Model-Based Clustering of Time Series Exhibiting Nonlinear Dynamics

A.B. Thesis

Model-Based Clustering of Time Series Exhibiting Nonlinear Dynamics

---

Alexander Lin

Advisers: Demba Ba, Pierre E. Jacob

Harvard University
Department of Computer Science

March 29th, 2019

Writing period

01. 01. 2019 – 04. 29. 2019

Readers

Demba Ba, Finale Doshi-Velez, Pierre E. Jacob

Advisers

Demba Ba, Pierre E. Jacob

Abstract

We consider the problem of time series clustering, which asks how to cluster objects with dynamically changing properties over time into sensible groups in an unsupervised manner. In tackling this problem, we pursue a model-based approach, introducing a mixture of nonlinear state-space models and corresponding inference algorithms. In particular, we develop a Monte Carlo expectation-maximization algorithm for finite mixtures and a Markov chain Monte Carlo algorithm for both finite and infinite mixtures. We apply our work to a variety of examples in computational neuroscience and demonstrate the utility of time series clustering in these real-world settings.

Contents

1 Introduction

Time series data are ubiquitous in everyday life. From daily stock returns to hourly seismic activity to second-by-second recordings of a patient's heart rate, it is important to develop methodologies that can process and analyze data characterized by significant changes over time. Over the years, the field of time series analysis has witnessed much growth, with advances in fundamental problems such as imputation, forecasting, anomaly detection, classification, segmentation, signal estimation, and many more (Shumway and Stoffer, 2017).

Time series clustering is one such task involving the unsupervised categorization of various objects that each exhibits some dynamically changing component(s) recorded as time series data. Based on the evolution of their respective time series, the objects are grouped into separate *clusters* in a way such that objects within the same cluster exhibit similar temporal dynamics that are sufficiently different from those of objects in other clusters.

The ability to cluster multiple time series has several benefits. Among these include creating the clustering to uncover previously unknown relationships between the objects of interest, and using an existing clustering to categorize a newly observed object into a group with known properties. Some examples of real-world time series clustering problems include:

- Determining common underlying trends among sets of stocks (Guo et al., 2008),
- Identifying pathological cases from streams of medical activity (Gullo et al., 2012),
- Finding climactic patterns in temperature or pressure data (Ji et al., 2013),
- Discovering neuron communities through analysis of spike trains (Humphries, 2011).

Domains for applying time series clustering cover a wide breadth of interdisciplinary fields, such as aviation, psychology, speech processing, finance, medicine, and robotics; Aghabozorgi et al. (2015) and Liao (2005) present comprehensive reviews of time series clustering applications.

1.1 A Motivating Example from Neuroscience

In computational neuroscience, a common type of recorded data is the *neuronal firing sequence* (also known as a *spike train*), which is a stream of binary 0-1 values over time that indicate when a neuron fired. More specifically, at every time point, we record a '1' if the neuron fired and a '0' otherwise. Typically, this data is recorded at the millisecond resolution (Boyden et al., 2005). Fundamental questions in neuroscience are built on how to efficiently analyze collections of these firing sequences from various neurons in the brain. The field of *neural coding*, for instance, is concerned with how we can characterize the relationship between patterns in these sequences and exogenous stimuli experienced by humans and animals (Doya et al., 2007).

We can think of a firing sequence from a single neuron as a time series. Specifically, this is a time series that exhibits *nonlinear* dynamics, because each observed point in time may depend on previously observed points, but this relationship is nonlinear. Clustering firing sequences from different neurons effectively clusters the neurons themselves. In a data set comprising hundreds to thousands of neuronal time series (Brown et al., 2004), such a clustering can provide insights into how neural computation is implemented at the level of groups of neurons. For instance, if a cluster of neurons all respond the same way to a stimulus, then it is reasonable to hypothesize that these neurons comprise a network of interactions in the brain. Thus, the ability to cluster times series is a useful tool in neuroscience research, because it can potentially reveal previously unknown relationships between groups of neurons, which can fuel the development of biological hypotheses about the brain. We give concrete examples of these applications throughout the remainder of this manuscript.

1.2 Problem Specification

Formally, the problem of time series clustering assumes we have a set of N *objects* indexed by $n = 1, \ldots, N$. From dynamically changing attributes of the objects, we observe N respective *time series* $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$. Each object's series $\boldsymbol{y}^{(n)} = \{y_1^{(n)}, \ldots, y_{T_n}^{(n)}\}$ is a vector of observations over T_n discrete time units. The dependence of T_n on n indicates that different time series can be of different lengths. As an example, consider multiple time series of daily stock returns for various companies in which the period of data collection ends after several companies go out of business.

The time series may either be univariate or multivariate, depending upon the specific

application at hand; that is, each time-point observation satisfies $y_t^{(n)} \in \mathcal{Y}$, where the set $\mathcal{Y}$ may have a single dimension (e.g. $\mathcal{Y} = \mathbb{R}$) or multiple dimensions (e.g. $\mathcal{Y} = \mathbb{R}^p$ for $p > 1$). For instance, the field goal percentage of a basketball player over the length of an NBA season is single-dimensional, whereas the geographic coordinates (i.e. latitude and longitude) of a plane over the duration of a flight are multidimensional.

The task of *clustering* time series requires identifying K *clusters* labeled $\{1, \ldots, K\}$ and N *cluster assignments* $Z = \{z^{(1)}, \ldots, z^{(N)}\}$ such that each $z^{(n)} \in \{1, \ldots, K\}$. Thus, the cluster assignments partition the set of all objects $\{1, \ldots, N\}$ into K disjoint sets. In contrast to supervised machine learning literature, the ground-truth cluster assignments are not known a priori, so the cluster assignments Z are solely inferred from the time series data $\boldsymbol{Y}$ in an unsupervised fashion. The number of clusters K may be supplied or not supplied in advance, depending upon the application of interest. The potential lack of specification for K adds an additional layer of unsupervision to the problem.

The partition, also known as the *clustering*, is done so that two objects $\{n, n'\}$ within the same cluster exhibit similar temporal dynamics $\{\boldsymbol{y}^{(n)}, \boldsymbol{y}^{(n')}\}$, while two objects in different clusters exhibit sufficiently different dynamics. The specific mechanism by which the clustering occurs is termed the *clustering method*, and there exist several such methods for clustering time series in the literature, which are elaborated on in Section 1.3.

The subject of this manuscript is a newly proposed method for clustering multiple time series – especially ones that exhibit some sort of nonlinear dynamics. The method models the series as the output of a generative mixture of state-space models and then performs statistical inference on the latent variables of the mixture. These latent variables are then used to determine the final clustering of the time series and their respective objects. Parts of this manuscript is published in Lin et al. (2019), which is referenced throughout the ensuing chapters.

1.3 Related Work

Existing methods in the rich literature on clustering multiple time series can be roughly categorized into three principal types – feature-based methods, shape-based methods, and model-based methods. This categorization was first established by Liao (2005) and later reiterated by Aghabozorgi et al. (2015) in their respective surveys of time series clustering methods.

Typical feature-based methods are two-stage processes that first extract a set of features from each time series and then perform clustering in feature space using standard

algorithms. For example, Guo et al. (2008) cluster stock time series data by constructing lower-dimensional representations with independent component analysis and then using the k-means algorithm to partition the resultant values. Similarly, Fu et al. (2001) reduce time series data to feature vectors comprised of "perceptually important points" and then utilize a modified version of the self-organizing map algorithm. A major advantage of feature-based approaches is that they do not require engineering a novel algorithm for handling time series in the clustering step. Thus, they can take advantage of classic clustering procedures, such as k-means (Wilpon and Rabiner, 1985; Owsley et al., 1997; Vlachos et al., 2003) and hierarchical agglomerative clustering (Shaw and King, 1992; Wang et al., 2006). However, in many cases, deciding on the most appropriate method to extract features can be difficult and highly application-specific. Furthermore, since the clustering is not done on the actual data themselves, key information present in the original data could be lost during the feature extraction step, thereby leading to potentially poor clustering results.

The appeal of shape-based methods is that they process the raw time series data themselves. Instead of employing an intermediate feature extraction step, they directly leverage a time series-specific distance metric in combination with standard algorithms to cluster the multiple time series. Popular metrics include Euclidean distance (Faloutsos et al., 1994), dynamic time warping (Sakoe, 1971), longest common subsequence (Vlachos et al., 2002), edit distance (Chen and Ng, 2004), and cross-correlation based distances (Golay et al., 1998). Standard clustering algorithms that have been used in conjunction with these metrics include k-means (Niennattrakul and Ratanamahatana, 2007), k-medioids (Liao et al., 2002), fuzzy c-means (Golay et al., 1998; Möller-Levet et al., 2003)), and hierarchical agglomerative clustering (Kakizawa et al., 1998; Van Wijk and Van Selow, 1999; Kumar et al., 2002; Shumway, 2003). Yet, although they are popular, many shape-based methods experience shortcomings in not having a straightforward way to deal with multiple time series with varying lengths, missing values, and/or unaligned recordings. Furthermore, while feature-based methods and shape-based methods are relatively simple to implement, they cannot be used to perform statistical inference on the parameters of a physical model by which the time series are generated.

Model-based approaches are advantageous because they establish a probabilistic framework outlining the generative process of the multiple time series. Thus, we can make powerful statements such as: *Under the model, two time series* $\{\boldsymbol{y}^{(n)}, \boldsymbol{y}^{(n')}\}$ *have a posterior probability* p *of being in the same cluster*, where p can be found using inference techniques. The typical generative model of choice for clustered data is the *mixture model*,

which is reviewed in Section 2.4. Performing statistical inference on the mixture model's parameters yields posterior probability distributions over the cluster assignments Z.

In existing literature, many model-based approaches for clustering multiple time series employ Bayesian mixtures over the parameters of established time series models. Examples of these time series models have included generalized autoregressive conditional heteroskedasticity (GARCH) models (Bauwens and Rombouts, 2007), integer valued autoregressive (INAR) models (Roick et al., 2019), and temporally reweighted Chinese restaurant process (TRCRP) models (Saad and Mansinghka, 2018).

State-space models are a well-known and flexible class of models for handling time series data (Durbin and Koopman, 2012). Several existing model-based approaches for clustering time series employ a mixture of linear Gaussian state-space models (LGSSM), which are reviewed in Section 2.5.1. Xiong and Yeung (2002) use the expectation-maximization algorithm for finite mixtures of autoregressive moving average (ARMA) models, which are a subset of LGSSM. Inoue et al. (2006) and Chiappa and Barber (2007) both examine general LGSSM mixtures and use Gibbs sampling and variational Bayes, respectively, for posterior inference. Nieto-Barajas and Contreras-Cristán (2014) and Middleton (2014) extend the framework by using a Dirichlet process mixture to infer the number of clusters and Gibbs sampling for full posterior inference. In all of these cases, the linear-Gaussian assumption is crucial for tractable inference; it enables exact evaluation of the likelihood using a Kalman filter and the ability to sample exactly from the state sequences underlying each of the time series. For nonlinear and non-Gaussian state-space models, this likelihood cannot be evaluated in closed form and exact sampling is not possible.

1.4 Overview of Thesis

The main contribution of this manuscript is to establish a model and corresponding inference algorithms for clustering multiple time series that exhibit *nonlinear* dynamics. Nonlinear time series are arguably more common than their linear counterparts and appear in a large breadth of examples, ranging from human breath rates to NMR laser data to foetal electrocardiogram recordings (Kantz and Schreiber, 2004). By relaxing the linear-Gaussian assumption used in much of previous literature, we make a particular form of model-based clustering more generalizable and more appropriate for a wider range of applications.

Chapter 2 provides some background of terminology and methods that are employed in the remainder of the manuscript. Chapters 3 and 4 introduce the primary results. These

results include:

- Introducing a mixture model of nonlinear state-space models for clustering time series (Section 3.1; Section 4.1),
- Deriving a Monte Carlo expectation-maximization algorithm for finding maximum a posteriori solutions in the case of finite mixtures (Section 3.2),
- Deriving a Markov chain Monte Carlo algorithm for performing full Bayesian inference in both finite and infinite mixtures (Section 4.2),
- Applying the framework and algorithms to a number of applications in neuroscience-related problems, such as epilepsy recognition (Section 3.3.1), neuronal firing clustering (Section 3.3.2), identification of stimulus-locked response profiles (Section 4.5.1), and analysis of interacting neuronal firing regions (Section 4.5.2).

The Markov chain Monte Carlo algorithm in Chapter 4 is the primary subject of Lin et al. (2019) and employs recent innovations in Monte Carlo, including particle marginal Metropolis-Hastings (Andrieu et al., 2010) and controlled sequential Monte Carlo (Heng et al., 2017). Finally, Chapter 5 concludes the manuscript with a discussion of implications and future work.

2 Background

This chapter establishes notation and reviews some key concepts that will be used throughout the rest of the manuscript. Section 2.1 introduces the concept of generative modeling, which forms the foundation of our methodologies. Then, Section 2.2 and Section 2.3 review the expectation-maximization algorithm and Markov chain Monte Carlo methods, respectively. Next, Section 2.4 elaborates on finite and infinite mixture models and relevant inference algorithms. And finally, Section 2.5 touches on state-space models and recently developed particle-based methods for handling nonlinear dynamics within these models.

2.1 Generative Models

Generative models for observed data $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$ are statistical models of the joint probability distribution $p(\mathcal{Z}, \boldsymbol{Y})$, where $\mathcal{Z}$ is a set of latent variables. The *generative* part of the model comes from the factorization $p(\mathcal{Z}, \boldsymbol{Y}) = p(\mathcal{Z}) \cdot p(\boldsymbol{Y} \mid \mathcal{Z})$, which suggests a natural process for the generation of $\boldsymbol{Y}$ – i.e. sample $\mathcal{Z} \sim p(\mathcal{Z})$ and then sample from the conditional distribution $\boldsymbol{Y} \sim p(\boldsymbol{Y} \mid \mathcal{Z})$. It is popular to picture this generative process in the form of a directed acyclic graph (DAG) that is known as a Bayesian network or graphical model (Figure 1).

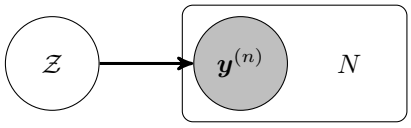

Figure 1: A graphical model which assumes $\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}$ are conditionally independent given $\mathcal{Z}$. Observed variables are indicated in gray.

Statistical inference for a generative model involves finding (or approximating) the posterior distribution $p(\mathcal{Z} \,|\, \boldsymbol{Y})$, which is a difficult task in many instances. The *expectation-maximization algorithm* can be used to provide a point estimate of the posterior, such as the maximum a posteriori estimate. A dominant methodology for full posterior inference is *Markov chain Monte Carlo*. We briefly review these methods in the next two sections.

In the case of time series clustering, we generatively model the raw time series as the observed data and the cluster assignments as latent variables. Thus, inference on the

model parameters allows us to find posterior probabilities for the cluster assignments. Further details can be found in Chapters 3 and 4.

2.2 Expectation-Maximization

The expectation-maximization (EM) algorithm (Dempster et al., 1977) is an optimization method for finding maximum likelihood estimates (MLE) and/or maximum a posteriori (MAP) estimates of unobserved parameters $\boldsymbol{\Theta}$ in a generative model with observed data $\boldsymbol{Y}$ and latent variables $\mathcal{Z}$. In the presence of latent variables, the log-posterior $\log p(\boldsymbol{\Theta} \mid \boldsymbol{Y}) \simeq \log p(\boldsymbol{\Theta}) + \log p(\boldsymbol{Y} \mid \boldsymbol{\Theta}) = \log p(\boldsymbol{\Theta}) + \log \left[\sum_{\mathcal{Z}} p(\boldsymbol{Y}, \mathcal{Z} \mid \boldsymbol{\Theta})\right]$ is intractable to optimize over in general, where $\simeq$ denotes equality up to an additive constant with respect to $\boldsymbol{\Theta}$. Thus, EM takes the approach of iteratively optimizing $\log p(\boldsymbol{\Theta}^{\langle i \rangle} \mid \boldsymbol{Y})$ until convergence for iterations $i = 1, 2, 3, \ldots$ in a two-step algorithm comprised of the expectation step and the maximization step (Algorithm 1).

Algorithm 1 `ExpectationMaximization`$(\boldsymbol{Y}, \boldsymbol{\Theta}^{\langle 0 \rangle}, \epsilon_0)$

1: Set $i = 0$
2: Set $\epsilon = \infty$
3: **while** $\epsilon_0 < \epsilon$ **do**
// Expectation Step
4: Define expected complete data log-likelihood $\ell(\boldsymbol{\Theta} | \boldsymbol{\Theta}^{\langle i \rangle}) = \mathbb{E}_{\mathcal{Z} \mid \boldsymbol{Y}, \boldsymbol{\Theta}^{\langle i \rangle}}[\log p(\boldsymbol{Y}, \mathcal{Z} | \boldsymbol{\Theta})]$
5: Define expected complete data log-posterior $R(\boldsymbol{\Theta} \mid \boldsymbol{\Theta}^{\langle i \rangle}) = \log p(\boldsymbol{\Theta}) + \ell(\boldsymbol{\Theta} \mid \boldsymbol{\Theta}^{\langle i \rangle})$
// Maximization Step
6: Optimize $\boldsymbol{\Theta}^{\langle i+1 \rangle} = \arg\max_{\boldsymbol{\Theta}} R(\boldsymbol{\Theta} \mid \boldsymbol{\Theta}^{\langle i \rangle})$
7: Compute $\epsilon = \|\boldsymbol{\Theta}^{\langle i+1 \rangle} - \boldsymbol{\Theta}^{\langle i \rangle}\|_2$.
8: Set $i = i + 1$
9: **end while**
10: **return** $\boldsymbol{\Theta}^{\langle i \rangle}$

It is possible to prove that $\log p(\boldsymbol{\Theta}^{\langle i \rangle} \mid \boldsymbol{Y}) \geq \log p(\boldsymbol{\Theta}^{\langle i-1 \rangle} \mid \boldsymbol{Y})$ for all $i = 1, \ldots, I$, thereby establishing the local optimality of the final solution $\boldsymbol{\Theta}^{\langle I \rangle}$ (Wu et al., 1983). However, there is no general guarantee of how close this local optimum is to the global optimum $\arg\max_{\boldsymbol{\Theta}} \log p(\boldsymbol{\Theta} \mid \boldsymbol{Y})$, and this is a well-known feature of EM.

2.2.1 Monte Carlo Expectation-Maximization

An assumption of the classic EM algorithm is that the expected complete data log-likelihood $\ell(\boldsymbol{\Theta} \mid \boldsymbol{\Theta}^{\langle i \rangle})$ of Algorithm 1 (Line 4) is computationally tractable. However,

this quantity is an integral with respect to the posterior $p(\mathcal{Z} \mid \boldsymbol{Y}, \boldsymbol{\Theta}^{\langle i \rangle})$, which may be impossible to calculate analytically for certain generative models. Monte Carlo Expectation-Maximization (MCEM) attempts to rectify this problem by approximating $\ell(\boldsymbol{\Theta} \mid \boldsymbol{\Theta}^{\langle i \rangle})$ with a Monte Carlo estimate; that is $\ell(\boldsymbol{\Theta} \mid \boldsymbol{\Theta}^{\langle i \rangle}) \approx 1/S \cdot \sum_{s=1}^{S} \log p(\boldsymbol{Y}, \mathcal{Z}^s \mid \boldsymbol{\Theta})$, where $\mathcal{Z}^s \sim p(\mathcal{Z} \mid \boldsymbol{Y}, \boldsymbol{\Theta}^{\langle i \rangle})$ for $s = 1, \ldots, S$. MCEM ultimately allows us to apply the EM algorithm to a wider range of generative models (Caffo et al., 2005).

2.3 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a family of algorithms for sampling from a distribution of interest when it is difficult to do so directly. The general idea is to construct an ergodic Markov chain whose stationary distribution is exactly the distribution of interest. Thus, MCMC is highly useful for using samples to approximate the (potentially intractable) posterior distribution of a generative model.

In this section, we review common MCMC methods, i.e. Metropolis-Hastings (Section 2.3.1) and Gibbs Sampling (Section 2.3.2). MCMC inference algorithms for specific generative models, such as mixture models and state-space models, are covered in the next sections.

2.3.1 Metropolis-Hastings

The Metropolis-Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970) is a remarkable invention that allows one to sample from any target distribution $q(\mathcal{Z})$ as long as we are able to compute a function $f(\mathcal{Z})$ that is proportional to this density $q(\mathcal{Z})$. Of course, $f(\mathcal{Z}) = q(\mathcal{Z})$ is a valid proportional function, yet $q(\mathcal{Z})$ may not always be tractable to compute.

For instance, in the generative model setting, the desired target distribution is usually the posterior over latent variables, i.e. $q(\mathcal{Z}) = p(\mathcal{Z} \mid \boldsymbol{Y}) = p(\mathcal{Z}, \boldsymbol{Y}) / \int_{\mathcal{Z}} p(\mathcal{Z}, \boldsymbol{Y})$. The integral in the denominator often makes $q(\mathcal{Z})$ impossible to compute, yet this denominator is also constant with respect to $\mathcal{Z}$. Thus, $q(\mathcal{Z})$ is proportional to $f(\mathcal{Z}) = p(\mathcal{Z}, \boldsymbol{Y})$, the joint distribution which is often tractable and allows us to use the MH algorithm to sample from the posterior $p(\mathcal{Z} \mid \boldsymbol{Y})$.

In addition to the target q, the MH algorithm requires us to specify a proposal distribution $r(\mathcal{Z}' \mid \mathcal{Z})$ for the Markov chain and a starting point $\mathcal{Z}^0$. We summarize MH in Algorithm 2. The output of the algorithm comprises samples $\mathcal{Z}^1, \ldots, \mathcal{Z}^L$ for $L \geq 1$.

The Markov chain almost never reaches stationarity immediately, so in practice, it is customary to throw away the first few samples as part of a burn-in period of length B, where $1 \leq B \leq L$. Thus, in subsequent tasks, the samples $\mathcal{Z}^{B+1}, \ldots, \mathcal{Z}^{L}$ are used as an approximation of the target distribution.

Algorithm 2 `MetropolisHastings`$(f, r, L, \mathcal{Z}^0)$

for $\ell = 1, \ldots, L$ **do**
  Draw proposal $\mathcal{Z}' \sim r(\mathcal{Z} \mid \mathcal{Z}^{\ell-1})$.
  Compute acceptance probability $a = \min\left(1, \frac{f(\mathcal{Z}') \cdot r(\mathcal{Z}^{\ell-1} \mid \mathcal{Z}')}{f(\mathcal{Z}^{\ell-1}) \cdot r(\mathcal{Z}' \mid \mathcal{Z}^{\ell-1})}\right)$.
  Sample $u \sim \text{Uniform}(0, 1)$.
  if $u \leq a$ **then**
    Accept proposal $\mathcal{Z}^{\ell} = \mathcal{Z}'$.
  else
    Reject proposal $\mathcal{Z}^{\ell} = \mathcal{Z}^{\ell-1}$.
  end if
end for
return samples $\mathcal{Z}^1, \ldots, \mathcal{Z}^L$

2.3.2 Gibbs Sampling

Gibbs sampling (Geman and Geman, 1987) is a special case of Metropolis-Hastings for sampling from the joint distribution of a set of random variables in which the proposal is accepted with probability 1. The basic idea is that we can sample from an M-dimensional joint distribution $q(\mathcal{Z})$ over many variables $\mathcal{Z} = \{\mathcal{Z}_1, \ldots, \mathcal{Z}_M\}$ by iteratively sampling from the univariate conditional distributions $q_m(\mathcal{Z}_m \mid \mathcal{Z}_1, \ldots, \mathcal{Z}_{m-1}, \mathcal{Z}_{m+1}, \ldots, \mathcal{Z}_M)$ for $m = 1, \ldots, M$. Algorithm 3 reviews the Gibbs sampling algorithm, which outputs samples $\{\mathcal{Z}^1, \ldots, \mathcal{Z}^L\}$ for the joint distribution. As with the MH algorithm, burn-in is typically applied to these samples.

Algorithm 3 `GibbsSampling`$(\{q_1, \ldots, q_M\}, L, \mathcal{Z}^0)$

for $\ell = 1, \ldots, L$ **do**
  for $m = 1, \ldots, M$ **do**
    Draw from conditional $\mathcal{Z}_m^{\ell} \sim q_m(\mathcal{Z}_m \mid \mathcal{Z}_1^{\ell}, , \ldots, \mathcal{Z}_{m-1}^{\ell}, \mathcal{Z}_{m+1}^{\ell-1}, \ldots, \mathcal{Z}_M^{\ell-1})$.
  end for
end for
return samples $\mathcal{Z}^1, \ldots, \mathcal{Z}^L$

2.4 Mixture Models for Clustered Data

Mixture models are popular generative models for clustered data (McLachlan et al., 2000). They follow a general structure of assuming that N observed data points $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$ have latent *cluster assignments* $Z = \{z^{(1)}, \ldots, z^{(N)}\}$ where each $z^{(n)} \in \{1, \ldots, K\}$. Usually, the number of clusters K is much fewer than the number of observations N. Furthermore, it is typically assumed that all data points assigned to the same cluster k are independent and identically distributed (i.i.d.) realizations of a generative process parameterized by shared *cluster parameters* $\boldsymbol{\theta}^{(k)}$, for $k = 1, \ldots, K$. The set of all cluster parameters is denoted $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(K)}\}$. In this section, we summarize the finite mixture model and the infinite mixture model for clustered data.

