CSE 601 : Data Mining and Bioinformatics
CLUSTERING ALGORITHMS

PRIYANKA PAI (ppai3@buffalo.edu) -50295903
HRISHIKESH NITTURKAR (hnitturk@buffalo.edu)-50291411
NAGA VAISHNAVI PAKYALA (nagavais@buffalo.edu) - 50290561

K-MEANS CLUSTERING

K-means Clustering algorithm uses K Centroids as centers for the clusters. The clustering happens based on the euclidean distances between every point in the dataset and each of the centroids. Each point belongs to the cluster with the nearest centroid as the center of the cluster. K-Means finds the best centroids by updating the current centroids by taking the mean of all the point coordinates of a particular cluster. K-Means continues to run until the centroids can no more be updated or when it reaches the specified maximum number of iterations.

IMPLEMENTATION OF K-MEANS CLUSTERING ALGORITHM

Language used: Python 3.6

Dynamic Inputs: Initial centroids, K, Max Iterations and a Dataset D.

This algorithm has three main steps. 1. Initialize K initial centroids. 2. Assignment of Clusters based on the minimum euclidean Distances. 3. Update the Centroids. Repeat Step 1, 2, 3 until no points are left.

1. The data set given at hand has a set of features, but has no labels. If no initial centroids are provided, initialize centroids at random from the given dataset based on the value of K.
2. Find the euclidean distances between every point in the dataset and the centroids and pick (point, centroid) with the minimum distance.
3. Assign points to clusters which has the centroid as the center.
4. After assigning every point in the input to a cluster and update each of the K centroids based on the mean of all the points in each of the clusters.
5. Repeat Steps 1,2,3,4 until the centroids are no longer updated with the new values.
6. Calculate the Jaccard and Rand index for the final cluster formed to provide similarities between the clusters using the following:

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

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

A pair of data object (O_i,O_j) falls into one of the following categories

-M_{11} : C_{ij} =1 and P_{ij}= 1 (agree)

M_{00} : C_{ij} =0 and P_{ij}= 0 (agree)

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

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

Where P is the set of ground truth clusters and C is a set of Clusters reported by the DBSCAN Algorithm.

7. Plot the graph after performing PCA on the input data. Use the data returned by PCA as the coordinates and the Cluster list returned by the algorithm as labels for the graph.

Pros:

- K-Means is very efficient and has linear runtime and easy to implement
- Runs with time complexity O(t * k * n), where k is the number of clusters, t is the number of iterations and n is the number of data points. Thus the algorithm works very efficiently for low values of k.
- Closely packed clusters are formed if the clusters are globular

Cons:

- Difficulty in predicting optimal value of k.
- K-means has problems when clusters are of differing Sizes, Densities, Irregular shapes. Initialization of parameters needs to be efficient.
- Works poorly when clusters aren't similarly sized and are of varying densities
- Clustering results may not be good when the original clusters aren't globular in shape
- The algorithm is dependent on the random points chosen initially, different initial points may result in different cluster assignment

EXPERIMENTAL FINDINGS FOR K-MEANS CLUSTERING

Input parameters:

Dataset = Cho.txt

Initial Centroids= [Gene IDs: 1, 3, 5, 7, 9]

Max Iterations= 100

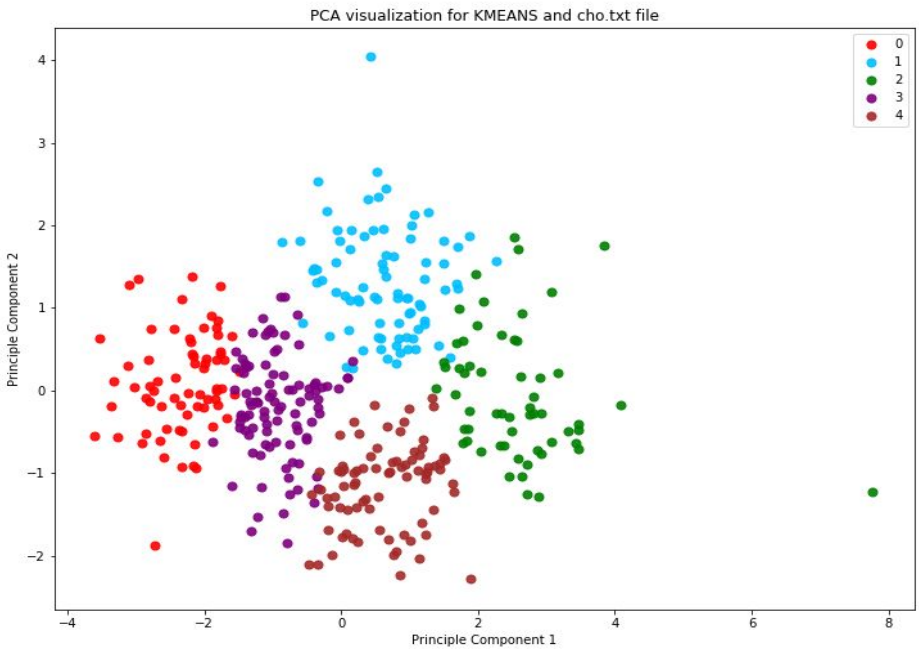
K= No. of Initial Centroids

Results:

Rand value: 0.7829069236758034

Jaccard value: 0.3350191192796349

Plot:



Input parameters:

Dataset = Iyer.txt

Initial Centroids= [Gene IDs: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21]

