INFERENCE AND PARAMETER ESTIMATION IN BAYESIAN CHANGE POINT MODELS

by

Çağatay Yıldız

B.S., Computer Engineering, Boğaziçi University, 2014

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2017

INFERENCE AND PARAMETER ESTIMATION IN
BAYESIAN CHANGE POINT MODELS

APPROVED BY:

Assoc. Prof. Ali Taylan Cemgil (Thesis Supervisor)	....
Assoc. Prof. Burak Acar	....
Assist. Prof. Umut Şimşekli	....

DATE OF APPROVAL: 28.07.2017

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dr. Ali Taylan Cemgil, for his assistance and patience. I feel truly privileged to study under his supervision. Also, I thank Prof. Bülent Sankur for his guidance through the years. He has been, and will be, a source of inspiration for me. Prof. Haluk Bingöl was also very helpful towards the end of my thesis, for which I am very grateful.

I have spent amazing years in my graduate study. Sharing the same office with Taha Yusuf Ceritli and Barış Kurt has always been fantastic, and will definitely be missed! Hazal Koptagel, Beyza Ermiş, Barış Evrim Demiröz, Orhan 'Abi' Sönmez and all other members of PILAB, many thanks for your support, academic discussions and nice coffee breaks. Our lovely Department Chair Prof. Cem Ersoy, those cookies will always be remembered.

I have been fortunate enough to meet wonderful people at Boğaziçi. It is not possible to mention everyone here, but I would like to thank Betül Uysal, Seray Yüksel, Berkant Kepez and Emre Erdoğan for letting me share all the happiness and stress of my graduate years, motivating nerdy talks and NBA chit-chats.

I would like to express my deepest gratitude to my beloved family, to whom this thesis is dedicated. I have felt their endless love and wholehearted support throughout my life. They made me -and everything I achieved- possible.

Finally, I would like thank the Scientific and Technological Research Council of Turkey (TÜBİTAK). This thesis has been supported by the M.Sc. scholarship (2211) from TÜBİTAK.

ABSTRACT

INFERENCE AND PARAMETER ESTIMATION IN BAYESIAN CHANGE POINT MODELS

In this work, we present a Bayesian change point model that identifies the time points at which a time series undergoes abrupt changes. Our model is a hierarchical hidden Markov model that treats the change points and the dynamics of the data stream as latent variables. We describe a generic generative model, forward-backward recursions for exact inference and an expectation-maximization algorithm for hyper-parameter learning. The model specifications discussed here can sense the changes in the state of the observed system as well as in the intensity and/or the ratio of the features. In addition to investigating the change point algorithm in generic notation, we also give an in-depth analysis and appropriate implementation of a particular model specification, namely, Dirichlet-Multinomial model.

We present a novel application of the model: Distributed Denial of Service (DDoS) attack detection in Session Initiation Protocol (SIP) networks. In order to generate DDoS attack data, we build a network monitoring unit and a probabilistic SIP network simulation tool that initiates real-time SIP calls between a number of agents. Using a set of features extracted from target computer's network connection and resource usage statistics, we show that our model is able to detect a variety of DDoS attacks in real time with high accuracy and low false-positive rates.

ÖZET

BAYESÇİ DEĞİŞİM NOKTASI MODELLERİNDE ÇIKARIM VE PARAMETRE KESTİRİMİ

Bu çalışmada, zaman serilerindeki ani değişimlerin tespiti için kullanılabilecek bir Bayesçi değişim noktası modelini sunuyoruz. Bu model, değişim noktalarını ve veri dinamiğini saklı değişkenler olarak temsil eden bir çok katmanlı saklı Markov modelidir. Çalışmamızda genel bir üreteç modeli, çıkarım yapmak için kullanılan ileri-geri yönlü algoritmayı ve parametre kestirimi için beklenti enyükseltme algoritmasını anlatıyoruz. Modelin incelediğimiz dört özel hali, gözlemlenen sistemin durumunda ve özniteliklerin yoğunluk ve/veya oranında meydana gelen değişiklikleri tespit edebilmektedir. Modelin genel halinin incelemesinin yanı sıra, özel hallerinden birinin -Dirichlet-Multinomial modeli- çözümlemesini yapıp nasıl gerçekleştirilmesi gerektiğini açıklıyoruz.

Modelin özgün bir uygulaması olarak Oturum Başlatma Protokolü (SIP) ağlarında Dağıtık Hizmet Engelleme (DDoS) saldırısı tespiti problemini inceledik. DDoS saldırı verisi üretmek için bir ağ izleme ünitesi ve belirli sayıda kullanıcı arasında gerçek zamanlı SIP aramaları üreten bir olasılıksal SIP ağı benzetim sistemi geliştirdik. Saldırılan bilgisayarın ağ bağlantısından ve kaynak kullanım istatistiklerinden çıkardığımız bir veri kümesi üzerinde yaptığımız deneylerde farklı niteliklere sahip birçok DDoS saldısını gerçek zamanlı olarak, yüksek doğruluk ve düşük yanlış kabul hatası oranlarıyla tespit edebildiğimizi gözlemledik.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\mathbb{E}\left[\cdot\right]$	Expectation
$\mathbb{E}_{q(\cdot)}\left[\cdot\right]$	Expectation with respect to $q(\cdot)$ function
$[\cdot]$	Indicator function, returning 1/0 if the argument is true/false
A_{ij}	Transition probability of hidden Markov model
$\mathbf{A}$	Transition matrix of hidden Markov model
$\mathcal{BE}(\cdot)$	Bernoulli distribution
$\mathbf{C}$	Observation matrix of a hidden Markov model
C_{ij}	Observation probability of a hidden Markov model
$\mathcal{C}at(\cdot)$	Categorical distribution
$\mathcal{D}ir(\cdot)$	Dirichlet distribution
$\mathcal{E}xp(\cdot)$	Exponential distribution
h_t	Hidden variable at time t
$\mathcal{G}(\cdot,\cdot)$	Gamma distribution
L	Lag in online smoothing
M	Maximum number of mixture components retained in forward-bacward algorithm
$\mathcal{M}(\cdot,\cdot)$	Multinomial distribution
$\mathcal{P}(\cdot)$	Poisson distribution
s_t	Switch variable of a change point model at time t
t	Time index
T	Number of time slices
$\mathcal{U}(\cdot,\cdot)$	Uniform distribution
v_t	Visible variable at time t
w	Reset parameter in Bayesian change point model
$\alpha_{t\|t}$	Forward variable
$\alpha_{t+1\|t}$	Forward predict variable
$\beta_{t\|t}$	Backward variable
$\beta_{t\|t+1}$	Backward postdict variable

$\delta(\cdot)$	Dirac delta function
$\Gamma(\cdot)$	Gamma function
$\Omega(\cdot)$	Reset distribution in Bayesian change point model
ν	Tolerated latency in matching change points and ground truth
π	Change point prior in Bayesian change point model
$\psi(\cdot)$	Digamma function
θ	Parameter set
$\Theta(\cdot)$	Observation distribution in Bayesian change point model

LIST OF ACRONYMS/ABBREVIATIONS

AS	Asterisk Stats
AL	Asterisk Logs
DDoS	Distributed Denial of Service
DM	Dirichlet-Multinomial
EM	Expectation-Maximization
GP	Gamma-Poisson
HMM	Hidden Markov Model
iid	Independent and Identically Distributed
KL	Kullback-Leibler
MCMC	Markov Chain Monte Carlo
NMF	Nonnegative Matrix Factorization
PSTN	Public Switched Telephone Network
RTP	Real-time Transport Protocol
RU	Resource Usage
SIP	Session Initiation Protocol
SReq	SIP Requests
SRes	SIP Responses
TCP	Transmission Control Protocol
UA	User Agent
UDP	User Datagram Protocol
VoIP	Voice Over the Internet Protocol

1. INTRODUCTION

Many machine learning problems start with the assumption that the data instances are independently and identically distributed (iid). The order of samples in such problems is not important. However, in cases where consecutive data instances are dependent, iid assumption would not be valid. Exploiting such dependencies typically results in better performance; hence, a substantial literature is devoted to building models for problems involving *sequential* data.

The particular problem we address in this thesis is change point detection in time series. Without loss of generality, we use the terms "sequential data" and "time series" interchangeably - the word "time" should connote some natural ordering. The fundamental assumption regarding the data streams of our interest is that they undergo abrupt changes at random points in time, which are referred as *change points*. We furthermore assume that change points divide the time series into non-overlapping and statistically independent subsequences, i.e., *segments*. Data points are modeled to be homogeneous within segments and heterogeneous across segments [1, 2].

Change point detection techniques have proven useful in a diverse array of applications. In industrial contexts, such models have been used for quality control [3] and measuring and controlling market risk [4]. Biology related applications include but are not limited to the segmentation of electroencephalogram (EEG) signals [5, 6], microarray data [7] and deoxyribonucleic acid (DNA) sequences [8]. Network intrusion detection can also be noted as another application domain [9, 10].

Above-mentioned applications do not come as a free lunch. The problem inherently incorporates certain issues that ought to be taken into consideration. First and foremost, determining the number of change points as well as their locations simultaneously is a remarkable challenge. In case of multiple time series, one would presumably like to take advantage of possibly correlated time series. Online detection of change point locations and algorithms that are designed particularly for lengthy time series also remain as difficult tasks [11].

1.1. Related Work

The literature on change point detection is quite extensive. A good overview of the methods relying mostly on hypothesis testing and cumulative sum algorithm is presented by Basseville and Nikiforov [12]. The technique developed by Desobry *et al.* makes use of single-class support vector machine for temporal segmentation [13]. Kawahara *et al.* presents a subspace identification based model that measures the distance between the subspace spanned by the columns of an observability matrix and the one spanned by the subsequences of time-series data [14]. Neural networks have also been applied to change detection problems [15, 16].

Frequentist methods listed above typically lack in modeling the uncertainty over change point locations, as opposed to Bayesian framework. Furthermore, Bayesian formulations of the problem can automatically capture a trade-off between model complexity and model fit [17]. Earlier Bayesian work on change point problems generally depends on Markov Chain Monte Carlo (MCMC). For example, methods presented by Stephens [18] and Chib [19, 20] use MCMC to sample from the joint posterior of change points. Nonetheless, due to their computational overhead and slow mixing rates of Markov chains, MCMC-based methods are later abandoned.

Fearnhead presents an efficient and elegant Bayesian treatment of the problem [21]. His formulation and its successors are closely tied with the Bayesian product partition change point model by Barry and Hartigan [22]. Caron *et al.* extend Fearnhead's work by a particle filter type algorithm that allows online detection and parameter estimation [2]. Xuan and Murphy show how to apply Fearnhead's algorithms to multidimensional time series by modeling the joint density of vector-valued observations using undirected Gaussian graphical models [17]. Later, Adams and Mackay describe another Bayesian change point detection algorithm for online inference, rather than retrospective segmentation [23].

1.2. Scope of the Thesis

We approach the change point detection problem from a Bayesian standpoint. More specifically, we describe a probabilistic graphical model and forward-backward type recursion for exact inference on multivariate data. Inferred variables denote our belief over change point locations as well as the latent dynamics within each segment. The recursions are designed both for online and offline scenarios; thus, the model can be applied to a large number of real-world problems.

In this thesis, we are concerned with the detection of Distributed Denial of Service (DDoS) attacks in Session Initiation Protocol (SIP) networks. A DDoS attack is defined as an explicit effort to overwhelm the processing power of a computer in order to prevent the legitimate use of a service. The problem is ever-growing since the attackers invent more sophisticated strategies as the Internet becomes more and more complex. The classical way of performing a DDoS attack is flooding the target computer by massive amounts of network packets [24]. Thus one would expect bursts in the counts of packets arriving and leaving the network of the target computer under a DDoS attack, which shows that our model is indeed plausible.

The main contributions of this thesis are the libraries that generate and collect SIP network traffic data, the change point model, a generic implementation of the model and a novel application. Details are as follows:

- *Simulator:* The module used for generating SIP traffic. It initiates real-time SIP calls between a number of users, whose characteristics are governed by a probabilistic generative model.
- *Monitor:* Feature collecting unit. Deployed on a SIP server, it extracts features from the network and operating system of the server, delivers collected data to a client computer in real time, and saves them in order to reproduce experiments.
- *Bayesian Change Point Model (BCPM):* We describe a generic generative model as well as exact and efficient inference and parameter learning algorithms. In addition, we present four specific instances of this generic formulation and

implement the model as an open source C++ library [25].

- *DDoS Attack Detection:* We conduct extensive experiments on the synthetic data generated by Simulator and a DDoS attack generating tool, and report the feature sets and model parameters that yield the best performance.

1.3. Organization of the Thesis

The rest of the thesis is organized as follows: We first provide the necessary background information to remind the main concepts this thesis is built upon. The first part of Chapter 3 explains our Bayesian change point model formulation, describes the inference and parameter learning algorithms, and specifies the generative model. The rest of this chapter is devoted to a detailed analysis of a particular model specification: Dirichlet-Multinomial model. Chapter 4 covers our SIP network data generation mechanism, experiment setup and the results. Finally, Chapter 5 concludes this thesis.

2. THEORETICAL BACKGROUND

This chapter is devoted to provide the theoretical background needed to understand subsequent parts of the thesis. We first examine different time series models in general terms and introduce our notation. Following section covers Expectation-Maximization (EM) algorithm for parameter estimation in latent state models. We finally give a brief overview of Nonnegative Matrix Factorization (NMF).

2.1. Time Series Models

The simplest way to model a data set is to assume that the instances are independent and identically distributed (iid) random variables. In applications such as speech recognition, stock market prediction, DNA sequence analysis, etc, however, this assumption would not properly model sequential dependencies. Therefore, iid assumption must be relaxed in problems that involve temporal relations. Probabilistic graphical models are powerful tools to model such dependencies as well as the uncertainty in the data. For time series data $v_1, v_2, \ldots, v_t$, a graphical model specifies the joint distribution $p(v_1, v_2, \ldots, v_t)$. Our standard notation for such sequential data is given below:

$$v_{1:T} \equiv v_1, v_2, \ldots, v_t \tag{2.1}$$

2.1.1. Markov Model

When we assume that the observation at time t depends only on the most recent observation, we obtain a first-order Markov chain. Mathematically, underlying modeling assumption is the following:

$$p(v_t|v_1, v_2, \ldots, v_{t-1}) = p(v_t|v_{t-1}) \tag{2.2}$$

The joint distribution over all variables can then be factorized as follows:

$$p(v_{1:T}) = p(v_1) \prod_{t=2}^{T} p(v_t | v_{t-1}) \tag{2.3}$$

Although a first-order Markov chain is conceptually sound and easy to implement, it lacks in anticipating patterns involving several consecutive data instances. One way to extend the model is to consider Markov chains of higher orders. More concretely, we may form an L'th order Markov chain if we allow the variable at time t to depend on previous L observations:

$$p(v_t | v_1, v_2, \ldots, v_{t-1}) = p(v_t | v_{t-L}, \ldots, v_{t-1}) \tag{2.4}$$

Graphical models associated with the first-order and second-order Markov models are depicted in Figure 2.1.

