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 $\mathbf{y}_i = (y_{i1}, \ldots, y_{\mathrm{iT}})'$, $i = 1, \ldots N.$, 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 $\delta_1 = \ldots = \delta_J = 1$

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 $\mathbf{\Theta}_j$: 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.

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

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

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:

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

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

, 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,

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

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

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

(5) $\mathbf{x}_0 \in \mathbb{R}^{Jp}$ and $\mathbf{Q}_0 \in \mathbb{R}^{Jp \times Jp}$: 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:

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

, 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 $\mathbf{\Theta'} = \mathbf{\Theta} \backslash \{\mathbf{x}_t\}_{t=1}^T$.

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

, 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: $\left\{\boldsymbol{\mu}_{\text{dc}}^{(j)}\right\}_j$ and $\left\{\mathbf{\Sigma}_{\text{dc}}^{(j)}\right\}_j$

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

(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 $\mathbf{T}_{dc0} = \mathbf{I}_{p+1}$

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

, 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 $\mathbf{F} = \begin{pmatrix} \mathbf{b}' \\ \mathbf{A}' \end{pmatrix}$, $\mathbf{f} = vec(\mathbf{F})$

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

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

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

$$\frac{\partial f}{\partial \boldsymbol{\zeta}_i} = \frac{\partial l}{\partial \boldsymbol{\zeta}_i} - \mathbf{\Sigma}_{\mathrm{dc}}^{(z_i)^{-1}}\left(\boldsymbol{\zeta}_i - \boldsymbol{\mu}_{\mathrm{dc}}^{(z_i)}\right) = \left[\sum_{t=1}^T \widetilde{\mathbf{x}}_t^{(z_i)}\left(y_{\mathrm{it}} - \lambda_{\mathrm{it}}\right)\right] - \mathbf{\Sigma}_{\mathrm{dc}}^{(z_i)^{-1}}\left(\boldsymbol{\zeta}_i - \boldsymbol{\mu}_{\mathrm{dc}}^{(z_i)}\right)$$

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

, 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 $\boldsymbol{\Sigma}_{\boldsymbol{\zeta}_i} = -\left[\frac{\partial^2 f}{\partial \boldsymbol{\zeta}_i \partial \boldsymbol{\zeta}_i'} \mid_{\boldsymbol{\zeta}_i = \boldsymbol{\mu}_{\boldsymbol{\zeta}_i}}\right]^{-1}$. 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 $\left(d_i, \mathbf{c}_i'\right)' = \boldsymbol{\zeta}_i \in \mathbb{R}^{p+1}$.

(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 $\mathbf{F} = \begin{pmatrix}\mathbf{b}' \\ \mathbf{A}'\end{pmatrix}$, $\mathbf{f} = vec(\mathbf{F})$, $\mathbf{Y_{bA}} = (\mathbf{x}_2, \mathbf{x}_3 \ldots, \mathbf{x}_T)'$ and $\mathbf{X_{bA}} = \begin{pmatrix}1 & \mathbf{x}_1' \\ 1 & \mathbf{x}_2' \\ \vdots & \vdots \\ 1 & \mathbf{x}_{T-1}'\end{pmatrix}$

$$\mathbf{Y_{bA}} = \mathbf{X_{bA}}\mathbf{F} + \mathbf{E}$$

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

$$\mathbf{Q}|\mathbf{Y_{bA}}, \mathbf{X_{bA}}, \ldots \sim W^{-1}(\Psi_{\mathbf{Q}}, \nu_{\mathbf{Q}})$$

$$\mathbf{f}|\mathbf{Y_{bA}}, \mathbf{X_{bA}}, \mathbf{Q}, \ldots \sim N(vec(\mathbf{F_{bA}}), \mathbf{Q} \bigotimes \boldsymbol{\Lambda}_{\mathbf{bA}}^{-1})$$

, with parameters be:

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

4 Simulations

All the simulation results are old and problematic. Update them later.

5 Problem from Affine Transformation

Let me review the model first.
Assume there are J clusters, with n neurons in each cluster. Each cluster is governed by p latent vectors. $\boldsymbol{\lambda}_t^{(j)} \in \mathbb{R}^n$, $\mathbf{x}_t^{(j)} \in \mathbb{R}^p$, $\mathbf{d}^{(j)} \in \mathbb{R}^n$ and $\mathbf{C}^{(j)} \in \mathbb{R}^{n\times p}$, with $j = 1, \ldots, J$, be the mean firing rate, latent state, intercept and loading (at time t) for each cluster. Then I just stack all the thing together, and denote corresponding things as $\boldsymbol{\lambda}_t \in R^{\mathrm{Jn}}$, $\mathbf{x}_t \in \mathbb{R}^{Jp}$, $\mathbf{d} \in \mathbb{R}^{\mathrm{Jn}}$ and $\mathbf{C} \in \mathbb{R}^{Jn\times Jp}$. Notice $\mathbf{C}$ is block diagonal.

$$
\log \begin{pmatrix} \boldsymbol{\lambda}_t^{(1)} \\ \vdots \\ \boldsymbol{\lambda}_t^{(J)} \end{pmatrix} = \log \boldsymbol{\lambda}_t = \begin{pmatrix} \mathbf{d}^{(1)} \\ \vdots \\ \mathbf{d}^{(J)} \end{pmatrix} + \begin{pmatrix} \mathbf{C}^{(1)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{C}^{(J)} \end{pmatrix} \begin{pmatrix} \mathbf{x}_t^{(1)} \\ \vdots \\ \mathbf{x}_t^{(J)} \end{pmatrix} = \mathbf{d} + \mathbf{C}\mathbf{x}_t
$$

The latent states progress linearly with a Gaussian noise:

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

, where $\mathbf{A} \in \mathbb{R}^{Jp\times Jp}$, $\mathbf{b} \in \mathbb{R}^{Jp}$ and $\mathbf{Q} \in \mathbb{R}^{Jp\times Jp}$.

