Data Mining and Bioinformatics

Project 2

Clustering Algorithms

Saeb Ragani Lamooki | 50205221

K-means Algorithm

In K-means algorithm K number of centroids are initialized first and the data points will be split into K clusters based on their minimum distance with the centroids. The centroids will be updated as the centroid of each clusters and the data will be re-clustered. This process will repeat until the centroids do not change in the centroid update step.

Implementation

We first stored the data in a matrix composed of the data points in each row and the features in columns. The second row of the input file was stored in a column matrix as the external ground truth cluster number. The centroids were initialized as K number of the input data points. The Euclidean distance of each data point with each centroid was compared and the point was assigned to the cluster belonging to closest centroid. After generating the cluster index for each data point, the mean of the matrix of the data points in each cluster was calculated as the new centroid of that cluster. The cluster assignment and centroid update was repeated until the centroids did not change in centroid update. Maximum number of iterations was set to 100.

Incidence matrix, Rand Index, and Jaccard Coefficients

The calculation of incidence matrix, Rand index, and Jaccard coefficient is the same for all the 5 algorithms. We explain their calculation here and it will be skipped for the next algorithms.

We created the incidence matrix for the algorithm's clustering results as well as the ground truth clustering results. For an input data with the length of N, each data point was compared against all other data points in a N x N matrix. If the two data points belonged to the same cluster the value of 1 was assigned to the corresponding element of the incidence matrix, otherwise 0 was assigned. The two incidence matrices were compared against each other element by element and Rand Index and Jaccard coefficient were calculated with the following equations:

$$Rand = \frac{|M_{11}| + |M_{00}|}{|M_{11}| + |M_{00}| + |M_{10}| + |M_{01}|}$$

$$Jaccard = \frac{|M_{11}|}{|M_{11}| + |M_{10}| + |M_{01}|}$$

Where M_{11}, M_{00}, M_{10}, and M_{01} are defined as follows:

M_{11} : C_{ij}=1 and P_{ij}=1

M_{00} : C_{ij}=0 and P_{ij}=0

M_{10} : C_{ij}=1 and P_{ij}=0

M_{01} : C_{ij}=0 and P_{ij}=1

C_{ij} represents the elements in the algorithm's incidence matrix and P_{ij} represents the elements in the ground truth's incidence matrix.

Results

The data points are visualized on 2D graphs by applying PCA and the clusters are distinguished by different colors. The graphs for "cho" and "iyer" data are presented below with the corresponding Rand Index and Jaccard coefficient.

