# CSE 601: Data Mining and Bioinformatics

## Project 2: Clustering Algorithms

*Submitted by*

Bhagutharivalan Natarajan Muthukkannu (50314871)
Harish Kannan Venkataramanan (50316999)
Praveen Mohan (50321225)

## K – Means Algorithm

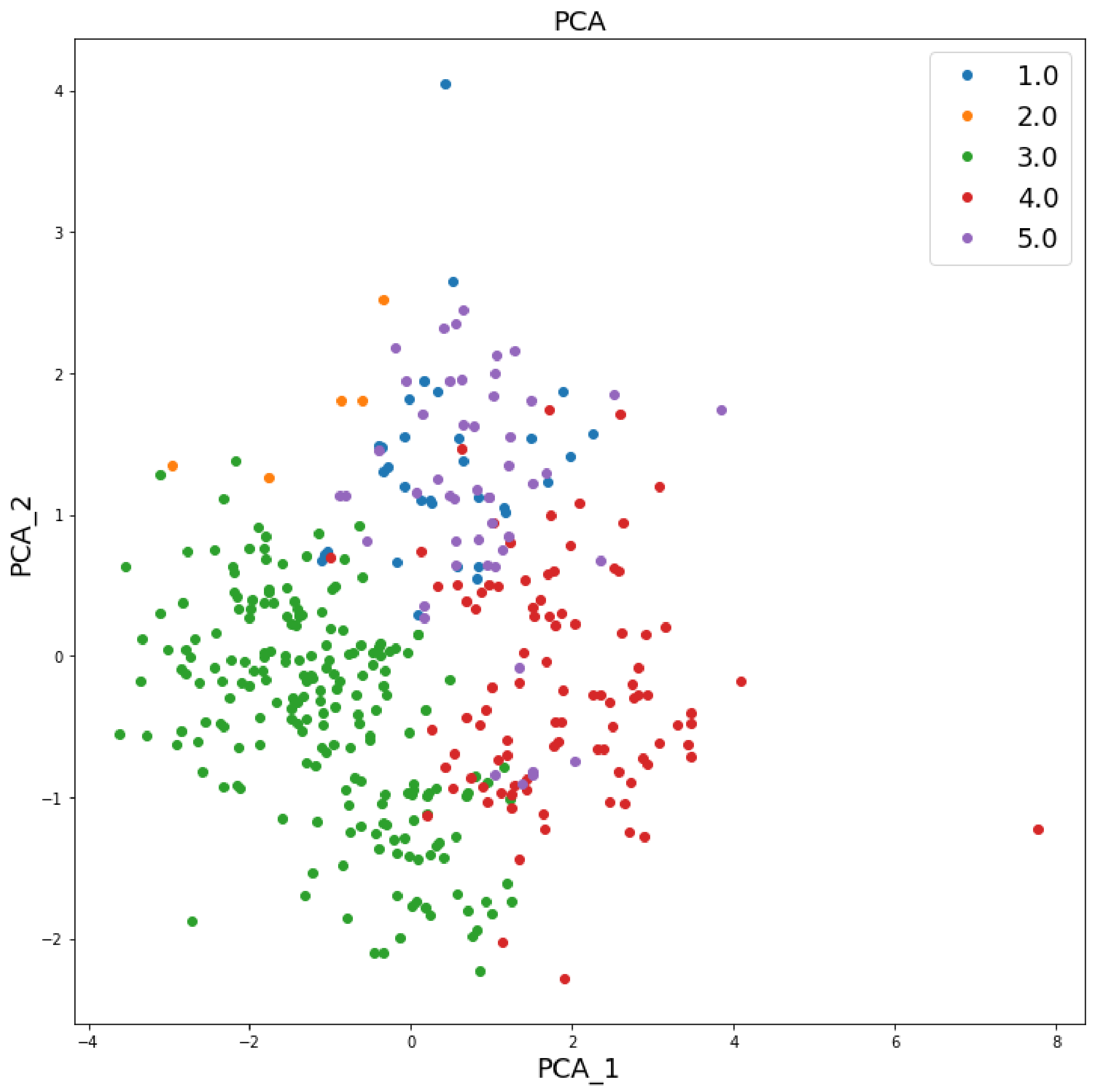
K-means is an iterative clustering algorithm that segregates the datapoints to a non-overlapping cluster.

**Pseudo code:**

```
1.  The required Libraries are imported.
2.  The input file is taken as a .txt format and stored as a data frame.
3.  The feature points are normalized to data points from (0 to 1).
4.  The number of clusters, number of iterations and initial cluster centroid (randomly generated or specific) is obtained from user input.
5.  The distance between each data point and clusters are calculated using Euclidean distance with SciPy package (pdist and squareform).
6.  The data points are assigned to the nearest cluster (i.e. the cluster which has minimum distance from the datapoint).
7.  The new centroid is formed by calculating the average of all points in a cluster.
8.  Repeat the iteration from step 4 till the sum of square error is negligibly minimum or zero (else the number of iterations specified is reached).
9.  If the data has more than two features or dimensions, the principal component analysis is implemented.
10. The clustering is evaluated using metric (Jaccard and Rand Index) and plotted.
```

**K-means clustering with cho.txt file:**

- Number of clusters = 5
- Number of iterations = 10
- Initial centroids = Randomly selected

