Comp 0086 Assignment 2

Alexandra Maria Proca (SN: 20047328)

November 12, 2020

Question 1.

a.

The likelihood for the model of K multivariate Bernoulli distributions over N images, $P(D|\pi, \mathsf{P})$ *can be written as the product of the probability of each mixing proportion and the likelihood of* $\mathbf{x}^{(n)}$ *given* P_k

$$\prod_{k=1}^{N} \sum_{k=1}^{K} \pi_k \mathrm{Ber}\left(\mathbf{x}^{(n)}|\mathsf{P}_k\right)$$
$$= \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \left(\prod_{d=1}^{D} \left(\mathsf{P}_{kd}^{x_d^{(n)}} \right) (1-\mathsf{P}_{kd})^{(1-x_d^{(n)}} \right)$$

b.

The responsibility of the kth cluster at the nth data point is the posterior probability distribution for each cluster.
Thus,

$$r_{nk} \equiv P\left(s^{(n)} = k|\mathbf{x}^{(n)}, \pi, \mathsf{P}\right)$$
$$= \frac{P\left(\mathbf{x}^{(n)}|s^{(n)}=k\right)P\left(s^{(n)}=k\right)}{P\left(\mathbf{x}^{(n)}|\pi,\mathsf{P}\right)}$$

The prior $P\left(s^{(n)} = k\right) = \pi_k$ *by the definition of the mixing proportions,*
The likelihood $P\left(\mathbf{x}^{(n)}|s^{(n)} = k\right) = \prod_{d=1}^{D} \left(\mathsf{P}_{kd}^{x_d^{(n)}} \right) (1-\mathsf{P}_{kd})^{(1-x_d^{(n)}}$
and the evidence is the probability of generating the data for all k,

$$\sum_{k=1}^{K} \left(\pi_k \prod_{d=1}^{D} \left(\mathsf{P}_{kd}^{x_d^{(n)}} \right) (1-\mathsf{P}_{kd})^{(1-x_d^{(n)}} \right)$$

Thus, the responsibility is

$$r_{nk} = \frac{\pi_k \prod_{d=1}^{D} \left(\mathsf{P}_{kd}^{x_d^{(n)}} \right) (1-\mathsf{P}_{kd})^{(1-x_d^{(n)}}}{\sum_{k=1}^{K} \left(\pi_k \prod_{d=1}^{D} \left(\mathsf{P}_{kd}^{x_d^{(n)}} \right) (1-\mathsf{P}_{kd})^{(1-x_d^{(n)}} \right)}$$

c.

$$\arg\max_{\pi,\ \mathsf{P}} < \sum_{n} \log P\left(\mathbf{x}^{(n)}, s^{(n)}|\pi, \mathsf{P}\right) >_{q\left(\left\{s^{(n)}\right\}\right)}$$
$$= \arg\max_{\pi,\ \mathsf{P}} < \sum_{n=1}^{N} \log\left(P\left(s^{(n)} = k|\mathbf{x}^{(n)}, \pi, \mathsf{P}\right) P\left(\mathbf{x}^{(n)}|\pi, \mathsf{P}\right)\right) >_{q\left(\left\{s^{(n)}\right\}\right)}$$
$$= \arg\max_{\pi,\ \mathsf{P}} \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \log\left(\pi_k \prod_{d=1}^{D} \left(\mathsf{P}_{kd}^{x_d^{(n)}} \right) (1-\mathsf{P}_{kd})^{(1-x_d^{(n)}} \right)$$

To solve for π*, the partial derivative with respect to* π_k *is taken*

$$\arg\max_{\pi} \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \log\left(\pi_k \prod_{d=1}^{D} \left(\mathsf{P}_{kd}^{x_d^{(n)}} \right) (1-\mathsf{P}_{kd})^{\left(1-x_d^{(n)}\right)} \right)$$

Because the expectation is being maximized with respect to π_k, $\prod_{d=1}^{D}\left(\mathsf{P}_{kd}^{x_d^{(n)}}\right)(1-\mathsf{P}_{kd})^{\left(1-x_d^{(n)}\right)}$ *remains constant and can be removed as it is not a function of* π_k,

$= \arg\max_{\pi} \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\log\pi_k$

Given the constraint on π, $\sum_{k=1}^{K}\pi_k = 1$, *a Lagrange multiplier can be used to solve the system of equations,*

$f(\pi) = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\log\pi_k$

$g(\pi) = \left(\sum_{k=1}^{K}\pi_k\right) - 1 = 0$

To first solve for the partial derivative with respect to π_k,

$\left(\left(\sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\log\pi_k\right) + \lambda\left(\left(\sum_{k=1}^{K}\pi_k\right) - 1\right)\right)\frac{\partial}{\partial\pi_k} = 0$

Because the derivative of all $\pi \neq \pi_k$ *is* 0, *the summation over* K *is removed,*

$\left(\left(\sum_{n=1}^{N} r_{nk}\log(\pi_k)\right) + \lambda(\pi_k - 1)\right)\frac{\partial}{\partial\pi_k}$

$= \left(\sum_{n=1}^{N}\frac{r_{nk}}{\pi_k}\right) + \lambda$

$= \frac{\sum_{n=1}^{N} r_{nk}}{\pi_k} + \lambda = 0$

$\pi_k = \frac{-\sum_{n=1}^{N} r_{nk}}{\lambda}$

To solve for the partial derivative with respect to λ,

$\left(\left(\sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\log\pi_k\right) + \lambda\left(\left(\sum_{k=1}^{K}\pi_k\right) - 1\right)\right)\frac{\partial}{\partial\lambda} = 0$

Replacing π_k *with the value found in terms of* λ,

$= \left(\left(\sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\log\left(\frac{-\sum_{n=1}^{N} r_{nk}}{\lambda}\right)\right) + \lambda\left(\left(\sum_{k=1}^{K}\frac{-\sum_{n=1}^{N} r_{nk}}{\lambda}\right) - 1\right)\right)\frac{\partial}{\partial\lambda}$

$= \left(\left(\sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\log\left(-\sum_{n=1}^{N} r_{nk}\right) - r_{nk}\log\lambda\right) + \left(\sum_{k=1}^{K}\left(-\sum_{n=1}^{N} r_{nk}\right)\right) - \lambda\right)\frac{\partial}{\partial\lambda}$

$= \left(\sum_{n=1}^{N}\sum_{k=1}^{K} -\frac{r_{nk}}{\lambda}\right) - 1$

$= \frac{1}{\lambda}\left(\sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\right) - 1 = 0$

$\lambda = -\sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}$

Solving for π_k *using* λ,

$\pi_k = \frac{-\sum_{n=1}^{N} r_{nk}}{-\sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}} = \frac{\sum_{n=1}^{N} r_{nk}}{\sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}}$

