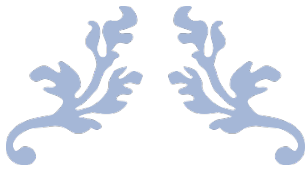


CS5487 PROGRAMMING ASSIGNMENT2

ZHANG Rui 54933930

Problem 1

1. K-means Algorithm

For K-means algorithm, I have used 2 initialization strategies for initializing the initial cluster – selecting random data points and set the points to be furthest apart from each other.

The result with random strategies is shown as Figure-P1.1, Figure-P1.2 and Figure-P1.3.

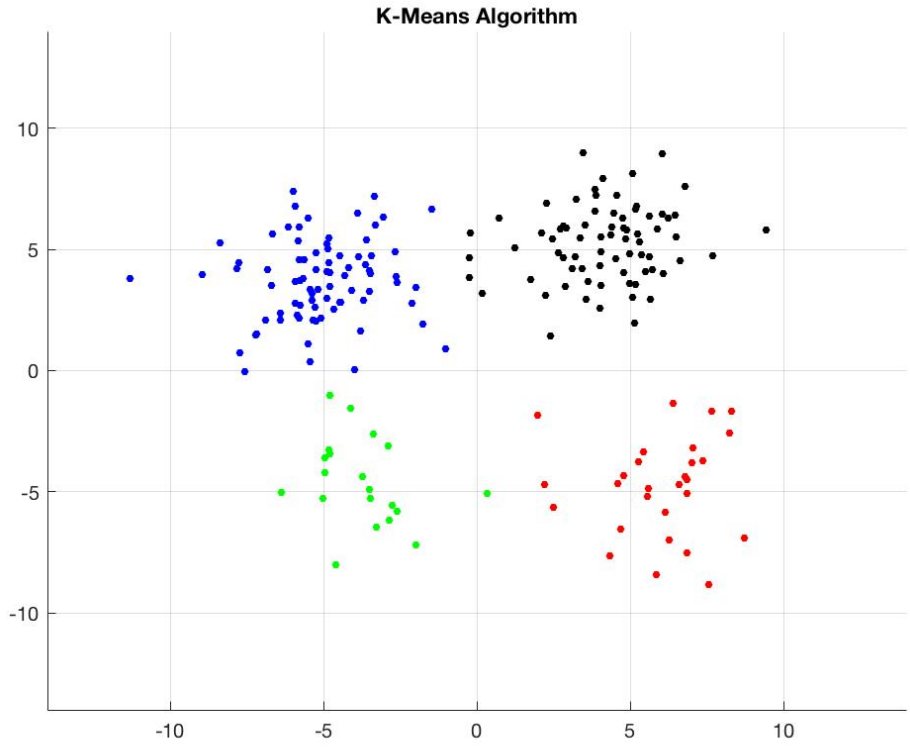


Figure-P1.1 Data_A with random initialization

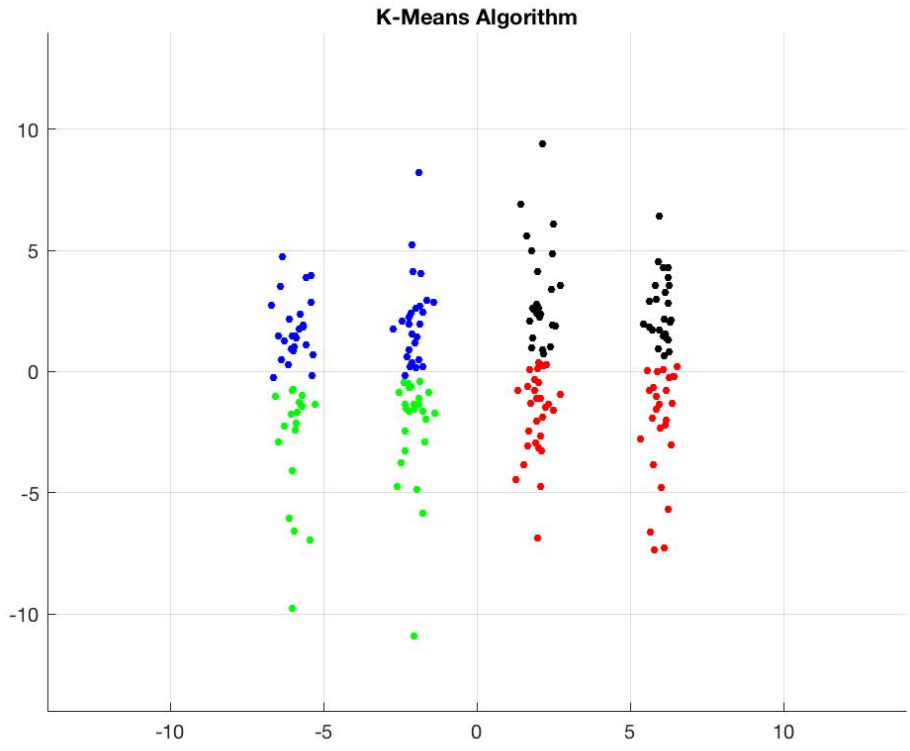


Figure-P1.2 Data_B with random initialization

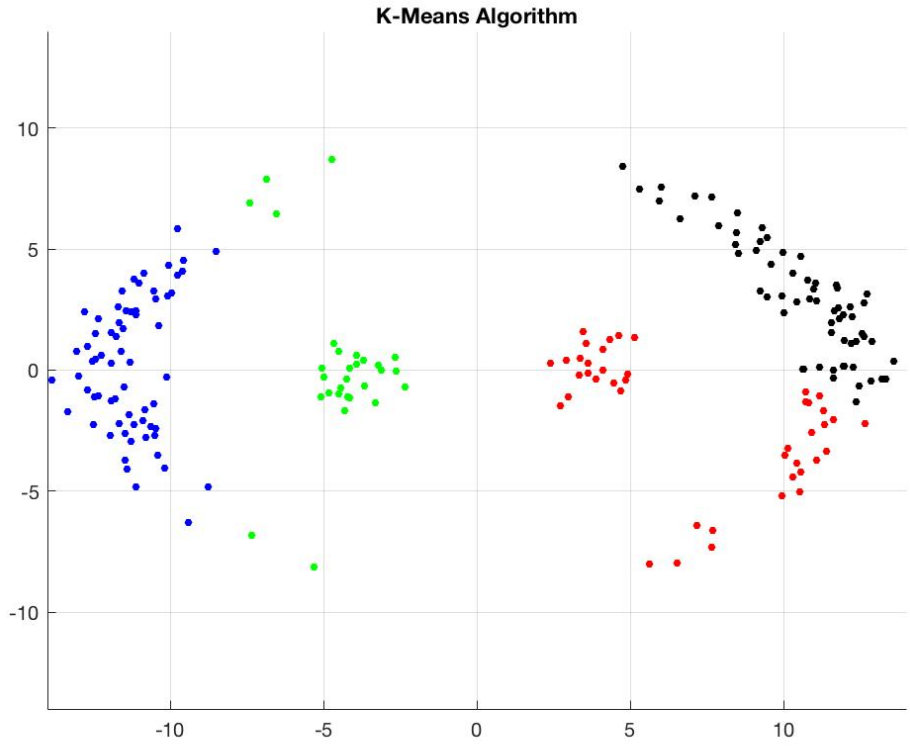


Figure-P1.3 Data_C with random initialization

From the above pictures, we can learn that the clustering result of using random initialization is unstable and show a large clustering error. Thus, we still cannot give a conclusion about the K-means performance regarding to the above figures.

Hence, I used another initialization strategy which is to set the points to be furthest apart from each other. The results are shown as Figure-P1.4, Figure-P1.5 and Figure-P1.6.