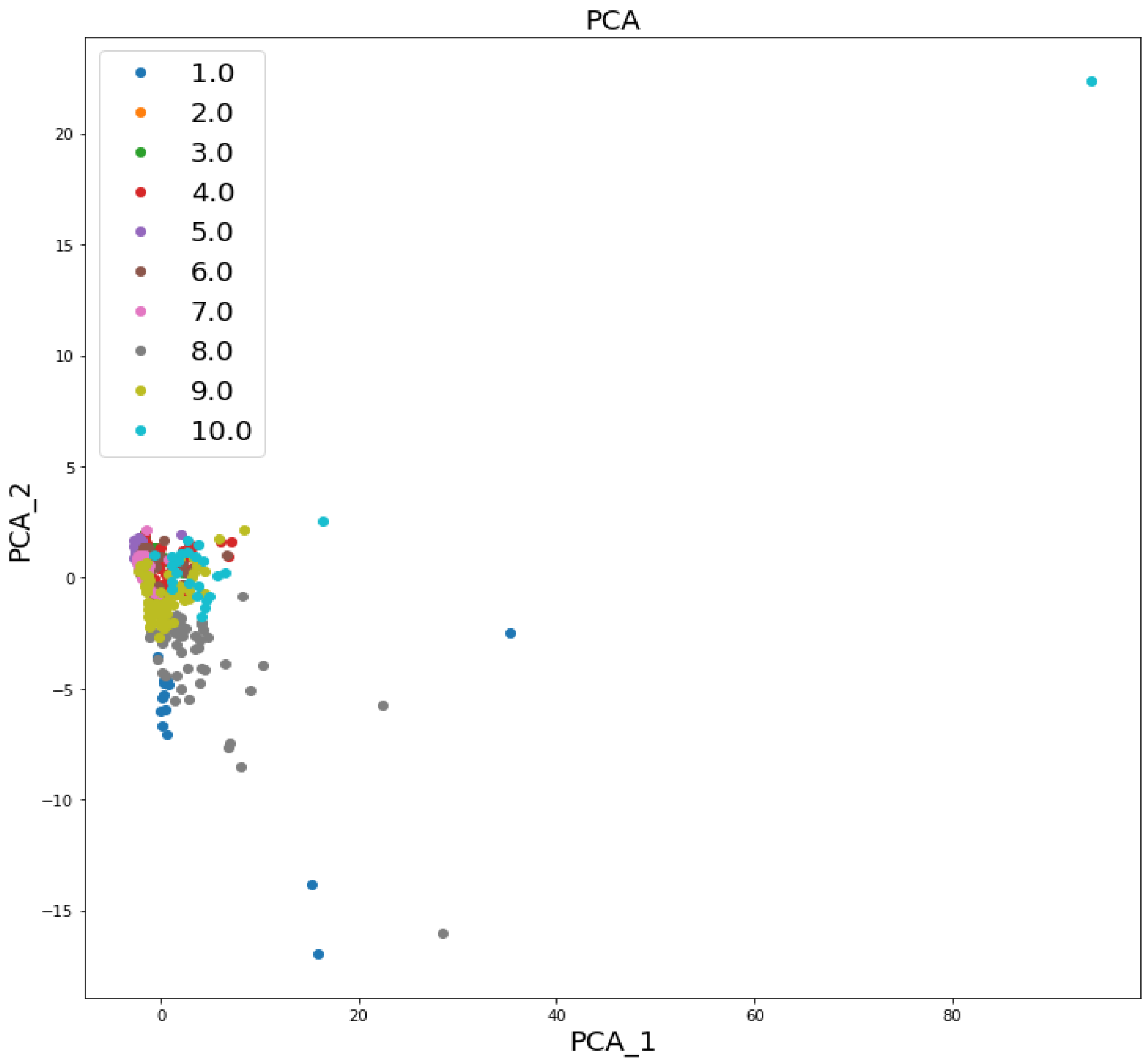


**Jaccard Index:** 0.3496

**Rand Index:** 0.7128

**K-means clustering with iyer.txt file:**

- Number of clusters = 10
- Number of iterations = 10
- Initial centroids = Randomly selected



**Jaccard Index:** 0.1801

**Rand Index:** 0.8114

**Result:**

- The cluster from the file cho.txt shows high degree of clustering even with outlier data points that are far away from centroids.
- The cluster from the file iyer.txt cluster has many dimensions and the datapoints are plotted from the PCA result. It can be seen that the data points are crowed in a certain location with notable outliers on side.

**Pros:**

- It is easy to implement than other clustering algorithms.
- The time complexity primarily depends on number of clusters k, n is the number of data points on the data sets and m being the maximum iteration.

**Cons:**

- The initial centroid determines the effectiveness of the clustering algorithm.
- Number of clusters plays a major role in the effectiveness of the algorithm.
- By calculating SSE local minimum can be identified but not the global minimum of the cluster.
- The initial clustering may lead to empty clusters.

## Hierarchical Agglomerative Clustering

Agglomerative clustering is a bottom up approach where each data point is a singleton cluster initially. At every stage of iteration, 2 clusters or datapoints are merged to one cluster, until we get the single cluster at the end.

**Single linkage (Min Approach)**

At each step of iteration, 2 datapoints with minimum distance are figured and merged into one, by updating the minimum values of 2 datapoints in n dimensional space. This method is also called as nearest neighbor method.