To solve for P, *the partial derivative with respect to* P_{kd} *is taken*

$$\left(\sum_{n=1}^{N} r_{nk}\log\left(\sum_{k=1}^{K}\left(\pi_k\prod_{d=1}^{D}\left(\mathsf{P}_{kd}^{x_d^{(n)}}\right)(1-\mathsf{P}_{kd})^{\left(1-x_d^{(n)}\right)}\right)\right)\right)\frac{\partial}{\partial \mathsf{P}_{kd}}$$

Because the derivative of all $\mathrm{P} \neq \mathrm{P}_{kd}$ *is* 0, *the summation over* K *and product over* D *can be removed*

$$= \sum_{n=1}^{N} r_{nk}\log\left(\pi_k\left(\mathsf{P}_{kd}^{x_d^{(n)}}\right)(1-\mathsf{P}_{kd})^{(1-x_d^{(n)}}\right)\frac{\partial}{\partial \mathsf{P}_{kd}}$$

$$= \left(\sum_{n=1}^{N} r_{nk}\log\pi_k + r_{nk}x_d^{(n)}\log\mathsf{P}_{kd} + r_{nk}\left(1-x_d^{(n)}\right)\log(1-\mathsf{P}_{kd})\right)\frac{\partial}{\partial \mathsf{P}_{kd}}$$

$$= \sum_{n=1}^{N}\frac{r_{nk}x_d^{(n)}}{\mathsf{P}_{kd}} - \frac{r_{nk}\left(1-x_d^{(n)}\right)}{1-\mathsf{P}_{kd}}$$

$$= \frac{\sum_{n=1}^{N} r_{nk}x_d^{(n)}}{\mathsf{P}_{kd}} - \frac{\sum_{n=1}^{N} r_{nk}-r_{nk}x_d^{(n)}}{1-\mathsf{P}_{kd}}$$

$$= (1-\mathsf{P}_{kd})\sum_{n=1}^{N} r_{nk}x_d^{(n)} - \mathsf{P}_{kd}\left(\sum_{n=1}^{N} r_{nk} - r_{nk}x_d^{(n)}\right)$$

$$= \sum_{n=1}^{N} r_{nk}x_d^{(n)} - \mathsf{P}_{kd}\sum_{n=1}^{N} r_{nk}x_d^{(n)} - \mathsf{P}_{kd}\sum_{n=1}^{N} r_{nk} + \mathsf{P}_{kd}\sum_{n=1}^{N} r_{nk}x_d^{(n)}$$

$$= \sum_{n=1}^{N} r_{nk}x_d^{(n)} - \mathsf{P}_{kd}\sum_{n=1}^{N} r_{nk}$$

$$\mathsf{P}_{kd}\sum_{n=1}^{N} r_{nk} = \sum_{n=1}^{N} r_{nk}x_d^{(n)}$$

$$\mathsf{P}_{kd} = \frac{\sum_{n=1}^{N} r_{nk}x_d^{(n)}}{\sum_{n=1}^{N} r_{nk}}$$

d.

The EM algorithm is implemented for the mixture of K multivariate Bernoullis. Because of numerical problems in python, the implementation uses a MAP calculation instead of a ML calculation, with a weak prior of B(1.0001,1.0001) and the likelihood is scaled by 0.001 before computing the log-likelihood of each image. The EM_Funct declares the prior, instantiates random parameters, and runs the algorithm for the given number of epochs. In each epoch, it first calls responsibility for the E step, which calculates the responsibility and likelihood by iterating through P, X, and pi and calculating the product of univariate Bernoullis, summing their product with pi for every K and dividing by the total sum of the product of Bernoullis over D and K and pis at K to find R at every N and K. The log is taken of the likelihood scaled by 0.001 and summed over all N. The EM_Funct then calls P_calc for the M step, which sums the product of R[n][k] and X[n][d] over all N and sums the responsibilities over all N. It then uses the MAP calculation of P, with the predefined values of alpha and beta. Finally, the EM_Funct calculates pi for the M step by summing R at each k and dividing by the number of images. The algorithm terminates once it reaches the epoch number of iterations or until the log-likelihood converges.

To maximize P *in* MAP *with a Beta prior,*

$$\arg\max_{\mathsf{P}} < \log P(\mathbf{x}, s|\pi, \mathsf{P})P(\mathsf{P}|\alpha,\beta) >$$

$$= \arg\max_{\mathsf{P}} \left(\sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \log \left(\pi_k \prod_{d=1}^{D} \left(\mathsf{P}_{kd}^{x_d^{(n)}} \right) (1-\mathsf{P}_{kd})^{\left(1-x_d^{(n)}\right)} \right) + \log \prod_{k=1}^{K} \prod_{d=1}^{D} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \mathsf{P}_{kd}^{\alpha-1}(1-\mathsf{P}_{kd})^{(\beta-1)} \right)$$

Substituting the derivation found in part (b) and simplifying,

$$= \frac{\sum\limits_{n=1}^{N} r_{nk}x_d^{(n)}}{\mathsf{P}_{kd}} - \frac{\sum\limits_{n=1}^{N} r_{nk}-r_{nk}x_d^{(n)}}{1-\mathsf{P}_{kd}} + \left(\sum_{k=1}^{K} \sum_{d=1}^{D} (\alpha-1)\log \mathsf{P}_{kd} + (\beta-1)\log(1-\mathsf{P}_{kd}) \right) \frac{\partial}{\partial \mathsf{P}_{kd}}$$

$$= \frac{\sum\limits_{n=1}^{N} r_{nk}x_d^{(n)}}{\mathsf{P}_{kd}} - \frac{\sum\limits_{n=1}^{N} r_{nk}-r_{nk}x_d^{(n)}}{1-\mathsf{P}_{kd}} + \frac{\alpha-1}{\mathsf{P}_{kd}} - \frac{\beta-1}{1-\mathsf{P}_{kd}}$$

$$= \frac{\sum\limits_{n=1}^{N} r_{nk}x_d^{(n)}+\alpha-1}{\mathsf{P}_{kd}} - \frac{\sum\limits_{n=1}^{N} \left(r_{nk}-r_{nk}x_d^{(n)}\right)-\beta+1}{1-\mathsf{P}_{kd}}$$

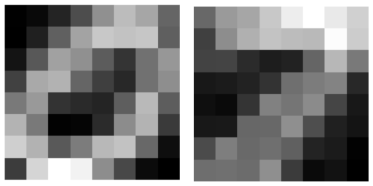
$$= (1-\mathsf{P}_{kd}) \left(\sum_{n=1}^{N} r_{nk}x_d^{(n)} + \alpha - 1 \right) - \mathsf{P}_{kd} \left(\sum_{n=1}^{N} \left(r_{nk} - r_{nk}x_d^{(n)} \right) - \beta + 1 \right)$$

