Bayesian Clustering of Neural Spiking Activity Using a Mixture of Dynamic Poisson Factor Analyzers

Ganchao Wei
Department of Statistics
University of Connecticut
ganchao.wei@uconn.edu

Ian. H. Stevenson
Department of Psychological Sciences
University of Connecticut
ian.stevenson@uconn.edu

Xiaojing Wang
Department of Statistics
University of Connecticut
xiaojing.wang@uconn.edu

Abstract

Modern neural recording techniques allow neuroscientists to observe the spiking activity of many neurons simultaneously. Although previous work has illustrated how activity within and between known populations of neurons can be summarized by low-dimensional latent vectors, in many cases what determines a unique population may be unclear. Neurons differ in their anatomical location, but also, in their cell types and response properties. Moreover, multiple distinct populations may not be well described by a single low-dimensional, linear representation. To tackle these challenges, we develop a clustering method based on a mixture of dynamic Poisson factor analyzers (mixDPFA) model, with the number of clusters treated as an unknown parameter. To do the analysis of DPFA model, we propose a novel Markov chain Monte Carlo (MCMC) algorithm to efficiently sample its posterior distribution. Validating our proposed MCMC algorithm with simulations, we find that it can accurately recover the true clustering and latent states and is insensitive to the initial cluster assignments. We then apply the proposed mixDPFA model to multi-region experimental recordings, where we find that the proposed method can identify novel, reliable clusters of neurons based on their activity, and may, thus, be a useful tool for neural data analysis.

1 Introduction

With modern high-density probes [Jun et al., 2017], neuroscientists can observe the spiking activity of many neurons from many different anatomical regions simultaneously. With these expanding capabilities, new methods to analyze neural data at the population-level and at the level of multiple populations become necessary. Several recent models have been developed to extract shared latent structures from simultaneous neural recordings, assuming that neural activity can be described through low-dimensional latent states. Many existing approaches are extensions of two basic models: the linear dynamical system (LDS) model [Macke et al., 2011] and a Gaussian process factor analysis (GPFA) model [Yu et al., 2009]. The LDS model is built on the state-space model and assumes latent factors evolve with linear dynamics. On the other hand, GPFA models the latent vectors by non-parametric Gaussian processes. However, in both cases, the observation model is generalized linear. Several variants of these models have been implemented to analyze multiple neural populations and their interactions [Semedo et al., 2019, Glaser et al., 2020]. However, in many cases, the total number of distinct populations and which neurons belong to a population is unclear.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Neurons in different anatomical locations may interact with each other or receive common input from unobserved brain areas, sharing the same latent structure. On the other hand, neurons of different cell-types within the same brain area may be better described by distinct latent structures. From a functional point of view, neither the anatomical location nor cell type (Fig. 1A) indicates which neurons should be grouped into the same populations. The incorrect population assignments can lead to biased and inconsistent inference on the latent structure [Ventura, 2009]. If we instead ignore multi-population structure and treat all neurons as a single population, then using linear model based methods may not describe their activity well, especially when the input is non-homogeneous. Besides, nonlinear models such as deep learning [Pandarinath et al., 2018, Whiteway et al., 2019] and Gaussian processes [Wu et al., 2017] have been developed, but these models do not explicitly distinguish among distinct populations of neurons.

Motivated by the mixture of (Gaussian) factor analyzers (MFA,Arminger et al. 1999, Ghahramani and Hinton 1996, Fokoué and Titterington 2003), which describes globally nonlinear data by combining a number of local factor analyzers, here we group neurons based on the latent factors (Fig. 1B). A similar idea was previously implemented using a mixture of Poisson linear dynamical system (PLDS) model (mixPLDS, Buesing et al. 2014). The mixPLDS model infers the subpopulations and latent factors using deterministic variational inference Wainwright and Jordan [2008], Jordan et al. [1999], Emtiyaz Khan et al. [2013] and the model parameters are estimated by Expectation Maximization (EM). Unlike MFA, the mixPLDS can capture temporal dependencies of neural activity as well as interactions between clusters over time. However, there are several limitations for mixPLDS: 1) it requires we predetermine the number of clusters, and 2) the clustering results are often sensitive to the initial cluster assignment.

Here we cluster the neurons by a mixture of dynamic Poisson factor analyzers (mixDPFA). The DPFA model takes the advantages of both Poisson factor analysis (FA) and PLDS and includes both a population baseline and baselines for individual neurons. The number of clusters is treated as an unknown parameter in the mixDPFA, and the posteriors are sampled using Markov Chain Monte Carlo (MCMC). To sample high dimensional latent factors, we approximate the full conditional distribution of the latent state by a Gaussian, which is similar to results by sampling from exact full conditional distribution. To improve mixing in the cluster assignments, we marginalize the loading out for clustering by Poisson-Gamma conjugacy. We also discuss the constraints necessity for successful sampling of the proposed models. After validating the proposed model with simulated data, we apply it to analyze multi-region experimental recordings from behaving mice: the Visual Coding - Neuropixels Dataset from the Allen Institute for Brain Science. Overall, the proposed method provides a way to efficiently cluster neurons into populations based on their activity.

2 Methods

Here we introduce a mixture of dynamic Poisson factor analyzers (mixDPFA) to cluster neurons based on multi-population latent structure. The number of mixture components is treated as an unknown parameter and the posteriors are sampled by MCMC. In this section, we first provide the single population DPFA for a given cluster. Then, we introduce a prior on the number of clusters and describe how we use the mixture of finite mixture model (MFM) to efficiently sample the posterior of the mixDPFA.

2.1 Dynamic Poisson Factor Analyzer

Denote the observed spike count of neuron $i \in \{1, \ldots, N\}$ at time bin $t \in \{1, \ldots, T\}$ as y_{it} (a non-negative integer), and let $\boldsymbol{y}_i = (y_{i1}, \ldots, y_{iT})'$. Further, let z_i be the cluster indicator of neuron i. Motivated by the nature of neural activity and the former PLDS model [Macke et al., 2011], we propose a new Poisson FA model by adding individual baselines δ_i. The proposed model is a combination of PLDS and Poisson FA, which includes both population baseline and individual baseline. Assume neuron i belongs to the j-th cluster (i.e., $z_i = j$), and its spiking activity is independently Poisson distributed, conditional on the low-dimensional latent state $\boldsymbol{x}_t^{(j)} \in \mathbb{R}^{p_j}$ and population baseline $\mu_t^{(j)}$ as follows:

$$
\begin{aligned}
y_{it} &\sim Poi(\lambda_{it}), \\
\log \lambda_{it} &= \delta_i + \mu_t^{(j)} + \boldsymbol{c}_i' \boldsymbol{x}_t^{(j)},
\end{aligned}
$$

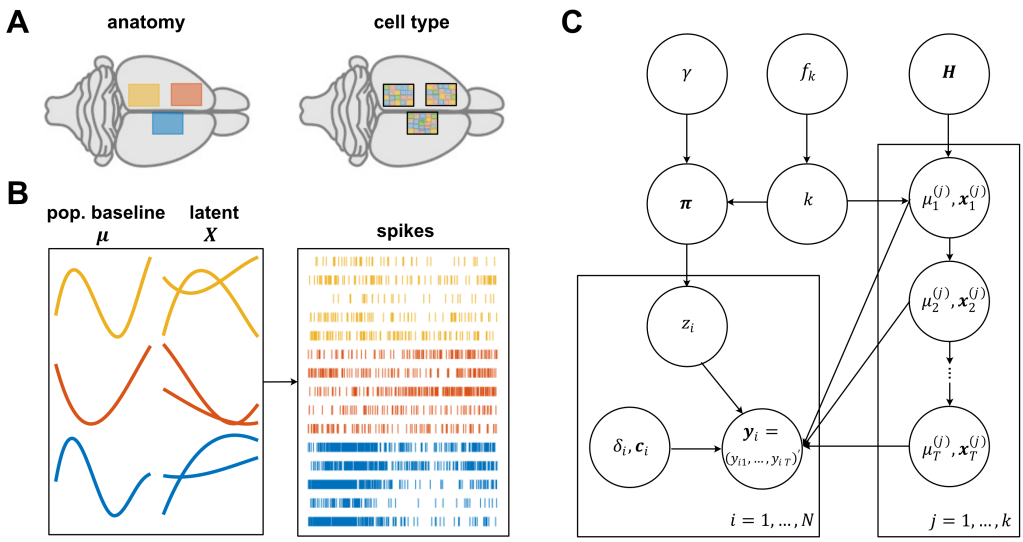

Figure 1: **Model overview. A.** There are multiple potential ways to define neural populations. For instance, populations could be defined by anatomical regions (left) or by cell types (right). Since the same latent structure could be shared across anatomical sites and cell types, a useful alternative may be define populations based on neural activity directly. **B.** The main goal for the proposed method is to cluster neurons according to their activity and extract functional grouping structure, based on spike train observations. The activity of each neuron is determined by a low dimensional latent state, specific to that neuron's cluster assignment (e.g. yellow, red, blue). **C.** Graphical representation of the mixture of finite mixtures (MFM) of dynamic Poisson factor analyzers (DPFA) generative model. Here the cluster number is treated as a random variable. The population baseline ($\mu_t^{(j)}$) and the latent factor ($\boldsymbol{x}_t^{(j)}$) for each cluster is generated by linear dynamics, with a Gaussian noise.

with $\boldsymbol{c}_i \sim \mathcal{N}_{p_j}(\boldsymbol{0}, \boldsymbol{I}_{p_j})$. The neuron-specific baseline δ_i is a constant across time for the ith neuron and unrelated to the cluster assignment. For simplicity, we assume the dimension of latent factors (states) is the same for all clusters, s.t. $p_j = p$, however, our method easily extends to the situation when p_js differs across clusters (see Discussion). Further, we assume the population baseline $\mu_t^{(j)}$ and the latent state $\boldsymbol{x}_t^{(j)}$ evolve linearly over time with Gaussian noise as following

$$\begin{aligned} \mu_{t+1}^{(j)} &= \quad g^{(j)} + h^{(j)}\mu_t^{(j)} + \epsilon_t^{(j)}, \\ \boldsymbol{x}_{t+1}^{(j)} &= \quad \boldsymbol{b}^{(j)} + \boldsymbol{A}^{(j)}\boldsymbol{x}_t^{(j)} + \boldsymbol{\eta}_t^{(j)}, \end{aligned}$$

where $\epsilon_t^{(j)} \sim \mathcal{N}(0, \sigma^{2(j)})$ and $\boldsymbol{\eta}_t^{(j)} \sim \mathcal{N}_p(\boldsymbol{0}, \boldsymbol{Q}^{(j)})$.

