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 $i \in \{1, \dots, N\}$ at time bin $t \in \{1, \dots, T\}$ as $y_{it} \in \mathbb{Z}_{\geq 0}$, and let $\boldsymbol{y}_i = (y_{i1}, \dots, y_{iT})'$. 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 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:

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

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

, 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

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

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 [**?**], 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 $\widetilde{\boldsymbol{X}}^{(j)} = \boldsymbol{X}^{(j)}\boldsymbol{U}$ 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 $\boldsymbol{\theta}^{(j)} = \{\boldsymbol{\mu}^{(j)}, \boldsymbol{X}^{(j)}, f^{(j)}, g^{(j)}, \Sigma^{(j)}, \{\boldsymbol{A}^{(j)}, \{\boldsymbol{b}^{(j)}, \{\boldsymbol{Q}^{(j)}\}$ 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 $\{f^{(j)}, g^{(j)}, \Sigma^{(j)}, \{\boldsymbol{A}^{(j)}, \{\boldsymbol{b}^{(j)}, \{\boldsymbol{Q}^{(j)}\}$ 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 [reference] 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 in 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.

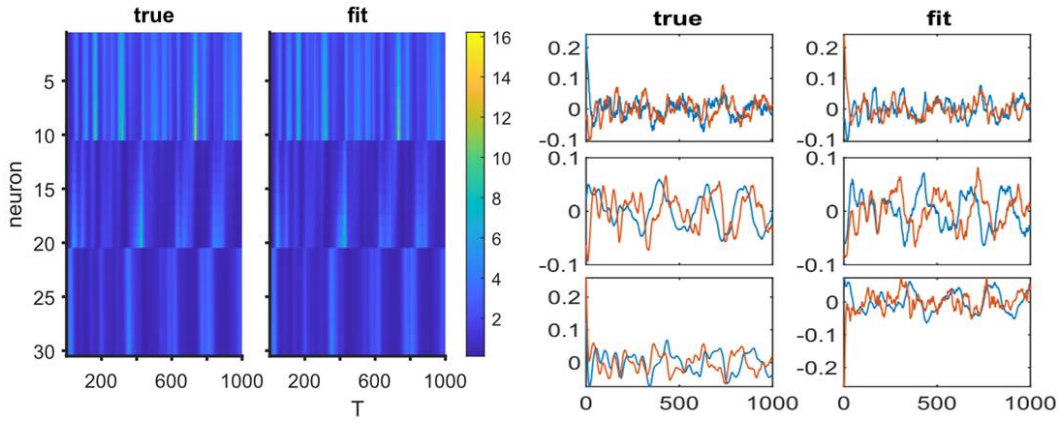
3 Simulations

3.1 Model Global Non-linearity by Clustering

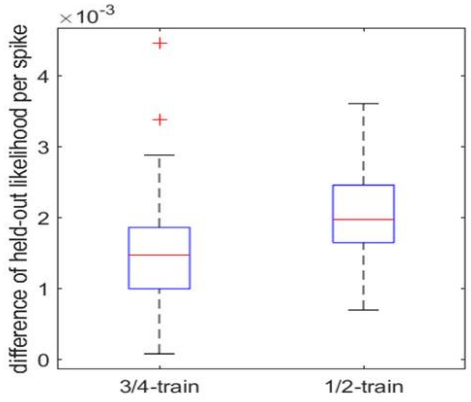
There were a rich research results for single PLDS model, but it provides only a global linear model to represent the data in a lower dimensional subspace, which makes the application scope limited. When the input space is nonhomogeneous, a large dimension of latent state is needed and this may lead 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.

We first simulated three neural populations, with 10 neurons in each. We set p=1 for each cluster and the recording length is T=1000. After checking the trace plots up to 10,000 iterations, the convergence achieved after several steps. The panel A and B in figure 1 show the posterior mean firing rate and fitted latent state, averaging from iteration 500 to 1000. The latent state is transformed into the

commonly used PLDS model: $\log \lambda_i = \delta_i \mathbf{1}_T + \boldsymbol{G}^{(j)} \boldsymbol{d}_i$, where $d_i \in \mathbb{R}^q$. In PLDS parametrization, the latent state has one more dimension, i.e. $p = q + 1$. The PLDS parametrization also doesn't have the unique solution. For comparison, we let $\boldsymbol{G}^{(j)}$ has mean zeros columns and is orthogonal.