Now assume we take an affine transformation of latent states as $\mathbf{x}_t^* = \mathbf{M}\mathbf{x}_t + \mathbf{g}$. To preserve the block diagonal structure of $\mathbf{C}$, $\mathbf{M}$ is also block diagonal. Then,

$$
\begin{aligned}
\log \boldsymbol{\lambda}_t &= \mathbf{d} + \mathbf{C}\mathbf{M}^{-1}\left(\mathbf{M}\mathbf{x}_t + \mathbf{g}\right) - \mathbf{C}\mathbf{M}^{-1}\mathbf{g} \\
&= \left(\mathbf{d} - \mathbf{C}\mathbf{M}^{-1}\mathbf{g}\right) + \mathbf{C}\mathbf{M}^{-1}(\mathbf{M}\mathbf{x}_t + \mathbf{g}) \\
\mathbf{M}\mathbf{x}_{t+1} + \mathbf{g}|\mathbf{x}_t &\sim N_{Jp}\left(\mathbf{M}\left(\mathbf{A}\mathbf{x}_t + \mathbf{b}\right) + \mathbf{g}, \mathbf{M}\mathbf{Q}\mathbf{M}'\right) \\
&= N_{Jp}(\mathbf{M}\mathbf{A}\mathbf{M}^{-1}\left(\mathbf{M}\mathbf{x}_t + \mathbf{g}\right) + \left(\mathbf{M}\mathbf{b} + \mathbf{g} - \mathbf{M}\mathbf{A}\mathbf{M}^{-1}\mathbf{g}\right), \mathbf{M}\mathbf{Q}\mathbf{M}')
\end{aligned}
$$

That means, when $\{\mathbf{d}, \mathbf{C}, \mathbf{x}_t, \mathbf{A}, \mathbf{b}, \mathbf{Q}\}$ is a solution, $\{\mathbf{d}^*, \mathbf{C}^*, \mathbf{x}_t^*, \mathbf{A}^*, \mathbf{b}^*, \mathbf{Q}^*\}$ is also a solution, where $\mathbf{d}^* = \mathbf{d} - \mathbf{C}\mathbf{M}^{-1}\mathbf{g}$, $\mathbf{C}^* = \mathbf{C}\mathbf{M}^{-1}$, $\mathbf{x}_t^* = \mathbf{M}\mathbf{x}_t + \mathbf{g}$, $\mathbf{A}^* = \mathbf{M}\mathbf{A}\mathbf{M}^{-1}$, $\mathbf{b}^* = \mathbf{M}\mathbf{b} + \mathbf{g} - \mathbf{M}\mathbf{A}\mathbf{M}^{-1}\mathbf{g}$ and $\mathbf{Q}^* = \mathbf{M}\mathbf{Q}\mathbf{M}'$, for any block diagonal $\mathbf{M}$. Therefore, we must add some constraints to ensure the unique solution/ stationary posterior!

5.1 Constraints

5.1.1 Constraint on latent vectors $\mathbf{x}_t$ only

The easiest way to constraint is to normalize or standardize each row of $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$.

Let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T) = \begin{pmatrix} \mathbf{X}'_{(1)} \\ \vdots \\ \mathbf{X}'_{(Jp)} \end{pmatrix}$, where $\mathbf{x}_{(k)} \in \mathbb{R}^T$ is the k-th row vector.

Then,

(1) Normalization: $\mathbf{x}^*_{(k)} = \frac{\mathbf{x}_{(k)} - \min \mathbf{x}_{(k)}}{\max \mathbf{x}_{(k)} - \min \mathbf{x}_{(k)}}$

(2) Standardization: $\mathbf{x}^*_{(k)} = \frac{\mathbf{x}_{(k)} - \bar{\mathbf{x}}_{(k)}}{S(\mathbf{x}_{(k)})}$

The purpose of this constraint is to let $\boldsymbol{d}$ capture the translation and $\boldsymbol{C}$ capture the scaling.

After normalization/ standardization, the solution should be unique.

5.1.2 Joint constraint on $\mathbf{x}_t$, C and linear dynamics

This is perfect when cluster is known (more robust than constraints on $\boldsymbol{x}_t$ only), but when doing clusters, this is not very useful... Since the constraint depends on the number of elements in the cluster, but the number of elements will change when doing clustering.

See details in the appendix.

5.2 Constraint Implementation

The constraint for normalization and standardization are fairly easy: just do transformation of the MCMC sample in each iteration.

Normalization and standardization are both linear transformation, so after some algebra, we can even write the full conditional distribution explicitly.

For example, let's write out the linear transformation for centering. Denote the k-th row mean as $\overline{\mathbf{x}}_{(k)} = \frac{1}{T}\mathbf{x}'_{(k)}\mathbf{1}_T$. The unconstraint full conditional distribution of $vec(\mathbf{X})$ is derived in previous section. After normalization in each iteration,

we are actually sampling from $P(vec(\mathbf{X}) - \mathbf{1}_T \bigotimes \begin{pmatrix} \overline{\mathbf{x}}_{(1)} \\ \vdots \\ \overline{\mathbf{x}}_{(Jp)} \end{pmatrix} | \mathbf{Y}, \ldots)$. Notice that,

$$\mathbf{1}_T \bigotimes \begin{pmatrix} \overline{\mathbf{x}}_{(1)} \\ \vdots \\ \overline{\mathbf{x}}_{(Jp)} \end{pmatrix} = \mathbf{1}_T \bigotimes \frac{1}{T} \mathbf{X} \mathbf{1}_T = \frac{1}{T} vec\left(\mathbf{1} \mathbf{X}_T \mathbf{1}_T^{'}\right) = \frac{\mathbf{J}_T \bigotimes \mathbf{I}_{Jp}}{T} vec(\mathbf{X})$$

, then we are just sampling from full conditional distribution of $(\mathbf{I}_{JpT} - \frac{\mathbf{J}_T \bigotimes \mathbf{I}_{Jp}}{T}) vec(\mathbf{X})$

To help the convergence, we may further add constraint on linear dynamics. Most of them can be fairly implemented, by diagonal $\mathbf{A}$ with full $\mathbf{Q}$ need some algebra. See details in the appendix.

5.3 Simulation

Here, I ran both normalization and standardization for 10,000 iterations. Although pure normalization or standardization should be enough to ensure convergence, further constraints in linear dynamics may help convergence. Several constraints in linear dynamics (i.e. $\boldsymbol{A}$ and $\boldsymbol{Q}$) are also implemented:

(1) No constraints in $\boldsymbol{A}$ or $\boldsymbol{Q}$

(2) No constraints in $\boldsymbol{A}$, but $\boldsymbol{Q}$ is block-diagonal.

(3) No constraints in $\boldsymbol{A}$, but $\boldsymbol{Q}$ is diagonal.

(4) $\boldsymbol{A}$ is diagonal, but no constraints in $\boldsymbol{Q}$ (See details in appendix).