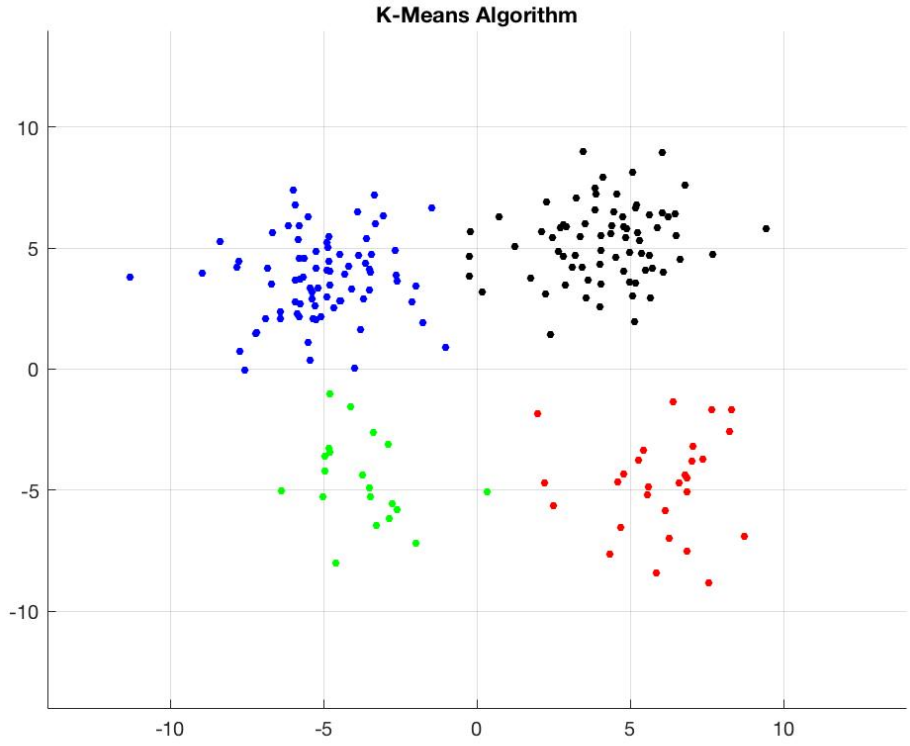


Figure-P1.4 Data_A with Furthest-first-like initialization

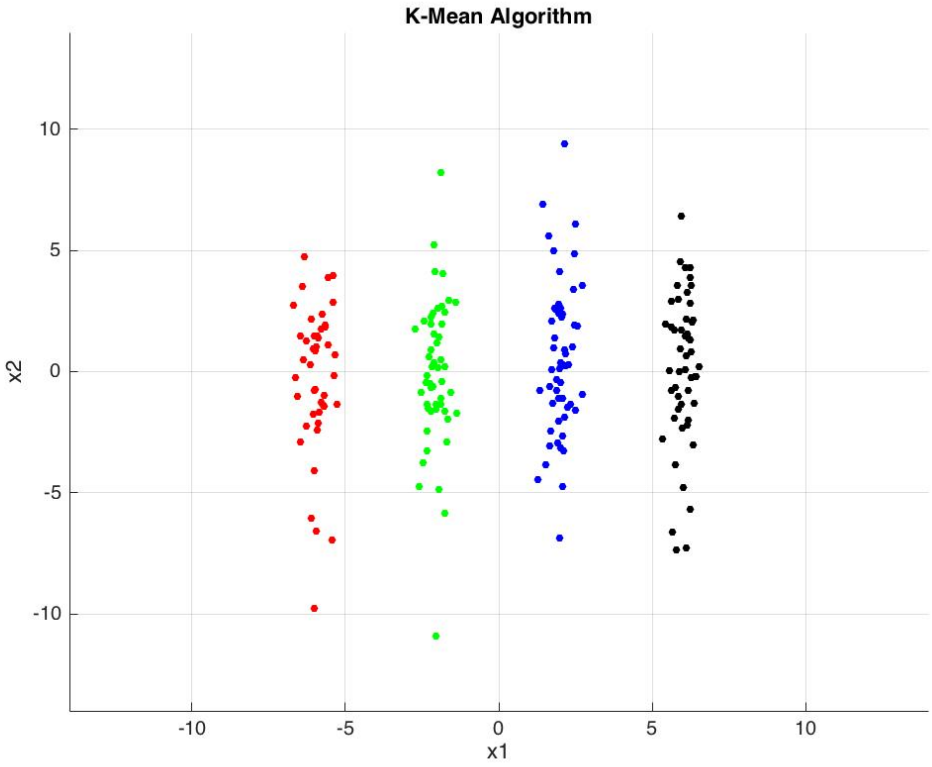


Figure-P1.5 Data_B with Furthest-first-like initialization

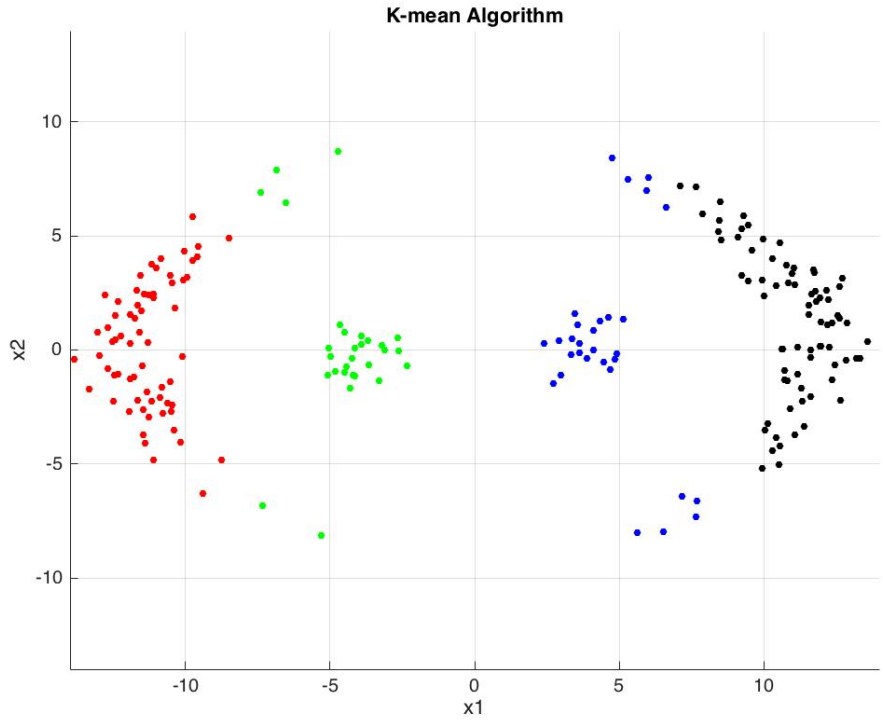


Figure-P1.6 Data_C algorithm with Furthest-first-like initialization

According the above pictures, we can learn that the result is stable. Although there is still a little error occurred at Data_A result, the K-means shows a good clustering performance on Data_A and Data_B. However, the Data_C cannot be clustered well using K-means algorithm whatever the initial point is.

Overall, I think K-mean is an easy-to-use and fast algorithm. However, the prior probability and the covariance of each cluster is fixed which is like a special situation of EM for GMM. The prior probability of each cluster is even and fixed which is unreal in many cases such as Data_C. In the same time, the covariance is fixed that means we cannot build the model regarding to the importance of data points in different dimensions. So, I think the K-means is a hard clustering that cannot be used generally and is very sensitive to outliers.

2. EM Algorithm for GMM

In the implementation of EM algorithm, I also use 2 initialization strategies – using the outcome of K-means and just using the minimum of distance of ||x-init_miu||^2 to assign a coarse label on each point and calculate the initial covariance of each cluster.

First, I used the outcome of K-means as initial parameter of EM algorithm. The results are shown as Figure-P1.7, Figure-P1.8, and Figure-P1.9.

