New Edge Activity and Anomaly Detection in a Large Computer Network

Submitted in part fulfilment of the
requirements for the degree of
Doctor of Philosophy of Imperial College London
and the
Diploma of Imperial College London
by

Silvia Metelli

Department of Mathematics,
Imperial College London

August 31, 2018

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.

Silvia Metelli,
March 2018

Copyright

Alla mia famiglia

Acknowledgements

First, I would really like to thank my supervisor, Nick Heard, for all the support he has given me throughout my time at Imperial: I feel very fortunate. With his patience and encouragement, Nick has guided me through this project since the start and he has always been available whether for meetings or last-minute clarifications with illegible, but fundamental, handwritten notes.

Then, thank you to all of the people with whom I have shared office 6M09 and to the people I have met during my visiting experience at Los Alamos National Laboratory. In both cases, I have been exposed to a very stimulating research environment.

Finally, I would like to thank my closest friends, who have always been there for me in good and difficult times, no matter the distance. Above all, thanks to my dad, my mum and my brother for all their love.

Silvia Metelli

Thesis Advisor: Dr. Nicholas Heard

Abstract

Computer networks are complex systems, and dynamically monitoring their structure in search for anomalies is both a challenging and important task for cyber security. In a computer network, new edges are connections from a host or client to a computer or server that has not been connected to before and can provide strong statistical evidence for detecting anomalies. However, performing meaningful anomaly detection on the arrivals of new edges is non-trivial as new edges can be indicative of both legitimate and illegitimate activity and occur with a considerable heterogeneity between network hosts.

This thesis presents a framework aimed at modelling *normal* new edge activity and performing anomaly detection in a large computer network graph. Specifically, the main contribution consists of a Bayesian method for modelling the intensity of new edges, simultaneously addressing the rate of occurrence of new edges and any underlying latent structural relationship between the clients and servers in the network. What constitutes normal behaviour for some hosts might be very unusual for some others and so examining existing network structure is key for accurately predicting likely future interactions. For this purpose, a notion of similarity between clients and servers is developed, first under hard-thresholding with a clustering model, and then extended to soft-thresholding in a flexible latent feature space. Finally, the model is used to construct an anomaly detection method, which successfully identifies some known compromised machines when demonstrated on real computer network data.

Contents

Abstract **5**

1 Introduction **21**
1.1 Anomaly detection for cyber-security 23
1.1.1 Cyber-attacks 24
1.1.2 Anomaly detection approach 25
1.2 Outline of this thesis 27

2 Computer Network Data **31**
2.1 Imperial College NetFlow data 32
2.2 Los Alamos National Laboratory data 33
2.2.1 LANL 2014 authentication data 33
2.2.2 LANL 2015 authentication data 35
2.3 Computer network data as marked point processes 39

3 Modelling the Rate of Occurrence of New Edges **43**
3.1 Bayesian variable selection 44
3.1.1 Bayesian inference 47
3.2 Modelling the rate of occurrence of new edges through variable selection 49
3.2.1 Logistic regression model for new edges 50
3.2.2 Bayesian model averaging 51
3.3 Detecting automated polling traffic 53
3.4 An application to Imperial College London NetFlow data 55
3.4.1 Network flow data 55
3.4.2 Variable selection results 55

4 Biclustering Methods for Computer Network Data **63**

4.1 Hierarchical agglomerative clustering methods 64
4.1.1 Bayesian model-based clustering 65
4.1.2 Clustering model for computer network data 66
4.1.3 Clustering algorithm 68
4.1.4 Bayesian model-based biclustering 68
4.1.5 Biclustering model for computer network data 70
4.1.6 Biclustering algorithm 72
4.1.7 An informative Beta prior 72
4.2 Flat clustering methods 74
4.2.1 Biclustering via singular value decomposition 75
4.2.2 Spectral biclustering 78
4.3 Simulation study 79
4.3.1 Simulated data 79
4.3.2 Results 80

5 A client model for the identity of new edges 87
5.1 Bayesian proportional hazards client model for new edges 88
5.1.1 Conditional Bayesian likelihood-based inference 90
5.2 Sequential two-step inference 92
5.2.1 Agglomerative clustering step 92
5.2.2 MCMC step 93
5.3 An application to LANL computer network authentication data . . 94

6 An entire-network model for the intensity of arrival of new edges 101
6.1 A Bayesian Cox model for new edges for the entire network 102
6.1.1 Conditional likelihood-based Bayesian inference 104
6.2 Cluster formulation 105
6.2.1 Surrogate spectral biclustering model 106
6.3 Latent feature formulation 107
6.3.1 The finite latent feature case 108
6.3.2 The infinite latent feature case 110
6.4 Posterior inference 118
6.4.1 Cluster formulation inference 118
6.4.2 Latent feature formulation inference 118
6.5 An application to LANL computer network data 122

7 Monitoring and Anomaly Detection 131
7.1 Monitoring 132
7.1.1 Predictive p-values 132

7.1.2 Results . . . 133
7.2 Anomaly Detection . . . 135
7.2.1 Combining p-values for ranking clients . . . 135
7.2.2 Results . . . 136

8 Conclusion and Future Work **139**
8.1 Conclusion . . . 139
8.2 Future Work . . . 141

Bibliography **145**

Appendix A Calculation of the marginal posterior distribution (4.6) **157**

Appendix B IBP as limit of a Beta-Bernoulli model **159**
B.1 A finite feature model . . . 159
B.2 Taking the infinite limit . . . 160

Appendix C Additional SVD results **163**

List of Figures

1.1 Toy motivating example of the evolution of a client-server bipartite network graph with new edges in green. 23

1.2 Lifecycle of a cyber-attack. 25

1.3 Example of traversal attack involving multiple machines. Filled circles represent compromised hosts (Neil et al., 2013). 26

2.1 The distribution of daily arrival times of Imperial College NetFlow events, estimated to five minute bins using the data obtained over 97 days, shown on [0,24] hours (left) to demonstrate the sinusoidal nature, and then as a circular distribution (right). 33

2.2 Log-log plot of out-degree distribution for users (left panel) and in-degree distribution (right panel) for computers for LANL network data gathered in 2014, respectively. 34

2.3 LANL bipartite authentication graph for the first minute of traffic. Clients are represented as blue nodes whilst servers are in green. . . 36

2.4 Log-log plot of out-degree distribution for clients (left panel) and in-degree distribution (right panel) for servers for LANL network data gathered in 2015, respectively. 36

2.5 Graphs of compromised activity just in the red team data, divided by week of traffic. 38

2.6 Histogram for the number of new edges per each day of traffic in the bulk data (blue bar) and just for new edges from the red team data (red bar). 38

3.1 The distribution of unfiltered client event times for IP $\tilde{x}$, estimated to five minute bins using the data obtained over 97 days. 58

3.2 The number of client event observed in the network flow data for a particular IP address on the network, IP z, which towards the end of the collection period was eventually found to be compromised. . . 59

4.1 A cartoon describing how to transform the adjacency matrix of a computer network graph into lower dimensional representations for clients and servers. 78
4.2 Probability density function of beta distributions with parameters $\alpha = 10$, $\beta = 1$ for the high degree servers and $\alpha = 1$, $\beta = 10$ for the low degree servers. 80
4.3 Simulated data matrix heatmaps, for three different matrix sizes, where the grayscale corresponds to different cluster parameters. . . 81
4.4 Algorithm runtimes (seconds) vs. data matrix dimension. 84

5.1 Frequency distribution of computer degrees (log-log scale). 95
5.2 Posterior distribution of the number of identified client clusters. . . 96
5.4 Log-likelihood vs. number of MCMC iterations, for the saturated model. 98
5.3 Adjacency data matrix heat map with cluster configuration identified. The greyscale represents different cluster allocation. 99

6.1 Example of the decomposition of U. A binary matrix Δ_U (first panel) indicates which features are active. Elementwise multiplication of Δ_U by $\tilde{U}$ of continuous values produces the representation in the second panel. 112
6.2 An illustration of the Indian Buffet Process for Δ_U. Note that $N = |X|$. (a) The first client samples Poisson(θ) features, which is recorded by changing the corresponding entries of Δ_U to one. (b) and (c) For the x^{th} client, the first step is activating the previously sampled features with probability proportional to the number of clients who already have these features active. The next step is to activate a Poisson(θ/x) number of new features. 114
6.3 Posterior estimates coefficients under the cluster formulation, with credible intervals. 124
6.4 Clustered graph induced by applying spectral biclustering, reordered by row clusters. 125
6.5 Posterior estimates with credible interval for the cluster formulation, where only client clusters were inferred. 126

6.6 Three sets of posterior estimates coefficients under the latent feature formulation, with credible intervals, obtained from full MCMC (●), sparse truncated SVD with stability selection (■), and standard truncated SVD (▲). 128
6.7 Number of identified row and column clusters during the MCMC run. 129
6.8 Scree plot for decay of singular values from the SVD of $\hat{\Lambda}$, in the interval $[0, 30]$, with the characteristic 'elbow' corresponding to the largest difference in magnitude. 130
6.9 Number of active latent features during the MCMC run. 130

7.1 Empirical cumulative distribution of observed p-values under the cluster model (left) and the latent feature model (right), against the Uniform$(0, 1)$ cumulative distribution function. 134
7.2 Log-likelihood vs. number of MCMC iterations, for the cluster formulation (top), and for the latent formulation (bottom). 135
7.3 ROC curves for each client, for each sample repetition, shown on both linear (left) and log scales (right). 137
7.4 Observed p-values (×) over time and the corresponding control chart (——) for the two identified infected clients and two of uninfected clients in the red bulk data. Top left: C17693; Top right: C19932; Bottom left: C349; Bottom right: C586. Control chart thresholds at the 1% (- - -) and 0.1% (- - -) significance levels are shown for each client. 138

8.1 A graph showing connections from user domains to server computers involving red team activity. 142

C.1 Scree plot for truncated-SVD operated directly on the adjacency matrix A. 163
C.2 Scree plots for truncated-SVD operated on $\hat{\Lambda}$, from Sample 2 to Sample 15. 164

List of Tables

2.1 Network flow data fields. 32
2.2 2014 LANL authentication data fields. 34
2.3 2015 LANL authentication data fields. 35
2.4 Numbers of events and unique server computers connected to by four client computers in the LANL authentication data identified as compromised both in the complete data and just in the red team. 37

3.1 Dummy variables included in the analysis. 51
3.2 Coefficient estimates from logistic regression for IP $\tilde{x}$. 56
3.3 Coefficient estimates from logistic regression for IP $\tilde{x}$ with polling data removed. 57
3.4 Coefficient estimates from logistic regression for IP z before infection. 60
3.5 Coefficient estimates from logistic regression for IP z after infection. 61
3.6 Bayes factors for the best three models against the null model, for the four different subsets of data analysed. 62

4.1 Quality measures for each algorithm over simulated data, for different data matrix sizes. 85

5.1 Summary statistics for the subset of data analysed. 95
5.2 Posterior model coefficient estimates with credible intervals. 96
5.3 Likelihood ratios for the different models. 98

6.1 Likelihood ratios for the three different model settings. 129

7.1 Out-of-training log likelihood under both formulations. All values are averages over the test pairs. 134

List of Algorithms

1 Standard Metropolis-Hastings Algorithm 49
2 Model-based Clustering 69
3 Model-based Iterative Biclustering 73
4 SSVD with stability selection 120

Chapter 1

Introduction

The deployment of statistical methodology for cyber-security research is a relatively new paradigm which has been attracting attention over the last decade due to the increasing cyber threat faced by government, industry, academia, the military and society as a whole. This has led to the routine bulk collection of high-volumes of traffic data, which has in turn fostered the development of novel statistical anomaly detection methods to quickly analyse those data, possibly in real time. However, computer network traffic data are complex and several challenges still hinder the employment of anomaly-based methods for large computer networks, such as computational speed and scalability. Statistical anomaly detection searches for outlying behaviour in a network with respect to a putative normal background. This approach aims at building a realistic probability model for the normal evolution of the network system and then look for any abnormal deviation (Neil et al., 2013; Turcotte et al., 2014), potentially providing a robust next layer of cyber-security defence. These tools are indeed intended to be complementary to existing signature-based enterprise network defence systems, which can only monitor for known security violations.

In statistical anomaly detection, it is therefore paramount to have reliable and robust models for learning the *normal* network background. Unfortunately, in cyber security applications, it is often very complex to determine appropriately

such a model. In addition, there is a considerable imbalance between the amount of available data for learning normal behaviour and data recording intrusions, which are very rare. As a result, the most of the detection power will come from the ability of building sophisticated models of normal behaviour, rather than from modelling intrusion data. In line with these considerations, the present thesis is mostly concerned with modelling normal network behaviour. Specifically, the work focuses on a specific aspect occurring in the evolution of a computer network system which has not been previously fully analysed, namely the formation of new edges over time. Arrivals of new edges in a computer network represent connections between a client and server pair not previously observed, and might suggest the presence of intruders or malicious implants.

The information about a computer network can be collected as an online stream of connection events, which will be summarised throughout this thesis as a series of time-varying directed graphs, denoted $\{G_t\}$. An intruder infecting a node at time t in the network does not typically have information about which nodes in G_t that node usually connects to. Therefore the intruder, despite not wishing to stand out in the network traffic, may be more likely to initiate new connections between hosts which have never communicated before. Such activity can provide a valuable signal for detecting the presence of the intruder. However, accurately capturing and accumulating evidence of new edge formation is challenging because new connections occur at a relatively high frequency for legitimate reasons, and with very different characteristics between network hosts. In particular, some hosts – for instance a new server – will have a high activity rate both of indegree and outdegree connections as soon as they come online, while some others – for example new single-user machines – will probably tend to have a lower activity rate at which they make new edges. Figure 1.1 provides a simple illustration of the evolution of a bipartite computer network graph with the purpose of illustrating the mechanism of new edge formation. Despite the system being too small to be representative of large-scale systems, the idea here is that Client 3 might be responsible of anomalous behaviour, as it makes a high number of new edges in a small window of time.

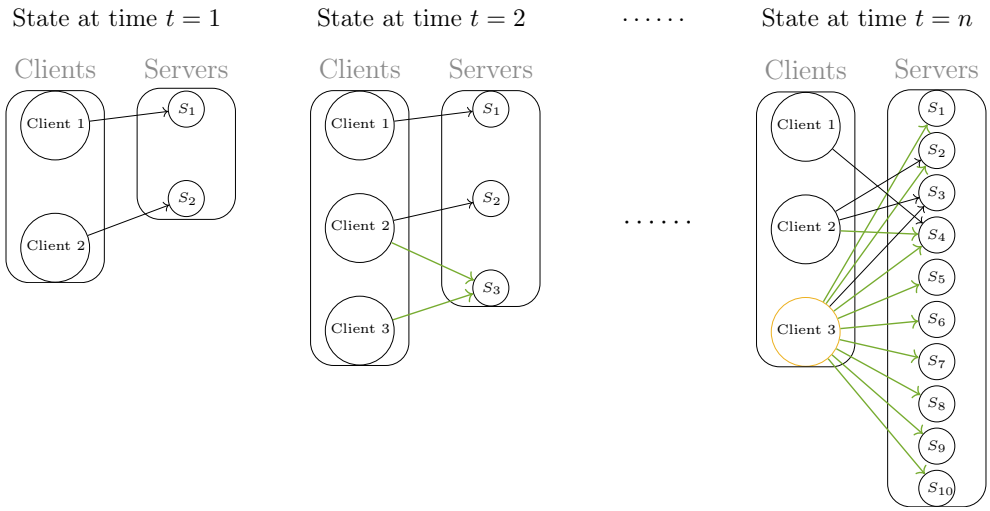


Figure 1.1: Toy motivating example of the evolution of a client-server bipartite network graph with new edges in green.

1.1 Anomaly detection for cyber-security

Anomalous computer network behaviour can be caused by different types of cyber-attacks, and examples include denial of service (DoS) attacks to cause disruption or further spread malware, or intruders seeking to escalate privileges by traversing through the host network. Although all modern networks are provided with security systems which try to prevent and monitor unauthorised access, these systems still remain permeable and most networks are arguably already compromised to some degree. This implies that some intruders will at some point succeed in gaining access through the network, capturing valuable data or consolidating their presence in the network for future malicious operations. In an attempt to better contextualise the general motivation for the application presented in this thesis, in the next subsection we give a brief description of the different stages of a modern, typically multi-stage, cyber-attack.

1.1.1 Cyber-attacks

Sophisticated attackers, typified by advanced persistent threat (Friedberg et al., 2015), are now able to easily circumvent perimeter based defences and to establish a persistent presence in the system. Furthermore, most of modern cyber-attacks are multi-stage by their nature and so detecting malicious activity, especially at early stages, is challenging. The typical stages involved in a breach can be described as a lifecycle, as illustrated in Figure 1.2. The first stage (Reconnaissance stage) consists in identifying potential targets and determining the attack methodology. In the second stage (Penetration stage), the attacker successfully executes malicious code on one or more systems, bypassing perimeter defences and gaining access to the internal network. Subsequently, the attacker gains a foothold on the network and establishes a persistent control, in order to have long-term, remote access over the victim's computer (Gaining foothold stage). The attempts to gain wider access to the network are then generally carried out via a "pass-the-hash" technique or by exploiting a vulnerable piece of software (Escalate Privileges stage). The next step is then to conduct internal exploration of the infected environment (Internal Reconnaissance stage) and subsequently to laterally compromise additional systems, using the access gained from the previous steps. By installing multiple remote entry points, the attacker ensures prolonged presence and control within the environment from outside (Maintain Presence stage). Finally, the attacker executes exfiltration, corruption, and disruption of sensitive data, reaching the end point of cyber exploitation life-cycle (Mission Complete stage).

As the final stage involves data exfiltration and system disruption, it is this stage the one responsible for the greatest damage. Thus, the ability to detect the attack before this final stage should be paramount to all modern anomaly detection techniques and is key to prevent large-scale impact.

Possible attack techniques which might be relevant to the analysis of new edge formation primarily include early activities directed to download some malicious payload, such as ransomware, and exploitation of a host computer as part of a botnet, e.g. taking part in a DDoS attack.

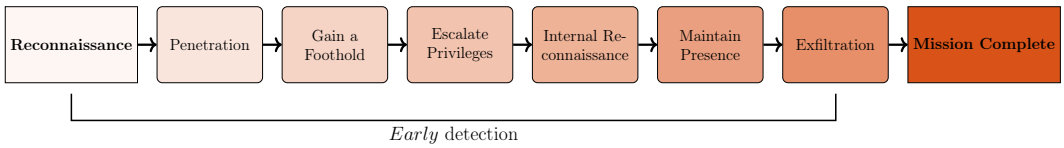


Figure 1.2: Lifecycle of a cyber-attack.

1.1.2 Anomaly detection approach

The two main approaches to intrusion detection on a computer network are signature-based methods and anomaly-based detection methods. Signature-based systems search for evidence of attacks based on signatures accumulated from previous attack vectors and stored on a so-called signature database (Cahill et al., 2002). Signature-based methods are often strong, resulting in very low levels of false positive detections. However, this approach requires the signature to be known beforehand, and is therefore less well-suited to detecting new anomalies. In contrast, anomaly detection methods, as already mentioned, aim at building a model reflecting the normal behaviour of the system and thus no prior knowledge about the characteristics of potential future attacks is required, allowing us to discover day-*zero* attacks, i.e. attacks which were never observed before (Patcha and Park, 2007).

A general overview of anomaly-based methods is provided by Chandola et al. (2009), which outlines a large number of approaches developed in literature and the challenges associated with their deployment. Heuristic measures, such as the minimum description length heuristic (Rissanen, 1989), and spectral graph techniques have been first attempts in detecting anomalous subsets of network data (Noble and Cook, 2003; Idé and Kashima, 2004). A considerable amount of research has then focused on scan statistics, which was established for online network intrusion detection for the first time by Priebe et al. (2005). The authors use a star shape to define a local area and then scan the data over those local windows, calculating a locality statistic for some graph invariant in each window. The locality statistic is calculated by testing the null hypothesis of normal behaviour in each local window and the scan statistic is defined as the maximum of the locality statistic across

all local areas. Park et al. (2013) then extended the theory of scan statistics on graphs by considering a fusion of graph invariants. Other notable work in this direction include Horn and Willett (2011); Sharpnack et al. (2012); Valko (2011). Subsequently, Neil et al. (2013) take a scan statistic approach to perform anomaly detection in small local areas of the graph, by looking at anomalous edges which form a path, i.e. a sequence of edges in which the destination node of the current edge is the source node for the next edge. By means of these paths, they seek to identify an intruder who is traversing the network. Figure 1.3 shows an example of traversal attack.

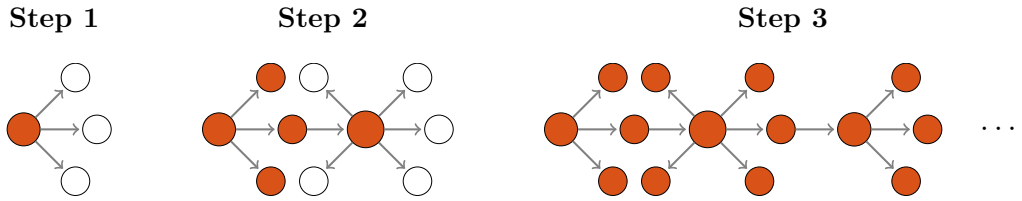


Figure 1.3: Example of traversal attack involving multiple machines. Filled circles represent compromised hosts (Neil et al., 2013).

Anomaly detection has also been studied in the field of communication networks, and relevant work include Heard et al. (2010) and Heard and Turcotte (2014), where discrete time models are used to model communications along edges in the network. A two-stage approach is proposed by the authors to first identify potentially anomalous nodes and subsequently build an anomalous subgraph induced by these nodes. The subgraph represents a much reduced portion of the initial network, thus allowing standard graph-theoretic tools to be deployed.

Despite the recent interest and efforts, statistical anomaly detection approach to cyber-security still poses significant statistical challenges, which are limiting the widespread deployment of these methods. One is modelling a complex data structure, with highly heterogeneous data. Then, a number of anomaly detection problems arise, for example it is difficult to incorporate model uncertainty while also combining anomalies through time and across the network without resulting in a high number of false positive detections. Finally, the high volume of traffic analysed raises significant computational challenges, making scalable and paral-

lelisable approaches of fundamental importance.

1.2 Outline of this thesis

The main contribution of this thesis is to propose a robust statistical model for the normal formation of new edges and subsequently to construct an anomaly detection method based on such a model. Specifically, we propose a framework for modelling the arrivals of new edges, based on two main components. First, we focus on understanding the *rate* at which individual client hosts form new edges. Second and more challenging, we focus on predicting the *identity* of new edges formed by each client. What constitutes a normal connection for some hosts could be a very unusual connection for others and in the absence of further information about the structure of the network, the second consideration requires extracting underlying latent structural relationship between the clients and servers in the network. Towards this end, we develop a notion of similarity of network hosts, such that *similar* clients may connect to *similar* servers. Similarity is considered under hard-thresholding with a clustering model, or soft-thresholding in a latent feature space.

Chapter 3-6 address modelling normal network behaviour, first proposing distinct models for the rate of occurrence and the identity of the new edges; and then including both aspects in a single, more complete model. Chapter 7 is finally devoted to performing anomaly-detection, based on the model learned from the previous chapters. The structure of this thesis is as follows.

- **Chapter 2** gives a general introduction to computer network data and describes the three motivating data sets used to perform the analyses. The first data set consists of network flow data gathered from the Imperial College London internal domain, while the second and the third ones consist of authentication events from the Los Alamos National Laboratory enterprise network. It also presents the mathematical formulation that will be used in all the following chapters to describe various aspects of such data.

- **Chapter 3** introduces some background material that will be needed throughout this thesis and then presents a logistic model for the rate of occurrence of new edges. Bayesian variable selection is deployed for identifying the most influential covariates to be included in the model, thus avoiding model redundancy and overfitting. To improve the data quality, a technique for removing automated polling traffic is also presented. The method is then demonstrated on Imperial College London network flow data, and results for filtered and unfiltered data are compared. Parts of this chapter can also be found in Metelli and Heard (2014).

- **Chapter 4** presents several methods for clustering and biclustering hosts in a large computer network. First, hierarchical clustering approaches such as Bayesian agglomerative model-based clustering and an extension to agglomerative model-based biclustering are presented. Second, two flat clustering approaches, namely singular value decomposition and spectral clustering, are introduced. A simulation study is then performed to compare the effectiveness of each method presented. Some of the clustering methods presented in this chapter will prove useful in Chapter 5 and Chapter 6.

- **Chapter 5** proposes a Bayesian client model aimed at predicting new edges in a large computer network while simultaneously identifying clustered structure. As a first step of the analysis, initial reliable cluster configurations are sequentially inferred through model-based agglomerative clustering and then jointly updated with model parameters via a Markov Chain Monte Carlo (MCMC) sampler. The method is then applied to one of the two authentication data sets gathered from Los Alamos National Laboratory internal network. Parts of this chapter can also be found in Metelli and Heard (2016).

- **Chapter 6** proposes a framework for modelling the intensity of arrival of new edges, which addresses both the rates at which the client forms new edges and any underlying latent structural relationship between the clients and servers in the network. Two formulations will be proposed for inferring latent structure, first with a clustering model under hard-thresholding and then under soft-thresholding in a flexible latent feature space. Posterior

inference under both frameworks is also described. Parts of this chapter are taken from Metelli and Heard (2018).

- **Chapter 7** finally shows how the model can be used to construct an anomaly detection method. This method, demonstrated on Los Alamos National Laboratory authentication data, was able to detect some known *red team* compromised machines.
- **Chapter 8** summarises the results presented in the thesis and discusses possible future work. Derivations for some of the equations used in the text and other supplementary materials are provided in the appendices.

Chapter 2

Computer Network Data

Computer networks are dynamic communication networks which allow computers to exchange data. Such data are mainly collected in the form of network flow events or as authentication events. Unfortunately, the number of released data sets from enterprise computer networks is still quite small, thus limiting the development of relevant research in the cyber security community. Many organisations are still reluctant to publicly release data, mainly due to security and privacy issues. In addition, in many cases the data collected lack accuracy or the volumes of collection is not sufficient for providing valuable cyber research.

The motivating data sets used in this thesis are obtained from two different available sources, namely Imperial College London and Los Alamos National Laboratory (LANL) internal networks. Imperial College London data are provided by the Information and Communication Technologies department in private form, whilst Los Alamos National Laboratory (LANL) has recently released data for public use (Kent, 2014, 2015*a*). Each data set is briefly described below. Specifically, Section 2.1 describes network flow data from Imperial College London internal domain, while Section 2.2 describes two different authentication data sets gathered from LANL internal network. Finally, Section 2.3 introduces the mathematical formulation used throughout the thesis to encode such data.

2.1 Imperial College NetFlow data

NetFlow data are aggregated summaries of the traffic passing around a computer network from one internet protocol (IP) address to another, over a period of time. A common format is Cisco's NetFlow protocol, which records information about IP addresses of the machines, both the source and the destination ports, the time of the connection and the number of packets and bytes transferred. As an example, the server port can provide rich information about the type of service being used: for example, ports 80 (HTTP) and 443 (HTTPS) are associated with web-browsing, and port 25 with email. A typical data set contains the following information:

Field Name	**Description**
Time	The start time of the event in epoch time format
Duration	The duration of the event (sec.)
SrcIP	The IP source address that likely initiated the event
DstIP	The IP destination address
Protocol	The protocol number
SrcPort	The port used by the *SrcIP*
DstPort	The port used by the *DstIP*
Packets	The number of packets of data exchanged
Bytes	The number of bytes of data exchanged

Table 2.1: Network flow data fields.

The Imperial College London network flow data are obtained from router level flow records gathered from one of the main networks. Imperial College London has a wide range of 345,098 IP addresses, in which approximately one seventh are generally active. The average level of traffic flow on the network equates to approximately 1.3 billion network flow records per day. Figure 2.1 shows the global daily pattern of the network, using data collected for 97 days between November 2013 and February 2014. As would be expected, the distribution of the time of day of the NetFlow events shows a sinusoidal diurnal scheme, where the majority of traffic happens in between 9am and 6pm.

This data set will be used to perform analyses in Chapter 3, where particular attention will be given to IP (Internet Protocol) source and destination addresses,

protocol, and source and destination port numbers for User Datagram Protocol (UDP) or Transmission Control Protocol (TCP). Both UDP and TCP are core members of the Internet protocol suite and their aim is to handle data communications between terminals in the Internet.

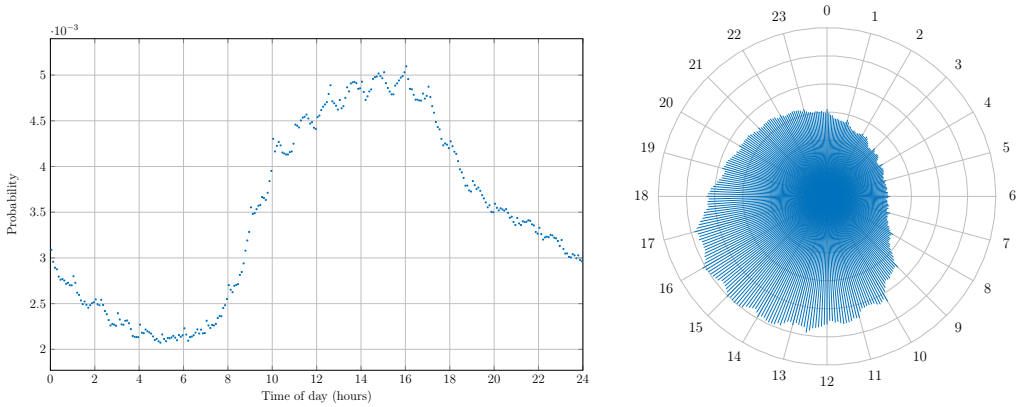


Figure 2.1: The distribution of daily arrival times of Imperial College NetFlow events, estimated to five minute bins using the data obtained over 97 days, shown on [0,24] hours (left) to demonstrate the sinusoidal nature, and then as a circular distribution (right).

2.2 Los Alamos National Laboratory data

Two data sets of authentication events from the Los Alamos National Laboratory (LANL) corporate, internal computer network are described below. The first data set encompasses authentication events collected in 2014 while the second one is part of comprehensive, multi-source event data gathered in 2015. The former will be used to demonstrate the method presented in Chapter 5, while the latter will be used in Chapter 6.

2.2.1 LANL 2014 authentication data

This data set consists of authentication data (Kent, 2014) from the enterprise computer network at LANL, representing 9 months of contiguous authentication

activity. More specifically, the data set contains 708,304,516 time-ordered, successful user to computer authentication events among 11,362 users and 22,284 computers in the network. The timing of each authentication pair of anonymised user and anonymised computer is recorded at one second resolution. Example records for the authentication data are as follows:

Field Name	**Description**
Time	Time of the authentication event (1 sec. resolution)
Usr	Anonymised user ID
Comp	Anonymised computer ID

Table 2.2: 2014 LANL authentication data fields.

Figure 2.2 shows the distributions of out-degrees for users and in-degrees for computers in the network. The outdegree for users is measured by the number of unique computers receiving authenticated connections, while the indegree for computers is measured by the number of unique users making authenticated connections. It can be noted that both the degree distributions approximately follow a power law.

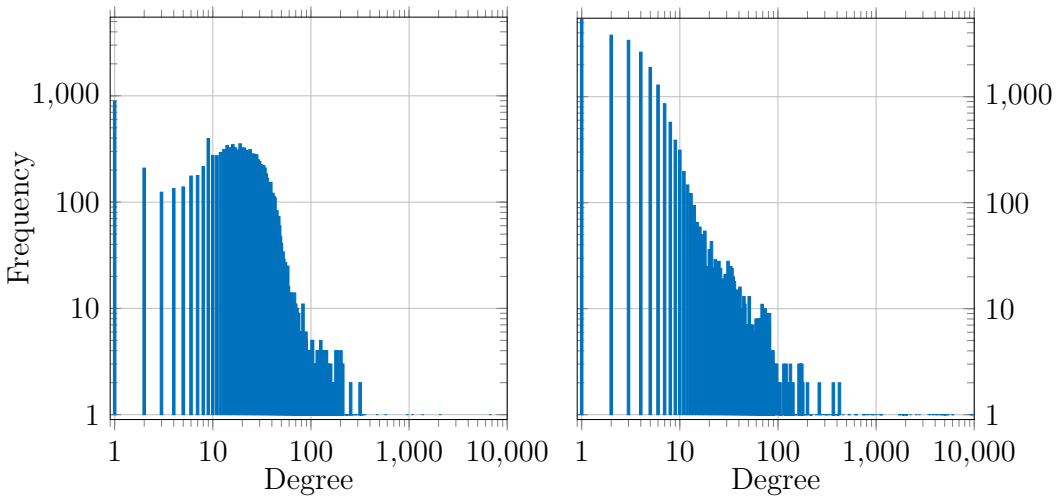


Figure 2.2: Log-log plot of out-degree distribution for users (left panel) and in-degree distribution (right panel) for computers for LANL network data gathered in 2014, respectively.

2.2.2 LANL 2015 authentication data

This more recent data set represents 58 consecutive days of de-identified authentication event data collected from the enterprise computer network at LANL (Kent, 2015*a*,*b*). The data are collected from individual Windows-based desktop computers, servers, and Active Directory servers. In total, the data set presents 336,806,387 time-ordered client to server authentication events among 16,230 clients and 15,417 servers. The timing of each authentication pair of anonymised client and anonymised server is again recorded at one second resolution. As an example, the different fields contained in this data set are reported in Table 2.3.

Field Name	**Description**
Time	Time of the authentication event (1 sec. resolution)
SrcUser@Domain	Anonymised user initiating the authentication event
DstUSer@Domain	Anonymised user that the authentication event is mapping to
SrcComputer	Anonymised user ID
DstComputer	Anonymised computer ID
AuthType	Type of authentication occurring (e.g. Negotiate, Kerberos etc.)
LogonType	Description of the type of logon (e.g. Batch, System service etc.)
Orientation	How the authentication event is being used
Status	Status of the authentication request (Success or Failure)

Table 2.3: 2015 LANL authentication data fields.

To illustrate, Figure 2.3 shows a graph of connections between clients and servers on the LANL data, observed over the first minute of traffic. It can be noted that there are a few high-degree clients responsible for the majority of the traffic. The distributions of out-degrees for clients and in-degrees for servers in the network in shown in Figure 2.2. As before, the outdegree for clients is measured by the number of unique server computers receiving authenticated connections, while the indegree for servers is measured by the number of unique clients making authenticated connections. Again, both degree distributions follow an approximate power law, although we can observe that the modal outdegree is higher than the modal indegree, since the majority of servers have a very small population of connecting clients.

This data set is particularly interesting as it contains a *red team* penetration

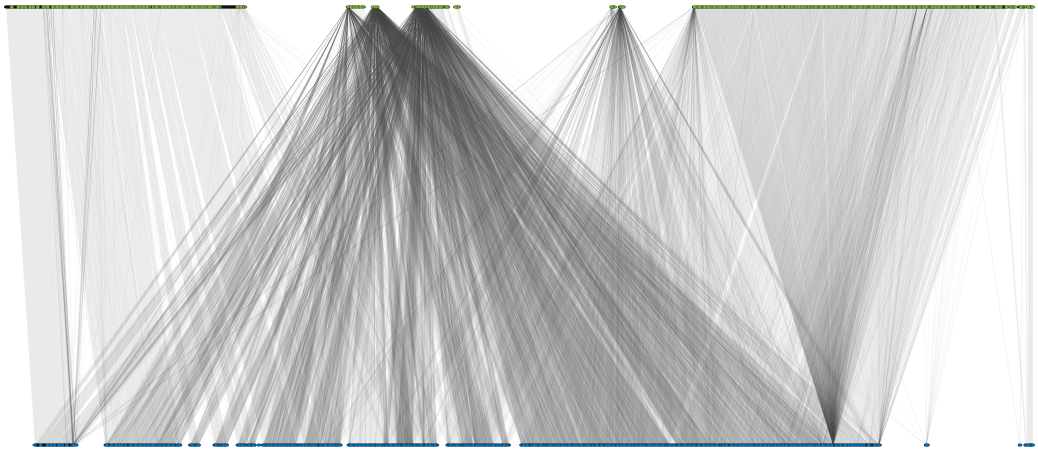


Figure 2.3: LANL bipartite authentication graph for the first minute of traffic. Clients are represented as blue nodes whilst servers are in green.