Simplifying and solving for P_{kd}*,*

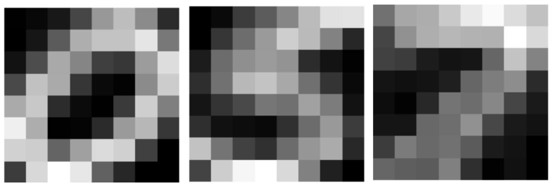
$$\mathsf{P}_{kd} = \frac{\alpha-1+\sum\limits_{n=1}^{N} r_{nk}x_d^{(n)}}{\alpha+\beta-2+\sum\limits_{n=1}^{N} r_{nk}}$$

```
import numpy as np
import random
import math
data=np.loadtxt('binarydigits.txt')

# Calculates the responsibility in the E step and the log-likelihood
def responsibility(K, P, X, pi, prior):
    N,D=X.shape
    R=np.zeros((N,K))
    loglike=0
    bitlike=0
    for n in range(0,N):
        denom=0
        for k in range(0,K):
            numer=1
            for d in range(0,D):
                # Calculates product of univariate Bernoulli at K and N over D
                numer=numer*((P[k][d]**(X[n][d]))*((1-P[k][d])**(1-X[n][d])  ))
            # Sums product of Bernoulli with pi for every K to calculate evidence
            denom=denom+pi[k]*numer
            # Product of Bernoulli and pi at K to calculate numerator of responsibility
            R[n][k]=numer*pi[k]
        # Divides numerator by denominator to calculate responsibility
        R[n]=R[n]/denom
        # Calculates log-likelihood, scaled by 0.001
        loglike=loglike+np.log(denom + 0.001)
        # Calculates log2-likelihood, scaled by 0.001
        bitlike=bitlike+np.log2(denom+0.001)
    return R, loglike, bitlike

# Calculates P in the M step
def P_calc(R, X, K, alpha, beta):
    N,D=X.shape
    P=np.zeros((K, D))
    sum=0
    for k in range(0,K):
        for d in range(0,D):
            for n in range(0,N):
                # Sums the product of responsibility and x over N and D
                P[k][d]=P[k][d]+R[n][k]*X[n][d]
                # Sums responsibilities over N
                sum=sum+R[n][k]
            #P[k][d]=P[k][d]/sum                         # ML Calculation
            P[k][d]=(alpha-1+P[k][d])/(alpha+beta-2+sum)  # MAP Calculation
            sum=0
    return P

def EM_Funct(K, X, epoch):
    alpha=1.0001
    beta=1.0001
    prior=math.gamma(alpha)*math.gamma(beta)/math.gamma(alpha+beta)
    N,D=X.shape
    P=np.random.random((K, D))
    plt.figure()
    pi=np.random.random(K)
    pi=pi/np.sum(pi)
    e=0
    LArr=[]
    EArr=[]
    prevL=0
    while e < epoch:
        # Calculates responsibility and log-likelihoods for E step
        R,L,BitL=responsibility(K,P,X,pi,prior)
        # Calculates P for M step
        P=P_calc(R,X,K, alpha, beta)
        # Calculates pi for M step
        for k in range(0,K):
            pi[k]=np.sum(R[:,k])/N
        if L-prevL < e-20:
            break
        prevL=L
        EArr.append(e)
        e=e+1
        LArr.append(L)
    return R, P, L, pi, BitL, LArr
```

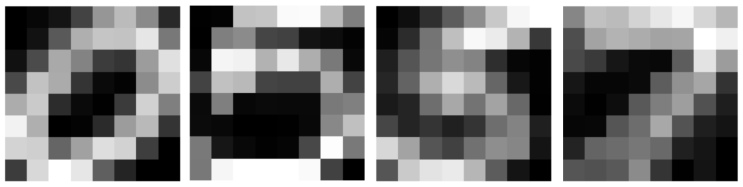
For K=2, an average log-likelihood of -3356.533485 is obtained over 20 runs. For one run, pi is [0.64495941 0.35504059]
and P displayed as images for each cluster is



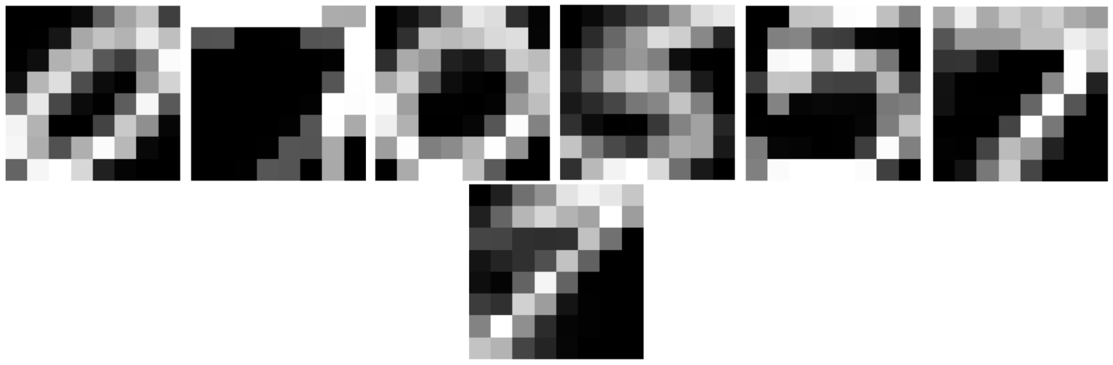
For K=3, an average log-likelihood of -3140.945762 is obtained over 20 runs. For one run, pi is [0.43175923 0.24968852 0.31855225]
and P displayed as images for each cluster is



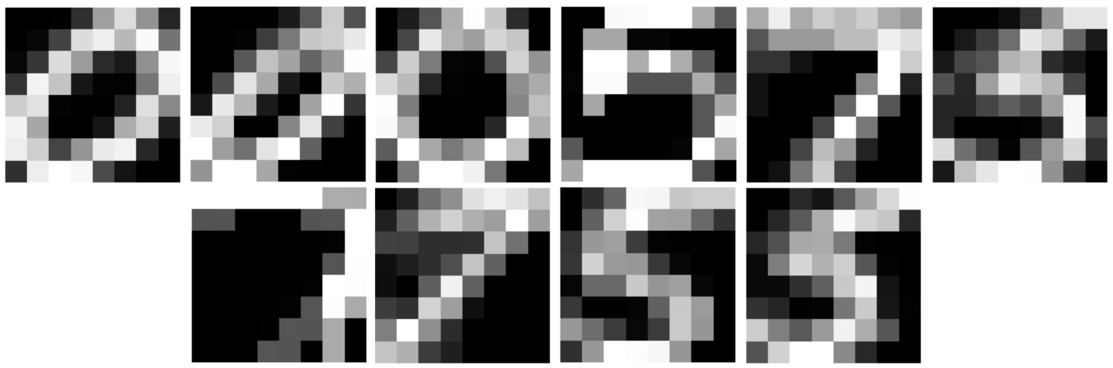
For K=4, an average log-likelihood of -2938.952761 is obtained over 20 runs. For one run, pi is
[0.42145892 0.04372178 0.26016971 0.27464959]
and P displayed as images for each cluster is