When the observations are discrete and can take at most N values, a first-order Markov model can be parameterized by an $N \times N$ transition matrix. Similarly, transition probabilities of an L'th order Markov chain are specified by a $K + 1$ dimensional array, which implies that the number of parameters required grows exponentially with the order of the chain.

2.1.2. Hidden Markov Model

In hidden Markov models (HMMs), we assume that each observation is a probabilistic function of a latent (hidden) state. Furthermore, in contrast to observable Markov model, a Markov chain is formed on latent states, not the observations. Consequently, this setup provides a more general framework for time series analysis and does not require exponentially many parameters - as long as the chain is of order one. Graphical model of a hidden Markov model is given in Figure 2.2. Note that shaded nodes in all graphical models in this thesis represent observations whereas the remain-

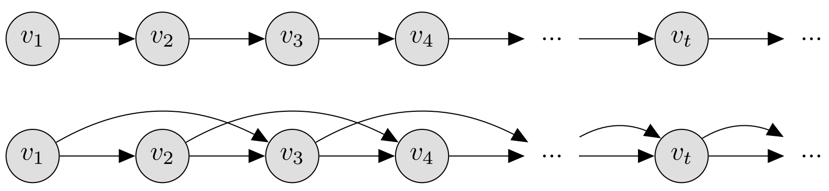

Figure 2.1. Graphical models of first-order and second-order Markov models

ing nodes are hidden variables.

Joint distribution corresponding to the model in Figure 2.2 is given by

$$p(h_{1:T}, v_{1:T}) = p(h_1)\left[\prod_{t=2}^{T} p(h_t|h_{t-1})\right]\left[\prod_{t=1}^{T} p(v_t|h_t)\right] \tag{2.5}$$

The hidden variable h_t in an HMM is always discrete whereas the observation model can either be discrete or continuous. When h_t can take N different values, we use an $N \times N$ matrix $\mathbf{A}$ to denote the state transition probabilities. More explicitly, $A_{ij} = p(h_t = i|h_{t-1} = j)$ denotes the probability of going from state j to state i at any time t. In this notation, $\mathbf{A}$ is a column stochastic matrix, $\sum_i A_{ij} = 1$, and made up of nonnegative entries, $A_{ij} \geq 0$. In case of discrete observations, $v_t \in \{1, 2, \ldots, M\}$, another column stochastic matrix $\mathbf{C}$ represents observation (emission) probabilities: $C_{ij} = p(v_t = i|h_t = j)$ is the probability of observing i, being in state j. We again have $\sum_i C_{ij} = 1$ and $C_{ij} \geq 0$.

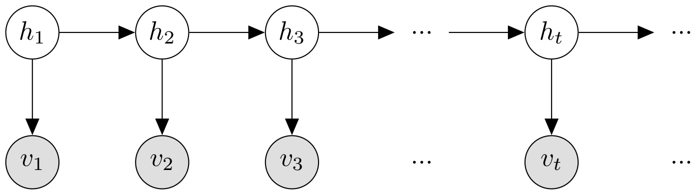

Figure 2.2. Graphical model of hidden Markov model

2.2. Parameter Estimation in Latent Variable Models

Computing the maximum likelihood estimates of many models is typically easy given that the values of all random variables are known. The task becomes trickier when there are missing data or latent variables. For such models, Expectation-Maximization algorithm [26, 27] is an elegant way of finding maximum likelihood solutions.

In this very general recipe, we denote all visible variables by $\mathbf{V}$ and hidden variables by $\mathbf{H}$. Also, let θ represent model parameters. Our goal is to find a set of parameters θ^* that maximizes the log likelihood of the observed data, which is the equivalent of maximizing the likelihood as log is a strictly increasing function:

$$\begin{aligned} \mathcal{L}(\theta) &= \log p(\mathbf{V}|\theta) && (2.6) \\ &= \log \sum_{\mathbf{H}} p(\mathbf{H}, \mathbf{V}|\theta) && (2.7) \end{aligned}$$

Without loss of generality, we suppose that $\mathbf{H}$ is discrete. In case it comprises continuous variables, the sum is replaced by the integral sign. The expression within the summation in Equation 2.7 is referred as *complete data likelihood*. The whole expression is hard to evaluate because the summation over latent variables is usually intractable. For this, classical EM algorithm attempts to maximize a lower bound for the likelihood.

Next, we introduce an arbitrary distribution $q(\mathbf{H})$ and re-write the log-likelihood as follows:

$$\mathcal{L}(\theta) = \log p(\mathbf{V}|\theta) \tag{2.8}$$

$$= \log \sum_{\mathbf{H}} p(\mathbf{H}, \mathbf{V}|\theta) \frac{q(\mathbf{H})}{q(\mathbf{H})} \tag{2.9}$$

$$= \log \mathbb{E}_q \left[\frac{p(\mathbf{H}, \mathbf{V}|\theta)}{q(\mathbf{H})} \right] \tag{2.10}$$

$$\geq \mathbb{E}_q \left[\log \frac{p(\mathbf{H}, \mathbf{V}|\theta)}{q(\mathbf{H})} \right] \tag{2.11}$$

$$= \mathbb{E}_q \left[\log p(\mathbf{H}, \mathbf{V}|\theta) - \log q(\mathbf{H}) \right] \tag{2.12}$$

$$\mathcal{L}(q, \theta) = \mathbb{E}_q \left[\log p(\mathbf{H}, \mathbf{V}|\theta) \right] - \mathbb{E}_q \left[\log q(\mathbf{H}) \right] \tag{2.13}$$

In the fourth line we use the Jensen's inequality, which states that $f(E[x]) \geq E[f(x)]$ is true when f is a concave function (such as log).

The EM algorithm is an iterative optimization method to maximize the likelihood in latent state models. The algorithm simply proceeds by coordinate ascent. At iteration t, assume that we have $q^{(t)}$ and $\theta^{(t)}$. In the E-step, $\theta^{(t)}$ is kept constant and the lower bound $\mathcal{L}(q^{(t)}, \theta^{(t)})$ is maximized with respect to $q(\mathbf{H})$. The update equation in classical EM is

$$q^{(t+1)} = p(\mathbf{H}|\mathbf{V}, \theta^{(t)}) \tag{2.14}$$

To verify the equation, notice that the new value of q function makes the bound tight:

$$\mathcal{L}(p(\mathbf{H}|\mathbf{V}, \theta^{(t)}), \theta^{(t)}) = \mathbb{E}_{p(\mathbf{H}|\mathbf{V},\theta^{(t)})} \left[\log p(\mathbf{H}, \mathbf{V}|\theta^{(t)}) - \log p(\mathbf{H}|\mathbf{V}, \theta^{(t)}) \right] \tag{2.15}$$

$$= \sum_{\mathbf{H}} p(\mathbf{H}|\mathbf{V}, \theta^{(t)}) \log \frac{p(\mathbf{H}, \mathbf{V}|\theta^{(t)})}{p(\mathbf{H}|\mathbf{V}, \theta^{(t)})} \tag{2.16}$$

$$= \sum_{\mathbf{H}} p(\mathbf{H}|\mathbf{V}, \theta^{(t)}) \log p(\mathbf{V}|\theta^{(t)}) \tag{2.17}$$

$$= \log p(\mathbf{V}|\theta^{(t)}) \sum_{\mathbf{H}} p(\mathbf{H}|\mathbf{V}, \theta^{(t)}) \tag{2.18}$$

$$= \log p(\mathbf{V}|\theta^{(t)}) \tag{2.19}$$

The M-Step updates the model parameter $\theta^{(t)}$ to maximize $\mathcal{L}(q,\theta)$ while holding the distribution $q^{(t+1)}$ unchanged. Observe that the second term on the right hand side of the equality in Equation 2.13 is constant with respect to θ. Thus, this step only maximizes $\mathbb{E}_q\left[\log p(\mathbf{H},\mathbf{V}|\theta^{(t)})\right]$, which is referred as *expected complete data log likelihood*. The update equation is the following:

$$\theta^{(t+1)} = \underset{\theta^{(t)}}{\operatorname{argmax}} \mathcal{L}(q^{(t+1)},\theta^{(t)}) \tag{2.20}$$

A nice property of the algorithm is that neither step causes a decrease in the log likelihood. In order to show that, we first need to define the Kullback-Leibler (KL) divergence [28]. KL divergence is a distance measure between two distributions $p(x)$ and $q(x)$, and is defined as follows:

$$\mathrm{KL}(p||q) = -\int p(x)\log\frac{q(x)}{p(x)}dx \tag{2.21}$$

KL divergence is always non-negative, $\mathrm{KL}(p||q) \geq 0$, and equality holds if and only if $p(x) = q(x)$. Also, it is not symmetrical quantity: $\mathrm{KL}(p||q) \neq \mathrm{KL}(q||p)$.

Now, let us re-write the log likelihood:

$$\begin{aligned}
\mathcal{L}(\theta) &= \log p(\mathbf{V}|\theta) && (2.22)\\
&= \sum_{\mathbf{H}} \log p(\mathbf{H},\mathbf{V}|\theta) - \log p(\mathbf{H}|\mathbf{V},\theta) && (2.23)\\
&= \sum_{\mathbf{H}} q(\mathbf{H})\left(\log p(\mathbf{H},\mathbf{V}|\theta) - \log p(\mathbf{H}|\mathbf{V},\theta) + \log q(\mathbf{H}) - \log q(\mathbf{H})\right) && (2.24)\\
&= \sum_{\mathbf{H}} q(\mathbf{H})\left(\log\frac{p(\mathbf{H},\mathbf{V}|\theta)}{q(\mathbf{H})} - \log\frac{p(\mathbf{H}|\mathbf{V},\theta)}{q(\mathbf{H})}\right) && (2.25)\\
&= \sum_{\mathbf{H}} q(\mathbf{H})\left(\log\frac{p(\mathbf{H},\mathbf{V}|\theta)}{q(\mathbf{H})}\right) - \sum_{\mathbf{H}} q(\mathbf{H})\left(\log\frac{p(\mathbf{H}|\mathbf{V},\theta)}{q(\mathbf{H})}\right) && (2.26)\\
&= \mathcal{L}(q,\theta) + \mathrm{KL}(q||p) && (2.27)
\end{aligned}$$

In E-step, q distribution is set to the posterior distribution $p(\mathbf{H}|\mathbf{V}, \theta^{(t)})$. Therefore, KL-divergence in Equation 2.27 vanishes, and the lower bound is equal to the log likelihood. Because E-step maximizes the lower bound, $\mathcal{L}(\theta)$ increases or stays the same if the maximum is already attained. In M-step, the model parameters are set to maximize the lower bound $\mathcal{L}(q, \theta)$. Furthermore, q distribution, which is held constant in this step, was computed conditioned on the old parameters. Hence, it must be different than the new posterior distribution $p(\mathbf{H}|\mathbf{V}, \theta^{(t+1)})$: $\mathrm{KL}(q||p) > 0$. Overall, log likelihood, $\mathcal{L}(\theta)$ increases. The algorithm terminates when it converges to some value.

2.3. Nonnegative Matrix Factorization

Nonnegative matrix factorization has been extensively used in machine learning as an alternative to other matrix decomposition methods such as principal component analysis and vector quantization [29]. Formally, the goal is to decompose a nonnegative matrix $\mathbf{X}$ into two matrices $\mathbf{T}$ and $\mathbf{V}$ with nonnegative entries:

$$\underbrace{\left(\quad \mathbf{X} : \text{Data Matrix} \quad \right)}_{W \times K} \approx \underbrace{\left(\mathbf{T} : \text{Template} \right)}_{W \times I} \times \underbrace{\left(\quad \mathbf{V} : \text{Excitation} \quad \right)}_{I \times K} \tag{2.28}$$

In the original NMF paper, Lee and Seung investigate two different cost functions (Euclidean distance and divergence) between $\mathbf{X}$ and $\mathbf{TV}$. They propose multiplicative update rules for both measures [29]. The rules differ in their multiplicative factors, and the convergence proofs use an auxiliary function analogous to that used for proving the convergence of the EM algorithm.

Since I is typically smaller than W, one interpretation of NMF is a low-rank matrix approximation, just like singular value decomposition. Besides, the nonnegativity constraint distinguishes NMF from other decomposition methods: For one thing, resulting matrices are usually very easy to interpret. For another, many problems, such as source separation and image processing, by nature require $\mathbf{T}$ and $\mathbf{V}$ to be nonnegative. Therefore, NMF has received a lot of attention in recent years.

The update equations presented in [29] provide point estimates for the template and excitation matrices. Consequently, the matrices may easily overfit the data and certainly lack in modeling uncertainty. Cemgil approaches NMF from a Bayesian standpoint and prensents a hierarchical generative model [30]:

$$t_{w,i} \sim \mathcal{G}\left(t_{w,i}; a^t_{w,i}, \frac{b^t_{w,i}}{a^t_{w,i}}\right) \tag{2.29}$$

$$v_{i,k} \sim \mathcal{G}\left(v_{i,k}; a^v_{i,k}, \frac{b^v_{i,k}}{a^v_{i,k}}\right) \tag{2.30}$$

$$s_{w,i,k} \sim \mathcal{P}(s_{w,i,k}; t_{w,i}v_{i,k}) \tag{2.31}$$

$$x_{w,k} = \sum_i s_{w,i,k} \tag{2.32}$$

where $\mathcal{G}$ and $\mathcal{P}$ denote Gamma and Poisson distributions, respectively. Inference in this setup can be carried out by approximate methods, namely, variational Bayes and the Gibbs sampler. Cemgil also shows that the original update equations that minimize KL divergence appear as the maximum likelihood solution (via EM algorithm) of this model when priors are omitted.

Above model assumes that each observation is a Poisson random variable - and thus integer valued. Schmidt *et al.* presents another Bayesian treatment of NMF, in which the likelihood is Gaussian [31]. Priors over the entries of template and excitation matrices are exponential densities multiplied with unit step function to enforce nonnegativity. Finally, the prior over the likelihood variance is an inverse Gamma density. They also present a Gibbs sampler solution to approximate the posterior.

In this work, we use NMF as a data pre-processing technique. More specifically, using the iterative conditional modes update equations presented by Cemgil [30], we extract templates from our data sets. These templates then become the parameters of one specific setting of the Bayesian change point model, which is named Discrete model and detailed in Subsection 3.1.5.4. Also, Figures 4.3 and 4.4 depict the template and excitation matrices inferred by the algorithm when executed on the network traffic data.

3. Bayesian Change Point Models

Bayesian change point model is an example of hierarchical hidden Markov models [21]. The model is described by three sets of variables: switches, latent dynamics and observations, denoted respectively by s_t, h_t and v_t. The switch variable s_t is a binary variable. Conditioned on the switches, h_t is a Markov chain. Furthermore, observations v_t are conditioned on latent states h_t. The graphical representation of the resulting model is given at Figure 3.1.

