Clustering Neural Populations by Poisson Dynamic Factor Model

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Modern recording techniques allow neuroscientists to study multiple neural populations over extended time periods in a large-scale, and the relationships within and between populations are summarized by low-dimensional latent vectors. When the neural activities are globally nonlinear, using single population analysis is inappropriate. However, defining the populations is usually difficult and wrong cluster assignments will lead to bias in latent structure inferences. To tackle this challenge, we develop a clustering method based mixture of Poisson dynamic factor model. The number of cluster is treated as a parameter in mixture of finite mixtures (MFM) model, and the posteriors are sampled by a MCMC algorithm. To sample the posteriors efficiently, we approximate the full conditional distribution of latent state by Gaussian and approximate the marginal likelihood by making use of the Poisson-Gamma conjugacy. We further apply our method to neuropixel data (and hippocampus data) for illustration.

1 Introduction

[TODO]

2 Method

2.1 Poisson Dynmic Factor Model

Denote the observed spike-count of neuron $i \in \{1, \ldots, N\}$ at time bin $t \in \{1, \ldots, T\}$ as $y_{it} \in \mathbb{Z}_{\geq 0}$, and let $\boldsymbol{y}_i = (y_{i1}, \ldots, y_{iT})'$. Further, let $z_i = j$ denote the cluster indicator of neuron i. To facilitate clustering, we re-parametrize the regular Poisson linear dynamical system (PLDS) model [Macke et al., 2011] to separate the mean log-firing-rate out. Assume neurons are independently Poisson distributed, conditional on the low-dimensional latent state $\boldsymbol{x}_t^{(j)} \in \mathbb{R}^p$ as follows:

$$y_{it} \sim Poi(\lambda_{it})$$
$$\log \lambda_{it} = \mu_t^{(j)} + \boldsymbol{c}_i' \boldsymbol{x}_t^{(j)}$$

, with $\boldsymbol{c}_i \sim N(\boldsymbol{0}, \boldsymbol{I}_p)$. We further assume the intercept $\mu_t^{(j)}$ and the latent state $\boldsymbol{x}_t^{(j)}$ progress linearly with Gaussian noise as

$$\begin{aligned}
\mu_1^{(j)} &\sim N(\mu_0, \Sigma_0) \\
\mu_{t+1}^{(j)} &\sim N(f^{(j)}\mu_t^{(j)} + g^{(j)}, \sigma^{2(j)}) \\
\boldsymbol{x}_1^{(j)} &\sim N(\boldsymbol{x}_0, \boldsymbol{Q}_0) \\
\boldsymbol{x}_{t+1}^{(j)} &\sim N(\boldsymbol{A}^{(j)}\boldsymbol{x}_t^{(j)} + \boldsymbol{b}^{(j)}, \boldsymbol{Q}^{(j)})
\end{aligned}$$

The bias term ($\boldsymbol{b}^{(j)}$) in the latent state is constant across time, unlike ind PLDS, which is $\boldsymbol{b}_t^{(j)}$. When the bias term changes across the time, the update order of $\boldsymbol{A}^{(j)}$ and $\boldsymbol{b}_t^{(j)}$ will influence the results (usually, the $\boldsymbol{A}^{(j)}$ is updated first). When ignoring the updating order, the linear dynamics will have infinite solutions. If the $\boldsymbol{A}^{(j)}$ is one solution, then $\boldsymbol{A}^{*(j)} = \boldsymbol{A}^{(j)} + \boldsymbol{M}$ and $\boldsymbol{b}_t^{*(j)} = \boldsymbol{b}_t^{(j)} - \boldsymbol{M}\boldsymbol{x}_t^{(j)}$ will be another set of solution, for any $\boldsymbol{M} \in \mathbb{R}^{p\times p}$. This causes some inconvenience for inference. The same reason for using the time-constant $g^{(j)}$.

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 model can be equivalently written as regular Poisson factor model as

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

However, since the condition $T/N \to 0$ doesn't hold [Johnstone and Lu, 2009], the latent state $\boldsymbol{X}^{(j)}$ cannot be consistently estimated, and assuming linear dynamics of $\boldsymbol{X}^{(j)}$ resolves the problem. Note that when $p > 1$, the model is not unique, since $\boldsymbol{X}^{*(j)} = \boldsymbol{X}^{(j)}\boldsymbol{U}$ also satisfies the equation for any orthogonal matrix $\boldsymbol{U}$ of order p. To ensure the model indefinability, we simply assume $\boldsymbol{A}^{(j)}$ and $\boldsymbol{Q}^{(j)}$ are both diagonal for convenience. See more detailed discussions of the constraints in discussion. Overall, denote the cluster-related parameters of cluster j as $\boldsymbol{\theta}^{(j)} = \{\boldsymbol{\mu}^{(j)}, \boldsymbol{X}^{(j)}, f^{(j)}, g^{(j)}, \sigma^{2(j)}, \boldsymbol{A}^{(j)}, \boldsymbol{b}^{(j)}, \boldsymbol{Q}^{(j)}\}$ and the spike counts of neuron i is generated by Poisson dynamic factor model (PDFM) as $\boldsymbol{Y}_i \sim PDFM(\boldsymbol{\theta}^{(z_i)})$, with the prior of $\boldsymbol{\theta}^{(j)}$ as $\boldsymbol{H}$. The priors for $\{f^{(j)}, g^{(j)}, \sigma^{2(j)}, \boldsymbol{A}^{(j)}, \boldsymbol{b}^{(j)}, \boldsymbol{Q}^{(j)}\}$ are regular normal and inverse-gamma distribution (or multivariate normal and inverse-Wishart when using other non-diagonal constraints).