If we denote $\boldsymbol{\lambda}_i = (\lambda_{i1}, \ldots, \lambda_{iT})'$, $\boldsymbol{\mu}^{(j)} = (\mu_1^{(j)}, \ldots, \mu_T^{(j)})'$ and $\boldsymbol{X}^{(j)} = (\boldsymbol{x}_1^{(j)}, \ldots, \boldsymbol{x}_T^{(j)})'$, the proposed model can be rewritten as

$$\begin{aligned} \boldsymbol{y}_i &\sim Poi(\boldsymbol{\lambda}_i), \\ \log \boldsymbol{\lambda}_i &= \delta_i \boldsymbol{1}_T + \boldsymbol{\mu}^{(j)} + \boldsymbol{X}^{(j)}\boldsymbol{c}_i. \end{aligned} \tag{1}$$

Generally, a factor model is consistent only when $T/N \to 0$ [Johnstone and Lu, 2009], but this is often not the case for most neural spike data. However, when we assume linear dynamics on $\boldsymbol{\mu}^{(j)}$ and $\boldsymbol{X}^{(j)}$, it resolves the consistency issue. As known in a FA model, when $p > 1$, the model is only identifiable up to orthogonal rotation on $\boldsymbol{X}^{(j)}$, with $\boldsymbol{c}_i \sim N(\boldsymbol{0}, \boldsymbol{I}_p)$. With including an individual baseline $\delta_i \boldsymbol{1}_T$ in our proposed DPFA model (1), it further makes the model invariant to translation of $\boldsymbol{\mu}^{(j)}$ and $\boldsymbol{X}^{(j)}$. That means if $\boldsymbol{\mu}^{(j)}$ and $\boldsymbol{X}^{(j)}$ is a set solution, then $\boldsymbol{\mu}^{(j)} + a\boldsymbol{1}_T$ and $\boldsymbol{X}^{(j)}\boldsymbol{U} + \boldsymbol{1}_T \otimes \boldsymbol{m}'$ also satisfy the model, for any a, $\boldsymbol{m}$ and orthogonal matrix $\boldsymbol{U}$. Thus, to make the model identifiable, we need to add several constraints. Although the clustering is invariant to orthogonal rotation, how we put constraints on translation will influence the cluster assignments. Here we assume $\boldsymbol{A}^{(j)}$ and

$\boldsymbol{Q}^{(j)}$ are diagonal [Peña and Poncela, 2004, Lopes et al., 2008], and, to encourage clustering based on the trajectories of latent factors, we set $\sum_{t=1}^{T} \mu_t^{(j)} = 0$ and $\sum_{t=1}^{T} \boldsymbol{x}_t^{(j)} = \boldsymbol{0}$. See Section 5 for more discussions about the choice of these constraints.

Given the parameters of the j-th cluster $\boldsymbol{\theta}^{(j)} = \left\{\boldsymbol{\mu}^{(j)}, \boldsymbol{X}^{(j)}, h^{(j)}, g^{(j)}, \sigma^{2(j)}, \boldsymbol{A}^{(j)}, \boldsymbol{b}^{(j)}, \boldsymbol{Q}^{(j)}\right\}$, the spike counts of neuron i are generated by the dynamic Poisson factor analyzer (DPFA) model as $[\boldsymbol{y}_i \mid z_i = j] \sim DPFA(\delta_i, \boldsymbol{c}_i, \boldsymbol{\theta}^{(z_i)})$. To faciliate the Bayesian computation, we have to impose priors $\boldsymbol{H}$ on $\boldsymbol{\theta}^{(j)}$, see more details of prior settings in Appendix A.1.

2.2 Clustering by Mixture of Finite Mixtures Model

When the population labels z_is are unknown, we cluster the neurons by a mixture of DPFA (mixDPFA). Since the number of neural populations is finite but unknown, we need to put priors on it. To make the Bayesian computation more efficient, we utilize the idea from the mixture of finite mixtures (MFM, Miller and Harrison 2018) model, by assigning the priors for the clusters in the following way:

$$\begin{aligned}
k &\sim f_k, && f_k \text{ is a p.m.f. on} \{1, 2, \ldots\}, \\
\boldsymbol{\pi} = (\pi_1, \ldots, \pi_k) &\sim Dir_k(\gamma, \ldots, \gamma) && \text{given } k, \\
z_1, \ldots, z_N &\overset{i.i.d.}{\sim} \boldsymbol{\pi} && \text{given } \boldsymbol{\pi}, \\
\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(k)} &\overset{i.i.d.}{\sim} \boldsymbol{H} && \text{given } k, \\
\boldsymbol{y}_i = (y_{i1}, \ldots, y_{iT})' &\sim \mathrm{DPFA}(\delta_i, \boldsymbol{c}_i, \boldsymbol{\theta}^{(z_i)}) && \text{given } \delta_i, \boldsymbol{c}_i, \boldsymbol{\theta}^{(z_i)}, z_i, \forall i = 1, \ldots, N,
\end{aligned} \tag{2}$$

where $\mathrm{p.m.f}$ denotes the probability mass function. By using the MFM, we can integrate the field knowledge about the number of neural populations into our analysis. In the analysis of this paper, we assume k follows a geometric distribution, i.e., $k \sim Geometric(\alpha)$ with its density defined as $f_k(k|\alpha) = (1-\alpha)^{k-1}\alpha$ for $k = 1, 2, \ldots$, and let $\gamma = 1$. The complete generative model is summarized in a graphical form shown in Fig. 1C.

2.3 Inference

Here the posteriors of the proposed mixDPFA model are sampled by an MCMC algorithm (see Appendix A.1). In each iteration, we sample the model parameters assuming the known cluster indices at first, and then sample the cluster indices given the model parameters. When sampling the (labeled) model parameters, the latent state $\boldsymbol{X}^{(j)}$ and population baseline $\boldsymbol{\mu}^{(j)}$ have no closed-form full conditional distributions. Although we could sample the posterior by particle MCMC directly, convergence may be too slow for clustering. Here, we approximate the full conditional distribution for $\boldsymbol{X}^{(j)}$ and $\boldsymbol{\mu}^{(j)}$ by a Gaussian distribution (a Laplace approximation) and generate samples according to this approximation. This Laplace approximation is widely used for EM [Macke et al., 2011] and variational inference [Glaser et al., 2020] with PLDS models and their variants, and Gaussian approximation of the intractable full conditional distribution of latent effects has also been used in Bayesian mixed effects binomial regression, where Berman et al. 2022 found that the approximation provided reasonable estimation accuracy with substantial computational speedups. We, thus, use a global Laplace approximation that can be efficiently computed in $\mathcal{O}(T)$ [Paninski et al., 2010]. To help convergence, sampling on (labeled) model parameters is repeated several times before updating the cluster indices. Approximating the intractable full conditional with a Laplace approximation also makes computation of the proposed mixDPFA more efficient. However, to assess the accuracy of the approximation we also compare our approach to directly sampling from the exact posterior of the model. We develop a Pólya-Gamma (PG) data augmentation approach [Windle et al., 2013, Linderman et al., 2017, 2016, **?**] with an additional Metropolis-Hastings (MH) step [**?**Hastings, 1970] to sample exactly from the full conditional of $\boldsymbol{X}^{(j)}$ and $\boldsymbol{\mu}^{(j)}$. We find that the proposed method using a Laplace apprixmation is faster but performs similarly as sampling from the exact joint posterior (Fig. 2, Fig. 5, Appendix A.2, and Table 1)).

Once we update the latent state $\boldsymbol{X}^{(j)}$ and population baseline $\boldsymbol{\mu}^{(j)}$, the cluster index is then sampled by the analogy of partition-based algorithm in Dirichlet process mixtures (DPM, Neal 2000). See details in Miller and Harrison 2018 and the Appendix (A.1). When doing the clustering, we need to evaluate the likelihood for neurons under each cluster. Although we can sample $\boldsymbol{c}_i$ directly and evaluate the full likelihood as in MCMC for Gaussian MFA (data-augmentation/ imputation-posterior

algorithm, Fokoué and Titterington 2003), the chain has poor mixing and stops after a few iterations, because of the high dimensionality. The heavy dependency on the starting point when fitting the mixture of PLDS (mixPLDS, Buesing et al. 2014) model may suggest a similar problem. To resolve this, we evaluate the marginal likelihood by integrating out the neuron-specific $\boldsymbol{c}_i$, i.e., the marginal likelihood of neuron i in cluster j is computed by

$$M_{\boldsymbol{\theta}^{(j)}}(\boldsymbol{y}_i) = P(\boldsymbol{y}_i|\boldsymbol{\theta}^{(j)}, \delta_i) = \int P(\boldsymbol{y}_i|\boldsymbol{\theta}^{(j)}, \delta_i, \boldsymbol{c}_i)P(\boldsymbol{c}_i)\, d\boldsymbol{c}_i. \tag{3}$$

However, this marginal likelihood has no closed form. Though we may evaluate it by a Laplace approximation, but iterating over all potential clusters for each neuron is computationally intensive. To make faster clustering, we approximate the marginal likelihood by utilizing a Poisson-Gamma conjugacy. This approach has been previously utilized to approximate posteriors [El-Sayyad, 1973] and predictive distributions [Chan and Vasconcelos, 2009]. In our situation, since $\boldsymbol{c}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_p)$, we have $\lambda_{it} = \exp(\delta_i + \mu_t^{(j)} + \boldsymbol{c}_i'\boldsymbol{x}_t^{(j)}) \sim lognormal(\delta_i + \mu_t^{(j)}, \boldsymbol{x}_t'^{(j)}\boldsymbol{x}_t^{(j)})$, and then we can approximate this lognormal distribution by a gamma distribution, i.e., assume λ_{it} follows $Gamma(a_{it}, b_{it})$ with $a_{it} = (\boldsymbol{x}_t'^{(j)}\boldsymbol{x}_t^{(j)})^{-1}$ and $b_{it} = \boldsymbol{x}_t'^{(j)}\boldsymbol{x}_t^{(j)} \cdot e^{\delta_i + \mu_t^{(j)}}$. Then, by the conjugate property with Poisson and Gamma random variables, we have

$$P(y_{it}|\boldsymbol{\theta}^{(j)}, \delta_i) = \int P(y_{it}|\lambda_{it})P(\lambda_{it})\, d\lambda_{it} \approx NB(y_{it}|\nu_{it}, p_{it}),$$

with $\nu_{it} = a_{it}$ and $p_{it} = 1/(1 + b_{it})$. Further, noticing that we have the conditional independence assumption for $P(\boldsymbol{y}_i|\boldsymbol{\theta}^{(j)}, \delta_i)$, that is $P(\boldsymbol{y}_i|\boldsymbol{\theta}^{(j)}, \delta_i) = \prod_{t=1}^{T} P(y_{it}|\boldsymbol{\theta}^{(j)}, \delta_i)$, we then have a closed-form for Equation (3).

Another possible idea is to approximate the log-likelihood by second-order polynomials, with coefficients determined by Chebyshev polynomial approximation [Keeley et al., 2020]. However, we find that this approximation doesn't work well in practice when spike counts have a wide range. The model is implemented in MATLAB and the code is available at `https://github.com/weigcdsb/MFM_DPFA_clean`.

3 Simulations

To validate and illustrate the proposed clustering method, we simulate neural data directly from the generative model (1) . The labels for each neuron are assumed known and fixed at first to check convergence and model identifiability. We then infer the labels to evaluate clustering performance. All experiments in this paper were performed using a 3.40 GHz processor with 16 GB of RAM.

3.1 Labeled data

We first simulate 10 clusters with 5 neurons in each, with recording length $T = 1000$ and $p = 2$ dimensional latent factors for each cluster. Individual baselines are generated by $\delta_i \sim N(0, 0.5^2)$, and the loading for the latent states are generated by $\boldsymbol{c}_i \sim N(\mathbf{0}, \boldsymbol{I}_2)$. The population baseline $\boldsymbol{\mu}^{(j)}$ and latent vector $\boldsymbol{X}^{(j)}$ are generated by the spline interpolation on 10 to 30 evenly spaced knots. The simulations are conducted 50 times with different seeds. Here we show results for one simulation, and the performance for the rest is similar.

Since the labels are known, each whole simulation is equivalent to 10 independent simulations, with 5 neurons in each. Running MCMC for 10,000 iterations, we find that the log-likelihood per spike converges rapidly for individual clusters and overall (Fig.2A). Trace plots (Fig.2B) of the (Frobenius) norms for linear dynamics samples $(h^{(j)}, g^{(j)}, \sigma^{2(j)}, \boldsymbol{A}^{(j)}, \boldsymbol{b}^{(j)}, \boldsymbol{Q}^{(j)})$ show rapid mixing and convergence for each DPFA. The fitted mean firing rate (mean response) (Fig. 2C), $\boldsymbol{\mu}^{(j)}$ and $\boldsymbol{X}^{(j)}$ (Fig. 2D) match the ground truth well. The convergence is fast, especially in terms of mean response and population baseline $\boldsymbol{\mu}^{(j)}$. Together, these results demonstrate the identifiability of the DPFA with appropriate constraints.

