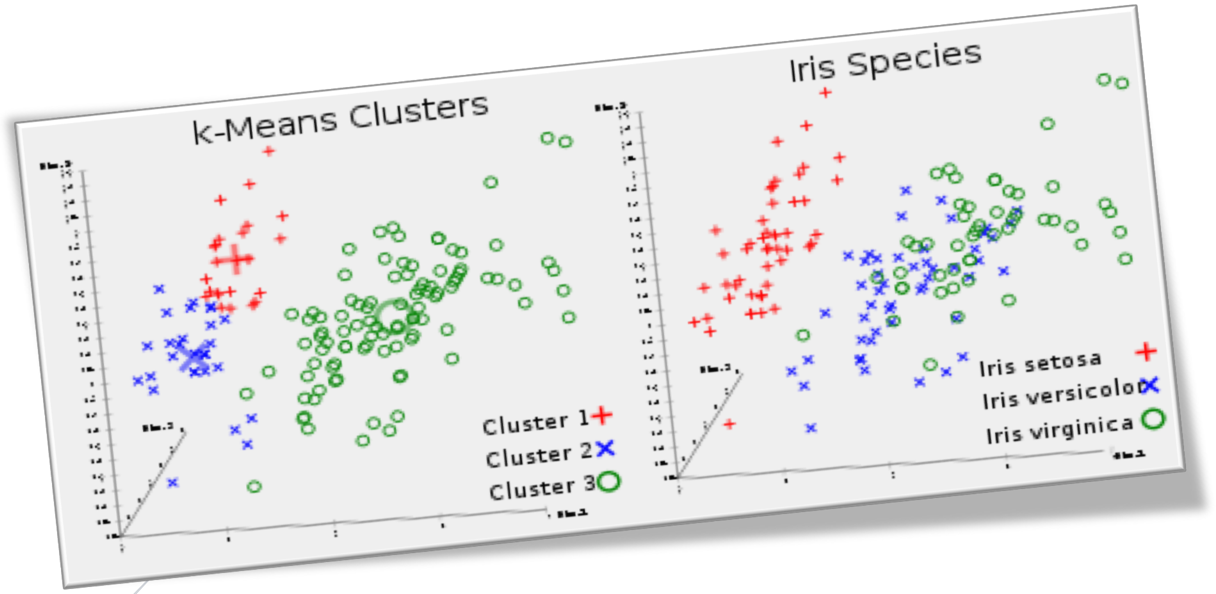
10/29/2018

Project 2: Clustering Algorithms

Report



Submitted By:

Kautuk Desai	50247648
Sujay Vijay Purandare	50205931
Palwinder Singh	50247454

Algorithms:

1. K-means Algorithm
2. Hierarchical Agglomerative
3. Density-Based Spatial Clustering and Application With Noise (DBSCAN)

K-Means Algorithm:

1.1 DESCRIPTION

K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the K-means clustering algorithm are:

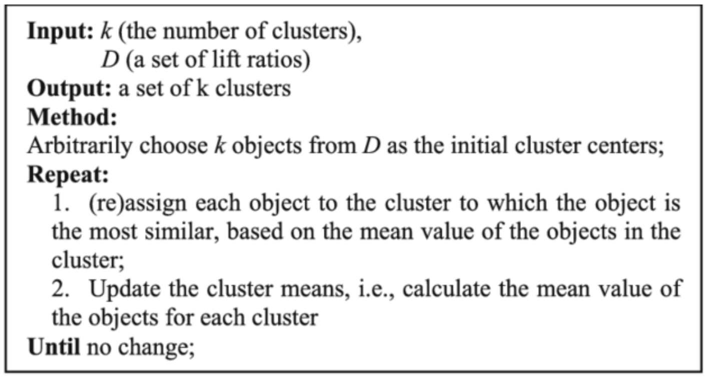
1. The centroids of the K clusters, which can be used to label new data
2. Labels for the training data (each data point is assigned to a single cluster)

Rather than defining groups before looking at the data, clustering allows you to find and analyze the groups that have formed organically. The "Choosing K" section below describes how the number of groups can be determined.

(Reference: https://www.datascience.com/blog/k-means-clustering)

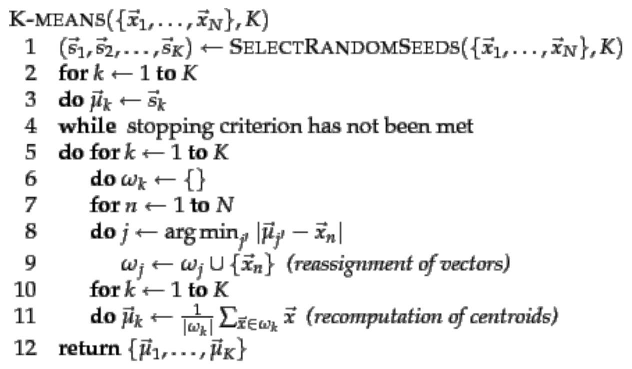
1.2 ALGORITHM AND IMPLEMENTATION

Pseudo code for K-means



```
Input: k (the number of clusters),
       D (a set of lift ratios)
Output: a set of k clusters
Method:
Arbitrarily choose k objects from D as the initial cluster centers;
Repeat:
   1. (re)assign each object to the cluster to which the object is
   the most similar, based on the mean value of the objects in the
   cluster;
   2. Update the cluster means, i.e., calculate the mean value of
   the objects for each cluster
Until no change;
```

Algorithm for K-means :



```
K-MEANS($\{\vec{x}_1, \ldots, \vec{x}_N\}, K$)
 1  $(\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_K) \leftarrow$ SELECTRANDOMSEEDS($\{\vec{x}_1, \ldots, \vec{x}_N\}, K$)
 2  for $k \leftarrow 1$ to $K$
 3  do $\vec{\mu}_k \leftarrow \vec{s}_k$
 4  while stopping criterion has not been met
 5  do for $k \leftarrow 1$ to $K$
 6      do $\omega_k \leftarrow \{\}$
 7      for $n \leftarrow 1$ to $N$
 8      do $j \leftarrow \arg\min_{j'} |\vec{\mu}_{j'} - \vec{x}_n|$
 9          $\omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}$ (reassignment of vectors)
10      for $k \leftarrow 1$ to $K$
11      do $\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$ (recomputation of centroids)
12  return $\{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$
```

▶ **Figure 16.2** The K-means algorithm. For most IR applications, the vectors $\vec{x}_n \in \mathbb{R}^M$ should be length-normalized. Alternative methods of seed selection and initialization are discussed on page 16.4 .

