# Clustering Time Series with Nonlinear Dynamics: A Bayesian Non-Parametric and Particle-Based Approach

**Alexander Lin**[1] **Yingzhuo Zhang**[1] **Jeremy Heng**[1] **Stephen A. Allsop**[2]
**Kay M. Tye**[3] **Pierre E. Jacob**[1] **Demba Ba**[1]
[1]Harvard University [2]MIT, Picower Institute [3]Salk Institute

## Abstract

We propose a general statistical framework for clustering multiple time series that exhibit nonlinear dynamics into an a-priori-unknown number of sub-groups. Our motivation comes from neuroscience, where an important problem is to identify, within a large assembly of neurons, subsets that respond similarly to a stimulus or contingency. Upon modeling the multiple time series as the output of a Dirichlet process mixture of nonlinear state-space models, we derive a Metropolis-within-Gibbs algorithm for full Bayesian inference that alternates between sampling cluster assignments and sampling parameter values that form the basis of the clustering. The Metropolis step is a PMMH iteration that requires an unbiased, low variance estimate of the likelihood function of a nonlinear state-space model, which we compute using the recently developed cSMC algorithm. We apply the framework to clustering time series acquired from the prefrontal cortex of mice in an experiment designed to characterize the neural underpinnings of fear.

## 1 INTRODUCTION

In a data set comprising hundreds to thousands of neuronal time series (Brown et al., 2004), the ability to automatically identify sub-groups of time series that respond similarly to an exogenous stimulus or contingency can provide insights into how neural computation is implemented at the level of groups of neurons.

Existing methods for clustering multiple time series can be classified broadly into feature-based approaches and model-based ones. The former extract a set of features from each time series, followed by clustering in feature space using standard algorithms, e.g. Humphries (2011). While simple to implement, feature-based approaches cannot be used to perform statistical inference on the parameters of a physical model by which the time series are generated.

Previous model-based approaches for clustering multiple time series typically employ Bayesian mixtures of time series models. Examples have included GARCH models (Bauwens and Rombouts, 2007), INAR models (Roick et al., 2019), and TRCRP models (Saad and Mansinghka, 2018).

State-space models are a well-known, flexible class of models for time series data (Durbin and Koopman, 2012). Many existing model-based approaches for clustering multiple time series use a mixture of linear Gaussian state-space models. Inoue et al. (2006) and Chiappa and Barber (2007) both consider the case of finite mixtures and use Gibbs sampling and variational-Bayes respectively for posterior inference. Nieto-Barajas et al. (2014) and Middleton (2014) use a Dirichlet Process (DP) mixture to infer the number of clusters, and Gibbs sampling for full posterior inference. In all cases, the linear-Gaussian assumption is crucial; 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.

We introduce a framework for clustering multiple time series that exhibit nonlinear dynamics into an a-priori-unknown number of clusters, each modeled as a nonlinear state-space model. We derive a Metropolis-within-Gibbs algorithm for inference in a Dirichlet process mixture of state-space models with linear-Gaussian states and binomial observations, a popular model in the analysis of neural spiking activity (Smith and Brown, 2003). The Metropolis step uses particle

Proceedings of the 22[nd] International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. 

marginal Metropolis Hastings (Andrieu et al., 2010), which requires likelihood estimates with small variance. We use controlled sequential Monte Carlo (Heng et al., 2017) to produce such estimates. We apply the framework to the clustering of 33 neural spiking time series acquired from the prefrontal cortex of mice in an experiment designed to characterize the neural underpinnings of fear. The framework produces a clustering of the neurons into groups that represent various degrees of neuronal signal modulation.

## 2 NONLINEAR TIME SERIES CLUSTERING MODEL

We begin by introducing, under a general framework, the Dirichlet Process nonlinear State-Space Mixture (DPnSSM) model for clustering multiple time series exhibiting nonlinear dynamics.

### 2.1 DPnSSM

Let $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$ be a set of observed time series in which each series $\boldsymbol{y}^{(n)} = y_1^{(n)}, \ldots, y_T^{(n)}$ is a vector of length $T$. Following the general framework of state-space models, we model $\boldsymbol{y}^{(n)}$ as the output of a latent, autoregressive process $\boldsymbol{x}^{(n)}$. In particular, for all $n = 1, \ldots, N$,

$$\begin{aligned} x_1^{(n)} \mid \tilde{\boldsymbol{\theta}}^{(n)} &\sim h(x_1^{(n)}; \tilde{\boldsymbol{\theta}}^{(n)}), \\ x_t^{(n)} \mid x_{t-1}^{(n)}, \tilde{\boldsymbol{\theta}}^{(n)} &\sim f(x_{t-1}^{(n)}, x_t^{(n)}; \tilde{\boldsymbol{\theta}}^{(n)}), \qquad 1 < t, \\ y_t^{(n)} \mid x_t^{(n)}, \tilde{\boldsymbol{\theta}}^{(n)} &\sim g(x_t^{(n)}, y_t^{(n)}; \tilde{\boldsymbol{\theta}}^{(n)}), \qquad 1 \le t, \end{aligned} \tag{1}$$

where $\tilde{\boldsymbol{\theta}}^{(n)}$ denotes a set of *hidden parameters* for series $n$, $f$ is some *state transition function*, $g$ is some *state-dependent likelihood*, and $h$ is some *initial prior*. Although this paper only considers examples in which each $x_t^{(n)}, y_t^{(n)} \in \mathbb{R}$, our model and inference algorithm are extendable to cases in which observed time series and/or latent states have multiple dimensions.

The hidden parameters $\tilde{\boldsymbol{\Theta}} = \{\tilde{\boldsymbol{\theta}}^{(1)}, \ldots, \tilde{\boldsymbol{\theta}}^{(N)}\}$ form the basis of clustering $\boldsymbol{Y}$; that is, if $\tilde{\boldsymbol{\theta}}^{(n)} = \tilde{\boldsymbol{\theta}}^{(n')}$, then series $n$ and $n'$ belong to the same cluster. However, in many applications, the number of clusters itself may be an unknown variable and therefore, we choose to model the parameters as coming from a distribution $Q$ that is sampled from a Dirichlet process (DP) with base distribution $G$ and inverse-variance parameter $\alpha$. Ferguson (1973) showed that $Q$ is almost surely discrete and that the number of distinct values within $N \to \infty$ draws from $Q$ is random. Thus, the DP serves as a prior for discrete distributions over elements in the support of $G$. The overall objective is to infer the joint distribution of $\tilde{\boldsymbol{\Theta}} \mid \boldsymbol{Y}, \alpha, G$.

The Chinese Restaurant Process (CRP) representation of the DP integrates out the intermediary distribution $Q$ used to generate $\tilde{\boldsymbol{\Theta}}$ (Neal, 2000). The CRP allows us to nicely separate the process of assigning a cluster (i.e. table) to each $\boldsymbol{y}^{(n)}$ from the process of choosing a hidden parameter (i.e. table value) for each cluster. This is similar to the finite mixture model, but we do not need to choose $K$, the number of clusters, a priori.