Max Iterations= 100

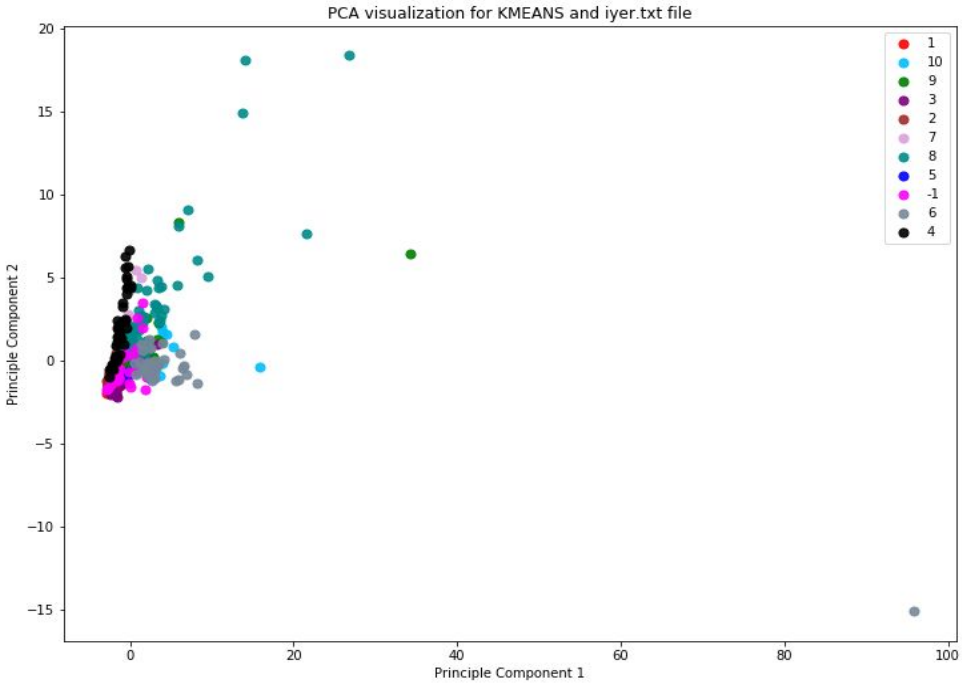
K= No. of Initial Centroids

Results:

Rand value: 0.7777686324540104

Jaccard value: 0.2948131967281232

Plot:

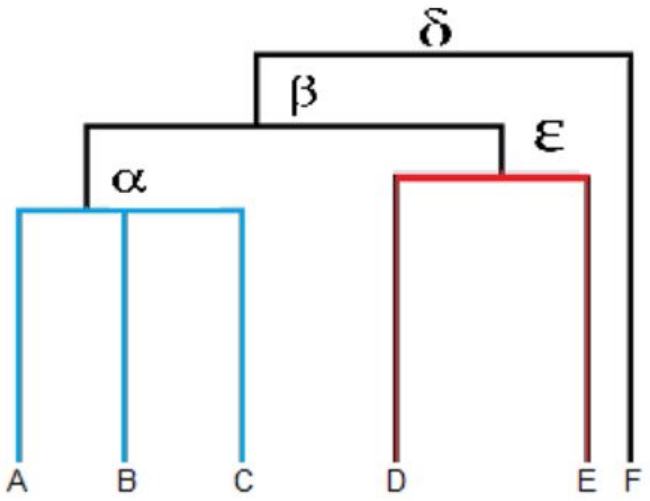

PCA visualization for KMEANS and iyer.txt file
Principle Component 2
Principle Component 1
20
15
10
5
0
−5
−10
−15
0
20
40
60
80
100
1
10
9
3
2
7
8
5
-1
6
4

HIERARCHICAL CLUSTERING

Hierarchical Agglomerative Clustering is a clustering algorithm which initially starts with each object being a cluster of its own and iterates over every point merging two clusters which are closest to each other until all objects are merged into a single cluster.

Dendrogram :

Hierarchical clustering is often displayed graphically using a tree-like diagram called a dendrogram,which displays both the cluster-subcluster relationships and the order in which the clusters were merged.The clades are arranged according to how similar (or dissimilar) they are. Clades that are close to the same height are similar to each other; clades with different heights are dissimilar — the greater the difference in height, the more dissimilarity



- Leaves A, B, and C are more similar to each other than they are to leaves D, E, or F.
- Leaves D and E are more similar to each other than they are to leaves A, B, C, or F.
- Leaf F is substantially different from all of the other leaves.

IMPLEMENTATION OF HIERARCHICAL CLUSTERING ALGORITHM

Language used: Python 3.6

Dynamic Inputs: Value of N (Number of Clusters) and a Dataset D.

1. Initially each item in the dataset is a cluster of its own and the distance matrix is calculated based on the euclidean distance between each of the clusters.
2. Find the closest pair of clusters by taking the minimum distance and merge them into a single cluster and add them as a new entry to the distance matrix. Remove the corresponding rows and columns of the points and now we have one less cluster.
3. Compute and update the distances between the new cluster and each of the old clusters by taking minimum distance between old clusters and each cluster in the new clusters.
4. Repeat Steps 2 and 3 until one single cluster is formed in the entire dataset.
5. Calculate the Jaccard and Rand index for the final cluster formed to provide similarities between the clusters using the following:

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

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

A pair of data object (O_i,O_j) falls into one of the following categories

-M_{11} : C_{ij} =1 and P_{ij}= 1 (agree)

M_{00} : C_{ij} =0 and P_{ij}= 0 (agree)

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

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

Where P is the set of ground truth clusters and C is a set of Clusters reported by the DBSCAN Algorithm.

6. Plot the graph after performing PCA on the input data. Use the data returned by PCA as the coordinates and the Cluster list returned by the algorithm as labels for the graph.

