CSE 601

Data Mining and Bioinformatics

Fall 2019

Project 2

Clustering Algorithms

Report

Amit Anilkumar Menon (amitanil@buffalo.edu)

Linus Castelino (linuscas@buffalo.edu)

Deepak Ranjan (dranjan@buffalo.edu)

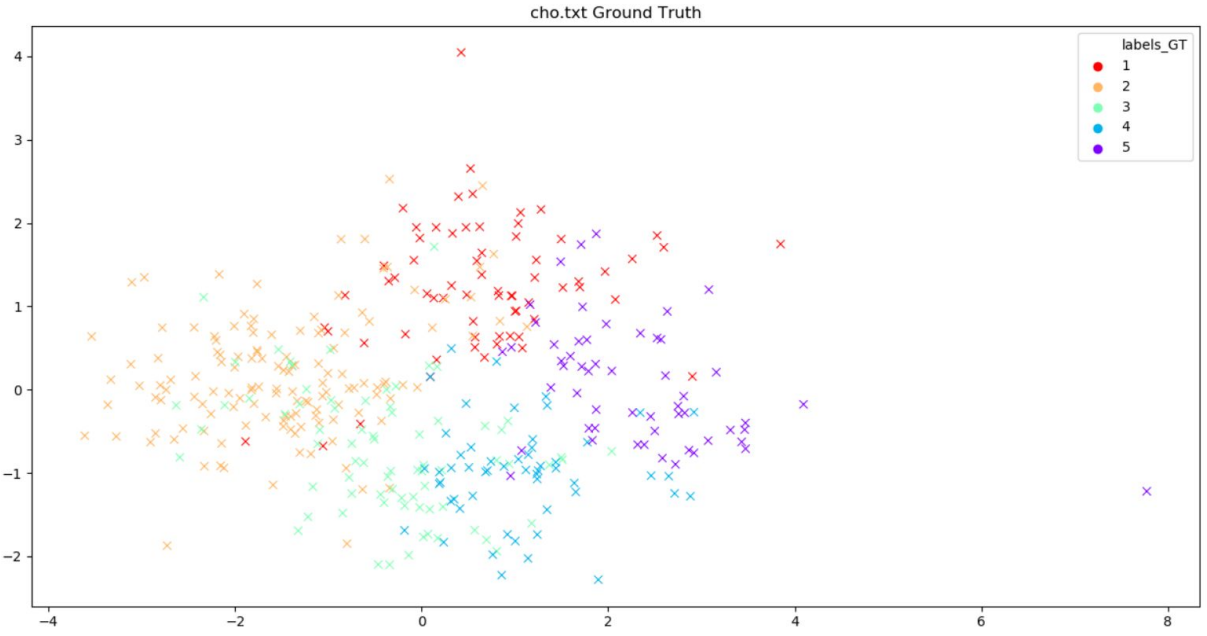
Input data set

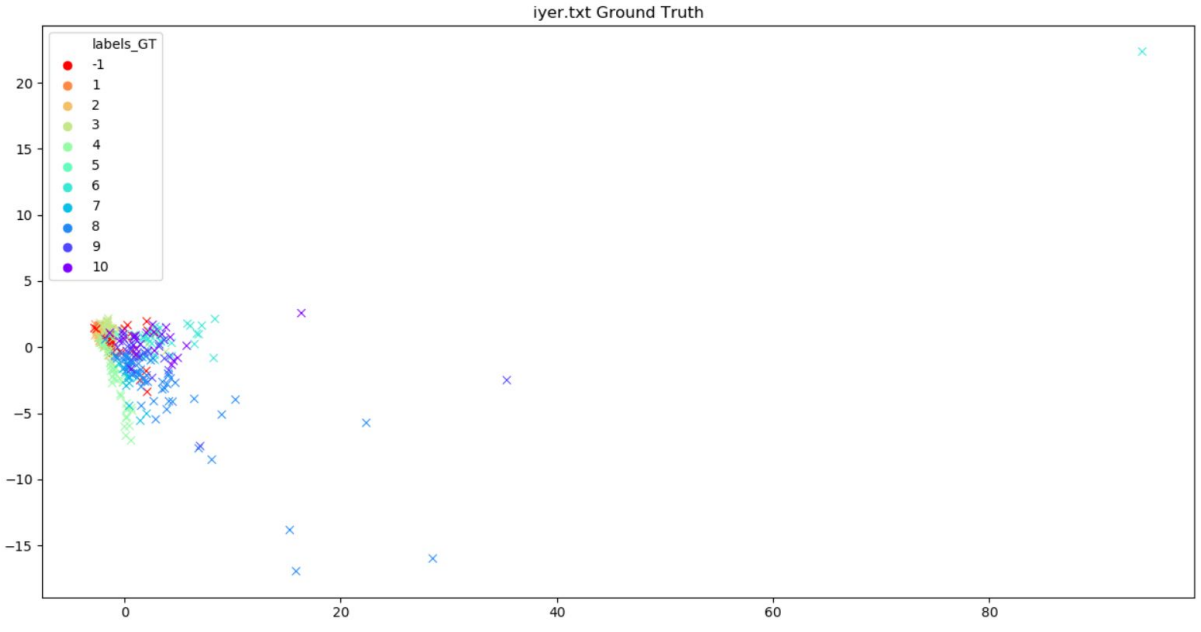
This project was implemented using two gene datasets as follows -

1. cho.txt (Dimensions - 386 x 16)
2. iyer.txt (Dimensions - 517 x 12)

The ground truths for each dataset was also provided that enabled us to validate the cluster assignments of each clustering algorithm.