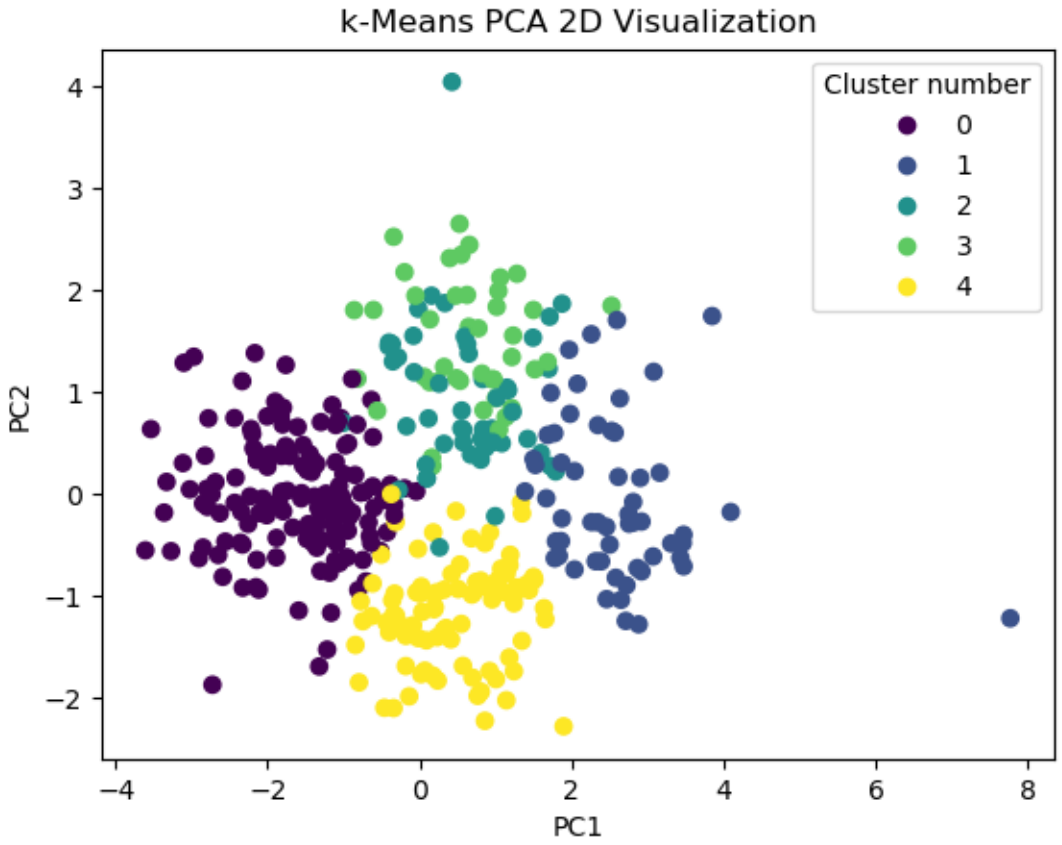


Figure 1 Rand Index = 0.81, Jaccard Coefficient = 0.43

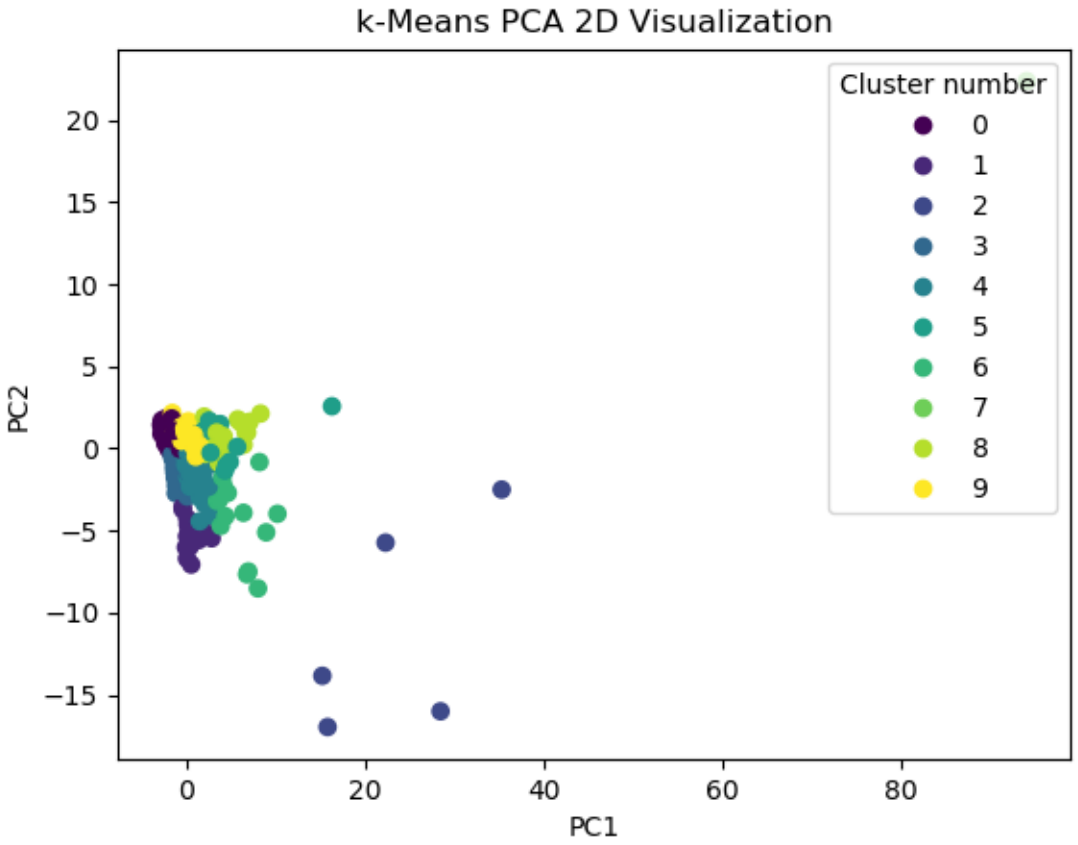


Figure 2 Rand Index = 0.74 Jaccard Coefficients = 0.33

Hierarchical Algorithm

Implementation

We applied hierarchical agglomerative clustering with min algorithm on our data set. We first stored the data using the same function we created in K-means algorithm. Then we created a distance dictionary of the data index as the keys and a dictionary as the values. Within each of these dictionaries again the data indices were used as the keys and the distance of the outer dictionary key and the inner keys was set to the value. Next, we created a function to find the minimum distance among all the distances and output the outer and inner keys associated with them. Then we updated the distance dictionaries by merging the two closest points and update all the distances by using the minimum distance of the two merged points with all other points as the distance of the newly generated point. The key for the new point was composed of the old point indices separated by a + sign. We implemented the closest points search and the distance dictionary update in a for loop until the number of keys in the outer dictionary reached a certain value.

After the loop was completed, in order to find and store the cluster index for all the points assigned to the same dictionary, we split each key element using the + separator and assigned a cluster number to all of them. Then we created the incidence matrices similar to K-means algorithm and calculated the Rand index and Jaccard coefficients similarly.

Results

We visualized the data by PCA and illustrated the data belonging to the same clusters by colors. The results are presented in the following graphs with the Rand index and Jaccard coefficients.