(5) Both $\boldsymbol{A}$ and $\boldsymbol{Q}$ are diagonal.

By limited experiments, it seems when $\boldsymbol{A}$ is diagonal, the convergence & mixing is best. If we put no constraint on $\boldsymbol{A}$, then putting no constraint on $\boldsymbol{Q}$ is the best. Here I show the 2-norm/ Frobenius norm of $\{\mathbf{d}, \mathbf{C}, \mathbf{X}, \mathbf{b}, \mathbf{A}, \mathbf{Q}\}$ for 4 constraints combos:

(1) Normalization + both $\boldsymbol{A}$ and $\boldsymbol{Q}$ are diagonal.

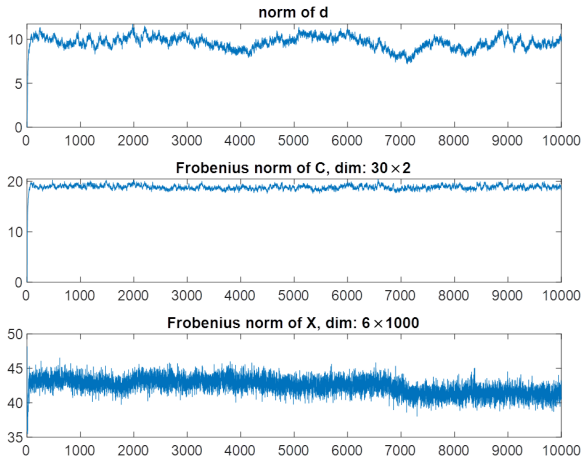
(2) Normalization + $\boldsymbol{A}$ diagonal but $\boldsymbol{Q}$ full.

(3) Standardization + both $\boldsymbol{A}$ and $\boldsymbol{Q}$ are diagonal.

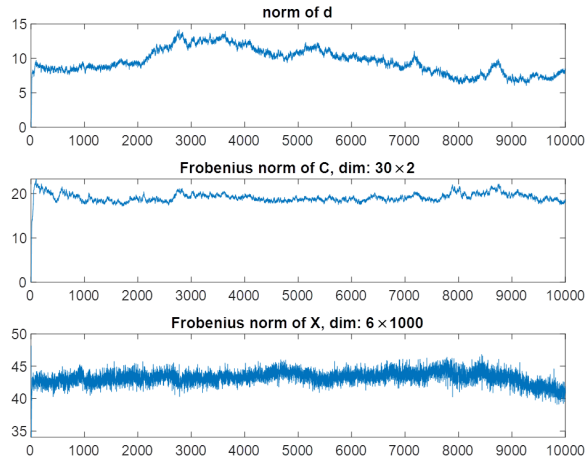
(4) Standardization + $\boldsymbol{A}$ diagonal but $\boldsymbol{Q}$ full.

5.4 Comments on Previous Models

When fitting the LDS or factor analysis (e.g. GPFA) model, I may only believe the temporal pattern or the change of the pattern. The pattern itself at a specific time point doesn't make a lot of sense, because of the affine transformation invariance.

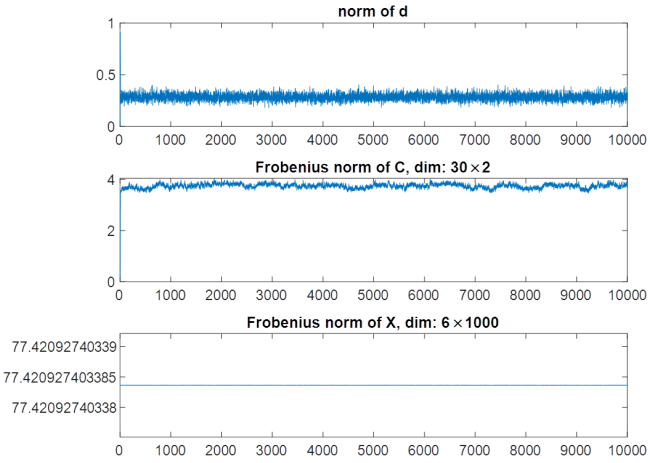


(a) normalization + $\boldsymbol{A}$ and $\boldsymbol{Q}$ diagonal

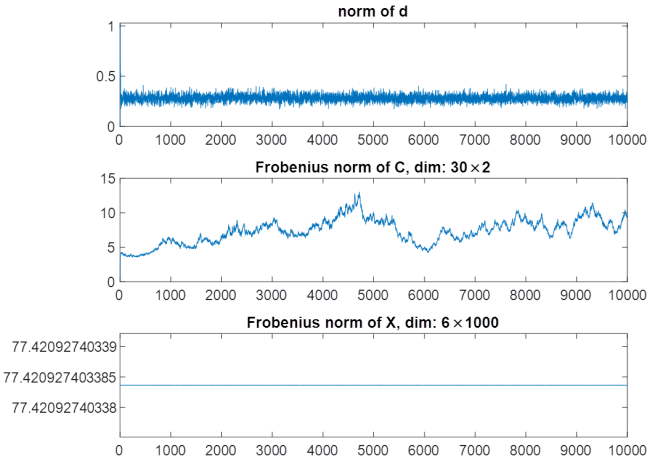


(b) normalization + $\boldsymbol{A}$ diagonal and $\boldsymbol{Q}$ full

Figure 1: normalization, full trace of $\boldsymbol{d}$, $\boldsymbol{C}$ and $\boldsymbol{X}$



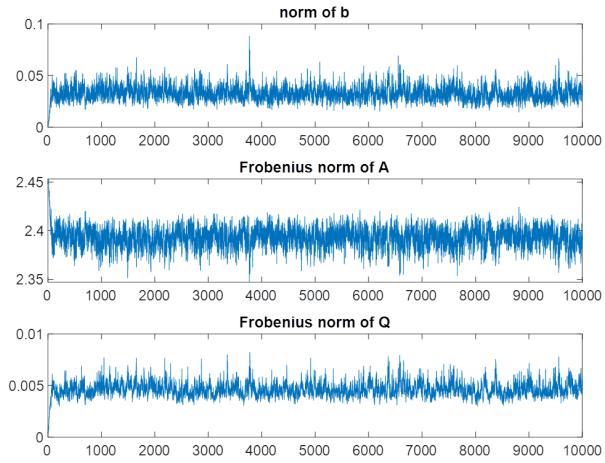
(a) standardization + $\boldsymbol{A}$ and $\boldsymbol{Q}$ diagonal