For K=7, an average log-likelihood of -2577.120503 is obtained over 20 runs. For one run, pi is
[0.25312355 0.03015581 0.16353391 0.19967551 0.04037124 0.15123922 0.16190076]
and P displayed as images for each cluster is



For K=10, an average log-likelihood of -2424.395138 is obtained over 20 runs. For one run, pi is
[0.22960152 0.09504444 0.0896707 0.03054327 0.15037351 0.10413275 0.03013278 0.15772364 0.04874147 0.06403593]
and P displayed as images for each cluster is



e.

The log-likelihoods expressed in bits are the $\log_2$(likelihood), found in the implementation of the EM algorithm. They vary each time because of the randomization of the initial parameters, but the absolute value of the average found over 20 separate runs is:

K	Log-likelihood in bits
2	4821.7473955
3	4430.8145303
4	4216.1918895
7	3769.6512523
10	3519.1871328

The naive encoding of the binary data is 100 image x 64 pixels = 6400, assuming the data is in bits. The absolute value of the log-likelihood in bits is smaller than the naive encoding by about 1500-3000, depending on the size of K (as K increases, the size of the encoding decreases). This means that the encoding of the data from the EM algorithm is more memory-efficient than the original data, dependent on K. A higher K yields a more optimal encoding of the data, as it models the probability distribution of the data more precisely than a lower K. The length of the gzip encoding yields 5416 bits, which is larger than the EM encoding and smaller than the naive encoding. The difference of the gzip compression is due to its use of a lossless data compression algorithm (symbol coding). Symbol coding uses shorter bit stream lengths to represent characters that appear more frequently and longer bit streams for characters that are less frequent. However, it requires at least one bit for each symbol and cannot deal with a change in frequency of characters elegantly. Gzip is deterministic rather than being probabilistic and does not produce an assumption about density, as seen in probabilistic modeling. Alternatively, the EM algorithm uses probabilistic modeling (arithmetic coding), which is closer to optimal compression. Probabilistic modeling attempts to learn from the data and predict the best model in order to compress it, as done in this implementation.

f.

Running the algorithm several times from random initializations, the same solutions are obtained up to permutations of the clusters, with very small variations in numerical values of the responsibility, P, and π. Visualized, each P cluster resembles the digits 0, 5, and 7. The clusters depend on K; as K increases, there are more forms/orientations of digits in clusters that appear and the clarity of the cluster images increases. The algorithm works well as it clusters images into the three types of images present (0,5,7), accurately distinguishing the data, as shown by each P at K. It also accurately quantifies the ratio of images for which a given cluster is most likely to have generated (the responsibilities). One way to improve the model would be by finding the appropriate number of clusters (presumably 3) without making it a hyperparameter K, perhaps by establishing a threshold of minimum variance between P_k's, as this would accurately label the data without further subdividing it into more clusters and increasing the cost of computation. Another way to improve the model could be to find a more accurate prior than a weak beta prior, as was used in the above MAP calculation.

g.

The total cost of encoding both the model parameters and the data given the model can be found by summing the absolute value of the encoding of the data (bit log-likelihood) with $64\times(K+K\times64)$ (the product of the number of floating bits and the size of the parameters, π and P).

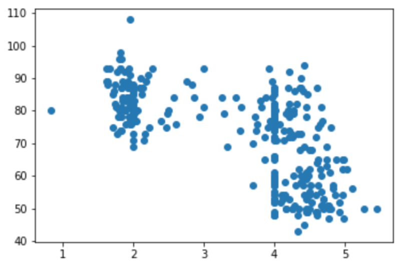
K	Total Log-likelihood in bits
2	13141.747395
3	16910.814530
4	20856.191889
7	32889.651252
10	45119.187132

The total cost grows to be much larger than gzip because it introduces more data (parameters) to be encoded with every K. The total cost of encoding both the model parameters and the data increases as K increases because each K enlarges the size of P, thus increasing the amount of data being encoded and stored. This indicates that there is a tradeoff between the number of clusters and specificity of P provided by a larger K and overall cost. As K increases, computation is more expensive and requires more memory. Thus, having a moderate value for K that allows important features to arise without highly increasing dimensionality of the data stored may be an important balance. Additionally, the parameter encoding far outweights the lower price of encoding the original data probabilistically in bits, becoming more expensive than both the gzip encoding and the naive encoding. Thus, this method of compression becomes unoptimal and expensive when parameters must be encoded.

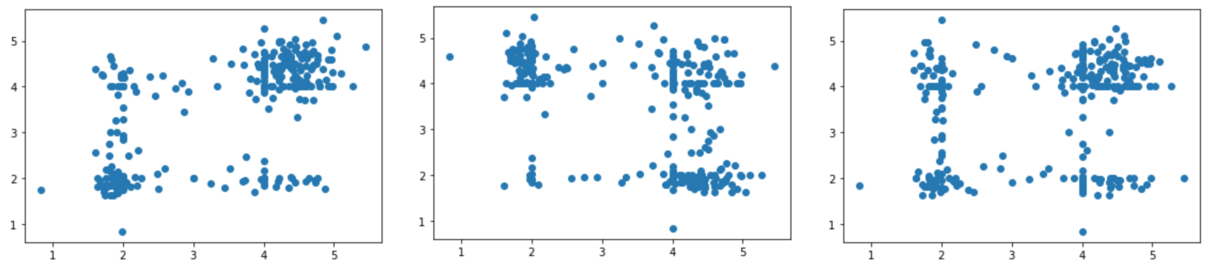
Question 2.

a.

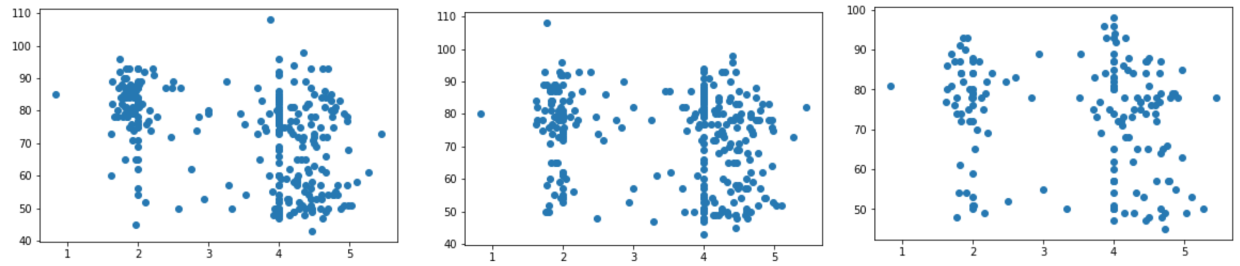
I believe that a hidden Markov model (HMM) would be most appropriate for this dataset.