Pros:

- No prior information about the number of clusters required.Any desired number of clusters can be obtained by 'cutting' the dendrogram at the proper level.
- It can produce an ordering of the objects, which is informative for data display (visualized mostly as dendrograms)
- It is shown to capture concentric clusters.We can create smaller clusters depending on where we cut the dendrograms which is helpful for discovery

Cons:

- Once two clusters are combined, it cannot be separated. No objective function is directly minimized. The algorithm can be sensitivity to noise and outliers and face difficulty handling different sized clusters and irregular shapes.
- Greedy – less accurate and computationally expensive
- A single-link clustering can produce a chaining effect i.e. it produces straggling clusters. As the merge criterion is local, a chain of points can be extended for long distances without regard to the overall shape of the emerging cluster

EXPERIMENTAL FINDINGS FOR HIERARCHICAL AGGLOMERATIVE CLUSTERING

Input parameters:

Dataset = Cho.txt

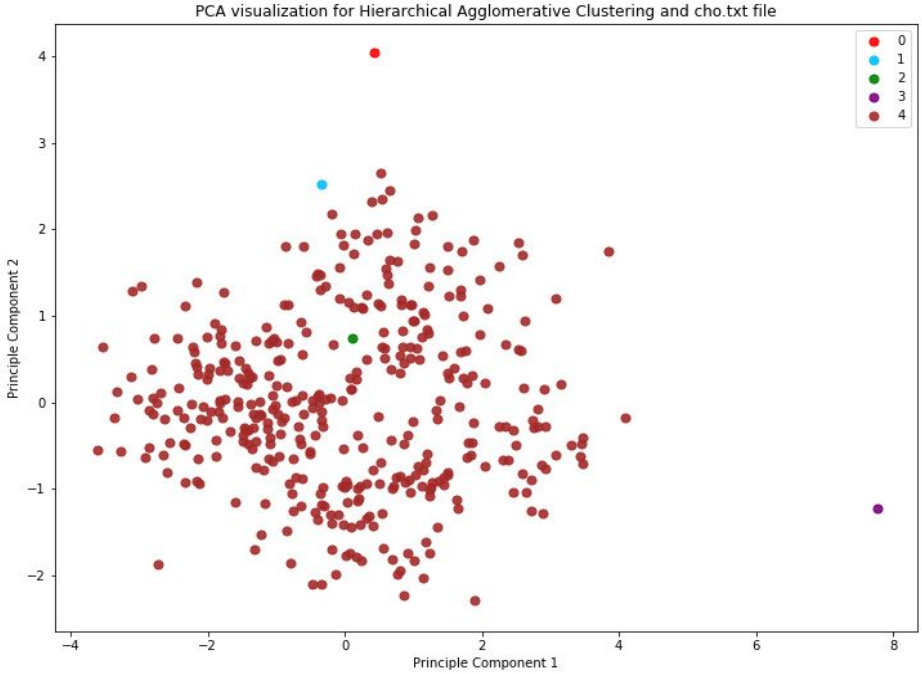
N= 5

Results:

Rand value: 0.24027490670890495

Jaccard value: 0.22839497757358454

Plot:



Input parameters:

Dataset = Iyer.txt

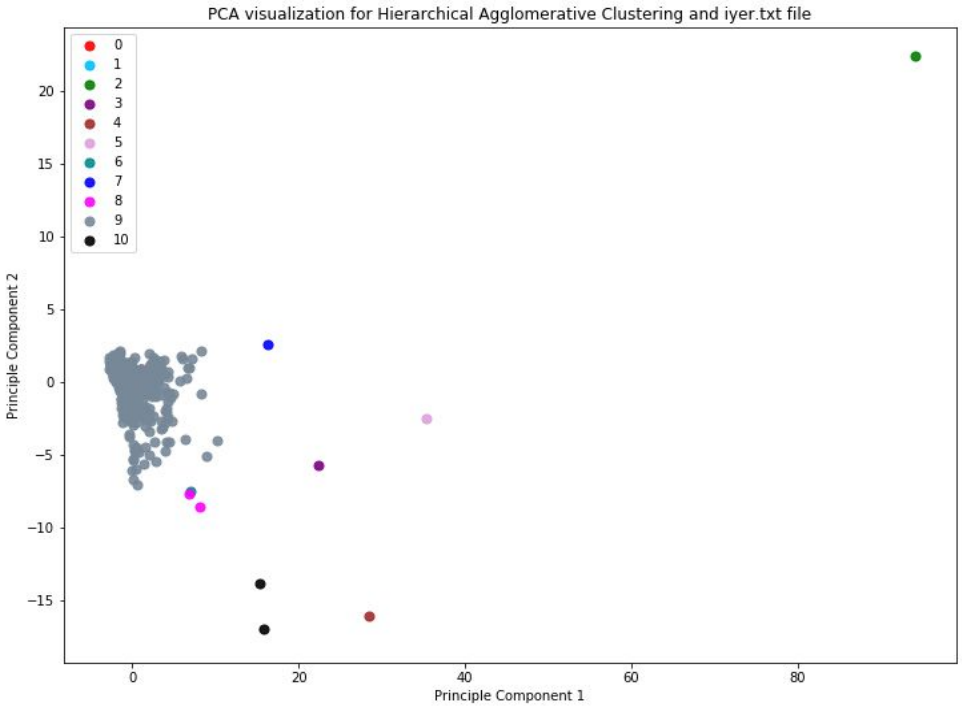
N= 11

Results:

Rand value: 0.1941381800223728

Jaccard value: 0.1584602101134175

Plot:



DENSITY BASED

The aim of density based clustering is to group together a set a points that are closely packed (points with many close neighbours) and marks the points that lie alone (in low density regions) as outliers.Density-reachable: An object q is directly density-reachable from object p if q is within the ε-neighborhood of p and p is a core object.Density-connectivity: An object p is density-connected to object q with respect to ε and minimum number of points if there is an object o such that both p and q are density-reachable from o with respect to ε and minimum number of points.