(Reference : Stanford NLP Group)

Implementation:

- We use a couple of functions to implement the algorithm.

   a) fillInput(): This module reads data file and creates 2d array named records where each row represents gene and its attributes.
   b) fillCentroids(): This module takes number of clusters to be formed and initial gene information to be considered as initial centers.
   c) Iterate (): This function asks for number of iterations we want to perform. In each iteration it calculates Euclidean distance from each current center and places each point in the cluster with minimum distance. It then calls function adjustCentroid() which calculates new centers by taking averages of distances and the process repeats.
   d) Find_Jaccard(): We use the original data and cluster indices computed to compute the jaccard co-efficient and random index.

   Jaccard coeff :

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

   Random Index :

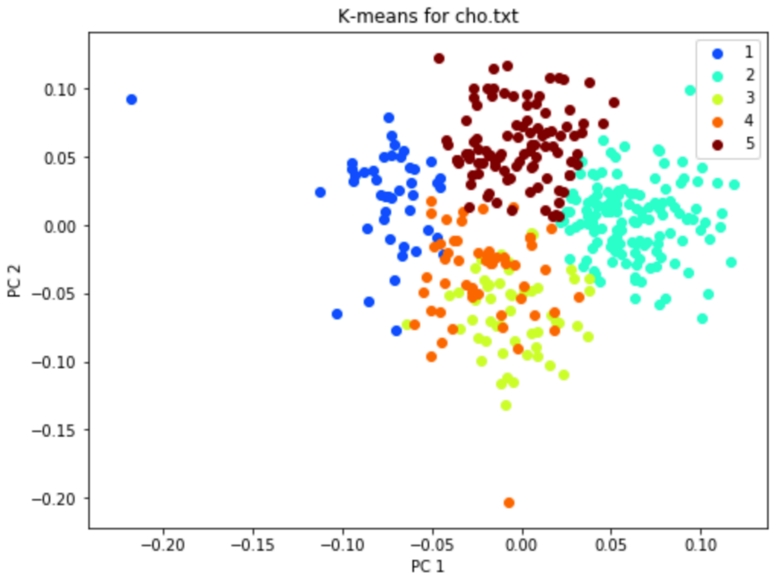
$$Rand = \frac{|Agree|}{|Agree| + |Disagree|} = \frac{|M_{11}| + |M_{00}|}{|M_{11}| + |M_{00}| + |M_{10}| + |M_{01}|}$$

Where, M_{11} : (agree, same cluster) M_{00} : (agree, different clusters)

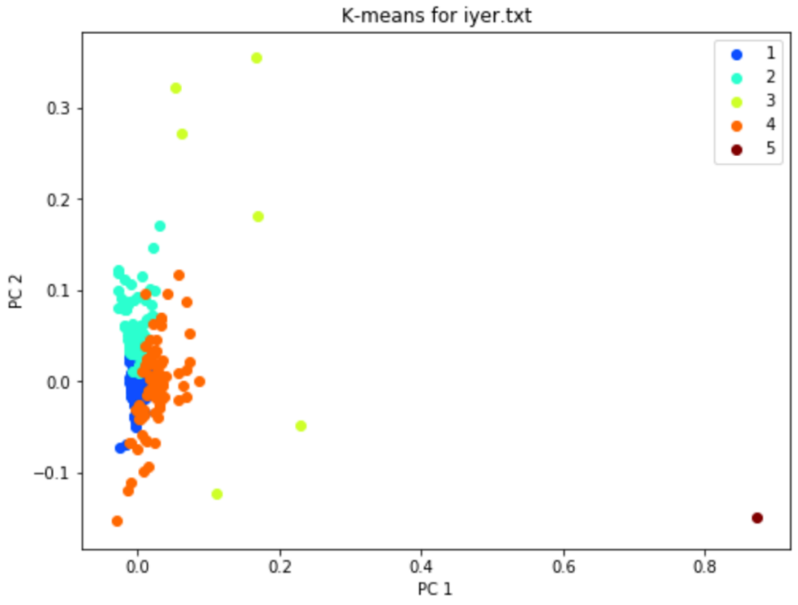
M_{10} : (disagree, different clusters) M_{01}: (disagree, different clusters)

1.3 PLOTS

Data file: cho.txt
initial clusters: 4,32,15,49,3.
Jaccard coefficient is = 0.4111645



data file: iyer.txt
initial clusters: 49,1,99,43,67. Jaccard coefficient is = 0.27837113



1.4 RESULTS, ADVANTAGES AND DISADVANTAGES

Results:

The Jaccard similarity scores are higher when we choose the centroids randomly .Also the initial centroids do adjust themselves.All in all we have good results for both cho.tct and iyer.txt.

Advantages:

- The k-means algorithm is Easy to implement.
- With a large number of variables, K-Means may be computationally faster than hierarchical clustering (if the k value is small).
- An instance can change cluster (move to another cluster) when the centroids are re-computed.
- The k-Means algorithm may produce tighter clusters than hierarchical clustering
- K means provides compact clustering results.

Disadvantages:

- It is often difficult to predict the number of clusters (K-Value).
- The algorithm is sensitive to scale: rescaling your datasets (normalization or standardization) will completely change results. While this itself is not bad, not realizing that you have to spend extra attention (on to scaling your data might be bad.
- Doesn't perform well when the clusters differ in sizes, densities and shapes.
- Sometimes using it may lead to empty clusters (Although most will be clustered).
- Require pre and post processing to give better results.