The generative process of the above-mentioned model is as follows: The initial latent variable h_0 is drawn from reset distribution $\Omega(h)$. At each time step t, the switch s_t is either *on* or *off*. In the former case, which is denoted as $s_t = 1$, the present is isolated from the past and h_t is re-drawn from the reset model (This is why such models are also known as *reset* models [32]). When $s_t = 0$, current latent dynamics are not reset but either updated or kept the same. Two other model assumptions are *(i)* the switches are conditionally independent from all other variables and *(ii)* the observation at time t, v_t, is a random variable sampled from observation distribution $\Theta(v)$ with an unknown parameter h_t. Thus, the complete generative model is given as:

$$h_0 \sim \Omega(h_0; w) \tag{3.1}$$

$$s_t \sim \mathcal{BE}(s_t; \pi) \tag{3.2}$$

$$h_t | s_t, h_{t-1} \sim [s_t = 0]\delta(h_t - h_{t-1}) + [s_t = 1]\Omega(h_t; w) \tag{3.3}$$

$$v_t | h_t \sim \Theta(v_t; h_t) \tag{3.4}$$

where δ is Dirac delta function. $\mathcal{BE}(\cdot; \pi)$ is the Bernoulli distribution with parameter π and defined as follows:

$$\mathcal{BE}(s; \pi) = \exp\left(s \log \pi + (1 - s)\log(1 - \pi)\right) \tag{3.5}$$

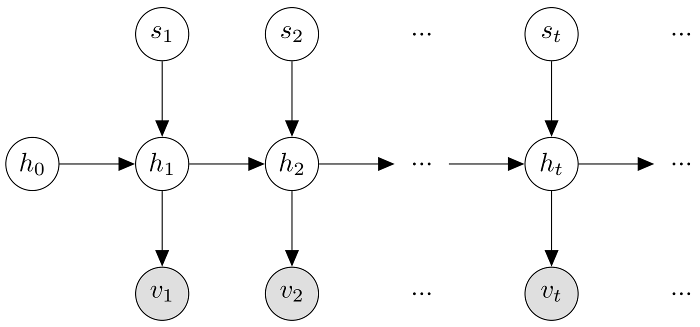

Figure 3.1. Graphical model of Bayesian change point model

Provided that the model parameters w and π are known, we can calculate exact posterior marginals of hidden variables $s_{1:T}$ and $h_{0:T}$ via forward-backward algorithm. Posterior probabilities of $s_{1:T}$ then lead the way to deciding whether a change has occurred at any particular time t or not. In next section, we cover the recursions needed to implement the forward-backward routine for the generic recipe described above and analyze its complexity. We also discuss possible settings of the reset and observation models. Section 3.2 investigates a particular setting of these models, namely, Dirichlet-Multinomial model, where we discuss how inference and parameter estimation are implemented.

3.1. Inference, Parameter Estimation and Model Specifications

3.1.1. Forward Recursion

Forward recursion allows us to infer the probability distribution over latent states s_t and h_t conditioned on observations until time t. In order to formulate the recursion, we need to define two variables:

$$\alpha_{t|t}(s_t, h_t) = p(s_t, h_t, v_{1:t}) \tag{3.6}$$

$$\alpha_{t|t-1}(s_t, h_t) = p(s_t, h_t, v_{1:t-1}) \tag{3.7}$$

These are refered as *forward* and *forward-predict* variables, respectively. What is called filtered change point probability $p(s_t|v_{1:t})$ can then be expressed as a function of the forward message:

$$p(s_t|v_{1:t}) = \int p(s_t, h_t|v_{1:t})dh_t \tag{3.8}$$

$$\propto \int \alpha_{t|t}(s_t, h_t)dh_t \tag{3.9}$$

As new observations arrive, forward-predict and forward messages are alternatively calculated and the posterior probabilities of latent states are propagated as follows:

$$\alpha_{t|t}(s_t, h_t) = p(s_t, h_t, v_{1:t}) \tag{3.10}$$

$$= p(v_t|s_t, h_t)\underbrace{p(s_t, h_t, v_{1:t-1})}_{\alpha_{t|t-1}(s_{t-1},h_{t-1})} \tag{3.11}$$

$$= p(v_t|h_t)\sum_{s_{t-1}}\int p(s_{t-1}, h_{t-1}, s_t, h_t, v_{1:t-1})dh_{t-1} \tag{3.12}$$

$$= p(v_t|h_t)\sum_{s_{t-1}}\int p(s_{t-1}, h_{t-1}, v_{1:t-1})p(s_t, h_t|s_{t-1}, h_{t-1})dh_{t-1} \tag{3.13}$$

$$= p(v_t|h_t)\sum_{s_{t-1}}\int \alpha_{t-1|t-1}(s_{t-1}, h_{t-1})p(h_t|s_t, h_{t-1})p(s_t)dh_{t-1} \tag{3.14}$$

$$= p(v_t|h_t)\sum_{s_{t-1}}\int \alpha_{t-1|t-1}(s_{t-1}, h_{t-1})\Big(\delta(h_t - h_{t-1})p(s_t = 0)+$$

$$p(h_t)p(s_t = 1)\Big)dh_{t-1} \tag{3.15}$$

3.1.2. Backward Recursion

Similar to forward messages, backward messages (*beta-postdict* and *beta*, respectively) are defined as follows:

$$\beta_{t-1|t}(s_{t-1}, h_{t-1}) = p(v_{t:T}|s_{t-1}, h_{t-1}) \tag{3.16}$$

$$\beta_{t|t}(s_t, h_t) = p(v_{t:T}|s_t, h_t) \tag{3.17}$$

Product of beta-postdict message and the observation model gives the beta message as shown below:

$$\beta_{t|t}(s_t, h_t) = \beta_{t|t+1}(s_t, h_t) p(v_t|h_t) \tag{3.18}$$

We now examine the backward propagation step:

$$\begin{aligned}
\beta_{t-1|t}(s_{t-1}, h_{t-1}) &= p(v_{t:T}|s_{t-1}, h_{t-1}) && (3.19)\\
&= \sum_{s_t} \int p(v_{t:T}, s_t, h_t|s_{t-1}, h_{t-1}) dh_t && (3.20)\\
&= \sum_{s_t} \int p(v_{t:T}|s_t, h_t) p(s_t, h_t|s_{t-1}, h_{t-1}) dh_t && (3.21)\\
&= \sum_{s_t} \int \beta_{t|t}(s_t, h_t) p(h_t|s_t, h_{t-1}) p(s_t) dh_t && (3.22)\\
&= \int \beta_{t|t}(s_t, h_t) \Big(\delta(h_t - h_{t-1}) p(s_t = 0) + p(h_t) p(s_t = 1) \Big) dh_t && (3.23)
\end{aligned}$$

3.1.3. Smoothing

Posterior probabilities conditioned on full history can be obtained by multiplying forward and backward messages:

$$\begin{aligned}
p(s_t, h_t|v_{1:T}) &\propto p(s_t, h_t, v_{1:T}) && (3.24)\\
&= p(s_t, h_t, v_{1:t}) p(v_{t+1:T}|s_t, h_t, v_{1:t}) && (3.25)\\
&= p(s_t, h_t, v_{1:t}) p(v_{t+1:T}|s_t, h_t) && (3.26)\\
&= \alpha_{t|t}(s_t, h_t) \beta_{t|t+1}(s_t, h_t) && (3.27)
\end{aligned}$$

When the inference task is online, using forward filtered change point probabilities is appropriate. In cases where a small amount of delay in change point decision at time t is allowed, one may use evidence from *future* in addition to the observations up to time t. For such cases, we use *online (fixed lag) smoothers*, in which a fixed number of backward recursions are performed after each new observation. This fixed number is known as *lag*. If the lag is L, posterior probabilities at time t are calculated conditioned on the observations up to time $t + L$, as formulated below:

$$p(s_t, h_t, v_{1:t+L}) = p(s_t, h_t, v_{1:t})p(v_{t+1:t+L}|s_t, h_t) \tag{3.28}$$

The second expression on the right hand side of the equality is computed by executing backward recursions for L steps, starting from the observation at time $t + L$.

3.1.4. Parameter Estimation

At the beginning of the inference, we assumed that model parameters are known beforehand. Simple search techniques for setting parameters to appropriate values, like grid search, may work in practice but are not typically scalable to large problems. The EM algorithm is a well-known tool in statistical estimation problems involving incomplete data. In this subsection, we derive an EM algorithm to estimate model parameters that maximize the (log)likelihood. Formally, our goal is to estimate $\theta = (w, \pi)$ that maximize

$$\mathcal{L}(\theta) = \log p(s_{1:T}, h_{0:T}, v_{1:T}, |\theta) \tag{3.29}$$

$$= \log \Omega(h_0; w) + \sum_{t=1}^{T} \Big[[s_t = 0]\big(\log(1-\pi) + \log \delta(h_t - h_{t-1})\big) + [s_t = 1]\big(\log \pi + \log \Omega(h_t; w)\big) + \log \Theta(v_t; h_t) \Big] \tag{3.30}$$

<u>3.1.4.1. E-Step.</u> The EM recipe presented in Section 2.2 requires us to take the expectation of Equation 3.30 with respect to the posterior distribution. If we denote the posterior at τ'th step of the algorithm as $p(s_{1:T}, h_{0:T}|v_{1:T}, \theta^{(\tau)})$, the expectation of $\mathcal{L}(\theta)$

with respect to the posterior is given by the following:

$$\begin{aligned}\mathbb{E}_{p(s_{1:T},h_{0:T}|v_{1:T},\theta^{(\tau)})}[\mathcal{L}(\theta)] &= \sum_{t=1}^{T}\mathbb{E}_{p(s_{1:T},h_{0:T}|v_{1:T},\theta^{(\tau)})}\Big[[s_t=0]\big(\log(1-\pi)\big)\Big] \\ &\quad+\sum_{t=1}^{T}\mathbb{E}_{p(s_{1:T},h_{0:T}|v_{1:T},\theta^{(\tau)})}\Big[[s_t=1]\big(\log\pi+\log\Omega(h_t;w)\big)\Big] \\ &\quad+\mathbb{E}_{p(s_{1:T},h_{0:T}|v_{1:T},\theta^{(\tau)})}[\log\Omega(h_0;w)] \\ &\quad+\sum_{t=1}^{T}\mathbb{E}_{p(s_{1:T},h_{0:T}|v_{1:T},\theta^{(\tau)})}\Big[\log\Theta(v_t;h_t)\Big] \qquad (3.31)\\ &=\log(1-\pi)\sum_{t=1}^{T}p(s_t=0|v_{1:T},\theta^{(\tau)}) \\ &\quad+\log\pi\sum_{t=1}^{T}p(s_t=1|v_{1:T},\theta^{(\tau)}) \\ &\quad+\sum_{t=0}^{T}p(s_t=1|v_{1:T},\theta^{(\tau)})\mathbb{E}_{p(h_t|v_{1:T},\theta^{(\tau)})}\Big[\log\Omega(h_t;w)\Big] \\ &\quad+\sum_{t=1}^{T}\mathbb{E}_{p(h_t|v_{1:T},\theta^{(\tau)})}\Big[\log\Theta(v_t;h_t)\Big] \qquad (3.32)\end{aligned}$$

where we introduce an auxillary variable s_0 and set $p(s_0=1|v_{1:T},\theta^{(\tau)})=1$ for compactness in the notation.

3.1.4.2. M-Step. In this step, we maximize Equation 3.32 with respect to w and π by taking partial derivatives. Observe that the last summation in Equation 3.32 is independent of the parameters and thus can be omitted. Partial derivative with respect to π yields:

$$0=\frac{\partial\mathbb{E}_{p(s_{1:T},h_{0:T}|v_{1:T},\theta^{(\tau)})}[\mathcal{L}(\theta)]}{\partial\pi} \qquad (3.33)$$

$$=\frac{1}{\pi}\sum_{t=1}^{T}p(s_t=1|v_{1:T},\theta^{(\tau)})-\frac{1}{1-\pi}\sum_{t=1}^{T}p(s_t=0|v_{1:T},\theta^{(\tau)}) \qquad (3.34)$$

$$\pi^{(\tau+1)}=\frac{1}{T}\sum_{t=1}^{T}p(s_t=1|v_{1:T},\theta^{(\tau)}) \qquad (3.35)$$

Above equation tells us to set the new value of the prior to the average of posterior change point probabilities, which we find very intiutive. Maximization with respect to w involves a single term:

$$0 = \frac{\partial \mathbb{E}_{p(s_{1:T}, h_{0:T} | v_{1:T}, \theta^{(\tau)})}[\mathcal{L}(\theta)]}{\partial w} \tag{3.36}$$

$$= \sum_{t=0}^{T} p(s_t = 1 | v_{1:T}, \theta^{(\tau)}) \frac{\partial}{\partial w} \mathbb{E}_{p(h_t | v_{1:T}, \theta^{(\tau)})} \Big[\log \Omega(h_t; w) \Big] \tag{3.37}$$

The term in Equation 3.37 depends directly on the Ω distribution. Once the derivative is computed for different reset models, a general framework incorparating the algorithm is straightforward to implement.

3.1.5. Model Specifications

Inference and parameter learning methods presented so far are independent of the observation model Θ and reset model Ω. Depending on the application and data set characteristics, these models can be set appropriately. A significant point is that the computations are much easier if the models are discrete or Θ is the conjugate prior of Ω since the inference requires their product. Below are four reset/observation model pairs we investigate in this thesis:

3.1.5.1. Gamma-Poisson (GP) Model. Given a data set consisting of nonnegative integers, this model assumes that the observation v_t is a Poisson random variable with intensity parameter h_t [21,33]. Gamma distribution is an appropriate choice to model the uncertainty over unknown Poisson intensity since both its support and intensity parameters are positive real numbers. Consequently, reset hyperparameter is a couple of real numbers, $w = (a, b)$, which denote the shape and scale parameters of Gamma distribution. As a consequence, forward and backward messages in this setup store a mixture of Gamma potentials over h_t. Below are the probability distribution/mass functions of Gamma and Poisson distributions:

$$\Omega(h; w) = \mathcal{G}(h; a, b) = \frac{b^a}{\Gamma(a)} h^{a-1} e^{-bh} \tag{3.38}$$

$$\Theta(v; h) = \mathcal{P}(v; h) \quad = \frac{h^v e^{-h}}{\Gamma(v+1)} \tag{3.39}$$

When the problem is multidimensional, one may treat each feature as an independent Poisson random variable. In this case, the intensity parameters are either reset altogether or stay the same. Obviously, the model defined by Equations 3.38 and 3.39 is one-dimensional instance of this more general formulation.

