

# CSE 601 - Data Mining Project

## **Clustering Algorithms**

---

Anjana Guruprasad(anjanagu - 50205233)

Apoorva Hejjib(apoorvah - 50206516)

Shreya Ravi Hegde (shegde3 - 50208485)

## Introduction

- To implement K-means, Hierarchical Agglomerative Clustering with Single Link(Min) and density based clustering.
- To set-up a single node hadoop cluster and implement K-means on it.
- Implemented external indexes - rand and jaccard to compare the clustering results
- Visualized the results using Principal Component Analysis

## Clustering

- Clustering is the most common form of unsupervised learning
- It tries to find structure in unlabelled data
- The aim is to form groups such that the objects within a group are similar to each other and objects from different groups are dissimilar to each other.

## K-means Clustering

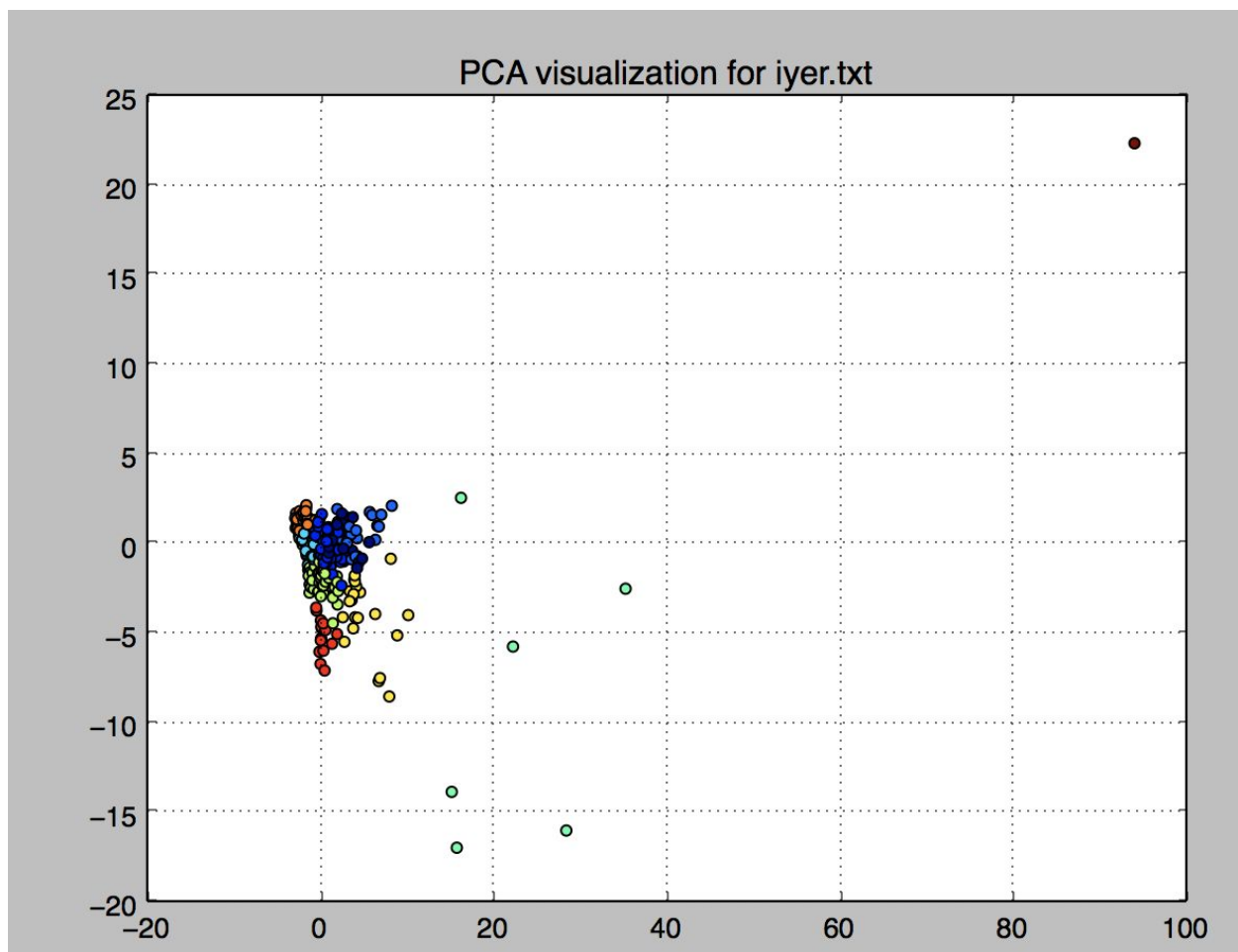
- In K-means clustering algorithm, the unlabelled data is classified into  $k$  clusters
- There are two steps to accomplish K-means:
  1. Assignment step: Each point is assigned to the cluster with the nearest centroid.
  2. Update step: Calculate the new means which will then be the centroids of the points in the new cluster
- The algorithm works iteratively to assign each data point to one of the  $K$  clusters based on the features that are provided.
- Data points are clustered based on feature similarity.
- Feature similarity is computed using functions such as euclidean distances.