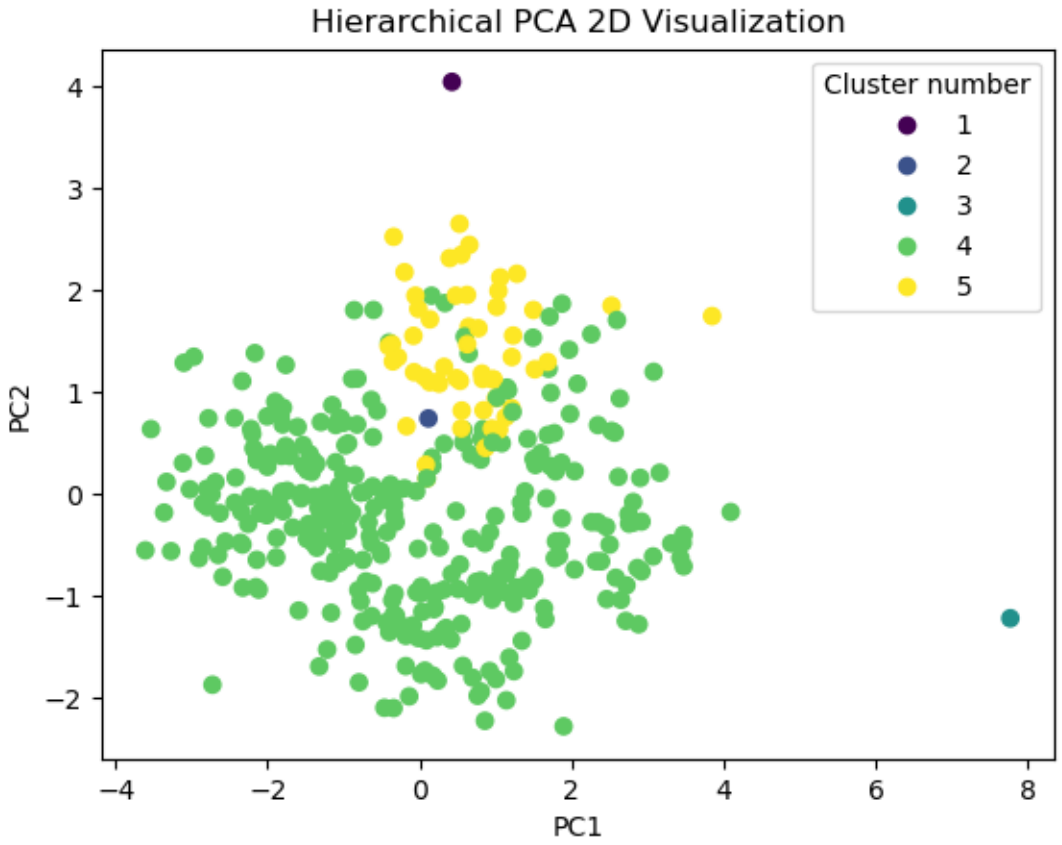


Figure 3 Rand = 0.40 Jaccard = 0.24

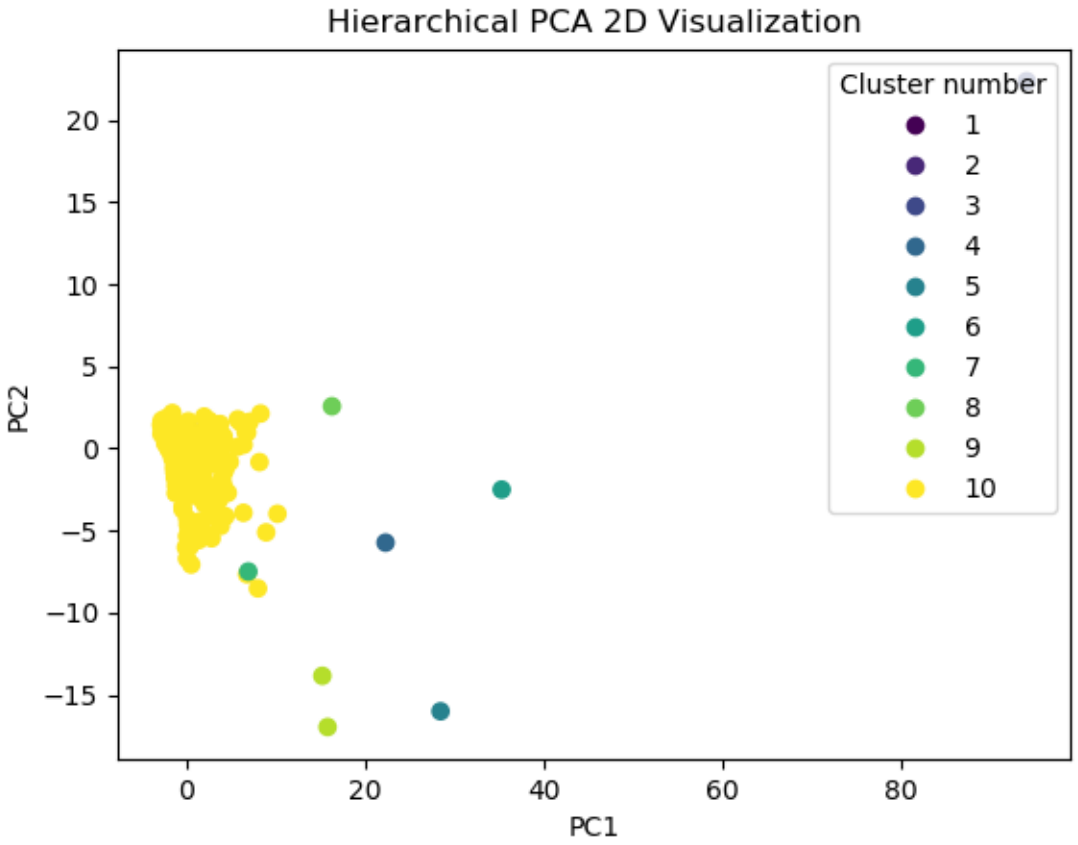


Figure 4 Rand = 0.19 Jaccard = 0.16

Density-based Algorithm

Density-based spatial clustering of applications with noise (DBSCAN) is a density-based algorithm which is designed to cluster higher density regions, separated with lower density regions. The data points classify into three types, core, border, and noise points. Points with more than or equal to Min-points within their epsilon neighborhood are considered core points, while the points with less than Min-points around them are considered noise unless they have a core point in their epsilon neighborhood.

