The idea is pretty straightforward: use the state vectors to do model-based clustering.

Here, I just show the simplest (and most commonly used) state-space for count data, the Poisson adaptive filter. This just use the information of mean, but we can further extend it to include information about variance & interactions, or make state space "sparse" etc.

In the 1st section, I do clustering by mixture model. Then I extend it and use DP for clustering.

Mixture Model (MM)

Model Details

Assume there are N neurons, and the recording length is T. The observation matrix is $\boldsymbol{Y} \in N^{N\times T}$, with each element to be a non-negative integer. Each row is the recording for neuron j, $\boldsymbol{y}_j = (y_{j1}, \dots, y_{jT})'$. Denote the cluster index for neuron j as z_j. In MM $z_j \in \{1, \dots, K\}$, but will include infinite element in DP. The proportion/ probability in cluster z_j is ρ_{z_j}.

For simplicity (in writing), assume each cluster corresponds to only 1 latent state vectors. Theses latent states are denoted as $\boldsymbol{\theta}_i \in R^T$, for $i = 1, \dots, K$. Therefore, the full likelihood for these N neurons is

$$L = \prod_{j=1}^{N} \rho_{z_j} f\left(\boldsymbol{y}_j | \boldsymbol{\theta}_{z_j}\right) = \prod_{i=1}^{K} \rho_i^{n_i} \left[\prod_{j:z_j=i} f\left(\boldsymbol{y}_j | \boldsymbol{\theta}_i\right)\right]$$

Where n_i is the number of neurons in cluster i, i.e. $n_i = \sum_{j=1}^{N} I(z_j = i)$. Obviously, $\sum_{i=1}^{K} n_i = N$.

In each neuron, assume the observation is generated by a state space model (which will give the prior). Specifically, when neuron j is in cluster i, the observation at t is generated as follows:

$$\theta_{it} \sim N(F_{it}\theta_{i,t-1}, Q_{it})$$

$$\log \lambda_{it} = x_{it}\theta_{it}$$

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

Since the progress noise is independent in state space model, $f\left(\boldsymbol{y}_j | \boldsymbol{\theta}_i\right) = \prod_{t=1}^{T} P(y_{it}|\theta_{it})$, where $P(\cdot)$ is the Poisson density. As usual Bayesian MM model, what I need to update/ sample by Gibbs sampler are: (1) cluster indicator $\{z_j\}_{j=1}^{N}$, (2) cluster proportion $\boldsymbol{\rho} = (\rho_1, \dots \rho_K)'$ and (3) state vectors $\{\boldsymbol{\theta}_i\}_{i=1}^{K}$. The prior for $\boldsymbol{\rho}$ is $Dir(\delta_1, \dots \delta_K)$. The full conditional distribution for all these 3 are as follows:

(1) cluster indicator $\{z_j\}_{j=1}^{N}$

$$P\left(z_j = i \middle| \boldsymbol{y}_j, \{\boldsymbol{\theta}_i\}_{i=1}^{K}\right) \propto \rho_i f\left(\boldsymbol{y}_j | \boldsymbol{\theta}_i\right)$$

(2) cluster proportion $\boldsymbol{\rho} = (\rho_1, \dots \rho_K)'$

$$\boldsymbol{\rho} | \boldsymbol{Y}, \{z_j\}_{j=1}^{N}, \{\boldsymbol{\theta}_i\}_{i=1}^{K} \sim Dir(\delta_1 + n_1, \dots \delta_K + n_K)$$

(3) state vectors $\{\boldsymbol{\theta}_i\}_{i=1}^{K}$

$$P\left(\boldsymbol{\theta}_i \middle| \boldsymbol{Y}, \{z_j\}_{j=1}^{N}, \boldsymbol{\rho}\right) \propto \prod_{j:z_j=i} \prod_{t=1}^{T} P(y_{it}|\theta_{it}, \boldsymbol{y}_i) P(\theta_{it}|y_{i,1:t-1})$$

Since the likelihood is Poisson but the prior is Gaussian, there's no closed form for the posterior. Instead of sampling from posterior directly, I still use Laplace approximation (approximate Posterior for filtering by Gaussian), and then sample posterior from Gaussian distribution (parameters are estimated by RTS smoothing). So we can actually call it "Laplace-Gibbs-Sampler".