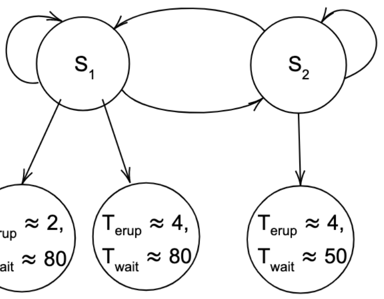
Plotting the within variable data above, it can be seen that there is a high density of points around 2 and 4 minute eruption time, and 50 and 80 minute waiting time, suggesting a distribution around these values and potentially a hidden variable generating these clusters of data. The plot indicates that the model is not multivariate normal, as it has several different areas of high density.



Plotting the latent time series for the eruption time above, the plot produces 4 clusters, with a clear average highest density around (4,4), moderate densities around (2,4) and (4,2), and lowest density around (2,2), indicating time dependency of clustering states. Time dependency of the model indicates that it is also not a mixture of Gaussians. Similar clustering can be seen when plotting the latent time series for waiting time.



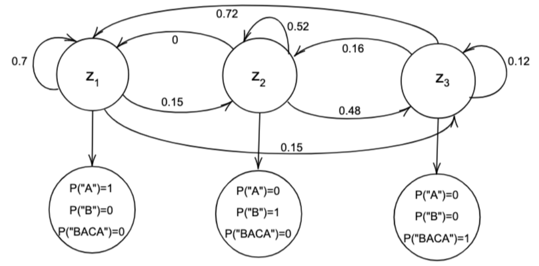
Plotting the latent time series for eruption time by waiting time above, four clusters again emerge with the highest density around (4,80), moderate densities around (2,80) and (4,50), and lowest density around (2,50). Visualizing these time-dependent clusterings, it can be concluded that the model has hidden variables producing certain distributions of data, thus eliminating the Markov chain. Furthermore, the discrete clusterings and seeming correspondence to particular distributions indicates the hidden states are discrete, not continuous, and therefore indicates that the model is not a linear Gaussian state-space model or a stochastic linear dynamical system. A hidden Markov model would thus be the most appropriate for the dataset, as it has hidden states producing the distribution of data. From these observations, I propose the following HMM:



where s_1 and s_2 are the latent variable states and the emission of each latent variable is a point sampled from a multivariate Gaussian distribution of the parameters shown above for the eruption time and waiting time variables. The transition probabilities of the latent variables would be such that each emission is equally probable. The HMM would produce the data observed and would further explain the distribution of clustering in the latent time series plots. Because $P(T_{erup} \approx 2) = \frac{1}{3}$ and $P(T_{erup} \approx 4) = \frac{2}{3}$, for the eruption time latent time series, there is a lower average density ($\approx \frac{1}{9}$ at (2,2), and a highest density ($\approx \frac{4}{9}$) at (4,4), with (2,4) and (4,2) approximately equal in density ($\approx \frac{2}{9}$ each), found by multiplying the probability of the corresponding emissions. The same can be done for the waiting time latent time series, where the data is similarly distributed. Finally, the HMM would produce the result of the plotting of all data points, as it ensures data points of parameters $T_{erup} \approx 2$ and $T_{wait} \approx 50$ are not possible and the other (2,80),(4,80), and (4,50) are approximately evenly distributed.

b.

From studying the string, several characteristics can be observed. Every time the character "C" appears, it is enclosed in a pattern of characters "BACA". If this string is itself treated as a single output of a latent variable, then it follows that outside of "BACA", the character "A" never follows "B" and "C" never appears elsewhere, meaning there are no patterns of "BA", "AC", "CA", "BC", or "CB". Another observation is that there are long strings of consecutive "A"s, smaller strings of consecutive "B"s, and only two strings of consecutive "BACA"s. A final observation is that "BACA" is often followed by a string of "A"s. Using the observations, a hidden Markov model can be suggested with three hidden states and three possible emissions: "A","B", and "BACA". I suggest the following HMM model that could have generated the string:



To find the emission probabilities, this HMM assumes that each state has a probability of 1 of generating a specific string and a probability of 0 of generating any other string to preserve the overall structure of the proposed model and given string. This yields the following emission matrix:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where E_{ij} = the probability of emitting string i at latent state j and the three possible emission strings are X_1="A", X_2="B", and X_3="BACA".

To approximate the probabilities of state transitions, the number of transitions from state i to state j can be counted and divided by the total number of transitions of state i. For z_1, it transitions into itself 39 times, into z_2 8 times, and into z_3 8 times. For z_2, it transitions into itself 10 times, into z_1 0 times, and into z_3 9 times. For z_3, it transitions into itself 2 times, into z_1 13 times, and into z_2 3 times. Dividing each number by the number of transitions of state i (55, 18, and 19, respectively) yields the following transition matrix:

$$\begin{bmatrix} 0.7 & 0.15 & 0.15 \\ 0 & 0.52 & 0.48 \\ 0.72 & 0.16 & 0.12 \end{bmatrix}$$

where T_{ij}=the probability of transition from hidden state i to state j.

Finally, to approximate the initial state probability, the probability of being in each state at any given point in the string can be calculated by dividing the number of instances of state i by the total number of state instances. z_1 has 55 state instances, z_2 has 21 state instances, and z_3 has 21 state instances. This yields the following initial state probabilities: $\begin{bmatrix} 0.56 & 0.22 & 0.22 \end{bmatrix}$.

Question 3.

a.

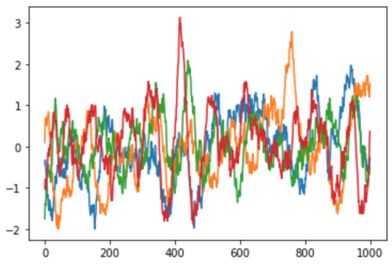
The Viterbi algorithm is guaranteed to converge. If the entire mass is placed on the single most probable state sequence and the other possible state sequences go to 0, the EM algorithm will operate on one sequence of the hidden variables to maximize it. In doing this, the algorithm will no longer make different partitions of other possible sequences, having only one. Instead it will maximize the parameters at the most-probable state sequence and reach convergence of that particular state sequence. Rather than computing the expected sufficient statistics from the posterior distribution over hidden states, it will compute the expected sufficient statistics over the averaged expectation.