The marginal likelihood of neuron i is

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

The marginal likelihood has no closed form and will be used for clustering. To help with fast clustering, instead of doing the Laplace approximation, we choose to make use of the Poisson-Gamma conjugacy. This approximation was originally used in El-Sayyad [1973] to derive approximate posterior and the same method was applied to derive other approximations in Chan and Vasconcelos [2009]. This approximation leads to the closed form approximation. By the conditional independency assumption, $M_{\boldsymbol{\theta}^{(j)}}(\boldsymbol{y}_i) = \prod_{t=1}^{T} P(y_{it}|\boldsymbol{\theta}^{(j)})$. Since $\boldsymbol{c}_i \sim N(\boldsymbol{0}, \boldsymbol{I}_p)$, $\lambda_{it} = \exp(\mu_t^{(j)} + \boldsymbol{c}_i'\boldsymbol{x}_t^{(j)}) \sim lognormal(\mu_t^{(j)}, \boldsymbol{x}_t'^{(j)}\boldsymbol{x}_t^{(j)})$. Approximate the lognormal distribution by Gamma distribution, s.t. $lognormal(\mu_t^{(j)}, \boldsymbol{x}_t'^{(j)}\boldsymbol{x}_t^{(j)}) \approx Gamma(a_{it}, b_{it})$ with $a_{it} = (\boldsymbol{x}_t'^{(j)}\boldsymbol{x}_t^{(j)})^{-1}$ and $b_{it} = \boldsymbol{x}_t'^{(j)}\boldsymbol{x}_t^{(j)} \cdot e^{\mu_t^{(j)}}$. Then by Poisson-Gamma conjugacy,

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

, with $r_{it} = a_{it}$ and $p_{it} = 1/(1 + b_{it})$.

Another more general idea is to approximate the log-likelihood by second-order polynomials, with coefficients determined by Chebyshev polynomial approximation Keeley et al. [2019]. However, the approximation doesn't work well in practice, especially when the neural spike counts have a wide range. When doing the integration, we need to exponentiate the log-likelihood and this will exaggerate the approximation error.

2.2 Cluster by Mixture of Finite Mixtures Model

When the label is unknown, we cluster the neurons by mixture models. In practice, it's usually impossible to know the number of neural population. One potential method is to do clustering by Dirichlet process mixtures (DPM) model. However, this is conceptually incorrect, since the number of neural populations is finite but unknown. Besides the conceptual incorrectness, using DPM is not easy to integrate the field knowledge about the number of neural populations. Here, we choose to use the mixture of finite mixtures (MFM, Miller and Harrison [2018]) model as follows

$$\begin{aligned}
K &\sim p_K && \text{where}\, p_K \text{ is a p.m.f. on} \{1, 2, \ldots\} \\
\boldsymbol{\pi} = (\pi_1, \ldots, \pi_k) &\sim Dir_k(\gamma, \ldots, \gamma) && \text{given} K = k \\
Z_1, \ldots, Z_n &\overset{i.i.d.}{\sim} \pi && \text{given} \pi \\
\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k &\overset{i.i.d.}{\sim} \boldsymbol{H} && \text{given} K = k \\
\boldsymbol{Y}_i = (y_{i1}, \ldots, y_{iT})' &\sim SSFM(\boldsymbol{\theta}^{(z_i)}) && \text{independently for}\, i = 1, \ldots, N, \text{given}\, \boldsymbol{\theta}_{1:K} \text{ and } Z_{1:N}
\end{aligned}$$

Besides the conceptual correctness, using MFM model allows us integrate the prior knowledge easily. Moreover, compared to DPM, MFM has some better properties for clustering, for example, MFM posterior on number of cluster is more concentrated and consistent, and MFM tend to give clusters size at the same order of magnitude while DPM may lead to a few large clusters and many small clusters. See Miller and Harrison [2018] for detailed discussion.

2.3 Inference

In this paper, we choose to do inference by MCMC. Because of the Poisson likelihood, the latent state $\boldsymbol{X}^{(j)}$has no closed full conditional distribution. We can sample the posterior by particle MCMC directly, but this can be slow. However, due to the Markovian structure of the model, the conditional log-posterior is concave and its Hessian is block-tridiagonal. Thus, we can do the global Laplace approximation efficiently in $\mathcal{O}(T)$ [Paninski et al., 2010].The cluster index and number of cluster are sampled by the analog of partition-based algorithm in DPM [Neal, 2000]. See details of the MCMC update in the Appendix.

In practice, using variational Bayes (VB) instead of MCMC may be more favorable. The PLDS can be updated by variational EM. Using the stick-breaking representation of MFM model, we can do VB easily similar to Blei and Jordan [2006]. However, checking by the "gold standard" MCMC before doing VB is always a good choice.

For the dimensionality p of of the latent state, we can treat it as a parameter and sample the posterior by RJMCMC as in Lopes and West [2004] or borrow the idea of adaptive Gibbs sampling with shrinkage prior [Bhattacharya and Dunson, 2011]. Here, we simply pre-set the p or select the optimized value by the cross-validation, which can be easily conducted when switching to the deterministic algorithm in the future.

3 Simulations

3.1 Model Global Non-linearity by Clustering

There were a rich research results for single PLDS model, but it provides only a global linear model to represent the data in a lower dimensional subspace, which makes the application scope limited. When the input space is nonhomogeneous, a large dimension of latent state is needed and this may lead to the overfitting and poor performance. Mixture of PLDS/ PDFM models allow us to partition the input space into clusters and can therefore capture global nonlinearity by combining local linear models.

