STATS 302 Homework 5

Name: Chenglin Zhang
NetID: cz155

Q1

We generate a dataset which is made up of two circles and we use the KMeans algorithm with different initializations to see the clustering result.

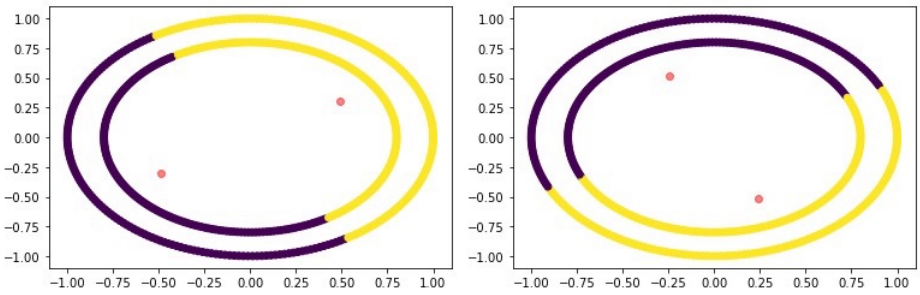

Figure 1: Example of two different initializations give different final clusterings for KMeans

As we can see, in the left figure, the initial points (centroids) are generated on the top-right and bottom-left. In the right figure, the initial centroids generated on the top-left and bottom-right. They yield different clustering results.

Q2

Epsilon (ϵ) is the maximum radius of the neighborhood. Data points will be valid neighbors if their mutual distance is less than or equal to the specified epsilon. In other words, it is the distance that DBSCAN uses to determine if two points are similar and belong together. A larger epsilon will produce broader clusters (encompassing more data points). Thus, if a data point q is reachable from p, with a larger ϵ, it will still be reachable because it is within the circle of radius ϵ.

Minimum Points (minPts) is the minimum number of data points within the radius of a neighborhood (i.e. ϵ) for the neighborhood to be considered a cluster. The initial point is also included in the minPts. A lower minPts helps the algorithm build clusters easily (with more potential noise or outliers). If a point is recognized in the cluster, then with a smaller minPts, it will remain in the cluster because the actual number of points inside the cluster does not change, so the point remains reachable.

Q3

Assume u_i is the corresponding eigenvector of the eigenvalue λ_i of the inverse of the covariance matrix Σ (which is Σ^{-1}). Assume Σ is a $n \times n$ matrix.

$$\begin{aligned}\Sigma^{-1} &= U\Lambda^{-1}U^{-1} \quad \text{(eigenvalue decomposition)} \\ &= U\Lambda^{-1}U^{T} \quad \text{(because } \Sigma \text{ is symmetric)} \\ &= \sum_{i=1}^{n} \lambda_i^{-1} u_i u_i^T\end{aligned}$$

Thus,

$$\begin{aligned}D &= (X-\mu)^T\Sigma^{-1}(X-\mu) \\ &= (X-\mu)^T \left(\sum_{i=1}^{n} \lambda_i^{-1} u_i u_i^T\right)(X-\mu) \\ &= \sum_{i=1}^{n} \lambda_i^{-1}(X-\mu)^T u_i u_i^T (X-\mu) \\ &= \sum_{i=1}^{n} \lambda_i^{-1}(X-\mu)^T u_i u_i^T (X-\mu) \\ &= \sum_{i=1}^{n} \lambda_i^{-1}\left(u_i^T(X-\mu)\right)^2 \\ &= \sum_{i=1}^{n} \left(\lambda_i^{-\frac{1}{2}} u_i^T(X-\mu)\right)^2\end{aligned}$$

Because $X \sim \mathcal{N}(\mu, \Sigma)$, $(X-\mu) \sim \mathcal{N}(0, \Sigma)$, $\lambda_i^{-\frac{1}{2}} u_i^T(X-\mu) \sim \mathcal{N}(0, (\lambda_i^{-\frac{1}{2}} u_i^T)\Sigma(\lambda_i^{-\frac{1}{2}} u_i^T)^T)$.