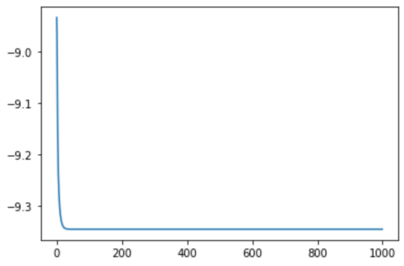
b.

The Viterbi algorithm will converge to the maximum of the likelihood at the particular chosen state sequence when isolated. However, this maximum parameterization of the the most-probable state sequence may not actually be the optimal parameterization of the model itself. If the maximized parameters are reintroduced into the model with all state sequences, there may actually be other parameterizations of all of the state sequences that could be more optimal. The Viterbi algorithm selects the path with the maximum number of expected correct states, but the path is not necessarily the single path with the highest probability of generating the data- it could even have a probability of 0.

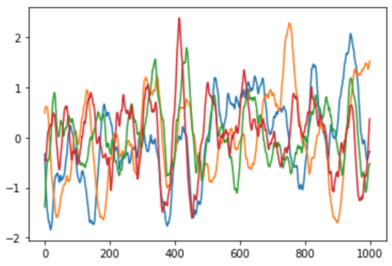
Question 4.

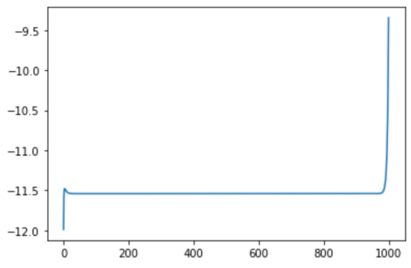
a.





The behavior of Y in the filtered Kalman, as shown in the left figure above, appears to be a periodic oscillation with additional noise contributing to its jagged unevenness in individual curves. The behavior of V in the filtered Kalman, as shown in the right figure above, is a rapid decrease at t=0 and a flat asymptote approximately -9.35 for all t afterwards.





The behavior of Y in the smooth Kalman, as shown in the left figure above, appears to be a periodic oscillation with relatively smooth curves. In contrast with the filtered Kalman, in the smooth Kalman there appears to be less noise in Y resulting in an almost identical pattern of curves with smoother plots and a slightly smaller range of Y values, suggesting the influence of a filter on the posterior means. The average values of Y oscillate similarly, as the hidden state is transferred in both Y. Additionally, both Y reach a maximum around t=410.

The behavior of V in the smooth Kalman, as shown in the right figure above, is an increase at t=0 and a flat asymptote around -11.5 until approximately t=1000, where V increases rapidly again to approximately -9.35, the same value that the filtered V converged to. Both the filtered and the smooth V's reach the same value at t=1000, but arrive at significantly different times (near t=0 vs. near t=1000). The variance in the filtered Kalman thus remains steady, as its posterior stabilizes it and it computes the variance at each time step. Conversely, the smooth variance is computed using all data, resulting in a lower variance until the final time step when it increases, given that it has passed through all the data at each time step.

b.

$$The\ log-likelihood\ can\ be\ written$$
$$l = \sum_{t=1}^{T} \log p(\mathbf{x}_t|\mathbf{x}_1, ..., \mathbf{x}_{t-1})$$
$$= \sum_{t=1}^{T} -\tfrac{1}{2}\log 2\pi - \tfrac{1}{2}\log|R| - \tfrac{1}{2}(\mathbf{x}_t - C\mathbf{y}_t)^\mathsf{T} R^{-1}(\mathbf{x}_t - C\mathbf{y}_t)$$
$$To\ solve\ for\ R_{new}\ in\ the\ maximization\ step\ of\ the\ EM\ algorithm,$$
$$R_{new} = \arg\max_R < l >_q$$
$$= \arg\max_R < \sum_{t=1}^{T} -\tfrac{1}{2}\log 2\pi - \tfrac{1}{2}\log|R| - \tfrac{1}{2}(\mathbf{x}_t - C\mathbf{y}_t)^\mathsf{T} R^{-1}(\mathbf{x}_t - C\mathbf{y}_t) >_q$$
$$= \arg\max_R < -\tfrac{T}{2}\log 2\pi - \tfrac{T}{2}\log|R| - \tfrac{1}{2}\sum_{t=1}^{T}(\mathbf{x}_t - C\mathbf{y}_t)^\mathsf{T} R^{-1}(\mathbf{x}_t - C\mathbf{y}_t) >_q$$
$$= \arg\max_R < -\tfrac{T}{2}\log 2\pi|R| - \tfrac{1}{2}\sum_{t=1}^{T}\left(\mathbf{x}_t^\mathsf{T} R^{-1} - \mathbf{y}_t^\mathsf{T} C^\mathsf{T} R^{-1}\right)(\mathbf{x}_t - C\mathbf{y}_t) >_q$$
$$= \arg\max_R < -\tfrac{T}{2}\log 2\pi|R| - \tfrac{1}{2}\sum_{t=1}^{T}\left(\mathbf{x}_t^\mathsf{T} R^{-1}\mathbf{x}_t - \mathbf{x}_t^\mathsf{T} R^{-1}C\mathbf{y}_t - \mathbf{y}_t^\mathsf{T} C^\mathsf{T} R^{-1}\mathbf{x}_t + \mathbf{y}_t^\mathsf{T} C^\mathsf{T} R^{-1}C\mathbf{y}_t\right) >_q$$
$$= \arg\max_R\{-\tfrac{T}{2}\log 2\pi|R| - \tfrac{1}{2}\sum_{t=1}^{T}(\mathbf{x}_t^\mathsf{T} R^{-1}\mathbf{x}_t - \mathbf{x}_t^\mathsf{T} R^{-1}C < \mathbf{y}_t >$$
$$- < \mathbf{y}_t^\mathsf{T} > C^\mathsf{T} R^{-1}\mathbf{x}_t + < \mathbf{y}_t^\mathsf{T} C^\mathsf{T} R^{-1}C\mathbf{y}_t >)\}$$
$$= \arg\max_R\{-\tfrac{T}{2}\log 2\pi|R| - \tfrac{1}{2}\mathrm{Tr}[R^{-1}\sum_{t=1}^{T}(\mathbf{x}_t^\mathsf{T}\mathbf{x}_t - \mathbf{x}_t^\mathsf{T} C < \mathbf{y}_t >$$
$$- < \mathbf{y}_t^\mathsf{T} > C^\mathsf{T}\mathbf{x}_t + < \mathbf{y}_t^\mathsf{T} C^\mathsf{T} C\mathbf{y}_t >)]\}$$