We then compare the 10-cluster model with the simplified model ignoring the clustering structure (1-cluster model). The p for 1-cluster model is 14, chosen by 5-fold speckled cross-validation described in [Williams et al., 2020]. To evaluate the fitting performance of these two models, we

hold out 1/2 of the data and compare the held-out log-likelihood per spike. The distribution training log-likelihood is evaluated by averaging over the samples in short chains (e.g. iteration 50 to 100), which is justified by the observed fast convergence. The same procedure is replicated for 50 times. In this case, ignoring the clustering structure leads to a worse performance (Fig 2E), and the single population analysis cannot describe the data as accurately, since the input is non-homogeneous and the data is global nonlinear.

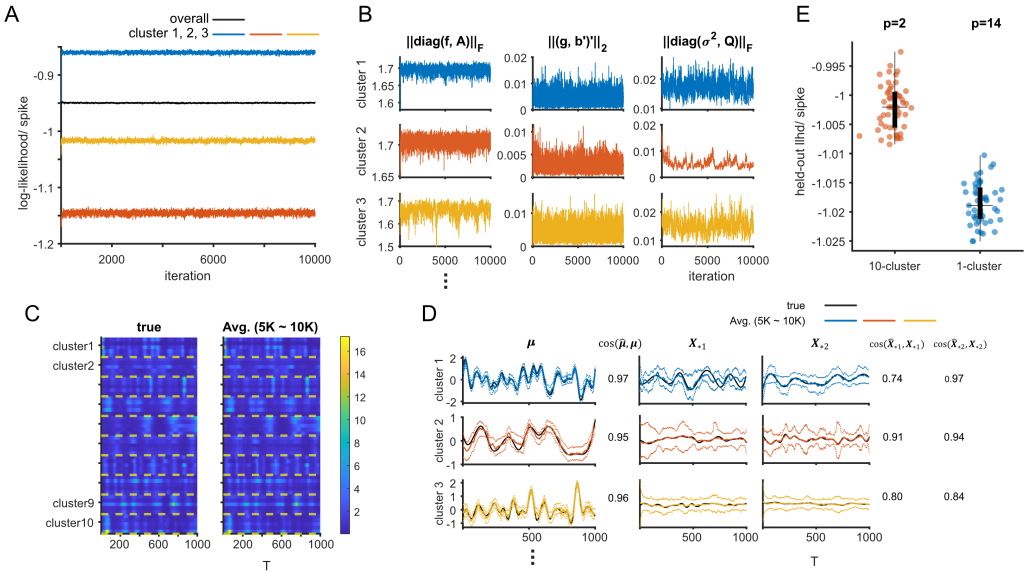

Figure 2: **Bayesian inference with labeled data** Here we simulate 10 clusters and assess convergence and mixing of for the DPFA. **A.** Traceplot of the log-likelihood per spike for all neurons and the first 3 clusters. **B.** The traceplot of (Frobenius) norms of linear dynamics of $\boldsymbol{\mu}^{(j)}$ and $\boldsymbol{X}^{(j)}$ for each cluster (showing the first 3). **C.** The true and the fitted mean firing rate, showing the averages over samples from iteration 5000 to 10000. **D.** The true (black) and the fitted (colored) population baseline and latent factor. The $\boldsymbol{X}_{*l}$ denotes the l-th latent factor (i.e. the l-th column of $\boldsymbol{X}$). The dashed lines show the 95% highest posterior density (HPD) interval. The cosine (the "overlap") between true values and posterior means shown besides. **E.** Comparison of the held-out likelihood per spike when fitting to 1/2 of the data: 1) 10-cluster model where each cluster has $p = 2$ (true value), and 2) 1-cluster model where a single DPFA describes all neurons, with $p = 14$ selected by 5-fold speckled cross-validation. Dots denote results from individual short (iterations 50-100), independent chains.

3.2 Clustering

Using the same simulation, we now infer the cluster labels. The latent factor dimension was first optimized with $p = 2$ selected by 5-fold speckled CV on small chains (100 iterations). We then compare three chains fitting with all data: two unique chains initialized using a single cluster and one chain initialized with $N = 50$ clusters (i.e. all clusters are singletons). Traceplots (Fig. 3A) of training log-likelihood and number of clusters show that all chains converge. When fit to the full data or only half, the number of clusters converges to 10, although the prior over the number of clusters is $K \sim Geometric(0.2)$. When the recording length is sufficiently long, the likelihood will dominate, and the number of clusters will not be much affected by the prior setting. The true mean firing rate for each neuron (Fig. 3B) can be well recovered, even with half data held out. To evaluate cluster membership, here we show a similarity matrix where the entry (i, l) is the posterior probability that data points i and l belong to the same cluster. The clustering results for all 4 chains recover the true clusters, no matter what the the starting assignment is (Fig. 3C), which suggest the convergence of MCMC. The overall performance of the full model, where cluster membership is inferred alongside the latent states, is similar to the case when cluster labels are known and substantially higher than the 1-cluster model (Fig. 3D and E).

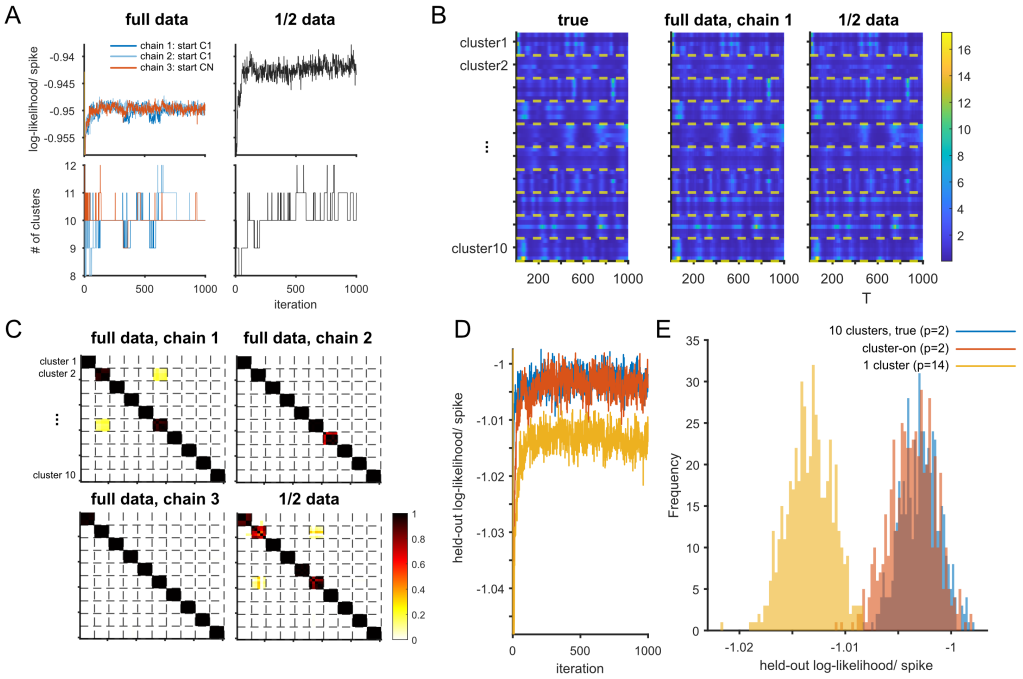

Figure 3: **Bayesian clustering** The same simulation setting as in Fig. 2, but inferring cluster labels from spike observations alone. **A.** The trace plots of training log-likelihood per spike and the number of clusters. The model is fitted by using all and half data as training with three chains shown for full data (initialized with a single cluster, C1 or $N = 50$ clusters, CN). **B.** The true and fitted mean firing rate, averaging from iteration 500 to 1000. **C.** The posterior similarity matrix for all chains (rows and columns ordered according to the ground truth). **D.** Traceplot of the held-out (1/2) log-likelihood for 1) the 10-cluster model with labels known, 2) the full model estimating labels and optimizing p, and 3) the 1-cluster, single population model with $p = 14$ chosen by CV. **E.** Held-out log-likelihood per spike for each model (samples for iteration 500 to 1000).

4 Multi-region neural spike recordings

We then apply the proposed clustering method to the Allen Institute Visual Coding Neuropixels dataset. The dataset contains spiking activity from hundreds of neurons from multiple brain regions of an awake mouse. See detailed data description in [Siegle et al., 2021]. Here we investigate the clustering structure of neurons from four anatomical sites (83 neurons): 1) hippocampal CA1 (24 neurons), 2) dorsal part of the lateral geniculate complex (LGd, 36 neurons), 3) lateral posterior nucleus of the thalamus (LP, 12 neurons) and 4) primary visual cortex (VISp, 11 neurons). And we analyze responses to 20s epochs during three visual stimuli: drifting gratings, spontaneous activity, and natural movies. Only neurons with rates > 1Hz within the selected epochs are included (72% of 115 neurons) and we analyze data with 40ms bins. We use a $Geometric(0.33)$ prior over the number of clusters, such that $p(k \leq 4) = 0.8$.

In responses to drifting gratings, the 5-fold speckled CV log-likelihood is optimized with $p = 2$, and, as in the simulations, the log-likelihood and number of clusters show rapid convergence and mixing (Fig 4A and B). Low firing rates and short recording lengths tend to cause confusions in clustering, reflecting uncertainty in cluster membership for neurons with little information. Here the average number of clusters is 16. To summarize the clustering results stored as posterior samples in MCMC, we give the single estimate for cluster indices $\hat{z}_i$ by maximizing the posterior expected adjusted Rand index (maxPEAR, Fritsch and Ickstadt 2009). The maxPEAR-sorted neural activity and posterior similarity matrix are shown in Fig. 4C and D. Results sorted by Maximum a posteriori (MAP) estimate are similar and are shown in the Appendix (Fig. 5). To examine the relationship between the clustering results and anatomy, we additionally sort the neurons according anatomical labels (upper left panel in Fig. 4E). Although many identified clusters are neurons from the same anatomical area,

clusters also include neurons from different regions and neurons within a region are often clustered into separate populations ($\hat{P}(\text{neuron } i, l \text{ in the same region} | z_i = z_l, \{\boldsymbol{y}_i\}_{i=1}^N) = 0.57$). Together, these results suggest that a simple assignment of populations based on anatomy many not accurately represent the latent structure.

We then evaluate the clustering patterns for different visual stimuli. We run 2 independent chains for each epoch (results from the second chain in Fig. 6D). The similarity matrices show that the pattern is consistent for the same epoch, but will change along the time even under the same experimental settings (D1 vs. D2 and S1 vs. S2). The changes in the clustering patterns may suggest long-term drift for neuron interactions. To quantify the observations, we evaluate the adjusted Rand index (ARI) of maxPEAR estimates (bottom right panel in Fig. 4E). Between-epoch comparisons tend to have lower similarity (average ARI from comparing 2 chains for each epoch) than within-epoch comparisons (different chains) for both maxPEAR and MAP (Fig. 6C).

The MAP number of clusters is largest (18) for the natural movie, suggesting this epoch has the most severe global non-linearity issue. Here we compare three models: 1) clustering model with $p = 2$, 2) single cluster model, with $p = 8$ selected by 5-fold speckled CV and 3) anatomical cluster model. The anatomical cluster model fits a single DPFA for each region, using the anatomical labels to define the clusters explicitly. Using cross-validation, we find that the optimized dimension for each region is $p = (1, 11, 5, 3)$, respectively. We find that the single cluster model tends to underfit the data, while the anatomical cluster model tends to overfit the data (Fig. 4F).