2.4.1 Finite Mixture Model

The Bayesian finite mixture model is a generative model of the following form:

$$\begin{aligned} \boldsymbol{q} &\sim \text{Dirichlet}(\boldsymbol{\alpha}), && (1) \\ z^{(n)} &\sim \text{Categorical}(\boldsymbol{q}), && 1 \le n \le N, \\ \boldsymbol{\theta}^{(k)} &\sim G, && 1 \le k \le K, \\ \boldsymbol{y}^{(n)} \mid z^{(n)} = k &\sim p(\boldsymbol{y}^{(n)} \mid \boldsymbol{\theta}^{(k)}), && 1 \le n \le N, \end{aligned}$$

where $\boldsymbol{\alpha} = \{\alpha^{(1)}, \ldots, \alpha^{(K)}\}$ is a K-dimensional vector of pseudo-counts and G is a base distribution for the parameters. The function p characterizes the distribution over $\boldsymbol{y}^{(n)}$ given that object n belongs to cluster k. This function will change depending on what type of mixture model we are considering.

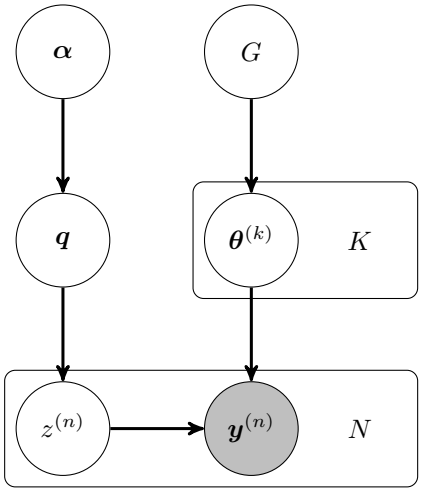

Figure 2: The graphical model of the finite mixture model.

Given observations $\boldsymbol{Y}$, a common objective is to perform posterior inference on this model to characterize posterior distributions for $Z \mid \boldsymbol{Y}$ and/or $\boldsymbol{\Theta} \mid \boldsymbol{Y}$. In many inference procedures, it is common to integrate out the vector $\boldsymbol{q}$ due to conjugacy between the Dirichlet and Categorical distributions. A graphical model of the finite mixture model is given in Figure 2.

In the Gaussian mixture model (GMM) setting, each observed data point $\boldsymbol{y}^{(n)}$ is a K-dimensional variable that is normally distributed when conditioned on cluster membership.

The cluster parameters are means and covariances of a Gaussian distribution, i.e. $\boldsymbol{\theta}^{(k)} = \{\boldsymbol{\mu}^{(k)}, \boldsymbol{\Sigma}^{(k)}\}$ and $p(\boldsymbol{y}^{(n)} \mid \boldsymbol{\theta}^{(k)}) = \mathcal{N}(\boldsymbol{y}^{(n)} \mid \boldsymbol{\mu}^{(k)}, \boldsymbol{\Sigma}^{(k)})$ for all k. A natural base distribution G that upholds conjugacy places a Gaussian $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ prior on $\boldsymbol{\mu}^{(k)}$ and an inverse Wishart $\mathcal{W}^{-1}(\boldsymbol{\Phi}, \nu)$ prior on $\boldsymbol{\Sigma}^{(k)}$, where $\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, \boldsymbol{\Phi}, \nu$ are all hyperparameters.

For the GMM, there exists a well-known EM algorithm for inferring an estimate $\boldsymbol{\Theta}^*$ that maximizes $p(\boldsymbol{\Theta}^* \mid \boldsymbol{Y})$ and finding the corresponding posterior distribution over cluster assignments $p(Z \mid \boldsymbol{Y}, \boldsymbol{\Theta}^*)$. There also exists a well-known Gibbs sampling algorithm for full Bayesian inference to determine $p(\boldsymbol{\Theta}, Z \mid \boldsymbol{Y})$. Both of these algorithms benefit from Gaussian-Gaussian and Gaussian-inverse Wishart conjugacy, and are commonly used in clustering applications (Murphy, 2012).

2.4.2 Infinite Mixture Model

An often-noted limitation of the finite mixture model is that a practitioner must specify the number of clusters K before running an inference algorithm. However, in many applications, this number may be unknown or unavailable. The infinite mixture model (Ferguson, 1973; Neal, 2000) attempts to rectify this problem by letting the number of clusters K also be a random variable in the model.

There are other ways to deal with this limitation, such as using model selection criteria to pick a single model among multiple finite mixtures with different values of K (Celeux and Soromenho, 1996) or using a mixture of finite mixtures (Miller and Harrison, 2018). We choose to explore the infinite mixture model, because it is a natural extension of the finite mixture model and as a result, shares similar inference techniques.

Dirichlet Process

The Dirichlet process (DP) (Ferguson, 1973) is at the heart of the infinite mixture model. Whereas the Dirichlet distribution provides a prior over all K-dimensional discrete distributions, the DP provides a prior over all discrete distributions in general. The DP is parameterized by an inverse-variance parameter α and a base distribution G. A sample from $\mathrm{DP}(\alpha, G)$ is an almost surely discrete distribution Q, in which the number of distinct values within $N \to \infty$ draws from Q (i.e. K) is random. Rewriting the finite mixture model of Equation (1), the Dirichlet process infinite mixture model can be written as:

$$\begin{aligned} Q &\sim \mathrm{DP}(\alpha, G), \\ \tilde{\boldsymbol{\theta}}^{(n)} &\sim Q, \qquad\qquad 1 \leq n \leq N, \end{aligned} \tag{2}$$

$$\boldsymbol{y}^{(n)} \mid \tilde{\boldsymbol{\theta}}^{(n)} \sim p(\boldsymbol{y}^{(n)} \mid \tilde{\boldsymbol{\theta}}^{(n)}), \qquad 1 \leq n \leq N.$$

In this setting, the parameters are indexed by the observation index n instead of by the cluster index k. Two objects $\{n, n'\}$ are in the same cluster if $\tilde{\boldsymbol{\theta}}^{(n)} = \tilde{\boldsymbol{\theta}}^{(n')}$.

Chinese Restaurant Process

Although the Dirichlet process mixture model of Equation (2) provides randomness for K, it is difficult to directly perform inference on this model due to the infinite dimensionality of Q. The Chinese restaurant process (CRP) representation of the DP integrates out this intermediary distribution Q (Neal, 2000). To generate a sample from the CRP, we can imagine the data points $1, \ldots, N$ as customers arriving successively at a restaurant. Given hyperparameter α, each customer n sits at an already occupied table k with probability $N^{(k)}/(n-1+\alpha)$, where $N^{(k)}$ is the number of people already sitting at table k, and starts a new table with probability $\alpha/(n-1+\alpha)$. Thus, the customers' table assignments $z^{(1)}, \ldots, z^{(N)}$, along with the number of tables K, are random variables. Each table has an associated value $\boldsymbol{\theta}^{(k)}$, which is independently drawn from a base distribution G.

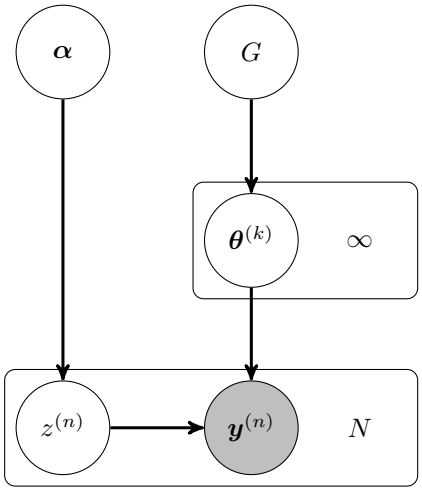

Figure 3: The graphical model of the infinite mixture model under the CRP representation.

In the mixture model, the CRP allows us to separate the process of assigning a cluster (i.e. table) to each n from the process of choosing a parameter (i.e. table value) for each cluster. The result is a form that is similar to the finite mixture model, but we do not need to choose K a priori:

$$\begin{aligned} z^{(1)}, \ldots, z^{(N)} &\sim \mathrm{CRP}(\alpha, N), && (3) \\ \boldsymbol{\theta}^{(k)} &\sim G, && 1 \leq k < \infty, \\ \boldsymbol{y}^{(n)} \mid z^{(n)} = k &\sim p(\boldsymbol{y}^{(n)} \mid \boldsymbol{\theta}^{(k)}), && 1 \leq n \leq N. \end{aligned}$$

A graphical model is depicted in Figure 3. In this case, the number of clusters K is simply the number of distinct values in $Z = \{z^{(1)}, \ldots, z^{(N)}\}$, where each $z^{(n)} \in \{1, \ldots, K\}$. Even though an infinite number of $\boldsymbol{\theta}^{(k)}$ are technically drawn from G, inference algorithms

only need to keep track of the finite number of cluster parameters that are involved in the generation of $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$. Nonetheless, inference is not necessarily easy to do. To the best of the author's knowledge, there is no straightforward method of using the EM algorithm for this model, although there are attempts at working with a truncated version (Kimura et al., 2013). In MCMC, there are several proposed methods for sampling from the posterior distribution of the infinite mixture model, many of which originated in Neal (2000). Neal's Algorithm 8 is one of the most general, as it handles the case in which there is no conjugacy between G and $p(\boldsymbol{y}^{(n)} \mid \boldsymbol{\theta}^{(k)})$.

2.5 State-Space Models for Time Series Data

State-space models are frequently used as generative models of time series data $\boldsymbol{y} = \{y_1, \ldots, y_T\}$ and appear in a wide variety of real-world applications (Durbin and Koopman, 2012). Given parameters $\boldsymbol{\theta}$, state-space models introduce an underlying latent process $\boldsymbol{x} = \{x_1, \ldots, x_T\}$ and assume the following form:

$$\begin{aligned} x_1 \mid \boldsymbol{\theta} &\sim h(x_1; \boldsymbol{\theta}), && \\ x_t \mid x_{t-1}, \boldsymbol{\theta} &\sim f(x_{t-1}, x_t; \boldsymbol{\theta}), && 1 < t \leq T, \\ y_t \mid x_t, \boldsymbol{\theta} &\sim g(x_t, y_t; \boldsymbol{\theta}), && 1 \leq t \leq T, \end{aligned} \tag{4}$$

where f is some *state transition density*, g is some *state-dependent likelihood*, and h is some *initial distribution*. The Bayesian setting places a prior G on $\boldsymbol{\theta}$. A graphical model is given in Figure 4. In this section, we summarize types of state-space models, as well as relevant inference algorithms.

2.5.1 Linear-Gaussian State-Space Models

The linear-Gaussian state-space model (LGSSM) assumes that f, g, and h are all Gaussian densities. Let the dimension of x_t be m and the dimension of y_t be p. In the most basic form, given parameters $\boldsymbol{\theta} = \{A, B, C, D, \mu_0, \Sigma_0\}$, where $A \in \mathbb{R}^{m \times m}, B \in \mathbb{R}^{m \times m}, C \in \mathbb{R}^{q \times m}, D \in \mathbb{R}^{q \times q}, \mu_0 \in \mathbb{R}^m$, and $\Sigma_0 \in \mathbb{R}^{m \times m}$, the LGSSM follows:

$$\begin{aligned} x_1 \mid \boldsymbol{\theta} &\sim \mathcal{N}(\mu_0, \Sigma_0), && \\ x_t \mid x_{t-1}, \boldsymbol{\theta} &\sim \mathcal{N}(Ax_{t-1}, B), && 1 < t \leq T, \\ y_t \mid x_t, \boldsymbol{\theta} &\sim \mathcal{N}(Cx_t, D), && 1 \leq t \leq T. \end{aligned} \tag{5}$$

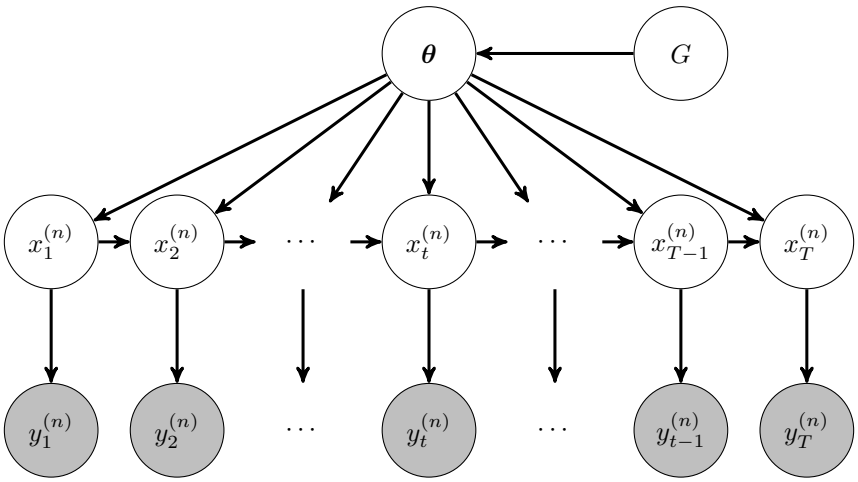

Figure 4: The graphical model of the state-space model.

The linear-Gaussian assumption makes exact inference computationally tractable due to conjugacy. For instance, there exists an EM algorithm for maximum likelihood estimation of $\boldsymbol{\theta}$ (Ghahramani and Hinton, 1996), which can be easily extended to MAP estimation if G is an inverse-Wishart distribution. There also exists a straightforward Gibbs sampling algorithm for full Bayesian inference of $p(\boldsymbol{x}, \boldsymbol{\theta} \mid \boldsymbol{y})$ in the linear-Gaussian setup (Carter and Kohn, 1994). Many of these methods take implicit advantage of the *Kalman filter* (Swerling, 1958; Kalman, 1960), which allows for exact computation of the parameter likelihood $p(\boldsymbol{y} \mid \boldsymbol{\theta})$.

2.5.2 Nonlinear State-Space Models

Nonlinear state-space models relax the linear-Gaussian assumption, allowing f, g, and h to take any form as distributions involving some nonlinearity. Inference then becomes a more difficult task. One early attempt to use the EM algorithm substitutes Gaussian radial basis function approximators for the nonlinearities during the M-step (Ghahramani and Roweis, 1999). This reduces the optimization problem to a solvable system of linear equations, yet depending on the specific nonlinearities, the approximation can be arbitrarily poor.

In MCMC literature, a recent innovation known as the *particle marginal Metropolis-Hastings* (PMMH) algorithm (Andrieu et al., 2010) allows one to construct a Markov chain with a stationary distribution exactly equal to $p(\boldsymbol{x}, \boldsymbol{\theta} \mid \boldsymbol{y})$ for nonlinear state-space models, provided that they have access to an algorithm which computes a Monte Carlo estimate of the parameter likelihood $\hat{p}(\boldsymbol{y} \mid \boldsymbol{\theta})$. It is an improvement over previously existing Gibbs sampling algorithms for accomplishing this task (Kim et al., 1999). The basic idea

of PMMH is to embed a *particle filter*, which is designed to produce such estimates, inside the MH algorithm.

Boostrap Particle Filter

A natural choice for particle filtering is the bootstrap particle filter (BPF) (Gordon et al., 1993), which is reviewed in Algorithm 4. The BPF is based on a sequential importance sampling procedure that iteratively approximates each filtering distribution $p(x_t \mid y_1, \ldots, y_t, \boldsymbol{\theta})$ with a set of S particles $\{x_t^1, \ldots, x_t^S\}$ so that

$$\hat{p}(\boldsymbol{y} \mid \boldsymbol{\theta}) = \prod_{t=1}^{T} \left(\frac{1}{S} \sum_{s=1}^{S} g(x_t^s, y_t; \boldsymbol{\theta}) \right) \tag{6}$$

is an unbiased estimate of the parameter likelihood $p(\boldsymbol{y}|\boldsymbol{\theta})$ (Doucet et al., 2001). Algorithm 4 provides a review of this algorithm.

Algorithm 4 `BootstrapParticleFilter`($\boldsymbol{y}$, $\boldsymbol{\theta}$, f, g, h)

for $s = 1, \ldots, S$ **do**
  Sample $x_1^s \sim h(x_1; \boldsymbol{\theta})$ and weight $w_1^s = g(x_1^s, y_1; \boldsymbol{\theta})$.
end for
Normalize $\{w_1^s\}_{s=1}^S = \{w_1^s\}_{s=1}^S / \sum_{s=1}^S w_1^s$.
for $t = 2, \ldots, T$ **do**
  for $s = 1, \ldots, S$ **do**
    Resample ancestor index $a_{t-1}^s \sim \text{Categorical}(w_{t-1}^1, \ldots, w_{t-1}^S)$.
    Sample $x_t^s \sim f(x_{t-1}^{a_{t-1}^s}, x_t; \boldsymbol{\theta})$ and weight $w_t^s = g(x_t^s, y_t; \boldsymbol{\theta})$.
  end for
  Normalize $\{w_t^s\}_{s=1}^S = \{w_t^s\}_{s=1}^S / \sum_{s=1}^S w_t^s$.
end for
return Particles $\{\{x_1^s\}_{s=1}^S, \ldots, \{x_T^s\}_{s=1}^S\}$

A common problem with the BPF is that although its estimate of $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ is unbiased, this approximation may have high variance for certain observation vectors $\boldsymbol{y}$ and certain parameters $\boldsymbol{\theta}$. The variance can be reduced at the price of increasing the number of particles, yet this often significantly increases computation time and is therefore unsatisfactory. A recently developed alternative to the BPF that attempts to remedy this problem is controlled sequential Monte Carlo.

Twisted Sequential Monte Carlo

Controlled sequential Monte Carlo (cSMC) (Heng et al., 2017) is based on the premise that we can modify a state-space model in such a way that standard bootstrap particle filters give lower variance estimates while the likelihood of interest is kept unchanged.

The basic idea of cSMC is to run several iterations of *twisted sequential Monte Carlo* (twisted SMC), a process in which we redefine the model's state transition density f, initial distribution h, and state-dependent likelihood g in a way that allows the BPF to produce lower-variance estimates without changing the parameter likelihood $p(\boldsymbol{y} \mid \boldsymbol{\theta})$. Guarniero et al. (2017) provide a different iterative approach.

Using a *policy* $\gamma = \{\gamma_1, \ldots, \gamma_T\}$ in which each γ_t is a positive and bounded function, we define,

$$\begin{aligned} h^\gamma(x_1; \boldsymbol{\theta}) &= \frac{h(x_1; \boldsymbol{\theta}) \cdot \gamma_1(x_1)}{H^\gamma(\boldsymbol{\theta})}, \\ f_t^\gamma(x_{t-1}, x_t; \boldsymbol{\theta}) &= \frac{f(x_{t-1}, x_t; \boldsymbol{\theta}) \cdot \gamma_t(x_t)}{F_t^\gamma(x_{t-1}; \boldsymbol{\theta})}, \qquad 1 < t \leq T, \end{aligned} \tag{7}$$

where $H^\gamma(\boldsymbol{\theta}) = \int h(x_1; \boldsymbol{\theta}) \gamma_1(x_1) dx_1$ and $F_t^\gamma(x_{t-1}; \boldsymbol{\theta}) = \int f(x_{t-1}, x_t; \boldsymbol{\theta}) \gamma_t(x_t) dx_t$ are normalization terms for the probability densities h^γ and f_t^γ, respectively. To ensure that the parameter likelihood estimate $\hat{p}(\boldsymbol{y} \mid \boldsymbol{\theta})$ remains unbiased under the twisted model, we define the twisted state-dependent likelihoods $g_1^\gamma, \ldots, g_T^\gamma$ as functions that satisfy:

$$\begin{aligned} & p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{\theta}) = h^\gamma(x_1; \boldsymbol{\theta}) \cdot \prod_{t=2}^{T} f_t^\gamma(x_{t-1}, x_t; \boldsymbol{\theta}) \cdot \prod_{t=1}^{T} g_t^\gamma(x_t, y_t; \boldsymbol{\theta}) \\ \iff & h(x_1; \boldsymbol{\theta}) \cdot \prod_{t=2}^{T} f(x_{t-1}, x_t; \boldsymbol{\theta}) \cdot \prod_{t=1}^{T} g(x_t, y_t; \boldsymbol{\theta}) \\ & = \frac{h(x_1; \boldsymbol{\theta}) \gamma_1(x_1)}{H^\gamma(\boldsymbol{\theta})} \cdot \prod_{t=2}^{T} \frac{f(x_{t-1}, x_t; \boldsymbol{\theta}) \gamma_t(x_t; \boldsymbol{\theta})}{F_t^\gamma(x_{t-1}; \boldsymbol{\theta})} \cdot \prod_{t=1}^{T} g_t^\gamma(x_t, y_t; \boldsymbol{\theta}) \\ \iff & \prod_{t=1}^{T} g(x_t, y_t; \boldsymbol{\theta}) = \frac{\gamma_1(x_1)}{H^\gamma(\boldsymbol{\theta})} \cdot \prod_{t=2}^{T} \frac{\gamma_t(x_t)}{F_t^\gamma(x_{t-1}; \boldsymbol{\theta})} \cdot \prod_{t=1}^{T} g_t^\gamma(x_t, y_t; \boldsymbol{\theta}). \end{aligned} \tag{8}$$

This equality can be maintained if we define $g_1^\gamma, \ldots, g_T^\gamma$ as follows:

$$g_1^\gamma(x_1, y_1; \boldsymbol{\theta}) = \frac{H^\gamma(\boldsymbol{\theta}) \cdot g(x_1, y_1; \boldsymbol{\theta}) \cdot F_2^\gamma(x_1; \boldsymbol{\theta})}{\gamma_1(x_1)}, \tag{9}$$

$$g_t^\gamma(x_t, y_t; \boldsymbol{\theta}) = \frac{g(x_t, y_t; \boldsymbol{\theta}) \cdot F_{t+1}^\gamma(x_t; \boldsymbol{\theta})}{\gamma_t(x_t)}, \qquad 1 < t < T,$$
$$g_T^\gamma(x_T, y_T; \boldsymbol{\theta}) = \frac{g(x_T, y_T; \boldsymbol{\theta})}{\gamma_T(x_T)}.$$

Thus, the parameter likelihood estimate of the twisted model is

$$\hat{p}^\gamma(\boldsymbol{y} \mid \boldsymbol{\theta}) = \prod_{t=1}^{T} \left(\frac{1}{S} \sum_{s=1}^{S} g_t^\gamma(x_t^s, y_t; \boldsymbol{\theta}) \right). \tag{10}$$

The optimal policy γ^* that minimizes the variance of this estimate may be intractable to compute in general. Thus, we typically choose γ as the policy that is closest to γ^* within a tractable set. The strategy for finding γ depends on the specific nonlinearities f, g, h. The BPF is simply a degenerate case of twisted SMC in which $\gamma_t = 1$ for all t.

Algorithm 5 `ControlledSMC`($\boldsymbol{y}$, f, g, h, $\boldsymbol{\theta}$, L)

Collect particles $\{x_1^s\}_{s=1}^S, \ldots, \{x_T^s\}_{s=1}^S =$ `BootstrapParticleFilter`($\boldsymbol{y}$, $\boldsymbol{\theta}$, f, g, h).
Initialize $\Gamma' = \{\Gamma'_1, \ldots, \Gamma'_T\}$ where $\Gamma'_t(x_t) = 1$ for all $t = 1, \ldots, T$.
for $\ell = 1, \ldots, L$ **do**
    Solve for cumulative policy $\Gamma =$ `FindPolicy`(Γ', f, g, h, $\{x_1^s\}_{s=1}^S, \ldots, \{x_T^s\}_{s=1}^S$, $\boldsymbol{y}$).
    // Bootstrap particle filter to sample particles.
    for $s = 1, \ldots, S$ **do**
        Sample $x_1^s \sim h^\Gamma(x_1)$ and weight $w_1^s = g_1^\Gamma(x_1^s, y_1)$.
    end for
    Normalize $\{w_1^s\}_{s=1}^S = \{w_1^s\}_{s=1}^S / \sum_{s=1}^S w_1^s$.
    for $t = 2, \ldots, T$ **do**
        for $s = 1, \ldots, S$ **do**
            Resample ancestor index $a_{t-1}^s \sim \text{Categorical}(w_{t-1}^1, \ldots, w_{t-1}^S)$.
            Sample $x_t^s \sim f_t^\Gamma(x_{t-1}^{a_{t-1}^s}, x_t; \boldsymbol{\theta})$ and weight $w_t^s = g_t^\Gamma(x_t^s, y_t)$.
        end for
        Normalize $\{w_t^s\}_{s=1}^S = \{w_t^s\}_{s=1}^S / \sum_{s=1}^S w_t^s$.
    end for
    Update $\Gamma' = \Gamma$.
end for
return Likelihood estimate $\hat{p}^\Gamma(\boldsymbol{y} \mid \boldsymbol{\theta})$ (Equation (10)).

