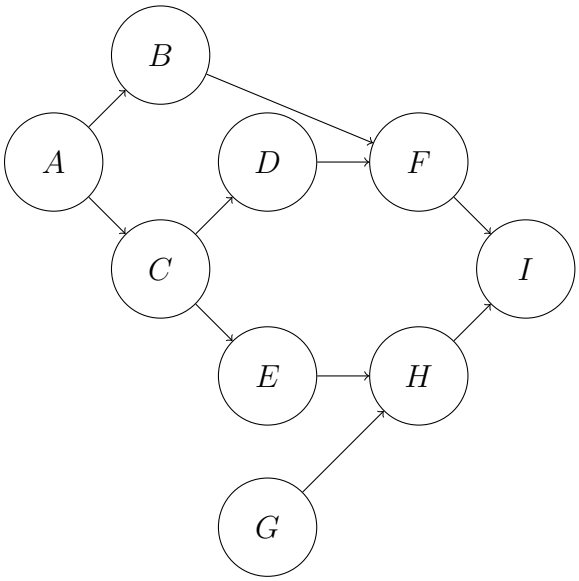
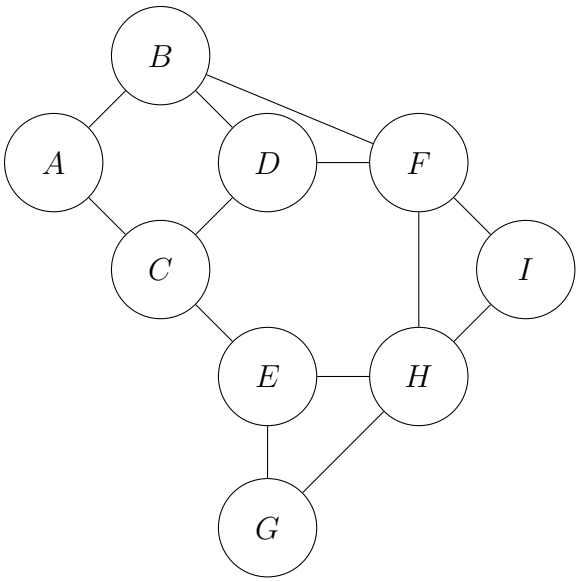
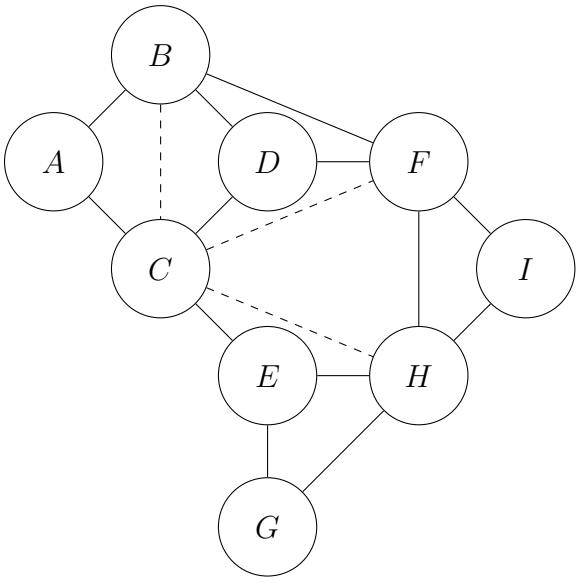
1. (a) The DAG is displayed as follow:



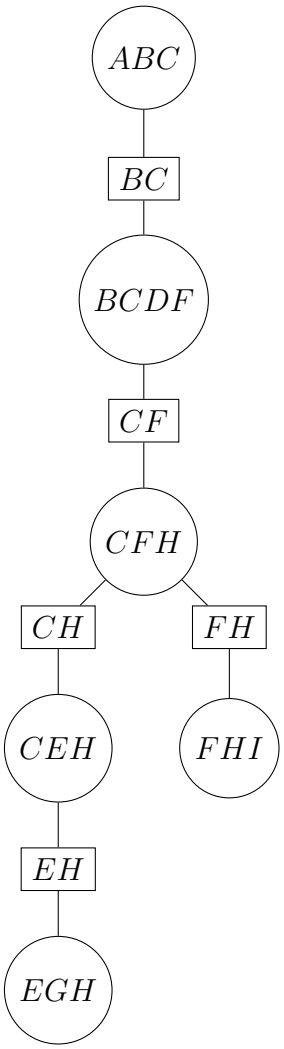
(b) The moralised graph:



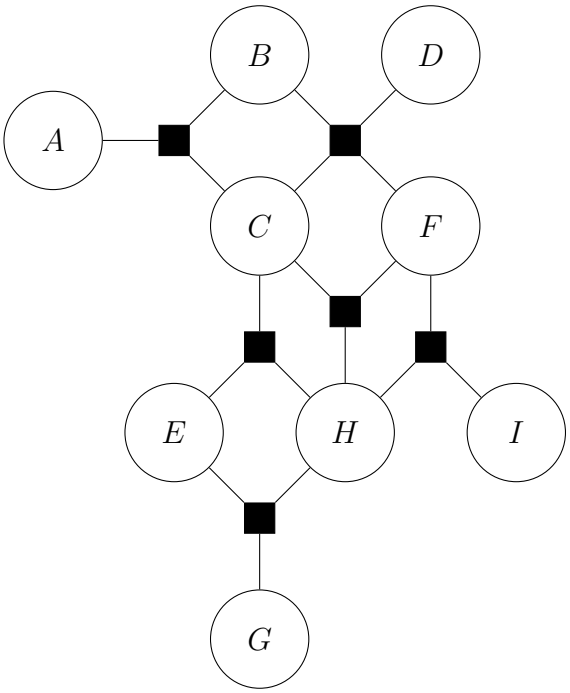
An efficient triangulation:



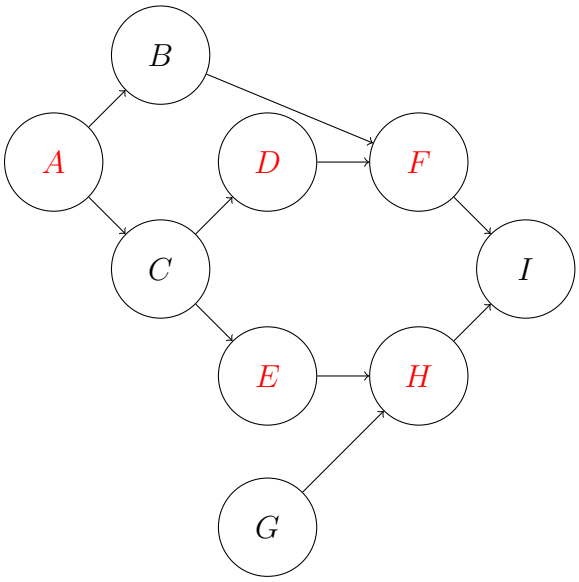
the resulting junction tree:



the junction tree redrawn as a factor graph:



(c) The below graph is the same as the one in (a):



The highlighted node is the set we are looking for, i.e. $\{A, D, E, F, H\}$.

(d) The perturbation model can be expressed as:

$$\delta(\mathbf{x}) = \mathbf{\Lambda}\mathbf{z} + \epsilon\,, \tag{1}$$

where $\epsilon \sim \mathcal{N}(0, \Psi)$ is the Gaussian noise, $\mathbf{z} \sim \mathcal{N}(0, \mathbb{I})$ is the latent factor, $\mathbf{\Lambda}$ is the factor loading matrix.

Given the DAG, we know:

$$\mathbf{\Lambda} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \Lambda_{BA} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \Lambda_{CA} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \Lambda_{DC} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \Lambda_{EC} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \Lambda_{FB} & 0 & \Lambda_{FD} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \Lambda_{HE} & 0 & \Lambda_{HG} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \Lambda_{IF} & 0 & \Lambda_{IH} & 0 \end{bmatrix}, \mathbf{z} = \begin{bmatrix} z_A \\ z_B \\ z_C \\ z_D \\ z_E \\ z_F \\ z_G \\ z_H \\ z_I \end{bmatrix} \tag{2}$$

Noted that Λ_{BA} indicates the factor between child B and parent A. Given that $\delta(B), \delta(D), \delta(E), \delta(G)$ are known, we plug in the perturbation model and simplify:

$$\begin{bmatrix} \delta(B) \\ \delta(D) \\ \delta(E) \\ \delta(G) \end{bmatrix} = \begin{bmatrix} \Lambda_{BA} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \Lambda_{DC} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \Lambda_{EC} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_A \\ z_B \\ z_C \\ z_D \\ z_E \\ z_F \\ z_G \\ z_H \\ z_I \end{bmatrix} + \begin{bmatrix} \epsilon_B \\ \epsilon_D \\ \epsilon_E \\ \epsilon_G \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} \delta(B) \\ \delta(D) \\ \delta(E) \end{bmatrix} = \begin{bmatrix} \Lambda_{BA} & 0 \\ 0 & \Lambda_{DC} \\ 0 & \Lambda_{EC} \end{bmatrix} \begin{bmatrix} z_A \\ z_C \end{bmatrix} + \begin{bmatrix} \epsilon_B \\ \epsilon_D \\ \epsilon_E \end{bmatrix}$$

Knowing that the model reveals the parent nodes' perturbation, thus from the above result, the perturbation of node A and node C are to be recovered.