Implementation of DBSCAN

The features were extracted from the input files and arbitrary values were assigned to Min-points and epsilon. The distance matrix was created such that ij_{th} member of the matrix represents the Euclidean distance between data points i and j. Next, we define a list of zeros as the cluster index list. We start to visit non-visited points and count the number of points in their epsilon neighborhood and assign them to noise if it is less than Min-points. If the number of points within their epsilon neighborhood is more than or equal to Min-points, it will be considered as core point therefore we try to expand the cluster around this core point and find all the directly and indirectly reachable points from it. During this step the reachable points that were previously labeled as noise will be added to the cluster as border points and all the non-visited points will be inquired for all their neighbors. All these reachable points will join the current cluster and after the current cluster reached to all the reachable points, the algorithm moves to the next non-visited point and repeat the above steps.

Ran index and Jaccard coefficient were calculated similar to previous algorithms.

Results

The data is visualized by PCA in the following graphs and the clusters are illustrated using different colors.

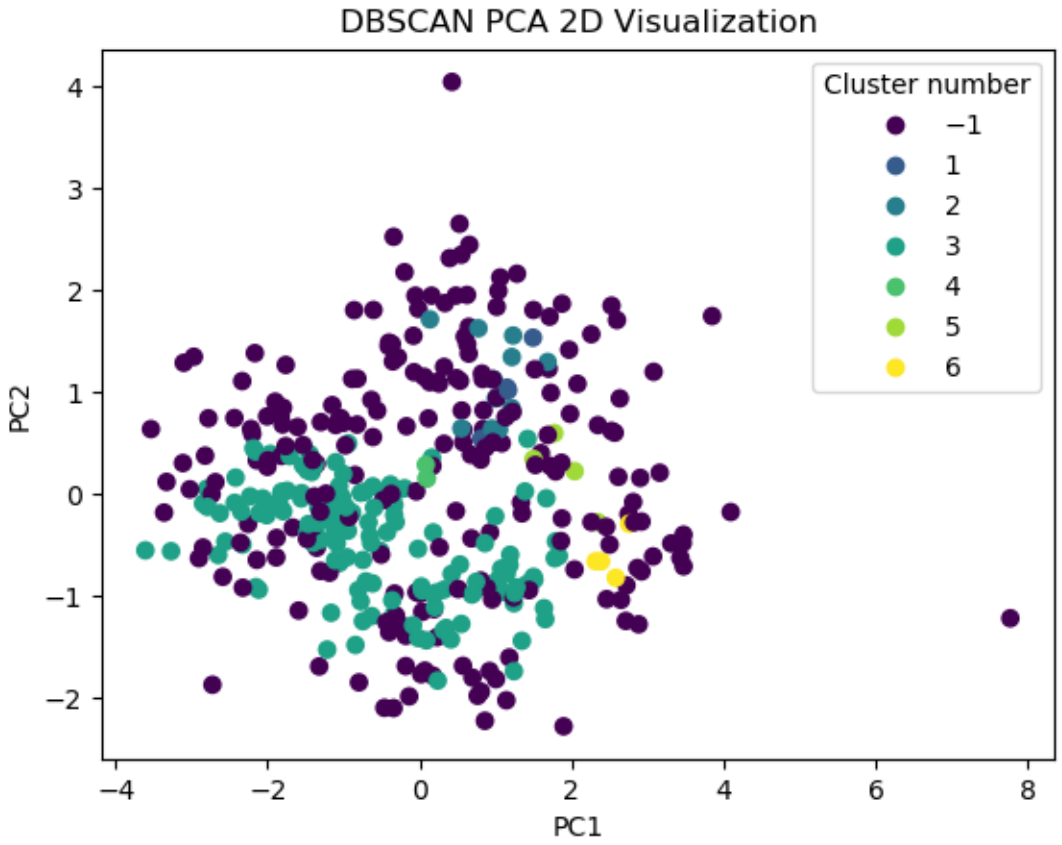


Figure 5 Rand Index = 0.54, Jaccard Coefficients = 0.204, MinPts = 3, epsilon = 1.0

