CSE 601 : Data Mining and Bioinformatics

Project 2 Clustering Report

rohanhem | 50291746

nd34 | 50289820

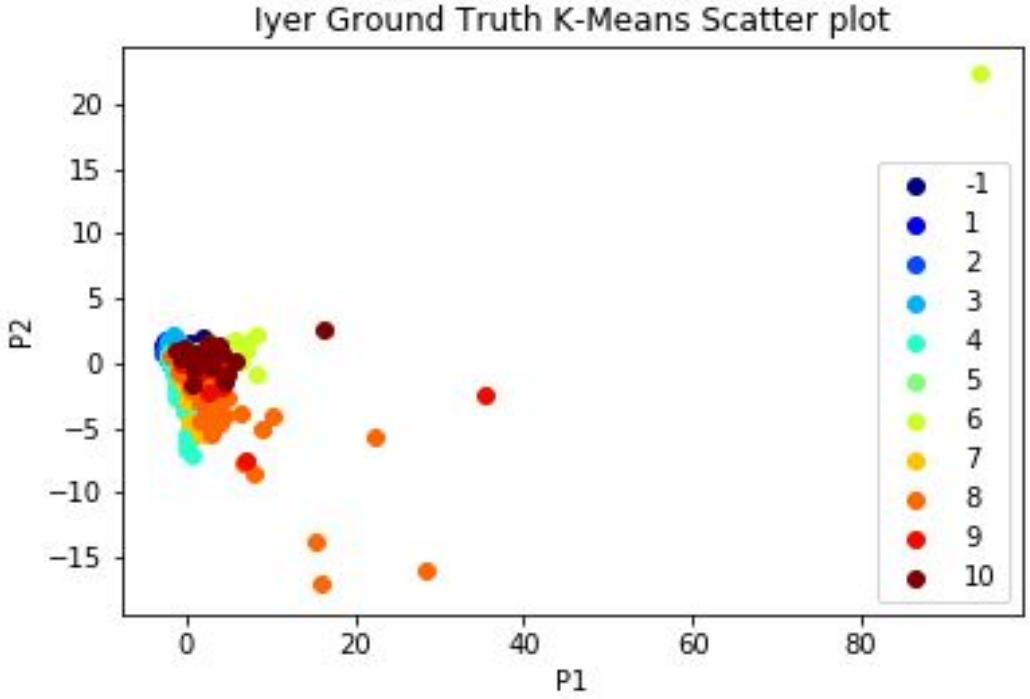
K-Means Algorithm: K-means clustering is an unsupervised learning algorithm used with unclassified data. It finds groups of high similarity within the data. It is an iterative process which assigns data to K clusters based on a similarity measure such as euclidean distance to the cluster centroids and updates the centroids for the next iteration.

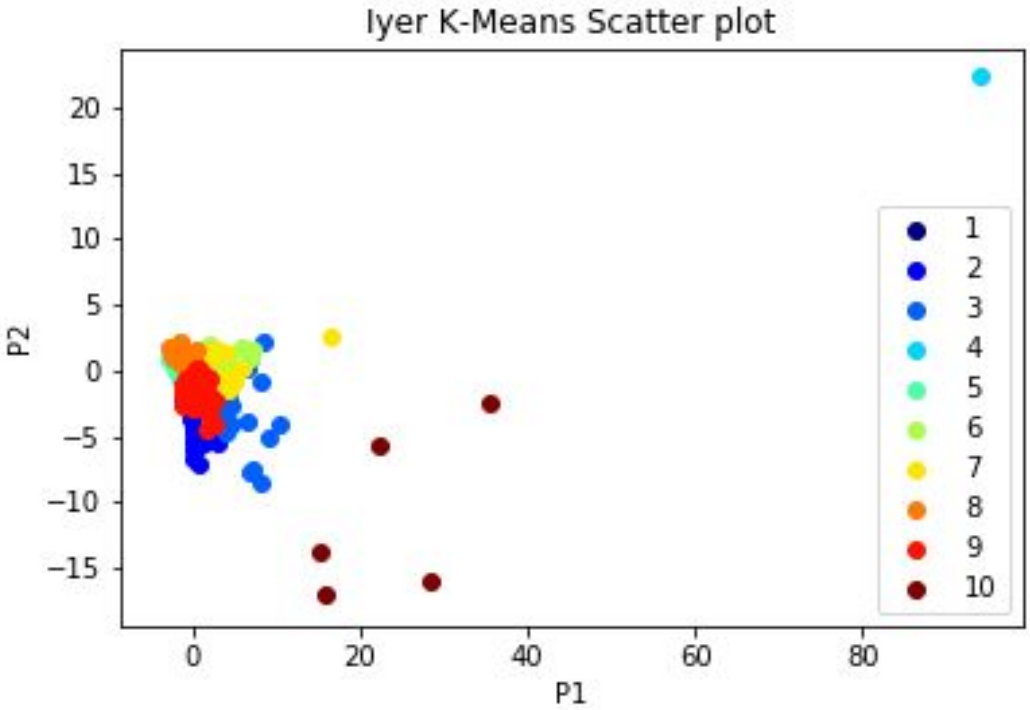
Implementation:

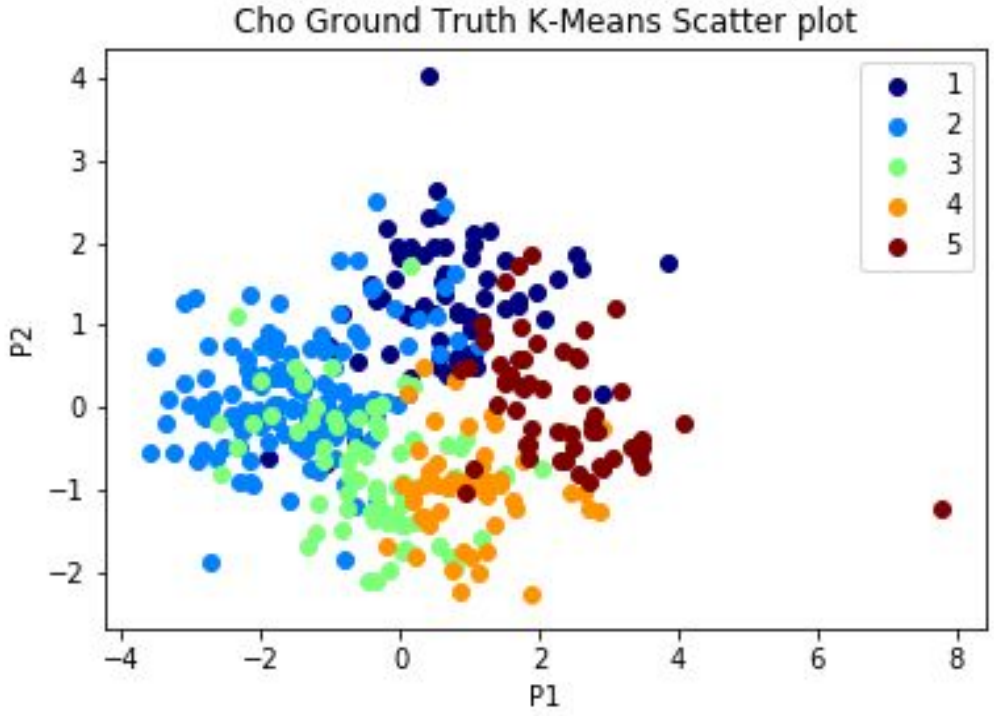
- First, we randomly initialize the initial cluster centroids.
- After initializing the centroids, we calculate the euclidean distance between every data point and the centroids to determine which centroid is the data point closest to. Once, we find the centroid, we assign that cluster to the data point.
- After all the data points have been assigned with their initial cluster labels, we update the centroids by calling the function updateCentroid(cluslist,k) where cluslist is the list containing initial cluster labels corresponding to each data point and k is the number of clusters.
- In the function updateCentroid, all the features of all data points within each cluster are averaged to find the new centroid for each cluster. The updated centroids are returned.
- Again, we calculate euclidean distances and find the new cluslist. If this new cluslist is equal to the previous iteration's cluslist, we stop and return the new cluslist and the centroids. If not, we again update our centroids and continue with the next iteration until max iterations are reached or the assigned cluster labels do not change.

