Comp 0085 Assignment 5

Alexandra Maria Proca (SN: 20047328)

January 6, 2021

Question 1.

a. The E step is computed by optimizing the free energy $\mathcal{F}(q, \boldsymbol{\theta})$ with respect to the distribution over the hidden variables $\mathcal{S}$ holding the parameters fixed. $\mathcal{F}(q, \boldsymbol{\theta})$ can be written as

$\mathcal{F}(q, \boldsymbol{\theta}) = \langle \log p(\mathcal{S}, \mathcal{X}|\boldsymbol{\theta}) \rangle_{q(\mathcal{S})} - \langle \log q(\mathcal{S}) \rangle_{q(\mathcal{S})}$

$= \sum_{n=1}^{N} \left\langle \log p\left(\boldsymbol{s}^{(n)}, \boldsymbol{x}^{(n)}|\boldsymbol{\theta}\right) \right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)} - \sum_{n=1}^{N} \left\langle \log q\left(\boldsymbol{s}^{(n)}\right) \right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}$

$= \sum_{n=1}^{N} \left\langle \log p\left(\boldsymbol{x}^{(n)}|\boldsymbol{s}^{(n)}, \boldsymbol{\theta}\right) + \log P\left(\boldsymbol{s}^{(n)}|\boldsymbol{\theta}\right) \right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)} - \sum_{n=1}^{N} \left\langle \log q\left(\boldsymbol{s}^{(n)}\right) \right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}$

Substituting the given distributions,

$$= \sum_{n=1}^{N} [-\frac{D}{2} \log 2\pi - D \log \sigma - \frac{1}{2\sigma^2} \left\langle \left(\boldsymbol{x}^{(n)} - \sum_{i=1}^{K} s_i^{(n)} \boldsymbol{\mu}_i \right)^{\mathsf{T}} \left(\boldsymbol{x}^{(n)} - \sum_{i=1}^{K} s_i^{(n)} \boldsymbol{\mu}_i \right) \right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}$$
$$+ \sum_{i=1}^{K} (\left\langle s_i^{(n)} \right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)} \log \boldsymbol{\pi}_i + (1 - \left\langle s_i^{(n)} \right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}) \log(1 - \boldsymbol{\pi}_i)$$
$$- \left\langle s_i^{(n)} \right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)} \log \lambda_{in} - (1 - \left\langle s_i^{(n)} \right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}) \log(1 - \lambda_{in}))]$$

$$= \sum_{n=1}^{N} [-\frac{D}{2} \log 2\pi - D \log \sigma - \frac{1}{2\sigma^2} (\boldsymbol{x}^{(n)\mathsf{T}} \boldsymbol{x}^{(n)} - 2 \sum_{i=1}^{K} \left\langle s_i^{(n)} \right\rangle_{q(\boldsymbol{s}^{(n)})} \boldsymbol{\mu}_i^{\mathsf{T}} \boldsymbol{x}^{(n)} + \sum_{i,j=1}^{K} \left\langle s_i^{(n)} s_j^{(n)} \right\rangle_{q(\boldsymbol{s}^{(n)})} \boldsymbol{\mu}_i^{\mathsf{T}} \boldsymbol{\mu}_j)$$
$$+ \sum_{i=1}^{K} \left(\left\langle s_i^{(n)} \right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)} \log \frac{\boldsymbol{\pi}_i}{\lambda_{in}} + (1 - \left\langle s_i^{(n)} \right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}) \log \frac{(1 - \boldsymbol{\pi}_i)}{(1 - \lambda_{in})} \right)]$$

Because the variational approximation $q_n(\boldsymbol{s}^{(n)})$ uses λ_{in} to model the posterior of s_i^n, $\langle s_i^n \rangle = \lambda_{in}$ and $\langle s_i^n s_j^n \rangle = \lambda_{in}\lambda_{jn} + \delta_{ij}(\lambda_{in} - \lambda_{in}^2)$.

Substituting in the expected values,

$$\mathcal{F}(q, \boldsymbol{\theta}) = \sum_{n=1}^{N} [-\frac{D}{2} \log 2\pi - D \log \sigma - \frac{1}{2\sigma^2} \left(\boldsymbol{x}^{(n)\mathsf{T}} \boldsymbol{x}^{(n)} - 2 \sum_{i=1}^{K} \lambda_{in} \boldsymbol{\mu}_i^{\mathsf{T}} \boldsymbol{x}^{(n)} + \left(\left(\sum_{i,j=1}^{K} \lambda_{in}\lambda_{jn} + \sum_{i=1}^{K} \left(\lambda_{in} - \lambda_{in}^2 \right) \right) \boldsymbol{\mu}_i^{\mathsf{T}} \boldsymbol{\mu}_i \right) \right)$$
$$+ \sum_{i=1}^{K} \left(\lambda_{in} \log \frac{\boldsymbol{\pi}_i}{\lambda_{in}} + (1 - \lambda_{in}) \log \frac{(1 - \boldsymbol{\pi}_i)}{(1 - \lambda_{in})} \right)]$$

Taking the derivative of $\mathcal{F}$ with respect to a single fixed point λ_{in} and setting to the expression to 0,

$$\frac{\partial \mathcal{F}}{\partial \lambda_{in}} = \frac{1}{\sigma^2} \boldsymbol{\mu}_i^{\mathsf{T}} \boldsymbol{x}^{(n)} - \frac{1}{2\sigma^2} \sum_{j \neg i}^{K} \lambda_{jn} \boldsymbol{\mu}_i^{\mathsf{T}} \boldsymbol{\mu}_j - \frac{1}{\sigma^2} \lambda_{in} \boldsymbol{\mu}_i^{\mathsf{T}} \boldsymbol{\mu}_i - \frac{1}{2\sigma^2} \boldsymbol{\mu}_i^{\mathsf{T}} \boldsymbol{\mu}_i + \frac{1}{\sigma^2} \lambda_{in} \boldsymbol{\mu}_i^{\mathsf{T}} \boldsymbol{\mu}_i$$
$$+ \log \boldsymbol{\pi}_i - 1 - \log \lambda_{in} - \log(1 - \boldsymbol{\pi}_i) + \frac{1}{1 - \lambda_{in}} - \frac{\lambda_{in}}{1 - \lambda_{in}} + \log(1 - \lambda_{in})$$
$$= \frac{1}{\sigma^2} \boldsymbol{\mu}_i^{\mathsf{T}} \boldsymbol{x}^{(n)} - \frac{1}{2\sigma^2} \sum_{j \neg i}^{K} \lambda_{jn} \boldsymbol{\mu}_i^{\mathsf{T}} \boldsymbol{\mu}_j - \frac{1}{2\sigma^2} \boldsymbol{\mu}_i^{\mathsf{T}} \boldsymbol{\mu}_i + \log \frac{\boldsymbol{\pi}_i}{1 - \boldsymbol{\pi}_i} - \log \frac{\lambda_{in}}{1 - \lambda_{in}} + \frac{1 - \lambda_{in}}{1 - \lambda_{in}} - 1$$

$$= \frac{1}{\sigma^2}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{x}^{(n)} - \frac{1}{2\sigma^2}\sum_{j\neg i}^{K}\lambda_{jn}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{\mu}_j - \frac{1}{2\sigma^2}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{\mu}_i + \log\frac{\boldsymbol{\pi}_i}{1-\boldsymbol{\pi}_i} - \log\frac{\lambda_{in}}{1-\lambda_{in}} = 0$$

Solving for λ_{in},

$$\log\frac{\lambda_{in}}{1-\lambda_{in}} = \frac{1}{\sigma^2}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{x}^{(n)} - \frac{1}{2\sigma^2}\sum_{j\neg i}^{K}\lambda_{jn}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{\mu}_j - \frac{1}{2\sigma^2}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{\mu}_i + \log\frac{\boldsymbol{\pi}_i}{1-\boldsymbol{\pi}_i}$$

$$\log\frac{1-\lambda_{in}}{\lambda_{in}} = -\left(\frac{1}{\sigma^2}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{x}^{(n)} - \frac{1}{2\sigma^2}\sum_{j\neg i}^{K}\lambda_{jn}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{\mu}_j - \frac{1}{2\sigma^2}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{\mu}_i + \log\frac{\boldsymbol{\pi}_i}{1-\boldsymbol{\pi}_i}\right)$$

$$\frac{1-\lambda_{in}}{\lambda_{in}} = \exp\left\{-\left(\frac{1}{\sigma^2}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{x}^{(n)} - \frac{1}{2\sigma^2}\sum_{j\neg i}^{K}\lambda_{jn}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{\mu}_j - \frac{1}{2\sigma^2}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{\mu}_i + \log\frac{\boldsymbol{\pi}_i}{1-\boldsymbol{\pi}_i}\right)\right\}$$

$$\frac{1-\lambda_{in}}{\lambda_{in}} + \frac{\lambda_{in}}{\lambda_{in}} = \frac{1}{\lambda_{in}} = \exp\left\{-\left(\frac{1}{\sigma^2}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{x}^{(n)} - \frac{1}{2\sigma^2}\sum_{j\neg i}^{K}\lambda_{jn}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{\mu}_j - \frac{1}{2\sigma^2}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{\mu}_i + \log\frac{\boldsymbol{\pi}_i}{1-\boldsymbol{\pi}_i}\right)\right\} + 1$$

$$\lambda_{in} = \frac{1}{\exp\left\{-\left(\frac{1}{\sigma^2}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{x}^{(n)} - \frac{1}{2\sigma^2}\sum_{j\neg i}^{K}\lambda_{jn}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{\mu}_j - \frac{1}{2\sigma^2}\boldsymbol{\mu}_i^\mathsf{T}\boldsymbol{\mu}_i + \log\frac{\boldsymbol{\pi}_i}{1-\boldsymbol{\pi}_i}\right)\right\} + 1}$$