(e) The results of the factor analysis cannot be used to recover the concentration perturbations of any other species in the cascade, since our model can only reveal the perturbation of parent nodes (upstream), but our DAG is going downstream instead of going upstream. Specifically, given $\delta(A)$ and $\delta(C)$, we can only recover the perturbation of parent nodes of A and C (if any), instead of the child nodes.

Thus for node identifiability, only A and C are identifiable up to an unknown scale factor, given $\delta(B)$, $\delta(D)$, $\delta(E)$ and $\delta(G)$.

For linear weight identifiability, similarly, only Λ_{BA}, Λ_{CA}, Λ_{DC} and Λ_{EC} are identifiable up to an unknown scale factor.

2. (a) We denote the covariance as $\sum = \begin{bmatrix} 100 & 0 \\ 0 & 10000 \end{bmatrix}$.

By lecture slides, the posterior mean and covariance are:

$$\sum_{a,b} = \left(\frac{\mathbf{X}\mathbf{X}^\top}{\sigma^2} + (\sum)^{-1} \right)^{-1}, \ \boldsymbol{\mu_{a,b}} = (\sum_{a,b}) \cdot \frac{\mathbf{X}\mathbf{Y}^\top}{\sigma^2} \tag{4}$$

Thus:

The posterior mean: $\boldsymbol{\mu}_{a,b} = \begin{bmatrix} 1.8194 \\ -3.6281 \times 10^3 \end{bmatrix}$.

The posterior covariance: $\sum_{a,b} = \begin{bmatrix} 1.3748 \times 10^{-5} & -0.0275 \\ -0.0275 & 55.0397 \end{bmatrix}$.

(b) The plot of residual $g_{obs}(t)$ is displayed as follow:

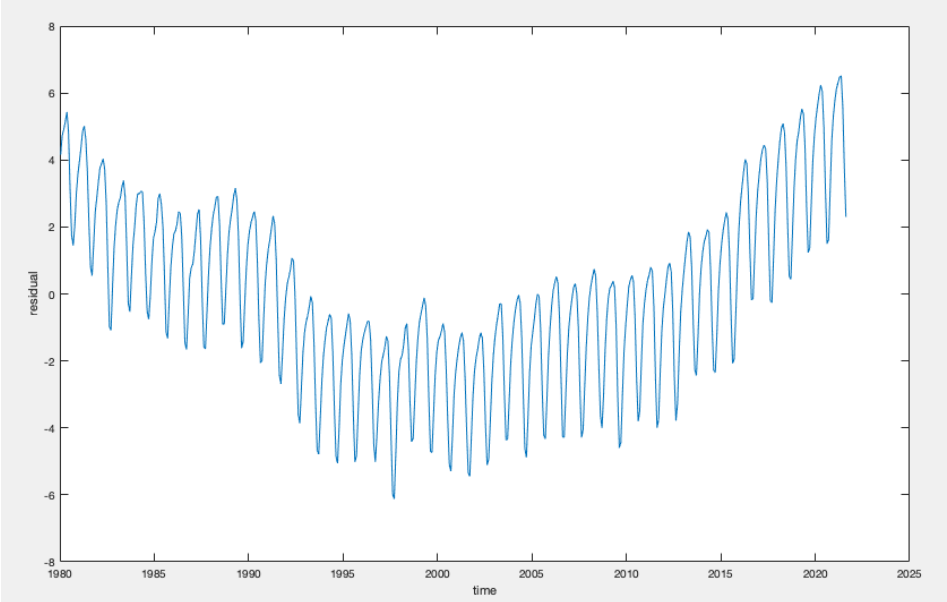


Figure 1: residual vs time

The residuals don't conform to our prior. The residuals reveal time-dependence (or say periodicity), which indicates that our independence assumption about $\epsilon(t)$ is wrong.

(c) The function that generates samples drawn from a GP:

```
function [y,cov] = GP(x,k)
    cov = covkernel(x,k);
    l = size(x,2);
    y = mvnrnd(zeros(1,l),cov);
end
```

```
function [y,cov] = GP(x,k)
    cov = covkernel(x,x,k);
    l = size(x,2);
    y = mvnrnd(zeros(1,l),cov);
end
```

Note that "k" is the kernel function with s and t as the input, i.e. k(s,t).

(d) By setting 7 different groups of hyperparameters, following figures are generated:

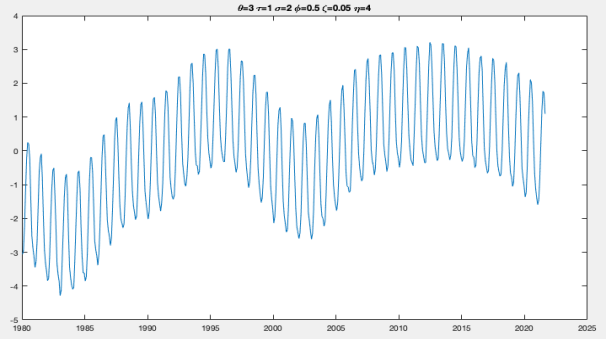


Figure 2: $\theta = 3$, $\tau = 1$, $\sigma = 2$, $\phi = 0.5$, $\zeta = 0.05$, $\eta = 4$

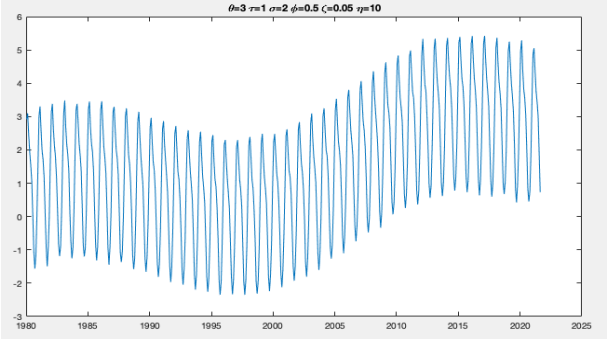


Figure 3: $\theta = 3$, $\tau = 1$, $\sigma = 2$, $\phi = 0.5$, $\zeta = 0.05$, $\eta = 10$

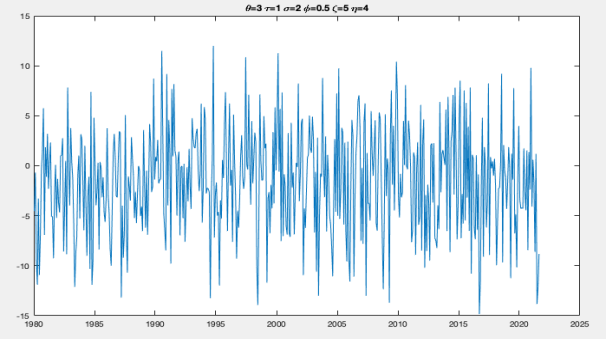


Figure 4: $\theta = 3$, $\tau = 1$, $\sigma = 2$, $\phi = 0.5$, $\zeta = 5$, $\eta = 4$

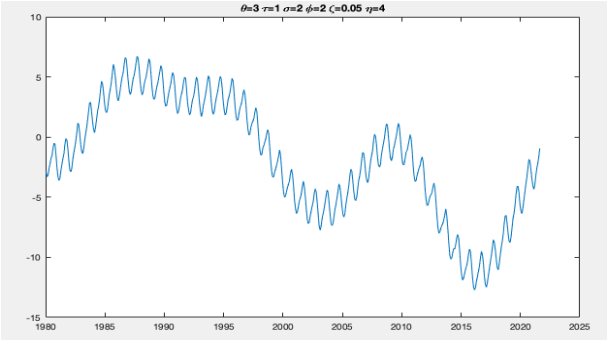


Figure 5: $\theta = 3$, $\tau = 1$, $\sigma = 2$, $\phi = 2$, $\zeta = 0.05$, $\eta = 4$