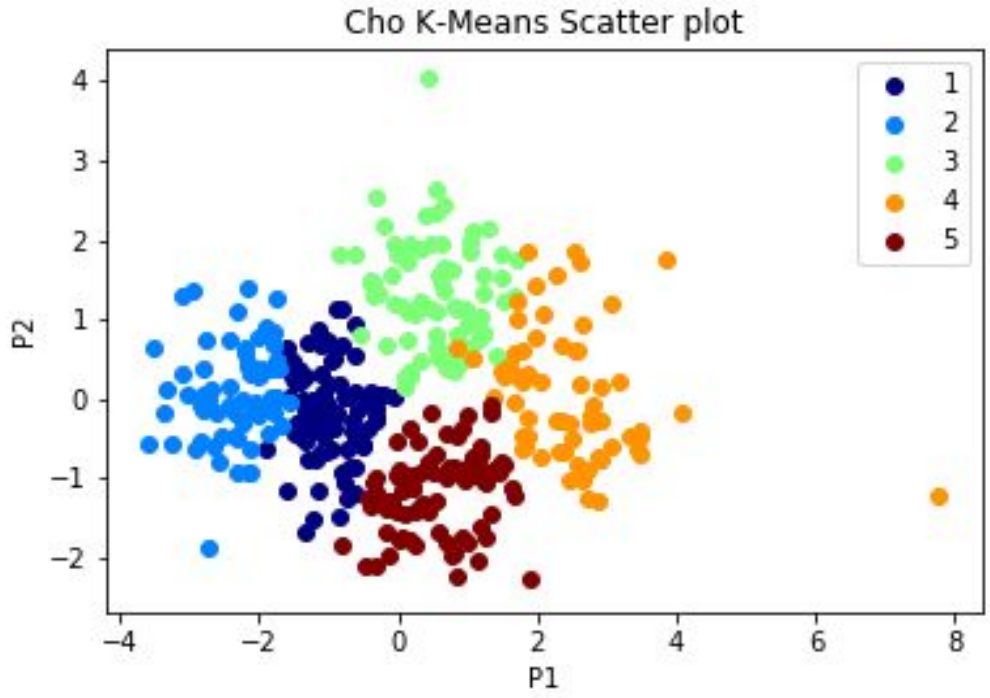
Initial center IDs for cho.txt: [147, 90, 24, 277, 203]
Rand index: 0.7820746865687 67
Jaccard index: 0.33153538930291926

Initial center IDs for iyer.txt: [34, 224, 475, 388, 113, 220, 484, 154, 195, 452]
Rand index: 0.8298545768812035
Jaccard index: 0.39339211161649174


Iyer Ground Truth K-Means Scatter plot
P2
P1
20
15
10
5
0
−5
−10
−15
0
20
40
60
80
-1
1
2
3
4
5
6
7
8
9
10


Iyer K-Means Scatter plot
P2
P1
20
15
10
5
0
−5
−10
−15
0
20
40
60
80
1
2
3
4
5
6
7
8
9
10


Cho Ground Truth K-Means Scatter plot
P1
P2
1
2
3
4
5


Cho K-Means Scatter plot
P1
P2
1
2
3
4
5

Conclusion:

- Initial centroids chosen at random are preferable, if we observe the jaccard similarity scores. Depending on the implementation, we may get new centroids though that is not guaranteed.
- K means output gives us a simple and intuitive classification that is efficient and easy to implement for the chosen k value.
- If there is further variation with the subgroups that we get in k-means, we can then reapply clustering as required. Therefore, k-means is often used as the first step in a big clustering problem.
- K means will generally give more preference to larger clusters to minimize variation within the clusters, thus potentially limiting the use in some specific cases.

Pros:

- As long as the value of k is not unreasonably large for the data, the computation time is remarkably small.
- The clustering achieved is compact and intuitive.
- This algorithm can often be used as a first step for bigger and more complex algorithms to make the computation easier and more efficient for later steps.

Cons:

- There can be a large variance in output for different initial centroids.
- Initial ground truth values are often instrumental in predicting K.
- Drawing from the first point, we can sometimes get empty clusters that provide a faulty clustering output.
- A healthy experimentation can be required in some cases to arrive at an acceptable output which means some pre and post processing.
- Application can be limited in terms of highly complex clusters with varying parameters.
- K means can only handle spherical clusters.

Hierarchical Agglomerative Clustering Algorithm with Min Approach:

It consists of a bottom up approach, we build a hierarchy of clusters represented using a dendogram. At the beginning every point is a single cluster. Then each cluster is merged with their nearest neighbour iteratively till we get a single cluster. The merging policy is greedy because we only merge a pair of clusters that have the smallest inter-cluster distance.

Implementation:

- Initially we maintain a dist_matrix and calculate the euclidean distance between every pair of clusters (data points) and fill the lower triangle of the dist_matrix of dimension (nxn). We also maintain a datalist that will contain all the clusters at the end of each iteration. Initially datalist contains all the data points as single clusters.
- Then we find out the minimum value in the matrix by traversing the matrix and record the corresponding index values i and j which correspond to the cluster indexes. This minimum value corresponds to the minimum inter cluster distance so that the two clusters can be merged into a larger cluster of size (size(cluster1)+size(cluster2)).
- Then we add the two clusters in a list to form a larger cluster and append this list to datalist at the end of it after dropping the two single clusters which we merged.
- This will ensure that the length of datalist decreases by 1 with each iteration.
- After the datalist is updated, we again maintain a dist_matrix with rows and columns equal to the length of the datalist and calculate the distance between each pair of clusters by selecting the closest pair of points (Single Linkage) within the two clusters. In this way, we fill the lower triangle of the dist_matrix.
- We repeat the process of finding the minimum value in the matrix, merging two clusters and updating the datalist until the length of the datalist is 1 which means all the data points belong to one single cluster.
- We have defined two functions mindist(l1,l2) to calculate the distance corresponding to the closest pair of points between two clusters and findIndex(d) to find the original index of the data point d in the input raw data.
- To find k=5 clusters, we can do while(k>5). Now the datalist contains 5 clusters and using findIndex(d), we can appropriately assign cluster numbers to the data points.