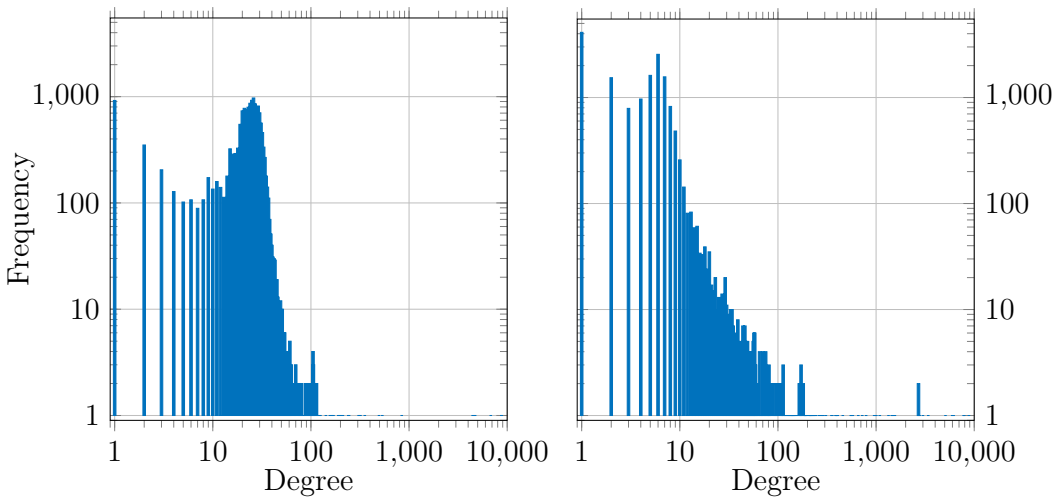


Figure 2.4: Log-log plot of out-degree distribution for clients (left panel) and in-degree distribution (right panel) for servers for LANL network data gathered in 2015, respectively.

testing operation, which occurred during the period of data collection. A subset of the data have been labelled as representing known compromise events, thus providing an ideal target for testing anomaly-detection methods.

The authentication event data labelled as red team events account for 48,079 of the total records in the bulk data, and Table 2.4 shows how these event data

Compromised client	Frequency		Unique server computers	
	Red Team	Total	Red Team	Total
C17693	701	1717	296	534
C18025	3	101	1	29
C19932	19	10,008	8	30
C22409	26	36,253	3	31

Table 2.4: Numbers of events and unique server computers connected to by four client computers in the LANL authentication data identified as compromised both in the complete data and just in the red team.

are distributed across those labelled machines, while Figure 2.5 shows four graphs of red team activity- one for each week of traffic- are reported.

As we will be restricting our interest to the formation of new edges over time, Figure 2.6 shows a bar plot of the rate of occurrence of new edges for each day of traffic, both for the bulk data and just for red team events. The total number of new edges in the bulk data is 419,744, representing 8,38% of the total traffic. Specifically, there are 134,688 new edges created during the first day, corresponding to approximately 32% of the total number of new edges in the data set. This is an expected behaviour, as at the beginning the clients tend to establish many new connections, which will later correspond to their regular traffic. The compromised clients in the red team data do not conform to this *normal* new edge behaviour: from the 29 days of traffic, the largest number of compromised new edges is formed during days 8 and 12.

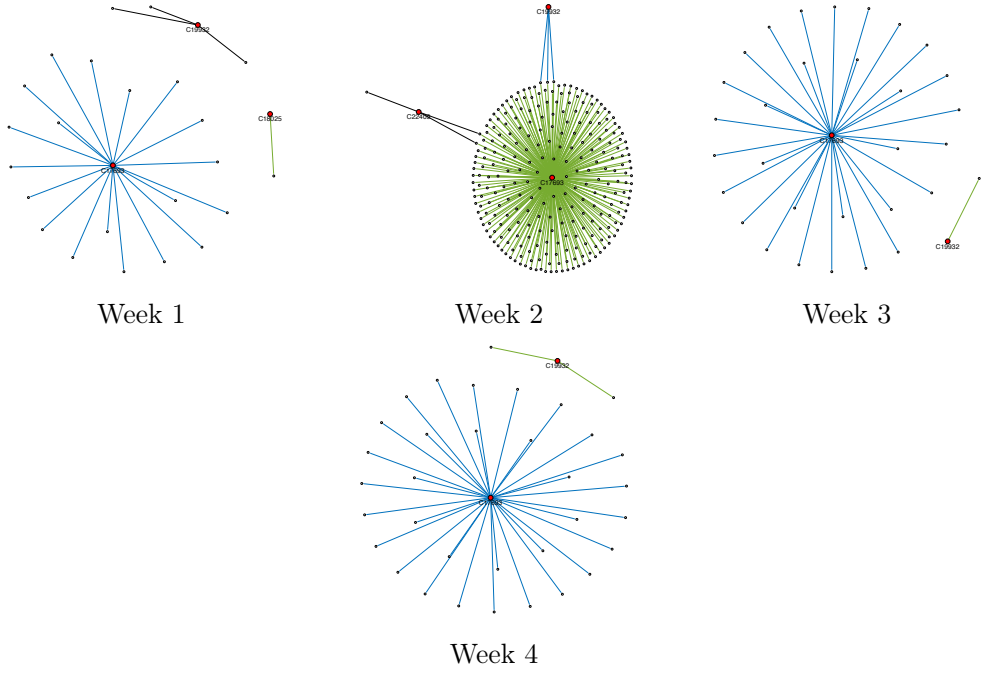


Figure 2.5: Graphs of compromised activity just in the red team data, divided by week of traffic.

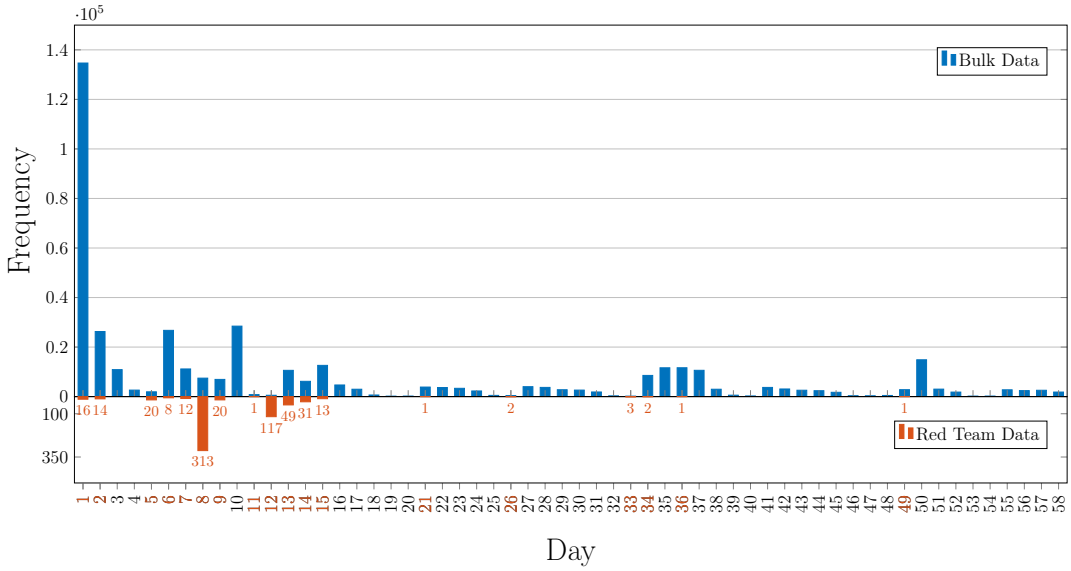


Figure 2.6: Histogram for the number of new edges per each day of traffic in the bulk data (blue bar) and just for new edges from the red team data (red bar).

2.3 Computer network data as marked point processes

In this section we provide a mathematical formulation for the aspects of computer network data which we will need throughout the thesis. In particular, we give a definition of a time-evolving computer network graph and we define the main variables that will be used for the rest of this work.

The information about the traffic observed on the computer network will be encoded as a directed graph of connections from a set of *clients* X to a set of *servers* Y, which can be naturally represented as a time-varying bipartite graph. Nodes in X and Y will represent active computers generating communication and edge activity will be seen as a point process of counts of connections over the time interval of interest. The sets X and Y may refer to the same collection of computers or IP addresses, but for the purposes of this thesis they will be considered as separate entities. Assuming fixed, but potentially very large sets of client nodes X and server nodes Y, $\{G_t\}$ will be a time-increasing set of directed edges in $X \times Y$, such that an edge (x, y) exists from $x \in X$ to $y \in Y$ in G_t if and only if client x has connected to server y by time t.

Specifically, we express the arrivals of connections between computers in the network as a marked point process $(\mathcal{T}, \mathcal{E}) = ((T_n)_{n\geq 1}, (E_n)_{n\geq 1})$ where each random variable T_n is a positive real-valued event time and E_n is a corresponding $(X \times Y)$-valued (client, server) mark. Let $0 \leq t_1 \leq t_2 \leq \ldots$ be the realised sequence of event times and let $e_n = (x_n, y_n) \in X \times Y$ be the corresponding mark for the nth event.

From $(\mathcal{T}, \mathcal{E})$ we can define a continuous-time, left-continuous stochastic series of random graphs $\{G_t | t \geq 0\}$ from the set of network graph edges observed. This can be expressed as

$$G_t = \{(x, y) | (x, y) \in X \times Y, N_{x,y}(t) > 0\}, \tag{2.1}$$

where

$$N_{x,y}(t) = \sum_{n \geq 1} \mathbb{1}_{[0,t)}(t_n)\mathbb{1}_{(x,y)}(e_n) \tag{2.2}$$

is the left-continuous counting process of connections from client x to server y prior to time t. Note that in this formulation the edges observed at time t are not included in G_t. The corresponding counting processes of connections from client x or to server y prior to time t can be achieved respectively summing (2.2) over X or Y,

$$N_{x,\cdot}(t) = \sum_{y \in Y} N_{x,y}(t), \;\; N_{\cdot,y}(t) = \sum_{x \in X} N_{x,y}(t). \tag{2.3}$$

For all the remaining chapters, we will be mainly interested in the following aspects of the dynamic graph G_t. First, we will be interested in capturing *bursts* of formation of new edges and second, we will be interested in incorporating the *popularity* of different client and server machines. The latter will be measured by the time-varying outdegrees of clients and indegrees of servers; while for the former we construct two variables which indicate whether the last connection made by each client x was new, or whether the last m connections were all new. Below we provide the mathematical definition of such variables.

For each client x, let $(\mathcal{T}^x, \mathcal{Y}^x) = ((T_n^x)_{n \geq 1}, (Y_n^x)_{n \geq 1})$ be the subprocess for which the client mark is x, corresponding to those indices n for which $\mathbb{1}_x(x_n) = 1$. From $(\mathcal{T}^x, \mathcal{Y}^x)$ we can first define a binary variable for the presence of a new edge

$$u_n^x = \mathbb{1}_{(X \times Y) \setminus G_{t_n^x}}\{(x, y_n^x)\} \tag{2.4}$$

such that $u_n^x = 1$ if and only if the nth connection from client x is new. Then, we can define the indicator variables for new edge 'burstiness' as

$$I_{x,1}(t) = \begin{cases} 1, & N_{x,\cdot}(t) = 0, \\ u^x_{N_{x,\cdot}(t)}, & N_{x,\cdot}(t) \geq 1; \end{cases} \tag{2.5}$$

and then recursively, for $m \geq 2$,

$$I_{x,m}(t) = \begin{cases} I_{x,m-1}(t), & N_{x,\cdot}(t) < m, \\ u^x_{N_{x,\cdot}(t)-1} I_{x,m}(t), & N_{x,\cdot}(t) \geq m. \end{cases} \tag{2.6}$$

The binary variable $I_{x,m}(t)$ will take value 1 if the last m connections made by client x were each new, and therefore represent a burst of new edge formation of length m.

Finally, from $(\mathcal{T}, \mathcal{E})$ we define the subprocess $(\mathcal{T}', \mathcal{E}') = ((T'_n)_{n\geq 1}, (E'_n)_{n\geq 1})$ of unique new edges observed in $(\mathcal{T}, \mathcal{E})$, corresponding to those indices n for which $\mathbb{1}_{G_{t_n}}\{(x_n, y_n)\} = 0$. From the realised sequence of unique edges $(e'_n)_{n\geq 1} = ((x'_n, y'_n))_{n\geq 1}$, the time-varying outdegrees of clients is simply given by

$$N^+_x(t) = \sum_{n\geq 1} \mathbb{1}_{[0,t)}(t'_n)\mathbb{1}_x(x'_n), \tag{2.7}$$

which measures the number of unique servers to which client x connected to before time t. Similarly, the time-varying indegrees of servers is

$$N^-_y(t) = \sum_{n\geq 1} \mathbb{1}_{[0,t)}(t'_n)\mathbb{1}_y(y'_n), \tag{2.8}$$

which measures the number of unique clients to which server y has received connections from, before time t. The formulation provided here refers to indegrees of clients and outdegrees of servers across the entire network graph; in Chapter 5 and 6 we will also consider the special case in which indegrees and outdegrees are measured just inside subsets (clusters) of similar clients or servers in the graph. In this case, the outdegrees of clients (2.7) and indegrees of servers (2.8) can be generalised by considering the degree of a node in the bipartite graph restricted to subsets of clients or servers. Respectively for a subset of clients $C \subseteq X$ and a

subset of servers $S \subseteq Y$, we can define

$$N^{+}_{x|S}(t) = \sum_{n \geq 1} \mathbb{1}_{[0,t)}(t'_n)\mathbb{1}_x(x'_n)\mathbb{1}_S(y'_n) \tag{2.9}$$

as the number of servers in S connected to by client x, prior to time t, and

$$N^{-}_{y|C}(t) = \sum_{n \geq 1} \mathbb{1}_{[0,t)}(t'_n)\mathbb{1}_y(y'_n)\mathbb{1}_C(x'_n) \tag{2.10}$$

as the number of clients in C which have connected to server y prior to time t.

Chapter 3

Modelling the Rate of Occurrence of New Edges

As discussed in the introduction, a compromised computer node tends to form a large number of new client edges in the network graph, connecting to server computers which have not previously been visited and so anomalous occurrence of new edges can signify malicious presence in the network. However, study of computer network flow data suggests new edges are also regularly formed by benign hosts, and often in bursts. Thus, when building a robust model for new edge formation, a first important component consists of understanding the rates at which new edges are formed by each host in the network. The formation of new edges is not a stationary process, and the rate at which new edges are formed should be viewed as time-varying: initially, many of the connections of the client will be new as its regular connections are first established, whilst in the longer term the rate of new edges will necessarily reduce, but will still be different or far from zero (see Figure 2.6).

In this chapter, Bayesian Model Averaging (BMA) (Raftery et al., 1997) is applied to a logistic regression model for new edge formation, with the purpose of performing variable selection. Network traffic data are complex, and so the potential number of variables which might be included in such a statistical model

can be large. Without proper treatment, this would lead to overfitting of models with poor predictive performance. Part of the work presented in this chapter can also be found in Metelli and Heard (2014).

Network flow data as described in Section 2.1 will be used to demonstrate the method presented. Here, the stream of network flow events from, say, a particular client IP address is viewed as a binary sequence, where the next value in the sequence is 1 if the client connects to a server IP address which has not previously been observed; otherwise, the next value in the sequence is 0. Specifically, Bayesian variable selection is used on NetFlow data from two IP addresses from the Imperial College London domain. After selecting the subset of regressors with highest posterior probability of inclusion, regression coefficient estimates are provided to understand which variables most affect the probability of observing a new edge.

A procedure to detect automatic periodic connections is also presented and variable selection is then repeated on the same data sets after removing automated polling behaviour, for the purpose of comparison. Normal connectivity in a computer network is generally a mixture of automated and human-driven traffic, and so understanding and separating the two is important when proposing models of normal behaviour of network hosts.

The remainder of the chapter is organised as follows: Section 3.1 presents the BMA technique and provides some background on Bayesian MCMC inference. The logistic regression model used to perform variable selection with BMA is presented in Section 3.2, while 3.3 describes the approach used to detect and filter out polling behaviour from the data. Finally, the main results of the application to Imperial College London's computer network are presented in Section 3.4, and results with and without the presence of polling behaviour are compared.

3.1 Bayesian variable selection

When the number of variables is very large, many of them could be unrelated to the feature of interest, for example computer network flow data (see Table 2.1)

contain a large number of fields and some of them, for instance some ports or protocols, may be unrelated to the presence of a new edge. Variable selection identifies a subset of the data related to the feature of interest, which can thus be used as a surrogate for the full set of variables. In the context of new edge formation in a computer network, this involves identifying a subset of variables highly connected to the presence of a new edge. In this way, models and predictions can be based only on the variables included in this subset without loss of quality. Variable selection is a special case of the wider problem of model selection, since each subset of variables can be regarded as a different regression model. A variety of algorithms for searching the model space and selection criteria for choosing between competing models have been proposed over the years (Miller, 2002; Broman and Speed, 2002; Sillanpää and Corander, 2002) both under a frequentist and Bayesian approach. In the Bayesian framework, rather than searching for the single optimal model, the posterior probability of all models within the considered class of models is estimated. Several techniques have been developed (Sillanpää et al., 2004; Meuwissen and Goddard, 2004; Xu, 2003), including the seminal paper on Bayesian model averaging by Raftery et al. (1997), which is the approach considered here.

Bayesian model averaging (BMA) provides a simple and useful way to account for the uncertainty associated with the model selection process. This approach tackles the problem of selection by estimating models for all possible combinations of variables and then constructing a weighted average over the model space. Considering p possible variables, there are $K = 2^p$ possible combinations of different models, without considering the interaction effects between variables.

Let $\mathcal{M} = (M_1, \ldots, M_K)$ be a set of possible models and θ a quantity of interest, such as a model parameter. The posterior distribution of θ given the observed data D, is:

$$\mathbb{P}(\theta|D) = \sum_{k=1}^{K} \mathbb{P}(\theta|D, M_k)\mathbb{P}(M_k|D). \tag{3.1}$$

This can be viewed as a weighted average posterior distribution for θ under each of the models considered, where the weights are the corresponding posterior model

probabilities. In a Bayesian approach, such posterior model probabilities are calculated by applying the Bayes theorem as follows:

$$\mathbb{P}(M_k|D) = \frac{\mathbb{P}(D|M_k)\mathbb{P}(M_k)}{\sum_{l=1}^{K} \mathbb{P}(D|M_l)\mathbb{P}(M_l)}, \tag{3.2}$$

where

$$\mathbb{P}(D|M_k) = \int \mathbb{P}(D|\theta_k, M_k)\mathbb{P}(\theta_k|M_k)\, d\theta_k \tag{3.3}$$

is the integrated, marginal likelihood of model M_k, θ_k is the vector of parameters of model M_k, $\mathbb{P}(\theta_k|M_k)$ is the prior density of θ_k under model M_k, $\mathbb{P}(D|\theta_k, M_k)$ is the model likelihood and $\mathbb{P}(M_k)$ is the prior probability that M_k is the true model. Note that all these probabilities are conditional on the model space $\mathcal{M}$. In Bayes rule, the marginal likelihood in the denominator normalises the posterior density, such that it integrates to one and is a correctly defined probability distribution. Then, the Bayesian model averaged estimate of the parameter θ is

$$\hat{\theta}_{\text{BMA}} = \sum_{k=1}^{K} \hat{\theta}_k \mathbb{P}(M_k|D), \tag{3.4}$$

where $\hat{\theta}_k$ is the posterior mean for model k. Variances of these estimates as well as estimates for other quantities of interest are available from Hoeting et al. (1999) and Viallefont et al. (2001).

The BMA approach has proved to have, on average, better predictive performance than any single model that could reasonably have been selected (Raftery et al., 1995). However, challenges can arise from the evaluation of the marginal likelihoods $\mathbb{P}(D|M_k)$ which do not typically have closed form, and the specification of suitable prior model probabilities $\mathbb{P}(M_k)$ (see Friel and Wyse (2012)). The model space exploration is generally performed following the Occam's principle of scientific parsimony (MacKay, 1991) or via Markov chain Monte Carlo (MCMC) methods (Chipman et al., 1998; Hoeting et al., 1999). Occam's principle will be discussed in Section 3.2, while some background material on MCMC inference is provided below. MCMC methods will be extensively used to perform model infer-

ence throughout this thesis and so some fundamental MCMC concepts are hereby reviewed.

3.1.1 Bayesian inference

When fitting statistical models in a Bayesian framework, we are seeking to identify the parameters which optimise the joint posterior distribution or the marginal posterior distribution of the model of interest, as in (3.1). Unfortunately, these distributions are often not analytically available, and could be multi-modal. Hence, Markov Chain Monte Carlo (MCMC) simulation methods are used to make inference and computing posterior quantities of interest. We next introduce some background theory that have been foundation to build the most commonly used MCMC methods.

3.1.1.1 MCMC methods

Markov chain Monte Carlo (MCMC) methods allows us to simulate an irreducible, aperiodic Markov chain[1] for which the stationary distribution equals the target distribution of interest. This simulation approach takes advantage of the property of Markov chain convergence to produce correlated samples from any target distribution, which after convergence need only be known up to proportionality. For extensive background on MCMC methods see Robert and Casella (2004).

Let π be the joint posterior distribution of interest, known up to a normalising constant, and let $\theta = (\theta_1, \theta_2, \ldots, \theta_M)$ be the sample of π generated by the MCMC sampler, where M is the length of the chain after removing a burn-in period.

At stage $t + 1$ of an MCMC sampling scheme, the so-called transition kernel, denoted $Q(\theta_{t+1}|\theta_t)$, is used to generate the new state θ_{t+1}, conditional on the present state θ_t. In practise, the following restrictive condition, known as detailed balance, is generally imposed to ensure the convergence of the Markov chain to

[1] A Markov chain is a stochastic process with the Markov property, i.e. the next state of the chain depends only on the preceding state.

the required stationary distribution:

$$\pi(\theta_{t+1})Q(\theta_t|\theta_{t+1}) = \pi(\theta_t)Q(\theta_{t+1}|\theta_t). \tag{3.5}$$

A chain which satisfies this property is said to be reversible, as the probability of moving from θ_t to θ_{t+1} is equal to the probability of moving from θ_{t+1} to θ_t, for all values of θ_t and θ_{t+1}. Although this is a stronger than necessary condition to ensure convergence, it is satisfied by a wide variety of algorithms that have been proposed, including the well-known Metropolis-Hastings algorithm.

Metropolis-Hastings algorithm A very general method for constructing a suitable Markov chain is the Metropolis-Hastings (M-H) algorithm (Metropolis et al., 1953; Hastings, 1970), which is outlined in Algorithm 1. The general form of this algorithm is briefly described below as it will be used throughout this thesis to perform posterior inference on the different models presented.

At each iteration t, a new value θ_{t+1} is drawn from a proposal distribution $q(\theta_t, \cdot)$. This new proposal θ_{t+1} is accepted with a probability $a(\theta_t, \theta_{t+1})$, whose form is given in Algorithm 1. A Markov chain generated according to Algorithm 1 will always converge because the detailed balance property is guaranteed to hold (Bernardo and Smith, 2007; Denison et al., 2002). Under this updating scheme, the elements of the vector θ can be conveniently updated jointly or alternatively in a sequential way, depending on how the proposal density is specified.

Other popular MCMC algorithms include Reversible-Jump MCMC (RJMCMC) (Green, 1995; Clyde, 1999) and the Gibbs sampler (Gelfand and Smith, 1990). RJMCMC is an extension of M-H to a varying-dimension problem. This algorithm is capable of reversible jumping between subspaces of differing dimensionality and thus it can be used for comparing models of different dimensions. This property makes it popular in Bayesian variable selection, where models with a varying number of parameters need to be compared. The Gibbs sampler is equivalent to combining Metropolis-Hastings samplers, where all proposed moves, sampled from their full conditional distribution, are accepted with probability 1.

Algorithm 1 Standard Metropolis-Hastings Algorithm

INPUT= an initial value θ_0, a proposal density $q(\cdot|\theta)$, a target density π, a number of samples M
for $t = 1, \ldots, M$ **do**
    Sample $\theta' \sim q(\cdot|\theta_{t-1})$
    $\alpha(\theta_{t-1}, \theta') = \min\{\frac{\pi(\theta')q(\theta_{t-1}|\theta')}{\pi(\theta_{t-1})q(\theta'|\theta_{t-1})}, 1\}$
    Sample $u \sim \text{Uniform}(0, 1)$
    if $u < \alpha(\theta_{t-1}, \theta')$ **then**
        $\theta_t = \theta'$
    else
        $\theta_t = \theta_{t-1}$
OUTPUT=Markov chain $\theta_1, \theta_2, \ldots, \theta_M$ with stationary distribution π

However, it is often the case that the full conditional distributions of some parameters are too difficult to compute, or not analytically available and hence the Gibbs sampler cannot be used. In such cases, the Gibbs sampler can be extended by sequentially updating each element of the parameter vector and using a single Metropolis-Hastings move for those elements that cannot be sampled from their full conditional distributions. Alternatively, a slice sampling (Neal, 2003) move can be used.

3.2 Modelling the rate of occurrence of new edges through variable selection

The work presented here refers to a logistic model, which is a generalised linear model (McCullagh, 1984) for a binary response with associated probability π and logit link function $\text{logit}(\pi) = \log\{\pi/(1-\pi)\}$, assumed to be a linear model of the regressors. Bayesian model averaging as described in Section 3.1 is then applied to this model to find an optimal subset of variables to be included in the model.

3.2.1 Logistic regression model for new edges

In this section, we focus on event data emanating from a single client IP address within the computer network, referred to from this point as IP $\tilde{x}$, i.e. from the time-ordered sequence of connections $(\mathcal{T}, \mathcal{E})$, we only focus on the subprocess $(\mathcal{T}^{\tilde{x}}, \mathcal{E}^{\tilde{x}})$ for which the client mark is $\tilde{x}$.

For the data analysed in this chapter, all event times were originally recorded to the nearest millisecond but then further rounded to the second for computational tractability. Hence, here we model new edge formation taking a discrete time perspective, although the underlying process of arrivals of connections in the graph operates in continuous time as described in Section 2.3. Let the event times take discrete steps $t = 1, 2, \ldots$ and let the increment $dN_{\tilde{x}}(t)$ be a Bernoulli$(\pi_{\tilde{x}}(t))$ random variable for the presence of a new edge at time t, which takes the value of 1 if the next observed edge is new and zero otherwise. Then, $\pi_{\tilde{x}}(t) = \mathbb{P}(dN_{\tilde{x}}(t) = 1)$ and the saturated model for $\pi_{\tilde{x}}(t)$ which we consider here is

$$\begin{aligned}
\text{logit}(\pi_{\tilde{x}}(t)) &= \beta_0 + \beta_1 t + \beta_2 N^+_{\tilde{x}}(t) + \beta_3 I_{\tilde{x},1}(t) + \beta_4 I_{\tilde{x},2}(t) \\
&+ \beta_5 Duration + \beta_6 Protocol + \beta_7 PortSSH + \\
&+ \beta_8 PortHTTP + \beta_9 PortDNS + \beta_{10} PortIMAP + \\
&+ \beta_{11} URGENT + \beta_{12} ACK + \beta_{13} PUSH + \\
&+ \beta_{14} RESET + \beta_{15} SYN + \beta_{16} FIN,
\end{aligned} \tag{3.6}$$

where $N^+_{\tilde{x}}(t)$ is the number of unique server IP addresses connected to by IP $\tilde{x}$ at time t, $I_{\tilde{x},1}(t)$ an indicator for whether the last edge was new, and $I_{\tilde{x},2}(t)$ an indicator for whether the last two edges were both new. These variables have been previously defined in (2.7), (2.5) and (2.6). Several terms summarising the last flow are also included: the duration of the connection, the protocol, four indicator variables for the server port used and six indicator variables derived from the TCP flag fields (see Table 3.1).

The dummy variable for the protocol field indicates whether the protocol for the last connection was TCP or UDP. From the full extent of ports used, four

Variables	Categories	Values
Protocol	TCP	6
	UDP	17
Port	SSH	22
	HTTP	80/8080
	DNS	53
	IMAP	143
TCP fields	URGENT	0/1
	ACK	0/1
	PUSH	0/1
	RESET	0/1
	SYS	0/1
	FIN	0/1

Table 3.1: Dummy variables included in the analysis.

dummy variables were created indicating whether the application was SSH (Secure Shell protocol), HTTP (Hypertext Transfer Protocol, used by web browsers), DNS (Domain Name System protocol for converting domain names to IP addresses) or IMAP (email applications including Outlook, Eudora and Thunderbird). Finally, the TCP variables indicate that the Urgent pointer field and Acknowledgement field are significant, the buffered data are pushed to the receiving application, the connection is reset, the sequence numbers are synchronised and finally that the connection with the sender is closed.

3.2.2 Bayesian model averaging

Following the approach described in Section 3.1, we seek to calculate the posterior probability of each possible model M_k, as in (3.2). Let $\beta_k = (\beta_0, \ldots, \beta_{d_k})$ the vector of model coefficients under model M_k and let $\mathbb{P}(\mathcal{E}|\mathcal{T}, \beta, M_k)$ be the logistic regression likelihood for all the parameters included in model M_k, conditioning on the event times $\mathcal{T}$. Note that each possible model is a subset of the saturated model presented in (3.6), for which $\beta_k = (\beta_0, \ldots, \beta_{16})$. The marginal likelihood in (3.3) does not here have closed form, and so we use the Bayesian Information

Criterion (BIC) (Schwarz, 1978) to obtain the following approximation for the marginal log-likelihood defined in (3.2),

$$\log\mathbb{P}(\mathcal{E}|\mathcal{T}, M_k) \approx \log\mathbb{P}(\mathcal{E}|\mathcal{T}, \hat{\beta}_k, M_k) - \frac{d_k}{2}\log n, \tag{3.7}$$

where n is the number of observed events in the time-ordered sequence of data $(\mathcal{E}, \mathcal{T})$. Then, for each model M_k the joint posterior distribution of all unknown parameters is given by

$$\mathbb{P}(M_k|\mathcal{T}, \mathcal{E}) \propto \mathbb{P}(\mathcal{E}|\mathcal{T}, M_k)\mathbb{P}(M_k). \tag{3.8}$$

The implementation of BMA involves the specification of two prior distributions: a prior for each parameter of the model and the prior probability of each model. We assume all combinations of predictors to be equally likely a priori so that $p(M_k) = \frac{1}{K}$, for each k, meaning that the posterior model probability for all possible models were computed using the diffuse, but proper, prior distributions (Raftery, 1988; Hoeting et al., 1999). Uniform prior distributions are then assumed for each regression coefficient and posterior estimates are inferred through the Metropolis-Hastings algorithm. We explore the model space with the Occam's principle of parsimony for scientific explanation (MacKay, 1991), following the procedure in Hoeting et al. (1999). Occam's razor is inherently invoked in the Bayesian framework: considering two models M_l and M_k, if $M_l \subset M_k$ then the marginal likelihood (3.7) will be lower for M_l than for M_k, if M_k already fits the data well; however, there will always be some other data that will be modelled better through M_l than M_k. Therefore, the Bayesian paradigm do not typically suffer from overfitting problems and provide a simple way to perform model comparison. In particular, the algorithm used here iteratively compares the two nested models M_l and M_k, and in case M_l is rejected, then all sub-models of M_l are also rejected. The set of selected model can be written as

$$\mathcal{M}' = \left\{ M_k : \frac{\max_l \mathbb{P}(M_l|\mathcal{T}, \mathcal{E})}{\mathbb{P}(M_k|\mathcal{T}, \mathcal{E})} \leq 1 \right\}, \tag{3.9}$$

and thus, models not belonging to $\mathcal{M}'$ will be excluded from the calculation of the posterior probability in (3.1). This will leave us with a set of nested plausible models for monitoring the normal rate of occurrence of the new edges in the graph.

When building such a model for new edge formation, a difficulty arises from the fact that a large portion of the traffic analysed generally tends to be automatically generated. This polling behaviour can be either indicative of malicious attacks or it can be caused by automated requests permitted by the client. In both cases, network flow data of this nature may interfere with the reliability of the model, since bursts of new edge formation would be predictably broken up by periodic connections to established hosts. Separating this behaviour from the human-level traffic is therefore very important in the present context. A method for detecting automated polling traffic is thus described below.

3.3 Detecting automated polling traffic

In a computer network, automated traffic tends to be strongly periodic at higher frequencies compared to the traffic generated by a human and thus detecting periodic behaviours can be indicative of polling. Examples of automated, legitimate polling may include connections to the network administrative servers; polling requests for file updates from a running Dropbox client; and regular automatic refreshing of a live.com page in a web browser. Following Heard et al. (2014), we sequentially conduct a Fourier analysis of the traffic occurring along each edge in the network between a specific IP address and a given host. In particular, a discrete Fourier transform, which is a powerful tool in signal processing, is used to obtain a spectrum, or *periodogram*, for each edge in the network. This will give us the contribution of each frequency to the signal on that edge and peaks in the spectrum indicate the presence of polling at that frequency. When polling is identified, the corresponding edges are removed from the bulk data. In this way, the remaining data, i.e. the portion of data that should reflect human-driven traffic only, can be used for suitable statistical modelling.

For two IP addresses x and y, and for $t \geq 0$, $N_{xy}(t)$ is the counting process of NetFlow events with source IP x and destination IP y occurring by time t, as defined in (2.2). This quantity, which monitors the activity on one edge of the network graph, is here treated as a discrete time process. Let the increment $dN_{xy}(t) \in \{0, 1\}$ correspond to the presence or absence of connections from x to y at time t. Then, the periodogram at frequency $f > 0$ is defined via the discrete Fourier transform (Halliday and Rosenberg, 1999),

$$S_{xy}(f) = \left| \sum_{t=1}^{T} (dN_{xy}(t) - N_{xy}(T)/T) e^{-i2\pi ft} \right|^2 \Big/ T, \tag{3.10}$$

where $N_{xy}(T)/T$ is the mean rate of the process, and by assuming the process to be approximately homogeneous, we have that $dN_{xy}(t) - N_{xy}(T)/T$ is approximately stationary.

A large value of $S(f)$ corresponds to strong periodic behaviour at frequency f. The fast Fourier transform allows $S(f)$ to be calculated efficiently for the Fourier frequencies $f \in \mathcal{F} = \{0, 1/(T\Delta t), 2/(T\Delta t), \ldots, (T/2 - 1)/(T\Delta t)\}$. For each frequency, we can test its periodicity via Fisher's g-test (Fisher, 1929) as follows

$$g_{xy} = \max_{f \in \mathcal{F}} S_{xy}(f) \Big/ \sum_{f' \in \mathcal{F}} S_{xy}(f'). \tag{3.11}$$

For a realised value g of (3.11), the corresponding p-value is the probability of observing a value of g_{xy} greater than or equal to g under the null hypothesis of a purely random process. Under asymptotic normality, the distribution of g can be computed exactly and the p-value is given by

$$1 - \exp(-m * \exp[\{-g * (m - 1 - \log m)/(m - g)\}]), \tag{3.12}$$

where $m = \lfloor T/2 \rfloor$. Full details for this calculation can also be found in (Wichert et al., 2004). Frequencies with p-values < 0.01 were classified as corresponding to periodic behaviour.

3.4 An application to Imperial College London NetFlow data

The model described in Section 3.2 and the approach to filter out polling as described in Section 3.3 is here applied to the real network flow data from the computer network of Imperial College London. The data have been obtained from router level flow records gathered from one of the main networks of the Imperial College London internal domain, which have been described in more details in Section 2.1.

3.4.1 Network flow data

The data were gathered from a large available collection of flow data, drawn from two collection periods of lengths 96 days and 53 days respectively, separated by a break in collection of 43 days. Two small, differently motivated subsets of these data have been analysed.

The first subset of data contains the client network flow records for IP $\tilde{x}$. Variable selection was conducted and then repeated on the same data set after removing automated periodic connections as described in Section 3.3. The second subset of data contains the client network flow records of another IP address, IP z, which was found to be infected towards the end of the data collection period. Again two analyses were performed, this time considering separating the two continuous collection intervals; the first interval contained no known malicious attack, whereas the second was known to contain infected data.

3.4.2 Variable selection results

There were 15 potential candidate variables in (3.6), implying a total number of 2^{15} possible models. For MCMC, the Metropolis-Hastings algorithm was used with a total number of iterations set to 10,000, after a burn-in period of size 1,000.

Tables 3.2 and 3.3 contain the posterior means, standard deviations and posterior effect probabilities $P(\beta \neq 0|\mathcal{T}, \mathcal{E})$, based on the data for IP $\tilde{x}$ with and without polling traffic, respectively. For each coefficient, the posterior probability of inclusion $P(\beta \neq 0|\mathcal{T}, \mathcal{E})$ is measured by the proportion of times a model containing that coefficient was selected in $\mathcal{M}'$. Note that these parameter estimates and standard deviations directly incorporate model uncertainty.

Variable	**Coefficient**	**Standard Error**	$\mathbb{P}(\boldsymbol{\beta \neq 0}|\boldsymbol{\mathcal{T}, \mathcal{E}})$ **(%)**
Intercept	-4.01516	0.04825	100
t	·	·	0
$N^+_{\tilde{x}}(t)$	-0.00019	0.00017	100
$I_{\tilde{x},1}(t)$	2.28091	0.00734	100
$I_{\tilde{x},2}(t)$	0.77250	0.0161	78
Duration	-0.00064	0.00142	100
Protocol	0.39602	0.04672	40
PortSSH	·	·	0
PortHTTP	0.67701	0.04472	70
PortDNS	·	·	0
PortIMAP	·	·	0
Urgent	·	·	0
Ack	0.60023	0.05237	100
Push	0.46302	0.06064	40
Reset	·	·	0
Syn	0.98544	0.07321	80
Fin	0.56254	0.06311	41

Table 3.2: Coefficient estimates from logistic regression for IP $\tilde{x}$.

