Document "data.py" is to generate GMM-based datasets with varying factors such as cluster number, sample size, dimension, and etc. Document "Kmeans.py" is to prepare for question 2. Document "KmeansCL.py" is code for question 2. Document "model selection of GMM.py" is code for question 3. They can generate corresponding pictures in the "figures" folder for latex.

1 K-means vs GMM

1.1 Variant of K-means

K-means can be regarded as a special case of GMM. The main superiority of GMM over K-means is the extra consideration of mixing weights, covariance matrix, and soft assignments, which also introduces high computation complexity.

Here a variant of K-means is put up by considering soft assignment instead of 1-of-K assignment, that is, in

$$J = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|\boldsymbol{x_n} - \boldsymbol{\mu_k}\|^2, \tag{1}$$

r_{nk} is not anymore a hard assignment of either 0 or 1, but a variable in $[0, 1]$ representing probability of point x_n belonging to cluster μ_k.

In this way, we might determine r_{nk} and $\boldsymbol{\mu_k}$ respectively:

1. Determination of r_{nk}. The terms involving different n are independent and so we can optimize for each n separately as follows:

$$r_{nk} = \frac{1}{1 + e^{\|\boldsymbol{x_n} - \boldsymbol{\mu_k}\|^2}}. \tag{2}$$

2. Optimization of $\boldsymbol{\mu_k}$ with r_{nk} held fixed. The object function J is a quadratic function of μ_k, and it can be minimized by setting its derivative w.r.t. μ_k to zero giving

$$2\sum_{n=1}^{N} r_{nk}(\boldsymbol{x_n} - \boldsymbol{\mu_k}) = 0, \tag{3}$$

which we can easily solve for μ_k to give

$$\boldsymbol{\mu_k} = \frac{\sum_n r_{nk}\boldsymbol{x_n}}{\sum_n r_{nk}}. \tag{4}$$

To summarize, the overall algorithm is shown in Alg. 1.

Algorithm 1: A variant of K-means

Input: Number of clusters K.
Output: r_{nk}, $\boldsymbol{\mu_k}(k = 1, 2, \cdots, K)$.

Initialize $\boldsymbol{\mu_k}, k = 1, 2, \cdots, K$, and evaluate the initial value of object function J;
while *the convergence condition of* $\boldsymbol{\mu_k}$ *is not satisfied* **do**
  E-step. Evaluate the responsibilities using the current parameter values

$$r_{nk} = \frac{1}{1 + e^{\|\boldsymbol{x_n} - \boldsymbol{\mu_k}\|^2}}.$$

  M-step. Re-estimate the parameters using the current responsibilities

$$\mu_k = \frac{\sum_n r_{nk} \boldsymbol{x_n}}{\sum_n r_{nk}}.$$

return r_{nk}, $\boldsymbol{\mu_k}(k = 1, 2, \cdots, K)$.

1.2 Discussion

Introduction of the probability-based soft assignment makes the variant more robust than K-means algorithm, while not increasing the computation complexity like GMM. However, this variant inevitably encounters the problems as sensitivity of initialization and inability to find global optimal as well as determine the best K.

2 K-means vs CL

2.1 Comparison of K-means and CL

Similarities between K-means and CL are as follows:

- Both K-means and CL are sensitive to initial values.
- They only consider distance metric rather than probability.
- They cannot determine the number of clusters from data.

Distances between K-means and CL are as follows:

- K-means is a batch-learning algorithm while CL is an online-learning algorithm.
- The update procedure of parameters $\boldsymbol{\mu_k}$ in K-means is to calculate the means without hyper parameters while in CL is to update with learning rate parameters.

- K-means has higher computation complexity than CL since each iteration of K-means considers a batch of data while CL updates $\boldsymbol{\mu_k}$ with one point a time.

2.2 Introduction of RPCL'idea into K-means

K-means algorithm has a disadvantage that the number of clusters cannot be estimated from data while RPCL can penalize the rival centers to control the cluster number. Thus, I slightly change the details of K-means with introduction of the idea to penalize the rivals. This new algorithm is simple and shown in Alg. 2

Algorithm 2: A variant of K-means with introduction of RPCL'idea

Input: Max number of clusters K.
Output: r_{nk}, $\boldsymbol{\mu_k}(k = 1, 2, \cdots, K)$.