Using the derived E step and equation for $\mathcal{F}$, the function `MeanField(X,mu,sigma,pie,lambda0,maxsteps)` is implemented, iteratively updating each λ_{in} using the fixed values of the other λ_{in} in the E step. $\mathcal{F}$ is then computed using the updated values of λ and the function terminates when $\mathcal{F}$ converges or the maximum number of steps have been reached, whichever occurs first.

```
import numpy as np
import matplotlib.pyplot as plt

# Computes F
def FreeEnergy(X,mu,sigma,pie,lambda0):
    F=0
    D=lambda0.shape[1]
    for n in range(lambda0.shape[0]):
        tF=X[n]-np.dot(lambda0[n],mu.T)
        tF=np.dot(tF.T,tF)
        tF2=np.sum(np.dot(np.dot(lambda0[n]-lambda0[n]**2,mu.T),mu))
        tF=(-1/(2*sigma**2))*(tF+tF2)-D*np.log(sigma)-(D/2)*np.log(2*np.pi)
        tF2=np.dot(lambda0[n],np.log(pie[0]).T)
            -np.dot(lambda0[n],np.log(lambda0[n]+.0000001).T)
        tF2=tF2+np.sum(np.log(1-pie[0]))-np.sum(np.log(1-lambda0[n]+.0000001))
        tF2=tF2-np.dot(lambda0[n],np.log(1-pie[0]).T)
            + np.dot(lambda0[n],np.log(1-lambda0[n]+.0000001))
        F+=tF+tF2
    return F

# Computes E step, maximizing F wrt S holding parameters fixed for s(n) at i
def EStep(X,mu,sigma,pie,lambda0,i,n):
    tlambda=np.dot(lambda0[n],mu.T)-np.dot(lambda0[n,i],mu[:,i].T)
    tlambda=(1/sigma**2)*np.dot((X[n]-tlambda),mu[:,i])
    tlambda=tlambda-(1/(2*sigma**2))*np.dot(mu[:,i].T,mu[:,i])
    tlambda=tlambda + np.log(pie[0][i]/(1-pie[0][i]))
    tlambda=1/(1+np.exp(-tlambda))
    return tlambda

# Updates each lambda iteratively in the E step and computes new F;
# halts when F has converged or maxsteps have been reached
def MeanField(X,mu,sigma,pie,lambda0,maxsteps):
```

```
lambda1=np.copy(lambda0)
prevlambda=lambda1
prevF=FreeEnergy(X,mu,sigma,pie,lambda1)
for m in range(maxsteps):
    for n in range(lambda1.shape[0]):
        for i in range(lambda1.shape[1]):
            lambda1[n][i]=EStep(X,mu,sigma,pie,lambda1,i,n)
    F=FreeEnergy(X,mu,sigma,pie,lambda1)
    if F-prevF < 0.0000001:
        break
    prevF=F
    prevlambda=lambda1
return lambda1,F
```

b. In the computation of the free energy $\mathcal{F}$, the expectation of the likelihood is found, given the probabilities $p(\boldsymbol{s}|\boldsymbol{\pi})$ and $p(\boldsymbol{x}|\boldsymbol{s},\boldsymbol{\mu},\sigma^2)$. The probability of $\boldsymbol{x}$ given the hidden states and parameters, $p\left(\boldsymbol{x}|s_1,\ldots,s_K,\boldsymbol{\mu},\sigma^2\right)=\mathcal{N}\left(\sum_{i=1}^{K}s_i\boldsymbol{\mu}_i,\sigma^2 I\right)$ is a Gaussian distribution.
Rewriting $p(\boldsymbol{x}|\boldsymbol{s},\boldsymbol{\mu},\sigma^2)$,
$$\sum_{n=1}^{N}\left[-\frac{D}{2}\log 2\pi - D\log\sigma - \frac{1}{2\sigma^2}\left\langle\left(\boldsymbol{x}^{(n)}-\sum_{i=1}^{K}s_i^{(n)}\boldsymbol{\mu}_i\right)^{\mathsf{T}}\left(\boldsymbol{x}^{(n)}-\sum_{i=1}^{K}s_i^{(n)}\boldsymbol{\mu}_i\right)\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}\right]$$
The term $\left\langle\left(\boldsymbol{x}^{(n)}-\sum_{i=1}^{K}s_i^{(n)}\boldsymbol{\mu}_i\right)^{\mathsf{T}}\left(\boldsymbol{x}^{(n)}-\sum_{i=1}^{K}s_i^{(n)}\boldsymbol{\mu}_i\right)\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)} \Longrightarrow \left\langle(\mathcal{X}-\mathcal{S}\boldsymbol{\mu})^{\mathsf{T}}(\mathcal{X}-\mathcal{S}\boldsymbol{\mu})\right\rangle_{q(\boldsymbol{s})}$ relates to linear regression as the optimization of the parameters and hidden states aim to minimize the squared difference between $\mathcal{X}$ and $\mathcal{S}\boldsymbol{\mu}$. In linear regression, the same form is found, $(\boldsymbol{y}-W\boldsymbol{x})^{\mathsf{T}}(\boldsymbol{y}-W\boldsymbol{x})$, minimizing the squared difference between $\boldsymbol{y}$ and $W\boldsymbol{x}$ (the sum of squared error).

The solution found for $\boldsymbol{\mu}$ in the M step $\boldsymbol{\mu}=\left\langle\mathcal{S}\mathcal{S}^{\mathsf{T}}\right\rangle^{-1}_{q\left(\boldsymbol{s}^{(n)}\right)}\left\langle\mathcal{S}\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}\boldsymbol{x}$ solves for this expression in the same closed form as linear regression $W=(XX^{\mathsf{T}})^{-1}X\boldsymbol{y}$, utilizing the pseudoinverse. This analogy is sensible as $\boldsymbol{\mu}$ represents "weight" parameters on the expected value of the hidden states (S) and $\mathcal{X}$ are the given values to model.

The solution found for σ^2 in the M step $\left\langle(\mathcal{X}-\mathcal{S}\boldsymbol{\mu})^{\mathsf{T}}(\mathcal{X}-\mathcal{S}\boldsymbol{\mu})\right\rangle_{q(\boldsymbol{s})}$ again models the sum of squared error in a similar manner as $p(\boldsymbol{x}|\boldsymbol{s},\boldsymbol{\mu},\sigma^2)$ previously shown. Thus, the optimization of σ^2 can be related to the sum of squared error of $\mathcal{X},\mathcal{S},\boldsymbol{\mu}$; this analogy is sensible given that σ^2 represents variance.

c. The computational complexity of `MStep(X,ES,ESS)` for the case where `ESS` is $K\times K$ can be calculated in terms of N,K,D by observing each command.
Each of the first 5 lines (`N`,`D`,`K`,`ES`, and `ESS` initialization) have a complexity of $O(1)$ each because they require constant time.
To calculate the complexity of the computation of `mu`, `sigma`, and `pie`, each line can be divided into individual commands. Each command below is represented by a variable in the commands following to signify that the internal commands have already been executed.
For `mu`,

Code	Command	Complexity
inv(ESS)⟹a	inversion of $K \times K$	$O(K^3)$
ES'⟹b	transpose of $N \times K$ matrix	$O(1)$
a*b⟹c	multiplication of $K \times K$ matrix with $K \times N$ matrix	$O(K^2N)$
c*X⟹d	multiplication of $K \times N$ matrix with $N \times D$ matrix	$O(DKN)$
d'	transpose of $K \times D$ matrix	$O(1)$

The total computational complexity of calculating `mu` is
$O\left(K^3 + K^2N + DKN + 2\right) \approx O\left(K^3 + K^2N + DKN\right)$.

For `sigma`,

Code	Command	Complexity
X'⟹a	transpose of $N \times D$ matrix	$O(1)$
a*X⟹b	multiplication of $D \times N$ matrix with $N \times D$ matrix	$O(D^2N)$
trace(b)⟹c	trace of $D \times D$ matrix	$O(D)$
mu'⟹d	transpose of $D \times K$ matrix	$O(1)$
d*mu⟹e	multiplication of $K \times D$ matrix with $D \times K$ matrix	$O(DK^2)$
e*ESS⟹f	multiplication of $K \times K$ matrix with $K \times K$ matrix	$O(K^3)$
trace(f)⟹g	trace of $K \times K$ matrix	$O(K)$
ES'⟹h	transpose of $N \times K$ matrix	$O(1)$
h*X⟹i	multiplication of $K \times N$ matrix with $N \times D$ matrix	$O(DKN)$
i*mu⟹j	multiplication of $K \times D$ matrix with $D \times K$ matrix	$O(DK^2)$
trace(j)⟹k	trace of $K \times K$ matrix	$O(K)$
2*k⟹l	multiplication of scalars	$O(1)$
c+g⟹m	addition of scalars	$O(1)$
m+l⟹n	addition of scalars	$O(1)$
N*D⟹o	multiplication of scalars	$O(1)$
n/o⟹ p	division of scalars	$O(1)$
sqrt(p)	square-root of scalars	$O(1)$

The total computational complexity of calculating `sigma` is
$O\left(D^2N + D + 2DK^2 + K^3 + K + DKN + K + 9\right) \approx O\left(D^2N + 2DK^2 + K^3 + DKN\right)$.

For `pie`,

Code	Command	Complexity
mean(ES,1)	mean of $N \times K$	$O(NK)$

The total computational complexity of calculating `pie` is
$O(NK)$.

Summing the complexities of all of the lines in `MStep` yields

$O(5) + O\left(K^3 + K^2N + DKN\right) + O\left(D^2N + 2DK^2 + K^3 + DKN\right) + O(NK)$
$= O\left(5 + 2K^3 + K^2N + 2DKN + D^2N + 2DK^2 + K^3 + NK\right)$
$\approx O\left(2K^3 + K^2N + 2DKN + D^2N + 2DK^2 + K^3\right)$

d. From looking through `images.jpg`, there appear to be 8 distinct features with noise. Namely,

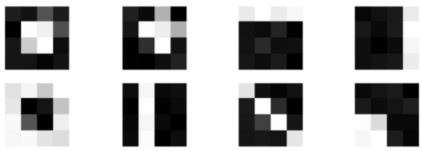
The images all contain some number of these features, which all appear to be independently distributed over all of the images.
Factor analysis would likely not be adequate because there does not seem to be a subset of (fewer) latent variables that would represent several of the features observed. Latent variables appear to be one-to-one with each feature, as multiple various features are often displayed simultaneously in an image and there does not appear to be a pattern with which features are displayed.
ICA would likely do well modelling this data because each feature appears to be independent and mixed, without Gaussian distributions. Each feature appears to be discrete and binary (either is in the image, or is not; presence of feature yields a particular pixel formation) and therefore, would fair well in ICA, without a Gaussian distribution of pixel values, and being identically and independently distributed.
A mixture of Gaussians would likely not be adequate because each feature is discrete (binary) and thus would not be modelled well with a continuous distribution. A mixture of Gaussians model would mix the latent variable distributions as Gaussian distributions, which would not accurately represent the model and result in displaying the features continuously rather than discretely.

