CS420 Homework 1

CS420-Machine Learning , Shikui Tu, Spring 2019.

Name: Jiayi Xu Student ID: 516021910396 Email: xujiayi925@sjtu.edu.cn

1 K-means vs GMM

In this section, I will give a variant of K-means algorithm between the original K-means and EM for GMM, and discuss the advantages and limitations.

Based on what I know, there are altogether 3 distinct differences between k-means and GMM:

1. K-means use a vector μ_k to represent the k-th cluster; GMM use a Gaussian distribution given by parameters μ_k and Σ_k to represent the k-th cluster.
2. K-means adopts distance framework, so its optimization objective is to minimize sum of square distance; GMM adopts probability framework, so its optimization objective is to maximize likelihood.
3. Based on the second difference, K-means assigns one point to one certain cluster while GMM assigns points to clusters softly. Therefore, K-means is a hard EM, and GMM is a soft EM.

To give a variant, we can start with these three differences. From my point of view, one of the biggest problems with K-means is its hard assignment, and its biggest advantage is easy and fast. Therefore, my variant simply changes the hard assignment to soft assignment.

The computational details can be seen in the pseudo-code Alg. 1.

Algorithm 1: A variant of K-means

Input : Data points $\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_N}$; The number of clusters K; Initial parameter values r_{nk} $(n = 1, 2, \cdots, N, k = 1, 2, \cdots, K)$

Output: Cluster centers $\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \cdots, \boldsymbol{\mu_K}$

Initialize the cluster centers $\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \cdots, \boldsymbol{\mu_K}$;

while *the assignment for data points to clusters changes* **do**

E step. Evaluate the responsibilities using the current parameter values:

$$r_{nk} = \frac{1/||\boldsymbol{x_n} - \boldsymbol{\mu_k}||^2}{\sum_{i=1}^{N} 1/||\boldsymbol{x_n} - \boldsymbol{\mu_i}||^2}$$

M step. Re-estimate the parameters with the current responsibilities:

$$\boldsymbol{\mu_k} = \frac{\sum_{i=1}^{N} r_{nk}\boldsymbol{x_n}}{\sum_{i=1}^{N} r_{nk}}$$

return $\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \cdots, \boldsymbol{\mu_K}$;

Similar to K-means, this variant is easy to implement, fast to converge, explainable and has only one parameter k to tune. Moreover, it may be more robust than the original K-means since this variant uses soft assignment to update the cluster centers. Its limits are obvious: sensitive to outliers; hard to choose K; cannot guarantee global optimum; can only find globular clusters. The limit "hard to choose K" can be solved by introducing the idea of RPCL. The limit "can only find globular clusters" can be solved by changing the distance measure from Euler distance to Mahalanobis distance.

2 K-means vs CL

2.1 Comparison between K-means and CL

- Differences

    1. K-means is for clustering a batch of data points (batch learning); CL is for clustering data points come one by one (adaptive learning / online learning).
    2. Since CL updates one cluster center according to one data point each time, K-means updates cluster centers according to all data points each iteration, CL requires less computing resources (CPU, memory and time).
    3. CL has a hyper parameter η to adjust while K-means does not. Therefore, K-means is simpler and more explainable.

- Similarities

    1. They both cannot determine the number of clusters K from data points.
    2. They both cannot guarantee global optimum and their performance both highly depend on initialization.
    3. They both adopt distance framework instead of probability framework.

2.2 Competitive K-means

One of the biggest drawbacks of K-means is that it cannot automatically determine the number of clusters K from data points. When confronting extra centers, K-means tend to assign more than one center to one cluster. RPCL (Rival Penalized Competitive Learning) can solve this problem by making extra centers far away. So, we can adopt the idea of RPCL to improve K-means in terms of automatically determining the number of clusters.

My algorithm adopts the idea of RPCL to make rival cluster centers far away. In original K-means, the extra centers tend to be close to some other true center. Therefore, in each iteration, if the true center can "kick" the extra center away a little bit, the extra center will bring less disturbance next time. When it comes to convergence, the extra centers will be far away from the dataset.

Suppose there are K cluster centers in the initial state, with one step of EM algorithm, we update the position of all cluster centers. Then we exam the distances between each pair of centers. We can assume that the extra center occupies part of the data points in another cluster. Therefore, we select a pair of centers which has shortest distance and kick the center with fewer data points away to some ratio. The pseudo-code of my K-means algorithm with the idea of RPCL can be seen

as Alg. 2.