The results show that the most influential variables on the arrival of new edges are $N^+_{\tilde{x}}(t)$, $I_{\tilde{x},1}(t)$, $I_{\tilde{x},2}(t)$, and the duration, the use of a web port and the acknowledgement field in the last connection. The variables corresponding to the other ports included in the analysis were never selected by the procedure, highlighting the significant weight of a web browser connection for forming new edges. Surprisingly, the variable t, intuitively influential on the response variable, was never selected. This may be due to the presence of a strong correlation with the variable $N^+_{\tilde{x}}(t)$, which possibly mitigates its effect. The variables $I_{\tilde{x},1}(t)$ and $I_{\tilde{x},2}(t)$ have the highest (positive) posterior mean effects; intuitively, this tells us that arrivals of new edges occur in bursts, and so knowledge that the last edges were new is very influential on the arrival of a new edge next time.

Variable	Coefficient	Standard Error	$\mathbb{P}(\boldsymbol{\beta} \neq \mathbf{0} \mid \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{E}})$ (%)
Intercept	-2.101000	0.074942	100
t	-0.000002	0.000003	100
$N_{\tilde{x}}^{+}(t)$	-0.004282	0.000039	100
$I_{\tilde{x},1}(t)$	0.468421	0.064454	100
$I_{\tilde{x},2}(t)$	1.342271	0.123445	100
Duration	·	·	0
Protocol	-2.067323	0.065836	100
PortSSH	-0.308322	0.070561	97
PortHTTP	0.313198	0.040813	100
PortDNS	·	·	0
PortIMAP	·	·	0
Urgent	·	·	0
Ack	0.584367	0.070741	100
Push	-0.098733	0.062122	0.6
Reset	-0.233741	0.103221	1.8
Syn	0.970156	0.063349	100
Fin	-0.143887	0.058094	3.5

Table 3.3: Coefficient estimates from logistic regression for IP $\tilde{x}$ with polling data removed.

IP $\tilde{x}$ is a relatively active node, which acts as a client in 2,644,780 NetFlow events within the period of data collection. The corresponding distribution of the time of day of the client connection events for IP $\tilde{x}$ is shown in Figure 3.1. It can be noted that this distribution is much flatter than the plot in Figure 2.1 for the global pattern of the network shown in the previous chapter. Furthermore, it has unusual peaks which are not consistent with diurnal human behaviour. This has motivated us to repeat the analysis after filtering out the automated periodic connections.

A total number of 1377 IP addresses were detected as responsible for polling traffic, using the Fourier analysis presented in Section 3.3. Some interesting examples of IP addresses removed include Turkish, Honk Kong and several Chinese boxes trying to gain access to the machine via remote SSH connection. All of these IP addresses were then found to be reported for malicious activity.

Comparing results obtained when the polling behaviour was removed, it can be noted that PortHTTP increases to a 100% probability of inclusion, while the indicators of data pushing and end of connection become less influential. Importantly,

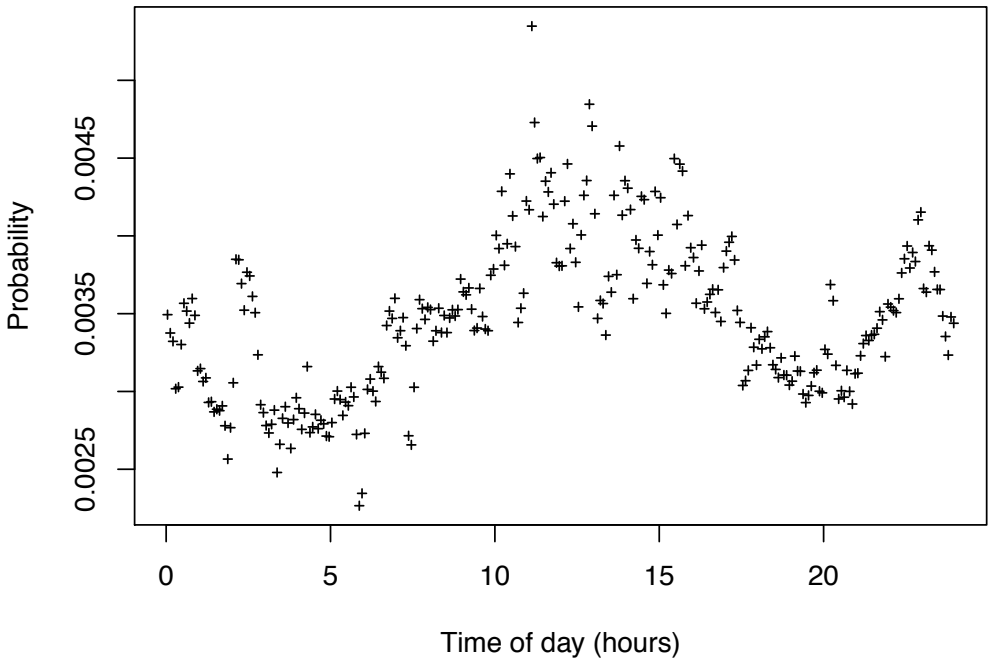


Figure 3.1: The distribution of unfiltered client event times for IP $\tilde{x}$, estimated to five minute bins using the data obtained over 97 days.

$I_{\tilde{x},2}(t)$ is now included in the model 100% of the time, supporting the hypothesis that bursts of new edge activity are more easily seen to be significant once polling traffic has been filtered out. Moreover, the variable t now recovers its intuitive influence on the response variable. The protocol variable increases to a 100% probability of inclusion, showing the importance of distinguishing whether the protocol for the last connection was TCP or UDP. Specifically, it has the highest negative posterior mean effect, indicating that the presence of a UDP connection decreases the probability of the next edge to be new. This might be due to the fact that UDP protocols are mostly used for live broadcasts and online connections, which may be less relevant in an attack setting. Furthermore, the indicator variable of an SSH connection is now another influential variable of the model, as well as the synchronisation of sequence numbers, which increases in posterior probability of inclusion. Surprisingly, the duration of the last connection was never selected, while in the unfiltered case this variable was always selected.

The other two data sets analysed consist of client connections of an IP address on the Imperial College network that was eventually infected, IP z (*cf.* Section 3.4.1). They correspond to the flow data collections either side of a 43 day break, where the second collection is suspected to contain the infection event. Figure 3.2 shows the number of events occurring in each 20-minute bin of time, before and after the break in data collection. The circled outlier is very pronounced, indicating clear anomalous behaviour occurring in the second period.

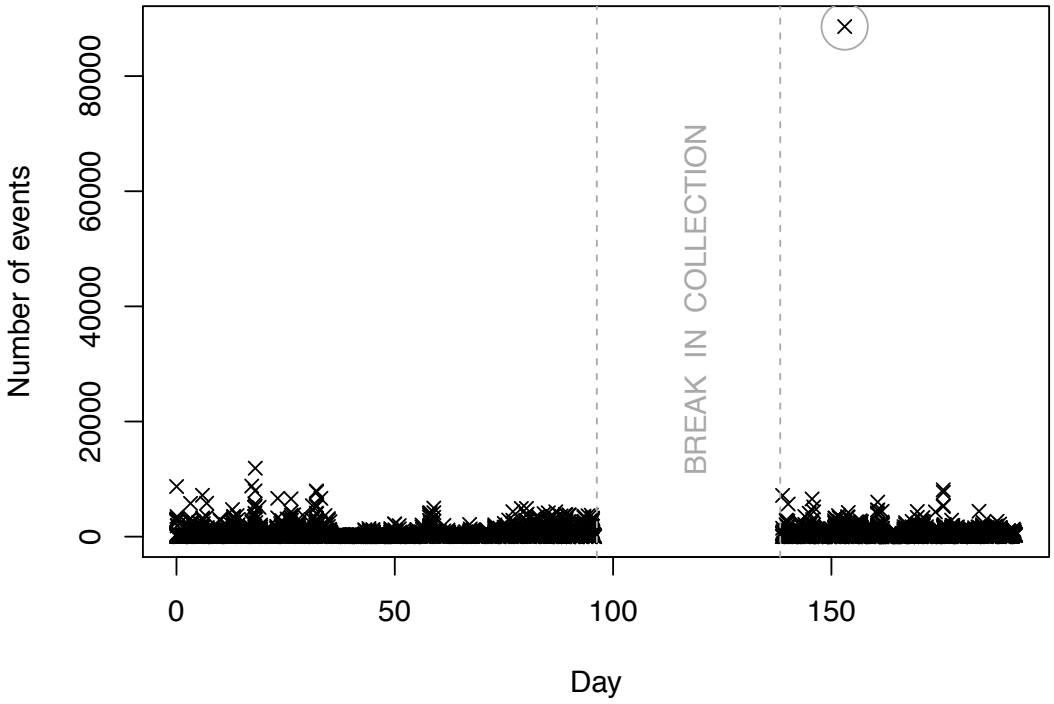


Figure 3.2: The number of client event observed in the network flow data for a particular IP address on the network, IP z, which towards the end of the collection period was eventually found to be compromised.

Results for the first and second collection periods are reported in Tables 3.4 and 3.5 respectively. Again, the most influential variables on the rate of arrival of new edges are $N_z^+(t)$, $I_{z,1}(t)$, $I_{z,2}(t)$, and the duration, the use of a web port and the acknowledgement field of the last connection. The indicator variable corresponding to a DNS connection acquires more importance with respect to the previous analysis, especially with the data set containing the malicious event: here

this indicator is always selected. Furthermore, the synchronisation variable has a very different impact on the two data sets. Whereas in the first one it does not seem influential, in the second it is selected with a 100% probability, indicating the importance of a matching connection after having sent a connection request.

Variable	**Coefficient**	**Standard Error**	$\mathbb{P}(\boldsymbol{\beta} \neq \mathbf{0}\|\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{E}})$ (%)
Intercept	-3.59102	0.05861	100
t	·	·	0
$N_z^+(t)$	-0.00731	0.00971	70
$I_{z,1}(t)$	1.74021	0.01025	90
$I_{z,2}(t)$	0.68375	0.03573	100
Duration	0.00184	0.00067	100
Protocol	1.43734	0.05111	57
PortSSH	·	·	0
PortHTTP	0.36513	0.03826	100
PortDNS	1.09809	0.07823	56
PortIMAP	·	·	0
Urgent	·	·	0
Ack	1.12033	0.06221	100
Push	·	·	0
Reset	·	·	0
Syn	0.09141	0.61321	12
Fin	·	·	0

Table 3.4: Coefficient estimates from logistic regression for IP z before infection.

Variable selection has narrowed the choice to a few plausible models, yielding a feasible implementation of Bayes factor's model comparison (Jeffreys, 1961; Kass and Raftery, 1995). Following the results of Section 3.4, we calculate Bayes factors B_{m0}, for the saturated model (3.6) against a null model M_0 (a model with no covariates included), for the different subsets of data analysed:

$$B_{m0} = \frac{\mathbb{P}(\mathcal{E}|\mathcal{T}, M_m)}{\mathbb{P}(\mathcal{E}|\mathcal{T}, M_0)}, \tag{3.13}$$

Here, the posterior probability for each model is now calculated via MCMC, using M-H moves. As exemplification, only three Bayes factors were computed for each dataset, each comparing the null model with the best three models selected by the variable selection procedure, i.e. the models with the highest probabilities of inclusion. Results for the four different datasets are reported in Table 3.6,

Variable	Coefficient	Standard Error	$\mathbb{P}(\boldsymbol{\beta} \neq \mathbf{0} \mid \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{E}})$ (%)
Intercept	-6.58301	0.06965	100
t	·	·	0
$N_z^+(t)$	-0.00053	0.00371	100
$I_{z,1}(t)$	1.24022	0.00524	90
$I_{z,2}(t)$	0.59323	0.02574	100
Duration	0.00084	0.00006	100
Protocol	1.03676	0.05913	60
PortSSH	·	·	0
PortHTTP	0.46661	0.02841	100
PortDNS	3.49845	0.64825	90
PortIMAP	·	·	0
Urgent	·	·	0
Ack	1.82077	0.05822	100
Push	·	·	0
Reset	·	·	0
Syn	0.51412	0.51389	100
Fin	·	·	0

Table 3.5: Coefficient estimates from logistic regression for IP z after infection.

confirming the considerable contribution provided by the covariates included in the model.

Data subset	Model	Bayes factor B_{m0}
IP $\tilde{x}$ unfiltered data	$M_1 : N^+_{\tilde{x}}(t) + I_{\tilde{x},1}(t) + I_{\tilde{x},2}(t) + Duration + Protocol + PortHTTP + Ack + Syn$	6.4×10^4
	$M_2 : N^+_{\tilde{x}}(t) + I_{\tilde{x},1}(t) + Duration + PortHTTP + Ack + Fin$	6.1×10^3
	$M_3 : N^+_{\tilde{x}}(t) + I_{\tilde{x},1}(t) + I_{\tilde{x},2}(t) + Duration + PortHTTP + Ack$	6.3×10^3
IP $\tilde{x}$ filtered data	$M_1 : N^+_{\tilde{x}}(t) + I_{\tilde{x},1}(t) + I_{\tilde{x},2}(t) + Duration + PortHTTP + Ack + Syn$	2.3×10^3
	$M_2 : N^+_{\tilde{x}}(t) + I_{\tilde{x},1}(t) + I_{\tilde{x},2}(t) + Duration + PortHTTP + Ack + Reset$	2.0×10^3
	$M_3 : N^+_z(t) + I_{z,1}(t) + I_{z,2}(t) + Duration + PortHTTP + Ack$	1.9×10^3
IP z before infection	$M_1 : N^+_z(t) + I_{z,1}(t) + I_{z,2}(t) + Duration + Protocol + PortHTTP + PortDNS + Ack + Syn$	1.8×10^4
	$M_2 : N^+_z(t) + I_{z,1}(t) + I_{z,2}(t) + Duration + PortHTTP + PortDNS + Ack$	1.7×10^4
	$M_3 : I_{z,1}(t) + I_{z,2}(t) + Duration + PortHTTP + Ack$	0.7×10^4
IP z after infection	$M_1 : N^+_z(t) + I_{z,1}(t) + I_{z,2}(t) + Duration + PortHTTP + PortDNS + Ack + Syn$	1.8×10^4
	$M_2 : N^+_z(t) + I_{z,2}(t) + Duration + PortHTTP + PortDNS + Ack + Syn$	1.8×10^4
	$M_3 : N^+_z(t) + I_{z,2}(t) + Duration + PortHTTP + Ack + Syn$	1.1×10^4

Table 3.6: Bayes factors for the best three models against the null model, for the four different subsets of data analysed.

Chapter 4

Biclustering Methods for Computer Network Data

The importance of clustering a computer network for cyber-security purposes is two-fold. Firstly, it allows us to learn the underlying latent structure of the network, where hosts sharing similar connection behaviour can be grouped together. Secondly, it represents an important step when modelling the characteristics of new edge formation, since it can provide relevant cluster-level covariates to include in the model, as later discussed in Chapter 5 and Chapter 6. Specifically, clustering a computer network involves clustering both the set of clients, or source IP addresses, and the set of servers, or destination IP addresses, and thus can be viewed as a biclustering problem. This chapter mainly addresses the problem of biclustering the adjacency matrix of a bipartite computer network graph.

Clustering is defined as the grouping of similar objects (Hartigan, 1975), while biclustering, a natural extension of the clustering problem, is the simultaneous clustering of both the rows and the columns of a data matrix creating a block-like pattern within the matrix (Hartigan, 1972). There are many different ways to define cluster similarity and depending on the definition chosen, the resulting cluster configurations might strongly differ. Most of clustering methods can be broadly divided into two categories: hierarchical clustering and flat clustering. The former

creates a hierarchy of clusters using a top-down or a bottom-up technique, whilst the latter is an efficient and simpler approach where no hierarchy is created. Hierarchical clustering is potentially more informative and accurate than unstructured, flat clustering and it does not require the a priori specification of the number of clusters. However, this comes at the cost of lower efficiency, with a complexity which is at least quadratic in the number of observations, compared to the linear complexity of algorithms such as standard k-means (MacQueen, 1967; Hartigan and Wong, 1979). Due to the fact that computer networks are large, our aim is for the methods in this chapter to be efficient while still retaining their desirable properties; for the hereby proposed hierarchical clustering such properties are resulting from a Bayesian paradigm. In the following, both hierarchical and fast, flat clustering techniques are introduced and compared, with the purpose of assessing the balance between clustering accuracy and computational feasibility.

The structure of this chapter is as follows: Section 3.1 introduces hierarchical agglomerative model-based clustering in the context of computer networks and its extension to agglomerative model-based biclustering. Section 3.2 presents two techniques for performing flat biclustering, while in Section 3.3 a simulation study is performed for assessing the performance of the different methods introduced.

4.1 Hierarchical agglomerative clustering methods

Hierarchical clustering creates a tree structure of cluster configurations, known as a *dendrogram*, each with a different number of clusters. This can be achieved in an *agglomerative* fashion, starting with all observations in singleton clusters and then iteratively merging clusters based on some optimisation criterion until only one remains, or in a *divisive* fashion, where the data are separated repeatedly into finer groups. Agglomerative clustering is faster since the number of possible merges is lower than possible splits. For this reason, agglomerative hierarchical clustering has been the most commonly used classification scheme (Duda and Hart, 1973).

Some popular optimisation criteria are the sum within-group sum of squares

(Ward, 1963) and some given distance measure such as the Euclidean distance between cluster means or the shortest distance between clusters (Gower and Ross, 1969). These heuristic approaches remain popular although it is not easy to assess some of their statistical properties, such as defining the uncertainty of the allocation of an item to a group or quantifying the probability of two items belonging to the same group. Model-based clustering (Banfield and Raftery, 1993; McLachlan and Basford, 1988) offers a principled alternative to these issues. In this approach, clusters are defined as groups of objects which have been generated by the same underlying probability distribution. An extensive review of model-based cluster methods in general can be found in Bock (1996).

In the frequentist framework, the estimation procedure for model-based agglomerative clustering methods consists of fitting a model at each stage of the hierarchy, and then successively merging those clusters with the greatest increase in the classification likelihood (Banfield and Raftery, 1993). In the Bayesian approach, which is the one adopted here, it is the marginal posterior distribution which is maximised by each merge. This approach is taken, for example, in the MCLUST procedure of Fraley and Raftery (2002), which fits Gaussian process clusters optimised against a BIC criterion. The advantage of using a Bayesian approach is to provide a posterior probability, which conveniently enables comparisons between each configuration found.

4.1.1 Bayesian model-based clustering

As discussed above, in model-based agglomerative clustering each data point is at first assigned to its own cluster and successively the cluster pairs which maximise an objective function are merged. A model-based approach defines a probability model as the clustering objective function, whilst the Bayesian formulation enables the partition of items into subsets to be a parameter of the probability model, subject to prior assumptions. Excellent background on Bayesian model-based clustering can be found in Banfield and Raftery (1993) and Fraley and Raftery (2002).

There are two main approaches to Bayesian model-based clustering. The first uses MCMC methods to obtain a sample from the posterior distribution in order to infer an optimal cluster configuration (Booth et al., 2008; Medvedovic et al., 2004; Qin, 2006). MCMC approaches allow to fit rather complex models and to estimate the number of groups at the same time as the other parameters (Richardson and Green, 1997). However, as the number of possible cluster configurations increases, MCMC schemes may suffer from poor mixing and very slow running times. The second is a deterministic approach which relies upon agglomerative hierarchical clustering to iteratively merging the cluster pair which provide the largest multiplicative change in posterior probability. Examples include Heller and Ghahramani (2005), which present an algorithm for Bayesian hierarchical clustering based on evaluating marginal likelihoods of Dirichlet process mixtures and Heard et al. (2006), which introduce a Bayesian model-based hierarchical clustering algorithm for locally maximise the marginal posterior distribution for each possible number of clusters. In the following sections, the latter approach is adopted.

4.1.2 Clustering model for computer network data

For notational convenience, suppose here that the clients and servers have been numbered such that $X = \{1, \ldots, |X|\}$ and $Y = \{1, \ldots, |Y|\}$. After observing n events in the computer network graph $(t'_1, (x'_1, y'_1)), \ldots, (t'_n, (x'_n, y'_n))$, let $A \in \{0,1\}^{|X|\times|Y|}$ be the $|X|\times|Y|$ adjacency matrix with entries $A_{x,y} = \sum_{i=1}^{n} \mathbb{1}_{(x,y)}\{(x'_i, y'_i)\}$ indicating which of the possible edges have been observed, i.e. $A_{xy} = 1$ if and only if client x connected to server x at least once. Note that here $A \equiv G_{t'_n}$. In this section we focus on clustering the client set only, with the purpose of dividing the data matrix A into K row blocks (after a permutation). In the next section this will be extended to enable biclustering of both clients and servers.

For the client set X, a cluster configuration $\mathbb{C}$ is a partition of the index set of clients $\{1, \ldots, |X|\}$ into K non-empty subsets $\{C_1, \ldots, C_K\}$, where the x^{th} client is allocated to the k^{th} cluster if and only if $x \in C_k$. Let $\boldsymbol{\theta}$ be a $K \times |Y|$ matrix of cluster-specific parameters, such that θ_{ky} is the probability that a client in cluster

k will connect to server y. For the data matrix A, the implied likelihood function for the cluster configuration and parameter is given by

$$L(A|\mathbb{C}, \boldsymbol{\theta}) = \prod_{k=1}^{K} \prod_{x \in C_k} \prod_{y=1}^{|Y|} \theta_{ky}^{A_{xy}} (1 - \theta_{ky})^{1 - A_{xy}}. \tag{4.1}$$

The Bayesian framework requires the specification of prior distributions for obtaining the joint posterior distribution $\mathbb{P}(\boldsymbol{\theta}, \mathbb{C}|A)$ of model parameters. We choose independent, conjugate Beta(a, b) priors for the probabilities θ_{ky} with density function:

$$f(\theta_{ky}) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} {\theta_{ky}}^{a-1} (1 - \theta_{ky})^{b-1}. \tag{4.2}$$

We assume exchangeability when specifying a prior for the cluster configuration $\mathbb{C}$ so that a priori no two observations are more likely to belong to the same cluster. Specifically, we use a uniform distribution over the space of all possible cluster configurations. Alternatively, the popular Dirichlet process prior could be employed (Kim et al., 2006).

The primary interest here is the allocation of the objects to clusters and so the marginal posterior distribution of $\mathbb{C}$ is the objective function that we seek to maximise. This can be obtained in closed-form up to proportionality as

$$\begin{aligned} \mathbb{P}(\mathbb{C}|A) \propto \mathbb{P}(A|\mathbb{C}) &= \int L(A|\mathbb{C}, \boldsymbol{\theta}) f(\boldsymbol{\theta}) d\boldsymbol{\theta} = \\ &= \prod_{k=1}^{K} \prod_{y=1}^{|Y|} \frac{\Gamma(a+b)\Gamma(a+m_{ky})\Gamma(b+n_k-m_{ky})}{\Gamma(a)\Gamma(b)\Gamma(a+b+n_k)}, \end{aligned} \tag{4.3}$$

where $m_{ky} = \sum_{x \in C_k} A_{xy}$ and $n_k = |C_k|$ is the number of clients in cluster k.

In the process of deciding upon possible merges and splits, agglomerative clustering requires a measure of similarity between any pair of clusters to be specified. An intuitive choice is the multiplicative change in the posterior probability in (4.3)

which results from merging the clusters pair:

$$S_{km}^{(X)} = \frac{\mathbb{P}(\mathbb{C}^{km}|A)}{\mathbb{P}(\mathbb{C}|A)}, \tag{4.4}$$

where $\mathbb{C}^{km}$ represents the cluster configuration obtained from $\mathbb{C}$ by merging cluster k and cluster m. By (4.3), this quantity is equivalent to the ratio of the marginal likelihoods of the two cluster configurations, with the model parameters integrated out, enabling a fair comparison between models with a different number of parameters. The Bayesian framework automatically embodies the principle of parsimony for scientific explanation which states that simpler models will always be preferred to unnecessarily complex ones.

4.1.3 Clustering algorithm

The algorithm is initiated with $\mathbb{C} = \{\{1\}, ..., \{|X|\}\}$, i.e. each client is placed in its own cluster. The pair of clusters which maximise the similarity measure in (4.4) are then iteratively merged until all clients reside in a single cluster. A nested sequence of cluster configurations is created and the optimal configuration corresponds to the configuration in the hierarchy with the largest marginal posterior probability in (4.3). An algorithm describing this procedure is given in Algorithm 2. In the next section, this algorithm is extended for performing biclustering, in order to cluster both the rows (i.e. clients) and the columns (i.e. servers) of the data matrix.

4.1.4 Bayesian model-based biclustering

Biclustering, also known as co-clustering or two-way clustering, refers to the simultaneous clustering of the rows and the columns of a data matrix, identifying "checkerboard" patterns, or sets of rows and sets of columns in the matrices that are significantly associated. This can reveal underlying structures which are potentially obscured when clustering either the rows or the columns independently. There are many different biclustering methods developed in the literature and these

Algorithm 2 Model-based Clustering

INPUT=($|X| \times |Y|$ data matrix A)
Set $\mathbb{C} = \{\{1\}, \ldots, \{|X|\}\}$, i.e. each client in its own cluster
Calculate $\mathbb{P}(\mathbb{C}|A)$
Calculate the similarity $S_{km}^{(X)}$ for each pair of clusters k and m
while $|\mathbb{C}| > 1$ **do**
    Identify the cluster pairs (k^*, m^*) with largest similarity measure
    Merge row clusters k^* and m^* to form a new cluster c^*
    Update $\mathbb{C}$
    Recalculate $S_{kk^*}^{(X)}$ for each $k \neq c^*$
Identify the optimal partition in the hierarchy : $\mathbb{C}^* = \arg\max_{\mathbb{C}} \mathbb{P}(\mathbb{C}|A)$
OUTPUT=(Cluster configurations hierarchy, optimal cluster configuration $\mathbb{C}^*$)

have been mostly popular in gene expression analysis (Li et al., 2012; Cheng and Church, 2000; Fowler and Heard, 2012) and text mining (Busygin et al., 2008). In its simplest form, biclustering consists of independently clustering rows and columns and subsequently applying both cluster structures to the data (Alon et al., 1999). However, this does not take into account possible interactions between rows and columns. Simultaneous clustering of both rows and columns has been at first introduced by Hartigan (1972), which discusses a method to iteratively partitioning either the rows or the columns of the data according to the partition which minimises the sum of squares. This partitioning method is computationally consuming and therefore not applicable to large data sets.

Fast and efficient methods suitable for large data sets have been later developed. For instance, Cheng and Church (2000) introduced an algorithm which defines biclusters as sub-matrices with the mean squared residue score below a user-defined threshold by iteratively adding or removing rows and columns of the data matrix, thus yielding a faster performance. A graph-theoretic approach, coupled with statistical modelling of the data, is presented by Tanay et al. (2002). In this

framework, the matrix is modelled as a bipartite graph, a bicluster is defined as a subgraph, and a likelihood score is used for assessing the significance of observed subgraphs.

4.1.5 Biclustering model for computer network data

Model-based biclustering is similar to the model-based clustering procedure described in the previous section with the assumption that here the observations assigned to each bicluster are generated by the same underlying distribution. Considering again the $|X| \times |Y|$ relational data matrix A, we define a bicluster configuration as a row partition for the client set X, $\mathbb{C} = \{C_1, \ldots, C_L\}$ into L non-empty row subsets and a column partition for the servers set Y, $\mathbb{S} = \{S_1, \ldots, S_M\}$ into M non-empty column subsets. The Cartesian product of $\mathbb{C}$ and $\mathbb{S}$ partitions the data matrix into biclusters, s.t. the $(x, y)^{th}$ element of A is assigned to the bicluster (l, m) if and only if $x \in C_l$ and $y \in S_m$.

Let $\boldsymbol{\theta}$ be a $L \times M$ matrix of bicluster-specific parameters, such that θ_{lm} is the probability that a client in cluster l will connect to a server in cluster m. The implied likelihood function for the data matrix A is now given by

$$L(A|\mathbb{C}, \mathbb{S}, \boldsymbol{\theta}) = \prod_{l=1}^{L} \prod_{m=1}^{M} \prod_{\substack{x \in C_l, \\ y \in S_m}} \theta_{lm}^{A_{xy}} (1 - \theta_{lm})^{1 - A_{xy}}. \tag{4.5}$$

As before, we choose a uniform distribution over the space of all possible cluster configurations, while the prior distributions for each bicluster parameter θ_{lm} are taken to be independent, conjugate beta distributions as per (4.2). The objective function we now seek to maximise is the marginal posterior distribution of the cluster configuration $(\mathbb{C}, \mathbb{S})$ and it can be again computed up to proportionality in

closed-form as

$$\begin{aligned}\mathbb{P}(\mathbb{C},\mathbb{S}|A) \propto \mathbb{P}(A|\mathbb{C},\mathbb{S}) = \int L(A|\mathbb{C},\mathbb{S},\boldsymbol{\theta}) f(\boldsymbol{\theta}) d\boldsymbol{\theta} = \\ = \prod_{l=1}^{L}\prod_{m=1}^{M} \frac{\Gamma(a+b)\Gamma(a+M_{lm})\Gamma(b-M_{lm}+n_l s_m)}{\Gamma(a)\Gamma(b)\Gamma(a+b+n_l s_m)},\end{aligned} \tag{4.6}$$

where $M_{lm} = \sum_{x\in C_l}\sum_{y\in S_m} A_{xy}$, $n_l = |C_l|$ is the number of clients in cluster k and $s_m = |S_m|$ is the number of servers in cluster m. Some of the calculations needed to derive this formula can be found in Appendix A.

The similarity measure in (4.4) used to perform agglomerative clustering of clients (rows) can be extended to servers (columns) as

$$S_{km}^{(Y)} = \frac{\mathbb{P}(\mathbb{C},\mathbb{S}^{km}|A)}{\mathbb{P}(\mathbb{C},\mathbb{S}|A)}. \tag{4.7}$$

However, simultaneous agglomerative clustering of rows (clients) and columns (servers) of the data matrix requires the value of the similarity measure $S_{km}^{(Y)}$ to be recalculated for all pairs of column clusters after a merge of two row clusters, and vice versa for mergers of column clusters. Thus, for a $|X| \times |Y|$ matrix, the computational complexity of the algorithm is $\mathcal{O}(|X|^2(1+|Y|^2) + |Y|^2(1+|X|^2))$. This computational load is not really affordable for large data matrices such as those from computer network data. Therefore, we investigate a different approach, based on iteratively clustering the rows and the columns separately and considering the impact each row cluster configuration found at the previous iteration step has on the column clustering and vice versa, the impact each column cluster configuration has on the row clustering. Agglomerative client clustering and server clustering are carried out iteratively until the same clustering configurations are found at two subsequent iteration steps. In this way, for client clustering, the computational complexity of the algorithm is $\mathcal{O}(|X|^2)$ and similarly for server clustering, the computational complexity is $\mathcal{O}(|Y|^2)$.

4.1.6 Biclustering algorithm

The algorithm is initiated with $\mathbb{C} = \{\{1\}, ..., \{|X|\}\}$ and $\mathbb{S} = \{\{1\}, ..., \{|Y|\}\}$, i.e. all entries of the data matrix are placed in $|X||Y|$ singleton biclusters. At each step, the algorithm iterates between applying Algorithm 2 to $S_{km}^{(X)}$ for client clustering and $S_{km}^{(\mathcal{C})}$ for server clustering, until convergence, i.e. the same clustering configurations are found at two iteration steps. Convergence is guaranteed if every time we switch from client to server clustering, or vice versa from server to client clustering, agglomerative clustering leads to a model that has the same or higher marginal likelihood than the one obtained at the previous iteration. A detailed algorithm is given in Algorithm 3. Note that the procedure is described assuming the client set is clustered first but in principle, the same algorithm can be applied by clustering the server set first.

4.1.7 An informative Beta prior

To aid the naive agglomerative clustering algorithm, an informative, empirical beta distribution was built for both the client and the server clustering steps. For ease of explanation, let us consider the case of client clustering, where the desired prior distribution was built for each server in the network with mean equal to the proportion of clients authenticated on that server computer. The rationale for using informative prior distributions is the following. Computer networks show highly skewed connectivity behaviour (as in Figure 2.2), where most of the nodes have few connections while some nodes are highly connected. For instance, the 2014 LANL computer network data contain three servers with over 9,000 unique client authentications. Therefore, observing client authentications on such high-degree servers may not be informative about the client profile, but might dominate the agglomerative clustering algorithm. Particularly for very high degree or very low degree nodes, a flat beta prior distribution, e.g. a beta distribution with parameters $a = b = 1$, would not be suitable.

Let p_y be the proportion of the population of clients that connect to server y.

Algorithm 3 Model-based Iterative Biclustering

INPUT=($|X| \times |Y|$ data matrix A)
Set iteration t=1. Set $\mathbb{C}(t{=}1)$=$\{\{1\},\ldots,\{|X|\}\}$, $\mathbb{S}(t{=}1)$=$\{\{1\},\ldots,\{|Y|\}\}$ and $\mathbb{I}(t{=}1)$=$\{(\mathbb{C}(1),\mathbb{S}(1))\}$
repeat
  procedure CLIENT-CLUSTERING(A)
    Calculate $\mathbb{P}(\mathbb{C}(t),\mathbb{S}(t)|A)$
    Calculate the similarity $S_{km}^{(X)}$ for each pair of client clusters k and m
    while $|\mathbb{C}(t)| > 1$ **do**
      Identify the cluster pairs (k^*, m^*) with largest similarity measure
      Merge row clusters k^* and m^* to form a new client cluster u^*
      Update $\mathbb{C}(t)$
      Recalculate $S_{ku^*}^{(X)}$ for each $k \neq u^*$
    Identify the optimal partition in the hierarchy: $\mathbb{C}(t)^* = \arg\max_{\mathbb{C}(t)} \mathbb{P}(\mathbb{C}(t),\mathbb{S}(t)|A)$
  Permute A based on $\mathbb{C}(t)^*$ and set $A = A^T$
  $\mathbb{C}(t) \leftarrow \mathbb{C}(t)^*$
  procedure SERVER-CLUSTERING(A)
    Calculate $\mathbb{P}(\mathbb{C}(t),\mathbb{S}(t)|A)$
    Calculate the similarity $S_{gl}^{(\mathbb{S})}$ for each pair of server clusters g and l
    while $|\mathbb{S}(t)| > 1$ **do**
      Identify the cluster pairs (g^*, l^*) with largest similarity measure
      Merge column clusters g^* and l^* to form a new server cluster c^*
      Update $\mathbb{S}(t)$
      Recalculate $S_{gc^*}^{(Y)}$ for each $g \neq c^*$
    Identify the optimal partition in the hierarchy: $\mathbb{S}(t)^* = \arg\max_{\mathbb{S}(t)} \mathbb{P}(\mathbb{C}(t),\mathbb{S}(t)|A)$
  Set $\mathbb{I}(t) = \left(\mathbb{C}^*(t),\mathbb{S}^*(t)\right)$
  Permute A based on $\mathbb{C}(t)^*$ and set $A = A^T$
  $t \leftarrow t+1$
  $\mathbb{S}(t) \leftarrow \mathbb{S}(t-1)^*$
until $\mathbb{I}(t) = \mathbb{I}(t-1)$ (same clustering configuration at iteration t-1 and t)
OUTPUT=(Optimal cluster configuration $\mathbb{I}^* = (\mathbb{C}^*,\mathbb{S}^*)$)

For each server y, the empirical beta distribution is specified with the following mean and variance:

$$\mu = \frac{a}{a+b} = p_y, \qquad \sigma^2 = \frac{ab}{(a+b)^2(a+b+1)} = \frac{p_y(1-p_y)}{|Y|}. \tag{4.8}$$

The corresponding a and b parameters of the beta distribution can be calculated by solving (4.8) with respect to a and b. The same rationale holds when clustering the server set, where observing servers receiving authenticated connections from high-degree clients may not be informative about the server profile. In this case, p_y in (4.8) can be replaced by the proportion of servers receiving authenticated connections from client x, denoted p_x.

4.2 Flat clustering methods

In contrast to hierarchical clustering, which creates a hierarchy of clusters, flat clustering methods create a flat partition of clusters without any explicit structure that relates the clusters to each other. In these partitional methods, data points are moved from a cluster to another until, conditional on some criterion, there are no further possible improvements. When the criterion used is the sum-of-square, iterative partitioning coincides with k-means clustering (MacQueen, 1967). k-means is the most traditionally used flat clustering algorithm, mainly due to its simplicity. However, a major drawback of this algorithm is that it cannot separate clusters which are non-linearly separable in the input space. A modified version of k-means, known as kernel k-means, has emerged to tackle this problem. In this reformulation, the data points are first transformed from input space to a new, possibly higher-dimensional space using a nonlinear function, and subsequently standard k-means clustering is performed in the new space. Kernel k-means is closely related to spectral algorithms (see Dhillon et al. (2004)), which can also overcome this problem by using graph cuts as objective functions for nonlinear data separation.