Applications of the algorithm:

- Widely used in feature extraction utilities.
- Customer segregation is one of the most important use of the algorithm in real life.
- Many other applications such as predicting performance of athletes ,students etc. also use the algorithm.

Hierarchical Agglomerative Clustering:

2.1DESCRIPTION

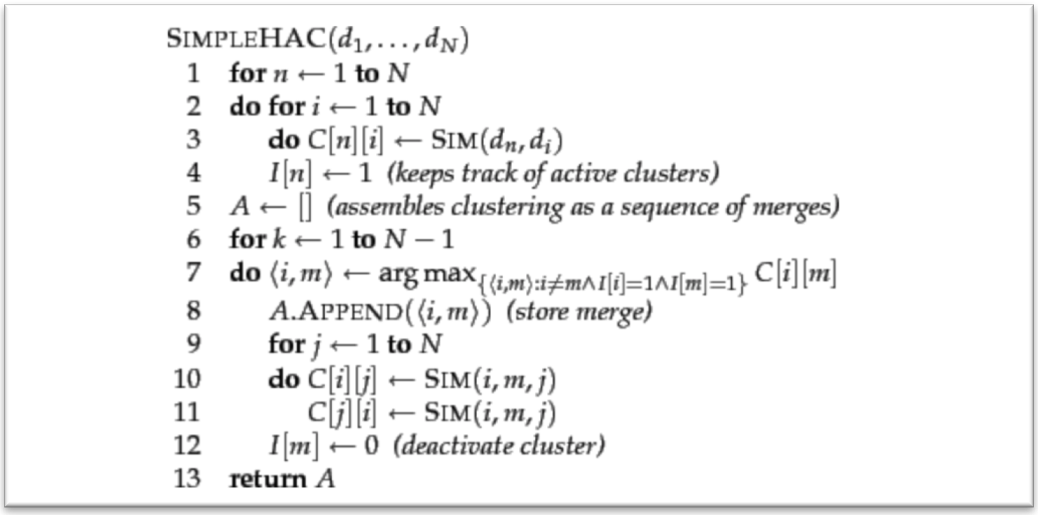
HCA is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:

- **Agglomerative**: This is a "bottom-up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- **Divisive**: This is a "top-down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

Hierarchical clustering algorithms are either top-down or bottom-up. Bottom-up algorithms treat each document as a singleton cluster at the outset and then successively merge (or agglomerate) pairs of clusters until all clusters have been merged into a single cluster that contains all documents. Bottom-up hierarchical clustering is therefore called hierarchical agglomerative clustering or HAC. Top-down clustering requires a method for splitting a cluster. It proceeds by splitting clusters recursively until individual documents are reached

2.2ALGORITHM AND IMPLEMENTATION

Algorithm of a simple HCA:



```
SimpleHAC($d_1, \ldots, d_N$)
for $n \leftarrow 1$ to $N$
do for $i \leftarrow 1$ to $N$
    do $C[n][i] \leftarrow$ Sim($d_n, d_i$)
    $I[n] \leftarrow 1$ (keeps track of active clusters)
$A \leftarrow []$ (assembles clustering as a sequence of merges)
for $k \leftarrow 1$ to $N - 1$
do $\langle i, m \rangle \leftarrow \arg\max_{\{\langle i,m \rangle : i \neq m \wedge I[i]=1 \wedge I[m]=1\}} C[i][m]$
    A.Append($\langle i, m \rangle$) (store merge)
    for $j \leftarrow 1$ to $N$
    do $C[i][j] \leftarrow$ Sim($i, m, j$)
        $C[j][i] \leftarrow$ Sim($i, m, j$)
    $I[m] \leftarrow 0$ (deactivate cluster)
return $A$
```

Implementation :

We used the following modules in our java program to implement the algorithm:

a) fillInput (): This module reads data file and creates 2d array named records where each row represents gene and its attributes.

b) fillCache(): This function creates a 2d array cache where cache[i][j] represents distance from gene with id i and gene with id j.

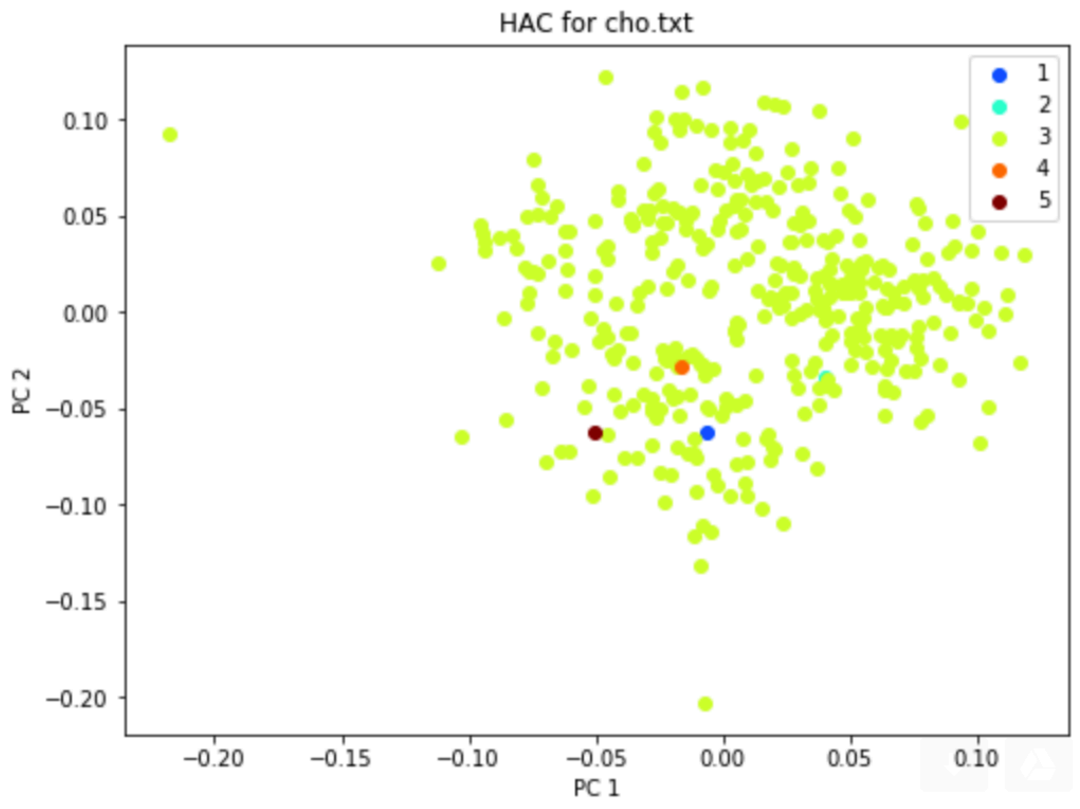
c)formClusters (): This function forms a new cluster by merging two entries i and j from cache[i][j] where the value of cache[i][j] is minimum and then recalculates the distances from new cluster for remaining entries and updates them in cache array

