Introduction

With high-density silicon probes and large-scale calcium imaging methods, neuroscientists now commonly record thousands of neurons at the same time. These new techniques can simultaneously capture large numbers of neurons from many different brain areas and multiple cell types. 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 state-space models (SSMs, and their variants) and Gaussian process (GP) can flexibly model these latent vectors and are widely used in neuroscience, to extract the low-dimensional structures in neural populations.

However, to implement these multi-population models, we need first define the population at the beginning. Unfortunately, it’s usually not an easy task. For example, in the calcium recording in the nematode *C.elegans*, there are many ways of defining neural populations because of anatomical vagueness, and this causes some confusions in multi-population analysis. 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 proposed a SSM based clustering algorithm, which let the latent structure help with clustering and vice versa. Although, this method is motivated by neuroscience and is based on the SSM, it can be used to cluster general multiple time series data, while extracting potential low-dimensional structures at the same time.

Model

The state space model (SSM) for neural spikes

Assume we can observe neural activities for neurons, with counting observation up to steps. Therefore, the observation is a -by- matrix, , with each row represents the recording from single neuron. The cluster indictor for neuron is . Then, the generating model for each neuron spike is:

, where and . The intercept and loading are cluster-dependent to help with clustering and interpretation. The latent vector progresses linearly with a Gaussian noise:

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

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 . To help with interpretation and make it possible for clustering, I put constraints on such that each row of is centered around 0 and . With further diagonal and constraints, the model is identifiable. Under this constraint, the parameters have intuitive interpretations. In other words, the spiking feature of the neuron is decomposed into three parts: (1) the baseline firing rate , (2) a set () of temporal firing patterns and (3) the “magnitude” of each temporal pattern . All these three features will be used for clustering later.

In summary, denote the cluster parameters of cluster as . The and contains auxiliary parameters, i.e. , to help clustering. The generating process is then denoted as .

Mixture of Finite Mixtures (MFM)

Due to the nature of the problem, the number of neural population is finite but unknown. Therefore, it’s conceptually incorrect to use Dirichlet process mixtures (DPM). Here, we choose to put prior on number of cluster directly with mixture of finite mixture (MFM) model. Besides the conceptual correctness, MFM also has some better properties for clustering than DPM. The MFM model:

where is a p.m.f.on given

given

given

independently for , given and

In the following implementation, I simply put the geometric prior on .

Inference

The posterior parameters are sampled from MCMC, with combination of Gibbs sampler, no-U-turn sampler and normal approximation.

1. The MFM-related parameters:

They are updated by the partition-based analog algorithm in DPM.

1. latent vector matrix :

First draw the sample without constraint and then project the sample to the constraint-space. To update efficiently, it is updated by Laplace-approximation. Due to unimodality and Markovian structure, the posterior mode can be found efficiently.

1. Intercept and loading :

First update by NUTS. Then update by Gibbs sampler based on . Finally, generating from the updated .

Diagram

Description automatically generated

1. Linear dynamics of latent vectors : all are updated by Gibbs samplers.

Results

Simulation

Application

Future direction/ Improvement

1. Currently, the latent state matrix is updated by Laplace approximation in the unconstrained step. Although it’s pretty efficient, it’s unknown how far the samples are deviated from the true distribution. Maybe a better way is to sample by particle MCMC, integrating with information from gradient and hessian or not.
2. The number of dimension of the latent vectors is assumed to be the same across clusters and need to be pre-specified. Since there is no constraint on loading, we can put a multiplicative gamma process shrinkage prior on the loading and update by adaptive Gibbs sampler as in sparse Bayesian infinite factor models. By doing this cluster-by-cluster, we can select the dimension separately for each cluster.

Acknowledgement

I thank my advisors Dr. Ian H. Stevenson and Dr. Xiaojing Wang for detailed and constructive discussions, comments and suggestions.