Homework1

Jizheng Chen 520021911182

March 26, 2023

1 K-mean algorithm

We need to prove that $J(C_i';\mu') < J(C_i;\mu)$, First we prove that step 1 will never increase the objective function: let the clusters in last interation C_i, and after assigning each point to the nearest μ_k the clusters becomes C_i', we then will show that : $J(C_i';\mu) < J(C_i;\mu)$,where μ_k' is the nearest cluster center to x_i among the centers. According to the process, we have for all i,k,j:

$$||x_i - \mu_j||^2 \geq ||x_i - \mu_k'||^2 \tag{1}$$

the equality holds when $j = k$ i.e.cluster of x_i hasn't change.

So for every x,

$$\sum_{k'=1}^{K} r_{tk'}||x_t - \mu_k'||^2 \leq \sum_{k=1}^{K} r_{tk}||x_t - \mu_k||^2 \tag{2}$$

As the property holds for every x, we then have $J(C_i';\mu) < J(C_i;\mu)$.

Second we will prove that step2 will never increase the objective function, that is to show:$J(C_i;\mu') < J(C_i;\mu)$, as in M-step only the cluster center is updated.

For each cluster C_k with center μ_k we show that the mean of x_k can minimize the square error, i.e:

$$\mu_k' = argmin_\mu \sum_{t=1}^{N_k} ||x_t - \mu||^2 \tag{3}$$

let

$$\frac{\partial f(\mu)}{\partial \mu} = \sum_{t=1}^{N_k} -2(x_t - \mu) = 0 \tag{4}$$

we can solve that

$$\mu = \frac{\sum_{t=1}^{N_k}}{N} \tag{5}$$

So for every cluster we have

$$\sum_{k'=1}^{K} ||x_t - \mu_k'||^2 \leq \sum_{k=1}^{K} ||x_t - \mu_k||^2 \tag{6}$$

thus $J(C_i;\mu') < J(C_i;\mu)$ holds. So that each of the above two steps will never increase the

k-mean objective function. #

2 k-mean vs GMM

We can convert k-mean algorithm to EM algorithm by doing the three things:

1. change the hard assignment to soft assignment.
2. add latent variable π_k and set it different from $\frac{1}{k}$
3. use the GMM M step to update the centroids.

Here I choose to use GMM to update the M step of k-mean algorithm, the steps is as follows:

1.Initialize K centroids randomly.

2.Assign each data point to the nearest centroid.

3.Calculate the mean and covariance matrix of each cluster using the assigned data points.

4.Calculate the soft assignments of each data point to each cluster using the Gaussian probability density function.

5.Update the centroids using the soft assignments.

6.Repeat steps 2-5 until convergence.

The computational details of the formulas are as follows:

Algorithm 1 My Algorithm

1: Initialize K centroids randomly.
2: **repeat**
3: Assign each data point to the nearest centroid using the Euclidean distance formula:$d(x_i, c_j) = \sqrt{\sum_{k=1}(x_{i,k} - c_{j,k})^2}$.
4: Calculate the mean and covariance matrix of each cluster using the assigned data points by $\mu_j = \frac{1}{n_j}\sum_{x_i \in C_j} x_i$.
5: Calculate the soft assignments of each data point to each cluster using the Gaussian probability density function using :$\Sigma_j = \frac{1}{n_j}\sum_{x_i \in C_j}(x_i - \mu_j)(x_i - \mu_j)^T$ and $p(x_i|\mu_j, \Sigma_j) = \frac{1}{(2\pi)^{p/2}|\Sigma_j|^{1/2}} exp(-\frac{1}{2}(x_i - \mu_j)^T \Sigma_j^{-1}(x_i - \mu_j))$.
6: Update the centroids using the soft assignments. $c_j = \frac{\sum_{i=1}^{n} p(x_i|\mu_j, \Sigma_j) x_i}{\sum_{i=1}^{n} p(x_i|\mu_j, \Sigma_j)}$
7: **until** convergence

The advantages of my algorithm are that it can give a non-spherical decision boundary and using soft assignments to update centroids also helps to converge. However, as we are using GMM-EM algorithm to do the M-step, it is more computationally expensive than K-Means and will still face the problems of KNN, such as local minima and hyper-parameter decision inaccuracy.

3 k-mean vs CL

The difference between k-mean and CL is as follows:

1. k-mean is used when the whole dataset is given, while CL is suitable when data points comes one by one

2. k-mean works by minimizing the square error of each data point and the centroids it is assigned to. While CL works by having a set of points compete to represent input data, a centroid wins if the new comer is closer to it.