**Steps to build the Hierarchical clustering:**

1. User is prompted to give name of the dataset to be processed and Open_file() is used to extract the essential features from the given dataset.
2. The number of clusters to be partitioned is given by the user as input (number_clusters).
3. Initially each individual data point in the dataset is a singleton cluster and dictionary is created containing N keys corresponding to N (singleton) clusters and the values of the key are the datapoints in the cluster.
4. Euclidean distance between each and every data point is calculated using scipy.pdist().
5. The diagonal values of the matrix are set to infinity, to find the minimum element in the matrix.
6. 2 clusters with minimum Euclidian distance is found and the new cluster is updated with minimum values of 2 clusters in n dimensional space.
7. Rows and columns corresponding to cluster 2 in the distance matrix is updated with value inf

8. The dictionary is updated by merging the values present in the 2 clusters into the cluster 1 and the cluster 2 is popped out from the dictionary.
9. The step 6 to 8 is repeated, till we get the desired number of clusters.
10. Jaccard and Rand index is calculated using function jac_rand() and displayed as a output.
11. PCA is used for reducing dimensions from m feature to 2 feature and scatter plot is drawn for different clusters.

**Packages used:**

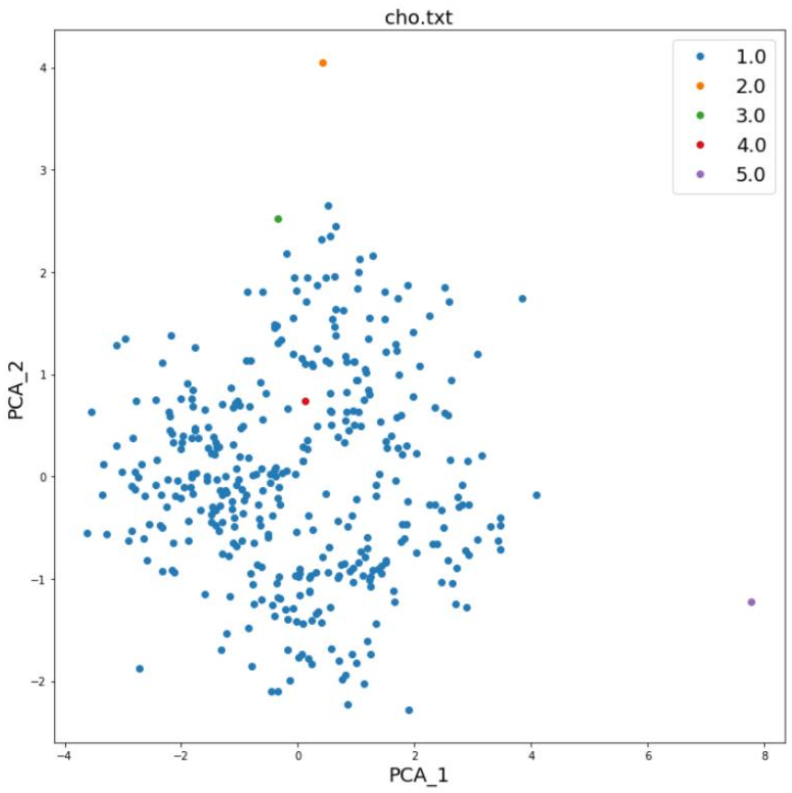
1. pandas
2. numpy
3. scipy
4. sklearn – PCA
5. matplotlib

**Advantages:**

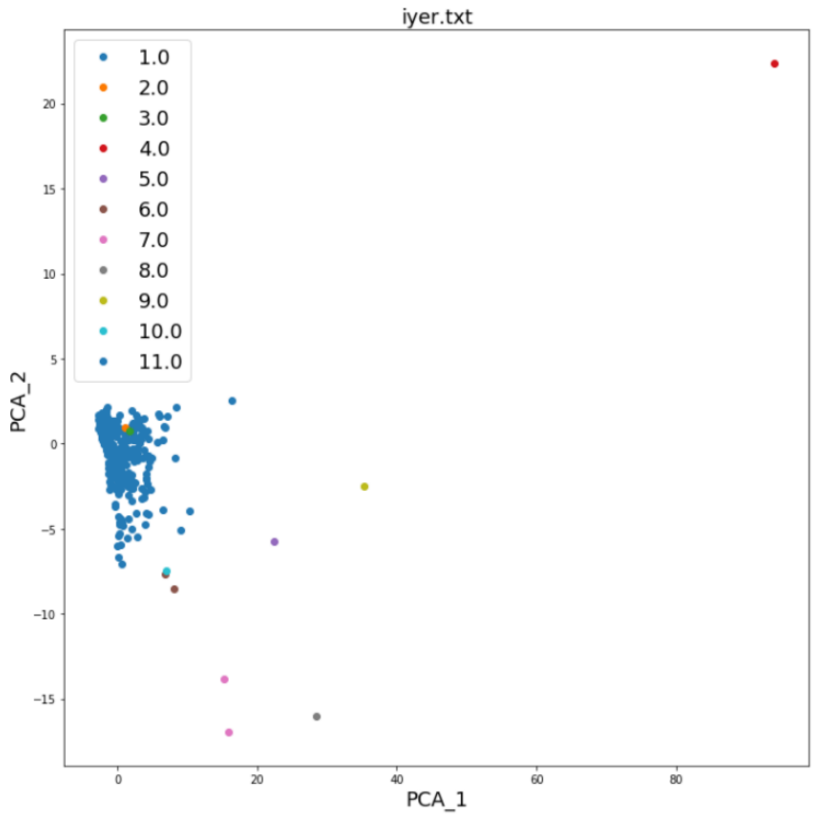
1. Hierarchical relation between the clusters are captured in the form of dendrogram.
2. We cut the dendrogram to get the desired number of clusters.