d)findJaccard () : Same as in K-means

2.3PLOTS

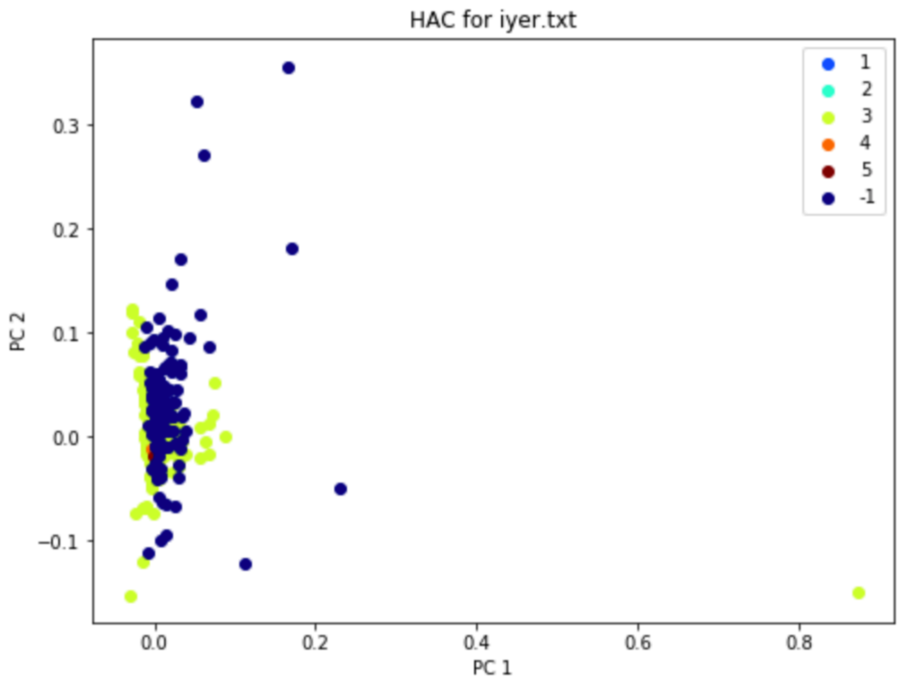
data file: cho.txt

clusters=5 Jaccard coefficient is = 0.2325819



```
data file: iyer.txt

clusters=13
Jaccard coefficient is = 0.15450801
```



2.4RESULTS, ADVANTAGES, DISADVANTAGES AND APPLICATIONS

Results:

- We had to implement hierarchical clustering using single linkage. The HCA algorithm does not identify outliers in the data as it merges all the points.
- Also, it does not provide evenly distributed clusters for user defined number of clusters.

Advantages:

- It is more informative than the unstructured set of flat clusters returned by k-means
- Also, it is easy to implement.
- No a priori information about the number of clusters required.
- it is easier to decide on the number of clusters by looking at the dendrogram.

Disadvantages:

- It is Very sensitive to the outliers
- The Initial seeds have a strong impact on the final results
- It is not suitable for too large a dataset because of time complexity.

- The order of the data has an impact on the final results.
- During the merging step only points with minimum distance are considered, irrespective of other points in the clusters.

Applications of the algorithm:

- Autofill, recommending ads etc.
- Determining patterns for images or other items.
- Charting Evolution through Phylogenetic Trees, Tracking Viruses through Phylogenetic Trees.
- Twitter data clustering is also a good use;

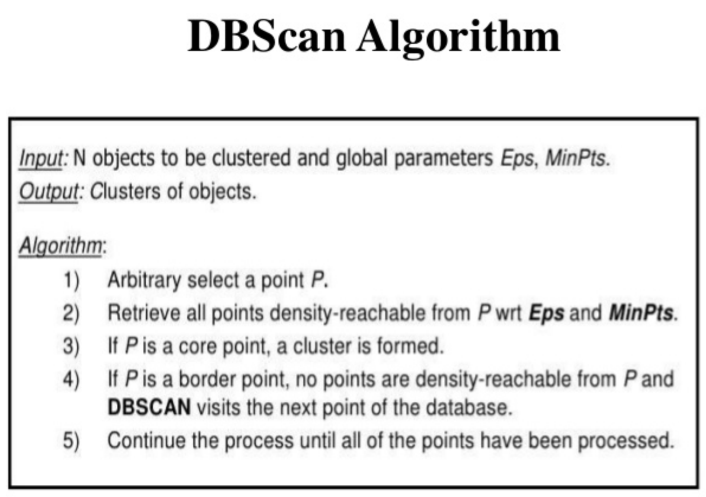
DBSCAN:

3.1 DESCRIPTION

Density-based spatial clustering of applications with noise (DBSCAN) is a well-known data clustering algorithm that is commonly used in data mining and machine learning.
The DBSCAN algorithm basically requires 2 parameters:

1. **eps**: the minimum distance between two points. It means that if the distance between two points is lower or equal to this value (eps), these points are considered neighbors.
2. **minPoints**: the minimum number of points to form a dense region. For example, if we set the minPoints parameter as 5, then we need at least 5 points to form a dense region.