Spectral clustering (Von Luxburg, 2007; Zhang and Jordan, 2008) is a flexible

flat clustering approach that makes few assumptions on the shape of the clusters. This method has proved to outperform k-means, which may often not be able to find a globally optimal partition, since it relies on randomly chosen initial centers. In spectral clustering, the data are represented as a similarity graph where data points are nodes and the edge-weights are pairwise similarities between points. The partition algorithm finds a k-way graph cut by firstly finding a spectral embedding through an eigenvalue decomposition of a Laplacian data matrix, and then based on this embedding a partition is found via a simplified clustering algorithm such as k-means. A nice review of several algorithm variants has been provided by Weiss (1999). When the eigenvalue decomposition operates on the standard adjacency matrix, rather than one of its Laplacian, spectral clustering coincides with singular value decomposition (SVD). Thus, the two embedding are strongly related, although they may emphasise different aspects of the graph under analysis. Both SVD and spectral clustering perform dimensionality reduction, allowing the extraction of an optimal lower-dimensional description of the data matrix, yet preserving the principal signal of association between data points.

The extension of spectral clustering to biclustering problems has been mainly introduced by Kluger et al. (2003) for analysing microarray cancer data sets and by Dhillon et al. (2004) and Cho et al. (2004) for bipartitioning word-document data matrices. Biclustering through singular value decomposition and spectral clustering are described in the following subsections. Both methods use k-means as a final step for extracting clusters from the eigenvectors of the spectral decompositions.

4.2.1 Biclustering via singular value decomposition

Singular value decomposition (SVD) provides a convenient method for factorising a matrix. Given the previously introduced adjacency matrix A, the SVD of A can be written as

$$A = U\Sigma V^T = \sum_{k=1}^{r} s_k u_k {v_k}^T, \tag{4.9}$$

where r is the rank of A, $U = (u_1, \ldots, u_r)$ is a matrix of orthonormal left singular

vectors, $V = (v_1, \ldots, v_r)$ is a matrix of orthonormal right singular vectors and $\Sigma = \text{diag}(s_1, \ldots, s_r)$ is a diagonal matrix of positive singular values $s_1 \geq s_2 \geq \cdots \geq s_r$. The adjacency matrix A is therefore decomposed into a summation of rank-one matrices $s_k u_k {v_k}^T$, also called *SVD layers*.

As pointed out by Busygin et al. (2008), biclustering can be related to the SVD by considering an idealised block diagonal data matrix with blocks representing biclusters and *zero* elements outside the blocks:

$$A = \begin{pmatrix} A_1 & 0 & \ldots & 0 \\ 0 & A_2 & 0 & \ldots \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \ldots & A_r \end{pmatrix}.$$

After performing SVD on A, each block submatrix A_k, $k = 1, \ldots, r$, is associated with a singular vector pair (u_k, v_k), such that non-zero components of u_k correspond to rows occupied by A_k and non-zero components of v_k correspond to columns occupied by A_k. In this way, it is natural to associate each block with a bicluster. Although in real applications we do not have perfect block data matrices, the SVD can still detect the rows and columns of the sub-matrices as the dominating coefficients in the singular vector pair. This makes SVD a suitable biclustering tool.

Applications generally focus on SVD layers with large eigenvalues (signal) while the remaining layers can be discarded as non-informative noise. Therefore, if we just consider the first $K \leq r$ rank-one matrices in (4.9), we obtain a rank-K approximation (or truncated SVD) to A:

$$A \approx A^{(K)} = \sum_{k=1}^{K} s_k u_k {v_k}^T, \tag{4.10}$$

and the matrix $A^{(K)}$ gives the best rank-K matrix approximation to A with respect

to the squared Frobenius norm (Eckart and Young, 1936), i.e.

$$A^{(K)} = \arg\min_{A^* \in \mathcal{A}_\mathcal{K}} \|A - A^*\|_F^2 = \arg\min_{A^* \in \mathcal{A}_\mathcal{K}} \mathrm{tr}\{(A - A^*)(A - A^*)^T\}, \tag{4.11}$$

where $\mathcal{A}_\mathcal{K}$ is the set of all $|X| \times |Y|$ matrices of rank K and $\|\cdot\|_F^2$ indicates the squared Frobenius norm. In this case, the decomposition gives an embedding of the nodes of the graph as vectors in a lower dimensional space.

Neglecting all but the first K components can be justified when the noise in the data perturbs the small eigenvalues, whereas the first K components capture the signal of the data. For this reason, selecting the threshold value K represents a central problem of truncated SVD. Furthermore, in the clustering setting choosing the value of the parameter K corresponds to choosing the number of clusters. A standard criterion for choosing K is to look for where the last large gap or elbow appears in a plot of singular values (Hoff, 2007), also referred to as *scree* plot, and subsequently applying k-means to extract clusters from the eigenvectors.

In a bipartite computer network graph, U represents a low-rank matrix of IP addresses of clients in latent space, V represents a low-rank matrix of IP addresses of servers in a latent space, and the best K-dimensional representation of X through U and V is given by taking the first K singular values from the SVD. This is illustrated in a cartoon in Figure 4.1. Biclustering is obtained by extracting clusters via k-means from both the eigenvectors of U and the eigenvectors of V. In k-means clustering, we are given a set of n data points in d-dimensional space $\mathbb{R}^d$ and an integer k, representing the number of clusters fixed a priori. The algorithm aims to partition the n points into a set of k points in $\mathbb{R}^d$, called *centers*, which minimise the mean squared distance from each data point to its nearest center. Here, we aim to partition the $|X|$ clients and the $|Y|$ servers in a K-dimensional space into $k = K$ clusters, where K is the rank of U and V obtained via the truncated SVD decomposition in (4.10).

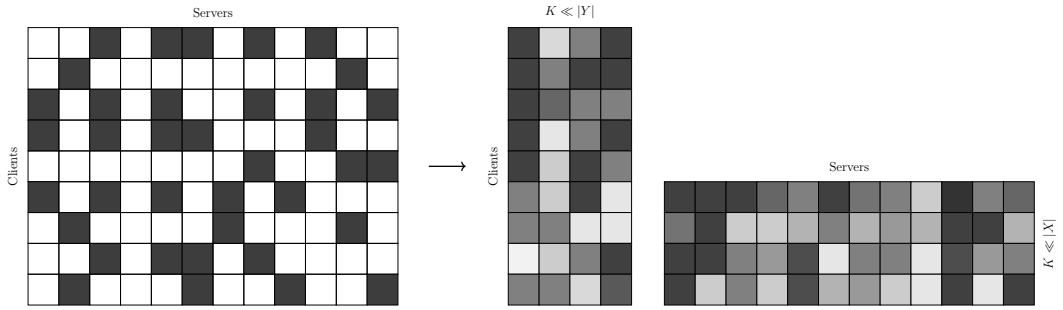


Figure 4.1: A cartoon describing how to transform the adjacency matrix of a computer network graph into lower dimensional representations for clients and servers.

4.2.2 Spectral biclustering

Spectral clustering approaches are closely related to singular value decomposition as they also use the information contained in the eigenvectors and eigenvalues of a suitable matrix representation of a graph to perform dimensionality reduction before clustering in fewer dimensions. The spectral biclustering embedding is similar to the one used in SVD presented in the previous section, but rather than operating directly on the adjacency matrix it operates on the Laplacian transformation.

Specifically, spectral biclustering uses SVD on the bipartite graph Laplacian matrix $L = D - A$, where $D = (D_{xy})$ is a diagonal degree matrix with $D_{xx} = \sum_m A_{xm}$. Spectral approaches operate on L or normalised variants of L to partition the graph recursively. An optimal bisection is found at each step with respect to an optimisation function based on Laplacian eigenvectors (Pothen et al., 1990). A detailed description of spectral clustering is provided by Von Luxburg (2007), while a commonly used extension to biclustering can be found in Dhillon (2001) and (Cho et al., 2004). Let D_X and D_Y be diagonal matrices of the row and column sums of A, respectively equal to the outdegrees of clients and indegrees of servers. Here, the spectral biclustering algorithm calculates a truncated-singular value decomposition of the normalised Laplacian

$$D_X^{-1/2} A D_Y^{-1/2}. \tag{4.12}$$

In this context, the left singular vectors of (4.12) correspond to the clients projected in a K-dimensional latent space, and similarly the right singular vectors correspond to K-dimensional latent positions for the servers. Once again, performing k-means on these latent positions yields cluster configurations of the clients and the servers.

4.3 Simulation study

4.3.1 Simulated data

We simulated 100 cluster configurations with the purpose of evaluating the effectiveness of the methods presented above. Three different data structures, which are known to potentially influence the algorithms, have been generated and analysed. The first data structure takes the form of a 100×100 square data matrix, the second is a 1000×100 matrix, while the third is a 1000×100 matrix. Examples of the three generated data matrix can be found in Figure 4.3. The structure determines the size of the data matrix and the form of the clusters. The number of clusters and their sizes were generated uniformly at random. As previously mentioned, computer networks have very skewed degree distributions which should be reflected in the simulated data: for high degree server groups, client clusters need to be created such that most client clusters have a high probability of connection, but a small proportion have a very low probability of connection. Similarly, for low degree server groups, client clusters need to be created such that most client clusters have a low probability of connection, but a small proportion have a very high probability of connection. This can be achieved by drawing the bicluster-specific parameters $\boldsymbol{\theta}$ from beta distributions with 'extreme' parameter values. Specifically, we choose a beta with parameters $\alpha = 10$ and $\beta = 1$ for the high degree servers and $\alpha = 1$ and $\beta = 10$ for the low degree servers. Figure 4.2 shows the beta probability density functions on varying the parameter values.

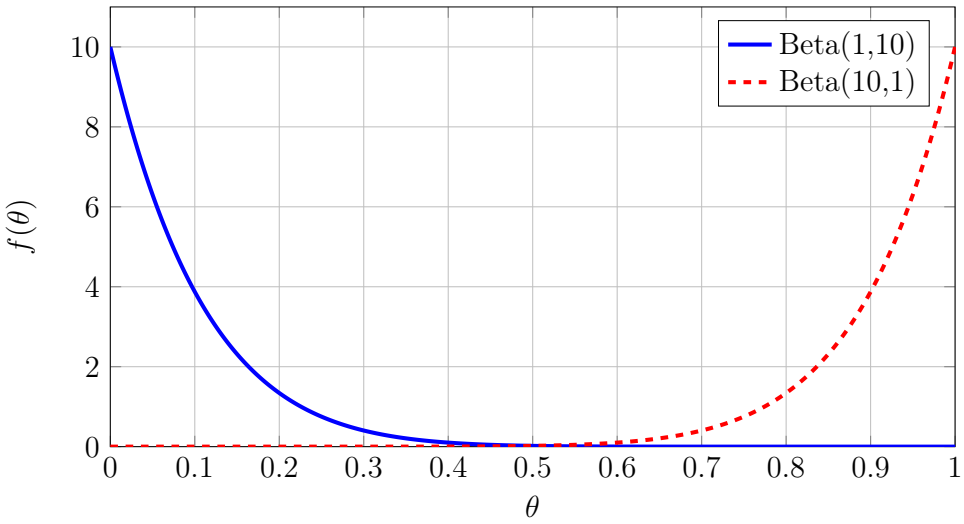


Figure 4.2: Probability density function of beta distributions with parameters $\alpha = 10$, $\beta = 1$ for the high degree servers and $\alpha = 1$, $\beta = 10$ for the low degree servers.

4.3.2 Results

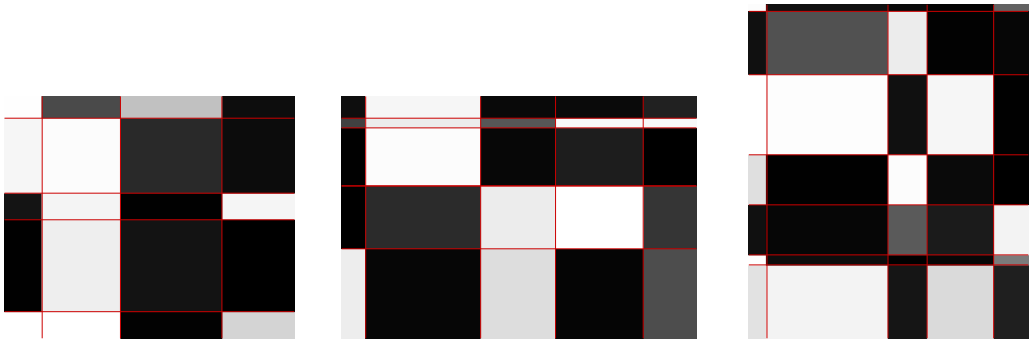
Simulated data are used to compare the model-based agglomerative biclustering algorithms with spectral and singular value decomposition biclustering approaches. Several scores are calculated to evaluate the cluster configurations returned by the algorithms against the true configurations. In particular, we use the three following scores, which compare a known target bicluster, A, with a retrieved bicluster, B (Turner et al., 2005):

$$\text{Sensitivity} = \frac{g_{A\cap B}}{g_B} \times \frac{s_{A\cap B}}{s_B}, \tag{4.13}$$

$$\text{Specificity} = \frac{g_{A\cap B}}{g_A} \times \frac{s_{A\cap B}}{s_A}, \tag{4.14}$$

$$F_1 \text{ measure} = \frac{2(g_{A\cap B})(s_{A\cap B})}{n_A + n_B}, \tag{4.15}$$

where g_C is the number of rows in cluster C, s_C is the number of columns in cluster C and $n_C = g_C s_C$ is the number of data points corresponding to C.



(a) 100×100 data matrix (b) 100×1000 data matrix (c) 1000×100 data matrix

Figure 4.3: Simulated data matrix heatmaps, for three different matrix sizes, where the grayscale corresponds to different cluster parameters.

Sensitivity measures the proportion of the identified cluster B which is contained in the target cluster A, while specificity measures the proportion of A that has been retrieved in B. The F_1 measure (Strehl, 2002) is the harmonic mean of the sensitivity and specificity, giving an overall measure of cluster quality. For each identified cluster, the cluster with the highest mean score across all three measures is assumed to be the known cluster and then a score for the whole configuration quality is obtained by calculating a weighted mean value, with weights given by the size of the identified clusters. Finally, a zero score is given in case a method detects no clusters.

The measures discussed so far do not evaluate the fact that agglomerative clustering produces a nested sequence of clusters. This aspect is evaluated using the so-called dendrogram purity score (Heller and Ghahramani, 2005), which measure how well a dendrogram clusters the known labels. This measure is calculated by randomly selecting two observations belonging to the same cluster in the true configuration and then finding the first cluster in the hierarchy which contains both the observations. Finally, the fraction of observations in this cluster which are also in the true cluster is calculated: the expected value of this proportion is the dendrogram purity. Note that the purity is 1 if and only if all observations in each group are contained in some pure cluster.

For agglomerative model-based iterative biclustering, informative beta prior distributions were used for the probability parameter $\boldsymbol{\theta}$. In particular, an empiri-

cal beta prior was built for each server when clustering the client set and for each client when clustering the server set, as described in Section 4.1.7, and the performance of the algorithm was compared with respect to a flat beta prior choice, i.e. choosing $a = b = 1$ in (4.2). Table 4.1 shows the mean score values from applying each clustering method to the 100 simulated cluster configurations, under the three different data matrix sizes. The naive model-based algorithm with flat beta priors showed poor performance, confirming a flat prior not to be suitable in this context. Simultaneous model-based agglomerative biclustering, where the similarity measure is recalculated after each row or column merge, was also used to provide a comparison with the iterative version proposed. This was possible due to the relatively small size of the simulated matrices analysed.

Changing the data structure has an effect on the performance of the algorithms but results show a good overall performance. In particular, model-based biclustering algorithms perform well, mainly correctly identifying the true cluster configuration in all the three data structures, as demonstrated by the high F_1 values and high purity. Spectral and SVD approaches are able to identify some clusters correctly, but the performance scores are always lower than the scores obtained by using agglomerative model-based algorithms. In particular, they both have a lower sensitivity than specificity score which indicates that although the clusters identified may contain the true clusters, they also contain other elements. Purity scores are not available for those two flat clustering methods, as no cluster hierarchy is created. SVD biclustering always achieves higher scores than spectral biclustering. Furthermore, the performance of the spectral methods is poorer when using large, but square data matrices, whereas agglomerative model-based algorithms achieve better results. Specifically, agglomerative model-based iterative biclustering always converged in less than ten iterations and performs well also in the case of highly imbalanced dimensionality, i.e. when $|X| \ll |Y|$. Since the number of rows and columns is not equal, there may be an initial algorithmic preference towards one type of merge. Furthermore, the algorithm can suffer from a known property of agglomerative clustering, i.e. a strong tendency for clusters included in the most recent merge to again be included in the next (Heard,

2011). As expected, the mean purity for the hierarchies created under model-based simultaneous biclustering is slightly higher than the mean purity score for the hierarchies created using the modified iterative form. However, this comes at the cost of higher computational load, which would not be feasible when using larger real computer network data matrices.

Finally, Figure 4.4 shows the algorithm runtimes to evaluate computational efficiency at varying the matrix dimension and dimensionality imbalance. Each slope line indicates the order of the runtime polynomial. As expected, model-based simultaneous biclustering has higher runtimes, since for each iteration, the similarity measure needs to be recalculated for all pairs of column clusters after a merge of two row clusters, and vice versa for column clusters mergers. Our modified version, ie. model-based iterative biclustering, scales better, potentially making its usage more feasible in the context of large-scale systems.

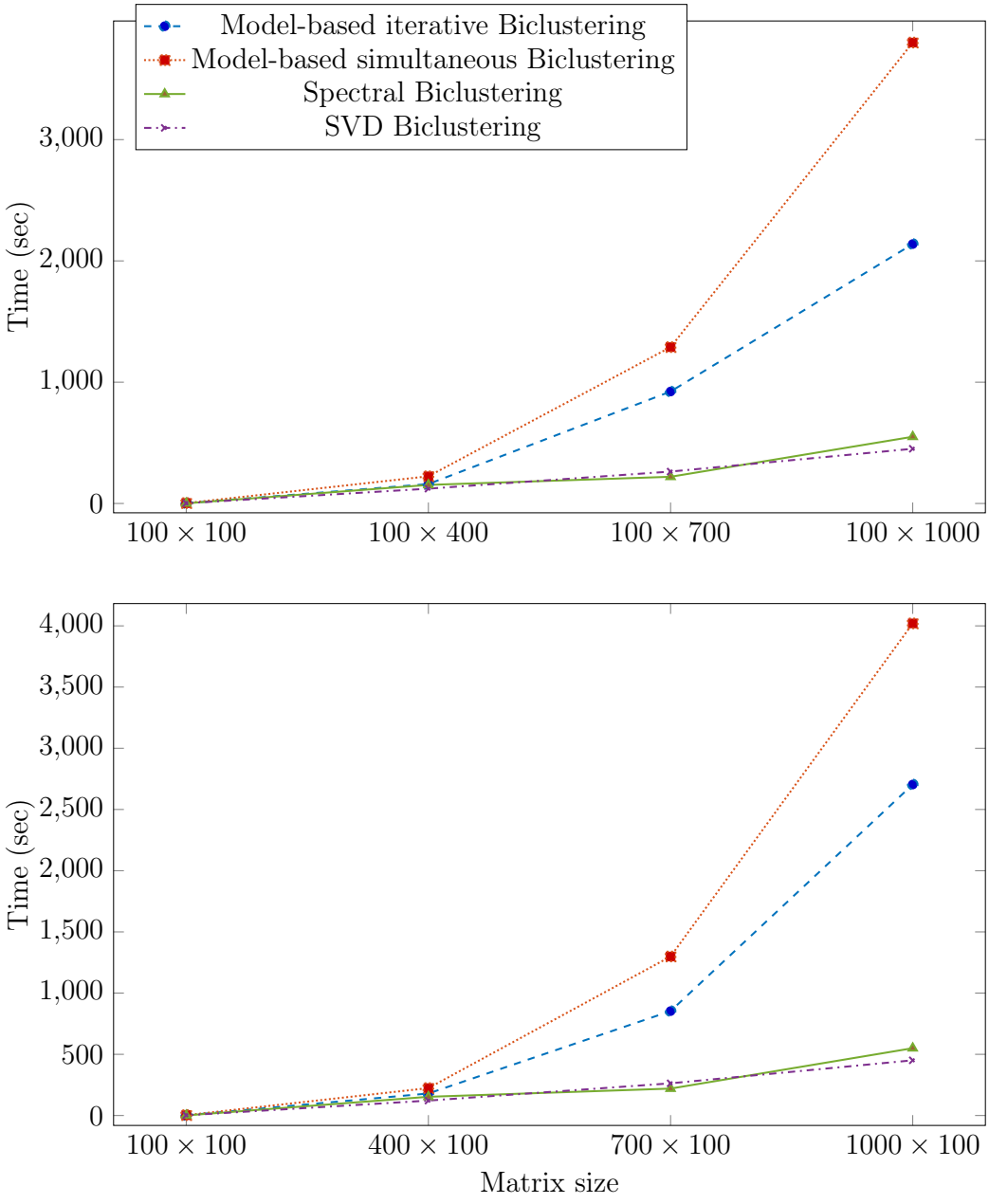


Figure 4.4: Algorithm runtimes (seconds) vs. data matrix dimension.

Measure	Method	Data Structure		
		$\lvert\mathcal{U}\rvert \times \lvert\mathcal{C}\rvert = 100 \times 100$	$\lvert\mathcal{U}\rvert \times \lvert\mathcal{C}\rvert = 1000 \times 100$	$\lvert\mathcal{U}\rvert \times \lvert\mathcal{C}\rvert = 100 \times 1000$
Sensitivity	Model-based Iterative Biclustering (flat prior)	0.581	0.382	0.579
	Model-based Iterative Biclustering (informative prior)	0.861	0.882	0.929
	Model-based Simultaneous Biclustering	0.922	0.961	0.938
	Spectral Biclustering	0.674	0.542	0.553
	SVD Biclustering	0.699	0.700	0.721
Specificity	Model-based Iterative Biclustering (flat prior)	0.384	0.261	0.413
	Model-based Iterative Biclustering (informative prior)	0.884	0.911	0.919
	Model-based Simultaneous Biclustering	0.916	0.957	0.951
	Spectral Biclustering	0.593	0.507	0.624
	SVD Biclustering	0.702	0.691	0.721
F_1 Measure	Model-based Iterative Biclustering (flat prior)	0.483	0.322	0.478
	Model-based Iterative Biclustering (informative prior)	0.884	0.908	**0.934**
	Model-based Simultaneous Biclustering	0.908	0.962	0.958
	Spectral Biclustering	0.633	0.537	0.584
	SVD Biclustering	0.689	0.720	0.753
Purity	Model-based Iterative Biclustering (flat prior)	0.551	0.546	0.507
	Model-based Iterative Biclustering (informative prior)	0.884	0.911	0.919
	Model-based Simultaneous Biclustering	0.857	0.875	0.908
	Spectral Biclustering	-	-	-
	SVD Biclustering	-	-	-

Table 4.1: Quality measures for each algorithm over simulated data, for different data matrix sizes.

Chapter 5

A client model for the identity of new edges

While the work on new edges presented in Chapter 3 was focused on modelling the rate of occurrence, in this chapter we seek to model and predict the identity of the new edges formed by each client. Specifically, we need a reliable and scalable model to monitor the characteristics of each new edge observed, on the basis of past behaviour of each network host. This could allow us to categorise new edges according to the different types of observed behaviour. As described in Chapter 4, computer networks tend to exhibit an underlying cluster structure, where nodes are naturally grouped together based on similar connection patterns. What constitutes normal behaviour might strongly differ between network hosts, and so inferring these peer groups constitutes an important step in modelling the types of new connections a client would make. If we observe a client forming a new edge, the question here is whether there are other similar clients which also share that connection. Other than revealing hidden network structure, such information about shared connectivity can be predictive of similar future interactions, thus aiding the problem of new edge prediction.

Towards this end, a notion of similarity is here built through a clustering model which partitions the client set into groups of clients sharing a similar connection

behaviour. This could give us a first indication of whether introducing information about the clustered network structure in the model can aid its predictive performance. This similarity measure will be further extended in Chapter 6 for measuring if *similar* clients may be more likely to connect to *similar* servers, which will be thus seen as a biclustering problem.

The method presented in this chapter is based on a sequential two-step Bayesian inference procedure to infer client cluster configurations while simultaneously modelling new edge formation. A Bayesian agglomerative model-based clustering algorithm is used as a first step of the analysis, with the purpose of identifying an initial, reliable cluster configuration of clients sharing similar connection behaviour. Subsequently, the identity of new edges is modelled through a Bayesian Cox proportional hazards model, where the initial cluster configuration and the model coefficient parameters are jointly updated with MCMC, to improve the predictive performance of the initial configurations. The procedure is then repeated sequentially following a linear updating scheme, as new data arrive. Parts of the work presented here can also be found in Metelli and Heard (2016).

The remainder of this chapter is organised as follows: Section 5.1 introduces a Bayesian proportional hazards model for new edges and describes the surrogate clustering model used to infer initial cluster configurations. Posterior inference is described in Section 5.2 while the main results are shown and discussed in Section 5.3.

5.1 Bayesian proportional hazards client model for new edges

From the start of an observation period of a computer network, new connections from clients to server computers are observed, each adding a new edge to the bipartite graph. Let $(t'_1, (x'_1, y'_1)), \ldots, (t'_n, (x'_n, y'_n))$ be the realised time-ordered sequence of new (client, server) edges observed in the network graph. We model new edge formation over time as a Cox proportional hazards model (Cox, 1972)

with time-dependent covariates. This model conveniently integrates the benefits of non-parametric and parametric approaches to statistical inference by dividing the hazard into a non-parametric part, namely the baseline hazard and a parametric part, namely the covariate effects.

The hazard for observing a new connection between a particular client x and server y at time t is modelled as the product of a baseline hazard and the exponential of a linear combination of the covariates. In this context, one relevant time-varying covariate is the indegree of the server, which provides a measure of 'popularity' for each server. The population of servers in an enterprise computer network typically has a heavily right-skewed degree distribution, with a small number of servers having a very high indegree, meaning they are connected to by most clients. In particular, we include two different degree effects: one representing the indegree of server y measured on the overall network graph at time t, denoted $N_y^-(t)$, and one representing the indegree of servers amongst a subset C of clients considered to be similar to the client x in question, denoted $N_{C,y}^-(t)$. These two variables have been already defined in (2.8) and (2.10). The idea is that if many of the clients with similar connection patterns to client x also connect to server y, then perhaps it is more likely that x will form a connection with y. This allows us to account for cluster-specific effects, which could be fundamental to improving the predictive ability of the model. As previously detailed in Section 4.1.2, clusters are defined as subsets of X and a cluster configuration $\mathbb{C} = \{C_1, \ldots, C_K\}$ is a partition of X into such subsets. Note that we only focus on clustering the client set for computational simplicity, while in the following chapter we will account for both client and server clustering.

In a slight abuse of notation, let $\mathbb{C}(x) \in \mathbb{C}$ be the unique cluster containing client x. Following the notation used in (2.10), we define the cluster-specific degree covariate for server y as

$$N_{y|\mathbb{C}(x)}^-(t) = \sum_{n \geq 1} \mathbb{1}_{[0,t)}(t'_n)\mathbb{1}_y(y'_n)\mathbb{1}_{\mathbb{C}(x)}(x'_n). \tag{5.1}$$

Note that $N_y^-(t) \equiv N_{X,y}^-(t)$. As we are interested in model the identity new edges for each client x, let $r_x(t)$ be the non-negative baseline intensity of new edge formation for client x. Then, for a client $x \in X$ and a server $y \in Y$, the hazard function takes the following form

$$\lambda_{xy}(t) = r_x(t) \exp\{\alpha N_y^-(t) + \beta N_{y|\mathbb{C}(x)}^-(t)\} \times \mathbb{1}_{(X\times Y)\setminus G_t}\{(x,y)\}. \tag{5.2}$$

The baseline hazard $r_x(t)$ is here treated as a nuisance parameter for each client, which does not affect model inference. Whilst this assumption will not fully hold in practice, it provides a vital inferential simplification as proposed in Cox (1972). Note that in this setting, treating $r_x(t)$ as a nuisance implies that we focus on a client-level analysis, with anomaly detection defined relative to the individual level, rather than the entire network graph. The latter case will be considered later in Chapter 6.

5.1.1 Conditional Bayesian likelihood-based inference

We condition on the event times $\mathcal{T}'$ of $(\mathcal{T}', \mathcal{E}')$ and work with the conditional likelihood of the event marks $\mathcal{E}'|\mathcal{T}'$. The conditional likelihood is here calculated sequentially as the product of predictive probabilities for the identities of each new edge, given the corresponding event time and the previous edges formed so far.

Given the sequence of time-ordered edges $(t'_1, e'_1), \ldots, (t'_{n-1}, e'_{n-1})$, the predictive distribution for the nth new edge is given by

$$\begin{aligned}\mathbb{P}_{E'_n}\{(x,y)|(t'_1,e'_1),\ldots,(t'_{n-1},e'_{n-1}),t'_n\} &= \frac{\lambda_{xy}(t'_n)}{\lambda(t'_n)} \\ &= \frac{\exp\{\alpha N_y^-(t'_n) + \beta N_{y|\mathbb{C}(x)}^-(t'_n)\}}{\sum_{(x',y')\notin G_{t'_n}} \exp\{\alpha N_{y'}^-(t'_n) + \beta N_{y'|\mathbb{C}(x')}^-(t'_n)\}}.\end{aligned} \tag{5.3}$$

Then, after observing n time-ordered edges $(t'_1, e'_1), \ldots, (t'_n, e'_n)$, the conditional

likelihood is simply given by the product of these predictive probabilities,

$$\mathbb{P}(\mathcal{E}'|\mathcal{T}', \alpha, \beta, \mathbb{C}) = \prod_{i=1}^{n} \mathbb{P}_{E'_i}\{(x'_i, y'_i)|(t'_1, e'_1), \ldots, (t'_{i-1}, e'_{i-1}), t'_i\}$$
$$= \prod_{i=1}^{n} \frac{\exp\{\alpha N^-_{y'_i}(t'_i) + \beta N^-_{y'_i|\mathbb{C}(x_i)}(t'_i)\}}{\sum_{(x',y')\notin G_{t'_i}} \exp\{\alpha N^-_{y'}(t'_i) + \beta N^-_{y'|\mathbb{C}(x')}(t'_i)\}}. \quad (5.4)$$

Here, $\alpha \in \mathbb{R}$, $\beta = (\beta_1, \ldots, \beta_K) \in \mathbb{R}^K$ and K is the total number of clusters in $\mathbb{C}$. This so-called 'partial' likelihood allows for estimation of covariates of the model without any restrictions placed on the baseline hazard.

A Bayesian formulation of the model requires the specification of prior distributions, in order to obtain the joint posterior distribution $\mathbb{P}(\alpha, \beta, \mathbb{C}|\mathcal{T}', \mathcal{E}')$ of the Cox model parameters and the cluster configuration. We choose standard normal distributions for the parameters α and β and we assume exchangeability when specifying a prior for the cluster configuration $\mathbb{C}$ so that a priori no two observations are more likely to belong to the same cluster. Specifically, we use a uniform distribution over the space of all possible cluster configurations. Alternatively, a Dirichlet process prior could be employed (Kim et al., 2006). Given the model likelihood provided in Eq. (5.4) and prior distributions $\mathbb{P}(\alpha)$, $\mathbb{P}(\beta|\mathbb{C})$ and $\mathbb{P}(\mathbb{C})$ for α, β and $\mathbb{C}$, the joint posterior distribution is then given by

$$\mathbb{P}(\alpha, \beta, \mathbb{C}|\mathcal{T}', \mathcal{E}') \propto \mathbb{P}(\mathcal{T}'|\mathcal{E}', \alpha, \beta, \mathbb{C})\mathbb{P}(\beta|\mathbb{C})\mathbb{P}(\mathbb{C})\mathbb{P}(\alpha). \quad (5.5)$$

As for most Bayesian models, exact inference is intractable, so Markov Chain Monte Carlo (MCMC) is required to perform posterior inference. The MCMC sampling scheme used is introduced in Section 5.2, while Bayesian model-based agglomerative clustering, as previously described in Section 4.1.1, is used as a surrogate clustering model to infer some initial cluster configurations. This gives us a simpler model that nonetheless provides an initial cluster configuration for (5.5) which can then be further updated according to the proportional hazards model, as described below.

5.2 Sequential two-step inference

The main inference procedure for the Cox model posterior distribution in (5.5) consists of two steps, embedded in a sequential updating scheme for computational tractability. The entire data set is divided into segments of equal length and the following procedure is repeated sequentially as new data arrive. Firstly, we perform agglomerative clustering (AC) as described in Section 4.1.2, and then we use MCMC sampling to simultaneously update the cluster configuration $\mathbb{C}$ and the parameters of the Cox model presented above.

An initial cluster configuration of the first segment of data is obtained via AC; subsequently, this configuration is updated via MCMC, jointly with Cox model parameters. The new cluster configuration is passed back to the AC step and a new data segment is added, where each data point is processed sequentially as described below; this results in an extended configuration which is then updated through MCMC. The procedure is repeated until the all data have been processed.

This two-step updating scheme provides a way for obtaining a good starting cluster configuration for the MCMC sampler. Although the simultaneous sampling of Cox model parameters and cluster configurations is fundamental for improving clusters predictive performance, MCMC algorithms can strongly depend on the initial values assigned to the chain. The agglomerative clustering step is therefore introduced for providing a deterministic initial cluster configuration, to achieve subsequent MCMC convergence quickly. This proves to be necessary because of the size of the clustering problem considered, where assigning random initial configurations could irreparably affect the convergence of the chain. The two different inference steps are described below.

5.2.1 Agglomerative clustering step

Agglomerative clustering under the surrogate model described in Section 4.1.1 is used to infer an initial cluster configuration. Although this algorithm generally provides a fast and efficient deterministic procedure for small data sets, large com-

puter networks carry a higher computational burden. We achieve computational saving by performing agglomerative clustering on a small initial segment of the entire data set, and then sequentially processing the remaining data. Each subsequent data point is added one at a time, initially as a new singleton cluster k' and then a possible merger with each previously created cluster k is evaluated on the basis of the similarity measure $S_{kk'}$ in (4.4). In this way, re-evaluation of all pairwise similarities is not required, providing a significant improvement in terms of computational speed.

5.2.2 MCMC step

The Metropolis-Hastings (M-H) algorithm, which has been described in Section 3.1.1, is used to draw approximate samples from the joint posterior distribution in (5.5) of the Cox model parameters α, β and the cluster configuration $\mathbb{C}$. The scheme used to sample Cox model parameters and cluster configurations is briefly presented below.

Let α^t, β^t and $\mathbb{C}^t$ be the values of the parameters α, β and cluster memberships in $\mathbb{C}$, at stage $t+1$ of the algorithm, with probability 0.5 a new value β_i^* is proposed for a uniformly sampled component i, conditional on the current value β_i^t, from a normal distribution $N(\beta_i^t, \sigma^2)$, with scaling parameter σ. Let β^* be equal to β_i^* at position i, and equal to β^t everywhere else. By symmetry of the normal proposal density, the proposed vector β^* is accepted with probability

$$\min\left(1, \frac{\mathbb{P}(\beta^*, \alpha^t, \mathbb{C}^t|\mathcal{T}', \mathcal{E}')}{\mathbb{P}(\beta^t, \alpha^t, \mathbb{C}^t|\mathcal{T}', \mathcal{E}')}\right). \tag{5.6}$$

Similarly for α, a new proposed value α^* is accepted with probability

$$\min\left(1, \frac{\mathbb{P}(\alpha^*, \beta^t, \mathbb{C}^t|\mathcal{T}', \mathcal{E}')}{\mathbb{P}(\alpha^t, \beta^t, \mathbb{C}^t|\mathcal{T}', \mathcal{E}')}\right). \tag{5.7}$$

For cluster allocations, with probability 0.5 we randomly choose a client $x \in X$ with current cluster label $\mathbb{C}^t(x)$, and propose a new cluster label $\mathbb{C}^*(x)$ from a

discrete uniform proposal distribution over the integer set $\{1, \ldots, K^t+1\}/\{\mathbb{C}^t(x)\}$, where K^t is the current number of client clusters in $\mathbb{C}^t$. The proposed value $\mathbb{C}^*(x)$ suggests a new cluster configuration $\mathbb{C}^*$ with K^* client clusters. If $|\mathbb{C}_{\mathbb{C}^t(x)}| = 1$ and $\mathbb{C}^*(x) \neq K^t + 1$, then $K^* = K^t - 1$; or else if $|\mathbb{C}_{\mathbb{C}^t(x)}| > 1$ and $\mathbb{C}^*(x) = K^t + 1$, then $K^* = K^t + 1$. In both of these cases, the dimension of the parameter β^t must change. Initially we set $\beta^t = \beta^*$, but if $K^* = K^t - 1$ then the $\mathbb{C}^t(x)$ component $\beta^*_{\mathbb{C}^t(x)}$ is deleted, and if $K^* = K^t + 1$ then a new component $\beta^*_{K^*}$ is proposed from the standard normal prior. Then taking, for example, the most common case where $K^* = K^t$, the resulting cluster configuration and parameter vector are accepted with probability

$$\min\left(1, \frac{\mathbb{P}(\alpha^t, \beta^t, \mathbb{C}^*|\mathcal{T}', \mathcal{E}')K^t}{\mathbb{P}(\alpha^t, \beta^t, \mathbb{C}^t|\mathcal{T}', \mathcal{E}')K^*}\right). \tag{5.8}$$

5.3 An application to LANL computer network authentication data

The procedure proposed in Section 5.2 was applied to the first 10,000 distinct network authentication events (Kent, 2014) of the 2014 LANL data sets, which has been previously described in Section 2.2.1. Segments of 1000 unique authentication events were added at each sequential updating step. Table 5.1 provides a summary of some basic graph statistical quantities of interest for the subset of data analysed, while Figure 5.1 shows a log-log plot of the indegree distribution of the server computers over those 10,000 events. The degree distribution appears to follow a power law, with the majority of the servers having only one authenticated client and only a few servers having high degree.