**Disadvantages:**

1. Algorithm is sensitive to outliers and noise and has high time complexity.
2. Algorithm is weak in handling different sized clusters and follows a greedy approach.
3. There is no objective function which can be optimized to get better clustering results.
4. The hierarchy formed cannot be undone.



```
Enter the File name :cho.txt
Enter the number of clusters :5
Jaccard :  0.22839497757358454
Rand :  0.24027490670890495
```



```
Enter the File name :iyer.txt
Enter the number of clusters :11
Jaccard :  0.1584602101134175
Rand :  0.1941381800223728
```

## Spectral Clustering Algorithm

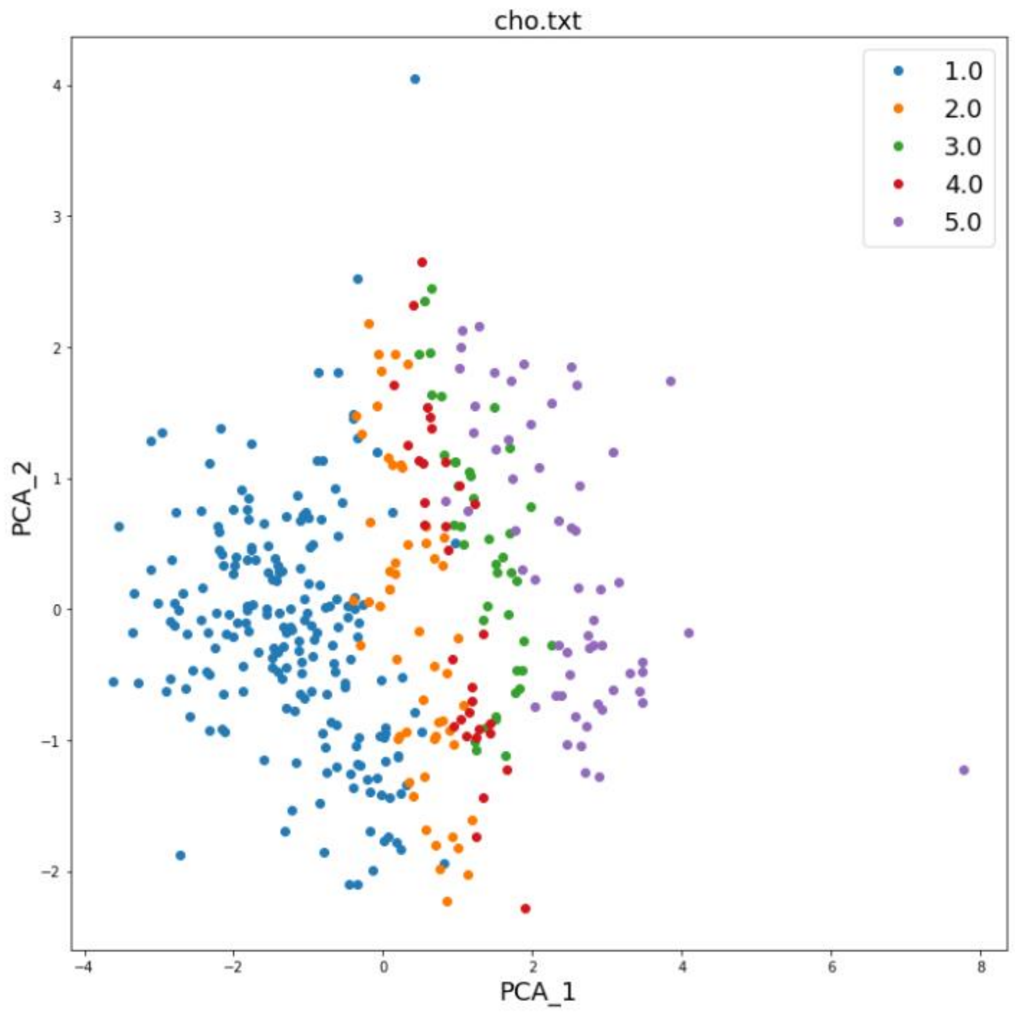
Spectral clustering is a technique used for segregation by the eigenvalues of similarity matrix for the data points to perform dimensionality reduction before clustering in fewer dimensions. The cluster points are treated as node, thus it’s a graph partition problem.

**Pseudo code:**

1. The input file is takes as a .txt format and stored as a dataset.
2. The data set is represented by a weighted graph G(V, E) where V denotes vertices and E represent the edge connecting the vertices.
3. The similarity matrix which is a square matrix is generated with the NxN, where N is the number of rows or data points.
4. The Diagonal matrix is formed by filling the diagonals with the sum of all features corresponding to the row of the similarity matrix.
5. The Laplacian matrix is created by subtracting similarity matrix from diagonal matrix.
6. Eigen values and eigen vectors are computed from the Laplacian matrix.
7. The eigen values are sorted in ascending order and the eigenvectors corresponding to the K smallest eigenvalue where eigen gap k.
8. K-means is applied to the formed N x K matrix to form clusters.