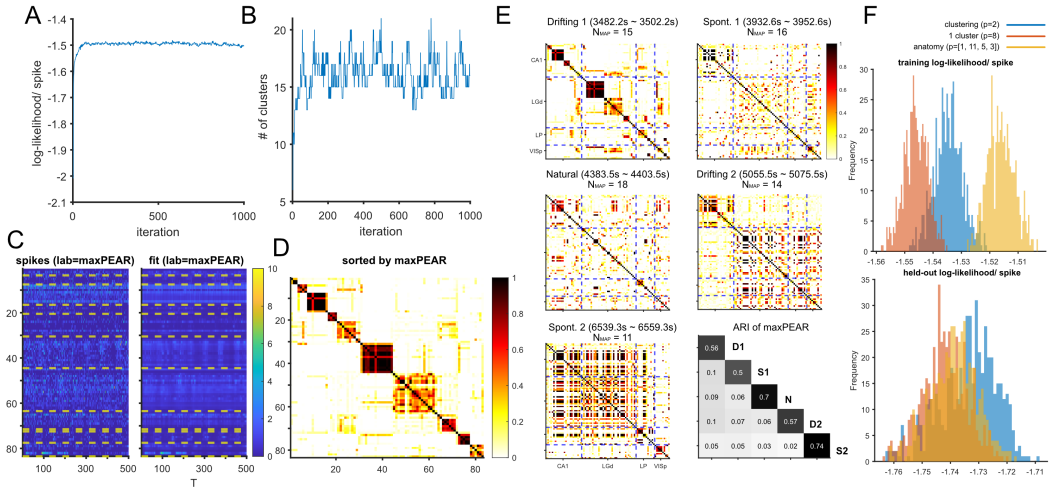

Figure 4: **Application in Neuropixels data. A. and B.** The trace plots of log-likelihood per spike and number of clusters for drifting grating responses. All results are averages from iteration 500 to 1000. **C.** The observed spikes and fitted mean firing rate, sorted by the maxPEAR label. **D.** The posterior similarity matrix, sorted by the maxPEAR label. **E.** The posterior similarity matrices for 4 adjacent epochs and 1 further epoch with different visual stimuli, sorted sorted according to maxPEAR estimate and anatomical label in the first drifting grating epoch. The last panel shows the adjusted Rand index of the maxPEAR estimates. The diagonal is the ARI between two chains for the same data, while off-diagonal values show the mean ARI of maxPEAR for the four comparisons between two chains from two different epochs. **F.** For the natural epoch, we hold out 1/2 data as the training, and show the histograms of training and held-out log-likelihood per spike, from iteration 500 to 1000 for three models: 1) clustering model, 2) single cluster model, and 3) anatomical cluster model.

5 Discussion

Here we introduce a Bayesian approach to cluster neural spike trains by MCMC. Previous approaches to multi-population latent variable modeling have used anatomical information to label distinct groups of neurons, but this choice is somewhat arbitrary. Brain region and cell-type, for instance, can give

contradictory population labels. The proposed method groups neurons by common latent factors, which may be useful for identifying "functional populations" of neurons. Here we use a mixDPFA model and infer the number of clusters by MFM with a partition-based algorithm similar to DPM. MFM may be more conceptually appropriate than DPM, since the number of neural "populations" is unknown but finite. Additionally, MFM produces more concentrated, evenly dispersed clusters (see Miller and Harrison 2018 for detailed discussion). The mixture modeling approach may also be appropriate in cases where neurons share non-homogeneous inputs, since it can approximate global nonlinearity with a mixture of locally linear models. Here we find that the mixture model outperforms globally linear (1-cluster) models in simulations and with experimental data.

Although the proposed method can describe data and cluster neural spiking activity successfully, there are some potential improvements. Firstly, as mentioned above, the unconstrained model does not have unique solutions. To ensure model identifiable, we put diagonal constraints on $\boldsymbol{A}^{(j)}$ and $\boldsymbol{Q}^{(j)}$ and constrain $\boldsymbol{\mu}^{(j)}$ and $\boldsymbol{X}^{(j)}$ to have mean zero. The assumption that $\boldsymbol{A}^{(j)}$ and $\boldsymbol{Q}^{(j)}$ are diagonal does not allow interaction between latent factors. However, these interactions could be allowed by instead constraining $\boldsymbol{X}'^{(j)}\boldsymbol{X}^{(j)}$ to be diagonal [Krzanowski and Marriott, 1994a,b, Fokoué and Titterington, 2003]. Such a constraint could allow unique solutions for the (P)LDS and GPFA. A second potential improvement would be to automatically infer the dimension of the latent factors (states). In this paper, we assume p_j is the same for all clusters, for convenience. p is a pre-selected value or can be selected by cross-validation (CV). This may limit the accuracy of the model, since populations of neurons in experimental data are likely to have different latent dimensionalities. In future work, it would also be possible to treat p_j as a parameter and sample the posterior by a reversible-jump (RJ)MCMC [Lopes and West, 2004], birth-death (BD)MCMC [Stephens, 2000, Fokoué and Titterington, 2003], or adaptive Gibbs sampling with shrinkage prior on $\boldsymbol{X}^{(j)}$ [Bhattacharya and Dunson, 2011]. Although RJMCMC and BDMCMC can be easily implemented, they perform poorly for high dimensional data and may be sensitive to priors. Adaptive Gibbs sampling with shrinkage, on the other hand, has been implemented with the infinite mixture of infinite factor analyzers (IMIFA, Murphy et al. 2020). The same idea may be useful here with an additional prior on linear dynamics ($\boldsymbol{A}^{(j)}$, $\boldsymbol{b}^{(j)}$ and $\boldsymbol{Q}^{(j)}$) to encourage shrinkage in $\boldsymbol{X}^{(j)}$. Finally, a deterministic approximation of MCMC, such as variational inference may be more computationally efficient. Standard methods for fitting the PLDS could be used directly in the VI updates, and if we further use a stick-breaking representation for the MFM model, it would be straightforward to use VI for clustering as well, similar to [Blei and Jordan, 2006].

As the number of neurons and brain regions that neuroscientists are able to record simultaneously continues to grow, understanding the latent structure of multiple populations will be a major statistical challenge. The Bayesian approach to clustering neural spike trains introduced here converges fast and is insensitive to the initial cluster assignments, and may, thus, be a useful tool for identifying "functional populations" of neurons.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1931249.

References

G. Arminger, P. Stein, and J. Wittenberg. Mixtures of conditional mean-and covariance-structure models. *Psychometrika*, 64(4):475–494, 1999.

B. Berman, W. O. Johnson, and W. Shen. Normal Approximation for Bayesian Mixed Effects Binomial Regression Models. *Bayesian Analysis*, pages 1 – 21, 2022.

A. Bhattacharya and D. B. Dunson. Sparse Bayesian infinite factor models. *Biometrika*, 98(2):291, 2011.

D. M. Blei and M. I. Jordan. Variational Inference for Dirichlet Process Mixtures. *Bayesian Analysis*, 1(1): 121–144, 2006.

L. Buesing, T. A. Machado, J. P. Cunningham, and L. Paninski. Clustered factor analysis of multineuronal spike data. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

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

A. B. Chan and N. Vasconcelos. Bayesian poisson regression for crowd counting. *Proceedings of the IEEE International Conference on Computer Vision*, pages 545–551, 2009.

S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2):216–222, 1987.

U. T. Eden, L. M. Frank, R. Barbieri, V. Solo, and E. N. Brown. Dynamic Analysis of Neural Encoding by Point Process Adaptive Filtering. *Neural Computation*, 16(5):971–998, 2004.

G. M. El-Sayyad. Bayesian and Classical Analysis of Poisson Regression. *Journal of the Royal Statistical Society: Series B (Methodological)*, 35(3):445–451, 1973.

M. Emtiyaz Khan, A. Aravkin, M. Friedlander, and M. Seeger. Fast dual variational inference for non-conjugate latent gaussian models. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 951–959, 2013.

E. Fokoué and D. M. Titterington. Mixtures of factor analysers. Bayesian estimation and inference by stochastic simulation. *Machine Learning*, 50(1-2):73–94, 2003.

A. Fritsch and K. Ickstadt. Improved criteria for clustering based on the posterior similarity matrix. *Bayesian Analysis*, 4(2):367–391, 2009.

S. Frühwirth-Schnatter. Data augmentation and dynamic linear models. *Journal of Time Series Analysis*, 15: 183–202, 1994.

Z. Ghahramani and G. E. Hinton. The EM Algorithm for Mixtures of Factor Analyzers. 1996.

J. Glaser, M. Whiteway, J. P. Cunningham, L. Paninski, and S. Linderman. Recurrent Switching Dynamical Systems Models for Multiple Interacting Neural Populations. In *Advances in Neural Information Processing Systems*, volume 33, pages 14867–14878, 2020.

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

S. Jain and R. M. Neal. Splitting and merging components of a nonconjugate Dirichlet process mixture model. *Bayesian Analysis*, 2(3):445 – 472, 2007.

I. M. Johnstone and A. Y. Lu. On Consistency and Sparsity for Principal Components Analysis in High Dimensions. *Journal of the American Statistical Association*, 104(486):682, 2009.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning 1999 37:2*, 37(2):183–233, 1999.

J. J. Jun, N. A. Steinmetz, J. H. Siegle, et al. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551(7679):232–236, 2017.

S. Keeley, D. Zoltowski, Y. Yu, S. Smith, and J. Pillow. Efficient non-conjugate Gaussian process factor models for spike count data using polynomial approximations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 5177–5186, 2020.

W. J. Krzanowski and F. H. C. Marriott. Multivariate analysis. Part I. Distributions, ordination and inference. 1994a.

W. J. Krzanowski and F. H. C. Marriott. Multivariate analysis. Part 2: Classification, covariance structures and repeated measurements. 1994b.

S. Linderman, R. P. Adams, and J. W. Pillow. Bayesian latent structure discovery from multi-neuron recordings. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

S. Linderman, M. Johnson, A. Miller, R. Adams, D. Blei, and L. Paninski. Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pages 914–922, 2017.

H. F. Lopes and M. West. Bayesian Model Assessment in Factor Analysis. *Statistica Sinica*, 14:41–67, 2004.

H. F. Lopes, E. Salazar, and D. Gamerman. Spatial dynamic factor analysis. *Bayesian Analysis*, 3(4):759–792, 2008.

J. H. Macke, L. Buesing, J. P. Cunningham, B. M. Yu, K. V. Shenoy, and M. Sahani. Empirical models of spiking in neural populations. *Advances in Neural Information Processing Systems*, 24, 2011.

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

K. Murphy, C. Viroli, and I. C. Gormley. Infinite Mixtures of Infinite Factor Analysers. *Bayesian Analysis*, 15 (3):937–963, 2020.

R. M. Neal. Markov Chain Sampling Methods for Dirichlet Process Mixture Models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

C. Pandarinath, D. J. O'Shea, J. Collins, R. Jozefowicz, S. D. Stavisky, J. C. Kao, E. M. Trautmann, M. T. Kaufman, S. I. Ryu, L. R. Hochberg, J. M. Henderson, K. V. Shenoy, L. F. Abbott, and D. Sussillo. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, 15(10):805–815, 2018.

L. Paninski, Y. Ahmadian, D. G. Ferreira, S. Koyama, K. Rahnama Rad, M. Vidne, J. Vogelstein, and W. Wu. A new look at state-space models for neural data. *Journal of Computational Neuroscience*, 29(1-2):107–126, 2010.

D. Peña and P. Poncela. Forecasting with nonstationary dynamic factor models. *Journal of Econometrics*, 119 (2):291–321, 2004.

H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, 1965.

J. D. Semedo, A. Zandvakili, C. K. Machens, B. M. Yu, and A. Kohn. Cortical areas interact through a communication subspace. *Neuron*, 102(1):249–259.e4, 2019.

J. H. Siegle, X. Jia, S. Durand, et al. Survey of spiking in the mouse visual system reveals functional hierarchy. *Nature*, 592(7852):86–92, 2021.

M. Stephens. Bayesian analysis of mixture models with an unknown number of components—an alternative to reversible jump methods. *The Annals of Statistics*, 28(1):40 – 74, 2000.

V. Ventura. Traditional waveform based spike sorting yields biased rate code estimates. *Proceedings of the National Academy of Sciences of the United States of America*, 106(17):6921–6926, 2009.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

M. R. Whiteway, K. Socha, V. Bonin, and D. A. Butts. Characterizing the nonlinear structure of shared variability in cortical neuron populations using latent variable models. *Neurons, behavior, data analysis and theory*, 3(1), 2019.