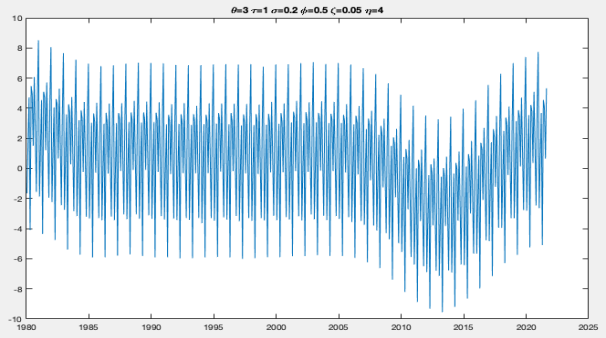


Figure 6: $\theta = 3$, $\tau = 1$, $\sigma = 0.2$, $\phi = 0.5$, $\zeta = 0.05$, $\eta = 4$

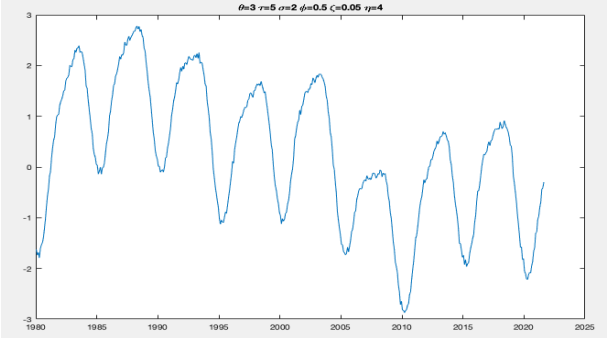


Figure 7: $\theta = 3$, $\tau = 5$, $\sigma = 2$, $\phi = 0.5$, $\zeta = 0.05$, $\eta = 4$

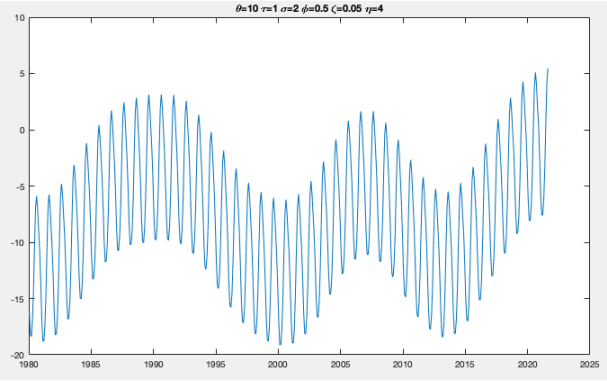


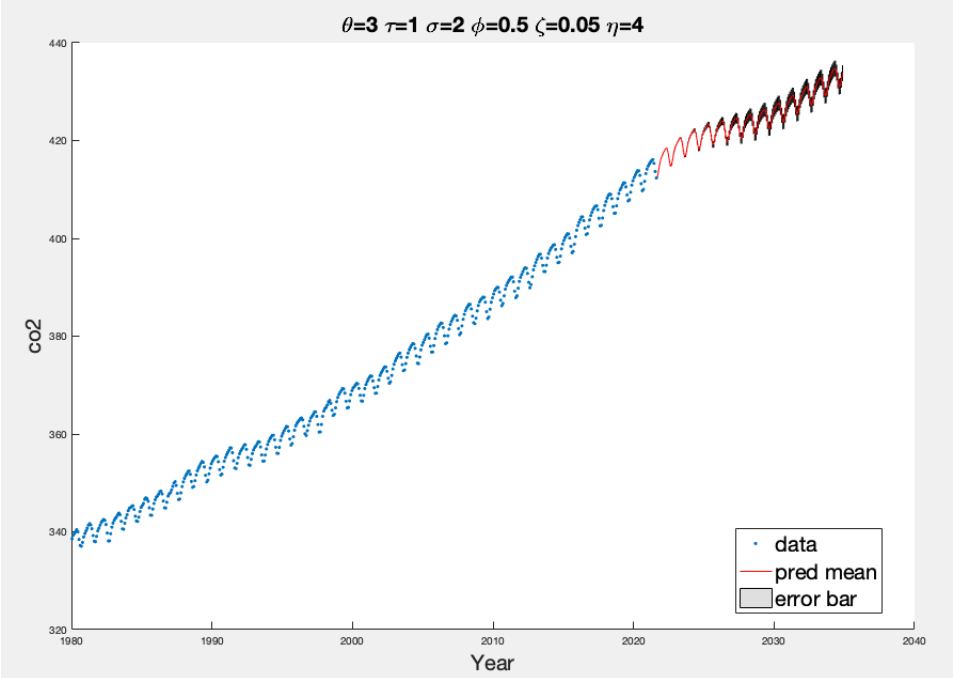
Figure 8: $\theta = 10$, $\tau = 1$, $\sigma = 2$, $\phi = 0.5$, $\zeta = 0.05$, $\eta = 4$

- η controls the general trend of the function: the larger the value of η, the more steady the function tends to be. (compare figure 3 and figure 2)

- ζ changes the noise variance at each single time point: the larger the value of ζ is, the larger noises will be detected. (compare figure 4 and figure 2)
- ϕ can influence both the general trend and the swing of the function. When we have larger ϕ value, we tend to have more fluctuating trend and smaller swing. (compare figure 5 and figure 2)
- σ affects how smooth the sinusoidal fluctuation is: as the value of σ decreases, the function gets more spikes. (compare figure 6 and figure 2)
- τ determines the periodicity of the sinusoidal fluctuation: the larger the value of τ, the number of the cycle is less. (compare figure 7 and figure 2)
- θ determines the extent of correlation for different data pairs, while also affects variance: the larger the θ value, more variance is detected and the figures fluctuates more significantly. (compare figure 8 and figure 2)

(e)
- Set $\eta = 4$, since generally the function has the global trend/fluctuation in 4-5 years.
- Set $\zeta = 0.05$, since it controls the variance of the random noise and therefore needs to be low.
- Set $\phi = 0.5$, since the swing of the function is more suitable.
- Set $\sigma = 2$, as the number may generate more steady function.
- The period length is 1 year, so we set $\tau = 1$.
- The extent of the correlation needs to dominate, so we may set $\theta = 3$.

(f) The means and one standard deviation error bars of the predicted CO_2 level is shown below:



Generally, the behaviour of the extrapolation conform to the expectation, as the prediction seems quite continuous, while the predicted periodic behaviours are also similar to the data given. Noticeably, the standard deviation (or say the error bar) increases when the prediction gets longer. This is also within the scope of our expectation, as the uncertainly will naturally increase when the time gap between the predicted value and the last observed data becomes larger.

In terms of the parameter sensitivity:

- η and σ have little effect on the prediction result. For η, it has limited effect on the sinusoidal function. Noticeably, when η gets smaller, the prediction continuity may raise concerns, since it generally creates a down shift compared with the original data. For σ, when it takes

small values, although there are fluctuations (in high frequency) for standard deviation, the prediction results still perform well.

- As ζ, ϕ increases, the standard deviation becomes larger. In terms of ϕ, the increasing effect is larger as the extrapolation gets longer.
- When τ and θ takes larger values, the amplitude of the predicted fluctuation will decrease, and vice versa. However, besides the amplitude, large values of θ also increases the standard deviation significantly.

(g) Why half Bayesian?

- When choosing the hyperparameters, there is no previous study or information included. Specifically, the parameters are not learned by having a prior and MAP estimate given the data.
- We only used the mean value in the prediction, and neglect any uncertainty between the original data and new data. In a full Bayesian framework, we need to take the original data and the new data as random variables that are generated from certain distributions, and integrate our prediction over all parameters.