## Implementation Details

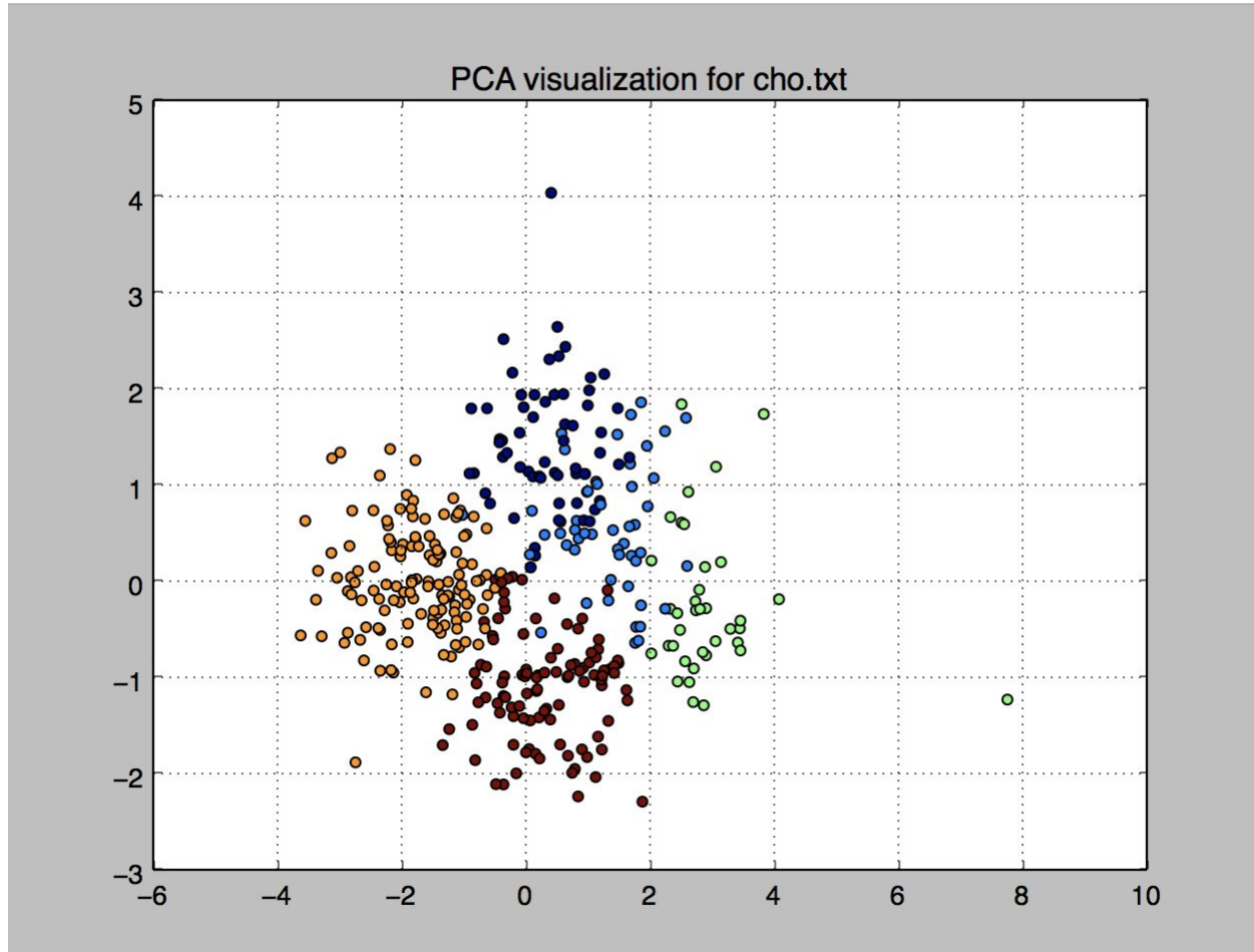
- Get the dataset on which you want to perform the clustering from the console.
- Get the initial parameters such as number of clusters ( $k$ ), number of iterations and the initial set of centroids.
- Repeat the following steps until number of iterations:

1. Calculate the euclidean distance between each point and centroid in `distanceFromCentroids()`.
  2. Assign each point to the centroid which is the closest to it.
  3. Once all the points are assigned to a cluster, new centroids are calculated in `findNewCentroids()`.
  4. Stop the process when either the number of iterations or convergence has occurred.
- Validate the output by computing the jaccard coefficient and rand coefficient using the ground truth specified.
  - Reduce your dataset to 2 dimensions by using principal component analysis.
  - Then, visualize the results using scatter plots.