Under the CRP, we index the parameters by the cluster index $k$ instead of by the observation index $n$. Let $z^{(n)} \in \{1, \ldots, K\}$ denote the cluster identity of series $n$ and let $\boldsymbol{\theta}^{(k)}$ denote the hidden parameters for cluster $k$. We formally define the model as follows,

$$\begin{aligned} z^{(1)}, \ldots, z^{(N)}, K \mid \alpha &\sim \mathrm{CRP}(\alpha, N), \\ \boldsymbol{\theta}^{(k)} \mid G &\sim G, \qquad 1 \le k \le K, \end{aligned} \tag{2}$$

and for all $n = 1, \ldots, N$,

$$\begin{aligned} x_1^{(n)} \mid z^{(n)} = k, \boldsymbol{\theta}^{(k)} &\sim h(x_1^{(n)}; \boldsymbol{\theta}^{(k)}), \\ x_t^{(n)} \mid x_{t-1}^{(n)}, z^{(n)} = k, \boldsymbol{\theta}^{(k)} &\sim f(x_{t-1}^{(n)}, x_t^{(n)}; \boldsymbol{\theta}^{(k)}), \quad 1 < t, \\ y_t^{(n)} \mid x_t^{(n)}, z^{(n)} = k, \boldsymbol{\theta}^{(k)} &\sim g(x_t^{(n)}, y_t^{(n)}; \boldsymbol{\theta}^{(k)}), \quad 1 \le t, \end{aligned}$$

Let $Z = \{z^{(1)}, \ldots, z^{(N)}\}$, $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(K)}\}$. Figure 1 shows a graphical representation of the DPnSSM.

For the case of finite mixtures with $K$ clusters, we can model $Z$ as instead being drawn from a $K$-dimensional multinomial that is itself drawn from a $K$-dimensional Dirichlet distribution with parameter $\boldsymbol{\alpha} = \{\alpha^{(1)}, \ldots, \alpha^{(K)}\}$, where $\alpha^{(k)} > 0$ for all $k$.

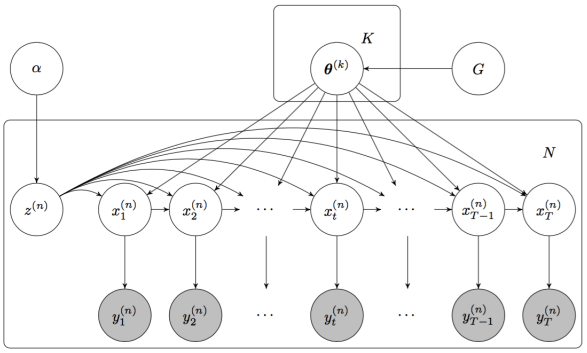


Figure 1: The graphical model representation of the DPnSSM. Observations are shown in grey, and states and parameters are shown in white. For simplicity, we omit the dependencies that are assumed in the DPnSSM between $\boldsymbol{Y}$ and $(Z, \boldsymbol{\Theta})$.

### 2.2 Point Process State-Space Model

While the DPnSSM is defined for generic nonlinear state-space models, in this paper we focus on a state-

space model commonly used for neural spike rasters.

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

$$\lambda^{(n)}(\tau \mid H_\tau^{(n)}) = \lim_{\Delta \to 0} \frac{p\left(N_r^{(n)}(\tau + \Delta) - N_r^{(n)}(\tau) = 1 \mid H_\tau^{(n)}\right)}{\Delta}, \tag{3}$$

where $H_\tau^{(n)}$ is the spiking history of the point process for neuron $n$ up to time $\tau$. The discrete-time process obtained by sampling $dN_r^{(n)}(\tau)$ at a resolution of $\Delta$, with $T = \left\lfloor \frac{\mathcal{T}}{\Delta} \right\rfloor$, is denoted by $\{\Delta N_{t,r}^{(n)}\}_{t=1,r=1}^{T,R}$. Define $M = \max_{n,t,r} \Delta N_{t,r}^{(n)}$. Given $x_0^{(n)}$ and $\psi_0^{(n)}$, a popular approach is to encode a discrete-time representation of the CIF $\{\lambda_t^{(n)}\}_{t=1}^T$ within an autoregressive process that underlies a binomial state-space model with observations $\{\Delta N_{t,r}^{(n)}\}_{t=1,r=1}^{T,R}$ (Smith and Brown, 2003):

$$\begin{aligned}
x_1^{(n)} \mid \tilde{\mu}^{(n)} &\sim \mathcal{N}(x_0^{(n)} + \tilde{\mu}^{(n)}, \psi_0^{(n)}), && \\
x_t^{(n)} \mid x_{t-1}^{(n)}, \tilde{\psi}^{(n)} &\sim \mathcal{N}(x_{t-1}^{(n)}, \tilde{\psi}^{(n)}), && 1 < t, \\
p_t^{(n)} = \lambda_t^{(n)} \cdot \Delta = \sigma(x_t^{(n)}) &= \frac{\exp x_t^{(n)}}{1 + \exp x_t^{(n)}}, && 1 \le t, \\
y_t^{(n)} = \sum_{r=1}^{R} \Delta N_{t,r}^{(n)} &\sim \mathrm{Bin}\left(R \cdot M, p_t^{(n)}\right), && 1 \le t.
\end{aligned} \tag{4}$$

where $\tilde{\boldsymbol{\theta}}^{(n)} = [\tilde{\mu}^{(n)}, \log \tilde{\psi}^{(n)}]^\top$ are the parameters of interest. We can cluster the neuronal units by these parameters by assuming that they arise from the Dirichlet process mixture of Equation 2, in which $\boldsymbol{\theta}^{(k)} = [\mu^{(k)}, \log \psi^{(k)}]^\top = \tilde{\boldsymbol{\theta}}^{(n)}$ for all $n$ such that $z^{(n)} = k$.

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 function $f$ 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 the sustainability of this response $\psi^{(k)}$ .

# 3 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 1. Outputs are samples $(Z^{\langle i \rangle}, \boldsymbol{\Theta}^{\langle i \rangle})$ for iterations $i = 1, 2, \ldots, I$.

For any set $S = \{s^{(1)}, \ldots, s^{(J)}\}$, we use the notation $S^{(-j)} = S \setminus \{s^{(j)}\}$ to denote set $S$ without the $j$-th element.