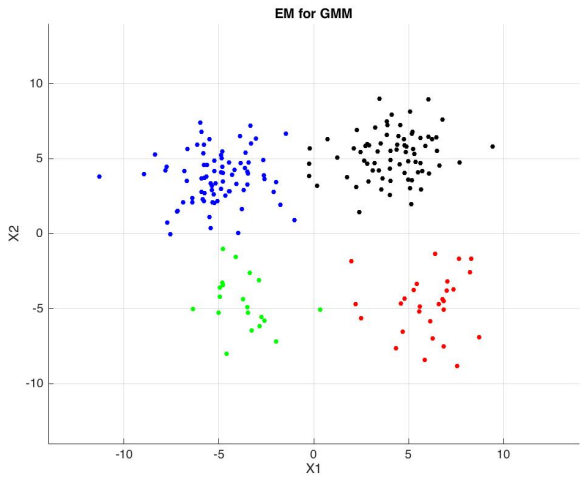


Figure-P1.7 Data_A with K-means initialization

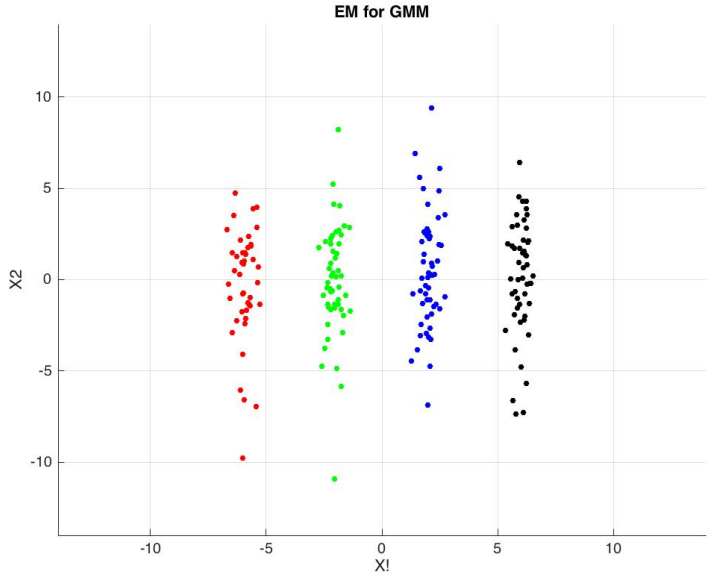


Figure-P1.8 Data_B with K-means initialization

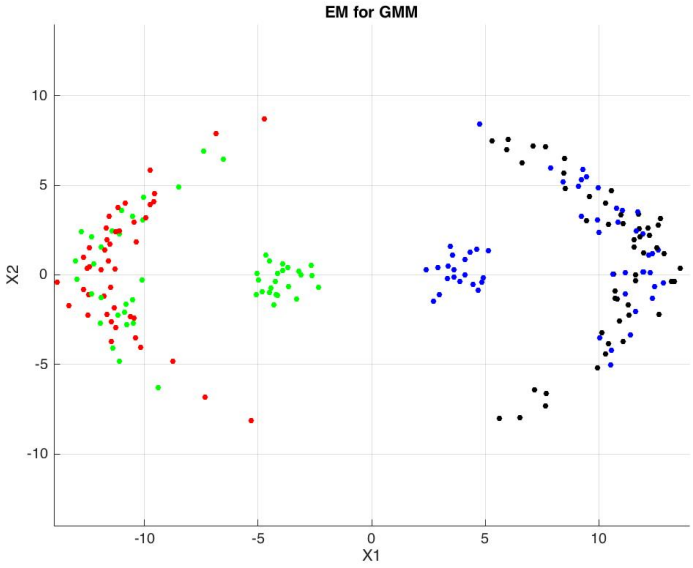


Figure-P1.9 Data_C with K-means initialization

We can find that the clustering performance of Data_C is very bad because of the bad outcome of K-means. Thus, the result of K-means will cause a huge impact on the outcome of the EM for GMM. Because of this interference factor, we still cannot judge the performance of EM. So, I use the second initialization strategy for eliminating this interference factor. The results are shown as Figure-P1.10, Figure-P1.11 and Figure-P1.10.