3. (a) When fully factored:

$$q(s) = \prod_n q_n(s^{(n)}) = \prod_{n,i} q_i^n(s_i^{(n)}) \tag{5}$$

Thus the VE-step:

$$\begin{aligned}
q_j^m(s_j^{(m)}) &\propto \exp \left\langle \log P(\mathbf{x}, \mathbf{s}|\boldsymbol{\theta}) \right\rangle_{\prod_{n,i\neq m,j} q_i^n(s_i^{(n)})} \\
&= \exp \left\langle \sum_n \log P(\mathbf{x}^{(n)}, \mathbf{s}^{(n)}|\boldsymbol{\theta}) \right\rangle_{\prod_{n,i\neq m,j} q_i^n(s_i^{(n)})} \\
&\propto \exp \left\langle \log P(\mathbf{x}^{(m)}, \mathbf{s}^{(m)}|\boldsymbol{\theta}) \right\rangle_{\prod_{i\neq j} q_i^m(s_i^{(m)})} \\
&= \exp \left[\left\langle \log P(\mathbf{x}^{(m)}|\mathbf{s}^{(m)}, \boldsymbol{\theta}) \right\rangle_{\prod_{i\neq j} q_i^m(s_i^{(m)})} + \left\langle \log P(\mathbf{s}^{(m)}|\boldsymbol{\theta}) \right\rangle_{\prod_{i\neq j} q_i^m(s_i^{(m)})} \right] \\
&\propto \exp \Bigg[s_j^{(m)} \log \pi_j + (1 - s_j^{(m)}) \log(1 - \pi_j) - \\
&\frac{1}{2\sigma^2} \left\langle (\mathbf{x}^{(m)} - \boldsymbol{\mu}\mathbf{s}^{(m)})^\top (\mathbf{x}^{(m)} - \boldsymbol{\mu}\mathbf{s}^{(m)}) \right\rangle_{\prod_{i\neq j} q_i^m(s_i^{(m)})} \Bigg] \\
&\propto \exp \Bigg[s_j^{(m)} \log \frac{\pi_j}{1 - \pi_j} - \\
&\frac{1}{2\sigma^2} \left(s_j^{(m)} \mu_j^\top \mu_j s_j^{(m)} - 2 s_j^{(m)} \mu_j^\top \mathbf{x}^{(m)} + 2 s_j^{(m)} \mu_j^\top \sum_{i\neq j} \mu_i \mathbb{E}(s_i^{(m)}) \right) \Bigg] \\
&= \exp \left[s_j^{(m)} \log \frac{\pi_j}{1 - \pi_j} - \frac{1}{2\sigma^2} \left(s_j^{(m)} \mu_j^\top \mu_j s_j^{(m)} - 2 s_j^{(m)} \mu_j^\top (\mathbf{x}^{(m)} - \sum_{i\neq j} \mu_i \lambda_i^{(m)}) \right) \right]
\end{aligned} \tag{6}$$

As we want the distribution:

$$q_j^m(s_j^{(m)}) = \lambda_{mj}^{s_j^{(m)}} (1 - \lambda_{mj})^{(1 - s_j^{(m)})} \Rightarrow \log q_j^m(s_j^{(m)}) \propto s_j^{(m)} \log \frac{\lambda_{mj}}{1 - \lambda_{mj}} \tag{7}$$

By the result from (6), we have:

$$\log q_j^m(s_j^{(m)}) \propto s_j^{(m)} \log \frac{\pi_j}{1 - \pi_j} - \frac{1}{2\sigma^2} \left(s_j^{(m)} \mu_j^\top \mu_j s_j^{(m)} - 2 s_j^{(m)} \mu_j^\top (\mathbf{x}^{(m)} - \sum_{i\neq j} \mu_i \lambda_i^{(m)}) \right) \tag{8}$$

When $s_j^{(m)} = 0$, both RHSs of (7) and (8) are equal obviously. When $s_j^{(m)} = 1$, we equate the RHSs of both (7) and (8):

$$\log \frac{\lambda_{mj}}{1 - \lambda_{mj}} = \log \frac{\pi_j}{1 - \pi_j} - \frac{1}{2\sigma^2} \left(\mu_j^\top \mu_j - 2 \mu_j^\top (\mathbf{x}^{(m)} - \sum_{i\neq j} \mu_i \lambda_i^{(m)}) \right) \tag{9}$$

Denote the RHS of (9) as S_{mj}, we have:

$$\lambda_{mj} = \frac{e^{S_{mj}}}{1 + e^{S_{mj}}} \tag{10}$$

Note that (since s_i is binary and independent):

$$\begin{aligned}
\left\langle s_i^{(n)} \right\rangle_{q_{in}} &= \mathbb{E}_{q_{in}}[s_i^{(n)}] = \lambda_i^{(n)} \\
\left\langle s_i^{(n)} s_j^{(n)} \right\rangle_{q_{in}q_{jn}} &= \mathbb{E}_{q_{in}q_{jn}}[s_i^{(n)} s_j^{(n)}] = \mathbb{E}_{q_{in}}[s_i^{(n)}]\mathbb{E}_{q_{jn}}[s_j^{(n)}] = \lambda_i^{(n)}\lambda_j^{(n)} \\
\left\langle (s_i^{(n)})^2 \right\rangle_{q_{in}} &= \mathbb{E}_{q_{in}}[(s_i^{(n)})^2] = \mathrm{Var}(s_i^{(n)}) + \mathbb{E}_{q_{in}}[s_i^{(n)}]^2 = \lambda_i^{(n)}(1-\lambda_i^{(n)}) + (\lambda_i^{(n)})^2 = \lambda_i^{(n)}
\end{aligned} \tag{11}$$

The free energy:

$$\begin{aligned}
\mathcal{F}(q(s), \boldsymbol{\theta}) &= \langle \log P(\mathbf{x}, \mathbf{s}|\boldsymbol{\theta}) \rangle_{q(s)} + H[q(s)] \\
&= \sum_n \left\langle \log P(\mathbf{x}^{(n)}, \mathbf{s}^{(n)}|\boldsymbol{\theta}) \right\rangle_{\prod_n q_n} + \sum_n H[q_n(s^{(n)})] \\
&= \sum_n \left\langle \log P(\mathbf{x}^{(n)}|\mathbf{s}^{(n)}, \boldsymbol{\theta}) + \log P(\mathbf{s}^{(n)}|\boldsymbol{\theta}) \right\rangle_{\prod_n q_n} + \sum_n H[q_n(s^{(n)})] \\
&= \sum_n \Bigg\langle -\frac{D}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\left(\mathbf{x}^{(n)} - \boldsymbol{\mu}\mathbf{s}^{(n)}\right)^\top \left(\mathbf{x}^{(n)} - \boldsymbol{\mu}\mathbf{s}^{(n)}\right) \\
&\quad + \sum_i s_i^{(n)} \log \pi_i + (1 - s_i^{(n)}) \log(1-\pi_i) \Bigg\rangle_{\prod_{n,i} q_{in}} \\
&\quad - \sum_n \sum_i \left\langle s_i^{(n)} \log \lambda_i^{(n)} + (1 - s_i^{(n)}) \log(1-\lambda_i^{(n)}) \right\rangle_{\prod_{n,i} q_{in}} \\
&= -\frac{ND}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_n \left[(\mathbf{x}^{(n)})^\top \mathbf{x}^{(n)} - 2(\mathbf{x}^{(n)})^\top \sum_i \boldsymbol{\mu}_i \langle s_i^{(n)} \rangle_{q_{in}} + \langle (\mathbf{s}^{(n)})^\top \boldsymbol{\mu}^\top \boldsymbol{\mu} \mathbf{s}^{(n)} \rangle_{q_{in}} \right] \\
&\quad + \sum_n \sum_i \Bigg[\langle s_i^{(n)} \rangle_{q_{in}} \log \pi_i + (1 - \langle s_i^{(n)} \rangle_{q_{in}}) \log(1-\pi_i) \\
&\quad - \left\langle s_i^{(n)} \right\rangle_{q_{in}} \log \lambda_i^{(n)} - (1 - \left\langle s_i^{(n)} \right\rangle_{q_{in}}) \log(1-\lambda_i^{(n)}) \Bigg] \\
&= -\frac{ND}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_n \Bigg[(\mathbf{x}^{(n)})^\top \mathbf{x}^{(n)} - 2(\mathbf{x}^{(n)})^\top \sum_i \boldsymbol{\mu}_i \lambda_i^{(n)} + \sum_{i \neq j} \sum_j \lambda_i^{(n)} \lambda_j^{(n)} \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j \\
&\quad + \lambda_i^{(n)} \boldsymbol{\mu}^\top \boldsymbol{\mu} \Bigg] + \sum_n \sum_i \Bigg[\lambda_i^{(n)} \log \pi_i + (1 - \lambda_i^{(n)}) \log(1-\pi_i) - \lambda_i^{(n)} \log \lambda_i^{(n)} \\
&\quad - (1 - \lambda_i^{(n)}) \log(1-\lambda_i^{(n)}) \Bigg] \\
&= -\frac{ND}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_n \Bigg[(\mathbf{x}^{(n)})^\top \mathbf{x}^{(n)} - 2(\mathbf{x}^{(n)})^\top \sum_i \boldsymbol{\mu}_i \lambda_i^{(n)} + \sum_{i \neq j} \sum_j \lambda_i^{(n)} \lambda_j^{(n)} \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j \\
&\quad + \lambda_i^{(n)} \boldsymbol{\mu}^\top \boldsymbol{\mu} \Bigg] + \sum_n \sum_i \left[\lambda_i^{(n)} \log \frac{\pi_i}{\lambda_i^{(n)}} + (1 - \lambda_i^{(n)}) \log \frac{1-\pi_i}{1-\lambda_i^{(n)}} \right]
\end{aligned} \tag{12}$$