## 3.1 Sampling Cluster Assignments

For a given time series $n \in \{1, \ldots, N\}$, we sample its cluster assignment from the distribution:

$$\begin{aligned}
p(z^{(n)} \mid Z^{(-n)}, \boldsymbol{\Theta}, \boldsymbol{Y}, \alpha, G) & \\
&\propto p(z^{(n)} \mid Z^{(-n)}, \boldsymbol{\Theta}, \alpha, G) \cdot p(\boldsymbol{Y} \mid Z, \boldsymbol{\Theta}, \alpha, G) \\
&\propto p(z^{(n)} \mid Z^{(-n)}, \alpha) \cdot p(\boldsymbol{y}^{(n)} \mid z^{(n)}, \boldsymbol{\Theta}, G).
\end{aligned} \tag{5}$$

Due to the CRP's exchangeability property, the first term in Equation (5) can be represented by the categorical distribution,

$$p(z^{(n)} = k) = \begin{cases} \dfrac{N^{(k)}}{N - 1 + \alpha}, & k = 1, \ldots, K', \\ \dfrac{\alpha/m}{N - 1 + \alpha}, & k = K' + 1, \ldots, K' + m, \end{cases} \tag{6}$$

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 $m \ge 1$ is some integer algorithmic parameter. For brevity, we drop the conditioning on $Z^{(-n)}$ and $\alpha$.

The second term in Equation 5 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 $k \in \{K' + 1, \ldots K' + m\}$. In the case of a linear-Gaussian state-space model, Middleton (2014) is able to evaluate this likelihood exactly using a Kalman filter. Since we are modeling $\boldsymbol{y}^{(n)}$ as the output of a nonlinear state-space model, we must use

particle methods to approximate this parameter likelihood. Though the standard bootstrap particle filter (BPF) (Doucet et al., 2001) offers a solution to this problem, it requires many particles to compute a low-variance estimate at the cost of increased computing time. To increase efficiency, we employ a recently proposed method, known as controlled sequential Monte Carlo (cSMC), that significantly reduces the variance of the likelihood estimators for a fixed computational cost (Heng et al., 2017). We outline the basic premise behind cSMC in Section 3.3.

For the finite mixture case with $K$ clusters, the only change in this step is that Equation 6 simply becomes $p(z^{(n)} = k \mid Z^{(-n)}, \boldsymbol{\alpha}) = N^{(k)}/(N - 1 + \alpha^{(k)})$ for all $k$.

### 3.2 Sampling Cluster Parameters

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

$$\begin{aligned} p(\boldsymbol{\theta}^{(k)} \mid \boldsymbol{\Theta}^{(-k)}, Z, \boldsymbol{Y}, \alpha, G) \\ \propto p(\boldsymbol{\theta}^{(k)} \mid \boldsymbol{\Theta}^{(-k)}, Z, \alpha, G) \cdot p(\boldsymbol{Y} \mid \boldsymbol{\Theta}, Z, \alpha, G) \\ \propto p(\boldsymbol{\theta}^{(k)} \mid G) \cdot \prod_{n \mid z^{(n)}=k} p(\boldsymbol{y}^{(n)} \mid \boldsymbol{\theta}^{(k)}). \end{aligned} \tag{7}$$

The first term of Equation 7 is the pdf 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.

To sample from the conditional posterior of parameters given cluster assignments, Middleton (2014) reintroduces 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 (PMMH) algorithm (Andrieu et al., 2010). To evaluate the second term of Equation 7 for PMMH, we once again choose to use cSMC (Section 3.3).

Note that for finite mixtures, this step of sampling cluster parameters is exactly the same in the inference algorithm.

### 3.3 Controlled Sequential Monte Carlo

Controlled SMC is based on the idea 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. More precisely, the algorithm introduces a collection of positive and bounded functions $\gamma = \{\gamma_1, \ldots, \gamma_T\}$, termed a policy, that alter the transition probabilities of the model in the following way,

$$\begin{aligned} h^{\gamma}(x_1; \boldsymbol{\theta}) &\propto h(x_1; \boldsymbol{\theta}) \cdot \gamma_1(x_1), \\ f_t^{\gamma}(x_{t-1}, x_t; \boldsymbol{\theta}) &\propto f(x_{t-1}, x_t; \boldsymbol{\theta}) \cdot \gamma_t(x_t), \quad 1 < t. \end{aligned} \tag{8}$$

To ensure that the likelihood associated with the modified model is the same as the original one, we introduce a modified version of the state-dependent likelihood $g$, denoted by $g_1^{\gamma}, \ldots, g_T^{\gamma}$. On the modified model defined by $h^{\gamma}, \{f_t^{\gamma}\}_{t=2}^{T}, \{g_t^{\gamma}\}_{t=1}^{T}$, we can run a bootstrap particle filter and compute the likelihood estimator:

$$\hat{p}^{\gamma}(\boldsymbol{y} \mid \boldsymbol{\theta}) = \prod_{t=1}^{\mathrm{T}} \left( \frac{1}{S} \sum_{s=1}^{S} g_t^{\gamma}(x_t^s, y_t; \boldsymbol{\theta}) \right), \tag{9}$$

where $S$ is the number of particles and $x_t^s$ is the $s$-th particle at time $t$. The policy $\gamma$ can be chosen so as to minimize the variance of the above likelihood estimator; the optimal policy minimizing that variance is denoted by $\gamma^*$.

When $h, f$ are Gaussian and $g$ is log-concave with respect to $x_t$ (such as in the point-process state-space model of Equation 4), we can justify the approximation of $\gamma^*$ with a series of Gaussian functions. This allows us to solve for $h^{\gamma}, \{f_t^{\gamma}\}_{t=2}^{T}, \{g_t^{\gamma}\}_{t=1}^{T}$ using an approximate backward recursion method that simply reduces to a sequence of constrained linear regressions. We provide a more rigorous treatment of the exact details in Appendix A.

Starting from an initial policy $\gamma^{\langle 0 \rangle}$, we can thus run a first bootstrap particle filter and obtain an approximation $\gamma^{\langle 1 \rangle}$ of $\gamma^*$. One can then iterate $L$ times to obtain refined policies, and consequently, lower variance estimators of the likelihood. Our empirical testing demonstrates that cSMC can significantly outperform the standard BPF in both precision and efficiency, while keeping $L$ very small. This justifies its use in the DPnSSM inference algorithm.

## 4 RESULTS

We investigate the efficacy of the DPnSSM model in clustering time series obtained from simulated and real neural spike rasters.

**Algorithm 1** `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** $(Z^{\langle 1\rangle}, \boldsymbol{\Theta}^{\langle 1\rangle}), \ldots, (Z^{\langle I\rangle}, \boldsymbol{\Theta}^{\langle I\rangle})$

### 4.1 Selecting Clusters

The output of Algorithm 1 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 choose to follow the work of Dahl (2006) and Nieto-Barajas et al. (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 $N$x$N$ co-occurrence matrix $\boldsymbol{\Omega}^{\langle i\rangle}$ in which,

$$\boldsymbol{\Omega}^{\langle i\rangle}_{(n,n')} = \begin{cases} 1, & z^{(n)} = z^{(n')} \mid z^{(n)}, z^{(n')} \in Z^{\langle i\rangle}, \\ 0, & z^{(n)} \neq z^{(n')} \mid z^{(n)}, z^{(n')} \in Z^{\langle i\rangle}. \end{cases} \quad (10)$$

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,

$$\boldsymbol{\Theta}^* = \frac{1}{J} \sum_{j=1}^{J} \boldsymbol{\Theta}^{\langle i_j^*\rangle} \approx \mathbb{E}[\boldsymbol{\Theta} \mid Z = Z^*]. \quad (11)$$

This averaging must be proceeded 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.