On performing PCA on the input datasets, the cluster assignments using the given ground truths can be visualized as follows -





In this project, we have implemented following 5 clustering algorithms-

1. K-Means Clustering
2. Hierarchical Agglomerative Clustering
3. Density-based Clustering (DBSCAN)
4. Mixture Model (Gaussian)
5. Spectral Clustering

K-Means Algorithm

1. Implementation details:

Kmeans is a prototype based clustering, in which the prototype is measured by the centroid of all the points in the cluster.

There are three main steps in Kmeans:

1) Initialization of initial centres:
    a) We have two modes to initialize the centres initially depending on the variable 'choice'. If choice is set to hard we initialize the centres by the row id values from the dataset given as input else we randomly choose K clusters where K is a parameter. We run multiple instances of K means initialized above and then run a final kmeans with the average value of the centres obtained after each run.
2) Cluster assignment:
    a) We calculate the distance of each point with every centroid in our implementation using Euclidean distance
    b) We then assign the index of the cluster to the datapoint with the minimum distance. We use the index values of centroids for cluster assignment.

3) Reassign centres:

    a) We recalculate the centres of each of the clusters again.
    b) Repeat the process from 2. till the centres do not change or number of iterations reached max number of iterations specified as parameter

Whenever we encounter the empty clusters, we take the point that contributes most to the SSE from the cluster.

While preprocessing the data, we remove outliers in data, before it is passed to 'process_kmeans' function.

2. Analysis:

Using randomly initialized centroids

Dataset 1 (cho.txt)

- RAND coefficient - 0.7969744154205483
- Jaccard coefficient - 0.4159104074145588

Dataset 2 (iyer.txt)

- RAND coefficient - 0.7163018236459259
- Jaccard coefficient - 0.3405112531258683

3. Observations:

When run with random initialization of cluster centres, we see a wide fluctuation in the values of Jaccard and Rand coefficients. For different runs of the data we see the values of Jaccard Coefficient to vary from 0.31 to 0.39 for cho.txt data set as seen from the values of 5 runs below:

	RAND	Jaccard
Run 1	0.7535235845257591	0.31805689667978904
Run 2	0.7506912937260061	0.35637800187129637
Run 3	0.766195065639346	0.32504068821204374
Run 4	0.7694703213509088	0.3938944767954826
Run 5	0.7765174904024269	0.3821117090369271

This is because poorly chosen initial centres makes the algorithm to converge at a local maxima instead of the global one.

When initialized with better centres given from the data set we see a marked improvement in the results.

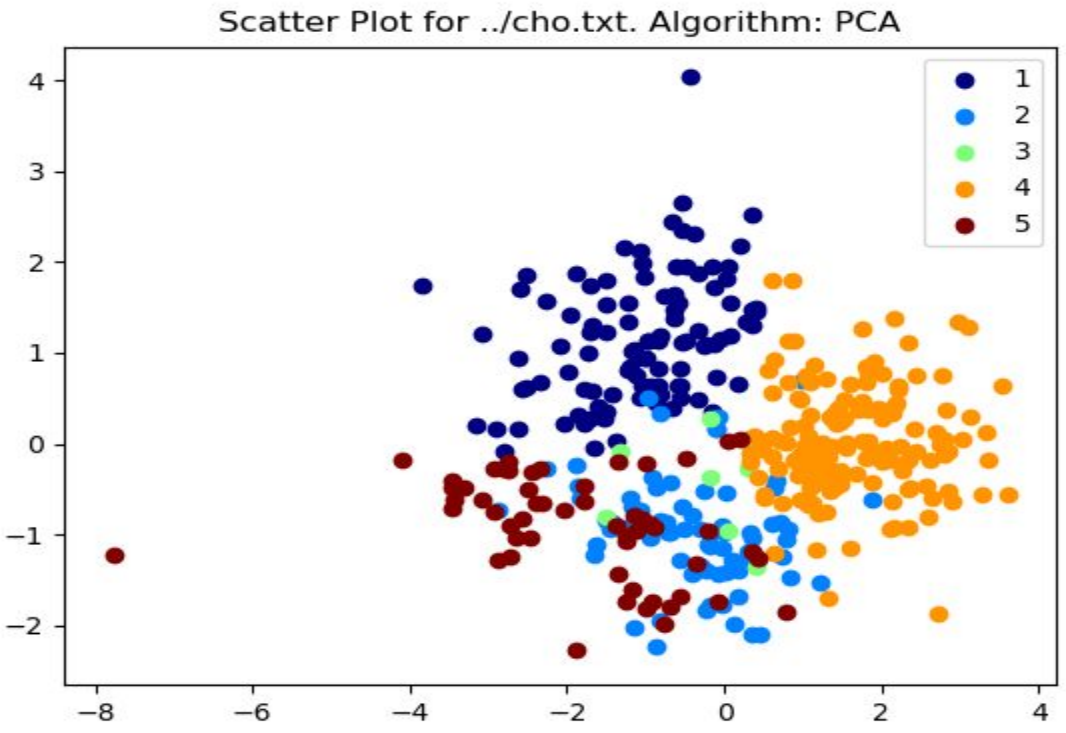
4. Advantages of K-Means Clustering:

- K-Means clustering algorithm is a very efficient algorithm. The runtime is linear in terms of data points and still less expensive as the number of clusters is always way less than the number of data points.
- Easy to implement.
- Can be used for a wide variety of data types.