**K-means clustering with cho.txt file:**

- Number of clusters = 5
- Sigma = 4
- Number of iterations = 10
- Initial centroids = randomly selected

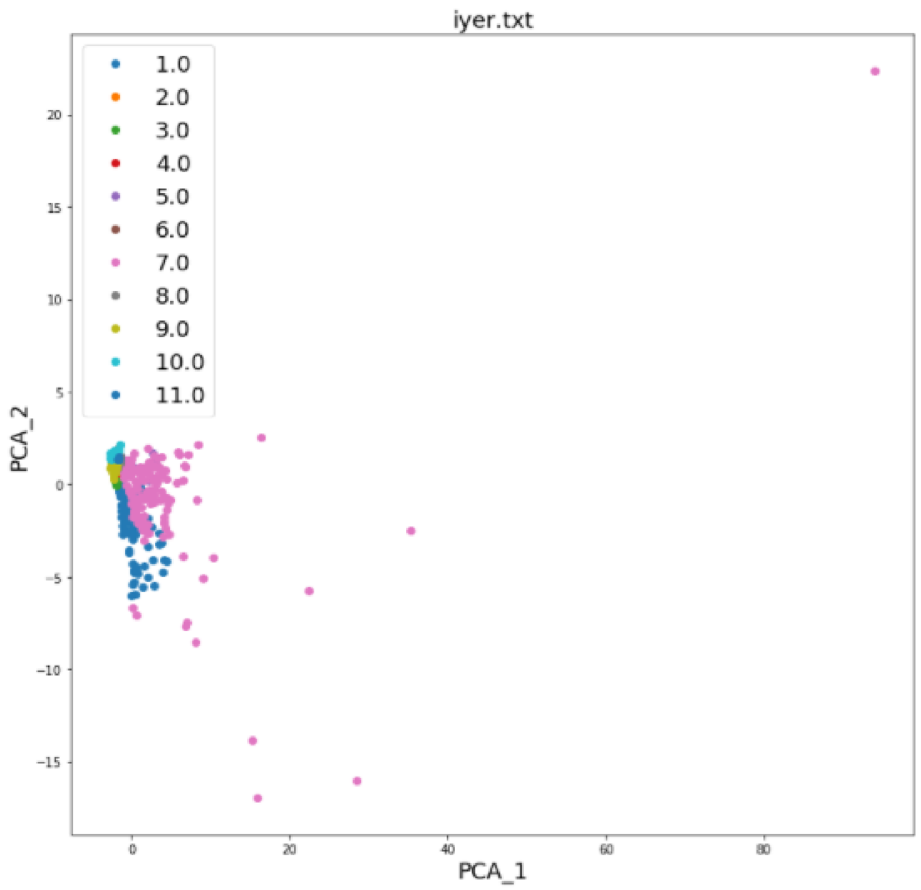


**Jaccard Index:** 0.3502

**Rand Index:** 0.7307

**K-means clustering with iyer.txt file:**

- Number of clusters = 11
- Sigma = 4
- Number of Iterations = 10
- Initial centroids = randomly selected



**Jaccard Index:** 0.1984

**Rand Index:** 0.7712

**Result:**

- The spectral clustering is computationally heavy unless it has clear spare datapoints.
- Kernel k-problem with the k-means where the input data points are mapped non linearly in the high dimensional feature space.
- The transformed weighted kernel k-means problem greatly reduces the computational expense.

**Pros:**

- It is easy to implement and gives better cluster results by correctly clustering the data points in the same cluster.
- It doesn’t make strong assumption on the statistics of the cluster.
- Fast for spare data set with numerous features.

**Cons:**

- Due to implementation of k-means, the clusters are not always same due to selection of initial centroids.
- For dense data, this might increase the time and computation complexity due to generation of eigen values and eigen vectors for every data.