IMPLEMENTATION OF DBSCAN ALGORITHM

Language used: Python 3.6

Dynamic Inputs: Minimum points(MinPts), epsilon(Eps) and Dataset D.

1. Given e and MinPts, categorize the objects into three exclusive groups- Border Point, Noise point and Core points. A point is a core point if it has more than a specified number of points (MinPts) within a specified minimum euclidean distance (Eps) —These are points that are at the interior of a cluster. A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point. A noise point is any point that is neither a core point nor a border point.
2. Traverse through all the points in the Dataset and for every point P in D, if P is encountered for the first time, mark it as visited and find all the points within P's eps-neighbourhood and mark them as neighbouring points.
3. Initialize a new cluster C. If the count of neighbouring points is greater than the MinPts, expand the cluster by adding P to the cluster C. Traverse through all points in the neighbouring points of P and for every point P' in the neighbouring points, repeat Step 2. If P' is not a member of any cluster, add it to the cluster C. If the count of neighbouring points is less than the MinPts, mark it as noise.
4. Repeat this until all the points are clustered.
5. Calculate the Jaccard and Rand index for the final cluster formed to provide similarities between the clusters using the following:

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

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

A pair of data object (O_i,O_j) falls into one of the following categories

-M_{11} : C_{ij} =1 and P_{ij}= 1 (agree)

M_{00} : C_{ij} =0 and P_{ij}= 0 (agree)

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

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

Where P is the set of ground truth clusters and C is a set of Clusters reported by the DBSCAN Algorithm.

6. Plot the graph after performing PCA on the input data. Use the data returned by PCA as the coordinates and the Cluster list returned by the algorithm as labels for the graph.

Pros:

- DBSCAN algorithm is resistant to noise and can handle clusters of different shapes and sizes.
- DBSCAN has a time complexity of $O(n^2)$ and space complexity of O(n), where n is the number of data points.
- DBSCAN doesn't require to specify the number of clusters, as opposed to k-means.
- DBSCAN can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster.
- DBSCAN requires just 2 parameters and is mostly insensitive to the ordering of the points in the database.

Cons:

- DBSCAN cannot handle varying densities because then choosing a meaningful distance measure (ε) can be difficult.
- DBSCAN cannot handle cluster data sets well with large differences in densities, since the minimum points – ε combination cannot then be chosen appropriately for all clusters.
- Due to the above 2 points, it is hard to determine the correct set of parameters, and thus it is highly sensitive to the parameters.
- DBSCAN is not entirely deterministic. Border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed.
- The quality of DBSCAN depends on the distance measure used in the regionQuery function. Mostly Euclidean distance is used. But for high dimensional data, this measure is rendered almost useless due to curse of dimensionality

EXPERIMENTAL FINDINGS FOR DBSCAN CLUSTERING

Input parameters:

Dataset = Cho.txt

Epsilon = 1

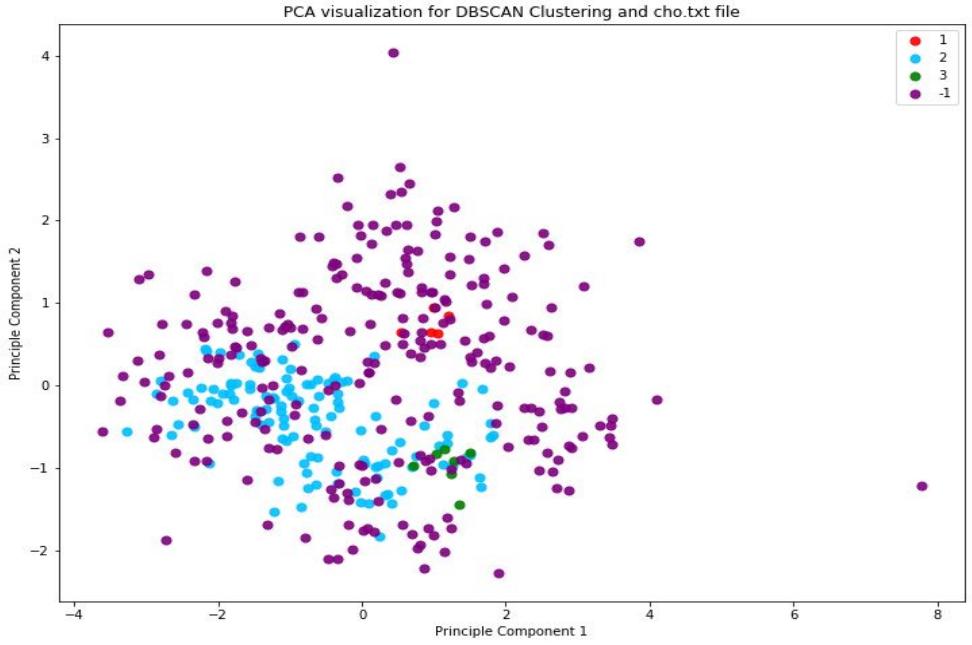
MinPoints = 5

Results:

Rand value: 0.4983086794276356

Jaccard value: 0.20487182214658015

Plot:



Input parameters:

Dataset = Iyer.txt

Epsilon = 1

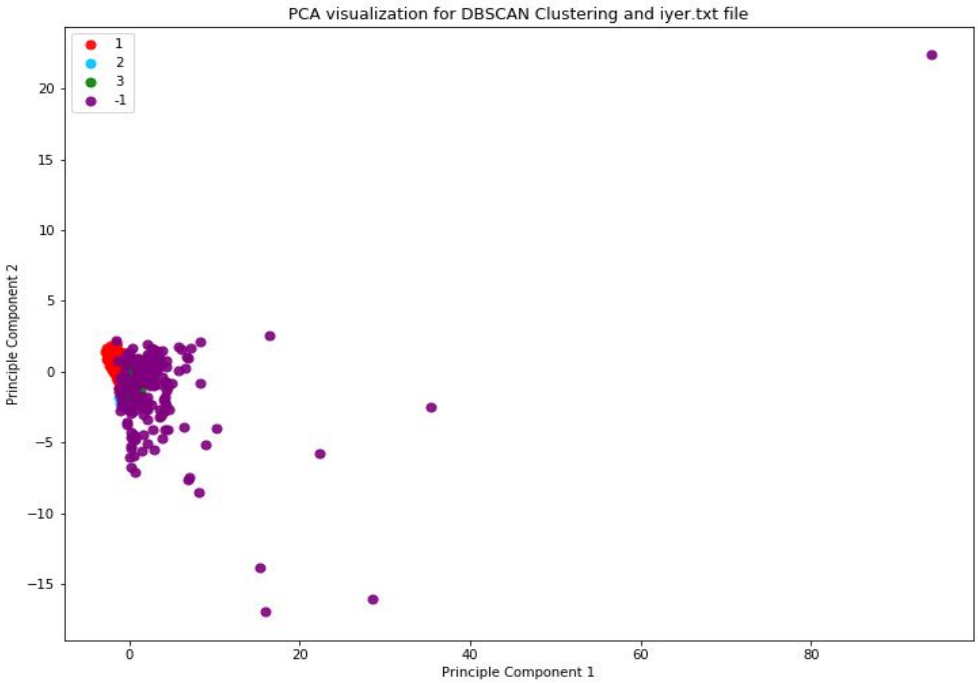
MinPoints = 5

Results:

Rand value: 0.646120865430302

Jaccard value: 0.28291902628366955

Plot:


PCA visualization for DBSCAN Clustering and iyer.txt file
1
2
3
-1
Principle Component 2
Principle Component 1
20
15
10
5
0
−5
−10
−15
0
20
40
60
80

SPECTRAL CLUSTERING

Spectral Clustering is a clustering algorithm with roots in graph theory, where the approach is used to identify communities of nodes in a graph based on the edges connecting them. The method is flexible and allows us to **cluster** non graph data as well.
Spectral Algorithm treats clustering as a graph partitioning problem without making specific assumptions on the form of the clusters.

IMPLEMENTATION OF SPECTRAL CLUSTERING ALGORITHM

Language used: Python 3.6

Dynamic Inputs: Value of K, sigma value, Initial Centroids, Max Iterations and a Dataset D

1. Initially each item in the dataset is a cluster of its own and the distance matrix is calculated based on the euclidean distance between each of the clusters.
2. Define an Affinity matrix, using a Gaussian Kernel using the formula. In higher dimensions, this is generalized to: $w_{ij} = \exp(-\| x_i - x_j \|^2 / \sigma^2)$
3. Calculate the degree matrix D where D_{ii} is the sum of all the elements in the i^{th} row of the affinity matrix.
4. Construct the Laplacian Matrix from the above two values by subtracting the affinity matrix from the degree matrix: Graph Laplacian= Degree Matrix – Affinity Matrix
5. Generate Eigen values and Eigen vectors of the laplacian matrix. Find the value of K that maximizes the Eigen gap.
6. Generate a reduced data space of order n x K from the eigen vectors where K is the number of columns in each row and n is the number of rows.
7. Perform K-Means clustering on the new data space according to the parameters provided.
8. Calculate the Jaccard and Rand index for the final cluster formed to provide similarities between the clusters using the following:

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

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

A pair of data object (O_i,O_j) falls into one of the following categories

-M_{11} : C_{ij} =1 and P_{ij}= 1 (agree)

M_{00} : C_{ij} =0 and P_{ij}= 0 (agree)

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

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

Where P is the set of ground truth clusters and C is a set of Clusters reported by the DBSCAN Algorithm.

9. Plot the graph after performing PCA on the input data. Use the data returned by PCA as the coordinates and the Cluster list returned by the algorithm as labels for the graph.

Pros:

- Does not make strong assumptions on the statistics of the clusters
- Easy to implement
- Good clustering results.
- Reasonably fast for sparse data sets of several thousand elements.

Cons:

- May be sensitive to choice of parameters
- Computationally expensive for large datasets

EXPERIMENTAL FINDINGS FOR SPECTRAL CLUSTERING

Input parameters:

Dataset = Cho.txt

K= [Gene IDs: 1, 2, 3, 4, 5]