To show the connection between the proposed model and PLDS parametrization, we simulate the data using the PLDS model, although these two are equivalent after some algebra. Denote the common PLDS model as $\log \lambda_i = \delta_i \mathbf{1}_T + \boldsymbol{G}^{(j)} \boldsymbol{d}_i$, where $d_i \in \mathbb{R}^q$. In PLDS parametrization, the latent state has one more dimension, i.e. $p = q + 1$.

We first simulated three neural populations, with 10 neurons in each. We set $q = 2$ for each cluster and the recording length is T=1000. The $\delta_i = 1$ for $i = 1, \ldots, 10$ and is 0 for the remaining neurons. The $\boldsymbol{G}^{(j)}$ is $0.1(1, \ldots, 10)' \otimes \mathbf{1}_2' + \mathbf{1}_{10} \otimes (-1, 2)$ to set up a gradient for the loading in each cluster and to differ the contribution of two latent vectors. We allow $\boldsymbol{A}^{(j)}$ be non-diagonal and possible interactions between clusters, which is used in single population PLDS model. The diagonal element is generated from $N(0.92, 0.1^2)$. The bias terms are $\boldsymbol{b}^{(j)}$ is $10^{-3}\mathbf{1}_2$ for the first 2 clusters and $10^{-4}\mathbf{1}_2$ for the last cluster. The $\boldsymbol{Q}^{(j)}$ are $10^{-3}\boldsymbol{I}_2$, $10^{-4}\boldsymbol{I}_2$ and $10^{-5}\boldsymbol{I}_2$.

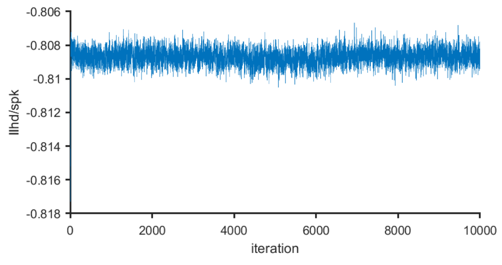
The Figure 1(a) shows the trace plot for the log-likelihood per spike, up to 10,000 iterations. The plot shows (Figure 1(b)) that the convergence achieved after several steps. Therefore, we show the results by averaging iteration 500 to 1000. Figure 1(c) shows the posterior mean of the mean firing rate, while Figure 1(d) shows the fitted latent state. The latent vectors are transformed to PLDS parametrization for comparison. Notice that the PLDS parametrization also doesn't have the unique solution, we let $\boldsymbol{G}^{(j)}$ has mean zeros columns and is orthogonal.

Then we held out $1/4$ and $1/2$ data as test set in a "speckled" pattern, i.e. randomly select subset of data for each neuron as held-out dataset. The "speckled" cross-validation was previously used in Williams et al. [2020]. We then fit the model with and without clusters, keeping the same latent dimension, i.e. (1) 3 clusters with $p = 1$ for each and (2) 1 cluster with $p = 2 \cdot 3 - 1 = 5$. The procedure is replicated for 100 times for two proportions, and the difference of held-out likelihood per spike between 2 fittings are shown in Figure 1(e). The difference between (1) and (2) are always positive, and this shows doing the mixture of PDFM performs better than single PDFM. Moreover, as the proportion of training decrease (less data), the benefit of clustering becomes more significant. This suggests that doing clustering is necessary.

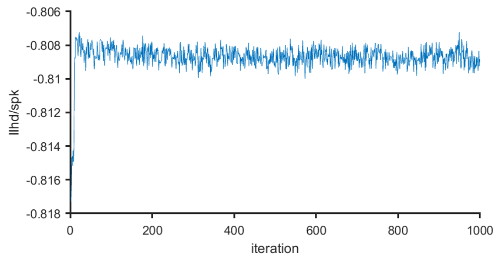
3.1.1 Clustering

Then use the same setting, we remove the label and use the mixture model to do the clustering by MFM. The prior of the cluster number is $K \sim Geometric(0.2)$. Figure 1(f) shows the trace of label z_i for each neuron for the first 100 iterations.

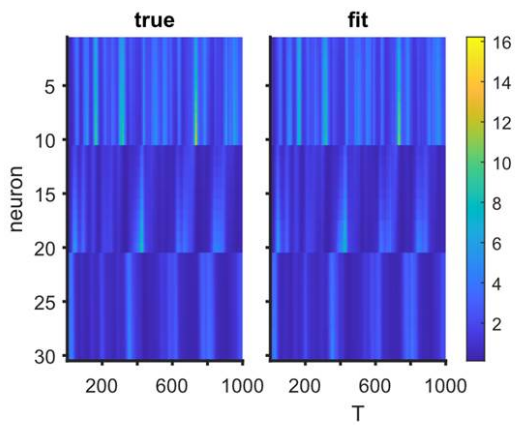
In this toy simulation example, the signal is strong and the spiking amplitudes of neurons in each population are similar. This might be too simplified for real situation. In practice, the neural activity is usually sparse and can be recorded in high resolution. Moreover, the spiking amplitudes for neuron in one nominated population vary a lot, and hence may form a few subpopulations. Therefore, we then simulate another more realistic example. In this simulation, there are 3 clusters and the dimension of latent state is $p = 1$ for each. However, there are 20 neurons with wider range but smaller value of loading c_i, corresponding to potential subpopulations and weak signal. The simulated firing rate is shown in Figure 2(a). The trace plot of number of cluster (Figure 2(b)) shows that the model tend to further split some clusters into sub-population, based on the spiking amplitude. This is more clearly shown in the posterior similarity matrix ((Figure 2(c)), averaging from iteration 500 to 1000. In similarity matrix, the entry (i, j) is the posterior probability that data points i and j belong to the same cluster. The rows and columns are ordered according to the ground truth. The similarity matrix shows that the model tend to split cluster 1 and 3 into two sub-clusters.