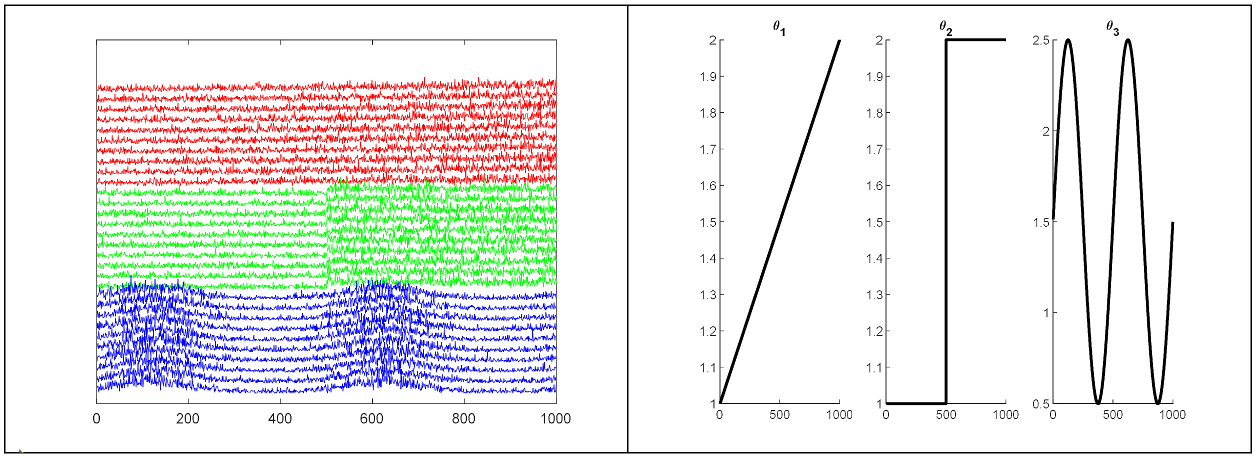
The detailed derivation for Poisson adaptive filter can be found in http://www.stat.columbia.edu/~liam/teaching/neurostat-spr11/papers/brown-et-al/eden2004.pdf.

When there's no $z_j = i$, just use the prior defined by $\theta_{it} \sim N(F_{it}\theta_{i,t-1}, Q_{it})$.