5. Disadvantages of K-Means Clustering:

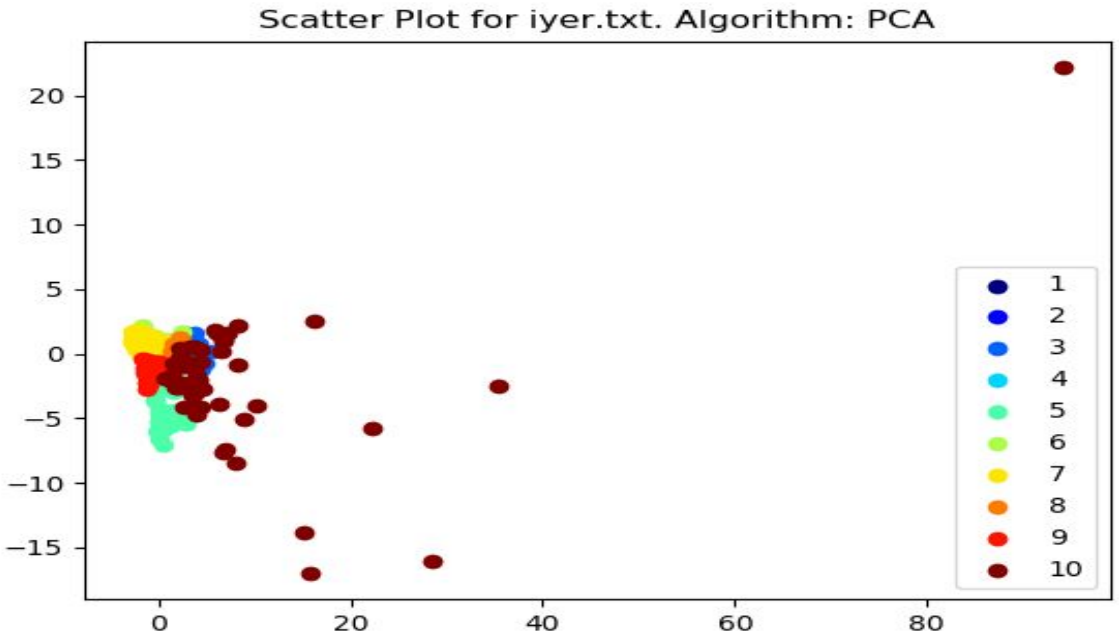
- K-Means is unable to handle clusters of different sizes or densities.
- It is very susceptible to outliers in the data.

Kmeans graph results



RAND coefficient: 0.7765174904024269

Jaccard coefficient: 0.3821117090369271



RAND coefficient: 0.7433235434738065

Jaccard coefficient: 0.3575793837343476

Hierarchical Agglomerative Clustering Algorithm (using Min approach)

1. Implementation details:

In Hierarchical Agglomerative Clustering, each datapoint is considered to be a singleton cluster and at each step, two clusters closest to each other are merged. This process is continued until the entire dataset is merged into one big cluster.

Hierarchical clustering is usually visualized using a dendogram to check the order in which the clusters are merged in a bottom-up approach. The dendogram is cut at a specific level when a certain number of clusters are required.

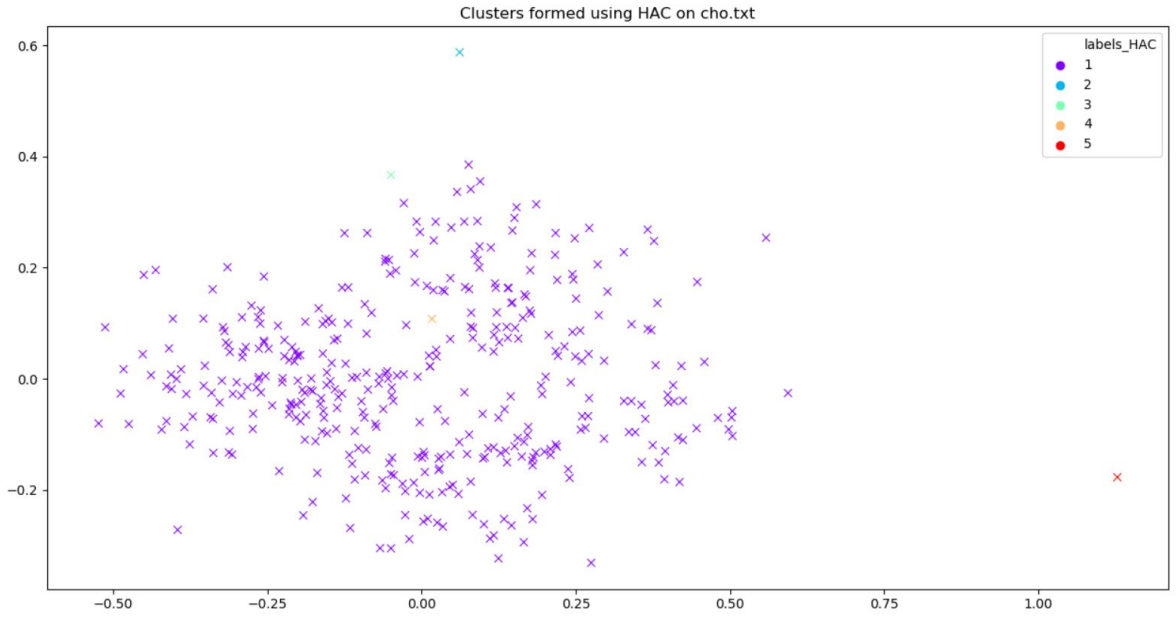
Our implementation of the DBSCAN clusters the given dataset in the following steps-