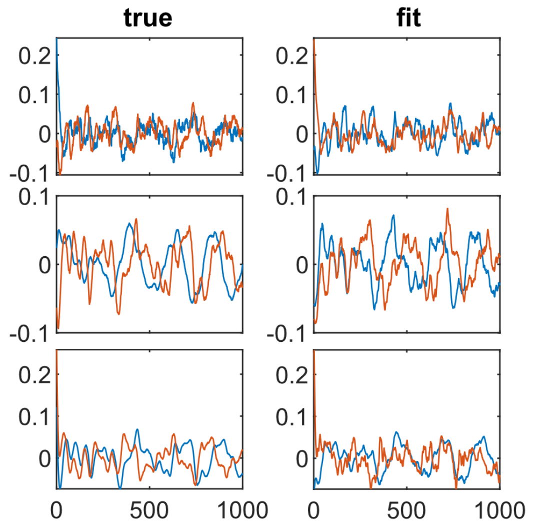
(a) Trace plot of log-likelihood per spike



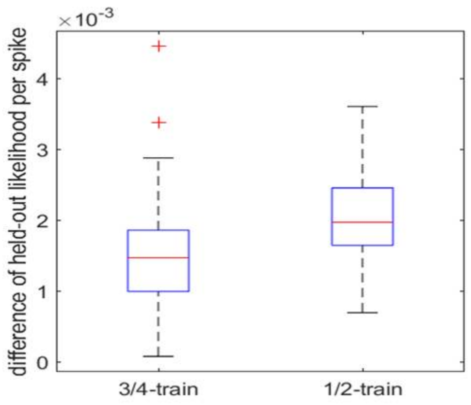
(b) Trace plot of log-likelihood per spike, truncated (1000)



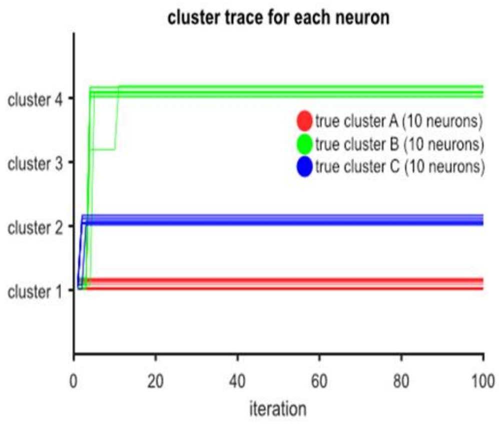
(c) true vs. fitted firing rate



(d) true vs. fitted latent vectors



(e) cross-validation: clustered vs. single population



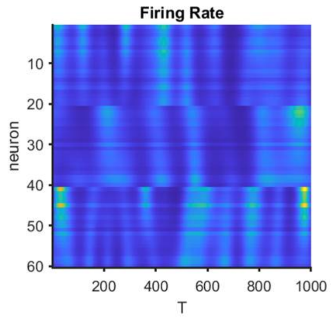
(f) trace plot for non-labeled case

Figure 1: Simple case

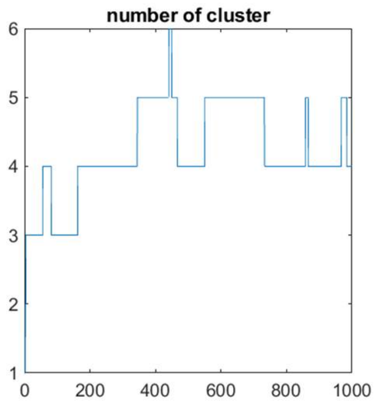
4 Application

4.1 Neuropixels data

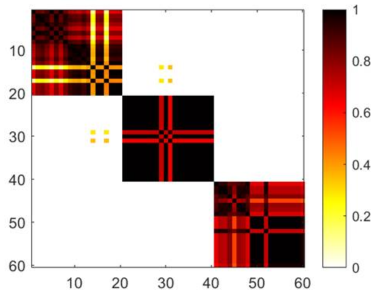
[TODO]: brief introduction of Neuropixels dataset.



(a) Firing rate



(b) trace plot for cluster number



(c) similarity matrix

Figure 2: More complicated case: more variety of amplitude & weaker signal

The Neuropixels dataset contains recording of neural activities in different brain regions of mouse. Here, we only consider neurons with SNR > 3 and the spiking counts > 1000. Then, we first use the recording activity from Lateral posterior nucleus of the thalamus (LP, 20 neurons), anteromedial visual area (VISam, 12 neurons) and ventral posteromedial nucleus of the thalamus (VPM, 14 neurons) during the spontaneous period. The bin size is 0.1s and truncate the data up to 1000 steps (100s recording) for convenience.

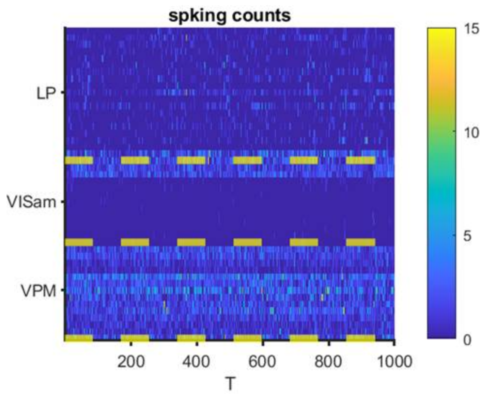
Figure 3(a) shows the spiking counts of these 46 neurons. Here we first fit model using all data, with $p = 1$ and $K \sim Geometric(0.3)$. For the number of clusters, Figure 3(b) shows the trace plot for the first 5000 iterations. All the following results are from averaging samples from iteration 2500 to 5000. The Figure 3(c) shows the histogram of the posterior samples. These plots show that these neurons are quite non-homogeneous and they tend to form many sub-populations. These 3 nominated populations by anatomy tend to form around 10 (sub)-clusters.