The code for the function:

```
function [lambda,F] = MeanField(X,mu,sigma,pie,lambda0,maxsteps)
    [~,K] = size(lambda0);
    [N,D] = size(X);
    lambda = lambda0;
    epsilon = 10^(-8);
    F = -Inf;

    % avoid numerical issue
    pie(pie==1) = 1 - eps;
    pie(pie==0) = eps;

    for i=1:maxsteps
        % update ES, ESS
        for j=1:K
            notj = [1:(j-1),(j+1):K];
            ES = lambda;
            loglik = log(pie(j)/(1-pie(j))) - 1/(2*sigma^2)*(mu(:,
               j)'*mu(:,j)-2*X*mu(:,j) ...
                +2*ES(:,notj)*mu(:,notj)'*mu(:,j));
            infty = exp(loglik) == Inf;
            fty = exp(loglik) < Inf;
            lambda(fty,j) = 1 ./ (1+exp(-loglik(fty)));
            lambda(infty,j) = 1;
        end
        ES = lambda;
        ESS = lambda' * lambda;
        for i = 1: size (lambda ,2)
            ESS(i,i) = sum(lambda(:,i));
        end

        % avoid numerical issue
        lambda(lambda==0) = eps ;
        lambda(lambda==1) = 1 - eps ;

        % free energy
        Hs = -sum(lambda.*log(lambda) + (1-lambda).*log(1-lambda)
           ,2);
        currentF = -N*D/2*log(2*pi*sigma^2) - 1/(2*sigma^2)*(sum(
           sum(X.*X))-2*sum(sum((ES*mu').*X)) ...
            +sum(sum(ESS.*(mu'*mu)))) + sum(ES*log(pie./(1-pie))'
               + sum(log(1-pie))) + sum(Hs);

        if currentF - F < epsilon
            break
        end
        F = currentF;
    end
end
```

(b) Each observation $\mathbf{x}^{(n)}|\mathbf{s}_i, \boldsymbol{\mu}_i, \sigma^2$ is drawn from the Gaussian distribution $\mathcal{N}(\sum_i s_i\boldsymbol{\mu}_i, \sigma^2\mathbb{I})$, thus we can denote that $X = M^\top S + \epsilon \sim \mathcal{N}(\sum_i s_i\boldsymbol{\mu}_i, \sigma^2\mathbb{I})$, where $\epsilon \sim \mathcal{N}(0, \sigma^2\mathbb{I})$. We can hence compute a closed-form solution of M, which is $M = (SS^\top)^{-1}SX$. Thus the estimation of M is similar to the derivation of the linear regression ($\mathbf{Y} = \beta^\top\mathbf{X}$) solution, which is $\hat{\beta} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{Y}$.

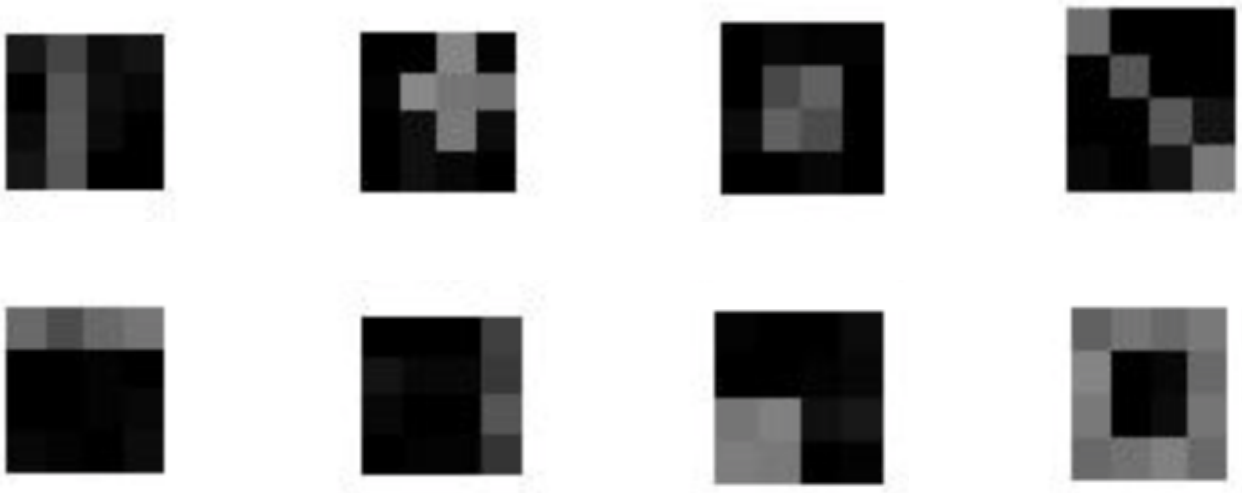
(c) Note that:

- X: $N \times D$, ES: $N \times K$, ESS: $K \times K$, mu: $D \times K$.
- Matrix multiplication of A($M \times N$) and B($N \times K$): $\mathcal{O}(MNK)$.
- Matrix inversion of A($N \times N$): $\mathcal{O}(N^3)$ (Gauss-Jordan elimination).
- Trace of matrix A($N \times N$): $\mathcal{O}(N)$.

The computational complexity of $\boldsymbol{\mu}$, $\boldsymbol{\pi}$ and σ are as followed:

- $\boldsymbol{\mu}$:
  - inv(ESS) $\rightarrow \mathcal{O}(K^3)$.
  - inv(ESS) $\times$ ES' $\rightarrow \mathcal{O}(NK^2)$.
  - (inv(ESS) $\times$ ES') $\times$ X $\rightarrow \mathcal{O}(KND)$.
  - total: $\mathcal{O}(K^3 + NK^2 + KND)$.
- $\boldsymbol{\pi}$:
  - mean(ES) $\rightarrow \mathcal{O}(NK)$.
- σ:
  - Tr(X'*X) $\rightarrow \mathcal{O}(ND^2 + D) \approx \mathcal{O}(ND^2)$.
  - Tr(mu'*mu*ESS) $\rightarrow \mathcal{O}(KD^2 + K^3 + K) \approx \mathcal{O}(KD^2 + K^3)$.
  - Tr(ES'*X*mu) $\rightarrow \mathcal{O}(KND + DK^2 + K) \approx \mathcal{O}(KND + DK^2)$.
  - total: $\mathcal{O}(ND^2 + KD^2 + K^3 + KND + DK^2)$.
- Total complexity for M-step:
  - $\mathcal{O}(ND^2 + KD^2 + K^3 + KND + DK^2 + NK + K^3 + NK^2 + KND)$
    $\approx \mathcal{O}(ND^2 + KD^2 + K^3 + KND + DK^2 + NK^2)$.

(d) The possible features (8 features in total) are displayed below:

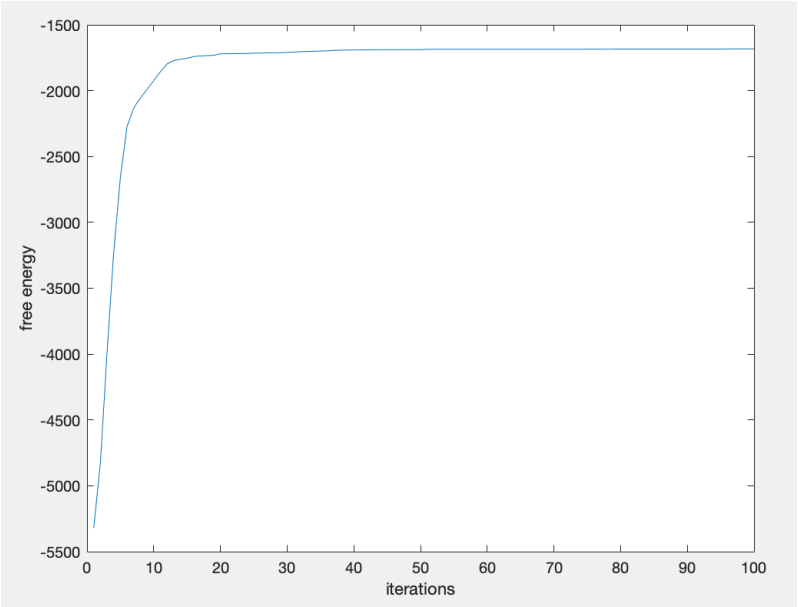


- Factor analysis doesn't work on the data. Our latent variable here is binary instead of continuous Gaussian.
- Mixture of Gaussian doesn't work either, which cannot capture the combined effect of different features. MOG produces single point with noise instead.
- ICA may work on the data. Treat different features as different sources. Latent variable S combines all the feature effects into a single output, with feature i represents by s_i.

