Lisa Meng
Shagun Gupta
INF 552
HW #2

Report

1. Data Structure:

   When coding the K-Means and Expectation Maximization (EM) with Gaussian Mixture Model (GMM) clustering algorithms from scratch, the data structure used was (numpy) array and Pandas DataFrame, respectively.

   Code-level optimizations:

   The main way we optimized our code was making functions for each mathematical formula, such as when calculating Euclidean distance, mu (mean), sigma (covariance), and pi (amplitude), which made it easier to call these functions and store previous values during the recursions. This helps us generalize the code a lot more such that no dimension is hard-coded.

   Challenges:

   The main challenge we faced was trying to reach convergence when coding the EM with GMM algorithm. We noticed that our code would continuously run when recalculating r_{ic} to reach convergence. Thus we set our maximum number of iterations to 100 because the default maximum number of iterations in the sklearn library is also 100 (if convergence is not reached before this limit).

   Outputs:

   a. K-Means:
      i. Centroid of each cluster:

```
[[ 3.0831826  1.7762138]
[ 5.620166   5.0262265]
[-0.9747657 -0.684193 ]]
```

   b. EM with GMM:
      i. Mean:

	0	1	2
0	-0.949639	0.169551	4.835910
1	-1.037989	1.086276	3.814162

      ii. Covariance matrix:

```
[          0         1
0  1.091275  0.010445
1  0.010445  1.219378,          0         1
0  4.209460 -0.052684
1 -0.052684  2.317009,          0         1
0  2.799781  1.512180
1  1.512180  4.263387]
```

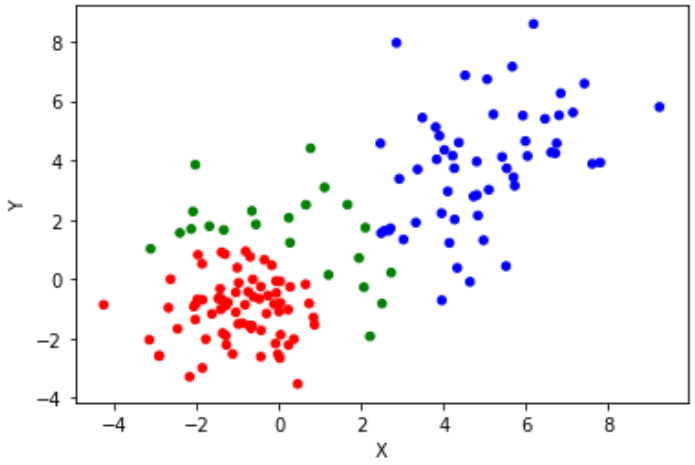
iii. Amplitude:

```
          0         1         2
0  0.130293  0.530318  0.339389
```

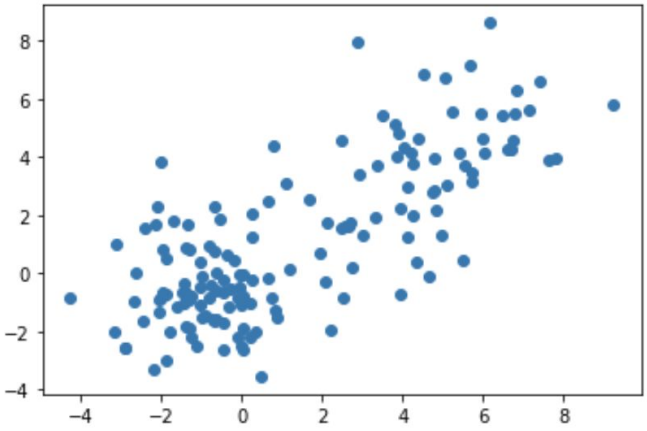
2. Upon researching libraries for clustering algorithms, we discovered "sklearn" offered the best implementations of the two algorithms.
    a. For K-Means: we imported "from sklearn.cluster import KMeans"
        i. When comparing the answer (centroid of each cluster) from our code versus that of the library, the answers were identical. Therefore, we feel that our code does not require further improvement in terms of determining the centroid of each cluster. But we discovered that the library offers a function (kmeans.labels_) that outputs which centroid each data point belongs to, which we did not implement in our code (because the homework did not state to do so), so we can improve our code in this manner.
    b. For EM with GMM: we imported "from sklearn.mixture import GaussianMixture"
        i. As stated above, reaching convergence was a challenge. The sklearn library offers attributes such as "converged_" and "n_iter_" that outputs a boolean value (True or False) if convergence was reached and how many iterations it took to reach convergence, respectively. Upon comparison, the library was able to reach convergence within a few (~5) iterations but our code truly never reaches convergence, thus we set the iteration to 100. Possible ways to improve our code would be to create functions that indicate whether convergence was reached and how many iterations it took to reach convergence, similar to that of the library. Due to the difference in the output results between these library functions and our code, we can also improve the efficiency and/or mathematical formula of our code. Another possible improvement to our code is implementing a tolerance or convergence threshold, where the EM iterations will stop when the lower bound (the E-step/recalculating r_{ic}) is below this threshold, which the library default sets to 0.001.

c. Another overall improvement to our code would be visualizations because visualizations are very useful to understanding the clustering algorithm concepts. Since Part 1 restricts the use of libraries, we mostly refrained from using plots until Part 2 of the assignment.