A. Williams, A. Degleris, Y. Wang, and S. Linderman. Point process models for sequence detection in high-dimensional neural spike trains. In *Advances in Neural Information Processing Systems*, volume 33, pages 14350–14361, 2020.

J. Windle, C. M. Carvalho, J. G. Scott, and L. Sun. Efficient Data Augmentation in Dynamic Models for Binary and Count Data. 2013.

A. Wu, N. A. Roy, S. Keeley, and J. W. Pillow. Gaussian process based nonlinear latent structure discovery in multivariate spike train data. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

B. M. Yu, J. P. Cunningham, G. Santhanam, S. I. Ryu, K. V. Shenoy, and M. Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Journal of Neurophysiology*, 102(1):614–635, 2009.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 MCMC updates

The posteriors are sampled by a Gibbs sampler. In each iteration, the sampling scheme has two main stages: 1) to sample the model parameters assuming known labels and 2) to sample the cluster indices given model parameters. Before moving into the second stage, the first stage is repeated several times. The sampling in the first stage is conducted without considering constraints for $\mu_t^{(j)}$ and $\boldsymbol{x}_t^{(j)}$ at first, and then we project the samples onto the constraint space for $\sum_{t=1}^{T} \mu_t^{(j)} = 0$ and $\sum_{t=1}^{T} \boldsymbol{x}_t^{(j)} = \mathbf{0}$ as used in [**?**].

Update $\boldsymbol{x}_t^{(j)}$ and $\mu_t^{(j)}$ The priors for initial population baseline and latent factor are:

$$\mu_1^{(j)} \sim \mathcal{N}(0,1),$$
$$\boldsymbol{x}_1^{(j)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_p).$$

Assume there are $n_j = \#\{i : z_i = j\}$ neurons belong to the j-th cluster, and denote the spike counts and firing rates of these neurons at time t as $\widetilde{\boldsymbol{y}}_t^{(j)} = vec(\{y_{it}|z_i = j\}) \in \mathbb{Z}_{\geq 0}^{n_j}$ and $\widetilde{\boldsymbol{\lambda}}_t^{(j)} = vec(\{\lambda_{it}|z_i = j\}) \in \mathbb{R}^{n_j}$, where $\mathbb{Z}_{\geq 0}^{n_j}$ denotes a n_j-dimensional vector with each element being non-negative integers. The corresponding loading and baseline for these n_j neurons is $\boldsymbol{C}^{(j)} \in \mathbb{R}^{n_j \times p}$ and $\boldsymbol{\Delta}_j = vec(\{\delta_i|z_i = j\}) \in \mathbb{R}^{n_j}$, such that $\log \widetilde{\boldsymbol{\lambda}}_t^{(j)} = \boldsymbol{\Delta}_j + \mu_t^{(j)} \boldsymbol{1}_{n_j} + \boldsymbol{C}^{(j)} \boldsymbol{x}_t^{(j)} = \boldsymbol{\Delta}_j + \left(\boldsymbol{1}_{n_j}, \boldsymbol{C}^{(j)}\right) \left(\mu_t^{(j)}, \boldsymbol{x}_t'^{(j)}\right)'$. Denote $\widetilde{\boldsymbol{C}}^{(j)} = \left(\boldsymbol{1}_{n_j}, \boldsymbol{C}^{(j)}\right)$, $\widetilde{\boldsymbol{x}}_t^{(j)} = \left(\mu_t^{(j)}, \boldsymbol{x}_t'^{(j)}\right)'$, $\widetilde{\boldsymbol{x}}^{(j)} = \left(\widetilde{\boldsymbol{x}}_1'^{(j)}, \ldots, \widetilde{\boldsymbol{x}}_T'^{(j)}\right)'$, $\widetilde{\boldsymbol{A}}^{(j)} = diag(h^{(j)}, \boldsymbol{A}^{(j)})$, $\widetilde{\boldsymbol{b}}^{(j)} = \left(g^{(j)}, \boldsymbol{b}'^{(j)}\right)'$ and $\widetilde{\boldsymbol{Q}}^{(j)} = diag(\sigma^{2(j)}, \boldsymbol{Q}^{(j)})$. The full conditional distribution $P(\widetilde{\boldsymbol{x}}^{(j)}|\ldots) = P(\widetilde{\boldsymbol{x}}^{(j)}|\{\widetilde{\boldsymbol{y}}_t^{(j)}\}_{t=1}^T, \boldsymbol{\Delta}_j, \widetilde{\boldsymbol{C}}^{(j)}, \widetilde{\boldsymbol{A}}^{(j)}, \widetilde{\boldsymbol{b}}^{(j)}, \widetilde{\boldsymbol{Q}}^{(j)})$ is approximated by a global Laplace approximation, i.e.,

$$P(\widetilde{\boldsymbol{x}}^{(j)}|\ldots) \approx \mathcal{N}_{(p+1)T}(\widetilde{\boldsymbol{x}}^{(j)}|\boldsymbol{\mu}_{\widetilde{\boldsymbol{x}}^{(j)}}, \boldsymbol{\Sigma}_{\widetilde{\boldsymbol{x}}^{(j)}}),$$
$$\boldsymbol{\mu}_{\widetilde{\boldsymbol{x}}^{(j)}} = \arg\max_{\widetilde{\boldsymbol{x}}^{(j)}} P(\widetilde{\boldsymbol{x}}^{(j)}|\ldots),$$
$$\boldsymbol{\Sigma}_{\widetilde{\boldsymbol{x}}^{(j)}}) = -(\nabla\nabla \log P(\widetilde{\boldsymbol{x}}^{(j)}|\ldots)|_{\widetilde{\boldsymbol{x}}^{(j)} = \boldsymbol{\mu}_{\widetilde{\boldsymbol{x}}^{(j)}}})^{-1}.$$

Then, taking the logarithm of the full conditional distribution, i.e., $\ell = \ell(\widetilde{\boldsymbol{x}}^{(j)}) = \log P(\widetilde{\boldsymbol{x}}^{(j)}|\ldots)$, we have

$$\ell = \text{const} + \sum_{t=1}^{T} \left(\widetilde{\boldsymbol{y}}_t'^{(j)}(\boldsymbol{\Delta}_j + \widetilde{\boldsymbol{C}}^{(j)}\widetilde{\boldsymbol{x}}_t^{(j)}) - \boldsymbol{1}_{n_j}'\widetilde{\boldsymbol{\lambda}}_t\right) - \frac{1}{2}(\widetilde{\boldsymbol{x}}_1^{(j)} - \widetilde{\boldsymbol{x}}_0'^{(j)})[\widetilde{\boldsymbol{Q}}_0^{(j)}]^{-1}(\widetilde{\boldsymbol{x}}_1^{(j)} - \widetilde{\boldsymbol{x}}_0^{(j)})$$
$$- \sum_{t=2}^{T} \frac{1}{2}(\widetilde{\boldsymbol{x}}_t^{(j)} - \widetilde{\boldsymbol{A}}^{(j)}\widetilde{\boldsymbol{x}}_{t-1}^{(j)} - \widetilde{\boldsymbol{b}}'^{(j)})[\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}(\widetilde{\boldsymbol{x}}_t^{(j)} - \widetilde{\boldsymbol{A}}^{(j)}\widetilde{\boldsymbol{x}}_{t-1}^{(j)} - \widetilde{\boldsymbol{b}}^{(j)}).$$

Here, $\ell(\widetilde{\boldsymbol{x}}^{(j)})$ is concave and unimodal. By the Markovian assumption for latent state vectors, the Hessian matrix is tri-block diagonal. We can thus compute Newton updates to get $\boldsymbol{\mu}_{\widetilde{\boldsymbol{x}}^{(j)}}$ for the Laplace approximation in $\mathcal{O}(T)$ [Paninski et al., 2010], similar to the E-step for the PLDS [Macke et al., 2011].

The gradient $\nabla\ell$ and the Hessian $\nabla\nabla\ell$ are provided as follows. For $t = 2, \ldots, T-1$, the gradient of $\ell(\widetilde{\boldsymbol{x}}^{(j)})$ is:

$$\nabla\ell = \left[\left(\frac{\partial\ell}{\partial\widetilde{\boldsymbol{x}}_1^{(j)}}\right)', \ldots, \left(\frac{\partial\ell}{\partial\widetilde{\boldsymbol{x}}_T^{(j)}}\right)'\right]',$$
$$\frac{\partial\ell}{\partial\widetilde{\boldsymbol{x}}_1^{(j)}} = \widetilde{\boldsymbol{C}}'^{(j)}(\widetilde{\boldsymbol{y}}_1^{(j)} - \widetilde{\boldsymbol{\lambda}}_1) - [\widetilde{\boldsymbol{Q}}_0^{(j)}]^{-1}(\widetilde{\boldsymbol{x}}_1^{(j)} - \widetilde{\boldsymbol{x}}_0^{(j)}) + \widetilde{\boldsymbol{A}}'^{(j)}[\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}(\widetilde{\boldsymbol{x}}_2^{(j)} - \widetilde{\boldsymbol{A}}^{(j)}\widetilde{\boldsymbol{x}}_1^{(j)} - \widetilde{\boldsymbol{b}}^{(j)}),$$
$$\frac{\partial\ell}{\partial\widetilde{\boldsymbol{x}}_t^{(j)}} = \widetilde{\boldsymbol{C}}'^{(j)}(\widetilde{\boldsymbol{y}}_t^{(j)} - \widetilde{\boldsymbol{\lambda}}_t) - [\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}(\widetilde{\boldsymbol{x}}_t^{(j)} - \widetilde{\boldsymbol{A}}^{(j)}\widetilde{\boldsymbol{x}}_{t-1}^{(j)} - \widetilde{\boldsymbol{b}}^{(j)})$$
$$+ \widetilde{\boldsymbol{A}}'^{(j)}[\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}(\widetilde{\boldsymbol{x}}_{t+1}^{(j)} - \widetilde{\boldsymbol{A}}^{(j)}\widetilde{\boldsymbol{x}}_t^{(j)} - \widetilde{\boldsymbol{b}}^{(j)}),$$
$$\frac{\partial\ell}{\partial\widetilde{\boldsymbol{x}}_T^{(j)}} = \widetilde{\boldsymbol{C}}'^{(j)}(\widetilde{\boldsymbol{y}}_t^{(j)} - \widetilde{\boldsymbol{\lambda}}_T) - [\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}(\widetilde{\boldsymbol{x}}_T^{(j)} - \widetilde{\boldsymbol{A}}^{(j)}\widetilde{\boldsymbol{x}}_{T-1}^{(j)} - \widetilde{\boldsymbol{b}}^{(j)}).$$

