

CSE 601: DATA MINING AND BIOINFORMATICS

FALL 2019

PROJECT 2: CLUSTERING ALGORITHMS

REPORT

Keerthana Baskaran UB NO: 50288944 (UBIT: kbaskara)

Sriram Venkataramanan UB NO : 50289666 (UB IT: sv84)

Vivek Nagaraju UB NO : 50288711 (UB IT: viveknag)

TASKS TO IMPLEMENT

Implement five clustering algorithms to find clusters of genes that exhibit similar expression profiles: K-means, Hierarchical Agglomerative clustering with Min approach, density-based, mixture model, and spectral clustering. Compare these five methods and discuss their pros and cons.

For each of the above tasks, you are required to validate your clustering results using the following methods:

- Using external index (Rand Index and Jaccard Coefficient) and compare the clustering results from different clustering algorithms. The ground truth clusters are provided in the datasets.
- Visualize data sets and clustering results by Principal Component Analysis (PCA). You can use the PCA you implemented in Project 1 or use any existing implementation or package.

CLUSTERING ALGORITHMS AND THEIR IMPLEMENTATION

CLUSTER EVALUATION:

Before we look into different clustering algorithms, let us see the coefficients with which we are going to evaluate them

Rand and Jaccard Indices are more often used to compare Partitionings/clustering than usual response characteristic statistics like sensitivity/specificity etc. We are going to use the same as well for evaluating.

Before we see how to calculate both the coefficients, we need to understand 4 terms related to it.

- **True Positives:** Relevant data retrieved.
It means two points are in same clusters according to the ground truth value given to us (our input file) and we also classify them in the same cluster. (Eg p is 1 and q is also 1)
- **True Negatives:** Irrelevant data omitted.
It means two points are not in same cluster according to the ground truth value given to us (our input file) and we also classify them in different clusters. (Eg p is 0 and q is also 0)
- **False Positives:** Irrelevant data retrieved
It means two points are not in same cluster according to the ground truth value given to us (our input file) but, we classify them in same cluster (Eg p is 1 and q is 0)
- **False Negatives:** Relevant data omitted.
It means two points are in same cluster according to the ground truth value given to us (our input file) but, we classify them in different clusters. (Eg p is 0 and q is 1)

JACCARD COEFFICIENT:

The **Jaccard index**, also known as **Intersection over Union** and the **Jaccard similarity coefficient**, is a statistic used for gauging the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets.

$$J = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}.$$

Where,

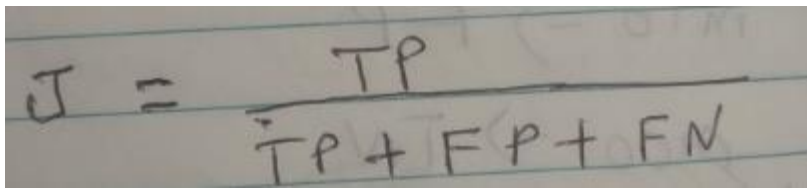
M_{11} represents the total number of attributes where A and B both have a value of 1.

M_{01} represents the total number of attributes where the attribute of A is 0 and the attribute of B is 1.

M_{10} represents the total number of attributes where the attribute of A is 1 and the attribute of B is 0.

M_{00} represents the total number of attributes where A and B both have a value of 0.

This is nothing but



A photograph of a handwritten formula on lined paper. The formula is $J = \frac{TP}{TP + FP + FN}$. The letters are written in black ink, and the fraction is clearly defined with a horizontal line.

True negatives are alone not included in the denominator.

Naturally, TN are neglected by Jaccard by definition. For very large datasets, the number of TN can be pretty huge. That term was driving all the analysis. Thus in Jaccard Index, when we neglect the contribution of TN, we will understand things better.

RAND COEFFICIENT:

It is also a metric to evaluate the quality of clustering technique and it measures the percentage of decisions which are correct.

The **Rand index** or **Rand measure** in statistics, and in particular in data clustering, is a measure of the similarity between two data clusterings. A form of the Rand index may be defined that is adjusted for the chance grouping of elements, this is the **adjusted Rand index**. From a mathematical standpoint, Rand index is related to the accuracy, but is applicable even when class labels are not used.

Similarly, one can also view the Rand index as a measure of the percentage of correct decisions made by the algorithm. It can be computed using the following formula:

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

KMEANS

K-means is a simple and popular unsupervised learning algorithm which is used to find groups in data. It clusters the data that has similarities.

The number of centroids is given beforehand. K means works in an iterative manner and it assigns every data point to its nearest cluster, while keeping the centroids as small as possible. The data points are assigned to clusters based on its similarity.

The k-means algorithm stops optimising cluster when either the centroids have been stabilized i.e. there is no change in the value of the centroids or the number of iterations specifies has got over.

ALGORITHM STEPS:

- Initially, you randomly pick k centroids in d -space. Try to make them near the data but different from one another.
- Assign each data point to closest centroid.
- Move the centroids to average location of data points assigned to it.
- Repeat the process until the assignments do not change or change very little.

IMPLEMENTATION ALGORITHM OF CODE:

- 1) We would initialize the K centroids corresponding to the K clusters.
- 2) In each iteration, we would assign the points to the clusters which have the minimum euclidean distance to the point.
- 3) We would calculate the updated centroids by calculating the mean of the points corresponding to each cluster.
- 4) If the centroids do not change, it means the clusters have been converged and we can stop out iteration
- 5) At the end of all iterations, the assigned clusters to each point denote the final clustering.

VISUALISATION:

- 1) File: iyer.txt

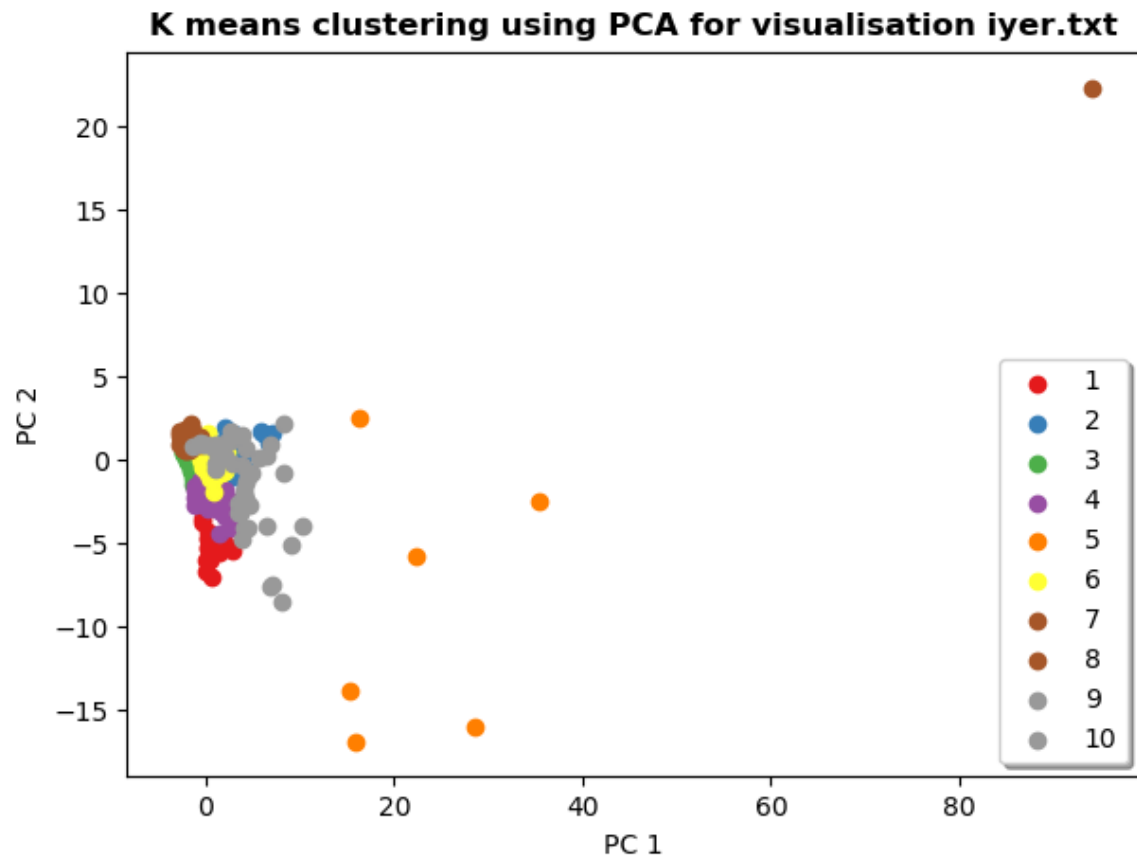
INPUT:

Enter the filename: `iyer.txt`

Enter the value of k : `10`

User want to enter centroids 1.Yes 2.No (Enter 1 or 2) `2`

Max number of iterations to be done: `30`



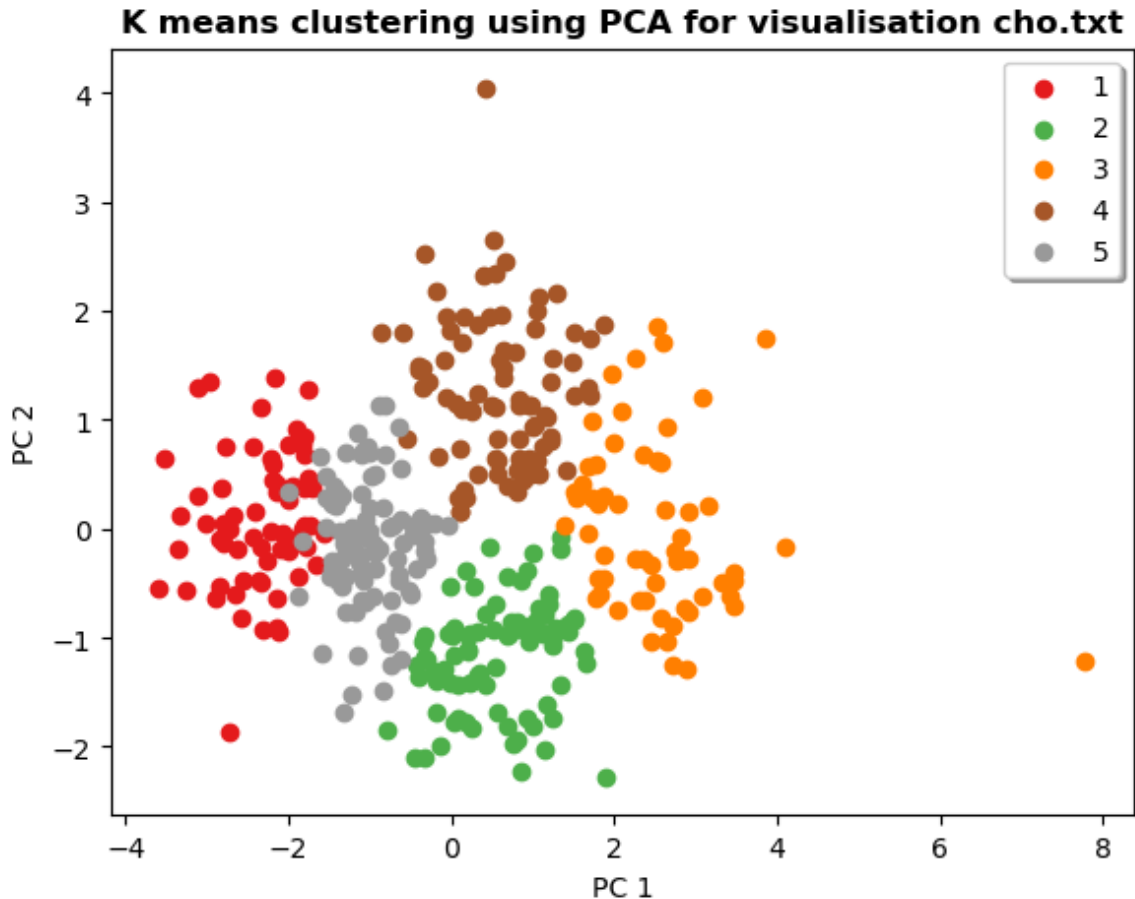
Jaccard Coefficient = 0.3150825608734427

Rand Index = 0.7155438495411334

2) File: Cho.txt

INPUT:

```
Enter the filename: cho.txt
Enter the value of k: 5
User want to enter centroids 1.Yes 2.No (Enter 1 or 2)2
Max number of iterations to be done: 30
```



Jaccard Coefficient = 0.336422804415884

Rand Index = 0.7837660071411313

RESULT EVALUATION:

- K-means is comparatively easy and efficient to implement.
- It was able to identify fairly distributed clusters for both data sets. As we know one key advantage of K-means is that it converges most of the time.
- In case of iyer.txt, we notice that the points in the centre form a cluster. It maintained a spherical shape in general and the points which are nearby in distance are put in same cluster.
- When we take the case of cho.txt, the clustering of data points looks better in terms of visualisation. It could be the reason, the data points are more moderate in terms of average distance from each other.
- The value of jaccard is high for both the datasets iyer.txt and cho.txt. -----Mention the range

EVALUATING FOR DIFFERENT VALUES OF K FOR BOTH FILES:

1) File: iyer.txt

K (Number of Centroids picked randomly)	Jaccard Coefficient	Rand Index
10	0.3150825608734427	0.7155438495411334
8	0.3081339795294705	0.6993067428887833
6	0.28022190979214956	0.6456569480973777

4	0.2515002009050608	0.5748721421382847
2	0.15606892547745807	0.16793433324977833

2) File : Cho.txt

K (Number of Centroids picked randomly)	Jaccard Coefficient	Rand Index
6	0.3079763029106208	0.7836183521705281
5	0.4071865871713817	0.8011356009557303
4	0.4391353811149033	0.801471180434374
3	0.4088574886337618	0.7574028832988805
2	0.3446946291307249	0.64411125130876

ADVANTAGES:

- It is easy to implement and efficient.
- It is computationally faster.
- Scales to large new data sets.
- It guarantees convergence.

DISADVANTAGES:

- It is difficult to predict the number of clusters. (We have to specify K in the beginning)
- It is dependent on the initial values and could result in local minima.
- It cannot handle non spherical shape clusters.
- Ordering of data sometimes has its impact on the results.

DBSCAN

DBSCAN is a well-known density-based clustering algorithm used in data mining and Machine Learning. The number of clusters need not be given as a parameter in case of DBSCAN, instead it infers the number of clusters based on the data and it discovers clusters of arbitrary shape.

The 2 parameters that are needed to be given in case of DBSCAN are:

- 1) **Eps ϵ** : The radius of our neighbours around a data point p.
- 2) **minPts**: The minimum number of data points needed in a neighbour to define a cluster.

With the help of these two parameters, DBSCAN categorises the data points into 3 categories.

- **Core point**: A core point is one which has more neighbours than the minimum number of points (minPts) specified initially in the algorithm.
- **Border point**: A border point is one it has number of neighbours less than the minimum number of points but it is reachable from a core point i.e. it is within the distance of eps.
- **Outlier**: It is neither a core point nor a border point.

IMPLEMENTATION ALGORITHM OF CODE:

These are the steps followed while implementing DBSCAN algorithm

Step 1) The file is obtained from the command line from the user.

Step 2) Once the file is obtained, it is converted into a np array using np.asarray() function.

Step 3) We pre-processed the data i.e. the points are converted into a matrix form using `np.matrix()` and the ground truth from the input file is stored in a variable separately.

Step 4) We obtain two input values from the user `epsilon` and `min_pts` whose functionality are mentioned above.

Step 5) Initially distance matrix for the points are computed by calling an in-built function `distane_matrix(points,points)`. It computes a distance of a point with other points and stores it in the form of a matrix.