1 Initialize $\boldsymbol{\mu_k}, k = 1, 2, \cdots, K$, and evaluate the initial value of object function J;
2 **for** $i \leq max\ iteration$ **do**
3 **for** *data in datasets* **do**
4 Calculate the distances towards all cluster centers;
5 Update r_{nk} by

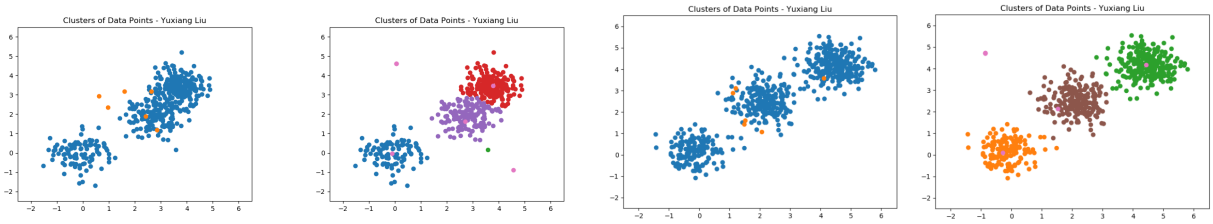
$$r_{nk} = \begin{cases} 1, & \text{if center k is the winner} \\ -\gamma, & \text{if center k is the rival} \\ 0 & \text{otherwise} \end{cases};$$

6 Update the coordinates of the winner center and the rival center according to

$$\boldsymbol{\mu_k}^{new} = \boldsymbol{\mu_k}^{old} + r_{nk} * rate * (\boldsymbol{x_n} - \boldsymbol{\mu_k}^{old});$$

7 **return** r_{nk}, $\boldsymbol{\mu_k}(k = 1, 2, \cdots, K)$.

I test this algorithm in a three-cluster dataset generated by myself based on GMM with adjusted parameter settings and the results are as in Fig. 7. Though in most situations this algorithm can find a pretty good k, it might perform similar to K-means sometimes.



From the pictures, we can find that applying the idea of RPCL to K-means helps to determine the k with pre-adjusted parameters.

3 Model selection of GMM

In this section, experimental comparisons on model selestion performance between AIC, BIC, and VBEM are presented and 'discussed.

3.1 Sensetivity on Sample Size

First I randomly generate 3-cluster datasets based on GMM by varying sample size from 30, 50, 80, 120 to 180. Then models as AIC, BIC, and VBEM are utilized to evaluate the performance. Classification results are shown as follows w.r.t. models and sample sizes.

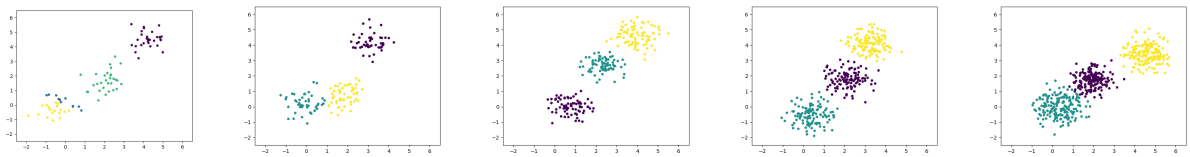


Figure 1: AIC with sample size in (30, 50, 80, 120, 180)

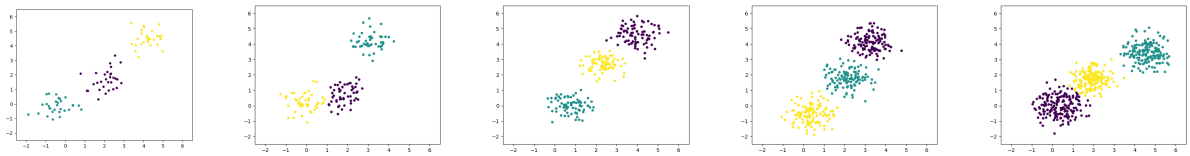


Figure 2: BIC with sample size in (30, 50, 80, 120, 180)

