DSCI 552 – Assignment 2: K-Means and Expectation Maximization with GMM

Team Members	Sanjana Gopnal Swamy (gopnalsw@usc.edu)
	Amit Sankhla (asankhla@usc.edu)

Implementation:

For implementation of both the algorithms, we have used standard python libraries such as numpy and python list/dict.

K-Means Algorithm

The approach to implement this algorithm has three main parts corresponding to the three steps.

First, randomly choose K centroids. In this assignment K=3.

Second, assign each data point to its nearest centroid. For this, we have initially added a new entry in every row of the data list to denote the centroid a particular point belongs to. To calculate the distance, we use Euclidean Distance to calculate the straight-line distance from point to centroid. For each point, we calculate its distance from all the three centroids and pick the nearest one.

Third, for each cluster, get a new centroid by using the “numpy.mean” function. Then, repeat step 2 and 3 until the new centroids do not change as compared with the previous ones. ‘max_num_of_iterations’ is a breakpoint set in case of a loop and is set to 200.

One of the sets of centroids with K-Means (rounded to 3 decimals) -

Centroid 1 - [5.433, 4.863]

Centroid 2 - [2.883, 1.358]

Centroid 3 - [-1.039, -0.679]

Expectation Maximization algorithm using a Gaussian Mixture Model (GMM)

The Gaussian mixture modelling used Expectation-Maximization Algorithm to cluster the datasets into different clusters. The algorithm keeps iterating “E-Step” and “M-Step” until the results converge. The most difficult problem we faced is determining what is a “good start” for the algorithm. Different mean points in “E-Step” can affect the result due to the difference in calculation while clustering the points in the very first iteration. At first, we randomly selected points from datasets. However, the three resulting clusters varied a lot among different runs since the “start centroids” are chosen randomly.

Then we decided to use K-means to initialize the “start centroids” for GMM. The reason for doing this is by the fact that centroids resulting from K-means have converged. It is more stable than randomly choosing the start centroid. The resulting clusters were now much stable as compared to choosing “start centroids” randomly.

There are two breakpoints set to for result convergence: max_iterations and threshold, which are both set to 200 and 0.01 respectively.

<u>One of the sets of Amplitude, Mean and Covariance (rounded to 3 decimals):</u>

Amplitudes:

[0.51, 0.033, 0.457]

Means:

[[-0.915, -0.566], [-3.091, -2.261], [4.212, 3.239]]

Covariance Matrix:

[[[0.854, -0.335], [-0.335, 1.883]],

[[0.458, -0.544], [-0.544, 0.657]],

[[4.139, 2.571], [2.571, 5.258]]]

<u>Contributions:</u>

1: Sanjana’s contribution

- Implemented code for Expectation Maximization algorithm with GMM

2: Amit’s contribution

- Implemented KMeans Algorithm
- Wrote the Report