The Figure 3(d) and Figure 3(e) show the trace of log-likelihood per spike. The fitted mean firing rate is shown in Figure 3(f). When investigating the similarity matrix in Figure 3(g) (sorted according to the anatomical population, such that 1-20 are LP, 21-32 are VISam and 33-46 are VPM), there are several subpopulations in each anatomical cluster. Generally, the clustering results are consistent with the anatomical assignment, although there are some tiny confusions between VISam and VPM.

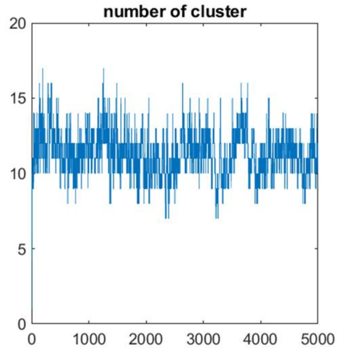
However, this is not the case when the mouse is in the experiment. We choose the same set of neurons but use recordings when the mouse are watching drift gratings. The Figure 3(h) shows the trace plot of cluster number. The Figure 3(i) and Figure 3(j) show the fitted firing rate and similarity matrix, again averaging from iteration 2500 to 5000.

When comparing the observed spiking activities, it's fairly hard to see differences when the mouse is exposed to the stimuli and not. However, the posterior similarity matrix in the experiment shows that there's more confusion between LP and VISam, compared to the spontaneous response. This suggests that the functional connection pattern between neurons in different areas when the stimuli are changed.

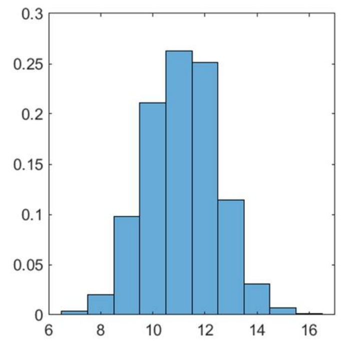
I tried to held-out half of the data in a speckled pattern and fit the model with clustering on and off (single population). The dimension is selected by held-out likelihood, which is p=1 for clustering on and p=2 for single population analysis. The trace plots of the held-out log-likelihood per spike (starting from iteration 2) shows that doing clustering doesn't improve things (although they are very close)… Maybe because "turning clustering on" introduce too much variance… The benefit of clustering will pop out if we use the deterministic algorithm, such as VB mentioned.



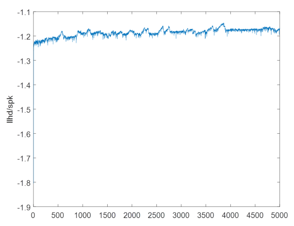
(a) spontaneous: spike counts



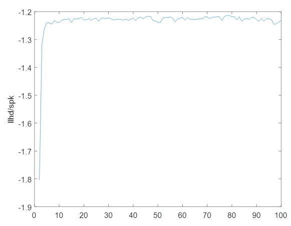
(b) spontaneous: cluster number trace



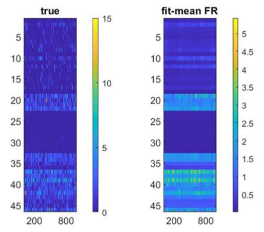
(c) spontaneous: cluster number histogram



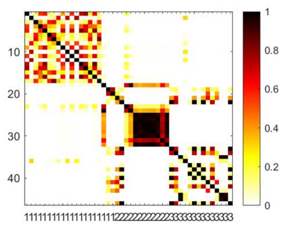
(d) spontaneous: llhd/spk trace



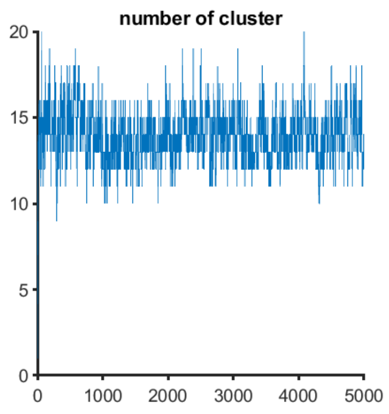
(e) spontaneous: llhd/spk trace truncated



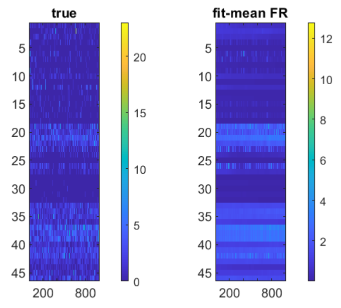
(f) spontaneous: true vs. fitted firing rate



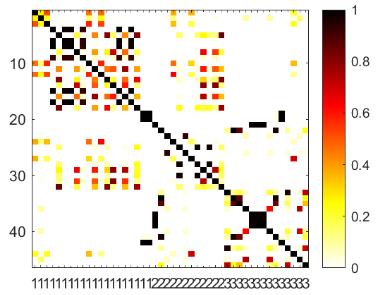
(g) spontaneous: similarity matrix



(h) drift-grating: cluster number trace



(i) drift-grating: true vs. fitted firing rate



(j) drift-grating: similarity matrix

Figure 3: Neuropixel

4.2 Hippocampus data

I prefer just give one application. Because 1) there's a page limit (9 pages excluding ref & appendix in 2021, and now it seems we will exceed the limit), and 2) the V1 data is enough to illustrate the idea If we really need to add the second example, we should tell the story in another angle.