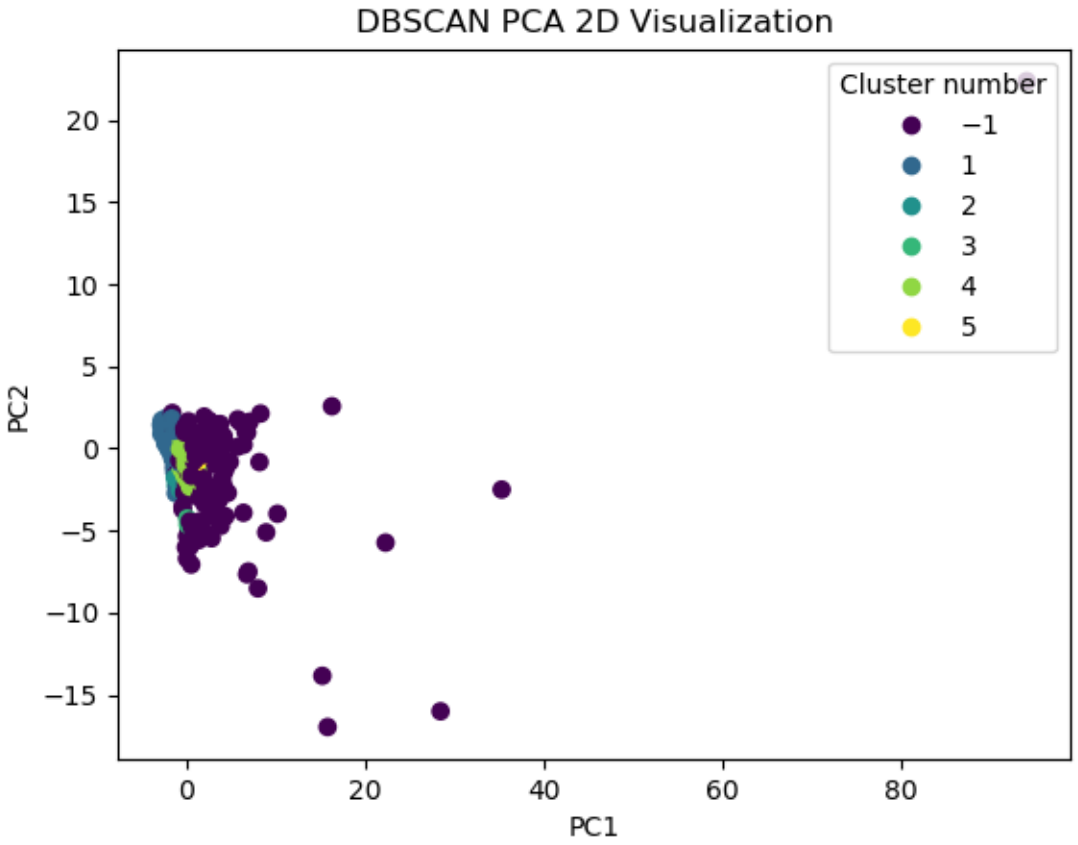


Figure 6 Rand Index = 0.65, Jaccard Coefficients = 0.28, MinPts = 3, epsilon = 1.0

Gaussian Mixture Model Algorithm

In the gaussian mixture model (GMM) algorithm the data is assumed to be representable by a number of gaussian distributions. The algorithm maximizes the log likelihood of the data by updating the mean and covariance of the gaussian distributions as well as the weights assigned to each distribution.

Implementation

We extract the feature matrix from the input file. Then we initialize the covariance matrices, means, and the weights for an arbitrary K number of gaussian distributions. Next, we implemented the E-step by computing the r_{ik}'s, from the following equation:

$$r_{ik} = \frac{\pi_k \mathcal{N}(x_i|u_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|u_k, \Sigma_k)}$$

Then we implemented the M-step where we used the computed r_{ik}'s to update the weights, means, and the covariance matrices as follows:

$$\pi_k = \frac{\sum_i r_{ik}}{n}$$

$$\mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}$$

$$\Sigma_k = \frac{\sum_i r_{ik}(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_i r_{ik}}$$

The E-step and M-step were repeated in a for loop and the corresponding log-likelihood was calculated as follows:

$$\ln p(x|\pi, \mu, \Sigma) = \sum_{i=1}^{n} \ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)\}$$

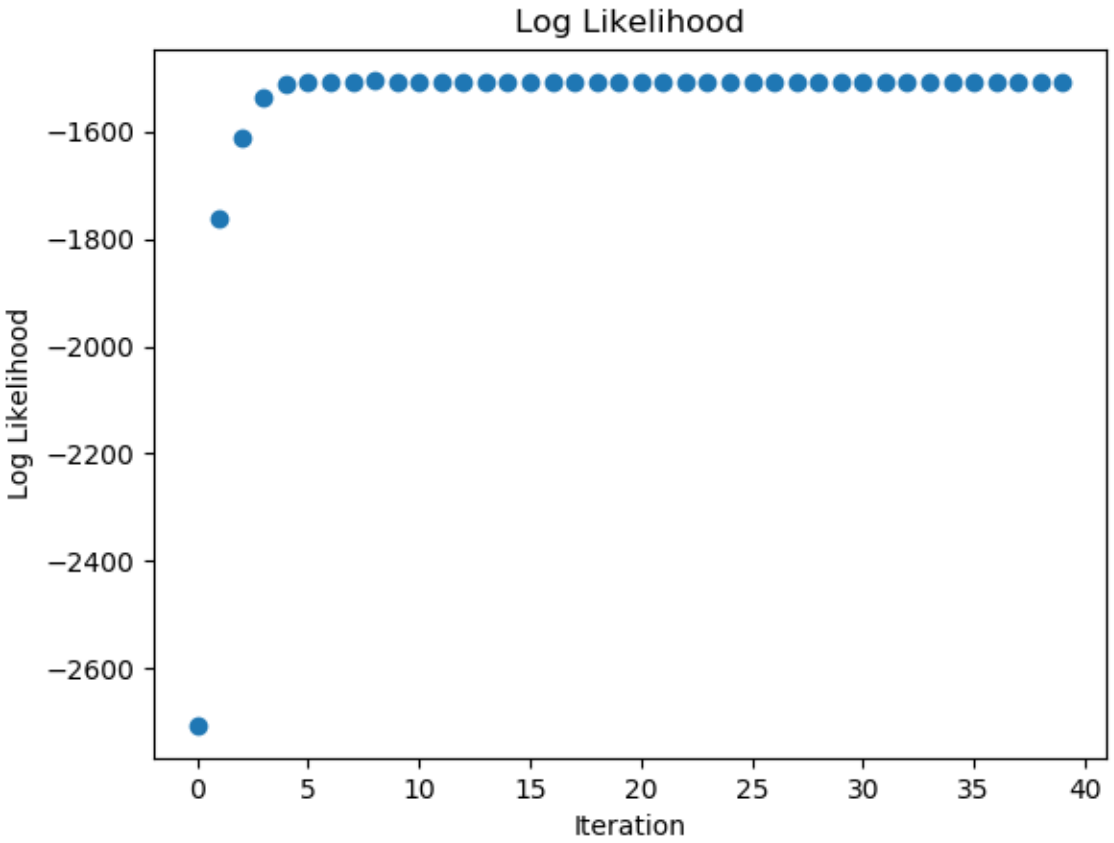
The for loop continues until the log-likelihood converges to a value and does not increase anymore.