3.2 ALGORITHM AND IMPLEMENTATION



DBScan Algorithm

Input: N objects to be clustered and global parameters *Eps*, *MinPts*.
Output: Clusters of objects.

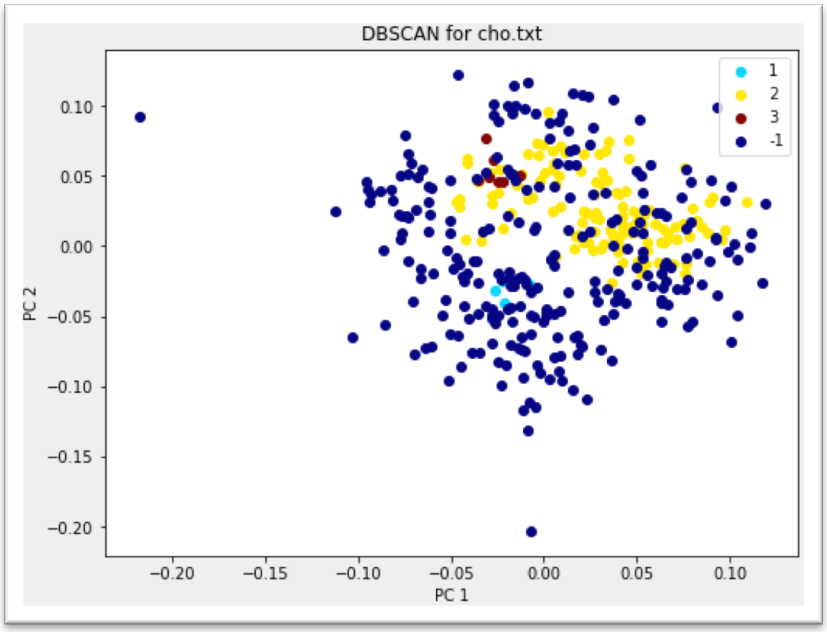
Algorithm:
1) Arbitrary select a point *P*.
2) Retrieve all points density-reachable from *P* wrt ***Eps*** and ***MinPts***.
3) If *P* is a core point, a cluster is formed.
4) If *P* is a border point, no points are density-reachable from *P* and **DBSCAN** visits the next point of the database.
5) Continue the process until all of the points have been processed.

Implemntation :

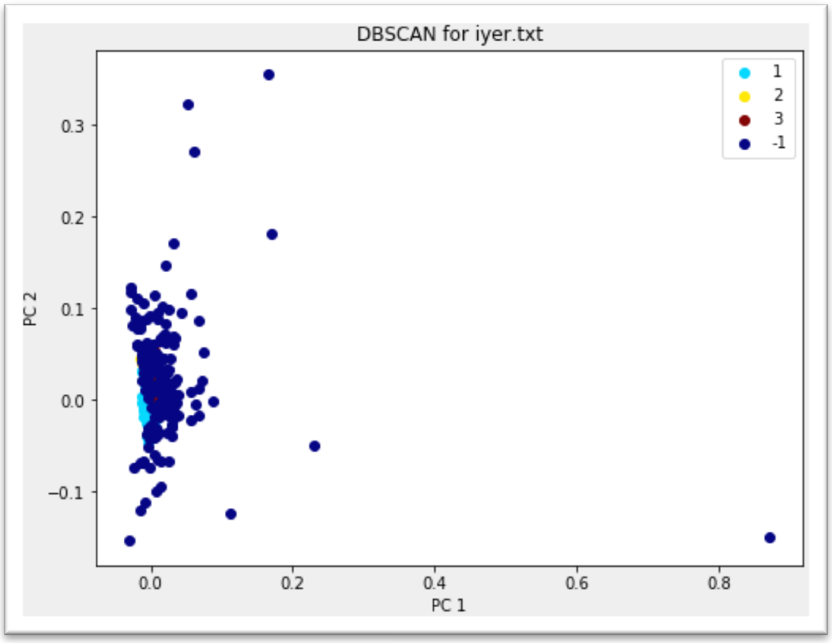
- We implement a function "def compute_dbscan(self, data, eps, min_pts):" , which takes the data and does the processing which includes finding all of P's neighboring points,and comparing the length of the neighbouring points with the min_pts.This function gives us the cluster labels.
- We have another function expand cluster , which assigns the cluster label to the seed point and then gets the next point in the queue. Assignment is done here.(Also explained in code comments)
- Next we use check neighbours fuction to get the neighbours of the P, and this function returns the neighbours.
- Finally we use plot_PCA to plot the pca scatter plots for the DBSCAN algorithm.

