Model-based Clustering for Neural Populations

The goal for this research is to do model-based clustering for neural populations, by making use of features for each counting process observation.

1 Notations

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, $\mathbf{Y} \in \mathbb{Z}_{\geq 0}^{N \times T}$,with each row represents the recording from single neuron. Denote the recording for neuron i as $\mathbf{y}_i = (y_{i1}, \ldots, y_{i\mathrm{T}})'$, $i = 1, \ldots N.$, with the cluster index for neuron i as $z_i \in \{1, \ldots\}$. The number of neurons in cluster j is $n_j = \sum_{i=1}^{N} I(z_i = j)$, and $\sum_{j=1,2,\ldots} n_j = N$. The proportion/ probability in cluster z_i is ρ_{z_i}.

2 Clustering Wrapper

The model-based clustering problem can be transformed into fitting the mixture model (MM). The likelihood for each cluster depends on how we model the counting observation, but fitting strategies for MM are the same for all models. Here, I choose to fit the MM by Gibbs sampler. Depending on whether the number of cluster is finite or not, there are two versions: finite mixture model (FMM) and Dirichlet process mixture model (DPMM).

2.1 Finite Mixture Model

Assume the number of cluster is J. The full likelihood for these N neurons is

$$L = \prod_{i=1}^{N} \rho_{z_i} f\left(\mathbf{y}_i | \mathbf{\Theta}_{z_i}\right) = \prod_{j=1}^{J} \rho_j^{n_j} \left[\prod_{i: z_i = j} f\left(\mathbf{y}_i | \mathbf{\Theta}_j\right) \right]$$

, where $\mathbf{\Theta}_j$ contains all parameters in cluster j defined by the specific model. Therefore, the parameters need to update are:

(1) Cluster indicator: $\{z_i\}_{i=1}^{N}$

(2) Cluster proportion: $\rho = (\rho_1, \ldots \rho_J)'$

(3) Model parameters: $\mathbf{\Theta}_j$

The (conditional) priors for clustering-related parameters:

(1) Cluster indicator $\{z_i\}_{i=1}^N$: $P(z_i = j) = \rho_j$

(2) Cluster proportion $\rho = (\rho_1, \dots \rho_J)'$:

$$\rho \sim Dir(\delta_1, \dots \delta_J)$$

, where $\delta_1 = \dots = \delta_J = 1$

So, the MCMC(Gibbs sampler) iteration for FMM is:

(1) Update $\{z_i\}_{i=1}^N$:

$$P\left(z_i = j \middle| \mathbf{y}_i, \{\mathbf{\Theta}_j\}_{j=1}^J\right) \propto \rho_j f(\mathbf{y}_i | \mathbf{\Theta}_j)$$

(2) Update $\rho = (\rho_1, \dots \rho_J)^{'}$:

$$\rho | \ \{\mathbf{y}_i\}_{i=1}^N, \ \{z_i\}_{i=1}^N, \{\mathbf{\Theta}_j\}_{j=1}^J \sim Dir(\delta_1 + n_1, \dots \delta_J + n_J)$$

(3) Update $\mathbf{\Theta}_j$: this is defined by the specific model. When there's no $z_i = j$, just sample $\mathbf{\Theta}_j$ from priors or by other observation-independent ways.

2.2 Dirichlet Process Mixture Model

Since calculation of posterior predictive distribution can be hard or even impossible for complicated models, instead of using the popular CRP representation of DP (Neal, 2020), I choose to use the slice sampler (Walker, 2007).

Use the "stick-breaking" construction for cluster proportion, i.e.

$$\rho_1 = \eta_1$$

$$\rho_j = (1 - \eta_1) \cdot \ldots \cdot (1 - \eta_{j-1}) \eta_j$$

$$\eta_j \sim Beta(1, \alpha)$$

In the slice sampler for DPMM, the parameters need to update are:

(1) "stick-breaking" elements: η_j

(2) Augment latent variable: $\{u_i\}_{i=1}^N$

(3) Model parameters: $\mathbf{\Theta}_j$

(4) Cluster indicator: $\{z_i\}_{i=1}^N$

So, the MCMC(Gibbs sampler) iteration for DPMM is:

(1) update η_j, for $j = 1, \ldots, z^* = max\left\{z_i\right\}_{i=1}^N$ as

$$\eta_j | \left\{z_i\right\}_{i=1}^N, \ldots \sim Beta(n_j + 1,\ N - \sum_{l=1}^{j} n_l + \alpha)$$

(2) update $\left\{u_i\right\}_{i=1}^N$:

$$u_i | \rho, \ldots \sim U(0,\ \rho_{z_i})$$

(3) update η_j, for $j = z^* + 1, \ldots,\ s^*$. s^* is the smallest value, s.t. $\sum_{j=1}^{s^*} \rho_j > 1 - \min\left\{u_1, \ldots, u_N\right\}$

$$\eta_j \sim Beta(1, \alpha)$$

(4) Update $\boldsymbol{\Theta}_j$: this is defined by the specific model. When there's no $z_i = j$, just sample $\boldsymbol{\Theta}_j$ from priors or by other observation-independent ways.

(5) Update $\left\{z_i\right\}_{i=1}^N$

$$P\left(z_i = j \middle| \mathbf{y}_i, \left\{\boldsymbol{\Theta}_j\right\}, \rho, \left\{u_i\right\}_{i=1}^N\right) = \frac{f\left(\mathbf{y}_i | \boldsymbol{\Theta}_j\right)}{\sum_{j:\rho_j > u_i} f\left(\mathbf{y}_i | \boldsymbol{\Theta}_j\right)}$$

3 linear Dynamical System Model

Here, I model the observations by a linear dynamical system (LDS) model.

LDS models the multi-dimensional time series using a lower dimensional latent representation of the system, which evolves over time according to linear dynamics. By specifying the linear dynamics and process noise covariance, we can also handle the interactions between different neural populations (clusters).

3.1 Model Details

Denote the latent vector in cluster j as $\mathbf{x}_t^{(j)} \in \mathbb{R}^{p_j}$. For simplicity, assume all $p_j = p$. Each observation follows a Poisson distribution:

$$\log \lambda_{\mathrm{it}} = d_i + \mathbf{c}'_i \mathbf{x}_t^{(z_i)}$$

$$y_{\mathrm{it}} \sim Poisson(\lambda_{\mathrm{it}})$$

, where $\mathbf{c}_i \in \mathbb{R}^p$ and $\mathbf{x}_t^{(z_i)} \in \mathbb{R}^p$.

Although the loading (d_i and $\mathbf{c}_i$) is determined by neuron index i, the distribution is also cluster-dependent. That is,

$$(d_i, \mathbf{c}'_i)' \sim N(\boldsymbol{\mu}_{\mathrm{dc}}^{(z_i)}, \boldsymbol{\Sigma}_{\mathrm{dc}}^{(z_i)})$$

By doing this, the loading within each cluster is also correlated.

Denote all latent states as $\mathbf{x}_t = \left(\mathbf{x'}_t^{(1)}, \mathbf{x'}_t^{(2)}, \ldots\right)'$ and they evolve linearly with a Gaussian noise:

$$\mathbf{x}_1 \sim N(\mathbf{x}_0, \mathbf{Q}_0)$$
$$\mathbf{x}_{t+1}|\mathbf{x}_t \sim N(\mathbf{A}\mathbf{x}_t + \mathbf{b}, \mathbf{Q})$$

For simplicity, assume $\mathbf{Q}_0$ is known (e.g. $\mathbf{Q}_0 = \mathbf{I} \times 10^{-2}$).