3.1.5.2. Dirichlet-Multinomial (DM) Model. If the observations are $K \geq 2$ dimensional vectors of nonnegative integers, they could be treated as Multinomial random variables. This assumption enforces the latent variable h_t to be a $K \geq 2$ dimensional vector whose elements are nonnegative and sum up to 1. Dirichlet distribution fits this setup very well since it is a distribution over $K-$simplex, which includes any possible value of h_t:

$$\Omega(\mathbf{h}; \mathbf{w}) = \mathcal{D}ir(\mathbf{h}; \mathbf{w}) = \frac{\Gamma\left(\sum_{k=1}^{K} w_k\right)}{\prod_{k=1}^{K} \Gamma(w_k)} \prod_{k=1}^{K} h_k^{w_k - 1} \tag{3.40}$$

$$\Theta(\mathbf{v}; \mathbf{h}) = \mathcal{M}(\mathbf{v}; \mathbf{h}) \quad = \frac{\Gamma(\sum_k v_k + 1)}{\prod_k \Gamma(v_k + 1)} \prod_k h_k^{v_k} \tag{3.41}$$

where $\mathcal{D}ir$ and $\mathcal{M}$ stand for Dirichlet and Multinomial distributions, respectively. One may similarly build a Dirichlet-Categorical model if the observations are vectors with one element being 1 and other elements being 0.

3.1.5.3. Compound Model. It is also possible to consider a compound model, in which a pre-determined subset of features is modeled with DM model and GP model is used for the rest. Without loss of generality, we may assume that the features can be arranged in such a way that the first M entries of an N dimensional observation vector v form the first subset and the last $N - M$ entries form the second one. We denote the resulting hyperparameter as $\mathbf{w} = \{\alpha_{1:M}, a_{1:N-M}, b_{1:N-M}\}$ and the Compound model is defined as follows:

$$\Omega(\mathbf{h}; \mathbf{w}) = \mathcal{D}ir(h_{1:M}; \alpha_{1:M}) \times \prod_{i=1}^{N-M} \mathcal{G}(h_{i+M}; a_i, b_i) \tag{3.42}$$

$$\Theta(\mathbf{v}; \mathbf{h}) = \mathcal{M}(v_{1:M}; h_{1:M}) \times \prod_{i=1}^{N-M} \mathcal{P}(v_{i+M}; h_{i+M}) \tag{3.43}$$

One may notice that even if we omit the DM part, i.e., $M = 0$, Equations 3.42 and 3.43 still define a compound (GP) model, provided $N > 1$. Also, the generative model of our BCPM enforces a binary change decision, that is to say, the latent parameters are reset altogether in case of a change.

3.1.5.4. Discrete Model. A discrete BCPM assumes that the system being examined can be at 1 of K possible *state*s. As opposed to DM/GP models, a latent state h_t in this setup does not track the parameter of the observation model but the probability mass over all states. Hence, h_t is modeled as a categorical random variable. We furthermore assume that the observations are Multinomial and that each state k is associated with its own emission probability vector $\mathbf{t}_k$. The overall model is expressed below:

$$\Omega(h; \mathbf{w}) = \mathcal{C}at(h, \mathbf{w}) = \prod_{k=1}^{K} w_k^{[h=k]} \tag{3.44}$$

$$\Theta(\mathbf{v}; h, \mathbf{t}_{1:K}) = \prod_{k=1}^{K} \mathcal{M}(\mathbf{v}; \mathbf{t}_k)^{[h=k]} \tag{3.45}$$

where $\mathcal{C}at$ denotes Categorical distribution.

3.1.6. Complexity Analysis of Forward-Backward

When the state space of h_t is discrete as in Subsection 3.1.5.4, the model reduces to classical HMMs, where the latent state space is simply the Cartesian product of that of s_t and h_t. In this case, forward variables are $2 \times |h_t|$ dimensional matrices, where $|h_t|$ denotes the length of the hidden variable. Nonetheless, other model specifications are in continuous domain and require more detailed analysis.

As we try to sum over exponentially many possible switch settings to calculate the posterior, the inference might seem intractable at first glance. However, thanks to the forgetting (reset) property of the model, it is possible to implement forward-backward routine efficiently, i.e., quadratic with the number of observations T. We first define the two components of the forward message given in Equation 3.15 as follows:

$$\begin{aligned} p(s_t = 0, h_t, v_{1:t}) &= p(v_t|h_t) \sum_{s_{t-1}} \int p(s_{t-1}, h_{t-1}, v_{1:t-1}) \delta(h_t - h_{t-1}) p(s_t = 0) dh_{t-1} \\ &= p(v_t|h_t) p(h_t, v_{1:t-1}) p(s_t = 0) \qquad (3.46) \\ p(s_t = 1, h_t, v_{1:t}) &= p(v_t|h_t) \sum_{s_{t-1}} \int p(s_{t-1}, h_{t-1}, v_{1:t-1}) p(h_t) p(s_t = 1) dh_{t-1} \\ &= p(v_t|h_t) p(v_{1:t-1}) p(h_t) p(s_t = 1) \qquad (3.47) \end{aligned}$$

In our terminology, components in Equations 3.46 and 3.47 denote *no-change* and *change* cases. In a no-change case, the forward message at time $t-1$ is multiplied by observation probability and change point prior. Informally speaking, probability distribution over h_{t-1} is updated and *transferred* to h_t, due to the integral and Dirac delta function. In a change case however, $p(h_t)$ is reset to $\Omega(h_t; w)$, meaning it becomes a single component whose parameter is independent of previous message.

Overall, the forward message $\alpha_{t|t}(s_t, h_t)$ is computed by updating the previous message (formulated in no-change case) and adding one more component (change case). Therefore, in general, if $\alpha_{t-1|t-1}(s_{t-1}, h_{t-1})$ is a mixture with K components, the next forward message would have $K + 1$ components. In other words, the number of components grows only linearly with time, and filtering consequently scales with $O(T^2)$. For smoothing, we multiply forward and backward messages; hence, the complexity becomes $O(T^4)$. Similarly, executing fixed lag smoothing, with lag being L, requires $O(T^2L)$ computational load.

Figure 3.2 demonstrates the forward messages obtained when GP model is fed with synthetic univariate data. Each subplot in Figure 3.2 illustrates the Gamma mixture in a forward message, and each curve is a Gamma potential over the latent Poisson intensity parameter. Components are drawn proportional to their weights. As shown, number of mixture components grows linearly.

Although the computational effort is far less than exponential, filtering is still too expensive for long time series and one needs to resort to approximations. A very simple way of doing so is setting an upper limit on the number of mixture components [34]. This procedure is known as *pruning* in the literature and can be done by eliminating the component with the smallest normalization constant after each update step of the forward-backward algorihtm. If M components are retained, the complexities of filtering and smoothing become $O(MT)$ and $O(M^2T^2)$. The value of M is usually chosen to be much smaller than T. As illustrated in Figure 3.2, retaining only a few mixture components is sufficient since many of them have very small normalization constants.

In order to empirically analyze the impact of M on the approximation, we run the forward-backward algorithm with different settings of M on the data generated by GP model. Figure 3.3 shows the true latent Poisson intensity as well as the expected intensities obtained by forward recursions with $M \in \{5, 10, 50\}$. As shown, the change in the mean intensity is negligible and inference with a very small setting of M gives a good estimate of true intensity.

3.2. Detailed Analysis of Dirichlet-Multinomial Model

In this section, we investigate the forward-backward algorithm and derive EM update equations specifically for the Dirichlet-Multinomial model. As discussed in Subsection 3.1.6, forward and backward messages are Dirichlet mixture distributions over latent variables. Thus, forward-backward routine requires multiplication of Dirichlet potentials as well as marginalization over latent variables. Before diving into the

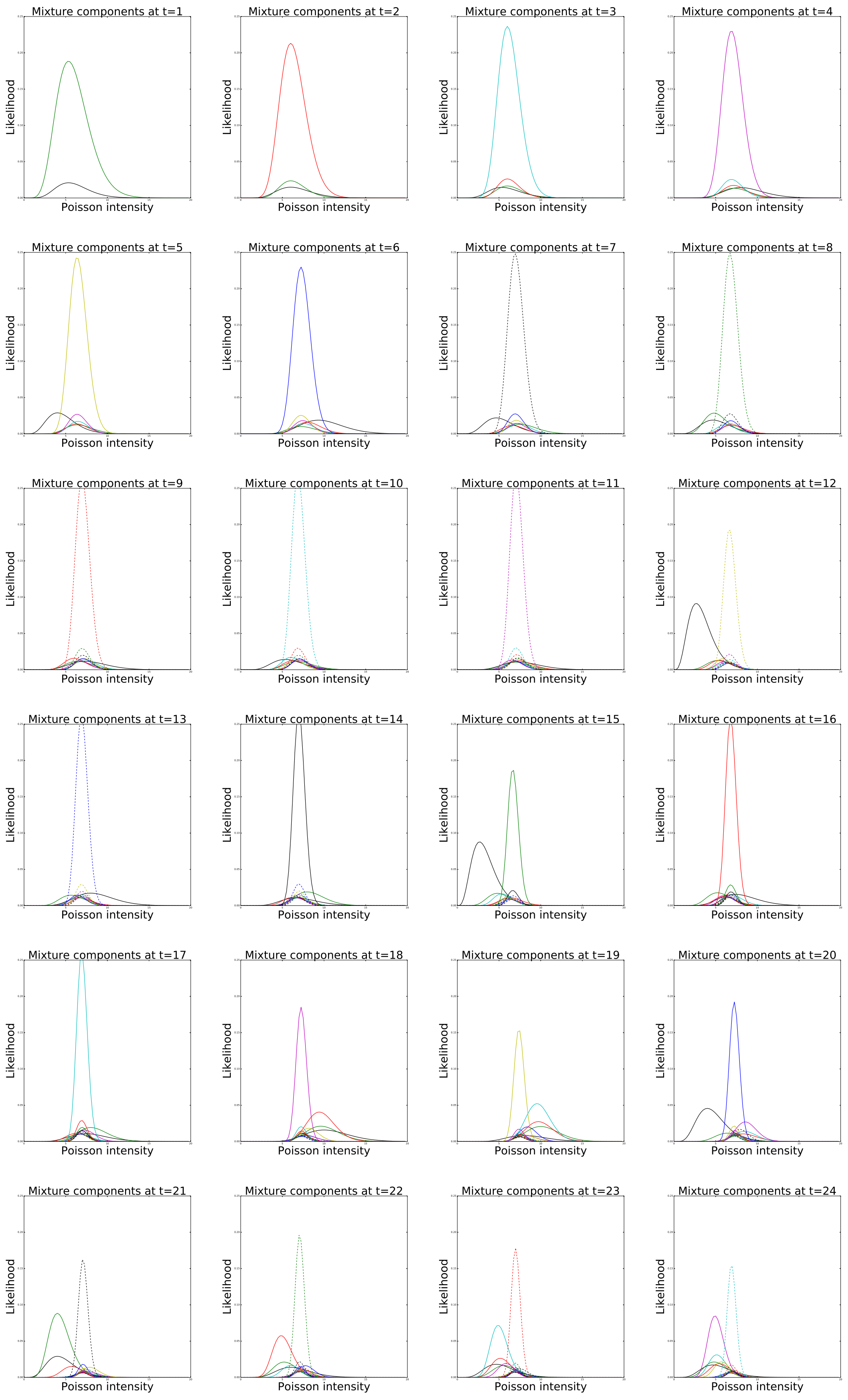

Figure 3.2. Gamma mixture components of forward messages

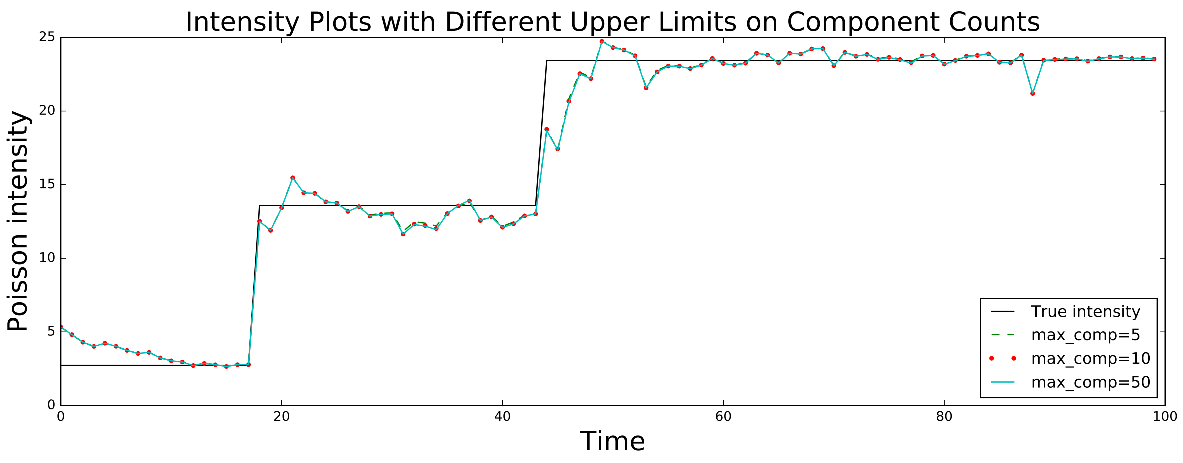

Figure 3.3. Mean intensities with different upper limits on component counts

details of the routine, let us introduce the notation and required operations.