1. We first assign a cluster number to each datapoint. This is done by considering the gene id is the cluster number for the corresponding datapoint.
2. We then normalize the data and compute the distance matrix that holds Euclidean distance of each datapoint pair (Euclidean distance computation is performed using cdist() from scipy.spacial.distance package).
3. The diagonal values of the distance matrix are 0 since the distance a point with itself will be 0. For ease of computation, we set the diagonal values to infinity.

4. For the given input indicating the number of clusters to be formed, say ‘N’, we perform the following operations until the number of rows remaining in the distance matrix is N -
    a. Find the indexes of the minimum value in the distance matrix. Say ‘x’ and ‘y’ are the indexes of the minimum value in the distance matrix.
    b. This means clusters x and y are the closest cluster and must be merged in this step of computation.
    c. We merge the two clusters and store the relation in a map. While merging, the distance matrix is updated in a way such that the xth row and xth column store the minimum values between x and y. This is done since the HAC algorithm is configured to use the Single Link approach to calculate the inter cluster distances. The yth row and yth column values are set to infinity.
5. After identifying the cluster numbers for all datapoints, validation of the clusters against the ground truth is performed using RAND and Jaccard coefficients, results of which are mentioned in the analysis section below. Visual validation is also performed after performing PCA on the input dataset.

2. Analysis:

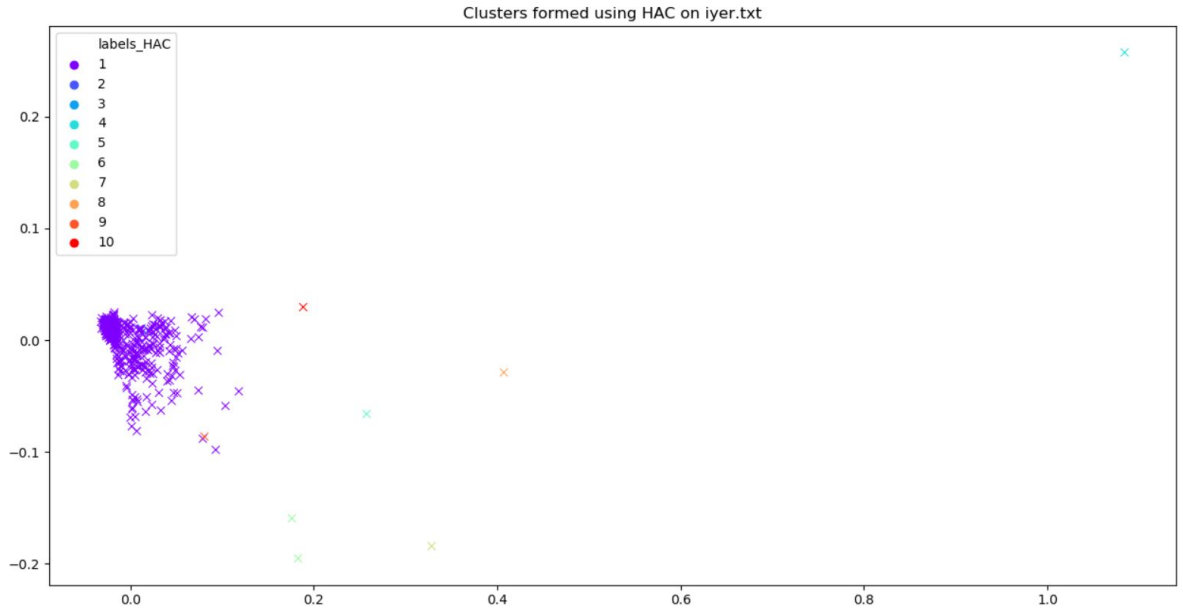
Dataset 1 (cho.txt)



RAND coefficient - 0.24027490670890495

Jaccard coefficient - 0.22839497757358454

Dataset 2 (iyer.txt)



RAND coefficient - 0.1882868355974245

Jaccard coefficient - 0.15824309696642858

3. Observations:

- It is observed that the validation coefficients for HAC algorithm are comparatively lower than those obtained in other clustering algorithms.
- Since the input dataset contains datapoint spread over a large space with varying densities and HAC algorithm using single link approach clusters datapoints based on the minimum distance between points at each step, it does not cluster the datapoints into distinct clusters very well.
- We observe that about 95% of the datapoints are clustered into a single cluster while other datapoints remained to be singleton or very small clusters.