e. `LearnBinFactors(X,K,iterations)` is implemented below. `lambda0` (`ES`) is randomly initialized, `ESS` is computed, and the parameters are initialized using the M step. The E and M steps are run for the given number of iterations and `F` is computed at each iteration; a check is included to determine whether `F` increases at each iteration.

```
# Calculates ESS using ES for m_step
def ComputeESS(ES):
    N=ES.shape[0]
    K=ES.shape[1]
    ESS=np.zeros((N,K,K))
    for n in range(N):
        tL=np.dot(ES[n].reshape(1,K).T,ES[n].reshape(1,K))
        ind=np.diag_indices(K)
        tL[ind]=ES[n]
        ESS[n]=tL
    return ESS

# Initializes ES randomly, computes ESS using ES,
# and initializes parameters using m_step
def InitParams(X,K):
    N=X.shape[0]
    D=X.shape[1]
    ES=np.random.rand(N,K)
    ESS=ComputeESS(ES)
    mu,sigma,pie=m_step(X,ES,ESS)
    return ES,ESS,mu,sigma,pie
```

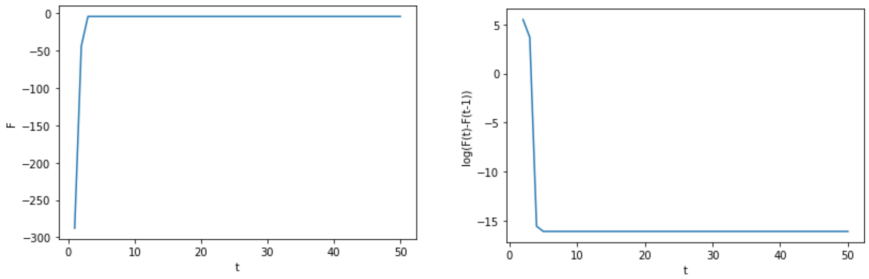
```
# Initializes parameters and runs EM for given number of iterations;
# calculates F for each EM iteration and includes a check that F increases
def LearnBinFactors(X,K,iterations):
    maxsteps=20
    ES,ESS,mu,sigma,pie=InitParams(X,K)
    F_array=[]
    prevF=FreeEnergy(X,mu,sigma,pie,ES)
    F_array.append(prevF)
    for i in range(iterations):
        ES,F=MeanField(X,mu,sigma,pie,ES,maxsteps)
        ESS=ComputeESS(ES)
        mu,sigma,pie=m_step(X,ES,ESS)
        F=FreeEnergy(X,mu,sigma,pie,ES)
        F_array.append(F)
        if F<=prevF:
            print("F decreasing by ",prevF-F)
        prevF=F
    return mu,sigma,pie,F_array
```

f. Running `LearnBinFactors` for 100 iterations and with a K of 8 yields good approximations of the features of the data. Namely, visualizing `mu`,

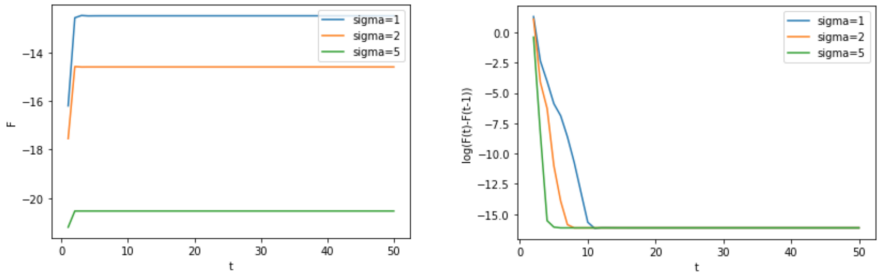


In this parameter setting and particular random initialization of `lambda0`, `mu` learns all of the features with a slight addition of noise. After `F` has converged, the algorithm could be improved by assigning binary values (0,1) to `mu` based on a threshold of 0.5; values less than 0.5 fixed to 0 and values greater than or equal to 0.5 fixed to 1. This would reduce the noise visible in the feature outputs and, in this particular run, would yield entirely accurate results for `mu`. In some runs, the features found appear to be more redundant: certain features are occasionally represented multiple times, while others are not found or are found in conjunction with other features (mixed features represented as one feature). The convergence to accurate predictions of `mu` is thus highly dependent on the random initialization of `lambda0`. Therefore, another way to improve the algorithm could be to introduce a weak prior (perhaps Beta) on `lambda`.
In order to achieve the results of `mu` above, K was set to 8 because of the a priori knowledge that the data is represented by 8 distinct features. `lambda0` was randomly initialized and used to compute `ES` and `ESS`. The parameters were then initialized using the M step, as the M step is a closed form function that computes new parameters based on the values of `ES` and `ESS`.

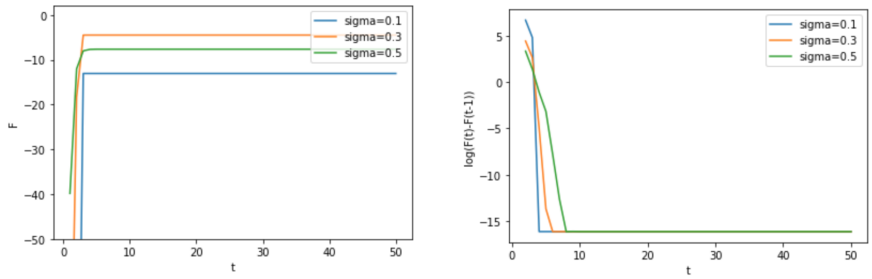
g. Running the variational approximation for the first data point ($q_1(\boldsymbol{s}^{(1)})$) using the parameters learned in the previous step yields the following plots for `F` and `log(F(t)-F(t-1))` as a function of iteration number `t` for `MeanField`.



As shown above, `F` converges relatively rapidly with the initialization of the previously-learned parameters and reaches a high value.
Running the variational approximation on 3 different values of `sigma` $(1, 2, 5)$ above the learned value of `sigma` (0.1773) yields



As `sigma` increases, the value of `F` decreases; `F` also converges more rapidly than smaller values of `sigma`. This demonstrates a precision-speed tradeoff. A higher value of `sigma` (increased noise) converges faster because a less-precise model will yield a converging `F`, being overtaken by noise. Alternatively, a lower value of `sigma` will converge slower as a more precise model is needed to reach a converging `F`, without noise obstructing the model. Intuitively, increased noise leads to less precise predictions, which require less time to arrive to.
Running the variational approximation on 3 different values of `sigma` $(0.1, 0.3, 0.5)$ around the learned value of `sigma` (0.1773) yields



A different effect is visible here: a low value `sigma` converges fastest, while yielding a lower `F`. An intermediate value for `sigma` yields the highest `F`, while also converging faster than a higher value of `sigma`, which takes the longest to converge. Thus the optimal `sigma` is an intermediate value between higher and lower values, as found by the learned model. The different effect of variance at lower values can be attributed to a very low value of `sigma` leading to overfitting, yielding a lower value of `F`; it converges faster due to the ease of fitting without significant noise present. Alternatively, a higher value of `sigma` does not overfit as much, but instead is more difficult to fit to, also resulting in a sub-optimal

`F` and longer convergence time. These observations are still consistent with those of the `sigma` greater than or equal to 1, as those cases are overtaken by noise in comparison to lower `sigma`. Intuitively, `sigma` below 1 yield higher values of `F`, as excessive noise makes the model more difficult to fit to and thus less precise.

Question 2.

a. The marginal $P(z_k)$ can be computed by integrating out u_k from the given probability distributions of z_k and u_k.

$P(z_k) = \int_{u_k} P(z_k|u_k,\alpha)P(u_k)\ du_k$

Substituting the given distributions for z_k and u_k,

$$= \int_{u_k} \left(\frac{1}{\sqrt{2\pi u_k^{-1}}} \exp\left\{ -\frac{(z_k-0)^2}{2u_k^{-1}} \right\} \right) \left(\frac{\left(\frac{\alpha}{2}\right)^{\frac{\alpha}{2}}}{\Gamma\left(\frac{\alpha}{2}\right)} u_k^{\frac{\alpha}{2}-1} \exp\left\{ -\frac{\alpha}{2}u_k \right\} \right) du_k$$

$$= \frac{\left(\frac{\alpha}{2}\right)^{\frac{\alpha}{2}}}{\sqrt{2\pi}\Gamma\left(\frac{\alpha}{2}\right)} \int_{u_k} \exp\left\{ -\frac{1}{2}z_k^2 u_k - \frac{\alpha}{2}u_k \right\} u_k^{\frac{\alpha}{2}-1} u_k^{\frac{1}{2}} du_k$$

$$= \frac{\left(\frac{\alpha}{2}\right)^{\frac{\alpha}{2}}}{\sqrt{2\pi}\Gamma\left(\frac{\alpha}{2}\right)} \int_{u_k} \exp\left\{ -u_k\left(\frac{1}{2}z_k^2 + \frac{\alpha}{2}\right) \right\} u_k^{\frac{\alpha}{2}-\frac{1}{2}} du_k$$

It can be observed that the integral takes the form of the Gamma distribution:

$\int x^{\alpha-1}e^{-\beta x} = \frac{\Gamma(\alpha)}{\beta^\alpha}$

with $x = u_k,\ \alpha = \frac{\alpha+1}{2}, \beta = \frac{z_k^2}{2} + \frac{\alpha}{2}$.