For cho.txt

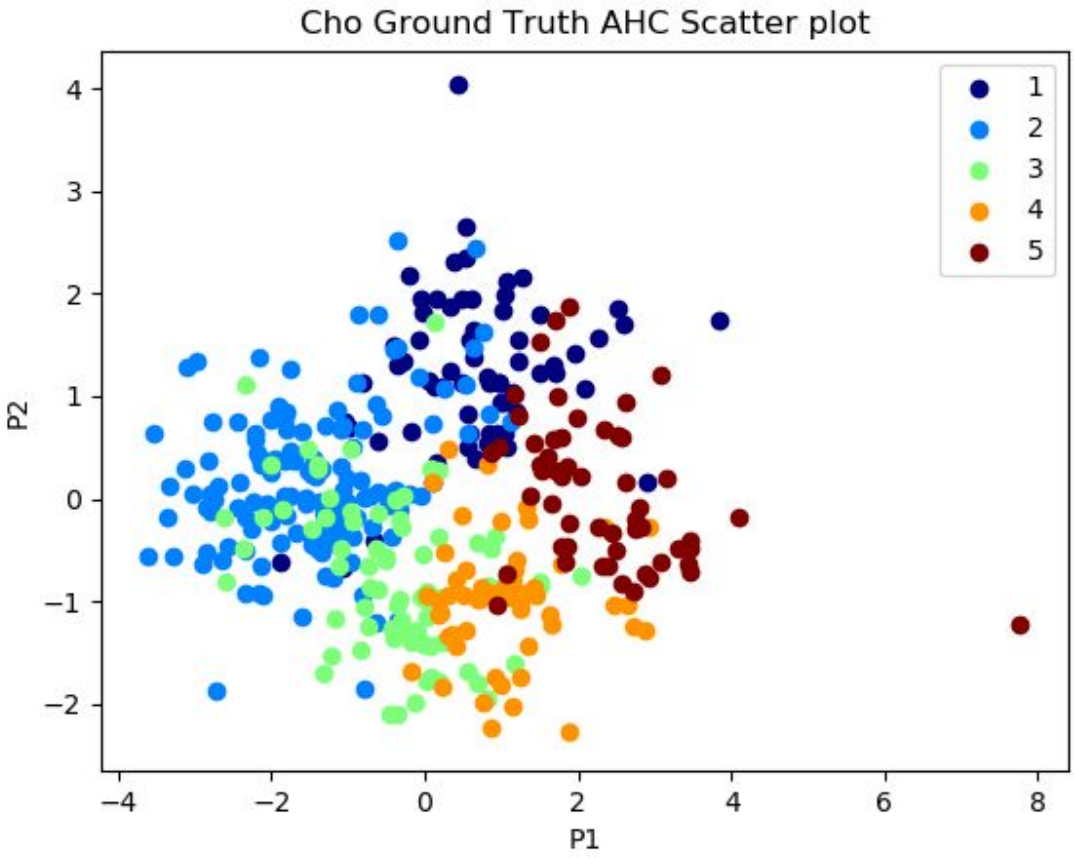
Rand index: 0.2402749067089049 5

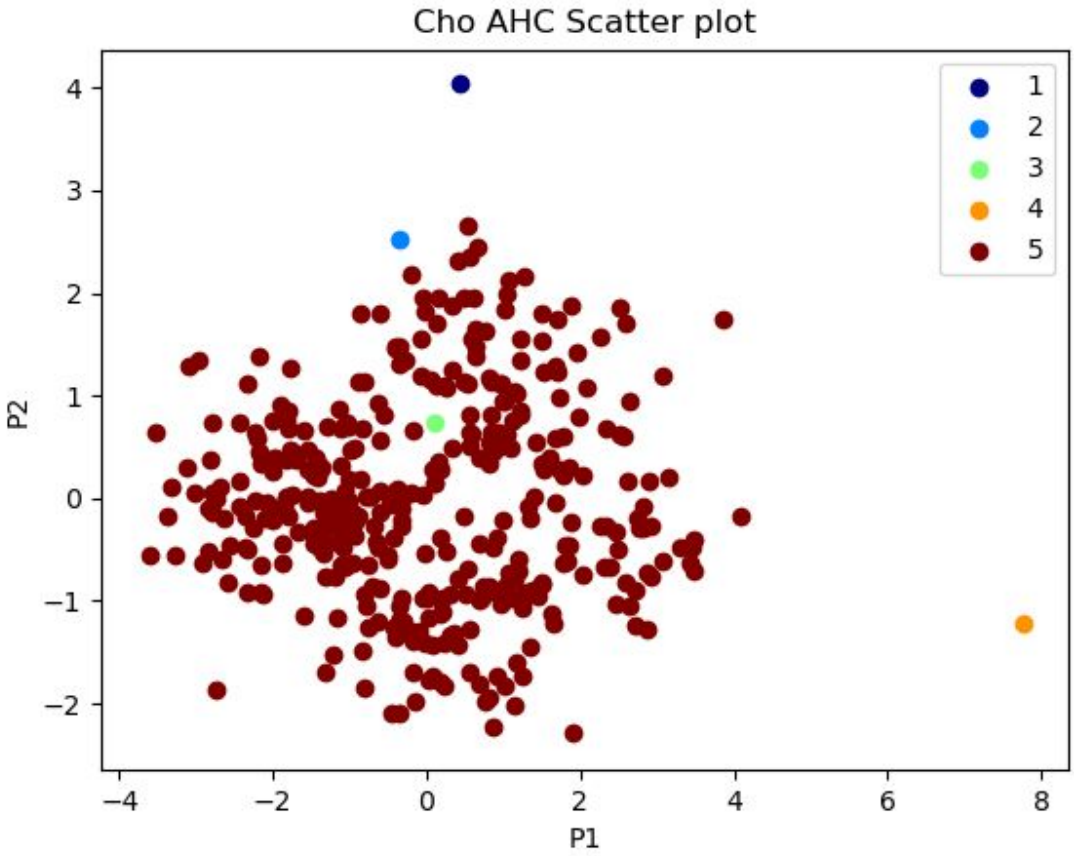
Jaccard index: 0.22839497757358454

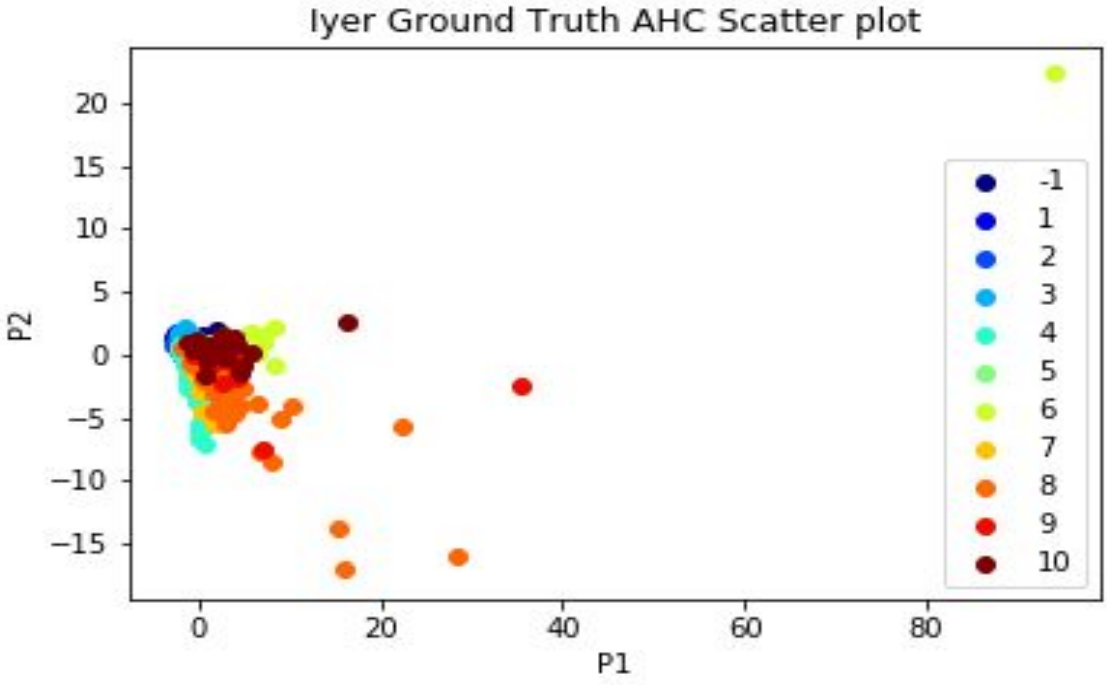
For iyer.txt

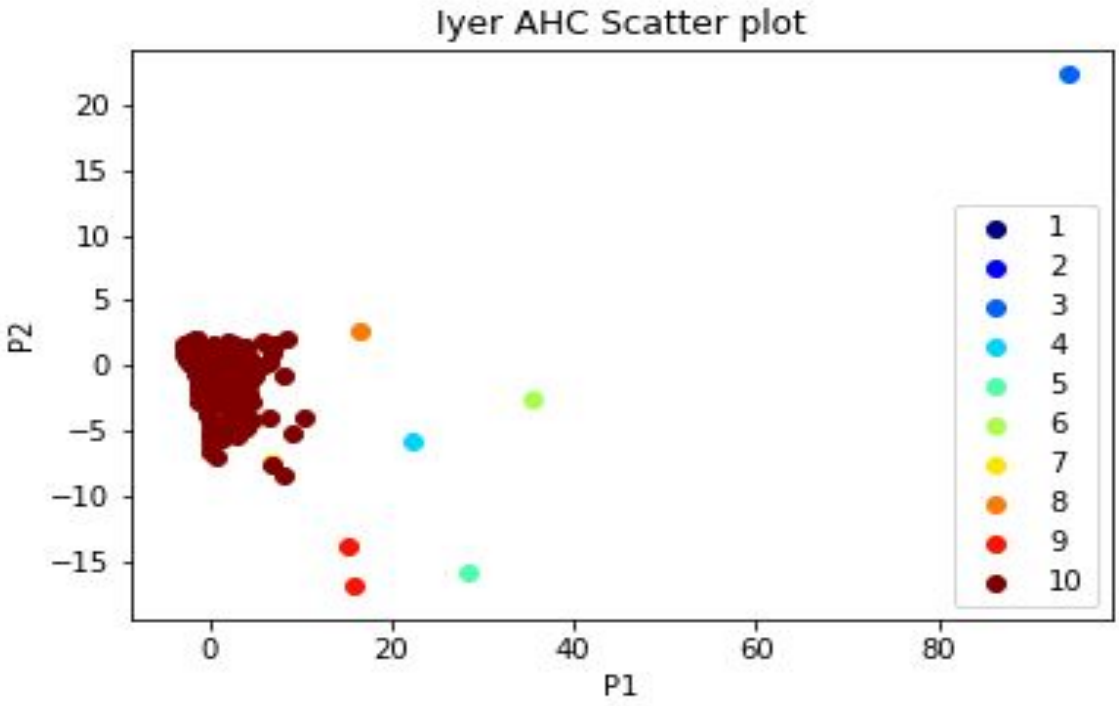
Rand index: 0.1882868355974245

Jaccard index: 0.15824309696642858




Cho AHC Scatter plot
P1
P2
1
2
3
4
5


Iyer Ground Truth AHC Scatter plot
P1
P2
-1
1
2
3
4
5
6
7
8
9
10



Conclusion:

- As required, we have implemented the Min approach where the minimum distance of the points is calculated and stored.
- The result of this approach is such that the linking is localized, i.e. remote points are not taken into consideration as the distance makes them less important.
- Outliers do not exist in this as at the end we get a single cluster though a distant points can create some problems in the penultimate steps.
- We can see the limitation of this HAC Min based approach as it requires a fairly noise-free data with larger inter-cluster distance.

Pros:

- We get a degree of separation and distances that could be used for further reading or just a better understanding of similarities and dissimilarities of the dataset.
- Specifying a number of clusters is not required and all data points covered.

Cons:

- As all data points are covered in the last step, outliers are identified.

- A Hierarchical approach is inherently costly in terms of time complexity, this effectively limiting its use in terms of dataset complexity and size.
- A Min-approach specific problem is that points further apart get negatively impacted.
- If there is a lot of noise, the effectiveness can be drastically reduced.
- A single point cluster forming is possible in datasets like the one provided for this demo, thus inserting inconsistencies.

DENSITY-BASED SPATIAL CLUSTERING AND APPLICATION WITH NOISE (DBSCAN) Algorithm:

It is a data clustering algorithm. It considers the basic idea that clusters are dense regions in the data space, separated by regions of lower object density. It groups together points that are closely packed together and marks points as outliers that lie in low-density regions.

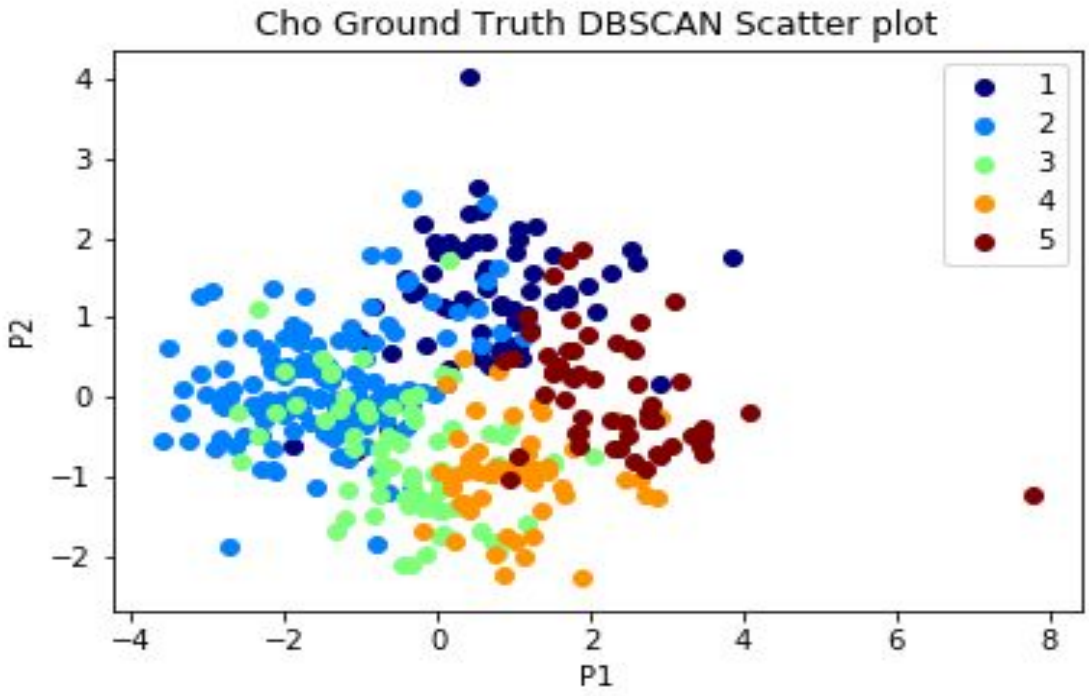
Implementation using Pseudocode:

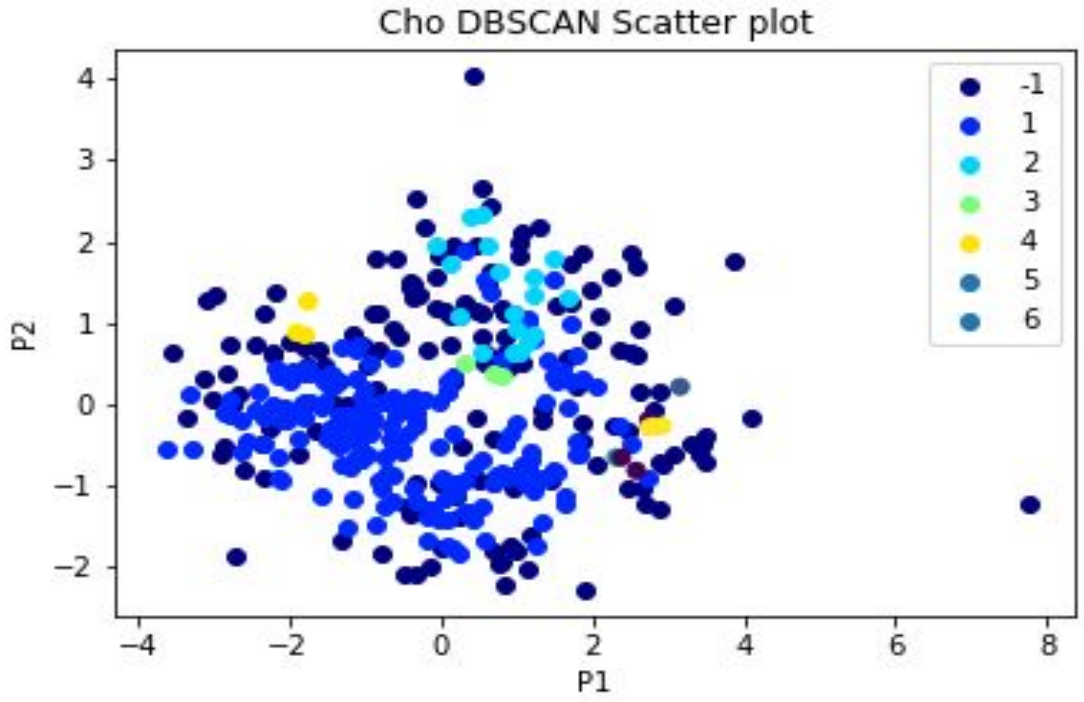
- DBSCAN is used to assign labels to each point. While labelling the cluster numbers to the data points, it also considers the neighbors that are density reachable.