## K-MEANS algorithm on dataset “iyer.txt”:



## K-MEANS algorithm on dataset “Cho.txt”:



## Results

K value	DataSet Used	Jaccard Coefficient
10	Iyer	0.333
5	Cho	0.376

## Pros of K-means:

- Simple and easy to implement
- Fast and efficient in terms of computation cost  $O(k*n*d)$
- Works really well if clusters are spherical
- It can be used for pre-clustering as it reduces the space into disjoint smaller sub-spaces where other clustering algorithms can be applied.

## Cons of K-means:

- Produces clusters with relatively uniform size even if the input data have different cluster sizes.
- Strong sensitivity to outliers and noise.
- Low capability to pass the local optimum.
- Need to specify the number of clusters initially.

## Hierarchical Agglomerative Clustering with single link (min):

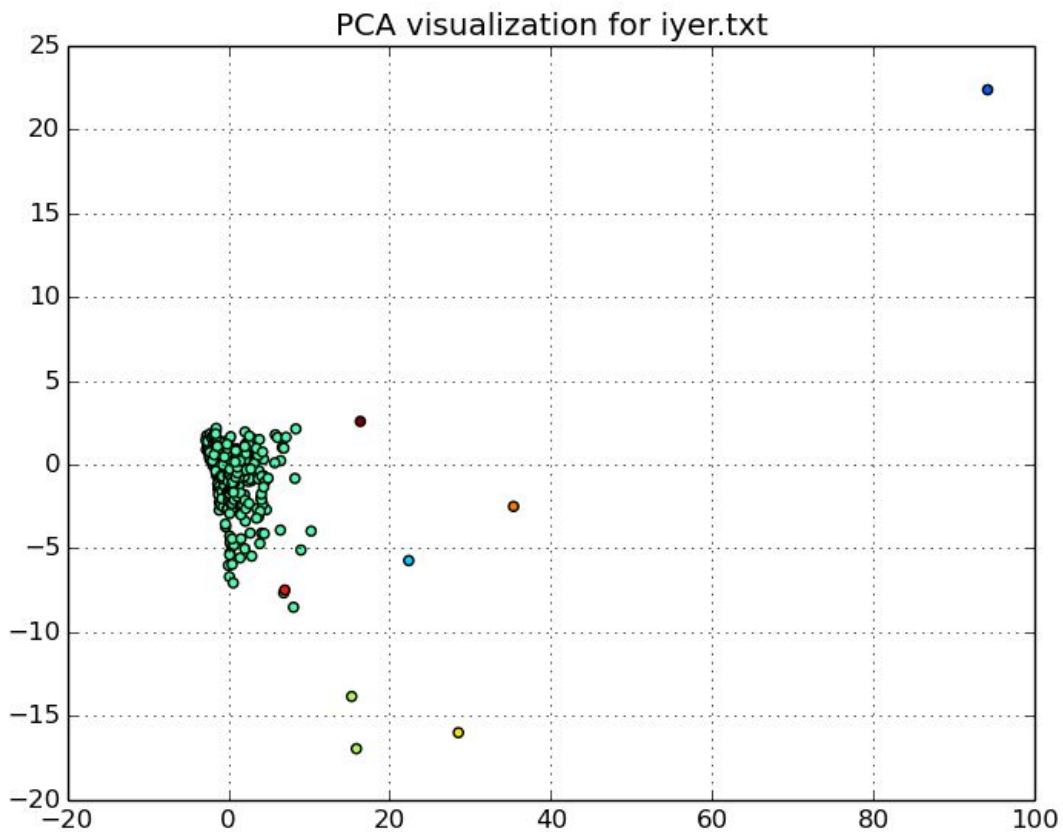
- In hierarchical clustering, a dendrogram(cluster tree) is used to represent clusters.
- Each node in the cluster tree contains a group of similar data.
- Nodes are placed on the graph next to other, similar nodes.
- There are two approaches that can be adopted for hierarchical clustering:
  1. **Divisive method** : In this method we assign all of the observations to a single cluster and then partition the cluster to two least similar clusters. Finally, we proceed recursively on each cluster until there is one cluster for each observation.
  2. **Agglomerative method** : In this method we assign each observation to its own cluster. Then, compute the similarity (e.g., distance) between each of the clusters and join the two most similar clusters. Finally, repeat steps 2 and 3 until there is only a single cluster left. The related algorithm is shown below.