The however have some similarities:

1. They are both unsupervised learning algorithms using for clustering.

2. Plain CL and k-mean both can't decide the number of clusters and need to set k as a hyper parameter.

3. When the learning rate of CL becomes $\frac{1}{N}$, CL is equivalent to k-means algorithm

To apply the idea of Rival Penalized Competitive Learning to k-mean, the following things can be down to expel the unwanted points:

1. Instead of updating the centroid to be the mean of a cluster, we update it by moving it a certain step towards the centroid.

2. We compel a rival penalty on each point to eliminate extra points in cluster assigning

3. We import a diversity term that affected by the total number of centroids.

To implement 1, we update each centroid by:

$$\mu_k' = \mu_k + \eta * (avg\{C_k\} - \mu_k) \tag{7}$$

where η is the step size(or learning rate)

I formulated 2 different penalty functions, the first is **sum penalty**,which can be formulated as:

$$P_k = \sum_{j \neq k} ||\mu_j - \mu_k||^2 \tag{8}$$

And the second is called**second penalty** which is formulated as:

$$P_k = \max_{j \neq k} ||\mu_j - \mu_k||^2 \tag{9}$$

To implement 3, we make diversity term to be $\frac{1}{k-1}$ where k is the number of centroids.

The above method is implemented in a dataset of 3 Gaussian distribution cluster, code is available at code.

From figure 3 and 4.2 we can know that KNN can handle a k equal to centroid number well, but can't work when k is greater than number of centroids. Using RPCL can deal with the problem well, and sum penalty works better than second-kick penalty.