$$Due\ to\ the\ cyclic\ property\ of\ traces,$$
$$= \arg\max_R\{-\tfrac{T}{2}\log 2\pi|R| - \tfrac{1}{2}\mathrm{Tr}[R^{-1}\sum_{t=1}^{T}(\mathbf{x}_t\mathbf{x}_t^\mathsf{T} - C < \mathbf{y}_t > \mathbf{x}_t^\mathsf{T}$$
$$-\mathbf{x}_t < \mathbf{y}_t^\mathsf{T} > C^\mathsf{T} + C < \mathbf{y_t}\mathbf{y}_t^\mathsf{T} > C^\mathsf{T})]\}$$
$$Using\ \frac{\partial \mathrm{Tr}[AB]}{\partial A} = B^\mathsf{T}\ and\ taking\ the\ derivative,$$
$$= \tfrac{T}{2}R - \tfrac{1}{2}\sum_{t=1}^{T}\left(\mathbf{x}_t\mathbf{x}_t^\mathsf{T} - \mathbf{x}_t < \mathbf{y}_t^\mathsf{T} > C^\mathsf{T} - C < \mathbf{y}_t > \mathbf{x}_t^\mathsf{T} + C < \mathbf{y_t}\mathbf{y}_t^\mathsf{T} > C^\mathsf{T}\right) = 0$$
$$Given\ that\ C = \left(\sum_{t=1}^{T}\mathbf{x}_t < \mathbf{y}_t^\mathsf{T} >\right)\left(\sum_{t=1}^{T} < \mathbf{y_t}\mathbf{y}_t^\mathsf{T} >\right)^{-1},\ it\ is\ substituted\ in\ the\ expression$$
$$= \tfrac{T}{2}R - \tfrac{1}{2}\sum_{t=1}^{T}\left(\mathbf{x}_t\mathbf{x}_t^\mathsf{T} - \mathbf{x}_t < \mathbf{y}_t^\mathsf{T} > C^\mathsf{T} - C < \mathbf{y}_t > \mathbf{x}_t^\mathsf{T} + C < \mathbf{y_t}\mathbf{y}_t^\mathsf{T} > \left(< \mathbf{y_t}\mathbf{y}_t^\mathsf{T} >\right)^{-1} < \mathbf{y}_t > \mathbf{x}_t^\mathsf{T}\right)$$
$$= \tfrac{T}{2}R - \tfrac{1}{2}\sum_{t=1}^{T}\left(\mathbf{x}_t\mathbf{x}_t^\mathsf{T} - \mathbf{x}_t < \mathbf{y}_t^\mathsf{T} > C^\mathsf{T} - C < \mathbf{y}_t > \mathbf{x}_t^\mathsf{T} + C < \mathbf{y}_t > \mathbf{x}_t^\mathsf{T}\right)$$
$$= \tfrac{T}{2}R - \tfrac{1}{2}\sum_{t=1}^{T}\left(\mathbf{x}_t\mathbf{x}_t^\mathsf{T} - \mathbf{x}_t < \mathbf{y}_t^\mathsf{T} > C^\mathsf{T}\right) = 0$$
$$\tfrac{T}{2}R = \tfrac{1}{2}\sum_{t=1}^{T}\left(\mathbf{x}_t\mathbf{x}_t^\mathsf{T} - \mathbf{x}_t < \mathbf{y}_t^\mathsf{T} > C^\mathsf{T}\right)$$
$$R_{new} = \tfrac{1}{T}\left[\sum_{t=1}^{T}\mathbf{x}_t\mathbf{x}_t^\mathsf{T} - \left(\sum_{\mathbf{t=1}}^{\mathbf{T}}\mathbf{x}_t < \mathbf{y}_t^\mathsf{T} >\right)C_{new}^\mathsf{T}\right]$$

$The\ log-likelihood\ can\ be\ written$

$l = \sum_{t=1}^{T-1} \log p(\mathbf{y}_{t+1}|\mathbf{y}_t)$

$= \sum_{t=1}^{T-1} -\frac{1}{2}\log 2\pi - \frac{1}{2}\log|Q| - \frac{1}{2}(\mathbf{y}_{t+1} - A\mathbf{y}_t)^\top Q^{-1}(\mathbf{y}_{t+1} - A\mathbf{y}_t)$

$To\ solve\ for\ Q_{new}\ in\ the\ maximization\ step\ of\ the\ EM\ algorithm,$

$Q_{new} = \arg\max_Q < l >_q$

$= \arg\max_Q < \sum_{t=1}^{T-1} -\frac{1}{2}\log 2\pi - \frac{1}{2}\log|Q| - \frac{1}{2}(\mathbf{y}_{t+1} - A\mathbf{y}_t)^\top Q^{-1}(\mathbf{y}_{t+1} - A\mathbf{y}_t) >_q$

$= \arg\max_Q < -\frac{T-1}{2}\log 2\pi - \frac{T-1}{2}\log|Q| - \frac{1}{2}\sum_{t=1}^{T}(\mathbf{y}_{t+1} - A\mathbf{y}_t)^\top Q^{-1}(\mathbf{y}_{t+1} - A\mathbf{y}_t) >_q$

$= \arg\max_Q < -\frac{T-1}{2}\log 2\pi|Q| - \frac{1}{2}\sum_{t=1}^{T-1}\left(\mathbf{y}_{t+1}^\top Q^{-1} - \mathbf{y}_t^\top A^\top Q^{-1}\right)(\mathbf{y}_{t+1} - A\mathbf{y}_t) >_q$

$= \arg\max_Q < -\frac{T-1}{2}\log 2\pi|Q| - \frac{1}{2}\sum_{t=1}^{T-1}(\mathbf{y}_{t+1}{}^\top Q^{-1}\mathbf{y}_{t+1} - \mathbf{y}_{t+1}{}^\top Q^{-1}A\mathbf{y}_t$

$-\mathbf{y}_t^\top A^\top Q^{-1}\mathbf{y}_{t+1} + \mathbf{y}_t^\top A^\top Q^{-1}A\mathbf{y}_t) >_q$

$= \arg\max_Q\{-\frac{T-1}{2}\log 2\pi|Q| - \frac{1}{2}\sum_{t=1}^{T-1}(< \mathbf{y}_{t+1}{}^\top Q^{-1}\mathbf{y}_{t+1} > - < \mathbf{y}_{t+1}{}^\top Q^{-1}A\mathbf{y}_t >$

$- < \mathbf{y}_t^\top A^\top Q^{-1}\mathbf{y}_{t+1} > + < \mathbf{y}_t^\top A^\top Q^{-1}A\mathbf{y}_t >)\}$

$= \arg\max_Q\{-\frac{T-1}{2}\log 2\pi|Q| - \frac{1}{2}\mathrm{Tr}[Q^{-1}\sum_{t=1}^{T-1}(< \mathbf{y}_{t+1}{}^\top \mathbf{y}_{t+1} > - < \mathbf{y}_{t+1}{}^\top A\mathbf{y}_t >$