And the Hessian matrix is:

$$\nabla\nabla\ell = \begin{pmatrix} \frac{\partial^2\ell}{\partial\widetilde{\boldsymbol{x}}_1^{(j)}\partial\widetilde{\boldsymbol{x}}_1'^{(j)}} & \widetilde{\boldsymbol{A}}'^{(j)}[\widetilde{\boldsymbol{Q}}^{(j)}]^{-1} & \mathbf{0} & \cdots & \mathbf{0} \\ [\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}\widetilde{\boldsymbol{A}}^{(j)} & \frac{\partial^2\ell}{\partial\widetilde{\boldsymbol{x}}_2^{(j)}\partial\widetilde{\boldsymbol{x}}_2'^{(j)}} & \widetilde{\boldsymbol{A}}'^{(j)}[\widetilde{\boldsymbol{Q}}^{(j)}]^{-1} & \cdots & \vdots \\ \mathbf{0} & [\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}\widetilde{\boldsymbol{A}}^{(j)} & \frac{\partial^2\ell}{\partial\widetilde{\boldsymbol{x}}_3^{(j)}\partial\widetilde{\boldsymbol{x}}_3'^{(j)}} & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \cdots & \cdots & \frac{\partial^2\ell}{\partial\widetilde{\boldsymbol{x}}_T^{(j)}\partial\widetilde{\boldsymbol{x}}_T'^{(j)}} \end{pmatrix},$$

$$\frac{\partial^2\ell}{\partial\widetilde{\boldsymbol{x}}_1^{(j)}\partial\widetilde{\boldsymbol{x}}_1'^{(j)}} = -\widetilde{\boldsymbol{C}}'^{(j)}diag(\widetilde{\boldsymbol{\lambda}}_1)\widetilde{\boldsymbol{C}}^{(j)} - [\widetilde{\boldsymbol{Q}}_0^{(j)}]^{-1} - \widetilde{\boldsymbol{A}}'^{(j)}[\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}\widetilde{\boldsymbol{A}}^{(j)},$$

$$\frac{\partial^2\ell}{\partial\widetilde{\boldsymbol{x}}_t^{(j)}\partial\widetilde{\boldsymbol{x}}_t'^{(j)}} = -\widetilde{\boldsymbol{C}}'^{(j)}diag(\widetilde{\boldsymbol{\lambda}}_t)\widetilde{\boldsymbol{C}}^{(j)} - [\widetilde{\boldsymbol{Q}}^{(j)}]^{-1} - \widetilde{\boldsymbol{A}}'^{(j)}[\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}\widetilde{\boldsymbol{A}}^{(j)},$$

$$\frac{\partial^2\ell}{\partial\widetilde{\boldsymbol{x}}_T^{(j)}\partial\widetilde{\boldsymbol{x}}_T'^{(j)}} = -\widetilde{\boldsymbol{C}}'^{(j)}diag(\widetilde{\boldsymbol{\lambda}}_T)\widetilde{\boldsymbol{C}}^{(j)} - [\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}.$$

Although we can conduct the Newton update efficiently in $\mathcal{O}(T)$, bad initial values may slow down the convergence. To facilitate convergence, we initialize the Newton update with a smoothing estimate by local Gaussian approximation. The forward filtering for a dynamic Poisson model has been previously described [Eden et al., 2004], and we use an additional backward pass to smooth [Rauch et al., 1965].

Let $\widetilde{\boldsymbol{x}}_{t|t-1}^{(j)} = E(\widetilde{\boldsymbol{x}}_t^{(j)}|\widetilde{\boldsymbol{y}}_1^{(j)},\ldots,\widetilde{\boldsymbol{y}}_{t-1}^{(j)})$ and $\boldsymbol{V}_{t|t-1}^{(j)} = Var(\widetilde{\boldsymbol{x}}_t^{(j)}|\widetilde{\boldsymbol{y}}_1^{(j)},\ldots,\widetilde{\boldsymbol{y}}_{t-1}^{(j)})$ be the mean and variance for the one-step prediction density, where $\widetilde{\boldsymbol{x}}_{t|t-1}^{(j)} = \widetilde{\boldsymbol{A}}^{(j)}\widetilde{\boldsymbol{x}}_{t-1|t-1}^{(j)} + \widetilde{\boldsymbol{b}}^{(j)}$ and $\boldsymbol{V}_{t|t-1}^{(j)} = \widetilde{\boldsymbol{A}}^{(j)}\boldsymbol{V}_{t-1|t-1}^{(j)}\widetilde{\boldsymbol{A}}'^{(j)} + \widetilde{\boldsymbol{Q}}^{(j)}$. Then, after we observe the data at time t, we can do a forward filtering step for the mean $\widetilde{\boldsymbol{x}}_{t|t}^{(j)} = E(\widetilde{\boldsymbol{x}}_t^{(j)}|\widetilde{\boldsymbol{y}}_1^{(j)},\ldots,\widetilde{\boldsymbol{y}}_t^{(j)})$ and the variance $\boldsymbol{V}_{t|t}^{(j)} = Var(\widetilde{\boldsymbol{x}}_t^{(j)}|\widetilde{\boldsymbol{y}}_1^{(j)},\ldots,\widetilde{\boldsymbol{y}}_t^{(j)})$, which are given by

$$\widetilde{\boldsymbol{x}}_{t|t}^{(j)} = \widetilde{\boldsymbol{x}}_{t|t-1}^{(j)} + \boldsymbol{V}_{t|t-1}^{(j)}[\widetilde{\boldsymbol{C}}'^{(j)}(\widetilde{\boldsymbol{y}}_t^{(j)} - \widetilde{\boldsymbol{\lambda}}_t)]_{\widetilde{\boldsymbol{x}}_t^{(j)}=\widetilde{\boldsymbol{x}}_{t|t-1}^{(j)}},$$

$$[\boldsymbol{V}_{t|t}^{(j)}]^{-1} = [\boldsymbol{V}_{t|t-1}^{(j)}]^{-1} + [\widetilde{\boldsymbol{C}}'^{(j)}diag(\widetilde{\boldsymbol{\lambda}}_t)\widetilde{\boldsymbol{C}}^{(j)} - [\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}]_{\widetilde{\boldsymbol{x}}_t^{(j)}=\widetilde{\boldsymbol{x}}_{t|t-1}^{(j)}}.$$

Derivation of the filtering estimates can be found in [Eden et al., 2004], and we can further get the smoothing estimates directly by standard Rauch-Tung-Striebel smoother [Rauch et al., 1965].

The smoother estimates $\widetilde{\boldsymbol{x}}_{t|T}^{(j)}$ and $\boldsymbol{V}_{t|T}^{(j)}$ are updated as follows:

$$\widetilde{\boldsymbol{x}}_{t-1|T}^{(j)} = \widetilde{\boldsymbol{x}}_{t-1|t-1}^{(j)} + \boldsymbol{J}_{t-1}(\widetilde{\boldsymbol{x}}_{t|T}^{(j)} - \widetilde{\boldsymbol{x}}_{t|t-1}^{(j)}),$$

$$\boldsymbol{V}_{t-1|T}^{(j)} = \boldsymbol{V}_{t-1|t-1}^{(j)} + \boldsymbol{J}_{t-1}(\boldsymbol{V}_{t|T}^{(j)} - \boldsymbol{V}_{t|t-1}^{(j)})\boldsymbol{J}_{t-1}',$$

where $\boldsymbol{J}_{t-1} = \boldsymbol{V}_{t-1|t-1}^{(j)}\widetilde{\boldsymbol{A}}'^{(j)}[\boldsymbol{V}_{t|t-1}^{(j)}]^{-1}$.

Update δ_i and $\boldsymbol{c}_i$ We specify the prior for neuron-specific baseline δ_i as $\delta_i \sim \mathcal{N}(0,1)$ and we have assumed the loading $\boldsymbol{c}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_p)$. Then, from the matrix representation of DPFA in (1), i.e., $\log\boldsymbol{\lambda}_i = \delta_i\mathbf{1}_T + \boldsymbol{\mu}^{(j)} + \boldsymbol{X}^{(j)}\boldsymbol{c}_i$, it is easy to see that given $\boldsymbol{\mu}^{(j)}$ and $\boldsymbol{X}^{(j)}$ are known, the update of δ_i and $\boldsymbol{c}_i$ is just a regular Bayesian Poisson regression problem. Thus, we can sample the full conditional distribution of δ_i and $\boldsymbol{c}_i$ by a Hamiltonian Monte Carlo (HMC, Duane et al. 1987) within the Gibbs sampler.

Update parameters of latent state The parameters for linear dynamics are $h^{(j)}$, $g^{(j)}$, $\sigma^{2(j)}$, $\boldsymbol{A}^{(j)}$, $\boldsymbol{b}^{(j)}$ and $\boldsymbol{Q}^{(j)}$. To make the model identifiable, we simply assume $\boldsymbol{A}^{(j)} = diag(a_1^{(j)},\ldots,a_p^{(j)})$ and $\boldsymbol{Q}^{(j)} = diag(q_1^{(j)},\ldots,q_p^{(j)})$. Therefore, we can update $\boldsymbol{A}^{(j)}$, $\boldsymbol{b}^{(j)}$ and $\boldsymbol{Q}^{(j)}$ for each diagonal

element separately, as the update in $h^{(j)}$, $g^{(j)}$ and $\sigma^{2(j)}$. Here, we update $h^{(j)}$, $g^{(j)}$ and $\sigma^{2(j)}$ as follows.

First, we specify the priors for $\sigma^{2(j)}$ following $IG\left(\nu_0/2, \nu_0\sigma_0^2/2\right)$ and $\left(g^{(j)}, h^{(j)}\right)' \sim \mathcal{N}(\boldsymbol{\tau}_0, \sigma^{2(j)}\boldsymbol{\Lambda}_0^{-1})$, with $\nu_0 = 1$, $\sigma_0 = 0.01$, $\boldsymbol{\tau}_0 = (0, 1)'$ and $\boldsymbol{\Lambda}_0 = \boldsymbol{I}_2$. Here, the "IG" denotes the inverse-gamma distribution.

Denote $\boldsymbol{\mu}_{2:T}^{(j)} = \left(\mu_2^{(j)}, \ldots, \mu_T^{(j)}\right)'$ and $\widetilde{\boldsymbol{\mu}}_{1:(T-1)}^{(j)} = \left(\mathbf{1}_{T-1}, \boldsymbol{\mu}_{1:(T-1)}^{(j)}\right)$, with $\boldsymbol{\mu}_{1:(T-1)}^{(j)} = \left(\mu_1^{(j)}, \ldots, \mu_{T-1}^{(j)}\right)'$. The full conditional distributions for $\sigma^{2(j)}$ and $\left(g^{(j)}, h^{(j)}\right)'$ are:

$$\sigma^{2(j)}|\{\mu_t^{(j)}\}_{t=1}^T \sim IG\left(\frac{\nu_0 + T - 1}{2}, \frac{\nu_0\sigma_0^2 + \boldsymbol{\mu}_{2:T}^{\prime(j)}\boldsymbol{\mu}_{2:T}^{(j)} + \boldsymbol{\tau}_0'\boldsymbol{\Lambda}_0\boldsymbol{\tau}_0 - \boldsymbol{\tau}_n'\boldsymbol{\Lambda}_n\boldsymbol{\tau}_n}{2}\right),$$

$$\left(g^{(j)}, h^{(j)}\right)'|\{\mu_t^{(j)}\}_{t=1}^T \sim \mathcal{N}(\boldsymbol{\tau}_n, \sigma^{2(j)}\boldsymbol{\Lambda}_n^{-1}),$$

with $\boldsymbol{\Lambda}_n = \widetilde{\boldsymbol{\mu}}_{1:(T-1)}^{\prime(j)}\widetilde{\boldsymbol{\mu}}_{1:(T-1)}^{(j)} + \boldsymbol{\Lambda}_0$, and $\boldsymbol{\tau}_n = \boldsymbol{\Lambda}_n^{-1}\left(\widetilde{\boldsymbol{\mu}}_{1:(T-1)}^{\prime(j)}\boldsymbol{\mu}_{2:T}^{(j)} + \boldsymbol{\Lambda}_0\boldsymbol{\tau}_0\right)$.

For completeness, we also provide the update of latent state parameters when using the more parsimonious constraint, i.e. diagonal $\boldsymbol{X}'^{(j)}\boldsymbol{X}^{(j)}$. According to results from previous research [Krzanowski and Marriott, 1994a,b, Fokoué and Titterington, 2003], this constraint is one of the most parsimonious one, which only put constraints on $p(p-1)/2$ parameters. The constraint can also be implemented in MCMC by "unconstrained sampling-projection" procedure [**?**].