### 4.2 Simulated Neural Spiking Data

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.

#### 4.2.1 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 $\mathcal{T} = 1500$. The resolution of the data is $\Delta = 5$ ms and we observe neuron $n$ firing $\Delta N_{t,r}^{(n)}$ 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 $\hat{\lambda}^{(n)} \sim \text{Uniform}(10, 15)$ 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 each of the following five types:

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

Within a time span of $\Delta$, the maximum possible number of firings for any neuron is $M = 5$; that is,

$\Delta N_{t,r}^{(n)} \leq 5$ for all $n, t, r$. Thus, following the point-process state-space model of Equation 4 – which assumes i.i.d. trials – we simulate,

$$y_t^{(n)} = \sum_{r=1}^{R=45} \Delta N_{t,r}^{(n)} \sim \text{Bin}(R \cdot M = 225, p_t^{(n)}), \quad (12)$$

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.

### 4.2.2 Modeling

In modeling these simulated data as coming from the DPnSSM, we employ the generative process specified by Equation 4; that is,

$$\begin{aligned} x_1^{(n)} \mid \mu^{(k)} &\sim \mathcal{N}(x_0^{(n)} + \mu^{(k)}, \psi_0), && (13) \\ x_t^{(n)} \mid x_{t-1}^{(n)}, \psi^{(k)} &\sim \mathcal{N}(x_{t-1}^{(n)}, \psi^{(k)}), && 1 < t \leq T, \\ y_t^{(n)} \mid x_t^{(n)} &\sim \text{Bin}(225, \sigma(x_t^{(n)})), && 1 \leq t \leq T, \end{aligned}$$

where cluster parameters are $\boldsymbol{\theta}^{(k)} = [\mu^{(k)}, \log \psi^{(k)}]^\top$.

The series are fed into Algorithm 1 with hyperparameter values $\alpha = 1$, $G = [\mathcal{N}(0, 2), \text{Unif}(-15, 0)]^\top$, 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 $(Z^{\langle 0 \rangle}, \boldsymbol{\Theta}^{\langle 0 \rangle}) = (\mathbf{1}, \boldsymbol{\theta}_0)$, 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 $\mathcal{N}(\boldsymbol{\theta}^{(k)}, 0.25 \cdot \mathcal{I})$ distribution, where $\mathcal{I}$ is the 2x2 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.

A heatmap of the resultant mean co-occurrence matrix $\boldsymbol{\Omega}$ (Equation 10) and the selected clustering $\boldsymbol{\Omega}^{\langle i^* \rangle}$ can be found in Figure 2. 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 1 summarizes the final cluster parameters $\boldsymbol{\Theta}^*$ computed using Equation 11. 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 seems to be 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 4.2.1 – e.g. $+1$ for $k \in \{1, 4\}$, $-1$ for $k \in \{2, 5\}$ and $\pm 0$ for $k = 3$. This provides an interpretation of the numerical value of $\mu^{(k)}$ in Equation 13. Indeed, given $\exp x_0^{(n)}, \exp x_1^{(n)} << 1$, as is often the case when modeling neurons, then the expected increase in the log of the firing probability due to the stimulus is:

$$\begin{aligned} \mathbb{E}\left[\log \frac{\sigma(x_1^{(n)})}{\sigma(x_0^{(n)})}\right] &= \mathbb{E}\left[\log \frac{\exp x_1^{(n)}/(1+\exp x_1^{(n)})}{\exp x_0^{(n)}/(1+\exp x_0^{(n)})}\right] \quad (14) \\ &\approx \mathbb{E}\left[\log \frac{\exp x_1^{(n)}}{\exp x_0^{(n)}}\right] = \mathbb{E}[x_1^{(n)} - x_0^{(n)}] = \mu^{(k)}. \end{aligned}$$

Looking at the values for $\log \psi^{*(k)}$ in Table 1, we can also see 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.

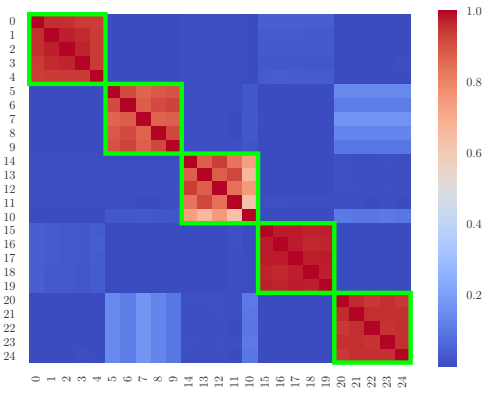


Figure 2: Heatmap of mean co-occurrence matrix $\boldsymbol{\Omega}$ for simulation results. Elements of the selected co-occurrence matrix $\boldsymbol{\Omega}^{\langle i^* \rangle}$ that are equal to 1 are enclosed in green squares. Each square corresponds to a distinct cluster.

## 4.3 Real Neural Spiking Data

In addition to simulations, we produce clusterings on real-world neural spiking data collected in a fear-conditioning experiment in mice designed to elucidate

Table 1: Parameters for simulation, where $\boldsymbol{\theta}^{*(k)} = [\mu^{*(k)}, \log \psi^{*(k)}]^\top$ for each $\boldsymbol{\theta}^{*(k)} \in \boldsymbol{\Theta}^*$.

| $k$ | $\mu^{*(k)}$ | $\log \psi^{*(k)}$ | **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 |

the nature of neural circuits that facilitate the associative learning of fear. The detailed experimental paradigm is described in Allsop et al. (2018). 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 is recorded from various neurons in the anterior cingulate cortex in the prefrontal cortex of the observer mouse. We apply our analysis to $N = 33$ 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 $\{\Delta N_{t,r}^{(n)}\}_{t=1,r=1}^{T,R}$, sampled at a resolution of $\Delta = 5$ ms with $T = 300$, where each $\Delta N_{t,r}^{(n)} \leq M = 5$.

We apply DPnSSM to identify various groups of responses in reaction to the auditory cue over time and over trials. 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.

#### 4.3.1 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 13). We also use all of the same hyperparameter values, as detailed in Section 4.2.2.

A heatmap of $\boldsymbol{\Omega}$ along with demarcations of $\boldsymbol{\Omega}^{\langle i^* \rangle}$ for this experiment can be found in Figure 3. Overall, five clusters are selected by the algorithm. Table 2 summarizes the chosen cluster parameters. Figure 8 shows two of the five clusters identified by the algorithm, namely those corresponding to $k = 1$ (Figure 8a) and $k = 5$ (Figure 8b). Figures for all other clusters can be found in Appendix B. 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 2 indicates that the algorithm has identified a sensible clustering of the neurons.

The algorithm is able to successfully differentiate various types of responses to the cue as well as the variability of the responses. 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 this data. In this case, the inference algorithm identifies five different clusters. We defer a scientific interpretation of this phenomenon to a later study.