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

$$\mathbf{x}_{t+1}^{(j)}|\mathbf{x}_t^{(1)}, \mathbf{x}_t^{(2)}, \ldots \sim N(\sum_{l=1,\ldots} \mathbf{A}_{j\leftarrow l}\mathbf{x}_t^{(l)} + \mathbf{b}_j, \mathbf{Q}^{(j)})$$

Notice $\{\mathbf{A}_{j\leftarrow l}\}$ forms the full transition matrix as:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{1\leftarrow 1} & \mathbf{A}_{1\leftarrow 2} & \ldots \\ \mathbf{A}_{2\leftarrow 1} & \mathbf{A}_{2\leftarrow 2} & \ldots \\ \ldots & \ldots & \ldots \end{pmatrix}$$

Denote the j^{th} row block of $\mathbf{A}$ as $\mathbf{A}_j = \begin{pmatrix} \mathbf{A}_{j\leftarrow 1} & \mathbf{A}_{j\leftarrow 2} & \ldots \end{pmatrix}$. Then, $\sum_{l=1,\ldots} \mathbf{A}_{j\leftarrow l}\mathbf{x}_t^{(l)} + \mathbf{b}_j = \mathbf{A}_j\mathbf{x}_t + \mathbf{b}_j$.

If we further let $\mathbf{Q}$ be diagonal, with the k^{th} row of $\mathbf{x}_t$, $\mathbf{A}$, $\mathbf{b}$ denoted as x_{kt}, $\mathbf{a}_k$, b_k. The corresponding process noise variance is q_k. Then:

$$x_{k,t+1}|x_{\text{kt}} \sim N\left(\mathbf{a}_k^{'}\mathbf{x}_t + b_k, q_k\right)$$

To facilitate derivation in Gibbs sampler, write the linear dynamics of $\mathbf{x}_t$ in MV-GLM form, i.e.

$$\mathbf{x}_{t+1}^{'} = \begin{pmatrix} 1 & \mathbf{x}_t^{'} \end{pmatrix} \begin{pmatrix} \mathbf{b}^{'} \\ \mathbf{A}^{'} \end{pmatrix} + \epsilon_i^{'}$$

, where $\epsilon \sim N(0, \mathbf{Q})$. Then if we stack the latent by row, the problem is reduced to a multivariate general linear model (MV-GLM) problem:

$$\begin{pmatrix} \mathbf{x}_2^{'} \\ \mathbf{x}_3^{'} \\ \vdots \\ \mathbf{x}_T^{'} \end{pmatrix} = \begin{pmatrix} 1 & \mathbf{x}_1^{'} \\ 1 & \mathbf{x}_2^{'} \\ \vdots & \vdots \\ 1 & \mathbf{x}_{T-1}^{'} \end{pmatrix} \begin{pmatrix} \mathbf{b}^{'} \\ \mathbf{A}^{'} \end{pmatrix} + \mathbf{E}$$

, where $\mathbf{E} = (\epsilon_1, \ldots, \epsilon_{T-1})^{'}$.

In summary, Let $\mathbf{\Theta} = \left\{\mathbf{z},\ \mathbf{d}, \mathbf{C}, \{\mathbf{x}_t\}_{t=1}^{T}, \mathbf{x}_0, \mathbf{Q}_0, \mathbf{A},\ \mathbf{b}, \mathbf{Q}\ \right\}$ be the set of parameters for LDS. The number of observation/ neuron is N, and they can group into J clusters. In each cluster, there are p latent state vectors. The recording length is T.

(1) $\mathbf{z} \in \mathbb{Z}^N$: cluster index for each neuron/ observation, with $z_i \in \{1, \ldots, J\}$ for $i = 1, \ldots, N$.

(2) $\mathbf{d} \in \mathbb{R}^N$ and $\mathbf{C} \in \mathbb{R}^{N \times p}$: baseline & loading of the latent

(3) $\mathbf{x}_t \in \mathbb{R}^{\mathrm{Jp}}$: all latent at t.

(4) $\mathbf{x}_t^{(j)} \in \mathbb{R}^p$ means the latent in cluster j at time t.

(5) $\mathbf{x}_0 \in \mathbb{R}^{\mathrm{Jp}}$ and $\mathbf{Q}_0 \in \mathbb{R}^{\mathrm{Jp} \times Jp}$: mean and covariance of $\mathbf{x}_1$. For simplicity, assume $\mathbf{Q}_0$ is known.

(6) $\mathbf{A} \in \mathbb{R}^{Jp \times Jp}$, $\mathbf{b} \in \mathbb{R}^{Jp}, \mathbf{Q} \in \mathbb{R}^{Jp \times Jp}$: linear dynamics of the latent.

Since the progress noise is independent at each step and the observation is assumed conditional independent, the likelihood is:

$$f(\mathbf{y}|\boldsymbol{\Theta}) = \prod_{i=1}^{N}\prod_{t=1}^{T} P(y_{it}|\boldsymbol{\Theta}) = \prod_{i=1}^{N}\prod_{t=1}^{T} POI(y_{it}|\exp{(d_i + \mathbf{c'}_i \mathbf{x}_t^{(z_i)})})$$

, where $POI(\cdot|\lambda)$ is the density of $Poisson(\lambda)$.

DETOUR:
Maybe in the future, we may switch to EM. To help with this, I also give the likelihood for complete observation $(\mathbf{y}, \{\mathbf{x}_t\}_{t=1}^T)$. Let $\boldsymbol{\Theta'} = \boldsymbol{\Theta} \backslash \{\mathbf{x}_t\}_{t=1}^T$.

$$f(\mathbf{y}, \{\mathbf{x}_t\}_{t=1}^T|\boldsymbol{\Theta'}) = \prod_{i=1}^{N}\prod_{t=1}^{T} P(y_{it}, \mathbf{x}_t|\boldsymbol{\Theta'}) = \prod_{i=1}^{N}\prod_{t=1}^{T} POI(y_{it}|\exp{(d_i + \mathbf{c'}_i \mathbf{x}_t^{(z_i)})}) \cdot N(\mathbf{x}_t^{(z_i)}|\mathbf{A}_{z_i}\mathbf{x}_t + \mathbf{b}_{z_i}, \mathbf{Q}_{z_i})$$

, where $\mathbf{A}_j \in \mathbb{R}^{p \times Jp}$ and $\mathbf{b}_j \in \mathbb{R}^p$ are the j^{th} row block of $\mathbf{A}$ and $\mathbf{b}$. $\mathbf{Q}_j \in \mathbb{R}^{p \times p}$ is the j^{th} row and column block of $\mathbf{Q}$.

3.2 Conditional Priors for Parameters

The parameters need to estimate:

(1) Latent vectors: $\{\mathbf{x}_t\}_{t=1}^T$

(2) Initials: $\mathbf{x}_0$

(3) Linear mapping (loading) for latent vectors: $\{d_i\}_{i=1}^N$ and $\{\mathbf{c}_i\}_{i=1}^N$

(4) Mean and covariance for loading in each cluster: $\left\{\boldsymbol{\mu}_{\mathrm{dc}}^{(j)}\right\}_j$ and $\left\{\boldsymbol{\Sigma}_{\mathrm{dc}}^{(j)}\right\}_j$

(5) Linear dynamics for latent vectors: $\mathbf{A}$ and $\mathbf{b}$

(6) Process noise: $\mathbf{Q}$

The conditional priors for these parameters:

(1) Latent vectors $\{\mathbf{x}_t\}_{t=1}^{T}$: the conditional prior is defined by

$$\mathbf{x}_1 \sim N(\mathbf{x}_0, \mathbf{Q}_0)$$

$$\mathbf{x}_{t+1}|\mathbf{x}_t \sim N(\mathbf{A}\mathbf{x}_t + \mathbf{b}, \mathbf{Q})$$