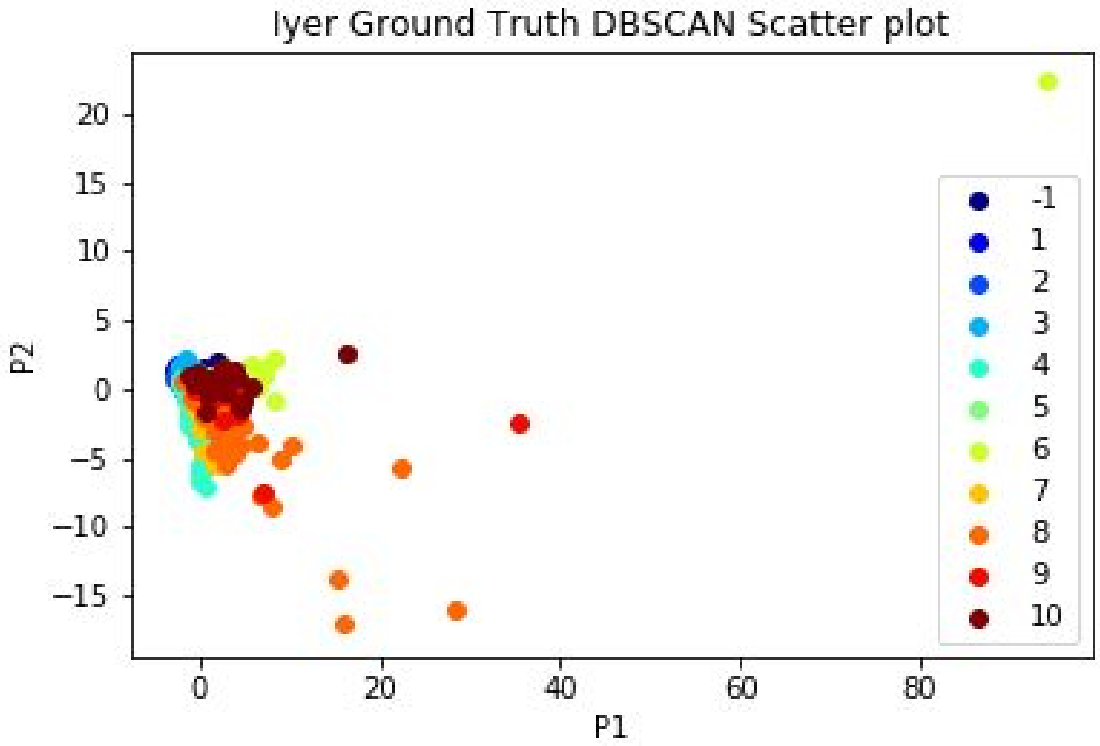
- If the neighboring points are density reachable, then expandCluster function is called and it adds the neighbors of these data points to the same cluster.
- regionQuery function returns the neighbors of a data point that are at or within epsilon distance from the point and satisfy the minPts condition.
- We mark each unvisited point as visited and calculate its neighbor points by calling regionQuery. If the number of neighbor points are less than MinPts, the point is marked as noise, or else the cluster number is incremented by 1 and we call expandCluster function which visits and adds all the density reachable points to the current cluster and also the points that have not been assigned to any cluster yet.
- This is how all the data points are labelled. Outliers are represented by the label -1.

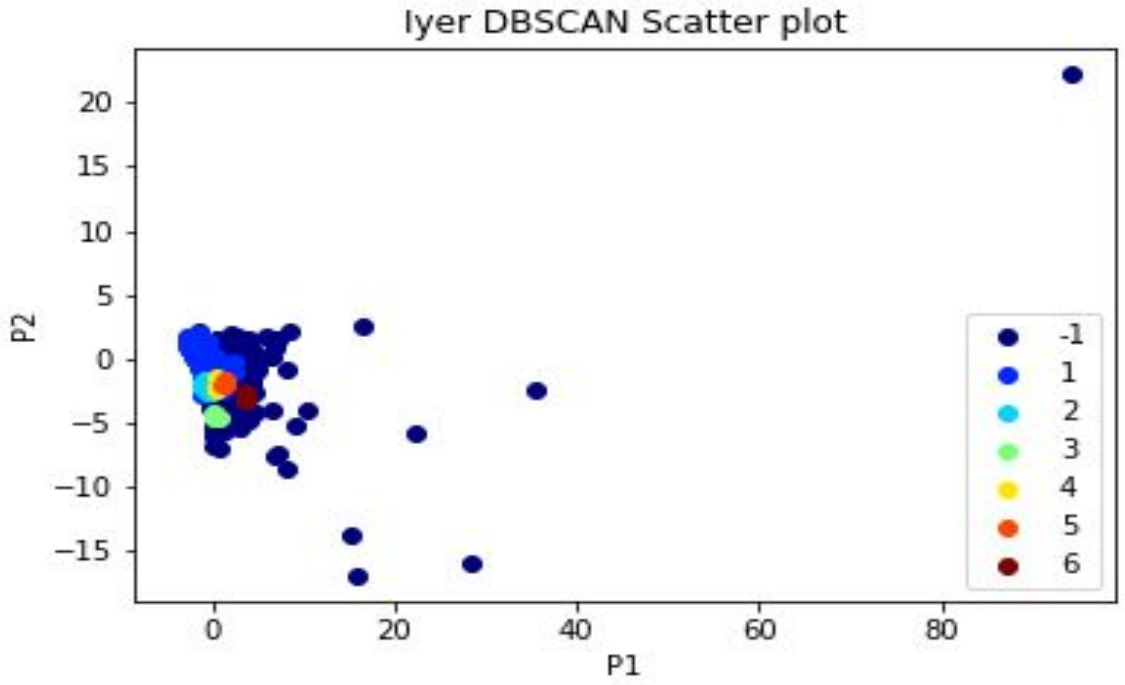
cho.txt:
Eps: 1.1
MinPts: 3
Rand index: 0.5664715831297484
Jaccard index: 0.2037425112793077

iyer.txt:
Eps: 1.1
MinPts: 3
Rand index: 0.5625184725147686
Jaccard index: 0.2385572609055213




Cho DBSCAN Scatter plot
P1
P2
-1
1
2
3
4
5
6


Iyer Ground Truth DBSCAN Scatter plot
P1
P2
-1
1
2
3
4
5
6
7
8
9
10



Conclusion:

- First thing we have to note about the DBSCAN algorithm is that we have to do a trial and error to get the suitable eps and minPoints values. In this, we have usd 1.1 and 3 for the respective variables that we found best for the result for the cho dataset and 1.1 and 3 for the iyer dataset.
- As can be seen from the graph, all points were covered with time complexity of $O(n^2)$.
- We see that no noise points are left out as unidentified. Everything is covered and correctly labeled, though depending on number of datapoints, complexity can increase.

Pros:

- DBSCAN can cover all points including noise with some flexibility as per the selected eps and minPoints.
- Specifying a number of clusters is not required, thus making the clustering easier.

- Cluster locating is independent of shapes, thus massively increasing its utility for a complex dataset.

Cons:

- Some points that are equidistant from more than one cluster can create problems.
- Since all the points have to be reached, the $O(n^2)$ time complexity can be bad for large datasets.
- Some tinkering with eps and minPoints is required to find the right values for classification. It is sensitive to parameters.

Gaussian Mixture Model (GMM) Algorithm:

It is an unsupervised learning algorithm which can be used for clustering. GMMs assume that there are a certain number of Gaussian distributions, and each of these distributions represent a cluster. Hence, it tends to group the data points belonging to a single distribution together. They are probabilistic models and use the soft clustering approach for distributing points to different clusters.

Implementation:

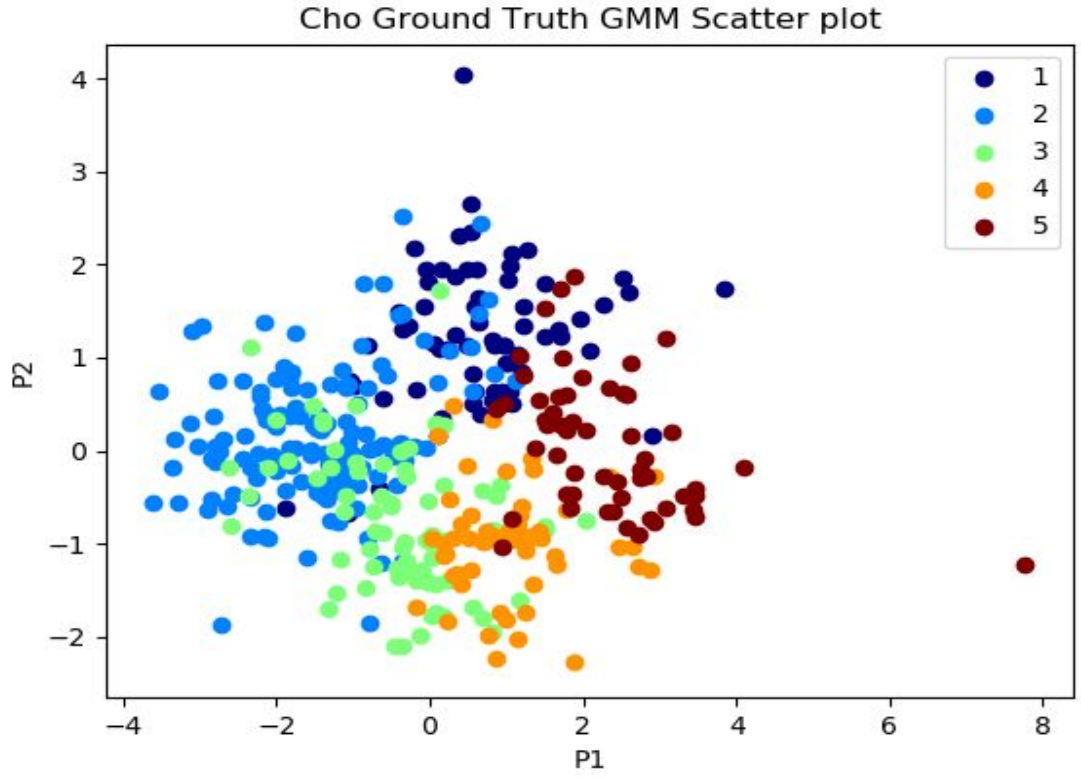
- Expectation Maximization algorithm is used.
- We define the initial means, covariance matrix and weights.
- In the E-Step, we calculate the responsibilities, that is the probability that a data point belongs to a cluster k based on the initial parameters using multivariate_normal(mu[k],BigSigma[k]).pdf(data)
- After the responsibilities are calculated, we need to update all the parameters in the M-Step. Means, Covariance matrix and weights are updated appropriately.
- We are adding a smoothing value of the order of e-6 to the diagonals of the covariance matrix to prevent singularity in the M-Step as well as during initialization.
- Then we calculate the log-likelihood. We keep iterating over E-Step and M-Step until log-likelihood has converged, or it has stopped increasing beyond a certain convergence threshold that is given as a parameter.
- When we are done iterating based on maximum iterations or convergence threshold, we assign clusters to the datapoints based on the largest responsibilities, or probabilities that the data points belongs to k clusters.

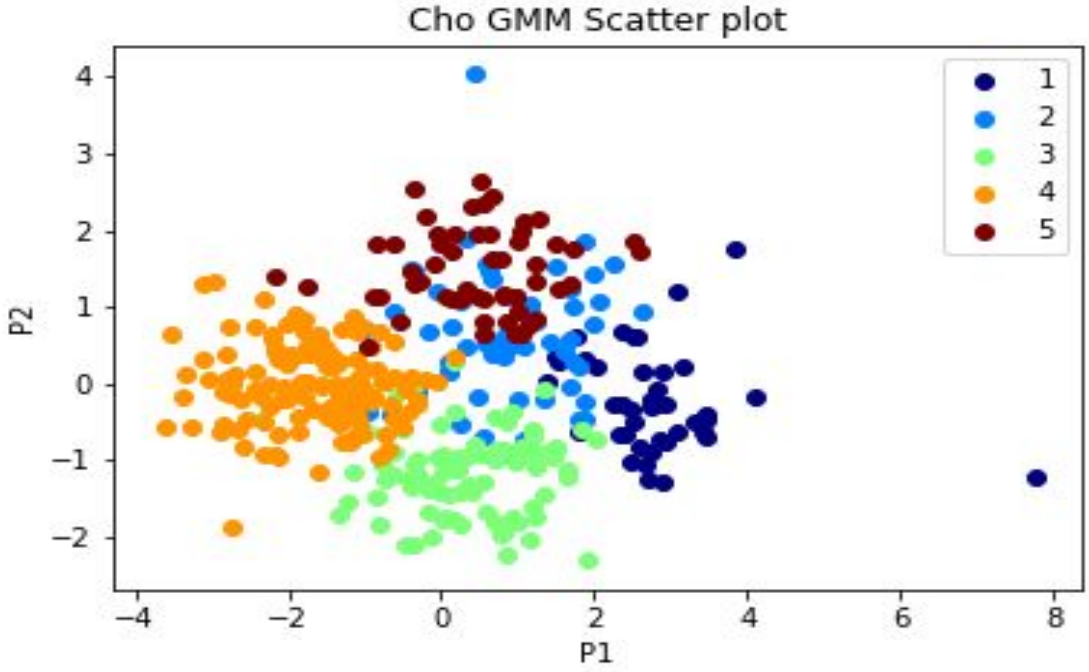
Initially, means are the centroids calculated by applying K-Means, Covariance matrix is calculated as BigSigma = np.full(shape, np.cov(data, rowvar = False)), where shape is

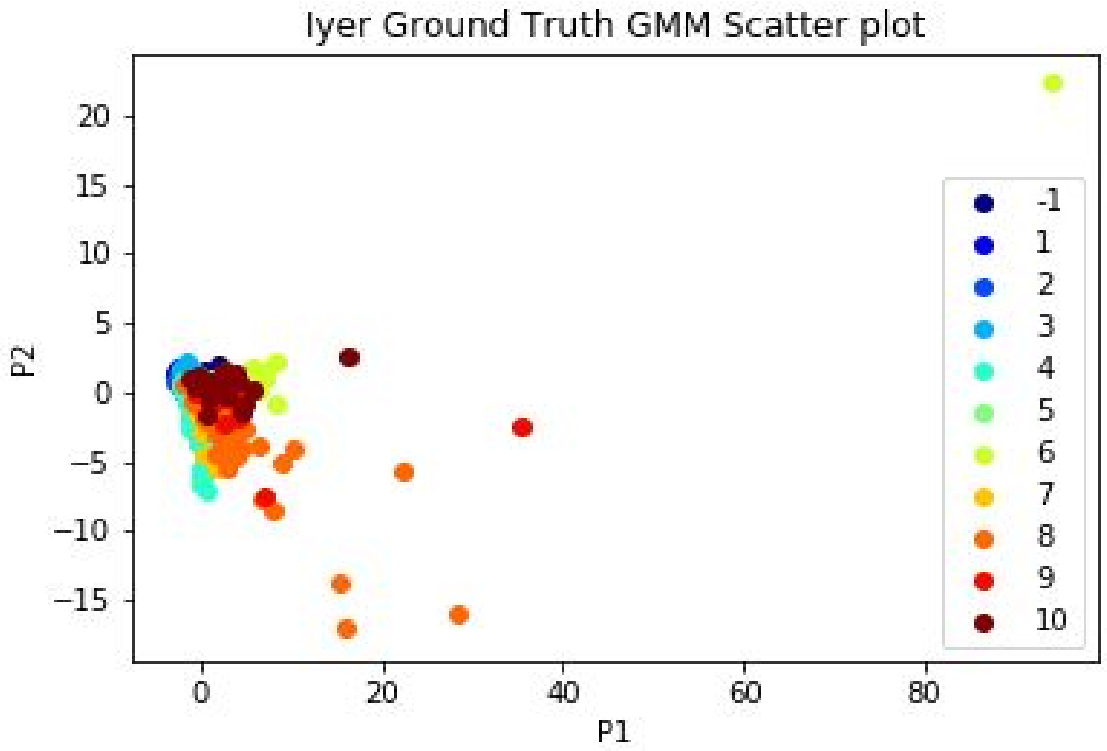
k x cols x cols (features)