## Density Based Clustering (DBSCAN)

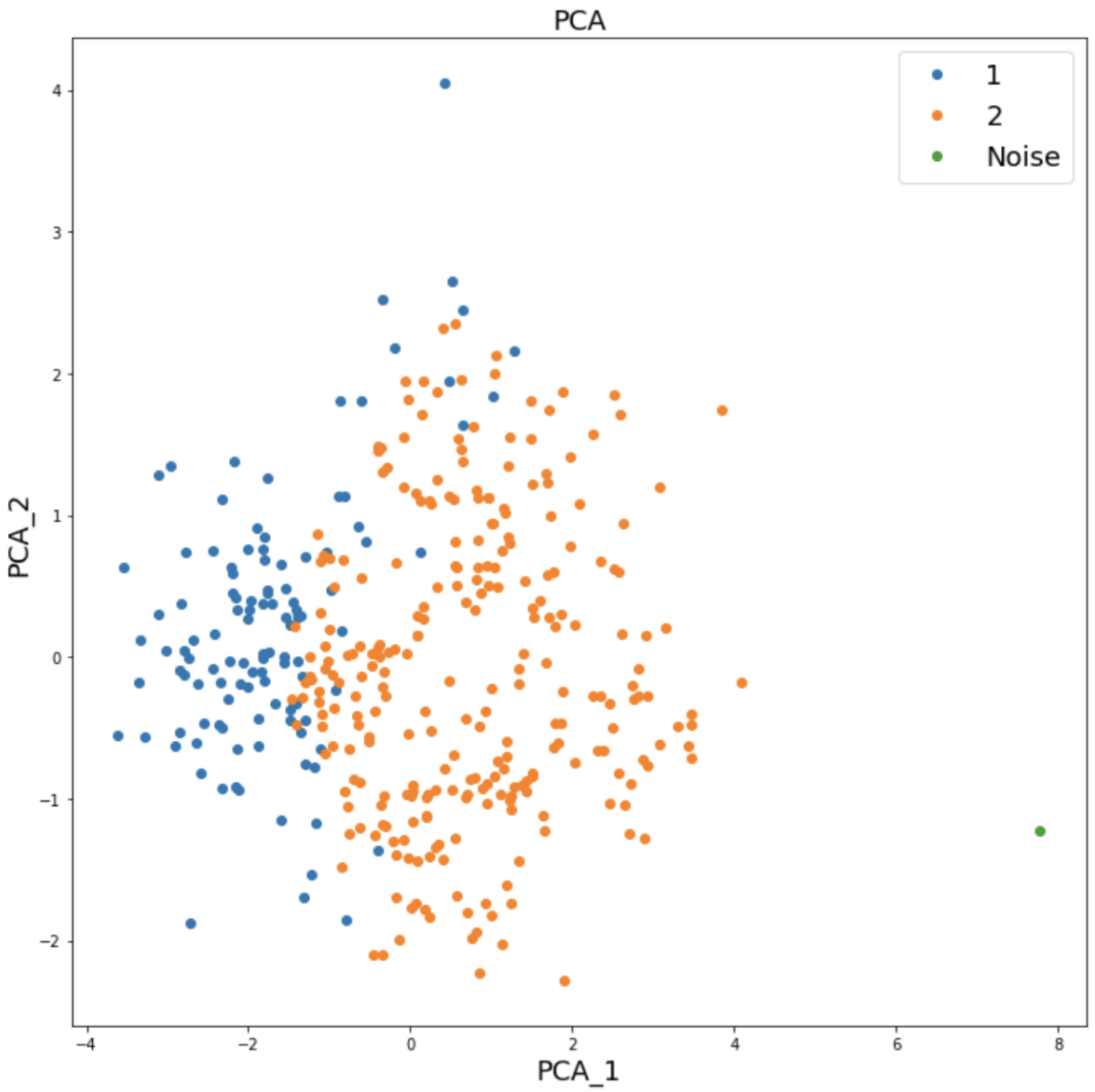
DBSCAN Algorithm utilizes the density (minimum points) of data points within a given radius (epsilon) for clustering. DBSCAN has the ability to cluster arbitrary shapes.

**Pseudo code:**

1. Initialize with first cluster. Inputs are eps and minpts.
2. For each unvisited data point p find the datapoints that the within the eps.
3. If the total number of neighbor points are higher than the minpts, assign p to the initial cluster and mark p as visited.
4. For each of these neighbor points find the next neighbor points and is its greater than the minpts assign them to initial cluster.
5. Move to next cluster and repeat the step 3 and 4 until every point in the data set are visited and assigned to a cluster.
6. If the number of neighbor points is less than the minpts then mark the corresponding data point as a Noise.

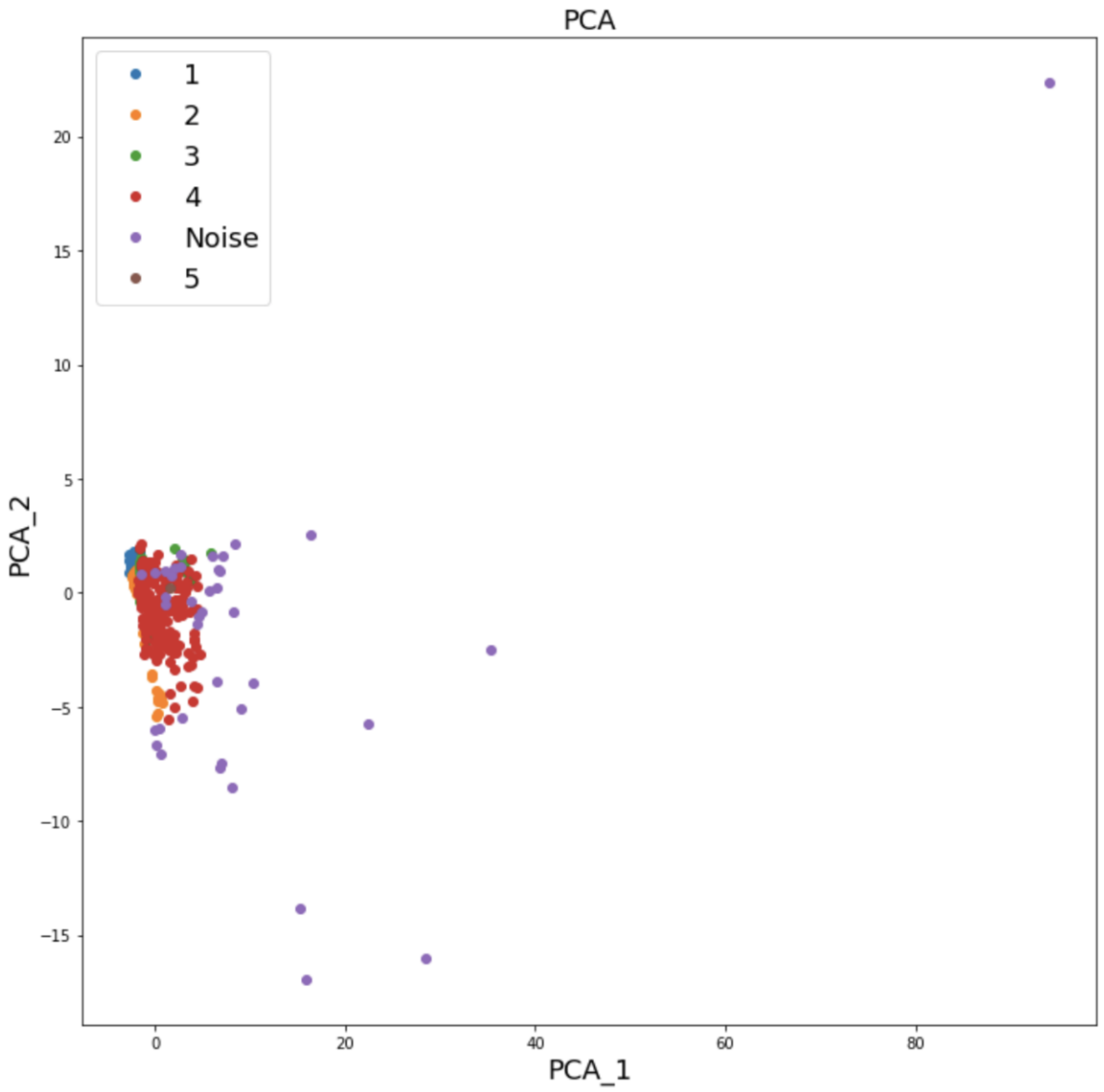
## DBSCAN on file cho.txt:

```
Enter the epsilon value : 5
Enter the minimum points value : 20
For DBSCAN on cho.txt with eps as 5.0 and min points as 20:
Jaccard :  0.23524571616645754
Rand :  0.5009664688984939


Plot for the file cho.txt
```