Without constraints, the sampling of $h^{(j)}$, $g^{(j)}$ and $\sigma^{2(j)}$ is the same as shown previously. The update of $\boldsymbol{A}^{(j)}$, $\boldsymbol{b}^{(j)}$ and $\boldsymbol{Q}^{(j)}$ is the standard multivariate Bayesian linear regression. Denote $\boldsymbol{X}_{2:T}^{(j)} = (\boldsymbol{x}_2^{(j)}, \ldots, \boldsymbol{x}_T^{(j)})'$ and $\widetilde{\boldsymbol{X}}_{1:(T-1)}^{(j)} = (\mathbf{1}_{T-1}, \boldsymbol{X}_{1:(T-1)}^{(j)})$, with $\boldsymbol{X}_{1:(T-1)}^{(j)} = (\boldsymbol{x}_1^{(j)}, \ldots, \boldsymbol{x}_{T-1}^{(j)})'$. Let us use the conjugate priors as following for $\boldsymbol{Q}^{(j)}$, $\boldsymbol{b}^{(j)}$ and $\boldsymbol{A}^{(j)}$:

$$\boldsymbol{Q}^{(j)} \sim \mathcal{W}^{-1}(\boldsymbol{\Psi}_0, \gamma_0),$$
$$vec((\boldsymbol{b}^{(j)}, \boldsymbol{A}'^{(j)})) \sim N(vec(\boldsymbol{T}_0), \boldsymbol{Q}^{(j)} \otimes \boldsymbol{\Gamma}_0^{-1}).$$

Here, the $\mathcal{W}^{-1}$ denotes the inverse-Wishart distribution. We can set the priors as $\boldsymbol{\Psi}_0 = 0.01\boldsymbol{I}_p$, $\gamma_0 = p+2$ and $\boldsymbol{T}_0 = (\mathbf{0}_p, \boldsymbol{I}_p)'$. Then, the full conditional distributions for $\boldsymbol{Q}^{(j)}$ and $vec((\boldsymbol{b}^{(j)}, \boldsymbol{A}^{(j)})')$ are:

$$\boldsymbol{Q}^{(j)}|\boldsymbol{X}^{(j)} \sim \mathcal{W}^{-1}(\boldsymbol{\Psi}_n, \gamma_n),$$
$$vec((\boldsymbol{b}^{(j)}, \boldsymbol{A}^{(j)})')|\boldsymbol{X}^{(j)} \sim N(vec(\boldsymbol{T}_n), \boldsymbol{Q}^{(j)} \otimes \boldsymbol{\Gamma}_n^{-1}),$$

with

$$\begin{aligned}
\boldsymbol{\Psi}_n &= \boldsymbol{\Psi}_0 + (\boldsymbol{X}_{2:T}^{(j)} - \widetilde{\boldsymbol{X}}_{1:(T-1)}^{(j)}\boldsymbol{T}_n)'(\boldsymbol{X}_{2:T}^{(j)} - \widetilde{\boldsymbol{X}}_{1:(T-1)}^{(j)}\boldsymbol{T}_n), \\
&\quad + (\boldsymbol{T}_n - \boldsymbol{T}_0)'\boldsymbol{\Gamma}_0(\boldsymbol{T}_n - \boldsymbol{T}_0), \\
\gamma_n &= \gamma_0 + T - 1, \\
\boldsymbol{T}_n &= \boldsymbol{\Gamma}_n^{-1}(\widetilde{\boldsymbol{X}}_{1:(T-1)}^{\prime(j)}\boldsymbol{X}_{2:T}^{(j)} + \boldsymbol{\Gamma}_0\boldsymbol{T}_0), \\
\boldsymbol{\Gamma}_n &= \widetilde{\boldsymbol{X}}_{1:(T-1)}^{\prime(j)}\widetilde{\boldsymbol{X}}_{1:(T-1)}^{(j)} + \boldsymbol{\Gamma}_0.
\end{aligned}$$

Before updating the cluster indices z_i, the sampling of $\{\boldsymbol{x}_t^{(j)}, \mu_t^{(j)}, \delta_i, \boldsymbol{c}_i, \boldsymbol{\theta}^{(j)}\}$ is repeated several times (5 times in both simulation and application for this paper). To save time, we can further allow the repetitions to be pre-stopped, when the training log-likelihood converges roughly.

Update z_i To update the cluster assignments for each neuron i, we use a partition based algorithm for MFM, similarly as described in Miller and Harrison 2018.

Let $\mathcal{C}$ denote a partition of neurons, and $\mathcal{C}\backslash i$ denote the partition obtained by removing neuron i from $\mathcal{C}$.

1. Initialize $\mathcal{C}$ and $\{\boldsymbol{\theta}^{(c)} : c \in \mathcal{C}\}$ (e.g. one cluster).
2. Repeat the following steps G times to obtain G samples. For $i = 1, \ldots, N$: remove neuron i from $\mathcal{C}$ and place it:
   (a) in $c \in \mathcal{C} \backslash i$ with probability $\propto (|c| + \gamma) M_{\boldsymbol{\theta}^{(c)}}(\boldsymbol{y}_i)$, where γ is defined in the MFM model in the main text (Equation 2) and $M_{\boldsymbol{\theta}^{(c)}}(\boldsymbol{y}_i)$ denotes the marginal likelihood of neuron i in cluster c, when integrating the loading $\boldsymbol{c}_i$ out (Equation (3)). The marginal likelihood is approximated by using Poisson-Gamma conjugacy.
   (b) in a new cluster c^* with probability $\propto \gamma \frac{V_n(t+1)}{V_n(t)} M_{\boldsymbol{\theta}^{(c^*)}}(\boldsymbol{y}_i)$, where t is the number of partitions obtained by removing the neuron i and $V_n(t) = \sum_{j=1}^{\infty} \frac{j_{(t)}}{(\gamma j)^{(n)}} f_k(j)$, with $x^{(m)} = x(x+1)\cdots(x+m-1)$, $x_{(m)} = x(x-1)\cdots(x-m+1)$, $x^{(0)} = 1$ and $x_{(0)} = 1$.

The update is an adaptation of partition-based algorithm for DPM [Neal, 2000], but with two substitutions: 1) replace $|c_i|$ by $|c_i| + \gamma$ and 2) replace α by $\gamma V_n(t+1)/V_n(t)$. See more details and discussions in [Miller and Harrison, 2018]. When evaluating the likelihood, we marginalize the cluster-independent loading $\boldsymbol{c}_i$ out. This is necessary for the high dimensional situation, otherwise the chain will stop moving.

One issue with incremental Gibbs samplers such as Algorithm 3 and 8 in Neal [2000], when applied to DPM, is that mixing can be somewhat slow. To further improve the mixing, we may intersperse the "split-merge" Metropolis-Hasting updates [Jain and Neal, 2007, **?**] between Gibbs sweeps, as in [Miller and Harrison, 2018].

A.2 Sample Latent Vectors Using Pólya-Gamma Augmentation and Metropolis-Hastings Algorithm

In this section, we provide details of sampling algorithms to draw the latent states $\boldsymbol{X}^{(j)}$ and population baseline $\boldsymbol{\mu}^{(j)}$ from the exact posterior of the model. In order to explain the algorithm much clearer, we just focus on a given cluster index, and thus in this section we suspend the superscript (j) as the cluster index in our notations. We present the sampling idea with two major parts as described below.

Pólya-Gamma Augmentation Although the mixDPFA model does not directly follow the PG augmentation scheme [**?**], we can approximate the Poisson distribution by a negative binomial (NB) distribution, i.e., $\lim_{r\to\infty} \text{NB}(r, \sigma(\psi - \log r)) = \text{Poisson}(e^{\psi})$, where $\sigma(\psi) = e^{\psi}/(1+e^{\psi})$ and $\text{NB}(r, p)$ denotes the NB distribution with $rp/(1-p)$ as its expectation. We approximate the proposed DPFA model using a NB, then we can instead sample a mixture of NB factor analyzers using the PG scheme [Windle et al., 2013]. Further, we use the forward-filtering-backward-sampling (FFBS) algorithm [Carter and Kohn, 1994, Frühwirth-Schnatter, 1994] to update the latent states $\boldsymbol{X}$ and population baseline $\boldsymbol{\mu}$. These two steps are summarized in the step 1 and step 2 of Algorithm 1 for updating $\widetilde{\boldsymbol{X}} = (\boldsymbol{\mu}, \boldsymbol{X})$.

Metropolis-Hastings Step Next, we use the samples of $\widetilde{\boldsymbol{X}}$ yielded from FFBS algorithm as a proposal, where we employ a Metropolis-Hastings (MH) step to reject or accept the proposal. In this step, the dispersion parameter r in NB distribution becomes a tuning parameter, to balance acceptance rate and autocorrelation in MH. When r is large, the approximation to Poisson observation is accurate and the MH performs similar to the Gibbs sampler. Here, we allow neurons at different time points to have unique tuning parameters r_{it}. The MH step is summarized in step 3 of Algorithm 1.

We have further implemented Algorithm 1 to fit DPFA model for cluster 1 data in Fig. 2 and the results have been presented in Fig. 5. We set the tuning parameters for each neuron at all time points as $r_{it} = 10$, to tune the acceptance rate around 0.4. When implementing the PG-MH algorithm in DPFA, the ω_{it} (i.e., PG random variable) was sampled using `pgdraw` function in R package `pgdraw`, which is called from our MATLAB code. Although we can program Algorithm 1 without calling `pgdraw` function from R to save some time, to sample the PG random variable is often much computationally expensive in comparison to sample from the Gaussian random variable. When fitting the simulated data with $N = 5$, $T = 1000$ and $p = 2$ (cluster 1 data in Fig. 2) using a 3.40 GHz processor with 16 GB of RAM, it takes 131.64s for PG-MH to draw 1000 posterior samples, while the proposed approximation method in the main context of the paper takes 83.86s. Although the

Algorithm 1: Pólya-Gamma-Metropolis-Hastings Algorithm (PG-MH) for Poisson Dynamic Model

Recall the DPFA model using the notations defined in A.1:

$$y_{it} \sim Poi(\lambda_{it}),$$
$$\log \lambda_{it} = \delta_i + \widetilde{\boldsymbol{c}}_i' \widetilde{\boldsymbol{x}}_t,$$

for $i = 1, \ldots, n$ and $t = 1, \ldots, T$. Here, $\widetilde{\boldsymbol{c}}_i = (1, \boldsymbol{c}_i')'$ and $\widetilde{\boldsymbol{x}}_t = (\mu_t, \boldsymbol{x}_t')'$. $\widetilde{\boldsymbol{x}}_t$ follows linear dynamics $\widetilde{\boldsymbol{x}}_{t+1}|\widetilde{\boldsymbol{x}}_t \sim \mathcal{N}(\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{x}}_t + \widetilde{\boldsymbol{b}}, \widetilde{\boldsymbol{Q}})$. Denote the prior as $\widetilde{\boldsymbol{x}}_1 \sim \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{V}_0)$. Given the sample from the $(G-1)$-th iteration $\widetilde{\boldsymbol{x}}_t^{(G-1)}$ and $\boldsymbol{U} = \{\widetilde{\boldsymbol{c}}_i, \delta_i, \widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{b}}, \widetilde{\boldsymbol{Q}}\}$.