Controlled Sequential Monte Carlo

The full cSMC algorithm is given in Algorithm 5. It iterates on twisted SMC for L iterations, building a series of policies $\gamma^{\langle 1\rangle}, \gamma^{\langle 2\rangle}, \ldots, \gamma^{\langle L\rangle}$ over time. Given two policies Γ' and γ, we can define,

$$h^{\Gamma'\cdot\gamma}(x_1) \propto h^{\Gamma'}(x_1)\gamma_1(x_1) = h(x_1;\boldsymbol{\theta}) \cdot \Gamma'_1(x_1) \cdot \gamma_1(x_1), \tag{11}$$
$$f_t^{\Gamma'\cdot\gamma}(x_{t-1}, x_t;\boldsymbol{\theta}) \propto f_t^{\Gamma'}(x_{t-1}, x_t;\boldsymbol{\theta}) \cdot \gamma_t(x_t) = f(x_{t-1}, x_t;\boldsymbol{\theta}) \cdot \Gamma'_t(x_t) \cdot \gamma_t(x_t), \quad 1 < t \leq T.$$

We can see from these relationships that twisting the original model using Γ' and then twisting the new model using γ has the same effect as twisting the original model using a cumulative policy Γ where each $\Gamma_t(x_t) = \Gamma'_t(x_t) \cdot \gamma_t(x_t)$.

In Algorithm 5, the `FindPolicy` function of Line 4 will depend on the specific type of nonlinear state-space model. In Section 4.3, we present an approach that works well for *dynamic generalized linear models*, a class of nonlinear state-space models with Gaussian latent states that are common in neuroscience applications.

2.5.3 Point Process State-Space Model

The point process state-space model is a particular nonlinear state-space model that is commonly used in neuroscience literature to describe neuronal firing data (Smith and Brown, 2003). We briefly review this model here.

Consider an experiment with R successive trials, during which we record the activity of a neuronal spiking unit. For each trial, let $(0, \mathcal{T}]$ be the continuous observation interval following the delivery of an exogenous stimulus at time $\tau = 0$. In addition, for each trial $r = 1, \ldots, R$, let S_r be the total number of spikes from the neuron during trial r, and the sequence $0 < \tau_{r,1} < \ldots < \tau_{r,S_r}$ corresponds to the times at which events from the neuronal unit occur. We assume that $\{\tau_{r,s}\}_{s=1}^{S_r}$ is the realization in $(0, \mathcal{T}]$ of a stochastic point-process with counting process $N_r(\tau) = \int_0^\tau dN_r(u)$, where $dN_r(\tau)$ is the indicator function in $(0, \mathcal{T}]$ of $\{\tau_{r,s}\}_{s=1}^{S_r}$. A point-process is fully characterized by its conditional intensity function (CIF) (Vere-Jones, 2003). Assuming all trials are independent and identically distributed realizations of the same point-process, the CIF $\lambda(\tau \mid H_\tau)$ of $dN_r(\tau)$ for $r = 1, \ldots, R$ is

$$\lambda(\tau \mid H_\tau) = \lim_{\delta \to 0} \frac{p\left(N_r(\tau + \delta) - N_r(\tau) = 1 \mid H_\tau\right)}{\delta}, \tag{12}$$

where H_τ is the event history of the point process up to time τ. Suppose we sample $dN_r(\tau)$ at a resolution of δ to yield a binary event sequence. We denote by $\{\Delta N_{t,r}\}_{t=1,r=1}^{T,R}$ the discrete-time process obtained by counting the number of events in $T = \lfloor \mathcal{T}/\Delta \rfloor$ disjoint bins of width $\Delta = M \cdot \delta$, where $M \in \mathbb{N}$. Given the initial state x_0, a popular approach is to encode a discrete-time representation of the CIF $\{\lambda_t\}_{t=1}^{T}$ within an autoregressive process that underlies a binomial state-space model with observations $\{\Delta N_{t,r}\}_{t=1,r=1}^{T,R}$ (Smith and Brown, 2003):

$$
\begin{aligned}
x_1 &\sim \mathcal{N}(x_0 + \mu_1, \psi_0), && \\
x_t \mid x_{t-1} &\sim \mathcal{N}(x_{t-1} + \mu_t, \psi), && 1 < t \leq T, \\
p_t = \lambda_t \cdot \delta = \sigma(x_t) &= \frac{\exp x_t}{1 + \exp x_t}, && 1 \leq t \leq T, \\
y_t = \sum_{r=1}^{R} \Delta N_{t,r} &\sim \text{Binomial}\left(R \cdot M, p_t\right), && 1 \leq t \leq T,
\end{aligned}
\tag{13}
$$

where $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_T\}$ is a vector of inputs to the system that are usually used to signify changes in the firing rate of the neuron at various time points, and ψ and ψ_0 are variance parameters that impose stochastic smoothness on the latent process $\boldsymbol{x}$.

Equation (13) is a nonlinear state-space model because the state transition density f and initial distribution h are Gaussian, but the state-dependent likelihood g is binomial. In the subsequent chapters, we will revisit this point process state-space model several times in applications of time series clustering to neuroscience.

3 State-Space Finite Mixture Model and MCEM Algorithm

The natural generative model for time series clustering problems is a mixture of state-space models, which we introduce here. This mixture can either be finite or infinite, depending on whether or not the true number of clusters K is known. This chapter focuses on the finite case. We present a Monte Carlo expectation-maximization algorithm for MAP inference and show its utility on a time series clustering application in neuroscience.

3.1 Finite Mixture of State-Space Models

Given N times series $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$ of length T, the model has the following form:

$$
\begin{aligned}
\boldsymbol{q} &\sim \text{Dirichlet}(\boldsymbol{\alpha}), && && (14)\\
z^{(n)} &\sim \text{Categorical}(\boldsymbol{q}), && 1 \le n \le N,\\
\boldsymbol{\theta}^{(k)} &\sim G, && 1 \le k \le K,\\
x_1^{(n)} \mid z^{(n)} = k, \boldsymbol{\theta}^{(k)} &\sim h(x_1^{(n)}; \boldsymbol{\theta}^{(k)}), && 1 \le n \le N,\\
x_t^{(n)} \mid x_{t-1}^{(n)}, z^{(n)} = k, \boldsymbol{\theta}^{(k)} &\sim f(x_{t-1}^{(n)}, x_t^{(n)}; \boldsymbol{\theta}^{(k)}), && 1 < t \le T, 1 \le n \le N,\\
y_t^{(n)} \mid x_t^{(n)}, z^{(n)} = k, \boldsymbol{\theta}^{(k)} &\sim g(x_t^{(n)}, y_t^{(n)}; \boldsymbol{\theta}^{(k)}), && 1 \le t \le T, 1 \le n \le N.
\end{aligned}
$$

Figure 5 depicts the graphical model representation.

3.2 Monte Carlo Expectation-Maximization Inference Algorithm

We denote the set of all cluster assignments $Z = \{z^{(1)}, \ldots, z^{(N)}\}$, the set of all hidden states $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$, and the set of all cluster parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(K)}\}$.

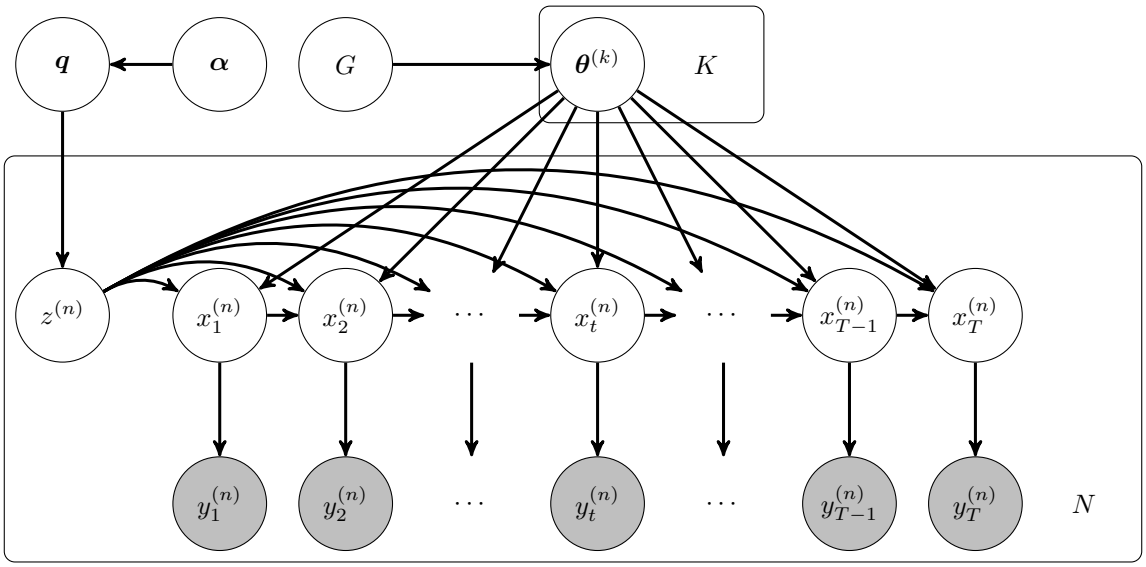

Figure 5: The finite state-space mixture model for time series clustering. For simplicity, we omit dependencies that are assumed between $\boldsymbol{Y}$ and $(Z, \boldsymbol{\Theta})$.

The goal is to recover a clustering of the time series data $\boldsymbol{Y}$ by performing MAP inference on $\{\boldsymbol{\Theta}, \boldsymbol{q}\}$, yielding estimates $\{\boldsymbol{\Theta}^*, \boldsymbol{q}^*\}$ which can then be used to find a distribution over cluster assignments $p(Z \mid \boldsymbol{Y}, \boldsymbol{\Theta}^*, \boldsymbol{q}^*)$.

3.2.1 Expectation Step (E-Step)

From the previous (i.e. i-th) M-Step, assume we have the parameter estimates $\boldsymbol{\Theta}^{\langle i \rangle}$ and $\boldsymbol{q}^{\langle i \rangle}$. We begin by defining the *expected complete-data log-likelihood* (ECDLL):

$$
\begin{aligned}
\ell(\boldsymbol{\Theta}, \boldsymbol{q}) &= \mathbb{E}_{Z, \boldsymbol{X} \mid \boldsymbol{Y}, \boldsymbol{\Theta}^{\langle i \rangle}, \boldsymbol{q}^{\langle i \rangle}} \left[\log p(Z, \boldsymbol{X}, \boldsymbol{Y} \mid \boldsymbol{\Theta}, \boldsymbol{q}) \right] \\
&= \mathbb{E}_{Z \mid \boldsymbol{Y}, \boldsymbol{\Theta}^{\langle i \rangle}, \boldsymbol{q}^{\langle i \rangle}} \left[\mathbb{E}_{\boldsymbol{X} \mid Z, \boldsymbol{Y}, \boldsymbol{\Theta}^{\langle i \rangle}, \boldsymbol{q}^{\langle i \rangle}} [\log p(Z \mid \boldsymbol{\Theta}, \boldsymbol{q}) + \log p(\boldsymbol{X}, \boldsymbol{Y} \mid Z, \boldsymbol{\Theta}, \boldsymbol{q})] \right] \\
&= \mathbb{E}_{Z \mid \boldsymbol{Y}, \boldsymbol{\Theta}^{\langle i \rangle}, \boldsymbol{q}^{\langle i \rangle}} \left[\mathbb{E}_{\boldsymbol{X} \mid Z, \boldsymbol{Y}, \boldsymbol{\Theta}^{\langle i \rangle}, \boldsymbol{q}^{\langle i \rangle}} [\log p(Z \mid \boldsymbol{q}) + \log p(\boldsymbol{X}, \boldsymbol{Y} \mid Z, \boldsymbol{\Theta})] \right] \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} p(z^{(n)} = k \mid \boldsymbol{y}^{(n)}, \boldsymbol{\Theta}^{\langle i \rangle}, \boldsymbol{q}^{\langle i \rangle}) \cdot \left(\log q^{(k)} + V^{(n,k)}(\boldsymbol{\Theta}) \right) \\
&\propto \sum_{n=1}^{N} \sum_{k=1}^{K} p(z^{(n)} = k \mid \boldsymbol{\Theta}^{\langle i \rangle}, \boldsymbol{q}^{\langle i \rangle}) \cdot p(\boldsymbol{y}^{(n)} \mid z^{(n)} = k, \boldsymbol{\Theta}^{\langle i \rangle}, \boldsymbol{q}^{\langle i \rangle}) \cdot \left(\log q^{(k)} + V^{(n,k)}(\boldsymbol{\Theta}) \right) \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} {q^{(k)}}^{\langle i \rangle} \cdot p\left(\boldsymbol{y}^{(n)} \mid {\boldsymbol{\theta}^{(k)}}^{\langle i \rangle} \right) \cdot \left(\log q^{(k)} + V^{(n,k)}(\boldsymbol{\Theta}) \right)
\end{aligned}
\tag{15}
$$

where the quantity $V^{(n,k)}(\boldsymbol{\Theta})$ is further expanded as:

$$
\begin{aligned}
V^{(n,k)}(\boldsymbol{\Theta}) &= \mathbb{E}_{\boldsymbol{x}^{(n)} \mid z^{(n)}=k, \boldsymbol{y}^{(n)}, \boldsymbol{\Theta}^{\langle i \rangle}, \boldsymbol{q}^{\langle i \rangle}} [\log p(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)} \mid z^{(n)} = k, \boldsymbol{\Theta})] \\
&= \mathbb{E}_{\boldsymbol{x}^{(n)} \mid z^{(n)}=k, \boldsymbol{y}^{(n)}, \boldsymbol{\Theta}^{\langle i \rangle}, \boldsymbol{q}^{\langle i \rangle}} [\log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{\theta}^{(k)}) + \log p(\boldsymbol{y}^{(n)} \mid \boldsymbol{x}^{(n)}, \boldsymbol{\theta}^{(k)})] \\
&= \mathbb{E}_{\boldsymbol{x}^{(n)} \mid \boldsymbol{y}^{(n)}, \boldsymbol{\theta}^{(k)\langle i \rangle}} [\log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{\theta}^{(k)}) + \log p(\boldsymbol{y}^{(n)} \mid \boldsymbol{x}^{(n)}, \boldsymbol{\theta}^{(k)})] \\
&= \mathbb{E}_{\boldsymbol{x}^{(n)} \mid \boldsymbol{y}^{(n)}, \boldsymbol{\theta}^{(k)\langle i \rangle}} \\
&\qquad \left[\log h(x_1^{(n)}; \boldsymbol{\theta}^{(k)}) + \sum_{t=2}^{T} \log f(x_{t-1}^{(n)}, x_t^{(n)}; \boldsymbol{\theta}^{(k)}) + \sum_{t=1}^{T} \log g(x_t^{(n)}, y_t^{(n)}; \boldsymbol{\theta}^{(k)}) \right].
\end{aligned} \tag{16}
$$

From Equation (15) and Equation (16), we can see that computing the ECDLL requires computing the parameter log-likelihood $p(\boldsymbol{y}^{(n)} \mid \boldsymbol{\theta}^{(k)\langle i \rangle})$ as well as expectations of the nonlinear functions f, g, h under the smoothing distribution $\boldsymbol{x}^{(n)} \mid \boldsymbol{y}^{(n)}, \boldsymbol{\theta}^{(k)\langle i \rangle}$. The first of these tasks can be done using a standard particle filtering method, such as the bootstrap particle filter or controlled sequential Monte Carlo (Heng et al., 2017). The latter task may be difficult to do in general; in Section 3.3.2, we give an application with specific nonlinearities in which this expectation is possible to approximate.

A potential further complication is the introduction of priors on $\boldsymbol{\Theta}$ and $\boldsymbol{q}$ in the Bayesian setting. Under such a scenario, we are instead interested in the *expected complete-data log-posterior* (ECDLP), which is:

$$
\begin{aligned}
R(\boldsymbol{\Theta}, \boldsymbol{q}) &\simeq \mathbb{E}_{Z, \boldsymbol{X} \mid \boldsymbol{Y}, \boldsymbol{\Theta}^{\langle i \rangle}, \boldsymbol{q}^{\langle i \rangle}} [\log p(\boldsymbol{\Theta}, \boldsymbol{q}) + \log p(Z, X, Y \mid \boldsymbol{\Theta}, \boldsymbol{q})] \\
&= \log p(\boldsymbol{\Theta}, \boldsymbol{q}) + \ell(\boldsymbol{\Theta}, \boldsymbol{q}),
\end{aligned} \tag{17}
$$

where $\simeq$ denotes equality up to an additive constant.

3.2.2 Maximization Step (M-Step)

After establishing the ECDLP in Equation (17), we wish to compute new values $\boldsymbol{\Theta}^{\langle i+1 \rangle}, \boldsymbol{q}^{\langle i+1 \rangle}$ that increase the log-likelihood of the data $\boldsymbol{Y}$. We can arrive at maximum ECDLL estimates by setting $\partial R / \partial \boldsymbol{\Theta} = \boldsymbol{0}$ and $\partial R / \partial \boldsymbol{q} = \boldsymbol{0}$, while obeying the constraint that $\sum_{k=1}^{K} q^{(k)} = 1$.

For $\boldsymbol{q}$, since conjugacy exists between the Dirichlet prior and the Categorical likelihood,

the update can be analytically derived as:

$$q^{(k)\langle i+1\rangle} = \frac{\alpha^{(k)} - 1 + \sum_{n=1}^{N} w^{(n,k)}}{\sum_{k'=1}^{K} \alpha^{(k')} - K + N}, \qquad 1 \leq k \leq K, \tag{18}$$

where

$$w^{(n,k)} = p(z^{(n)} = k \mid \boldsymbol{y}^{(n)}, \boldsymbol{q}^{\langle i\rangle}, \boldsymbol{\Theta}^{\langle i\rangle}) \propto q^{(k)\langle i\rangle} \cdot p\left(\boldsymbol{y}^{(n)} \mid \boldsymbol{\theta}^{(k)\langle i\rangle}\right). \tag{19}$$

The parameter likelihood $p(\boldsymbol{y}^{(n)} \mid \boldsymbol{\theta}^{(k)\langle i\rangle})$ can be exactly computed for linear-Gaussian state-space models with a Kalman filter, but must be approximated for the general nonlinear case using particle filtering methods.

For $\boldsymbol{\Theta}$, a closed form solution for $\boldsymbol{\Theta}^{\langle i+1\rangle} = \arg\max_{\boldsymbol{\Theta}} R(\boldsymbol{\Theta}, \cdot)$ may or may not be derivable, depending on f, g, h and G. Section 3.3.2 gives an example of when derivation is possible using a Monte Carlo approximation for certain quantities.

3.2.3 Computing Cluster Distributions

Alternately repeating the E-Step and M-Step yields a sequence of parameter estimates $\{\boldsymbol{\Theta}^{(1)}, \boldsymbol{q}^{(1)}\}, \{\boldsymbol{\Theta}^{(2)}, \boldsymbol{q}^{(2)}\}, \ldots$, until some convergence criterion returns a final estimate $\{\boldsymbol{\Theta}^*, \boldsymbol{q}^*\}$. To find a posterior distribution over possible clusterings Z of the time series $\boldsymbol{Y}$, we calculate:

$$\begin{aligned} \log p(Z \mid \boldsymbol{Y}, \boldsymbol{\Theta}^*, \boldsymbol{q}^*) &\simeq \log p(Z \mid \boldsymbol{\Theta}^*, \boldsymbol{q}^*) + \log p(\boldsymbol{Y} \mid Z, \boldsymbol{\Theta}^*, \boldsymbol{q}^*) \\ &= \sum_{n=1}^{N} \log p(z^{(n)} \mid \boldsymbol{\Theta}^*, \boldsymbol{q}^*) + \sum_{n=1}^{N} \log p(\boldsymbol{y}^{(n)} \mid Z, \boldsymbol{\Theta}^*, \boldsymbol{q}^*) \\ &= \sum_{k=1}^{K} N^{(k)} \cdot \log q^{(k)*} + \sum_{n=1}^{N} \log p\left(\boldsymbol{y}^{(n)} \mid \boldsymbol{\theta}^{(k)*}\right), \end{aligned} \tag{20}$$

where $N^{(k)}$ is the number of $z^{(n)}$ equal to k for $k = 1, \ldots, K$. In the nonlinear state-space case, we need a particle filter to compute the cluster parameter likelihoods $p\left(\boldsymbol{y}^{(n)} \mid \boldsymbol{\theta}^{(k)*}\right)$.

3.3 Applications

We present two applications of using the EM algorithm to cluster time series. The first concerns epileptic seizure recognition. The time series in this example are modeled using

linear-Gaussian state-space models, which allows us to exactly compute the ECDLL in Equation (15) (and by extension, the ECDLP in Equation (17)) without Monte Carlo methods. The relatively successful clustering in this setting serves as a demonstration of the abilities of the EM algorithm in the absence of nonlinearities.

The second application concerns neuronal firing rates. In this example, we use a mixture of nonlinear state-space models, which requires us to employ particle-based methods for tractable inference. Thus, the EM Algorithm becomes a MCEM algorithm in this setting. We show that for this specific problem, the algorithm is able to identify a sensible clustering of the time series in question.

3.3.1 Epilepsy Recognition

Epilepsy is a chronic disorder that affects 50 million people every year. It is characterized by subjects undergoing a central nervous system breakdown, also known as a seizure. In a recent report by the World Health Organization, it is estimated that 70% of people with epilepsy could live seizure-free if they were properly diagnosed and treated. A common way of recognizing these seizures is by using electroencephalography (EEG). This is a way to monitor electrical activity in the brain and record any unusual behavior. EEG usually comes as data in the form time series, with points given at the centi-second or milli-second resolution. There is motivation to cluster these time series according to whether or not a particular patient is undergoing a seizure. These clusters can then be used in downstream tasks, such as determining whether a future patient is undergoing a seizure based on which cluster their EEG data falls into.

The specific dataset we use is collected by Andrzejak et al. (2001). It contains $N = 115,000$ series of length $T = 178$ from a total of 500 individuals. Each series $\boldsymbol{y}^{(n)} \in \boldsymbol{Y}$ is a sequence of EEG recordings keeping track of electrical activity in the brain for a total of one second. The total number of true clusters is $K = 2$. Of the 115,000 series, a total of 23,000 (or 20%) come from patients who experience epileptic seizures ($k = 1$). The rest come from patients who do not ($k = 2$).

The dataset comes with cluster labels, indicating which time series correspond to epileptic seizures. Our objective is to separate the data without explicitly building these labels into the inference procedure. Afterwards, we can use the labels to benchmark the performance of the algorithm. Figure 6 presents examples of EEG recordings from an epileptic brain and a normal brain.

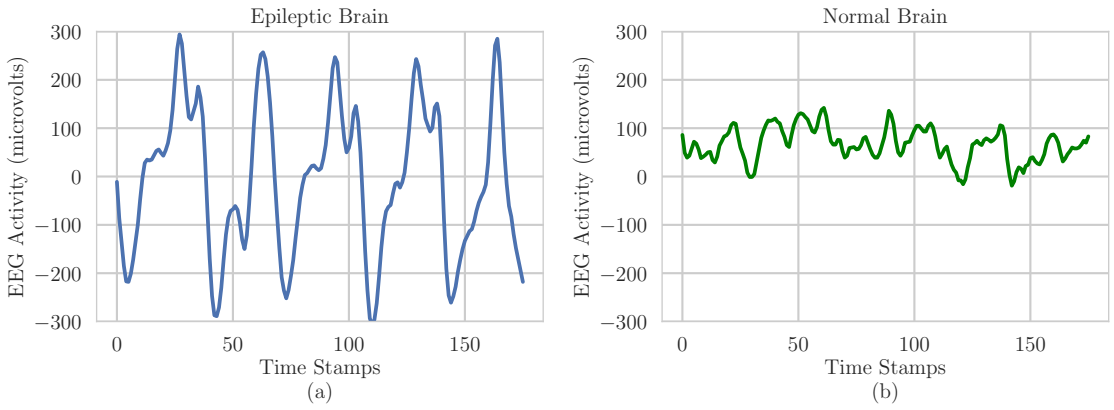

Figure 6: Sample EEG recordings from (a) an epileptic brain and (b) a normal brain.

Modeling Details

Each data point $y_t^{(n)} \in \mathbb{R}$, so it is possible to adopt the linear-Gaussian state-space (LGSSM) framework. In addition, there are multiple time series belonging to two different clusters, which prompts us to employ a finite mixture. In the form of Equation (14), we generatively model the time series as:

$$\begin{aligned}
\boldsymbol{q} &\sim \text{Dirichlet}(\boldsymbol{\alpha}), && \\
z^{(n)} &\sim \text{Categorical}(\boldsymbol{q}), && 1 \le n \le N, \\
\boldsymbol{\theta}^{(k)} = \psi^{(k)} &\sim G, && 1 \le k \le K, \\
x_1^{(n)} &\sim \mathcal{N}(x_0{}^{(n)}, \psi_0), && 1 \le n \le N, \\
x_t^{(n)} \mid x_{t-1}^{(n)}, z^{(n)} = k, \psi^{(k)} &\sim \mathcal{N}(x_{t-1}^{(n)}, \psi^{(k)}), && 1 < t \le T, 1 \le n \le N, \\
y_t^{(n)} \mid x_t^{(n)} &\sim \mathcal{N}(x_t^{(n)}, \sigma^2), && 1 \le t \le T, 1 \le n \le N.
\end{aligned} \tag{21}$$

where $x_0^{(n)}, \psi_0, \sigma^2$ are all supplied constants. Let the base distribution G be inverse Gamma with hyperparameters a, b. The one cluster parameter in this case is the variance $\psi^{(k)}$ in the state transition density. This was chosen with the knowledge that in general, patients experiencing epileptic seizures exhibit EEG recordings with relatively greater degrees of oscillation than those who are not experiencing seizures.