## Implementation Details:

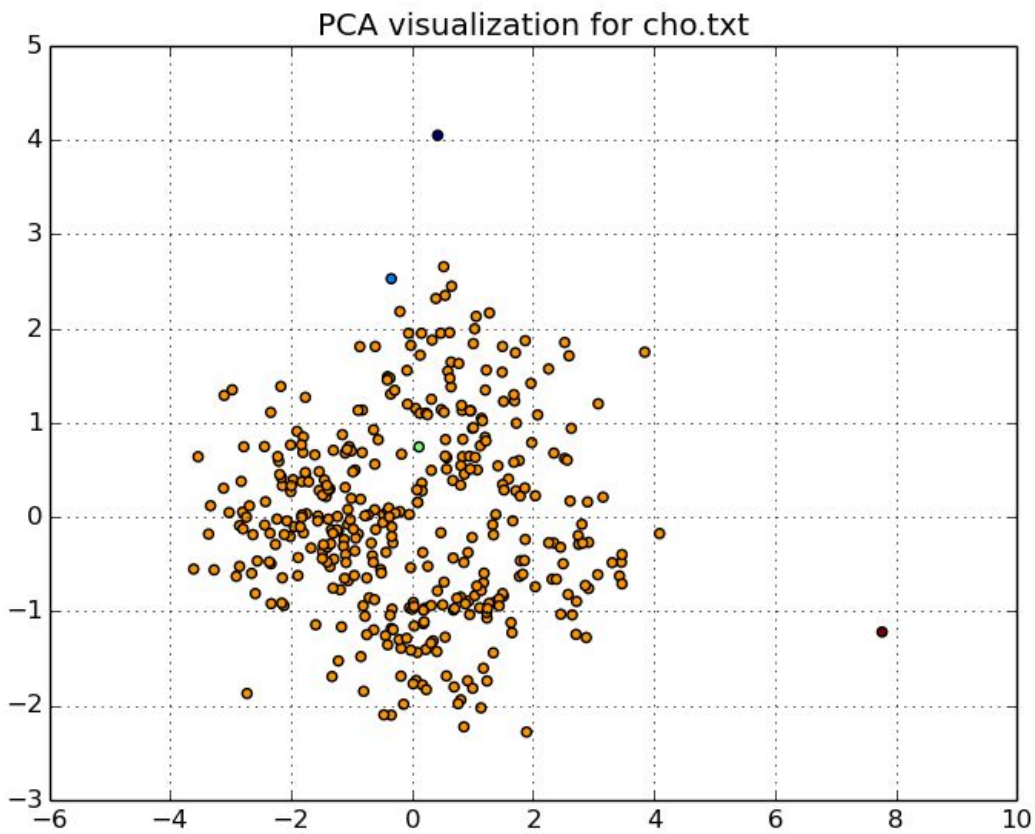
- Get the dataset on which you want to perform the clustering from the console.
- Get the the number of clusters you want to partition the data into as input from the console.
- Construct a distance matrix of size  $n * n$  where  $n$  is the number of data points.

- The matrix should store the value in the form  $\text{distance\_matrix}[i][j] = \text{euclidean distance between the } i^{\text{th}} \text{ and } j^{\text{th}} \text{ data point.}$
- Repeat the following steps until the specified number of clusters are formed:
  1. From the `distance_matrix`, identify the two closest points in the data space.
  2. The indexes corresponding to these points are merged into one row and column in the `distance_matrix`.
  3. The merged data points are added into the same cluster.
  4. The distance values of all the other data points to the merged data points are computed and updated in the `distance_matrix`.
- Validate the output by computing the jaccard coefficient and rand coefficient using the ground truth specified.
- Reduce your dataset to 2 dimensions by using principal component analysis.
- Then, visualize the results using scatter plots.

## Hierarchical Agglomerative Clustering on Iyer DataSet:



## Hierarchical Agglomerative Clustering on Cho DataSet:



## Results

Dataset Used	Number of clusters	Rand Coefficient	Jaccard Coefficient
Iyer	10	0.1883	0.1582
Cho	5	0.2402	0.2284

## Pros Of Hierarchical agglomerative clustering

- No apriori information about the number of clusters required
- Hierarchical Clustering can give different partitionings depending on the level-of-resolution we are looking at

- It is good for datasets of smaller sizes(produces quality results).

