CLUSTERING REPORT

Hima Sujani Adike (50246828)

Ratnamala Korlepara (50247117)

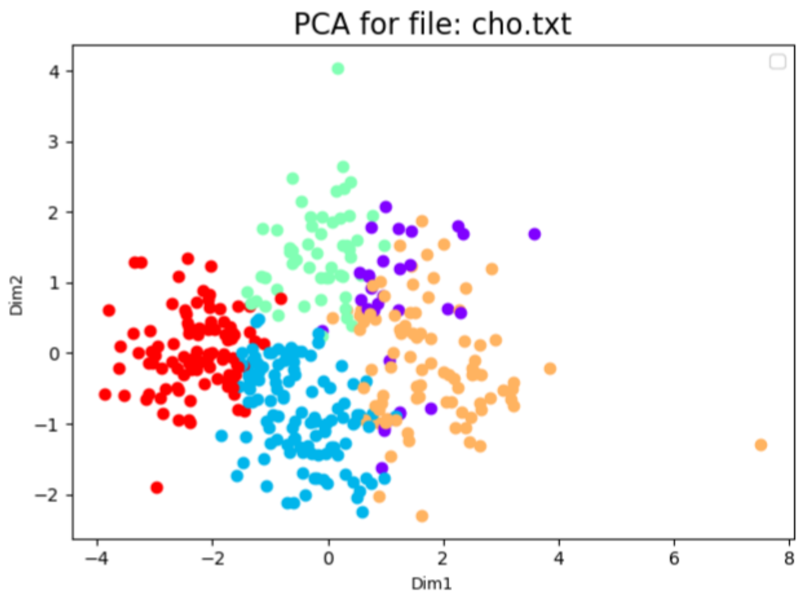
K-Means

K Means is clustering method where the assignment of each cluster depends on the mean from each individual cluster centroid. The data points are merged based on features and the algorithm runs until a stable centroids are obtained.

Implementation

1. Stored each input gene as an object in a list (FileReader -> readFile(filename))
2. Assigned initial centroids based on the number of input clusters. (initializeCentroids(List<Genes> , int k))
3. For each gene, obtain the distance from all centroids and assign the gene to closest cluster. (getClusterFor(Gene, List<Genes> centroids))
4. Process repeats until the cluster centroids remain stable or the iterations exceeded than required. (canStop(previous_centroid, current_centroid, iteration))
5. Calculated external index(rand index) based on the obtained cluster indexes and ground truth. (Index – > randIndex(cluster_indexes, groundtruth))

For cho.txt:



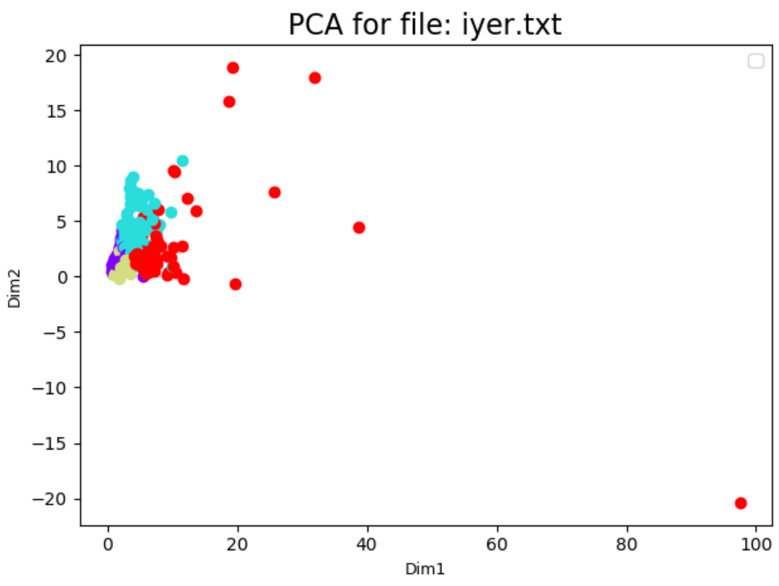
For cluster centroid = [10, 30, 50, 70, 100], K=5

Rand Index: 0.78912

For iyer.txt:

For cluster centroid = [5, 10, 20, 25], k=4

Rand Index: 0.7772523



Advantages:

1. KMeans is easier to implement and efficient for larger datasets
2. For smaller datasets, computations performed are much smaller.

Disadvantages:

1. It is problematic when clusters are differing sizes, densities, irregular shapes.
2. Difficult to predict the initial centroids.
3. For basic KMeans algorithm may produce empty clusters.