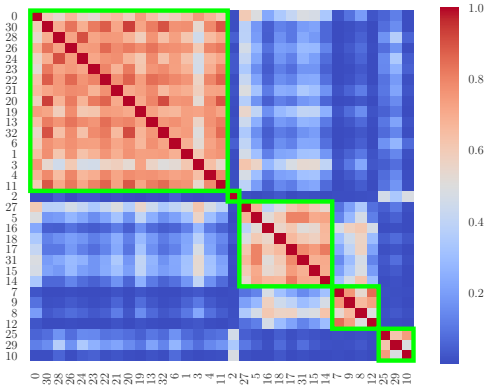


Figure 3: Heatmap of mean co-occurence matrix for cue data over time and selected clusters (green).

Table 2: Cluster parameters for cue data over time. $N^{*(k)}$ denotes how many neurons are in cluster $k$.

| $k$ | $\mu^{*(k)}$ | $\log \psi^{*(k)}$ | $N^{*(k)}$ |
|---|---|---|---|
| 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 |

#### 4.3.2 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 at the 16th trial over the 45 trials. 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 4.3.1). In this setting, each $y_r^{(n)} = \sum_{t=1}^{T} \Delta N_{t,r}^{(n)} \in \{0, 1, \ldots, 2000\}$ represents the

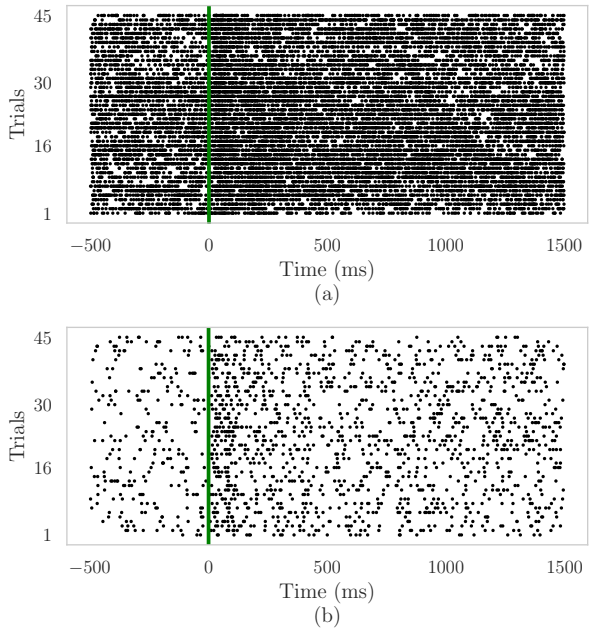


Figure 4: Overlaid raster plots of neuronal clusters with (a) moderately excited ($\mu^{*(1)} = 0.54$) and somewhat sustained ($\log \psi^{*(1)} = -6.27$) responses to the cue; and (b) more excited ($\mu^{*(5)} = 1.21$) and less sustained responses ($\log \psi^{*(5)} = -4.52$) to the cue. A black dot at $(\tau, r)$ indicates a spike from one of the neurons in the corresponding cluster at time $\tau$ during trial $r$. The vertical green line indicates cue onset.

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 framework:

$$\begin{aligned} x_{16}^{(n)} &\sim \mathcal{N}(x_0^{(n)} + \mu^{(k)}, \psi_0), && (15) \\ x_r^{(n)} \mid x_{r-1}^{(n)} &\sim \mathcal{N}(x_{r-1}^{(n)}, \psi^{(k)}), && 16 < r \leq R, \\ y_r^{(n)} \mid x_r^{(n)} &\sim \mathrm{Bin}(2000, \sigma(x_r^{(n)})), && 16 \leq r \leq R, \end{aligned}$$

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 4.2.2.

The corresponding heatmap, representative raster plots, and clustering results can be found in Figures 5, 6, and Table 3, 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. We will explore the implications of these findings to the neuroscience of observational learning of fear in future work.

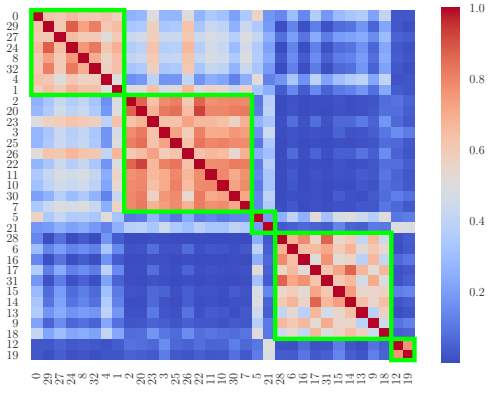


Figure 5: Heatmap of mean co-occurence matrix for cue data over trials and selected clusters (green).

Table 3: Cluster parameters for cue data over trials.

| $k$ | $\mu^{*(k)}$ | $\log \psi^{*(k)}$ | $N^{*(k)}$ |
|---|---|---|---|
| 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 |

### 4.4 Controlled SMC Versus BPF

Finally, we present some computational results on the advantages of using controlled sequential Monte Carlo over the bootstrap particle filter. Computing the parameter likelihood is a fundamental task in Algorithm 1. In each Gibbs iteration, we perform $O(N \cdot K)$ particle filter computations during the sampling of the cluster assignments and another $O(N)$ particle filter computations during the sampling of the cluster parameters. Thus, for both the efficiency and precision of the algorithm, it is necessary to find a fast way to compute low-variance estimates.

Figure 3 demonstrates the benefits of using cSMC over BPF for likelihood evaluation for a fixed computational cost. In some cases, cSMC estimates have variances that are several orders of magnitude lower than those produced by BPF. This is especially true for low values of the variability parameter $\log \psi$, which is crucial for this application since these are often the ones that maximize the parameter likelihood.

## 5 CONCLUSION

We proposed a general framework to cluster time series with nonlinear dynamics modeled by nonlinear state-space models. To the best of the authors' knowledge, this is the first Bayesian framework for clustering time series that exhibit nonlinear dynamics. The back-

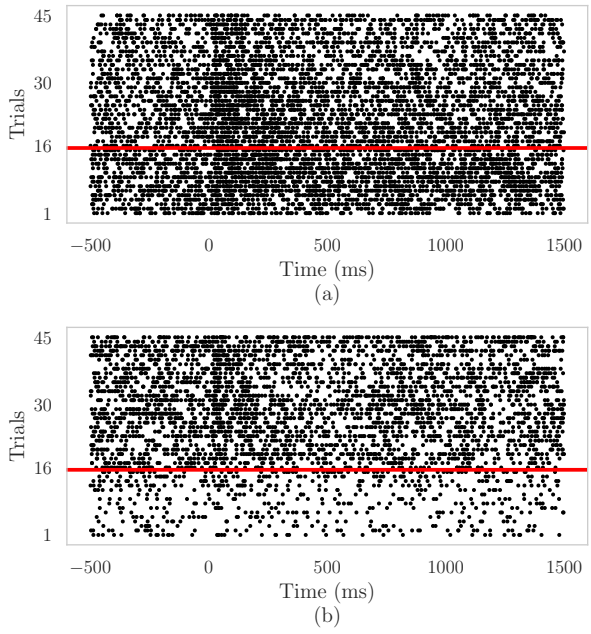


Figure 6: (a) Overlaid raster plots of neuronal clusters with (a) slightly inhibited, variable responses ($k = 2$) and (b) very excited, variable responses ($k = 4$). The red line marks the first trial with shock administration.

