Project 2: Clustering Algorithms

Abhav Luthra(50288904) - Krishna Sehgal(50291124)
Tanmay Pradeep Singh(50291086)

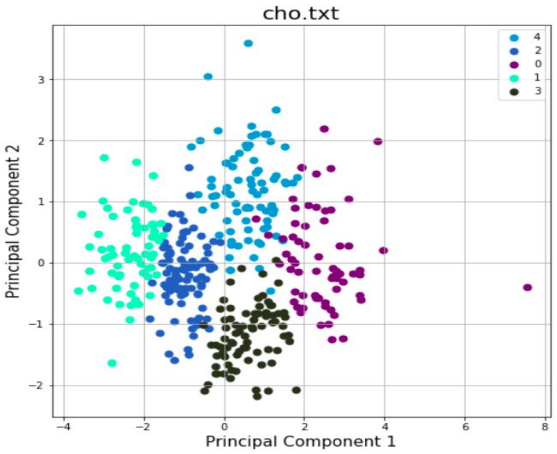
1. K-means Clustering

K-means algorithm, cluster's the data based on the euclidean distance of points from K cluster centroids.

The Algorithm for K-means clustering is as follows:

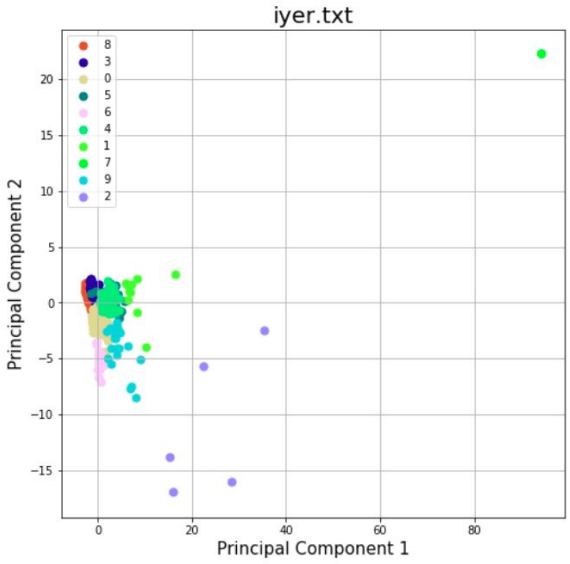
1. The features are extracted from the txt file and stored in a list.
2. The type of centroid initialization (i.e Random or Specific), number of centroids in K-means and maximum number of iterations are taken as input from the user. If the type of initialization is specific the user can specify the centroids.
3. Each point in the dataset is assigned to a cluster which has the minimum euclidean distance to the cluster's centroid.
4. The points in each cluster are averaged to get the new cluster centroid.
5. Error is calculated using the old cluster centroid and the newly calculated cluster centroids.
6. The steps 3 to 5 are repeated until the error becomes 0 or the maximum specified number of iterations is reached.

K-means clustering on cho.txt and iyer.txt with values as follows:
K = 4 , Centroid initialization=random , Maximum number of iterations=10



Jaccard Index: 0.33 | Rand Index: 0.78

Figure 1: Visualisation of K-means Clustering using PCA for Cho Data-set



Jaccard Index: 0.35 | Rand Index: 0.77

Figure 2: Visualisation of K-means Clustering using PCA for Iyer Data-set

Result Evaluation

1. The K-means clustering for cho.txt is very well clustered as shown by high jaccard and rand values.
2. The image of the plot shows that even the outliers are mapped to the clusters whose centroids are closest to them.

Pros and Cons for K- means Clustering

Pros

1. K-means is easy to implement.
2. The time complexity of K-means is O(m*k*n) where m is the maximum number of iterations, k is the number of clusters and n is the number of points in the dataset.

Cons

1. The initial number of centroids has to be specified therefore it is difficult to predict the optimal value of k.
2. K-means performance is not good for clusters of different size,shape or density.
3. K-means clustering may result in empty clusters.
4. The initial initialization matters. Different initial centroids may result in different clusters.