Algorithm 2: K-means with the idea of RPCL

Input : Data points $\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_N}$; Penalization term η

Output: Cluster centers $\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \cdots, \boldsymbol{\mu_K}$

Initialize the number of cluster centers K;

Initialize the cluster centers $\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \cdots, \boldsymbol{\mu_K}$;

while *the assignment for data points to clusters changes* **do**

E step. Evaluate the responsibilities using the current parameter values:

$$r_{nk} = \begin{cases} 1 & k = \arg\min_j ||\boldsymbol{x_n} - \boldsymbol{\mu_k}||^2 \\ 0 & otherwise \end{cases}$$

M step. Re-estimate the parameters with the current responsibilities:

$$\boldsymbol{\mu_k} = \frac{\sum_{i=1}^{N} r_{nk} \boldsymbol{x_n}}{\sum_{i=1}^{N} r_{nk}}$$

RP step. Penalize the extra cluster centers:
find the two centers with smallest distance, $\boldsymbol{\mu_p}$, $\boldsymbol{\mu_q}$.

if $\sum_{i=1}^{N} r_{ip} >= \sum_{i=1}^{N} r_{iq}$ **then**

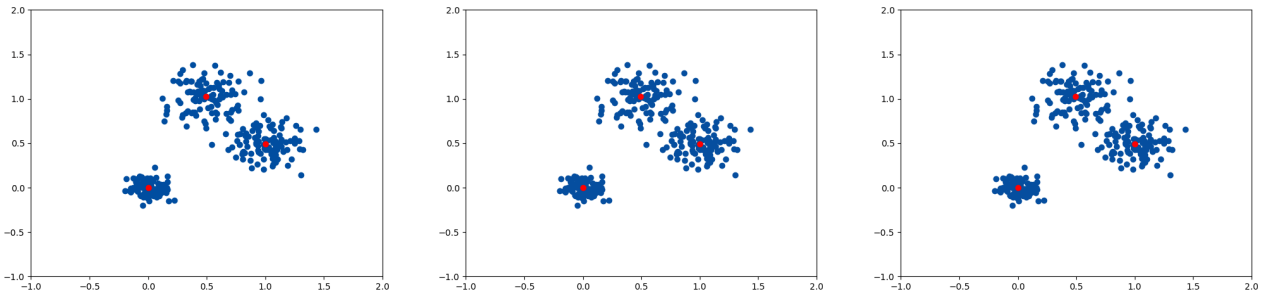
$$\boldsymbol{\mu_q} = \boldsymbol{\mu_q} + \eta * (\boldsymbol{\mu_q} - \boldsymbol{\mu_p})$$

else

$$\boldsymbol{\mu_p} = \boldsymbol{\mu_p} + \eta * (\boldsymbol{\mu_p} - \boldsymbol{\mu_q})$$

return $\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \cdots, \boldsymbol{\mu_K}$;

This K-means algorithm with the idea of RPCL is implemented on a three-cluster dataset generated by myself. The dataset is generated from Gaussian distribution with random sigma between 0.05 and 0.2 and three fixed centers (0,0), (0.5,1), (1,0.5). I initialize the number of clusters to be 4, 5, 6, and the centers' positions are initialized randomly. The clustering results are shown as Fig. 1 and Tab. 1.



(a) Result of 4 initialized centers (b) Result of 5 initialized centers (c) Result of 6 initialized centers

Figure 1: The results of K-means with the idea of RPCL. Blue points denotes data and red points denotes cluster centers.

	4		5		6	
	coordinate	# points	coordinate	# points	coordinate	# points
center 1	(4.9e-1,1.0e0)	101	(1.0e0,4.9e-1)	99	(1.0e0,4.9e-1)	99
center 2	(4.5e-3,5.3e-5)	100	(4.9e-1,1.0e0)	101	(4.9e-1,1.0e0)	101
center 3	(-1.0e2,-4.9e1)	0	(1.1e2,2.7e2)	0	(3.9e6,-7.7e5)	0
center 4	(1.0e0,4.9e-1)	99	(1.0e6,8.2e5)	0	(4.7e5,1.0e6)	0
center 5	None	None	(4.5e-3,5.3e-5)	100	(8.5e7,5.0e7)	0
center 6	None	None	None	None	(4.5e-3,5.3e-5)	100

Table 1: The results of K-means with the idea of RPCL. 'coordinate' represents the coordinate of the cluster centers, '# points' represents the number of data points in the cluster.