Step 6) **`dbscan(points,epsilon,min_points,visited_points,cluster_final,dist_matrix)`**

- The next step is to call the `dbscan` function. All the necessary arguments are passed to the function, that are mentioned above.
- For every point that is not visited yet, i.e by looping to the length of points, get the point's neighbours within the `epsilon` distance we obtained from the user.

Step 7) **`regionQuery(query_point_index,points,epsilon,dist_matrix):`**

- This function is used to return the neighbour points from the distance matrix calculated above.
- Once the neighbour points are obtained, if the number of points is greater than the `min_pts`, we know it's a core point, add it to a cluster and call the `expandCluster()` function
- If it is not a core point, i.e. the number of neighbour points is less than the `min_pts`, it is classified as a noise.

Step 8) **`expandCluster(corepoint_index,points, neighbour_pts,cluster,epsilon,min_points,cluster_final,visited_points,dist_matrix):`**

- For every neighbour point we obtained above, if the point is not in visited list, get it's neighbours by calling the `regionQuery()` function.
- If the number of points is greater than the `min_pts`, the neighbour points are merged with the neighbour points, i.e the points are added in the same cluster.
- Once all the points have been finised visiting, then we obtain the cluster list of all points, which are maintained in `cluster_final`.
- Once the `cluster_final` is obtained, we plotted those clusters by calling the `PCA()` method provided by `sklearn`.
- `pca_plot_matrix = PCA(n_components=2).fit_transform(points)`.

Step 9) The plotting is done by using `matplotlib.pyplot`, by setting all the legends and attributes and the two coefficients(Jaccard and Rand index) are calculated.

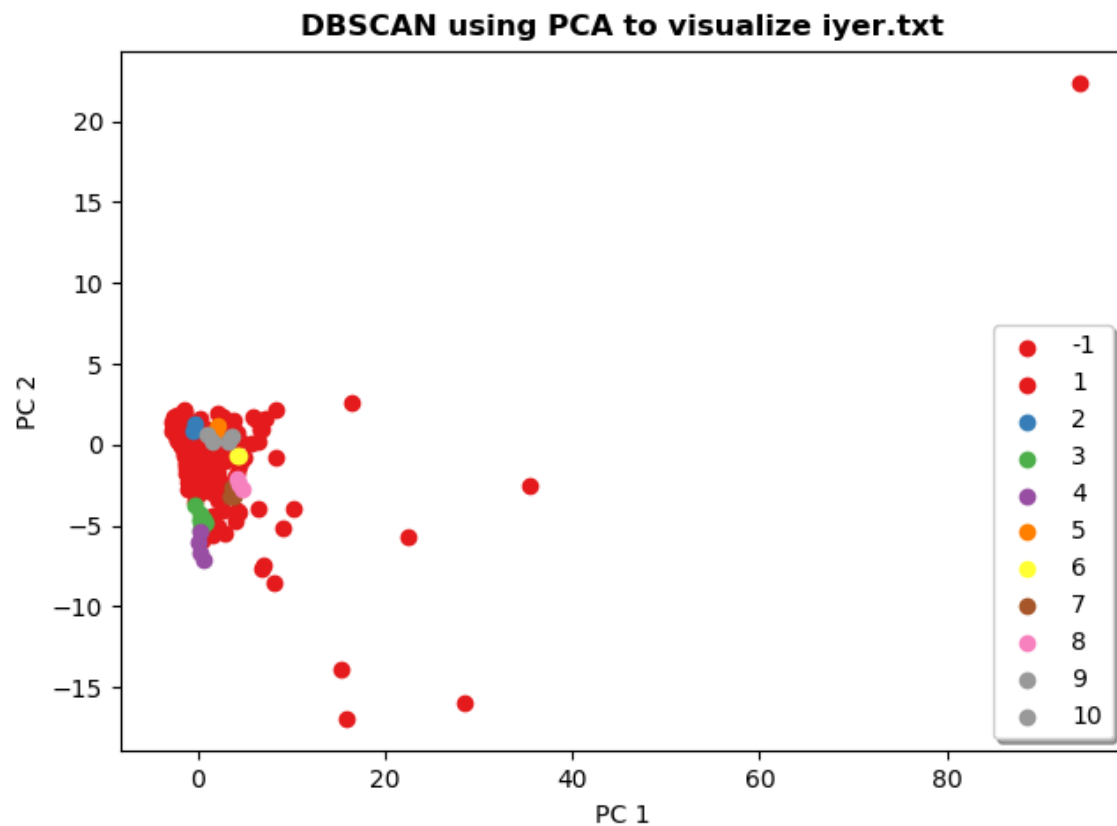
VISUALISATION:

Experiment: Different values of `eps` and `min_pts` resuted in different number of clusters. We obtained exactly 10 clusters for the following values.

- `Eps=1.42`
- `Min_pts = 2`

1) File : `iyer.txt`


```
Enter the filename: iyer.txt
Enter the eps value: 1.42
Enter the minimum points: 2
```



Jaccard Coefficient = 0.20501094324625382

Rand Index = 0.46865003797387844

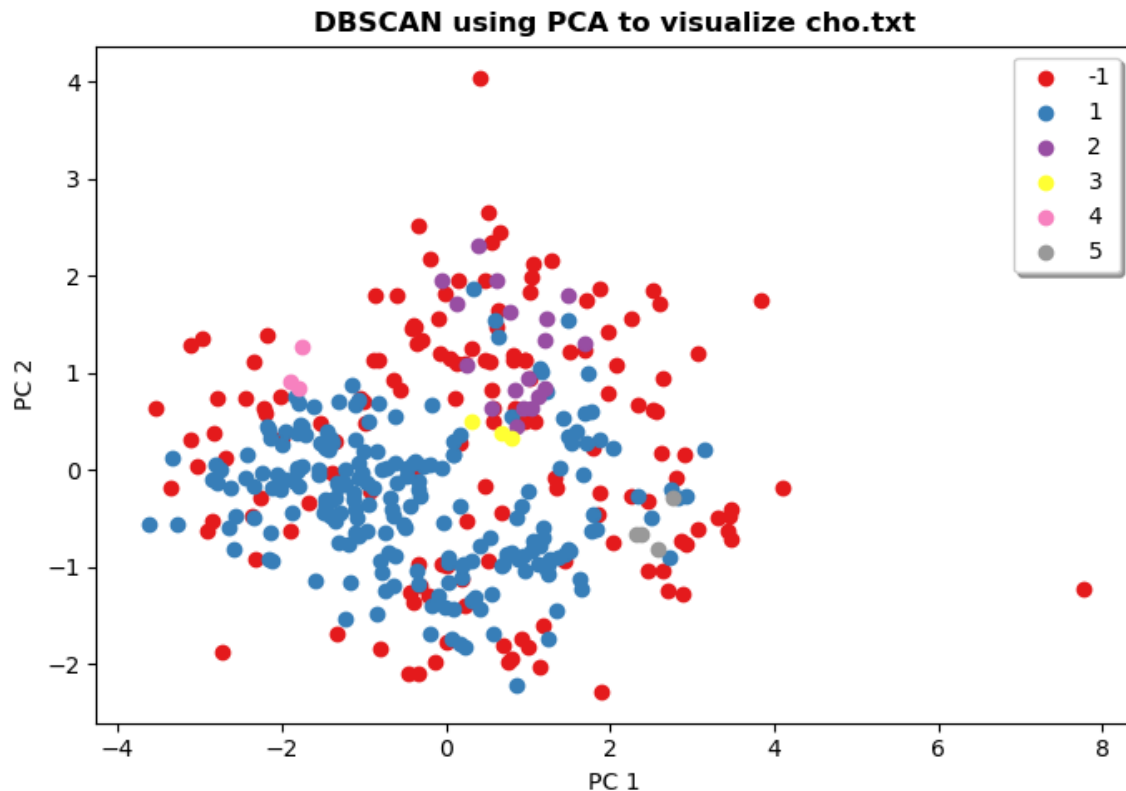
Cho.txt

Experiment: Different values of eps and min_pts resulted in different number of clusters.

We obtained exactly 5 clusters for the following values.