(2) Initials $\mathbf{x}_0$: assume there are J clusters,

$$\mathbf{x}_0 \sim N(\boldsymbol{\mu}_{\mathbf{x}_{00}},\ \boldsymbol{\Sigma}_{\mathbf{x}_{00}})$$

, where $\boldsymbol{\mu}_{\mathbf{x}_{00}} = \mathbf{0}_{\mathrm{Jp}}$ and $\boldsymbol{\Sigma}_{\mathbf{x}_{00}} = \mathbf{I}_{\mathrm{Jp}}$

(3) Linear mapping (loading) for latent vectors $\{d_i\}_{i=1}^{N}$ and $\{\mathbf{c}_i\}_{i=1}^{N}$:

$$(d_i, \mathbf{c}_i')' \sim N(\boldsymbol{\mu}_{\mathrm{dc}}^{(z_i)}, \boldsymbol{\Sigma}_{\mathrm{dc}}^{(z_i)})$$

(4) Mean and covariance for loading in each cluster Mean and covariance for loading in each cluster $\left\{\boldsymbol{\mu}_{\mathrm{dc}}^{(j)}\right\}_j$ and $\left\{\boldsymbol{\Sigma}_{\mathrm{dc}}^{(j)}\right\}_j$:

$$\boldsymbol{\mu}_{\mathrm{dc}}^{(j)} \sim N(\delta_{dc0}, \mathbf{T}_{dc0})$$

, where $\boldsymbol{\delta}_{dc0} = \mathbf{0}_{p+1}$ and $\mathbf{T}_{dc0} = \mathbf{I}_{p+1}$

$$\boldsymbol{\Sigma}_{\mathrm{dc}}^{(j)} \sim W^{-1}\left(\Psi_{dc0}, \nu_{dc0}\right)$$

, where $\nu_{dc0} = p + 1 + 2$ and $\Psi_{dc0} = \mathbf{I}_{p+1} \times 10^{-4}$

(5) Linear dynamics for latent vectors: $\mathbf{A}$, $\mathbf{b}$ and process noise $\mathbf{Q}$.
Denote $\mathbf{F} = \begin{pmatrix} \mathbf{b}' \\ \mathbf{A}' \end{pmatrix}$, $\mathbf{f} = vec(\mathbf{F})$

$$\begin{aligned} P\left(\mathbf{f},\ \mathbf{Q}\right) &= P(\mathbf{Q})P(\mathbf{f}|\mathbf{Q}) \\ \mathbf{Q} &\sim W^{-1}(\Psi_{\mathbf{Q}_0}, \nu_{\mathbf{Q}_0}) \\ \mathbf{f} &\sim N(\mathbf{f}_0, \mathbf{Q} \bigotimes \boldsymbol{\Lambda}_0^{-1}) \end{aligned}$$

, where $\mathbf{f}_0 = vec(\mathbf{F}_0)$.If the number of cluster is J, $\nu_{\mathbf{Q}_0} = Jp + 2$, $\Psi_{\mathbf{Q}_0} = \mathbf{I}_{\mathrm{Jp}} \times 10^{-4}$. (To make the mean of $\mathbf{Q}$ loosely centered around $\mathbf{I}_{Jp} \times 10^{-4}$), $\mathbf{F}_0 = \begin{pmatrix} \mathbf{0}'_{Jp} \\ \mathbf{I}_{\mathrm{Jp}} \end{pmatrix}$ and $\boldsymbol{\Lambda}_0 = \mathbf{I}_{Jp+1}$

3.3 MCMC (Gibbs Sampler)

3.3.1 Update $\{\mathbf{x}_t\}_{t=1}^{T}$

Use Laplace approximation and make use of the block tri-diagonal Hessian.

Denote t^{th} column of mean firing rate and observation as $\widetilde{\boldsymbol{\lambda}}_t = (\lambda_{1t}, \ldots, \lambda_{\text{Nt}})'$ and $\widetilde{\mathbf{y}}_t = (y_{1t}, \ldots, y_{\text{Nt}})'$. The linear mapping matrix for all observations is $\mathbf{C}$, such that $\log \widetilde{\boldsymbol{\lambda}}_t = \mathbf{d} + \mathbf{C}\mathbf{x}_t$. Let $\mathbf{x} = (\mathbf{x}'_1, \ldots, \mathbf{x}'_T)'$ and $f(\mathbf{x}) = \log P(\mathbf{x} | \{\mathbf{y}_i\}_{i=1}^{N}, \mathbf{C}, \mathbf{Q}_0, \mathbf{A}, \mathbf{b}, \mathbf{Q}, \ldots)$ The first and second derivative with respect to $\mathbf{x}$, for $t = 2, \ldots, T-1$:

$$
\begin{aligned}
\frac{\partial f}{\partial \mathbf{x}_1} &= \mathbf{C}'\left(\widetilde{\mathbf{y}}_1 - \widetilde{\boldsymbol{\lambda}}_1\right) - \mathbf{Q}_0^{-1}(\mathbf{x}_1 - \mathbf{x}_0) + \mathbf{A}'\mathbf{Q}^{-1}(\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1 - \mathbf{b}) \\
\frac{\partial f}{\partial \mathbf{x}_t} &= \mathbf{C}'\left(\widetilde{\mathbf{y}}_t - \widetilde{\boldsymbol{\lambda}}_t\right) - \mathbf{Q}^{-1}(\mathbf{x}_t - \mathbf{A}\mathbf{x}_{t-1} - \mathbf{b}) + \mathbf{A}'\mathbf{Q}^{-1}(\mathbf{x}_{t+1} - \mathbf{A}\mathbf{x}_t - \mathbf{b}) \\
\frac{\partial f}{\partial \mathbf{x}_T} &= \mathbf{C}'\left(\widetilde{\mathbf{y}}_T - \widetilde{\boldsymbol{\lambda}}_T\right) - \mathbf{Q}^{-1}(\mathbf{x}_T - \mathbf{A}\mathbf{x}_{T-1} - \mathbf{b}) \\
\frac{\partial^2 f}{\partial \mathbf{x}_1 \partial \mathbf{x}_1'} &= -\mathbf{C}'\text{Diag}\left(\widetilde{\boldsymbol{\lambda}}_1\right)\mathbf{C} - \mathbf{Q}_0^{-1} - \mathbf{A}'\mathbf{Q}^{-1}\mathbf{A} \\
\frac{\partial^2 f}{\partial \mathbf{x}_t \partial \mathbf{x}_t'} &= -\mathbf{C}'\text{Diag}\left(\widetilde{\boldsymbol{\lambda}}_t\right)\mathbf{C} - \mathbf{Q}^{-1} - \mathbf{A}'\mathbf{Q}^{-1}\mathbf{A} \\
\frac{\partial^2 f}{\partial \mathbf{x}_T \partial \mathbf{x}_T'} &= -\mathbf{C}'\text{Diag}\left(\widetilde{\boldsymbol{\lambda}}_T\right)\mathbf{C} - \mathbf{Q}^{-1} \\
\frac{\partial^2 f}{\partial \mathbf{x}_1 \partial \mathbf{x}_2'} &= \frac{\partial^2 f}{\partial \mathbf{x}_t \partial \mathbf{x}_{t+1}'} = \mathbf{A}'\mathbf{Q}^{-1} \qquad \frac{\partial^2 f}{\partial \mathbf{x}_t \partial \mathbf{x}_{t-1}'} = \mathbf{Q}^{-1}\mathbf{A}
\end{aligned}
$$

So, the gradient is:

$$
\nabla = \frac{\partial f}{\partial \mathbf{x}} = \left(\left(\frac{\partial f}{\partial \mathbf{x}_1}\right)', \ \ldots, \left(\frac{\partial f}{\partial \mathbf{x}_T}\right)'\right)'
$$