5 Discussion

As mentioned above, the factor model doesn't have unique solution. Although this issue had been thoroughly discussed in statistics [reference], it has been ignored in some neuroscience research. Since the (P)LDS related model are usually fitted by deterministic algorithms such as variants of

EM, the issue is kind of hard to detect. And because of the ignorance, there are some problematic comments on linear dynamics $\boldsymbol{A}$ and $\boldsymbol{Q}$. When using the variants of factor model, e.g. (P)LDS and GPFA, we should only focus on the “shape” of latent state but not the specific values of them.

In this paper, to ensure the unique solution, we put the diagonal constraints in $\boldsymbol{A}^{(j)}$ and $\boldsymbol{Q}^{(j)}$ are used for convenience. However, only constraining on $p(p-1)/2$ s enough [reference]. Since we the label is unknown and will be switched when doing clustering, it’s inappropriate and convenient to put constraint on $\boldsymbol{c}_i$. Therefore, we instead treat $\boldsymbol{X}^{(j)}$ as the "loading" and can put constraints on it. In traditional Gaussian factor model, there are two equivalent types of constraints: (1) diagonal constraint: $\boldsymbol{X}'^{(j)}\boldsymbol{X}^{(j)}$ is diagonal; (2) block lower triangular constraint [Fokoué and Titterington, 2003]: The first p rows of $\boldsymbol{X}^{(j)}$ is lower triagular and the diagonal elements are positive. Since we assume $\boldsymbol{X}^{(j)}$ has linear progression, using the diagonal $\boldsymbol{X}'^{(j)}\boldsymbol{X}^{(j)}$ constraint is more appropriate. This constraint can be easily achieved when conducting deterministic optimization algorithms (e.g. the VB mentioned in “method-inference” section). This can also be done in MCMC, by turning off the reflection of the projection (e.g. let the diagonal elements in the projection matrix be positive). Using diagonal $\boldsymbol{X}'^{(j)}\boldsymbol{X}^{(j)}$ leads to similar results.

Moreover, the idea of doing clustering by mixture model can be extended beyond the Poisson distribution. We can further include other information, such as, the dispersion information by assuming negative-binomial distributed or even Conway-Maxwell-Poisson distributed neural spikes [reference to our paper in CMP]. This further information can be used for more detailed clustering, by expanding the state space.

Acknowledgments

Broader Impact

References

Jakob H. Macke, Lars Buesing, John P. Cunningham, Byron M. Yu, Krishna V. Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. *Advances in Neural Information Processing Systems*, 24, 2011.

Iain M. Johnstone and Arthur Yu Lu. On Consistency and Sparsity for Principal Components Analysis in High Dimensions. *Journal of the American Statistical Association*, 104(486):682, jun 2009. ISSN 01621459. doi: 10.1198/JASA.2009.0121.

G. M. El-Sayyad. Bayesian and Classical Analysis of Poisson Regression. *Journal of the Royal Statistical Society: Series B (Methodological)*, 35(3):445–451, jul 1973. ISSN 2517-6161. doi: 10.1111/J.2517-6161.1973.TB00972.X.

Antoni B. Chan and Nuno Vasconcelos. Bayesian poisson regression for crowd counting. *Proceedings of the IEEE International Conference on Computer Vision*, pages 545–551, 2009. doi: 10.1109/ICCV.2009.5459191.

Stephen L. Keeley, David M. Zoltowski, Yiyi Yu, Jacob L. Yates, Spencer L. Smith, and Jonathan W. Pillow. Efficient non-conjugate Gaussian process factor models for spike count data using polynomial approximations. *37th International Conference on Machine Learning, ICML 2020*, PartF16814:5133–5142, jun 2019.

Jeffrey W. Miller and Matthew T. Harrison. Mixture models with a prior on the number of components. *Journal of the American Statistical Association*, 113(521):340, jan 2018. doi: 10.1080/01621459.2016.1255636.

Liam Paninski, Yashar Ahmadian, Daniel Gil Ferreira, Shinsuke Koyama, Kamiar Rahnama Rad, Michael Vidne, Joshua Vogelstein, and Wei Wu. A new look at state-space models for neural data, aug 2010. ISSN 09295313. URL `https://link.springer.com/article/10.1007/s10827-009-0179-x`.

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

David M Blei and Michael I Jordan. Variational Inference for Dirichlet Process Mixtures. *Bayesian Analysis*, 1 (1):121–144, 2006. URL `http://www.cs.berkeley.edu/$\sim$blei/`.

Hedibert Freitas Lopes and Mike West. Bayesian Model Assessment in Factor Analysis. *Statistica Sinica*, 14: 41–67, 2004.

A. Bhattacharya and D. B. Dunson. Sparse Bayesian infinite factor models. *Biometrika*, 98(2):291, jun 2011. ISSN 00063444. doi: 10.1093/BIOMET/ASR013.

Alex H. Williams, Anthony Degleris, Yixin Wang, and Scott W. Linderman. Point process models for sequence detection in high-dimensional neural spike trains. *Advances in Neural Information Processing Systems*, 2020-Decem, oct 2020. ISSN 10495258.

Ernest Fokoué and D. M. Titterington. Mixtures of factor analysers. Bayesian estimation and inference by stochastic simulation. *Machine Learning*, 50(1-2):73–94, jan 2003. ISSN 08856125. doi: 10.1023/A:1020297828025.

