Name: Phuc H. Lam

1. Explanation

- Detect overfitting: I originally split the data into a train set and a validation set, then trained the parameters using the train set only. However, this method is not sensitive to the number of clusters: for dataset A, $K \geq 60$ for dataset A still gives increasing log likelihood for the validation set. Thus, this method is not suitable for detecting overfitting. For problems that include log likelihood, either AIC (Akaike Information Criterion) or BIC (Bayesian information criterion) works. Here, we stick to BIC. What we do is that we calculate the BIC score as follows:

  $$\boxed{BIC \text{ score } = (\text{num. of parameters}) \times \log(\text{num. of data points}) - 2\log(\text{max Loss})}$$

  .

  Here, the number of parameters is $\underbrace{K * d}_{\text{mean}} + \underbrace{K * d(d+1)/2}_{\text{covariance}} + \underbrace{(K-1)}_{\text{weights}}$.

  The value of K that we choose is the one right before the BIC scores "stabilize".
- To use the program: in *main.py*, the user sets the values for K and $iter$ on lines 17 and 18. To change how the parameters are initialized, go to *EMprogram.py* and modify the function BIC_score.
- Warning: sometimes, the error "ValueError: array must not contain infs or NaNs" will appear randomly. This is not to be worried about; this error appears due to the term $\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \mu_k, \mathbf{\Sigma}_k)$ being too close to 0 such that its log is tending to $-\infty$. When $K > 1$ is small, this error appears more frequently due to the randomized initial means are too far away from the data points. When K is too big, this error also appears more frequently. This is due to the initial weights of those some summands $\mathcal{N}(\mathbf{x}_n \mid \mu_k, \mathbf{\Sigma}_k)$ in the above being too small ($\frac{1}{K}$), and the remaining summands are too close to 0 (again, due to the randomized initial means are too far away from the data points. In these cases, we shall rerun the program until we succeed. If the program fails, it seems to fail after at most 1 iteration (this means if it runs well for 5 iterations, we can be sure that it is working). For example, for dataset A, with $K = 7$ and $iter = 50$, it takes as much as 40 times to rerun the program until the randomized initial parameters finally work.

2. Results

There are cases for which, when we look at the loss plotted, we realize that the parameters are stuck at a local maxima, in which case we rerun the program. All the log likelihood for each K in each dataset can be found in the folder "Results".

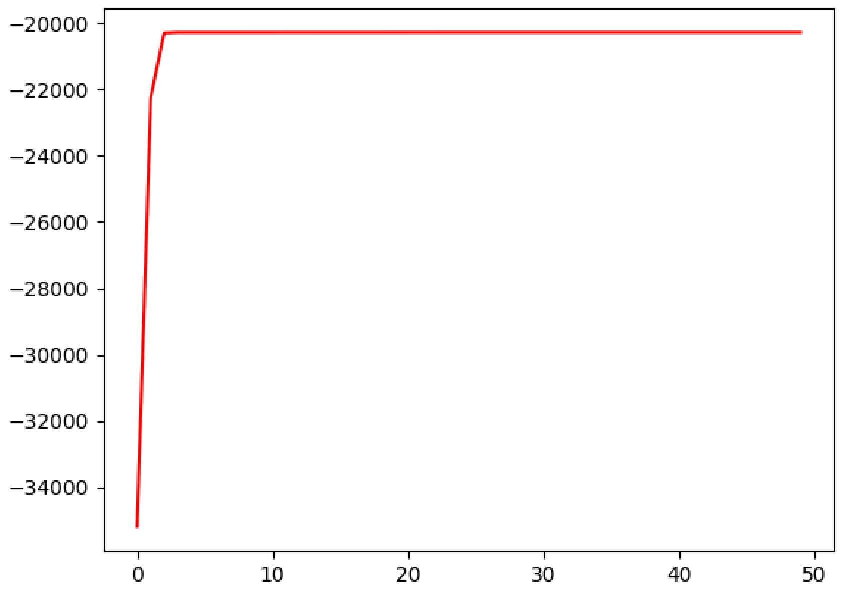
---

a. Dataset A:

Here is a table of BIC scores (prone to change depending on how the parameters are initialized), with $iter = 50$. It is impossible to run the program for $K = 2$.

K	**BIC scores**
3	103355.58190376672
4	45976.64886160516
5	40736.13131391161
6	13040.294770827344
7	10439.062578778217
8	10720.454329130722
9	8411.672834550594
10	7572.128697402681

We see that there is a huge jump in BIC score between $K = 5$ and $K = 6$, and the BIC score does not change significantly afterwards, i.e. there is not much gain in increasing the number of clusters. Thus, we estimate $\boxed{K = 5}$. The plot for the log likelihood is as follows.