As we can see from the clustering results, K-means with the idea of RPCL performs quite well in the presence of extra centers. It can make the extra centers far away from the generated dataset.

However, there also exists some problems with this algorithm. First, it needs to tune the parameter η. In my experiments, η equals 2, 3, 5 respectively in the situation of 4, 5, 6 initialized centers. Second, it highly depends on the initial coordinate of cluster centers as well. In some initialization, the results are not so good.

3 Model selection of GMM

In this section, I will report my experimental comparison on model selection performance between AIC, BIC and VBEM. I will make comparisons from three perspectives: the number of data points in each cluster (sample size), the number of clusters in the dataset (cluster number), the dimensionality of the data points (dimension).

3.1 Sample sizes

To compare the performance of different model selection methods in terms of sample sizes, I generate 6 datasets. They are all 2-dimension 3-cluster datasets, but have 10, 20, 50, 100, 200, 300 data points in each cluster respectively. The clustering results are shown as Tab. 2 and Fig. 2.

Sample Sizes	10	20	50	100	200	300
The K AIC chooses	15	12	2	2	3	3
The K BIC chooses	3	3	2	2	3	3
The K VBEN chooses	2	3	3	4	3	4

Table 2: This table shows the number of clusters AIC, BIC and VBEM choose on 2-dimension 3-cluster datasets with different sample sizes.

As we can see, AIC performs the worst, while BIC and VBEM achieve pretty good performance on different sample sizes. What's more, with larger sample sizes, more information about the clusters are given, so all three model selection methods perform well. Sometimes, BIC and VBEM will make mistakes due to the deceptive distribution, but the wrong clustering results look acceptable.

3.2 Cluster numbers

To compare the performance of different model selection methods in terms of cluster numbers, I generate 5 datasets. They are all 2-dimension clusters with 100 samples in each cluster, but have 2, 3, 4, 5, 6 clusters respectively. The clustering results are shown as Tab. 3 and Fig. 3.

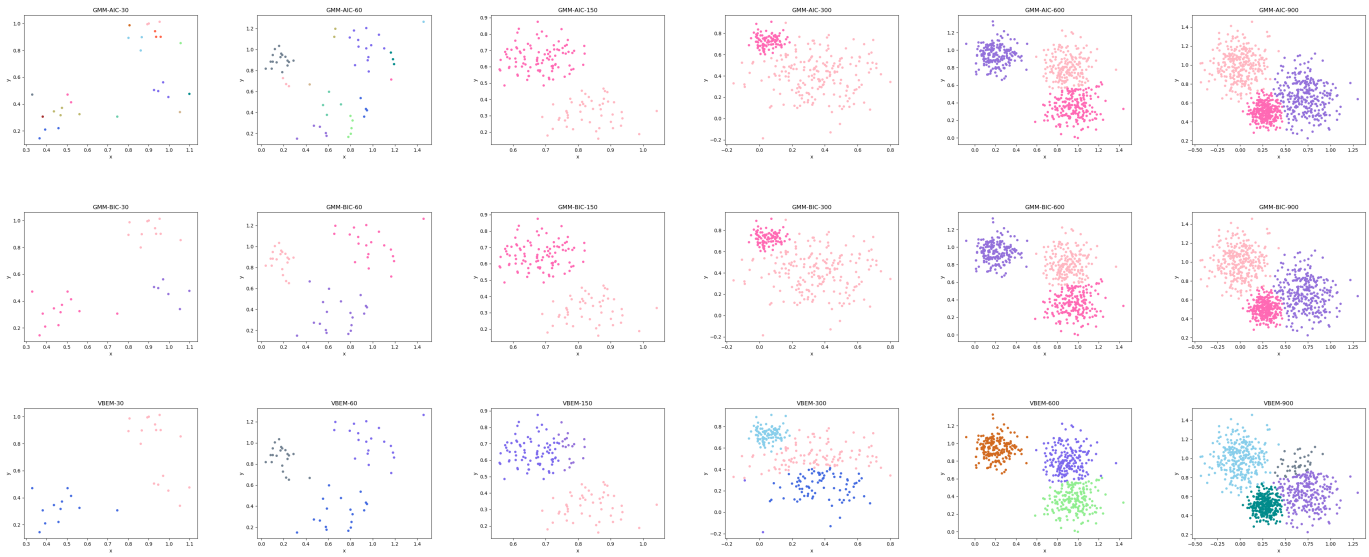


Figure 2: This picture shows the performance of AIC, BIC and VBEM on 2-dimension 3-cluster datasets with different sample sizes. The first, second and third row represents the performance of AIC, BIC and VBEM respectively. Each row from left to right, the sample size increases. Sample sizes 10, 20, 50, 100, 200, 300 in each cluster is tested.