4. Advantages of HAC Clustering:

- HAC does not depend on any input parameters. All datapoints are eventually clustered into one large cluster and the merging of the clusters can be viewed as a dendogram. The required number of clusters can be obtained by cutting the dendogram at a specific level.
- HAC clustering technique is usually helpful in generating meaningful taxonomies.
- The single link approach used to compute the inter-cluster distance in our implementation is able to handle non-circular/non-elliptical clusters.

5. Disadvantages of HAC Clustering:

- Once two clusters are merged, the decision cannot be undone at a later stage.
- Each approach used to compute inter-cluster distance (single link, complete link and average link) have different problems and therefore it is necessary to determine the correct approach to use for the given dataset based on the size, shape, density and noise in the dataset.
- The single link approach used to compute the inter-cluster distance in our implementation is sensitive to outliers. As a reason, for the given datasets (cho.txt and iyer.txt) majority of the datapoints are clustered into one large cluster.

Density-based Clustering Algorithm (DBSCAN)

1. Implementation details:

DBSCAN algorithm implemented in this project is based on the algorithm discussed in class (Clustering4.pptx Slide-11)

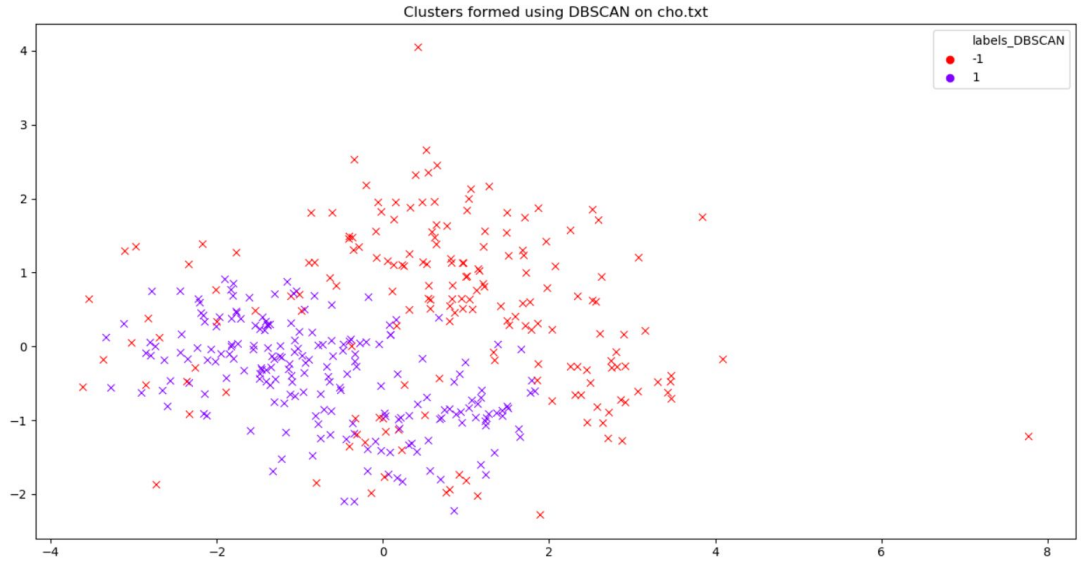
Our implementation of the DBSCAN clusters the given dataset in the following steps-

1. Initialize a cluster map that will hold the cluster numbers for each data point. Set the cluster number of each datapoint to be -1, indicating a noise point. As the algorithm proceeds, relevant cluster numbers will be assigned and stored for each datapoint.
2. Start with any datapoint (say ‘P’) in the input dataset and mark it visited.
3. Find all the neighbouring datapoints within the given ‘eps’ distance from P.
4. If the number of neighbouring datapoints is greater than or equal to the input ‘minPts’ value, it means that point P is a core point and must be expanded further to check for other density-reachable points.
5. The neighbouring datapoints are identified by computing the Euclidean distance between point P and the rest of the datapoints (Euclidean distance computation is performed using cdist() from scipy.spacial.distance package).
6. We assign a cluster number to P and scan the neighbouring datapoints that are within ‘eps’ distance of P’s unvisited neighbours. If a density-reachable neighbour under consideration has not been assigned a cluster yet, the cluster number of P is assigned to it.
7. Above operations are performed until all the datapoints in the input dataset are visited.
8. After identifying the cluster numbers for all datapoints for the given input parameters ‘eps’ and ‘minPts’ respectively, validation of the clusters against the ground truth is performed using RAND and Jaccard coefficients, results of which are mentioned in the analysis section below. Visual validation is also performed after performing PCA on the input dataset.