Uri T Eden, Loren M. Frank, Riccardo Barbieri, Victor Solo, and Emery N. Brown. Dynamic Analysis of Neural Encoding by Point Process Adaptive Filtering. *Neural Computation*, 16(5):971–998, may 2004. ISSN 0899-7667. doi: 10.1162/089976604773135069. URL `https://doi.org/10.1162/089976604773135069`.

H E Rauch, F Tung, and C T Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, aug 1965. ISSN 0001-1452. doi: 10.2514/3.3166. URL `https://doi.org/10.2514/3.3166`.

Matthew D. Hoffman and Andrew Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15:1593–1623, nov 2011. ISSN 15337928. URL `https://arxiv.org/abs/1111.4246v1`.

A Appendix

A.1 MCMC updates

The posteriors are sampled by a Gibbs sampler, the full conditional distributions, if available, are shown as follows:

Update $x_t^{(j)}$ and $\mu_t^{(j)}$: Let the t^{th} column observation and firing rate of the j^{th} cluster be $\widetilde{\boldsymbol{y}}_t = vec\left(\{y_{it}^{(j)}|z_i = j\}\right)$ and $\widetilde{\boldsymbol{\lambda}}_t^{(j)} = vec(\{\lambda_{it}|z_i = j\})$. The number of neurons in cluster j is $n_j = |\{i : z_i = j\}|$. The corresponding loading for these n_j neurons is $\boldsymbol{C}^{(j)} \in \mathbb{R}^{n_j \times p}$, such that $\log \widetilde{\boldsymbol{\lambda}}_t^{(j)} = \mu_t^{(j)} \mathbf{1}_{n_j} + \boldsymbol{C}^{(j)} \boldsymbol{x}_t^{(j)} = \left(\mathbf{1}_{n_j}, \boldsymbol{C}^{(j)}\right) \cdot \left(\mu_t^{(j)}, \boldsymbol{x}_t^{'(j)}\right)'$. Denote $\widetilde{\boldsymbol{C}}^{(j)} = \left(\mathbf{1}_{n_j}, \boldsymbol{C}^{(j)}\right)$, $\widetilde{\boldsymbol{x}}_t^{(j)} = \left(\mu_t^{(j)}, \boldsymbol{x}_t^{'(j)}\right)'$, $\widetilde{\boldsymbol{x}}^{(j)} = \left(\widetilde{\boldsymbol{x}}_1^{'(j)}, \ldots, \widetilde{\boldsymbol{x}}_T^{'(j)}\right)'$, $\widetilde{\boldsymbol{A}}^{(j)} = diag(f^{(j)}, \boldsymbol{A}^{(j)})$, $\widetilde{\boldsymbol{b}}^{(j)} = \left(g^{(j)}, \boldsymbol{b}^{'(j)}\right)'$ and $\widetilde{\boldsymbol{Q}}^{(j)} = diag(\sigma^{2(j)}, \boldsymbol{Q}^{(j)})$. The full conditional distribution $P(\widetilde{\boldsymbol{x}}^{(j)}|\ldots) = P(\widetilde{\boldsymbol{x}}^{(j)}|\{\widetilde{\boldsymbol{y}}_t\}_{t=1}^T, \widetilde{\boldsymbol{C}}^{(j)}, \widetilde{\boldsymbol{A}}^{(j)}, \widetilde{\boldsymbol{b}}^{(j)}, \widetilde{\boldsymbol{Q}}^{(j)})$ is approximated by a global Laplace approximation, i.e. $P(\widetilde{\boldsymbol{x}}^{(j)}|\ldots) \approx N_{(p+1)T}(\widetilde{\boldsymbol{x}}^{(j)}|\boldsymbol{\mu}_{\widetilde{\boldsymbol{x}}^{(j)}}, \boldsymbol{\Sigma}_{\widetilde{\boldsymbol{x}}^{(j)}})$, with $\boldsymbol{\mu}_{\widetilde{\boldsymbol{x}}^{(j)}} = \arg\max_{\widetilde{\boldsymbol{x}}^{(j)}} P(\widetilde{\boldsymbol{x}}^{(j)}|\ldots)$ and $\boldsymbol{\Sigma}_{\widetilde{\boldsymbol{x}}^{(j)}}) = -\left(\nabla\nabla \log P(\widetilde{\boldsymbol{x}}^{(j)}|\ldots)\big|_{\widetilde{\boldsymbol{x}}^{(j)}=\boldsymbol{\mu}_{\widetilde{\boldsymbol{x}}^{(j)}}}\right)^{-1}$. The log full conditional distribution $h(\widetilde{\boldsymbol{x}}^{(j)}) = \log P(\widetilde{\boldsymbol{x}}^{(j)}|\ldots)$ is given by:

$$\begin{aligned} h = \text{const} + \sum_{t=1}^{T} \left(\widetilde{\boldsymbol{y}}_t' \widetilde{\boldsymbol{C}}^{(j)} \widetilde{\boldsymbol{x}}_t^{(j)} - \widetilde{\boldsymbol{\lambda}}_t\right) - \frac{1}{2}(\widetilde{\boldsymbol{x}}_1^{(j)} - \widetilde{\boldsymbol{x}}_0^{(j)})' \widetilde{\boldsymbol{Q}}_0^{(j)} (\widetilde{\boldsymbol{x}}_1^{(j)} - \widetilde{\boldsymbol{x}}_0^{(j)}) \\ - \sum_{t=2}^{T} \frac{1}{2}(\widetilde{\boldsymbol{x}}_t^{(j)} - \widetilde{\boldsymbol{A}}^{(j)} \widetilde{\boldsymbol{x}}_{t-1}^{(j)} - \widetilde{\boldsymbol{b}}^{(j)})' \widetilde{\boldsymbol{Q}}^{(j)} (\widetilde{\boldsymbol{x}}_t^{(j)} - \widetilde{\boldsymbol{A}}^{(j)} \widetilde{\boldsymbol{x}}_{t-1}^{(j)} - \widetilde{\boldsymbol{b}}^{(j)}) \end{aligned}$$

, where "const" represents the constant. The $h(\widetilde{\boldsymbol{x}}^{(j)})$ is concave and hence unimodal, and the Markovian structure of the latent dynamics, and hence the tri-block-diagonal Hessian, makes it possible to compute a Newton update in $\mathcal{O}(T)$ [Paninski et al., 2010]. The same technique is used in the E-step for PLDS model [Macke et al., 2011].