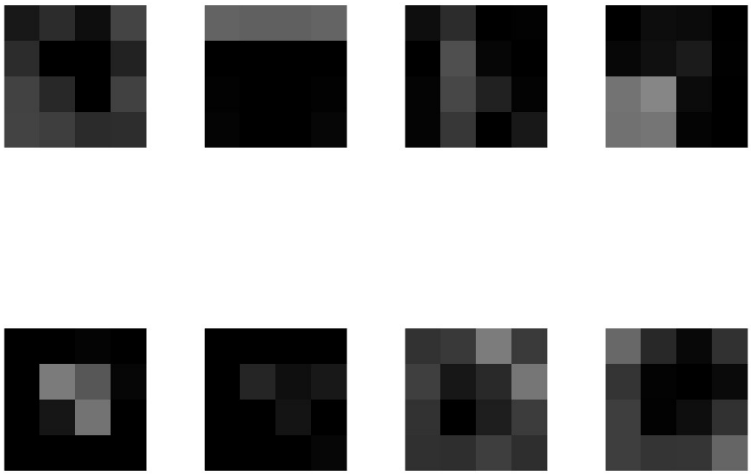
(e) The function is shown below:

```
function [mu, sigma, pie] = LearnBinFactors(X,K,iterations)
    [N,~]=size(X);
    epsilon = 10^(-8);
    F = -inf;
    maxsteps = 50;

    % initialise
    lambda0 = rand(N,K);
    ES = lambda0;
    ESS = zeros(N,K,K);
    for n=1:N
        tmp = lambda0(n,:)'*lambda0(n,:);
        tmp(logical(eye(K)))=lambda0(n,:);
        ESS(n,:,:) = tmp;
    end
    [mu, sigma, pie] = MStep(X,ES,ESS);

    % EM
    for j=1:iterations
        % E-step
        [lambda,F_new] = MeanField(X,mu,sigma,pie,lambda0,maxsteps
           );

        % M-step
        ES = lambda;
        ESS = zeros(N,K,K);
        for n=1:N
            tmp = lambda(n,:)'*lambda(n,:);
            tmp(logical(eye(K)))=lambda(n,:);
            ESS(n,:,:) = tmp;
        end
        [mu, sigma, pie] = MStep(X,ES,ESS);

        if F_new-F < epsilon
            break;
        end
        F = F_new;
        lambda0 = lambda;
    end
end
```

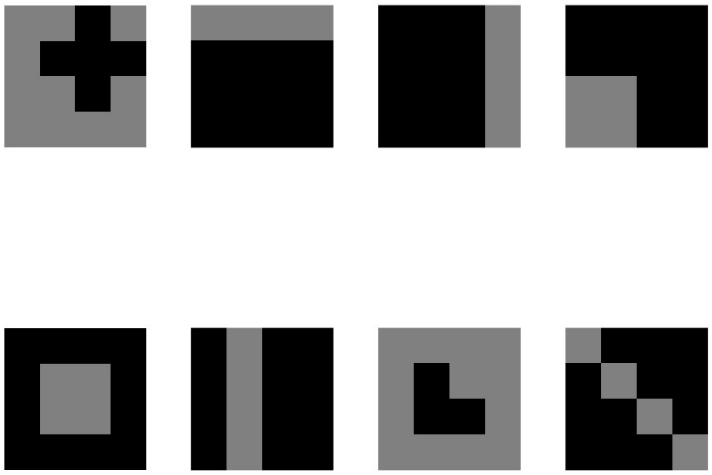
The plot below indicates that the free energy is increasing in each iteration:



(f) The learned features are displayed below:

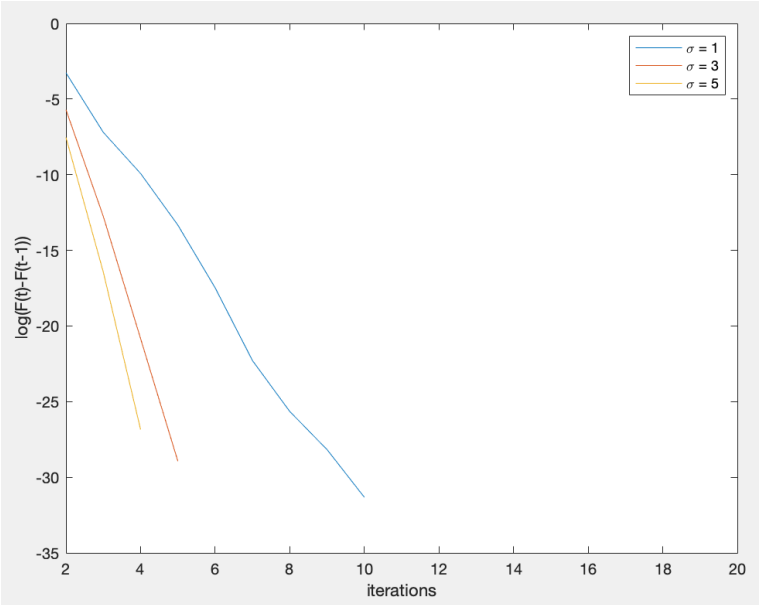


- **Improve the algorithm**:
  - Run the algorithm several times and choose the parameters with highest free energy, since our results depend significantly on the initialised values, which are random.
  - Add priors to our parameters μ, σ and π for more accurate estimation.
  - Approximate the likelihood by BIC or Laplace approximation.
- **Improve the features**: build a classifier to remap the learned features (μ) to binary data. For example, we set the threshold as 0.5: the μ values that are under 0.5 as set to 0, otherwise 1. We obtain the following improved results:



- **What to set K?** From (d), we set $K = 8$. K can be larger than 8, however, a too-large K may cause overfitting.
- **How to initialise parameters?** We first randomise λ_0 uniformly, then we calculate other parameters (μ, σ and π) based on the random λ_0 by M-step.

(g) The figure including 3 different sigmas is displayed below:



We discover that smaller σ has larger free energy, but it also requires longer time to converge. We know that σ quantifies the noise of $\mathbf{x}$. When the noise is large (σ is large), a rough model can make the free energy converges, this explains the reason why larger σ has faster speed of convergence. But the model is less precise in this case. When the noise is small (σ is small), the model we obtain is closer to the "truth", which is more precise. However, it takes much longer time to converge. Thus, there is a tradeoff between the convergence speed and the model accuracy.

5. (a) Given that

$$\begin{aligned} P(\mathbf{s}) &\sim \prod_i^K \pi_i^{s_i}(1-\pi_i)^{(1-s_i)} \\ P(\mathbf{x}|\mathbf{s}) &\sim \mathcal{N}(\sum_i s_i\boldsymbol{\mu}_i, \sigma^2\mathbb{I}) \,, \end{aligned} \tag{13}$$

we derive the log joint:

$$\begin{aligned} \log P(\mathbf{x},\mathbf{s}) &= \log P(\mathbf{x}|\mathbf{s}) + \log P(\mathbf{s}) \\ &= -\frac{K}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\left(\mathbf{x}^\top\mathbf{x} - 2\mathbf{x}^\top\sum_i s_i\boldsymbol{\mu}_i + \sum_{i,j} s_i s_j \boldsymbol{\mu}_i^\top\boldsymbol{\mu}_j\right) \\ &\quad + \sum_i \left(s_i \log\pi_i + (1-s_i)\log(1-\pi_i)\right) \\ &= -\frac{1}{2\sigma^2}\left(-2\mathbf{x}^\top\sum_i s_i\boldsymbol{\mu}_i + \sum_{i,j} s_i s_j \boldsymbol{\mu}_i^\top\boldsymbol{\mu}_j\right) + \sum_i s_i \log\frac{\pi_i}{1-\pi_i} + C \\ &= \sum_i s_i\left(\frac{\mathbf{x}^\top\boldsymbol{\mu}_i}{\sigma^2} + \log\frac{\pi_i}{1-\pi_i}\right) - \sum_{i,j}\frac{s_i s_j\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_j}{2\sigma^2} + C \\ &= \sum_i s_i\left(\frac{\mathbf{x}^\top\boldsymbol{\mu}_i}{\sigma^2} + \log\frac{\pi_i}{1-\pi_i}\right) - \left(\sum_i \frac{s_i^2\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_i}{2\sigma^2} + \sum_{i\neq j}\frac{s_i s_j\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_j}{2\sigma^2}\right) + C \\ &= \sum_i \log\left(\exp\left[s_i\left(\frac{\mathbf{x}^\top\boldsymbol{\mu}_i}{\sigma^2} + \log\frac{\pi_i}{1-\pi_i} - \frac{\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_i}{2\sigma^2}\right)\right]\right) \\ &\quad + \sum_{i\neq j}\log\left(\exp\left[-\frac{s_i s_j\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_j}{2\sigma^2}\right]\right) + C \,, \text{since } s \text{ are binary, thus } s_i^2 = s_i \\ &= \sum_i \log f_i(s_i) + \sum_{i\neq j}\log g_{ij}(s_i,s_j) + C \,, \end{aligned} \tag{14}$$

where

$$\begin{aligned} f_i(s_i) &= \exp\left[s_i\left(\frac{\mathbf{x}^\top\boldsymbol{\mu}_i}{\sigma^2} + \log\frac{\pi_i}{1-\pi_i} - \frac{\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_i}{2\sigma^2}\right)\right] \\ g_{ij}(s_i,s_j) &= \exp\left[-\frac{s_i s_j\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_j}{2\sigma^2}\right] \end{aligned} \tag{15}$$