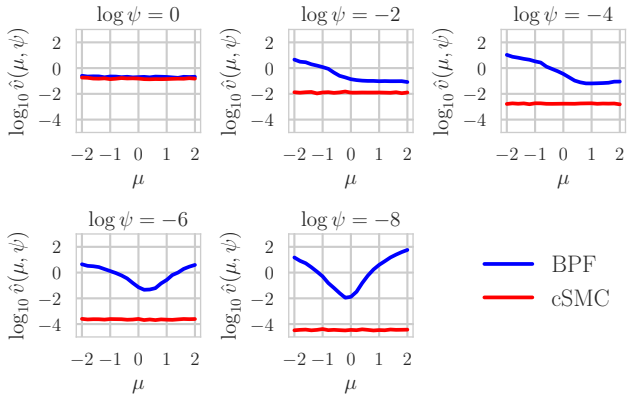


Figure 7: Results on running BPF (with $S = 1024$ particles) and cSMC (with $S = 64$ particles, $L = 3$ iterations) on computing the log parameter likelihood for a representative neuron's cue responses over time $\boldsymbol{y}$. Computational cost is fixed at $\approx 35$ ms per likelihood evaluation. For each particle filtering method, let $v(\mu, \psi) = \mathbb{V}\mathrm{ar}[\log p(\boldsymbol{y} \mid \boldsymbol{\theta})]$, where $\boldsymbol{\theta} = [\mu, \log \psi]^\top$. We plot an empirical estimation of $v$, i.e. $\hat{v}$, over different values of $(\mu, \psi)$ for the two methods. For each empirical variance data point, we use 500 estimations of $\log p(\boldsymbol{y} \mid \boldsymbol{\theta})$. As $\log \psi$ decreases, cSMC performs substantially better than BPF, especially for extreme values of $\mu$.

bone of the framework is the cSMC algorithm for low-variance evaluation of parameter likelihoods in nonlinear state-space models. We applied the framework to neural data in an experiment designed to elucidate the neural underpinnings of the observational learning of fear in mice. We were able to identify potential clusters of neurons that allow an observer to understand when the demonstrator is in distress.

In future work, we plan to perform detailed analyses of the data from these experiments (Allsop et al., 2018), and the implications of these analyses on the neuroscience of the observational learning of fear in mice. We will also explore applications of our model to data in other application domains such as sports and sleep research (St Hilaire et al., 2017), to name a few.

**Acknowledgements**

The authors thank Amazon Web Services (AWS) for the assistance of computational power in running experiments for this paper. Demba Ba thanks the Harvard Data Science Initiative. Pierre E. Jacob thanks the Harvard Data Science Initiative and the National Science Foundation (Grant DMS-1712872). Kay M. Tye thanks the McKnight Foundation, NIH (Grant R01-MH102441-01), and NCCIH (Pioneer Award DP1-AT009925).

**References**

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.

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

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.

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.

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.

Durbin, J. and Koopman, S. J. (2012). *Time series analysis by state space methods*, volume 38. Oxford University Press.

Ferguson, T. S. (1973). A bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230.

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

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.

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

Neal, R. M. (2000). Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265.

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

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.

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

St Hilaire, M. A., Rüger, M., Fratelli, F., Hull, J. T., Phillips, A. J., and Lockley, S. W. (2017). Modeling neurocognitive decline and recovery during repeated cycles of extended sleep and chronic sleep deficiency. *Sleep*, 40(1).

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

# Appendices

## A CONTROLLED SEQUENTIAL MONTE CARLO

A key step in sampling both the cluster assignments and the cluster parameters of Algorithm 1 is computing the parameter likelihood $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ for an observation vector $\boldsymbol{y} = y_1, \ldots, y_T$ and a given set of parameters $\boldsymbol{\theta}$.

Recall the state-space model formulation:

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

### A.1 Bootstrap Particle Filter

The bootstrap particle filter (BPF) of Doucet et al. (2001) 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

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

is an unbiased estimate of the parameter likelihood $p(\boldsymbol{y} \mid \boldsymbol{\theta})$. Algorithm 2 provides a review of this algorithm.

**Algorithm 2** `BootstrapParticleFilter`($\boldsymbol{y}$, $\boldsymbol{\theta}$, $f$, $g$, $h$)

1: **for** $s = 1, \ldots, S$ **do**
2: Sample $x_1^s \sim h(x_1; \boldsymbol{\theta})$ and weight $w_1^s = g(x_1^s, y_1; \boldsymbol{\theta})$.
3: **end for**
4: Normalize $\{w_1^s\}_{s=1}^S = \{w_1^s\}_{s=1}^S / \sum_{s=1}^S w_1^s$.
5: **for** $t = 2, \ldots, T$ **do**
6: **for** $s = 1, \ldots, S$ **do**
7: Resample ancestor index $a \sim \mathcal{C}\text{ategorical}(w_{t-1}^1, \ldots, w_{t-1}^S)$.
8: Sample $x_t^s \sim f(x_{t-1}^a, x_t; \boldsymbol{\theta})$ and weight $w_t^s = g(x_t^s, y_t; \boldsymbol{\theta})$.
9: **end for**
10: Normalize $\{w_t^s\}_{s=1}^S = \{w_t^s\}_{s=1}^S / \sum_{s=1}^S w_t^s$.
11: **end for**
12: **return** Particles $\{\{x_1^s\}_{s=1}^S, \ldots, \{x_T^s\}_{s=1}^S\}$

There are a variety of algorithms for the resampling step of Line 8. We use the systematic resampling method.

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 $\boldsymbol{y}$. 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. To remedy our problem, we follow the work of Heng et al. (2017) in using controlled sequential Monte Carlo (cSMC) as an alternative to the standard bootstrap particle filter.