3.3 PLOTS

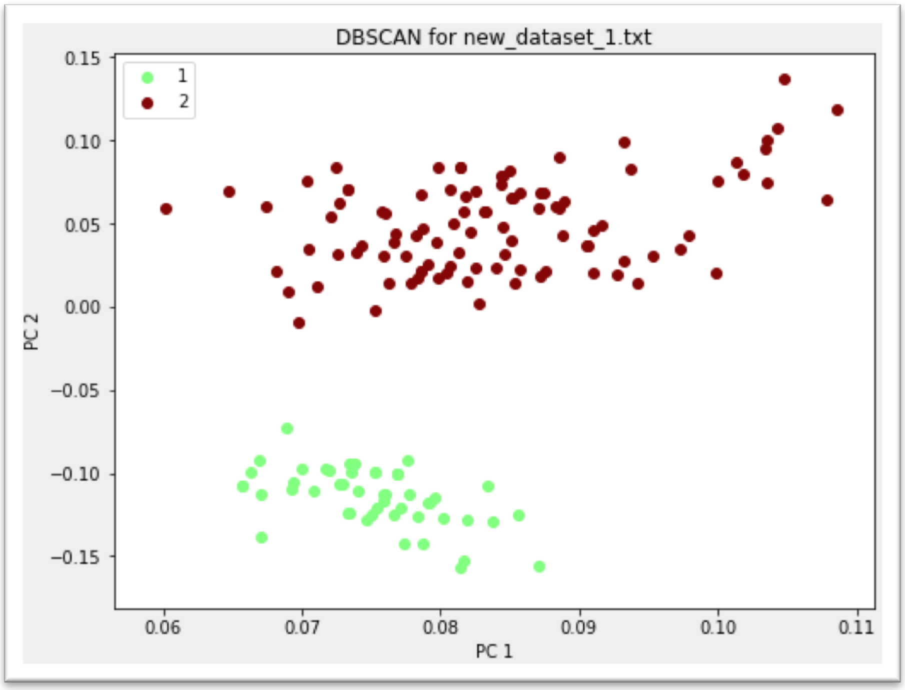
1)cho.txt



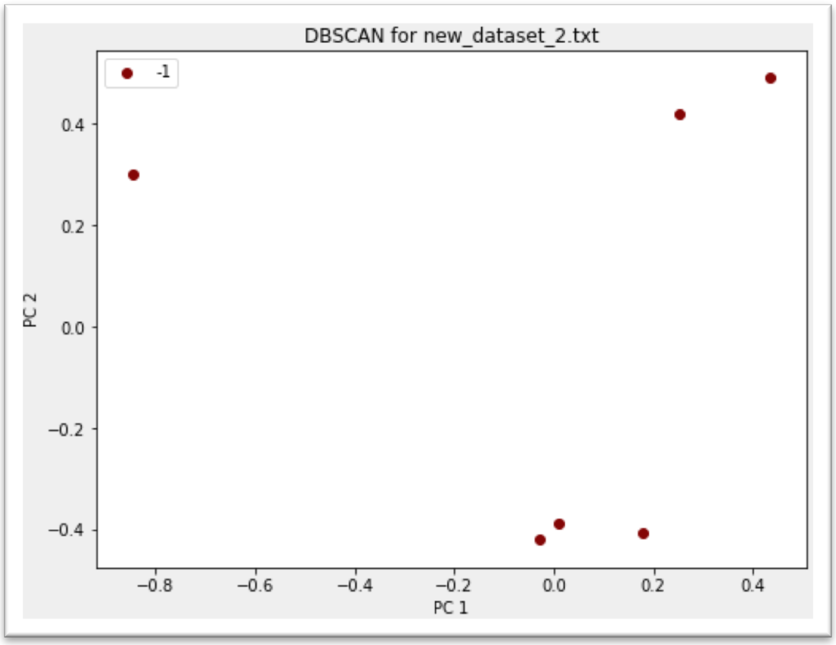
2)iyer.txt



3)new_dataset_1.txt



4) new_dataset_2.txt



3.4 RESULTS, ADVANTAGES, DISADVANTAGES AND APPLICATIONS

Results :

- DBSCAN algorithm assigns labels to each point based on the number of neighbors (n) that are reachable within the ‘ε’ distance.
- The algorithm has O(n2) complexity .
- If n is too big , small clusters are likely to be labeled as noise and if n is too small Even a small number of closely spaced that are noise or outliers will be incorrectly labeled as clusters.

Advantages:

- The algorithm can help us discover arbitrarily shaped clusters.
- It is robust towards outlier detection(noise).
- It does not require one to specify the number of clusters in the data a priori.
- The algorithm is conceptually simple and is very good for small data sets.

Disadvantages:

- While using this algorithm, If the data and scale are not well understood, choosing a meaningful distance threshold ε can be really difficult.
- It is tough to use when working with datasets with altering densities.
- Also, sensitive to clustering parameters minPoints and EPS.
- It fails to identify cluster, if the density varies and if the dataset is too sparse.

Applications of the algorithm:

- Used in Imaging in satellites.
- Crystallography of X-ray.
- Anomaly detection in temperature data.
- Also, in recommendation of goods.

MapReduce K-means:

4.1 DESCRIPTION

Now, we implement the K-means algorithm on a distributed environment.In order to do that we setup a single node Hadoop cluster on our machine , and we implement the Hadoop MapReduce model.
For this we use implement two functions, i.e., mapper and reducer .We use these to implement the algorithm and get plots for the algorithm.

4.2 ALGORITHM AND IMPLEMENTATION

Algorithm :
Same as K-means in 1.2

Implementation:

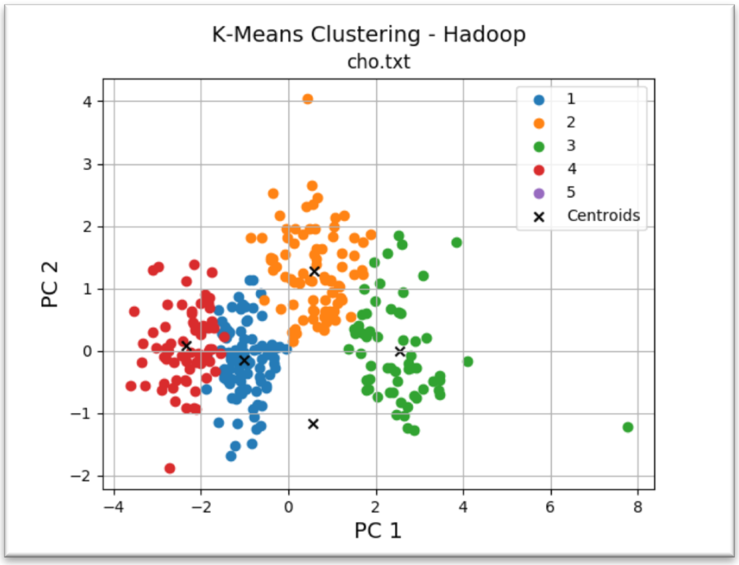
- We program a Mapper.py which is the mapper and reducer.py which will be the reducer. We know the functioning of these two modules.
- In k-means.py , we have a main function in which we load out data file , and also we specify the code regarding the Hadoop functioning(inclusing the Hadoop streaming jar).
- We also programmed utilities to do the following: clear input data of previous runs, create input folder, upload input data on hadoop fs.
- Now we've programmed another file, utilities.py which has all the functions that we require to implement the algorithm. The functions are used to initialize the centroids, check if we have convergence (or if convergence has reached),compute the centroids over and over ,generate the matrices , compute the Jaccard Coefficients and the random index, and finally calculate the PCA and plot them. After all these we use a function map_centroid_data() to map the centroids to the data, according to the gene ids.
- So, this is how we implement the K-means algorithm in a distributed/parallel environment., using a single node Hadoop cluster.