Our log joint can also be arranged to:

$$\begin{aligned} \log P(\mathbf{x},\mathbf{s}) &= \sum_i s_i\left(\frac{\mathbf{x}^\top\boldsymbol{\mu}_i}{\sigma^2} + \log\frac{\pi_i}{1-\pi_i} - \frac{\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_i}{2\sigma^2}\right) - \sum_{i\neq j}\frac{s_i s_j\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_j}{2\sigma^2} + C \\ &= \sum_i b_i s_i + \sum_{i<j} W_{ij} s_i s_j - \log Z \,, \end{aligned} \tag{16}$$

where

$$b_i = \frac{\mathbf{x}^\top\boldsymbol{\mu}_i}{\sigma^2} + \log\frac{\pi_i}{1-\pi_i} - \frac{\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_i}{2\sigma^2}, \quad W_{ij} = -\frac{\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_j}{2\sigma^2}, \quad C = -\log Z \tag{17}$$

The Boltzmann Machine is:

$$\begin{aligned} P(\mathbf{s}|W,\mathbf{b}) &= \frac{1}{Z}\exp\left\{\sum_{ij} W_{ij}s_is_j + \sum_i b_is_i\right\} \\ \log P(\mathbf{s}|W,\mathbf{b}) &= \sum_{ij} W_{ij}s_is_j + \sum_i b_is_i - \log Z \end{aligned} \tag{18}$$

By comparing the result of (12) and (14), we conclude that the log joint is the Boltzmann Machine.

(b) Our approximate probability function is:

$$q(\mathcal{S}) = \frac{1}{Z}\prod_i f_i(s_i)\prod_{i\neq j} g_{ij}(s_i,s_j) = \frac{1}{Z}\prod_i \tilde{f}_i(s_i)\prod_{i\neq j}\tilde{g_{ij}}(s_i,s_j) \tag{19}$$

Approximate $f_i(s_i)$:
To approximate the marginal probability of s_i:

$$\tilde{f}_i(s_i) = \underset{f'\in Bern}{\arg\min}\,\mathrm{KL}\left[f_i(s_i)\prod_{i\neq j}\tilde{g_{ij}}(s_i,s_j)\middle\|f'_i(s_i)\prod_{i\neq j}\tilde{g_{ij}}(s_i,s_j)\right] \tag{20}$$

Since $f'_i(s_i)$ is constrained to be Bernoulli distribution, and the true $\tilde{f}_i(s_i)$ is Bernoulli, the projection step is just: $\tilde{f}_i(s_i) = f_i(s_i)$.

Approximate $g_{ij}(s_i,s_j)$:
We approximate the pairwise factor by the product of two Bernoulli distributions, i.e. $\tilde{g_{ij}}(s_i,s_j) \approx \tilde{g_{j\to i}}(s_i)\tilde{g_{i\to j}}(s_j)$. Take $\tilde{g_{j\to i}}(s_i)$ as an example, say $\tilde{g_{j\to i}}(s_i) \sim Bern(\theta_{ji})$, then:

$$\begin{aligned} \tilde{g_{j\to i}}(s_i) &= \theta_{ji}^{s_i} + (1-\theta_{ji})^{(1-s_i)} \\ \log \tilde{g_{j\to i}}(s_i) &\propto s_i \log\frac{\theta_{ji}}{1-\theta_{ji}} \\ &= s_iW_{j\to i} \\ \text{Thus, } \tilde{g_{j\to i}}(s_i) &\propto \exp(s_iW_{j\to i})\,, \end{aligned} \tag{21}$$

where we denote the natural parameter of $\tilde{g_{j\to i}}(s_i)$ as $W_{j\to i}$, and similarly, we denote the natural parameter of $\tilde{g_{i\to j}}(s_j)$ as $W_{i\to j}$.
By combining the results derived in (15) and (17), we have:

$$f_i(s_i) = \exp[s_ib_i],\ g_{ij}(s_i,s_j) = \exp[s_is_jW_{ij}]\,, \tag{22}$$

where b_i and W_{ij} are the natural parameters of $f_i(s_i)$ and $g_{ij}(s_i,s_j)$ respectively.
Thus:

$$\begin{aligned} \tilde{g_{ij}}(s_i,s_j) = \underset{g'_{m\to i},g'_{n\to j}\in Bern}{\arg\min}\ \mathrm{KL}\Bigg[& g_{ij}(s_i,s_j)f_i(s_i)f_j(s_j)\prod_{m\neq i,j}\tilde{g_{m\to i}}(s_i)\prod_{n\neq i,j}\tilde{g_{n\to j}}(s_j) \\ & \Bigg\|\left(g'_{j\to i}(s_i)\tilde{f}_i(s_i)\prod_{m\neq i,j}\tilde{g_{m\to i}}(s_i)\right)\left(g'_{i\to j}(s_j)\tilde{f}_j(s_j)\prod_{n\neq i,j}\tilde{g_{n\to j}}(s_j)\right)\Bigg] \\ = \underset{g'_{m\to i},g'_{n\to j}\in Bern}{\arg\min}\ \mathrm{KL}\Bigg(& \exp(s_is_jW_{ij} + s_ib_i + s_jb_j + \sum_{m\neq i,j}s_iW_{m\to i} + \sum_{n\neq i,j}s_jW_{n\to j}) \\ & \Bigg\|\exp(s_i(W'_{j\to i} + b_i + \sum_{m\neq i,j}W_{m\to i}))\cdot\exp(s_j(W'_{i\to j} + b_j + \sum_{n\neq i,j}W_{n\to j}))\Bigg) \end{aligned} \tag{23}$$

The minimum is found by matching sufficient statistics.

We denote:

$$\begin{aligned} \text{the LHS of KL:} \quad & L_{ij}(s_i, s_j) \\ \text{the RHS of KL:} \quad & R_{ij}(s_i, s_j) \\ \eta_i &= b_i + \sum_{m \neq i,j} W_{m \to i} \\ \eta_j &= b_j + \sum_{n \neq i,j} W_{n \to j} \end{aligned} \tag{24}$$

Thus:

$$\begin{aligned} \mathbb{E}[L_{ij}(s_i)] &= \frac{\exp(W_{ij} + \eta_i + \eta_j) + \exp(\eta_i)}{\exp(W_{ij} + \eta_i + \eta_j) + \exp(\eta_i) + \exp(\eta_j) + 1} \\ \mathbb{E}[L_{ij}(s_j)] &= \frac{\exp(W_{ij} + \eta_i + \eta_j) + \exp(\eta_j)}{\exp(W_{ij} + \eta_i + \eta_j) + \exp(\eta_j) + \exp(\eta_i) + 1} \\ \mathbb{E}[R_{ij}(s_i)] &= \frac{1}{1 + \exp(-(W'_{j \to i} + \eta_i))} \\ \mathbb{E}[R_{ij}(s_j)] &= \frac{1}{1 + \exp(-(W'_{i \to j} + \eta_j))} \end{aligned} \tag{25}$$

Equating:

$$\begin{aligned} \mathbb{E}[L_{ij}(s_i)] &= \mathbb{E}[R_{ij}(s_i)] \\ \mathbb{E}[L_{ij}(s_j)] &= \mathbb{E}[R_{ij}(s_j)] \end{aligned} \tag{26}$$

By solving (26), we get the natural parameters of the desired factors:

$$\begin{aligned} W'_{j \to i} &= \log\left(\frac{\exp(W_{ij} + \eta_j) + 1}{\exp(\eta_j) + 1}\right) = \log\left(\frac{\exp(W_{ij} + b_j + \sum_{n \neq i,j} W_{n \to j}) + 1}{\exp(b_j + \sum_{n \neq i,j} W_{n \to j}) + 1}\right) \\ W'_{i \to j} &= \log\left(\frac{\exp(W_{ij} + \eta_i) + 1}{\exp(\eta_i) + 1}\right) = \log\left(\frac{\exp(W_{ij} + b_i + \sum_{m \neq i,j} W_{m \to i}) + 1}{\exp(b_i + \sum_{m \neq i,j} W_{m \to i}) + 1}\right) \end{aligned} \tag{27}$$

The final message-passing scheme:

- Set b_i by parameters and observations, set $W_{ij} = 0$.
- Update $W_{i \to j}$ according to (27) in each iteration until reaching the convergence.