2. Analysis:

- eps = 1
- minPts = 4

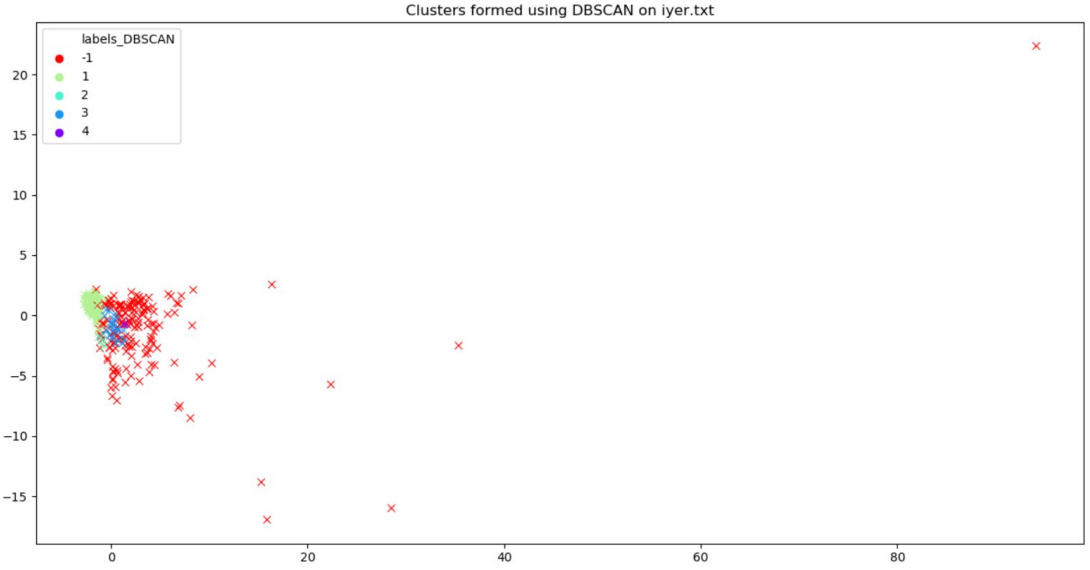
Dataset 1 (cho.txt)



RAND coefficient - 0.5232220999221455

Jaccard coefficient - 0.20366343071093873

Dataset 2 (iyer.txt)



RAND coefficient - 0.6507375911466615

Jaccard coefficient - 0.28373805961560594

3. Observations:

- It is observed that the for the given input parameters (eps = 1 and minPts = 4), DBSCAN performs poorly for dataset 1 (cho.txt) which is reflected in the corresponding coefficient values, however, DBSCAN performs fairly well for the same input parameters for dataset 2 (iyer.txt) with comparatively higher coefficient values.
- This shows that the DBSCAN algorithm is sensitive to the input parameters and differs for each dataset.
- A better result was obtained for dataset 1 when the input parameters were set to eps = 1.3 and minPts = 16 (RAND coefficient - 0.566 and Jaccard coefficient - 0.255).
- From the plots it can be observed that DBSCAN was able to identify the outliers.
- Furthermore, one can observe that since the density of the datapoints vary in different regions, the algorithm is unable to perfectly distinguish between clusters and noise points, thereby resulting in a lower Jaccard coefficient.

4. Advantages of Density-based Clustering:

- Density-based clustering identifies cluster of different shapes and sizes.
- It is useful in identifying outliers.

5. Disadvantages of Density-based Clustering:

- It is difficult to determine the correct input parameters (epsilon and minPts).
- The algorithm is very sensitive to the input parameters. Slight change in the input parameters affects the clusters formed in a significant way.
- DBSCAN does not handle varying densities within the data very well.

Mixture Model Clustering

Gaussian mixture model is a probabilistic clustering algorithm. In this model, each point is assumed to be coming from a set of Gaussian distributions and each distribution is considered to be a cluster.

Each cluster is parameterized by gaussian parameters such as mean, covariance matrix. Eventually our problem reduces to maximising likelihood of a point to find which distribution it's more likely to belong to.

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

1. Implementation details:

- The likelihood function is difficult to optimize due to the summation inside the log function in MLE.
- Therefore, we use Expectation Maximization(EM) algorithm in order to find the parameters.
- EM algorithm works in two steps:
    - E-Step: In this step we calculate the expected values of the latent variables

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

    - M-Step:
        - In this step, we update the parameters that would maximize the log likelihood.

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

        - We repeat the above two steps until the there is a very small change in the log likelihood.
- Finally we find the max value from ‘r’ that will represent the cluster to which the point belongs to.

2. Analysis:

We initialized the parameters with the following values:

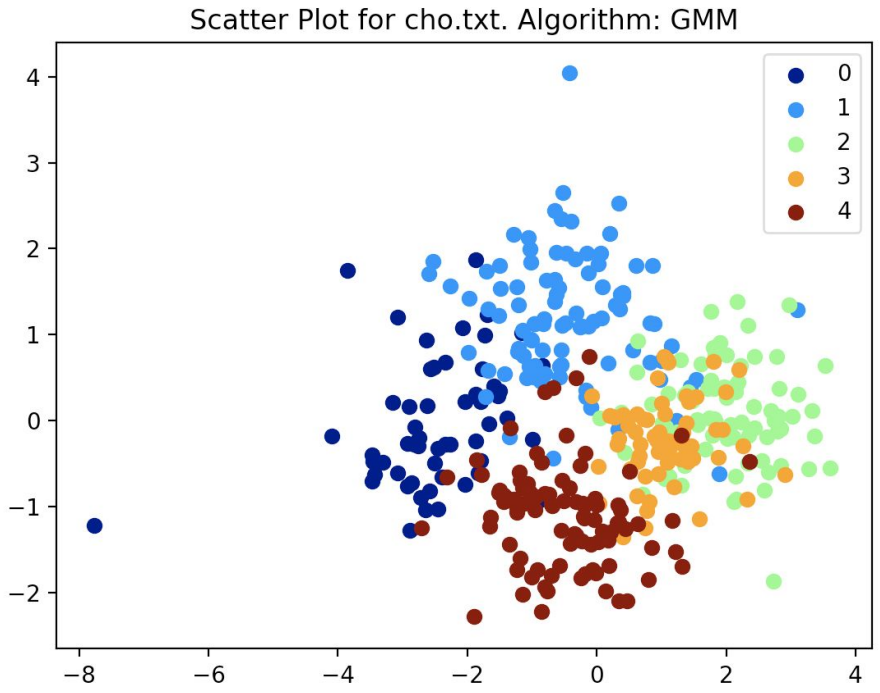
pi = [0.2, 0.2, 0.2, 0.2, 0.2]

mu = data[rand_data] where rand_data = np.random.choice(n_data, no_of_cluster, replace=False)

cov = np.zeros((no_of_cluster, dim, dim), dtype='float64')
and each diagonal is assigned a number 1.

Gaussian mixture model produces the following plots in 2D space-

Dataset 1 (cho.txt)



- RAND coefficient: 0.7784235818411233
- Jaccard coefficient: 0.32901101581236536

Dataset 2 (iyer.txt)

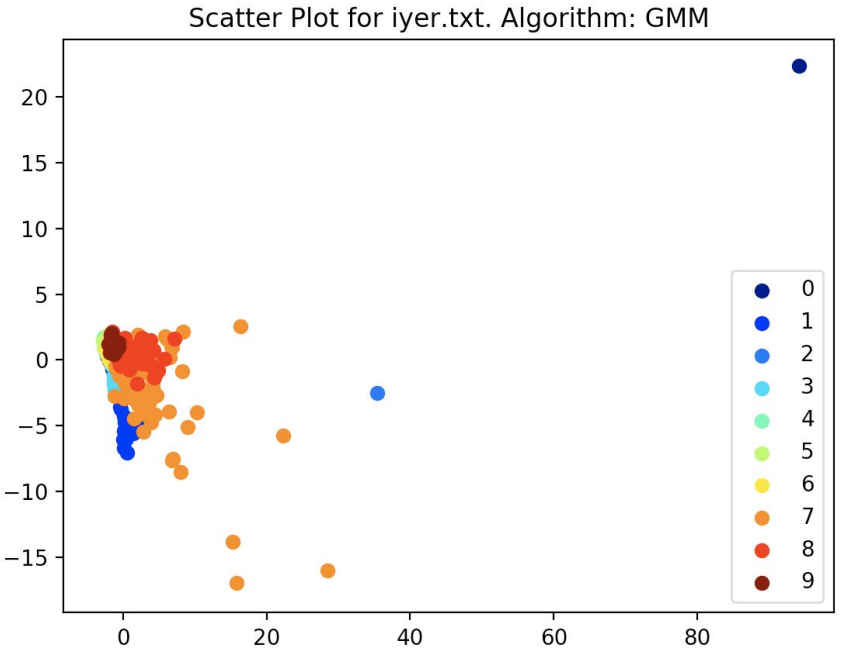
We initialized the parameters with the following values:

pi = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]

mu = data[rand_data] where rand_data = np.random.choice(n_data, no_of_cluster, replace=False)

cov = np.zeros((no_of_cluster, dim, dim), dtype='float64')
and each diagonal is assigned a number 1.

Gaussian mixture model produces below plot in 2D space.



- RAND coefficient: 0.8194538495785461
- Jaccard coefficient: 0.24207252909487836

3. Observations:

- We ran GMM on various input parameters and found that rand and jaccard coefficients changes with varying parameters. This can be attributed to the fact that initialization determines initial mean and covariance and that contributes to what points are more likely to belong to which cluster.
- If given the near-accurate initial latent variables and parameters, GMM learns the clusters only in a few iterations.

4. Advantages of GMM Clustering:

- Gaussian mixture model can learn cluster with varying latent variables.
- Unlike K-Means, GMM can learn non-circular clusters as well.

5. Disadvantages of GMM Clustering:

- Results depend on the initial values of the parameters such as mu, sigma, and pi.
- Deciding on distribution plays an important role in performance on a mixture model.