$- < \mathbf{y}_t^\top A^\top \mathbf{y}_{t+1} > + < \mathbf{y}_t^\top A^\top A\mathbf{y}_t >)]\}$

$Due\ to\ the\ cyclic\ property\ of\ traces,$

$= \arg\max_Q\{-\frac{T-1}{2}\log 2\pi|Q| - \frac{1}{2}\mathrm{Tr}[Q^{-1}\sum_{t=1}^{T-1}(< \mathbf{y}_{t+1}\mathbf{y}_{t+1}{}^\top > -A < \mathbf{y_t}\mathbf{y}_{t+1}{}^\top >$

$- < \mathbf{y}_{t+1}\mathbf{y}_t^\top > A^\top + A < \mathbf{y}_t\mathbf{y}_t^\top > A^\top)]\}$

$Using\ \frac{\partial \mathrm{Tr}[AB]}{\partial A} = B^\top\ and\ taking\ the\ derivative,$

$= \frac{T-1}{2}Q - \frac{1}{2}\sum_{t=1}^{T-1}(< \mathbf{y}_{t+1}\mathbf{y}_{t+1}{}^\top > - < \mathbf{y_{t+1}}\mathbf{y}_t{}^\top > A^\top - A < \mathbf{y}_t\mathbf{y}_{t+1}^\top > +A < \mathbf{y}_t\mathbf{y}_t^\top > A^\top = 0$

$Given\ that\ A = \left(\sum_{t=1}^{T-1} < \mathbf{y_{t+1}}\mathbf{y}_t^\top >\right)\left(\sum_{t=1}^{T-1} < \mathbf{y_t}\mathbf{y}_t^\top >\right)^{-1},\ it\ is\ substituted\ in\ the\ expression,$

$= \frac{T-1}{2}Q - \frac{1}{2}\sum_{t=1}^{T-1}(< \mathbf{y}_{t+1}\mathbf{y}_{t+1}{}^\top > - < \mathbf{y_{t+1}}\mathbf{y}_t{}^\top > A^\top - A < \mathbf{y}_t\mathbf{y}_{t+1}^\top >$

$+A < \mathbf{y}_t\mathbf{y}_t^\top > \left(< \mathbf{y}_t\mathbf{y}_t^\top >\right)^{-1} < \mathbf{y}_t\mathbf{y}_{t+1}^\top >)$

$= \frac{T-1}{2}Q - \frac{1}{2}\sum_{t=1}^{T-1}\left(< \mathbf{y}_{t+1}\mathbf{y}_{t+1}^\top > - < \mathbf{y_{t+1}}\mathbf{y}_t^\top > A^\top - A < \mathbf{y}_t\mathbf{y}_{t+1}^\top > +A < \mathbf{y}_t\mathbf{y}_{t+1}^\top >\right)$

$= \frac{T-1}{2}Q - \frac{1}{2}\sum_{t=1}^{T-1}\left(< \mathbf{y}_{t+1}\mathbf{y}_{t+1}^\top > - < \mathbf{y_{t+1}}\mathbf{y}_t^\top > A^\top\right) = 0$

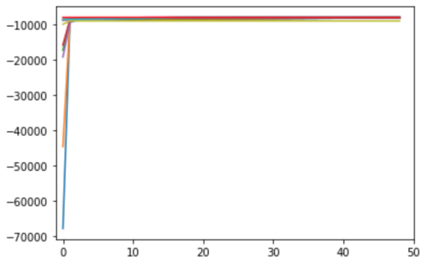
$\frac{T-1}{2}Q = \frac{1}{2}\sum_{t=1}^{T-1}\left(< \mathbf{y}_{t+1}\mathbf{y}_{t+1}^\top > - < \mathbf{y_{t+1}}\mathbf{y}_t^\top > A^\top\right)$

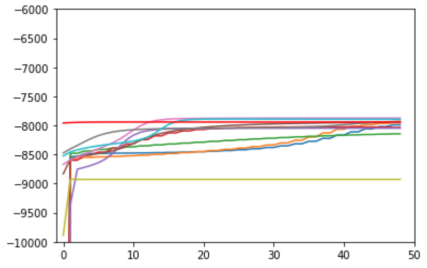
$Q_{new} = \frac{1}{T-1}\left[\sum_{t=1}^{T-1} < \mathbf{y}_{t+1}\mathbf{y}_{t+1}^\top > - \left(\sum_{t=1}^{T-1} < \mathbf{y_{t+1}}\mathbf{y}_t^\top >\right)A_{new}^\top\right]$

$which\ is\ equivalent\ to$

$\frac{1}{T-1}\left[\sum_{t=2}^{T} < \mathbf{y}_t\mathbf{y}_t^\top > - \left(\sum_{t=2}^{T} < \mathbf{y_t}\mathbf{y}_{t-1}^\top >\right)A_{new}^\top\right]$

The log-likelihoods of the initialization with the generating parameters (in bright red) and with 10 random initializations are plotted on the same set of axes below, at different axes scales to better observe their features.

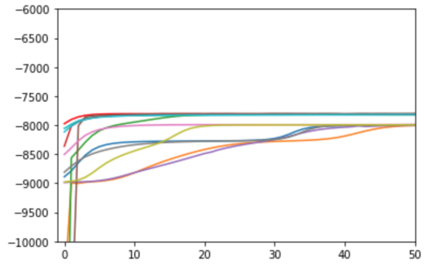




The likelihood of the plot with the initial features converges very quickly, with an initial likelihood close to the converging value, near -8000. The EM does not terminate immediately because it continuously updates based on the data provided and adjusts slightly to fit the data. The likelihoods of the randomly initialized parameters converge to different likelihood values, with varying rates. The initial likelihoods of some randomly-initialized values are very small, but all likelihoods rapidly converge to likelihood values near that of the generated parameter EM, indicating that the EM algorithm rapidly optimizes the likelihood even for significantly nonoptimal initial parameters. The convergence likelihoods of the randomly-initialized iterations differ suggesting that the algorithm may occasionally optimize for a local maximum.

c.

The log-likelihoods of the test set initialized with the generating parameters (in bright red), the test set with the EM initialized at the true parameters (cyan), and the test set with the EM initialized at random parameters are plotted on the same set of axes below.



Unlike the random initializations on the training data having a higher variance of convergence values, the test data converges to two distinct values. The likelihoods of the true parameter initializations converge around a particular likelihood (≈-7820), and the majority of the random initializations converge around a different likelihood (≈-8000), with a few exceptions. It appears that the random initialization EM parameters do not always reach the optimal solution, perhaps overfitting the data and thus do not perform as well as the generating parameter initialization or generating parameter EM on the test set.