3.2.1. Preliminaries

3.2.1.1. Dirichlet Potential. A Dirichlet potential is represented by a tuple (a, c) where a is the Dirichlet parameter and c is the logarithm of the normalizing constant.

$$(a, c) = e^c \mathcal{D}ir(x; a) \tag{3.48}$$

$$c = \log \int (a, c) dx \tag{3.49}$$

3.2.1.2. Multiplication of Two Dirichlet Potentials. We previously showed that in order to compute the smoothed density, forward and backward messages -and hence Dirichlet mixtures- must be multiplied. The good news is that the product of two Dirichlet potentials (a_1, c_1) and (a_2, c_2) over the same random variable is another Dirichlet potential:

$$
\begin{aligned}
(a_1, c_1)(a_2, c_2) &= e^{c_1}\mathcal{D}ir(x; a_1)e^{c_2}\mathcal{D}ir(x; a_2) & (3.50)\\
&= \exp\Bigg(c_1 + c_2 + \log\Gamma\left(\sum_{k=1}^{K} a_{1,k}\right) - \sum_{k=1}^{K}\log\Gamma(a_{1,k}) + \sum_{k=1}^{K}(a_{1,k}-1)\log x_k \\
&\qquad + \log\Gamma\left(\sum_{k=1}^{K} a_{2,k}\right) - \sum_{k=1}^{K}\log\Gamma(a_{2,k}) + \sum_{k=1}^{K}(a_{2,k}-1)\log x_k\Bigg) & (3.51)\\
&= \exp\Bigg(c_1 + c_2 + \log\Gamma\left(\sum_{k=1}^{K} a_{1,k}\right) - \sum_{k=1}^{K}\log\Gamma(a_{1,k}) + \log\Gamma\left(\sum_{k=1}^{K} a_{2,k}\right) \\
&\qquad - \sum_{k=1}^{K}\log\Gamma(a_{2,k}) + \sum_{k=1}^{K}(a_{1,k}+a_{2,k}-2)\log x_k\Bigg) & (3.52)\\
&= (a_1 + a_2 - 1, g(a_1, c_1, a_2, c_2)) & (3.53)
\end{aligned}
$$

where the new normalizing constant is

$$
\begin{aligned}
g(a_1, c_1, a_2, c_2) = c_1 + c_2 &+ \log\Gamma\left(\sum_{k=1}^{K} a_{1,k}\right) - \sum_{k=1}^{K}\log\Gamma(a_{1,k}) \\
&+ \log\Gamma\left(\sum_{k=1}^{K} a_{2,k}\right) - \sum_{k=1}^{K}\log\Gamma(a_{2,k}) \\
&- \log\Gamma\left(\sum_{k=1}^{K}(a_{1,k}+a_{2,k}-1)\right) + \sum_{k=1}^{K}\log\Gamma(a_{1,k}+\alpha_{2,k}-1)
\end{aligned}
\qquad (3.54)
$$

3.2.1.3. Multinomial Distribution as a Dirichlet Potential. As mentioned before, update step corresponds to multiplication of a message with the observation model $p(v_t|h_t) = \mathcal{M}(v_t; h_t)$. We know that Dirichlet distribution is the conjugate prior of Multinomial; therefore, the multiplication is easy. Below, we express Multinomial distribution in terms of a Dirichlet potential:

$$\mathcal{M}(x;\pi) = \exp\left(\log\Gamma\left(1+\sum_{k=1}^{K} x_k\right) - \sum_{k=1}^{K}\log\Gamma(x_k+1) + \sum_{k=1}^{K} x_k \log\pi_k\right) \tag{3.55}$$

$$= \exp\left(\log\Gamma\left(1+\sum_{k=1}^{K} x_k\right) - \sum_{k=1}^{K}\log\Gamma(x_k+1) + \sum_{k=1}^{K} x_k \log\pi_k + \log\Gamma\left(\sum_{k=1}^{K}(x_k+1)\right) - \log\Gamma\left(\sum_{k=1}^{K}(x_k+1)\right)\right) \tag{3.56}$$

$$= \mathcal{D}ir(\pi; x+1)\exp\left(\log\Gamma\left(1+\sum_{k=1}^{K} x_k\right) - \log\Gamma\left(\sum_{k=1}^{K}(x_k+1)\right)\right) \tag{3.57}$$

$$= (x+1, c) \tag{3.58}$$

where $c = \log\Gamma\left(1+\sum_{k=1}^{K} x_k\right) - \log\Gamma\left(\sum_{k=1}^{K}(x_k+1)\right)$. The update step can then be easily implemented using Equations 3.52 and 3.57.

3.2.2. Implementation of the Inference

3.2.2.1. Computing Posteriors of Latent Variables. To track the density of h_t and the posterior probability of change, we need the marginals over latent variables. First, let us express a forward message explicitly:

$$\alpha_{t|t}(s_t, h_t) = \sum_m \alpha_{t|t}^m(s_t, h_t) \tag{3.59}$$

$$= \sum_m (a_{t|t}^m, c_{t|t}^m) \tag{3.60}$$

Here, superscript m goes over mixture components. In other words, $a_{t|t}^m$ and $c_{t|t}^m$ correspond to Dirichlet parameter and the log normalizing constant of m'th component. In order to compute both filtered and smoothed change point probabilities, we need to calculate the marginal of a mixture of Dirichlet potentials over h_t:

$$p(s_t|v_{1:t}) \propto \int p(s_t, h_t, v_{1:t}) dh_t \tag{3.61}$$
$$= \int \sum_m (a^m_{t|t}, c^m_{t|t}) dh_t \tag{3.62}$$
$$= \sum_m \int (a^m_{t|t}, c^m_{t|t}) dh_t \tag{3.63}$$
$$= \sum_m \exp(c^m_{t|t}) \tag{3.64}$$

As already shown, EM algorithm requires computing the expected value of the filtering/smoothing distribution over h_t:

$$\mathbb{E}\left[p(h_t|v_{1:t})\right] \propto \mathbb{E}\left[\sum_{s_t} p(h_t, s_t, v_{1:t})\right] \tag{3.65}$$
$$= \sum_m \mathbb{E}\left[(a^m_{t|t}, c^m_{t|t})\right] \tag{3.66}$$
$$= \sum_m \exp(c^m_{t|t}) \frac{a^m_{t|t}}{|a^m_{t|t}|} \tag{3.67}$$

since the expected value of a Dirichlet random variable with parameter a is the unit vector pointing the same direction as a.

3.2.2.2. Implementation of Forward Recursion. In order to start forward recursion, we first need to define the first forward message, which is readily available given the generative model:

$$\alpha_{0|0}(h_0) = (w, 0) \tag{3.68}$$

Let us further introduce the shorthand for logarithms of change point priors as

$$l_0 = \log p(s_t = 0) = \log(1 - \pi) \tag{3.69}$$
$$l_1 = \log p(s_t = 1) = \log \pi \tag{3.70}$$

As shown in Equations 3.46 and 3.47, one step prediction gives two sets of components, representing no-change and change cases. The potential in no-change case becomes (w, l_0) because the parameter of the Dirichlet potential stays the same and the prior probability of no-change is l_0. In change case, the parameter is reset (to the same value) and the potential is equal to (w, l_1). Since the parameters of both Dirichlet potentials are the same, we can merge them into a single component (which we have not done while plotting Figure 3.2):

$$\alpha_{1|0}(h_1, s_1) = (w, 0) \tag{3.71}$$

What is next is the update step. Representing Multinomial distribution as a Dirichlet potential, we have the following:

$$\alpha_{1|1}(h_1, s_1) = p(v_1|h_1) \times \alpha_{1|0}(h_1, s_1) \tag{3.72}$$

$$= (v_t + 1, \kappa_1) \times (w, 0) \tag{3.73}$$

$$= (w + v_t, \zeta_1) \tag{3.74}$$

How to calculate exact values of κ_1 and ζ_1 is given in Subsections 3.2.1.2 and 3.2.1.3. Next predict step yields a mixture of two Dirichlet potentials whose parameters are $w+v_t$ (no change) and w (change). Following this pattern, we deduce that the forward message at time t is made up of t mixture components.

We now consider the prediction and update steps in general. The formulas for prediction step at time t were given in Equations 3.46 and 3.47:

$$\alpha_{t|t-1}(h_t, s_t) = p(s_t = 1, h_t, v_{1:t-1}) \cup p(s_t = 0, h_t, v_{1:t-1}) \tag{3.75}$$

Here, $\cup$ means $\alpha_{t|t-1}(h_t, s_t)$ is composed of the components in $p(s_t = 1, h_t, v_{1:t-1})$ plus $p(s_t = 0, h_t, v_{1:t-1})$. Let us express each term explicitly in terms of our Dirichlet potential representation:

$$p(s_t = 1, h_t, v_{1:t-1}) = p(s_t = 1) \times p(h_t) \times \sum_{s_{t-1}} \int \alpha_{t-1|t-1}(h_{t-1}, s_{t-1}) dh_{t-1} \tag{3.76}$$

$$= \exp(l_1) \times (w, 0) \times \sum_m \exp(c^m_{t-1|t-1}) \tag{3.77}$$

$$= \left(w, l_1 + \log \sum_m \exp(c^m_{t-1|t-1}) \right) \tag{3.78}$$

Concentrating on no-change case:

$$p(s_t = 0, h_t, v_{1:t-1}) = p(s_t = 0) \times \sum_{s_{t-1}} \int \alpha_{t-1|t-1}(h_{t-1}, s_{t-1}) \delta(h_t - h_{t-1}) dh_{t-1} \tag{3.79}$$

$$= p(s_t = 0) \sum_m (a^m_{t-1|t-1}, c^m_{t-1|t-1}) \tag{3.80}$$

$$= \sum_m (a^m_{t-1|t-1}, l_0 + c^m_{t-1|t-1}) \tag{3.81}$$

The update operation in general is simply tantamount to computing t Dirichlet potential products:

$$\alpha_{t|t}(h_t, s_t) = p(v_t|h_t) \times \alpha_{t|t-1}(h_t, s_t) \tag{3.82}$$

$$= (v_t + 1, \kappa_t) \times \sum_m (a^m_{t|t}, c^m_{t|t}) \tag{3.83}$$

$$= \sum_m (v_t + a^m_{t|t}, \rho^m_t) \tag{3.84}$$

3.2.2.3. Implementation of Backward Recursion. Implementation of backward recursion is a little bit trickier than the forward recursion. We first note that the update step is exactly the same as described in Subsection 3.2.2.2 and turn our attention to backward propogation. To start the recursion, we need to define the last beta-postdict message $\beta_{T|T+1}(s_T, h_T) = p(v_{T+1:T}|s_T, h_T)$. Because $v_{T+1:T}$ has no practical equivalent, what we define is a placeholder message, i.e., an identity element with respect to Dirichlet potential multiplication. If K, $\mathbf{1}$, and gammaln(x) denote the input dimensionality, a K dimensional vector of ones, and logarithm of the absolute value of the Gamma function on a real input x, respectively, our placeholder is $(\mathbf{1}, \text{gammaln}(K))$. Then the first update step becomes the following:

$$\begin{aligned}\beta_{T|T}(s_T, h_T) &= p(v_T|s_t, h_t)\beta_{T|T+1}(s_T, h_T) && (3.85)\\ &= (v_T + 1, \kappa_T) \times (\mathbf{1}, \text{gammaln}(K)) && (3.86)\\ &= (v_T + 1, \kappa_T) && (3.87)\end{aligned}$$

Equation 3.23 formulates the postdict step. In no-change case,

$$\begin{aligned}p(s_t = 0, v_{t:T}|s_{t-1}, h_{t-1}) &= \int p(v_{t:T}|s_t, h_t)\delta(h_t - h_{t-1})p(s_t = 0)dh_t && (3.88)\\ &= \sum_m (a^m_{t|t}, l_0 + c^m_{t|t}) && (3.89)\end{aligned}$$

Respectively, in change case,

$$\begin{aligned}p(s_t = 1, v_{t:T}|s_{t-1}, h_{t-1}) &= \int_{h_t} p(v_{t:T}|s_t, h_t)p(h_t)p(s_t = 1)dh_t && (3.90)\\ &= \sum_m \int (a^m_{t|t}, c^m_{t|t}) \times (w, l_1)dh_t && (3.91)\end{aligned}$$

Because we marginalize over h_t, we end up with a normalization constant and again use $\mathbf{1}$ as the Dirichlet parameter of the placeholder message. Let us introduce the function const$[\cdot]$, which takes a Dirichlet potential as parameter and returns the log normalizing constant. Taking all into consideration, the change component is equal to

$$p(s_t = 1, v_{t:T}|s_{t-1}, h_{t-1}) = \left(\mathbf{1}, l_1 + \log \sum_m \exp(\text{const}[(w, 0) \times (a^m_{t|t}, c^m_{t|t})])\right) \quad (3.92)$$

3.2.2.4. Implementation of Expectation-Maximization Algorithm. An EM algorithm for BCPM was previously derived in Subsection 3.1.4. In order to optimize with respect to π, we simply compute the average of smoothed change point probabilities, which are readily available once the forward-backward procedure is implemented. In this subsection, we analyze how to optimize with respect to w. For this, we need to take

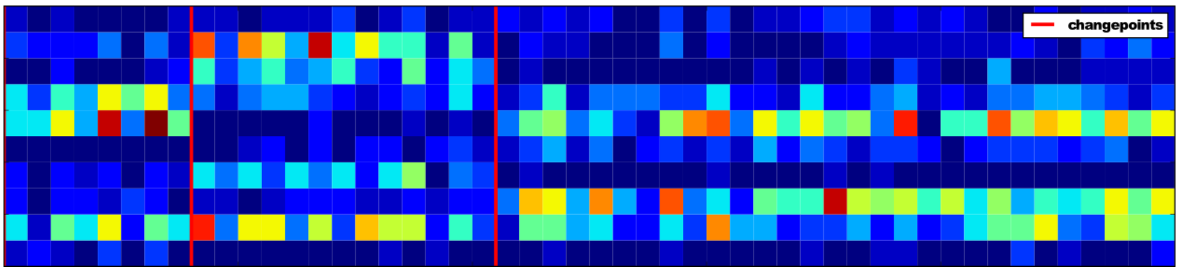

Figure 3.4. Randomly generated data from the generative model using Dirichlet-Multinomial reset-observation models

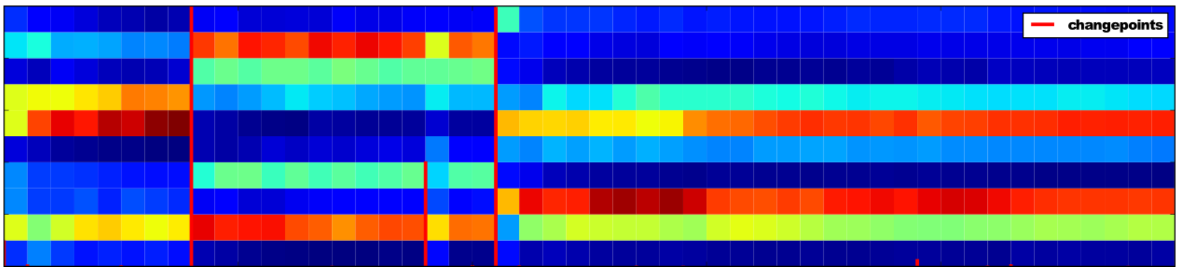

Figure 3.5. Forward filtered change point probabilities on forward-filtered mean densities

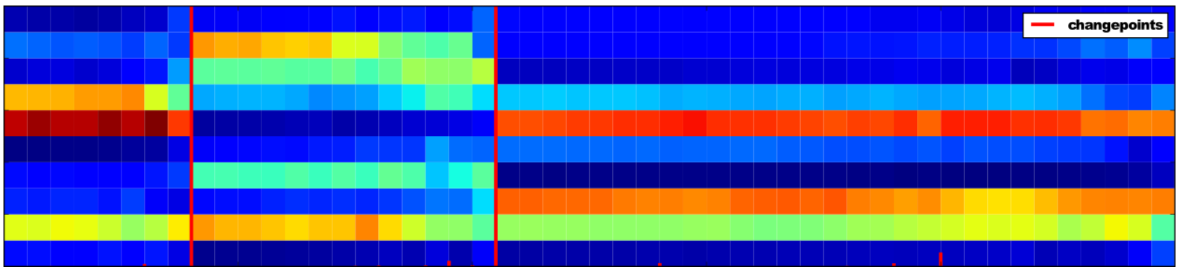

Figure 3.6. Backward-filtered change point probabilities on backward-filtered mean densities