For agglomerative clustering, informative beta prior distributions were used for the probability parameter $\boldsymbol{\theta}$ as described in Section 4.1.7. For MCMC, the Metropolis-Hastings algorithm was used with a total number of iterations set to 5,000, after a burn-in period of size 1000.

Figure 5.2 shows the number of identified clusters during the MCMC run: the

Events	10,000
Clients	1272
Servers	1432
Min server degree	1
Mean server degree	1.86
Max server degree	93

Table 5.1: Summary statistics for the subset of data analysed.

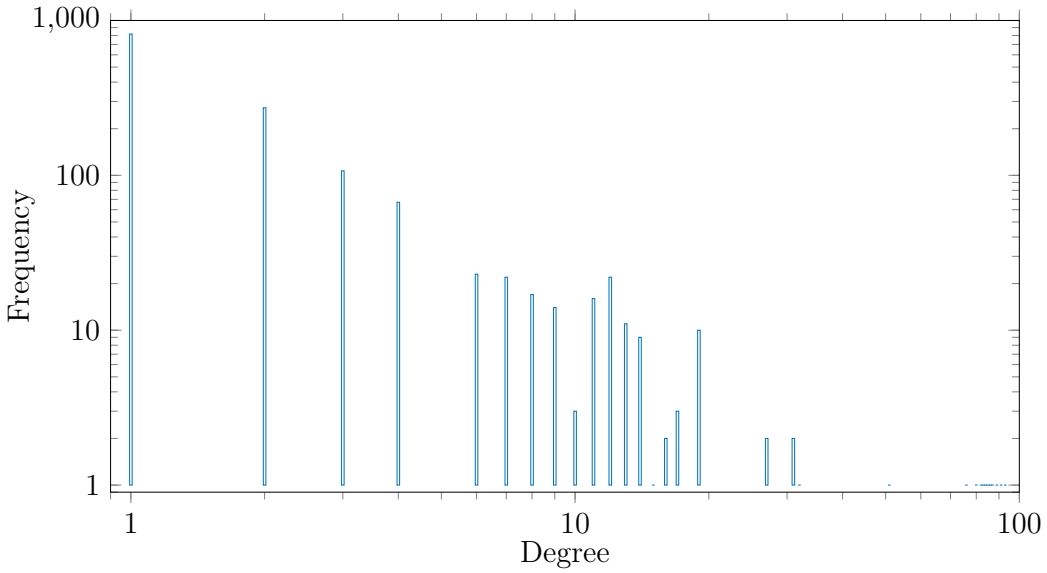


Figure 5.1: Frequency distribution of computer degrees (log-log scale).

most probable number of clusters is $K = 6$. Conditional on the most probable configuration with 6 clusters, Table 5.2 shows posterior means, standard deviations and 95% highest posterior density (HPD) credible intervals for the coefficients of the covariates, i.e. an overall degree effect α and cluster-specific degree effects β. The estimated coefficients are all positive, indicating that a high computer degree strengthens the probability of a new client establishing a connection to that computer, particularly when clients with similar connectivity patterns have also connected to that server. In addition, the effect is stronger for β_1, β_2 and β_3, which correspond to the largest clusters.

Coefficient	Mean	95 % HPD Credible Interval	
		Lower	Upper
α	1.0601	1.0556	1.0746
β_1	2.2532	2.1134	2.3987
β_2	2.0145	1.9586	2.0709
β_3	1.9753	1.8512	2.0802
β_4	0.9643	0.8923	1.0465
β_5	1.1112	1.0634	1.1534
β_6	0.4367	0.3754	0.5183

Table 5.2: Posterior model coefficient estimates with credible intervals.

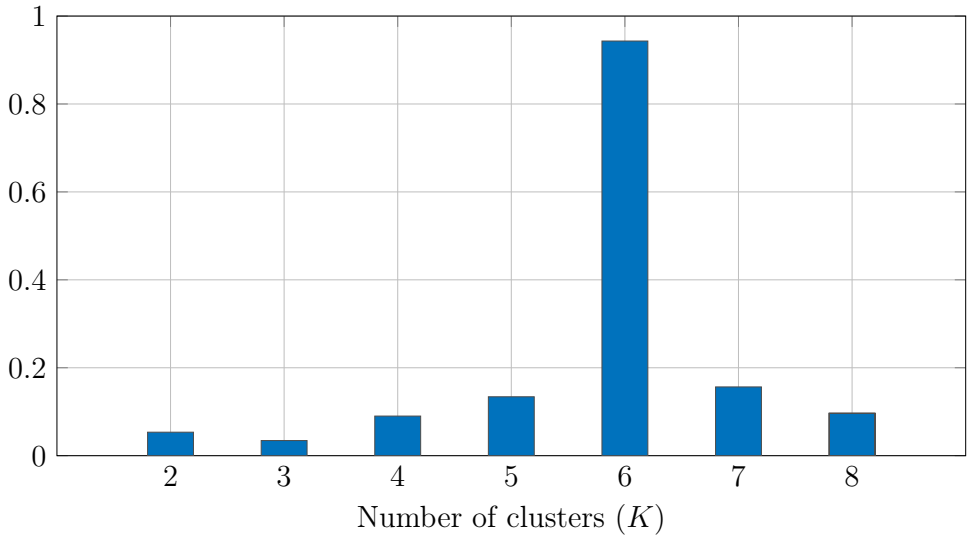


Figure 5.2: Posterior distribution of the number of identified client clusters.

The heat map in Figure 5.3 shows the clustered adjacency matrix, where the greyscale indicates different cluster allocations and the dendrogram reflects the hierarchical structure created through agglomerative clustering. Within the six client clusters identified, there are two dominant groups corresponding to the clusters accounting for the highest posterior coefficient effects.

Furthermore, the appropriateness of the probability model used is investigated through an analysis of the model's predictive performance. Assessing the plausibility of a posited model is an integral part of any statistical analysis and especially

fundamental in the Bayesian framework, where prior knowledge is included in the model.

Goodness-of-fit A principled way to evaluate a model is to analyse its out-of-sample predictive performance. The validation process involves comparison of the goodness-of-fit of models with a different number of covariates included. In particular, we analyse the predictive performance of three different models on 1,000 out-of-sample authentication events from the LANL data set. Table 5.3 shows a comparison of the three following models:

- Model 1: $\lambda_{xy}(t) = 1, \quad (\alpha = 0, \beta = 0)$
- Model 2 : $\lambda_{xy}(t) \propto \exp\{\alpha N_y^-(t)\}, \quad (\beta = 0)$
- Model 3: $\lambda_{xy}(t) \propto \exp\{\alpha N_y^-(t) + \beta N_{y|\mathbb{C}(x)}^-(t)\}$

The first model represents a null model where the probability of establishing a new connection is the same for each server. In the second model, the overall degrees of the servers are included while the third saturated model includes both overall and cluster-level effects. In this way, we are able to assess the contribution of including cluster information into our model. Furthermore, the analysis was repeated without updating the initial agglomerative clustering configuration jointly with Cox model parameters, in order to evaluate the improvement in the predictive performance of MCMC-based clusters. Comparisons were obtained by using the ratio of the averaged marginal likelihood, over the MCMC samples, for each pair of models. Note that the marginal likelihoods of Model 1 and Model 2 are each invariant to the cluster configuration.

The results show a significant improvement in the predictive performance of Model 3 with respect to both the null model and Model 2, suggesting there is a considerable contribution provided by the cluster-level covariates. Finally, the

Comparison	**LR - $\mathbb{C}$ not updated -**	**LR - $\mathbb{C}$ updated -**
Model 3 *vs* Model 1	589.03	607.56
Model 3 *vs* Model 2	2.76	3.46

Table 5.3: Likelihood ratios for the different models.

cluster-degree effect is stronger when cluster configurations are updated jointly with Cox model parameters, confirming an improved predictive performance of the MCMC-based clusters.

Finally, convergence diagnostic is used to assess the performance of an MCMC sampler by monitoring the convergence of the Markov chain to the stationary distribution. Here a plot of the marginal likelihood for each MCMC iteration is provided in Figure 5.4 in order to measure the goodness-of-fit of the saturated model. The process appears stationary as the number of iterations increases.

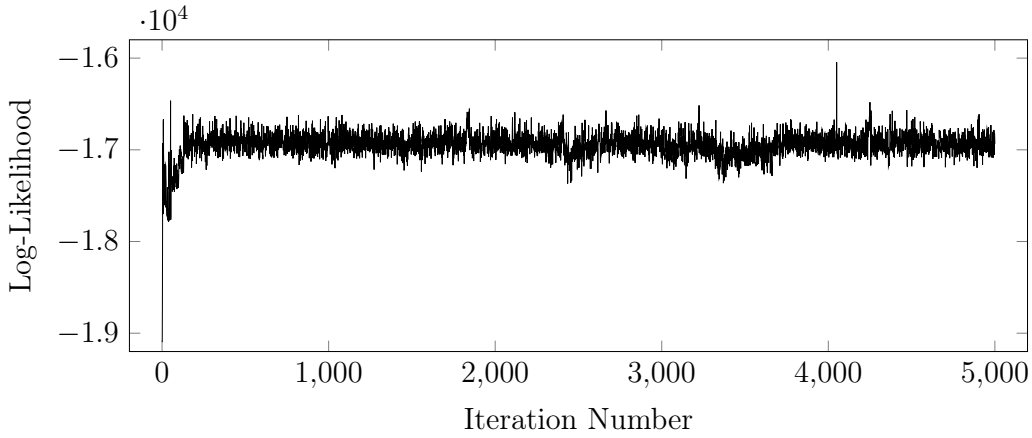


Figure 5.4: Log-likelihood vs. number of MCMC iterations, for the saturated model.

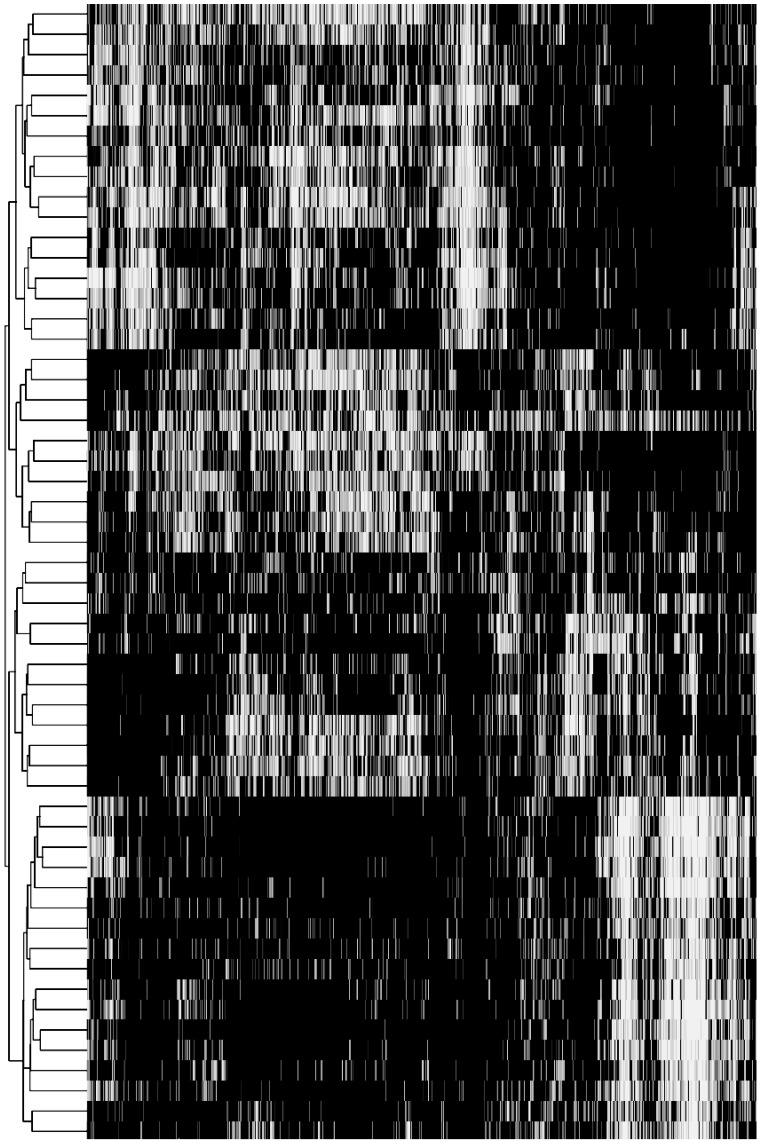


Figure 5.3: Adjacency data matrix heat map with cluster configuration identified. The greyscale represents different cluster allocation.

Chapter 6

An entire-network model for the intensity of arrival of new edges

The model introduced in the previous chapter has proved to have good predictive performance but does not allow to detect *which* client is responsible for causing anomalous behaviour, thus offering in principle a weaker detection power. For this reason, in this chapter we seek to extend the model taking a graph-level perspective which also takes into account the fact that different clients might make a very different number of new edges. This will offer a stronger detection power, with the potential to be able to detect anomalies in which client is forming new edges as well as which servers it is connecting to.

A robust model of new edge formation for the entire graph needs to capture the *rates* at which individual client hosts form new edges and then it must predict the *identity* of new edges formed by each client. After having devoted attention to separately modelling these two components in Chapter 3 and Chapter 5 respectively, in this chapter we propose a model which combines together these two aspects. Specifically, we extend the model proposed in Chapter 5 by constructing a richer, more robust semi-parametric Cox model for new edge intensity which simultaneously addresses the rates at which the client forms new edges and any underlying latent structural relationship between the clients and servers in the

network. In Chapter 5, we have seen the importance of introducing a notion of similarity between clients in the model: this is here extended to take into account that *similar* clients may be more likely to connect to *similar* servers. Two different formulations of this similarity measure will be proposed: first, we will encode cluster memberships under hard-thresholding similarly to Chapter 5 and then we will use dot-products of respective latent feature positions under soft-thresholding. Both models again allow us to examine whether shared connectivity can be predictive of similar future interactions. Part of the work presented here is taken from Metelli and Heard (2018).

The remainder of this chapter is organised as follows: Section 6.1 proposes a framework for modelling the intensity of arrival of new edges, with the two specific latent feature formulations outlined in Sections 6.2 and 6.3. Section 6.4 describes posterior inference under both formulations and finally, results from real computer network data are shown and discussed in Section 6.5.

6.1 A Bayesian Cox model for new edges for the entire network

Similarly to the model presented in (5.2), new edge formation over time is modelled as a Bayesian semi-parametric Cox model for the hazard function of each potential (client, server) edge. Here, the shape of the baseline rate will be determined by a function $r(t) \geq 0$ which is considered common to all network hosts and therefore a seasonal nuisance parameter whose functional form does not affect the model inference.

To incorporate the highly variable *popularity* of different client and server machines we now include both the time-varying outdegree of each client computer x, N_x^+, and the indegree of each server computer y, N_y^-, which have been respectively defined in (2.7) and (2.8). As already mentioned, the population of servers has typically a heavily right-skewed degree distribution and similarly, the outdegree distribution of clients can also follow a power law for large degree values, although

very small outdegrees are less common. These observations have been illustrated for real data in Figures 2.2 and 2.4.

Furthermore, in Chapter 3 we found the indicator variables $I_{x,1}$ (2.5) and $I_{x,2}$ (2.6) to be strongly significant predictors of the rate of occurrence of new edges in a computer network. These two variables, which indicate whether the last connection was new, or whether the last two connections were new, are therefore included in the model.

Finally, we propose a family of covariates $\{Z_{xy}(t)|(x,y) \in X \times Y, t \geq 0\}$ representing a general notion of *attraction* between clients and servers. This will be the quantity of central interest for the remainder of the chapter and two alternative formulations will be considered in the next two sections: a hard-threshold clustering model and a soft-threshold latent feature model.

Specifically, using the above described covariates, the intensity at time t for observing a new connection between a client $x \in X$ and a server $y \in Y$ is given by

$$\lambda_{xy}(t) = r(t) \exp\{\alpha \cdot (N_x^+(t), N_y^-(t), I_{x,1}(t), I_{x,2}(t)) + \beta_{xy} \cdot Z_{xy}(t)\} \times \mathbb{1}_{(X\times Y)\setminus G_t}\{(x,y)\}, \quad (6.1)$$

where $Z_{xy}(t) \in \mathbb{R}^k$ is a vector of length $k > 0$ quantifying the relative attraction of client x to server y at time t. Note that the intensity (6.1) becomes zero once the pair (x,y) have been observed. The coefficients $\alpha = (\alpha_1, \ldots, \alpha_4) \in \mathbb{R}^4$ and $\beta_{xy} \in \mathbb{R}^k$ for each $(x,y) \in X \times Y$.

The conditional intensity function for the counting process $\mathcal{T}'$ of new connections being made across the entire network is the double sum of (6.1) over X and Y,

$$\lambda(t) = r(t) \sum_{x\in X} \sum_{y\in Y} \exp\{\alpha \cdot (N_x^+(t), N_y^-(t), I_{x,1}(t), I_{x,2}(t)) + \beta_{xy} \cdot Z_{xy}(t)\} \times \mathbb{1}_{(X\times Y)\setminus G_t}\{(x,y)\}. \quad (6.2)$$

The covariates $N_x^+(t)$, $N_y^-(t)$, $I_{x,1}(t)$, $I_{x,2}(t)$ have already shown in the previous chapters to be informative about the rate of occurrence and the hazard of making new edges and so we will treat the corresponding coefficients α as nuisance parameters. The important question here is whether, having accounted for these intuitive covariates, we can find any underlying latent structure in the network which can provide further information about which (client,server) pairs might be more likely to connect. This aspect will be measured by the inferred magnitude of the coefficients $\beta = \{\beta_{xy}\}$ in (6.1).

6.1.1 Conditional likelihood-based Bayesian inference

Similarly to Chapter 5, we condition on the event times of $\mathcal{T}'$ of $(\mathcal{T}', \mathcal{E}')$ and so we work with the conditional likelihood of the event marks $\mathcal{E}'|\mathcal{T}'$. Given time-ordered edge sequence $(t'_1, e'_1), \ldots, (t'_{n-1}, e'_{n-1})$, the predictive distribution for the nth new edge is given by

$$\begin{aligned}
&\mathbb{P}_{E'_n}\{(x,y)|(t'_1,e'_1),\ldots,(t'_{n-1},e'_{n-1}),t'_n\} = \frac{\lambda_{xy}(t'_n)}{\lambda(t'_n)} \\
&= \frac{\exp\{\alpha \cdot (N_x^+(t'_n), N_y^-(t'_n), I_{x,1}(t'_n), I_{x,2}(t'_n)) + \beta_{xy} \cdot Z_{xy}(t'_n)\}}{\sum_{(x',y')\notin G_{t'_n}} \exp\{\alpha \cdot (N_{x'}^+(t'_n), N_{y'}^-(t'_n), I_{x',1}(t'_n), I_{x',2}(t'_n)) + \beta_{x'y'} \cdot Z_{x'y'}(t'_n)\}}. \quad (6.3)
\end{aligned}$$

Then, the conditional likelihood after observing n time-ordered edges $(t'_1, e'_1), \ldots, (t'_n, e'_n)$ is simply the product of the predictive probabilities in (6.3),

$$\begin{aligned}
\mathbb{P}(\mathcal{E}'|\mathcal{T}', \alpha, \beta, \{Z_{xy}\}) &= \prod_{i=1}^{n} \mathbb{P}_{E'_i}\{(x'_i, y'_i)|(t'_1,e'_1),\ldots,(t'_{i-1},e'_{i-1}),t'_i\} \\
&= \prod_{i=1}^{n} \frac{\exp\{\alpha \cdot (N_{x'_i}^+(t'_i), N_{y'_i}^-(t'_i), I_{x'_i,1}(t'_i), I_{x'_i,2}(t'_i)) + \beta_{x'_i y'_i} \cdot Z_{x'_i y'_i}(t'_i)\}}{\sum_{(x,y)\notin G_{t'_i}} \exp\{\alpha \cdot (N_x^+(t'_i), N_y^-(t'_i), I_{x,1}(t'_i), I_{x,2}(t'_i)) + \beta_{xy} \cdot Z_{xy}(t'_i)\}}. \quad (6.4)
\end{aligned}$$

Standard normal prior distributions are then chosen both for components of α and the free parameters of $\beta = \{\beta_{xy}|(x,y) \in X \times Y\}$. We now describe in Sections

6.2 and 6.3 the two proposed constructions for $\{Z_{x,y}|(x,y) \in X \times Y\}$ and their respective prior distributions.

6.2 Cluster formulation

The first proposal for constructing $Z_{xy}(t)$ is to build "peer group" clusters within both X and Y, based on similarity in connectivity patterns. This will allow us to account for bicluster effects and can be viewed as an extension of the covariate (5.1) proposed in Chapter 5, which accounted for peer group clusters within the client set X only.

Let $\mathbb{C} = \{C_1, \ldots, C_L\}$ be a partition of X, and $\mathbb{S} = \{S_1, \ldots, S_M\}$ a partition of Y and let $\mathbb{C}(x) \in \mathbb{C}$ be the unique cluster containing client x, and $\mathbb{S}(y) \in \mathbb{S}$ be the cluster containing server y. Recalling that $N^-_{y|\mathbb{C}(x)}(t)$ proposed in (5.1) represented the outdegrees of servers restricted to client cluster $\mathbb{C}(x)$, we define analogously the indegrees of clients restricted to subsets of servers as

$$N^+_{x|\mathbb{S}(y)}(t) = \sum_{n \geq 1} \mathbb{1}_{[0,t)}(t'_n)\mathbb{1}_x(x'_n)\mathbb{1}_{\mathbb{S}(y)}(y'_n), \tag{6.5}$$

which represents the number of servers in $\mathbb{S}(y)$ connected to by client x, prior to time t. Then, we can define the attraction covariate for the pair (x, y) as

$$Z_{xy}(t) = \left(N^+_{x|\mathbb{S}(y)}(t), N^-_{y|\mathbb{C}(x)}(t)\right). \tag{6.6}$$

For $L \geq 1$ client clusters and $M \geq 1$ server clusters, the specification (6.6) for $Z_{xy}(t)$ implies LM free parameters for $\beta = \{\beta_{xy}\}$, i.e. the effective dimensionality of β is reduced to the number of existing row and column clusters. These coefficients are assigned independent standard normal priors. Note that under the simpler model presented in Chapter 5, $Z_{xy}(t) \equiv N^-_{y|\mathbb{C}(x)}(t)$, and this scenario will later serve as a comparison for assessing the additional value of introducing server clusters when presenting results in Section 6.5. To complete a Bayesian model specification, the prior distributions for the clustering configurations $\mathbb{C}$ and $\mathbb{S}$ are

again assumed uniform over the space of all possible configurations.

Under the conditional likelihood (6.4), posterior inference is required for the joint distribution of all unknown parameters,

$$\mathbb{P}(\alpha, \beta, \mathbb{C}, \mathbb{S}|\mathcal{T}', \mathcal{E}') \propto \mathbb{P}(\mathcal{E}'|\mathcal{T}', \alpha, \beta, \{Z_{xy}\})\mathbb{P}(\alpha)\mathbb{P}(\beta|\mathbb{C}, \mathbb{S})\mathbb{P}(\mathbb{C})\mathbb{P}(\mathbb{S}), \tag{6.7}$$

Once again, exact inference is not analytically tractable and so MCMC is required to perform posterior inference. The exact MCMC sampling scheme employed can be found in Section 6.4, while below we describe a spectral clustering approach used to provide initial cluster configurations for both clients and servers to seed the MCMC sampling algorithm.

6.2.1 Surrogate spectral biclustering model

The commonly used spectral biclustering algorithm of Dhillon (2001) and Cho et al. (2004) described in Section 4.2.2 is used as surrogate clustering model to infer an initial cluster configuration for both the clients and the servers in the network. The left singular-vectors obtained from the spectral decomposition correspond to the clients projected into a K-dimensional latent space, and similarly the right singular vectors correspond to latent-space positions for the servers. We then perform k-means clustering on these latent representations to extract initial cluster configurations of the clients and the servers. This gives us a simpler model that nonetheless provides an initial cluster configuration for (6.7) which can then be further updated according to the Cox proportional hazards model, as further described in Section 6.4.

As most clustering models, the model presented in this section takes a class-oriented representation based on stochastic blockmodels (Nowicki and Snijders, 2001). The main limitation is to assume that there is a finite number of clusters (classes) and that these classes entirely determine the structure of the graph. Additional flexibility can be achieved by using the Dirichlet process (e.g. the Chinese restaurant process, Pitman (2002)) to allow a potentially infinite number of

clusters. However, this representation still limits each data point to one cluster, whilst ideally we would like to learn a representation which allows for overlapping clusters. Although computationally convenient, it may be statistically inaccurate to assume that there exists a single partition that correctly characterises the observed data. Therefore, a latent feature approach is explored in the next section, with the purpose of increasing the flexibility of the generative process.

6.3 Latent feature formulation

Latent feature models increase the flexibility of class-oriented models by letting each entity possess a (potentially unbounded) vector of latent features. These models allow us to learn a compact representation where the observed as well as the unobserved data can be explained through a small number of components. Note that any latent class model can be represented as a latent feature model by restricting each row to have only a single non-zero entry. Conversely, we could also view a latent feature model as a latent class representation by constructing as many different classes as the number of available features. For instance, supposing that there are K features, we would need 2^K classes to construct the analog of a latent feature model with K features. Clearly, this would result in an exponential growth of the number of clusters, making inference soon unfeasible.

One first class of latent feature models has been developed by (Hoff et al., 2005) and (Hoff, 2009, 2002) in the field of social network analysis. There, they use real-valued vectors as latent representations of the entities and the link probability between two entities is determined by the similarity of their real-valued feature vector. Then Miller et al. (2009) use instead a vector of binary features. The number of features is assumed not to be known a priori and the Indian Buffet Process (IBP) (Ghahramani and Griffiths, 2005) is used to determine the number of latent clusters. For the remaining of this chapter, we will determine similarity using real-valued latent feature and we will assume the number of features not to be known a priori.

In Section 6.2.1, latent-space representations were used as data locations within the clustering algorithm, for grouping together similar clients or servers. In this representation, each client and each server of the network graph is associated with a vector of latent features, with a common (but potentially unbounded) dimension $K < \infty$. The latent-space vectors for each client and each server of the network graph are then combined by a simple dot-product to provide a score of attraction between client and server. Two different scenarios for determining the number of features and the subset of features selected will be explored. First, we will assume the number of features to be unknown but finite, and then we will extend the flexibility by allowing the vector of latent features to be potentially infinite. In the latter case the Indian Buffet Process will be used. These two cases are now described in Section 6.3.1 and 6.3.2.

6.3.1 The finite latent feature case

Let $U = (u_1, \ldots, u_{|X|}) \in \mathbb{R}^{|X| \times K}, V = (v_1, \ldots, v_{|Y|}) \in \mathbb{R}^{|Y| \times K}$ be matrices containing the K-dimensional, real-valued latent feature vectors for the client and server computers respectively. Then the latent feature model score of attraction between client x and server y is simply given by

$$Z_{xy}(t) = u_x \cdot {v_y}^T. \tag{6.8}$$

Note that (6.8) is fixed and not time-varying. Since the magnitude of the vectors u_x and v_y provide a *sociability* effect for each client x or server y respectively, we restrict the $\{\beta_{xy}\}$ regression coefficients to a single constant,

$$\beta_{xy} = \beta \tag{6.9}$$

implying just one free parameter β which is assigned a standard normal prior distribution. Note that there is a slight identifiability issue between the free matrix entries of U, V and the single coefficient β. However, the relative dimensionality of the large matrices against a single scalar means that if the event data support a

strong latent feature effect, under the chosen priors this will be reflected through the parameter β which incurs just a single penalty.

Under this scenario, the subset of real-valued features selected U and V is inferred via truncated-SVD. Conditional on U and V and given the conditional likelihood (6.4), the joint posterior distribution of model parameters is given up to proportionality by

$$\mathbb{P}(\alpha, \beta | \mathcal{T}', \mathcal{E}', U, V) \propto \mathbb{P}(\mathcal{E}' | \mathcal{T}', \alpha, \beta, U, V)\mathbb{P}(\alpha)\mathbb{P}(\beta). \tag{6.10}$$

Details about model parameters inference will be given in Section 6.4, while below we propose a truncated-SVD approach to infer latent positions U and V.

6.3.1.1 Truncated SVD

In Section 6.2, truncated-singular value decomposition was applied to a normalised transformation (4.12) of the binary adjacency matrix A. Under this model formulation, we propose to operate the SVD decomposition on a weighted adjacency matrix whose (x, y) entry is an empirical estimate of the intensity of connections between client x and server y. As later shown in Section 6.5, applying SVD on a weighted adjacency matrix leads to better results than from using standard SVD directly on A. The idea here is that the SVD decomposition is used as part of a log-linear model for the intensity function (6.1) and so it is likely more suitable to provide a covariate which is derived from the empirical estimate of the intensity of connections in the graph.

From observing the arrivals of new connections over a time interval $[0, T]$, we construct a $|X| \times |Y|$ matrix, denoted $\hat{\Lambda}$, where the (x, y) component $\hat{\Lambda}_{x,y}$ is an estimate of the intensity in (6.1) obtained from a simple conjugate Bayesian model. For the stochastic process of directed graphs $\{G_t | t \geq 0\}$, we can define a random variable for the waiting time until the edge (x, y) is first observed as

$$T_{x,y} = \inf_t \{t | (x, y) \in G_t\}. \tag{6.11}$$

To specify our simple Bayesian model, we suppose $T_{x,y} \sim \text{Exp}(\Lambda_{x,y})$, with conjugate prior for the unknown intensity $\Lambda_{x,y} \sim \Gamma(\gamma, \upsilon)$. Correspondingly, after having observed the evolution of G_t on $[0, T]$, we define

$$\tilde{t}_{xy} = \begin{cases} T_{x,y} & T_{x,y} \leq T \\ T & T_{x,y} > T \end{cases} \tag{6.12}$$

to be either the observed value or else a right-censoring time, for (6.11). Then conditioning on this observed value (6.12), the posterior distribution for the intensity is simply

$$\Lambda_{x,y} | \tilde{t}_{xy} \sim \Gamma(\gamma + \mathbb{1}_{[0,T]}(t_{xy}), \upsilon + t_{xy}), \tag{6.13}$$

which yields the posterior mean estimate for the (x, y) entry of $\hat{\Lambda}$,

$$\hat{\Lambda}_{x,y} = \frac{\gamma + \mathbb{1}_{[0,T]}(t_{xy})}{\upsilon + t_{xy}}. \tag{6.14}$$

Then, the truncated rank-K singular value decomposition of $\hat{\Lambda}$ is given by

$$\hat{\Lambda} \approx \hat{\Lambda}^{(K)} \equiv U \Sigma V^T = \sum_{k=1}^{K} s_k u_k {v_k}^T. \tag{6.15}$$

Here, the left singular vectors correspond to the client computers projected into a K-dimensional latent space, and the right singular vectors corresponding to the servers. In case these K-dimensional latent positions are further updated with MCMC, the Indian Buffet Process is used as a prior distribution and the full process is described in the next section.

6.3.2 The infinite latent feature case

Under this scenario, the Indian Buffet Process is used to determine the number of latent features and the subset of features selected for each client and server. This provides a flexible nonparametric approach which assumes that there is a poten-

tially infinite number of latent features. Bayesian inference requires the specification of a prior over this infinite binary matrix: the Indian Buffet Process is such a prior and its generative process will be discussed in the following.

In particular, while specifying such prior on the triple (K, U, V), we look to introduce some cluster structure in the features by encouraging sparsity in the matrices U and V, so that each client/server only possesses a subset of the possible latent feature measurements. Sparsity reflects the idea that only some small proportion of coefficients should be non-zero. The reason why sparsity is often sought is that statistical real world data often exhibit sparsity. In addition, sparsity assumptions can be a very good regulariser to avoid model overfitting and can be exploited for efficient computation. The IBP naturally incorporates sparsity and the resulting sparse representation allows increased data reduction compared to standard data reduction tools. The assumption that the observed entities only manifest a sparse subset of an unbounded number of latent classes is often used in nonparametric Bayesian statistics. Bayesian nonparametric models are very flexible and can be able to model data better than less flexible models, for instance models in which the number of classes is chosen a priori. This is due to the fact that Bayesian methods are most accurate when the prior adequately captures the a priori assumptions.

Suppose F represents a matrix containing K-dimensional latent feature vectors, one for each entity of interest. Features can be binary or real-valued, and so it is possible to break F into two components: a binary matrix Δ indicating which features are possessed by each entity, and a matrix W representing the value of each feature for each entity. Thus, F can be viewed as the Hadamard elementwise product of Δ and W, $F = \Delta \odot W$.

Following this notation, the cluster feature matrix U can be therefore decomposed into two components: a binary matrix $\Delta_U \in \{0, 1\}^{|X| \times K}$ with entry $\Delta_{xk} = 1$ if and only if client x possesses feature k, and a second matrix $\tilde{U} \in \mathbb{R}^{|X| \times K}$ comprising the continuous feature values of each feature for each client. The feature matrix U can then be expressed as the elementwise Hadamard product of these

two matrices,

$$U = \Delta_U \odot \tilde{U}, \tag{6.16}$$

as illustrated in Figure 6.1. Similarly, the server feature matrix $V = (v_1, \ldots, v_{|Y|}) \in \mathbb{R}^{|Y| \times K}$ is expressed as the Hadamard product of matrices $\Delta_V \in \{0,1\}^{|Y| \times K}$ and $\tilde{V} \in \mathbb{R}^{|Y| \times K}$,

$$V = \Delta_V \odot \tilde{V}. \tag{6.17}$$

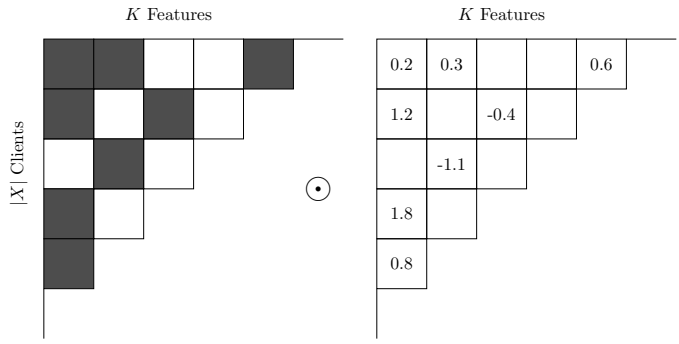


Figure 6.1: Example of the decomposition of U. A binary matrix Δ_U (first panel) indicates which features are active. Elementwise multiplication of Δ_U by $\tilde{U}$ of continuous values produces the representation in the second panel.

Note that in the latent class representation described in the previous section Δ_U (Δ_V) would be a binary matrix with each row corresponding to each data point and each column corresponding to a class of clients (servers). For determining K and the subset of features selected for each client and server, independent Indian buffet process priors with Poisson parameter $\theta > 0$ are assigned to Δ_U and Δ_V. The IBP defines a distribution over the rows of an infinite binary matrix, and its generative process is described below.

Indian Buffet Process This model posits that each entity can be explained by a *sparse* subset of latent features. The number of features is unbounded a priori but with probability 1, a feature matrix drawn from an IBP will only have a finite number of non-zero features. Inspired by the derivation of the Chinese restaurant process by Pitman (2002), the name of the IBP is a culinary metaphor describing

how to generate the non-zero columns of a matrix of features. We illustrate the IBP process by considering the binary client feature matrix $\Delta_U \in \mathbb{R}^{|X|\times K}$. The following steps equivalently hold for the computer binary matrix $\Delta_V \in \mathbb{R}^{|Y|\times K}$. In this metaphor, each row of Δ_U corresponds to a customer at an Indian buffet and each column corresponds to one of infinitely many dishes. For a client x, let Δ_{xk} be the entry at (x, k), then we have $\Delta_{xk} = 1$ if the x^{th} customer tastes the k^{th} dish and zero otherwise. The process is shown in Figure 6.2. Without referring to the culinary metaphor, the feature matrix Δ_U can be generated from an IBP(θ), with $\theta > 0$, starting with an all-zero matrix and performing the following:

- In the first row, set the first Poisson(θ) columns to one. Leave the rest all zero.
- Assuming the first $x-1$ rows are filled, then fill in the x^{th} row as follows:
  - for each non-zero column, set the $\Delta_{xk} = 1$ with probability m_k/x, with m_k number of non-zero entries in the k^{th} column,
  - add an additional Poisson(θ/x) ones after that last non-zero column.