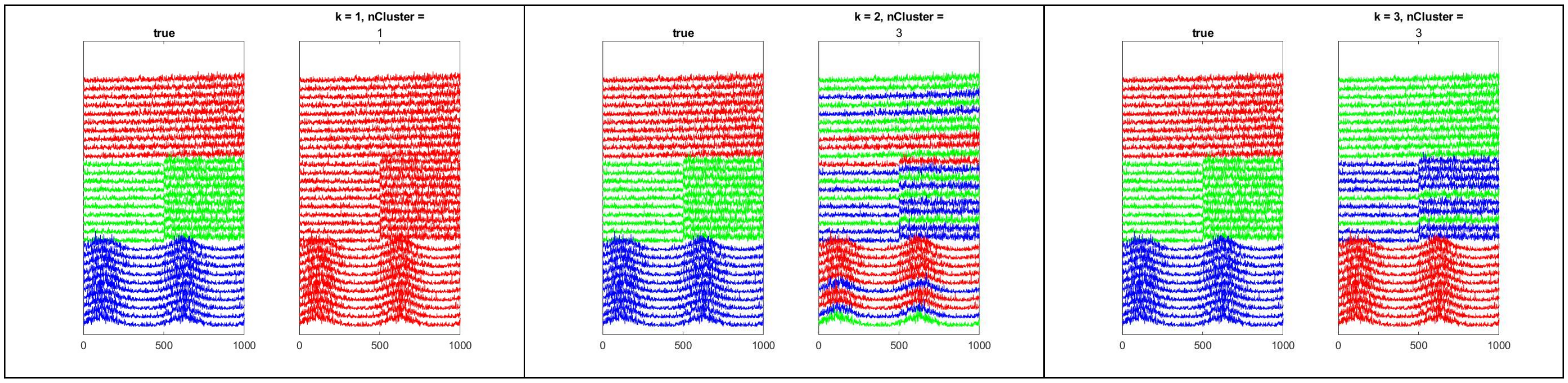
Results

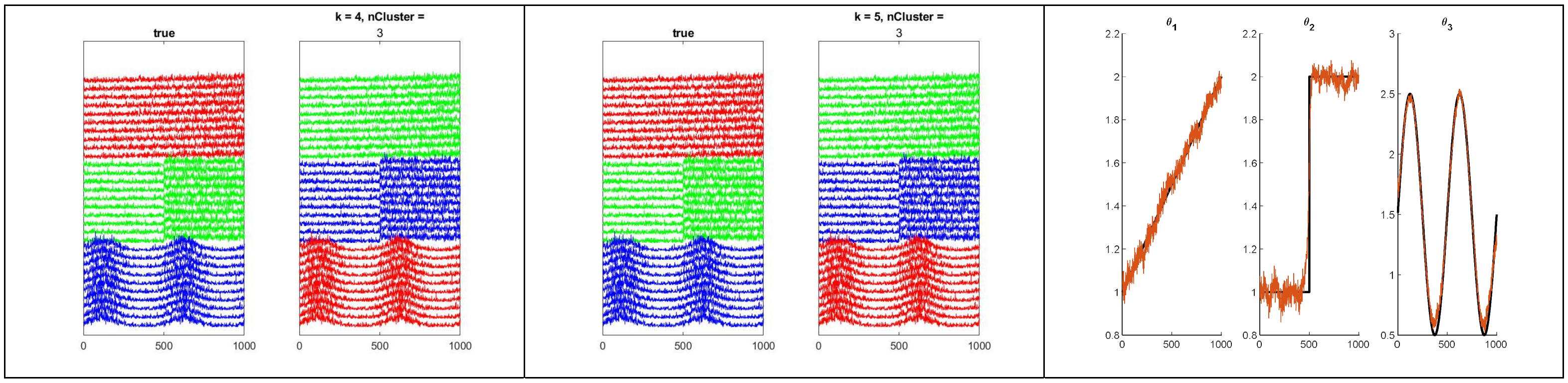
Simulation 1

Here I simulate 3 clusters, each cluster has 10 neurons. The recording length is T = 1000. The simulated data and corresponding state vectors for each cluster is:



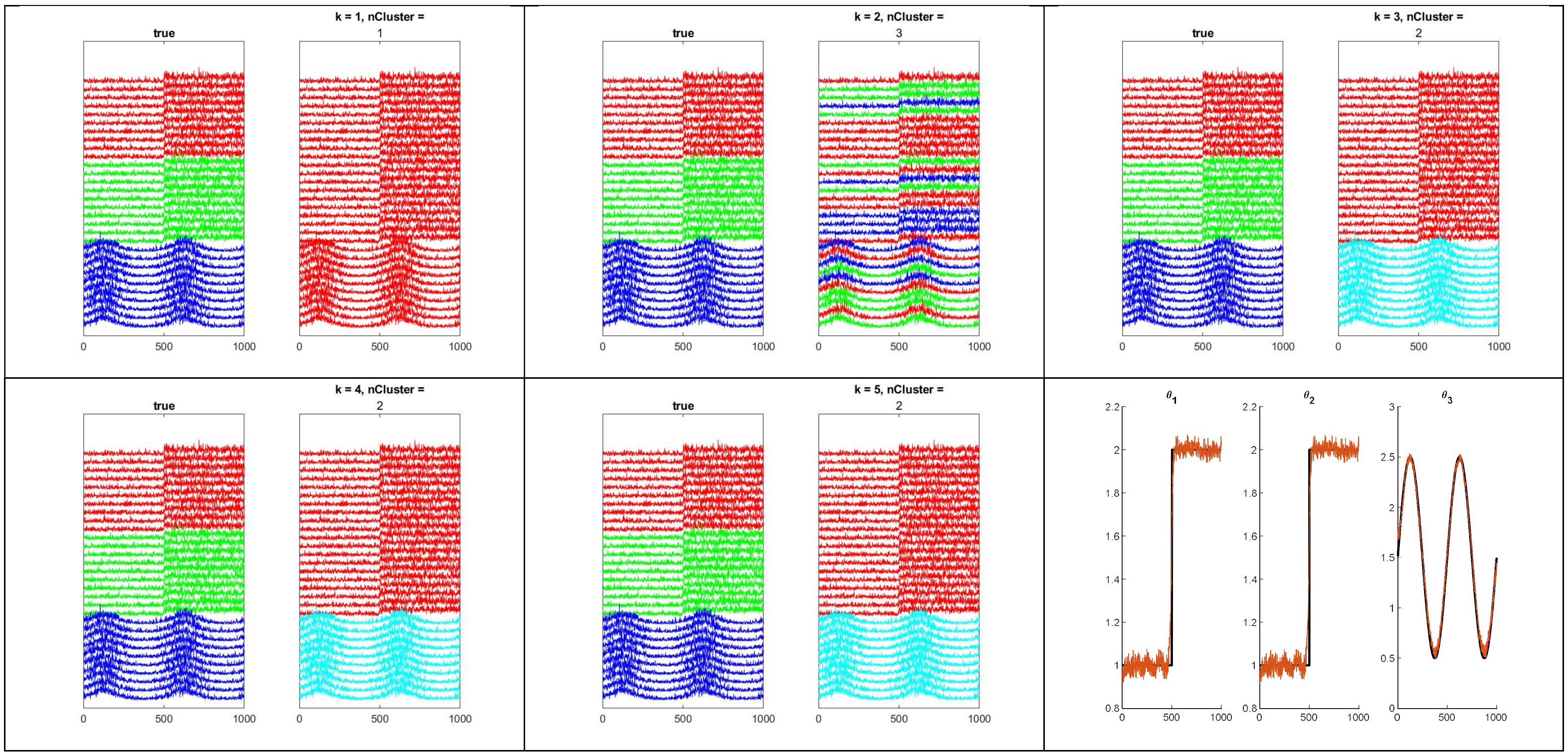
The prior for $\boldsymbol{\rho}$ is $Dir(1, \ldots, 1)$. The iteration runs from below (single cluster), and the clustering results for each iteration is shown here. The last panel just show fitting for state vectors at the last iteration.





Simulation 2

Now, I just make 1st and 2nd cluster have the same state vectors. Therefore, the true cluster number should be K = 2. Again, the clustering results:



Dirichlet Process (DP)

Model Details

The natural extension of MM is to use DP for clustering. However, since the posterior is not conjugate/ has no closed form, when implementing the popular CRP version of DP (Neal 2000), calculate the posterior predictive distribution would be tricky. Here I choose to do DMM by slices (Walker 2007, https://www.tandfonline.com/doi/full/10.1080/03610910601096262).

The proportion of cluster is constructed by “stick-breaking”, i.e.

$$\rho_1 = \eta_1$$

$$\rho_i = (1-\eta_1)\cdot \ldots \cdot (1-\eta_{i-1})\eta_i$$

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

Followed by the algorithm in the paper, the Gibbs sampler is as follows (this algorithm further introduce another latent variables u_j), with some initial allocation $\{z_j\}_{j=1}^N$

(1) update η_i, for $i = 1, \ldots, \mathrm{z}^* = \max\{z_j\}_{j=1}^N$ as

$$\eta_i|\{z_j\}_{j=1}^N, \ldots \sim Beta(n_i+1, N-\sum_{l=1}^{i} n_l + \alpha)$$

(2) update “helper” latent variable $\{u_j\}_{j=1}^N$

$$u_j|\boldsymbol{\rho}, \ldots \sim U(0, \rho_{z_j})$$

(3) update η_i, for $i = \mathrm{z}^*+1, \ldots, s^*$. s^* is the smallest value, s.t. $\sum_{i=1}^{s^*}\rho_i > 1-\min\{u_1, \ldots, u_N\}$

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

(4) update state vectors $\{\boldsymbol{\theta}_i\}_{i=1}^K$ as shown in MM. Again, when there’s no $z_j = i$, just use the prior defined by $\theta_{it} \sim N(F_{it}\theta_{i,t-1}, Q_{it})$.