And the block tri-diagonal Hessian:

$$
H = \frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{x}'} = \begin{pmatrix}
\frac{\partial^2 f}{\partial \mathbf{x}_1 \partial \mathbf{x}_1'} & \mathbf{A}'\mathbf{Q}^{-1} & 0 & \cdots & 0 \\
\mathbf{Q}^{-1}\mathbf{A} & \frac{\partial^2 f}{\partial \mathbf{x}_2 \partial \mathbf{x}_2'} & \mathbf{A}'\mathbf{Q}^{-1} & \cdots & \vdots \\
0 & \mathbf{Q}^{-1}\mathbf{A} & \frac{\partial^2 f}{\partial \mathbf{x}_3 \partial \mathbf{x}_3'} & \cdots & \vdots \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & \cdots & \cdots & \frac{\partial^2 f}{\partial \mathbf{x}_T \partial \mathbf{x}_T'}
\end{pmatrix}
$$

Use Newton-Raphson to find $\boldsymbol{\mu}_{\mathbf{x}} = \text{argmax}_{\mathbf{x}}(f(\mathbf{x}))$ and $\boldsymbol{\Sigma}_{\mathbf{x}} = -\left[\frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{x}'} \ |_{\mathbf{X}=\boldsymbol{\mu}_{\mathbf{x}}}\right]^{\mathbf{-1}}$, such that $(P(\mathbf{x} | \{\mathbf{y}_i\}_{i=1}^{N}, \ldots) \approx N(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}})$. When using Newton-Raphson

(NR), $H\backslash\nabla$ in MATLAB will make use of block tri-diagonal structure automatically.

However, NR is not robust to bad initials. At the first few iterations, simply using fitting from previous step may lead to infinite Hessian. When the initial from previous step fails, use the approximation at recursive priors, , i.e. the adaptive smoother estimates, as the initial. The adaptive smoother estimates are from backward RTS smoother from adaptive filter, and the details about Poisson adaptive filter can be found in Eden et al., 2004.

To sample efficiently and make best use of sparse covariance, use Cholesky decomposition of $\mathbf{\Sigma}_{\mathbf{x}}^{-1} = \mathbf{R}\mathbf{R}'$: sample $\mathbf{Z} \sim N(\mathbf{R}'\mu_{\mathbf{x}}, \mathbf{I})$, then $\mathbf{x} = (\mathbf{R}')^{-1}\mathbf{Z} \sim N(\boldsymbol{\mu}_{\mathbf{x}}, \mathbf{\Sigma}_{\mathbf{x}})$.

Step Further:
Using normal approximation may not be accurate enough, and this may lead to a bad mixing. We can step further to use the Metropolis-Hasting, based on the normal approximated variance. To be specific, the proposal distribution is $Q(\mathbf{x}^*|\mathbf{x}^{(s)}) \sim N(\mathbf{x}^{(s)}, \alpha\mathbf{\Sigma}_{\mathbf{x}})$, where α is a scalar to make the acceptance rate to be 0.4 to 0.6. According to Roberts and Rosenthal, 2001. For high dimensional MH, the optimal proposal is $N(x, 2.38^2\Sigma/d)$. See details & experiments in Section 5: Simulations.

3.3.2 Update $\mathbf{x}_0$

$$P\left(\mathbf{x}_0|\mathbf{x}_1,\ \mathbf{Q}_0\ldots\right) \propto N(\mathbf{x}_1|\mathbf{x}_0,\ \mathbf{Q}_0)N(\mathbf{x}_0|\boldsymbol{\mu}_{\mathbf{x}_{00}},\ \mathbf{\Sigma}_{\mathbf{x}_{00}})$$

By conjugacy, $\mathbf{x}_0|\mathbf{x}_1,\ \mathbf{Q}_0\ldots \sim N(\boldsymbol{\mu}_{\mathbf{x}_0},\ \mathbf{\Sigma}_{\mathbf{x}_0})$

$$\begin{aligned}\mathbf{\Sigma}_{\mathbf{x}_0} &= \left[\mathbf{\Sigma}_{\mathbf{x}_{00}}^{-1} + \mathbf{Q}_0^{-1}\right]^{-1}\\ \boldsymbol{\mu}_{\mathbf{x}_0} &= \mathbf{\Sigma}_{\mathbf{x}_0}\left(\mathbf{\Sigma}_{\mathbf{x}_{00}}^{-1}\boldsymbol{\mu}_{\mathbf{x}_{00}} + \mathbf{Q}_0^{-1}\mathbf{x}_1\right)\end{aligned}$$

3.3.3 Update $\{d_i\}_{i=1}^N$ and $\{\mathbf{c}_i\}_{i=1}^N$

To update efficiently, use Laplace approximation and the error is independent for each row (i.e. update things row by row). Denote $(d_i, \mathbf{c}_i')' = \boldsymbol{\zeta}_i \in \mathbb{R}^{p+1}$ and $\left(1, \mathbf{x}_t'^{(z_i)}\right) = \widetilde{\mathbf{x}}_t'^{(z_i)}$.

$$P\left(\boldsymbol{\zeta}_i \middle| \mathbf{y}_i,\ \left\{\mathbf{x}_t^{(z_i)}\right\}_{t=1}^T, \ldots\right) = \exp f\left(\boldsymbol{\zeta}_i\right) \approx N\left(\boldsymbol{\zeta}_i|\boldsymbol{\mu}_{\boldsymbol{\zeta}_i}, \mathbf{\Sigma}_{\boldsymbol{\zeta}_i}\ \right)$$

$$\begin{aligned}\frac{\partial f}{\partial \boldsymbol{\zeta}_i} &= \frac{\partial l}{\partial \boldsymbol{\zeta}_i} - \mathbf{\Sigma}_{\text{dc}}^{(z_i)^{-1}}\left(\boldsymbol{\zeta}_i - \boldsymbol{\mu}_{\text{dc}}^{(z_i)}\right) = \left[\sum_{t=1}^T \widetilde{\mathbf{x}}_t^{(z_i)}\left(y_{\text{it}} - \lambda_{\text{it}}\right)\right] - \mathbf{\Sigma}_{\text{dc}}^{(z_i)^{-1}}\left(\boldsymbol{\zeta}_i - \boldsymbol{\mu}_{\text{dc}}^{(z_i)}\right)\\ \frac{\partial^2 f}{\partial \boldsymbol{\zeta}_i \partial \boldsymbol{\zeta}_i'} &= \frac{\partial^2 l}{\partial \boldsymbol{\zeta}_i \partial \boldsymbol{\zeta}_i'} - \mathbf{\Sigma}_{\text{dc}}^{(z_i)^{-1}} = -\left[\sum_{t=1}^T \lambda_{\text{it}}\widetilde{\mathbf{x}}_t^{(z_i)}\widetilde{\mathbf{x}}_t'^{(z_i)}\right] - \mathbf{\Sigma}_{\text{dc}}^{(z_i)^{-1}}\end{aligned}$$

, where l is Poisson log-likelihood. Then use Newton-Raphson to find $\boldsymbol{\mu}_{\boldsymbol{\zeta}_i} = \text{argmax}_{\boldsymbol{\zeta}_i}\left(f\left(\boldsymbol{\zeta}_i\right)\right)$ and $\boldsymbol{\Sigma}_{\boldsymbol{\zeta}_i} = -\left[\frac{\partial^2 f}{\partial \boldsymbol{\zeta}_i \partial \boldsymbol{\zeta}_i'} |_{\boldsymbol{\zeta}_i = \boldsymbol{\mu}_{\boldsymbol{\zeta}_i}}\right]^{-1}$. If initial as the previous step fits fails, simply use prior mean of $\boldsymbol{\mu}_{\boldsymbol{\zeta}_i}$, i.e. $\boldsymbol{\delta}_{dc0}$.