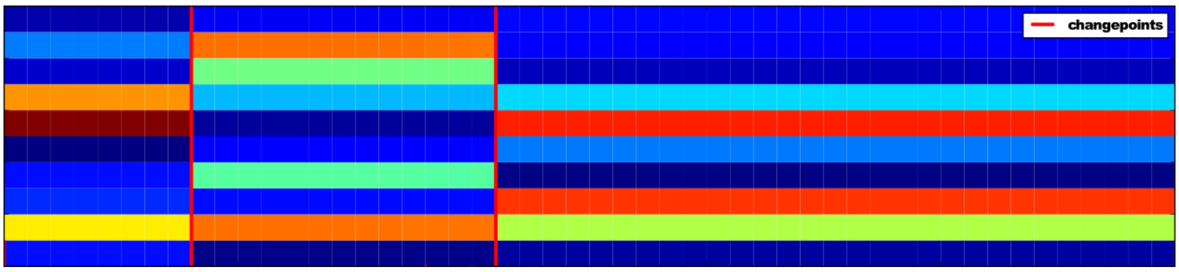

Figure 3.7. Smoothed change point probabilities on smoothed mean densities

the derivative of expected complete data log likelihood, given in Equation 3.36, with respect to w_k, k'th component of w, and set it equal to zero:

$$
\begin{aligned}
0 &= \sum_{t=0}^{T} p(s_t = 1 | v_{1:T}, \theta^{(\tau)}) \frac{\partial}{\partial w_k} \mathbb{E}_{p(h_t|v_{1:T},\theta^{(\tau)})} \Big[\log \mathcal{D}ir(h_t; w) \Big] && (3.93) \\
&= \sum_{t=0}^{T} p(s_t = 1 | v_{1:T}, \theta^{(\tau)}) \mathbb{E}_{p(h_t|v_{1:T},\theta^{(\tau)})} \Bigg[\frac{\partial}{\partial w_k} \sum_k (w_k - 1) \log h_{t,k} + \frac{\partial}{\partial w_k} \log \Gamma \left(\sum_k w_k \right) \\
&\qquad - \frac{\partial}{\partial w_k} \sum_k \log \Gamma(w_k) \Bigg] && (3.94) \\
&= \sum_{t=0}^{T} p(s_t = 1 | v_{1:T}, \theta^{(\tau)}) \mathbb{E}_{p(h_t|v_{1:T},\theta^{(\tau)})} \left[\log h_{t,k} + \psi \left(\sum_k w_k \right) - \psi(w_k) \right] && (3.95) \\
&= \sum_{t=0}^{T} p(s_t = 1 | v_{1:T}, \theta^{(\tau)}) \left\{ \psi \left(\sum_k w_k \right) - \psi(w_k) + \mathbb{E}_{p(h_t|v_{1:T},\theta^{(\tau)})} \left[\log h_{t,k} \right] \right\} && (3.96)
\end{aligned}
$$

Here, ψ is known as the digamma function. If we set $C = \sum_t p(s_t = 1|v_{1:T}, \theta^{(\tau)})$, above expression becomes

$$
\frac{1}{C} \sum_{t=0}^{T} p(s_t = 1 | v_{1:T}, \theta^{(\tau)}) \mathbb{E}_{p(h_t|v_{1:T},\theta^{(\tau)})} \left[\log h_{t,k} \right] = \psi(w_k) - \psi \left(\sum_k w_k \right) \qquad (3.97)
$$

Next, we need to compute the expectation, and hence need the posterior. If smoothed posterior is expressed as below

$$
p(h_t | v_{1:T}, \theta^{(\tau)}) = \frac{1}{Z} \sum_m \exp(c^m) \mathcal{D}ir(h_t | a^m) \qquad (3.98)
$$

where $Z = \sum_m \exp(c^m)$ is the normalization constant, then the expectation in Equation 3.97 can be explicitly written:

$$\begin{aligned}
\mathbb{E}_{p(h_t|v_{1:T},\theta^{(\tau)})}\left[\log h_{t,k}\right] &= \int p(h_t|v_{1:T},\theta^{(\tau)}) \log h_{t,k} dh_t && (3.99)\\
&= \int \frac{1}{Z}\sum_m \exp(c^m)\mathcal{D}ir(h_t|a^m) \log h_{t,k} dh_t && (3.100)\\
&= \frac{1}{Z}\sum_m \exp(c^m) \int \mathcal{D}ir(h_t|a^m) \log h_{t,k} dh_t && (3.101)\\
&= \frac{1}{Z}\sum_m \exp(c^m)\mathbb{E}[\log h_{t,k}]_{p(h_t|a^m)} && (3.102)\\
&= \frac{1}{Z}\sum_m \exp(c^m)\left(\psi(a^m_k) - \psi(a^m_0)\right) && (3.103)
\end{aligned}$$

where we define $a^m_0 \equiv \sum_k a^m_k$ and Minka shows how to compute the expectation in the last step [35]. Plugging Equation 3.103 in Equation 3.97, we finally have the following:

$$\frac{1}{C}\sum_{t=0}^{T}\left\{p(s_t = 1|v_{1:T},\theta^{(\tau)})\left(\frac{1}{Z}\sum_m \exp(c^m)\left(\psi(a^m_k) - \psi(a^m_0)\right)\right)\right\} = \psi(w_k) - \psi\left(\sum_k w_k\right) \tag{3.104}$$

Minka also shows that the following fixed-point iteration gives us the new estimate of w_k [35]:

$$\psi\left(w^{\text{new}}_k\right) = \frac{1}{C}\sum_{t=0}^{T}\left\{p(s_t = 1|v_{1:T},\theta^{(\tau)})\left(\frac{1}{Z}\sum_m \exp(c^m)\left(\psi(a^m_k) - \psi(a^m_0)\right)\right)\right\} + \psi\left(\sum_k w^{\text{old}}_k\right) \tag{3.105}$$

Finally, ψ function must be inverted. Here, we do not give the details but a quickly converging Newton iteration to perform the inversion is given in [35].

4. DDoS ATTACK DETECTION in SIP NETWORKS

In this chapter, we present an application of the Bayesian change point model described in Chapter 3: DDoS attack detection in SIP networks. BCPM is a good choice to tackle this problem because of two fundamental reasons. First, DDoS attacks cause instantaneous changes in the quantities being observed, particularly in the network traffic data. Furthermore, the problem naturally requires online data processing rather than retrospective segmentation. It is worth pointing out that the model allows us to formulate the inference problem in a batch settings as well, which typically results in better performance.

Volumes of research that discuss a wide range of techniques for detecting (D)DoS attacks have been produced. Some of the methods are problem-specific and hence require incorporating the knowledge of the problem domain. On the other hand, statistical methods, relying heavily on learning the normal state of the observed system, identify anomalies in general (rather than particular instances of DDoS attacks). Although BCPM belongs to the second category, our application deals exclusively with one type of DDoS attacks, namely flooding attacks, that target a particular network protocol. Following subsections cover the details of flooding attacks and SIP networks. We then describe our data generation mechanism and the data sets. The section is concluded by the experiments conducted and the results.

4.1. SIP Networks

Voice Over the Internet Protocol (VoIP) technology has emerged as a strong alternative to public switched telephone networks (PSTN) mainly because VoIP services allow text and video communication in addition to the voice, and provide means for the migration from PSTN. The transition from PSTN to VoIP leads to an increase in the popularity of SIP [36]. Nowadays, SIP is considered to be one of the standard protocols to setup, modify and end VoIP sessions. Two general types of SIP entities are *(i)*user agents (UAs) and *(ii)*servers. A user agent is an endpoint entity that generates and receives SIP messages. In a typical SIP session, UAs communicate with each other by sending request and response messages. Servers are responsible for registering users, storing their locations, delivering messages when needed and so forth.

SIP messages are divided into two basic categories: SIP requests and SIP responses. Each SIP request sent by a UA is supposed to be answered by one of the corresponding SIP response messages. For example, a UA can make a REGISTER request to the server in order to change its status to online, make an INVITE request to another UA to start a call, or make a BYE request to terminate an ongoing conversation. A SIP response message generated for a request can be of one of the 6 SIP response categories: 1xx-provisional, 2xx-success, 3xx-redirection, 4xx-client error, 5xx-server error or 6xx-global failure.

An illustrative example of SIP communication is given in Figure 4.1. It shows the flow of exchanged messages between a server and two users during a normal call. In this scenario, Alice initiates a call to Bob by sending an *INVITE* packet to the SIP server. After authentication, the SIP server forwards this request to Bob. Similarly, Bob's response, in this case *ACK* packet showing that the call is accepted, is transmitted to Alice through the SIP server. At the end, *BYE* messages terminate the conversation.

Once a SIP session is established, two endpoints start exchanging multimedia data such as audio conversations, video streams, etc. Recall that SIP, being a signaling protocol, is not involved in the multimedia data exchange between agents. Handshake on the type of conversation, encoding, addresses and ports to be used for transfer and other details regarding the data exchange are usually achieved using Session Description Protocol [37]. Additionally, real-time media delivery relies on Real-time Transport Protocol (RTP) [38].

4.2. DDoS Attacks

Denial of Service attacks intend to prevent legitimate use of a service. When performed by multiple hosts, such attacks are called Distributed Denial of Service at-

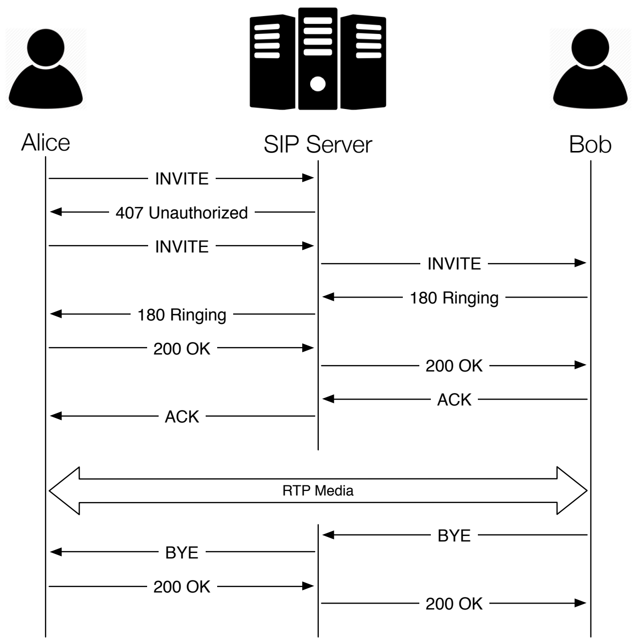

Figure 4.1. A call scenario in SIP network.

tacks. One frequently exercised way of performing a DDoS attack is to send malformed packets to victim computer to confuse an application or a protocol. Alternatively, attackers may aim to consume a key resource such as CPU, memory, bandwith and socket [24,39]. Such attacks usually enforce victim computer to freeze or reboot.

Flooding attack is a particular type of DDoS attacks, which is performed by sending a vast amount of legitimate network packets to the target. These attacks are difficult to combat with since filtering network traffic typically harms legitimate requests. The impact of the attack is contingent on the attack settings such as flood rate, network protocol and packet types used to perform it, and thus may differ substantially from one attack to another. Also, settings in many attacks do not stay constant throughout an attack, which typically results in detection latency [24].

An example DDoS flooding attack in SIP networks is the *INVITE* attack, in which the attacker sends a large number of *INVITE* packets to a SIP server. Unless particular counter-measures are implemented, these packets seem to be legitimate call initiation requests. In case the server attempts to process all requests, it would reach its memory capacity and eventually stop responding to normal users.

4.3. Data Set Generation

Many machine learning algorithms proposed to sense DDoS attacks lack in privacy guarantees and may be exploited to uncover personal information, even from anonymized data sets with the help of extra information [40]. Thus, in order to train a machine learning solution, real world SIP traffic must be collected with uttermost care in order not to reveal sensitive information. As expected, not many SIP data sets are publicly available. To combat with the absence of real-world data, we build a simulation setup that consists of following components:

- *Simulator:* To mimic the traffic on a SIP server, we build a probabilistic SIP network simulation system that initiates calls between a number of users in real-time [41]. The details of the underlying probabilistic model are presented in Appendix A.
- *Monitor:* This module is responsible for collecting a set of features from a SIP server. Collected data are delivered to a client computer in real time and DDoS detection algorithm runs on streaming data.
- *Trixbox:* The simulation setup naturally requires a SIP server to operate. We use Asterisk-based private branch exchange software named *Trixbox* as a SIP server [42].
- *NOVA V-Spy:* To generate a rich variety of DDoS attacks concurrently with normal traffic simulation, we make use of a commercial vulnerability scanning tool, called *NOVA V-Spy* [43].

We control the simulation environment with two variables: network traffic intensity and DDoS attack intensity. The former is a parameter of the Simulator and controls the frequency of the calls whereas the latter is the rate of flooded packets, governed by NOVA V-Spy. Both variables are binary, that is, they are set to be either *low* or *high*, which makes up a total of four data sets. In the rest of this work, we refer to the data sets as *LOW-LOW*, *LOW-HIGH*, *HIGH-LOW* and *HIGH-HIGH*, where the

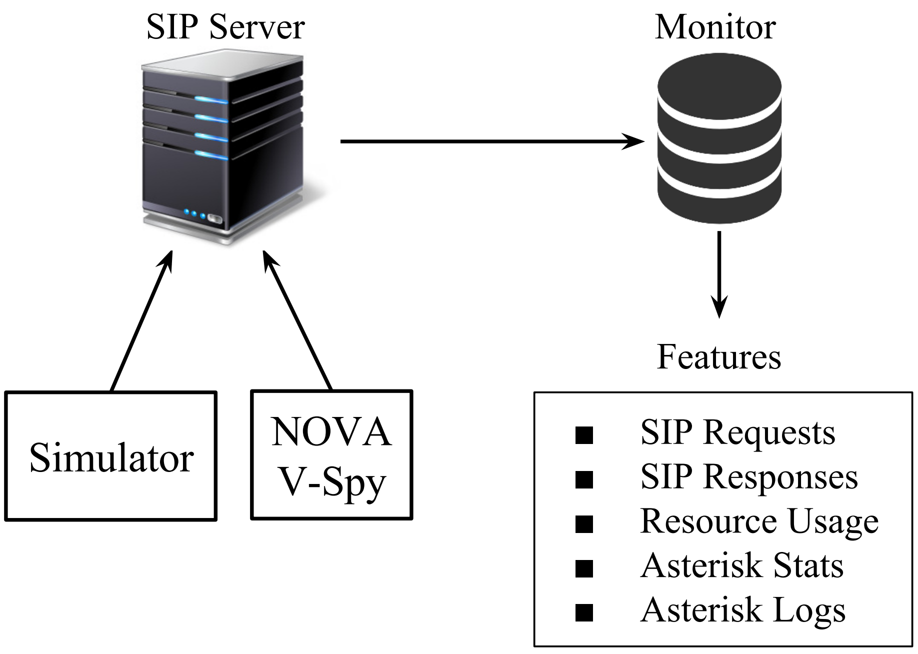

Figure 4.2. SIP simulation environment

adjectives qualify the network traffic intensity and the flood rate, respectively. During the simulations, 500 users are registered to the server. IP addresses and the user id's of attackers are randomly chosen.