1. sample ω_{it} from PG distribution and calculate $\hat{y}_{it}$, which follows $\mathcal{N}(\widetilde{\boldsymbol{c}}_i'\widetilde{\boldsymbol{x}}_t^{(G-1)}, \omega_{it}^{-1})$
for $t = 1, \ldots, T$ **do**
  for $i = 1, \ldots, n$ **do**
    sample $\omega_{it} \sim P_{PG}(r_{it} + y_{it}, \delta_i + \widetilde{\boldsymbol{c}}_i'\widetilde{\boldsymbol{x}}_t^{(G-1)} - \log r_{it})$
    $\kappa_{it} = (y_{it} - r_{it})/2 + \omega_{it}(\log r_{it} - \delta_i)$
    $\hat{y}_{it} = \omega_{it}^{-1}\kappa_{it}$
  end
end

2. Forward-filtering-backward-sampling (FFBS) for $\widetilde{\boldsymbol{X}}$
Denote $\hat{\boldsymbol{y}}_t = (\hat{y}_{1t}, \ldots, \hat{y}_{Nt})'$, $\boldsymbol{\Omega}_t = Diag([\omega_{1t}, \ldots, \omega_{Nt}])$ and $\widetilde{\boldsymbol{C}} = (\widetilde{\boldsymbol{c}}_1, \ldots, \widetilde{\boldsymbol{c}}_N)'$
for $t = 1, \ldots, T$ **do**
  $\boldsymbol{m}_{t|t-1} = \widetilde{\boldsymbol{A}}\boldsymbol{m}_{t-1} + \widetilde{\boldsymbol{b}}$
  $\boldsymbol{V}_{t|t-1} = \widetilde{\boldsymbol{A}}\boldsymbol{V}_{t-1}\widetilde{\boldsymbol{A}}' + \widetilde{\boldsymbol{Q}}$
  $\boldsymbol{K}_t = \boldsymbol{V}_{t|t-1}\widetilde{\boldsymbol{C}}'(\widetilde{\boldsymbol{C}}\boldsymbol{V}_{t|t-1}\widetilde{\boldsymbol{C}}' + \boldsymbol{\Omega}_t^{-1})^{-1}$
  $\boldsymbol{m}_t = \boldsymbol{m}_{t|t-1} + \boldsymbol{K}_t(\hat{\boldsymbol{y}}_t - \widetilde{\boldsymbol{C}}\boldsymbol{m}_{t|t-1})$
  $\boldsymbol{V}_t = (\boldsymbol{I} - \boldsymbol{K}_t\widetilde{\boldsymbol{C}})\boldsymbol{V}_{t|t-1}$
end
sample $\widetilde{\boldsymbol{x}}_T^* \sim \mathcal{N}(\boldsymbol{m}_T, \boldsymbol{V}_T)$
for $t = T-1, \ldots, 1$ **do**
  $\boldsymbol{J}_t = \boldsymbol{V}_t\widetilde{\boldsymbol{A}}'(\widetilde{\boldsymbol{A}}\boldsymbol{V}_t\widetilde{\boldsymbol{A}}' + \widetilde{\boldsymbol{Q}})^{-1}$
  $\boldsymbol{m}_t^* = \boldsymbol{m}_t + \boldsymbol{J}_t(\boldsymbol{x}_{t+1}^* - \widetilde{\boldsymbol{A}}\boldsymbol{m}_t - \widetilde{\boldsymbol{b}})$
  $\boldsymbol{V}_t^* = (\boldsymbol{I} - \boldsymbol{J}_t\widetilde{\boldsymbol{A}})\boldsymbol{V}_t$
  sample $\boldsymbol{x}_t^* \sim \mathcal{N}(\boldsymbol{m}_t^*, \boldsymbol{V}_t^*)$
end

3. Accept or reject the proposal $\widetilde{\boldsymbol{X}}^*$
compute the acceptance ratio

$$\begin{aligned}\zeta &= \frac{\pi(\widetilde{\boldsymbol{X}}^*|\{\boldsymbol{y}_i\}_{i=1}^n, \boldsymbol{U})}{\pi(\widetilde{\boldsymbol{X}}^{(G-1)}|\{\boldsymbol{y}_i\}_{i=1}^n, \boldsymbol{U})} \frac{q(\widetilde{\boldsymbol{X}}^{(G-1)}|\widetilde{\boldsymbol{X}}^*, \boldsymbol{U})}{q(\widetilde{\boldsymbol{X}}^*|\widetilde{\boldsymbol{X}}^{(G-1)}, \boldsymbol{U})} \\ &= \frac{P(\{\boldsymbol{y}_i\}_{i=1}^n|\widetilde{\boldsymbol{X}}^*)}{P(\{\boldsymbol{y}_i\}_{i=1}^n|\widetilde{\boldsymbol{X}}^{(G-1)})} \frac{NB(\{\boldsymbol{y}_i\}_{i=1}^n|\widetilde{\boldsymbol{X}}^{(G-1)}, \boldsymbol{R})}{NB(\{\boldsymbol{y}_i\}_{i=1}^n|\widetilde{\boldsymbol{X}}^*, \boldsymbol{R})},\end{aligned}$$

where $\boldsymbol{R} = \{r_{it}\}$ is matrix for dispersion parameters for each neuron at all time points. $P(\cdot)$ denotes the Poisson likelihood, and $NB(\cdot)$ denotes the negative binomial likelihood. Accept the proposal $\widetilde{\boldsymbol{X}}^*$ with probability $\min(1, \zeta)$.

proposed method takes the approximation to the full conditional of the latent state $\boldsymbol{X}$ and population baseline $\boldsymbol{\mu}$, the results are similar as to sample directly from the exact full conditional. Table 1 shows

Table 1: PG-MH vs. Gaussian approximation for $\boldsymbol{\mu}$ and $\boldsymbol{X}$

	$\boldsymbol{\mu}$	$\boldsymbol{X}_{*1}$	$\boldsymbol{X}_{*2}$
$\cos\left(\hat{\boldsymbol{\xi}}_{\text{PGMH}}, \boldsymbol{\xi}_{\text{true}}\right)$	0.9724	0.7663	0.9728
$\cos\left(\hat{\boldsymbol{\xi}}_{\mathcal{N}}, \boldsymbol{\xi}_{\text{true}}\right)$	0.9680	0.7443	0.9699
$\cos\left(\hat{\boldsymbol{\xi}}_{\text{PGMH}}, \hat{\boldsymbol{\xi}}_{\mathcal{N}}\right)$	0.9435	0.9664	0.9959

the similarities, defined by cosine function, of fitted $\boldsymbol{\mu}$ and $\boldsymbol{X} = (\boldsymbol{X}_{*1}, \boldsymbol{X}_{*2})$ between the PG-MH and the Gaussian approximation for full conditional, where $\boldsymbol{X}_{*l}$ denotes the l-th latent vectors (i.e. the l-th column of $\boldsymbol{X}$). Here, $\hat{\boldsymbol{\xi}}_{\text{PGMH}}$ denotes the average from iteration 5000 to 10,000 for chain 1 of PG-MH (Fig. 5), $\hat{\boldsymbol{\xi}}_{\mathcal{N}}$ denotes the average from iteration 5000 to 10,000 for the chain in Fig. 2 of the Gaussian approximation, and $\boldsymbol{\xi}_{\text{true}}$ denotes the ground truth. In each column, $\boldsymbol{\xi}$ is replaced by $\boldsymbol{\mu}$, $\boldsymbol{X}_{*1}$ and $\boldsymbol{X}_{*2}$ respectively. The posterior means for these two method are close to each other, i.e. $\cos\left(\hat{\boldsymbol{\xi}}_{\text{PGMH}}, \hat{\boldsymbol{\xi}}_{\mathcal{N}}\right) \approx 1$.

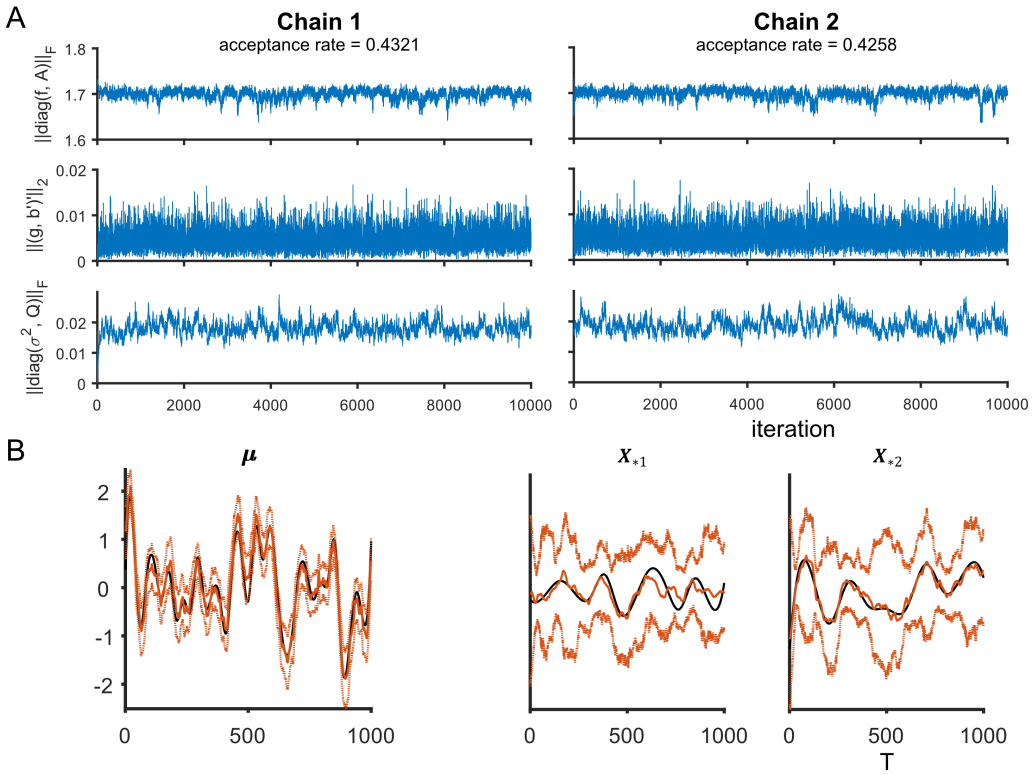

Figure 5: **PG-MH for DPFA** Two independent chains (10,000 iterations) for cluster 1 data in Figure 2. The full conditional distributions of $\boldsymbol{\mu}$ and $\boldsymbol{X}$ are sampled by PG-MH algorithm (Algorithm 1).**A.** Traceplot of Frobenius norms of linear dynamics. The acceptance rates in subtitle are for sampling $\boldsymbol{\mu}$ and $\boldsymbol{X}$ (PG-MH step). **B.** The true (black) and fitted (colored) population baseline and latent factor. Use samples from iteration 5000 to 10,000 in chain 1, the solid orange lines show averages and the dashed lines show 95% HPD intervals.

A.3 Supplementary Results for Neuropixels Application

This section shows supplementary results when applying the proposed model to the Neuropixels dataset (4 Multi-region neural spike recordings). Here, we show 1) results sorted by maximum a posteriori probability (MAP) estimates and 2) clustering results in another independent chain (Fig. 6).

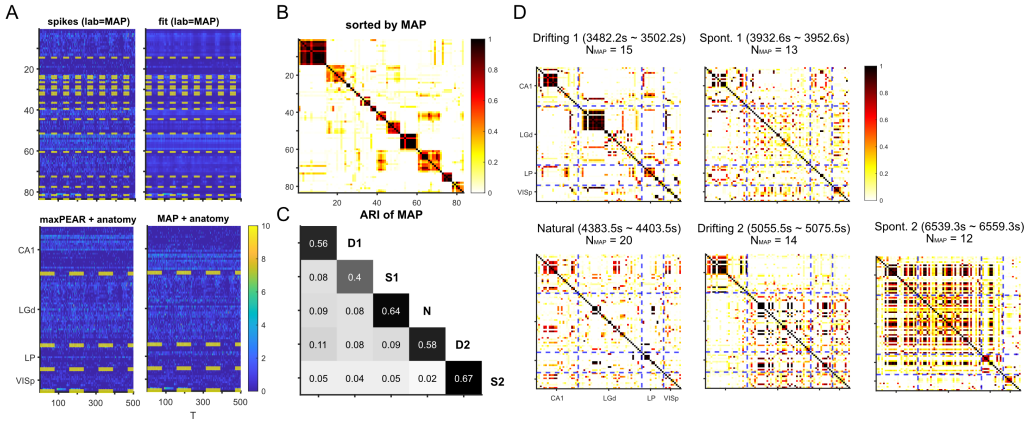

Figure 6: **Supplementary Results for Neuropixels Application A.** The first row shows the spike counts and fitted mean firing rate, sorted according to the MAP label estimates. The second row sort the spikes further according the anatomical sites, which shows clustering structure within each region. **B.** The posterior similarity matrix sorted by MAP estimates. **C.** The ARI of MAP estimates. The diagonal is ARI between 2 chains, and the off-diagonal is mean ARI of MAP for 4 combinations. **D.** Posterior similarity matrices for the second chains. Neurons are sorted as in the first panel of Fig. 4E