More specifically, the prior distribution resulting from the IBP is constructed as the limit of a finite Beta-Bernoulli process model (Hjort, 1990). This model is described in Appendix B.1 for completeness. Ghahramani and Griffiths (2005) show that the likelihood of Δ_U (Δ_V) generated by the Beta-Bernoulli process in the limit that the number of features $K \to \infty$ is equivalent to the one generated by the Indian Buffet Process. Ghahramani et al. (2007) then define a scheme to order the non-zero rows of Δ_U and Δ_V which allows us to take the limit $K \to \infty$ and find that the probabilities of being produced by an IBP(θ) are respectively given by

$$\begin{aligned}
\mathbb{P}(\Delta_U) &= \frac{\theta^K}{\prod_{x=1}^{|X|} K_x!}\exp\{-\theta H_{|X|}\}\prod_{k=1}^{K}\frac{(|X|-m_k)!(m_k-1)!}{|X|!}, \\
\mathbb{P}(\Delta_V) &= \frac{\theta^K}{\prod_{y=1}^{|Y|} K_y!}\exp\{-\theta H_{|Y|}\}\prod_{k=1}^{K}\frac{(|Y|-t_k)!(t_k-1)!}{|Y|!},
\end{aligned} \tag{6.18}$$

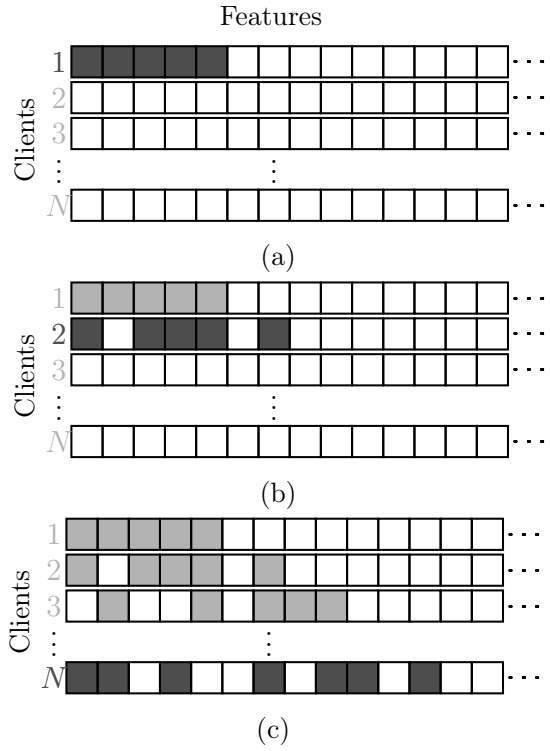


Figure 6.2: An illustration of the Indian Buffet Process for Δ_U. Note that $N = |X|$. (a) The first client samples Poisson(θ) features, which is recorded by changing the corresponding entries of Δ_U to one. (b) and (c) For the x^{th} client, the first step is activating the previously sampled features with probability proportional to the number of clients who already have these features active. The next step is to activate a Poisson(θ/x) number of new features.

where K_x and K_y indicate the number of new features sampled respectively by client x and computer y; $H_{|X|} = \sum_{x=1}^{|X|} \frac{1}{x}$, $H_{|Y|} = \sum_{y=1}^{|Y|} \frac{1}{y}$ and t_k is the number of non-zero entries in the k^{th} column of Δ_V. Derivations of these equations are provided in Appendix B.2. The IBP assumes exchangeability of the rows of Δ_U and Δ_V, while the columns are independent. This means that the order in which the clients activate each feature thus the ordering in the IBP does not affect the likelihood. Note that this is not necessarily intuitive as, recalling the buffet construction, the client chooses the features based solely on their 'popularity'. In summary, the IBP is a simple generative process for which each client and server has an expected number of features equal to the Poisson parameter θ.

In the present context, independent Indian buffet process (IBP) priors with Poisson parameter θ will be assigned to both Δ_U and Δ_V. This prior will allow us to select a final number of features which may be different from the initial number obtained through SVD. Under this scenario, if K_U (K_V) is the total number of features activated within Δ_U (Δ_V), then the resulting dimension for the model is taken to be the maximum, $K = \max\{K_U, K_V\}$.

Conditional on K, the continuous-valued entries of $\tilde{U}$ and $\tilde{V}$ are assigned independent standard normal priors. Given the conditional likelihood (6.4), the joint posterior distribution is given up to proportionality by

$$\mathbb{P}(\alpha, \beta, U, V|\mathcal{T}', \mathcal{E}') \propto \mathbb{P}(\mathcal{E}'|\mathcal{T}', \alpha, \beta, U, V)\mathbb{P}(\alpha)\mathbb{P}(\beta)\mathbb{P}(U)\mathbb{P}(V). \tag{6.19}$$

Full details on posterior inference for α, β, U and V can be found in Section 6.4. Following Section 6.2, a surrogate truncated-SVD model is used to provide initial low-rank latent positions for clients and servers. These initial positions are then used to seed MCMC sampling. Under the IBP prior, the implied assumption for Δ_U and Δ_V is to retain only a *sparse* subset of an unbounded number of features. Thus, to obtain initial matrices resembling draws from an IBP we need to enforce sparsity on both the left and the right singular vectors obtained in Section 6.3 from the truncated-SVD of $\hat{\Lambda}$. The desired structure can be obtained by using a so-called sparse singular value decomposition, as described below.

6.3.2.1 Sparse truncated SVD incorporating stability selection

Sparse, initial latent feature matrices can be achieved by interpreting singular vectors of a standard SVD problem as the regression coefficients of a linear regression and imposing sparsity-inducing penalties in the optimisation objective in (6.15). The way the SVD approximation relates to linear regression is extensively discussed by Lee et al. (2010).

Sparse singular vectors are obtained by finding the first SVD-*layer* which min-

imises the following penalised regression with respect to (s_1, u_1, v_1):

$$\|\hat{\Lambda} - s_1 u_1 v_1^T\|_F^2 + \rho_{u_1} P_1(s_1, u_1) + \rho_{v_1} P_2(s_1, v_1), \tag{6.20}$$

where $P_1(s_1, u_1)$ and $P_2(s_1, v_1)$ are sparsity-inducing penalty terms and ρ_{u_1} and ρ_{v_1} tuning parameters that determine the amount of regularisation. As further described in Section 6.4, the penalty functions will be adaptive lasso penalties. Note that $\rho_{u_1} = \rho_{v_1} = 0$ leads to the plain SVD criterion. Subsequent *layers*, or biclusters, can be extracted in the same way from the residual matrices obtained by sequentially subtracting the preceding *layers* and applying standard SVD again. However, the sparseness of U and V can strongly depend on the choice of the penalisation parameters. Following Sill et al. (2011), we use stability selection to obtain stable penalisation parameters and to control the degree of sparsity, i.e. the number of non-zero coefficients.

Stability selection (Meinshausen and Bühlmann, 2010) is a subsampling method which combines resampling with variable selection methods, such as penalised regression, allowing Type I (false discoveries) error rates to be controlled. Here, the selection probability for each variable is estimated by randomly resampling from the data and calculating the relative frequency, as described below.

Suppose $S_{u_1} = \{x : u_{1,x} \neq 0\}$ is the true set of variables with non-zero coefficients in the left singular vector u_1 and $\mathcal{P}_{u_1}$ the set of possible penalty parameters. Different choices of $\rho_{u_1} \in \mathcal{P}_{u_1}$ lead to different estimated subsets $\hat{S}_{u_1}^{\rho_{u_1}} \subseteq \{1, \ldots, |X|\}$. For any given ρ_{u_1}, $\hat{S}_{u_1}^{\rho_{u_1}}$ is implicitly a function of the samples $I = \{1, \ldots, |Y|\}$. We denote $\hat{S}_{u_1}^{\rho_{u_1}} = \hat{S}_{u_1}^{\rho_{u_1}}(I)$ where necessary to express this dependence. Let I^* be a random subsample of $\{1, \ldots, |Y|\}$ of size $\lfloor |Y|/2 \rfloor$ drawn without replacement. Then, the estimated probability of being in the selected set $\hat{S}_{u_1}^{\rho_{u_1}}(I)$ is

$$\hat{\pi}_x^{\rho_{u_1}} = P(x \in \hat{S}_{u_1}^{\rho_{u_1}}(I^*)). \tag{6.21}$$

For a cut-off threshold $\pi_{\text{thr}} \in (0, 1)$ and a given set of regularisation parameters

ρ_{u_1}, the set of stable variables is

$$\hat{S}_{u_1}^{\text{stable}} = \left\{ x : \max_{\rho_{u_1} \in \mathcal{P}_{u_1}} \hat{\pi}_x^{\rho_{u_1}} \geq \pi_{\text{thr}} \right\}. \tag{6.22}$$

We denote the corresponding stable coefficient by $\rho_{u_1}^{\text{stable}}$. Meinshausen and Bühlmann (2010) show that the threshold value has a very small influence and they suggest to choose $\pi_{\text{thr}} \in (0.6, 0.9)$. Let $\hat{S}^{\mathcal{P}_{u_1}} = \bigcup_{\rho_{u_1} \in \mathcal{P}_{u_1}} \hat{S}_{u_1}^{\rho_{u_1}}$ be the set of variables selected under some regularisation parameter ρ_{u_1} and let $q_{\mathcal{P}_{u_1}} = E(|\hat{S}^{\mathcal{P}_{u_1}}(I^*)|)$ be the average number of selected coefficients and $N_{u_1} = \{x : u_{1,x} = 0\}$ be the set of variables with zero coefficients. Then, $V_{u_1} = |N_{u_1} \bigcap \hat{S}_{u_1}^{\text{stable}}|$ is the number of falsely selected variables with stability selection. The expected number of falsely selected coefficients, $E(V_{u_1})$, is very hard to control and under some simplifying assumptions, Meinshausen and Bühlmann (2010) find that $E(V_{u_1})$ is bounded by

$$E(V_{u_1}) = \frac{1}{(2\pi_{\text{thr}} - 1)} \frac{q_{\mathcal{P}_{u_1}}^2}{|X|}. \tag{6.23}$$

Let $e_{\mathcal{P}_{u_1}} = \sqrt{E(V_{u_1})(2\pi_{\text{thr}} - 1)|X|}$, then the error level can be controlled as long as $q_{\mathcal{P}_{u_1}} < e_{\mathcal{P}_{u_1}}$, which provides an upper bound for the average number of falsely selected coefficients. Most importantly, the error control also gives a stopping criterion for the algorithm and automatically determines the number of reasonable clusters.

Clearly, the same procedure is used to infer the true set of non-zero coefficients in the right singular vector v_1. In brief, the selection probabilities $\pi_y^{\rho_{v_1}}$ are estimated for each ρ_{v_1} by taking subsets $J^* \subset \{1, \ldots, |X|\}$. The regularisation region is again delimited by $q_{\mathcal{P}_{v_1}} \leq e_{\mathcal{P}_{v_1}}$, where $e_{\mathcal{P}_{v_1}} = \sqrt{E(V_{v_1})(2\pi_{\text{thr}} - 1)|Y|}$ and the set of stable non-zero coefficients is given by

$$\hat{S}_{v_1}^{\text{stable}} = \left\{ y : \max_{\rho_{v_1} \in \mathcal{P}_{v_1}} \hat{\pi}_y^{\rho_{v_1}} \geq \pi_{\text{thr}} \right\}. \tag{6.24}$$

The detailed algorithm used for efficient estimation of the sparse SVD-*layers*

incorporating stability selection is further described in Section 6.4.2.1.

6.4 Posterior inference

The inference procedure used to estimate the quantities of interest for the Bayesian proportional hazards model proposed in Section 6.1 under both the clustering formulation presented in Section 6.2 and the latent feature extension presented in Section 6.3 are described below.

6.4.1 Cluster formulation inference

The main inference procedure for the Cox model posterior distribution in (6.7) consists of two steps. To initialise the algorithm, row and column cluster configurations are first obtained through the spectral biclustering algorithm described in Section 6.2; subsequently, these configurations are updated via MCMC jointly with Cox model parameters. The Metropolis-Hastings (M-H) algorithm is used to draw approximate samples from the joint posterior distribution in (6.7) of the Cox model parameters α and β and cluster configurations $\mathbb{C}$ and $\mathbb{S}$. The sampling procedure is analogous to what described in Section 5.2.2, where for altering α and β, simple random walks with Gaussian steps were applied to each randomly selected component. Similarly, the same procedure described for sampling the clustering configuration is now applied to both the client cluster configuration $\mathbb{C}$ and the server cluster configuration $\mathbb{S}$.

6.4.2 Latent feature formulation inference

The main inference procedure for the Cox model posterior distribution in (6.19) consists of two steps. First, sparse singular value decomposition as described in Section 6.3 is used to provide reliable initial latent positions, parametrised through $\tilde{U}, \tilde{V}, \Delta_U, \Delta_V$; subsequently the Cox model parameters and initial latent positions are jointly updated through an MCMC sampling scheme. The two steps

are separately described below.

6.4.2.1 Sparse SVD stability selection algorithm

Singular value decomposition incorporating stability selection is used as described in Section 6.3, in order to provide initial sparse latent configurations. Recall that sparse singular vectors are obtained by finding the first SVD-*layer* which minimises (6.20). The lasso adaptive penalties P_1 and P_2 are here chosen with weights $w_{1,i}$ and $w_{2,j}$ as

$$P_1(s_1, u_1) = s_1 \sum_{x=1}^{|X|} w_{1,x}|u_{1,x}|, \quad P_2(s_1, v_1) = s_1 \sum_{y=1}^{|Y|} w_{2,y}|v_{1,y}|. \tag{6.25}$$

Fixing v_1, the minimisation in (6.20) becomes

$$\|\hat{\Lambda} - s_1 u_1 v_1^T\|_F^2 + \rho_{u_1} s_1 \sum_{x=1}^{|X|} w_{1,x}|u_{1,x}| = \|\hat{\Lambda}\|_F^2 + \sum_{i=1}^{|X|} \{\tilde{u}_{1,x}^2 - 2\tilde{u}_{1,x}(\hat{\Lambda}v_1)_i + \rho_{u_1} w_{1,x}|\tilde{u}_{1,x}|\}, \tag{6.26}$$

and the estimated left singular vector $\hat{\tilde{u}}_{1,x}$ can be obtained by using the well known soft threshold estimator proposed by Tibshirani (1996) and the stable coefficient $\rho_{u_1}^{\text{stable}}$ resulting from stability selection, as

$$\hat{\tilde{u}}_{1,x} = \text{sign}\left\{(\hat{\Lambda}v_1)_x\right\}(|(\hat{\Lambda}v_1)_x| - \rho_{u_1}^{\text{stable}} w_{1,x}/2)_+, \tag{6.27}$$

where $(a)_+ = \max\{0, a\}$. Then $\hat{\tilde{u}}_1 = (\hat{\tilde{u}}_{1,1}, \ldots, \hat{\tilde{u}}_{1,|X|})^T$ gives an estimate of the product of the first left singular vector multiplied with the first singular value. Therefore, the estimated sparse singular vector becomes $\hat{u}_1 = \hat{\tilde{u}}_1/\|\hat{\tilde{u}}_1\|$.

Similarly, fixing u_1 and following the same procedure we have

$$\hat{\tilde{v}}_{1,y} = \text{sign}\left\{(\hat{\Lambda}^T u_1)_y\right\}(|(\hat{\Lambda}^T u_1)_y| - \rho_{v_1}^{\text{stable}} w_{2,y}/2)_+. \tag{6.28}$$

Then $\hat{\tilde{v}}_1 = (\hat{\tilde{v}}_{1,1}, \ldots, \hat{\tilde{v}}_{1,|Y|})^T$ and $\hat{v}_1 = \hat{\tilde{v}}_1/\|\hat{\tilde{v}}_1\|$. Penalised regression models for

both u_1 and v_1 are alternated until convergence, i.e. either $\|u_1 - \hat{u}_1\| < \epsilon$ or $\|v_1 - \hat{v}_1\| < \epsilon$, for a chosen convergence threshold $\epsilon > 0$. Finally, coefficients which are not contained in the two stable sets $\hat{S}_{u_1}^{\text{stable}}$ and $\hat{S}_{v_1}^{\text{stable}}$ are set to zero. Therefore, the components of $\hat{u}_1$ becomes $\hat{u}_{1,x} = \mathbb{1}(x \in \hat{S}_{u_1}^{\text{stable}})\hat{u}_{1,x}$ and similarly the components of $\hat{v}_1$ becomes $\hat{v}_{1,y} = \mathbb{1}(y \in \hat{S}_{v_1}^{\text{stable}})\hat{v}_{1,y}$. The next layers are sequentially obtained by repeating the steps above on the residual matrix subtracting the preceding approximation derived by applying regular SVD to the submatrices defined by $\hat{S}_{u_1}^{\text{stable}}$ and $\hat{S}_{v_1}^{\text{stable}}$. The whole procedure stops if either $\hat{S}_{u_1}^{\text{stable}} = \emptyset$ or $\hat{S}_{v_1}^{\text{stable}} = \emptyset$. The full algorithm is outlined in Algorithm 4.

Algorithm 4 SSVD with stability selection

INPUT=($|X| \times |Y|$ matrix $\hat{\Lambda}$)
repeat<Find SSVD layers>
  Apply SVD to $\hat{\Lambda}$.
  Denote the first SVD layer as $\{s_1, u_1, v_1\}$, choose $E(V_{v_1})$, $E(V_{u_1})$, π_{thr}, ϵ.
  while $\|u_1 - \hat{u}_1\| > \epsilon$ or $\|v_1 - \hat{v}_1\| > \epsilon$ **do**
    $\forall \rho_{u_1}$ draw I^* and estimate $\hat{\pi}_x^{\rho_{u_1}}$, define $\mathcal{P}_{u_1}$ s.t. $q_{\mathcal{P}_{u_1}} < e_{\mathcal{P}_{u_1}}$.
    Estimate $\hat{S}_{u_1}^{\text{stable}}$.
    Set $\hat{\tilde{u}}_{1,x} = \text{sign}\left\{(\hat{\Lambda}v_1)_x\right\}(|(\hat{\Lambda}v_1)_x| - \rho_{u_1}^{\text{stable}} w_{1,x}/2)_+$
    Let $\hat{\tilde{u}}_1 = (\hat{\tilde{u}}_{1,1}, \ldots, \hat{\tilde{u}}_{1,|X|})^T$ and $\hat{u}_1 = \hat{\tilde{u}}_1/\|\hat{\tilde{u}}_1\|$.
    $\forall \rho_{v_1}$ draw J^* and estimate $\hat{\pi}_i^{\rho_{v_1}}$, define $\mathcal{P}_{v_1}$ s.t. $q_{\mathcal{P}_{v_1}} < e_{\mathcal{P}_{v_1}}$.
    Estimate $\hat{S}_{v_1}^{\text{stable}}$.
    Set $\hat{\tilde{v}}_{1,y} = \text{sign}\left\{(\hat{\Lambda}^T u_1)_y\right\}(|(\hat{\Lambda}^T u_1)_y| - \rho_{v_1}^{\text{stable}} w_{2,y}/2)_+$.
    Let $\hat{\tilde{v}}_1 = (\hat{\tilde{v}}_{1,1}, \ldots, \hat{\tilde{v}}_{1,|Y|})^T$ and $\hat{v}_1 = \hat{\tilde{v}}_1/\|\hat{\tilde{v}}_1\|$.
    Set $v_1 = \hat{v}_1$ and $u_1 = \hat{u}_1$.
  Set $\hat{s}_1 = \hat{u_1}^T \hat{\Lambda} \hat{v}_1$ so that $\hat{u}_{1,i} = \mathbb{1}(i \in \hat{S}_{u_1}^{\text{stable}})\hat{u}_{1,x}$, $\hat{v}_{1,y} = \mathbb{1}(y \in \hat{S}_{v_1}^{\text{stable}})\hat{v}_{1,y}$.
  Set $\hat{\Lambda} = \hat{\Lambda} - s_1 \hat{u}_1 \hat{v_1}^T$ (submatrix defined by $\hat{S}_{u_1}^{\text{stable}}$ and $\hat{S}_{v_1}^{\text{stable}}$)
until $\hat{S}_{u_1}^{\text{stable}} = \emptyset$ or $\hat{S}_{v_1}^{\text{stable}} = \emptyset$
OUTPUT=(Set of stable SSVD layers)

6.4.2.2 Inference of features and Cox model parameters

The latent feature approach allows us to simultaneously infer the number of latent features while at the same time inferring which features each entity has and how observed data are influenced by those features. Under the Indian Buffet process prior, our algorithm uses Metropolis sampling moves to sample both the Cox model parameters α and β and the feature matrices U and V. Again, for altering α and β, simple random walks with Gaussian steps were applied to each randomly selected component and so attention is below focused on sampling $\tilde{U}, \tilde{V}, \Delta_U, \Delta_V$.

Different steps are respectively needed to sample the feature matrices Δ_U and Δ_V, under the IBP. For simplicity, we describe the feature sampling scheme focusing only on the client feature matrix Δ_U, although the same sampler is used for the server matrix Δ_V. For each client x, we need to sample both the value of an existing feature k, Δ_{xk}, and the number of new features K_x. The latter step is necessary as the IBP prior treats the number of latent features as itself a random variable.

At iteration $t+1$, we propose changes to $\tilde{U}^t$ and Δ_U^t (analogous approaches are used to sample $\tilde{V}^t$ and Δ_V^t) as follows.

- **Sampling $\boldsymbol{\Delta_{xk}}$:** For a randomly sampled client x, let

$$K_x^t = \{k | 1 \leq k \leq K_U, \Delta_{xk}^t = 1\} \tag{6.29}$$

be the features currently activated for that client in the latent feature model. Further, for $k \in \{1, \ldots, K_U\}$ let $d_k^t = \sum_{x \in X} \Delta_{xk}^t$be the number of clients with feature k currently active. For a randomly chosen feature $k \in \{1, \ldots, K_U + 1\}$, each value Δ_{xk} can be resampled from its full conditional distribution,

$$\mathbb{P}(\Delta_{xk} | \Delta_{-(x,k)}^t, \ldots) \propto \mathbb{P}(\mathcal{E}' | \mathcal{T}', U, V, \alpha, \beta) \mathbb{P}(\Delta_{xk} | \Delta_{-(x,k)}^t), \tag{6.30}$$

where the second term is the conditional prior distribution for the new value

of Δ_{xk}. If $d_k^t > 1$ or $\Delta_{xk}^t = 0$, then we have

$$\mathbb{P}(\Delta_{xk}|\Delta^t_{-(x,k)}) = \frac{(d_k^t)^{\Delta_{xk}}(|X| - d_k^t)^{1-\Delta_{xk}}}{|X|}. \tag{6.31}$$

Alternatively, if $d_k^t = \Delta_{xk}^t = 1$ such that x is the only client with feature k active, then U may potentially decrease in dimension. Then, by using the recursive formula for the Poisson distribution, we have the following updates

$$\frac{\mathbb{P}(\Delta_{xk} = 1|\Delta^t_{-(x,k)})}{\mathbb{P}(\Delta_{xk} = 0|\Delta^t_{-(x,k)})} = \frac{\theta}{|X||K_x^t|}. \tag{6.32}$$

Finally if $k = (K_U + 1)$, proposing an increase in dimension of U,

$$\frac{\mathbb{P}(\Delta_{xk} = 1|\Delta^t_{-(x,k)})}{\mathbb{P}(\Delta_{xk} = 0|\Delta^t_{-(x,k)})} = \frac{\theta}{|X|(|K_x^t| + 1)}. \tag{6.33}$$

- **Sampling $\tilde{u}_{xk}$:** Simple random walks with Gaussian steps are applied to each randomly selected value $\tilde{u}_{xk}$ of $\tilde{U}$.

- **Sampling θ:** Under the IBP model, for each client x the distribution of the number of sampled features is Poisson(θ/x). By making use of the convenience of conjugacy with the prior $\Gamma(a_\theta, b_\theta)$ for the hyperparameter θ, samples can be drawn directly from the the posterior distribution

$$\Gamma(a_\theta + K_U, b_\theta + \sum_{x=1}^{|X|} 1/x). \tag{6.34}$$

6.5 An application to LANL computer network data

For computational tractability, the methods discussed in the previous sections have been tested on a sample of event data corresponding to a random selection of 1,000 clients of the 2015 LANL authentication data set described in Section 2.2. These events were analysed along with the red team event data and 15 sample repetitions were used for robustness. For MCMC sampling, the total number of iterations was

set to 5,000, after a burn-in period of size 1,000.

The posterior means of model coefficients and box-plots of their 95% highest posterior density (HPD) credible intervals, under both the cluster formulation and the latent-feature formulation, are shown in Figures 6.3 and 6.6 respectively. For the cluster formulation, to allow coherent posterior model averaging across the cluster dimensions L and M (*cf.* Section 6.2), which may vary across samples, we calculated mean coefficients $\bar{\beta}_1 = \sum_{l=1}^{L} \beta_{1,l}/L$ and $\bar{\beta}_2 = \sum_{m=1}^{M} \beta_{2,m}/M$ respectively, summarising the coefficients over the fitted client and server clusters. This still allows us to evaluate the additional value of introducing cluster information, although we loose information about the specific performance of each cluster.

The posterior estimates of the parameters are all significantly positive, with some natural, but acceptably small, level of variation across samples. For the nuisance parameters α, results confirm the *popularity* effect for both client and servers, where computers that have many connecting neighbours are more likely to make further new connections. The presence of bursts of new edge formation by clients also was found significative. Specifically, the two indicator variables for whether the last edge, or the last two edges were new, are also always positive, which intuitively suggests that new edges occur in bursts. More interestingly, the significant β parameters confirm that strong additional information is provided by the identity of the links already formed and the latent communities they suggest, whether this is characterised by hard clustering of clients and servers, or through a dot-product model using latent positions. For the cluster model, the initial number of client and server clusters identified with spectral clustering and extracted via k-means is $L = 2$ and $M = 3$ as shown in Figure 6.4, while MCMC selects on average $L = 5$ client clusters and $M = 4$ server clusters, as shown from the trace plots of the number of row and column clusters in Figure 6.7.

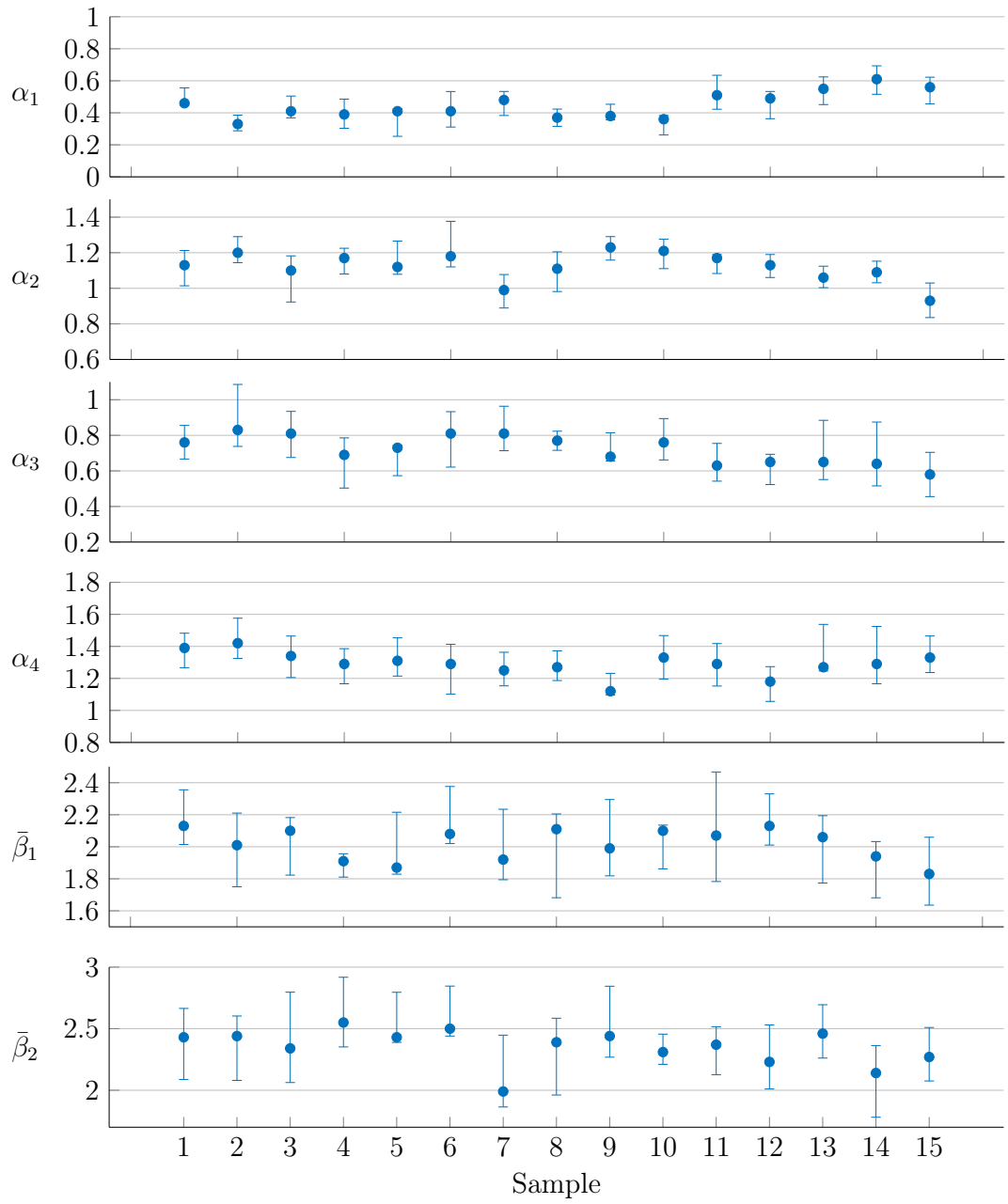


Figure 6.3: Posterior estimates coefficients under the cluster formulation, with credible intervals.

These results have been then compared to a simpler scenario, where only client clusters were inferred. Following the procedure shown in Chapter 5, we initially

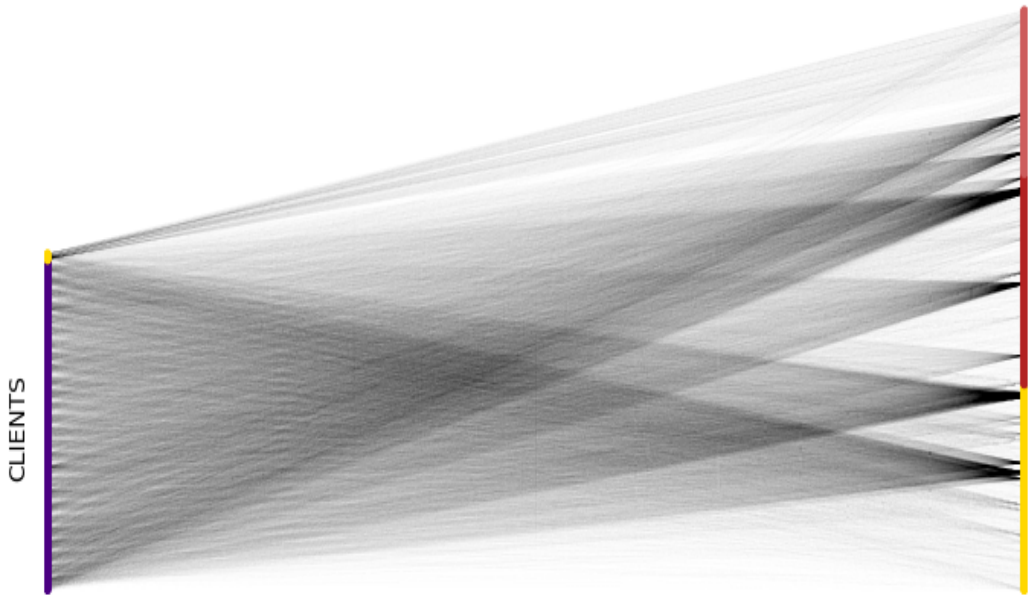


Figure 6.4: Clustered graph induced by applying spectral biclustering, reordered by row clusters.

infer reliable cluster configurations through the sequential agglomerative clustering described in Section **??**. This can provide a way to account for the additional value of including simultaneous clustering of clients and servers in the model. Posterior estimates with HPD credible intervals are reported in Figure 6.5. It can be noted that the estimates are still significantly positive but lower in magnitude, especially for the client cluster parameter $\bar{\beta}_1$, suggesting that simultaneously clustering both clients and servers enhances the performance of the model.

Figure 6.6 then shows three sets of estimates for the latent-feature model. The blue points correspond to the full inference procedure proposed in Section 6.3.2, where sparse truncated SVD is used to seed an MCMC exploration of the variable dimension latent feature space. The other two sets of points represent the resulting estimates from not performing MCMC and just using the truncated SVD to propose latent features, either with sparsity imposed (green points) or with no sparsity (yellow points) as discussed in Section 6.3.1. For stability selection, we selected the threshold π_{thr} for each value of ρ so as to guarantee $E(V_u) < 20$,

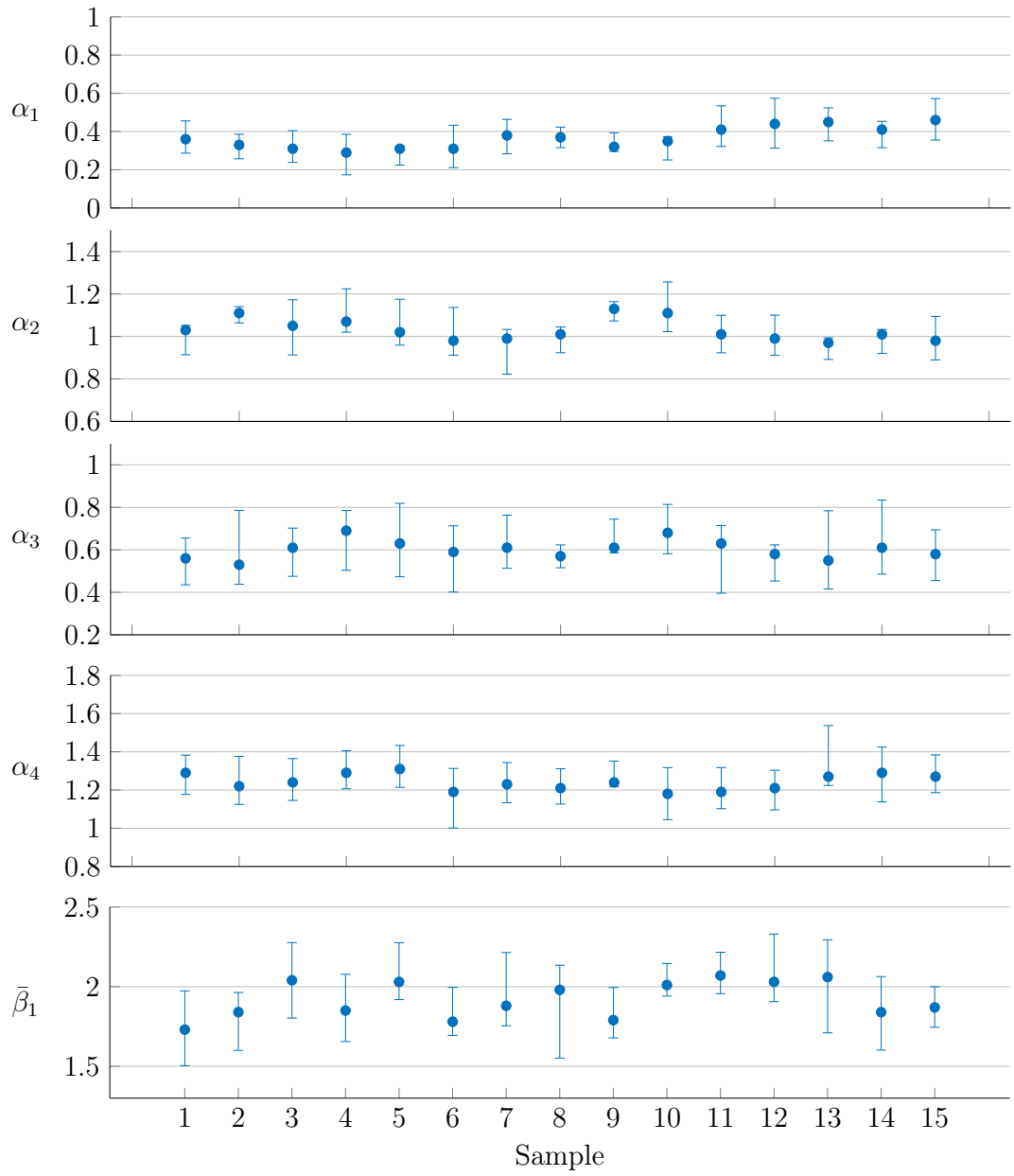


Figure 6.5: Posterior estimates with credible interval for the cluster formulation, where only client clusters were inferred.

i.e. we expect fewer than 20 wrong selections. When MCMC is performed, the magnitudes of the latent position matrices U and V will be different compared to when no MCMC moves are performed. Thus, to ensure a meaningful comparison

of these coefficients, after MCMC the latent positions U and V are rescaled so that the mean absolute value of the features $\{Z_{xy}\}$ is the same as that from the initial values from sparse truncated SVD.

