Clustering Report

Authors: Na Li, Yash Naik

Part 1

In this assignment, Yash is mainly responsible for implementing K-means algorithm, and we two implemented Gaussian Mixture Model together.

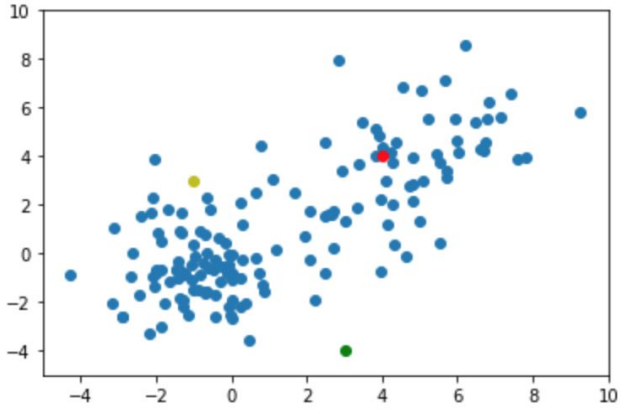
KMeans Clustering

When implementing the K-means algorithm, we used dataframe as our data structure from pandas library, since it is easy to do operations with it. We also used matplotlib.pyplot library to draw the output scatter plots at different stages of clustering.

	0	1
0	-1.861331	-2.991683
1	-2.170092	-3.292318
2	-1.014081	0.385795
3	-2.912943	-2.579539
4	0.035721	-0.799698

	0	1
count	150.000000	150.000000
mean	1.358728	1.118653
std	3.067421	2.704068
min	-4.260794	-3.530785
25%	-1.104974	-0.886802
50%	0.235210	0.491569
75%	4.082047	3.132417
max	9.246253	8.595481

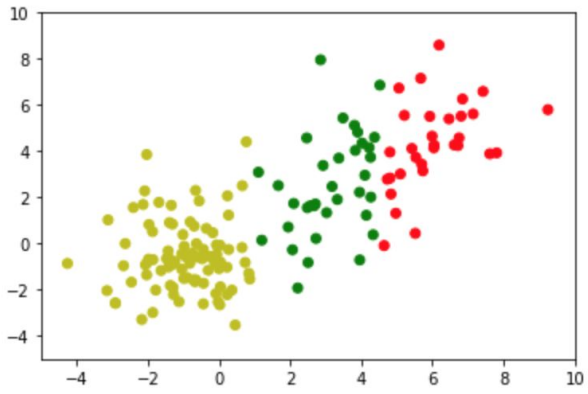
KMeans was not much complicated to implement on the given dataset.



This image above represents the starting point for our algorithm. We have initialized three randomly generated centroids for each of the 3 clusters as required by the assignment.

Output: Image below represents the final three centroids computed from the result of Kmeans.

```
.....................................................................
{1: [6.043653607099999, 4.307911598066667], 2: [-0.9838221813953492, -0.5511124860930233], 3: [3.150244341382353, 2.5281246747941175]}
```



This image represents the final output of our KMeans algorithm. The algorithm worked recursively assigning each data point to its nearest centroid until convergence. Finally, our data has fitted well into three distinct clusters.

GMM Clustering

When implementing the Gaussian Mixture Model using Expectation Maximization algorithm. The data structure we used is mainly the matrix from the numpy library. In this format, we could easily get the sum, establish the 2-d array, generate the matrix with random numbers, and etc. The most tricky part is how to convert the mathematics of GMM into python code.
The termination condition of iterations is when the difference between previous probability and current probability is less than 0.000001, or the iteration time is more than 500.

The mean, covariance, amplitude of our three GGM are shown below:

```
Means of the three Gaussian distributions:  [array([[-0.99189116],
       [-0.62110624]]), array([[3.75211127],
       [2.98225714]]), array([[6.48852878],
       [4.60937921]])]
Covariance of the three Gaussian distributions:  [array([[ 1.17588551, -0.06721079],
       [-0.06721079,  2.04685982]]), array([[2.50118724, 1.65958453],
       [1.65958453, 5.65202292]]), array([[1.44890636, 0.64821354],
       [0.64821354, 1.07667991]])]
Amplitude of the three Gaussian distributions:  [0.5628866927349826, 0.3359033779399118, 0.10120992932510516]
```

Analysis: Comparing the results of two algorithms, we could see both of them work well. But the Gaussian Mixture Model algorithm works much better, since it works on the principle of soft clustering, that clusters may overlap. Each point in the dataset is assigned to a gaussian according to probability of its proximity to that particular gaussian.

Part 2

Python has a really nice library called Sci-kit learn or sklearn which has functions that can easily solve the problems. Using KMEANS() and GuassianMixtureModel() from sklearn library comes extremely handy when fitting highly complex data. Both there library functions can do the job within one line of code and can easily be fine tuned as well.

Part 3

Kmeans is an extremely popular algorithm that helps in data segmentation, data clustering, image segmentation, etc. It is quite a versatile algorithm with many real life applications. Gaussian Mixture Model is yet another popular clustering technique. It is somewhat an extension of kmeans in that it can handle higher dimensional data which is much more complex and doesn't fit well under Kmeans. GMM is very efficient for data analysis and is widely applied in signal and information processing.