Thus, simplifying the integral,

$$= \frac{\left(\frac{\alpha}{2}\right)^{\frac{\alpha}{2}}}{\sqrt{2\pi}\Gamma\left(\frac{\alpha}{2}\right)} \times \frac{\Gamma\left(\frac{\alpha+1}{2}\right)}{\left(\frac{z_k^2}{2}+\frac{\alpha}{2}\right)^{\frac{\alpha+1}{2}}}$$

$$= \frac{\Gamma\left(\frac{\alpha+1}{2}\right)}{\sqrt{2\pi}\Gamma\left(\frac{\alpha}{2}\right)} \left(\frac{\alpha}{2}\right)^{\frac{\alpha}{2}} \left(\frac{z_k^2}{2}+\frac{\alpha}{2}\right)^{-\frac{\alpha+1}{2}}$$

$$= \frac{\Gamma\left(\frac{\alpha+1}{2}\right)}{\sqrt{2\pi}\Gamma\left(\frac{\alpha}{2}\right)} \left(\frac{2}{\alpha}\right)^{-\left(\frac{\alpha}{2}+\frac{1}{2}-\frac{1}{2}\right)} \left(\frac{z_k^2}{2}+\frac{\alpha}{2}\right)^{-\frac{\alpha+1}{2}}$$

$$= \frac{\Gamma\left(\frac{\alpha+1}{2}\right)}{\sqrt{2\pi}\Gamma\left(\frac{\alpha}{2}\right)} \left(\frac{2}{\alpha}\right)^{-\left(-\frac{1}{2}\right)} \left(\frac{2}{\alpha}\right)^{-\left(\frac{\alpha+1}{2}\right)} \left(\frac{z_k^2}{2}+\frac{\alpha}{2}\right)^{-\frac{\alpha+1}{2}}$$

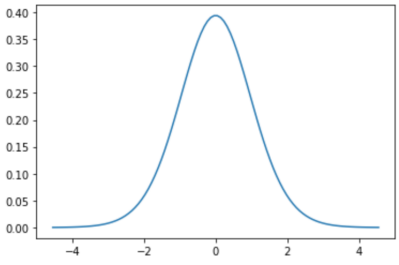
$$= \frac{\Gamma\left(\frac{\alpha+1}{2}\right)}{\sqrt{2\pi}\Gamma\left(\frac{\alpha}{2}\right)} \left(\frac{2}{\alpha}\right)^{\frac{1}{2}} \left(\frac{2}{\alpha}\left(\frac{z_k^2}{2}+\frac{\alpha}{2}\right)\right)^{-\frac{\alpha+1}{2}}$$

$$= \frac{\Gamma\left(\frac{\alpha+1}{2}\right)}{\sqrt{2\pi}\Gamma\left(\frac{\alpha}{2}\right)} \left(\frac{2}{\alpha}\right)^{\frac{1}{2}} \left(\frac{z_k^2}{\alpha}+1\right)^{-\frac{\alpha+1}{2}}$$

$$= \frac{\Gamma\left(\frac{\alpha+1}{2}\right)}{\sqrt{2\pi}\sqrt{\frac{\alpha}{2}}\Gamma\left(\frac{\alpha}{2}\right)} \left(1+\frac{z_k^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}$$

$$= \frac{\Gamma\left(\frac{\alpha+1}{2}\right)}{\sqrt{\alpha\pi}\Gamma\left(\frac{\alpha}{2}\right)} \left(1+\frac{z_k^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}$$

The resulting marginal $P(z_k)$ is a Student's t-distribution, displayed below



ICA cannot resolve independent components which have Gaussian distributions. The Student's t-distribution is somewhat similar to the Gaussian distribution in shape, but since it is non-Gaussian, it can be modeled using ICA.

b. Because $\mathcal{U}$ can be integrated out as shown in (a), the model can be represented in terms of $\mathcal{Z}$. Thus, a minimal Variational Bayesian factorization of the joint posterior on the latent variables and parameters can written as the following
$Q(\mathcal{Z}, A, \Psi) \approx Q_{\mathcal{Z}}(\mathcal{Z}) Q_{A,\Psi}(A, \Psi)$

c. With the conditions that the joint probability over variables is in the exponential family and the prior over parameters is conjugate to this joint probability, $Q_{A,\Psi}(A, \Psi)$ is also conjugate. Thus,

$$Q_{A,\Psi}(A,\Psi) \propto P(A,\Psi) \exp\left\langle \sum_{n=1}^{N} \log P\left(\boldsymbol{z}^{(n)}, \boldsymbol{x}^{(n)} | A, \Psi\right) \right\rangle_{Q_{\mathcal{Z}}}$$

$$= P(A,\Psi) \exp\left\langle \sum_{n=1}^{N} \left(\log P\left(\boldsymbol{x}^{(n)} | \boldsymbol{z}^{(n)}, A, \Psi\right) + \sum_{k=1}^{K} \log P\left(z_k^{(n)}\right) \right) \right\rangle_{Q_{\mathcal{Z}}}$$

Substituting the given distribution of $\boldsymbol{x}$ and solving for $\langle P\left(\boldsymbol{x}^{(n)} | \boldsymbol{z}^{(n)}, A, \Psi\right) \rangle_{Q_{\mathcal{Z}}}$,

$$\langle P\left(\boldsymbol{x}^{(n)} | \boldsymbol{z}^{(n)}, A, \Psi\right) \rangle_{Q_{\mathcal{Z}}} = \left\langle \frac{1}{\sqrt{2\pi\Psi}} \exp\left\{ -\frac{1}{2} \left(\boldsymbol{x}^{(n)} - A\boldsymbol{z}^{(n)}\right)^{\mathsf{T}} \Psi^{-1} \left(\boldsymbol{x}^{(n)} - A\boldsymbol{z}^{(n)}\right) \right\} \right\rangle_{Q_{\mathcal{Z}}}$$

$$= \left\langle \frac{1}{\sqrt{2\pi\Psi}} \exp\left\{ -\frac{1}{2} \boldsymbol{x}^{(n)\mathsf{T}} \Psi^{-1} \boldsymbol{x}^{(n)} + \frac{1}{2} \boldsymbol{x}^{(n)\mathsf{T}} \Psi^{-1} A \boldsymbol{z}^{(n)} + \frac{1}{2} \boldsymbol{z}^{(n)\mathsf{T}} A^{\mathsf{T}} \Psi^{-1} \boldsymbol{x}^{(n)} - \frac{1}{2} \boldsymbol{z}^{(n)\mathsf{T}} A^{\mathsf{T}} \Psi^{-1} A \boldsymbol{z}^{(n)} \right\} \right\rangle_{Q_{\mathcal{Z}}}$$

$$= \left\langle \frac{1}{\sqrt{2\pi\Psi}} \exp\left\{ -\frac{1}{2} \left(\boldsymbol{x}^{(n)\mathsf{T}} \Psi^{-1} \boldsymbol{x}^{(n)} - 2\boldsymbol{z}^{(n)\mathsf{T}} A^{\mathsf{T}} \Psi^{-1} \boldsymbol{x}^{(n)} + \boldsymbol{z}^{(n)\mathsf{T}} A^{\mathsf{T}} \Psi^{-1} A \boldsymbol{z}^{(n)} \right) \right\} \right\rangle_{Q_{\mathcal{Z}}}$$

$$= \left\langle \frac{1}{\sqrt{2\pi\Psi}} \exp\left\{ -\frac{1}{2} \left(\mathrm{Tr}\left(\boldsymbol{x}^{(n)\mathsf{T}} \Psi^{-1} \boldsymbol{x}^{(n)} \right) - 2\mathrm{Tr}\left(\boldsymbol{z}^{(n)\mathsf{T}} A^{\mathsf{T}} \Psi^{-1} \boldsymbol{x}^{(n)} \right) + \mathrm{Tr}\left(\boldsymbol{z}^{(n)\mathsf{T}} A^{\mathsf{T}} \Psi^{-1} A \boldsymbol{z}^{(n)} \right) \right) \right\} \right\rangle_{Q_{\mathcal{Z}}}$$

$$= \frac{1}{\sqrt{2\pi\Psi}} \exp\left\{ -\frac{1}{2} \left(\mathrm{Tr}\left(\Psi^{-1} \boldsymbol{x}^{(n)} \boldsymbol{x}^{(n)\mathsf{T}} \right) - 2\mathrm{Tr}\left(A^{\mathsf{T}} \Psi^{-1} \boldsymbol{x}^{(n)} \left\langle \boldsymbol{z}^{(n)} \right\rangle_{Q_{\mathcal{Z}}}^{\mathsf{T}} \right) + \mathrm{Tr}\left(A^{\mathsf{T}} \Psi^{-1} A \left\langle \boldsymbol{z}(n) \boldsymbol{z}^{(n)\mathsf{T}} \right\rangle_{Q_{\mathcal{Z}}} \right) \right) \right\}$$

$$= \frac{1}{\sqrt{2\pi\Psi}} \exp\left\{ \begin{bmatrix} vec\left(-\frac{1}{2}\Psi^{-1}\right) & vec\left(A^{\mathsf{T}}\Psi^{-1}\right) & vec\left(-\frac{1}{2}A^{\mathsf{T}}\Psi^{-1}A\right) \end{bmatrix} \begin{bmatrix} vec\left(\boldsymbol{x}^{(n)}\ \boldsymbol{x}^{(n)\mathsf{T}}\right) \\ vec\left(\boldsymbol{x}^{(n)}\ \left\langle \boldsymbol{z}^{(n)} \right\rangle_{Q_{\mathcal{Z}}}^{\mathsf{T}}\right) \\ vec\left(\left\langle \boldsymbol{z}^{(n)} \boldsymbol{z}^{(n)\mathsf{T}} \right\rangle_{Q_{\mathcal{Z}}}\right) \end{bmatrix} \right\}$$

Thus, the natural parameters $\phi(A, \Psi)$ can be written as

$$\phi(A,\Psi) = \begin{bmatrix} vec\left(-\frac{1}{2}\Psi^{-1}\right) \\ vec\left(A^{\mathsf{T}}\Psi^{-1}\right) \\ vec\left(-\frac{1}{2}A^{\mathsf{T}}\Psi^{-1}A\right) \end{bmatrix}$$

Additionally, the sufficient statistics $\langle \mathsf{T}\left(\boldsymbol{z}^{(n)}, \boldsymbol{x}^{(n)}\right) \rangle_{Q_{\mathcal{Z}}}$ can be written as

$$\langle \mathsf{T}\left(\boldsymbol{z}^{(n)}, \boldsymbol{x}^{(n)}\right) \rangle_{Q_{\mathcal{Z}}} = \begin{bmatrix} vec\left(\boldsymbol{x}^{(n)} \boldsymbol{x}^{(n)\mathsf{T}}\right) \\ vec\left(\boldsymbol{x}^{(n)} \left\langle \boldsymbol{z}^{(n)} \right\rangle_{Q_{\mathcal{Z}}}^{\mathsf{T}}\right) \\ vec\left(\left\langle \boldsymbol{z}^{(n)} \boldsymbol{z}^{(n)\mathsf{T}} \right\rangle_{Q_{\mathcal{Z}}}\right) \end{bmatrix}$$

Written in the terms above yields
$P\left(\boldsymbol{x}^{(n)}|\boldsymbol{z}^{(n)},A,\Psi\right)\rangle_{Q_{\mathcal{Z}}}=\frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{\Psi}}\exp\left\{\phi(A,\Psi)^{\mathsf{T}}\langle\mathsf{T}\left(\boldsymbol{z}^{(n)},\boldsymbol{x}^{(n)}\right)\rangle_{Q_{\mathcal{Z}}}\right\}$

Additionally, the marginal probability of $z_k^{(n)}$ can be written as
$$\langle P\left(z_k^{(n)}\right)\rangle_{Q_{\mathcal{Z}}}=\frac{\Gamma\left(\frac{\alpha+1}{2}\right)}{\sqrt{\alpha\pi}\Gamma\left(\frac{\alpha}{2}\right)}\left(1+\frac{\langle {z_k^{(n)}}^2\rangle_{Q_{\mathcal{Z}}}}{\alpha}\right)^{-\frac{\alpha+1}{2}}$$