Step further: Again, use MH to sample the posterior, based on the normal approximated variance. Since the dimension of d and C is not large (just d = 3) in the simulation, I didn't use the optimal scalar yet. **Modify it later**.

3.3.4 Update $\left\{\mu_{\mathbf{dc}}^{(j)}\right\}_j$ and $\left\{\boldsymbol{\Sigma}_{\mathbf{dc}}^{(j)}\right\}_j$

Purpose: to make loading within each cluster depends on each other, and this will help with clustering. As above, denote $(d_i, \mathbf{c}_i')' = \boldsymbol{\zeta}_i \in \mathbb{R}^{p+1}$.

(1) Mean $\left\{\boldsymbol{\mu}_{\text{dc}}^{(j)}\right\}_j$: by conjugacy, $\boldsymbol{\mu}_{\text{dc}}^{(j)} \sim N\left(\boldsymbol{\delta}_{\text{dc}}, \mathbf{T}_{\text{dc}}\right)$

$$\mathbf{T}_{\text{dc}}^{-1} = \left(\mathbf{T}_{dc0}^{-1} + n_j \boldsymbol{\Sigma}_{\text{dc}}^{(j)^{-1}}\right)^{-1}$$

$$\boldsymbol{\delta}_{\text{dc}} = \mathbf{T}_{\text{dc}}\left(\mathbf{T}_{dc0}^{-1}\boldsymbol{\delta}_{dc0} + \boldsymbol{\Sigma}_{\text{dc}}^{(j)^{-1}} \sum_{i:z_i=j} \boldsymbol{\zeta}_i\right)$$

(2) Covariance $\left\{\boldsymbol{\Sigma}_{\text{dc}}^{(j)}\right\}_j$: by conjugacy, $\boldsymbol{\Sigma}_{\text{dc}}^{(j)} \sim W^{-1}\left(\Psi_{\text{dc}}, \nu_{\text{dc}}\right)$

$$\nu_{\text{dc}} = n_j + \nu_{dc0}$$

$$\Psi_{\text{dc}} = \Psi_{dc0} + \sum_{i:z_i=j}\left(\boldsymbol{\zeta}_i - \boldsymbol{\mu}_{\text{dc}}^{(j)}\right)\left(\boldsymbol{\zeta}_i - \boldsymbol{\mu}_{\text{dc}}^{(j)}\right)'$$

3.3.5 Update A, b and Q

Here, I only give the update for full $\mathbf{Q}$. If we want to assume block-diagonal or diagonal structure of $\mathbf{Q}$, just update things cluster-by-cluster or latent-by-latent.

As shown before, the problem is the usual Bayesian MV-GLM problem. Denote $\mathbf{F} = \begin{pmatrix}\mathbf{b}' \\ \mathbf{A}'\end{pmatrix}$, $\mathbf{f} = vec(\mathbf{F})$, $\mathbf{Y}_{\mathbf{bA}} = (\mathbf{x}_2, \mathbf{x}_3 \ldots, \mathbf{x}_T)'$ and $\mathbf{X}_{\mathbf{bA}} = \begin{pmatrix}1 & \mathbf{x}_1' \\ 1 & \mathbf{x}_2' \\ \vdots & \vdots \\ 1 & \mathbf{x}_{T-1}'\end{pmatrix}$

$$\mathbf{Y}_{\mathbf{bA}} = \mathbf{X}_{\mathbf{bA}}\mathbf{F} + \mathbf{E}$$

, where $\mathbf{E} = (\epsilon_1, \ldots, \epsilon_{T-1})'$ and $\epsilon \sim N(0, \mathbf{Q})$. Then posteriors are

$$\mathbf{Q}|\mathbf{Y}_{\text{bA}}, \mathbf{X}_{\text{bA}}, \ldots \sim W^{-1}(\Psi_{\mathbf{Q}}, \nu_{\mathbf{Q}})$$

$$\mathbf{f}|\mathbf{Y}_{\text{bA}}, \mathbf{X}_{\text{bA}}, \mathbf{Q}, \ldots \sim N(vec(\mathbf{F}_{\mathbf{bA}}), \mathbf{Q} \bigotimes \boldsymbol{\Lambda}_{\mathbf{bA}}^{-1})$$

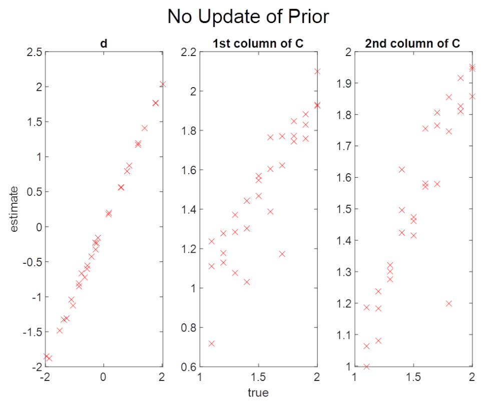
, with parameters be:

$$\begin{aligned}
\Psi_{\mathbf{Q}} &= \Psi_{\mathbf{Q}_0} + (\mathbf{Y}_{\mathbf{bA}} - \mathbf{X}_{\mathbf{bA}}\mathbf{F}_{\mathbf{bA}})'(\mathbf{Y}_{\mathbf{bA}} - \mathbf{X}_{\mathbf{bA}}\mathbf{F}_{\mathbf{bA}}) + (\mathbf{F}_{\mathbf{bA}} - \mathbf{F}_0)'\mathbf{\Lambda}_0(\mathbf{F}_{\mathbf{bA}} - \mathbf{F}_0)\\
\nu_{\mathbf{Q}} &= \nu_{\mathbf{Q}_0} + T - 1\\
\mathbf{F}_{\mathbf{bA}} &= (\mathbf{X}'_{\mathbf{bA}}\mathbf{X}_{\mathbf{bA}} + \mathbf{\Lambda}_0)^{-1}(\mathbf{X}'_{\mathbf{bA}}\mathbf{Y}_{\mathbf{bA}} + \mathbf{\Lambda}_0\mathbf{F}_0)\\
\mathbf{\Lambda}_{\mathbf{bA}} &= \mathbf{X}'_{\mathbf{bA}}\mathbf{X}_{\mathbf{bA}} + \mathbf{\Lambda}_0
\end{aligned}$$

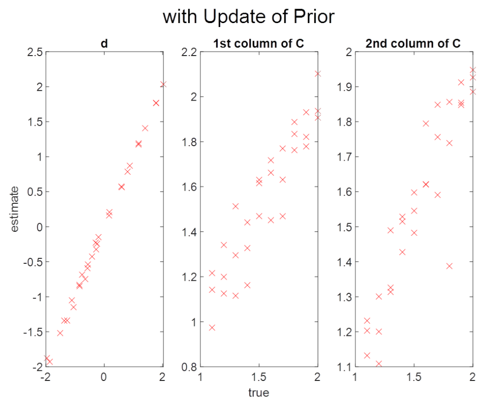
4 Model Check [Problems for b and A partially solved]

4.1 Estimation of d and C alone

First, turn off all others except for loading (related) parameters and set them to be true values. There are 2 versions: (1) update priors as in subsection 3.3.4 and (2) no update of priors, and just set the prior for each as the standard normal. For the estimation with prior update, the results are mean from iteration 50 to 100 (Figure 1).