(5) Update $\{z_j\}_{j=1}^N$

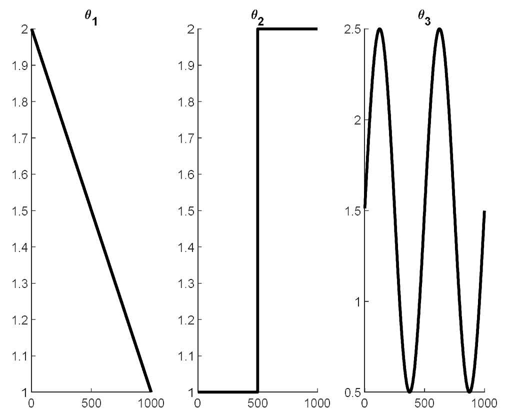
$$P\left(z_j = i \middle| \boldsymbol{y}_j, \{\boldsymbol{\theta}_i\}_{i=1}^K, \boldsymbol{\rho}, \{u_j\}_{j=1}^N\right) = \frac{f(\boldsymbol{y}_j|\boldsymbol{\theta}_i)}{\sum_{i:\rho_i>u_j} f(\boldsymbol{y}_j|\boldsymbol{\theta}_i)}$$

Results

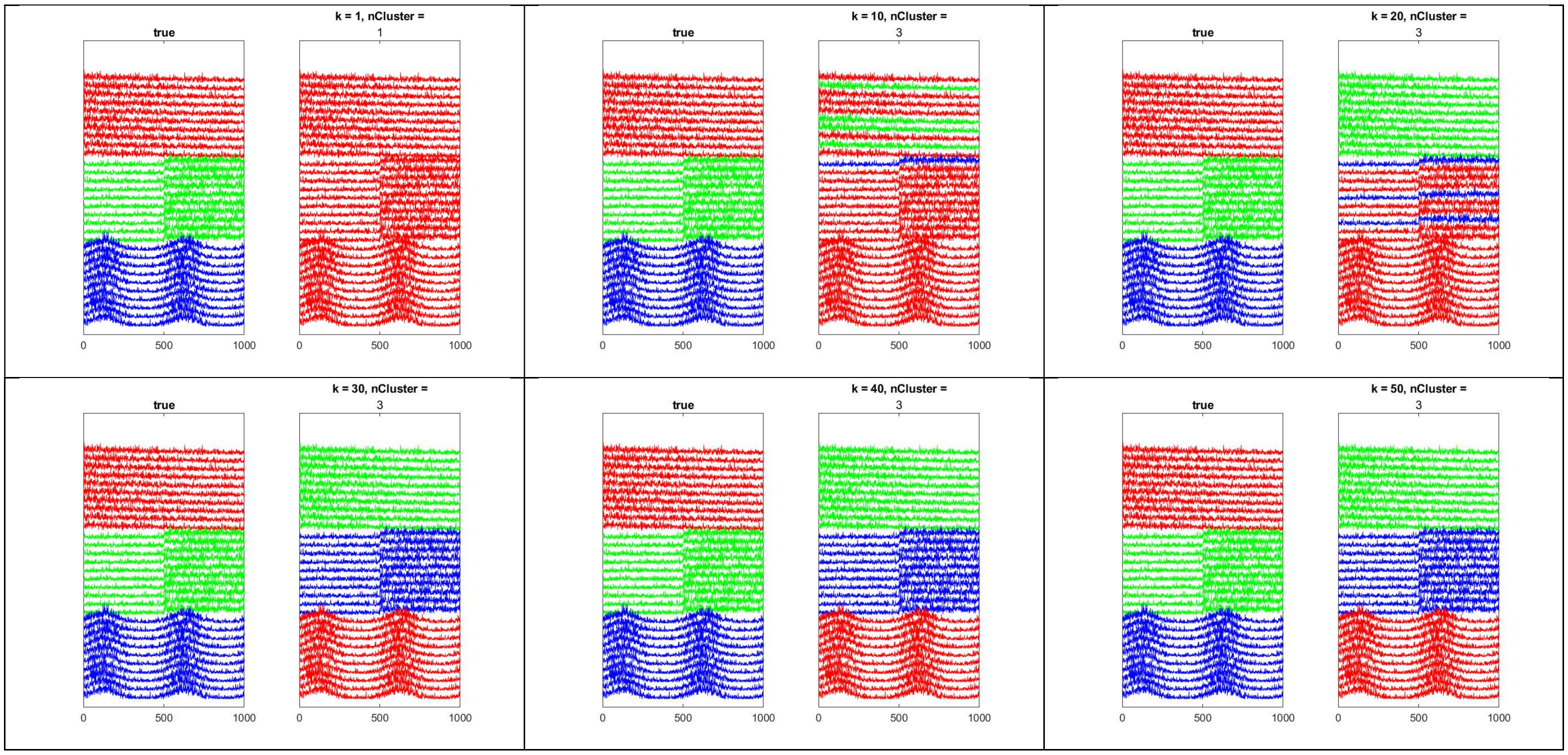
I won’t show results for state vector fitting here. Since when the clustering is correct, the fitted state vector should be OK.