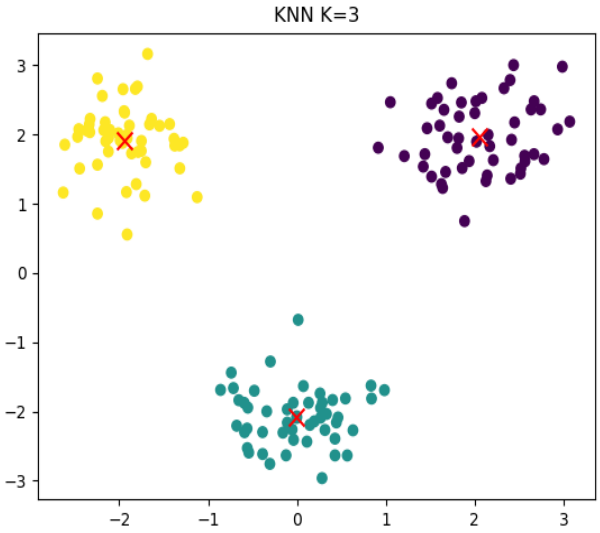

Figure 1: knn with k=3

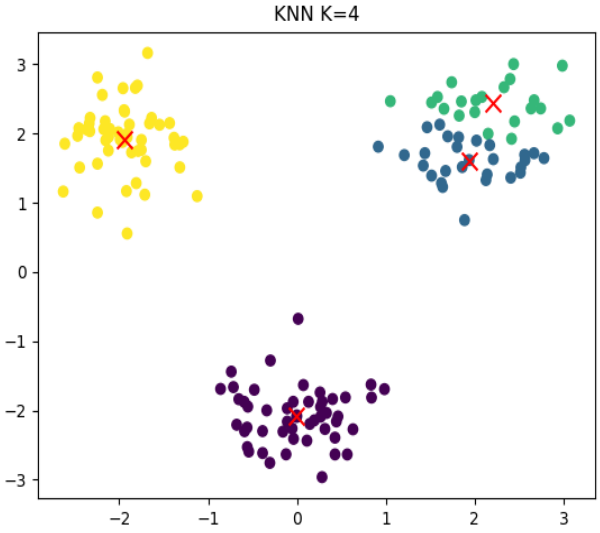

Figure 2: knn with k=4

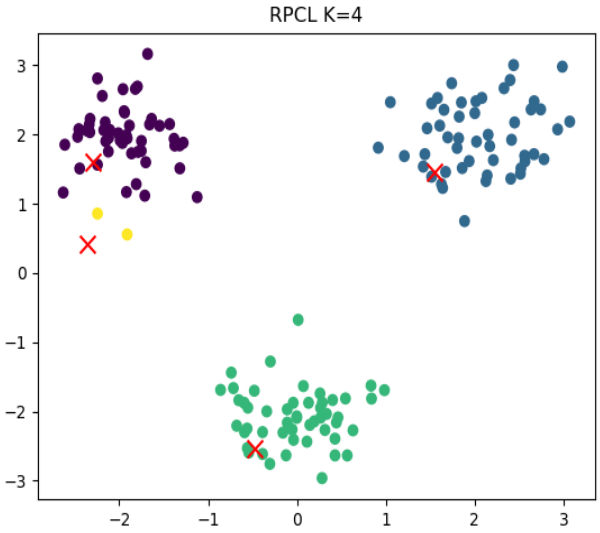

Figure 3: RPCL with sum penalty

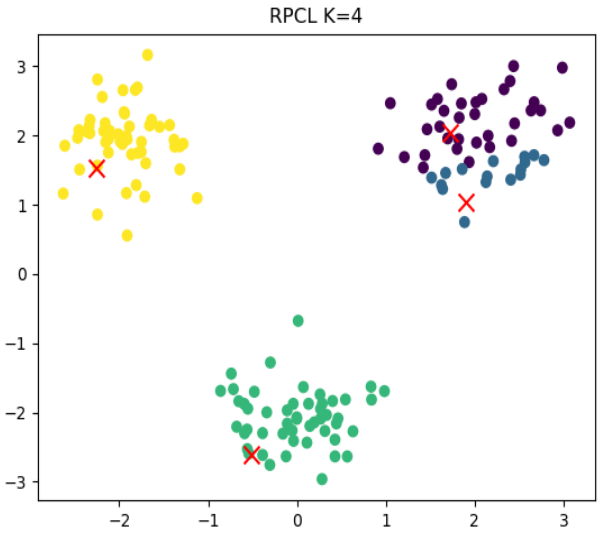

Figure 4: RPCL with second penalty

4 model selection of GMM

4.1 experiment setting

In my experiment, I adjust the sample sizes, dimensionality and fix the cluster of the distribution of data to 10. The GMM models are trained using a series of hyper parameter k.

1. sample size: varies from 50 to 250, with step interval 50
2. dimensionality of data: varies from 2 to 8, with step interval 3
3. number of clusters in the GMM model: varies from 1 to 19, with step interval 1.

After a GMM model with hyper parameter k is trained on a given data of a certain sample size and dimensionality, its evaluation score J(k) can be acquired with aic, bic,or vbem methods. The the optimal setting k^* can be acquired using:

$$k^* = argmax_{k=1,...,K} J(k) \tag{10}$$

where J(k) is generated using the formula of aic, bic and vbem seperately, where:

$$J_{AIC}(k) = ln([p(X|\overline{\Theta}_k]) - d_m \tag{11}$$

and

$$J_{BIC}(k) = ln([p(X|\overline{\Theta}_k]) - \frac{lnN}{2} d_m \tag{12}$$

Then the best log-likelihood under the best choosen k can be calculated by:

$$LL_{best} = ln([p(X|\overline{\Theta}_{k^*}]) \tag{13}$$

4.2 experiment results

Here I'll give the experiment results both in table and in graph.

Table 1: Best log likelihood with different dimension and sample size

Method	$50,2$	$50,5$	$50,8$	$100,2$	$100,5$	$100,8$	$150,2$	$150,5$	$150,8$
AIC	-2.8561	-6.9964	-10.1792	-2.8358	-7.1131	-11.0954	-2.8857	-7.0545	-11.0714
BIC	-2.8561	-6.1951	-9.6988	-2.8358	-6.7997	-10.4656	-2.8377	-6.9208	-10.8257
VBEM	-2.6896	-6.1951	-9.6988	-2.7675	-6.6766	-10.4656	-2.7870	-6.8338	-10.8131

Method	$200,2$	$200,5$	$200,8$	$250,2$	$250,5$	$250,8$
AIC	-2.8955	-7.0206	-11.4522	-2.8638	-7.1290	-11.1941
BIC	-2.8503	-6.9274	-10.8829	-2.8638	-7.0067	-11.0924
VBEM	-2.7966	-6.8721	-10.8829	-2.7539	-6.9572	-11.0056

From both the tabular and the plot we can see the model selection performance between BIC, AIC and VBEM. In different experiment settings, VBEM has a better model selection performance concerning on best log likelihood over BIC, and BIC performs better than AIC.

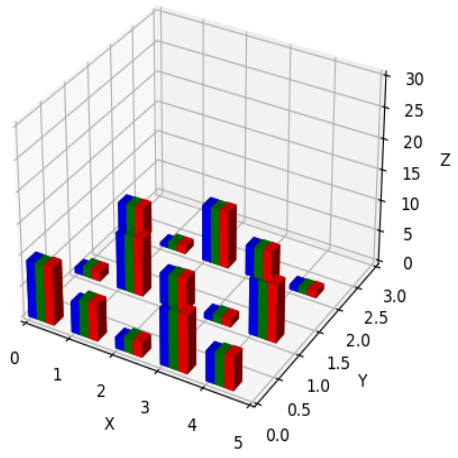

Figure 5: Best log likelihood with different dimension and sample size