Robustness is one of the key properties of a reliable DDoS attack detector. To test how our detection mechanism responds to attacks with different characteristics, we generate 40 DDoS attacks per data set. We leave an interval of at least 25 seconds between two consecutive attacks; this results in a simulation sequence around half an hour. Attack parameters are listed below:

- *Attack Type:* We flood the server with five SIP packets: *REGISTER*, *INVITE*, *OPTIONS*, *CANCEL* and *BYE* packets.
- *Transport Protocol:* SIP is designed to run over several transport layer protocols. In our experiments, we send attack packets both over Transmission Control Protocol (TCP) and User Data Protocol (UDP).
- *Fluctuation:* A binary variable denoting whether the packets are generated uni-

formly with time or fluctuates over time.

- *Content Size:* NOVA V-Spy can optionally insert dummy strings to the end of SIP messages. Because such strings affect the bandwidth consumption and thus are relevant to attack detection task, we generate two sorts of DDoS attacks in that manner: with or without dummy strings.

Even though there is no consensus on how to classify network traffic as abnormal, the traffic caused by DDoS attacks is said to exhibit unique properties [44]. The main source of information to establish whether the system is under attack or not is the network load, which implies that significant changes in the pattern of SIP message histograms possibly indicate a DDoS attack. Therefore, our first two feature categories consist of 14 *SIP Request (SReq)* and 14 *SIP Response (SRes)* packets (the incoming and outgoing packets) counted for fixed observation intervals (e.g., 1 sec).

We monitor three additional modules of the victim: *Resource Usage (RU)*, *Asterisk Stats (AS)* and *Asterisk Logs (AL)*. The first category is the pair of total CPU and memory usage of the server. The second group consists of operating system statistics, such as the number of open file handlers, connections, running threads, etc. The last category counts the keywords in the log files generated by Asterisk. The listed features are all expected to diverge from their settled values in case of an attack and hence potentially signal it.

4.4. Evaluation Criteria

We measure the performances of different model specifications using precision (P) and recall (R) measures. Both P and R are desired to be close to 1. As the rates of false alarm and/or missed change points increase, P and R, respectively, diverge more from 1. We also calculate the F-score (F), or the harmonic mean of precision and recall

Table 4.1. Five categories of the features collected from network and OS sides of the SIP server

Category	Feature	Description
SIP Requests	REGISTER	Num. of "register" requests
	INVITE	Num. of "invite" requests
	SUBSCRIBE	Num. of "subscription" requests
	NOTIFY	Num. of "notification" requests
	OPTIONS	Num. of "options" requests
	ACK	Num. of "acknowledgment" requests
	BYE	Num. of "bye" requests
	CANCEL	Num. of "cancellation" requests
	PRACK	Num. of "provisional acknowledgement" requests
	PUBLISH	Num. of "event publish" requests
	INFO	Num. of "information update" requests
	REFER	Num. of "call transfer" requests
	MESSAGE	Num. of "instant message" requests
	UPDATE	Num. of "session state update" requests
Resource Usage	TOT_CPU	Percentage of total CPU usage
	TOT_MEM	Percentage of total virtual memory usage
Asterisk Stats	A_CPU	Percentage of CPU used by Asterisk
	MEM	Percentage of physical memory utilized by Asterisk
	FH	Num. of Asterisk file descriptors
	THREADS	Num. of Asterisk threads
	TCP_CONN	Num. of Asterisk TCP connections
	UDP_CONN	Num. of Asterisk UDP connections

Table 4.2. Five categories of the features collected from network and OS sides of the SIP server, cont.

Category	**Feature**	**Description**
SIP Responses	100	Num. of “trying” responses
	180	Num. of “ringing” responses
	183	Num. of “session progress” responses
	200	Num. of “success” responses
	400	Num. of “bad request” errors
	401	Num. of “unauthorized” errors
	403	Num. of “forbidden” errors
	404	Num. of “not found” errors
	405	Num. of “not allowed” errors
	481	Num. of “dialog does not exist” errors
	486	Num. of “busy” errors
	487	Num. of “request terminated” errors
	500	Num. of “server internal” errors
	603	Num. of “decline” errors
Asterisk Logs	A_WARNING	Num. of Asterisk “warning” log messages
	NOTICE	Num. of Asterisk “notice” log messages
	VERBOSE	Num. of Asterisk “verbose” log messages
	ERROR	Num. of Asterisk “error” log messages
	DEBUG	Num. of Asterisk “debug” log messages

$$F\text{-score} = 2 \times \frac{P \times R}{P + R} \tag{4.1}$$

$$\text{Precision (P)} = \frac{\#\text{ true alarms}}{\#\text{ alarms}} = \frac{T_a}{T_a + F_a} \tag{4.2}$$

$$\text{Recall (R)} = \frac{\#\text{ true alarms}}{\#\text{ ground truth}} = \frac{T_a}{G_a} \tag{4.3}$$

where T_a and F_a are the true and false alarm counts respectively, and G_a is the true number of change points. An alarm g_t means signaling of a change event at time t, which is triggered whenever the change point probability exceeds a certain threshold λ_a. An alarm g_t is said to be "correct" if it is signalled within ν seconds following an attack onset at time $\hat{t}$. In other words, we increment the number of true alarms if $0 \leq t - \hat{t} < \nu$. Alarms not matched with any ground truth are regarded as false positives.

4.5. Model and Parameter Settings

We conduct experiments with all models specificed in Subsection 3.1.5. For each model, we either include or exclude a feature category, which makes a total of $2^5 - 1 = 31$ sets of experiments per each model specification.

For a given feature setting and model, we perform a grid search in order to find the hyperparameters yielding the best performance. Because the data sets are considerably long and high-dimensional, each experiment is very time-consuming; therefore, we keep the search space small. More concretely, we assign a single parameter ϕ for the Dirichlet priors by setting $w_{DM} = [\phi, \phi, \ldots, \phi]$. Similarly, we fix the shape parameter of all Gamma priors in GP and Compound model to a single value $\psi = w_{GP} = w_C$ and scale parameter to 1. We also set π to three different values.

Parameter setting in Discrete model is slightly different. The first parameter to be set is the number of states K. Regardless of this value, we assume that the prior distribution over hidden states to be flat: $w_D = 1/K[1, 1, \ldots, 1]$. In order to determine

Table 4.3. Grid search parameters

Parameter	Search values
Change point prior (π)	$10^{-2}, 10^{-4}, 10^{-8}$
DM parameter (ϕ)	$1, 10, 100$
GP parameter (ψ)	$1, 10, 100$
Number of states (K)	$6, 8, 10, 12, 14$

the emission probability vector T_k associated with each state $k \in \{1, \ldots, K\}$, we run the NMF algorithm with $I = K$. Each column of the resulting template matrix corresponds to an emission probability vector. Overall search space is given in Table 4.3.

Figure 4.3 illustrates how an example data set is decomposed into two factor matrices using NMF. The matrix on the left side contains the network data, i.e., histograms of SIP network packets over time. Other two matrices are the template and excitation matrices. In this work, we simulate five types of DDoS attacks; thus along with the normal traffic on the background, observed vectors could roughly be classified into six patterns. When we force the template matrix to be sparse with $K = 6$ and the excitation matrix to be dense by tuning the priors, we observe that each basis corresponds to either the signature of one of the DDoS attacks or the normal traffic (see Figure 4.4).

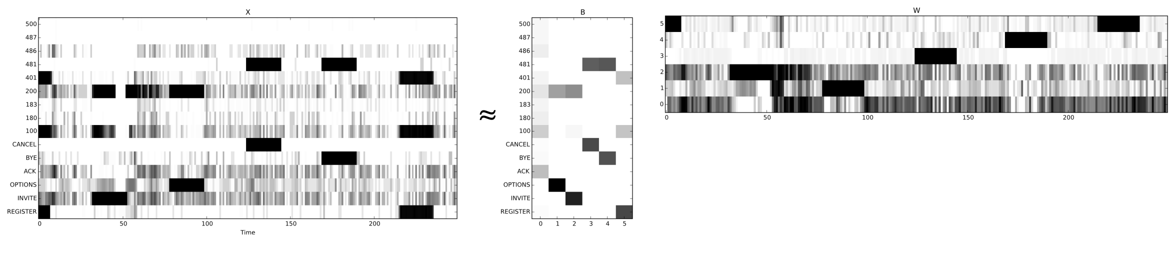

Figure 4.3. Data, template and excitation matrices, respectively from left to right. The (x,y) axes of the matrices are (time, message types), (template indices, message types), (time, template indices) respectively.

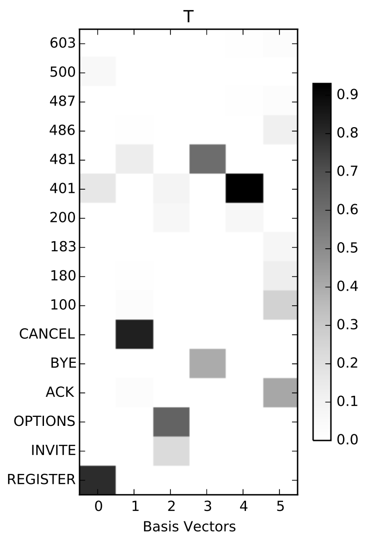

Figure 4.4. Normalized basis vectors extracted by NMF from whole data set.

4.6. Results

Best F-scores achieved by four model settings after grid search are presented in Tables 4.4 and 4.5. Results in these tables are computed based on filtered and online smoothed change point probabilities, respectively. Both L and ν, denoting the number of backward steps in online smoothing and the toleranced latency while matching change points with true attack onset labels, are set to 5. This number seems to be a reasonable choice, considering the experiment setup allows a successful attack detection to be at most $L+\nu$ seconds after the onset (In the worst scenario, a successful signal might have been produced ν seconds after the attack and computed after L steps backward recursion). One may adjust the value depending on the cruciality of latency in detection.

The most striking finding is that online smoothing results in better performance than filtering. This is not surprising since change points are computed given evidence from future. Along with that, increasing lag L does not enhance F-score after a certain threshold, which we emprically determine as 5. As importantly, we observe that DM model, which only tracks the ratios of features usually gives better accuracy than GP model, which tracks the magnitudes of signals. It should also be noticed that

Compound model, which is able to tract the ratios and the magnitudes at the same time, performs slightly better than DM model in online smoothing.

Higher F-scores are attained when attack intensity is boosted because more intense DDoS attacks cause greater divergence from normal traffic and hence facilitate detection. It is also worth noting that the quality of detection usually rises with the network traffic volume, which may seem contradictory with our prior claims at first glance. Nonetheless, in case of low background traffic intensity, tiny changes in the traffic generated by Simulator are easier to be mislabeled (as attack). The findings indeed confirm that F-scores worsen due to lower precision values, rather than missed DDoS attacks.

As mentioned in Subsection 3.1.6, inference is intractable for long time series. Consequently, we maintain $M = 50$ mixture components having the highest weights in forward recursion (for time series consisting of 2000-2100 observations). Empirically, we observe that keeping additional components does not increase F-score.

Because all feature settings in Tables 4.4 and 4.5 include SIP Requests, we deduce that they constitute the most significant feature set. This comes as no surpise due to the fact that flooded packets are all SIP request packets. In the second place, Resource Usage statistics help improving the accuracy of our system. It can be furthermore noticed that Asterisk Stats assist in sensing DDoS attacks with high intensity in particular, noting that specifications of the target computer as well as the operating system would definitely affect how useful this feature set can be.

Finally, we calculate model performances with hyperparameters learned by the EM algorithm described in Subsection 3.1.4. We observe that the F-scores obtained after the maximum likelihood estimation of the parameters are much lower than the scores obtained by grid search. A possible explanation is that EM update equations weight the sufficient statistics by respective change point probabilities (see Equation 3.37). Because the model raises alarms to the fluctuations in the background traffic in addition to DDoS attacks, learned parameters converge to such values that

Table 4.4. Highest F-scores attained after grid search with filtered change point probabilities

Model Specification	Traffic Intensity	Attack Intensity	Feature Set	F-score	Average F-score
Dirichlet-Multinomial	Low	Low	SReq, RU	0.83	0.91
	Low	High	SReq, RU, AS	0.94	
	High	Low	SReq, RU	0.89	
	High	**High**	**SReq, RU, AS**	**0.98**	
Gamma-Poisson	Low	Low	SReq	0.90	0.9
	Low	High	SReq	0.93	
	High	Low	SReq	0.79	
	High	**High**	**SReq**	**0.98**	
Compound Model	Low	Low	SReq(GP), RU(DM)	0.91	0.91
	Low	High	SReq(GP), RU(DM)	0.94	
	High	Low	SReq(GP), RU(DM)	0.79	
	High	**High**	**SReq(GP), RU(DM)**	**0.98**	
Discrete Model	Low	Low	SReq, RU, AS	0.84	0.88
	Low	**High**	**SReq, RU, AS**	**0.94**	
	High	Low	SReq, RU, AS	0.84	
	High	High	SReq, RU AS	0.9	

Table 4.5. Highest F-scores attained after grid search with online smoothed change point probabilities

Model Specification	Traffic Intensity	Attack Intensity	Feature Set	F-score	Average F-score
Dirichlet-Multinomial	Low	Low	SReq	0.92	0.95
	Low	High	SReq, RU, AS	0.96	
	High	Low	SReq, RU	0.92	
	High	**High**	**SReq, RU, AS**	**0.98**	
Gamma-Poisson	Low	Low	SReq, RU	0.94	0.93
	Low	**High**	**SReq, RU**	**0.95**	
	High	Low	SReq, RU	0.89	
	High	High	SReq, RU	0.94	
Compound Model	Low	Low	SReq(GP), AS(DM)	0.95	0.96
	Low	High	SReq(GP), AS(DM)	0.96	
	High	Low	SReq(DM), RU(GP)	0.94	
	High	**High**	**SReq(GP), AS(DM)**	**0.99**	
Discrete Model	Low	Low	SReq, RU	0.91	0.9
	Low	**High**	**SReq, AS**	**0.94**	
	High	Low	SReq, RU	0.84	
	High	High	SReq, AS	0.92	

eventually make the model more sensitive to the fluctuations, whereas we would like to detect DDoS attack onsets only.

5. CONCLUSION

In this work, we approached the change point detection problem from a Bayesian standpoint. Our model is a hierarchical hidden Markov model that comprises three sets of variables: switches, latent dynamics and observations. Our formulation reduces the task of deciding whether a change occurs or not at a given time to computing the posterior probability of the switch variable at that time. We provided a forward-backward type algorithm for inference, which allows both real-time and batch processing. We also derived update equations for learning two model parameters, namely, the prior probability of a change and reset parameter, through the Expectation-Maximization algorithm.