Spectral Clustering Algorithm

1. Implementation details:

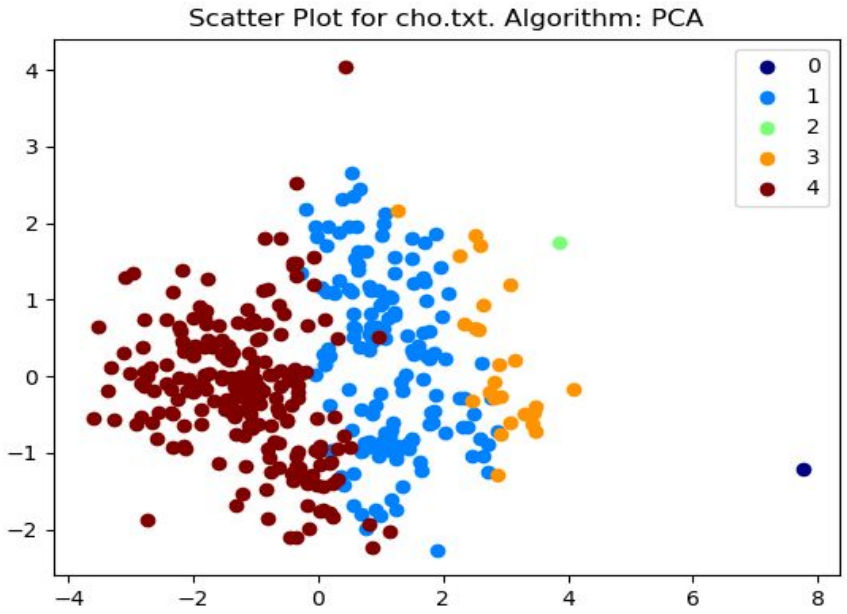
Sigma, num_clusters, max_iters, choice, and centers are the input parameters required by the algorithm.

Following are the steps implemented to perform Spectral Clustering for the given datasets-

1. We first get the affinity matrix (W) using the Gaussian kernel as a similarity measure between each point.
2. The diagonal degree matrix 'D' is computed
3. L is the laplacian matrix as D - W.
4. We calculate the eigenvalue and eigenvectors of the Laplacian matrix as eig_val and eig_vec respectively.
5. We then determine 'k', which is the higher value index of the eigenvalue that maximises the eigen_gap
6. No_of_clusters and max_iters is the user specified parameter that is passed to KMeans function from the scikit library.
7. When the choice value is "hard" we take the specified row values as the initial centres for the KMeans step, else we use random initialization of values.

2. Analysis:

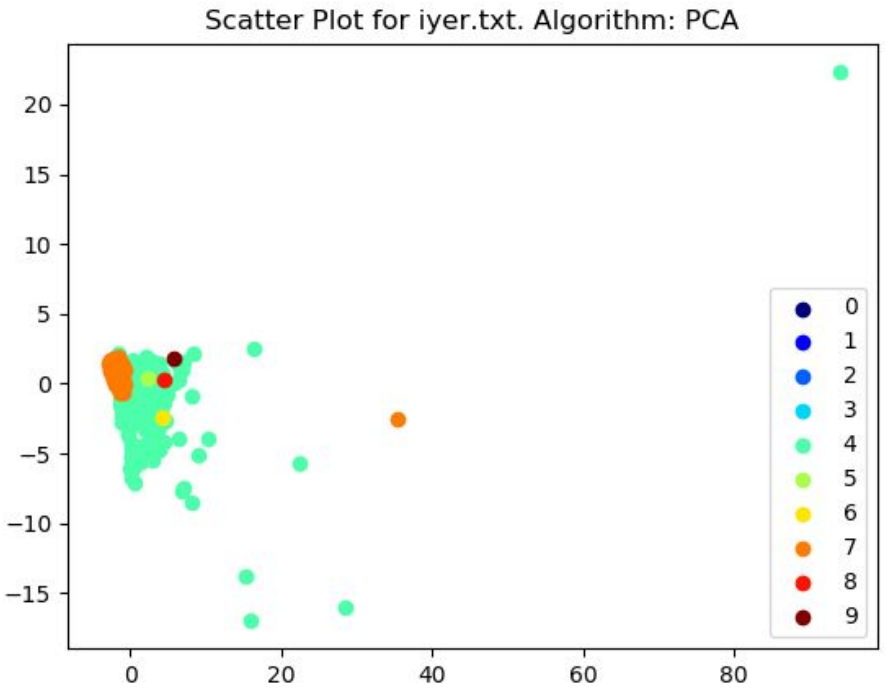
Dataset 1 (cho.txt) for sigma = 3.5



Rand Coefficient: 0.6620311954683347

Jaccard Coefficient: 0.33551502995434274

Dataset 2 (iyer.txt) sigma =2.75



RAND coefficient - 0.6894896535210951

Jaccard coefficient - 0.3070095603890953

3. Observations:

- Increasing the sigma value increases the segregation of points into different clusters, but this is not necessarily accurate as the clusters observed by the algorithm is different than actual truth. Hence there is a great difference in the Rand coefficient than jaccard. For cho.txt dataset best results were observed with sigma around 3.5
- Since points in iyer.txt are very close to each other (when observed in a two dimensional plot), the best results for iyer.txt are observed around a much lower sigma value(2.75).
- Zero values of eigenvectors correspond to the number of connected components in the graph. Near zero eigenvalues thus indicate an almost connected component or a cluster.

4. Advantages of Spectral Clustering:

- The algorithm is reasonably fast for small sparse data sets.
- It makes no assumption about space of the clusters.
- It is easy to implement.

5. Disadvantages of Spectral Clustering:

- Since the final step involves the use of K-Means, it is not guaranteed to give the same clusters and suffers from similar drawbacks as of K-Means.
- Calculation of eigenvectors and eigenvalues make this algorithm an expensive affair for large datasets.