The coefficients are again all significantly positive, confirming the findings of Figure 6.3. In particular, all the coefficient estimates are higher in magnitude when using the full inference procedure, suggesting that the MCMC exploration of the feature space is worthwhile for identifying more significant latent feature covariates. Figure 6.9 shows the trace plot of the number of active features from MCMC in the full inference procedure: there is some mixing and the distribution has a strong mode at $K = 8$, with some small probability associated with $K = 9$. When performing sparse truncated-SVD incorporating stability selection, a dimensionality of $K = 6$ is automatically chosen, while in the simplest case when no sparsity is imposed $K = 5$ appears to be an optimal choice, as shown from the scree plot of the SVD eigenvalues in Figure 6.8. The scree plots obtained from the remaining samples can be found in Appendix C for completeness. Appendix C also contains the scree plot obtained by applying SVD directly on the adjacency matrix A, rather than on our proposed matrix, $\hat{\Lambda}$. In such case, the SVD decomposition only found $K = 2$, confirming that best results are achieved by using $\hat{\Lambda}$.

Finally, the predictive performance of the latent feature model using the different inference methods used in Figure 6.6 was assessed on 10,000 out-of-training sample authentication events. Table 6.1 reports the averaged likelihood ratios, showing that performing MCMC leads to a distinct improvement in model fit. When no MCMC moves are performed on the latent matrices, the plain SVD leads to better results than the sparse decomposition, which in this case may just remove useful information or add unnecessary noise. In this case it is clear that we require the extra effort of MCMC to find a useful sparse solution. In this way, we have assessed the strength of the effects of each included covariate, for both the cluster and latent feature models. Therefore, in the next chapter we can finally proceed with performing anomaly-detection, based on the model which has here proved to lead to the highest predictive performance.

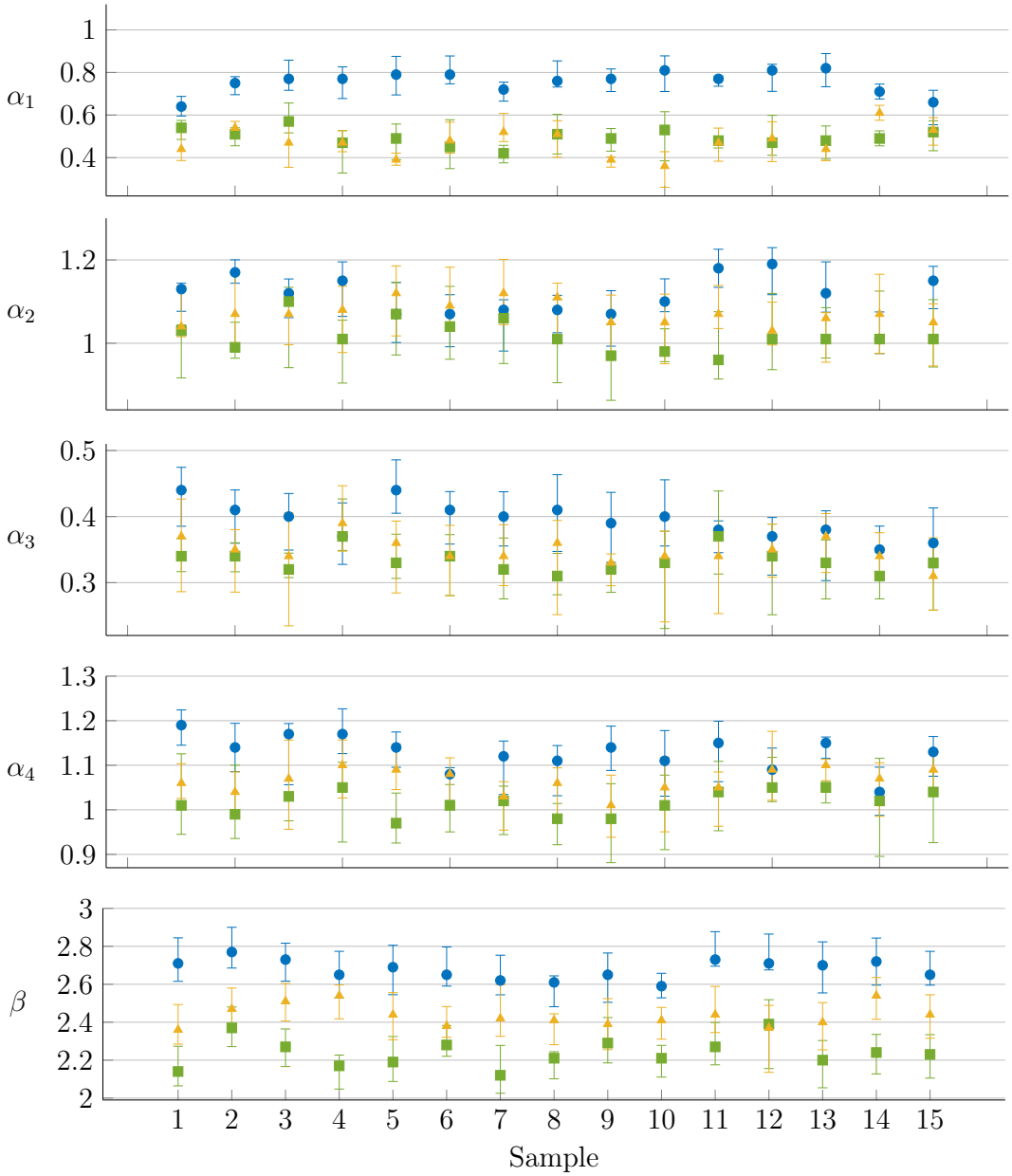


Figure 6.6: Three sets of posterior estimates coefficients under the latent feature formulation, with credible intervals, obtained from full MCMC (●), sparse truncated SVD with stability selection (■), and standard truncated SVD (▲).

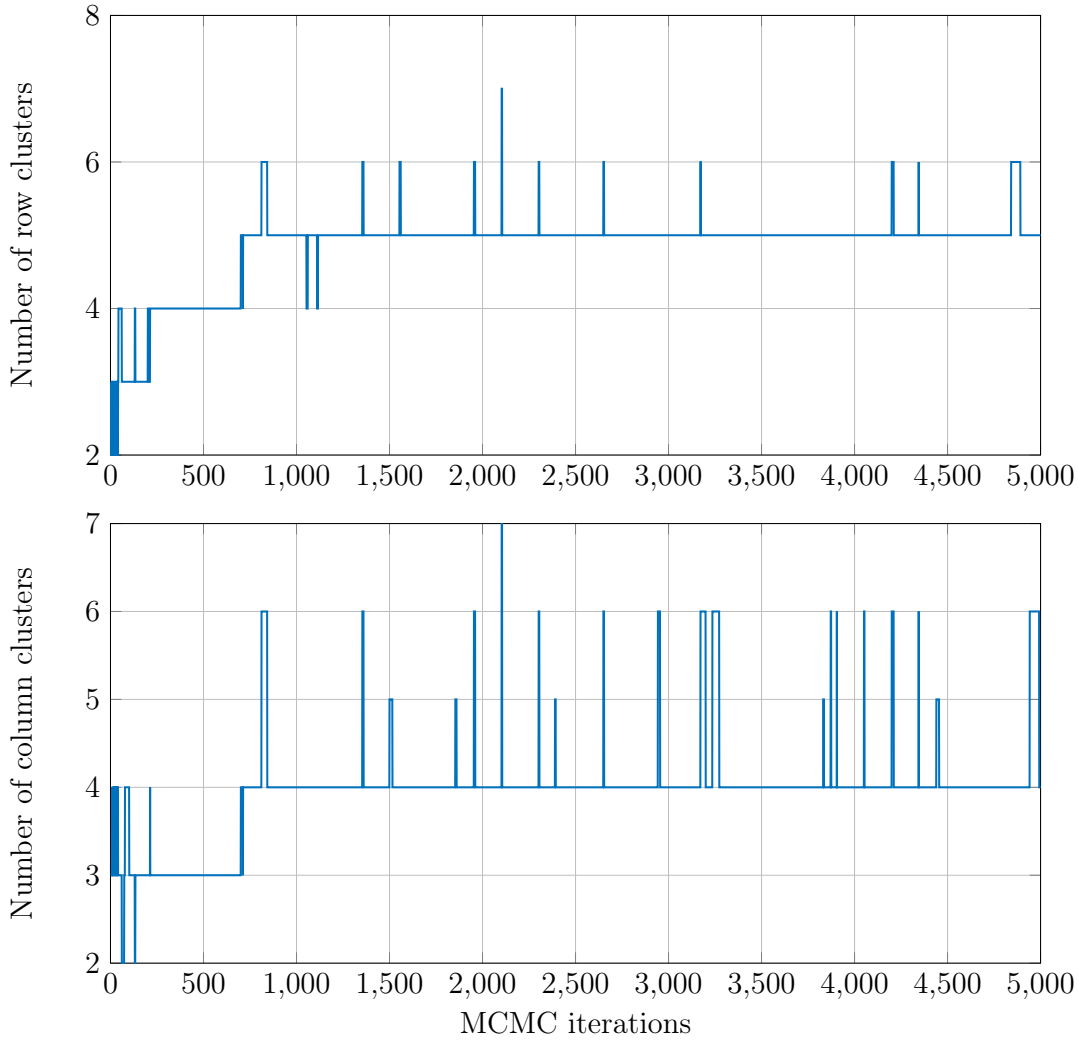


Figure 6.7: Number of identified row and column clusters during the MCMC run.

Comparison	Likelihood Ratio
MCMC *vs* Sparse SVD with stability selection	3.01
MCMC *vs* SVD	2.37

Table 6.1: Likelihood ratios for the three different model settings.

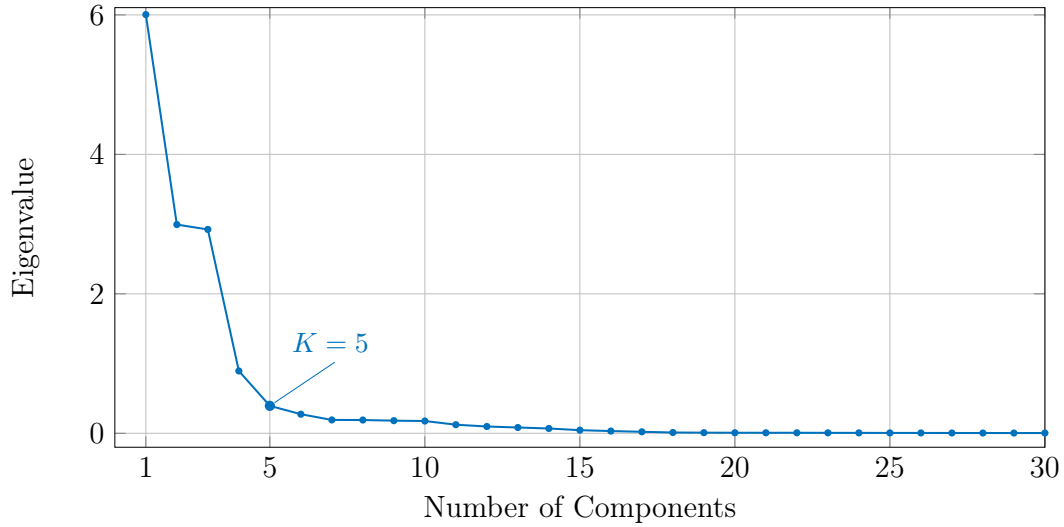


Figure 6.8: Scree plot for decay of singular values from the SVD of $\hat{\Lambda}$, in the interval $[0, 30]$, with the characteristic 'elbow' corresponding to the largest difference in magnitude.

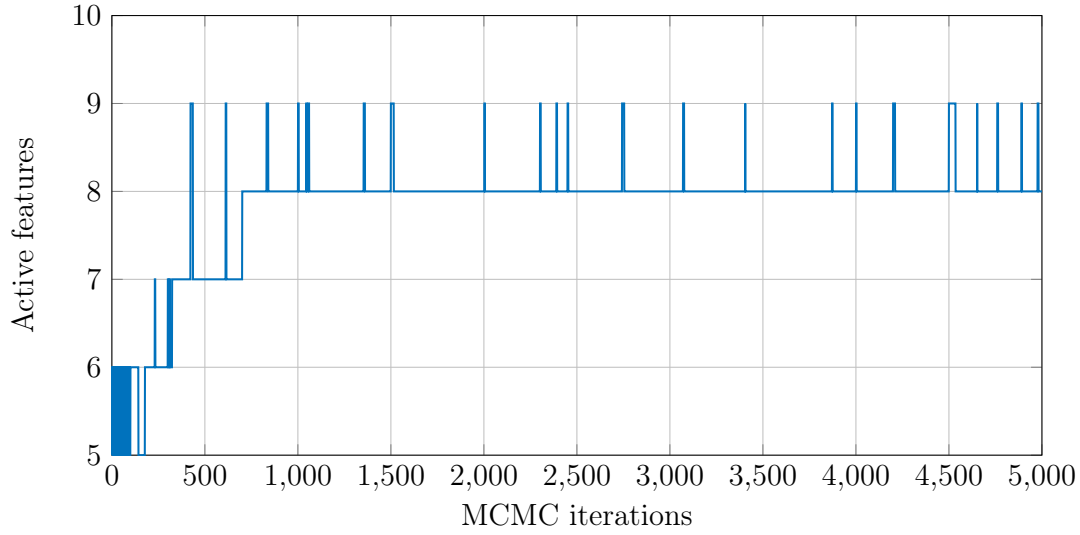


Figure 6.9: Number of active latent features during the MCMC run.

Chapter 7

Monitoring and Anomaly Detection

Anomaly detection is the problem of identifying subsets of data or patterns which do not conform with a posited model and thus can be seen as a hypothesis testing problem. In the context of this work, we are concerned with determining if the new authentication connections observed over some time period can be regarded as relatively normal - with respect to the model proposed in Chapter 6 - or whether they should be flagged as anomalous. In particular, the work presented in this chapter considers the following hypothesis testing:

H_0 : normal new edge behaviour,
H_1 : some departure from *normal* new edge behaviour.

How anomalous a test statistics is with respect to the null hypothesis is usually represented by a p-value, which will be a quantity of central interest in this chapter. Note that here, we will be only interested in one-sided p-values resulting from the test above: while traversing through the network, an intruder will always necessarily increase the number of new edges being created.

In this chapter, we will first assess the goodness-of-fit of the learned intensity model (6.1) and then, we will show how to construct an anomaly detection method based on such a model. Specifically, each new edge will be sequentially scored according to our model by creating a measure of surprise, i.e. a p-value, from the predictive distribution of each new observed edge. This sequence will

then be combined into a single time-varying score for each client in the network (alternatively, in future work interest could equally be focused on servers), such as a control chart (Nelson, 2001). The method will be finally demonstrated on real computer network data.

The remainder of this chapter is organised as follows. Section 7.1 introduces the posterior predictive p-values for each new edge observed, which will be later used for determining if observed new connections over some time period can be considered normal, and performs goodness-of-fit analyses based on such quantity; while Section 7.2 shows how the model proposed can be used to construct an anomaly detection method, which is then applied to real computer network data.

7.1 Monitoring

7.1.1 Predictive p-values

Before proceeding with anomaly detection, it is fundamental to evaluate the plausibility of the model proposed. In Chapter 6 we have already assessed the strength of the effect of each covariate included, for both the cluster and latent feature formulation. Here, for assessing fit, the quantity of interest is the posterior predictive distribution (6.3) of each new edge observed in $(\mathcal{T}', \mathcal{E}')$, from which we can calculate a corresponding p-value.

After observing time-ordered edges $(t'_1, e'_1), \ldots, (t'_{n-1}, e'_{n-1})$, let $e'_n = (x'_n, y'_n)$ be the nth new edge in $(\mathcal{T}', \mathcal{E}')$, observed at time t'_n. A p-value for this observation is given by the model probability of observing an edge no more probable than e'_n, denoted p_n. The calculation of p_n is given by

$$p_n = \frac{\sum_{(x,y)\notin G_{t'_n}} \lambda_{xy}(t'_n) \mathbb{1}_{[0,\lambda_{x'_n y'_n}(t'_n)]}\{\lambda_{xy}(t'_n)\}}{\sum_{(x,y)\notin G_{t'_n}} \lambda_{xy}(t'_n)}, \tag{7.1}$$

which quantifies the probability of observing an event as rare as the realised $e'_n = (x'_n, y'_n)$. Note that the numerator is the sum of intensities over all possible new

edges with intensity less than or equal to the edge observed.

Note that the p-values in (7.1) are generated from discrete random variables, and as a result they are conservative. This means that the evidence against the null hypothesis of normal behaviour will be understated. When such p-values are combined this effect is exacerbated, possibly resulting in extremely conservative p-values. Therefore, efficiently combining p-values presents important issues, and these will be further considered in Section 7.2.

7.1.2 Results

We perform a Kolmogorov-Smirnov (KS) test (Lewis, 1956) for the set of observed p-values (7.1), under the null hypothesis that they are distributed as uniform random variables on the unit interval, to test for model fit. The test is performed for both the cluster and the latent feature model formulations, with the purpose of comparison. In both cases, the p-values are generated from discrete random variables and so they are stochastically larger than uniform. Thus, to perform the test we use randomised p-values (Pearson, 1950), which are known to be marginally uniform on the unit interval if the model is correct. Figure 7.1 shows the empirical cumulative distribution of the observed p-values under both model formulations. We can observe that the distributions are approximately uniform in both cases, with the KS test yielding a p-value of 0.364 and 0.678, respectively for the cluster and the latent feature model.

Furthermore, inference quality and computational efficiency of both model formulations proposed in Chapter 6 are compared in terms of convergence diagnostics and algorithm run-times, as shown in Table 7.1. Results confirm that the latent feature model, under the IBP, outperforms the cluster model. Unfortunately, this comes at a cost of a higher running time of the sampler, which is dominated by the computation of the likelihood term: if only one element of the latent position vectors is changed, then the likelihood may be updated in $\mathcal{O}(K^2)$ time. The Indian Buffet Process provides a very flexible framework but inference might suffer from slow mixing or poor scalability, due to the very large number of possible

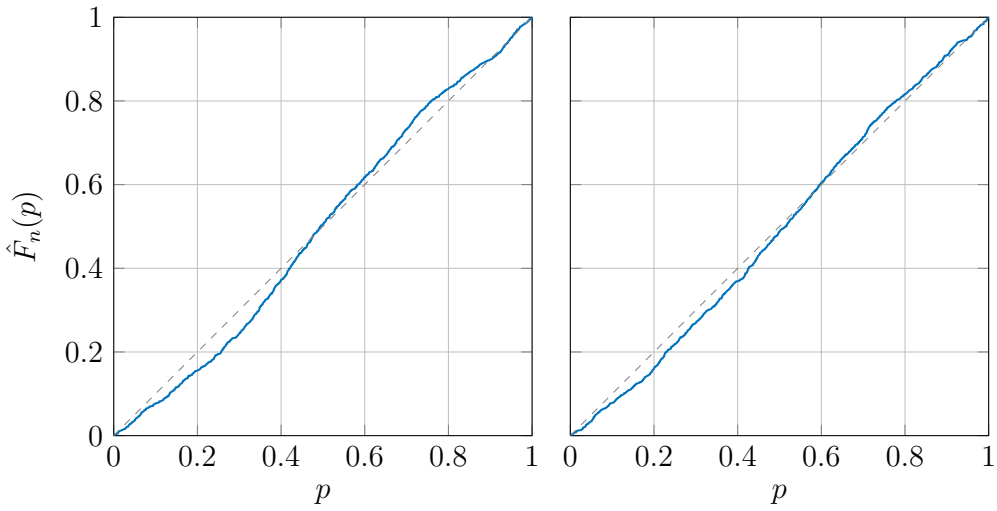


Figure 7.1: Empirical cumulative distribution of observed p-values under the cluster model (left) and the latent feature model (right), against the Uniform$(0, 1)$ cumulative distribution function.

configuration states.

Model formulation	**Log Likelihood**	**Iteration Time**
Cluster	-18804.34	81.4s
Latent-feature (IBP)	-18379.93	131.7s

Table 7.1: Out-of-training log likelihood under both formulations. All values are averages over the test pairs.

For convergence diagnostics, the log-likelihood for each MCMC iteration under both methods for the first data sample analysed is shown in Figure 7.2. The top panel shows the marginal likelihood for each iteration of the cluster formulation model, while the log-likelihood for the latent feature model under the IBP is shown in the bottom panel. For both models, the process appears stationary as the number of iterations increases. In particular, the plot suggests that the more complex latent feature model requires more time to complete burning-in.

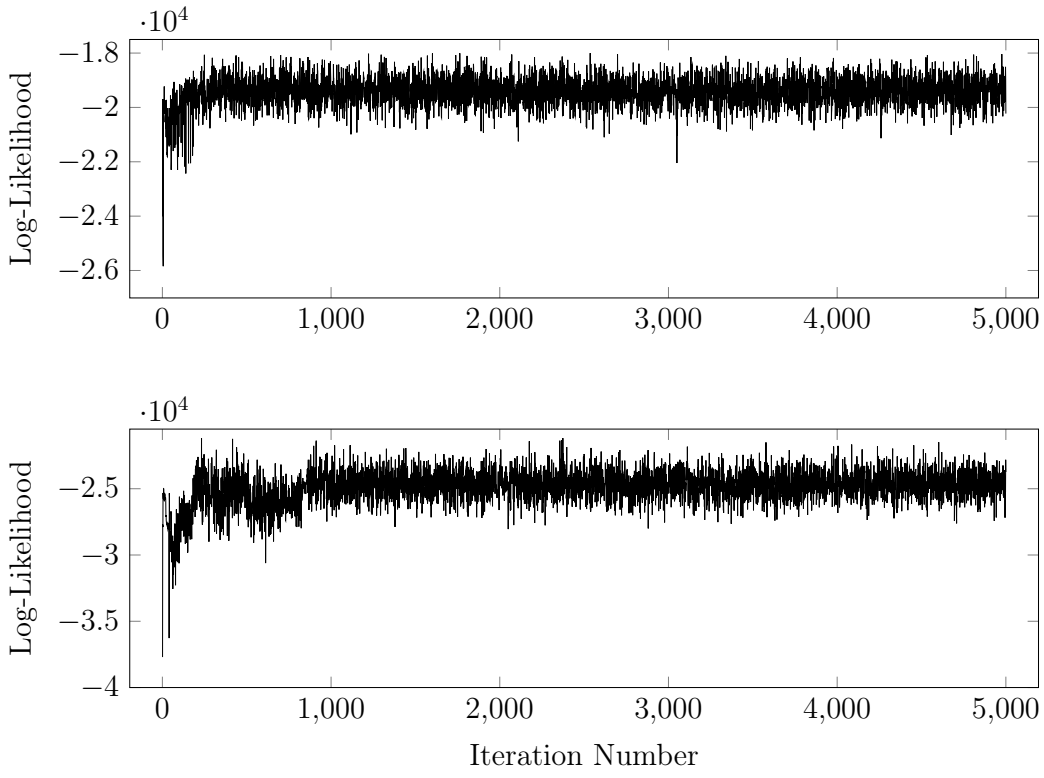


Figure 7.2: Log-likelihood vs. number of MCMC iterations, for the cluster formulation (top), and for the latent formulation (bottom).

7.2 Anomaly Detection

7.2.1 Combining p-values for ranking clients

To identify local deviations in the network, we are interested in finding anomalous behaviour over time for specific client computers. In this section, we describe a method to create time-varying anomaly score for each client in the network using the popular Fisher's method (Fisher, 1925) for combining p-values.

We utilise the sequence of predictive p-values $(p_n)_{n\geq 1}$ (7.1) presented in the previous section, derived from the sequence of new edges $(\mathcal{T}', \mathcal{E}') = ((T'_n)_{n\geq 1}, (E'_n)_{n\geq 1})$ arriving in the dynamic graph G_t. As we are interested in finding anomalous be-

haviour for specific client computers in the network, it is convenient to define $(\mathcal{T}'^x, \mathcal{Y}'^x) = ((T'^x_n)_{n\geq 1}, (Y'^x_n)_{n\geq 1})$ to be the subprocess of the new edge process $(\mathcal{T}', \mathcal{E}')$ for client x, corresponding to those indices n for which $\mathbb{1}_x(x_n) = 1$. Correspondently, let $(p^x_n)_{n\geq 1}$ be the subsequence of p-values of client x. We combine the sequence of p-values for client x over time using Fisher's method. This defines the following control chart

$$s_x(t) = \bar{\chi}^2_{2\{1+N^+_x(t)\}}\left(-2\sum_{n\geq 1}\mathbb{1}_{[0,t)}(t'^x_n)\log p^x_n\right), \tag{7.2}$$

where $N^+_x(t)$ is the outdegree of client x before time t and $\bar{\chi}^2_\nu(\cdot)$ is the survivor function of the chi-squared distribution with ν degrees of freedom. Under the null hypothesis of no attacks, and if the model of normal behaviour holds, the p-values, and by extension $s_x(t)$ for any $t \geq 0$, are approximately uniform on the unit interval. Clearly, extreme, small values of (7.2) represent anomalous behaviour of surprising new edge formation, and so here we are interested in anomaly scores such as

$$\inf_{t\geq 0}\ s_x(t). \tag{7.3}$$

In cyber-security applications, the distribution of (7.3) varies for different values of x, since some clients make many more new connections than others (see for example Figure 2.2). For this reason, tailored rejection regions for each client, based on their number of connections, must be calculated when building the control chart; this can be achieved through simple Monte Carlo estimation.

7.2.2 Results

The anomaly detection results presented in this section are restricted to the latent-feature model, since this has shown to lead to the highest predictive accuracy. Figure 7.4 shows the p-values (7.1) and control chart scores (7.2) (on the log-scale) for two of the known compromised clients (C17693, C19932) in the red team, and two randomly selected uninfected clients (C349, C586) that we have used as benchmark. In both infected cases, we can see some extreme values in the p-

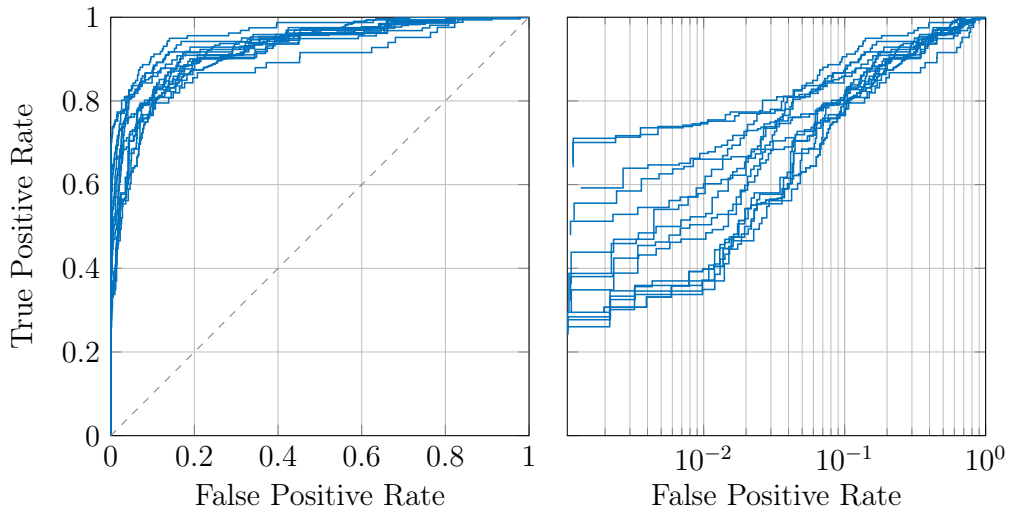


Figure 7.3: ROC curves for each client, for each sample repetition, shown on both linear (left) and log scales (right).

values and the control charts, leading to clear detection at the indicated 1% and 0.1% significance thresholds estimated through Monte Carlo as discussed above. In contrast, the control charts of the uninfected clients seem to indicate a normal behaviour, staying well above the thresholds for all times t.

Finally, Figure 7.3 shows the receiver operating characteristic (ROC) curves for each of the 15 sample repetitions for the sequence of p-values given by (7.1) and the scores obtained using (7.3). The results show a very high true positive to false positive ratio over low thresholds, although in the right panel of the figure, focusing on low false positive thresholds, we still see a number of true positives which we would ideally like to detect.

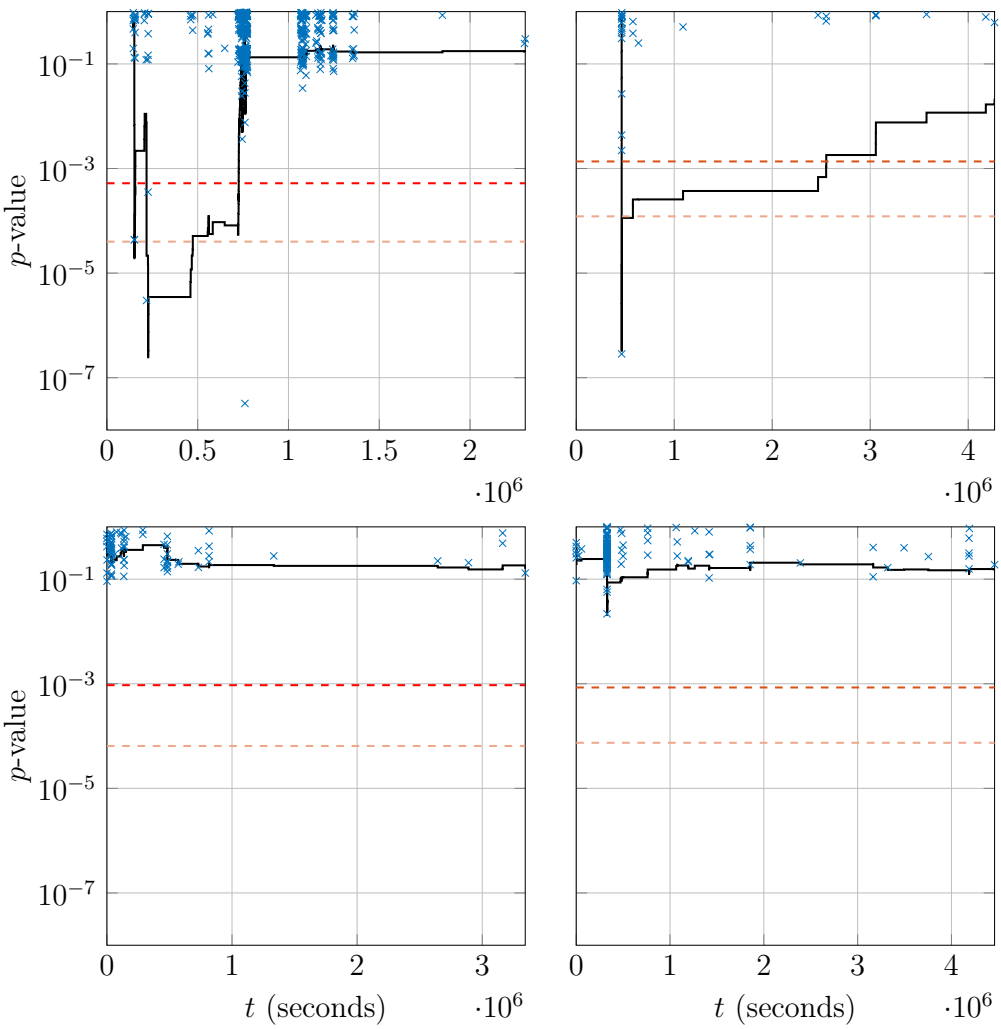


Figure 7.4: Observed p-values (×) over time and the corresponding control chart (——) for the two identified infected clients and two of uninfected clients in the red bulk data. Top left: C17693; Top right: C19932; Bottom left: C349; Bottom right: C586. Control chart thresholds at the 1% (- - -) and 0.1% (- - -) significance levels are shown for each client.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

This thesis has addressed the problem of modelling the arrival of new edges in a large computer network graph. When an intruder starts infecting a network machine he or she does not typically have information about which servers that machine usually connects to. As a result, the intruder will often make new edges and so study of new edge behaviour can be strongly informative about the presence of malicious activity. The work presented in this thesis was motivated by the need to understand such behaviour. In particular, Chapter 2 to Chapter 6 have been devoted to modelling the normal behaviour of new edges over time, while finally in Chapter 7 an anomaly detection method is constructed, based on the model learned in the previous chapters.

From a modelling point of view, two main aspects of new edges were investigated: first, we have focused on the rates at which clients form new edges, and second, we have focused on predicting the likely identity of each (client, server) potential new edge. In the first case, we performed variable selection for finding relevant covariates to be included in such a model; while in the second, we simultaneously modelled the hazard of observing new edges and any unobserved structural features inferred from historic connections. Examining network struc-

ture has proved to be fundamental for improving the predictive performance of our model. In both cases, inference was carried out in a Bayesian framework and so MCMC algorithms were used. To aid quick convergence, surrogate models for identifying initial, reliable cluster configurations were used to seed the sampler. When the size of the clustering problem analysed is so large, random initial allocations of the clustering or latent positions could have resulted in infeasibly long algorithm running times.

In particular, the main contribution of this work consisted of a robust Bayesian model which simultaneously addressed both the rates at which clients make new edges and any underlying latent network structure. We have further developed two different formulations for inferring the latent structure of the network. In both cases, the mechanism of new edge formation was modelled as a Bayesian Cox proportional hazards model and surrogate models were used to infer initial latent positions of clients and servers in the network. In the first formulation, a clustering model under hard-thresholding has been used, while in the second the flexibility of the model has been extended by letting each client and each server be associated with a (potentially unbounded) vector of latent features.

The methodology proposed has proved to be well suited for modelling new edges in a large network when demonstrated on real computer network data: results from both methods showed a considerable significance of the time-varying covariates on the predictive probability of observing a particular new connection and strongly indicate the positive impact of introducing a notion of similarity in the model. In particular, the most flexible, nonparametric latent feature approach, under the Indian Buffet Process prior, has led to the highest performance in terms of model fitting and predictive ability. Furthermore, our anomaly detection method has shown to drive encouraging performance in detecting compromised clients at low false positive rates. However, these results are not conclusive and adequately combining p-values might need to be taken into more consideration in future work.

8.2 Future Work

Throughout this thesis, the sequence of new edges generated from each unique (client, server) pair in the network has been employed to demonstrate our methods on real data. A first, straightforward extension may simply consider replicating the analyses presented here on the sequence of new (user domain, server) edges, in place of the (client, server) pairs. In this way, each client computer and user ID pairing would be treated as a separate network entity. As an example, figure 8.1 shows a graph of connections in the red team data from each user domain to each destination server. We can notice that users with a relatively high degree seem to connect to servers in disconnected graph components. This might suggest that our clustering method, under both formulations proposed, could easily separate the user domains into well-defined clusters, thus enhancing the model prediction performance.

From a modelling perspective, an avenue of research could involve directly capturing the power-law phenomenon of computer networks as a part of the modelling process. For instance, Pitman and Yor (1997) have proposed a generalisation of the Dirichlet process which include power-law properties. Following a similar procedure, Teh and Gorur (2009) have proposed a three-parameter generalisation of the IBP, which allows to capture power-law behaviour in the context of latent feature models. Both these frameworks could act as interesting comparisons with our model, where power-law behaviour is not part of the modelling process but rather, it has been incorporated in the model by using several time-varying degree covariates. However, these methods can be computationally very demanding; to reduce the computational times, more efficient MCMC schemes, such as split-merge sampling (Meeds et al., 2006), could be explored. Splitting and merging features in the binary matrices of latent features would signify performing non-incremental moves. Such moves can produce significative changes in the configuration state just in a single iteration and could thus help the sampler exploring the parameter space more efficiently.