We described our change point detection framework independent of particular distributions modeling observations and latent dynamics. To specify the model, we explained four different observation and reset distribution pairs: Gamma-Poisson, Dirichlet-Multinomial, Compound and Discrete models, which track the magnitude, ratio, a mixture of magnitude and ratio of the input signal, and the discrete states of the observed system. Section 3.2 presented an exhaustive analysis of the Dirichlet-Multinomial model, in which forward-backward messages and the predict and update equations were explicitly derived. We also discussed the complexity analysis and a simple trick that allows real-time approximate inference.

The novel application of the model is DDoS attack detection in SIP networks. Since we were not able to find any publicly available data set collected from a SIP server under a DDoS attack, we first built a simulation environment consisting of a SIP traffic generator, SIP server and DDoS attack generator. We conducted several experiments with different model specifications and feature combinations, and showed that the Compound model outperforms the others. Moreover, we identified SIP Request and Resource Usage feature categories as the most informative ones as they very frequently appear in the settings yielding the highest F-scores.

As we demonstrated in our experiments, the proposed method is powerful and flexible, and yields good results in an arguably challenging problem. Nonetheless, both the model and the applications can be further improved in many respects. A list of future research directions are as follows:

- Despite pruning is shown to work very well in practice, other approximation techniques such as particle filters or collapsing multiple components into a single one may be investigated for problems involving highly heterogeneous time series data.
- Currently, the switch variables are assumed to be independent of all other variables in the model. A Markov chain on the switches may be considered.
- The latent dynamics when there is no change do not necessarily stay the same. New transition dynamics could be explored.
- We can build a more generic framework if multiple reset templates are allowed to exist. Such a setup would better represent the uncertainty over the latent dynamics.
- New observation-reset distribution pairs may be worked out. As long as the reset distribution is the conjugate prior for the observation distribution, the generic inference and parameter learning recipes stay valid.

REFERENCES

1. Carlin, B. P., A. E. Gelfand and A. F. M. Smith, “Hierarchical Bayesian Analysis of Changepoint Problems”, *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, Vol. 41, No. 2, pp. 389–405, 1992, `http://www.jstor.org/stable/2347570`.

2. Caron, F., A. Doucet and R. Gottardo, “On-line changepoint detection and parameter estimation with application to genomic data”, *Statistics and Computing*, Vol. 22, No. 2, pp. 579–595, 2012.

3. Aroian, L. A. and H. Levene, “The effectiveness of quality control charts”, *Journal of the American Statistical Association*, Vol. 45, No. 252, pp. 520–529, 1950.

4. Bormetti, G., M. E. De Giuli, D. Delpini and C. Tarantola, “Bayesian value-at-risk with product partition models”, *Quantitative Finance*, Vol. 12, No. 5, pp. 769–780, 2012.

5. Barlow, J., O. Creutzfeldt, D. Michael, J. Houchin and H. Epelbaum, “Automatic adaptive segmentation of clinical EEGs”, *Electroencephalography and Clinical Neurophysiology*, Vol. 51, No. 5, pp. 512–525, 1981.

6. Bodenstein, G. and H. M. Praetorius, “Feature extraction from the electroencephalogram by adaptive segmentation”, *Proceedings of the IEEE*, Vol. 65, No. 5, pp. 642–652, 1977.

7. Erdman, C. and J. W. Emerson, “A fast Bayesian change point analysis for the segmentation of microarray data”, *Bioinformatics*, Vol. 24, No. 19, pp. 2143–2148, 2008.

8. Braun, J. V., R. Braun and H.-G. Muller, “Multiple changepoint fitting via quasilikelihood, with application to DNA sequence segmentation”, *Biometrika*, pp. 301–

314, 2000.

9. Zhang, H., Z. Gu, C. Liu and T. Jie, "Detecting VoIP-specific Denial-of-service Using Change-point Method", *Proceedings of the 11th International Conference on Advanced Communication Technology - Volume 2*, pp. 1059–1064, IEEE Press, 2009.

10. Rebahi, Y. and D. Sisalem, "Change-point detection for voice over IP denial of service attacks", *KiVS 2007*, 2007.

11. Nyamundanda, G., A. Hegarty and K. Hayes, "Product partition latent variable model for multiple change-point detection in multivariate data", *Journal of Applied Statistics*, Vol. 42, No. 11, pp. 2321–2334, 2015.

12. Basseville, M., I. V. Nikiforov *et al.*, *Detection of abrupt changes: theory and application*, Vol. 104, Prentice Hall Englewood Cliffs, 1993.

13. Desobry, F., M. Davy and C. Doncarli, "An online kernel change detection algorithm", *IEEE Transactions on Signal Processing*, Vol. 53, No. 8, pp. 2961–2974, 2005.

14. Kawahara, Y., T. Yairi and K. Machida, "Change-point detection in time-series data based on subspace identification", *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pp. 559–564, IEEE, 2007.

15. Gupta, V., "Speaker change point detection using deep neural nets", *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 4420–4424, IEEE, 2015.

16. Bulunga, M. L., "Change-point detection in dynamical systems using auto-associative neural networks", , 2012.

17. Xuan, X. and K. Murphy, "Modeling changing dependency structure in multi-

variate time series", *Proceedings of the 24th international conference on Machine learning*, pp. 1055–1062, ACM, 2007.

18. Stephens, D., "Bayesian retrospective multiple-changepoint identification", *Applied Statistics*, pp. 159–178, 1994.

19. Chib, S., "Calculating posterior distributions and modal estimates in Markov mixture models", *Journal of Econometrics*, Vol. 75, No. 1, pp. 79–97, 1996.

20. Chib, S., "Estimation and comparison of multiple change-point models", *Journal of econometrics*, Vol. 86, No. 2, pp. 221–241, 1998.

21. Fearnhead, P., "Exact and efficient Bayesian inference for multiple changepoint problems", *Statistics and computing*, Vol. 16, No. 2, pp. 203–213, 2006.

22. Barry, D. and J. A. Hartigan, "Product partition models for change point problems", *The Annals of Statistics*, pp. 260–279, 1992.

23. Adams, R. P. and D. J. MacKay, "Bayesian online changepoint detection", *arXiv preprint arXiv:0710.3742*, 2007.

24. Mirkovic, J. and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms", *ACM SIGCOMM Computer Communications Review*, Vol. 34, No. 2, pp. 39–53, 2004.

25. Yıldız, Ç., B. Kurt and T. Y. Ceritli, "Bayesian Change Point Model", `github.com/pilab-sigma/bcpm`, 2013, [Online; accessed 4-July-2017].

26. Dempster, A. P., N. M. Laird and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

27. McLachlan, G. and T. Krishnan, *The EM algorithm and extensions*, Vol. 382, John Wiley & Sons, 2007.

28. Kullback, S. and R. A. Leibler, "On information and sufficiency", *The annals of mathematical statistics*, Vol. 22, No. 1, pp. 79–86, 1951.

29. Lee, D. D. and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization.", *Nature*, Vol. 401, pp. 788–91, 1999.

30. Cemgil, A. T., "Bayesian inference for nonnegative matrix factorisation models", *Computational Intelligence and Neuroscience*, Vol. 2009, 2009.

31. Schmidt, M. N., O. Winther and L. K. Hansen, "Bayesian non-negative matrix factorization", *International Conference on Independent Component Analysis and Signal Separation*, pp. 540–547, Springer, 2009.

32. Barber, D., *Bayesian Reasoning and Machine Learning*, Cambridge University Press, New York, NY, USA, 2012.

33. Barber, D. and A. T. Cemgil, "Graphical models for time-series", *Signal Processing Magazine, IEEE*, Vol. 27, No. 6, pp. 18–28, 2010.

34. Bracegirdle, C. and D. Barber, "Switch-Reset Models : Exact and Approximate Inference", G. J. Gordon and D. B. Dunson (Editors), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, Vol. 15, pp. 190–198, Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011, `http://proceedings.mlr.press/v15/bracegirdle11a.html`.

35. Minka, T., *Estimating a Dirichlet distribution*, Tech. rep., 2000, `https://tminka.github.io/papers/dirichlet/minka-dirichlet.pdf`.

36. Rosenberg, J., H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and S. E, *RFC 3261: SIP: Session Initiation Protocol*, Tech. rep., IETF, 2002, `www.ietf.org/rfc/rfc3261.txt`.

37. Handley, M. and V. Jacobson, “SDP: Session Description Protocol”, , 1998.

38. Schulzrinne, H., S. Casner, R. Frederick and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications”, , 2003.

39. Zargar, S. T., J. Joshi and D. Tipper, “A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks”, *IEEE communications surveys & tutorials*, Vol. 15, No. 4, pp. 2046–2069, 2013.

40. Narayanan, A. and V. Shmatikov, “Robust de-anonymization of large sparse datasets”, *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pp. 111–125, IEEE, 2008.

41. Kurt, B., Ç. Yıldız, Y. T. Ceritli, M. Yamaç, M. Semerci, B. Sankur and A. T. Cemgil, “A Probabilistic SIP Network Simulation System”, *24th IEEE Conference on Signal Processing and Communications Applications (Accepted, in Turkish)*, 2016.

42. “Trixbox”, `http://www.fonality.com/trixbox`, 2016, [Online; accessed 29-April-2016].

43. “Nova V-SPY”, `http://www.netas.com.tr/en/`, 2016, [Online; accessed 29-April-2016].

44. Xu, Y. and R. Guérin, “On the robustness of router-based denial-of-service (DoS) defense systems”, *ACM SIGCOMM Computer Communication Review*, Vol. 35, No. 3, pp. 47–60, 2005.

45. Goldenberg, A., A. X. Zheng, S. E. Fienberg and E. M. Airoldi, “A Survey of Statistical Network Models”, *Foundations and Trends in Machine Learning*, Vol. 2, No. 2, pp. 129–233, 2010.

APPENDIX A: SIP Network Traffic Generation

Simulator is driven by a probabilistic generative model to recreate typical user behaviors in SIP networks, such as making calls, answering, rejecting or ignoring an incoming call, and holding an ongoing call. User actions generated by the Simulator are realized as actual SIP communications, where SIP messages are exchanged between UAs and Trixbox. Simulator omits the RTP messages carrying actual conversational data between users, since these do not pass through the SIP server, and are not relevant to the outcome of the simulation.

A.1. The Social Network

The relationship between users in the simulation is structured by a stochastic graph block model [45]. N users in the simulation are first divided into K different groups. The probability of a user to be in group k is denoted by the k'th element of a K dimensional vector π, which is a Dirichlet random variable with parameter χ, which is by default flat:

$$\pi \sim \mathcal{D}ir(\pi; \chi) \tag{A.1}$$

We use a $N \times K$ dimensional binary-valued matrix G to denote group identities, encoded in 1-of-K manner. In other words, each row g_n consists of $k-1$ zeros and a single one ($g_{n,k} = 1$ if and only if n'th user belongs to k'th group). Rows are sampled from a Categorical distribution with parameter π:

$$g_n \sim \mathcal{C}at(g_n; \pi) \tag{A.2}$$

A.2. Phone Books

First, a phone book b_n is generated for each user n to determine persons that the user is most likely to call. b_n is a vector of length N such that $b_{n,m}$ is the probability of user n calling user m whenever it decides to make a call. This vector is drawn from a Dirichlet distribution with a parameter α_n that is specific to each user.

$$b_n \sim \mathcal{D}ir(b_n; \alpha_n) \tag{A.3}$$

All entries except for n'th one of $\alpha_n \in \mathbb{R}^N$ are 1, and $\alpha_{n,n} = 0$. This means the phone book b_n is drawn from an (almost) flat Dirichlet distribution, but b_n itself is not flat. Each user n has different probabilities of calling other users.

Whenever user n decides to make a call, it selects a user c from its phone book to call from a Categorical distribution

$$c \sim \mathcal{C}at(c; b_n) \tag{A.4}$$

A.3. Registration

When the simulation starts, all users are offline. The amount of time user n spends before registration is denoted by r_n and generated as follows:

$$r_n \sim \mathcal{G}(r_n; 2, 10) \tag{A.5}$$

Furthermore, each user n sends a re-register packet with a fixed period t_n that is again sampled from Gamma distribution:

$$t_n \sim \mathcal{G}(t_n; 30, 10) \tag{A.6}$$

A.4. Call Rates

The rate of phone calls for each user n is determined by β_n, which is generated from Gamma distribution:

$$\beta_n \sim \mathcal{G}(\beta_n; \mu, 10) \tag{A.7}$$

During the simulation, whenever user n becomes idle, it waits for a random i_n seconds, which is drawn from Exponential distribution with rate β_n.

$$i_n \sim \mathcal{E}xp(i_n; \beta_n) \tag{A.8}$$

$$\mathcal{E}xp(i_n|\beta_n) = 1/\beta_n \exp(-i_n/\beta_n) \tag{A.9}$$

To generate low traffic, we set μ to 20 and $\mu = 5$ in case of high traffic.

A.5. Call Durations

Each user n has its own average call duration δ_n, drawn from Uniform distribution:

$$\delta_n \sim \mathcal{U}(\delta_n; 20, 200) \tag{A.10}$$

Some users tend to make short calls, while others prefer to talk longer. Whenever a call is set up between users n and m, they both sample the duration of the call independently and the one who wants to talk shorter hangs up. Therefore, we generate a call duration $d_{n,m}$ as follows

$$d_n|\delta_n \sim \mathcal{E}xp(d_n; \delta_n) \tag{A.11}$$

$$d_m|\delta_m \sim \mathcal{E}xp(d_m; \delta_m) \tag{A.12}$$

$$d_{n,m} = \min(d_n, d_m) \tag{A.13}$$

A.6. Call Responses

Every user n is associated with a call answering probability a_n and call holding probability h_n. We draw them from following uniform priors:

$$a_n \sim \mathcal{U}(a_n; 0.8, 1) \tag{A.14}$$

$$h_n \sim \mathcal{U}(h_n; 0, 0.1) \tag{A.15}$$

Whenever a user in available state receives a call, we draw a Bernoulli variable x_n with parameter a_n for the decision. The call is accepted if $x_n = 1$. If the user is already talking when she receives the call, we draw another binary random variable z_n with parameter h_n. If z_n turns out to be 1, she holds the other party and accepts the incoming call. Otherwise, incoming call is rejected.

$$x_n \sim \mathcal{BE}(x_n; a_n) \tag{A.16}$$

$$z_n \sim \mathcal{BE}(z_n; h_n) \tag{A.17}$$

The actions that users take for incoming calls is given in Figure A.1.

```
User $A$ calls user $B$
if $B$.status == “Available” then
    generate $x \in \mathcal{U}[0, 1)$
    if x $\leq$ $B$.answering_probability then
        return “ACCEPT”
    else
        return “REJECT”
    end if
else
    generate $z \in \mathcal{U}[0, 1)$
    if z $\leq$ $B$.holding_probability then
        return “HOLD_OTHER_PARTY”
    else
        return “BUSY”
    end if
end if
```

Figure A.1. Algorithm to generate responses to calls