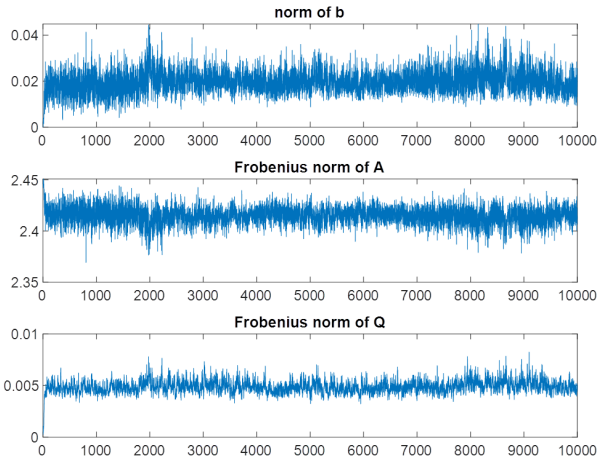
(b) standardization + $\boldsymbol{A}$ diagonal and $\boldsymbol{Q}$ full

Figure 2: standardization, full trace of $\boldsymbol{d}$, $\boldsymbol{C}$ and $\boldsymbol{X}$

In previous work, they fit models by EM, variational Bayes or some other MAP/ MLE method. **This is quite dangerous** if they don't check the uniqueness (Thanks to the gold standard MCMC, otherwise I will never check this issue...).

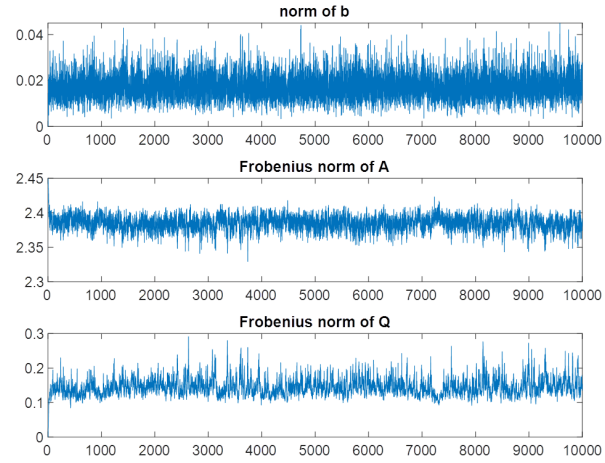


(a) normalization + $\boldsymbol{A}$ and $\boldsymbol{Q}$ diagonal

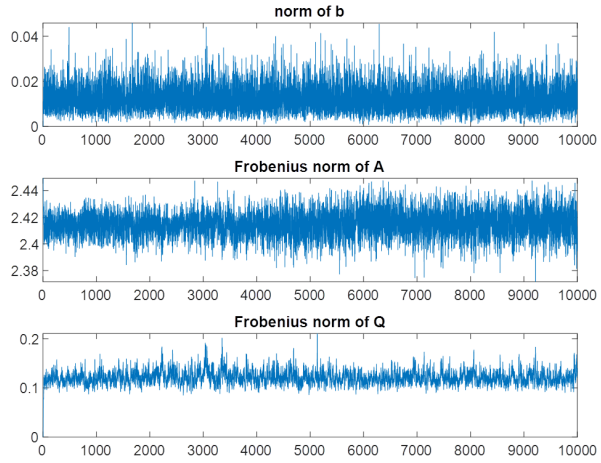


(b) normalization + $\boldsymbol{A}$ and $\boldsymbol{Q}$ full

Figure 3: normalization, full trace of $\boldsymbol{b}$, $\boldsymbol{A}$ diagonal and $\boldsymbol{Q}$



(a) standardization + $\boldsymbol{A}$ and $\boldsymbol{Q}$ diagonal



(b) standardization + $\boldsymbol{A}$ and $\boldsymbol{Q}$ full

Figure 4: standardization, full trace of $\boldsymbol{b}$, $\boldsymbol{A}$ diagonal and $\boldsymbol{Q}$

In the PLDS code that Ian shared me before, the default model has no constraint. They fit it by EM, and use ELBO as the convergence criteria. Although the parameters are not unique, the likelihood keeps the same, using ELBO is

justified. However, since there's no randomness in EM, it's impossible to detect the non-uniqueness. Actually, I think their results are even not local optima, but just a single point in the optimal surface (I guess if they use something like stochastic EM, they will find the issue).

The PLDS also provides some constraint options, but some of them are still not enough to ensure unique solution. BTW, this might be another reason they choose to use EM, since adding constraint in EM is trivial: just do unconstrained EM in each iteration and project the value to the constraint space (i.e. add constraint) at the end of each iteration. But this doesn't work for MCMC, since we are sampling the distribution but not just the single optimal value.

I guess similar problems may happen in other LDS/ GPFA research.

Even if they make enough constraints to ensure uniqueness, these research should be careful about the pattern they found. For example, in Joshua et al., 2020, the changes/ different states of interaction make sense to me, but I'm quite skeptical about their results in the dynamic matrix $\mathbf{A}$ itself. Maybe because of block-diagonal structure in loading $\mathbf{C}$, the interaction between population make sense somewhat. But at least the relationship within population doesn't make any sense, because of transformation invariance.

6 Some Issues in clustering

The first problem is how to update the unobserved components. In DP, this is the same as how to generate new components.

Traditionally, the unobserved ones are just sampled from prior. However, this doesn't work in my case, since the dimension of $\boldsymbol{X}$ is super high (6×1000) in simulation. It's nearly impossible to accept the newly generated components.

Here's my solution: other parameters are generated from prior, but $\boldsymbol{X}$ in new component is just the same as, or just another sample for the largest cluster.

In DP, we can do things even more aggressive: at the first few steps (e.g. pre-burn-in = 10), let all newly generated component parameters be the same (or samples from the same distribution) as the largest cluster. **This strategy encourages more clusters but keeps the clustering structure at the same time**.

After pre-burn-in, we goes back to normal: all parameters except for $\boldsymbol{X}$ are generated from priors. This gives the algorithm some chances to hit a "good" one, but the probability is not as high as in pre-burn-in.

The second problem comes when I let $\boldsymbol{Q}$ be full. For example, at one step, I have cluster index 1, 2, 4. But the sub-matrix of $\boldsymbol{Q}$ buying 1, 2, 4 columns and rows is not necessarily positive-definite. Here, I just do the easy thing: if the sub-matrix is not p.d., only keep the diagonal elements or block diagonal elements.