(a)



(b)

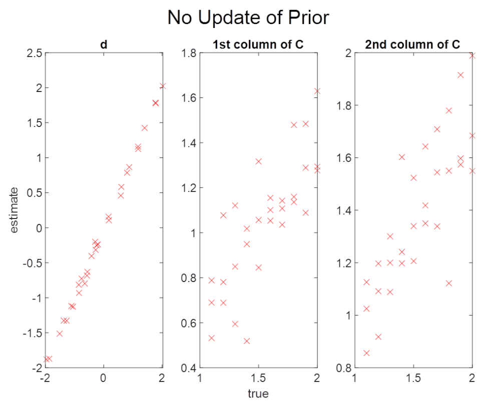
Figure 1: estimation of loading

The mixing of convergence for loading priors (in subsection 3.3.4) are fine. The update of priors don't influence results a lot, but this will help clustering a lot. It seems estimation of loading is fine. Let's see what happens when estimation of latents $\mathbf{x}_t$ is on at the same time.

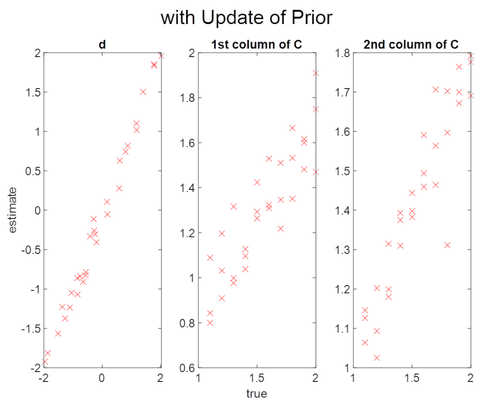
4.2 Estimation of d, C and $\mathbf{x}_t$

When the estimation of $\mathbf{x}_t$ is on, we need more iterations. The results are mean from iteration 500 to 1000 (Figure 2).

The estimation of loading:



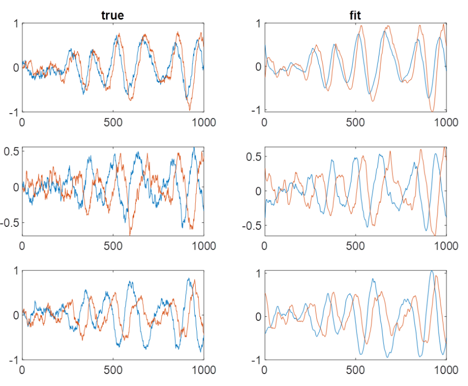
(a)



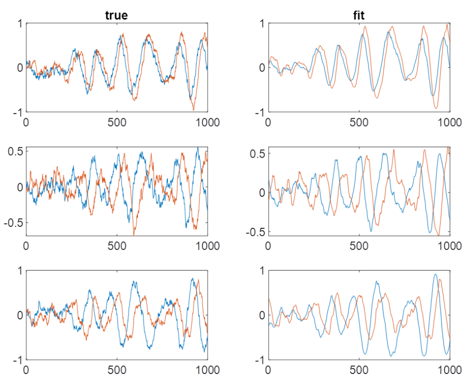
(b)

Figure 2: estimation of loading, with latents on

And the estimation of corresponding latents (Figure 3):



(a) no update of prior



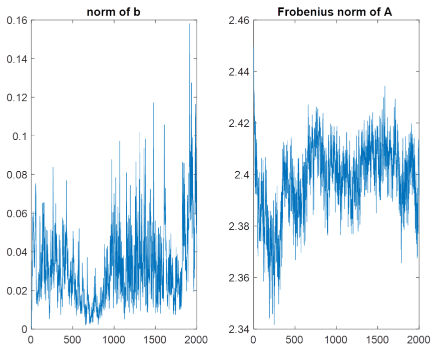
(b) with update of prior

Figure 3: estimation of loading

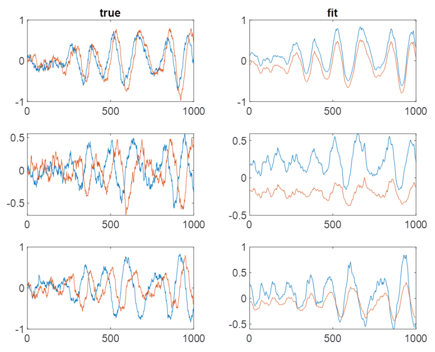
Things are still fine. The remaining parts may influence latent estimations are dynamics ($\mathbf{b}$, $\mathbf{A}$) and prior process noise $\mathbf{Q}$.

4.3 Estimation of b, A, Q and $\mathbf{x}_t$

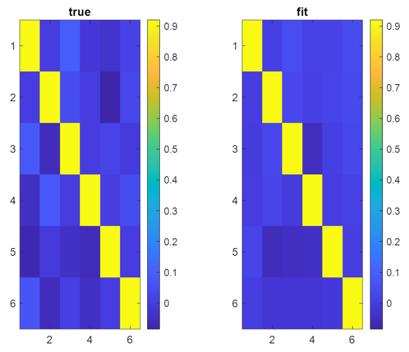
I show the results of block-diagonal (Figure 4) and full (Figure 5) **Q**. These results are from 2000 MCMC samples. The latent and dynamics are from average from 500 to 2000 iterations.



(a) trace

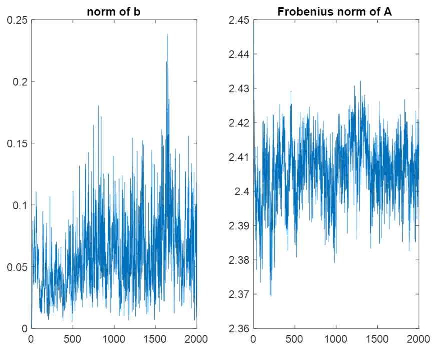


(b) latent

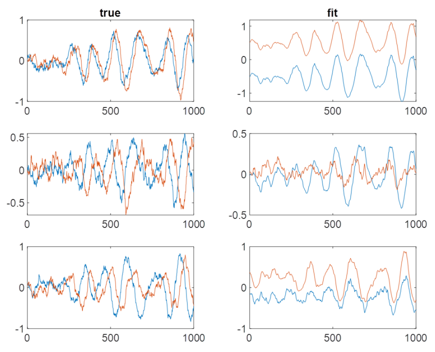


(c) dynamics **A**

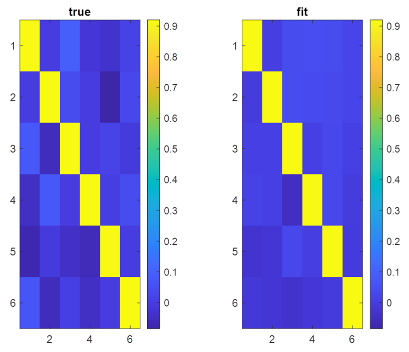
Figure 4: sampling of **b**, **A**, **Q** and $\mathbf{x}_t$, block-diagonal process noise



(a) trace



(b) latent



(c) dynamics **A**

Figure 5: sampling of **b**, **A**, **Q** and $\mathbf{x}_t$, full process noise

Both block-diagonal and full looks fine, but the mixing of full process noise version looks better.

5 Simulations

There are two set of simulation examples. The first example is generated from LDS model directly, while in the second example the latents are generated directly without explicit specifying linear dynamics. In all following results, I fit things with both block-diagonal and full $\mathbf{Q}$. The fitting results for block-diagonal version is a bit better, because the underlying true $\mathbf{Q}$ is diagonal. In the following part, I only show results from block-diagonal fitting for labeled data. For unlabeled data, i.e. clustering, all the results can be found in this folder.

5.1 Simulation 1: Generate from LDS Directly

In this simulation, there are 3 clusters with 10 neurons in each cluster. The dimension of latents in each cluster is 2. In the linear dynamics, the bias term $\mathbf{b}$ is zero. There's no within-population interaction but has some weak between-population interactions. In other words, the linear dynamics matrix $\mathbf{A}$ is roughly diagonal. The details of simulation can be found in the first section of the simulation 1.