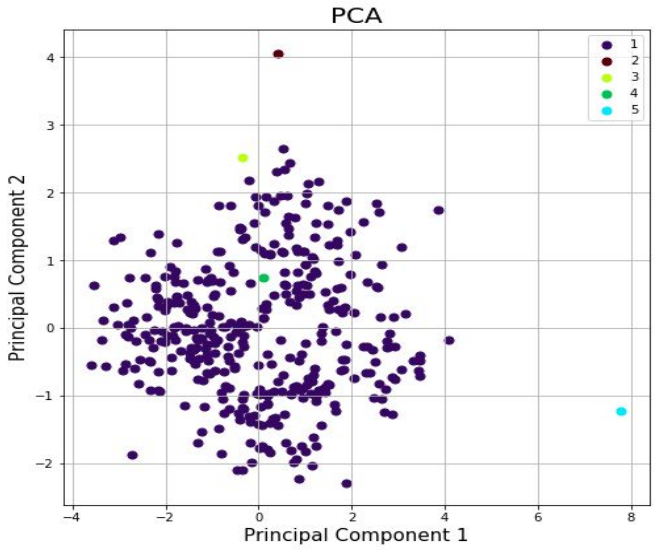
2. Hierarchical Agglomerative Clustering

Agglomerative Clustering is a bottom-up approach starting with each point in a separate cluster, repeatedly joining the most similar pair of clusters and updating the similarity of the new cluster to other clusters until there is only one cluster

We use **single linkage** i.e. we consider the distance of the closest pair of data objects belonging to different clusters. In single-link clustering or single-linkage clustering, the similarity of two clusters is the similarity of their most similar members. This single-link merge criterion is minimum distance.

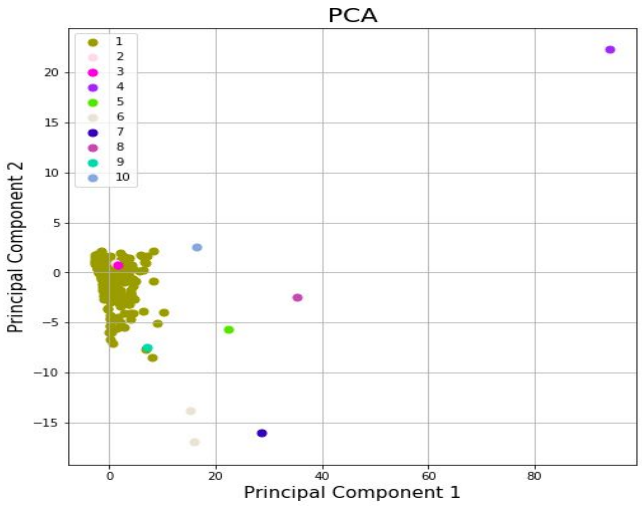
The Algorithm for HAC clustering is as follows:

1. The features are extracted from the txt file and stored in a numpy array.
2. The number of clusters in HACs are taken as input from the user
3. Each data point is considered as an individual cluster.
4. We calculate the proximity of individual points and insert them into a heap.
5. Smallest euclidean distance points are taken from heap are merged together and formed as a single cluster.
6. Finally, all the clusters are merged together till the no of cluster reaches the user defined cluster size.



Jaccard Index: 0.22839497757358454 | Rand Index: 0.24027490670890495

Figure 3: Visualisation of Agglomerative Clustering using PCA for Cho Data-set



Jaccard Index: 0.15824309696642858 | Rand Index: 0.1882868355974245

Figure 4: Visualisation of Agglomerative Clustering using PCA for Iyer Data-set

Pros and Cons for Hierarchical Clustering

Pros

- No prior information about the number of clusters required as they merge into one and can be stopped if needed
- It captures hierarchical relations between clusters
- We can create smaller clusters depending on where we cut the dendrograms which is helpful for discovery to similarity of clusters
- Meaningful taxonomies might be produced

Cons

- Algorithm is Greedy, therefore less accurate and computationally expensive
- A single-link clustering can produce a chaining effect i.e. algorithm merge points with minimum distance irrespective of other points and final shape of the clusters
- Algorithm is sensitive to noise and outliers
- hierarchical clustering has high in time complexity
- There is no objective function that is directly minimized
- Steps are final in algorithm as merged clusters can’t be undone
- Sensitive to outliers
- Algorithm have difficulty in handling different sized clusters and convex shapes