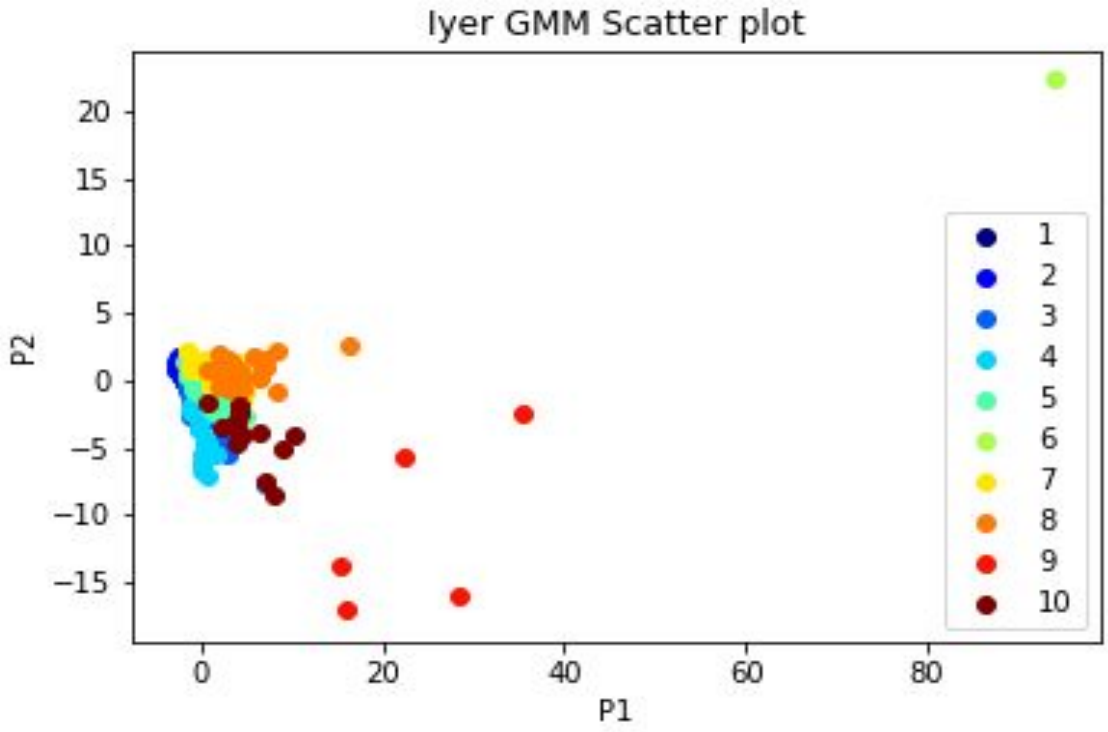
Cho.txt:
Number of clusters k = 5
Conv_threshold = 1e-4
smooth = 1e-6
MaxIter = 100
Rand index: 0.8016054122258316
Jaccard index: 0.4039120790481952

iyer.txt:
Number of clusters k = 10
Conv_threshold = 1e-4
smooth = 1e-6
MaxIter = 100
Rand index: 0.7760027535738471
Jaccard index: 0.3625009316737119


Cho Ground Truth GMM Scatter plot
P2
P1
1
2
3
4
5


Cho GMM Scatter plot
P2
P1
1
2
3
4
5


Iyer Ground Truth GMM Scatter plot
P2
P1
20
15
10
5
0
−5
−10
−15
0
20
40
60
80
-1
1
2
3
4
5
6
7
8
9
10


Iyer GMM Scatter plot
P2
P1
20
15
10
5
0
−5
−10
−15
0
20
40
60
80
1
2
3
4
5
6
7
8
9
10

Conclusion:

- As can be seen from the graphs, we get a desirable probabilistic distribution of points to the clusters for K = 5 and K = 10 (cho and iyer respectively).
- We can see that the outliers are correctly classified in both the cases.

:

Pros:

- The GMM gives us a probabilistic distribution of the data points with respect to clusters, thus giving us the relationship between every data point and cluster.
- Works across a wide variety of datasets and complex scenarios
- Excellent for visualization of dataset analysis.
- Works well when we have missing data points or for an incomplete dataset.

Cons:

- If a Gaussian collapses on a single data point, it can complicate the likelihood function (singularity issues).
- Computing the gaussians can sometimes be expensive.

Spectral Clustering Algorithm:

Spectral Clustering follows the approach of connectivity in data clustering. In spectral clustering, the data points are treated as graph nodes. Thus, clustering is treated as a graph partitioning problem. The nodes are then mapped to a low-dimensional space that can be easily clustered.. Also, there is no assumption is made about the shape of the clusters.

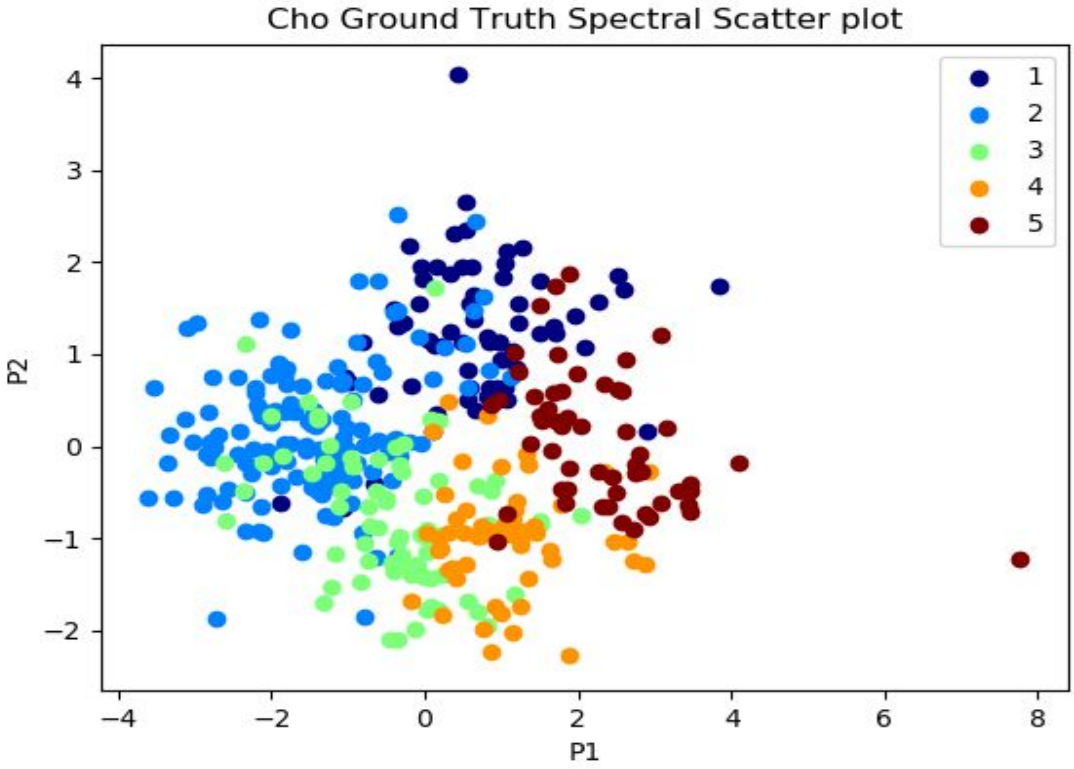
Implementation:

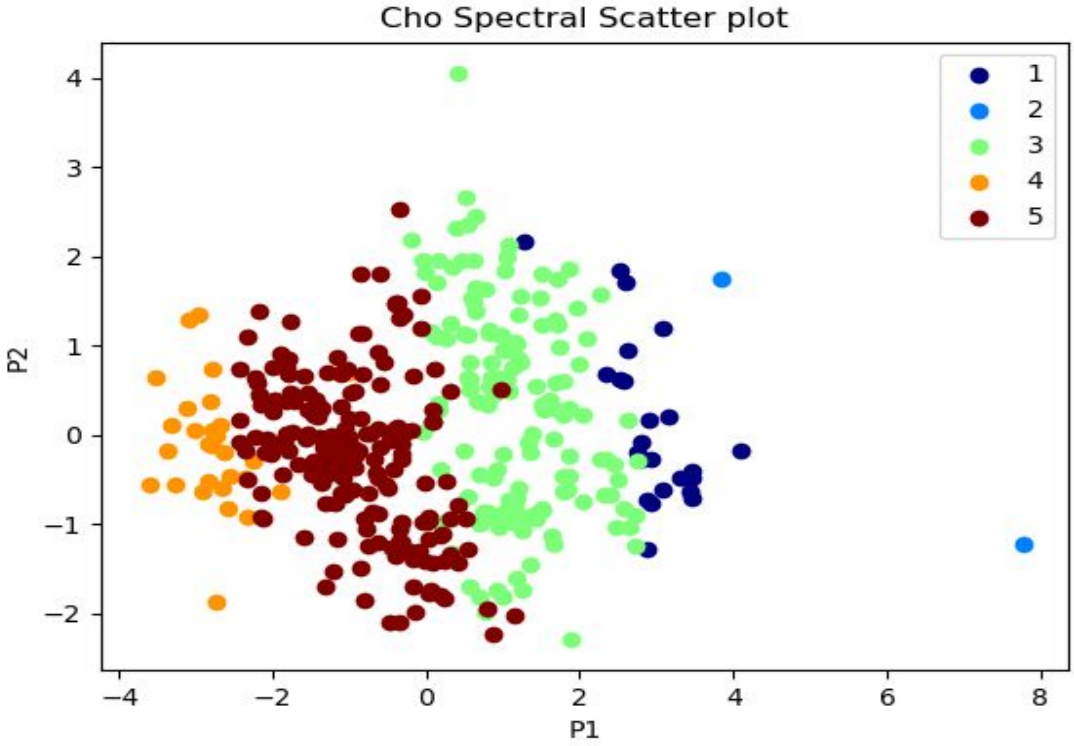
- Initially we calculate a similarity matrix W using the Gaussian kernel and then degree matrix D.
- Then we calculate Laplacian Matrix as L = D - W.