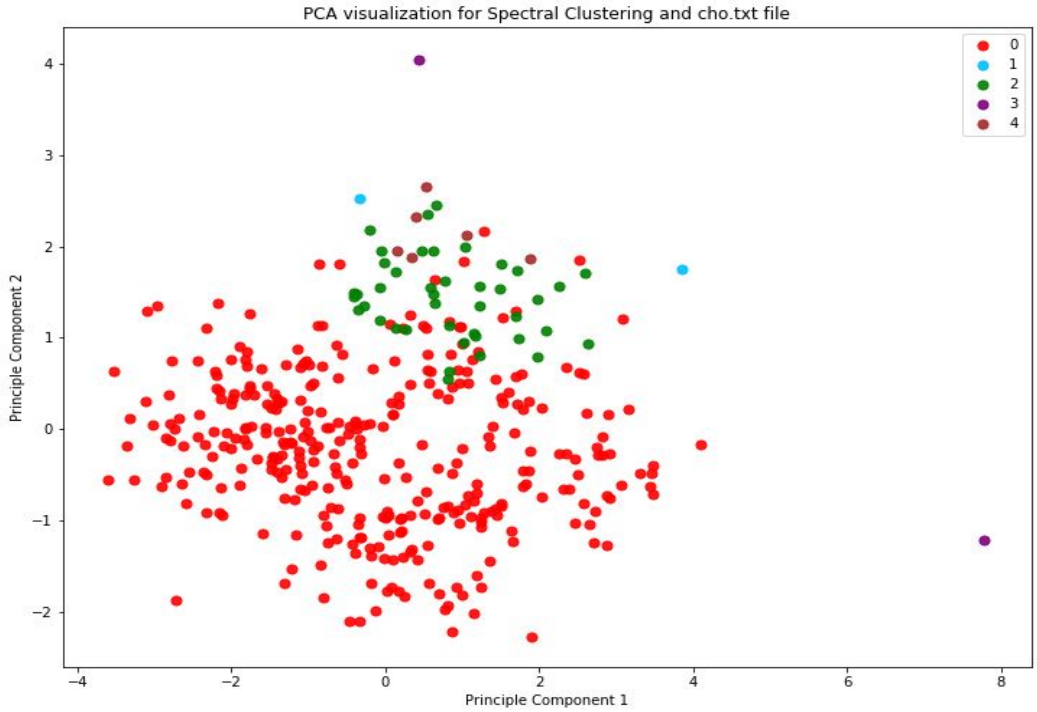
Sigma = 2

Results:

Rand value: 0.3927622218046122

Jaccard value: 0.24008063161431212

Plot:



Input parameters:

Dataset = lyer.txt

K= [Gene IDs: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

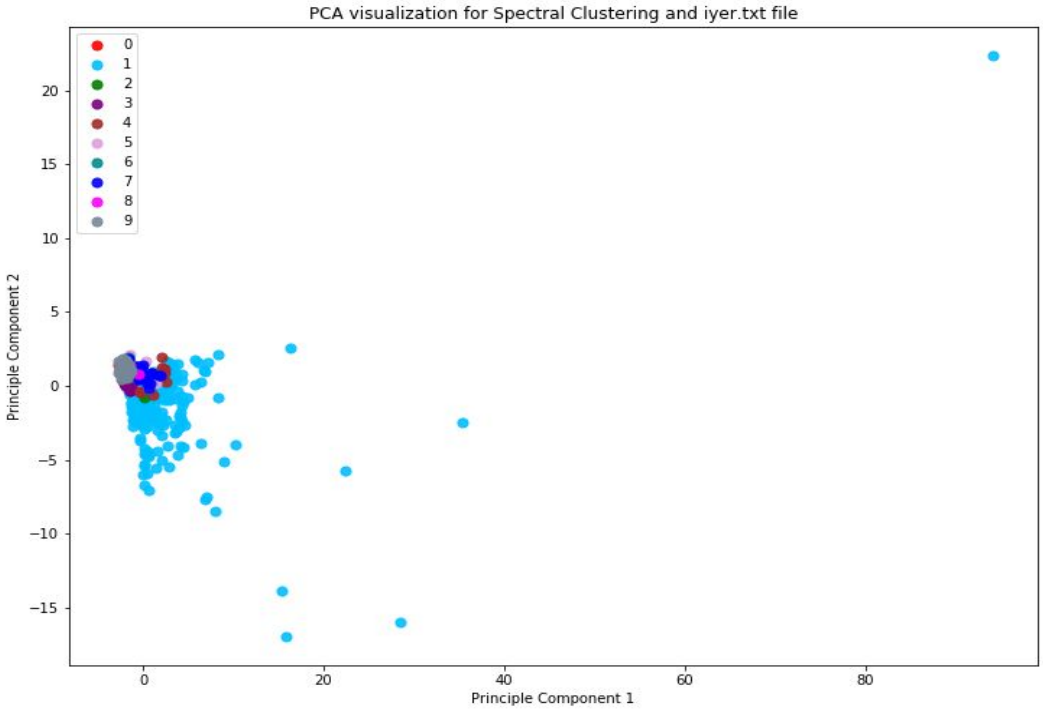
Sigma = 2

Results:

Rand value: 0.7458630920090239

Jaccard value: 0.27522593173500637

Plot:


PCA visualization for Spectral Clustering and iyer.txt file
0
1
2
3
4
5
6
7
8
9
Principle Component 2
Principle Component 1
20
15
10
5
0
−5
−10
−15
0
20
40
60
80

GAUSSIAN MIXTURE MODEL CLUSTERING

Expectation–maximization (**EM**) **algorithm** is an iterative method to find maximum likelihood estimates of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the *E* step.

IMPLEMENTATION OF GAUSSIAN MIXTURE MODEL CLUSTERING ALGORITHM

Language used: Python 3.6

Dynamic Inputs: This algorithm takes in the value of sigma,mu and phi

In this approach we initialise each latent variable such as centroid (mean), covariance , and the size of the cluster(Weight) with estimated or random values. Here rather than identifying clusters by "nearest" centroids, we fit a set of k gaussians to the data. And we estimate gaussian distribution parameters such as mean and Variance for each cluster and weight of a cluster. After learning the parameters for each data point we can calculate the probabilities of it belonging to each of the clusters.