Result Evaluation

1. Single-Linkage is not a very effective method in HAC as it fails to separate clusters properly if there is noise between clusters. Other methods like ‘ward’ do a much better job.
2. The jaccard and rand coefficients is extremely low as we are unable to differentiate between clusters.
3. No of points - cluster size -1 points come in 1 cluster and other clusters get 1 point each

3. Density-based Clustering

- DBSCAN is density based spatial clustering of applications with noise.
- DBSCAN is used to cluster the points based on the density of the regions. Here density is the number of points present in a specific region.
- Given epsilon which is used as a radius and min points which is a threshold of minimum number of points in a region points are classified into 3 groups as follows:
    1. Core points:-A point is a core point if it has more than a specified number of points (Min Points) within Epsilon—These are points that are at the interior of a cluster.
    2. Border points:-A border point has fewer than Min Points within Epsilon, but is in the neighborhood of a core point.
    3. Noise points:-A noise point is any point that is not a core point nor a border point.

The algorithm for DBSCAN is as follows:

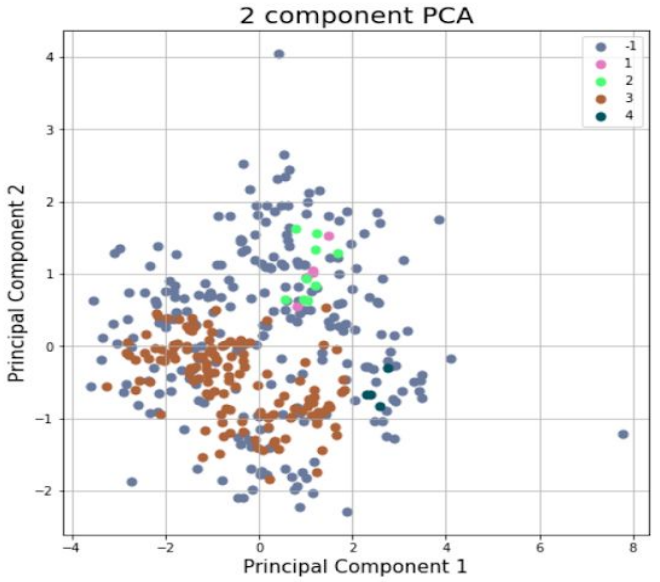
1. The features are extracted from the txt file and stored in a list.
2. Epsilon and min points are taken as input from the user and passed to the dbscan function.
3. The cluster_id is first initialized to 0.
4. For each unvisited point we find the points in its epsilon neighbourhood using Get_Neighbours function which returns all the neighbouring points in epsilon distance from the initial point. The distance is calculated using the euclidean distance formula.
5. If the number of neighbours is less then the min_points is is assigned a cluster_i =-1 and next point is evaluated.
6. If the number of neighbours is more then the min_points it is assigned a cluster_id which increments every time a new cluster is initialized.
7. The Grow_Cluster function is called using the original point to assign the same cluster_id as the original point to all the unvisited points in its neighbourhood. The Get_Neighbour function is then called for the neighbouring points and the neighbours are then appended to the neighbour list of the original point.

8. Every points which was classified as noise (i.e cluster_id=-1) and is in the neighbourhood of the original point is classified as border point to that cluster and assigned the same cluster_id.
9. Steps 4 to 8 are repeated until all the points are visited.