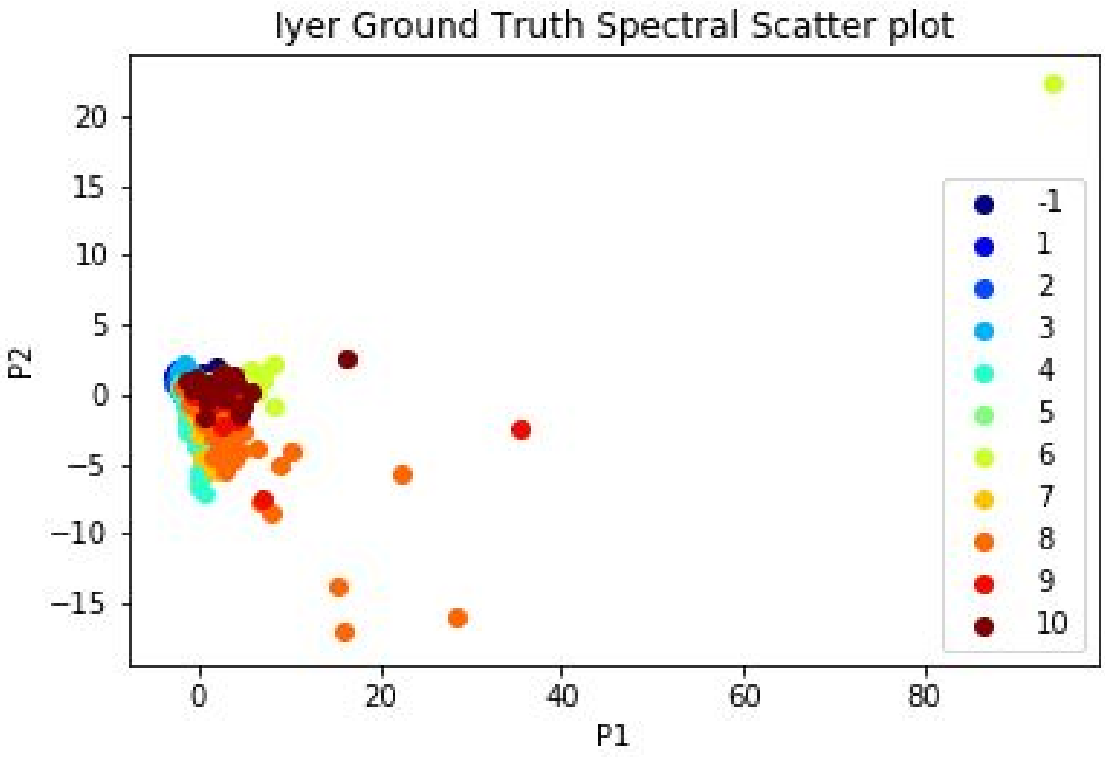
- Then we calculate the lower-dimensional embedded space by calculating eigenvalues and eigenvectors of L using np.linalg.eig.
- We sort the eigenvalues and select a k such that k corresponds to the largest eigengap which is the absolute difference between consecutive eigenvalues.
- Then we transform the original m x n data into m x k space and perform K-Means clustering on it to obtain the clusters.

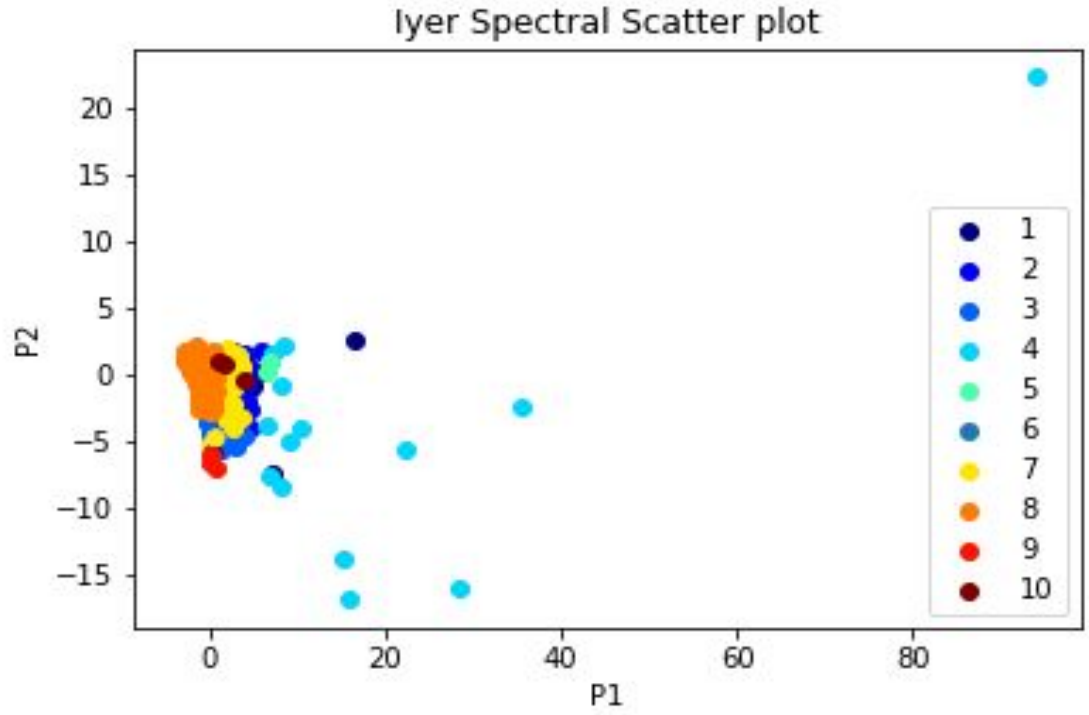
cho.txt:
Sigma=8.0
Rand index: 0.6570511960052619
Jaccard index: 0.2840208496805291

iyer.txt:
Sigma=6.8
Rand index: 0.5701057656693691
Jaccard index: 0.2366520736867979


Cho Ground Truth Spectral Scatter plot
1
2
3
4
5
P2
P1
4
3
2
1
0
−1
−2
−4
−2
0
2
4
6
8


Cho Spectral Scatter plot
1
2
3
4
5
P2
P1
4
3
2
1
0
−1
−2
−4
−2
0
2
4
6
8


Iyer Ground Truth Spectral Scatter plot
P1
P2
-1
1
2
3
4
5
6
7
8
9
10


Iyer Spectral Scatter plot
P1
P2
1
2
3
4
5
6
7
8
9
10

Conclusion:

- For K = 5 and 10 (cho and iyer respectively), we see that the points are mapped according to the eigenvectors we get in the laplacian matrix.
- We also see a lack of local minima concentration problem as well as an elegant and simple clustering output.

Pros:

- No need to make a strong assumption about the statistics of the clusters.
- Easy to implement, elegant and mathematically sound.
- Local minima can be avoided by finding a solution to the minicut problem.

Cons:

- Problematic when there is a lot of noise in the dataset.
- Can be very expensive computationally for a large dataset.

Final summary:

Since the Jaccard coefficient measures similarity between finite sample sets and the Rand index measures similarity between two data clusterings, GMM and K-Means have better results than the other approaches.