Clustering Neural Populations by State-space Factor Models

Ganchao Wei
University of Connecticut, Department of Statistics

UCONN

Introduction

With high-density silicon probes and large-scale calcium imaging methods, neuroscientists now commonly record thousands of neurons at the same time. With these expanding capabilities, instead of studying single neuron or neurons from one population, we can further study neurons in the multi-population level.

In the multi-population studies, we are mostly interested in time-varying relationships within and between neural populations. Usually, these relationships can be captured by low-dimensional latent state vectors. Both AR(1) and Gaussian process (GP) are widely used models for latent vectors.

However, to implement these multi-population models, we need first define the populations. Unfortunately, it's usually not an easy task. Routinely, when it's hard or even impossible to define populations, neurons are clustered by distance-based algorithms at first. But the resulting latent structures will be biased, unless the neurons can be classified with perfect accuracy.

Here, I use the state-space factor model (SSFM) to do clustering, which let the latent structure help with clustering and vice versa. Although, this method is motivated by neuroscience and is based on the SSFM, it can be used to cluster general multiple time series data, while extracting potential low-dimensional structures at the same time.

Models for Neural Population: SSFM

Neural spikes for multi-population are modeled by the **state-space factor model (SSFM)**.

Assume we can observe neural activities for N neurons, with counting observation up to T steps. Therefore, the observation is a N-by-T matrix, $Y = (y_{it}) \in \mathbb{Z}_{\geq 0}^{N\times T}$, with each row represents the recording from single neuron. The cluster indictor for neuron i is z_i. Then, the generating model for each neuron spike is:

$$y_{it} \sim Poi(\lambda_{it})$$
$$\log(\lambda_{it})\,|z_i = d_i^{(z_i)} + \boldsymbol{c}_i^{(z_i)}\boldsymbol{x}_t^{(z_i)}$$
$$\left(d_i^{(z_i)}, \boldsymbol{c}_i'^{(z_i)}\right)' \sim N_{p+1}\left(\boldsymbol{\mu}_{dc}^{(z_i)}, \boldsymbol{\Sigma}_{dc}^{(z_i)}\right)$$

, where $\boldsymbol{c}_i^{(z_i)} \in \mathbb{R}^p$ and $\boldsymbol{x}_t^{(z_i)} \in \mathbb{R}^p$. The latent vector $\boldsymbol{x}_t^{(z_i)}$ progresses linearly with a Gaussian noise:

$$\boldsymbol{x}_1^{(z_i)} \sim N_p(\boldsymbol{x}_0, \boldsymbol{Q}_0)$$
$$\boldsymbol{x}_{t+1}^{(z_i)}|\boldsymbol{x}_t^{(z_i)} \sim N_p(\boldsymbol{A}^{(z_i)}\boldsymbol{x}_t^{(z_i)} + \boldsymbol{b}^{(z_i)}, \boldsymbol{Q}^{(z_i)})$$

We can further model interaction between clusters by allowing non-zero elements in transition matrix across clusters.

Comment 1: Cluster-dependent Loading $\boldsymbol{c}_i^{(z_i)}$

Denote the latent state matrix for cluster k as $\boldsymbol{X}^{(k)} = \left(\boldsymbol{x}_1^{(k)}, \dots, \boldsymbol{x}_T^{(k)}\right) \in \mathbb{R}^{p\times T}$. If the p is large enough to capture sufficient temporal patterns and let all clusters share the same latent state matrix, neurons in different clusters can just tune the loading to "buy" different combinations of temporal pattern. Therefore, the label-independent loading makes clustering impossible.

Comment 2: Constraints for Model Identifiability

Like factor models, this is an over-parameterized model, so that we need to put some constraints to ensure identifiability, otherwise the model can be rewritten with any affine transformation of $\boldsymbol{x}_t^{(k)}$.

In neuroscience, the fitted latent state is also interesting. Therefore, to help with interpretation, I put constraints on latent state matrix $\boldsymbol{X}^{(k)}$ directly, such that each row of $\boldsymbol{X}^{(k)}$ is centered around $\boldsymbol{0}$ and $\boldsymbol{X}^{(k)}\boldsymbol{X}'^{(k)} = \boldsymbol{I}_p$. With further constraints for diagonal $\boldsymbol{A}^{(k)}$ and $\boldsymbol{Q}^{(k)}$, the model is identifiable.

Comment 3: Interpretations of Parameters

With the constraints above, the parameters have intuitive interpretations. In other words, the spiking feature of the neuron i is decomposed into three parts:

1) The baseline firing rate $d_i^{(z_i)}$
2) A set (p) of temporal firing patterns $\boldsymbol{X}^{(k)}$. These patterns, i.e. rows of $\boldsymbol{X}^{(k)}$, are centered and orthogonal to each other.
3) The "magnitude" of each temporal pattern $\boldsymbol{c}_i^{(z_i)}$.