(c) From (b), we know that:

$$\begin{aligned} \eta_i &= b_i + \sum_{m \neq i,j} W_{m \to i} \\ \eta_j &= b_j + \sum_{n \neq i,j} W_{n \to j} \end{aligned} \tag{28}$$

Take the first equation as an example. The RHS reveals the summation of the singleton factor i's natural parameter and all the natural parameters of all i's neighbours except j.

This leads to a loopy BP algorithm, since it indicates that every node is the neighbour of all other nodes (all nodes are fully connected).

(d) We apply ARD (automatic relevance determination) to loopy-BP.

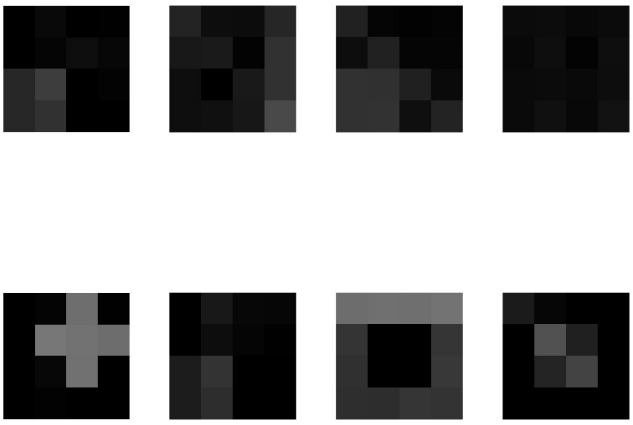
Add $\boldsymbol{\pi}$ as a latent variable (or say a prior) generated from hyperparameters. Then the joint:

$$P(\mathbf{s}, \boldsymbol{\pi} | \boldsymbol{\theta}) = P(\mathbf{s} | \boldsymbol{\pi}, \boldsymbol{\theta}) P(\boldsymbol{\pi} | \boldsymbol{\theta}) \,, \tag{29}$$

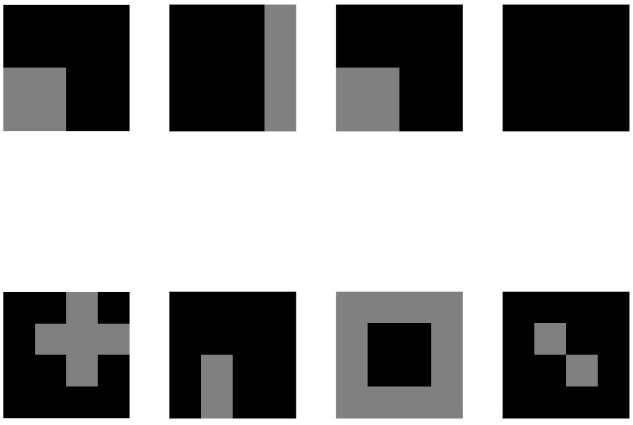
where $\boldsymbol{\pi}|\boldsymbol{\theta} \sim \exp(\boldsymbol{\theta}^\top \boldsymbol{\pi})$ and $\boldsymbol{\theta}$ is the natural parameter. Then use loopy-BP to approximate the joint. In M-step, we obtain the optimised $\boldsymbol{\theta}^*$, and we check each entry of $\boldsymbol{\theta}^*$: find $\boldsymbol{\theta}^*_d \nrightarrow -\infty$ (relevant dimensions), which is K.

6. We use loopy BP to replace the MeanField as E-step, while the M-step remain the same.

   The learned features by loopy-BP are as followed:



   The improved features are as followed:



   While the iterations for loopy BP and mean field are both 100 times, the mean field performs better by obtaining much clearer features. Specifically, from the above 2 figures obtained by loopy BP, some features are not clear (even goes all black), while all the features obtained by mean field (see figures in 3(f)) are much more obvious.

   This may be caused by: the loopy-BP does not necessarily converge due to the model setting.

The function LoopyBP:

```
function [lambda,Message,F] = LoopyBP(X,mu,sigma,pie,Message0,maxsteps)
    [N,D] = size(X);
    [~,K,~] = size(Message0);
    Message = zeros(size(Message0));
    F = -inf;
    epsilon = 10^(-8);

    % natural parameter of fi(si): bi
    f = zeros(N,K);
    for n=1:N
       f(n,:) = log(pie./(1-pie)) + X(n,:)*mu/(sigma^2) - diag(mu'*mu)
          '/(2*sigma^2);
    end

    for iter=1:maxsteps
        for n=1:N
            message_n = Message0(:,:,n);
            for i=1:K
                for j=(i+1):K
                    alpha = 0.5;
                    % update wji
                    Wij = - mu(:,i)'*mu(:,j)/(sigma^2);
                    nj = f(n,j) + sum(message_n(:,j)) - message_n(i,j)
                       ;
                    wji = (exp(Wij+nj)+1)/(exp(nj)+1);
                    % message_n(j,i) = log(wji);
                    message_n(j,i) = alpha*message_n(j,i) + (1-alpha)*
                       log(wji);

                    %update wij
                    Wji = - mu(:,j)'*mu(:,i)/(sigma^2);
                    ni = f(n,i) + sum(message_n(:,i)) - message_n(j,i)
                       ;
                    wij = (exp(Wji+ni)+1)/(exp(ni)+1);
                    % message_n(i,j) = log(wij);
                    message_n(i,j) = alpha*message_n(i,j) + (1-alpha)*
                       log(wij);
                end
            end
            Message(:,:,n) = message_n;
        end

        lambda = zeros(N,K);
        for n=1:N
            p =  f(n,:)+sum(Message(:,:,n));
            lambda(n,:) = 1./(1+exp(-p));
        end

```

```
        ES = lambda;
        ESS = lambda' * lambda;
        for i = 1: size (lambda ,2)
            ESS(i,i) = sum(lambda(:,i));
        end

        % avoid numerical issue
        lambda(lambda==0) = eps ;
        lambda(lambda==1) = 1 - eps ;

        % free energy
        Hs = -sum(lambda.*log(lambda) + (1-lambda).*log(1-lambda),2);
        currentF = -N*D/2*log(2*pi*sigma^2) - 1/(2*sigma^2)*(sum(sum(X
           .*X))-2*sum(sum((ES*mu').*X)) ...
            +sum(sum(ESS.*(mu'*mu)))) + sum(ES * log(pie./(1-pie))' +
               sum(log(1-pie))) + sum(Hs);

        diff = max(max(max(Message-Message0)));

        if diff < epsilon
            break
        end
        F = currentF;
        Message0 = Message;

    end
end
```

The updated function LearnBinFactors:

```
function [mu,sigma,pie,lambda] = LearnBinFactors(X,K,iterations)
    [N,D] = size(X);
    F = -inf;
    epsilon = 10^(-8);
    maxsteps = 50;

    % initialisation
    lambda0 = rand(N,K);
    ES = lambda0;
    ESS = zeros(N,K,K);
    for n=1:N
        tmp = lambda0(n,:)'*lambda0(n,:);
        tmp(logical(eye(K)))=lambda0(n,:);
        ESS(n,:,:) = tmp;
    end
    [mu, sigma, pie] = MStep(X,ES,ESS);
    Message0 = rand(K,K,N);
    for n=1:N
        b = Message0(:,:,n);
        b = b-diag(diag(b));
        Message0(:,:,n) = b;
```

```
    end

    Ftotal = zeros(1,iterations);
    for j=1:iterations
        % E-step
        [lambda,Message,currentF] = LoopyBP(X,mu,sigma,pie,Message0,
           maxsteps);
        Ftotal(:,j) = currentF;

        % M-step
        ES = lambda;
        ESS = zeros(N,K,K);
        for n=1:N
            tmp = lambda(n,:)'*lambda(n,:);
            tmp(logical(eye(K)))=lambda(n,:);
            ESS(n,:,:) = tmp;
        end
        [mu, sigma, pie] = MStep(X,ES,ESS);

        if currentF - F < epsilon
            break
        end
        F = currentF;
        Message0 = Message;

    end
    figure
    plot(Ftotal);xlabel("iterations");ylabel("free energy")
end
```