Time Complexity: O(NKI) (N-> number of objects, K->Number of clusters,

I-> Number of iterations)

Findings:

1. Cluster centroid initialization is sensitive
2. Some filtering like pre and post processing is required.

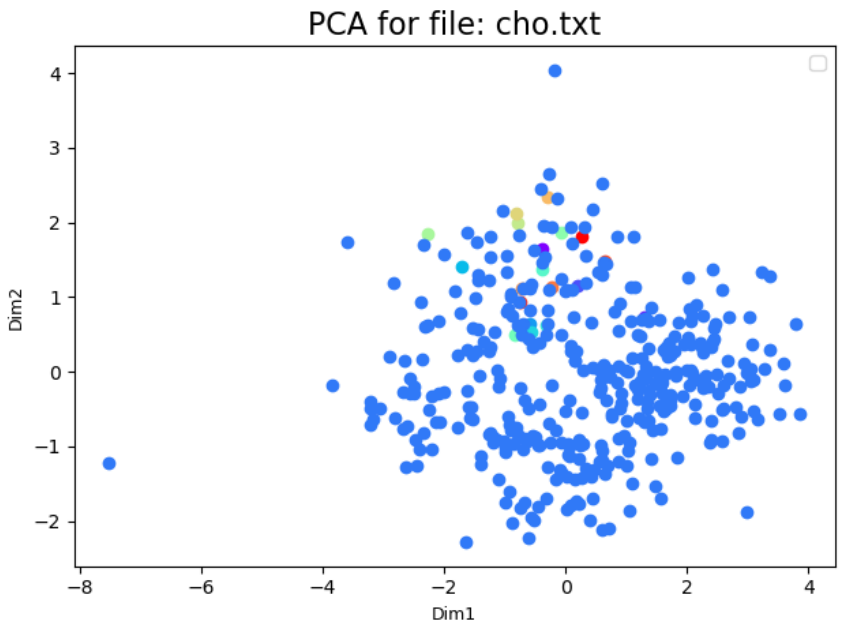
Hierarchical Clustering

Hierarchical clustering is one of the clustering method where clusters are built in hierarchy using dendograms. The clusters are formed based on distance matrix calculated among all the clusters(where every object is viewed as cluster) and then finding the minimum distance between 2 clusters and updating the distance matrix with distance between merged clusters and remaining clusters till all clusters merged into one.

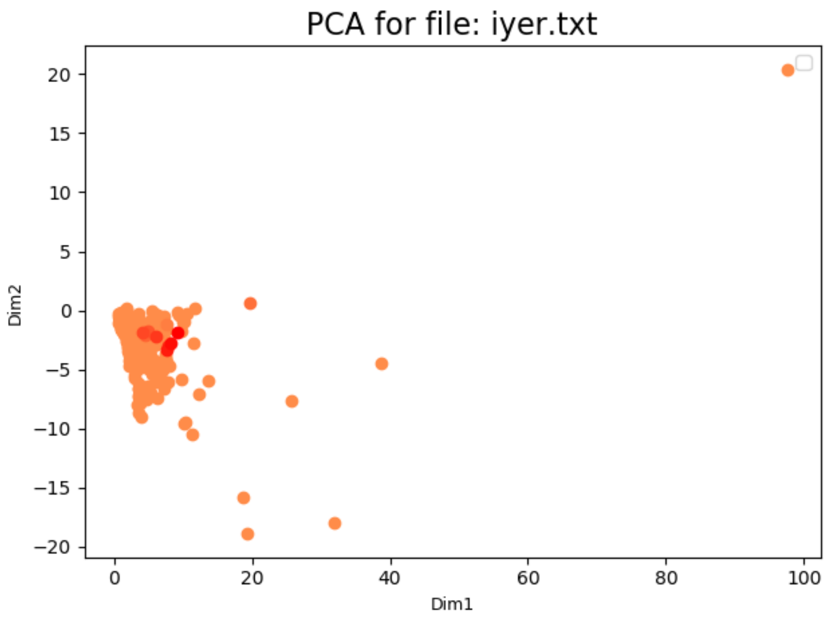
Implementation

1. Stored each input gene as an object in a list (FileReader -> readFile(filename))

2. Initialize the distance matrix with distance among objects (initialize(List<Genes>), distance(<Object1, Object2>))
3. Find the closest clusters to merge (getMergingClusterIndexes(distanceMatrix))
4. Update the distance matrix by calculating the distance between every cluster to the newly merged cluster.(getDistanceMatrix(distanceMatrix))
5. Iterating till the cluster merges as a singe cluster



Rand Index: 0.2968



Rand Index: 0.2673

Advantages:

1. There is no initialization of cluster centroids
2. No need to presume the number of clusters

Disadvantages:

1. Once clusters are merged there is no revert back.
2. Cannot handle different sized and shaped clusters.
3. Due to chaining of clusters the clusters with minimum distance are merged sooner along with outliers.