$$\begin{aligned}(\lambda_i^{-\frac{1}{2}} u_i^T)\Sigma(\lambda_i^{-\frac{1}{2}} u_i^T)^T &= (\lambda_i^{-\frac{1}{2}} u_i^T)\Sigma(\lambda_i^{-\frac{1}{2}} u_i) \\ &= \lambda_i^{-1} u_i^T \Sigma u_i \\ &= \lambda_i^{-1} u_i^T \left(\sum_{j=1}^{n} \lambda_j u_j u_j^T\right) u_i \\ &= \sum_{j=1}^{n} \lambda_i^{-1} u_i^T \lambda_j u_j u_j^T u_i\end{aligned}$$

In the problem statement, we know that u_i are orthogonal to each other, which means

$$(\lambda_i^{-\frac{1}{2}} u_i^T)\Sigma(\lambda_i^{-\frac{1}{2}} u_i^T)^T = \sum_{j=1}^{n} \lambda_i^{-1} u_i^T \lambda_j u_j u_j^T u_i = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$$

Thus, $D \sim \mathcal{N}(0, I)$, in which I is a $n \times n$ identity matrix, which is equivalent to

$$Prob(Z_1^2 + Z_2^2 + \cdots + Z_n^2 = D^2, Z_i \sim \mathcal{N}(0, I))$$

Thus, we have proved that the square of the Mahalanobis distance in $\mathbb{R}^n$ is χ_n^2 distributed when x is sampled from a normal distribution $\mathcal{N}(\mu, \Sigma)$.

Challenge #1

We make the Gaussian blobs and the moons data like what is shown below:

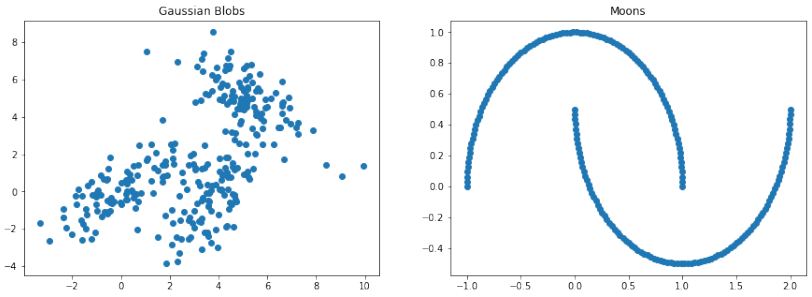

Figure 2: Datasets

The inertia of the KMeans clustering model for the two dataset is shown below:

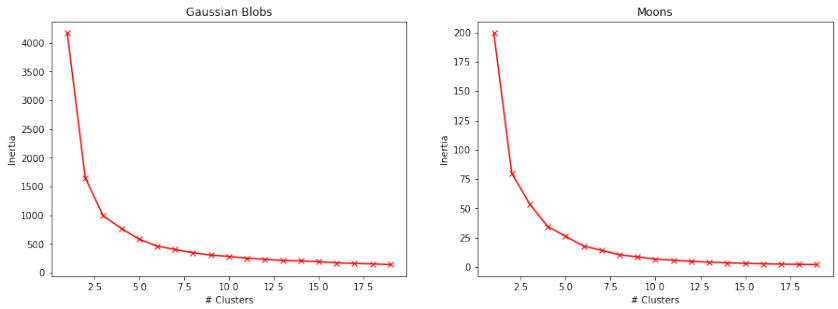

Figure 3: Inertia of the KMeans

Thus, we choose $K = 3$ for the Gaussian blobs and $K = 4$ for the moons as they are the turning points in their inertia curves. The clustering results is shown below:

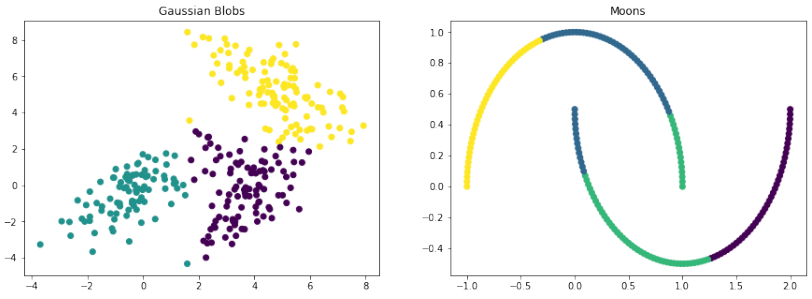

Figure 4: Clustering results

For the Gaussian blobs, the clustering result is satisfactory. For the moon blobs, it is certainly not what we want. For the moons dataset, DBSCAN will work well in clustering, while KMeans and GMM will not.