1. Calculate the Probability density function which returns the pdf value for a given row in the input data and each of the means and the variances using the following formula:

PDF= **1/(np.sqrt(2*np.pi*variance))* np.exp(-(((data - mean))*(data-mean))/(2*variance))**

2. Perform the **E-Step (Expectation Step)** where for every element 't' in the dataset calculate the sum of all the pdf values and update the pdf value matrix with this value and then take average across the row of the matrix.
3. Perform the **Maximization Step** the values for the **mean(mu), Variance(sigma) and weights (pi).** This involves updating these values for each iteration.
4. Calculate the Log-Likelihood for the updated mean and variance matrices.
5. Check the **Exit condition**. If the present Log-Likelihood is greater than the previous Log-Likelihood by the provided convergence threshold value,the algorithm stops and exits or else repeat the procedure from step 2.
6. Repeat this procedure till the Log-Likelihood value of the Latent variables doesn't increase further and we get the maximum log-likelihood

7. Take the index of the maximum pdf value from the weight matrix corresponding to each point in the data and assign the point to the maximum indexed cluster.
8. Calculate the Jaccard and Rand index for the final cluster formed to provide similarities between the clusters using the following:

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

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

A pair of data object (O_i,O_j) falls into one of the following categories

-M_{11} : C_{ij} =1 and P_{ij}= 1 (agree)

M_{00} : C_{ij} =0 and P_{ij}= 0 (agree)

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

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

Where P is the set of ground truth clusters and C is a set of Clusters reported by the DBSCAN Algorithm.

9. Plot the graph after performing PCA on the input data. Use the data returned by PCA as the coordinates and the Cluster list returned by the algorithm as labels for the graph.

Pros:

- GMM gives probabilistic cluster assignments and have probabilistic interpretation
- GMM can handle clusters with varying sizes, variance etc.

Cons:

- The way the clusters are initialized and choosing appropriate distributions matters
- GMM is also prone overfitting issues

EXPERIMENTAL FINDINGS FOR GMM CLUSTERING

Input parameters:

Dataset = Cho.txt

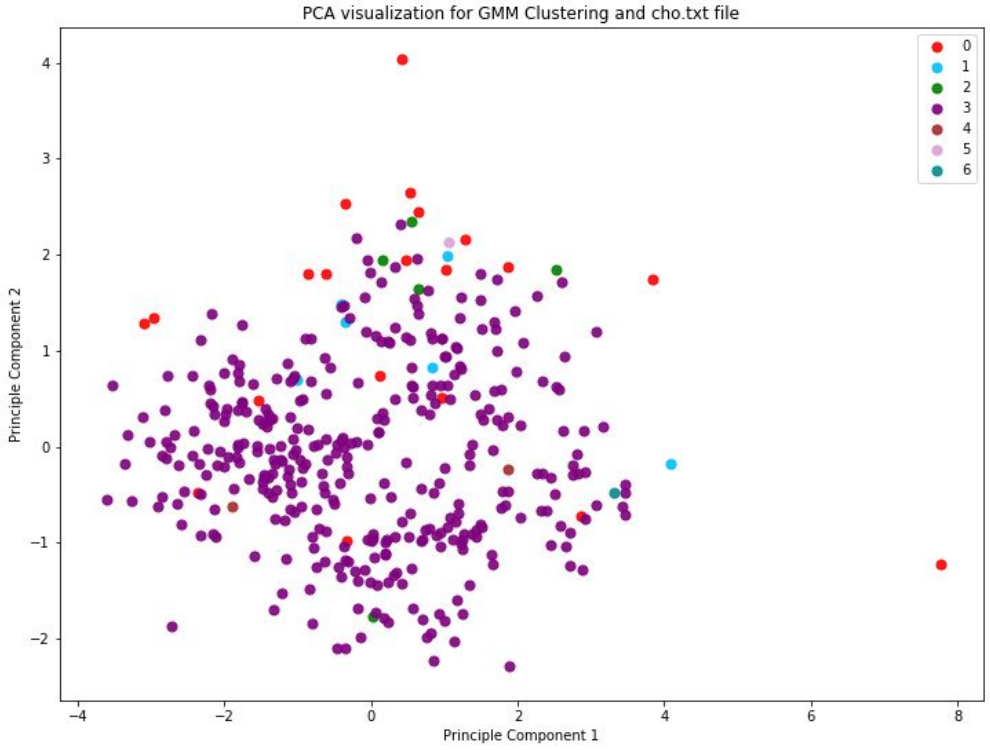
K= [Gene IDs: 1, 2, 3, 4, 5]

Results:

Rand value: 0.9526312217626476

Jaccard value: 0.856722635537266

Plot:



<u>Input parameters:</u>

Dataset = Iyer.txt

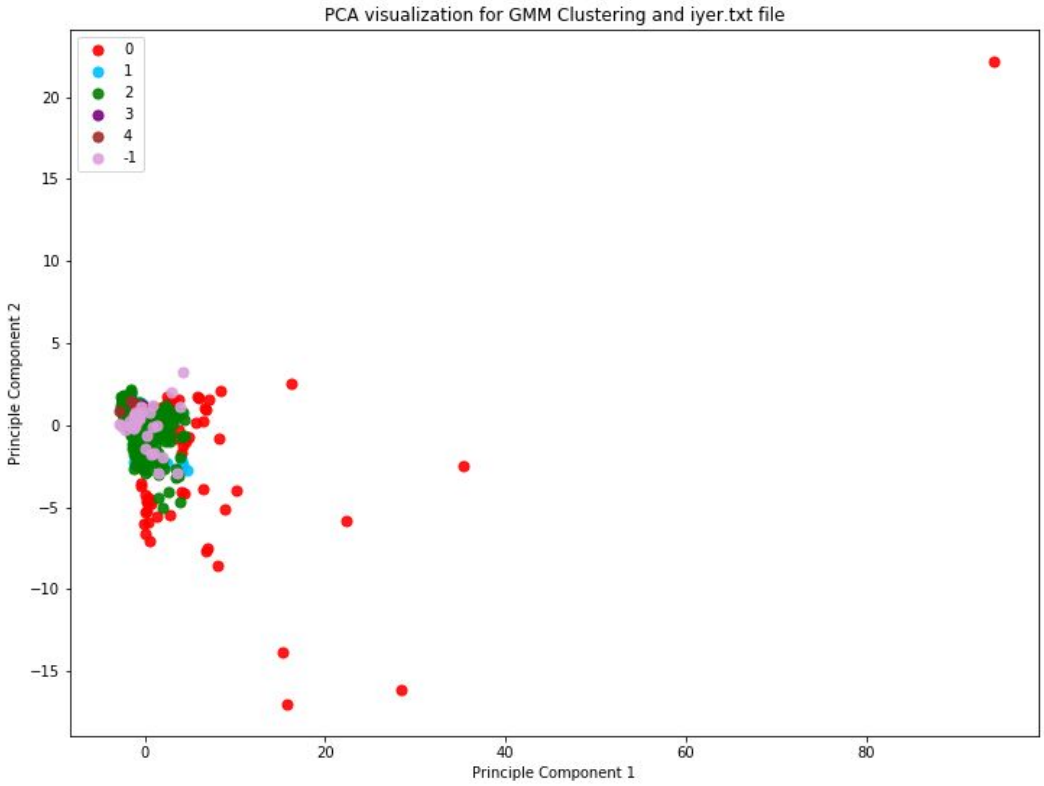
K= [Gene IDs: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

<u>Results:</u>

Rand value: 0.7458630920090239

Jaccard value: 0.27522593173500637

Plot:


PCA visualization for GMM Clustering and iyer.txt file
0
1
2
3
4
-1
Principle Component 2
Principle Component 1
20
15
10
5
0
−5
−10
−15
0
20
40
60
80

COMPARISON OF RAND AND JACCARD VALUES OF ALL THE ABOVE CLUSTERING ALGORITHMS

	K-Means	Hierarchical	DBScan	GMM	Spectral
Rand Index	Cho- 0.782906923675803 4	Cho- 0.24027490670890495	Cho- 0.4983086794276 356	Cho- 0.9526312217626 476	Cho- 0.392762221804 6122
	Iyer- 0.777768632454010 4	Iyer- 0.1941381800223728	Iyer- 0.6461208654303 02	Iyer- 0.7458630920090 239	Iyer- 0.745863092009 0239
Jaccard Index	Cho- 0.335019119279634 9	Cho- 0.22839497757358454	Cho- 0.2048718221465 8015	Cho- 0.8567226355372 66	Cho- 0.240080631614 31212
	Iyer-: 0.294813196728123 2	Iyer- 0.1584602101134175	Iyer- 0.2829190262836 6955	Iyer- 0.2752259317350 0637	Iyer- 0.275225931735 00637

PCA Data Visualization :

- PCA in conjunction with clustering is a powerful method for visualizing high dimensional data.The data we were given contained nearly 16 features (columns) and we wanted to get a broad overview of all of them but traditional methods of visualization were not doing well. For this reason we went for PCA which takes a ton features, project them onto a lower-dimensional space, reduce them down to just a few important principal ones, and visualize them.We found the optimal number of components which captured the greatest amount of variance in the data.
- We first fit the input matrix to the clustering algorithm and determine the best number of clusters.The ability to notice otherwise unseen patterns and to come up with a model to generalize those patterns onto observations is precisely why tools like PCA are essential
- When we perform PCA we collapse the variables (first normalize) and when we perform cluster analysis we collapse the rows.
- Doing PCA after clustering can validate the clustering algorithm so it helps to reveal clusters and plot the data on a lower dimension space.Applying PCA with only 1 component (after standardization) will give the direction where we will observe all 10 clusters.
- Analyzing explained variance ('elbow' approach), we will see that 1 component is enough to describe this data.PCA is used only to build some hypotheses regarding the data.
- Rand and Jaccard Index Values :
- Rand's Index is used in the classification problems in which the result of a classification scheme has to be compared to a correct classification. The most common performance measure for this problem calculates the fraction of correctly classified (respectively misclassified) elements to all
- For Rand, comparing two clusterings is just a natural extension of this problem which has a corresponding extension of the performance measure: instead of counting single elements we can count correctly classified pairs of elements.
- Once, we obtain the final clusters, we'll compute the Jaccard coefficient and randIndex of the clusters. Jaccard index and randIndex will provide similarity between the clusters as defined by the size of the intersection divided by the size of the union of the sample set.
- It helps to verify answers like :
- Is the algorithm sensitive to small perturbations, i.e. can small changes in the data (so-called "noise") entail large changes in the clustering?
- Is the algorithm sensitive to the order of the data, i.e. can another
- Order of the data result in a very different clustering?

- How similar are the solutions of two different algorithms?
- If an optimal solution is available: How close is the clustering solution to the optimal one?
- For examining these aspects, it would be desirable to have a ”measure” for the similarity between two clusterings. In a more general context, it can be necessary to combine different clusterings to a single one, i.e. calculating a ”mean value” of the clusterings.
- Rand and Jaccard Indices are more often used to compare Partitionings/clusterings than usual response characteristic statistics like sensitivity/specificity etc. But they can in some sense be extended to the idea of a true positive or a true negative.