(a) a (b) b



(c) c

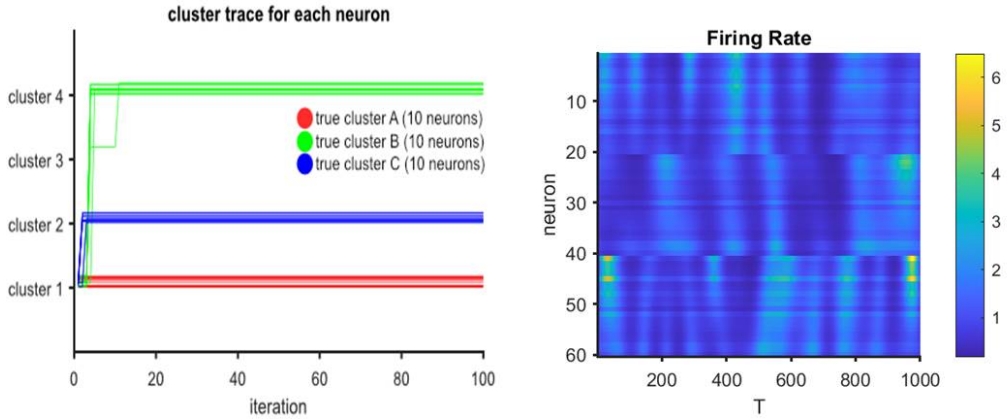
Figure 1: Model Global Non-linearity by Clustering

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. Then we 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 panel C in figure 1. 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

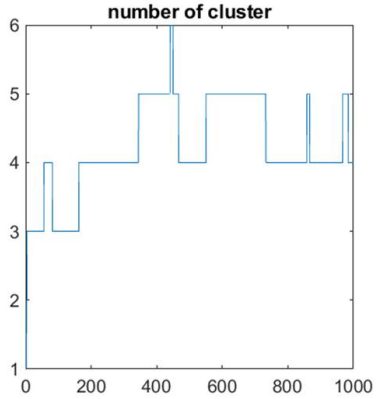
Then use the same setting, we remove the label and use the mixture model to do the clustering by MFM. The prior of the cluster number is $K \sim Geometric(0.2)$. The panel A of figure 2 shows the trace of label z_i for each neuron for the first 100 iterations.

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 amplitude for neuron in one population vary a lot. Therefore, we simulate another more realistic example. In this simulation,

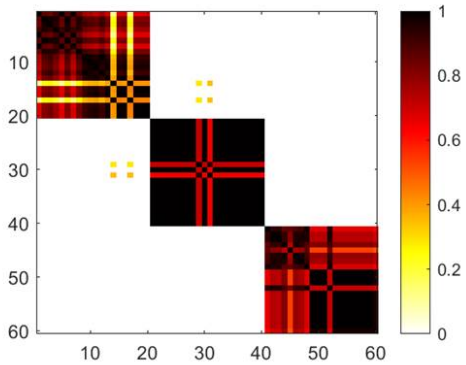


(a) a

(b) b



(c) c



(d) c

Figure 2: Clustering

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. The simulated firing rate is shown in panel B in figure 2. The trace plot of number of cluster (panel C in figure 2) shows that the model tend to further split some clusters into sub-population, based on the spiking amplitude. The similarity matrix (panel D in figure 2) of posterior, averaging from iteration 500 to 1000, shows that the model tend to split cluster 1 and 3 into two sub-clusters.

4 Application

4.1 Neuropixels data

[TODO]: brief introduction of Neuropixels dataset.

The Neuropixels dataset contains recording of neural activities in different brain regions, when doing the drift-grating experiment. Here, we only consider neurons with SNR > 3 and the spiking counts > 1000. Then, we 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.

Panel A in figure 3 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, panel B in figure 2

shows the trace plot for the first 5000 iterations and panel C shows the histogram of iteration 2500 to 5000. These plots show that these neurons are quite non-homogenous and they tend to form many sub-populations.The posterior similarity matrix, averaging from iteration 2500 to 5000 (panel D in figure 3), shows that there are several sub-populations in each anotomical cluster, and there are some tiny confusions between VISam and VPM.

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... 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.