### A.2 Twisted Sequential Monte Carlo

The basic idea of cSMC is to run several iterations of twisted sequential Monte Carlo, a process in which we redefine the model's state transition function $f$, initial prior $h$, and state-dependent likelihood $g$ in a way that allows the BPF to produce lower-variance estimates without changing the parameter likelihood $p(\boldsymbol{y} \mid \boldsymbol{\theta})$. See also Guarniero et al. (2017) for a different iterative approach. Using a *policy* $\gamma = \{\gamma_1, \ldots, \gamma_T\}$ in which each $\gamma_t$ is a

positive and bounded function, we define,

$$h^\gamma(x_1;\boldsymbol{\theta}) = \frac{h(x_1;\boldsymbol{\theta})\cdot\gamma_1(x_1)}{H^\gamma(\boldsymbol{\theta})},$$
$$f_t^\gamma(x_{t-1},x_t;\boldsymbol{\theta}) = \frac{f(x_{t-1},x_t;\boldsymbol{\theta})\cdot\gamma_t(x_t)}{F_t^\gamma(x_{t-1};\boldsymbol{\theta})}, \qquad 1<t\le T,$$

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 $h^\gamma$ and $f_t^\gamma$, 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 $g_1^\gamma,\dots,g_T^\gamma$ as functions that satisfy:

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

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

$$g_1^\gamma(x_1,y_1;\boldsymbol{\theta}) = \frac{H^\gamma(\boldsymbol{\theta})\cdot g(x_1,y_1;\boldsymbol{\theta})\cdot F_2^\gamma(x_1;\boldsymbol{\theta})}{\gamma_1(x_1)},$$
$$g_t^\gamma(x_t,y_t;\boldsymbol{\theta}) = \frac{g(x_t,y_t;\boldsymbol{\theta})\cdot F_{t+1}^\gamma(x_t;\boldsymbol{\theta})}{\gamma_t(x_t)}, \qquad 1<t<T,$$
$$g_T^\gamma(x_T,y_T;\boldsymbol{\theta}) = \frac{g(x_T,y_T;\boldsymbol{\theta})}{\gamma_T(x_T)}.$$

Thus, the parameter likelihood estimate of the twisted model is

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

The BPF is simply a degenerate case of twisted SMC in which $\gamma_t = 1$ for all $t$.

### A.3 Determining the Optimal Policy $\gamma^*$

The variance of the estimate $\hat{p}^\gamma$ comes from the state-dependent likelihood $g$. Thus, to minimize the variance, we would like $g_t^\gamma$ to be as uniform as possible with respect to $x_t$. Let the optimal policy be denoted $\gamma^*$. It follows that

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

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

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

which may be impossible to directly sample from if $f$ and $g$ form an intractable posterior (e.g. if $f$ is Gaussian and $g$ is binomial). In such a case, we must choose a suboptimal policy $\gamma$.

### A.4 Choosing a Policy $\gamma$ for the Neuroscience Application

Recall the point-process state-space model (Equation 4), in which we have

$$\begin{aligned} h(x_1;\boldsymbol{\theta}) &= \mathcal{N}(x_1 \mid x_0+\mu,\psi_0),\\ f(x_{t-1},x_t;\boldsymbol{\theta}) &= \mathcal{N}(x_t \mid x_{t-1},\psi),\\ g(x_t,y_t) &= \text{Binomial}\left(M\cdot R,\frac{\exp x_t}{1+\exp x_t}\right), \end{aligned}$$

where we define the parameters $\boldsymbol{\theta}=\{\mu,\log\psi\}$ and $x_0,\psi_0,M,R$ are supplied constants.

Here, we can show that $F^{\gamma^*}_{t+1}(x_t;\boldsymbol{\theta}) = \int f(x_t,x_{t+1};\boldsymbol{\theta})\gamma^*_{t+1}(x_{t+1})dx_{t+1}$ must be log-concave in $x_t$. This further implies that for all $t$, $\gamma^*_t(x_t) = g(x_t,y_t)\cdot F^{\gamma^*}_{t+1}(x_t;\boldsymbol{\theta})$ 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,

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

and thus, $f^\gamma_t(x_{t-1},x_t;\boldsymbol{\theta}) \propto f(x_{t-1},x_t;\boldsymbol{\theta})\cdot\gamma_t(x_t)$ 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 $\gamma_t$ and $\gamma^*_t$ in log-space. That is, given a set of samples $\{x^1_t,\ldots,x^S_t\}$ for the random variable $x_t$, we solve for:

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

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

$$\gamma^*_t(x_t) = g(x_t,y_t)\cdot F^\gamma_{t+1}(x_t;\boldsymbol{\theta}_t),$$

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

In the aforementioned least-squares optimization problem, there is one additional constraint that we must take into account. Recall that $f^\gamma_t(x_{t-1},x_t;\boldsymbol{\theta})\propto f(x_{t-1},x_t;\boldsymbol{\theta})\cdot\gamma_t(x_t)$ is a Gaussian pdf that we sample from. Therefore, we must ensure that the variance of this distribution is positive, which places a constraint on $\gamma_t$ 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 $h^\gamma$ and $f^\gamma_t$:

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

The corresponding normalizing terms for these densities are

$$\begin{aligned} H^\gamma(\boldsymbol{\theta}) &= \frac{1}{\sqrt{1+2a_1\psi_0}}\exp\left(\frac{\psi_0^{-1}\cdot(x_0+\mu)-(b_1)^2}{2(\psi_0^{-1}+2a_1)}-\frac{(x_0+\mu)^2}{2\psi_0}-c_1\right),\\ F^\gamma_t(x_{t-1};\boldsymbol{\theta}) &= \frac{1}{\sqrt{1+2a_t\psi}}\exp\left(\frac{\psi^{-1}\cdot x_{t-1}-(b_t)^2}{2(\psi^{-1}+2a_t)}-\frac{x^2_{t-1}}{2\psi}-c_t\right), \qquad 1<t\le T. \end{aligned}$$

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

$$a_1 > -\frac{1}{2\psi_0}, \qquad\qquad a_t > -\frac{1}{2\psi}, \quad 1 < t \leq T.$$