Cluster Numbers	2	3	4	5	6
The K AIC chooses	2	3	5	5	7
The K BIC chooses	2	3	4	5	7
The K VBEN chooses	2	3	3	5	6

Table 3: This table shows the number of clusters AIC, BIC and VBEM choose on 2-dimension 100-sample size datasets with different cluster numbers.

As we can see, BIC and VBEM still perform better than AIC on different cluster numbers. Moreover, VBEM is better than BIC in my opinion because its wrong clustering result is more reasonable than BICs.

3.3 Dimensions

To compare the performance of different model selection methods in terms of different dimensions , I generate 5 datasets. They are all 3-cluster datasets with 100 samples in each cluster, but have different dimensions 3, 4, 5, 6, 7 respectively. The clustering results are shown as Tab. 4 and Fig. 4.

Dimension	3	4	5	6	7
The K AIC chooses	3	3	3	9	3
The K BIC chooses	3	3	3	3	3
The K VBEN chooses	3	3	3	3	3

Table 4: This table shows the number of clusters AIC, BIC and VBEM choose on 3-cluster 100-sample size datasets with different dimensions.

As we can see, AIC's performance is a little bit worse than BIC and VBEM. Even if high dimension data may not be easy to separate on some dimensions, they can be clustered accurately according to information from all dimensions.

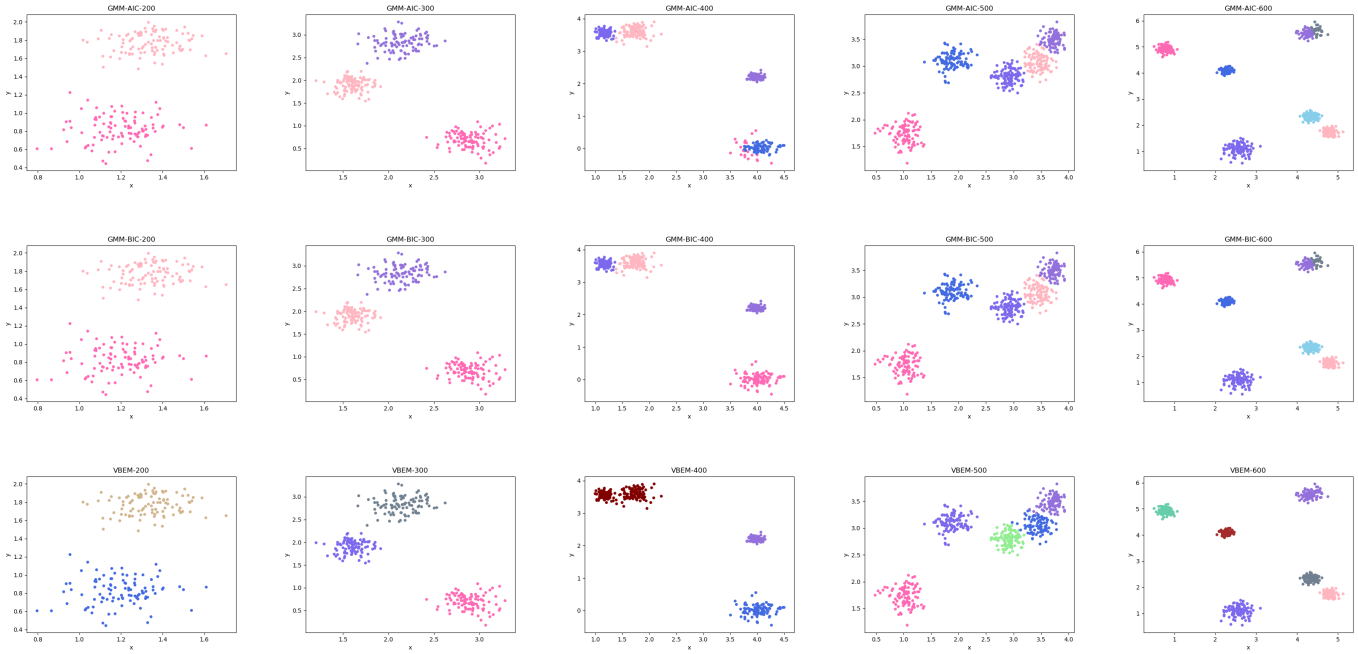


Figure 3: This picture shows the performance of AIC, BIC and VBEM on 2-dimension 100-sample size datasets with different cluster numbers. The first, second and third row represents the performance of AIC, BIC and VBEM respectively. Each row from left to right, the cluster numbers increases. Cluster numbers 2, 3, 4, 5, 6 is tested.