7 TODO

(1) Improve the NUTS.

(2) Debug the clustering.

(3) Check and clear typo.

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

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

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

8 Appendix

8.1 Joint constraint on $\mathbf{x}_t$, C and linear dynamics

For $\mathbf{x}_t$, we can row center it as previous. But instead of doing scaling on $\boldsymbol{x}_t$, we can put the scaling to the loading $\mathbf{C}$.

To mimic PCA, one natural way is to make the loading orthogonal. Here, I simply make $\mathbf{C}'^{(j)}\mathbf{C}^{(j)} = \mathbf{I}_p$.

However, this is not enough! Consider $\mathbf{x}_t^* = \mathbf{U}\mathbf{x}_t$, to preserve the orthogonality and block diagonal on $\mathbf{C}$, $\mathbf{U}$ must be block diagonal and each diagonal block $\mathbf{U}^{(j)}$ is an orthogonal matrix. More explicitly,

$$\begin{pmatrix} \mathbf{C}^{(1)} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{C}^{(J)} \end{pmatrix} \begin{pmatrix} \mathbf{U}'^{(1)} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{U}'^{(J)} \end{pmatrix} = \begin{pmatrix} \mathbf{C}^{(1)}\mathbf{U}'^{(J)} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{C}^{(1)}\mathbf{U}'^{(J)} \end{pmatrix}$$

This transformation will also preserve the Frobenius norm of $\mathbf{x}_t$, but the element value is not preserved. To ensure the unique solution, we need to add more constraints on linear dynamics, i.e. $\mathbf{A}$ or/and $\mathbf{Q}$. Simply making $\mathbf{A}$ or $\mathbf{Q}$ be block diagonal or even diagonal with same absolute value (e.g., $\mathbf{I}_{Jp}$) is still not enough. One solution is to force at least one of $\mathbf{A}$ and $\mathbf{Q}$ be diagonal.

Take $\mathbf{A}$ for example. Assume the true $\mathbf{A}$ is diagonal and true $\mathbf{C}^{\{j\}}$ has orthogonal column. As shown previous, the linear transformation $\mathbf{U}$ must be block diagonal and each diagonal block $\mathbf{U}^{(j)}$ is an orthogonal matrix. Further, to preserve the diagonal structure of $\mathbf{A}$ (the elements on the diagonal are generally not the same), we need $\mathbf{U}^{(j)}$ to be diagonal. These two constraints force $\mathbf{U}$ be diagonal with diagonal elements be ± 1. So the solution is unique up to sign. The same rationale for constraint on $\mathbf{Q}$.

In summary, we need add three constraints if do things jointly: (1) translation of $\mathbf{x}_t$, (2) orthogonality of loading and (3) diagonality of linear dynamics. In the following implementation, I tried 3 sets of constraints:

(1) row mean in $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$ is 0, $\mathbf{C}'^{(j)}\mathbf{C}^{(j)} = \mathbf{I}_p$, $\mathbf{A}$ is full while $\mathbf{Q}$ is diagonal

(2) row mean in $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$ is 0, $\mathbf{C}'^{(j)}\mathbf{C}^{(j)} = \mathbf{I}_p$, $\mathbf{Q}$ is full while $\mathbf{A}$ is diagonal

(3) row mean in $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$ is 0, $\mathbf{C}'^{(j)}\mathbf{C}^{(j)} = \mathbf{I}_p$, both $\mathbf{A}$ and $\mathbf{Q}$ are diagonal

Constraint (1) and (2) allows interaction between clusters.

Although, constraint (1) (diagonal $\mathbf{Q}$) is valid in theory, it doesn't work well in practice. Since elements $\mathbf{Q}$ is usually very small (10^{-3} to 10^{-5}). Therefore, I will discard this constraint later.

8.2 sample the orthogonal loading

The constraint on orthogonality is kind of tricky. Here, I just transformed $\mathbf{C}^{(j)}$ be $\mathbf{C}^{(j)} = \mathbf{K}^{(j)} \left(\mathbf{K}^{'(j)} \mathbf{K}^{(j)} \right)^{-1/2}$. Therefore, we can just sample $\mathbf{K}^{(j)}$ without any constrained. Since $\mathbf{K}^{(j)}$ has the same structure as $\mathbf{C}^{(j)}$, we can put some hierarchical prior to model $\mathbf{C}^{(j)}$. Specially, when $\mathbf{K}^{(j)} \sim MN_{n,p}(\mathbf{0}, \mathbf{\Sigma}^{(j)}, \mathbf{I})$ ($MN_{n,p}$ means a matrix normal distribution), $\mathbf{C}^{(j)} \sim MACG(\mathbf{\Sigma}^{(j)})$. $MACG(\mathbf{\Sigma})$ means the matrix angular central Gaussian distribution, with density $f_{\mathbf{X}}(\mathbf{X}) = |\mathbf{\Sigma}\ |^{-p/2} |\mathbf{X}' \Sigma^{-1} \mathbf{X}|^{-n/2}$. When $\mathbf{\Sigma}^{(j)} = \mathbf{I}_n$ or $n = p$, $\mathbf{C}^{(j)}$ is just uniformly distributed on the Stiefel manifold $S(p, n) = \{\mathbf{C} \in \mathbb{R}^{n \times p} | \, \mathbf{C}'\mathbf{C} = \mathbf{I}_p\}$.

Since now there's no closed form for full conditional, I just sample it by no U-turn sampler (NUTS).

8.3 diagonal A and full Q

Denote the diagonal $\mathbf{A}$ as $\mathbf{A} = diag(a_1, \ldots, a_{Jp})$. Then,

$$\mathbf{Y}_{BA} = \begin{pmatrix} \mathbf{x}_2^{'} \\ \mathbf{x}_3^{'} \\ \vdots \\ \mathbf{x}_T^{'} \end{pmatrix} = \begin{pmatrix} 1 & x_{1,1} & \cdots & x_{1,Jp} \\ 1 & x_{2,1} & \cdots & x_{2,Jp} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{T-1,1} & \cdots & x_{T-1,Jp} \end{pmatrix} \begin{pmatrix} b_1 & \cdots & b_{Jp} \\ a_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{Jp} \end{pmatrix} + \mathbf{E} = \mathbf{X}_{BA} \mathbf{F}_{BA} + \mathbf{E}$$

, where $\mathbf{E} = (\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_{T-1})^{'}$ and $\boldsymbol{\epsilon}_t \sim N(\mathbf{0}, \mathbf{Q})$. Notice that $\mathbf{x}_t = (x_{t,1}, \cdots x_{t,Jp})'$.