Time Complexity: O(n^2)

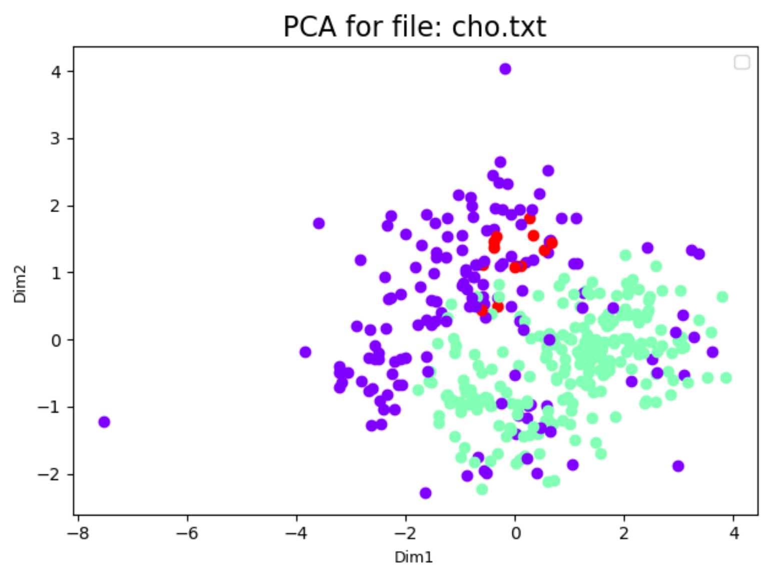
Findings:

1. It does not remove outliers from the cluster as the algorithm converges with single cluster
2. More computations are needed.

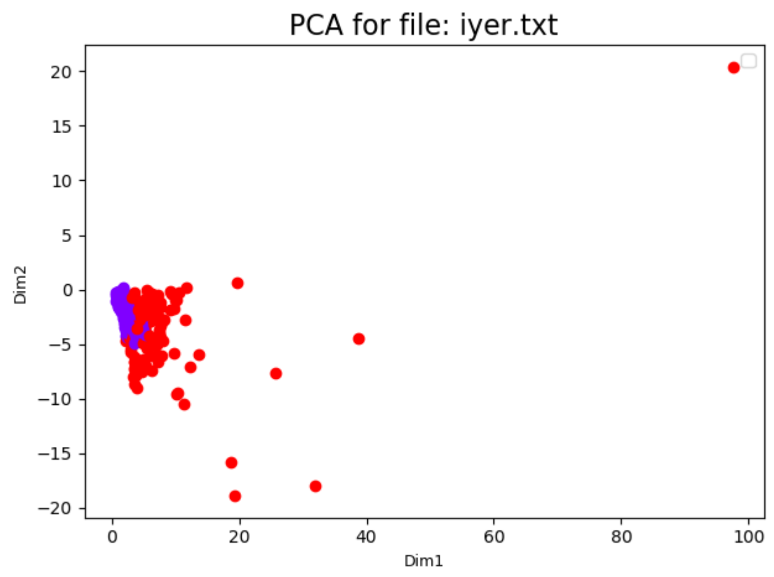
DB Scan

Implementation

1. Stored each input gene as an object in a list (FileReader -> readFile(filename))
2. Calculate the epsilon for the given dataset using KNN (getEpsilon())
3. For each data point get the neighbours i.e., within the epsilon distance and if the number of points are greater than minimum count we mark the object as core object (rangeQuery(Gene, epsilon))
4. If object is classified as core object then get the core object's neighbours every neighbour to expand the cluster so that all the points with particular density fall within the cluster (expandNeighbours(coreNeighbours, data, epsilon, minPoints))



Rand Index: 0.5845



Rand Index: 0.6179

Advantages:

1. It can handle clusters with different shapes and sizes
2. Algorithm is resistant to noise
3. No need to initialize the cluster centroids

Disadvantages:

1. It cannot handle dataset of varying densities
2. It is sensitive to find the appropriate parameters (epsilon, minPoints)
3. If minPoints is large, most of the clusters will be labelled noise and if minPoints is small noise will be grouped into cluster

Time Complexity: O(n^2)

K Means Map reduce

K Means on Hadoop map reduce is used to process the data parallel and this very efficient in larger datasets as the dataset grows time needed to compute kmeans on a system vs single node cluster varies significantly.