Results 1

Similar setting as in simulation 1 in MM. The state vectors for 3 clusters are:

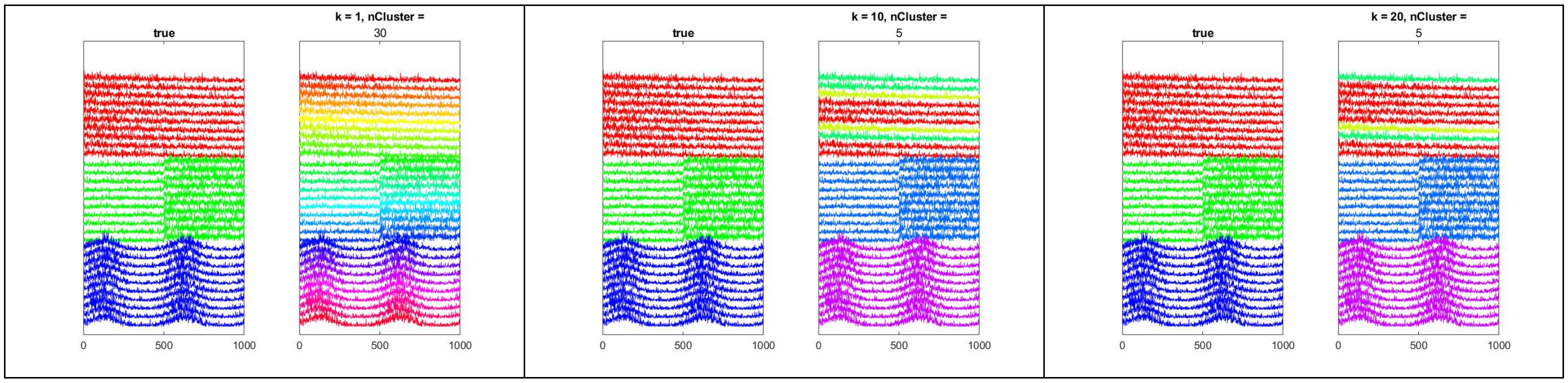


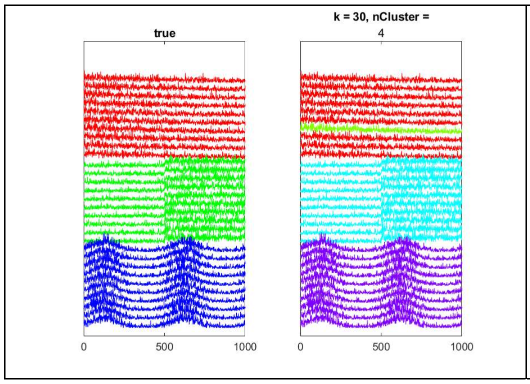
The $\alpha = 1$. I first run from below (single cluster). Show results for iteration 10, 20,…,50:

