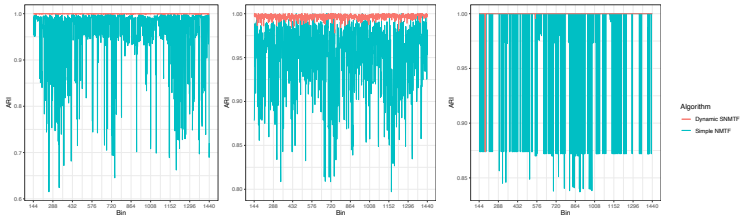


(a) Computer **(b)** User **(c)** Port

Figure E.4: Number of cluster changes for each node type for network stream as described in Section 7.2.1 where there exist no cluster changes.

E.2 Single Node Cluster Change

We now allow a single node for each node type to change cluster assignment once throughout the stream. This will help determine whether the algorithm is capable of adapting to cluster changes. Similar behaviour to the no change case is seen for the similarity between consecutive clusterings for this new simulation case in Figure E.5. The similarity is again very high for Dynamic SNMTF suggesting there are very few changes which is expected as only a single node has a cluster

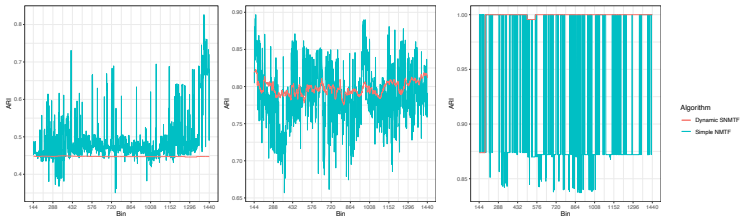


(a) Computer **(b)** User **(c)** Port

Figure E.5: ARI of consecutive clusterings over network stream as described in Section 7.2.1 where for a single node from each node type there is a cluster change at 1261, 847 and 502 for the computer, user and port node respectively.

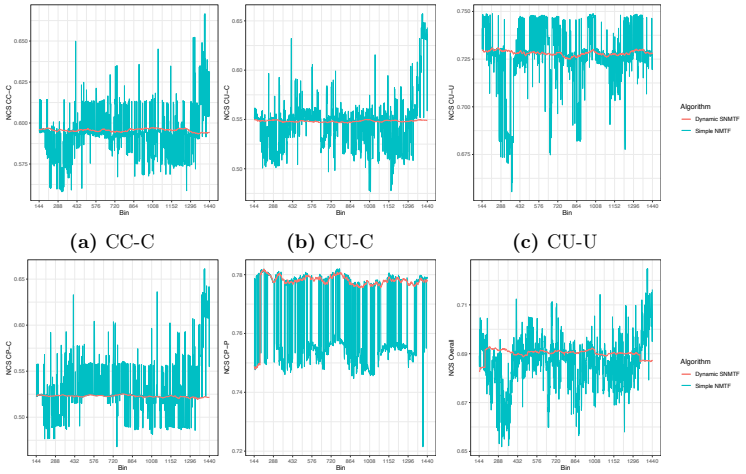
assignment change. Again Simple NMTF has more variability in comparison.

Again for the similarity to the true clustering in Figure E.6, the port clustering for Dynamic SNMTF is very high where the computer and user ports have stable good performance in comparison to Simple NMTF. Similar to NCS in Figure E.7, Dynamic SNMTF has a value in the middle of the extremes of Simple NMTF for the computer and user clusterings and very high NCS for the port clustering. Finally, when looking at the number of cluster changes in Figure E.8, there exist more large scale changes for the computers, users and ports in comparison to the no change example. This may be an artefact of the simulation or due to the inclusion of cluster changes in the data. However as the performance is stable, these cluster changes may correspond to merging or splittings of clusters which may not dramatically affect the performance. As the cluster performance of Dynamic SNMTF does not decrease due to the cluster changes present, this suggests the procedure is robust to cluster changes.



(a) Computer **(b)** User **(c)** Port

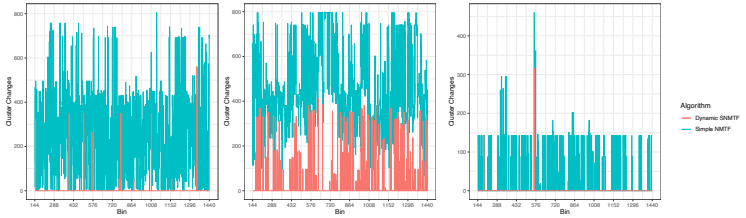
Figure E.6: ARI to true clustering over network stream as described in Section 7.2.1 where for a single node from each node type there is a cluster change at 1261, 847 and 502 for the computer, user and port node respectively.