Equivalently, we can rewrite the model as:

$$\mathbf{Y}_{ba} = vec\,(\mathbf{Y}_{BA}) = \begin{pmatrix} 1 & x_{1,1} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{T-1,1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & x_{1,Jp} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & x_{T-1,Jp} \end{pmatrix} \begin{pmatrix} b_1 \\ a_1 \\ \vdots \\ b_{Jp} \\ a_{Jp} \end{pmatrix} + vec(\mathbf{E}) = \mathbf{X}_{ba} \mathbf{f}_{ba} + vec(\mathbf{E})$$

, where $Cov\,(vec(\mathbf{E})) = \mathbf{Q} \bigotimes \mathbf{I}_{T-1}$.

If we use the prior:

$$\mathbf{Q} \sim W^{-1}(\Psi_{Q_0}, \nu_{Q_0})$$
$$\mathbf{f}_{ba} | \mathbf{Q} \sim N_{2Jp}(\mathbf{1}_{Jp} \bigotimes \mathbf{f}_{ba,0},\ \mathbf{Q} \bigotimes \mathbf{\Lambda}_{ba,0}^{-1})$$

, where $\mathbf{f}_{ba,0} = (0,1)'$ and $\mathbf{\Lambda}_{ba,0} = \mathbf{I}_2$. Then the corresponding prior for $vec(\mathbf{F}_{BA})$ is:

$$vec\left(\mathbf{F}_{BA}\right)|\mathbf{Q} \sim N(vec\left(\mathbf{F}_{BA,0}\right),\ \mathbf{Q} \bigotimes \mathbf{\Lambda}_{BA,0}^{-1})$$

, where $\mathbf{F}_{BA,0} = \begin{pmatrix} \mathbf{0}_{Jp}^{'} \\ \mathbf{I}_{Jp} \end{pmatrix}$ and $\mathbf{\Lambda}_{BA,0} = \mathbf{I}_{Jp+1}$.

Then the posteriors are:

$$\begin{aligned} \mathbf{Q}|\mathbf{Y}, \ldots &\sim W^{-1}(\Psi_Q, \nu_Q) \\ \mathbf{f}_{ba}|\mathbf{Y}, \mathbf{Q}, \ldots &\sim N(\widehat{\mathbf{f}}_{ba},\ (\mathbf{Q} \bigotimes \mathbf{I}_2)\Lambda_{ba}^{-1}) \end{aligned}$$

, with parameters be:

$$\begin{aligned} \mathbf{\Lambda}_{ba} &= \mathbf{X}_{ba}^{'}\mathbf{X}_{ba} + \mathbf{I}_{Jp} \bigotimes \mathbf{\Lambda}_{ba,0} \\ \widehat{\mathbf{f}}_{ba} &= \mathbf{\Lambda}_{ba}^{-1}\left(\mathbf{X}_{ba}^{'}\mathbf{Y}_{ba} + \ \mathbf{1}_{Jp} \bigotimes \mathbf{\Lambda}_{ba,0}\mathbf{f}_{ba,0}\right) \end{aligned}$$

The corresponding estimate for $\mathbf{F}_{BA}$ is denoted as $\widehat{\mathbf{F}}_{BA}$, which is just a transformation of $\widehat{\mathbf{f}}_{ba}$.

$$\Psi_Q = \Psi_{Q0} + \left(\mathbf{Y}_{BA} - \mathbf{X}_{BA}\widehat{\mathbf{F}}_{BA}\right)^{'}\left(\mathbf{Y}_{BA} - \mathbf{X}_{BA}\widehat{\mathbf{F}}_{BA}\right) + \left(\widehat{\mathbf{F}}_{BA} - \mathbf{F}_{BA,0}\right)^{'}\Lambda_{BA,0}^{-1}\left(\widehat{\mathbf{F}}_{BA} - \mathbf{F}_{BA,0}\right)$$
$$\nu_Q = \nu_{Q0} + T - 1$$

9 Simulation on joint constraint

The simulation example is as before, and all the code can be found in this folder.

Since now we constrain the loading has orthogonal column for each cluster, just showing the Frobenius norm of $\mathbf{x}_t$ is not enough to check convergence. Here, I further check the value of $\mathbf{x}_{[T/2]}$.

For $\mathbf{C}^{(j)} = \mathbf{K}^{(j)}\left(\mathbf{K}'^{(j)}\mathbf{K}^{(j)}\right)^{-1/2}$, I just put a simple hierarchical prior on $\mathbf{K}^{(j)}$ such that each element of $\mathbf{K}^{(j)}$ is i.i.d. $N(\mu_k^{(j)}, (\sigma_k^{(j)})^2)$. Well, maybe we can put some more careful priors later to reflect the covariance among neurons.

Basically, I ran both MCMCs for 10,000 iterations. Here, I show 6 sets of plots:

(1) Full trace of 2-norm/ Frobenius norm in $\boldsymbol{d}$, $\boldsymbol{C}$ and $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$

(2) Full trace of 2-norm/ Frobenius norm in $\mathbf{b}$, $\mathbf{A}$ and $\mathbf{Q}$

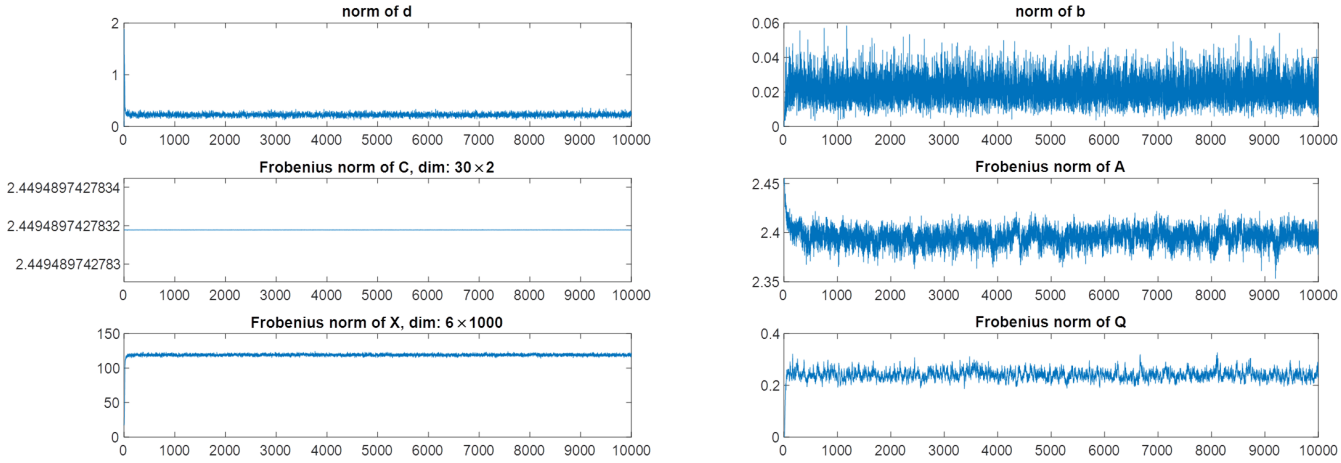
(3) partial trace plot of (1) from iteration 1 to 1000