EM Algorithm

Since Equation (21) is a mixture of linear-Gaussian state-space models, we can exactly compute the quantities in Equation (15) of the E-Step. For example, a *Kalman filter* gives

the exact parameter log-likelihood $p(\boldsymbol{y}^{(n)} \mid \boldsymbol{\theta}^{(k)\langle i \rangle})$. In addition, observe that the initial distribution h and the state-dependent likelihood g do not depend on the cluster parameter $\boldsymbol{\theta}^{(k)} = \psi^{(k)}$. Thus, the other major quantity from Equation (15) and Equation (16), $V^{(n,k)}(\boldsymbol{\Theta})$, can be expanded as:

$$
\begin{aligned}
V^{(n,k)}(\boldsymbol{\Theta}) &= \mathbb{E}_{\boldsymbol{x}^{(n)} \mid \boldsymbol{y}^{(n)}, \boldsymbol{\theta}^{(k)\langle i \rangle}} \\
&\qquad \left[\log h(x_1^{(n)}; \boldsymbol{\theta}^{(k)}) + \sum_{t=2}^{T} \log f(x_{t-1}^{(n)}, x_t^{(n)}; \boldsymbol{\theta}^{(k)}) + \sum_{t=1}^{T} \log g(x_t^{(n)}, y_t^{(n)}; \boldsymbol{\theta}^{(k)}) \right] \\
&\simeq \mathbb{E}_{\boldsymbol{x}^{(n)} \mid \boldsymbol{y}^{(n)}, \boldsymbol{\theta}^{(k)\langle i \rangle}} \left[\sum_{t=2}^{T} \log f(x_{t-1}^{(n)}, x_t^{(n)}; \boldsymbol{\theta}^{(k)}) \right] \\
&\simeq \mathbb{E}_{\boldsymbol{x}^{(n)} \mid \boldsymbol{y}^{(n)}, \boldsymbol{\theta}^{(k)\langle i \rangle}} \left[\sum_{t=2}^{T} -\frac{1}{2} \log \psi^{(k)} - \frac{(x_t^{(n)} - x_{t-1}^{(n)})^2}{2\psi^{(k)}} \right] \\
&= -\frac{(T-1)}{2} \log \psi^{(k)} - \frac{1}{2\psi^{(k)}} \sum_{t=2}^{T} \mathbb{E}_{\boldsymbol{x}^{(n)} \mid \boldsymbol{y}^{(n)}, \boldsymbol{\theta}^{(k)\langle i \rangle}} \left[(x_t^{(n)} - x_{t-1}^{(n)})^2 \right] \\
&= -\frac{(T-1)}{2} \log \psi^{(k)} - \frac{1}{2\psi^{(k)}} \sum_{t=2}^{T} \mathbb{E} \left[(x_t^{(n)} - x_{t-1}^{(n)})^2 \mid \boldsymbol{y}^{(n)}, \psi^{(k)\langle i \rangle} \right].
\end{aligned}
\tag{22}
$$

The final quantity, $\sum_{t=2}^{T} \mathbb{E} \left[(x_t^{(n)} - x_{t-1}^{(n)})^2 \mid \boldsymbol{y}^{(n)}, \psi^{(k)\langle i \rangle} \right]$, is a statistic from the smoothing distribution. We can arrive at an exact calculation using a *Kalman smoother*, which is explained in the next section.

In the M-Step, we wish to find $\psi^{(k)\langle i+1 \rangle}$ which maximizes the ECDLP R, a function of $\psi^{(k)}$ in Equation (17). Setting $\partial R / \partial \psi^{(k)} = 0$, we have:

$$
\begin{aligned}
0 &= \frac{\partial}{\partial \psi^{(k)}} \left[\log p(\psi^{(k)}; a, b) + \mathbb{E}_{Z, \boldsymbol{X} \mid \boldsymbol{Y}, \boldsymbol{\Theta}^{\langle i \rangle}, \boldsymbol{q}^{\langle i \rangle}} [\log p(Z, X, Y \mid \boldsymbol{\Theta}, \boldsymbol{q})] \right] \\
0 &= -\frac{a+1}{\psi^{(k)}} + \frac{b}{(\psi^{(k)})^2} + \frac{\partial}{\partial \psi^{(k)}} \left[\sum_{n=1}^{N} \underbrace{p(z^{(n)} = k \mid \boldsymbol{y}^{(n)}, \boldsymbol{q}^{\langle i \rangle}, \boldsymbol{\Theta}^{\langle i \rangle})}_{w^{(n,k)}} \cdot V^{(n,k)}(\boldsymbol{\Theta}) \right] \\
0 &= -\frac{a+1}{\psi^{(k)}} + \frac{b}{(\psi^{(k)})^2} + \\
&\qquad \sum_{n=1}^{N} w^{(n,k)} \cdot \left[-\frac{(T-1)}{2\psi^{(k)}} + \frac{1}{2(\psi^{(k)})^2} \sum_{t=2}^{T} \mathbb{E} \left[(x_t^{(n)} - x_{t-1}^{(n)})^2 \mid \boldsymbol{y}^{(n)}, \psi^{(k)\langle i \rangle} \right] \right] \\
0 &= -\frac{1}{\psi^{(k)}} \cdot \left(a + 1 + \frac{T-1}{2} \sum_{n=1}^{N} w^{(n,k)} \right) +
\end{aligned}
\tag{23}
$$

$$\frac{1}{(\psi^{(k)})^2} \cdot \left(b + \frac{1}{2} \sum_{n=1}^{N} \sum_{t=2}^{T} w^{(n,k)} \cdot \mathbb{E}\left[(x_t^{(n)} - x_{t-1}^{(n)})^2 \mid \boldsymbol{y}^{(n)}, {\psi^{(k)}}^{\langle i \rangle} \right] \right).$$

Solving this final equation yields

$${\psi^{(k)}}^{\langle i+1 \rangle} = \frac{b + \frac{1}{2} \sum_{n=1}^{N} \sum_{t=2}^{T} w^{(n,k)} \cdot \mathbb{E}\left[(x_t^{(n)} - x_{t-1}^{(n)})^2 \mid \boldsymbol{y}^{(n)}, {\psi^{(k)}}^{\langle i \rangle} \right]}{a + 1 + \frac{T-1}{2} \sum_{n=1}^{N} w^{(n,k)}}. \tag{24}$$

Notice that this solution is the MAP of the ECDLP, an inverse Gamma distribution with hyperparameters a', b', where

$$\begin{aligned} a' &= a + \frac{T-1}{2} \sum_{n=1}^{N} w^{(n,k)}, \\ b' &= b + \frac{1}{2} \sum_{n=1}^{N} \sum_{t=2}^{T} w^{(n,k)} \cdot \mathbb{E}\left[(x_t^{(n)} - x_{t-1}^{(n)})^2 \mid \boldsymbol{y}^{(n)}, {\psi^{(k)}}^{\langle i \rangle} \right]. \end{aligned} \tag{25}$$

Equation (19) describes how to compute $w^{(n,k)}$ and the next section details how to compute $\mathbb{E}\left[(x_t^{(n)} - x_{t-1}^{(n)})^2 \mid \boldsymbol{y}^{(n)}, {\psi^{(k)}}^{\langle i \rangle} \right]$.

Algorithm 6 summarizes our EM algorithm for the epilepsy model of Equation (21).

Kalman Smoothing

As previously mentioned, for a given time series of observations $\boldsymbol{y}$ and supplied parameters $\{\psi, x_0, \psi_0, \sigma^2\}$, we can compute $\mathbb{E}\left[(x_t - x_{t-1})^2 \mid \boldsymbol{y} \right]$ for all $t = 2, \ldots, T$ by running a Kalman smoother (Haykin, 2004). Expanding this quantity further, we see that it is an expression involving five common statistics from the smoothing distribution $p(\boldsymbol{x} \mid \boldsymbol{y})$:

$$\begin{aligned} \mathbb{E}\left[(x_t - x_{t-1})^2 \mid \boldsymbol{y} \right] &= \mathbb{E}[(x_t)^2 \mid \boldsymbol{y}] + \mathbb{E}[(x_{t-1})^2 \mid \boldsymbol{y}] - 2\mathbb{E}[x_t \cdot x_{t-1} \mid \boldsymbol{y}] \\ &= \mathbb{V}\mathrm{ar}[x_t \mid \boldsymbol{y}] + \mathbb{V}\mathrm{ar}[x_{t-1} \mid \boldsymbol{y}] - 2\mathbb{C}\mathrm{ov}[x_t, x_{t-1} \mid \boldsymbol{y}] + \left(\mathbb{E}[x_t \mid \boldsymbol{y}] - \mathbb{E}[x_{t-1} \mid \boldsymbol{y}] \right)^2. \end{aligned} \tag{26}$$

Algorithm 7 summarizes the Kalman smoother, which is designed to calculate $\mu_{t \mid T} = \mathbb{E}[x_t \mid \boldsymbol{y}, \boldsymbol{\theta}]$ and $\nu_{t \mid T} = \mathbb{V}\mathrm{ar}[x_t \mid \boldsymbol{y}, \boldsymbol{\theta}]$ for all t. The final statistic, the covariance, can be computed as $\mathbb{C}\mathrm{ov}[x_t, x_{t-1} \mid \boldsymbol{y}, \boldsymbol{\theta}] = L_{t-1} \cdot \nu_{t \mid T}$, where L_t is the *Kalman gain* for all $t < T$.

Algorithm 6 `EpilepsyExpMax`$(\boldsymbol{Y}, K, a, b, \boldsymbol{\alpha}, \psi_0, \sigma^2, \{x_0^{(1)}, \ldots, x_0^{(N)}\}, \epsilon_0)$

Set $i = 0$.
Draw parameters $\psi^{(k)\langle 0\rangle} \sim \text{InvGamma}(a, b)$ for $k = 1, \ldots, K$ and $\boldsymbol{q}^{\langle 0\rangle} \sim \text{Dirichlet}(\boldsymbol{\alpha})$.
Set $\epsilon = \infty$.
while $\epsilon_0 < \epsilon$ **do**
  // Expectation Step
  for $n = 1, \ldots, N$ **do**
    for $k = 1, \ldots, K$ **do**
      Run `KalmanSmoother`$(\boldsymbol{y}, \psi^{(k)\langle i\rangle}, x_0^{(n)}, \psi_0, \sigma^2)$.
      Compute $\sum_{t=2}^{T} \mathbb{E}\left[(x_t^{(n)} - x_{t-1}^{(n)})^2 \mid \boldsymbol{y}^{(n)}, \psi^{(k)\langle i\rangle}\right]$ using Equation (26).
      Compute $w^{(n,k)} = p(z^{(n)} = k \mid \boldsymbol{y}^{(n)}, \boldsymbol{q}^{\langle i\rangle}, \boldsymbol{\Theta}^{\langle i\rangle}) \propto q^{(k)\langle i\rangle} \cdot p\left(\boldsymbol{y}^{(n)} \mid \psi^{(k)\langle i\rangle}\right)$.
    end for
  end for
  // Maximization Step
  for $k = 1, \ldots, K$ **do**
    Optimize and solve for $q^{(k)\langle i+1\rangle}$ using Equation (18).
    Optimize and solve for $\psi^{(k)\langle i+1\rangle}$ using Equation (24).
  end for
  Compute $\epsilon = \|\boldsymbol{\Theta}^{\langle i+1\rangle} - \boldsymbol{\Theta}^{\langle i\rangle}\|_2$.
  Set $i = i + 1$.
end while
return $\boldsymbol{q}^{\langle i\rangle}, \boldsymbol{\Theta}^{\langle i\rangle}$

Algorithm 7 `KalmanSmoother`$(\boldsymbol{y}, \psi, x_0, \psi_0, \sigma^2)$

Initialize $\mu_{0\,|\,0} = x_0$ and $\nu_{0\,|\,0} = \psi_0$.
// Forward Kalman Filter
for $t = 1, \ldots, T$ **do**
  Set $\mu_{t\,|\,t-1} = \mu_{t-1\,|\,t-1}$ and $\nu_{t\,|\,t-1} = \nu_{t-1\,|\,t-1} + \psi$
  Set $K_t = \nu_{t\,|\,t-1}/(\nu_{t\,|\,t-1} + \sigma^2)$
  Set $\mu_{t\,|\,t} = \mu_{t\,|\,t-1} + K_t \cdot (y_t^{(n)} - \mu_{t\,|\,t-1})$
  Set $\nu_{t\,|\,t} = K_t \cdot \sigma^2$
end for
// Backward Kalman Smoother
for $t = T - 1, \ldots, 1$ **do**
  Compute *Kalman gain* $L_t = \nu_{t\,|\,t}/\nu_{t+1\,|\,t}$
  Set $\mu_{t\,|\,T} = \mu_{t\,|\,t} + L_t \cdot (\mu_{t+1\,|\,T} - \mu_{t+1\,|\,t})$
  Set $\nu_{t\,|\,T} = \nu_{t\,|\,t} + L_t^2 \cdot (\nu_{t+1\,|\,T} - \nu_{t+1\,|\,t})$
end for
return $(\mu_{1\,|\,T}, \nu_{1\,|\,T}, L_1), \ldots, (\mu_{T\,|\,T}, \nu_{T\,|\,T}, L_T)$

Table 1: Confusion matrix for a representative run of Algorithm 6 with $K = 2$, along with calculated true positive rate (TPR) and true negative rate (TNR).

	True Seizure	True Non-Seizure
Clustered Seizure	2160	580
Clustered Non-Seizure	140	8620
	TPR: 0.939	TNR: 0.937

Results

We feed at $N = 115,000$ time series into Algorithm 6 with $K = 2$, $a = 1$, $b = 1$, and $\boldsymbol{\alpha} = [1, 1]$. We let $\psi_0 = 1$ and $\sigma^2 = 1$. For each time series n, the initial state is estimated as the average of the first five data points, $x_0^{(n)} = 1/5 \cdot \sum_{t=1}^{5} y_t^{(n)}$. Finally, we use a convergence threshold of $\epsilon_0 = 10^{-5}$. After convergence, cluster parameters Θ^* and $\boldsymbol{q}^*$ are returned by the algorithm. We use Equation (20) to compute posterior cluster distributions for all the time series $p(Z \mid \boldsymbol{Y}, \boldsymbol{\Theta}^*, \boldsymbol{q}^*)$. To determine a final clustering, we assign each time series n to the cluster k such that $p(z^{(n)} = k \mid \boldsymbol{y}^{(n)}, \boldsymbol{\Theta}^*, \boldsymbol{q}^*)$ is maximized. We compare this final clustering with the ground-truth labels of seizure vs. no seizure.

It is well-documented that the EM algorithm tends to get stuck in local minima. Thus, we run Algorithm 6 a total of 20 times for different random initializations. Over the 20 times, the algorithm reports an average accuracy of 93.87%. In Table 1, we report the confusion matrix of a representative run. We see that the true positive rate (TPR) and true negative rate (TNR) are roughly equal, meaning neither cluster dominates in terms of accuracy. On average, the two final cluster parameter estimates are ${\psi^{(1)}}^* = 13,785.23$ for seizure-related time series and ${\psi^{(2)}}^* = 260.12$ for non-seizure related time series. This is consistent with the initial belief that seizure-related time series are more volatile. Interestingly, in all runs of the algorithm, over 99% of the time series have a cluster distribution $p(z^{(n)}|\boldsymbol{y}^{(n)}, \boldsymbol{\Theta}^*, \boldsymbol{q}^*)$ in which one of the clusters has at least a 0.95 calculated probability of being the true cluster for time series n.

One may expect the results to improve if we use more than two clusters. However, upon repeating the experiment for $K = 3$, there is no such improvement in the corresponding results. We see that one cluster clearly contains all seizure-related time series while a second cluster clearly contains all non-seizure-related time series. However, the third cluster becomes a roughly even, 50-50 mix of the two classes, thereby bringing the overall accuracy down to 86.55% – even if we group the third cluster with one of the other two.

Nonetheless, it seems that the EM inference algorithm does a relatively clean job of

correctly separating the data in an unsupervised manner when $K = 2$. This example with linear-Gaussian state-space models establishes that the finite mixture model has some utility in real-world settings. In the subsequent experiment, we explore time series that exhibit nonlinear dynamics. Thus, exact inference becomes impossible and the EM algorithm is forced to rely on particle methods instead of Kalman smoothing.

3.3.2 Neuronal Firing Clustering

We explore an application of time series clustering in a nonlinear setting. A neuron is a nerve cell that transmits electrical signals as part of the nervous system of every human. Neurons fire in response to different stimuli. These firings can be recorded as binary indicators over the course of a time period. Thus, a neuronal firing sequence is effectively a binary time series. Different neurons are believed to exhibit different firing patterns in response to different stimuli. Therefore, the problem of clustering neuronal firing patterns is important, because it can reveal hidden relationships between neurons that were not noticed before. The neuron is arguably the fundamental unit of learning and thought that exists in our brains, so it is of great interest to understand its characterizations in greater detail. In this application, we build a model and algorithm designed to recover a ground-truth clustering among several neuronal firing patterns. If the algorithm is able to yield successful results and perform as expected, then it may have utility in problems in which the ground-truth clustering is not known.

The objective is to build a model that inputs a time series of 0-1 neuronal firing indicators and accurately outputs different clusterings of neurons by their original stimulus. For this experiment, we used a dataset collected by Temereanca et al. (2008). The experimenters artificially moved the whiskers of mice at different velocities (50 mm/s and 16 mm/s). Theses velocities form the ground-truth clustering of the data, i.e. $K = 2$. For the two velocities, the experimenters recorded the corresponding firing sequences for N neurons over $T = 3000$ time points, with certain neurons receiving the *fast* stimulus (i.e. $k = 1$, 50 mm/s) and others receiving the *slow* stimulus (i.e. $k = 2$, 16 mm/s). At each time point in the experiment, a '1' indicates that the neuron fired, while a '0' indicates that the neuron did not fire. The experiment was repeated over a total of $R = 50$ trials.

Modeling Details

Thus, the data for each neuron n comes in the form of a raster $\{\Delta N_{t,r}^{(n)}\}_{t=1,r=1}^{T,R}$, where each $\Delta N_{t,r}^{(n)} \in \{0, 1\}$. Assuming independence between trials, we sum this raster across trials

to form a univariate time series $\boldsymbol{y}^{(n)}$, where each $y_t^{(n)} = \sum_{r=1}^{R} \Delta N_{t,r}^{(n)} \in \{0, \ldots, 50\}$. A popular way to specify a generative model for this time series is to use the binomial point process state-space model of Equation (13). Given multiple neurons that share common attributes, we impose a finite mixture of point process state-space models on the data:

$$
\begin{aligned}
\boldsymbol{q} &\sim \text{Dirichlet}(\boldsymbol{\alpha}), && (27)\\
z^{(n)} &\sim \text{Categorical}(\boldsymbol{q}), && 1 \leq n \leq N,\\
\boldsymbol{\theta}^{(k)} = \psi^{(k)} &\sim G, && 1 \leq k \leq K,\\
x_1^{(n)} &\sim \mathcal{N}({x_0}^{(n)}, \psi_0), && 1 \leq n \leq N,\\
x_t^{(n)} \mid x_{t-1}^{(n)}, z^{(n)} = k, \psi^{(k)} &\sim \mathcal{N}(x_{t-1}^{(n)}, \psi^{(k)}), && 1 < t \leq T, 1 \leq n \leq N,\\
y_t^{(n)} \mid x_t^{(n)} &\sim \text{Binomial}\left(R, \frac{\exp x_t^{(n)}}{1 + \exp x_t^{(n)}}\right), && 1 \leq t \leq T, 1 \leq n \leq N.
\end{aligned}
$$

Similar to the previous application, the cluster parameter in this problem is $\psi^{(k)}$, the variance of the underlying latent process. Looking at representative plots of neurons with fast and slow stimuli (Figure 7), we expect those receiving the fast stimulus to have a higher variance than those receiving a slow stimulus. Thus, the expectation is that this parameter can partition the data in a sensible manner.

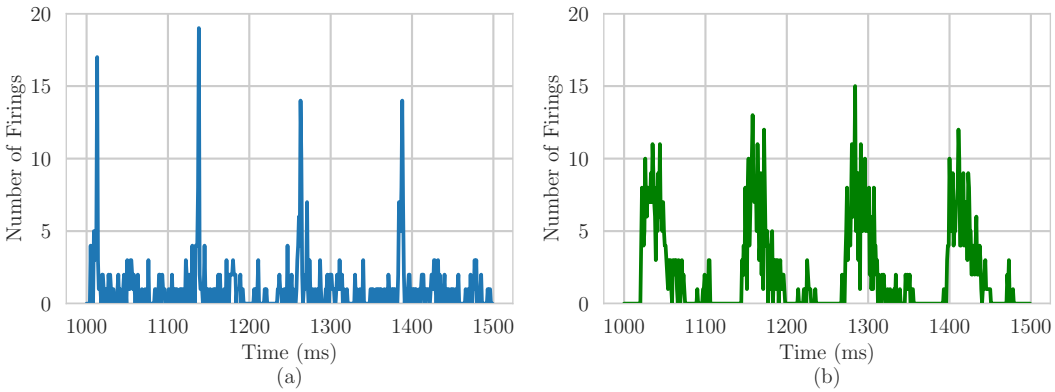

Figure 7: Sample segments of representative firing sequences $\{y_{1000}^{(n)}, \ldots, y_{1500}^{(n)}\}$, in which a neuron n experiences (a) the fast stimulus and (b) the slow stimulus.

MCEM Algorithm

Inference for this model essentially follows the same form as Algorithm 6, in the epilepsy example. However, there are a couple of notable differences. Two key quantities can no longer be directly computed due to the nonlinearities of Equation (27) and the lack of conjugacy between the Gaussian and the binomial. These quantities are the parameter log-likelihood $p\left(\boldsymbol{y}^{(n)} \mid \psi^{(k)\langle i \rangle}\right)$ and the smoothing quantity $\sum_{t=2}^{T} \mathbb{E}\left[(x_t^{(n)} - x_{t-1}^{(n)})^2 \mid \boldsymbol{y}^{(n)}, \psi^{(k)\langle i \rangle}\right]$. Thus, we introduce Monte Carlo methods for the E-Step. The parameter log-likelihood can be estimated using a *particle filter*. Controlled sequential Monte Carlo (Heng et al., 2017) is a natural choice, because it is designed to provide low-variance estimates. The smoothing quantity can be similarly estimated using a *particle smoother*. In fact, the output of the bootstrap particle filter (and cSMC by implication) can be altered to give estimates from the smoothing distribution of a state-space model. We elaborate on how this is done in the next section.

The M-Step updates are exactly the same as those in Algorithm 6, except we use estimates for the parameter log-likelihood and the smoothing quantity instead of exact calculations. A summary of the resultant MCEM algorithm is given in Algorithm 8.

Particle Smoothing

Consider the point process state-space model of Equation (13):

$$\begin{aligned} x_1 &\sim \mathcal{N}(x_0, \psi_0), \\ x_t \mid x_{t-1} &\sim \mathcal{N}(x_{t-1}^{(n)}, \psi), && 1 < t \leq T, \\ y_t \mid x_t &\sim \text{Binomial}\left(R, \frac{\exp x_t}{1 + \exp x_t}\right), && 1 \leq t \leq T. \end{aligned}$$

The problem of *particle smoothing* inputs a given time series of observations $\boldsymbol{y}$ and supplied parameters $\{\psi, x_0, \psi_0, \sigma^2\}$ and outputs sets of S particles $\{x_{1 \mid T}^s\}_{s=1}^S, \ldots, \{x_{T \mid T}^s\}_{s=1}^S$ that approximate the smoothing distribution $p(\boldsymbol{x} \mid \boldsymbol{y})$. Once we have these particles, we can compute the following unbiased estimate of $\sum_{t=2}^{T} \mathbb{E}\left[(x_t - x_{t-1})^2 \mid \boldsymbol{y}\right]$:

$$\frac{1}{S} \sum_{s=1}^{S} \sum_{t=2}^{T} (x_{t \mid T}^s - x_{t-1 \mid T}^s)^2. \tag{28}$$

As noted by Doucet and Johansen (2009), the output of a bootstrap particle filter (BPF) can be modified to produce $\{x_{1 \mid T}^s\}_{s=1}^S, \ldots, \{x_{T \mid T}^s\}_{s=1}^S$ from the smoothing distribution.