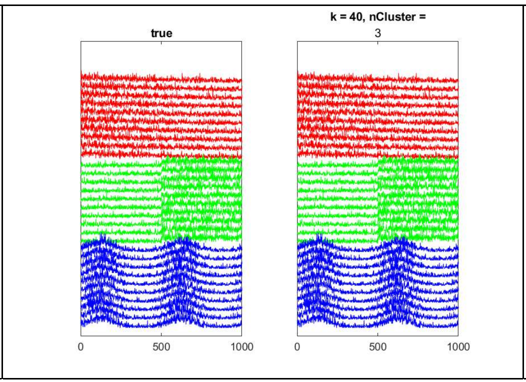


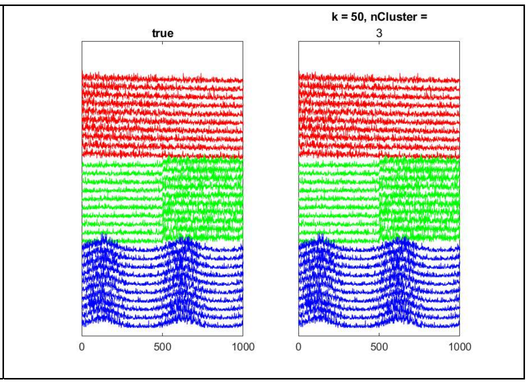
Results 2

What about run from above (each neuron form its own cluster)? The results:



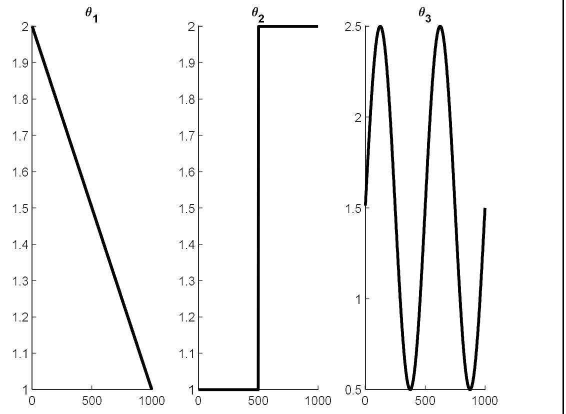
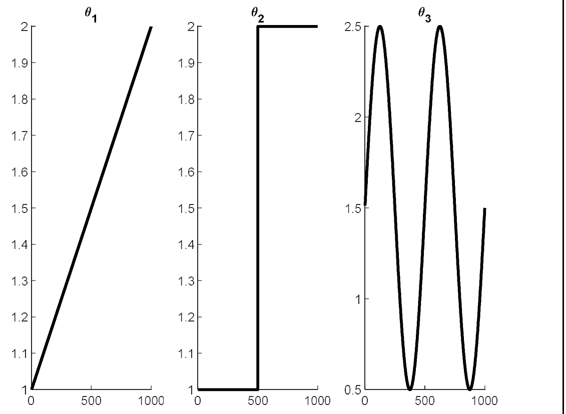






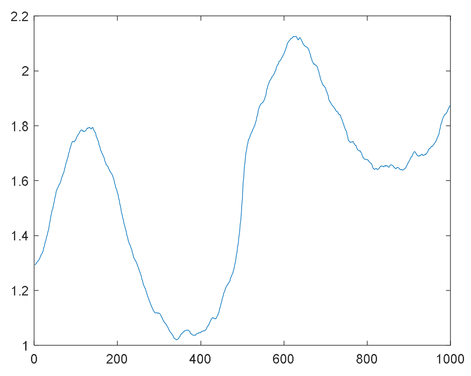
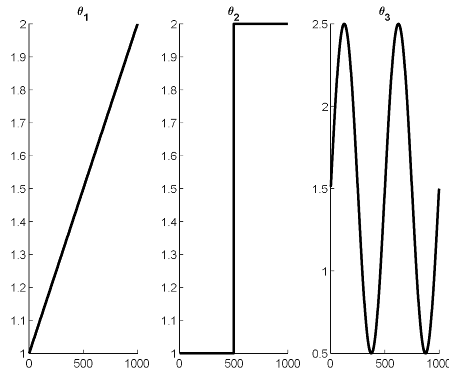
Results 3

I further tried simulation as in MM, which is harder to distinguish than simulation in result 1 & 2.

Result 1 & 2	Result 3
	

Now searching from above is still fine, but searching from below just stay in 1 cluster (iteration = 50).

This is because these 3 state vectors are very similar, fitting state vectors by using all data is always better than the random prior (defined by state equation). So the convergence will be super slow for from-below search for a small α.

Posterior mean by using all dataset	True state vectors
	

Personally, I **will always search from above (begin with single neuron in each cluster)** with small (at least not large) α to achieve a faster convergence.