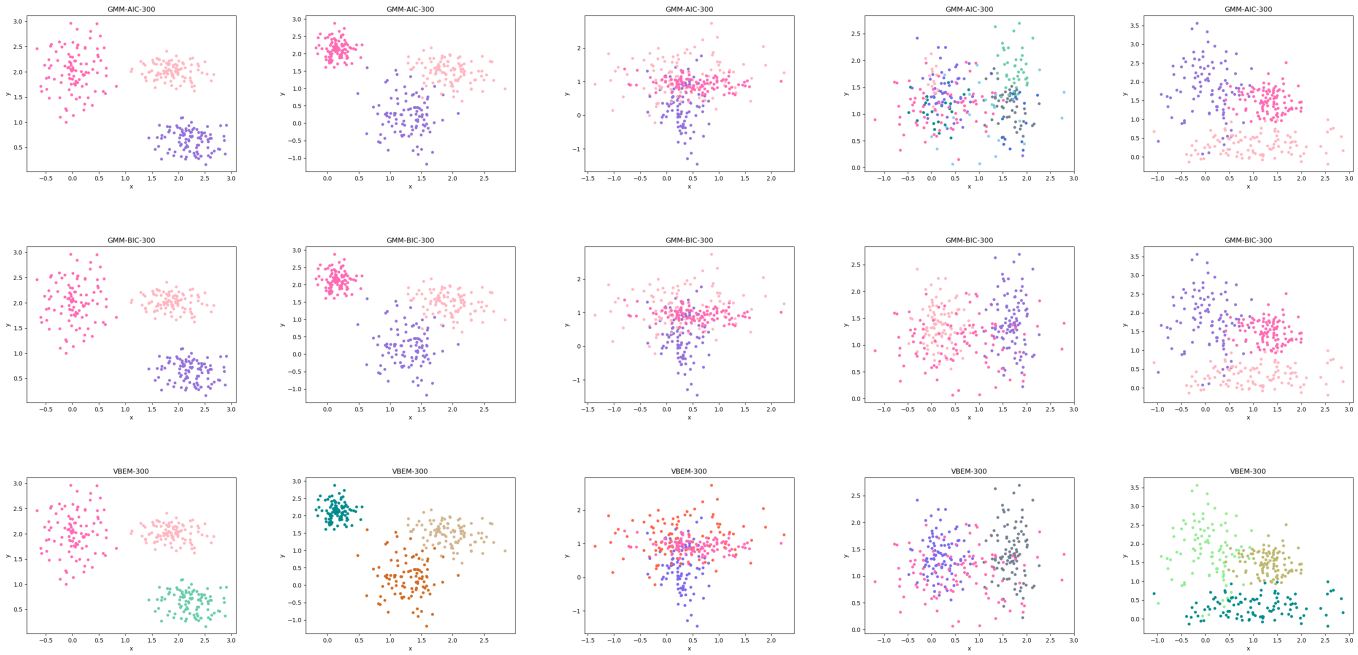


Figure 4: This picture shows the performance of AIC, BIC and VBEM on 3-cluster 100-sample size datasets with different dimensions on the first two dimensions. The first, second and third row represents the performance of AIC, BIC and VBEM respectively. Each row from left to right, the dimension increases. dimension 3, 4, 5, 6, 7 is tested.

4 Conclusion

In this project, I revised EM algorithm for K-means and GMM and enhanced my understanding; I revised the difference between K-means and GMM and their advantages and disadvantages; I implemented K-means using sklearn and by myself; I implemented GMM using sklearn; I used AIC and BIC to select the best cluster number; I implemented VBEM to automatically select the best model; I varied parameters to compare the performance of AIC, BIC and VBEM. To conclude, I harvest a lot in this project. My codes can be found on github: https://github.com/JiayiXuDaisy/Machine-Learning.git.