(a) CC-C **(b)** CU-C **(c)** CU-U

(d) CP-C **(e)** CP-P **(f)** Overall

Figure E.7: NCS for calculated clusterings for simulated network as described in Section 7.2.1 where for a single node from each node type there is a cluster change at 1261, 847 and 502 for the computer, user and port node respectively.



(a) Computer **(b)** User **(c)** Port

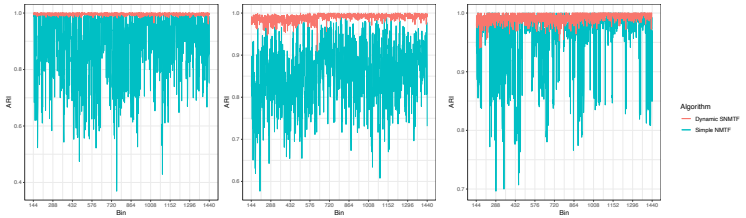
Figure E.8: Number of cluster changes for each node type for network stream as described in Section 7.2.1 where for a single node from each node type there is a cluster change at 1261, 847 and 502 for the computer, user and port node respectively.

E.3 Anomalous Node

In addition to allowing many nodes to change clusters where we set $\lambda_i = 3\ \forall i$, we include an anomalous node for each node type which is in its own cluster and has random connections throughout the network. This is considered anomalous as it

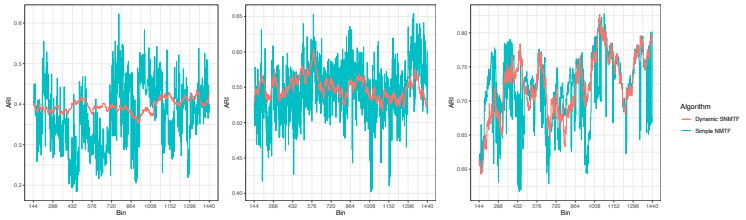
does not have a regular pattern of connection, unlike the other nodes. Additionally, at each time point, these anomalous nodes have large numbers of edges where the probability of a connection to any other node in the graph is 0.05.

Once again the similarity between clusterings for Dynamic SNMTF is still very high in comparison to Simple NMTF and is seen in Figure E.9. This is however lower than that seen in Section 7.2.3 due to more cluster changes in the stream. For the similarity to the true clustering in Figure E.10, there is more similarity between the performance of Dynamic NMTF and Simple NMTF. As more cluster changes are introduced into the graph, it is reasonable that the restart method may become more favourable as there is less dependence on the previous clustering. However, the performance of Dynamic NMTF is more stable and is not worse than Simple NMTF.



(a) Computer **(b)** User **(c)** Port

Figure E.9: ARI of consecutive clusterings over network stream as described in Section 7.2.1 where $\lambda = 3$ in the cluster membership generation process and an anomalous node for each node type is included.

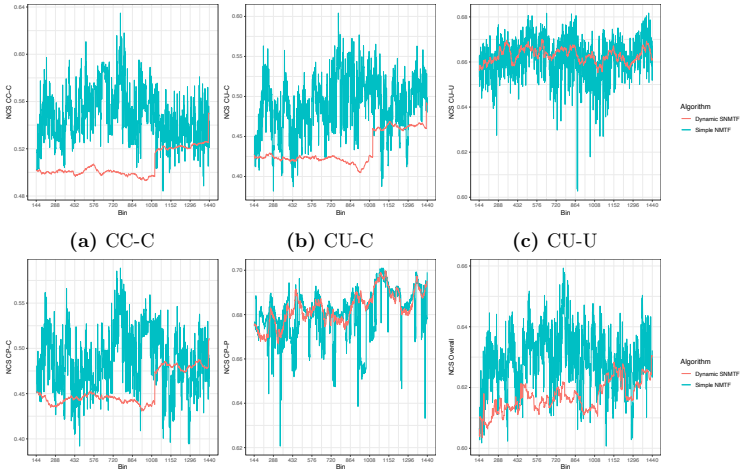


(a) Computer **(b)** User **(c)** Port

Figure E.10: ARI to the true clustering over network stream as described in Section 7.2.1 where $\lambda = 3$ in the cluster membership generation process and an anomalous node for each node type is included.

