# Notations (common)

Each row is the recording for neuron , , Denote the cluster index for neuron as . The number of neurons in cluster is , and .

# Model

Denote the latent vector in cluster as . For simplicity, assume all . Each observation follows a Poisson distribution as follows:

Where and .

Denote all latent states as and they evolve linearly with Gaussian noise:

To simplify, assume is known (e.g. ).

If we assume block diagonal (as in Joshua et al., 2020) for process noise covariance, we can write things as:

Notice forms the full transition matrix as:

If the row block of is . Then, .

If we further let be diagonal: denote the row of , , as , , . The corresponding process noise variance as . Then:

The parameters need to estimate:

1. Latent vectors:
2. Initials:
3. Linear mapping for latent vectors: and
4. Linear dynamics for latent vectors: and
5. Process noise:

Since the progress noise is independent in the model, , where is the Poisson density and is the parameters in cluster .

# Conditional Priors

1. Latent vectors : the conditional prior is defined by

(Assume there are clusters)

1. Initials :

(Assume there are clusters)

and

1. Linear mapping for latent vectors and :

Where and

1. Linear dynamics for latent vectors and :

(if the number of cluster is )

* 1. Assume be diagonal:

Where and

* 1. Assume beblock-diagonal:

Denote

Where and

* 1. No constraints on :

Denote

Where and

1. Process noise :
   1. Assume be diagonal:

Where and

* 1. Assume beblock-diagonal:

Where and .

(To make the mean of loosely centered around )

* 1. No constraints on :

(if the number of cluster is )

Where and .

# MCMC iteration

1. Update :

use local Normal approximation at prior, i.e. update by adaptive smoothing.

Notice: adaptive filtering is not the “exact Laplace approximation”. In Laplace approximation, we are evaluating at the mode/ maximum of posterior, but in adaptive filtering, it is evaluated at the prior…

**An IMPROTANT drawback for adaptive filtering/ smoothing is that it don’t give the covariance between different time steps.** This will ruin the sampling and estimations for other parameters.

But the mean estimations for adaptive filtering/ smoothing is perfect.

**An easy solution:** Only use the posterior mean from adaptive smoothing (discard the variance estimation), and then calculate the covariance matrix by the Hessian as follows (“THE EXACT LAPLACE APPROXIMATION”).

THE EXACT LAPLACE APPROXIMATION

Denote column of mean firing rate and observation as and . The linear mapping matrix for all observations is , such that . Let and

The first and second derivative with respect to

So, the gradient is:

And the block tri-diagonal Hessian:

Use Newton-Raphson to find and

When using Newton-Raphson, in MATLAB will make use of block tri-diagonal structure automatically.

This can be much slower than smoother in some case, and my implementation is not robust (sometimes). See two examples in [newtonTest.m](https://github.com/weigcdsb/state-space-clustering/blob/main/test/newtonTest.m).

To sample efficiently, use Cholesky decomposition of : sample , then .

1. Update :

Because of independence, we can update element by element. To write things easily, just write in matrix form. By conjugacy,

1. Update and :

Notice . Then the problem is reduced to Bayesian Poisson regression. Denote and . To get an efficient update, use the Laplace approximation:

Where is Poisson log-likelihood.

Use Newton-Raphson to find and

1. Update and
   1. Assume be diagonal:

Again, rewrite Then the problem is reduced to Bayesian linear regression. Denote and . Then by conjugacy,

* 1. Assume beblock-diagonal:

Again, notice , then the problem is reduced to Bayesian linear regression. Denote , and . Then

By conjugacy:

* 1. No constraints on :

Let , and . By conjugacy, .

1. Update :
   1. Assume be diagonal:

Let

By conjugacy,

* 1. Assume beblock-diagonal:

Let

By conjugacy,

* 1. No constraints on :

Let

By conjugacy,