Parameters:

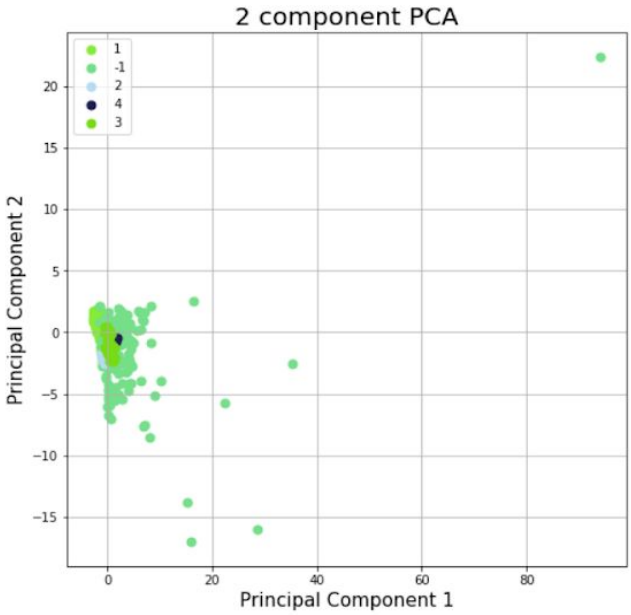
Epsilon = 1

Min_points = 4



Jaccard Index: 0.203663430711| Rand Index: 0.523222099922

Figure 5: Visualisation of Density-based Clustering using PCA for Cho Data-set



Jaccard Index: 0.650737591147 | Rand Index: 0.283738059616

Figure 6: Visualisation of Density-based Clustering using PCA for Iyer Data-set

Result Evaluation

1. Among all the clustering algorithms only DBSCAN is able to show noise (i.e -1) in the PCA plots as seen in the above visualizations.
2. It can be seen that the clusters formed are of different shapes and sizes which is advantageous in case the distribution of points is not regular.
3. The jaccard and rand coefficients is much lower than what was obtained by k-means.

Pros and Cons for Density- Based Clustering

Pros

1. DBSCAN has a time complexity of O(n2) and space complexity of O(n), where n is the number of points in the dataset.
2. The number of centroids does not have to be specified at the beginning as opposed to k-means clustering.
3. DBSCAN can find arbitrarily shaped clusters.
4. DBSCAN can handle clusters of different shapes and sizes.

Cons

1. DBSCAN cannot handle varying densities as it will make choosing the epsilon value difficult.
2. DBSCAN is highly sensitive to parameters making it difficult to determine the correct set of parameters.
3. Border points assignment which are reachable from more than one cluster depends on the order of data processing.

4. Gaussian Mixture Model

Gaussian Mixture Models (GMMs) assume that there are a certain number of Gaussian distributions, and each of these distributions represent a cluster. Hence, a Gaussian Mixture Model tends to group the data points belonging to a single distribution together.

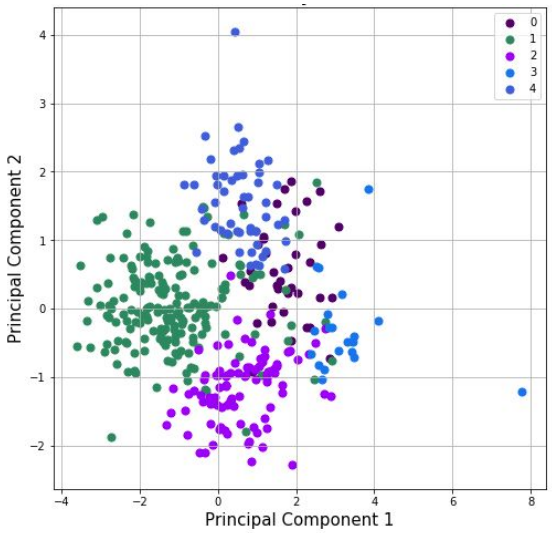
Gaussian Mixture Models are probabilistic models and use the soft clustering approach for distributing the points in different clusters.

The probability density function would be given by:

$$f(x \mid \mu, \Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left[-\tfrac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]$$

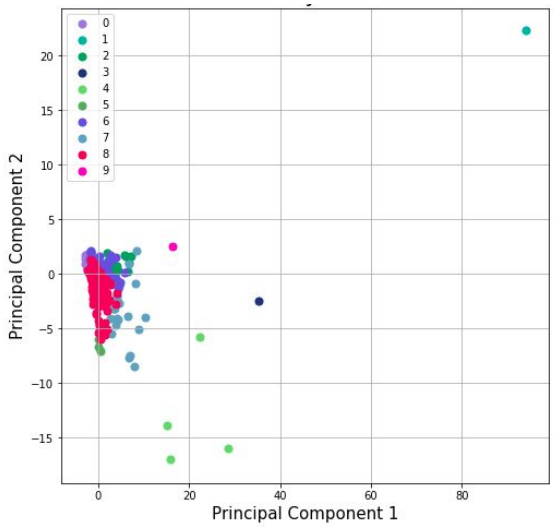
The algorithm for GMM is as follows:

1. The features are extracted from the txt file and stored in a numpy array.
2. No of clusters, iteration count, convergence threshold, smoothing value, mu, sigma, pi are taken as input from the user and passed to the GMM function. Many of these values are calculated using K-means or set default for easy implementation.
3. Then we run the em step to calculate weight and recalculate mean and variance value. Expectation-Maximization (EM) is a statistical algorithm for finding the right model parameters.
4. E-step: In this step, the mean and variance data is used to estimate the weight value
5. M-step: Based on the estimated values generated in the E-step, the complete data is used to update the mean and variance.
6. E-M step is run for iteration count provided or if difference in likelihood is less than the convergence threshold.



Jaccard Index: 0.3684033839129851 | Rand Index: 0.7474563075518806

Figure 7: Visualisation of Gaussian Mixture Model using PCA for Cho Data-set



Jaccard Index: 0.34097286726961623 | Rand Index: 0.7478309994051383

Figure 8: Visualisation of Gaussian Mixture Model using PCA for Iyer Data-set

Pros and Cons for Gaussian Mixture Model

Pros

1. Gives probabilistic cluster assignments, therefore these probabilities can be used to interpret suspected classifications

2. Can handle clusters with varying sizes, variance etc. therefore can be used for non-spherical clusters unlike K-means that work only with spherical clusters

Cons

1. Initialization matters for mean and sigma values to reach maximum likelihood, otherwise it might only reach local maxima.
2. Overfitting issues

Result Evaluation

1. The results obtained are close to the library implementation and is heavily dependent on cluster centroid provided by k means. Therefore, incorrect values from k-means lead to incorrect clusters in gaussian as well.
2. Rand and jaccard value are close to K-means results for the 2 dataset provided initially.

5. Spectral Clustering

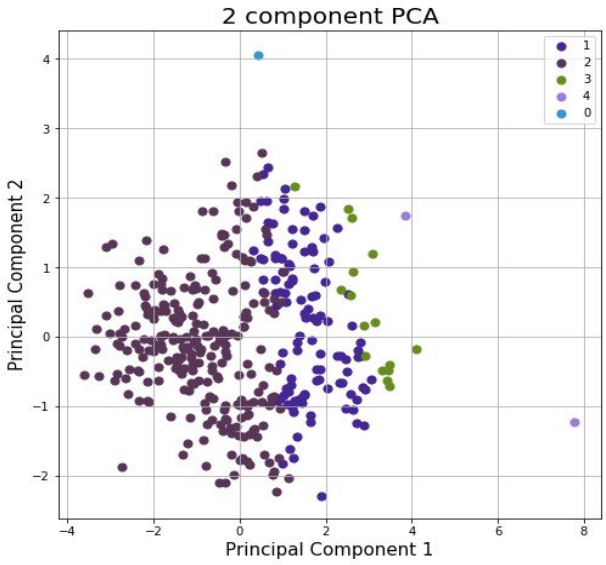
In spectral clustering, the data points are treated as nodes of a graph. Thus, clustering is treated as a graph partitioning problem. The nodes are then mapped to a low-dimensional space that can be easily segregated to form clusters. Spectral clustering uses information from the eigenvalues of special matrices built from the graph or the data set.

The algorithm for Spectral is as follows:

1. In Spectral Clustering, we represent dataset as a weighted graph G(V,E) where V are the vertices which represent data points and E represent edges connecting the vertices.
2. We construct the similarity matrix, which is a NxN matrix where N is the number of rows in the data-set. We fill the cells with the edge weight Wij called as Gaussian Kernel which represent similarity between each pair of points.
3. Once we've built the similarity matrix S, we construct the degree matrix D. For each row of the degree matrix we fill the cell along the diagonal by summing all the elements of the corresponding row in the similarity matrix.
4. After that, we compute the Laplacian Matrix L by subtracting Similarity matrix S from the Diagonal matrix D.
5. After computing the Laplacian Matrix, we compute the Eigenvectors and Eigenvalues for the Laplacian Matrix L.
6. We Build embedded space from the eigenvectors corresponding to the k smallest eigenvalues where k is the Eigen Gap.
7. Apply K-means to the reduced N X K matrix to produce K clusters.