All these three features will be used for clustering.
In summary, the cluster parameters of cluster k are $\boldsymbol{\Theta}_k = \left\{\boldsymbol{d}^{(k)}, \boldsymbol{C}^{(k)}, \boldsymbol{\mu}_{dc}^{(k)}, \boldsymbol{\Sigma}_{dc}^{(k)}, \boldsymbol{X}^{(k)}, \boldsymbol{A}^{(k)}, \boldsymbol{b}^{(k)}, \boldsymbol{Q}^{(k)}\right\}$, with prior $\boldsymbol{H}$. Since $\boldsymbol{d}^{(k)} \in \mathbb{R}^N$ and $\boldsymbol{C}^{(k)} \in \mathbb{R}^{N\times p}$ are both neuron- and cluster-dependent, they will contain auxiliary parameters, i.e. $\left\{d_i^{(k)}, \boldsymbol{c}_i^{(k)} : z_i \neq k\right\}$ to help clustering. The generating process is denoted as $\boldsymbol{Y}_i = (Y_{i1}, \dots, Y_{iT})' \sim SSM(\boldsymbol{\Theta}_{z_i})$.

Models for Clustering: MFM

In practice, it's impossible to know the number of clusters. One usual way is to implement the Dirichlet process mixtures (DPM) model. However, due to the nature of the problem, the number of neural population is finite but unknown. Therefore, using DPM is conceptually incorrect.

Here, I choose to put prior on number of cluster directly with **mixture of finite mixture (MFM) model**. Besides the conceptual correctness, by using MFM, we can easily integrate the field knowledge about number of clusters into the model. Furthermore, MFM also has some better clustering properties than DPM. The MFM model:

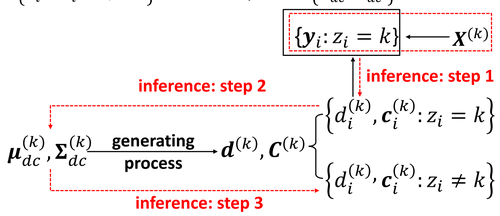
$K \sim p_k$	where p_k is a p.m.f. on $\{1,2,\dots\}$
$\boldsymbol{\pi} = (\pi_1, \dots \pi_k) \sim Dirichlet_k(\gamma, \dots, \gamma)$	given $K = k$
$Z_1, \dots, Z_N \overset{\text{i.i.d.}}{\sim} \boldsymbol{\pi}$	given $\boldsymbol{\pi}$
$\Theta_1, \dots \Theta_k \overset{\text{i.i.d.}}{\sim} \boldsymbol{H}$	given $K = k$
$\boldsymbol{Y}_i = (Y_{i1}, \dots, Y_{iT})' \sim SSM(\boldsymbol{\Theta}_{z_i})$ given $\boldsymbol{\Theta}_{1:K}$ and $Z_{1:N}$	independently for $i = 1, \dots, N$,

In the following implementations, I simply put the geometric prior on $K \sim Geomatric(r)$, with $r = 0.2$.

Inference

The posterior parameters are sampled by MCMC. To sample efficiently, I updated the SSFM-related parameters by normal approximation with Gibbs sampler, instead of implementing particle MCMC directly.

1) The MFM-related parameters: They are updated by the analog of partition-based algorithm in DPM.
2) The latent state matrix $\boldsymbol{X}^{(k)} = \left(\boldsymbol{x}_1^{(k)}, \dots, \boldsymbol{x}_T^{(k)}\right) \in \mathbb{R}^{p\times T}$: First draw the sample without constraint by the Laplace-approximation and then project the sample to the constraint space by singular value decomposition (SVD). Due to unimodality and Markovian structure, the posterior mode can be found efficiently in $O(T)$.
3) Intercept and loading $\boldsymbol{d}^{(k)}, \boldsymbol{C}^{(k)}, \boldsymbol{\mu}_{dc}^{(k)}, \boldsymbol{\Sigma}_{dc}^{(k)}$: First update $\left\{d_i^{(k)}, \boldsymbol{c}_i^{(k)} : z_i = k\right\}$ by NUTS. Then update $\left\{\boldsymbol{\mu}_{dc}^{(k)}, \boldsymbol{\Sigma}_{dc}^{(k)}\right\}$ by Gibbs sampler based on $\left\{d_i^{(k)}, \boldsymbol{c}_i^{(k)} : z_i = k\right\}$. Finally, generating $\left\{d_i^{(k)}, \boldsymbol{c}_i^{(k)} : z_i \neq k\right\}$ from the updated $\left\{\boldsymbol{\mu}_{dc}^{(k)}, \boldsymbol{\Sigma}_{dc}^{(k)}\right\}$.