5.1.1 Labeled Data: No Clustering

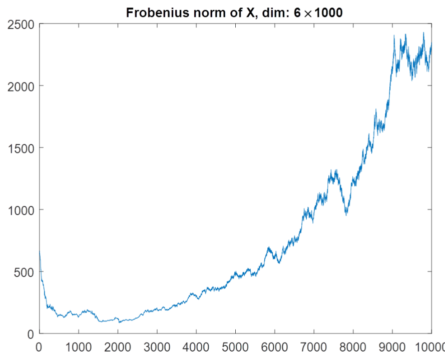
The code for fitting is here.

Here, I first compare 10k iterations for three methods (consider high dimension of $\mathbf{x}_t$): (1) both $\mathbf{x}_t$ and loading ($\mathbf{d}$ and $\mathbf{C}$) are updated by normal approximation. (2) update $\mathbf{x}_t$ by normal approximation, but update loading by MH. (3) update both $\mathbf{x}_t$ and loading by MH.

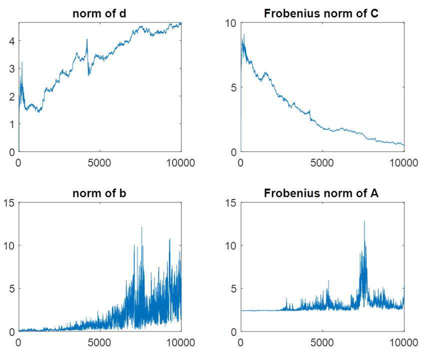
The fitting results are average of iteration 5k to 10k.

Update $\mathbf{x}_t$ and (d, C) by normal approximation

The trace plots are in Figure 6



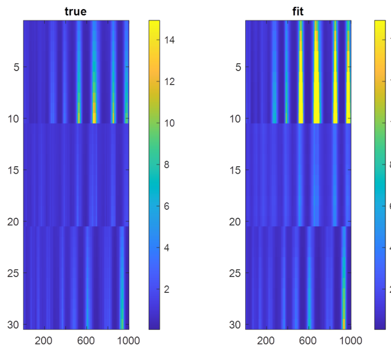
(a) tracd of latent



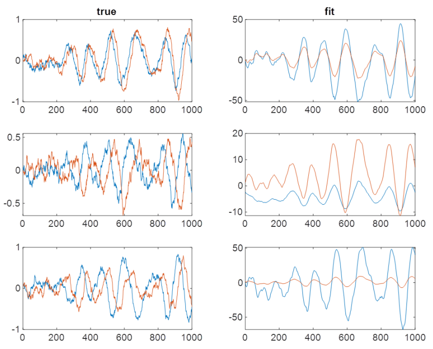
(b) trace of loading and dynamics $\mathbf{A}$

Figure 6: both Normal Approx., trace plots

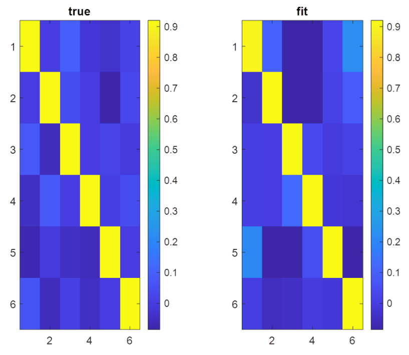
Well, the chain hasn't achieve the stationary distribution. So there's no need to see further results. But to show the improvement and for comparison, I still show the fitting of overall firing rate, latents and dynamics in Figure 7.



(a) overall mean firing rate



(b) latents

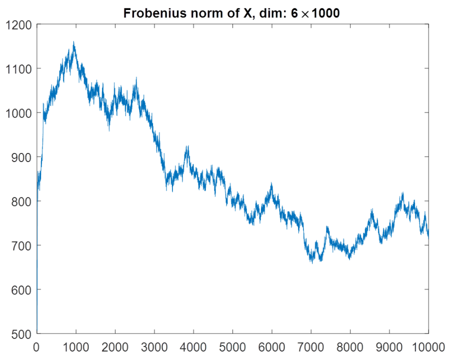


(c) dynamics $\mathbf{A}$

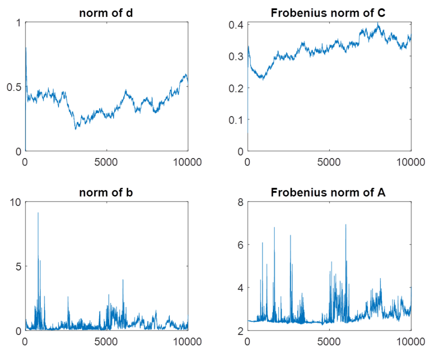
Figure 7: both Normal Approx., fit

Update $\mathbf{x}_t$ by normal approximation and (d, C) by MH

By using the normal approximated posterior variance as the proposal variance, the acceptance rates are around 0.45 (Since there are 30 neurons, there are 30 acceptance rates). The trace plots are in Figure 8



(a) tracd of latent

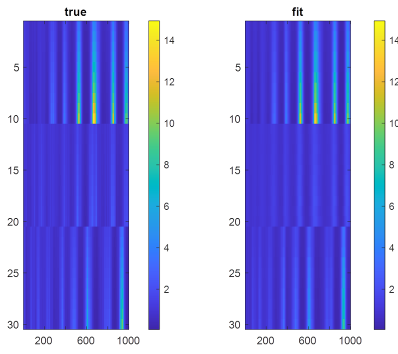


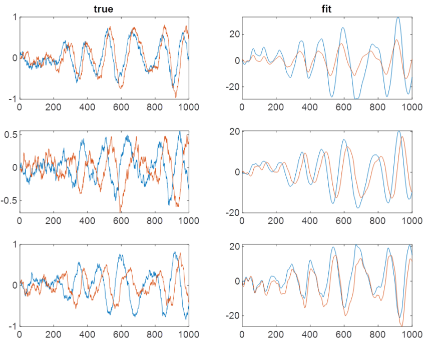
(b) trace of loading and dynamics $\mathbf{A}$

Figure 8: latent Normal Approx. + loading MH, trace plots

Looks much better, although still not perfect yet. The fitting of overall firing rate, latents and dynamics are in Figure 9.

See, nearly perfect (especially) for latent.


(a) overall mean firing rate


(b) latents

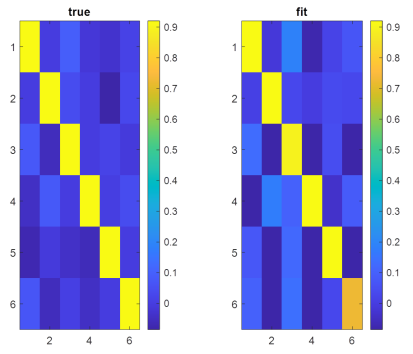
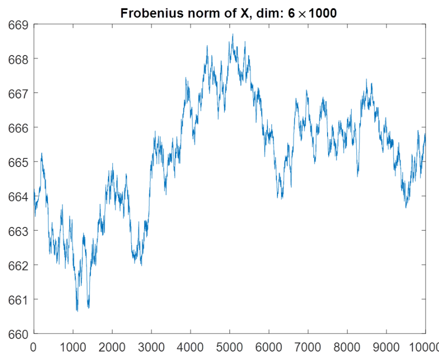

(c) dynamics $\mathbf{A}$

Figure 9: latent Normal Approx. + loading MH, fit

Update both $\mathbf{x}_t$ and (d, C) by MH

The acceptance rate (by using the optimal scalar) for latent is 0.6227, which is a bit high (but acceptable). Again, the acceptance rates for loading are around 0.45.