- Eps=1.13
- Min_pts = 3

```
Enter the filename: cho.txt
Enter the eps value: 1.13
Enter the minimum points: 3
```



Jaccard Coefficient = 0.21188257110517064

Rand Index = 0.560726462455368

RESULT EVALUATION:

- As we know that the DBSCAN classifies the points based on the number of neighbours that falls within a particular radius. Both of these are given as inputs to the algorithm min_pts and eps value.
- The time complexity of the algorithm is $O(n^2)$ where n is the number of data points when the dimensionality of data is low.
- We ran the code for different values of eps and min_pts and identified that the algorithm is sensitive to the parameters.
- In case of Cho.txt, visualisation of clusters is very good i.e. it was able to identify well-spaced clusters and we can differentiate it easily.
- In case of iyer.txt, the points look closer to each other along the principal axis, however in reality they could be farther apart from each other.
- From the graph we can see that it handles outliers better unlike hierarchical clustering.

EVALUATING FOR DIFFERENT VALUES OF EPS AND MINPTS FOR BOTH FILES:

File: Iyer.txt

EPS	MINPTS	CLUSTERS FORMED	Jaccard Coefficient	Rand Index
1.42	2	10	0.20501094324625382	0.46865003797387844

1.5	3	4	0.19474242541851017	0.4178136773305299
2	2	5	0.1760975808606719	0.31945572021295304
3	2	6	0.16459589992330492	0.24608943877226522
4	20	2	0.16244762605306684	0.22894694506695001

File: Cho.txt

EPS	MIN_PTS	CLUSTERS FORMED	Jaccard Coefficient	Rand Index
1.13	3	5	0.21188257110517064	0.560726462455368
1.1	5	4	0.2110538575562928	0.5499073800638943
1	3	6	0.20433630786391524	0.5404037692287041
3	1	2	0.2303719983280974	0.23379151118150823
1.1	4	4	0.20370195262635338	0.547840210475449

ADVANTAGES:

- It detects arbitrary shaped clusters.
- It does not require to specify number of clusters in the beginning.
- It is robust towards outliers i.e it has high resistance towards noise.

DISADVANTAGES

- It is sensitive to parameters min_pts and eps.
- It fails to identify the cluster if the dataset is too sparse.
- It is tricky when the density of the dataset varies.

Hierarchical Agglomerative Clustering

It is a clustering method which assigns n data points to n clusters initially and joins the most similar cluster by computing the similarity. Similarity can be found by measuring the distance each of the cluster. The process is continued until we obtain a single cluster.

- It starts with one cluster, individual item in its own cluster and iteratively merge clusters until all items belong to one cluster.
- Bottom up approach is followed to merge the clusters together.
- Dendrograms are pictorially used to represent the HAC

The various distance metric used to form the cluster are:

Type	Explanation
Single Linkage	This is the distance between the closest members of its two clusters.
Complete Linkage	This is the distance between the members that are farthest apart.
Average Linkage	This method involves looking at the distances between all pairs and averages all of these distances. This is also called Unweighted Pair Group Mean Averaging.

From this, we used Single Linkage for the implementation.

Generic Algorithm

- 1) Compute the proximity matrix.
- 2) Let each data point be a cluster.
- 3) Repeat: Merge two closest clusters and update the proximity matrix.
- 3) Until only a single cluster remains.

For the given dataset:

- 1) For the given objects and attributes in the input data, calculate the distance matrix using Euclidean distance.
- 2) On each iteration, Identify the least value i.e. minimum value from the distance matrix.
- 3) Once the minimum value is found, merge the object id's for which the value is minimum and update the distance matrix based on the new cluster.
- 4) Repeat the above steps, until it comes to only one cluster.

IMPLEMENTATION ALGORITHM OF CODE:

Step 1) The file is obtained from the command line from the user.

Step 2) Once the file is obtained, it is converted into a np array using np.asarray() function.

Step 3) We pre-processed the data i.e. the points are converted into a matrix form using np.matrix() and the ground truth from the input file is stored in a variable separately.

Step 4) The number of clusters is obtained from the user as input.

Step 5) Initially distance matrix for the points are computed by calling an in-built function `distane_matrix(points,points)`. It computes a distance of a point with other points and store it in the form of a matrix.

Step 6) The `update_cluster()` method is called, which will return the final cluster list of all the points.

`update_cluster()`

Step 7) By lopping to the number of clusters, we perform single linkage (i.e MIN) Hierarchial clustering.

- Initially we find the minimum value from the distance matrix and find the cluster those points belong to.
- `find_cluster(cluster_list,point)`: this method is used to return the index of the cluster that point belongs to.
- Once the index of the cluster of the points whose distance was minimum from the distance matrix was found, the net step is to merge those clusters.

- Once the clusters are merged, the next step is to update the distance matrix.
- The matrix is updated based on MIN Technique i.e. Single Linkage.
- Once the loop is finished to the number of clusters, it will return a list.
- `final_point_cluster_name` : This list will return the clusters each point belongs to.

Step 8) Once the cluster_final is obtained, we plotted those clusters by calling the PCA() method provided by sklearn.

```
pca_plot_matrix = PCA(n_components=2).fit_transform(points)
```

Step 9) The plotting is done by using **matplotlib.pyplot**, by setting all the legends and attributes.

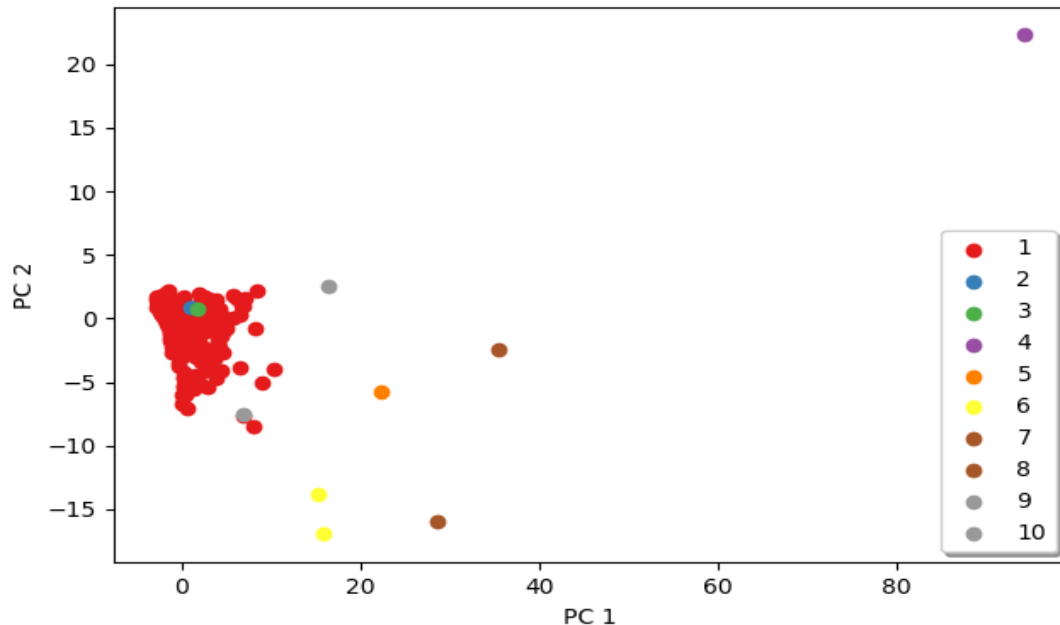
VISUALISATION:

1) File: Iyer.txt

Number of Clusters=10

```
Enter the filename: iyer.txt
Enter the number of Clusters: 10
```

