Implementation:

For both k-means and GMM, I use numpy array to store all the points.

1)For K-means, I first initialize 3 centroids randomly, then 1) compute the distance of centroids to each data point, assignment points to their minimum distance cluster and recording their belonged cluster, 2) re-compute the centroids for each cluster. Repeat this two steps until no more data points change its cluster.

1)For the GMM initialization, I use the clustering result of K-means. For example, If one point i is belong to one cluster j in K-means, then in GMM, I initialize Ric[i][j]=1, otherwise, Ric[i][j]=0; using Ric matrix to calculate ∏ . Then 1)compute each Uc and ∑c using Ric and ∏. 2)calculate probabilities for each data points in all k Gaussian functions, and recompute the Ric and ∏, assign each point into its new cluster which has maximum Ric[i][j],{j∈0-k}. repeat this 2 steps.

The challenges

1) The end condition in GMM I set is: if least than 5 data points change their clusters, I will end the loop. I do this because if I use Maximum Likelihood, sometimes I will meet the single matrix error because the covariance matrix will contain numbers much smaller than others. This problem is equal to covariance matrix contain 0, so there won’t exist inv(∑). This is the largest challenge I meet. So I change the end condition.

2) how to improve my code:

the library GMM and K-means functions are faster than mine. So I think instead of using numpy, I should use a better data structure like pandas in python, it’s quicker and easier for the matrix processing. It will decrease the number of circulations efficiently.