## DBSCAN on file iyer.txt:

```
Enter the epsilon value : 4
Enter the minimum points value : 20
For DBSCAN on iyer.txt with eps as 4.0 and min points as 20:
Jaccard :  0.22195719653651363
Rand :  0.7327387210098433


Plot for the file iyer.txt
```



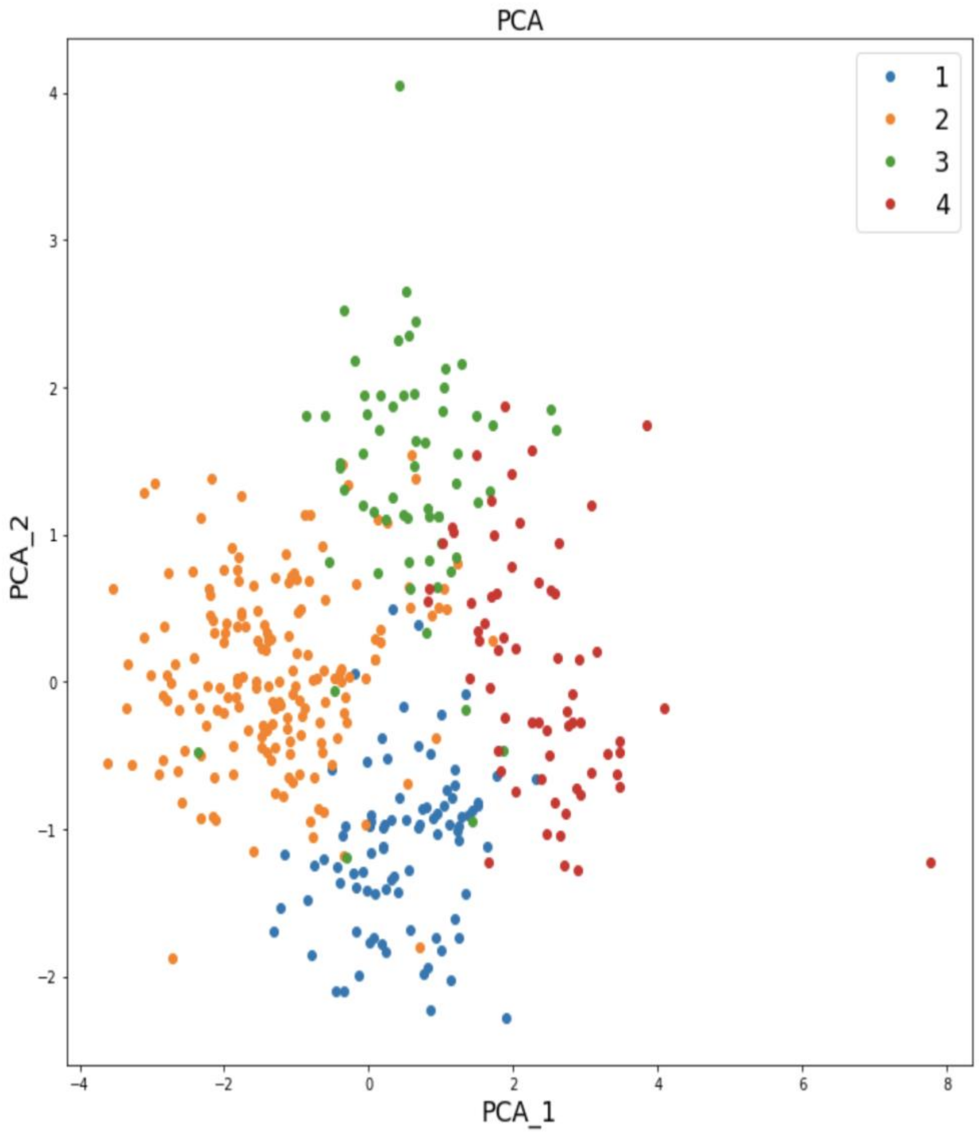
## Gaussian Mixture Model (EM Algorithm)