A,Ψ are assumed to have conjugate priors to the probability of $\boldsymbol{x}$ above with hyperparameters $\upsilon,\boldsymbol{\tau}$ and thus can be written as
$P(A,\Psi|\upsilon,\boldsymbol{\tau})=\left(\frac{1}{\sqrt{2\pi}}\right)^{\upsilon}\left(\frac{1}{\sqrt{\Psi}}\right)^{\upsilon}\exp\left\{\phi(A,\Psi)^{\mathsf{T}}\boldsymbol{\tau}\right\}$

Substituting into the original expression,

$$\begin{aligned}
Q_{A,\Psi}(A,\Psi)&\propto\left(\frac{1}{\sqrt{2\pi}}\right)^{\upsilon}\left(\frac{1}{\sqrt{\Psi}}\right)^{\upsilon}\exp\left\{\phi(A,\Psi)^{\mathsf{T}}\boldsymbol{\tau}\right\}\exp<\sum_{n=1}^{N}\log\left(\frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{\Psi}}\exp\left\{\phi(A,\Psi)^{\mathsf{T}}\mathsf{T}\left(\boldsymbol{z}^{(n)},\boldsymbol{x}^{(n)}\right)\right\}\right)\\
&\quad+\sum_{k=1}^{K}\log\frac{\Gamma\left(\frac{\alpha+1}{2}\right)}{\sqrt{\alpha\pi}\Gamma\left(\frac{\alpha}{2}\right)}\left(1+\frac{{z_k^{(n)}}^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}>_{Q_{\mathcal{Z}}}\\
&=\left(\frac{1}{\sqrt{2\pi}}\right)^{\upsilon}\left(\frac{1}{\sqrt{\Psi}}\right)^{\upsilon}\exp\left\{\phi(A,\Psi)^{\mathsf{T}}\boldsymbol{\tau}\right\}\exp\left\{\sum_{k=1}^{K}\log\frac{\Gamma\left(\frac{\alpha+1}{2}\right)}{\sqrt{\alpha\pi}\Gamma\left(\frac{\alpha}{2}\right)}\left(1+\frac{\left\langle {z_k^{(n)}}^2\right\rangle_{Q_{\mathcal{Z}}}}{\alpha}\right)^{-\frac{\alpha+1}{2}}\right\}\\
&\quad\exp\left\{\sum_{n=1}^{N}\log\left(\frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{\Psi}}\exp\left\{\phi(A,\Psi)^{\mathsf{T}}\langle\mathsf{T}\left(\boldsymbol{z}^{(n)},\boldsymbol{x}^{(n)}\right)\rangle_{Q_{\mathcal{Z}}}\right\}\right)\right\}\\
&\propto\left(\frac{1}{\sqrt{2\pi}}\right)^{\upsilon}\left(\frac{1}{\sqrt{\Psi}}\right)^{\upsilon}\exp\left\{\phi(A,\Psi)^{\mathsf{T}}\boldsymbol{\tau}\right\}\left(\frac{1}{\sqrt{2\pi}}\right)^{N}\left(\frac{1}{\sqrt{\Psi}}\right)^{N}\exp\left\{\phi(A,\Psi)^{\mathsf{T}}\left(\sum_{n=1}^{N}\langle\mathsf{T}\left(\boldsymbol{z}^{(n)},\boldsymbol{x}^{(n)}\right)\rangle_{Q_{\mathcal{Z}}}\right)\right\}\\
&\propto\left(\frac{1}{\sqrt{2\pi}}\right)^{\upsilon+N}\left(\frac{1}{\sqrt{\Psi}}\right)^{\upsilon+N}\exp\left\{\phi(A,\Psi)^{\mathsf{T}}\left(\boldsymbol{\tau}+\sum_{n=1}^{N}\langle\mathsf{T}\left(\boldsymbol{z}^{(n)},\boldsymbol{x}^{(n)}\right)\rangle_{Q_{\mathcal{Z}}}\right)\right\}\\
&=\left(\frac{1}{\sqrt{2\pi}}\right)^{\tilde{\upsilon}}\left(\frac{1}{\sqrt{\Psi}}\right)^{\tilde{\upsilon}}\exp\left\{\phi(A,\Psi)^{\mathsf{T}}\tilde{\boldsymbol{\tau}}\right\}
\end{aligned}$$

where $\tilde{\upsilon}=\upsilon+N$ and $\tilde{\boldsymbol{\tau}}=\boldsymbol{\tau}+\sum_{n=1}^{N}\langle\mathsf{T}\left(\boldsymbol{z}^{(n)},\boldsymbol{x}^{(n)}\right)\rangle_{Q_{\mathcal{Z}}}$

d. i. To learn the shape of the marginal distributions $P(z_k)$, α can be optimized in a hyper-M step. The model itself is specified by α and thus the Variational Bayesian free energy can be written
$$\mathcal{F}(Q_{\mathcal{Z}},Q_{A,\Psi},\boldsymbol{\alpha})=\int\int d\mathcal{Z}dA,\Psi Q_{\mathcal{Z}}(\mathcal{Z})Q_{A,\Psi}(A,\Psi)\log\frac{P(\mathcal{X},\mathcal{Z},A,\Psi|\boldsymbol{\alpha})}{Q_{\mathcal{Z}}(\mathcal{Z})Q_{A,\Psi}(A,\Psi)}\leq P(\mathcal{X}|\alpha)$$

A hyper-M step maximizes the current bound with respect to α and thus,
$$\begin{aligned}
\boldsymbol{\alpha}&\Longleftarrow\arg\max_{\alpha}\int\int d\mathcal{Z}dA,\Psi\ Q_{\mathcal{Z}}(\mathcal{Z})Q_{A,\Psi}(A,\Psi)P(\mathcal{X},\mathcal{Z},A,\Psi|\boldsymbol{\alpha})\\
&\propto\arg\max_{\alpha}\langle\log P(\mathcal{Z}|\boldsymbol{\alpha})\rangle_{Q_{\mathcal{Z}}}
\end{aligned}$$

Solving for a particular α,
$$\begin{aligned}
\alpha&=\arg\max_{\alpha}\left\langle\log\frac{\Gamma\left(\frac{\alpha+1}{2}\right)}{\sqrt{\alpha\pi}\Gamma\left(\frac{\alpha}{2}\right)}\left(1+\frac{z_k^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}\right\rangle_{Q_{\mathcal{Z}}}\\
&=\arg\max_{\alpha}\left\langle\log\Gamma\left(\frac{\alpha+1}{2}\right)-\log\sqrt{\alpha\pi}\Gamma\left(\frac{\alpha}{2}\right)-\frac{\alpha+1}{2}\log\left(1+\frac{z_k^2}{\alpha}\right)\right\rangle_{Q_{\mathcal{Z}}}
\end{aligned}$$

Taking the derivative with respect to α,

$$\left(\log\Gamma\left(\frac{\alpha+1}{2}\right)-\frac{1}{2}\log\alpha-\frac{1}{2}\log\pi-\log\Gamma\left(\frac{\alpha}{2}\right)-\frac{\alpha+1}{2}\log\left(1+\frac{\langle z_k^2\rangle}{\alpha}\right)\right)d\alpha$$
$$=\frac{\Gamma\left(\frac{\alpha+1}{2}\right)\psi_0\left(\frac{\alpha+1}{2}\right)}{2\Gamma\left(\frac{\alpha+1}{2}\right)}-\frac{1}{2\alpha}-\frac{\Gamma\left(\frac{\alpha}{2}\right)\psi_0\left(\frac{\alpha}{2}\right)}{2\Gamma\left(\frac{\alpha}{2}\right)}+\frac{(\alpha+1)\langle z_k^2\rangle}{2\left(1+\frac{\langle z_k^2\rangle}{\alpha}\right)\alpha^2}-\frac{1}{2}\log\left(1+\frac{\langle z_k^2\rangle}{\alpha}\right)$$
$$=\frac{\psi_0\left(\frac{\alpha+1}{2}\right)}{2}-\frac{1}{2\alpha}-\frac{\psi_0\left(\frac{\alpha}{2}\right)}{2}+\frac{(\alpha+1)\langle z_k^2\rangle}{2\left(1+\frac{\langle z_k^2\rangle}{\alpha}\right)\alpha^2}-\frac{1}{2}\log\left(1+\frac{\langle z_k^2\rangle}{\alpha}\right)$$
$$=\frac{1}{2}\left(\psi_0\left(\frac{\alpha+1}{2}\right)-\alpha^{-1}-\psi_0\left(\frac{\alpha}{2}\right)+\frac{(\alpha+1)\langle z_k^2\rangle}{\alpha^2\left(1+\frac{\langle z_k^2\rangle}{\alpha}\right)}-\log\left(1+\frac{\langle z_k^2\rangle}{\alpha}\right)\right)=0$$
$$\psi_0\left(\frac{\alpha+1}{2}\right)-\alpha^{-1}-\psi_0\left(\frac{\alpha}{2}\right)+\frac{(\alpha+1)\langle z_k^2\rangle}{\alpha^2\left(1+\frac{\langle z_k^2\rangle}{\alpha}\right)}-\log\left(1+\frac{\langle z_k^2\rangle}{\alpha}\right)=0$$

The maximum α should satisfy the function above and can be found by using a technique such as a root calculator or gradient step.

ii. To derive the hyperparameter optimization rule for learning the number of features, the model parameterization is modified. In particular, A is parameterized with a column-wise prior over all D,

$A_{:k}\sim\mathcal{N}\left(0,\beta_k^{-1}I\right)$

Thus, to solve for the optimal hyperparameter $\boldsymbol{\beta}$ of A in order to determine the number of relevant features K,

$\boldsymbol{\beta}=\arg\max_{\beta}\langle\log P(A|\boldsymbol{\beta})\rangle_{Q_A}$