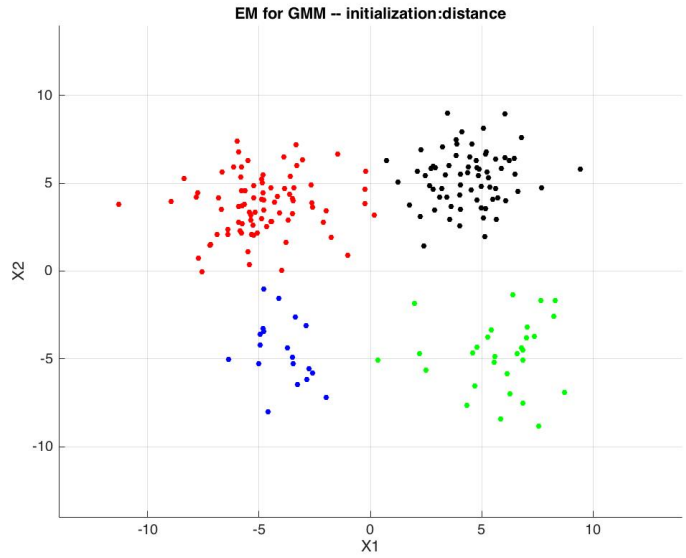


Figure-P1.10 Data_A with minimal distance initialization

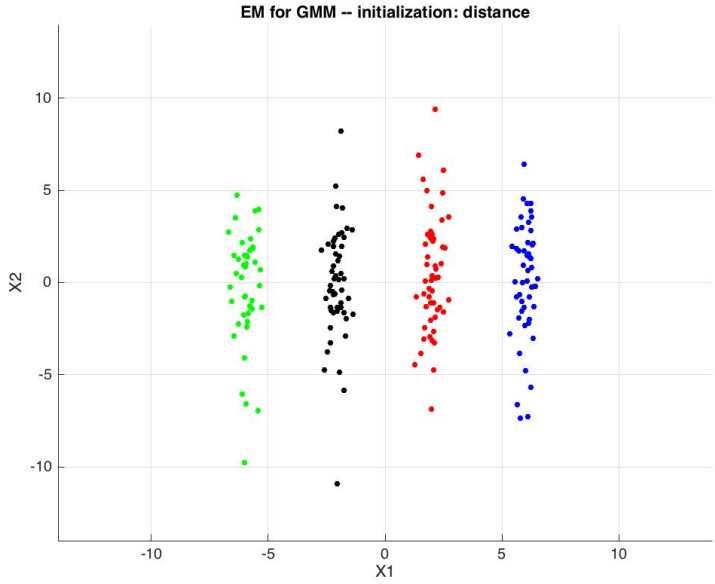


Figure-P1.11 Data_B with minimal distance initialization

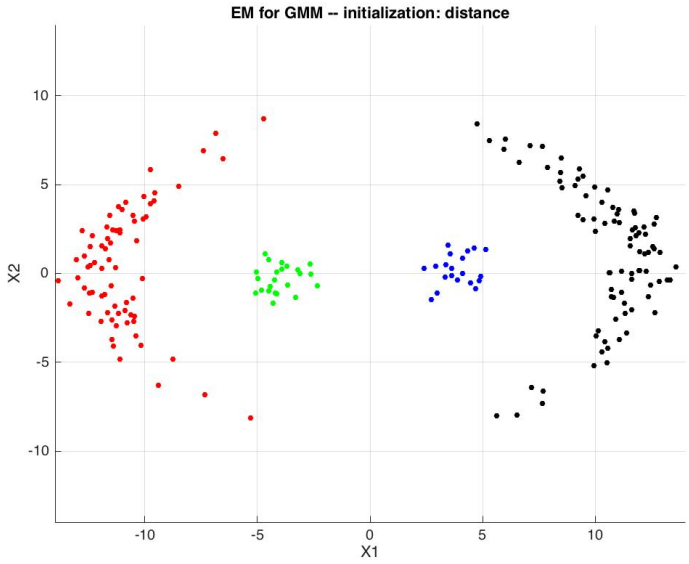


Figure-P1.12 Data_C with minimal distance initialization

After using the second initialization strategy, we can observe that the EM of GMM performed well on the three sets of data. Meanwhile, the Data_C can be clustered very well, rather than the performance of K-means.

Thus, I think EM for GMM is a fast, accurate but difficult-to-implement algorithm. The prior probability and covariance is not fixed anymore which means that the data have a better model of prior probability and using covariance on calculating the posterior probability to cluster the data in proportion. Thus, the EM of GMM is a fuzzy clustering that can fit general case well.

3. Mean-shift Algorithm

In the implementation of Mean-shift algorithm, we do not need to initial any parameter but need to set a bandwidth h. The results of different bandwidths are shown below.

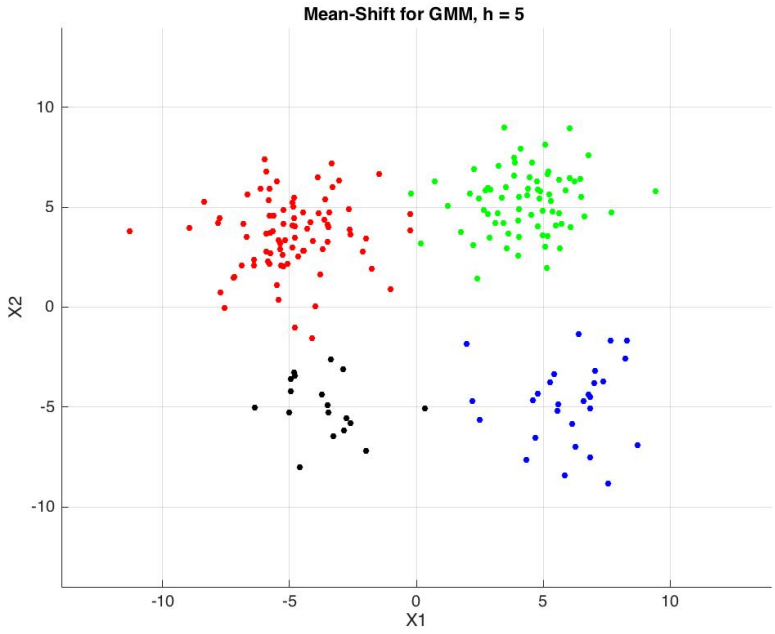


Figure-P1.13 Data_A with Mean-shift (h=5)