Algorithm 8 `NeuronClustMCExpMax`$(\boldsymbol{Y}, K, a, b, \boldsymbol{\alpha}, \psi_0, R, \{x_0^{(1)}, \ldots, x_0^{(N)}\}, S, L, \epsilon_0)$

Set $i = 0$.
Draw parameters $\psi^{(k)\langle 0\rangle} \sim \text{InvGamma}(a, b)$ for $k = 1, \ldots, K$ and $\boldsymbol{q}^{\langle 0\rangle} \sim \text{Dirichlet}(\boldsymbol{\alpha})$.
Set $\epsilon = \infty$.
while $\epsilon_0 < \epsilon$ **do**
  // Expectation Step
  for $n = 1, \ldots, N$ **do**
    for $k = 1, \ldots, K$ **do**
      Run `ParticleSmoother`$(\boldsymbol{y}, \psi^{(k)\langle i\rangle}, x_0^{(n)}, \psi_0, R)$.
      Estimate $\sum_{t=2}^{T} \mathbb{E}\left[(x_t^{(n)} - x_{t-1}^{(n)})^2 \mid \boldsymbol{y}^{(n)}, \psi^{(k)\langle i\rangle}\right]$ using Equation (28).
      Estimate $w^{(n,k)} = p(z^{(n)} = k \mid \boldsymbol{y}^{(n)}, \boldsymbol{q}^{\langle i\rangle}, \boldsymbol{\Theta}^{\langle i\rangle}) \propto q^{(k)\langle i\rangle} \cdot p\left(\boldsymbol{y}^{(n)} \mid \psi^{(k)\langle i\rangle}\right)$.
    end for
  end for
  // Maximization Step
  for $k = 1, \ldots, K$ **do**
    Optimize and solve for $q^{(k)\langle i+1\rangle}$ using Equation (18) and E-Step estimates.
    Optimize and solve for $\psi^{(k)\langle i+1\rangle}$ using Equation (24) and E-Step estimates.
  end for
  Compute $\epsilon = \|\boldsymbol{\Theta}^{\langle i+1\rangle} - \boldsymbol{\Theta}^{\langle i\rangle}\|_2$.
  Set $i = i + 1$.
end while
return $\boldsymbol{q}^{\langle i\rangle}, \boldsymbol{\Theta}^{\langle i\rangle}$

The BPF (Algorithm 4) returns particles $\{x_1^s\}_{s=1}^S, \ldots, \{x_T^s\}_{s=1}^S$ where each set of particles $\{x_t^s\}_{s=1}^S$ approximates $p(x_t \,|\, y_1, \ldots, y_{t-1})$. In Line 7 of Algorithm 4, we also sample ancestor indices $\{a_t^s\}_{s=1}^S$ to determine the trajectory of particles in the next time step $\{x_{t+1}^s\}_{s=1}^S$ for $t = 1, \ldots, T-1$. We can use these ancestor indices to get particles from the smoothing distribution by defining a new set of indices $\{b_1^s\}_{s=1}^S, \ldots, \{b_T^s\}_{s=1}^S$.

First, define weights for the T-th time step $w_T^s = g(x_T^s, y_T)$ for $s = 1, \ldots, S$ and normalize them so that $\sum_{s=1}^S w_T^s = 1$; for the point process state-space model, this state-dependent likelihood g is the probability mass function of the binomial. Then, sample the following indices:

$$b_T^s \sim \text{Categorical}(w_T^1, \ldots, w_T^S), \qquad s = 1, \ldots, S. \tag{29}$$

We can see that the set of particles $\{x_T^{b_T^s}\}_{s=1}^S$ form an approximation of $p(x_T \mid y_1, \ldots, y_T)$.

Algorithm 9 `ParticleSmoother`$(\boldsymbol{y}, \psi, x_0, \psi_0, R, S, L)$

Define $f(x_{t-1}, x_t) := \mathcal{N}(x_t^{(n)} \mid x_{t-1}^{(n)}, \psi)$, $g(x_t, y_t) := \text{Binomial}(y_t \mid R, \exp x_t/(1+\exp x_t))$, and $h(x_1) := \mathcal{N}(x_1 \mid x_0, \psi_0)$.
Collect particles $\{x_1^s\}_{s=1}^S, \ldots, \{x_T^s\}_{s=1}^S$, ancestor indicies $\{a_1^s\}_{s=1}^S, \ldots, \{a_T^s\}_{s=1}^S$, and final policy Γ from `ControlledSMC`$(\boldsymbol{y}, f, g, h, \{\psi, x_0, \psi_0\}, L)$.
for $s = 1, \ldots, S$ **do**
    Weight $w_T^s = g^\Gamma(x_T^s, y_T)$.
end for
Normalize $\{w_T^s\}_{s=1}^S = \{w_T^s\}_{s=1}^S / \sum_{s=1}^S w_T^s$.
for $s = 1, \ldots, S$ **do**
    Sample index $b_T^s \sim \text{Categorical}(w_T^1, \ldots, w_T^S)$.
    Define smoothing particle $x_{T \mid T}^s := x_T^{b_T^s}$.
end for
for $t = T-1, \ldots, 1$ **do**
    for $s = 1, \ldots, S$ **do**
        Define index $b_t^s = a_t^{b_{t+1}^s}$.
        Define smoothing particle $x_{t \mid T}^s := x_t^{b_t^s}$.
    end for
end for
return $\{x_{1 \mid T}^s\}_{s=1}^S, \ldots, \{x_{T \mid T}^s\}_{s=1}^S$

We can then use the following recursive procedure for all $t < T$:

$$b_t^s = a_t^{b_{t+1}^s}, \qquad s = 1, \ldots, S. \tag{30}$$

Then, it follows that $\{x_t^{b_t^s}\}_{s=1}^S$ is drawn from the smoothing distribution $p(x_t \mid y_1, \ldots, y_t)$ for all $1 \leq t \leq T$. Thus, we can define $\{x_{t \mid T}^s\}_{s=1}^S := \{x_t^{b_t^s}\}$ for all t. Doucet and Johansen (2009) note that this approximation of the smoothing distribution is typically poor, because of the particle degeneracy problem in which earlier time steps are really only represented by a single unique particle.

However, controlled sequential Monte Carlo can produce much better approximations of the smoothing distribution due to its twisting mechanism. After twisting the model, cSMC outputs particles using the BPF. Thus, we can modify this output using the process detailed by Equation (29) and Equation (30) to generate particles from the smoothing distribution $\{x_{1 \mid T}^s\}_{s=1}^S, \ldots, \{x_{T \mid T}^s\}_{s=1}^S$. Algorithm 9 summarizes this procedure.

Results

In total, we have data from $N = 24$ neurons that are fed into Algorithm 8. As stated earlier, for each neuron n, we know whether the true stimulus is the fast stimulus ($k = 1$) or the slow stimulus ($k = 2$). There are 12 neurons with the fast stimulus and 12 neurons with the slow stimulus. We let $K = 2$, the base distribution be InvGamma(1, 1), $\boldsymbol{\alpha} = [1, 1]$, and $\psi_0 = 10^{-10}$. The initial state $x_0^{(n)}$ for neuron n is empirically estimated as $\log(p_0^{(n)}/(1 - p_0^{(n)}))$, where $p_0^{(n)} = 1/(20 \cdot R) \cdot \sum_{t=1}^{20} y_t^{(n)}$. For cSMC hyperparameters, we let $L = 3$ iterations and $S = 128$ particles. Finally, we let $\epsilon_0 = 10^{-4}$. The algorithm returns final cluster parameters $\boldsymbol{\Theta}^*$ and $\boldsymbol{q}^*$.

Over 20 runs of Algorithm 8, we notice that $\boldsymbol{\Theta}^*$ and $\boldsymbol{q}^*$ are very stable for this problem. The average final parameter estimates are ${\psi^{(1)}}^* = 0.261$ and ${\psi^{(2)}}^* = 0.094$. We make a final assignment of neuron n to the cluster k that maximizes $p(z^{(n)} = k \mid \boldsymbol{y}^{(n)}, \boldsymbol{q}^*, \boldsymbol{\Theta}^*)$ over all $k = 1, \ldots, K$. Table 2 presents the resultant confusion matrix, which ended up being the same for all runs.

Table 2: Confusion matrix for a representative run of Algorithm 8 with $K = 2$.

	True Fast Stimulus	True Slow Stimulus
Clustered Fast Stimulus	11	4
Clustered Slow Stimulus	1	8

The results confirm the initial expectation that ${\psi^{(1)}}^* > {\psi^{(2)}}^*$, because there is greater variance for time series corresponding to the faster stimulus. The algorithm seems somewhat able to separate the two stimuli in an unsupervised manner, with 19 out of the original 24 time series being placed in correct clusters. Perhaps if more than one parameter were used for cluster assignment, then the partition would be more aligned with expectations.

4 State-Space Infinite Mixture Model and MCMC Algorithm

While the finite mixture model seems to successfully cluster time series in certain applications, an obvious limitation is that it relies on the specification of the number of clusters K. Thus, for real-world problems where it is difficult to determine this quantity, the finite mixture model may not be appropriate.

Furthermore, while the MCEM algorithm is efficient, it does have certain drawbacks, depending on the specific applications of interest. First of all, the final MCEM parameter estimates are often only *locally optimal* and can be arbitrarily far from the true MAP estimates. The sensitivity of EM to starting conditions often requires us to run the algorithm several times. In addition, the algorithm does not allow us to perform *full Bayesian inference* to find a joint posterior distribution over cluster assignments and cluster parameters $p(Z, \boldsymbol{\Theta} \mid \boldsymbol{Y})$. Finally, the tractability of the optimization problem in the M-Step depends on the specific nonlinearities in question; for certain nonlinearities, this step may be difficult or impossible to take.

In an attempt to address these issues, we develop an infinite mixture of nonlinear state-space models, also known as the *Dirichlet process nonlinear state-space mixture* (DPnSSM). We present a corresponding Markov chain Monte Carlo inference algorithm that leverages Metropolis-within-Gibbs and particle MCMC. We apply the model and inference algorithm to problems in computational neuroscience. The vast majority of results in this chapter can also be found in Lin et al. (2019).

4.1 Dirichlet Process Nonlinear State-Space Mixture

We employ the Chinese restaurant representation of the Dirichlet process mixture, which provides randomness over the number of clusters K. Let α be an inverse-variance hyperparameter and G be a base distribution. Given N times series $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$

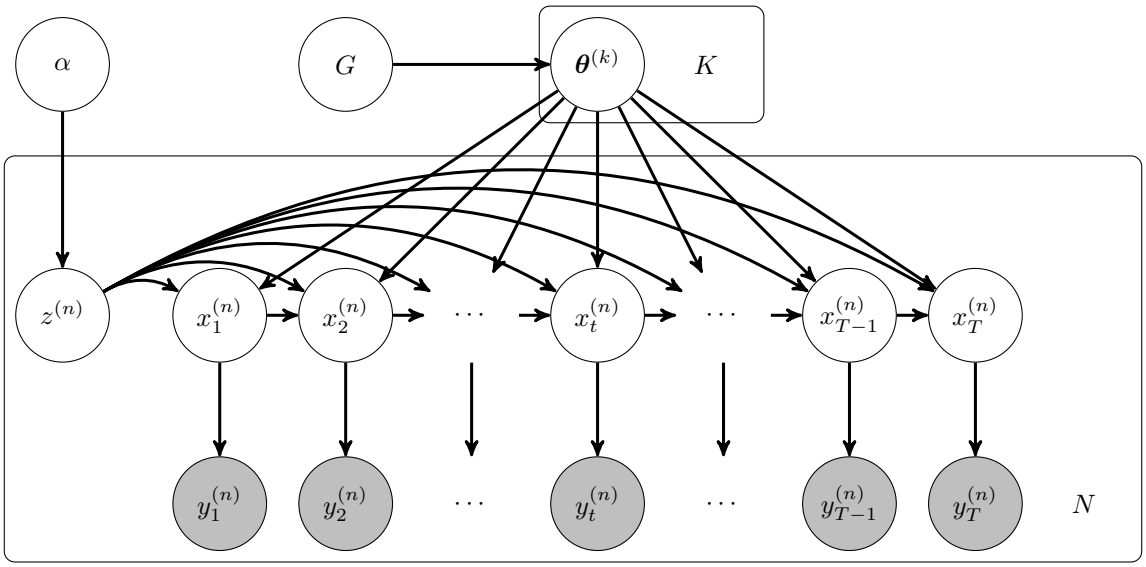

Figure 8: The graphical model representation of the DPnSSM. For simplicity, we omit dependencies that are assumed between $\boldsymbol{Y}$ and $(Z, \boldsymbol{\Theta})$.

of length T, the model has the following form:

$$
\begin{aligned}
z^{(1)}, \ldots, z^{(N)}, K &\sim \mathrm{CRP}(\alpha, N), && (31)\\
\boldsymbol{\theta}^{(k)} &\sim G, && 1 \leq k \leq K,\\
x_1^{(n)} \mid z^{(n)} = k, \boldsymbol{\theta}^{(k)} &\sim h(x_1^{(n)}; \boldsymbol{\theta}^{(k)}), && 1 \leq n \leq N,\\
x_t^{(n)} \mid x_{t-1}^{(n)}, z^{(n)} = k, \boldsymbol{\theta}^{(k)} &\sim f(x_{t-1}^{(n)}, x_t^{(n)}; \boldsymbol{\theta}^{(k)}), && 1 < t \leq T, 1 \leq n \leq N,\\
y_t^{(n)} \mid x_t^{(n)}, z^{(n)} = k, \boldsymbol{\theta}^{(k)} &\sim g(x_t^{(n)}, y_t^{(n)}; \boldsymbol{\theta}^{(k)}), && 1 \leq t \leq T, 1 \leq n \leq N.
\end{aligned}
$$

Figure 8 depicts the graphical model representation.

4.2 Inference Algorithm

For conducting posterior inference on the DPnSSM, we introduce a Metropolis-within-Gibbs sampling procedure inspired by Algorithm 8 from Neal (2000). We derive the following process for alternately sampling (1) the cluster assignments $Z \mid \boldsymbol{\Theta}, \boldsymbol{Y}, \alpha, G$ and (2) the cluster parameters $\boldsymbol{\Theta} \mid Z, \boldsymbol{Y}, \alpha, G$. A summary of the inference algorithm is given in Algorithm 10. Outputs are samples $\{Z^{\langle i \rangle}, \boldsymbol{\Theta}^{\langle i \rangle}\}$ for iterations $i = 1, 2, \ldots, I$.

4.2.1 Sampling Cluster Assignments

For any set $\mathcal{S} = \{s^{(1)}, \ldots, s^{(J)}\}$, we denote set $\mathcal{S}$ without the j-th element as $\mathcal{S}^{(-j)} = \mathcal{S} \setminus \{s^{(j)}\}$. The posterior distribution of the cluster assignment $z^{(n)}$ is:

$$\begin{aligned} p(z^{(n)} \mid Z^{(-n)}, \boldsymbol{\Theta}, \boldsymbol{Y}, \alpha, G) &\propto p(z^{(n)} \mid Z^{(-n)}, \boldsymbol{\Theta}, \alpha, G) \cdot p(\boldsymbol{Y} \mid Z, \boldsymbol{\Theta}, \alpha, G) \\ &\propto p(z^{(n)} \mid Z^{(-n)}, \alpha) \cdot p(\boldsymbol{y}^{(n)} \mid z^{(n)}, \boldsymbol{\Theta}, G). \end{aligned} \tag{32}$$

The first term in Equation (32) can be represented by the categorical distribution,

$$p(z^{(n)} = k \mid Z^{(-n)}, \alpha) = \begin{cases} \dfrac{N^{(k)}}{N - 1 + \alpha}, & k = 1, \ldots, K', \\ \dfrac{\alpha / m}{N - 1 + \alpha}, & k = K' + 1, \ldots, K' + m, \end{cases} \tag{33}$$

where K' is the number of unique k in $Z^{(-n)}$, $N^{(k)}$ is the number of cluster assignments equal to k in $Z^{(-n)}$, and $m \geq 1$ is some integer algorithmic parameter that represents the infinitely many clusters not found in $Z^{(-n)}$.

The second term in Equation (32) is equivalent to the parameter likelihood $p(\boldsymbol{y}^{(n)} \mid \boldsymbol{\theta}^{(k)})$, where $\boldsymbol{\theta}^{(k)}$ is known if $k \in \{1, \ldots, K'\}$; otherwise, $\boldsymbol{\theta}^{(k)}$ must first be sampled from G if $k \in \{K' + 1, \ldots, K' + m\}$. Since $\boldsymbol{y}^{(n)}$ is the output of a nonlinear state-space model, we must use particle methods to approximate this parameter likelihood. We employ controlled sequential Monte Carlo to produce low-variance estimates of this likelihood (Heng et al., 2017). We outline the details behind how we use cSMC in Section 4.3.

4.2.2 Sampling Cluster Parameters

For a given cluster $k \in \{1, \ldots, K\}$, we wish to sample from the distribution:

$$\begin{aligned} p(\boldsymbol{\theta}^{(k)} \mid \boldsymbol{\Theta}^{(-k)}, Z, \boldsymbol{Y}, \alpha, G) &\propto p(\boldsymbol{\theta}^{(k)} \mid \boldsymbol{\Theta}^{(-k)}, Z, \alpha, G) \cdot p(\boldsymbol{Y} \mid \boldsymbol{\Theta}, Z, \alpha, G) \\ &\propto p(\boldsymbol{\theta}^{(k)} \mid G) \cdot \prod_{n \mid z^{(n)} = k} p(\boldsymbol{y}^{(n)} \mid \boldsymbol{\theta}^{(k)}). \end{aligned} \tag{34}$$

The first term of Equation (34) is the probability density function of the base distribution, and the second term is a product of parameter likelihoods. Because the likelihood conditioned on class membership involves integration of the state sequence $\boldsymbol{x}^{(n)}$, and the prior G is on the parameters of the state sequence, marginalization destroys any conjugacy that might have existed between the state sequence prior and parameter priors.

Algorithm 10 `InferDPnSSM`$(\boldsymbol{Y}, \alpha, G, m, r, I, Z^{\langle 0 \rangle}, \boldsymbol{\Theta}^{\langle 0 \rangle})$

for $i = 1, \ldots, I$ **do**
  Let $Z = Z^{\langle i-1 \rangle}$ and $\boldsymbol{\Theta} = \boldsymbol{\Theta}^{\langle i-1 \rangle}$.
  // Sample cluster assignments.
  for $n = 1, \ldots, N$ **do**
    Let K' be the number of distinct k in $Z^{(-n)}$.
    for $k = 1, \ldots, K' + m$ **do**
      Run cSMC to compute $p(\boldsymbol{y}^{(n)} \mid \boldsymbol{\theta}^{(k)})$.
    end for
    Sample $z^{(n)} \mid Z^{(-n)}, \boldsymbol{\Theta}, \boldsymbol{Y}, \alpha, G$.
  end for
  Let K be the number of distinct k in Z.
  // Sample cluster parameters.
  for $k = 1, \ldots, K$ **do**
    Sample proposal $\boldsymbol{\theta}' \sim r(\boldsymbol{\theta}' \mid \boldsymbol{\theta}^{(k)})$.
    for $n \in \{1, \ldots, N\} \mid z^{(n)} = k$ **do**
      Run cSMC to compute $p\left(\boldsymbol{y}^{(n)} \mid \boldsymbol{\theta}'\right)$.
    end for
    Let $a = \frac{p(\boldsymbol{\theta}' \mid \boldsymbol{\Theta}^{(-k)}, Z, \boldsymbol{Y}, \alpha, G) \cdot r(\boldsymbol{\theta}^{(k)} \mid \boldsymbol{\theta}')}{p(\boldsymbol{\theta}^{(k)} \mid \boldsymbol{\Theta}^{(-k)}, Z, \boldsymbol{Y}, \alpha, G) \cdot r(\boldsymbol{\theta}' \mid \boldsymbol{\theta}^{(k)})}$.
    Let $\boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}'$ with probability $\min(a, 1)$.
  end for
  Let $Z^{\langle i \rangle} = Z$ and $\boldsymbol{\Theta}^{\langle i \rangle} = \boldsymbol{\Theta}$.
end for
return $(Z^{\langle 1 \rangle}, \boldsymbol{\Theta}^{\langle 1 \rangle}), \ldots, (Z^{\langle I \rangle}, \boldsymbol{\Theta}^{\langle I \rangle})$

To sample from the conditional posterior of parameters given cluster assignments, Middleton (2014) re-introduces the state sequence as part of his sampling algorithm for the linear Gaussian state-space case. We use an approach that obviates the need to re-introduce the state sequence and generalizes to scenarios where the prior on parameter and the state sequence may not have any conjugacy relationships. In particular, our sampler uses a Metropolis-Hastings step with proposal distribution $r(\boldsymbol{\theta}' \mid \boldsymbol{\theta})$ to sample from the class conditional distribution of parameters given cluster assignments. This effectively becomes one iteration of the well-known particle marginal Metropolis-Hastings algorithm (Andrieu et al., 2010). To evaluate the second term of Equation (34) for PMMH, we once again choose to use cSMC (Section 4.3).

4.2.3 The Case of Finite Mixture

It is simple to convert the DPnSSM into a finite mixture model in which the true number of clusters K is known a priori. Instead of using a Dirichlet process, we can simply use a Dirichlet($\boldsymbol{\alpha}$) distribution in which $\boldsymbol{\alpha}$ is a K-dimensional vector with each $\alpha^{(k)} > 0$ for $k = 1, \ldots, K$. Then, we can modify Equation (31) with:

$$\begin{aligned} \boldsymbol{q} \mid \boldsymbol{\alpha} &\sim \text{Dirichlet}(\boldsymbol{\alpha}), \\ z^{(1)}, \ldots, z^{(N)} \mid \boldsymbol{q} &\sim \text{Multinomial}(N, \boldsymbol{q}), \\ \boldsymbol{\theta}^{(k)} \mid G &\sim G, \qquad\qquad 1 \leq k \leq K, \end{aligned} \tag{35}$$

where $\boldsymbol{q}$ is an intermediary variable that is easy to integrate over.

The resultant inference algorithm is simpler. The only necessary modification to Algorithm 10 is that, when sampling cluster assignments, there is no longer any need for an auxiliary integer parameter $m \geq 1$ to represent the infinite mixture. Thus, Equation (33) becomes

$$p(z^{(n)} = k \mid Z^{(-n)}, \boldsymbol{\alpha}) = \frac{N^{(k)} + \alpha^{(k)} - 1}{N - 1 + \sum_{k'=1}^{K} \alpha^{(k')} - K}, \qquad k = 1, \ldots, K, \tag{36}$$

where $N^{(k)}$ is the number of cluster assignments equal to k in $Z^{(-n)}$. The process of sampling cluster parameters remains exactly the same as in the infinite mixture case.

4.3 Controlled Sequential Monte Carlo

Controlled sequential Monte Carlo (Heng et al., 2017) is the workhorse of the DPnSSM inference algorithm, appearing in both the sampling cluster assignments step and the sampling cluster parameters step. It is used to provide low-variance estimates of the parameter likelihood $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ for a fixed computational cost. Alternatively, we could use the bootstrap particle filter to provide these estimates, but it would not be as efficient. Section 2.5.2 and Algorithm 5 provide a general summary of how cSMC works. The remaining unspecified part of the algorithm is the `FindPolicy` function in Line 6. As stated earlier, the implementation of this function will vary depending on the identities of f, g, h in the nonlinear state-space model. In this section, we present a general scheme for finding a twisting policy γ that is justifiable for specific classes of nonlinear state-space models. This scheme, known as *approximate backwards recursion*, was first identified by

Heng et al. (2017) for the binomial state-space model.

4.3.1 Determining the Optimal Policy

In Equation (10), the variance of the estimate $\hat{p}^{\gamma}(\boldsymbol{y} \mid \boldsymbol{\theta})$ comes from the twisted state-dependent likelihoods $g_1^{\gamma}, \ldots, g_T^{\gamma}$. Thus, to minimize this variance, we would like g_t^{γ} to be as uniform as possible with respect to x_t. Let the optimal policy which minimizes this variance be denoted γ^*. It follows that

$$\begin{aligned} \gamma_T^*(x_T) &= g(x_T, y_T; \boldsymbol{\theta}), \\ \gamma_t^*(x_t) &= g(x_t, y_t; \boldsymbol{\theta}) \cdot F_{t+1}^{\gamma^*}(x_t; \boldsymbol{\theta}), \qquad 1 \leq t < T. \end{aligned} \tag{37}$$

Under γ^*, the likelihood estimate $\hat{p}^{\gamma^*}(\boldsymbol{y} \mid \boldsymbol{\theta}) = H^{\gamma^*} = p(\boldsymbol{y} \mid \boldsymbol{\theta})$ has zero variance. However, it may be infeasible for us to use γ^* in many cases, because the BPF algorithm requires us to sample x_t from $f_t^{\gamma^*}$ for all t. For example, under γ^*, we would have

$$f_T^{\gamma^*}(x_{T-1}, x_T; \boldsymbol{\theta}) \propto f(x_{T-1}, x_T; \boldsymbol{\theta}) \cdot \gamma_T^*(x_T) = f(x_{T-1}, x_T; \boldsymbol{\theta}) \cdot g(x_T, y_T; \boldsymbol{\theta}), \tag{38}$$

which may be impossible to directly sample from if f and g form an intractable posterior. In such a case, we must choose a suboptimal policy γ.

4.3.2 Choosing a Gaussian Policy

Consider the special case in which the latent state sequence $\boldsymbol{x}$ is Gaussian, i.e. in the popular class of nonlinear, Gaussian state-space models. This means that the state transition density and the initial distribution follow the form:

$$\begin{aligned} h(x_1; x_0, \psi_0, \mu_1) &= \mathcal{N}(x_1 \mid x_0 + \mu_1, \psi_0), \\ f(x_t, x_{t+1}; \psi, \mu_t) &= \mathcal{N}(x_{t+1} \mid x_t + \mu_t, \psi), \end{aligned} \tag{39}$$

for parameters $\boldsymbol{\theta} = \{x_0, \psi_0, \psi, \boldsymbol{\mu}\}$, where $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_T\}$. Assume further that the state-dependent likelihood $g(x_t, y_t)$ is a *log-concave* function of x_t; in subsequent sections, we give a class of examples in which this condition is satisfied.

Thus, we can show that $F_{t+1}^{\gamma^*}(x_t; \boldsymbol{\theta}) = \int f(x_t, x_{t+1}; \boldsymbol{\theta}) \gamma_{t+1}^*(x_{t+1}) dx_{t+1}$ must be log-concave in x_t. This further implies that for all t, $\gamma_t^*(x_t) = g(x_t, y_t) \cdot F_{t+1}^{\gamma^*}(x_t; \boldsymbol{\theta})$ is a log-concave function of x_t since the product of two log-concave functions is log-concave.