After the convergence was reached, the data points were assigned to the gaussian distribution for which their r_{ik} value was maximum.

Results

Similar to the other algorithms, we visualized the data with PCA and illustrated each cluster by a different color.

The results for the "cho" dataset was created with smoothing value = 10^{-9}, maximum iterations = 40, and convergence threshold = 10^{-9}. The code did not converge within the maximum iterations, however the following likelihood graph shows stable converging values.



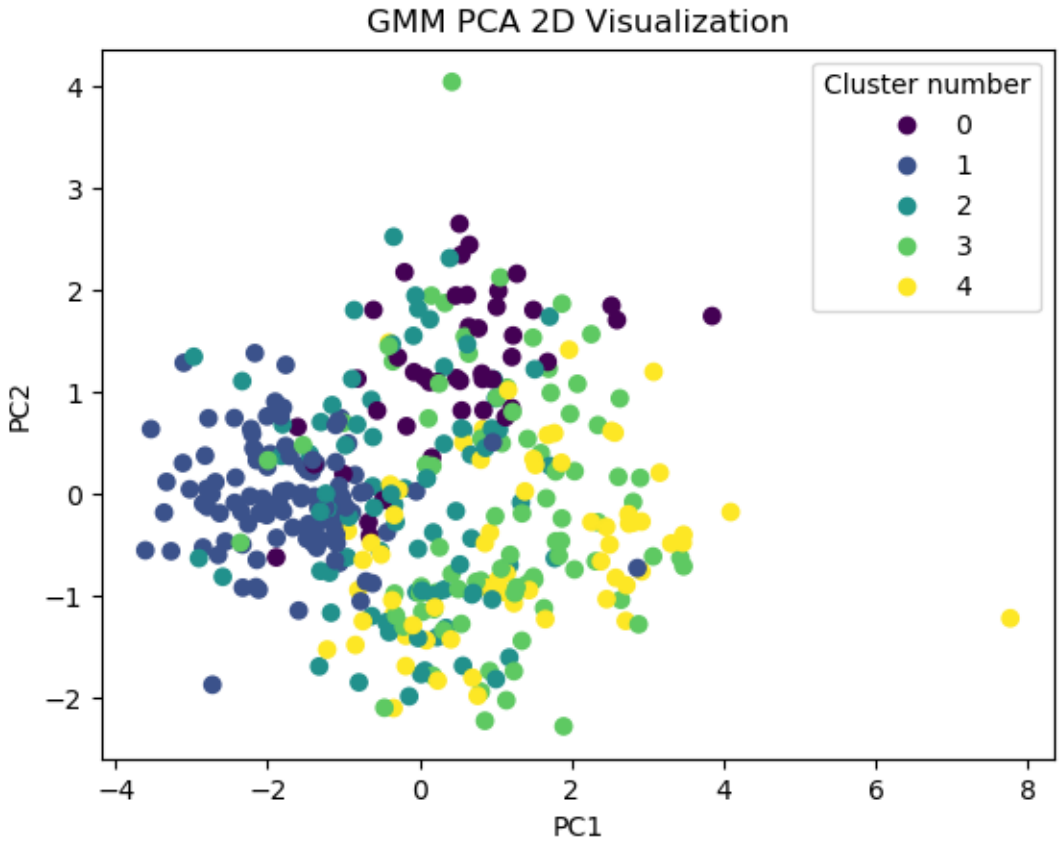


Figure 7 Rand Index= 0.74 Jaccard Coefficients= 0.27

For “iyer” dataset, the algorithm did not work with smoothing value = 10^{-9}, therefore smoothing value of 10^{-5} was used instead. However, even with this smoothing value we did not get the log likelihood values and we could not check convergence criterion. The final clustering, Rend index and Jaccard coefficients are presented below.