Hierarchical Agglomerative clustering using PCA for visualisation iyer.txt



Jaccard Coefficient = 0.15824309696642858

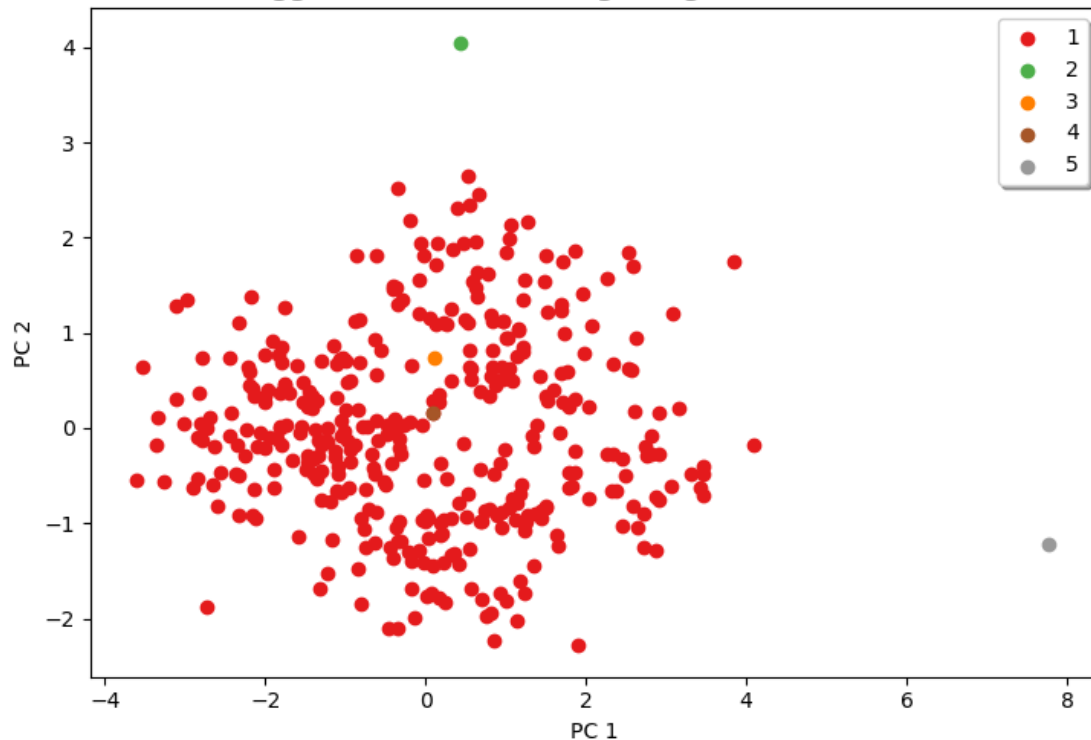
Rand Index = 0.1882868355974245

2) File: Cho.txt

Number of Clusters=5

```
Enter the filename: cho.txt
Enter the number of Clusters: 5
```

Hierarchical Agglomerative clustering using PCA for visualisation cho.txt



Jaccard Coefficient = 0.2297361848480299

Rand Index = 0.24242261537222476

RESULT EVALUATION:

- We used Single Linkage Hierarchical agglomerative clustering, which is used to group clusters in bottom up fashion, at each step combining two clusters that contain the closest pair of elements not yet belonging to the same cluster as each other. Thus points closest to each other first form a cluster.
- In visualisation, we can see the clusters are not evenly distributed. It looks like a one dominant cluster. The points which are farther apart falls in separate cluster.
- We don't specify the clusters or eps or min value at the start. The clusters are determined by pair of points, just by a link in the proximity graph.
- By looking at the graph, we can identify it is more sensitive to outliers and noise. At times Jaccard coefficient is a lesser value when compared to other algorithms for the same reason.

EVALUATING FOR DIFFERENT NUMBER OF CLUSTERS FOR BOTH FILES:

File : lyer.txt

CLUSTERS	Jaccard Coefficient	Rand Index
2	0.15548413017276014	0.15853626598924758
4	0.15632743078279093	0.16560726404752907
6	0.1564646060675313	0.17143616085959393
8	0.15710452461338867	0.1808566757330081
10	0.15824309696642858	0.1882868355974245

File: Cho.txt

CLUSTERS	Jaccard Coefficient	Rand Index
1	0.23038858776258664	0.23381835753979974
2	0.23038858776258664	0.23381835753979974
3	0.23068692778590388	0.23752315498402643
4	0.22964654082922256	0.23906682058578754
5	0.2297361848480299	0.24242261537222476

ADVANTAGES:

- It outputs dendrogram, which can be visualized for the clusters present in data.
- It can handle non elliptical shape clusters.

DISADVANTAGES:

- It is sensitive to noise and therefore outliers.
- Here again, ordering of data has impact on its results.
- Time complexity of the algorithm is quite high and therefore not suitable for large datasets.
- Intermediate decisions cannot be undone.

MIXTURE MODEL CLUSTER:

- It is a probabilistic grounded way of doing soft clustering.
- Each cluster is a generative model (Gaussian in this case)
- The parameters of the model are mean and covariance which are unknown.
- We estimate Gaussian distribution parameters mentioned above for each cluster and weight of a cluster.
- Once the parameters are learnt, we can calculate probability of it belonging to each of the cluster.

Mathematically we can define Gaussian mixture model as mixture of K Gaussian distribution that means it's a weighted average of K Gaussian distribution.

$$p(x) = \sum_{k=1}^K \pi_k \mathcal{N}(x | \mu_k, \Sigma_k)$$

where $\mathcal{N}(x | \mu_k, \Sigma_k)$ represents cluster in data with mean μ_k and co variance Σ_k and weight π_k .

MATH BEHIND THE MODEL:

- 1) Decide on how many sources or clusters you want to fit the data in.
- 2) Initialize the mean, covariance, fraction per class for each cluster.

Expectation maximization for mixture models consists of two steps.

- The first step, known as the **expectation** step or **E** step, consists of calculating the expectation of the component assignments $C_{k|i}$ for each data point $x_i \in X$ given the model parameters ϕ , μ , and σ .
- The second step is known as the **maximization** step or **M** step, which consists of maximizing the expectations calculated in the E step with respect to the model parameters. This step consists of updating the values ϕ , μ , and σ .
- The entire iterative process repeats until the algorithm converges, giving a maximum likelihood estimate. By alternating between which values are assumed fixed, or known, maximum likelihood estimates of the non-fixed values can be calculated in an efficient manner.

IMPLEMENTATION ALGORITHM:

- 1) Initialize the mean, covariance and prior probability for the K Gaussian distribution functions (this will be our k clusters).
- 2) We should add a very small value to the diagonal of the covariance matrix to avoid the singular matrix error.
- 3) Now in each iteration we will perform two operations
 - a) E Step
In this step, we would calculate posterior probability for each point belong to each of the K clusters
 - b) M Step
 - In this step, we would update the parameters (μ , σ and ϕ) based on the posterior probability.
 - If the parameter updation is less than the convergence threshold, we would stop the iteration.
 - We can also use the log likelihood function for checking the convergence
- 4) At the end of all iterations, for each point we would choose the cluster which gives the maximum posterior probability.
- 5) Once the cluster_final is obtained, we plotted those clusters by calling the PCA() method provided by sklearn.
`pca_plot_matrix = PCA(n_components=2).fit_transform(points)`
- 6) The plotting is done by using **matplotlib.pyplot**, by setting all the legends and attributes.