Hence, we have shown that the optimal policy $\gamma^* = \{\gamma_1^*, \ldots, \gamma_T^*\}$ is a series of log-concave functions. This justifies the approximation of each $\gamma_t^*(x_t)$ with a Gaussian function,

$$\gamma_t(x_t) = \exp(-a_t x_t^2 - b_t x_t - c_t), \quad (a_t, b_t, c_t) \in \mathbb{R}^3, \tag{40}$$

and thus, $f_t^\gamma(x_{t-1}, x_t; \boldsymbol{\theta}) \propto f(x_{t-1}, x_t; \boldsymbol{\theta}) \cdot \gamma_t(x_t)$ is also a Gaussian density that is easy to sample from when running the BPF algorithm.

We want to find the values of (a_t, b_t, c_t) that enforce $\gamma_t \approx \gamma_t^*$ for all t. One simple way to accomplish this goal is to find the (a_t, b_t, c_t) that minimizes the least-squares difference between γ_t and γ_t^* in log-space. That is, given a set of samples $\{x_t^1, \ldots, x_t^S\}$ for the random variable x_t, we solve for:

$$\begin{aligned}(a_t, b_t, c_t) &= \arg \min_{(a_t, b_t, c_t) \in \mathbb{R}^3} \sum_{s=1}^{S} \left[\log \gamma_t(x_t^s) - \log \gamma_t^*(x_t^s)\right]^2 \\ &= \arg \min_{(a_t, b_t, c_t) \in \mathbb{R}^3} \sum_{s=1}^{S} \left[-(a_t(x_t^s)^2 + b_t(x_t^s) + c_t) - \log \gamma_t^*(x_t^s)\right]^2.\end{aligned} \tag{41}$$

Also, in a slight abuse of notation, we redefine for all $t < T$,

$$\gamma_t^*(x_t) = g(x_t, y_t) \cdot F_{t+1}^\gamma(x_t; \boldsymbol{\theta}), \tag{42}$$

because when performing approximate backwards recursion, it is not possible to analytically solve for the intractable integral $F_{t+1}^{\gamma^*}(x_t; \boldsymbol{\theta})$.

In the aforementioned least-squares optimization problem, there is one additional constraint that we must take into account. Recall that $f_t^\gamma(x_{t-1}, x_t; \boldsymbol{\theta}) \propto f(x_{t-1}, x_t; \boldsymbol{\theta}) \cdot \gamma_t(x_t)$ is a Gaussian density that we sample from. Therefore, we must ensure that the variance of this distribution is positive, which places a constraint on γ_t and more specifically, the domain of (a_t, b_t, c_t). Using properties of Gaussians, we can perform algebraic manipulation to work out the following parameterizations of h^γ and f_t^γ:

$$\begin{aligned}h^\gamma(x_1; \boldsymbol{\theta}) &= \mathcal{N}\left(x_1 \,\middle|\, \frac{\psi_0^{-1} \cdot (x_0 + \mu_1) - b_1}{\psi_0^{-1} + 2a_1}, \frac{1}{\psi_0^{-1} + 2a_1}\right), \\ f_t^\gamma(x_{t-1}, x_t; \boldsymbol{\theta}) &= \mathcal{N}\left(x_t \,\middle|\, \frac{\psi^{-1} \cdot (x_{t-1} + \mu_t) - b_t}{\psi^{-1} + 2a_t}, \frac{1}{\psi^{-1} + 2a_t}\right), \qquad 1 < t \leq T.\end{aligned} \tag{43}$$

The corresponding normalizing terms for these densities are

$$H^{\gamma}(\boldsymbol{\theta}) = \frac{1}{\sqrt{1+2a_1\psi_0}} \exp\left(\frac{\psi_0^{-1}\cdot(x_0+\mu_1)-(b_1)^2}{2(\psi_0^{-1}+2a_1)} - \frac{(x_0+\mu_1)^2}{2\psi_0} - c_1\right), \tag{44}$$

$$F_t^{\gamma}(x_{t-1};\boldsymbol{\theta}) = \frac{1}{\sqrt{1+2a_t\psi}} \exp\left(\frac{\psi^{-1}\cdot(x_{t-1}+\mu_t)-(b_t)^2}{2(\psi^{-1}+2a_t)} - \frac{(x_{t-1}+\mu_t)^2}{2\psi} - c_t\right), \quad 1 < t \leq T.$$

Thus, to obtain (a_t, b_t, c_t) and consequently γ_t for all t, we solve the aforementioned least-squares minimization problem (Equation (41)) subject to the following constraints:

$$\begin{aligned} a_1 &> -\frac{1}{2\psi_0}, \\ a_t &> -\frac{1}{2\psi}, \qquad\qquad 1 < t \leq T. \end{aligned} \tag{45}$$

4.3.3 Repeated Twisting Mechanism

The mechanism of cSMC iteratively performs twisted sequential Monte Carlo with policies $\gamma^{\langle 1\rangle}, \gamma^{\langle 2\rangle}, \gamma^{\langle 3\rangle}, \ldots$ As established in Equation (11), twisting the original model using Γ' and twisting the new model using γ is equivalent to twisting the original model by a cumulative policy $\Gamma = \Gamma' \cdot \gamma$. Thus, assuming that both Γ' and γ are series of Gaussian functions with coefficients $\{(A'_1, B'_1, C'_1), \ldots, (A'_T, B'_T, C'_T)\}$ and $\{(a_1, b_1, c_1), \ldots, (a_T, b_T, c_T)\}$, respectively, then the cumulative policy Γ has Gaussian coefficients $\{(A_1, B_1, C_1), \ldots, (A_T, B_T, C_T)\}$, where each $(A_t, B_t, C_t) = (A'_t + a_t, B'_t + b_t, C'_t + c_t)$.

Algorithm 11 outlines the complete scheme in detail, assuming an existing policy Γ'. By embedding it inside Algorithm 5, we can run cSMC to produce low-variance likelihood estimates for state-space models with Gaussian latent states and nonlinear, log-concave state-dependent likelihoods.

4.3.4 Log-Concave State-Dependent Likelihoods

The justification in using Algorithm 11 relies on the key requirement that the state-dependent likelihood is log-concave. In this section, we give examples of state-space models in which this requirement is satisfied.

Dynamic Generalized Linear Model

The *dynamic generalized linear model* (West et al., 1985) is an extension of generalized linear models to the state-space framework. The state-dependent likelihood g is said to

Algorithm 11 `FindPolicy`(Γ', f, g, h, $\{x_1^s\}_{s=1}^S, \ldots, \{x_T^s\}_{s=1}^S$, $\boldsymbol{y}$)

Define $\gamma_T^*(x_T) = g_T^{\Gamma'}(x_T, y_T)$.
for $t = T, \ldots, 2$ **do**
  Solve $(a_t, b_t, c_t) = \arg\min_{(a_t,b_t,c_t)} \sum_{s=1}^S \left[-(a_t(x_t^s)^2 + b_t(x_t^s) + c_t) - \log \gamma_t^*(x_t^s)\right]^2$ subject to $a_t > -1/(2\psi) - A_t'$ using linear regression.
  Define new policy function $\gamma_t(x_t) = \exp(-a_t x_t^2 - b_t x_t - c_t)$.
  Define cumulative policy function $\Gamma_t(x_t) = \Gamma_t'(x_t) \cdot \gamma_t(x_t) = \exp(-A_t x_t^2 - B_t x_t - C_t)$.
  Define $f_t^\Gamma(x_{t-1}, x_t; \boldsymbol{\theta})$ and $F_t^\Gamma(x_{t-1}; \boldsymbol{\theta})$.
  if $t = T$ **then**
    Define $g_T^\Gamma(x_T, y_T) = g(x_T, y_T)/\Gamma_T(x_T)$.
  else
    Define $g_t^\Gamma(x_t, y_t) = g(x_t, y_t) \cdot F_{t+1}^\Gamma(x_t; \boldsymbol{\theta})/\Gamma_t(x_t)$.
  end if
  Define $\gamma_{t-1}^*(x_{t-1}) = g_{t-1}^{\Gamma'}(x_{t-1}, y_{t-1}) \cdot F_t^\Gamma(x_{t-1}; \boldsymbol{\theta})/F_t^{\Gamma'}(x_{t-1}; \boldsymbol{\theta})$.
end for
Solve $(a_1, b_1, c_1) = \arg\min_{(a_1,b_1,c_1)} \sum_{s=1}^S \left[-(a_1(x_1^s)^2 + b_1(x_1^s) + c_1) - \log \gamma_1^*(x_1^s)\right]^2$ subject to $a_1 > -1/(2\psi_0) - A_1'$ using linear regression.
Define new policy function $\gamma_1(x_1) = \exp(-a_1 x_1^2 - b_1 x_1 - c_1)$.
Define cumulative policy function $\Gamma_1(x_1) = \Gamma_1'(x_1) \cdot \gamma_1(x_1) = \exp(-A_1 x_t^2 - B_1 x_t - C_1)$.
Define Γ-twisted initial distribution $h^\Gamma(x_1; \boldsymbol{\theta})$ and $H^\Gamma(\boldsymbol{\theta})$.
Define $g_1^\Gamma(x_1, y_1) = H^\Gamma(\boldsymbol{\theta}) \cdot g(x_1, y_1) \cdot F_2^\Gamma(x_1; \boldsymbol{\theta})/\Gamma_1(x_1)$.
return Cumulative policy $\Gamma = \{\Gamma_1, \ldots, \Gamma_T\}$.

be part of the exponential family if it follows the form:

$$g(x_t, y_t) = \exp\left[\boldsymbol{\chi}(x_t)^\top \boldsymbol{\phi}(y_t) - \zeta(\boldsymbol{\chi}(x_t)) + \eta(y_t)\right], \tag{46}$$

where $\boldsymbol{\chi}(x_t)$ are the natural parameters, $\boldsymbol{\phi}(y_t)$ are the sufficient statistics, ζ is the log partition function and η is the log scaling function.

The log partition function ζ is so-named because it is the log of the partition function, or normalizing constant, $\int \exp[\boldsymbol{\chi}(x_t)^\top \boldsymbol{\phi}(y_t) + \eta(y_t)] dy_t$. We can prove that it is a convex function of the natural parameters $\boldsymbol{\chi}(x_t)$. We begin with the observation that $\int g(x_t, y_t) dy_t = 1$ for any value of x_t, since g must be a valid pdf of y_t. Then, we can differentiate both sides by $\boldsymbol{\chi}(x_t)$, which gives:

$$\begin{aligned} \mathbf{0} &= \int g(x_t, y_t) \left(\boldsymbol{\phi}(y_t) - \frac{d\zeta}{d\boldsymbol{\chi}(x_t)}\right) dy_t \\ \frac{d\zeta}{d\boldsymbol{\chi}(x_t)} &= \mathbb{E}[\boldsymbol{\phi}(y_t)]. \end{aligned} \tag{47}$$

Differentiating by $\boldsymbol{\chi}(x_t)$ again, it follows that:

$$\begin{aligned}
\mathbf{0} &= \frac{d}{d\boldsymbol{\chi}(x_t)} \left[\int g(x_t, y_t) \left(\boldsymbol{\phi}(y_t) - \frac{d\zeta}{d\boldsymbol{\chi}(x_t)} \right) dy_t \right] \\
0 &= \int \left[g(x_t, y_t) \left(-\frac{d^2\zeta}{d\boldsymbol{\chi}^2(x_t)} \right) + g(x_t, y_t) \left(\boldsymbol{\phi}(y_t) - \frac{d\zeta}{d\boldsymbol{\chi}(x_t)} \right)^2 \right] dy_t \\
\frac{d^2\zeta}{d\boldsymbol{\chi}^2(x_t)} &= \int g(x_t, y_t) \left(\boldsymbol{\phi}(y_t) - \mathbb{E}[\boldsymbol{\phi}(y_t)] \right)^2 dy_t \\
\frac{d^2\zeta}{d\boldsymbol{\chi}^2(x_t)} &= \mathbb{V}\mathrm{ar}[\boldsymbol{\phi}(y_t)].
\end{aligned} \tag{48}$$

Hence, the first and second derivatives of ζ are the mean and the variance of $\boldsymbol{\phi}(y_t)$ respectively. The fact that the second derivative is non-negative everywhere guarantees that ζ is a convex function of $\boldsymbol{\chi}(x_t)$ and consequently, g is a log-concave function of $\boldsymbol{\chi}(x_t)$.

In the special case where $\boldsymbol{\phi}(y_t) = y_t$, it follows that g belongs to the well-known subset of distributions called the *natural exponential family*. If we follow the literature of generalized linear models and let $\boldsymbol{\chi}(x_t) = w_t x_t$ for some scalar weight $w_t \in \mathbb{R}$ (i.e. this amounts to using the *canonical* link function), then we can rewrite Equation (46) as:

$$g(x_t, y_t) = \exp[w_t x_t \cdot y_t - \zeta(w_t x_t) + \eta(y_t)]. \tag{49}$$

In combining this equation with a Gaussian state transition density f and a Gaussian initial distribution h, the resultant model is introduced by West et al. (1985) as the dynamic generalized linear model. These models uphold the log-concavity of g as a function of x_t, thereby justifying the use of a Gaussian approximation to the optimal policy γ^*, as detailed Section 4.3.2. They are also widely used in applications such as neuroscience. Here are some notable examples of state-space models with a state-dependent likelihood belonging to the natural exponential family:

- **Binomial State-Space Model** The binomial log probability mass function follows the form:

$$\log g(x_t, y_t) = \log \binom{R_t}{y_t} + y_t \log \pi_t + (R_t - y_t) \log(1 - \pi_t), \tag{50}$$

  where the probability of success $\pi_t = \exp w_t x_t / (1 + \exp w_t x_t)$ and R_t is the total

number of trials at time t. Rewriting in natural exponential family form, we have

$$\log g(x_t, y_t) = w_t x_t \cdot y_t - R_t \log(1 + \exp w_t x_t) + \log \binom{R_t}{y_t}. \tag{51}$$

Neuroscience applications for the binomial case include Smith and Brown (2003).

- **Poisson State-Space Model** The poisson log probability mass function follows the form:

$$\log g(x_t, y_t) = y_t \log \lambda_t - \lambda_t - \log y_t!, \tag{52}$$

where the average rate $\lambda_t = \exp w_t x_t$. Rewriting in natural exponential family form, we have

$$\log g(x_t, y_t) = w_t x_t \cdot y_t - \exp w_t x_t - \log y_t!. \tag{53}$$

Neuroscience applications for the Poisson case include Paninski et al. (2007) and Pillow et al. (2008).

- **Negative Binomial State-Space Model** The negative binomial log probability mass function follows the form:

$$\log g(x_t, y_t) = \log \binom{y_t + r_t - 1}{y_t} + y_t \log \pi_t + r_t \log(1 - \pi_t), \tag{54}$$

where the probability of success $\pi_t = \exp w_t x_t / (1 + \exp w_t x_t)$ and r_t is the number of failed trials at time t. Rewriting in natural exponential family form, we have

$$\log g(x_t, y_t) = w_t x_t \cdot y_t - (y_t + r_t) \log(1 + \exp w_t x_t) + \log \binom{y_t + r_t - 1}{y_t}. \tag{55}$$

Neuroscience applications for the negative binomial case include Linderman et al. (2015) and Gao et al. (2015).

Stochastic Volatility

The stochastic volatility model is a well-known state-space model that is commonly used for financial data to explain the phenomenon of volatility clustering (Durbin and Koopman, 2012). The observations $y_1, \ldots, y_T$ typically correspond to the differences in the log of

asset prices. The simplest stochastic volatility model follows the state-space framework:

$$y_t \mid x_t \sim \mathcal{N}\left(\mu, \sigma^2 \exp w_t x_t\right), \tag{56}$$

with a Gaussian state-sequence $\boldsymbol{x}$, where μ and σ^2 are supplied parameters. Thus, the log state-dependent likelihood g follows the form:

$$\log g(x_t, y_t) = -\frac{1}{2}\log 2\pi - \frac{1}{2}\log \sigma^2 - \frac{1}{2} w_t x_t - \frac{1}{2\sigma^2}(y_t - \mu)^2 \exp(-w_t x_t), \tag{57}$$

which is a concave function of x_t.

4.4 Simulation

We conduct a simulated experiment to test the ability of the DPnSSM to yield desired results in a setting in which the ground truth clustering is known. The application is neuroscience, where is it common to model neuronal firing activity as arising from a point process state-space model (Section 2.5.3). We begin by describing the data simulation process, continue by elaborating on the modeling procedure, and finish with a discussion of the results.

4.4.1 Data Generation

We simulate $N = 25$ neuronal rasters that each record data for $R = 45$ trials over the time interval $(-500, \mathcal{T}]$ milliseconds (ms) before/after an exogenous stimulus is applied at 0 ms, where $\mathcal{T} = 1500$. For each trial, the resolution of the binary event sequence is $\delta = 1$ ms. We create bins of size $\Delta = M \cdot \delta$, where $M = 5$, and observe neuron n firing $\Delta N_{t,r}^{(n)} \leq M$ times during the t-th discrete time interval $(t\Delta - \Delta, t\Delta]$ for trial r.

We use the following process for generating the simulated data: For each neuron n, the initial rate is independently drawn as $\lambda^{(n)} \sim \text{Uniform}(10, 15)$ Hz. Each neuron's *type* is determined by the evolution of its discrete-time CIF $\lambda_t^{(n)}$ over time. We split the discrete-time intervals into three parts – $\boldsymbol{t}_1 = \{-99, \ldots, 0\}$, $\boldsymbol{t}_2 = \{1, \ldots, 50\}$, and $\boldsymbol{t}_3 = \{51, \ldots, 300\}$. We generate five neurons from each of the following five types:

1. *Excited, sustained* neurons with rate $\lambda_t^{(n)} = \lambda^{(n)}$ for $t \in \boldsymbol{t}_1$; rate $\lambda_t^{(n)} = \lambda^{(n)} \cdot \exp(1)$ for $t \in \boldsymbol{t}_2, \boldsymbol{t}_3$.
2. *Inhibited, sustained* neurons with rate $\lambda_t^{(n)} = \lambda^{(n)}$ for $t \in \boldsymbol{t}_1$; rate $\lambda_t^{(n)} = \lambda^{(n)} \cdot \exp(-1)$ for $t \in \boldsymbol{t}_2, \boldsymbol{t}_3$.

3. *Non-responsive* neurons with rate $\lambda_t^{(n)} = \lambda^{(n)}$ for $t \in \boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3$.
4. *Excited, unsustained* neurons with rate $\lambda_t^{(n)} = \lambda^{(n)}$ for $t \in \boldsymbol{t}_1$; rate $\lambda_t^{(n)} = \lambda^{(n)} \cdot \exp(1)$ for $t \in \boldsymbol{t}_2$; rate $\lambda_t^{(n)} = \lambda^{(n)}$ for $t \in \boldsymbol{t}_3$.
5. *Inhibited, unsustained* neurons with rate $\lambda_t^{(n)} = \lambda^{(n)}$ for $t \in \boldsymbol{t}_1$; rate $\lambda_t^{(n)} = \lambda^{(n)} \cdot \exp(-1)$ for $t \in \boldsymbol{t}_2$; rate $\lambda_t^{(n)} = \lambda^{(n)}$ for $t \in \boldsymbol{t}_3$.

Following the point-process state-space model of Equation (13) – which assumes independent and identically distributed trials – we simulate,

$$y_t^{(n)} = \sum_{r=1}^{R=45} \Delta N_{t,r}^{(n)} \sim \text{Binomial}(R \cdot M = 225, p_t^{(n)}), \tag{58}$$

where $p_t^{(n)} = \lambda_t^{(n)} \cdot \delta$ for $t = -99, \ldots, 300$. These are the observations $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$ that are fed to the DPnSSM. The model is then tasked with figuring out the original ground-truth clustering.

4.4.2 Modeling

In modeling these simulated data as coming from the DPnSSM, we employ a mixture of the generative process specified by Equation (13); that is,

$$\begin{aligned}
z^{(1)}, \ldots, z^{(N)}, K &\sim \text{CRP}(\alpha, N), && && (59)\\
\boldsymbol{\theta}^{(k)} = [\mu^{(k)}, \log \psi^{(k)}] &\sim G, && 1 \le k \le K,\\
x_1^{(n)} \mid z^{(n)} = k, \mu^{(k)} &\sim \mathcal{N}(x_0^{(n)} + \mu^{(k)}, \psi_0), && 1 \le n \le N,\\
x_t^{(n)} \mid x_{t-1}^{(n)}, z^{(n)} = k, \psi^{(k)} &\sim \mathcal{N}(x_{t-1}^{(n)}, \psi^{(k)}), && 1 < t \le T, 1 \le n \le N,\\
y_t^{(n)} \mid x_t^{(n)} &\sim \text{Binomial}(225, \sigma(x_t^{(n)})), && 1 \le t \le T, 1 \le n \le N,
\end{aligned}$$

where cluster parameters $\boldsymbol{\theta}^{(k)}$ arise from a Dirichlet process with base distribution G and σ is the sigmoid function. In Equation (59), we only include a stimulus parameter $\mu^{(k)}$ for time $t = 1$, because the stimulus is only applied at time $\tau = 0$. The parameter $\mu^{(k)}$ describes the extent to which the exogenous stimulus modulates the response of the neuron – a positive value of $\mu^{(k)}$ indicates excitation, a negative value indicates inhibition, and a value close to zero indicates no response. The state transition density f imposes a stochastic smoothness constraint on the CIF of neuron n, where $\psi^{(k)}$ controls the degree of smoothness. A small value of $\psi^{(k)}$ suggests that the neurons exhibit a sustained change in response to the stimulus, whereas a large value of $\psi^{(k)}$ indicates that the change is

unsustained. With respect to the DPnSSM, the goal is to cluster the neurons according to the extent of the initial response $\mu^{(k)}$ and how sustained the response is $\psi^{(k)}$.

The series are fed into Algorithm 10 with hyperparameter values $\alpha = 1$, $G[\mu, \log \psi] = [\mathcal{N}(0, 2), \text{Uniform}(-15, 0)]$, and $m = 5$. For every series n, we compute the initial state $x_0^{(n)} = \sigma^{-1}(1/500 \cdot \sum_{t=-99}^{0} y_t^{(n)})$ from the observations before the stimulus in that series. In addition, we let $\psi_0 = 10^{-10}$, a very small value that forces any change in the latent state at $t = 1$ to be explained by the cluster parameter $\mu^{(k)}$. The initial clustering is $(Z^{\langle 0 \rangle}, \boldsymbol{\Theta}^{\langle 0 \rangle}) = (\mathbf{1}, \boldsymbol{\theta}_0)$, where $\mathbf{1}$ is a vector of N ones denoting that every series begins in the same cluster and $\boldsymbol{\theta}_0$ is sampled from G. For the proposal $r(\boldsymbol{\theta}' \mid \boldsymbol{\theta})$, we use a $\mathcal{N}(\boldsymbol{\theta}, 0.25 \cdot \mathbf{I})$ distribution, where $\mathbf{I}$ is the 2×2 identity matrix. We run the sampling procedure for I = 10,000 iterations and apply a burn-in of B = 1,000 samples. To compute likelihood estimates, we use $L = 3$ cSMC iterations and $S = 64$ particles.

4.4.3 Selecting Clusters

The output of Algorithm 10 is a set of Gibbs samples $(Z^{\langle 1 \rangle}, \boldsymbol{\Theta}^{\langle 1 \rangle}), \ldots, (Z^{\langle I \rangle}, \boldsymbol{\Theta}^{\langle I \rangle})$. Each sample $(Z^{\langle i \rangle}, \boldsymbol{\Theta}^{\langle i \rangle})$ may very well use a different number of clusters. The natural question that remains is how to select a single final clustering $(Z^*, \boldsymbol{\Theta}^*)$ of our data from this output. There is a great deal of literature on answering this subjective question. We follow the work of Dahl (2006) and Nieto-Barajas and Contreras-Cristán (2014).

Each Gibbs sample describes a clustering of the time series; we therefore frame the objective as selecting the most representative sample from our output. To start, we take each Gibbs sample i and construct an $N \times N$ co-occurrence matrix $\boldsymbol{\Omega}^{\langle i \rangle}$ in which,

$$\boldsymbol{\Omega}^{\langle i \rangle}_{(n,n')} = \begin{cases} 1, & z^{(n)} = z^{(n')} \mid z^{(n)}, z^{(n')} \in Z^{\langle i \rangle}, \\ 0, & z^{(n)} \neq z^{(n')} \mid z^{(n)}, z^{(n')} \in Z^{\langle i \rangle}. \end{cases} \tag{60}$$

This is simply a matrix in which the (n, n') entry is 1 if series n and series n' are in the same cluster for the i-th Gibbs sample and 0 otherwise. We then define $\boldsymbol{\Omega} = (I - B)^{-1} \sum_{i=B+1}^{I} \boldsymbol{\Omega}^{\langle i \rangle}$ as the mean co-occurrence matrix, where $B \geq 1$ is the number of pre-burn-in samples. This matrix summarizes information from the entire trace of Gibbs samples. The sample i^* that we ultimately select is the one that minimizes the Frobenius distance to this matrix, i.e. $i^* = \arg\min_i \|\boldsymbol{\Omega}^{\langle i \rangle} - \boldsymbol{\Omega}\|_F$. We use the corresponding assignments and parameters as the final clustering $(Z^*, \boldsymbol{\Theta}^*) = (Z^{\langle i^* \rangle}, \boldsymbol{\Theta}^{\langle i^* \rangle})$. The appeal of this procedure is that it makes use of global information from all the Gibbs samples,

yet ultimately selects a single clustering produced by the model. If there are $J > 1$ Gibbs samples $i_1, \ldots, i_J$ such that $\boldsymbol{\Omega}^{\langle i_j \rangle} = \boldsymbol{\Omega}^{\langle i^* \rangle}$ for $j = 1, \ldots, J$, our final cluster parameters $\boldsymbol{\Theta}^*$ can be redefined as the average among the corresponding parameter samples,

$$\boldsymbol{\Theta}^* = \frac{1}{J} \sum_{j=1}^{J} \boldsymbol{\Theta}^{\langle i_j \rangle} \approx \mathbb{E}[\boldsymbol{\Theta} \mid Z = Z^*]. \tag{61}$$

This averaging must be preceded by a permutation of each set of $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(K)} \in \boldsymbol{\Theta}^{\langle i_j^* \rangle}$ to fix any potential label switching.