Challenge #2

The dataset is generated by

```
X_blobs = []
X_blobs.extend(np.random.multivariate_normal([-0.5,-0.5],
    [[-0.5,0.4],[0.4,-0.5]], 100))
X_blobs.extend(np.random.multivariate_normal([1,1],
    [[-0.5,0.48],[0.48,-0.5]], 100))
X_blobs.extend(np.random.multivariate_normal([2,2],
    [[-0.2,0.3],[0.3,-0.6]], 100))
X_blobs = np.asarray(X_blobs)

gmm = GMM(n_components=3,covariance_type='full').fit(X_blobs).
    predict(X_blobs)
dbscan = DBSCAN(eps=0.4, min_samples=5).fit(X_blobs)
kmeans = KMeans(n_clusters = 3).fit(X_blobs)
```

The original data, the performances for the Gaussian Mixture Model, DBSCAN and KMeans are shown below.

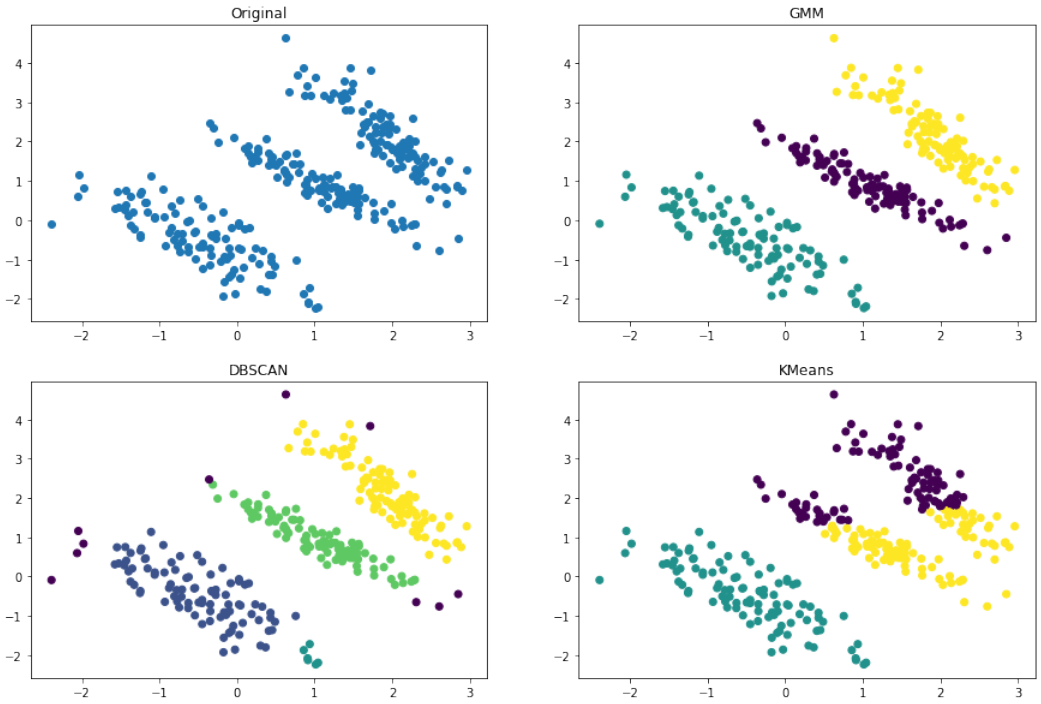

Figure 5: Clustering results

As we can see, both the GMM and DBSCAN succeed, while KMeans fails for this dataset.

Challenge #3

The original representation of each pixel of the PNG image is through the array [R, G, B, transparency], in which R, G, B are float point values ranging from 0 to 1. An example of a PNG pixel may be [0.03529412, 0.08627451, 0.01176471, 1.], which is the output of the top-left pixel in the PNG picture "ladybug.png".

Figure 6: ladybug.png

To do spatial segmentation, we need to take the actual position of each pixel into account when doing clustering. For example, we can augment the array representing each pixel from size (4,) to size (6,) by adding the row index and column index to the array. For example, for the first pixel, the array is augmented into [0.03529412, 0.08627451, 0.01176471, 1. , 0, 0]. By doing so and modifying the objective function of the clustering algorithm, we can conduct spatial segmentation.