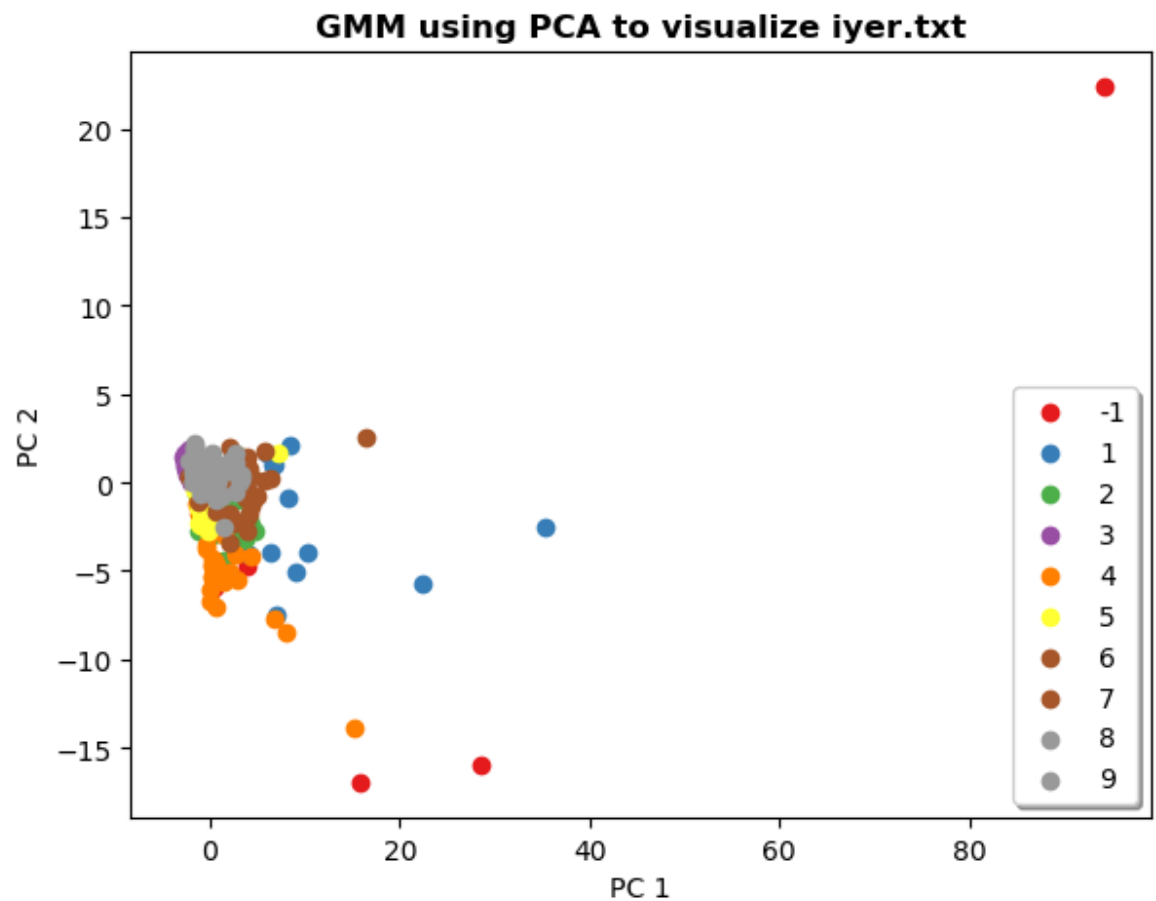
VISUALISATION:

1) File: Iyer.txt

Input:

Enter the name of the file: `iyer.txt`

```
Enter the the number of clusters: 10
```



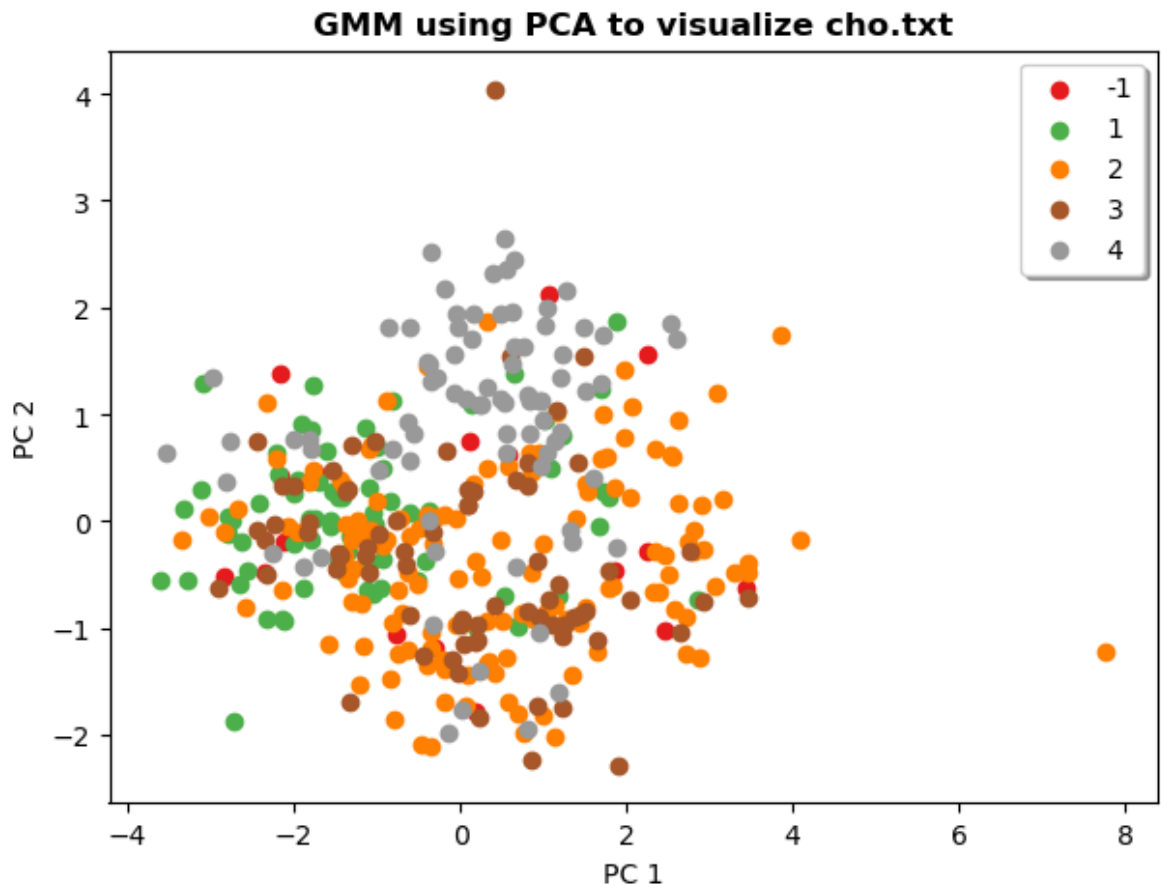
Jaccard Coefficient = 0.3642947358663283

Rand Index = 0.7947465103315138

2) Cho.txt

Enter the name of the file: *cho.txt*

Enter the the number of clusters: 5



Jaccard Coefficient = 0.17338353249304306

Rand Index = 0.6610781497489866

RESULT EVALUATION:

- The result of this is that each cluster is associated not with a hard-edged sphere, but with a smooth Gaussian model. Just as in the k-means expectation–maximization approach, this algorithm can sometimes miss the globally optimal solution, and thus in practice multiple random initializations are used.
- Unlike KMeans , In GMM we do not have a binary assignment of each point to a particular cluster . Instead we have the Gaussian distribution determine the likelihood of a point belonging to a particular cluster .

EVALUATING FOR DIFFERENT NUMBER OF CLUSTERS

1)FILE : Iyer.txt

ITERATIONS = 60

CLUSTERS	JACCARD VALUE	RAND INDEX
2	0.15523254286154042	0.15810227880683456

4	0.30528884858859523	0.693298265173651
6	0.3324705286554315	0.7337039683638309
8	0.3276340602485373	0.7398845444444029
10	0.34114963976571816	0.762219919263419

2) Cho.txt
ITERATIONS: 60

CLUSTERS	JACCARD VALUE	RAND INDEX
1	0.2300732905581358	0.2300732905581358
2	0.23684084026342356	0.3606808236462724
3	0.2501757345705047	0.5704448441568901
4	0.21064375767042284	0.5251550377191334
5	0.2058293945532298	0.6210904990738007

ADVANTAGE:

- 1) It can handle more complex and ellipsoid shape cluster compared to other models.
- 2) It is based on likelihood of a point falling on a cluster and this approach is more flexible.

DISADVANTAGE:

- 1) It has strong assumptions about the data

SPECTRAL CLUSTER:

It is a popular unsupervised machine learning algorithm. It can be solved efficiently by standard linear algebra methods. In this type of clustering, the affinity determines what points fall under which cluster.

In this case, the data points are treated as nodes of a graph. It is treated as a graph partitioning problem. The nodes are then mapped to a lower-dimensional space that can be easily segregated to form clusters. It is useful in tackling problems where the data forms complicated shapes.