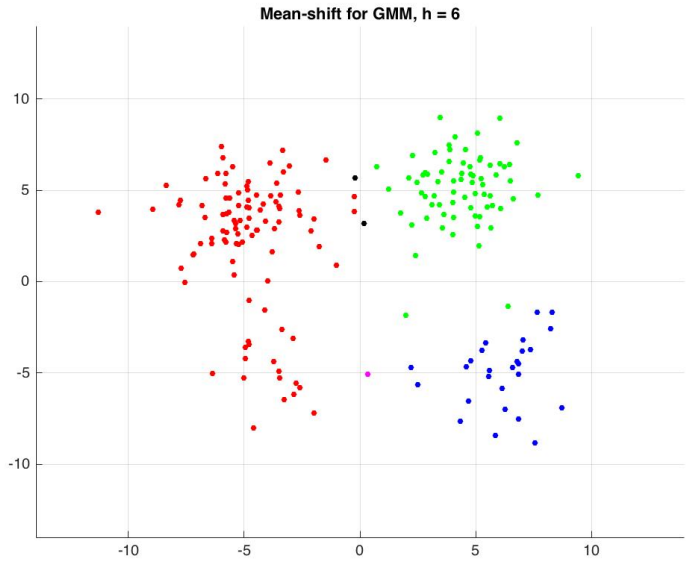


Figure-P1.14 Data_A with Mean-shift (h=6)

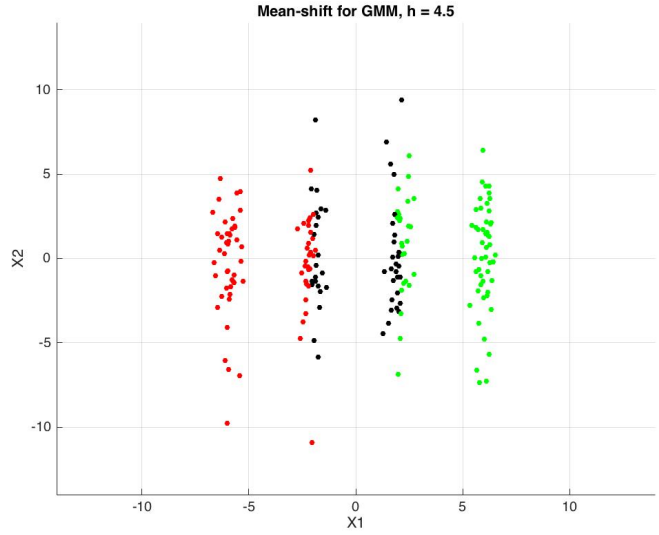


Figure-P1.15 Data_B with Mean-shift (h=4.5)

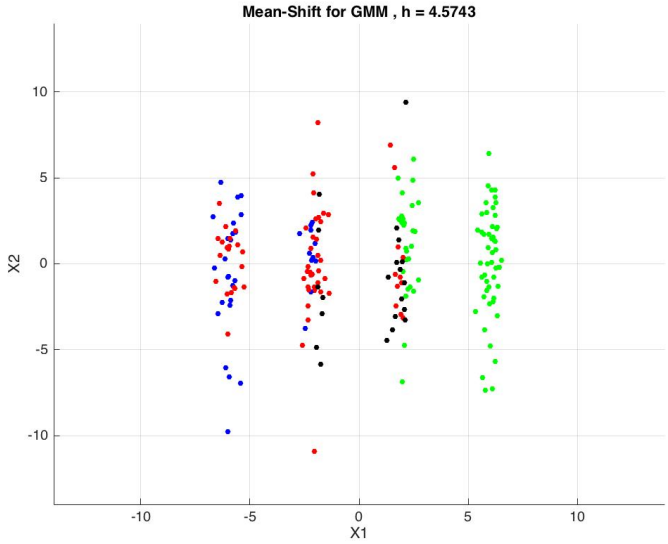


Figure-P1.16 Data_B with Mean-shift (h=4.5743)

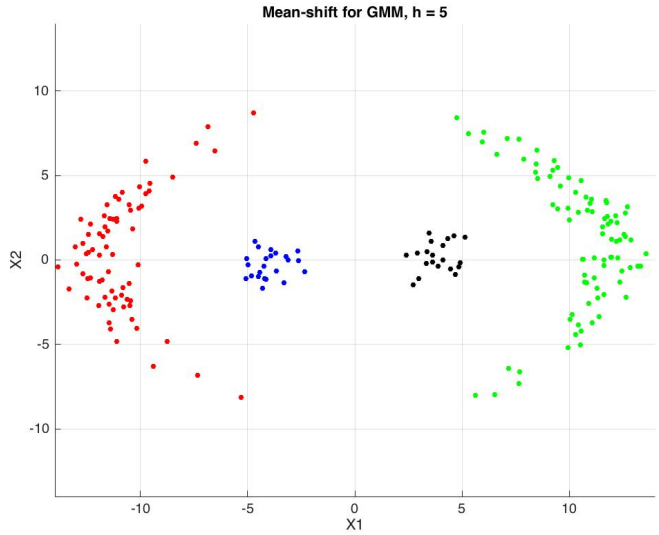


Figure-P1.17 Data_C with Mean-shift (h=5)