4.3 PLOTS

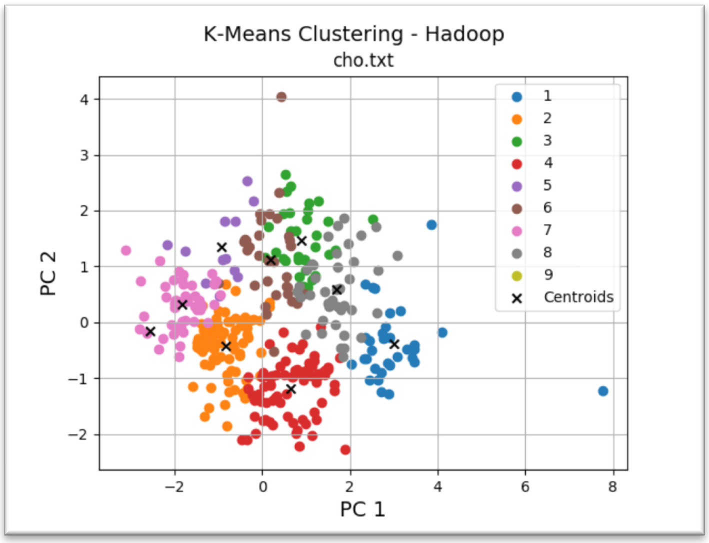
These are the cluster scatter plots obtained by the Map Reduce K-means Algorithm:

1)cho.txt

Initial centroids: [236 173 186 344 154]
Total num of iterations to converge: 10
Jaccard: 0.33703184293
Rand: 0.78424924159

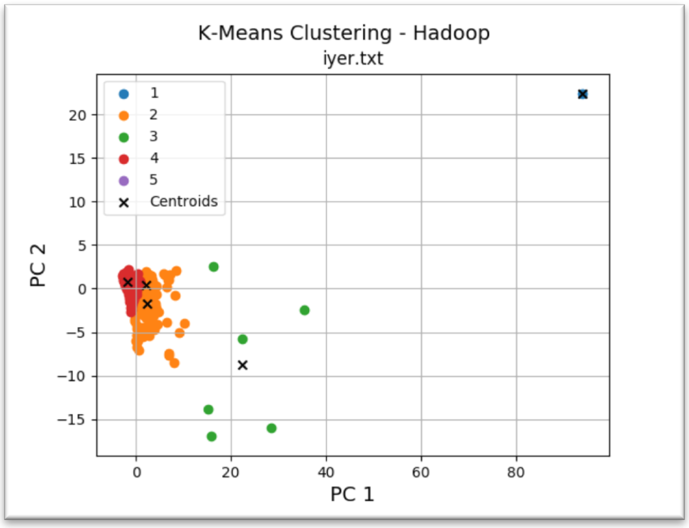


Initial centroids: [112 381 154 60 203 119 186 181 186 5]
Total num of iterations to converge: 14
Jaccard: 0.250172516907
Rand: 0.781215603103

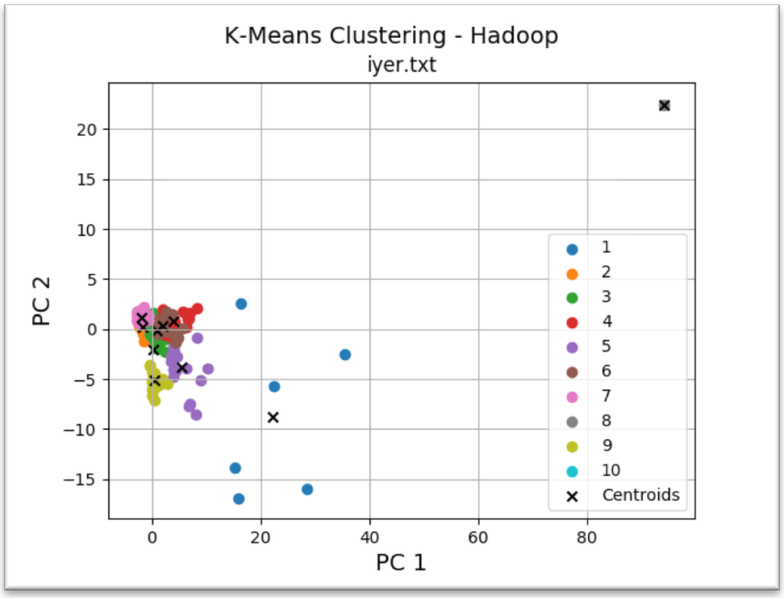


2)iyer.txt

initial centroids: [343 366 313 381 172]
Total num of iterations to converge: 23
Jaccard: 0.266750736775
Rand: 0.606253904949