4.2 Hippocampus data

What kind of story do I need to tell here? This should be different from the story from Neuropixel data. Maybe focus on the potential improvement of held-out likelihood after clustering? But I guess similar problem will happen. When fitting data with 84 neurons, the number of component jumps around 10.

5 Discussion

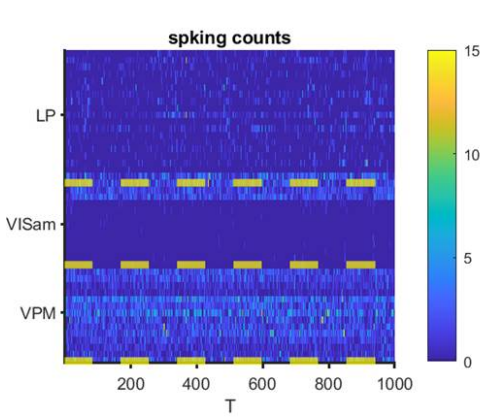
As mentioned above, the factor model doesn't have unique solution. Although this issue had been thoroughly discussed in statistics [reference], it has been ignored in some neuroscience research. Since the (P)LDS related model are usually fitted by deterministic algorithms such as variants of EM, the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

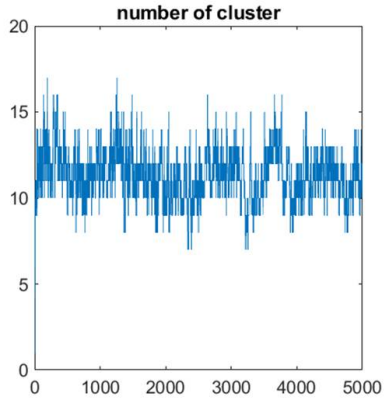
Moreover, the idea of doing clustering by mixture model can be extended beyond the Poisson distribution. We can further include other information, such as, the dispersion information by assuming negative-binomial distributed or even Conway-Maxwell-Poisson distributed neural spikes [reference to our paper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

Acknowledgments

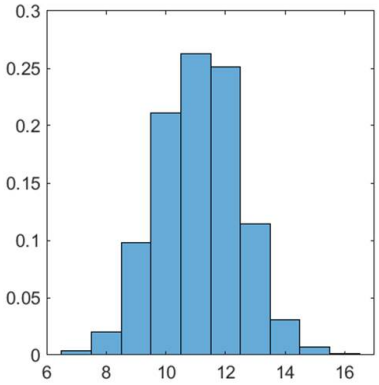
Broader Impact



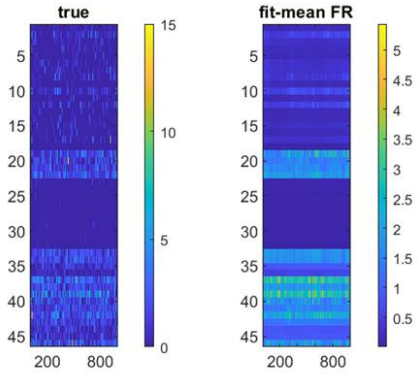
(a) a



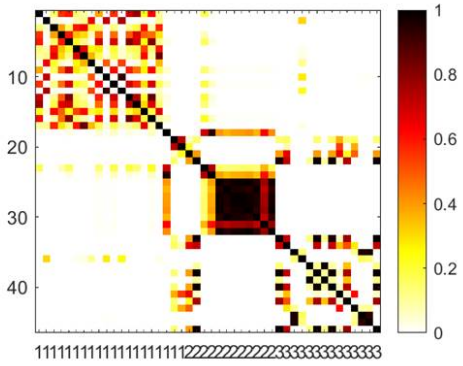
(b) b



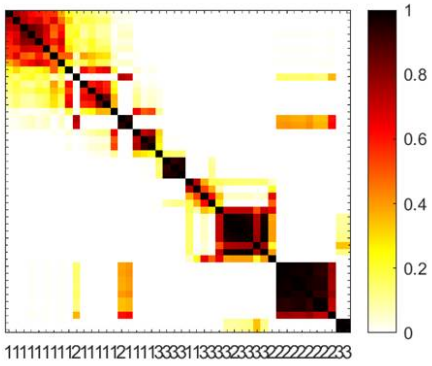
(c) c



(d) c



(e) c



(f) c

Figure 3: Neuropixel

References

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.

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.

A Appendix