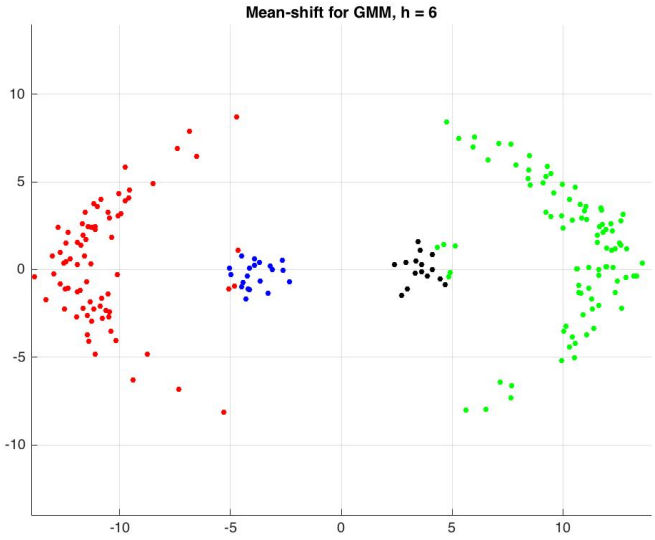


Figure-P1.18 Data_C with Mean-shift (h=6)

According to the above figures, we can find that using Mean-shift algorithm with bandwidths being equal to 5 to cluster Data_A and Data_C have more accurate outcome comparing with the first two algorithms, since the kernel density estimator avoid the problem of choosing the number of clusters and initialization strategies, but consider every data point as one cluster center. So, there is no error caused by initialization phase. Meanwhile, we need to use all data to get the center points so that the

algorithm is robust and insensitive to outliers. However, it is difficult to be manipulated: the size of data is large and the operation is very slow.

In the same time, the only remaining parameter -- covariance of each cluster (a Gaussian kernel whose centered point is data point) is equal to the bandwidth, which means the critical part of the Mean-shift algorithm is how to choose a bandwidth. Comparing between different data with different bandwidth h, we can find that Mean-shift algorithm is very sensitive to bandwidth. For example, Data_C with h1=6 (Figure-P1.18), that shows a different result comparing with the one of h2=5 (Figure-P1.17), although h1 is only 1 more than h2. There also has a remarkable difference of Data_B with different h (Figure-P1.15 and Figure-P1.16) although the clustering performance is very bad.

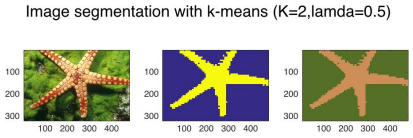
For the performance of Data_B, I still cannot figure out where is going wrong.

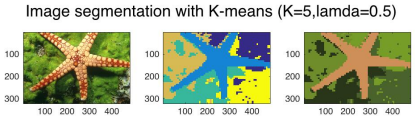
Problem 2

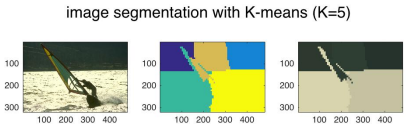
(a)

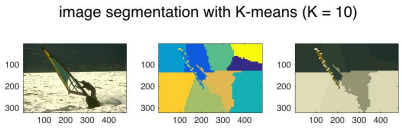
I have segmented three pictures using the above three algorithms respectively. Meanwhile, I tried to change with different K and h to observe what performance they have. The results are shown below.