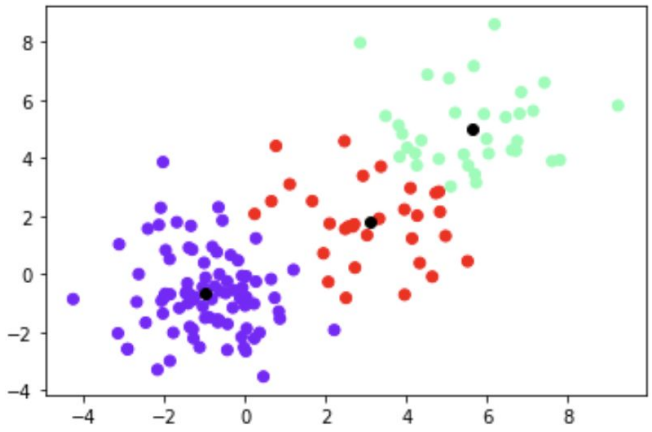
   i. Within Part 1, we used a plot to visualize our solution when using the EM with GMM algorithm:



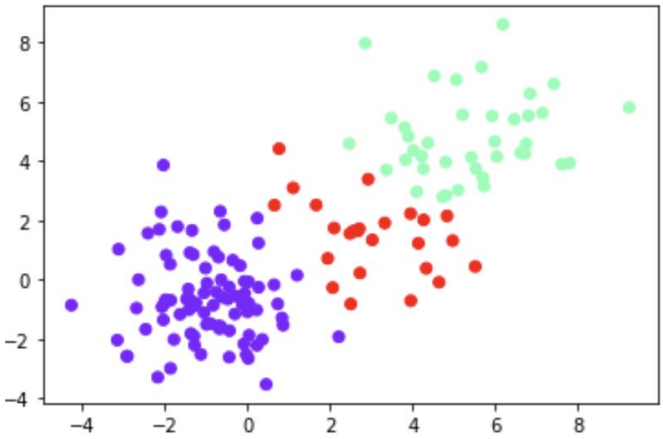
   ii. In Part 2, we used matplotlib to initially view our data points:



Next, we plotted the clusters and centroids from the K-Means library:



And finally, we plotted the clusters from the EM with GMM library:



3. Since K-Means and EM with GMM are types of unsupervised learning, below are a list of interesting applications that use these clustering algorithms:
    a. Spam emails are a nuisance. To avoid these annoying emails from entering your inbox, spam filters can be created by using the K-Means clustering algorithm in order to classify if an email is spam or legitimate. First, different features of an email (eg. header, greeting, sender, content) are identified and the data are grouped together. The K-Means algorithm is then used on the unlabelled data in the classification process. After training the model, it will be able to flag spam emails.
    b. The K-Means clustering algorithm can also be used for customer segmentation, where companies want to divide their customers into groups based on common characteristics/features so they can market and advertise to each group effectively (eg. Amazon or Youtube recommendations). K-Means accomplishes this by forming clusters (ie. groups) based on similar interests/features (eg. spending or watching behavior, respectively).

c. Because EM with GMM allows for soft-membership, this clustering algorithm would be useful for detecting epileptic seizure from EEG signals. EEG data is a type of time-series data, thus decomposition is required when pre-processing the data in order to determine the remainder component (ie. the irregularities part of the EEG once the trend and seasonal components are removed). But due to the fact that white noise is common when measuring neural activity from an EEG (due to head/eye movement and the environment), it is difficult to determine what is a signal from a stimulus versus white noise. Therefore, the EM algorithm will be able to classify the EEG signals using the probability density function (ie. probabilistic cluster assignments).

Contributions:

- ★ For Part 1, Lisa and Shagun collectively discussed the concepts of both clustering algorithms and decided to initially code the algorithms individually. From the individual work, we then picked the best code to submit, which was Lisa's code for the K-Means algorithm and Shagun's code for the EM with GMM algorithm.
- ★ Part 2 and 3 were researched and completed by Lisa.
- ★ The report was compiled by Lisa and reviewed by Shagun.