GMM clustering is a type of a probabilistic clustering where each cluster is mathematically represented by a distribution. In GMM clustering each data point belongs to every cluster with a probability degree.

**Pseudo code:**

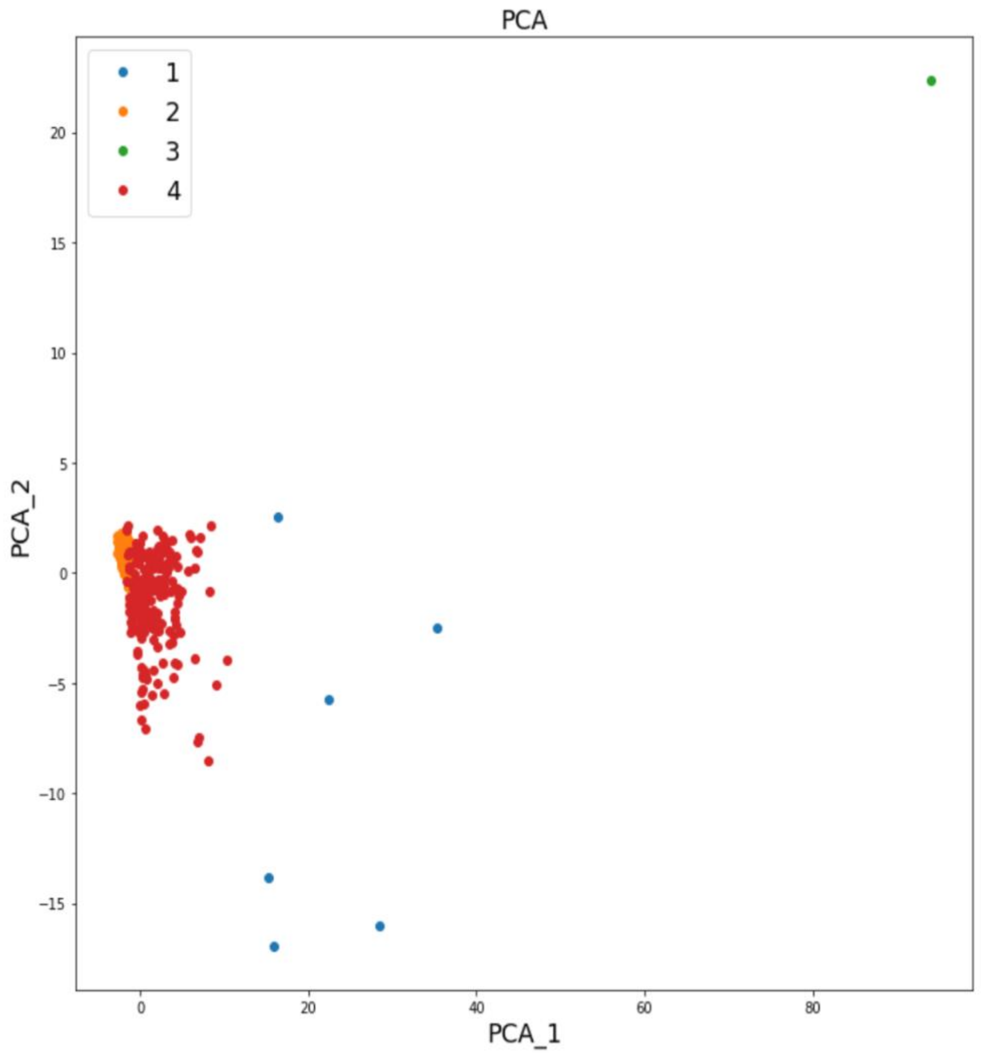
1. Initialize mu, pi, sigma, where mu is the mean, pi is the cluster probability values, and sigma is the covariance matrix.
2. Find Gamma values in E step, where we calculate the expectation.
3. Then during the M step, we maximize the expectation the find the cluster with maximum probability for each data points
4. During each iteration we keep updating mu, pi, and sigma.
5. Repeat steps 2, 3, 4 until convergence or the degree of loss during each iteration reaches the maximum threshold.
6. For the initial guess of clusters, we can use the K-Means clustering method to initially assign each data points to a cluster.

## EM Algorithm on cho.txt:



```
For GMM on cho.txt with number of clusters as 4 and number of iterations as 100 and maximum threshold as 1e-08:
Jaccard :  0.39300899142677914
Rand :  0.7662084888184918
```

## EM Algorithm on iyer.txt:



```
For GMM on iyer.txt with number of clusters as 4 and number of iterations as 100 and maximum threshold as 1e-08:
Jaccard :  0.2823348335752686
Rand :  0.6319190090127166
```