Furthermore, computer network traffic data typically exhibit seasonal patterns

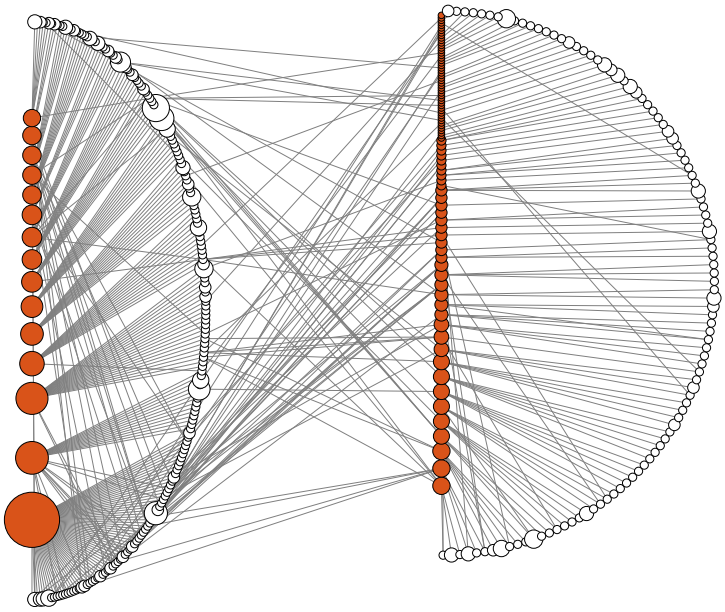


Figure 8.1: A graph showing connections from user domains to server computers involving red team activity.

since human users are more likely to be active on weekdays during the day time than they are at night time or at the weekend; thus a robust client model of normal network behaviour should also aim at identifying this key feature. In the context of this work, this would signify capturing seasonality through the client baseline hazard function $r_x(t)$ for the client model (5.2) in Chapter 5. This would aid detecting anomalous behaviour for each client x, who is forming bursts of new edges at particular times of day.

Finally, more attention should be addressed to efficiently combining p-values, with the purpose of improving the performance of our anomaly method. One of the main issues encountered in cyber-security applications is the imbalance between the vast amount of p-values generated under the null distribution and the tiny proportion of activity which corresponds to cyber threat. As a result, there are only few, very tiny significant p-values and unfortunately Fisher's method is known to

fail to some degree to adequately address this. For this reason, when building the control chart presented in Chapter 7, the choice of the construction of the control chart should be adapted according to the precise target of the anomaly detection scheme. A diverse range of p-value combination methods appear in the literature, each with different statistical properties (e.g. Stouffer (1949); Pearson (1933)), which have not been considered here. A natural extension of the anomaly detection work presented here would be to examine the performance of our method under different combiners. For instance, mid-p-values (Lancaster, 1952; Rubin-Delanchy et al., 2018) have been shown to perform well in cyber-security applications (Turcotte et al., 2016). In addition, previous studies of computer network data have indicated evidence of higher correlations between the timings of the anomalous events for the true positives rather than the false positives (Turcotte et al., 2016). Thus, further extensions could focus on modelling the correlation in timings of the anomalous events between client computers and server computers, which could in turn aid reducing false alarms.

Bibliography

Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D. and Levine, A. (1999), 'Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays', *Proceedings of the National Academy of Sciences* **96**(12), 6745–6750.

Banfield, J. D. and Raftery, A. E. (1993), 'Model-based gaussian and non-gaussian clustering', *Biometrics* **49**, 803–821.

Bernardo, J. and Smith, A. (2007), *Bayesian Theory*, John Wiley & Sons, New York, NY.

Bock, H. H. (1996), 'Probabilistic models in cluster analysis', *Computational Statistics & Data Analysis* **23**(1), 5–28.

Booth, J., Casella, G. and Hobert, J. (2008), 'Clustering using objective functions and stochastic search', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **70**(1), 119–139.

Broman, K. W. and Speed, T. P. (2002), 'A model selection approach for identification of quantitative trait loci in experimental crosses', *Journal of the Royal Statistical Society, Series B (Statistical Methodology)* **64**, 641–656.

Busygin, S., Prokopyev, O. A. and Pardalos, P. M. (2008), 'Biclustering in data mining', *Computers & Operations Research* **35**(9), 2964–2987.

Cahill, M. H., Lambert, D., Pinheiro, J. C. and Sun, D. X. (2002), *Handbook of Massive Data Sets.*, Kluwer Academic Publishers, chapter Detecting Fraud in the Real World, pp. 911–929.

Chandola, V., Banerjee, A. and Kumar, V. (2009), 'Anomaly detection: A survey', *ACM Computing Surveys (CSUR)* **41**(3), 15.

Cheng, Y. and Church, G. M. (2000), Biclustering of expression data, *in* 'Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology', AAAI Press, pp. 93–103.

Chipman, H., George, E. and McCulloch, R. (1998), 'Bayesian cart estimation (with discussion and rejoinder)', *Journal of the American Statistical Association* **93**, 935–960.

Cho, H., Dhillon, I., Guan, Y. and Sra, S. (2004), Minimum sum-squared residue co-clustering of gene expression data., *in* 'Proceedings of the Fourth SIAM International Conference on Data Mining', pp. 114–125.

Clyde, M. (1999), 'Discussion of Bayesian model averaging: A tutorial with discussion by J. A. Hoeting, D. Madigan and A. E. Raftery and C. T. Volinsky', *Statistical Science* **14**, 401–404.

Cox, D. R. (1972), 'Regression models and life-tables (with discussion)', *Journal of the Royal Statistical Society, Series B (Statistical Methodology)* **34**, 187–220.

Denison, D., Holmes, C., Mallick, B. and Smith, A. (2002), *Bayesian methods for nonlinear classification and regression*, John Wiley & Sons, New York, NY.

Dhillon, I. S. (2001), Co-clustering documents and words using bipartite spectral graph partitioning, *in* 'Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', ACM, pp. 269–274.

Dhillon, I. S., Guan, Y. and Kulis, B. (2004), Kernel k-means: Spectral clustering and normalized cuts, *in* 'Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '04, ACM, pp. 551–556.

Duda, R. O. and Hart, P. E. (1973), *Pattern Classification and Scene Analysis*, John Willey & Sons, New Yotk.

Eckart, C. and Young, G. (1936), 'The approximation of a matrix by another one of lower rank', *Psychometrika* **1**, 211–218.

Fisher, R. (1925), *Statistical methods for research workers*, Edinburgh Oliver & Boyd.

Fisher, R. A. (1929), 'Tests of significance in harmonic analysis', *Proceedings of the Royal Society of London. Series A* **125**(796), 54–59.

Fowler, A. and Heard, N. (2012), 'On two-way Bayesian agglomerative clustering of gene expression data', *Statistical Analysis and Data Mining* **5**, 463–476.

Fraley, C. and Raftery, A. E. (2002), 'Model-based clustering discriminant analysis, and density estimation', *Journal of the American Statistical Association* **97**, 611–631.

Friedberg, I., Skopik, F., Settanni, G. and Fiedler, R. (2015), 'Combating advanced persistent threats: From network event correlation to incident detection.', *Computers & Security* **48**, 35–57.

Friel, N. and Wyse, J. (2012), 'Bayesian interpolation estimating the evidence–a review', *Statistica Neerlandica* **66**(3), 288–308.

Gelfand, A. and Smith, A. F. M. (1990), 'Sampling-based approaches to calculating marginal densities', *Journal of the American Statistical Association* **410**(85), 398—409.

Ghahramani, Z. and Griffiths (2005), Infinite latent feature models and the indian buffet process, *in* 'Advances in Neural Information Processing Systems 18', MIT Press, pp. 475–482.

Ghahramani, Z., Griffiths, T. and Sollich, P. (2007), *Bayesian nonparametric latent feature models.*, Oxford University Press, pp. 201–225.

Gower, J. C. and Ross, G. J. S. (1969), 'Minimum spanning trees and single linkage cluster analysis', *Journal of the Royal Statistical Society. Series C (Applied Statistics)* **18**(1), 54–64.

Green, P. J. (1995), 'Reversible jump Markov chain Monte Carlo computation and bayesian model determination', *Biometrika* **82**, 711–732.

Halliday, D. and Rosenberg, J. (1999), *Time and frequency domain analysis of spike train and time series data*, U. Windhorst and H. Johansson, Eds., Springer Berlin Heidelberg.

Hartigan, J. (1975), *Clustering Algorithms*, John Wiley & Sons, New York, NY.

Hartigan, J. A. (1972), 'Direct clustering of a data matrix', *Journal of American Statistical Association* **67**, 123–129.

Hartigan, J. A. and Wong, M. A. (1979), 'Algorithm AS 136: A K-means clustering algorithm', *Applied Statistics* pp. 100–108.

Hastings, W. (1970), 'Monte Carlo sampling methods using Markov chains and their applications', *Biometrika* **57**(1), 97–109.

Heard, N. (2011), 'Iterative reclassification in agglomerative clustering', *Journal of Computational and Graphical Statistics* **20**(4), 920–936.

Heard, N. A., Holmes, C. C. and Stephens, D. A. (2006), 'A quantitative study of gene regulation involved in the immune response of anopheline mosquitoes: An application of bayesian hierarchical clustering of curves', *Journal of the American Statistical Association* **101**, 18–29.

Heard, N. A. and Turcotte, M. J. (2014), *Monitoring a Device in a Communication Network*, Imperial College Press, chapter 6, pp. 151–188.

Heard, N., Rubin-Delanchy, P. and Lawson, D. (2014), Filtering automated polling traffic in computer network flow data, IEEE, pp. 268–271.

Heard, N., Weston, D., Platanioti, K. and Hand, D. (2010), 'Bayesian anomaly detection methods for social networks', *Annals of Applied Statistics* **4**, 645–662.

Heller, K. and Ghahramani, Z. (2005), Bayesian hierarchical clustering, *in* 'Proceedings of the 22nd International Conference on Machine learning'.

Hjort, N. L. (1990), 'Nonparametric Bayes estimators based on beta processes in models for life history data', *The Annals of Statistics* **18**(3), 1259–1294.

Hoeting, J. A., Madigan, D., Raftery, A. E. and Volinsky, C. T. (1999), 'Bayesian model averaging: A tutorial with discussion', *Statistical Science* **14**, 382–417.

Hoff, P. D. (2002), 'Latent space approaches to social network analysis', *Journal of the American Statistical Association* **97**(460), 1090—1098.

Hoff, P. D. (2007), 'Model averaging and dimension selection for the singular value decomposition', *Journal of the American Statistical Association* **102**(478).

Hoff, P. D. (2009), 'Multiplicative latent factor models for description and prediction of social networks', *Computational and Mathematical Organization Theory* **15**(4), 261–272.

Hoff, P. D., Raftery, A. and Handcock, M. S. (2005), 'Bilinear mixed-effects models for dyadic data', *Journal of the American Statistical Association* **100**(469), 286–295.

Horn, C. and Willett, R. (2011), Online anomaly detection with expert system feedback in social networks., *in* 'ICASSP', IEEE, pp. 1936–1939.

Idé, T. and Kashima, H. (2004), Eigenspace-based anomaly detection in computer systems, *in* 'Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04', ACM Press, pp. 440–449.

Jeffreys, H. (1961), *Theory of Probability (3rd Edition)*, Oxford University Press, New York.

Kass, R. E. and Raftery, A. E. (1995), 'Bayes factors', *Journal of the American Statistical Association* **90**(430), 773–795.

Kent, A. D. (2014), 'User-computer authentication associations in time', Los Alamos National Laboratory.

Kent, A. D. (2015*a*), 'Comprehensive, multi-source cyber-security events', Los Alamos National Laboratory.

Kent, A. D. (2015*b*), Cyber security data sources for dynamic network research, *in* 'Dynamic Networks in Cybersecurity', Imperial College Press.

Kim, S., M., T. and Vannucci, M. (2006), 'Variable selection in clustering via dirichlet process mixture models', *Biometrika* **93**, 877–893.

Kluger, Y., Basri, R., Chang, J. T. and Gerstein, M. (2003), 'Spectral Biclustering of Microarray Data: Coclustering Genes and Conditions', *Genome Research* **13**(4), 703–716.

Lancaster, H. (1952), 'Statistical control of counting experiments', *Biometrika* **39**, 419—-422.

Lee, M., Shen, H., Huang, J. Z. and Marron, J. S. (2010), 'Biclustering via sparse singular value decomposition', *Biometrics* **66**(4), 1087–1095.

Lewis, P. A. W. (1956), 'Some results on tests for Poisson processes', *Biometrika* **52**(1-2), 67–77.

Li, L., Guo, Y., Wu, W., Shi, Y., Cheng, J. and Tao, S. (2012), 'A comparison and evaluation of five biclustering algorithms by quantifying goodness of biclusters for gene expression data.', *BioData Mining* **5**, 8.

MacKay, D. (1991), 'Bayesian interpolation', *Neural Computation* **4**, 415–447.

MacQueen, J. (1967), Some methods for classification and analysis of multivariate observations, *in* 'Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics', University of California Press, pp. 281–297.

McCullagh, P. (1984), 'Generalized linear models', *European Journal of Operational Research* **16**(3), 285–292.

McLachlan, G. and Basford, K. (1988), *Mixture Models: Inference and Applications to Clustering*, Marcel Dekker, New York.

Medvedovic, M., Yeung, K. and Bumgarner, R. (2004), 'Bayesian mixture model based clustering of replicated microarray data', *Bionformatics* **20**, 1222–1232.

Meeds, E., Ghahramani, Z., Neal, R. and Roweis, S. (2006), Modeling dyadic data with binary latent factors, *in* 'Proceedings of the 19th International Conference on Neural Information Processing Systems', NIPS'06, Cambridge, MA, USA, pp. 977–984.

Meinshausen, N. and Bühlmann, P. (2010), 'Stability selection', *Journal of the Royal Statistical Society, Series B (Statistical Methodology)* **72**, 417–473.

Metelli, S. and Heard, N. (2014), Modelling new edge formation in a computer network through bayesian variable selection, *in* 'Joint Intelligence and Security Informatics Conference (JISIC), 2014 European', IEEE, pp. 272–275.

Metelli, S. and Heard, N. (2016), Model-based clustering and new edge modelling in a large computer network, *in* 'IEEE International Conference on Intelligence and Security Informatics (ISI), 2016', IEEE, pp. 91–96.

Metelli, S. and Heard, N. (2018), On Bayesian new edge modelling and anomaly detection in computer networks. Submitted to the Annals of Applied Statistics.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. (1953), 'Equation of state calculations by fast computing machines', *The Journal of Chemical Physics* **21**(6), 1087–1092.

Meuwissen, T. H. E. and Goddard, M. E. (2004), 'Mapping multiple qtl using linkage disequilibrium and linkage analysis information and multitrait data', *Genetics Selection Evolution* **36**, 261–279.

Miller, A. (2002), *Subset Selection in Regression.*, Chapman & Hall/CRC.

Miller, K., Jordan, M. and Griffiths, T. L. (2009), Nonparametric latent feature models for link prediction, *in* 'Advances in Neural Information Processing Systems 22', pp. 1276–1284.

Neal, R. (2003), 'Slice sampling', *Annals of Statistics* **31**(3), 705–767.

Neil, J., Storlie, C., Hash, C., Brugh, A. and M., F. (2013), 'Scan statistics for the online detection of locally anomalous subgraphs', *Technometrics* **55**(4), 403–414.

Nelson, P. R. (2001), 'Book review: Normality and the process behavior chart by Donald J. Wheeler', *Technometrics* **43**(3), 371–371.

Noble, C. C. and Cook, D. J. (2003), Graph-based anomaly detection., *in* L. Getoor, T. E. Senator, P. M. Domingos and C. Faloutsos, eds, 'KDD', ACM, pp. 631–636.

Nowicki, K. and Snijders, T. A. B. (2001), 'Estimation and prediction for stochastic blockstructures', *Journal of the American Statistical Association* **96**(455), 1077–1087.

Park, Y., Priebe, C. E. and Youssef, A. (2013), 'Anomaly detection in time series of graphs using fusion of graph invariants.', *Journal of Selected Topics in Signal Processing* **7**(1), 67–75.

Patcha, A. and Park, J. (2007), 'An overview of anomaly detection techniques: Existing solutions and latest technological trends', *Computer Networks* **51**(12), 3448–3470.

Pearson, E. S. (1950), 'On questions raised by the combination of tests based on discontinuous distributions', *Biometrika* **37**(3/4), 383–398.

Pearson, K. (1933), 'On a method of determining whether a sample of size n supposed to have been drawn from a parent population having a known probability integral has probably been drawn at random.', *Biometrika* **25**(3-4), 379–410.

Pitman, J. (2002), Combinatorial stochastic processes., Technical Report 621, Dept. Statistics, U.C. Berkeley. Lecture notes for St. Flour course.

Pitman, J. and Yor, M. (1997), 'The two-parameter poisson-dirichlet distribution derived from a stable subordinator', *Annals of Probability* **25**, 855–900.

Pothen, A., Simon, H. D. and Liou, K. (1990), 'Partitioning sparse matrices with eigenvectors of graphs', *SIAM Journal on Matrix Analysis and Applications* **11**(3), 430–452.

Priebe, C. E., Conroy, J. M., Marchette, D. J. and Park, Y. (2005), Scan statistics on enron graphs, SDM 05.

Qin, Z. S. (2006), 'Clustering microarray gene expression data using weighted chinese restaurant process', *Bionformatics* **22**, 1988–1997.

Raftery, A. (1988), Approximate bayes factors for generalized linear models, Technical Report 121, Department of Statistics, University of Washington.

Raftery, A. E., Madigan, D. and Volinsky, C. T. (1995), 'Accounting for model uncertainty in survival analysis improves predictive performance (with discussion)', *In Bayesian Statistics 5, Bernardo JM, Berger JO, Dawid AP, Smith FM. (eds)* .

Raftery, A., Madigan, D. and Hoeting, J. (1997), 'Bayesian model averaging for regression models', *Journal of the American Statistical Association* **92**, 179–191.

Richardson, S. and Green, P. (1997), 'On Bayesian analysis of mixtures with an unknown number of components', *Journal of the Royal Statistical Society, Series B (Statistical Methodology)* **59**(4), 731–792.

Rissanen, J. (1989), *Stochastic Complexity in Statistical Inquery*, World Scientific.

Robert, C. and Casella, G. (2004), *Monte Carlo Statistical Methods*, Springer-Verlag, 2nd edition, New York.

Rubin-Delanchy, P. T. G., Heard, N. A. and Lawson, D. J. (2018), Meta-analysis of mid-p-values: some new results based on the convex order. Journal of the American Statistical Association, to appear.

Schwarz, G. (1978), 'Estimating the dimension of a model', *The Annals of Statistics* **6**(2), 461–464.

Sharpnack, J., Rinaldo, A. and Singh, A. (2012), 'Changepoint detection over graphs with the spectral scan statistic', *arXiv/1206.0773* .

Sill, M., Kaiser, S., Benner, A. and Kopp-Schneider, A. (2011), 'Robust biclustering by sparse singular value decomposition incorporating stability selection', *Bioinformatics* **27**(15), 2089–2097.

Sillanpää, M. and Corander, J. (2002), 'Model choice in gene mapping: what and why', *Trends in Genetics* **18**, 301–307.

Sillanpää, M., Gasbarra, D. and Arjas, E. (2004), 'Comment on "on the metropolis hastings acceptance probability to add or drop a quantitative trait locus in markov chain monte carlo-based bayesian analyses', *Genetics* **167**, 1037.

Stouffer, S. (1949), *The American soldier. Vol. 1: Adjustment during army life*, Studies in social psychology in World War II, Princeton University Press.

Strehl, A. (2002), Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining, PhD thesis, The University of Texas at Austin.

Tanay, A., Sharan, R. and Shamir, R. (2002), 'Discovering statistically significant biclusters in gene expression data', *Bioinformatics* **18**(90001), 136–144.

Teh, Y. W. and Gorur, D. (2009), Indian buffet processes with power-law behavior, *in* 'Advances in Neural Information Processing Systems 22', Curran Associates, Inc., pp. 1838–1846.

Tibshirani, R. (1996), 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society, Series B (Statistical Methodology)* **58**, 267–288.

Turcotte, M. J., Heard, N. A. and Kent, A. D. (2016), Modelling user behaviour in a network using computer event logs, *in* 'Dynamic Networks and Cyber-Security', Imperial College Press, London, UK, pp. 67–87.

Turcotte, M. J., Heard, N. A. and Neil, J. (2014), Detecting localised anomalous behaviour in a computer network., *in* 'Advances in Intelligent Data Analysis XIII - 13th International Symposium', Springer, pp. 321–332.

Turner, H., Bailey, T. and Krzanowski, W. (2005), 'Improved biclustering of microarray data demonstrated through systematic performance tests', *Computational Statistics & Data Analysis* **48**(2), 235–254.

Valko, M. (2011), Adaptive graph-based algorithms for conditional anomaly detection and semi-supervised learning, PhD thesis, University of Pittsburgh.

Viallefont, V., Raftery, A. and Richardson, S. (2001), 'Variable selection and bayesian model averaging in case-control studies', *Statistics in Medicine* **20**(21), 3215–3230.

Von Luxburg, U. (2007), 'A tutorial on spectral clustering', *Statistics and Computing* **17**(4), 395–416.

Ward, J. (1963), 'Hierarchical grouping to optimize an objective function.', *Journal of the American Statistical Association* **58**, 236–244.

Weiss, Y. (1999), Segmentation using eigenvectors: a unifying view, *in* 'Proceedings of the IEEE International Conference on Computer Vision', AAAI Press, pp. 975–982.

Wichert, S., Fokianos, K. and Strimmer, K. (2004), 'Identifying periodically expressed transcripts in microarray time series data', *Bioinformatics* **20**(1), 5–20.

Xu, S. (2003), 'Estimating polygenic effects using markers of the entire genome', *Genetics* **163**, 789–801.

Zhang, Z. and Jordan, M. I. (2008), 'Multiway spectral clustering: A margin-based perspective', *Statistical Science* **23**, 383–403.

Appendix A

Calculation of the marginal posterior distribution (4.6)

We assume that $\mathbb{P}(A_{xy} = 1 | x \in C_l, y \in S_m) = \theta_{lm}$, with $\theta_{lm} \sim \text{Beta}(a, b)$, i.e.

$$f(\theta_{lm}) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} {\theta_{lm}}^{a-1}(1-\theta_{lm})^{b-1}.$$

Then

$$\begin{aligned}
L(A|\mathbb{C}, \mathbb{S}, \boldsymbol{\theta}) f(\boldsymbol{\theta}) &= \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \prod_{l=1}^{L} \prod_{x \in C_l} \prod_{m=1}^{M} \prod_{y \in S_m} \theta_{lm}^{A_{xy}} (1-\theta_{lm})^{A_{xy}} \theta_{lm}^{a-1} (1-\theta_{lm})^{b-1} = \\
&= \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \prod_{l=1}^{L} \prod_{m=1}^{M} \theta_{lm}^{a-1} (1-\theta_{lm})^{b-1} \prod_{x \in C_l} \prod_{y \in S_m} \theta_{lm}^{A_{xy}} (1-\theta_{lm})^{A_{xy}} \\
&= \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \prod_{l=1}^{L} \prod_{m=1}^{M} \theta_{lm}^{a-1} (1-\theta_{lm})^{b-1} \theta_{lm}^{\sum_{x \in C_l} \sum_{y \in S_m} A_{xy}} (1-\theta_{lm})^{\sum_{x \in C_l} \sum_{y \in S_m} 1 - \sum_{x \in C_l} \sum_{y \in S_m} A_{xy}} \\
&= \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \prod_{l=1}^{L} \prod_{m=1}^{M} \theta_{lm}^{a-1+M_{lm}} (1-\theta_{lm})^{b-1+n_l s_m - M_{lm}},
\end{aligned}$$

where

$$M_{lm} = \sum_{x \in C_l} \sum_{y \in S_m} A_{xy}, \qquad n_l = \sum_{x \in C_l} 1, \qquad s_m = \sum_{y \in S_m} 1.$$

Recalling that

$$\int_0^1 \theta^a (1-\theta)^b \mathrm{d}\theta = \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)},$$

we easily find

$$\int L(A|\mathbb{C}, \mathbb{S}, \boldsymbol{\theta}) f(\boldsymbol{\theta}) d\boldsymbol{\theta} = \prod_{l=1}^{L} \prod_{m=1}^{M} \frac{\Gamma(a+b)\Gamma(a+M_{lm})\Gamma(b-M_{lm}+n_l s_m)}{\Gamma(a)\Gamma(b)\Gamma(a+b+n_l s_m)}.$$

Clearly, the same result holds for (4.3), with $s_m = 1$.

Appendix B

IBP as limit of a Beta-Bernoulli model

B.1 A finite feature model

Suppose we have $|X|$ clients (or equivalently $|Y|$ servers) and K features. As before, the possession of feature k by client x is represented by Δ_{xk}, which belongs to the binary feature matrix Δ_U, with independently generated features. For each client, let $\pi_k \in [0,1]$ be the probability of possessing feature k and let $\pi = \{\pi_1, \ldots, \pi_K\}$, then the probability of Δ_U conditional on π is given by

$$\mathbb{P}(\Delta_U|\pi) = \prod_{k=1}^{K}\prod_{x=1}^{|X|}\mathbb{P}(\Delta_{xk}|\pi_k) = \prod_{k=1}^{K}{\pi_k}^{m_k}(1-\pi_k)^{|X|-m_k}, \tag{B.1}$$

where $m_k = \sum_{x=1}^{|X|}\Delta_{xk}$ is the number of clients with feature k active. The conjugate prior on π for this model is the beta distribution and thus independent Beta(r,s) distributions are assumed for each π_k:

$$p(\pi_k) = \frac{\pi_k^{r-1}(1-\pi_k^{s-1})}{B(r,s)}, \tag{B.2}$$

where $B(r,s)$ is the Beta function. Exploiting the recursive property of the Gamma

function and taking $r = \frac{\theta}{K}$ and $s = 1$ we have:

$$B(\frac{\theta}{K}, 1) = \frac{\Gamma(\frac{\theta}{K})}{\Gamma(1 + \frac{\theta}{K})} = \frac{K}{\theta}. \tag{B.3}$$

This defines the following probability model:

$$\begin{aligned} \pi_k|\theta &\sim B(\frac{\theta}{K}, 1), \\ \Delta_{xk}|\pi_k &\sim \text{Bernoulli}(\pi_k). \end{aligned} \tag{B.4}$$

By integrating over all possible values for π_k and exploiting the conjugacy between the binomial and beta distributions, we can obtain the marginal distribution of Δ_U as follow

$$\begin{aligned} \mathbb{P}(\Delta_U) &= \prod_{k=1}^{K} \int \big(\prod_{x=1}^{|X|} P(\Delta_{xk}|\pi_k)\big) p(\pi_k)\, d\pi_k \\ &= \prod_{k=1}^{K} \frac{B(m_k + \frac{\theta}{K}, |X| - m_k + 1)}{B(\frac{\theta}{K}, 1)} \\ &= \prod_{k=1}^{K} \frac{\frac{\theta}{K}\Gamma(m_k + \frac{\theta}{K})\Gamma(|X| - m_k + 1)}{\Gamma(|X| + 1 + \frac{\theta}{K})} \end{aligned} \tag{B.5}$$

This corresponds to a Beta-Binomial model. We can notice that this distribution is exchangeable, since it only depends on m_k.

B.2 Taking the infinite limit

The distribution on infinite binary matrices can be derived from the finite model described in Section B.1 by taking the limit as $K \to \infty$ and finding equivalence classes of binary matrices. Recalling that permuting the columns does not affect the model, it is convenient to represent the model by using a canonical ordering such that all Δ_U matrices that are the same up to column-permutations are equivalent. Ghahramani and Griffiths (2005) define a canonical representation called the

left-ordered form of Δ_U, denoted as $[\Delta_U] = lof(\Delta_U)$. Here, the binary sequences of $\Delta_{xk} = 0$ and $\Delta_{xk} = 1$ are taken for each column (referred to as a history h), treating the first customer as the most significant bit, and converts the binary sequence to a number. Thus, each column, or feature, receives a single value. Then, the columns are ordered by descending value. It is then fundamental to calculate the cardinality of $[\Delta_U]$, since this will give us the number of matrices that map to the same left-ordered form. If Δ_U contains identical columns, the number of existing matrices in $[\Delta_U]$ is reduced, as re-ordering will produce the same matrix. Specifically, the cardinality of $[\Delta_U]$ is given by

$$\binom{K}{K_0 \dots K_{2^{|X|}-1}} = \frac{K!}{\prod_{h=0}^{2^{|X|}-1} K_h!} \tag{B.6}$$

with K_h is the count of the number of columns which have full history h. Under the distribution in (B.5), the distribution of a *lof* equivalence class $[\Delta_U]$ is

$$\mathbb{P}([\Delta_U]) = \sum_{\Delta_U \in [\Delta_U]} \mathbb{P}(\Delta_U) = \frac{K!}{\prod_{h=0}^{2^{|X|}-1} K_h!} \prod_{k=1}^{K} \frac{\frac{\theta}{K}\Gamma(m_k + \frac{\theta}{K})\Gamma(|X| - m_k + 1)}{\Gamma(|X| + 1 + \frac{\theta}{K})}. \tag{B.7}$$

For computing the limit of (B.7) as $K \to \infty$, it is convenient to break up the features in Δ_U into two parts: the features for which $m_k = 0$ and the features for which $m_k > 0$. Re-ordering the columns so that $m_k > 0$ if $k \leq K+$, and $m_k = 0$ otherwise, the product in (B.7) becomes

$$\begin{aligned}
&\prod_{k=1}^{K} \frac{\frac{\theta}{K}\Gamma(m_k + \frac{\theta}{K})\Gamma(|X| - m_k + 1)}{\Gamma(|X| + 1 + \frac{\theta}{K})} \\
&= \Big(\frac{\frac{\theta}{K}\Gamma(\frac{\theta}{K})\Gamma(|X| + 1)}{\Gamma(|X| + 1 + \frac{\theta}{K})}\Big)^{K-K+} \prod_{k=1}^{K+} \frac{\frac{\theta}{K}\Gamma(m_k + \frac{\theta}{K})\Gamma(|X| - m_k + 1)}{\Gamma(|X| + 1 + \frac{\theta}{K})} \\
&= \Big(\frac{\frac{\theta}{K}\Gamma(\frac{\theta}{K})\Gamma(|X| + 1)}{\Gamma(|X| + 1 + \frac{\theta}{K})}\Big)^{K} \prod_{k=1}^{K+} \frac{\frac{\theta}{K}\Gamma(m_k + \frac{\theta}{K})\Gamma(|X| - m_k + 1)}{\Gamma(\frac{\theta}{K})\Gamma(|X| + 1)} \\
&= \Big(\frac{|X|!}{\prod_{q=1}^{|X|}(q + \frac{\theta}{K})}\Big)^{K} \Big(\frac{\theta}{K}\Big)^{K+} \prod_{k=1}^{K+} \frac{(|X| - m_k)! \prod_{q=1}^{m_k - 1}(q + \frac{\theta}{K})}{|X|!}
\end{aligned} \tag{B.8}$$

Finally, substituting (B.8) into (B.7) and rearranging we have

$$\begin{aligned}\lim_{K\to\infty} \quad & \frac{\theta^{K+}}{\prod_{h=0}^{2^{|X|}-1} K_h!} \cdot \frac{K!}{K_0! K^{K+}} \cdot \Big(\frac{|X|!}{\prod_{q=1}^{|X|}(q+\frac{\theta}{K})}\Big)^K \quad \cdot \prod_{k=1}^{K+} \frac{(|X|-m_k)! \prod_{q=1}^{m_k-1}(q+\frac{\theta}{K})}{|X|!} \\ = \quad & \frac{\theta^{K+}}{\prod_{h=0}^{2^{|X|}-1} K_h!} \cdot \quad 1 \quad \cdot \exp\{-\theta H_{|X|}\} \quad \cdot \prod_{k=1}^{K+} \frac{(|X|-m_k)(m_k-1)!}{|X|!},\end{aligned} \tag{B.9}$$

where $H_{|X|} = \sum_{q=1}^{|X|} = \frac{1}{q}$.

Appendix C

Additional SVD results

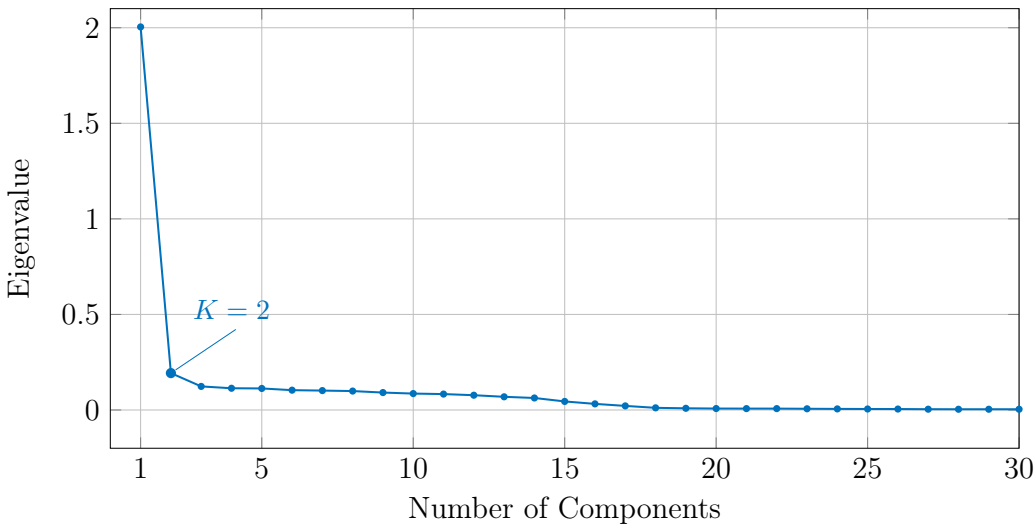


Figure C.1: Scree plot for truncated-SVD operated directly on the adjacency matrix A.

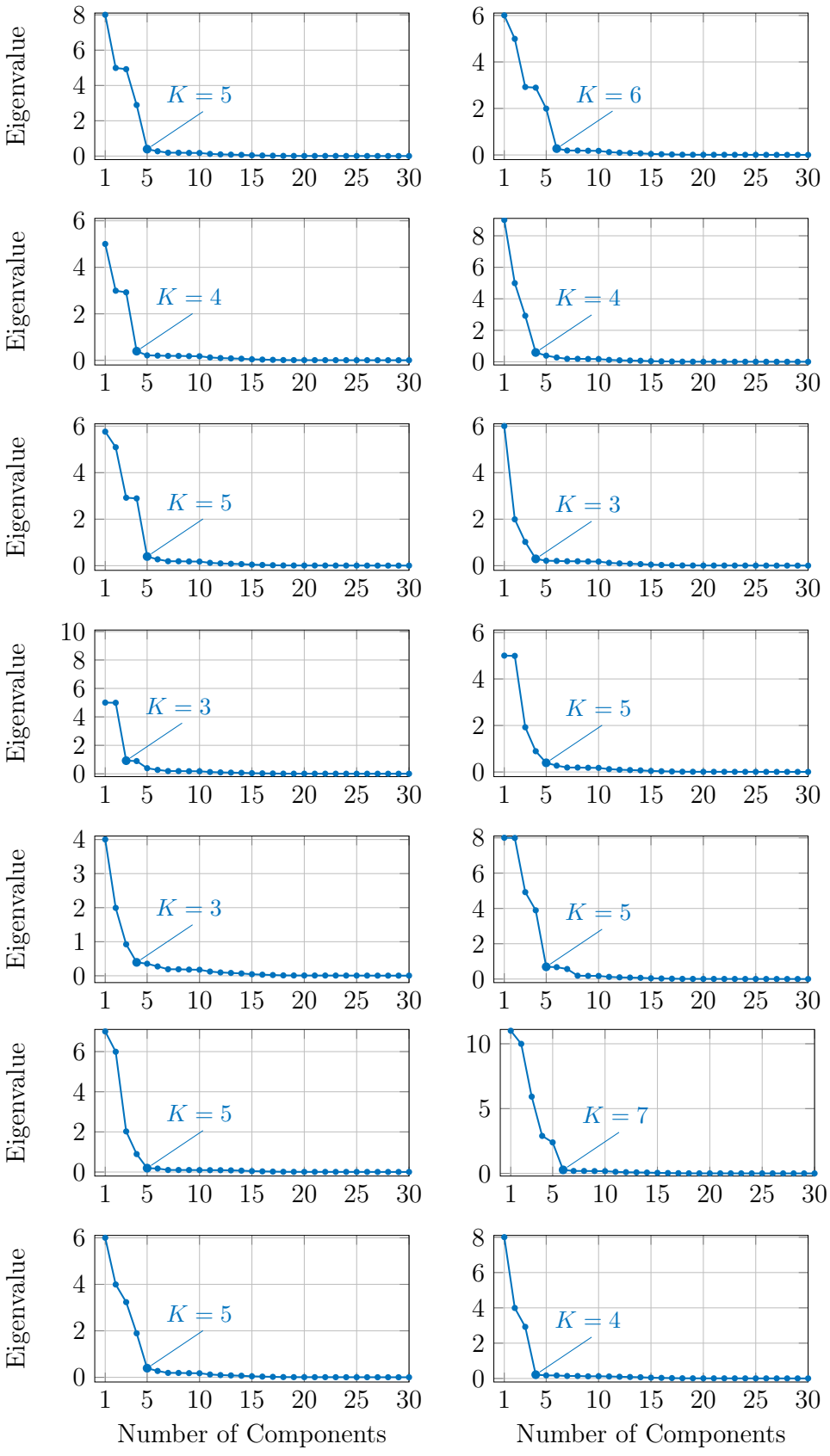


Figure C.2: Scree plots for truncated-SVD operated on $\hat{\Lambda}$, from Sample 2 to Sample 15.