It is clear from Figure E.11 that the computer NCS is poorer for Dynamic SNMTF than Simple NMTF whilst being very similar for users and ports in Figures

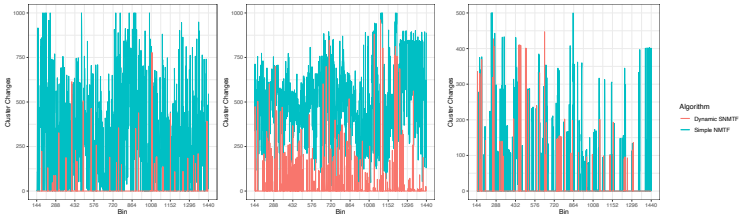
E.11 (c) and (e) respectively. However, the difference in performance is marginal. Once again the number of cluster changes in Figure E.12 is larger than before but for the computers and ports, there are only discrete spikes of change rather than continuous change as seen by Simple NMTF. As Dynamic SNMTF still has a similar performance to Simple NMTF but is much more stable with smooth clusterings, it is favourable. It is important to note that the simulated case in this section is more extreme than that observed in practice where nodes rarely change clusterings or do so less often than simulated here.



(a) CC-C **(b)** CU-C **(c)** CU-U

(d) CP-C **(e)** CP-P **(f)** Overall

Figure E.11: NCS for calculated clusterings for simulated network as described in Section 7.2.1 where $\lambda = 3$ in the cluster membership generation process and an anomalous node for each node type is included.



(a) Computer **(b)** User **(c)** Port

Figure E.12: Number of cluster changes for each node type for network stream as described in Section 7.2.1 where $\lambda = 3$ in the cluster membership generation process and an anomalous node for each node type is included.

Appendix F

Cyber Application Appendix

Table F.1: Descriptions of Authentication Event IDs in LANL WLS data.

Event ID	Name	Description
4768	Kerberos authentication ticket requested (TGT)	Logged on domain controllers which logs both success and failure instances. Event occurs for initial logon.
4769	Kerberos service ticket requested (TGS)	Service tickets obtained whenever user or computer accesses a server on the network. Logged on domain controllers.
4770	Kerberos service ticket renewed	Kerberos limits how long ticket valid for. If the ticket expires when the user is still logged on, windows automatically contacts the domain controller to renew the ticket.
4774	An account was mapped for logon	Logged on the domain controller.
4776	Domain controller (DC) attempted to validate credentials	When domain controller successfully authenticates a user via other authentication packages (instead of Kerberos), the DC logs this event.
4624	Account has successful log on	Has specific logon type. If the entry has a source field, the local event will be the same as LogHost.
4625	Account failed to Logon	Has a specific logon type and sometimes has a source.
4634	Account was Logged off	Logoff resulting from idle network session with specific logon type.
4647	User initiated logoff	Logoff where user physically logs off from console.
4648	Logon attempted using explicit credentials	User connects to a server or runs a program locally using alternative credentials. Also when a process logs on as a different account such as when the scheduled task service starts a task as the specified user. Logging on interactively to a member server with a domain account produces this event with 2 instances of 4624.
4672	Special privileges assigned to a new logon	When an account assigned any "administrator equivalent" user rights logs on. Will see these events close to logon events (4624) for administrators. This also logged for any server or applications accounts logging on as a batch job or system service.
4800	Workstation locked	
4801	Workstation unlocked	
4802	Screensaver invoked	
4803	Screensaver dismissed	

Table F.2: Descriptions of Process Event IDs in LANL WLS data.

Event ID	Name	Description
4688	Process Start	Documents each program that is executed and the process that started this process.
4689	Process End	When start program creating a process that stays open until program exits.

Table F.3: Descriptions of System Event IDs in LANL WLS data.

Event ID	Name
4608	Windows starting up
4609	Windows shutting down
1100	Event logging services has shut down

Table F.4: Descriptions of Logon Types in LANL WLS data.

Logon Type	Description
0	Used only by the system account
2	Interactive (logon at keyboard and screen of system)
3	Network (connection to shared folder on this computer from elsewhere in the network)
4	Batch (scheduled task)
5	Service (service startup)
7	Unlock(unattended workstation with password protected screen saver)
8	NetworkCleartext (logon with credentials sent in clear text. Indicates logon to Internet Information Services with basic authentication.)
9	NewCredentials such as with RunAs or mapped network drive with alternative credentials.
10	RemoteInterative (terminal services, remote desktop or remote assistance)
11	cachedinteractive (logon with cached domain credentials such as when logging onto a laptop when away from the network)