**Algorithm 3** `ControlledSMC`($\boldsymbol{y}$, $g$, $\mu$, $\psi$, $x_0$, $\psi_0$, $L$)

Define $f(x_{t-1}, x_t; \boldsymbol{\theta}) = \mathcal{N}(x_t \mid x_{t-1}, \psi)$ and $h(x_1; \boldsymbol{\theta}) = \mathcal{N}(x_1 \mid x_0 + \mu, \psi_0)$.
Define parameters $\boldsymbol{\theta} = \{\mu, \log \psi\}$.
Collect particles $\{x_1^s\}_{s=1}^S, \ldots, \{x_T^s\}_{s=1}^S$ from `BootstrapParticleFilter`($\boldsymbol{y}$, $\boldsymbol{\theta}$, $f$, $g$, $h$).
Initialize $\Gamma' = \{\Gamma'_1, \ldots, \Gamma'_T\}$ where $\Gamma'_t(x_t) = 1$ for all $t = 1, \ldots, T$.
Initialize $g_t^{\Gamma'}(x_t, y_t) = g(x_t, y_t)$ for all $t = 1, \ldots, T$.
Initialize $a_t^{\langle 0 \rangle} = 0, b_t^{\langle 0 \rangle} = 0, c_t^{\langle 0 \rangle} = 0$ for all $t = 1, \ldots, T$.
**for** $\ell = 1, \ldots, L$ **do**
  *// Approximate backward recursion to determine policy and associated functions*
  Define $\gamma_T^*(x_T) = g_T^{\Gamma'}(x_T, y_T)$.
  **for** $t = T, \ldots, 2$ **do**
    Solve $(a_t^{\langle \ell \rangle}, b_t^{\langle \ell \rangle}, c_t^{\langle \ell \rangle}) = \arg\min_{(a_t, b_t, c_t)} \sum_{s=1}^S \left[-(a_t(x_t^s)^2 + b_t(x_t^s) + c_t) - \log \gamma_t^*(x_t^s)\right]^2$ subject to $a_t > -1/(2\psi) - \sum_{\ell'=0}^{\ell-1} a_t^{\langle \ell' \rangle}$ using linear regression.
    Define new policy function $\gamma_t(x_t) = \exp(-a_t^{\langle \ell \rangle} x_t^2 - b_t^{\langle \ell \rangle} x_t - c_t^{\langle \ell \rangle})$.
    Define cumulative policy function $\Gamma_t(x_t) = \Gamma'_t(x_t) \cdot \gamma_t(x_t) = \exp(-A_t x_t^2 - B_t x_t - C_t)$ where $A_t = \sum_{\ell'=0}^{\ell} a_t^{\langle \ell' \rangle}$, $B_t = \sum_{\ell'=0}^{\ell} b_t^{\langle \ell' \rangle}$, and $C_t = \sum_{\ell'=0}^{\ell} c_t^{\langle \ell' \rangle}$.
    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 $(a_1^{\langle \ell \rangle}, b_1^{\langle \ell \rangle}, c_1^{\langle \ell \rangle}) = \arg\min_{(a_1, b_1, c_1)} \sum_{s=1}^S \left[-(a_1(x_1^s)^2 + b_1(x_1^s) + c_1) - \log \gamma_1^*(x_1^s)\right]^2$ subject to $a_1 > -1/(2\psi_0) - \sum_{\ell'=0}^{\ell-1} a_1^{\langle \ell' \rangle}$ using linear regression.
  Define new policy function $\gamma_1(x_1) = \exp(-a_1^{\langle \ell \rangle} x_1^2 - b_1^{\langle \ell \rangle} x_1 - c_1^{\langle \ell \rangle})$.
  Define cumulative policy function $\Gamma_1(x_1) = \Gamma'_1(x_1) \cdot \gamma_1(x_1) = \exp(-A_1 x_t^2 - B_1 x_t - C_1)$ where $A_1 = \sum_{\ell'=0}^{\ell} a_1^{\langle \ell' \rangle}$, $B_1 = \sum_{\ell'=0}^{\ell} b_1^{\langle \ell' \rangle}$, and $C_1 = \sum_{\ell'=0}^{\ell} c_1^{\langle \ell' \rangle}$.
  Define $\Gamma$-twisted initial prior $h^\Gamma(x_1; \boldsymbol{\theta})$ and $H^\Gamma(\boldsymbol{\theta})$.
  Define $g_1^\Gamma(x_1, y_1) = H^\Gamma(\boldsymbol{\theta}) \cdot g(x_1, y_1) \cdot F_2^\Gamma(x_1; \boldsymbol{\theta})/\Gamma_1(x_1)$.
  *// Forward bootstrap particle filter to sample particles and compute weights*
  **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 $a \sim \mathcal{C}\text{ategorical}(w_{t-1}^1, \ldots, w_{t-1}^S)$.
      Sample $x_t^s \sim f_t^\Gamma(x_{t-1}^a, x_t; \boldsymbol{\theta})$ 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** Likehood estimate $\hat{p}^\Gamma(\boldsymbol{y} \mid \boldsymbol{\theta})$.

### A.5 Full cSMC Algorithm

The full controlled sequential Monte Carlo algorithm 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 $\Gamma'$ and $\gamma$, we can define

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

We can see from these relationships that twisting the original model using $\Gamma'$ and then twisting the new model using $\gamma$ has the same effect as twisting the original model using a cumulative policy $\Gamma$ where each $\Gamma_t(x_t) = \Gamma'_t(x_t)\cdot\gamma_t(x_t)$. We state the full cSMC algorithm in Algorithm 3.

## B ADDITIONAL RASTER PLOTS FOR CUE DATA OVER TIME

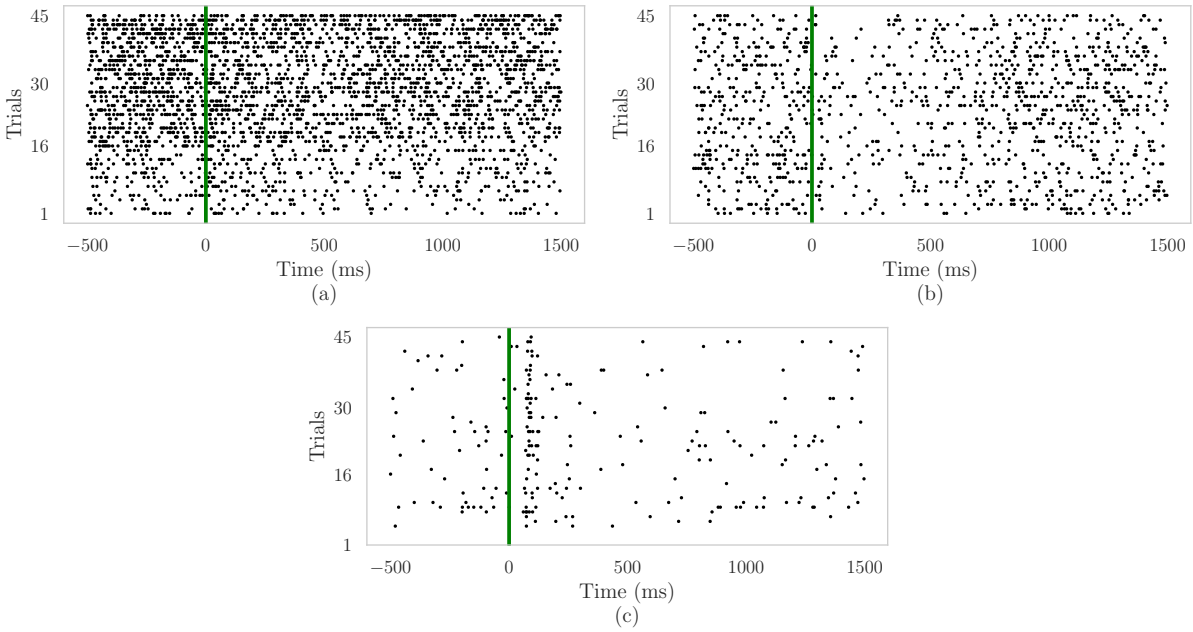


Figure 8: Overlaid raster plots of neuronal clusters found in Section 4.3.2 for (a) cluster $k = 2$ with eight neurons, (b) cluster $k = 4$ with four neurons, and (c) cluster $k = 2$ with a single neuron. A black dot at $(\tau, r)$ indicates a spike from one of the neurons in the corresponding cluster at time $\tau$ during trial $r$. The vertical green line indicates cue onset.