Solving for a particular β_k

$\beta_k=\arg\max_{\beta_k}\langle\log P(A_{:k}|\beta_k)\rangle_{Q_A}$

Taking the derivative with respect to β_k,

$$\left\langle\log\left(\frac{D}{\sqrt{2\pi\beta_k^{-1}}}\exp\left\{-\frac{1}{2\beta_k^{-1}}A_{:k}^{\mathsf{T}}A_{:k}\right\}\right)\right\rangle_{Q_A}d\beta_k$$
$$=\left(\log\left(\frac{D}{\sqrt{2\pi\beta_k^{-1}}}\right)-\frac{1}{2\beta_k^{-1}}\left\langle A_{:k}^{\mathsf{T}}A_{:k}\right\rangle_{Q_A}\right)d\beta_k$$
$$=\left(-\frac{D}{2}\log(2\pi)+\frac{D}{2}\log(\beta_k)-\frac{1}{2}\beta_k\left\langle A_{:k}^{\mathsf{T}}A_{:k}\right\rangle_{Q_A}\right)d\beta_k$$
$$=\frac{D}{2\beta_k}-\frac{1}{2}\left\langle A_{:k}^{\mathsf{T}}A_{:k}\right\rangle$$
$$=\frac{D}{2\beta_k}-\frac{1}{2}\left\langle A_{:k}^{\mathsf{T}}A_{:k}\right\rangle=0$$
$$\frac{D}{2\beta_k}=\frac{1}{2}\left\langle A_{:k}^{\mathsf{T}}A_{:k}\right\rangle$$
$$\frac{D}{\beta_k}=\left\langle A_{:k}^{\mathsf{T}}A_{:k}\right\rangle$$
$$\beta_k=\frac{D}{\left\langle A_{:k}^{\mathsf{T}}A_{:k}\right\rangle}$$

Question 3.

a. In order to automatically determine K, the number of hidden binary variables in the model, μ is parameterized with a column-wise prior
$\boldsymbol{\mu}_{:i} \sim \mathcal{N}\left(0, \alpha_i^{-1} I\right)$
Thus, the free energy is now taken with expectations over both $Q_{\mathcal{S}}$ and Q_{μ}
$\mathcal{F}(q, \boldsymbol{\theta}) = \langle \log p(\mathcal{S}, \mathcal{X}|\boldsymbol{\theta})\rangle_{q(\mathcal{S})q(\mu)} - \langle \log q(\mathcal{S})\rangle_{q(\mathcal{S})}$

$$= \sum_{n=1}^{N} \left\langle \log p\left(\boldsymbol{x}^{(n)}|\boldsymbol{s^{(n)}}, \boldsymbol{\mu}, \sigma^2\right) + \log P\left(\boldsymbol{s^{(n)}}|\boldsymbol{\pi}\right) + \log P(\boldsymbol{\mu}|\boldsymbol{\alpha})\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)q(\mu)} - \sum_{n=1}^{N} \left\langle \log q\left(\boldsymbol{s}^{(n)}\right)\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}$$

$$= \sum_{n=1}^{N} [-\frac{D}{2}\log 2\pi - D\log\sigma - \frac{1}{2\sigma^2}\left\langle \left(\boldsymbol{x}^{(n)} - \sum_{i=1}^{K} s_i^{(n)}\boldsymbol{\mu}_i\right)^{\mathsf{T}} \left(\boldsymbol{x}^{(n)} - \sum_{i=1}^{K} s_i^{(n)}\boldsymbol{\mu}_i\right)\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)q(\mu)}$$
$$+ \sum_{i=1}^{K}\left(\left\langle s_i^{(n)}\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)} \log \boldsymbol{\pi}_i + \left(1 - \left\langle s_i^{(n)}\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}\right)\log(1-\boldsymbol{\pi}_i)\right)$$
$$+ \sum_{i=1}^{K}\left(-\frac{D}{2}\log 2\pi\alpha_i^{-1} \; - \frac{1}{2}\langle\boldsymbol{\mu}_i\rangle_{q(\mu)}^{\mathsf{T}}\alpha_i\langle\boldsymbol{\mu}_i\rangle_{q(\mu)}\right)$$
$$- \sum_{i=1}^{K}\left(\left\langle s_i^{(n)}\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)} \log \lambda_{in} - \left(1 - \left\langle s_i^{(n)}\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}\right)\log(1-\lambda_{in})\right) \;\;]$$

$$= \sum_{n=1}^{N} [-\frac{D}{2}\log 2\pi - D\log\sigma - \frac{1}{2\sigma^2}\left\langle \left(\boldsymbol{x}^{(n)} - \sum_{i=1}^{K} s_i^{(n)}\boldsymbol{\mu}_i\right)^{\mathsf{T}} \left(\boldsymbol{x}^{(n)} - \sum_{i=1}^{K} s_i^{(n)}\boldsymbol{\mu}_i\right)\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)q(\mu)}$$
$$+ \sum_{k=1}^{K}\left(\left\langle s_i^{(n)}\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)} \log \frac{\boldsymbol{\pi}_i}{\lambda_{in}} + \left(1 - \left\langle s_i^{(n)}\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}\right)\log\frac{(1-\boldsymbol{\pi}_i)}{(1-\lambda_{in})}\right)$$
$$+ \sum_{i=1}^{K}\left(-\frac{D}{2}\log 2\pi\alpha_i^{-1} \; - \frac{1}{2}\langle\boldsymbol{\mu}_i\rangle_{q(\mu)}^{\mathsf{T}}\alpha_i\langle\boldsymbol{\mu}_i\rangle_{q(\mu)}\right)$$

Solving for
$$\sum_{n=1}^{N} -\frac{1}{2\sigma^2}\left\langle \left(\boldsymbol{x}^{(n)} - \sum_{i=1}^{K} s_i^{(n)}\boldsymbol{\mu}_i\right)^{\mathsf{T}} \left(\boldsymbol{x}^{(n)} - \sum_{i=1}^{K} s_i^{(n)}\boldsymbol{\mu}_i\right)\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)q(\mu)}$$
which can be rewritten in the d dimension,
$$= -\frac{1}{2\sigma^2}\left\langle \sum_{d=1}^{D}(\boldsymbol{x}_d - S\boldsymbol{\mu}_d)^{\mathsf{T}}(\boldsymbol{x}_d - S\boldsymbol{\mu}_d)\right\rangle_{q(\mathcal{S})q(\mu)}$$
For square forms where x $\mathcal{N}(m, \Sigma)$,
$\mathbb{E}\left[(\boldsymbol{x}-\boldsymbol{m}')^{\mathsf{T}}\boldsymbol{A}(\boldsymbol{x}-\boldsymbol{m}')\right] = (\boldsymbol{m}-\boldsymbol{m}')^{\mathsf{T}}\boldsymbol{A}(\boldsymbol{m}-\boldsymbol{m}') + \mathrm{Tr}(\boldsymbol{A}\boldsymbol{\Sigma})$

Thus, taking the expectation with respect to μ,
$$= -\frac{1}{2\sigma^2}\left\langle \sum_{d=1}^{D}(\boldsymbol{x}_d - S\langle\boldsymbol{\mu}_d\rangle_{q(\mu)})^{\mathsf{T}}(\boldsymbol{x}_d - S\langle\boldsymbol{\mu}_d\rangle_{q(\mu)}) + \mathrm{Tr}\left(S^{\mathsf{T}}S\sum_{d=1}^{D}\Sigma_{\mu_d}\right)\right\rangle_{q(\mathcal{S})}$$
$$= -\frac{1}{2\sigma^2}\left\langle \sum_{n=1}^{N}\left(\boldsymbol{x}^{(n)} - \langle\boldsymbol{\mu}\rangle_{q(\mu)}\boldsymbol{s}^{(n)}\right)^{\mathsf{T}}\left(\boldsymbol{x}^{(n)} - \langle\boldsymbol{\mu}_d\rangle_{q(\mu)}\boldsymbol{s}^{(n)}\right) + \mathrm{Tr}\left(S^{\mathsf{T}}S\sum_{d=1}^{D}\Sigma_{\mu_d}\right)\right\rangle_{q(\mathcal{S})}$$

Taking the expectation with respect to $\mathcal{S}$,
$$= \sum_{n=1}^{N} -\frac{1}{2\sigma^2}\left(\boldsymbol{x}^{(n)} - \langle\boldsymbol{\mu}\rangle_{q(\mu)}\langle\boldsymbol{s}^{(n)}\rangle_{q(\mathcal{S})}\right)^{\mathsf{T}}\left(\boldsymbol{x}^{(n)} - \langle\boldsymbol{\mu}\rangle_{q(\mu)}\langle\boldsymbol{s}^{(n)}\rangle_{q(\mathcal{S})}\right)$$

$$-\frac{1}{2\sigma^2}\mathrm{Tr}\left(\langle\boldsymbol{\mu}\rangle_{q(\mu)}^{\mathsf{T}}\langle\boldsymbol{\mu}\rangle_{q(\mu)}\sum_{n=1}^{N}\Sigma_{\boldsymbol{s}_n}\right)-\frac{1}{2\sigma^2}\mathrm{Tr}\left(\langle S^{\mathsf{T}}S\rangle\sum_{d=1}^{D}\Sigma_{\mu_d}\right)$$

Substituting back into $\mathcal{F}$,

$$\mathcal{F}(q,\boldsymbol{\theta})=\sum_{n=1}^{N}[-\frac{D}{2}\log 2\pi-D\log\sigma-\frac{1}{2\sigma^2}\left(\boldsymbol{x}^{(n)}-\langle\boldsymbol{\mu}\rangle_{q(\mu)}\langle\boldsymbol{s}^{(n)}\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}\right)^{\mathsf{T}}\left(\boldsymbol{x}^{(n)}-\langle\boldsymbol{\mu}\rangle_{q(\mu)}\langle\boldsymbol{s}^{(n)}\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}\right)$$
$$+\sum_{k=1}^{K}\left(\left\langle s_i^{(n)}\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}\log\frac{\boldsymbol{\pi}_i}{\lambda_{in}}+\left(1-\left\langle s_i^{(n)}\right\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}\right)\log\frac{(1-\boldsymbol{\pi}_i)}{(1-\lambda_{in})}\right)$$
$$+\sum_{i=1}^{K}\left(-\frac{D}{2}\log 2\pi\alpha_i^{-1}\ -\frac{1}{2}\langle\boldsymbol{\mu}_i\rangle_{q(\mu)}^{\mathsf{T}}\alpha_i\langle\boldsymbol{\mu}_i\rangle_{q(\mu)}\right.$$
$$\left.-\frac{1}{2\sigma^2}\mathrm{Tr}\left(\langle\boldsymbol{\mu}\rangle_{q(\mu)}^{\mathsf{T}}\langle\boldsymbol{\mu}\rangle_{q(\mu)}\sum_{n=1}^{N}\Sigma_{\boldsymbol{s}_n}\right)-\frac{1}{2\sigma^2}\mathrm{Tr}\left(\langle S^{\mathsf{T}}S\rangle\sum_{d=1}^{D}\Sigma_{\mu_d}\right)\right.$$