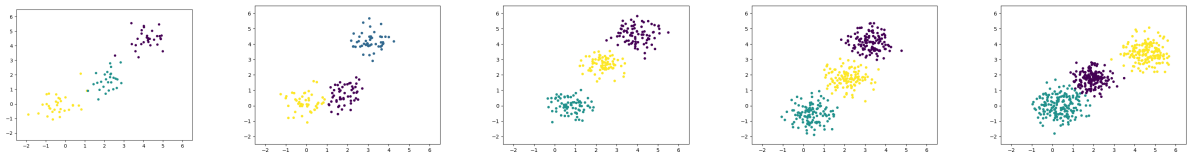


Figure 3: VBEM with sample size in (30, 50, 80, 120, 180)

From the comparisons of Fig. 1 with Fig. 2 and Fig. 3, we get that when the dataset is large, their performances are approximately equal. When dataset is as small as 30 it is obvious that BIC and VBEM perform better than AIC and this is not an accidental event because after lots of trainings, the results are almost the same. It can be inferred from that BIC considers sample size N in its object function while AIC not. With many times of experiments, their performance differences vary as well and AIC is not always worse than the others.

3.2 Sensetivity on Cluster Number

Number of clusters also influences the model performance. For this part, I generate 1, 2, 3, 4, and 5 clusters and apply AIC, BIC, and VBEM on them, respectively, as shown in Fig. 4, Fig. 5, and Fig. 6.

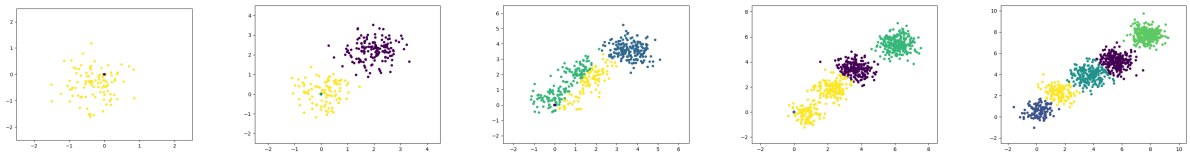


Figure 4: AIC with cluster number in (1, 2, 3, 4, 5)

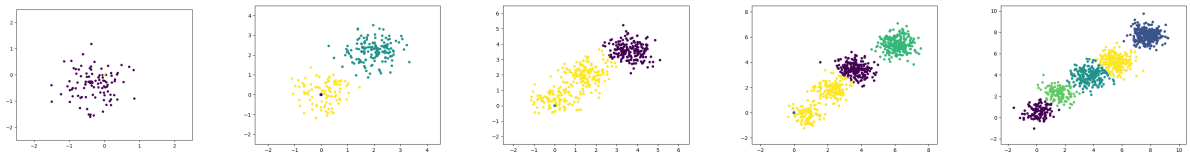


Figure 5: BIC with cluster number in (1, 2, 3, 4, 5)

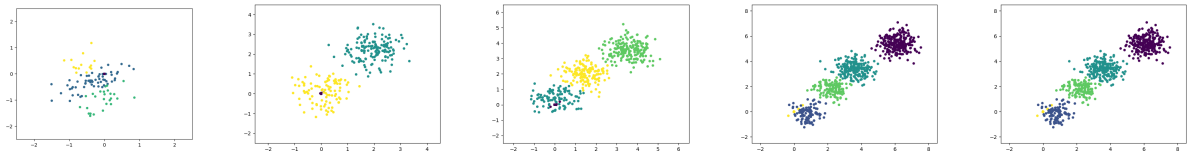


Figure 6: VBEM with cluster number in (1, 2, 3, 4, 5)

From the figures, we can find that when the number of cluster is small such as 1, three models cannot well get that and always classify them into two clusters or more. Besides, after many times experiments, with the number gets larger, AIC, BIC perform almost the same, which consists with Section 3.1 that when the sample size is not that small (here sample size for each cluster of GMM is not equal but are all at least 100), there is almost no difference between AIC and BIC in our experiment settings.