(4) partial trace plot of (2) from iteration 1 to 1000

(5) average mean firing rate from iteration 5000 to 10,000.

(6) average $\mathbf{X}$ from iteration 5000 to 10,000.

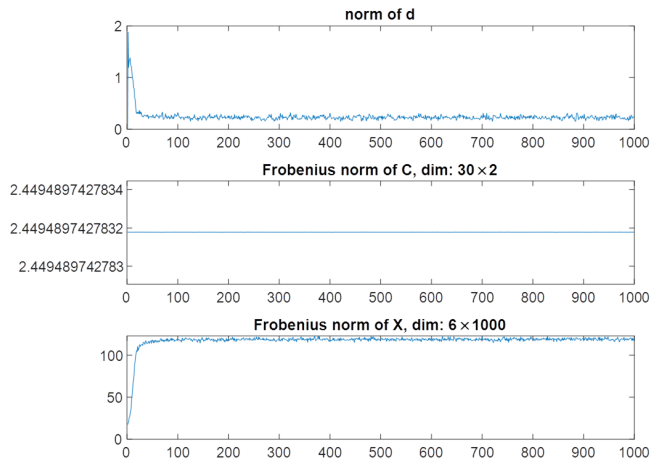
9.0.1 Q is full while A is diagonal



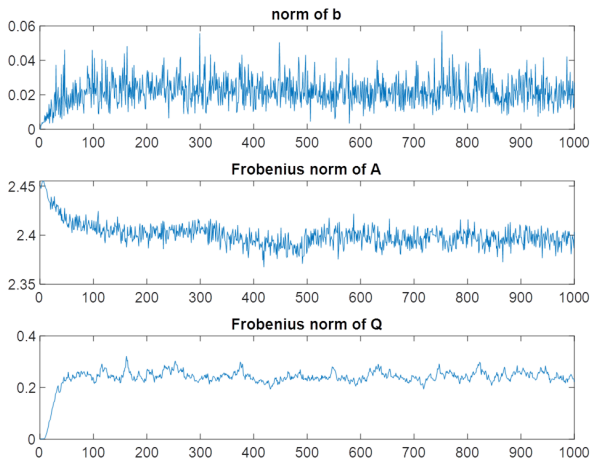
(a) tracd of $\mathbf{X}$

(b) trace of loading and dynamics

Figure 5: $\mathbf{Q}$ is full while $\mathbf{A}$ is diagonal, full trace

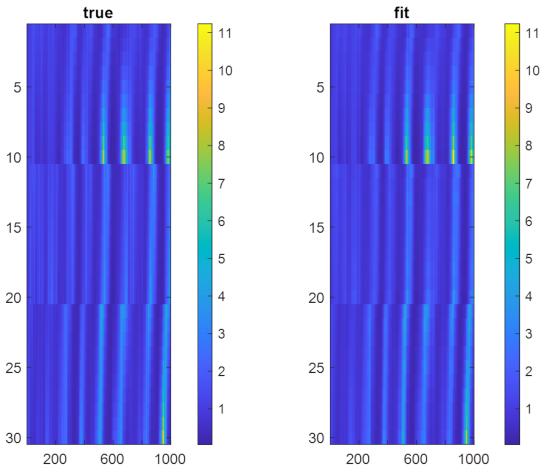


(a) tracd of $\mathbf{X}$

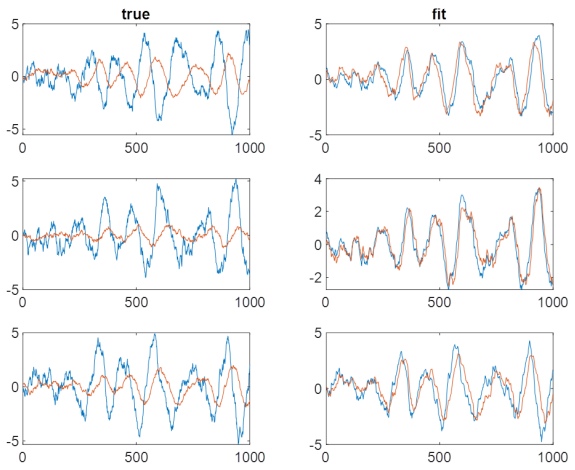


(b) trace of loading and dynamics

Figure 6: $\mathbf{Q}$ is full while $\mathbf{A}$ is diagonal, partial trace



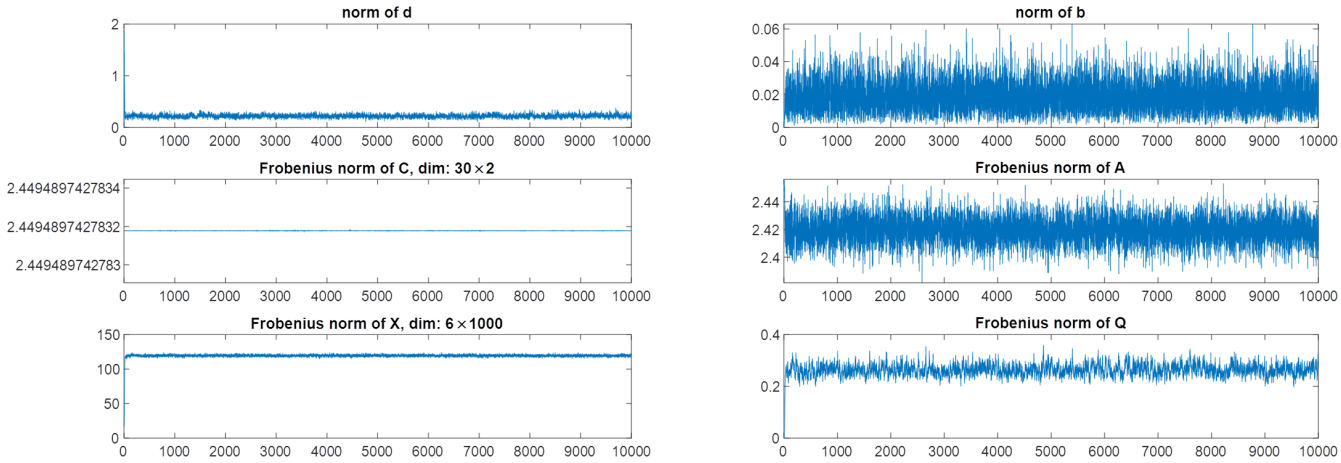
(a) mean firing rate



(b) latent vectors

Figure 7: $\mathbf{Q}$ is full while $\mathbf{A}$ is diagonal, mean FR and $\mathbf{x}_t$

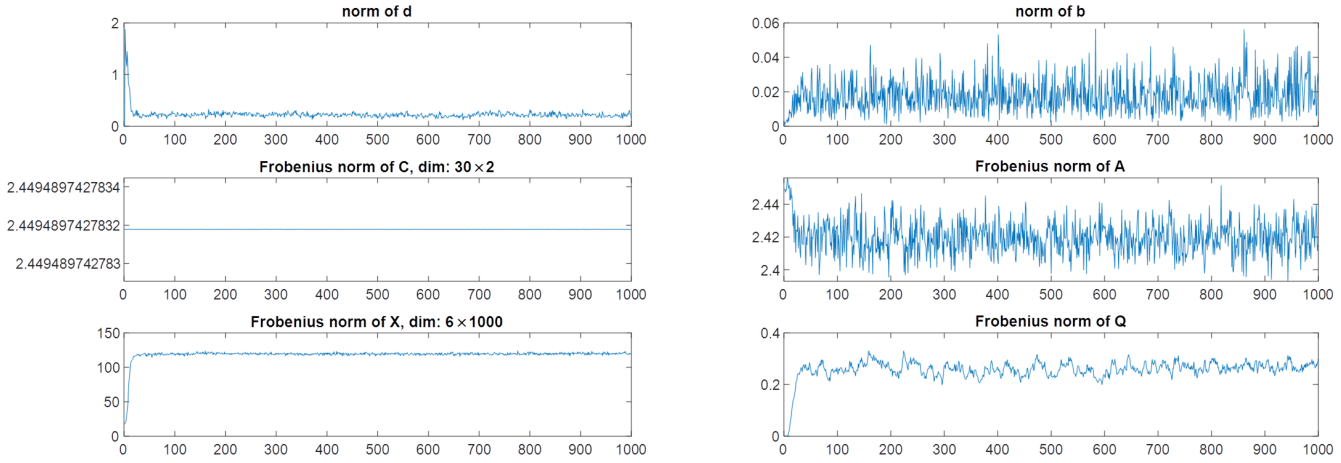
9.0.2 both A and Q are diagonal



(a) tracd of $\mathbf{X}$

(b) trace of loading and dynamics