The covariance of $\mathcal{S}$ is

$$\sum_{n=1}^{N}\Sigma_{\boldsymbol{s}_n}\ =\langle S^{\mathsf{T}}S\rangle_{Q_{\mathcal{S}}}-\langle S\rangle_{Q_{\mathcal{S}}}^{\mathsf{T}}\langle S\rangle_{Q_{\mathcal{S}}}$$

The posterior on $\boldsymbol{\mu}$ is normal. Thus,

$\Sigma_{\mu_d}=\left(\sigma^{-2}\langle S^{\mathsf{T}}S\rangle_{Q_{\mathcal{S}}}+diag(\boldsymbol{\alpha})^{-1}\right)^{-1}$

$\overline{\boldsymbol{\mu}}_d=\sigma^{-2}\Sigma_{\mu_d}\langle S\rangle_{Q_{\mathcal{S}}}^{\mathsf{T}}\boldsymbol{x}_d$

$Q(\mu)\sim\mathcal{N}(\overline{\boldsymbol{\mu}},\Sigma_\mu)$

Solving for α_i, $\frac{\partial\mathcal{F}}{\partial\alpha_i}=\left(-\frac{D}{2}\log 2\pi\alpha_i^{-1}-\frac{1}{2}\boldsymbol{\mu}_i\rangle_{q(\mu)}^{\mathsf{T}}\alpha_i\langle\boldsymbol{\mu}_i\rangle_{q(\mu)}\right)d\alpha_i$

$$=\left(-\frac{D}{2}\log 2\pi+\frac{D}{2}\log\alpha_i-\frac{1}{2}\langle\boldsymbol{\mu}_i\rangle_{q(\mu)}^{\mathsf{T}}\alpha_i\langle\boldsymbol{\mu}_i\rangle_{q(\mu)}\right)d\alpha_i$$
$$=\frac{D}{2\alpha_i}-\frac{1}{2}\langle\boldsymbol{\mu}_i\rangle_{q(\mu)}^{\mathsf{T}}\langle\boldsymbol{\mu}_i\rangle_{q(\mu)}=0$$
$$\frac{D}{2\alpha_i}=\frac{1}{2}\langle\boldsymbol{\mu}_i\rangle_{q(\mu)}^{\mathsf{T}}\langle\boldsymbol{\mu}_i\rangle_{q(\mu)}$$
$$\alpha_i=\frac{D}{\langle\boldsymbol{\mu}_i\rangle_{q(\mu)}^{\mathsf{T}}\langle\boldsymbol{\mu}_i\rangle_{q(\mu)}}$$

Solving for σ yields

$$\sigma^2=\frac{1}{ND}[\sum_{n=1}^{N}\left(\boldsymbol{x}^{(n)}-\langle\boldsymbol{\mu}\rangle_{q(\mu)}\langle\boldsymbol{s}^{(n)}\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}\right)^{\mathsf{T}}\left(\boldsymbol{x}^{(n)}-\langle\boldsymbol{\mu}\rangle_{q(\mu)}\langle\boldsymbol{s}^{(n)}\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}\right)$$
$$+\mathrm{Tr}\left(\langle\boldsymbol{\mu}\rangle_{q(\mu)}^{\mathsf{T}}\langle\boldsymbol{\mu}\rangle_{q(\mu)}\sum_{n=1}^{N}\Sigma_{\boldsymbol{s}_n}\right)+\mathrm{Tr}\left(\langle S^{\mathsf{T}}S\rangle\sum_{d=1}^{D}\Sigma_{\mu_d}\right)]$$

The updates for the parameter π remain the same as the additional terms in $\mathcal{F}$ become 0 when the derivative with respect to π is taken.

Finally, solving for $\langle s_i^{(n)}\rangle_{q\left(\boldsymbol{s}^{(n)}\right)}$,

$$q\left(\boldsymbol{s}_i^{(n)}\right)\propto\exp\left\langle\log p\left(\boldsymbol{x}^{(n)}|\boldsymbol{s}^{(\boldsymbol{n})},\boldsymbol{\mu},\sigma^2\right)+\log P\left(\boldsymbol{s}^{(\boldsymbol{n})}|\boldsymbol{\pi}\right)+\log P(\boldsymbol{\mu}|\boldsymbol{\alpha})\right\rangle_{q\left(\boldsymbol{s}_{\neg i}^{(n)}\right)q(\boldsymbol{\theta})}$$
$$\propto\exp\langle-\frac{D}{2}\log 2\pi-D\log\sigma-\frac{1}{2\sigma^2}\left(\boldsymbol{x}^{(n)}-\sum_{i=1}^{K}s_i^{(n)}\boldsymbol{\mu}_i\right)^{\mathsf{T}}\left(\boldsymbol{x}^{(n)}-\sum_{i=1}^{K}s_i^{(n)}\boldsymbol{\mu}_i\right)$$
$$+\sum_{i=1}^{K}\left(s_i^{(n)}\log\boldsymbol{\pi}_i+\left(1-s_i^{(n)}\right)\log(1-\boldsymbol{\pi}_i)\right)$$

$$+\sum_{i=1}^{K}\left(-\frac{D}{2}\log 2\pi\alpha_i^{-1} \; -\frac{1}{2\alpha_i^{-1}}\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_i\right)\rangle_{q\left(\boldsymbol{s}_{\neg i}^{(n)}\right)q(\boldsymbol{\mu})}$$

Taking the expectation with respect to $q(\boldsymbol{\mu})$,

$$\begin{aligned}
&= \exp\langle -\frac{D}{2}\log 2\pi - D\log\sigma - \frac{1}{2\sigma^2}\left(\boldsymbol{x}^{(n)} - \sum_{i=1}^{K} s_i^{(n)}\boldsymbol{\mu}_i\right)^{\mathsf{T}}\left(\boldsymbol{x}^{(n)} - \sum_{i=1}^{K} s_i^{(n)}\boldsymbol{\mu}_i\right) - \frac{1}{2\sigma^2}\mathrm{Tr}\left(S^{\mathsf{T}}S\sum_{d=1}^{D}\Sigma_{\mu_d}\right)\\
&\quad + \sum_{i=1}^{K}\left(s_i^{(n)}\log\boldsymbol{\pi}_i + \left(1 - s_i^{(n)}\right)\log(1-\boldsymbol{\pi}_i)\right)\\
&\quad + \sum_{i=1}^{K}\left(-\frac{D}{2}\log 2\pi\alpha_i^{-1} \; -\frac{1}{2\alpha_i^{-1}}\langle\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_i\rangle_{q(\boldsymbol{\mu})}\right)\rangle_{q\left(\boldsymbol{s}_{\neg i}^{(n)}\right)}\\
&= \exp\langle -\frac{D}{2}\log 2\pi - D\log\sigma\\
&\quad -\frac{1}{2\sigma^2}\left(\boldsymbol{x}^{(n)\mathsf{T}}\boldsymbol{x}^{(n)} - 2\sum_{j\neg i}^{K} s_j^{(n)}\mu_j^{\mathsf{T}}\boldsymbol{x}^{(n)} - 2s_i^{(n)}\mu_i^{\mathsf{T}}\boldsymbol{x}^{(n)} + \sum_{j,j\neg i}^{K} s_j^{(n)}s_j^{(n)}\boldsymbol{\mu}_j^{\mathsf{T}}\boldsymbol{\mu}_j + 2\sum_{j\neg i}^{K} s_i^{(n)}s_j^{(n)}\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_j + s_i^{(n)}s_i^{(n)}\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_i\right)\\
&\quad -\frac{1}{2\sigma^2}\left(\sum_{m\neg n}^{N}\sum_{i,j=1}^{K} s_i^{(m)}s_j^{(m)} + \sum_{j,j\neg i}^{K} s_j^{(n)}s_j^{(n)} + 2\sum_{j\neg i}^{K} s_i^{(n)}s_j^{(n)} + s_i^{(n)}s_i^{(n)}\right)\sum_{d=1}^{D}\left(\sum_{j,j\neg i}^{K}\Sigma_{\mu_d}^{jj} + 2\sum_{j\neg i}^{K}\Sigma_{\mu_d}^{ij} + \Sigma_{\mu_d}^{ii}\right)\\
&\quad + \sum_{i=1}^{K}\left(\langle s_i^{(n)}\rangle_{q\left(\boldsymbol{s}_{\neg i}^{(n)}\right)}\log\boldsymbol{\pi}_i + \left(1 - \langle s_i^{(n)}\rangle_{q\left(\boldsymbol{s}_{\neg i}^{(n)}\right)}\right)\log(1-\boldsymbol{\pi}_i)\right)\\
&\quad + \sum_{i=1}^{K}\left(-\frac{D}{2}\log 2\pi\alpha_i^{-1} \; -\frac{1}{2\alpha_i^{-1}}\langle\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_i\rangle_{q(\boldsymbol{\mu})}\right)\rangle_{q\left(\boldsymbol{s}_{\neg i}^{(n)}\right)}
\end{aligned}$$

Taking the expectation with respect to $q\left(\boldsymbol{s}_{\neg i}^{(n)}\right)$

$$\begin{aligned}
&\propto \exp\left\{-\frac{1}{2\sigma^2}\left(-2s_i^{(n)}\mu_i^{\mathsf{T}}\boldsymbol{x}^{(n)} + 2\sum_{j\neg i}^{K} s_i^{(n)}s_j^{(n)}\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_j + s_i^{(n)}s_i^{(n)}\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_i\right)\right.\\
&\quad \left.-\frac{1}{2\sigma^2}\left(2\sum_{j\neg i}^{K} s_i^{(n)}s_j^{(n)} + s_i^{(n)}s_i^{(n)}\right)\sum_{d=1}^{D}\left(2\sum_{j\neg i}^{K}\Sigma_{\mu_d}^{ij} + \Sigma_{\mu_d}^{ii}\right) + s_i^{(n)} \; \log\frac{\boldsymbol{\pi}_i}{1-\boldsymbol{\pi}_i}\right.
\end{aligned}$$

Given that $q\left(s_i^n\right) = \lambda_{in}^{s_i^{(n)}}\left(1-\lambda_{in}\right)^{\left(1-s_i^{(n)}\right)}$, $\log q\left(s_i^n\right) \propto s_i^n \log\frac{\lambda_{in}}{1-\lambda_{in}}$. Thus, equal to the log of the expression above,

$$\begin{aligned}
s_i^{(n)}\log\frac{\lambda_{in}}{1-\lambda_{in}} &= -\frac{1}{2\sigma^2}\left(-2s_i^{(n)}\mu_i^{\mathsf{T}}\boldsymbol{x}^{(n)} + 2\sum_{j\neg i}^{K} s_i^{(n)}s_j^{(n)}\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_j + s_i^{(n)}s_i^{(n)}\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_i\right)\\
&\quad -\frac{1}{2\sigma^2}\left(2\sum_{j\neg i}^{K} s_i^{(n)}s_j^{(n)} + s_i^{(n)}s_i^{(n)}\right)\sum_{d=1}^{D}\left(2\sum_{j\neg i}^{K}\Sigma_{\mu_d}^{ij} + \Sigma_{\mu_d}^{ii}\right) + s_i^{(n)} \; \log\frac{\boldsymbol{\pi}_i}{1-\boldsymbol{\pi}_i}
\end{aligned}$$