4.4.4 Results

After running the algorithm, a heatmap of the resultant mean co-occurrence matrix $\boldsymbol{\Omega}$ (Equation (60)) and the selected clustering $\boldsymbol{\Omega}^{\langle i^* \rangle}$ can be found in Figure 9. The rows and columns of this matrix have been reordered to aid visualization of clusters along the diagonal of $\boldsymbol{\Omega}$. From this experiment, we can see that the DPnSSM inference algorithm is able to successfully recover the five ground-truth clusters.

Table 3 summarizes the final cluster parameters $\boldsymbol{\Theta}^*$. As one may expect, a highly positive $\mu^{*(k)}$ corresponds to neurons that are excited by the stimulus, while a highly negative $\mu^{*(k)}$ corresponds to neurons that are inhibited. The one cluster with $\mu^{*(k)} \approx 0$ corresponds to non-responsive neurons. With $\mu^{*(k)}$, the algorithm is able to approximately recover the true amount by which the stimulus increases or decreases the log of the firing rate/probability, which is stated in Section 4.4.1 – i.e. $+1$ for $k \in \{1, 4\}$, -1 for $k \in \{2, 5\}$ and 0 for $k = 3$. This provides an interpretation of the numerical value of $\mu^{(k)}$. Indeed, if $\exp x_0^{(n)} << 1, \exp x_1^{(n)} << 1$, as is often the case when modeling neurons, then the expected increase in the log of the firing probability due to the stimulus is:

$$\mathbb{E}\left[\log \frac{\sigma(x_1^{(n)})}{\sigma(x_0^{(n)})}\right] \approx \mathbb{E}\left[\log \frac{\exp x_1^{(n)}}{\exp x_0^{(n)}}\right] = \mu^{(k)}. \tag{62}$$

In addition, the values for $\log \psi^{*(k)}$ in Table 3 reveal that the algorithm uses this dimension to correctly separate unsustained clusters from sustained ones. For $k \in \{1, 2, 3\}$, the algorithm infers smaller values of $\log \psi^{*(k)}$ because the change in the firing rate is less variable after the stimulus has taken place, whereas the opposite is true for $k \in \{4, 5\}$. In summary, the DPnSSM is able to recover some of the key properties of the data in an unsupervised fashion, thereby demonstrating its utility on this toy example.

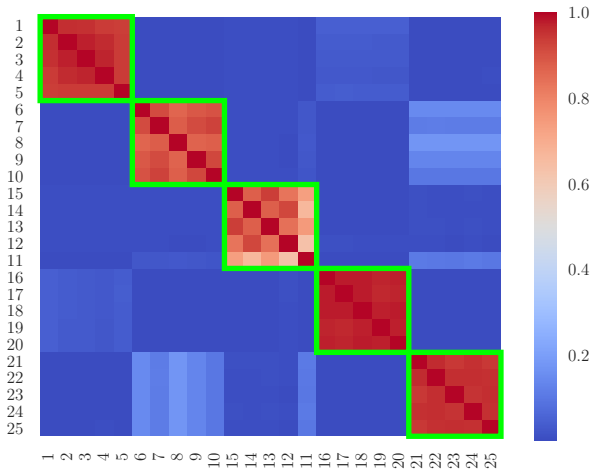

Figure 9: Heatmap of mean co-occurrence matrix $\mathbf{\Omega}$ for simulation results. Elements of the selected co-occurrence matrix $\mathbf{\Omega}^{\langle i^* \rangle}$ that are equal to 1 are enclosed in green squares. Each square corresponds to a distinct cluster.

Table 3: Parameters for simulation, where $\boldsymbol{\theta}^{*(k)} = [\mu^{*(k)}, \log \psi^{*(k)}]^\top$ for each $\boldsymbol{\theta}^{*(k)} \in \mathbf{\Theta}^*$.

k	$\mu^{*(k)}$	$\log \psi^{*(k)}$	**Effect**
1	$+0.96$	-10.88	Excited, Sustained
2	-0.92	-12.32	Inhibited, Sustained
3	$+0.03$	-10.04	Non-responsive
4	$+1.04$	-5.55	Excited, Unsustained
5	-0.89	-5.89	Inhibited, Unsustained

4.5 Applications

We present two real-world applications of the DPnSSM to clustering time series problems in neuroscience. In both of these problems, the true number of clusters is unknown, which prompts us to use the infinite mixture version. Both problems address data from the same experiment, which we briefly outline.

Allsop et al. (2018) conduct a fear-conditioning experiment designed to elucidate the nature of neural circuits that facilitate the observational learning of fear. In short, an observer mouse observes a demonstrator mouse receive conditioned cue-shock pairings through a perforated transparent divider. The experiment consists of $R = 45$ trials. During the first 15 trials of the experiment, both the observer and the demonstrator hear an auditory cue at time $\tau = 0$ ms. From trial 16 onwards, the auditory cue is followed by

the delivery of a shock to the demonstrator at time $\tau = 10{,}000$ ms, i.e. 10 seconds after the cue's administration. The data are recorded from various neurons in the prefrontal cortex of the observer mouse. The two main regions of neuronal recording are the anterior cingulate cortex (ACC) and the basolateral amygdala (BLA). The detailed experimental paradigm is described in Allsop et al. (2018).

4.5.1 Identification of Stimulus-Locked Response Profiles

First, we apply our analysis to $N = 33$ ACC neurons from this experiment that form a network hypothesized to be involved in the observational learning of fear. Our time interval of focus is $(-500, \mathcal{T} = 1500]$ ms before/after the administration of the cue. The raster data comes in the form of $\{\Delta N_{t,r}^{(n)}\}_{t=1,r=1}^{T,R}$, binned at a resolution of $\Delta = 5$ ms with $T = 300$, where each $\Delta N_{t,r}^{(n)} \leq M = 5$.

The goal is to use the DPnSSM to identify various groups, or clusters, of responses in reaction to the auditory cue over time and over trials. Each cluster of neurons form a common stimulus-locked response, which suggests that they may be involved in the same learning process. For example, a group of neurons that respond significantly after trial 16 can be interpreted as one that allows the observer to understand when the demonstrator is in distress.

Clustering Cue Data over Time

To cluster neurons by their cue responses over time, we collapse the raster for all neurons over the $R = 45$ trials. Thus, for neuron n, define $y_t^{(n)} = \sum_{r=1}^{R} \Delta N_{t,r}^{(n)}$. We apply the exact same model as the one used for the simulations (Equation (59)). We also use all of the same hyperparameter values, as detailed in Section 4.4.2.

To choose clusters, we use the same process as the one described in Section 4.4.3. A heatmap of $\mathbf{\Omega}$ along with demarcations of $\mathbf{\Omega}^{\langle i^* \rangle}$ for this experiment can be found in Figure 10. Overall, five clusters are selected by the algorithm. Table 4 summarizes the chosen cluster parameters. Figure 11 shows two of the five clusters identified by the algorithm, namely those corresponding to $k = 1$ (Figure 11a) and $k = 5$ (Figure 11b).

Table 4: Cluster parameters for cue data over time.

k	$\mu^{*(k)}$	$\log \psi^{*(k)}$	# of Neurons
1	$+0.54$	-6.27	17
2	$+0.03$	-6.80	1
3	-0.07	-7.25	8
4	-0.70	-6.01	4
5	$+1.21$	-4.52	3

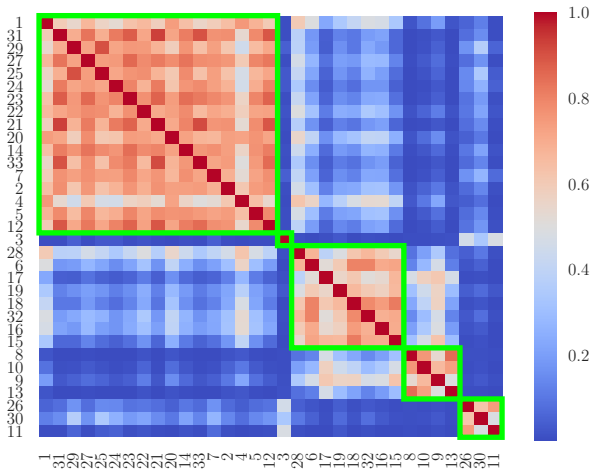

Figure 10: Heatmap of mean co-occurrence matrix for cue data over time and selected clusters (green).

Each of the figures was created by overlaying the rasters from neurons in the corresponding cluster. The fact that the overlaid rasters resemble the raster from a single unit (as opposed to random noise), with plausible parameter values in Table 4, indicates that the algorithm has identified a sensible clustering of the neurons. One advantage of not restricting the algorithm to a set number of classes a priori is that it can decide what number of classes best characterizes these data. In this case, the inference algorithm identifies five different clusters. We defer a scientific interpretation of this phenomenon to a later study.

Clustering Cue Data over Trials

We also apply DPnSSM to determine if neurons can be classified according to varying degrees of neuronal signal modulation when shock is delivered to another animal, as opposed to when there is no shock delivered. The shock is administered starting from the 16th trial onwards. Thus, to understand the varying levels of shock effect, we collapse the raster across the $T = 300$ time points (instead of the $R = 45$ trials, as was done in Section 4.5.1). In this setting,

Table 5: Cluster parameters for cue data over trials.

k	$\mu^{*(k)}$	$\log \psi^{*(k)}$	# of Neurons
1	$+0.19$	-5.29	8
2	-0.14	-4.40	11
3	-0.08	-2.38	2
4	$+0.87$	-2.95	10
5	-0.42	-0.41	2

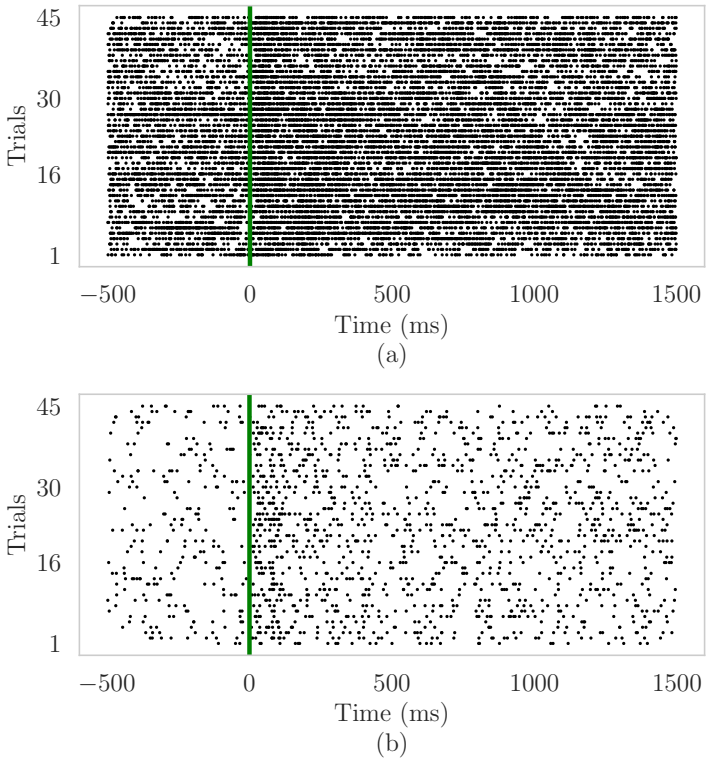

Figure 11: Overlaid raster plots of neuronal clusters with (a) moderately excited ($\mu^{*(1)} = 0.54$) and somewhat sustained ($\log \psi^{*(1)} = -6.27$) responses to the cue; and (b) more excited ($\mu^{*(5)} = 1.21$) and less sustained responses ($\log \psi^{*(5)} = -4.52$) to the cue. A black dot at (τ, r) indicates a spike from one of the neurons in the corresponding cluster at time τ during trial r. The vertical green line indicates cue onset.

each $y_r^{(n)} = \sum_{t=1}^{T} \Delta N_{t,r}^{(n)} \in \{0, 1, \ldots, 2000\}$ represents the number of firings during the r-th trial. For each neuron n, let the initial state be $x_0^{(n)} = \sigma^{-1}(1/15 \cdot \sum_{r=1}^{15} y_r^{(n)})$. Then, we use the following state-space mixture:

$$
\begin{aligned}
z^{(1)}, \ldots, z^{(N)}, K &\sim \mathrm{CRP}(\alpha, N), && && (63)\\
\boldsymbol{\theta}^{(k)} = [\mu^{(k)}, \log \psi^{(k)}] &\sim G, && 1 \le k \le K,\\
x_{16}^{(n)} \mid \mu^{(k)} &\sim \mathcal{N}(x_0^{(n)} + \mu^{(k)}, \psi_0), && 1 \le n \le N,\\
x_r^{(n)} \mid x_{r-1}^{(n)}, \psi^{(k)} &\sim \mathcal{N}(x_{r-1}^{(n)}, \psi^{(k)}), && 16 < r \le R, 1 \le n \le N,\\
y_r^{(n)} \mid x_r^{(n)} &\sim \mathrm{Binomial}(2000, \sigma(x_r^{(n)})), && 16 \le r \le R, 1 \le n \le N,
\end{aligned}
$$

where once again the cluster parameters are $\boldsymbol{\theta}^{(k)} = [\mu^{(k)}, \log \psi^{(k)}]^\top$. All other hyperparameter values are the same as those listed in Section 4.4.2.

The corresponding heatmap, representative raster plots, and clustering results can be found in Figure 12, Figure 13, and Table 5, respectively. We speculate that the results suggest the existence of what we term *empathy clusters*, namely groups of neurons that allow an observer to understand when the demonstrator is in distress.

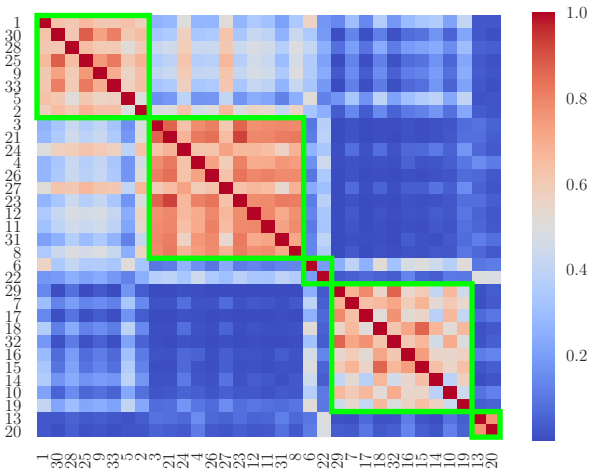

Figure 12: Heatmap of mean co-occurrence matrix for cue data over trials and selected clusters (green).

4.5.2 Network Analysis of Interacting Neuronal Firing Regions

In their study, Allsop et al. (2018) hypothesize on the nature of the interaction between the ACC and the BLA regions of the brain by analyzing firing data from neurons one at a time. Their analysis leads them to speculate that ACC neurons transmit socially derived information to the BLA during the cue presentation, thereby allowing the BLA within the observer to associate the cue with a shock to the demonstrator. This is a hypothesis on the neural mechanism behind observational learning. In this section, we use the DPnSSM to reaffirm this hypothesis by analyzing the firing data is a more automated way.

Allsop et al. (2018) follow a two-step process to arrive at this conclusion: First, they identify specific neurons in both the ACC and the BLA as *cue responsive neurons*, i.e. neurons that respond significantly to the cue in some way. The response can be either excitation or inhibition with respect to the cue. This categorization is mainly done by eye. Next, Allsop et al. (2018) use state-space analysis (Smith and Brown, 2003) on each cue responsive neuron to determine its *rate change trial*, i.e. the first trial after shock delivery in which the neuron begins to respond significantly differently to the cue. They

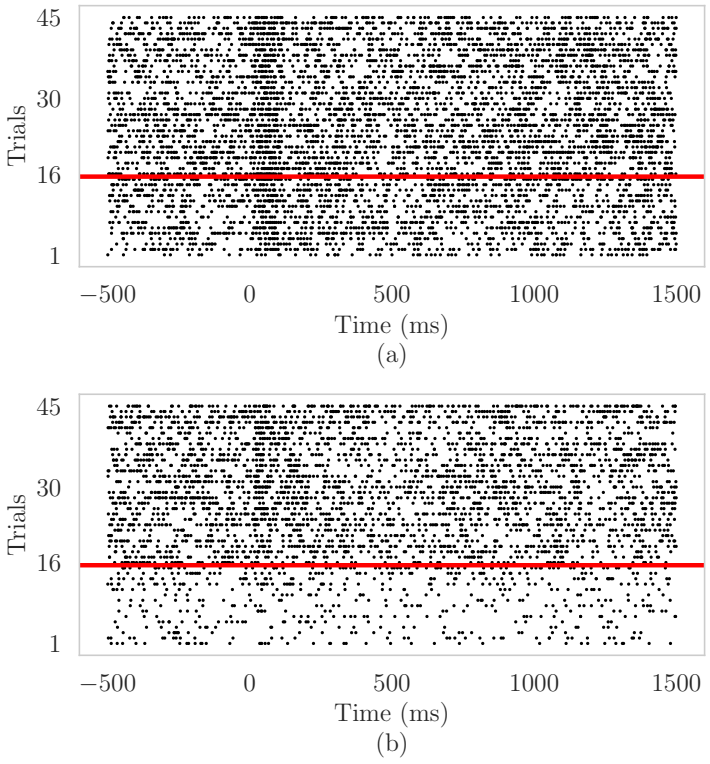

Figure 13: (a) Overlaid raster plots of neuronal clusters with (a) slightly inhibited, variable responses ($k = 2$) and (b) very excited, variable responses ($k = 4$). The red line marks the first trial with shock administration.

find that BLA neurons have, on average, a larger rate change trial than ACC neurons. This leads them to conclude that ACC neurons transmit information to the BLA during fear conditioning.

In this section, we follow the same two-step process, but use the DPnSSM to obviate the need to analyze the neuron data individually and manually. We show that our analysis leads us to similar conclusions as (Allsop et al., 2018), suggesting that the DPnSSM may have some utility in supplementing neuroscience research.

Similar to the last experiment, the time interval of focus is $(-500, \mathcal{T} = 1500]$ ms before/after the administration of the cue. Once again, the raster data comes in the form of $\{\Delta N_{t,r}^{(n)}\}_{t=1,r=1}^{T,R}$, binned at a resolution of $\Delta = 5$ ms with $T = 300$, where each $\Delta N_{t,r}^{(n)} \leq M = 5$. We examine a total of $N = 95$ neurons with an average firing rate of at least 1 Hz over the time interval of interest. There are 68 neurons from the ACC and 27 neurons from the BLA.

Identification of Cue Responsive Neurons

We use the DPnSSM to identify which of the 96 neurons are *cue responsive*. The neurons are hypothesized to work as a greater network, so one would expect various clusters of responses to the cue. This is the rationale behind using a mixture model for the data. Instead of the two-parameter clusterings of Equation (59) and Equation (63), we only use a one-parameter mixture model in this setting, because we only wish to separate the neurons based on the level of initial response to the cue. Instead of the binomial state-space model (Equation (13) and Equation (51)), we use the equally popular Poisson state-space model (Equation (53)) to demonstrate the flexibility of the DPnSSM. This is the full generative model that we employ:

$$
\begin{aligned}
z^{(1)}, \ldots, z^{(N)}, K &\sim \mathrm{CRP}(\alpha, N), && && (64)\\
\boldsymbol{\theta}^{(k)} = \mu^{(k)} &\sim G, && 1 \le k \le K,\\
x_1^{(n)} \mid z^{(n)} = k, \mu^{(k)} &\sim \mathcal{N}(x_0^{(n)} + \mu^{(k)}, \psi_0), && 1 \le n \le N,\\
x_t^{(n)} \mid x_{t-1}^{(n)} &\sim \mathcal{N}(x_{t-1}^{(n)}, \psi), && 1 < t \le T, 1 \le n \le N,\\
y_t^{(n)} \mid x_t^{(n)} &\sim \mathrm{Poisson}(\exp x_t^{(n)}), && 1 \le t \le T, 1 \le n \le N.
\end{aligned}
$$

where the data $y_t^{(n)} = \sum_{r=1}^{R} \Delta N_{t,r}^{(n)}$ are collapsed across trial.

Posterior inference on this model is done using Algorithm 10 with hyperparameter values $\alpha = 1$, $G = \mathcal{N}(0, 2)$, and $m = 5$. For every series n, we compute the initial state $x_0^{(n)} = \log(1/500 \cdot \sum_{t=-99}^{0} y_t^{(n)})$ from the observations before the stimulus in that series. In addition, we let $\psi = 10^{-5}, \psi_0 = 10^{-10}$, i.e. very small values forcing any significant change in the latent state to be explained by the cluster parameter $\mu^{(k)}$.

For the initial clustering, we use $(Z^{\langle 0 \rangle}, \boldsymbol{\Theta}^{\langle 0 \rangle}) = (\mathbf{1}, \boldsymbol{\theta}_0)$, where $\mathbf{1}$ is a vector of N ones and $\boldsymbol{\theta}_0 \sim G$. We set the proposal $r(\boldsymbol{\theta}' \mid \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}, 0.25)$, run the algorithm for $I = 10{,}000$ iterations, and apply a burn-in of $B = 1{,}000$ samples. The cSMC parameters are $L = 3$ iterations and $S = 64$ particles.

After burn-in, we collect posterior samples $(Z^{\langle B+1 \rangle}, \boldsymbol{\Theta}^{\langle B+1 \rangle}), \ldots, (Z^{\langle I \rangle}, \boldsymbol{\Theta}^{\langle I \rangle})$. In the model detailed by Equation (64), the cue response parameter $\mu^{(k)}$ is indexed by the cluster index k. However, we can index this parameter by the neuron index n instead by defining the variable $\tilde{\mu}^{(n)} = \mu^{(k)}$ where $k = z^{(n)}$. That is, $\tilde{\mu}^{(n)}$ is the cue response parameter for neuron n.

We identify a neuron n as *cue responsive* if the posterior probability of a change in the latent firing rate at time τ is at least 90%. In terms of the model, we define this as either

$p(\tilde{\mu}^{(n)} > 0 \mid \boldsymbol{Y}) \geq 0.9$ or $p(\tilde{\mu}^{(n)} < 0 \mid \boldsymbol{Y}) \geq 0.9$. From the Gibbs samples, we can estimate these quantities as:

$$\hat{p}(\tilde{\mu}^{(n)} > 0 \mid \boldsymbol{Y}) = \frac{1}{I-B} \sum_{i=B+1}^{I} \mathbb{I}\left(\mu^{(k_i)\langle i \rangle} > 0\right), \quad \text{where } k_i = z^{(n)\langle i \rangle}, \quad 1 \leq n \leq N, \quad (65)$$

$$\hat{p}(\tilde{\mu}^{(n)} < 0 \mid \boldsymbol{Y}) = \frac{1}{I-B} \sum_{i=B+1}^{I} \mathbb{I}\left(\mu^{(k_i)\langle i \rangle} < 0\right), \quad \text{where } k_i = z^{(n)\langle i \rangle}, \quad 1 \leq n \leq N,$$

where $\mathbb{I}(\cdot)$ is the indicator function of whether an event is true. Figure 14 shows a visualization of which neurons are deemed *cue responsive*. Out of the $N = 96$ neurons, a total of 59 (42 ACC, 17 BLA) satisfy the aforementioned requirement. We will use these 59 in the subsequent analysis.

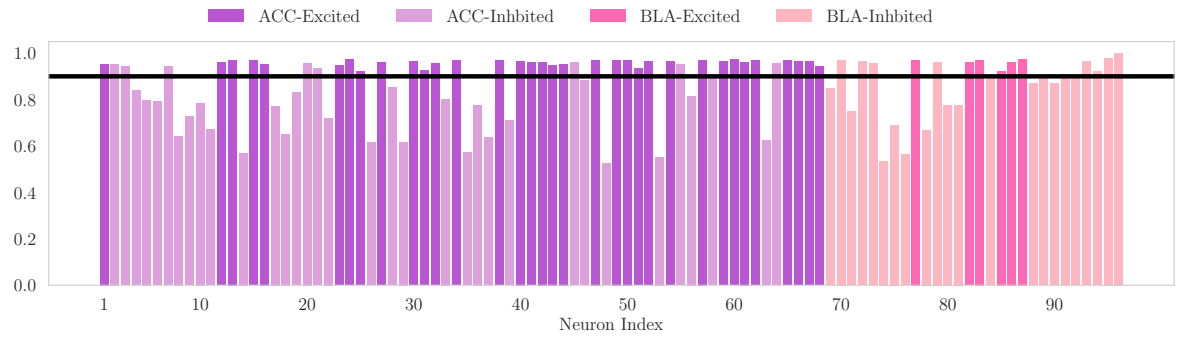

Figure 14: A bar plot of $\max\{\hat{p}(\tilde{\mu}^{(n)} > 0 \mid \boldsymbol{Y}), \hat{p}(\tilde{\mu}^{(n)} < 0 \mid \boldsymbol{Y})\}$ for neuron $n = 1, \ldots, N$ in the context of Equation (64). Neurons that exhibit this quantity being greater than a threshold of 0.9 are deemed *cue responsive*. Excited neurons satisfy $\hat{p}(\tilde{\mu}^{(n)} > 0 \mid \boldsymbol{Y}) > 0.5$, while inhibited neurons satisfy $\hat{p}(\tilde{\mu}^{(n)} < 0 \mid \boldsymbol{Y}) > 0.5$.