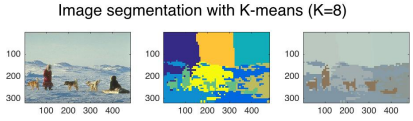
- K-means Algorithm

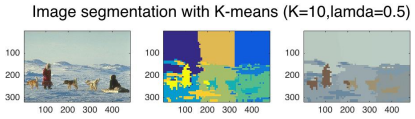




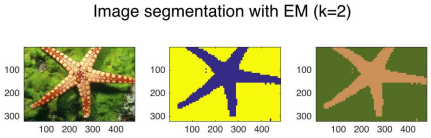


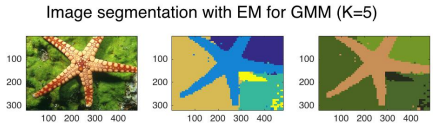


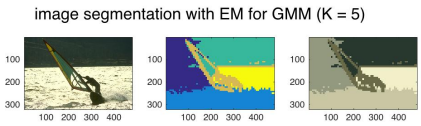


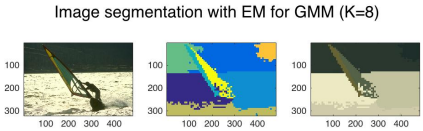


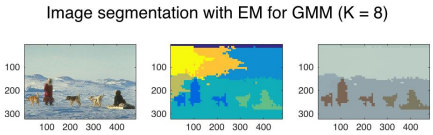
- EM for GMM

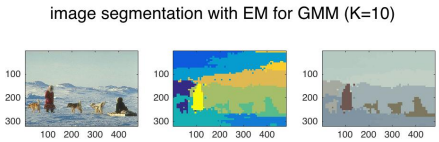




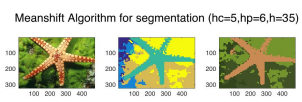


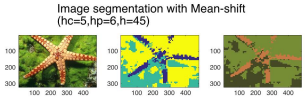


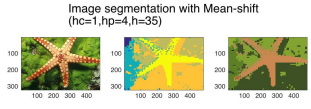


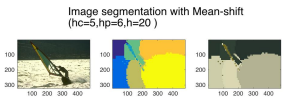


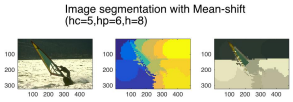
- Mean-shift Algorithm

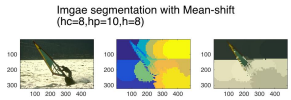


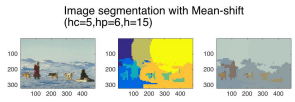


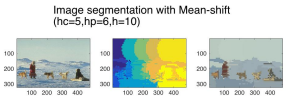


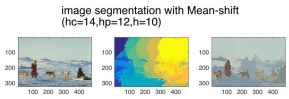












Firstly, comparing among the performances of K-means, EM for GMM (both with the same parameter K) and Mean-shift, it is obviously that the outcome of EM is better than the others. The performance of K-means is the worst.

Secondly, I have changed their parameters K or h and found that:

- As the parameter K becoming bigger, the time of calculating K-means will be longer. However, it seems that the performance does not get any improvement.
- As the parameter K becoming bigger, there is only slightly change on the time of calculation, but in some case the better result will be got using the smaller parameter K.
- In the Mean-shift cases, h is more critical than hc and hp. There is a remarkable change of the result with different h. However, if I change the parameters hc and hp, the outcome is only changed mildly. Meanwhile, as the parameter h becoming bigger, the segmentation will be fuzzier.
- EM for GMM still shows the fastest calculation performance and get the better segmentation results.

Overall, there are some properties about the three algorithms:

- K-means: Because of the fixed prior probability, fixed isotropic covariance matrix and only using distance between data point and

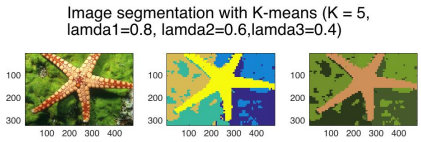
center point to cluster, it will assign the point to the cluster with the closest center point, but in some case, the probability of assigning to that cluster (i.e. posterior probability) actually is very low. Thus, according to some results of segmentation using K-means, it seems that some section of the picture cannot be segmented (such as the picture of sailing). However, it can perform well on some simple pictures with contrasting colors (such as the picture of starfish).

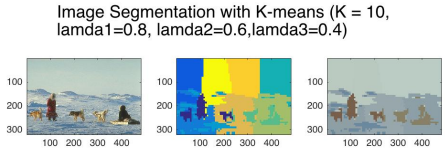
- EM for GMM: Contrast to K-means, the data point will be any cluster which depends on the prior probability and posterior probability. Thus, the performance of segmentation is better than K-means. Meanwhile, the parameter K is very critical to this algorithm. A suitable K can produce a good result.
- Mean-shift: In the Mean-shift case, our goal is to find such a cluster center that assigns pixels within a certain range of this center to the same label. The similar colors will be gathered together into a cluster. So, we can see that there are some circle-like areas in the segmentation results, and these areas will be increased as the h being bigger. When the areas become very big, the segmentation will be fuzzier. Hence, the hc, hp and h is very critical to this algorithm. A small change of these parameters will cause a huge difference of the result, especially the parameter h. In the same time, it is easy to get into the local optimal solution and cannot get the global one.

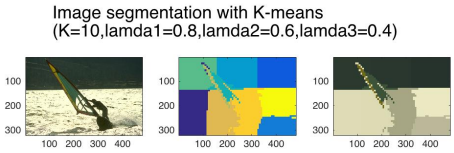
(b)

I have modified my K-means and mean-shift implementations to allow different feature scaling with changing the distance d and kernel k is equivalent to scaling each dimension in the feature vector x by an appropriate amount. The new results of K-means and Mean-shift are shown below

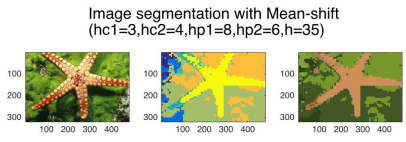
- K-means

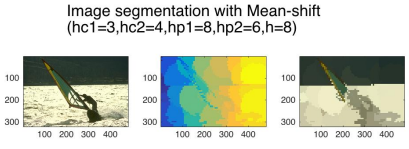


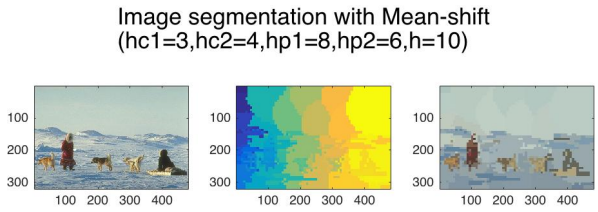




- Mean-shift







According to the above results, we can find that both algorithms have different degrees of improvement, especially Mean-shift.