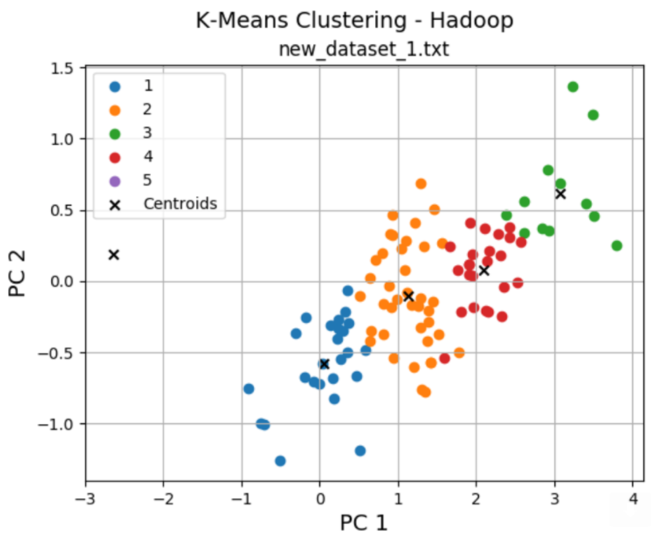


initial centroids: [156 460 48 57 55 163 220 77 347 302]
Total num of iterations to converge: 40
Jaccard: 0.294757207801
Rand: 0.777708772153

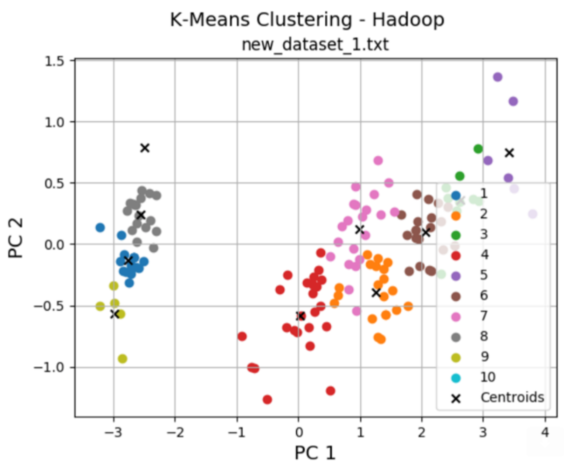


3) new_dataset_1.txt

```
initial centroids: [ 83  80 135  34 127]
Total num of iterations to converge:  5
Jaccard: 0.564445525292
Rand: 0.8408
```

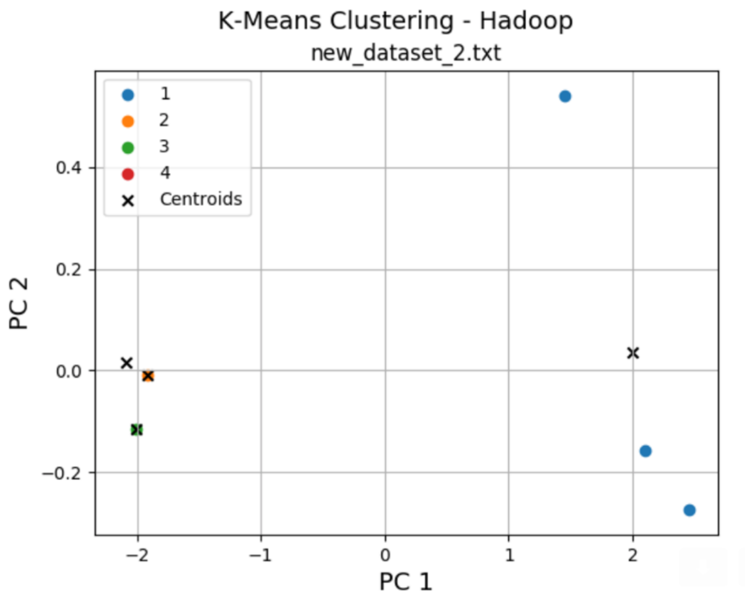


```
initial centroids: [ 91  14  12 140 113  92 103  27 129  39]
Total num of iterations to converge:  11
Jaccard: 0.314752398237
Rand: 0.765066666667
```



4) new_dataset_2.txt

```
initial centroids: [2 5 2 1 0]
Total num of iterations to converge:  2
Jaccard: 0.666666666667
Rand: 0.833333333333
```



4.4 RESULTS, ADVANTAGES, DISADVANTAGES AND APPLICATIONS

Results :

Comparison with non-parallel K-means implemented in 1.1:

- The visualization results show that the map reduce k-means provides similar results to a non-parallel implementation of k-means.
- The running of the algorithm on the single node Hadoop cluster takes some time to give us results.This is due to using Hadoop and various utilities that need to run in order for the HDFS to function.(setting up nodes,job trackers etc).
- For larger data sets the map reduce k means will be better but for the small data sets it is better to use the regular k-means algorithm(non-parallel).
- Parallel calculations do reduce the total time , and also cuts time off the sequential alculations.In the non-parallel k-means more time is spent since all computations are sequential.
- The data sets that we were given to test the algorithms are quite small so the k-means without the map reduce will be better.

Improving running time:

- Since we have used small data sets the time improvement is not a prominent factor.
- We have used functions , which are in one file and are called whenever needed.This setup of functions helped us make the code more efficient .
- So all in all we tried to implement efficient code to make it less time expensive.
- For further reducing the time , we can program a combiner but for the data sets that we were given (small) , it'll perform same as there is a built in combiner in the Hadoop processing system.
- The combiner can be used to average the values if we had huge data set .

Advantages:

- When we perform the k-means using parallel architecture we still have all the advantages that we get in a regular kmeans algorithm.
- The time complexity of the parallel is: O (log k2), wheres as for the non-parallel is : O(tkn), hence parallel is better.
- When we have extremely large datasets it is better to use the parallel k-means.

Disadvantages:

- For large dataset computing the value of K is not feasible.
- When used on a smaller dataset, a parallel K-means needs more time for execution as the parallel setup is time expensive.
- More resources are needed and it is more complex than the non-parallel algorithm implementation.

References :

- Wikipedia.com for definitions.
- https://hadoop.apache.org
- https://www.datascience.com
- Google for algorithms , and images .