Spectral clustering on cho.txt and iyer.txt with values as follows:

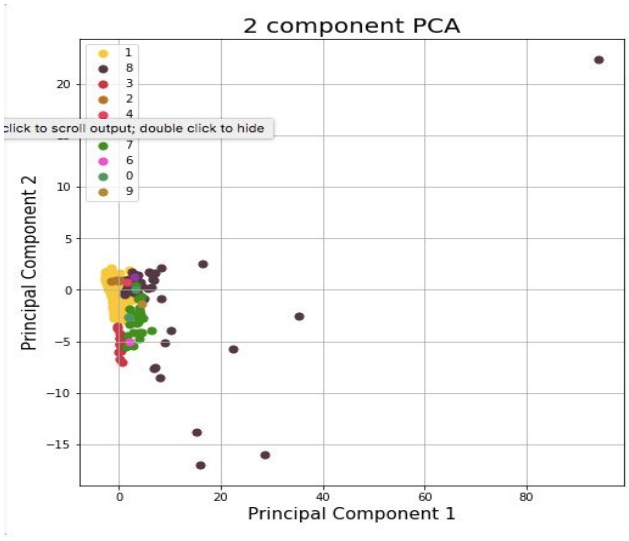
Parameters - Sigma = 3, Number of Clusters = 5



Jaccard Index: 0.308656674604286 | Rand Index: 0.602499395956938

Figure 9: Visualisation of Spectral Clustering using PCA for Cho Data-set

Parameters - Sigma = 5, Number of Clusters = 10



Jaccard Index: 0.19759611515037 | Rand Index: 0.440530661568564

Figure 10: Visualisation of Spectral Clustering using PCA for Iyer Data-set

Evaluation

1. Spectral clustering is computationally expensive unless the graph is sparse and the similarity matrix can be efficiently constructed.
2. The kernel K-means problem is an extension of the K-means problem where the input data points are mapped non-linearly into a higher-dimensional feature space via a kernel function
3. Transforming the spectral clustering problem into a weighted kernel K-means problem greatly reduces the computational burden.

Pros and Cons for Spectral Clustering

Pros

1. It does not make strong assumptions on the statistics of the clusters i.e Clustering techniques like K-Means Clustering assume that the points assigned to a cluster are spherical about the cluster center.
2. It is easy to implement and gives good clustering results. It can correctly cluster observations that actually belong to the same cluster but are farther off than observations in other clusters due to dimension reduction.
3. Reasonably fast for sparse data sets of several thousand elements.

Cons

1. Use of K-Means clustering in the final step implies that the clusters are not always the same. They may vary depending on the choice of initial centroids.
2. It is computationally expensive for large datasets. This is because eigenvalues and eigenvectors need to be computed and then we have to do clustering on these vectors. For large, dense datasets, this may increase the time complexity.

Table 1: Comparing the Clustering results using Jaccard and Rand Index

	Cho.txt	Cho.txt	Iyer.txt	Iyer.txt
Type of Clustering	**Jaccard Index**	**Rand Index**	**Jaccard Index**	**Rand Index**
K-means	**0.33567531823**	**0.7842358184**	**0.35796226653**	**0.771977148**
Hierarchical Agglomerative	**0.22839497757358**	**0.24027490670890**	**0.1582430969664**	**0.18828683559742**
Density-based Clustering	**0.203663430711**	**0.523222099922**	**0.650737591147**	**0.283738059616**
Gaussian Mixture Model	**0.36671398154719**	**0.7604499449649**	**0.3450848666805**	**0.7527170964760**
Spectral Clustering	**0.308656674604286**	**0.602499395956938**	**0.19759611515037**	**0.440530661568564**