ACC Versus BLA Rate Change Trial Analysis

To identify neurons that are involved in the observational learning of fear, we must find cue responsive neurons whose responses change significantly after trial 16, the trial in which the cue begins to be paired with the shock. The exact trial in which such a significant change occurs is known as the *rate change trial* ${r_0}^{(n)}$. The rate change trial ${r_0}^{(n)}$ is a random variable and varies for different neurons n. However, the versatility of particle filtering methods within the DPnSSM algorithm allows us to simply sample r_0 as an additional parameter in the inference algorithm.

In the previous analysis, we identified $N = 59$ cue responsive neurons. We reindex these neurons $n = 1, \ldots, N$ and ignore the others. We collapse the neuronal rasters across time to create time series $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$ over trials. More specifically, we let $y_r^{(n)} = \sum_{t=1}^{T} \Delta N_{t,r}^{(n)}$ for all $r = 1, \ldots, R$ and all n. Once again, we adopt the Poisson state-space model. With respect to the DPnSSM, the full generative model is, as follows:

$$
\begin{aligned}
z^{(1)}, \ldots, z^{(N)}, K &\sim \text{CRP}(\alpha, N), && && (66)\\
\boldsymbol{\theta}^{(k)} = [\mu^{(k)}, {r_0}^{(k)}] &\sim G, && 1 \leq k \leq K,\\
x_1^{(n)} \mid z^{(n)} = k, \mu^{(k)}, {r_0}^{(k)} &\sim \mathcal{N}(x_0^{(n)} + \mu^{(k)} \cdot \mathbb{I}({r_0}^{(k)} = 16), \psi_0), && 1 \leq n \leq N,\\
x_t^{(n)} \mid x_{t-1}^{(n)}, z^{(n)} = k, \mu^{(k)}, {r_0}^{(k)} &\sim \mathcal{N}(x_{t-1}^{(n)} + \mu^{(k)} \cdot \mathbb{I}({r_0}^{(k)} = r), \psi), && 16 < r \leq R, 1 \leq n \leq N,\\
y_t^{(n)} \mid x_t^{(n)} &\sim \text{Poisson}(\exp x_t^{(n)}), && 16 \leq r \leq R, 1 \leq n \leq N,
\end{aligned}
$$

where $\mathbb{I}(\cdot)$ is the indicator function. Again, we run Algorithm 10 for full Bayesian inference. For each neuron n, the initial state $x_0^{(n)} = \log(1/15 \cdot \sum_{r=1}^{15} y_r^{(n)})$ is computed from the trials prior to the shock modulation. We set the base distribution for the parameters $G[\mu^{(k)}, {r_0}^{(k)}] = [\mathcal{N}(0, 2), \text{Categorical}(16, \ldots, 35)]$. This is done with the expectation that the rate change trial will occur in one of the first 20 out of 30 trials after shock delivery begins; Allsop et al. (2018) estimated the latest rate change trial as occurring 17 trials after shock delivery.

The MH proposal distribution $r(\boldsymbol{\theta}' \mid \boldsymbol{\theta})$ has two components for this two-parameter problem. The response parameter $\mu' \sim \mathcal{N}(\mu, 0.25)$, as before. Since the rate change trial r_0 is a discrete variable, we adopt a categorical proposal distribution in which $r_0' = r_0 - 1$ with probability $1/3$, $r_0' = r_0$ with probability $1/3$, and $r_0' = r_0 + 1$ with probability $1/3$. All other hyperparameters are the same as in the previous analysis, in which we identified cue responsive neurons.

From the output of Algorithm 10, we collect samples $(Z^{\langle B+1 \rangle}, \boldsymbol{\Theta}^{\langle B+1 \rangle}), \ldots, (Z^{\langle I \rangle}, \boldsymbol{\Theta}^{\langle I \rangle})$. We are now interested in identifying *learning neurons*, i.e. cue responsive neurons that also exhibit significant changes after trial 16. These neurons encode a response to the cue that is modified when the cue is paired with the shock, thereby suggesting that they are involved in observational learning. A sensible way to find these neurons is simply to repeat the exercise of Equation (65). We define cue responsive neuron n as a *learning neuron* if it satisfies the following condition: Either $\hat{p}(\tilde{\mu}^{(n)} > 0 \mid \boldsymbol{Y}) \geq 0.9$ or $\hat{p}(\tilde{\mu}^{(n)} < 0 \mid \boldsymbol{Y}) \geq 0.9$, where $\tilde{\mu}^{(n)}$ is now defined for the generative model of Equation (66). A corresponding visualization of can be found in Figure 15. From the $N = 59$ cue responsive neurons, we

find 44 learning neurons, with 31 from the ACC and 13 from the BLA.

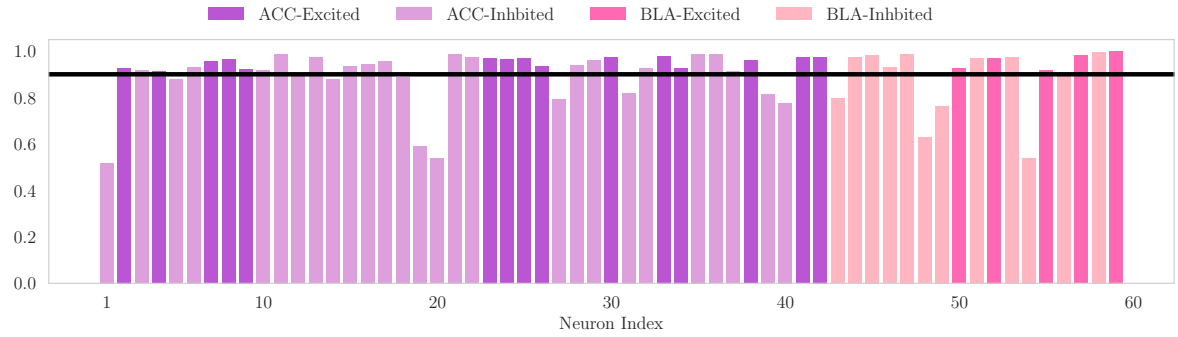

Figure 15: A bar plot of $\max\{\hat{p}(\tilde{\mu}^{(n)} > 0 \,|\, \boldsymbol{Y}), \hat{p}(\tilde{\mu}^{(n)} < 0 \,|\, \boldsymbol{Y})\}$ for cue responsive neuron $n = 1, \ldots, N$ in the context of Equation (66). Neurons that exhibit this quantity being greater than a threshold of 0.9 are deemed *learning neurons*. Excited neurons satisfy $\hat{p}(\tilde{\mu}^{(n)} > 0 \,|\, \boldsymbol{Y}) > 0.5$, while inhibited neurons satisfy $\hat{p}(\tilde{\mu}^{(n)} < 0 \,|\, \boldsymbol{Y}) > 0.5$.

A key finding in Allsop et al. (2018) is that the rate change trial for learning neurons is smaller in the ACC than in the BLA. For any neuron n, we can estimate its mean rate change trial $\tilde{r}_0^{(n)}$ as

$$\hat{\tilde{r}}_0^{(n)} = \frac{1}{I - B} \sum_{i=B+1}^{I} \mathbb{I}(r_0^{(k_i)\langle i \rangle}), \quad \text{where } k_i = z^{(n)\langle i \rangle}, \qquad 1 \leq n \leq N. \tag{67}$$

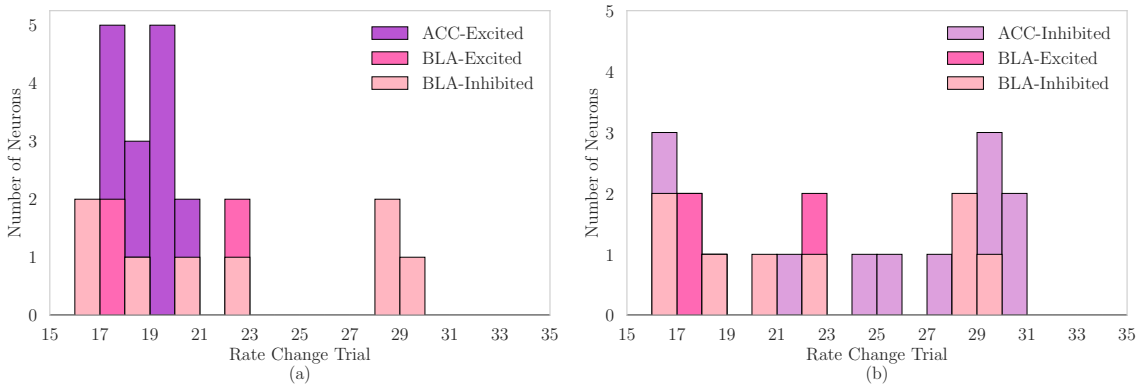

Figure 16: A comparison of the mean rate change trial for (a) excited ACC vs. BLA neurons and (b) inhibited ACC vs. BLA neurons.

In Figure 16a, we present a histogram of the mean rate change trial of excited ACC neurons vs. BLA neurons. This figure confirms the finding of Allsop et al. (2018), as the

average BLA rate change trial occurs after the average excited ACC rate change trial. It further supports the hypothesis that the ACC may transmit information to the BLA during fear conditioning. On the other hand, inhibited ACC neurons (Figure 16b) seem to exhibit much more variance in terms of the rate change trial. This suggests a more complicated relationship between these inhibited ACC neurons and the BLA neurons, which may be worth exploring in a future neuroscience study. We provide this section as an example of how the DPnSSM may be used to uncover findings that drive the formulation of new neuroscience hypotheses.

5 Conclusion

This manuscript introduced a new model-based method for clustering time series. The proposed approach combined two existing classes of generative models – mixture models and state-space models – to create a mixture of state-space models for time series exhibiting nonlinear dynamics in particular. We also developed two inference algorithms, a Monte Carlo expectation-maximization algorithm for maximum a posteriori inference and a Markov Chain Monte Carlo algorithm for full Bayesian inference. We applied these algorithms to problems in computational neuroscience, analyzing prototypical examples of nonlinear time series in this domain. The clustering results produced by the algorithm seemed to be consistent with expectations for both simulations and real-world experiments.

In comparison to other approaches for clustering time series, this model-based approach has certain advantages. The state-space framework is quite flexible, easily accommodating time series with different lengths and missing values. In addition, although this paper only considers examples in which each latent state and each observed value is one-dimensional, our model and inference algorithms can be extended to cases in which observed time series and/or latent states have multiple dimensions. For example, as demonstrated in Section 5.2 of Heng et al. (2017), controlled sequential Monte Carlo, the main workhorse of the algorithm, scales well with a 64-dimensional time series model. This suggests that our proposed clustering approach with particle filtering is scalable to applications involving multivariate time series.

However, one potential disadvantage of model-based clustering methods is that they are usually much slower to execute than feature-based or shape-based methods. Both EM and MCMC sampling are iterative processes that are much more complex than simply computing a distance metric between time series. In addition, although the state-space framework is flexible, we typically need some domain knowledge in order to choose an appropriate type of state-space model, along with cluster parameters, for the mixture framework. Furthermore, the inference algorithm cannot necessarily be treated as a black-box approach, because the specific type of state-space model (e.g. non-exponential family) may require us to modify the optimal method for policy selection in controlled sequential

Monte Carlo. Thus, as with many clustering methods, the utility of our model-based approach will depend on the specific application at hand.

In future work, one avenue of interest is finding ways to speed up the inference algorithms. For the MCMC case, this could involve running multiple chains in parallel. When doing so, it would be important to make sure that each chain returns an unbiased approximation of the posterior distribution; thus, we are interested in applying recent advances in unbiased Markov chain Monte Carlo (Jacob et al., 2017) to this work.

Another interesting idea is to explore methods in variational inference, the other dominant methodology for posterior inference in generative models. Such an approach would likely combine previously developed variational methods for inference in state-space models (Krishnan et al., 2017) and Dirichlet process mixtures (Blei et al., 2006) .

Furthermore, we are always interested in exploring other settings in which our model can be applied. Immediate examples that come to mind include sports (e.g. using binomial state-space models to cluster NBA players based on their field goal percentages) and finance (e.g. using stochastic volatility models to cluster various companies based on their asset returns).

6 Acknowledgments

First and foremost, I would like to thank my advisors, Demba and Pierre, to whom I am deeply appreciative for continually inspiring me to do research. I am very grateful for their invaluable time, guidance, and moral support throughout the past few years. In addition, I would like to thank my co-authors and collaborators – Diana Zhang for her mentorship, Jeremy Heng for many enlightening discussions, and Stephen Allsop and Kay Tye for providing meaningful neuroscience perspectives. Also, thank you to Demba, Finale, and Pierre for taking the time to read and evaluate this manuscript.

Finally, I would like to thank all of my friends and family for always supporting me, especially throughout the duration of this project. Thank you to my father and my mother for sparking my initial interest in mathematical reasoning and always being available in times of need. Thank you to my sister Katrina for her undying optimism. And last but not least, thank you to Sarah Wang for her ever-present support and companionship.

Bibliography

Aghabozorgi, S., Shirkhorshidi, A. S., and Wah, T. Y. (2015). Time-series clustering–a decade review. *Information Systems*, 53:16–38.

Allsop, S. A., Wichmann, R., Mills, F., Burgos-Robles, A., Chang, C.-J., Felix-Ortiz, A. C., Vienne, A., Beyeler, A., Izadmehr, E. M., Glober, G., et al. (2018). Corticoamygdala transfer of socially derived information gates observational learning. *Cell*, 173(6):1329–1342.

Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342.

Andrzejak, R. G., Lehnertz, K., Mormann, F., Rieke, C., David, P., and Elger, C. E. (2001). Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E*, 64(6):061907.

Bauwens, L. and Rombouts, J. (2007). Bayesian clustering of many GARCH models. *Econometric Reviews*, 26(2-4):365–386.

Blei, D. M., Jordan, M. I., et al. (2006). Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143.

Boyden, E. S., Zhang, F., Bamberg, E., Nagel, G., and Deisseroth, K. (2005). Millisecond-timescale, genetically targeted optical control of neural activity. *Nature neuroscience*, 8(9):1263.

Brown, E. N., Kass, R. E., and Mitra, P. P. (2004). Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nature Neuroscience*, 7(5):456.

Caffo, B. S., Jank, W., and Jones, G. L. (2005). Ascent-based Monte Carlo expectation–maximization. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):235–251.

Carter, C. K. and Kohn, R. (1994). On gibbs sampling for state space models. *Biometrika*, 81(3):541–553.

Celeux, G. and Soromenho, G. (1996). An entropy criterion for assessing the number of clusters in a mixture model. *Journal of classification*, 13(2):195–212.

Chen, L. and Ng, R. (2004). On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 792–803. VLDB Endowment.

Chiappa, S. and Barber, D. (2007). Output grouping using Dirichlet mixtures of linear Gaussian state-space models. In *Image and Signal Processing and Analysis, 2007. ISPA 2007. 5th International Symposium on*, pages 446–451. IEEE.

Dahl, D. B. (2006). Model-based clustering for expression data via a Dirichlet process mixture model. *Bayesian Inference for Gene Expression and Proteomics*, 201:218.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Doucet, A., De Freitas, N., and Gordon, N. (2001). An introduction to sequential Monte Carlo methods. In *Sequential Monte Carlo Methods in Practice*, pages 3–14. Springer.

Doucet, A. and Johansen, A. M. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3.

Doya, K., Ishii, S., Pouget, A., and Rao, R. P. (2007). *Bayesian brain: Probabilistic approaches to neural coding*. MIT press.

Durbin, J. and Koopman, S. J. (2012). *Time Series Analysis by State Space Methods*, volume 38. Oxford University Press.

Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. (1994). *Fast subsequence matching in time-series databases*, volume 23. ACM.

Ferguson, T. S. (1973). A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230.

Fu, T.-c., Chung, F.-l., Ng, V., and Luk, R. (2001). Pattern discovery from stock time series using self-organizing maps. In *Workshop Notes of KDD2001 Workshop on Temporal Data Mining*, pages 26–29.

Gao, Y., Busing, L., Shenoy, K. V., and Cunningham, J. P. (2015). High-dimensional neural spike train analysis with generalized count linear dynamical systems. In *Advances in neural information processing systems*, pages 2044–2052.

Geman, S. and Geman, D. (1987). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. In *Readings in computer vision*, pages 564–584. Elsevier.

Ghahramani, Z. and Hinton, G. E. (1996). Parameter estimation for linear dynamical systems. Technical report, Technical Report CRG-TR-96-2, University of Totronto, Dept. of Computer Science.

Ghahramani, Z. and Roweis, S. T. (1999). Learning nonlinear dynamical systems using an EM algorithm. In *Advances in neural information processing systems*, pages 431–437.

Golay, X., Kollias, S., Stoll, G., Meier, D., Valavanis, A., and Boesiger, P. (1998). A new correlation-based fuzzy logic clustering algorithm for fmri. *Magnetic Resonance in Medicine*, 40(2):249–260.

Gordon, N. J., Salmond, D. J., and Smith, A. F. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113. IET.

Guarniero, P., Johansen, A. M., and Lee, A. (2017). The iterated auxiliary particle filter. *Journal of the American Statistical Association*, 112(520):1636–1647.

Gullo, F., Ponti, G., Tagarelli, A., Tradigo, G., and Veltri, P. (2012). A time series approach for clustering mass spectrometry data. *Journal of Computational Science*, 3(5):344–355.

Guo, C., Jia, H., and Zhang, N. (2008). Time series clustering based on ica for stock data analysis. In *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4. IEEE.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications.

Haykin, S. (2004). *Kalman filtering and neural networks*, volume 47. John Wiley & Sons.

Heng, J., Bishop, A. N., Deligiannidis, G., and Doucet, A. (2017). Controlled sequential Monte Carlo. *arXiv preprint arXiv:1708.08396*.

Humphries, M. D. (2011). Spike-train communities: finding groups of similar spike trains. *Journal of Neuroscience*, 31(6):2321–2336.

Inoue, L. Y., Neira, M., Nelson, C., Gleave, M., and Etzioni, R. (2006). Cluster-based network model for time-course gene expression data. *Biostatistics*, 8(3):507–525.

Jacob, P. E., O'Leary, J., and Atchadé, Y. F. (2017). Unbiased markov chain monte carlo with couplings. *arXiv preprint arXiv:1708.03625*.

Ji, M., Xie, F., and Ping, Y. (2013). A dynamic fuzzy cluster algorithm for time series. In *Abstract and Applied Analysis*, volume 2013. Hindawi.

Kakizawa, Y., Shumway, R. H., and Taniguchi, M. (1998). Discrimination and clustering for multivariate time series. *Journal of the American Statistical Association*, 93(441):328–340.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45.

Kantz, H. and Schreiber, T. (2004). *Nonlinear time series analysis*, volume 7. Cambridge university press.

Kim, C.-J., Nelson, C. R., et al. (1999). State-space models with regime switching: classical and gibbs-sampling approaches with applications. *MIT Press Books*, 1.

Kimura, T., Tokuda, T., Nakada, Y., Nokajima, T., Matsumoto, T., and Doucet, A. (2013). Expectation-maximization algorithms for inference in Dirichlet processes mixture. *Pattern Analysis and Applications*, 16(1):55–67.

Krishnan, R. G., Shalit, U., and Sontag, D. (2017). Structured inference networks for nonlinear state space models. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Kumar, M., Patel, N. R., and Woo, J. (2002). Clustering seasonality patterns in the presence of errors. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 557–563. ACM.

Liao, T., Bolt, B., Forester, J., Hailman, E., Hansen, C., Kaste, R., and O'May, J. (2002). Understanding and projecting the battle state. In *23rd Army Science Conference, Orlando, FL*, volume 25.

Liao, T. W. (2005). Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874.

Lin, A., Zhang, Y., Heng, J., Allsop, S. A., Tye, K. M., Jacob, P. E., and Ba, D. (2019). Clustering time series with nonlinear dynamics: a Bayesian non-parametric and particle-based approach. In *International Conference on Artificial Intelligence and Statistics*, page (to appear).

Linderman, S. W., Adams, R. P., and Pillow, J. W. (2015). Inferring structured connectivity from spike trains under negative-binomial generalized linear models. *Neuron*, 50(100):150.

McLachlan, G. J., Lee, S. X., and Rathnayake, S. I. (2000). Finite mixture models. *Annual Review of Statistics and Its Application*, (0).

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Middleton, L. (2014). Clustering time series: a Dirichlet process mixture of linear-Gaussian state-space models. Master's thesis, Oxford University, United Kingdom.

Miller, J. W. and Harrison, M. T. (2018). Mixture models with a prior on the number of components. *Journal of the American Statistical Association*, 113(521):340–356.

Möller-Levet, C. S., Klawonn, F., Cho, K.-H., and Wolkenhauer, O. (2003). Fuzzy clustering of short time-series and unevenly distributed sampling points. In *International Symposium on Intelligent Data Analysis*, pages 330–340. Springer.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Neal, R. M. (2000). Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265.

Niennattrakul, V. and Ratanamahatana, C. A. (2007). On clustering multimedia time series data using k-means and dynamic time warping. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pages 733–738. IEEE.

Nieto-Barajas, L. E. and Contreras-Cristán, A. (2014). A Bayesian nonparametric approach for time series clustering. *Bayesian Analysis*, 9(1):147–170.

Owsley, L. M., Atlas, L. E., and Bernard, G. D. (1997). Self-organizing feature maps and hidden markov models for machine-tool monitoring. *IEEE Transactions on Signal Processing*, 45(11):2787–2798.

Paninski, L., Pillow, J., and Lewi, J. (2007). Statistical models for neural encoding, decoding, and optimal stimulus design. *Progress in brain research*, 165:493–507.

Pillow, J. W., Shlens, J., Paninski, L., Sher, A., Litke, A. M., Chichilnisky, E., and Simoncelli, E. P. (2008). Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995.

Roick, T., Karlis, D., and McNicholas, P. D. (2019). Clustering discrete valued time series. *arXiv preprint arXiv:1901.09249*.

Saad, F. and Mansinghka, V. (2018). Temporally-reweighted chinese restaurant process mixtures for clustering, imputing, and forecasting multivariate time series. In *International Conference on Artificial Intelligence and Statistics*, pages 755–764.

Sakoe, H. (1971). Dynamic-programming approach to continuous speech recognition. In *1971 Proc. the International Congress of Acoustics, Budapest.*

Shaw, C. and King, G. (1992). Using cluster analysis to classify time series. *Physica D: Nonlinear Phenomena*, 58(1-4):288–298.

Shumway, R. H. (2003). Time-frequency clustering and discriminant analysis. *Statistics & probability letters*, 63(3):307–314.

Shumway, R. H. and Stoffer, D. S. (2017). *Time series analysis and its applications: with R examples*. Springer.

Smith, A. C. and Brown, E. N. (2003). Estimating a state-space model from point process observations. *Neural computation*, 15(5):965–991.

Swerling, P. (1958). *A proposed stagewise differential correction procedure for satellite tracking and prediciton*. Rand Corporation.

Temereanca, S., Brown, E. N., and Simons, D. J. (2008). Rapid changes in thalamic firing synchrony during repetitive whisker stimulation. *Journal of Neuroscience*, 28(44):11153–11164.

Van Wijk, J. J. and Van Selow, E. R. (1999). Cluster and calendar based visualization of time series data. In *Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis' 99)*, pages 4–9. IEEE.

Vere-Jones, D. (2003). *An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods*. Springer.

Vlachos, M., Gunopoulos, D., and Kollios, G. (2002). Discovering similar multidimensional trajectories. In *icde*, page 0673. IEEE.

Vlachos, M., Lin, J., Keogh, E., and Gunopulos, D. (2003). A wavelet-based anytime algorithm for k-means clustering of time series. In *In proc. workshop on clustering high dimensionality data and its applications*. Citeseer.

Wang, X., Smith, K., and Hyndman, R. (2006). Characteristic-based clustering for time series data. *Data mining and knowledge Discovery*, 13(3):335–364.

West, M., Harrison, P. J., and Migon, H. S. (1985). Dynamic generalized linear models and bayesian forecasting. *Journal of the American Statistical Association*, 80(389):73–83.

Wilpon, J. and Rabiner, L. (1985). A modified k-means clustering algorithm for use in isolated work recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(3):587–594.

Wu, C. J. et al. (1983). On the convergence properties of the em algorithm. *The Annals of statistics*, 11(1):95–103.

Xiong, Y. and Yeung, D.-Y. (2002). Mixtures of ARMA models for model-based time series clustering. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 717–720. IEEE.