Substituting $s_i^{(n)} = 1$ and solving for λ_{in},

$$\begin{aligned}
\log\frac{\lambda_{in}}{1-\lambda_{in}} &= -\frac{1}{2\sigma^2}\left(-2\mu_i^{\mathsf{T}}\boldsymbol{x}^{(n)} + 2\sum_{j\neg i}^{K} s_j^{(n)}\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_j + \boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_i\right)\\
&\quad -\frac{1}{2\sigma^2}\left(2\sum_{j\neg i}^{K} s_j^{(n)} + 1\right)\sum_{d=1}^{D}\left(2\sum_{j\neg i}^{K}\Sigma_{\mu_d}^{ij} + \Sigma_{\mu_d}^{ii}\right) + \log\frac{\boldsymbol{\pi}_i}{1-\boldsymbol{\pi}_i}
\end{aligned}$$

$$\lambda_{in} = \frac{1}{-\exp\left\{-\frac{1}{2\sigma^2}(-2\mu_i^\top \boldsymbol{x}^{(n)} + 2\sum_{j\neg i}^{K} s_j^{(n)} \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j + \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_i) - \frac{1}{2\sigma^2}(2\sum_{j\neg i}^{K} s_j^{(n)} + 1)\sum_{d=1}^{D}(2\sum_{j\neg i}^{K}\Sigma_{\mu_d}^{ij} + \Sigma_{\mu_d}^{ii}) + \log\frac{\boldsymbol{\pi}_i}{1-\boldsymbol{\pi}_i}\right\}}$$

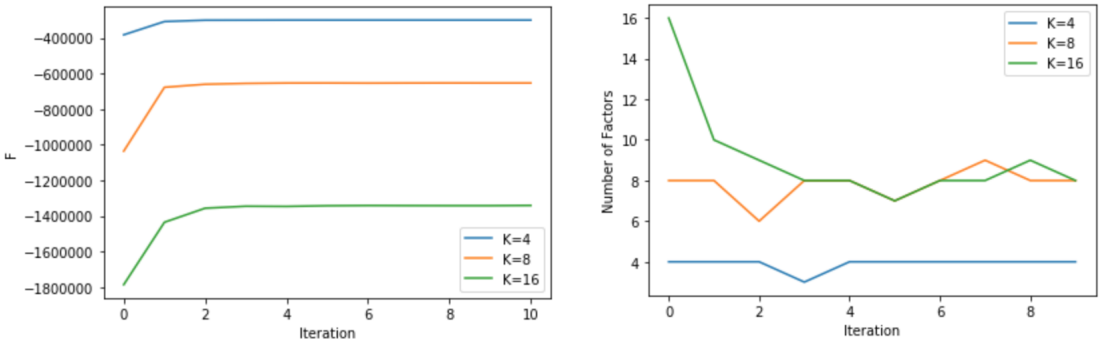
b. The implementation in (1) is modified to reflect the new algorithm. In particular, the update of `lambda`, `mu`, and `sigma` are altered. Additionally, the parameter `alpha` and the covariance of `mu` is computed. `F` is also altered to reflect the new computation of free energy.

```
def EStep(X,mu,sigma,pie,lambda0,i,n,pcovm):
    D=X.shape[1]
    tlambda=np.dot(lambda0[n],mu.T)-np.dot(lambda0[n,i],mu[:,i].T)
    tlambda=(1/sigma**2)*np.dot((X[n]-tlambda),mu[:,i])
    tlambda=tlambda-(1/(2*sigma**2))*np.dot(mu[:,i].T,mu[:,i])
    tlambda=tlambda + np.log(pie[0][i]/(1-pie[0][i])+.0000001)
    tlambda2=2*(np.sum(lambda0[n])-lambda0[n,i])+1
    tlambda3=(2*np.sum(pcovm[i])-pcovm[i,i])
    tlambda=tlambda+D*tlambda2*tlambda3
    tlambda=1/(1+np.exp(-tlambda))
    return tlambda

def FreeEnergy(X,mu,sigma,pie,lambda0,pcovm,alpha):
    F=0
    D=lambda0.shape[1]
    N=X.shape[0]
    ESS=compute_ESS(lambda0)
    ESS=np.sum(ESS,axis=0)
    for n in range(lambda0.shape[0]):
        tF=X[n]-np.dot(lambda0[n],mu.T)
        tF=np.dot(tF.T,tF)
        tF2=np.sum(np.dot(np.dot(lambda0[n]-lambda0[n]**2,mu.T),mu))
        tF=(-1/(2*sigma**2))*(tF+tF2)-D*np.log(sigma)-(D/2)*np.log(2*np.pi)
        tF2=np.dot(lambda0[n],np.log(pie[0]+.0000001).T)
            -np.dot(lambda0[n],np.log(lambda0[n]+.0000001).T)
        tF2=tF2+np.sum(np.log(1-pie[0]+.0000001))
            -np.sum(np.log(1-lambda0[n]+.0000001))
        tF2=tF2-np.dot(lambda0[n],np.log(1-pie[0]+.0000001).T)
            +np.dot(lambda0[n],np.log(1-lambda0[n]+.0000001))
        F+=tF+tF2
    tF3=(-D/2)*np.log(2*np.pi)+(D/2)*np.log(alpha+0.00001)
    tF3=np.sum(-0.5*np.dot(alpha,np.dot(mu.T,mu))) +np.sum(tF3) # blows up
    Scov=np.sum(ESS, axis=0)-np.dot(lambda0.T,lambda0)
    tF4=(-1/(2*sigma**2))*np.trace(np.dot(np.dot(mu.T,mu),Scov))
    tF5=(-1/(2*sigma**2))*np.trace(np.dot(ESS,D*pcovm))
    F+=tF4+tF5+tF3
    return F

def calc_alpha(mu):
    alpha=np.zeros((mu.shape[1]))
    for k in range(0,mu.shape[1]):
        alpha[k]=mu.shape[0]/np.dot(mu[:,k].T,mu[:,k])
    return alpha

def calc_mu(X,ES,sigma,alpha):
    pcovm=np.zeros((ES.shape[1],ES.shape[1]))
    mu=np.zeros((X.shape[1],ES.shape[1]))
    ESS=compute_ESS(ES)
```

```
    ESS=np.sum(ESS, axis=0)
    pcovm=LA.inv(ESS*(sigma**(-2)) + LA.inv(np.diag(alpha)))
    for d in range(X.shape[1]):
        pmean=np.dot(np.dot(pcovm, ES.T),X[:,d])*(sigma**(-2))
        mu[d]=pmean
    return mu,pcovm

def m_step(X, ES, ESS,alpha,mu,pcovm):
    N, D = X.shape
    if ES.shape[0] != N:
        raise TypeError('ES must have the same number of rows as X')
    K = ES.shape[1]
    if ESS.shape == (N, K, K):
        ESS = np.sum(ESS, axis=0)
    if ESS.shape != (K, K):
        raise TypeError('ESS must be square and have the same number of columns as ES')
    try:
        sigma
    except NameError:
        sigma=np.random.rand(1)
    sigma = np.sqrt((np.trace(np.dot(X.T, X)) + np.trace(np.dot(np.dot(mu.T, mu), ESS))
        - 2 * np.trace(np.dot(np.dot(ES.T, X), mu))+
        np.trace(np.dot(np.dot(mu.T,mu),ESS-np.dot(ES.T,ES)))
        + np.trace(np.dot(ESS,pcovm))
        / (N * D)))
    pie = np.mean(ES, axis=0, keepdims=True)
    return mu, sigma, pie

def init_params(X,K):
    N=X.shape[0]
    D=X.shape[1]
    ES=np.random.rand(N,K)
    ESS=compute_ESS(ES)
    alpha=np.random.rand(K)
    sigma=np.random.rand()
    mu,pcovm=calc_mu(X,ES,sigma,alpha)
    mu,sigma,pie=m_step(X,ES,ESS,alpha,mu,pcovm)
    return ES,ESS,mu,sigma,pie,pcovm,alpha

def MeanField(X,mu,sigma,pie,lambda0,maxsteps,pcovm,alpha):
    lambda1=np.copy(lambda0)
    prevlambda=lambda1
    prevF=FreeEnergy(X,mu,sigma,pie,lambda1,pcovm,alpha)
    for m in range(maxsteps):
        for n in range(lambda1.shape[0]):
            for i in range(lambda1.shape[1]):
                lambda1[n][i]=EStep(X,mu,sigma,pie,lambda1,i,n,pcovm)
        mu,pcovm=calc_mu(X,lambda1,sigma,alpha)
        F=FreeEnergy(X,mu,sigma,pie,lambda1,pcovm,alpha)
        if F-prevF < 0.0000001:
            break
        prevF=F
        prevlambda=lambda1
    return lambda1,F,mu,pcovm

def LearnBinFactors(X,K,iterations):
```

```
maxsteps=20
ES,ESS,mu,sigma,pie,pcovm,alpha=init_params(X,K)
F_array=[]
prevF=FreeEnergy(X,mu,sigma,pie,ES,pcovm,alpha)
F_array.append(prevF)
for i in range(iterations):
    ES,F,mu,pcovm=MeanField(X,mu,sigma,pie,ES,maxsteps,pcovm,alpha)
    ESS=compute_ESS(ES)
    mu,sigma,pie=m_step(X,ES,ESS,alpha,mu,pcovm)
    alpha=calc_alpha(mu)
    F=FreeEnergy(X,mu,sigma,pie,ES,pcovm,alpha)
    print(F)
    F_array.append(F)
    if F<=prevF:
        print("Dif ",F-prevF)
    prevF=F
return mu,sigma,pie,F_array,ES,alpha
```

Models are fit with maximum values of $K = 4, 8, 16$ and the VB free energy and number of effective factors is plotted as a function of iteration.



The ordering of free energies is sensible as lower maximum values of K have higher values of free energy than higher maximum values. With more factors to discern and learn from, precision decreases and thus the free energy is also lower. Notably, it also takes longer for `F` to converge with higher values of K. The effective number of factors is also sensible as the they converge to approximately the accurate number of factors available for the maximum value of K.