---

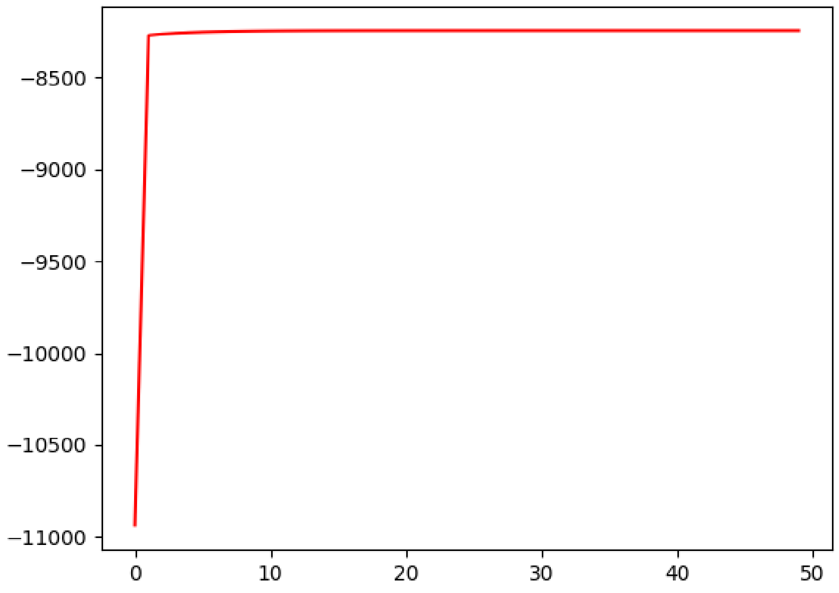
b. Dataset B:

Here is a table of BIC scores (prone to change depending on how the parameters are initialized), with $iter = 50$. For this dataset, I slightly modified how I choose my initial means (due to randomized means giving ValueError almost consistently). The modification is made in file *EMprogram.py*. Here, the initial means are chosen from the data points (without replacement), and then ran through K-means for a number of iterations.

It is impossible to run the program for $K = 2$ and $K = 3$, even after having initialized the means using K-means and initializing the covariances to different scalar multiplications of the identity matrix. During one of the tests for $K = 3$, the initialized means are $[[-2.64026118, -0.27564658], [-1.43058243, -1.853747], [0.34297784, -1.19167775]]$, which is quite close to the centers of the (visual) clusters that we see when plotting the dataset. Yet, the program still returns ValueError.

K	**BIC scores**
4	16813.603530144894
5	16659.827417841258
6	8141.79742838447
7	5434.3337305833
8	4681.167751171882
9	6472.055624040349
10	4301.77144336708

We see that there is a huge jump in BIC score between $K = 5$ and $K = 6$, and the BIC score does not change significantly afterwards, i.e. there is not much gain in increasing the number of clusters. Thus, we estimate $\boxed{K = 5}$. The plot for the log likelihood is as follows.



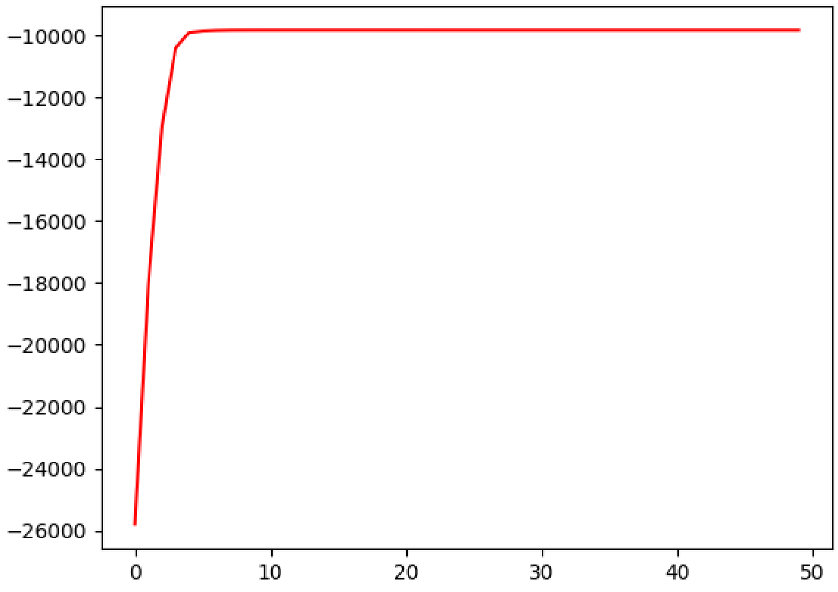
---

c. Dataset C:

Here is a table of BIC scores (prone to change depending on how the parameters are initialized), with $iter = 50$. Again, it is impossible to run the program for $K = 2$.

K	**BIC scores**
3	37561.6972296452
4	21887.716835491945
5	21454.258991158407
6	21444.2748905924
7	19871.973039570606
8	4743.819027394598
9	3602.732765678583
10	3827.313060855648
11	2349.4136886006636
12	1977.6589441466733

We see that there is a huge jump in BIC score between $K = 7$ and $K = 8$, and the BIC score does not change significantly afterwards, i.e. there is not much gain in increasing the number of clusters. Thus, we estimate $\boxed{K = 7}$. The plot for the log likelihood is as follows.



---

d. Dataset Z:

I was not able to run my code on this dataset. I attempted to make a better initialization of parameters by using K-mean to find the means, and setting the initial covariances to scalar multiples of the identity matrix. I also attempted to avoid the NaNs errors by catching the cases where both the numerator and denominator of gamma in the E-step are 0. In these cases, I catch the case where the denominator is 0, and only divide if it is nonzero. In the M-step, I also caught the case where N_k might be 0, and made the same adjustment. However, at some point, that made the weights (pi) all zeros, which I have yet to figure out how to resolve this problem.