To facilitate convergence, we use a smoothing estimate with local Gaussian approximation as a "warm start". The forward filtering for a dynamic Poisson model has been previously described in Eden et al. [2004]. Because the approximated filtering estimates are Gaussian distributed, we can further find the smoothing estimates using a backward pass as in Rauch et al. [1965].

Update $\boldsymbol{c}_i$: When writing the observation mapping in matrix form, $\log \boldsymbol{\lambda}_i = \boldsymbol{\mu}^{(j)} + \boldsymbol{X}^{(j)} \boldsymbol{c}_i$, the update of $\boldsymbol{c}_i$ reduces to a regular Poisson regression problem, given $\boldsymbol{\mu}^{(j)}$ and $\boldsymbol{X}^{(j)}$ are known. Here, we sample the posterior by no a No U-Turn Sampler (NUTS, Hoffman and Gelman [2011]), within the Gibbs sampler.

Update Linear dynamics of latent state: The parameters for linear dynamics are $f^{(j)}$, $g^{(j)}$, $\sigma^{2(j)}$, $\boldsymbol{A}^{(j)}$, $\boldsymbol{b}^{(j)}$ 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 $\boldsymbol{Q}^{(j)}$ dimension-by-dimension, as the update in $f^{(j)}$, $g^{(j)}$ and $\sigma^{2(j)}$. Use the priors $\sigma^{2(j)} \sim IG\left(\nu_0/2, \nu_0\sigma_0^2/2\right)$ and $\left(g^{(j)}, f^{(j)}\right)' \sim N(\boldsymbol{\mu}_0, \sigma^{2(j)} \boldsymbol{\Lambda}_0^{-1})$. The prior μ_0 is specified as $(0,1)'$. Denote $\boldsymbol{y}_{\mu^{(j)}} = \left(\mu_2^{(j)}, \ldots, \mu_T^{(j)}\right)'$ and $\boldsymbol{X}_{\mu^{(j)}} = \left(\mathbf{1}_{T-1}, \boldsymbol{\mu}_{1:(T-1)}^{(j)}\right)$, with $\boldsymbol{\mu}_{1:(T-1)}^{(j)} = \left(\mu_1^{(j)}, \ldots, \mu_{T-1}^{(j)}\right)'$. The full conditional distributions are $\sigma^{2(j)}|\ldots \sim IG\left(\frac{\nu_0+T-1}{2}, \frac{\nu_0\sigma_0^2 + \boldsymbol{y}_{\mu^{(j)}}' \boldsymbol{y}_{\mu^{(j)}} + \boldsymbol{\mu}_0' \boldsymbol{\Lambda}_0 \boldsymbol{\mu}_0 - \boldsymbol{\mu}_n' \boldsymbol{\Lambda}_n \boldsymbol{\mu}_n}{2}\right)$ and $\left(g^{(j)}, f^{(j)}\right)'|\ldots \sim N(\boldsymbol{\mu}_n, \sigma^{2(j)} \boldsymbol{\Lambda}_n^{-1})$, with $\boldsymbol{\Lambda}_n = \boldsymbol{X}_{\mu^{(j)}}' \boldsymbol{X}_{\mu^{(j)}} + \boldsymbol{\Lambda}_0$ and $\boldsymbol{\mu}_n = \boldsymbol{\Lambda}_n^{-1}\left(\boldsymbol{X}_{\mu^{(j)}}' \boldsymbol{X}_{\mu^{(j)}} \left(\boldsymbol{X}_{\mu^{(j)}}' \boldsymbol{X}_{\mu^{(j)}}\right)^{-1} \boldsymbol{X}_{\mu^{(j)}}' \boldsymbol{y}_{\mu^{(j)}} + \boldsymbol{\Lambda}_0 \boldsymbol{\mu}_0\right)$. The updates of $\boldsymbol{A}^{(j)}$, $\boldsymbol{b}^{(j)}$ and $\boldsymbol{Q}^{(j)}$ are similar, with diagonal assumption.

Update z_i: To update the cluster assignments for each neuron i, we modify a partition based algorithm in DPM for MFM. Because of the non-conjugacy, the marginal likelihood in terms of all cluster parameters θ_{z_i} can not be easily computed. This issue can be solved by introducing auxiliary variable, as in "Algorithm 8" in Neal [2000] for DPM. We can use the same idea for inference in the MFM by two substitutions: 1) replace $|c_i|$ by $|c_i| + \gamma$ and 2) replace α by $\gamma V_n(t+1)/V_n(t)$. Here, t is the number of partition obtained by removing the neuron i. The $V_n(t) = \sum_{k=1}^{\infty} \frac{k_{(t)}}{(\gamma k)^{(n)}} p_K(k)$, with $x^{(m)} = x(x+1)\cdots(x+m-1)$, $x_{(m)} = x(x-1)\cdots(x-m+1)$, $x^{(0)} = 1$ and $x_{(0)} = 1$. See details in Miller and Harrison [2018]