GENERIC ALGORITHM:

Step 1) Construct a similarity graph.

Step 2) Determine the adjacency matrix W , Degree matrix D and the Laplacian matrix L .

Step 3) Compute the eigen vectors of Matrix L .

Step 4) Using the second smallest eigenvector as input, we should train a K means model and use it to classify the data.

IMPLEMENTATION ALGORITHM:

- 1) We first construct the similarity matrix from the given data set . We define similarity to be a metric that determines how close two points are in our space. We use Gaussian Kernel to determine our similarity
- 2) We then construct a Degree matrix from our similarity matrix. Degree matrix is defined as a diagonal matrix consisting of the sum of weights into each vertex.
- 3) Next we calculate the Laplacian matrix . Let us denote the degree matrix as D and the similarity matrix as A . Then the Laplacian Matrix $L = D - A$.
- 4) Now calculate the eigenvalues and their corresponding eigenvectors of the Laplacian Matrix.
- 5) Let the number of cluster be $K \Rightarrow$ take the smallest K eigenvalues and their corresponding eigenvectors. Reduce the $N * M$ space to $N * K$ space (N = number of points , M = number of features).
- 6) Apply KMeans Algorithm to this reduced space and Label the points to their corresponding clusters.

VISUALISATION:

- 1) Iyer.txt

Input:

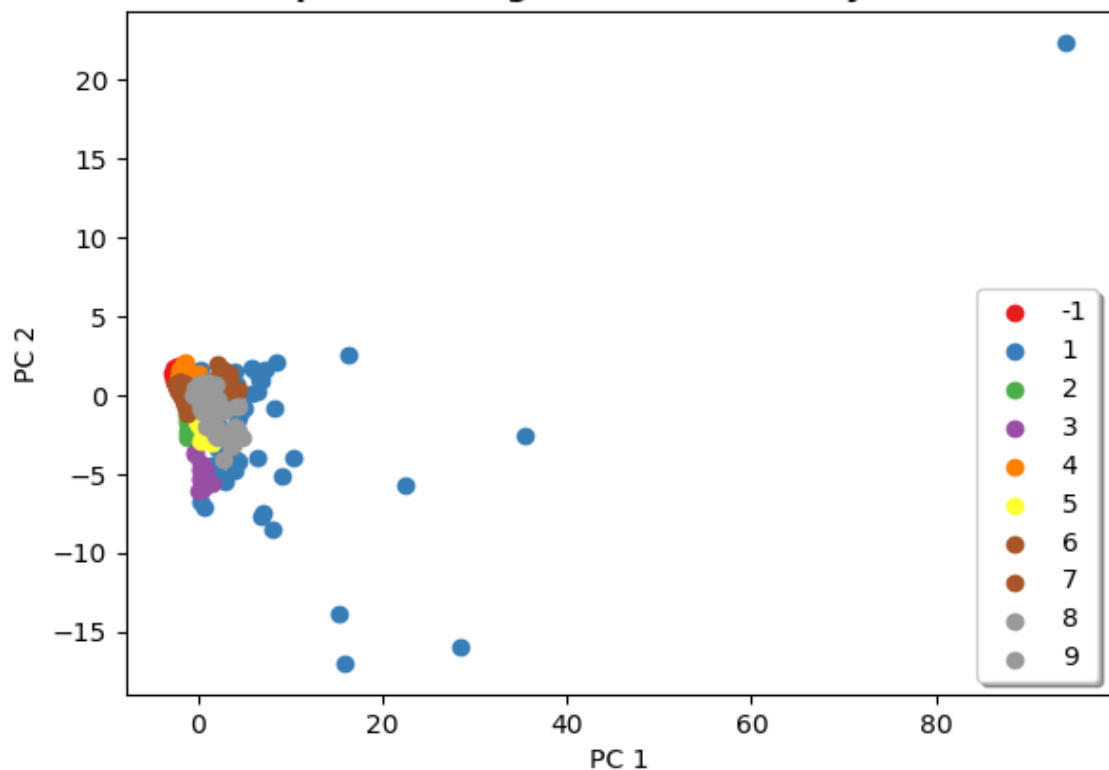
Enter the filename: `iyer.txt`

Enter the number of Clusters: `10`

Enter the sigma value `1.4`

do you want to enter the centroid points ? 1: Yes 2: No `2`

Spectral using PCA to visualize iyer.txt



Jaccard Coefficient = 0.27387633970445774
Rand Index = 0.8233298040697522

2) Cho.txt

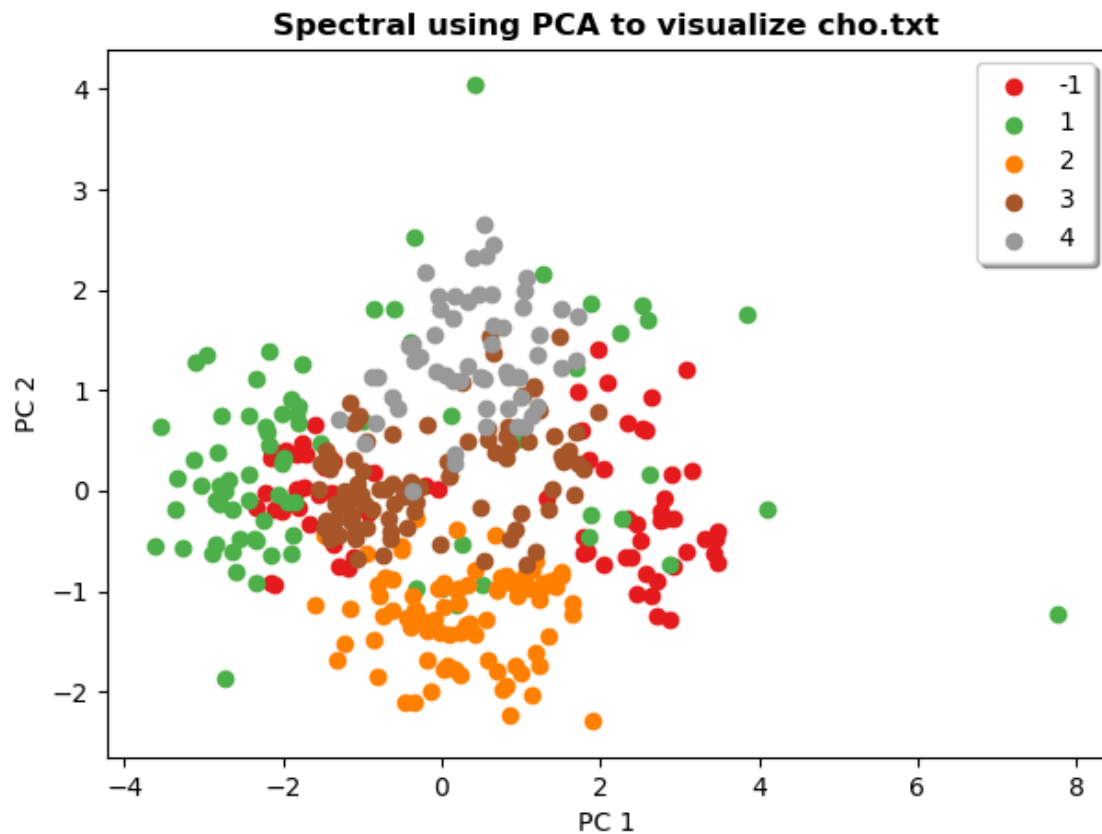
Input:

Enter the filename: *cho.txt*

Enter the number of Clusters: 5

Enter the sigma value 1.4

do you want to enter the centroid points ? 1: Yes 2: No 2



Jaccard Coefficient = 0.20684895736188308
Rand Index = 0.7145963650030873

USING EIGGEN GAP

File: Cho.txt

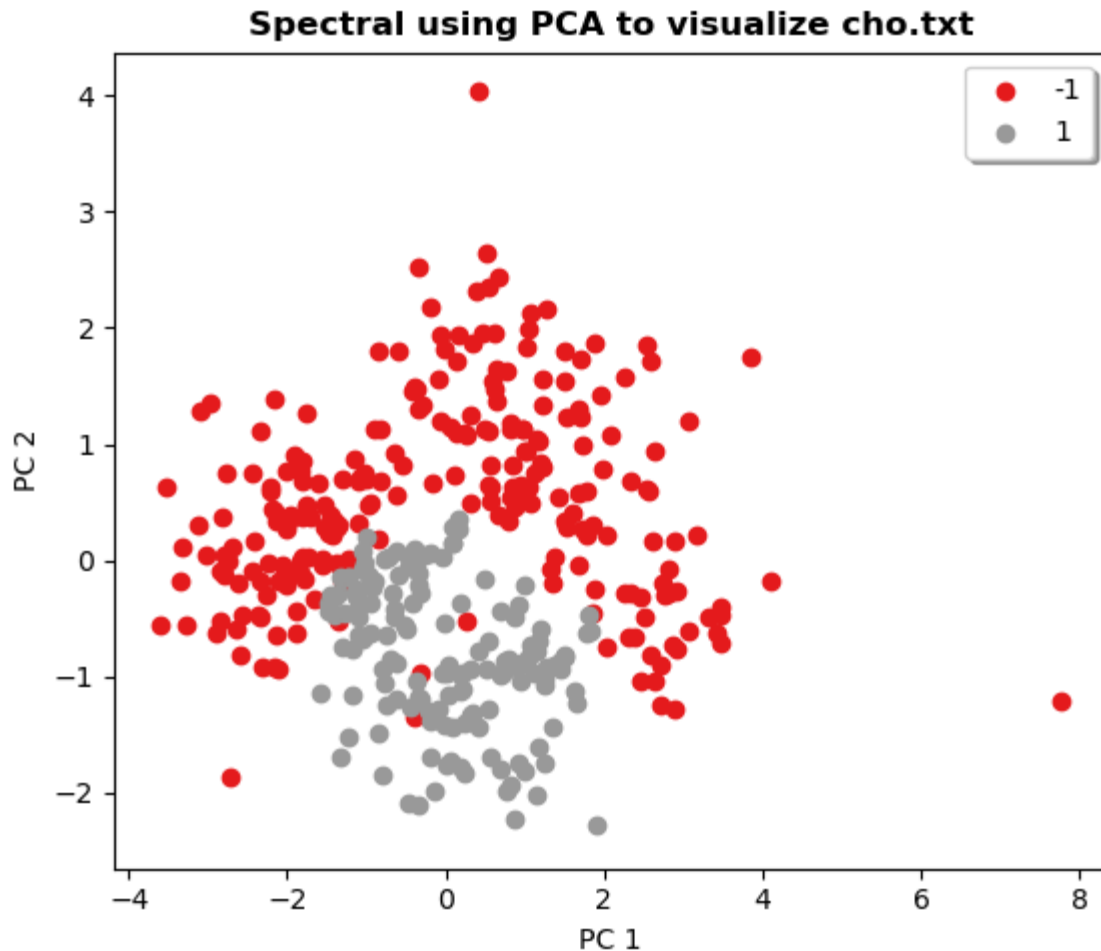
Enter the filename: *cho.txt*

Enter the number of Clusters: 5

Enter the sigma value 1.4

do you want to enter the centroid points ? 1: Yes 2: No 3. Eigen Gap 3

number of clusters = 2



Jaccard Coefficient = 0.2566161099497398

Rand Index = 0.5513033906950522

Using Eigengap we get two clusters.

RESULT EVALUATION

- We are calculating the similarity of points using Gaussian Kernel. We should choose the value of sigma that gives good clustering.
- Sigma exponentially weighs the proximity of two data points and decays exponential with the distance between two points.
- We can see that as we increase the sigma , we get better clustering result and better rand and jaccard value.
- Result of spectral clustering is influenced by how we define our similarity matrix . It is important to choose a good affinity function like Gaussian Kernel to accurately determine the correct clusters.
- Choosing proper similarity function and choosing proper centroids when applying KMeans to the reduced feature space will ensure that we get accurate clustering result.

EVALUATING FOR DIFFERENT NUMBER OF CLUSTERS FOR BOTH FILES:

File : lyer.txt

SIGMA	JACCARD VALUE	RAND INDEX
0.1	0.14526725540789914	0.20807814762298485
0.4	0.1290307279264579	0.6565215927329594
0.8	0.13423079184057374	0.7245004470816233
1.2	0.14054743305643932	0.7402661538634212
1.4	0.27387633970445774	0.8233298040697522
1.6	0.27343617562298694	0.823614140499609

File: Cho.txt

SIGMA	JACCARD VALUE	RAND INDEX
0.1	0.22297689058229106	0.24446293860237858
0.4	0.2047164771426391	0.30061209696904617
0.8	0.21094279900520008	0.6720717334693549
1.2	0.23113136486893643	0.7259657977395366
1.4	0.20684895736188308	0.7145963650030873
1.6	0.20851969412783147	0.7054551800048323

ADVANTAGES:

- 1) It is easy to implement and gives good clustering results.
- 2) It does not make strong assumptions on the statistics of clusters and helps creating more accurate clusters.
- 3) It is reasonably fast on sparse data set of several thousand elements.

DISADVANTAGES:

- 1) It is computationally expensive for large datasets. It is because eigen values and eigen vectors needs to be computed and then we have to do clustering on these vectors.
- 2) Since we use K-means in the final step, this implies clusters are not the same

COMPARISON OF ALGORITHMS

ALGORITHM	FILE	CLUSTERS	JACCARD	RAND INDEX
K MEANS	Iyer.txt	10	0.25424942362493375	0.8051622027094268
	Cho.txt	5	0.3666228122632859	0.7867459529114875
	MAIN ADVANTAGE			
	It is dependent on the initial values and could result in local minima			
	MAIN DISADVANTAGE			
It cannot handle non spherical shape clusters.				
DBSCAN	Iyer.txt	10	0.20501094324625382	0.46865003797387844
	Cho.txt	5	0.21188257110517064	0.560726462455368
	MAIN ADVANTAGE			
	Detects arbitrary shaped clusters and no need to specify the number of clusters in the beginning.			
	MAIN DISADVANTAGE			
It is sensitive to the parameters.				
HIERARCHIAL	Iyer.txt	10	0.15824309696642858	0.1882868355974245
	Cho.txt	5	0.2297361848480299	0.24242261537222476
	MAIN ADVANTAGE			
	It outputs dendogram, which gives better visualisation of data.			
	MAIN DISADVANTAGE			
It is sensitive to outliers.				
SPECTRAL	Iyer.txt	10	0.27387633970445774	0.8233298040697522
	Cho.txt	5	0.20684895736188308	0.7145963650030873
	MAIN ADVANTAGE			
	Handles arbitrary shaped clusters.			
	MAIN DISADVANTAGE			
Inefficient for extremely large datasets.				
GMM	Iyer.txt	10	0.3642947358663283	0.7947465103315138
	Cho.txt	5	0.17338353249304306	0.6610781497489866
	MAIN ADVANTAGE			
	Gives probabilistic cluster assignments.			
	MAIN DISADVANTAGE			
Have Strong assumptions about the data				

CONCLUSION:

- All 5 clustering algorithms have been implemented.
- Plots have been visualized and the coefficients (jaccard and rand index) are calculated to differentiate the algorithms.
- Every algorithm has been analysed and we identified the pros and cons of all algorithms.

REFERENCES:

- 1) <https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1>
- 2) <https://blog.dominodatalab.com/topology-and-density-based-clustering/>
- 3) <https://towardsdatascience.com/understanding-the-concept-of-hierarchical-clustering-technique-c6e8243758ec>
- 4) <https://medium.com/clustering-with-gaussian-mixture-model/clustering-with-gaussian-mixture-model-c695b6cd60da>
- 5) <https://brilliant.org/wiki/gaussian-mixture-model/>
- 6) https://en.wikipedia.org/wiki/Jaccard_index
- 7) https://en.wikipedia.org/wiki/Rand_index