Let

, where .

Use AR(1) for

**Connection to the “old” model**

The old model

For any (e.g. the column mean of )

That means,

**The effective in new model is 1 larger than the old one, i.e.**

The new model and old model have the **same number of constraints**, in terms of observation equation:

1. Constraints in old model:
   1. Columns of are centered around
   2. has orthonormal columns, i.e.
2. Constraints in new model:
   1. or
   2. The 1st element of is 1.

Both parametrizations have their own strengths/ explanations. The parametrization in old model may have a better interpretation.

**Note: when showing the results, center and orthogonalize the columns**

**Constraints:**

Easy one: Just let and be diagonal. (Now is free, yay!)

Maybe just let be upper/lower-trangular is enough? Why?

Letter

Description automatically generated with low confidence

Because the number of constraint parameters is , which is equivalent to the number of constraints in traditional constraint for **normal factor model**. **Try later.**

Using less constraints will lead to a faster convergence.

After doing Schur decomposition for

Therefore,

**Key derivation for clustering: Marginal distribution of .**

I no longer need to use auxiliary parameters.

To save notations, denote as , as , as and as . Further, denote . Then, I need to evaluate the marginal likelihood

**Way 1: use normal approximation (the Laplace approximation)**

That is,

, where is the Poisson density with mean .

In the Laplace approximation, , i.e., MAP and is the inverse negative hessian at . Since is usually super large, I can just use MLE to replace MAP, by doing Poisson regression. However, there are some drawbacks:

1. This is quite painful. If there are neurons and clusters at current step, I need to run Newton-Raphson times! (Maybe I can do OLS instead, i.e., )
2. It’s not quite robust, especially when generating a new set of parameters from prior. The inverse-hessian might be close to singular.

Another easy way is just set (evaluate the approximation at prior). Well, the approximation should be super bad…

**Way 2: approximate by gamma distribution**

Assume conditional independence: . Denote . Then, since

Then approximate as

So,

, where and

This approximation is quite accurate when is not super small, i.e. should be small to achieve a good performance. This is just our case, since I separate mean and variance and just capture the variance part. **The will even be nearly guaranteed to be small when doing the clustering, since the outliers will just be clustered to another cluster.**

Empirically, we can see fitting in simulation 1. The just bounces around 0 at each step . For example, even when , the performance is good enough.

Chart, histogram, box and whisker chart

Description automatically generatedChart, histogram

Description automatically generated

**Way 3: Use 2nd-order polynomial to approximate exponential function**

Chebyshev polynomial approximation in :

Here, I derive things in the most general case, i.e., each spike count correspond to separate set of . More specifically, let

Since there’s an offset, I need to redo the algebra…

The log-integrand is:

, where and .

Then the marginal likelihood is:

Denote , then:

Let

i.e.

In summary,

, where

**I first tried as in Keely’s paper, i.e., use and .** And let , for all

However, this only works when the range on is small. In other words, the resolution should be high.

Take the 1st neuron for example, let and

|  |  |
| --- | --- |
| Within | Across all range of |
|  |  |

let and

|  |  |
| --- | --- |
| Within | Across all range of |
|  |  |

Let and



**When I divide the spike counts by 5, i.e. Y\_new = round(Y/5), things kind of works.** But still not as good as gamma approximation.

|  |  |  |
| --- | --- | --- |
| Trace- PAL | Fitted FR in 100th sample | Trace-Gamma |
|  |  |  |

**Y\_new = round(Y/6)**

|  |  |  |
| --- | --- | --- |
| Trace- PAL | Fitted FR in 100th sample | Trace-Gamma |
|  |  |  |

OK, let’s only do 2nd order approximation locally.

**That is, I tried to let each spike count correspond to separate set of .** More specifically, let

, for or . When , just reset it as 0.1. The following is the approximation using , from to



Looks promising, but it actually doesn’t work.

|  |  |
| --- | --- |
| Simulation 2 (unrobust)  Most are (1,3), (2); sometimes others…  Rarely be (1), (2), (3) | Simulation 3 |
|  |  |

The reason is that when comparing different marginal likelihood for different sets of cluster parameters, the is not necessarily center around , when .

**To solve this, I evaluate in the interval** , where means . However, doing this even make things diverge (even for the simple simulation 2)!



**Simulation 1: known label**

When

A picture containing calendar

Description automatically generated

I further use .



Trace for X



**Simulation 2: same setting as in simulation 1 but remove the label.**

I checked these 2 things:

1. Using normal approximation (with MLE) and gamma approximation are similar (at least for the first 20 iterations)
2. give similar results, at least for the first 20 iterations

So, in the following 2 fittings, I just use gamma approximation and and to save time.

|  |  |  |
| --- | --- | --- |
|  | trace | Fit at iteration 20 |
| P=1 |  |  |
| P=2 |  |  |

OK, perfect. Let’s see what happens in a more challenging setting.

**Simulation 3: as in simulation 3 in the slides**

:



|  |  |  |
| --- | --- | --- |
| Iteration 10 | Iteration 50 | Iteration 100 (end of chain) |
|  |  |  |
|  |  |  |

:



|  |  |  |
| --- | --- | --- |
| Iteration 10 | Iteration 50 | Iteration 100 (end of chain) |
|  |  |  |
|  |  |  |

Although the results are similar, it seems that using larger gives a bit better capture of variance in this case.

**Although, the results seem (just seem) not as good as previous, but I strongly prefer the current model:**

1. The new model is cleaner & more elegant. Moreover, it has a better interpretation with less constraints, i.e., mean part () + variance part (). The value of just controls how detailed we want to model the variance.
2. The convergence & fitting in terms of mean firing rate is (**so**) much faster, especially when doing the clustering.
3. The seemly perfectness (, **which is fake**) in the previous method is because the convergence of is slow, and this gives a huge space for to tune itself… The convergence of clustering is achieved even before convergence of fitting. After convergence, the auxiliary will never beat , since is observed for . So, after a few iterations, the algorithm will just stop searching. It will even stop at some totally irrational results sometimes.
4. The previous method is not robust… When change the variance of or change , the results will change a lot.
5. The new method may just do a more detailed clustering: not only consider the waveform but also take amplitude into consideration.

**Choose**

The number of latent factor should be optimized. Choose by using the shrinkage prior and adaptive Gibbs sampler. See if I can easily do that.

But conceptually, I think it would be hard… since they put the shrinkage prior on . If I change prior of , the orthogonal structure and some other convince will be lost… Think.

**Maybe put shrinkage-type prior on or , so that the last few latent vectors just decay to 0? It should be doable.**