The trace plots are in Figure 10


(a) tracd of latent

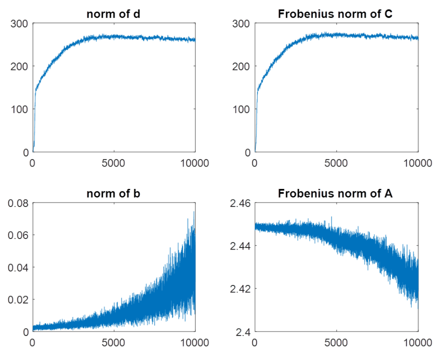
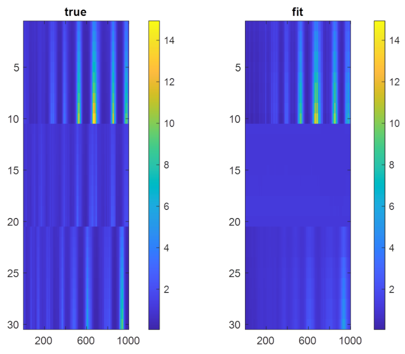

(b) trace of loading and dynamics $\mathbf{A}$

Figure 10: both MH, trace plots

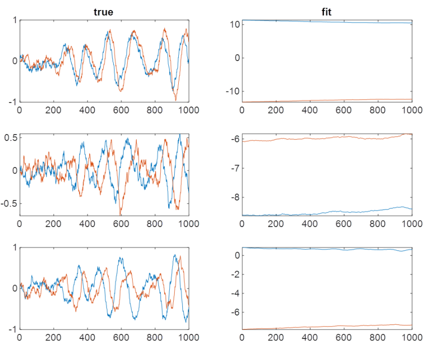
The trace for dynamics is terrible... Maybe this is caused by high dimension of latent ($d = 6000$ in this case).

The fitting of overall firing rate, latents and dynamics are in Figure 11.

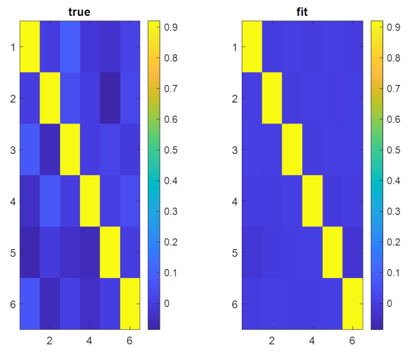
OK, the fitting sucks...



(a) overall mean firing rate



(b) latents



(c) dynamics $\mathbf{A}$

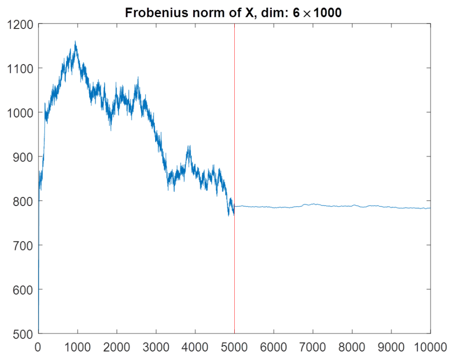
Figure 11: both MH, fit

One natural idea is to combine MH and normal approximation in latent: at first few steps, use normal approximation at conditional posterior mode to bring things into somewhere close to mode quickly. Then use MH to stabilize everything.

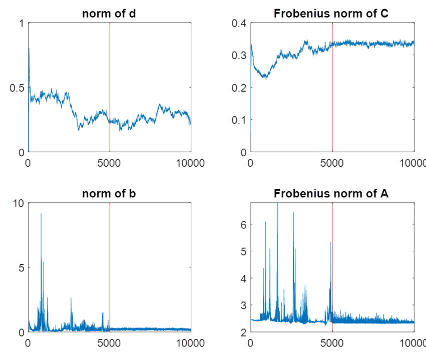
In the following, I still sample loading by MH. But for latent, the first half iteration is sampled by normal approximation, while the remainder is sampled from MH.

The red line marks when I switch the sampling method for $\mathbf{x}_t$.

The trace plots are in Figure 12



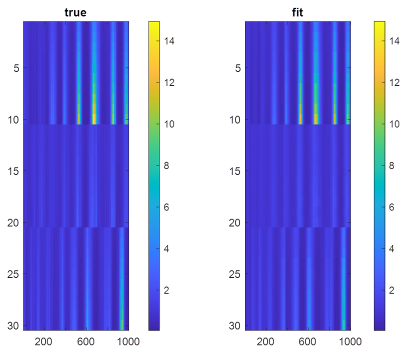
(a) tracd of latent



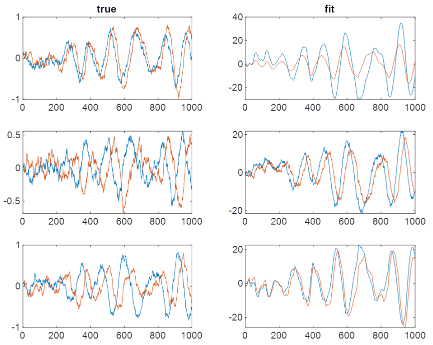
(b) trace of loading and dynamics $\mathbf{A}$

Figure 12: half MH for latent, trace plots

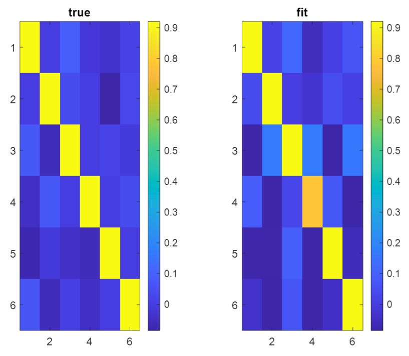
The fitting of overall firing rate, latents and dynamics are in Figure 13.



(a) overall mean firing rate



(b) latents



(c) dynamics **A**

Figure 13: half MH for latent, fit

Well, everything is stable now. But this implementation is too naive. Maybe use some more advanced adaptive sampling method?

I also fit the model by assuming one cluster. Again, the fitting for overall mean firing rate is perfect. This is why it's important to make the loading, **d** and **C**, also be cluster-dependent (loading within clusters are correlated).

If the loading only depends on neuron index and will not change for different clustering assignments, it's impossible to do clustering (at least in this case), since the loading is enough to capture all the patterns.

5.1.2 Unlabeled Data: Clustering

To give the full path of clustering, I show results in GIFs. All results can be found in this folder. The code can be found in this folder.

Basically, in my current implementation, algorithms are good to merge clusters. However, generating new clusters is very hard... In other words, the newly generated cluster will usually not be sampled.

This is a big problem. This makes DPMM loses its power. **Fix that later**.

5.2 Simulation 2: Generate Latents, without Specifying Linear Dynamics

In this simulation, there are again 3 clusters with 2 latents and 10 neurons in each. Now, the latents are generated directly without specifying the underlying linear dynamics of latents. However, each latent is generated independently, so the linear dynamics matrix **A** should be roughly diagonal. The details of simulation can be found in the code.

5.2.1 Labeled Data: No Clustering

As in simulation 1, the data is fitted by the true cluster assignment (3 clusters) and forcing all neurons belong to single cluster. The code can be found here.

I have deleted the old results to avoid confusion. I will post the new results here later.

5.2.2 Unlabeled Data: Clustering

The clustering results are old. Update when the MFM is implemented

Again, all results can be found in this folder. The code can be found in this folder.

Besides fitting 3 clusters with 10 neurons each, I further fit a larger scale example (50 neurons for each cluster).

6 TODO

(1) Do MH adaptively.

(2) Implement the mixture of finite mixture (MFM) by Jeff Miller.

(3) Find a more efficient way to generate new cluster parameters, otherwise the newly generated ones will always be rejected.

(4) After all of them are resolved, switch to GP version