Figure 8: both $\mathbf{A}$ and $\mathbf{Q}$ are diagonal, full trace

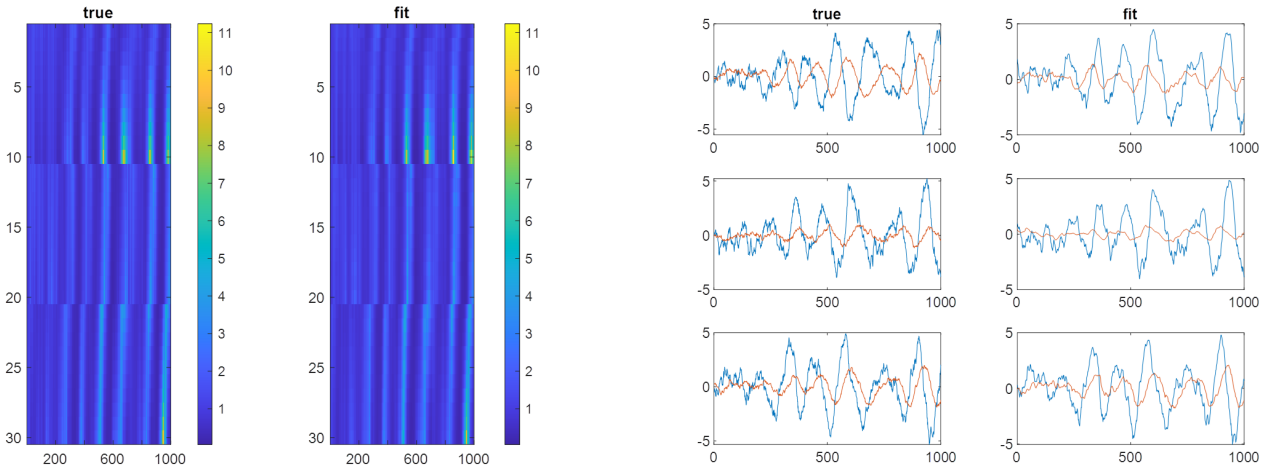


(a) tracd of $\mathbf{X}$

(b) trace of loading and dynamics

Figure 9: both $\mathbf{A}$ and $\mathbf{Q}$ are diagonal, partial trace

We can see that either is fine, and the convergence is achieved at nearly 20 to 30 iterations.



(a) mean firing rate

(b) latent vectors

Figure 10: both $\mathbf{A}$ and $\mathbf{Q}$ are diagonal, mean FR and $\mathbf{x}_t$

In summary, to ensure convergence, we can put the following two constraints:

(1) row mean in $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$ is 0, $\mathbf{C}'^{(j)}\mathbf{C}^{(j)} = \mathbf{I}_p$, $\mathbf{Q}$ is full while $\mathbf{A}$ is diagonal

(2) row mean in $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$ is 0, $\mathbf{C}'^{(j)}\mathbf{C}^{(j)} = \mathbf{I}_p$, both $\mathbf{A}$ and $\mathbf{Q}$ are diagonal

If we really care about the interactions between clusters, just use constraint (1). Otherwise, either is fine.