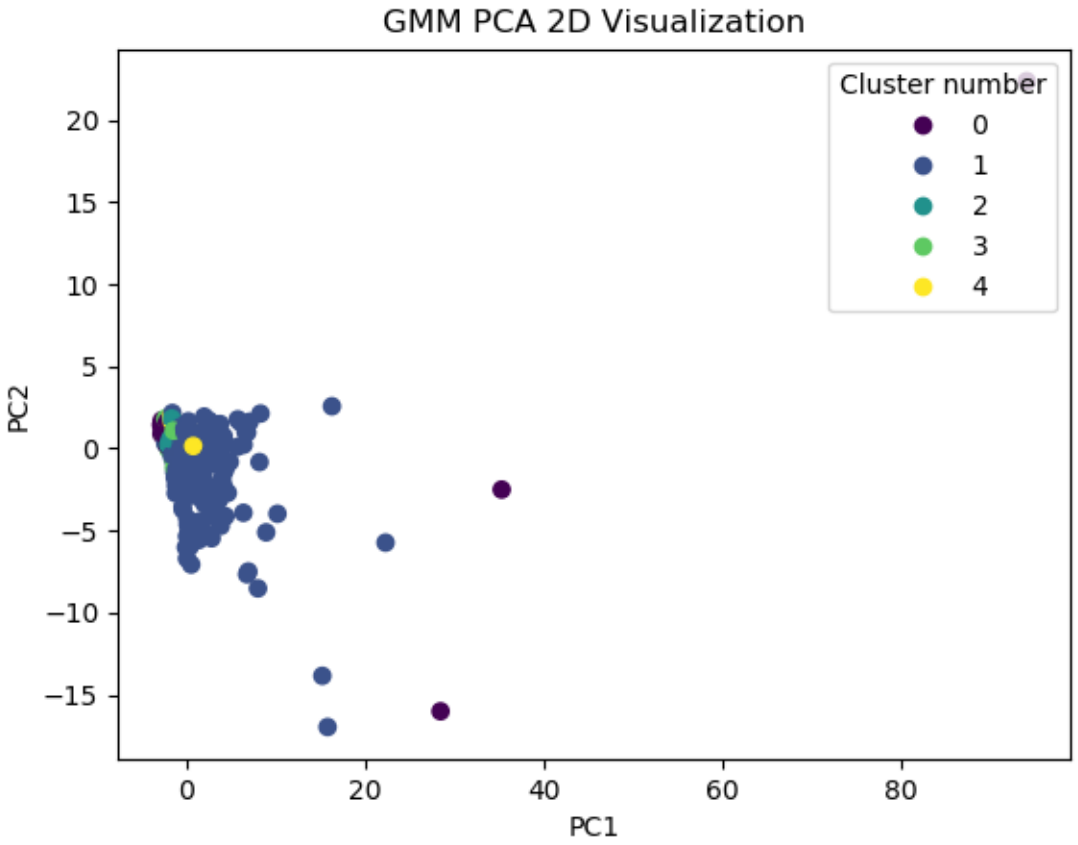


Figure 8 Rand Index= 0.71 Jaccard Coefficients= 0.198

Spectral Algorithm

Spectral algorithm is a graph-based clustering method where each data point is assumed a vertex and the vertices are connected by weighted edges. The graph will be split by a cut that minimizes the cut value and clusters the data.

Implementation

We created the data matrix similar to previous algorithms. Next, we created the gaussian kernel matrix using the following equation:

$$w_{ij} = \exp(-\| x_i - x_j \|^2 / \sigma^2)$$

Where sigma is a parameter to be tuned. The diagonal members of the gaussian matrix were enforced zero. Then we created a diagonal matrix by summing all the row values in the gaussian matrix to create the degree matrix. The Laplacian matrix wans subsequently computed by subtracting the gaussian matrix from the degree matrix. Then we calculated the eigenvalues and the eigenvectors of the Laplacian matrix and found the indices of the N smallest eigenvalues. The eigenvectors of the corresponding indices were then collected in a new matrix. We then clustered the eigenvectors matrix using K-means algorithm and used the result as the clustering output for the original data.

Results

The data were visualized using PCA and the clusters were illustrated by different colors.