Implementation:

Mapper:

1. The input dataset is divided into multiple parts and is given as input to mappers.
2. The mapper calls map function for every single object and output the closest centroid.
3. The centroids are stored in a file and updated continuously.
   Input: Data
   Ouptut: Centroid ID

Reducer:

1. Reducer calculates the new centroid for that particular cluster
2. It calculates the average for all the objects in the cluster and output the new centroid.
   Input: Centroid ID, List<Genes>
   Output: Centroid ID, new Centroid

Advantages:

1. It is much more faster than the basic k means since the processing is parallel
2. It has better complexity and much efficient on huge datasets.

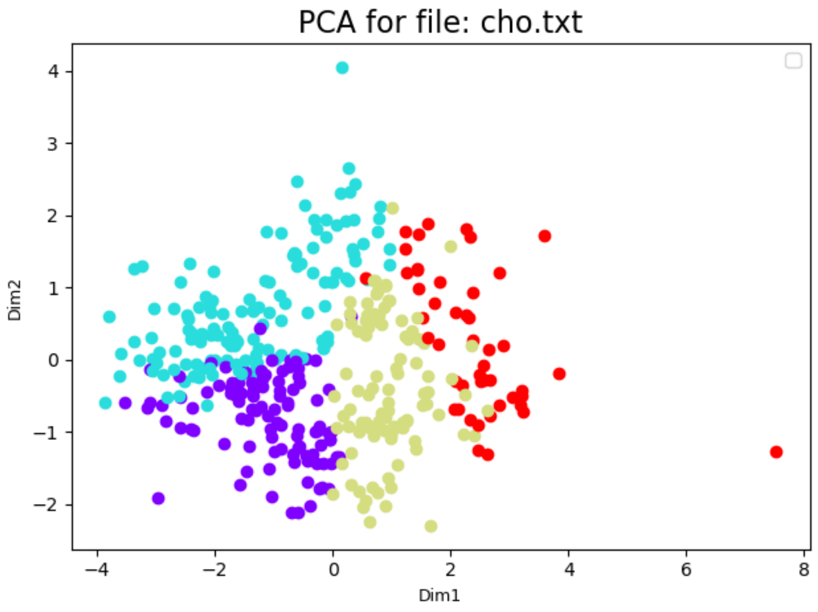
Disadvantages:

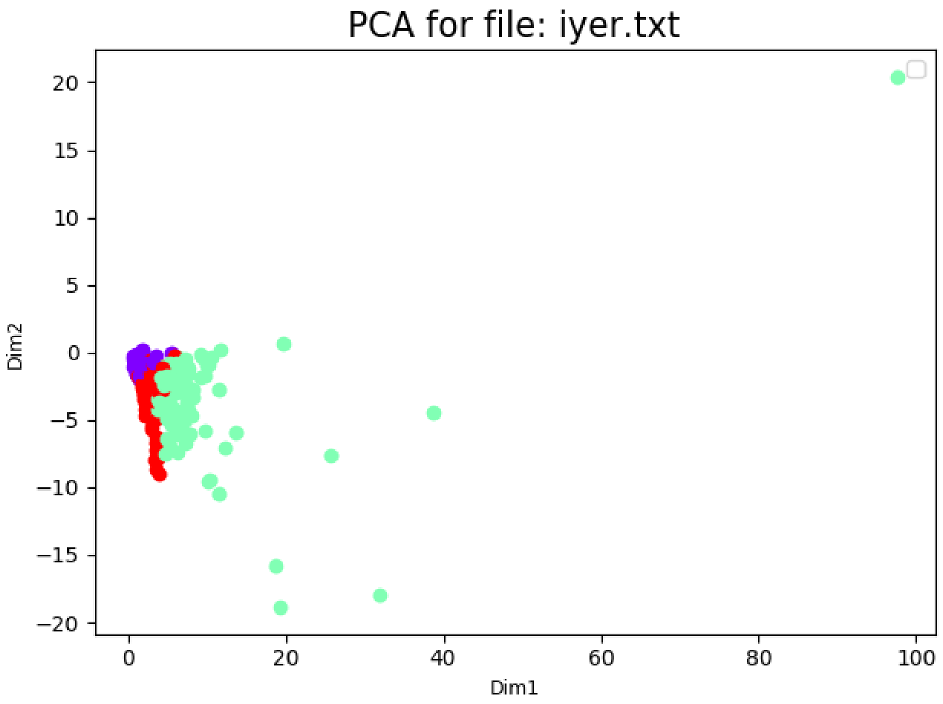
1. It is problematic when clustering sizes or density is different or if it has irregular shape
2. More computations are needed for smaller datasets

Time Complexity: O(nlog k)

KMeans vs KMeans Map reduce:

1. KMeans map reduce is slower than the basic KMeans as the datasets are smaller.
2. It outperforms the basic KMeans with larger datasets because of the parallel processing.


PCA for file: cho.txt
Dim1
Dim2


PCA for file: iyer.txt
Dim1
Dim2