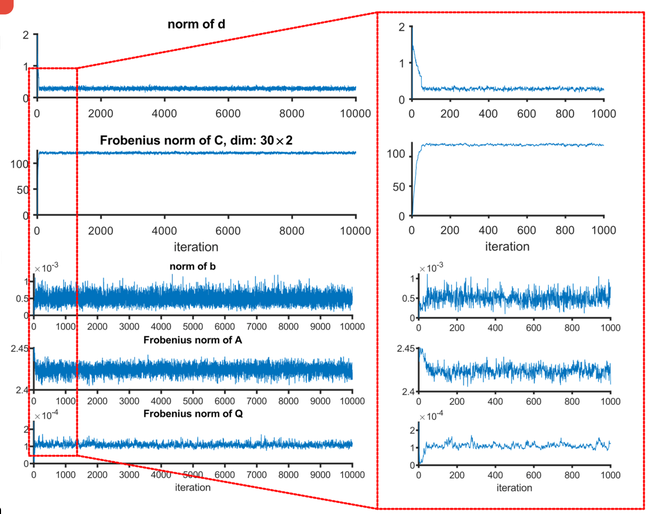


4) Linear dynamics of latent vectors $\boldsymbol{A}^{(k)}, \boldsymbol{b}^{(k)}, \boldsymbol{Q}^{(k)}$: all of them are updated by Gibbs samplers.

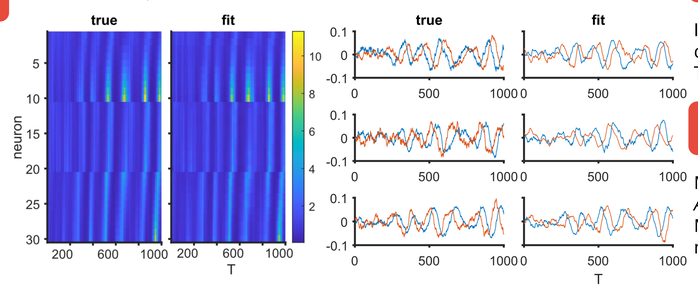
Simulations

Simulation 1: Neurons with Known Labels

3 clusters, 10 neuron in each cluster. The dimension of latent vectors is $p = 2$ and recording length is $T = 1000$. Run MCMC for 10,000 iterations. Some trace plots:

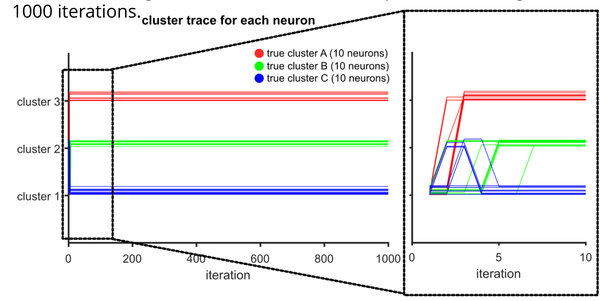


The averages of fitted mean firing rate and latent sate, from iteration 1000 to 10,000.



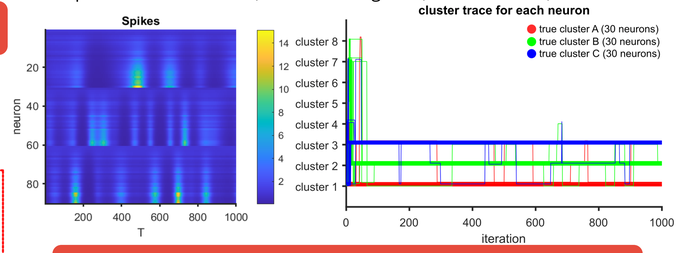
Simulation 2: Neurons with Unknown Labels

The same settings as simulation 1. The trace plot of clustering for 1000 iterations.



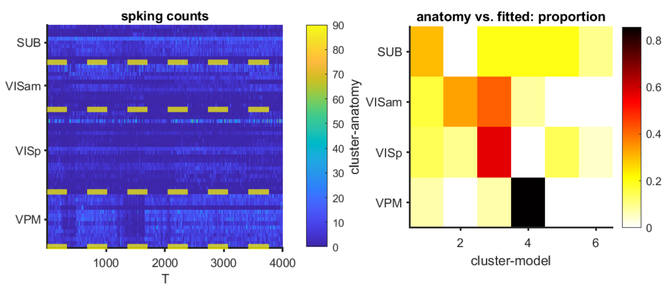
Simulation 3: A More Challenging Setting

30 neurons in each cluster. In each cluster, some neurons (left panel: tops within each cluster) have weak signals (hard to cluster).



Application

57 neurons from 4 anatomical sites. Use ~30 min recordings for clustering (bin size = 0.5s). Set $p = 4$. The average results from iteration 1000 to 3000.



Acknowledgements

I thank my advisors **Dr. Ian H. Stevenson** and **Dr. Xiaojing Wang** for detailed and constructive discussions, comments and suggestions. Thank the Allen Institute for sharing the neuropixels data.

References

Macke, J. H. et al. Empirical models of spiking in neural populations. *Adv. Neural Inf. Process. Syst.* **24**, (2011).
Miller, J. W. & Harrison, M. T. Mixture models with a prior on the number of components. *J. Am. Stat. Assoc.* **113**, 340 (2018).