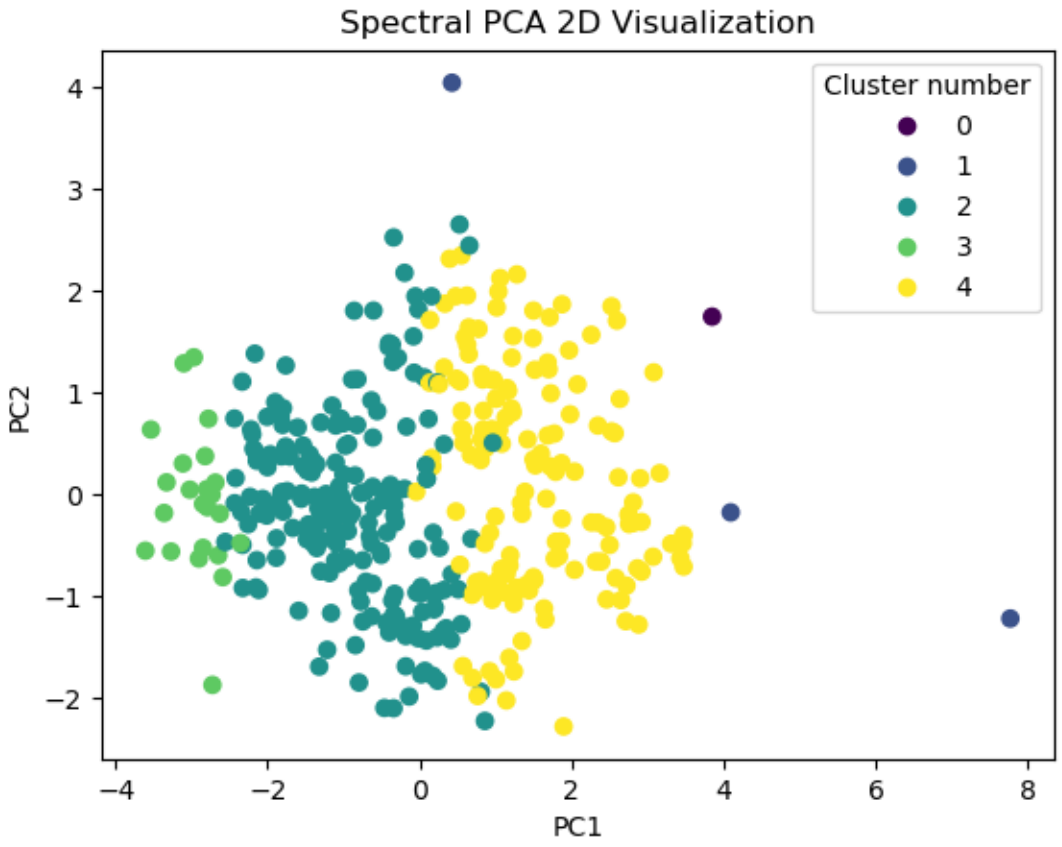


Figure 9 Rand Index = 0.63, Jaccard Coefficients = 0.29, sigma = 20, # of minimum eigen-values = 5, # of Kmeans clusters = 5

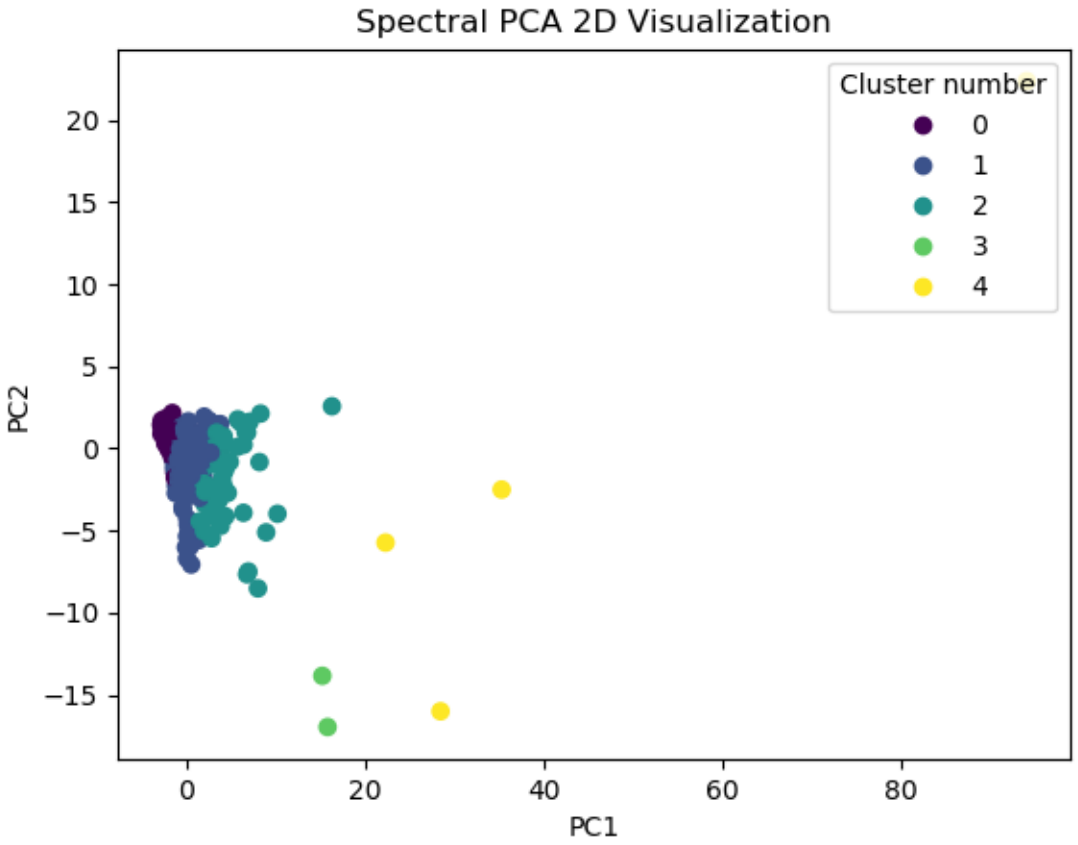


Figure 10 Rand Index = 0.66, Jaccard Coefficients = 0.28, sigma = 20, # of minimum eigen-values = 5, # of Kmeans clusters = 5