## Cons Of Hierarchical agglomerative clustering

- The algorithm can't undo what was previously done.
- Time Complexity of at least  $O(n^2 \log n)$  where  $n$  is the number of data points.
- Hierarchical clustering can be slow (has to make several merge/split decisions).
- Based on the distance matrix :
  - 1) It can be sensitive to noise and outliers
  - 2) It can be hard to handle clusters of different sizes and convex shapes


## Density-based Clustering

- According to density - based clustering, clusters are dense regions in the data space separated by regions of lower object density.
- It can find non-linear shaped structures based on density
- It uses the concept of  $\epsilon$ -Neighborhood, density reachability and density connectivity.
  1.  **$\epsilon$ -Neighborhood** – Objects within a radius of  $\epsilon$  from an object.
 
$$N_{\epsilon}(p) : \{q \mid d(p, q) \leq \epsilon\}$$
  2. **Density Reachability** - A point "p" is said to be density reachable from a point "q" if point "p" is within  $\epsilon$  distance from point "q" and "q" has sufficient number of points in its neighbors which are within distance  $\epsilon$ .
  3. **Density Connectivity** - A point "p" and "q" are said to be density connected if there exist a point "r" which has sufficient number of points in its neighbors and both the points "p" and "q" are within the  $\epsilon$  distance. This is chaining process. So, if "q" is neighbor of "r", "r" is neighbor of "s", "s" is neighbor of "t" which in turn is neighbor of "p" implies that "q" is neighbor of "p".

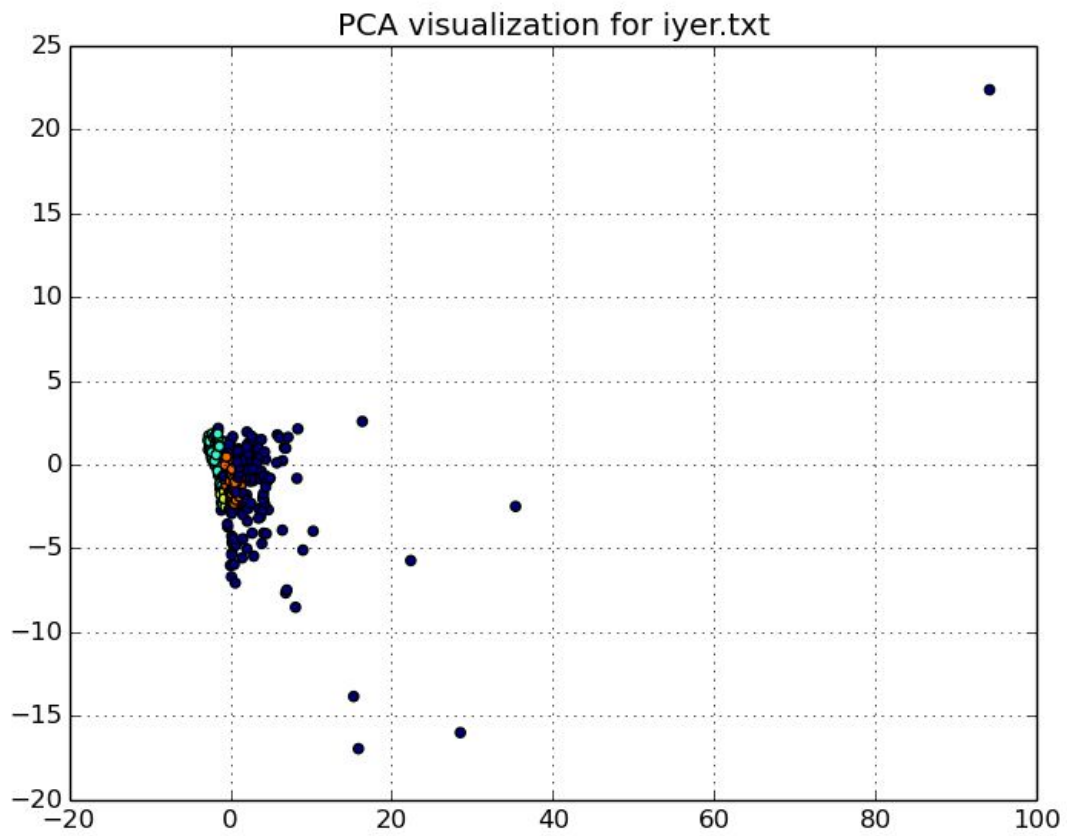
## Implementation Details

- Get the dataset on which you want to perform the clustering from the console.
- Get the initial parameters eps(maximum distance) and MinPts(minimum number of points) from the console.
- Set initial cluster\_id as 1
- For each unvisited data point P, do the following:

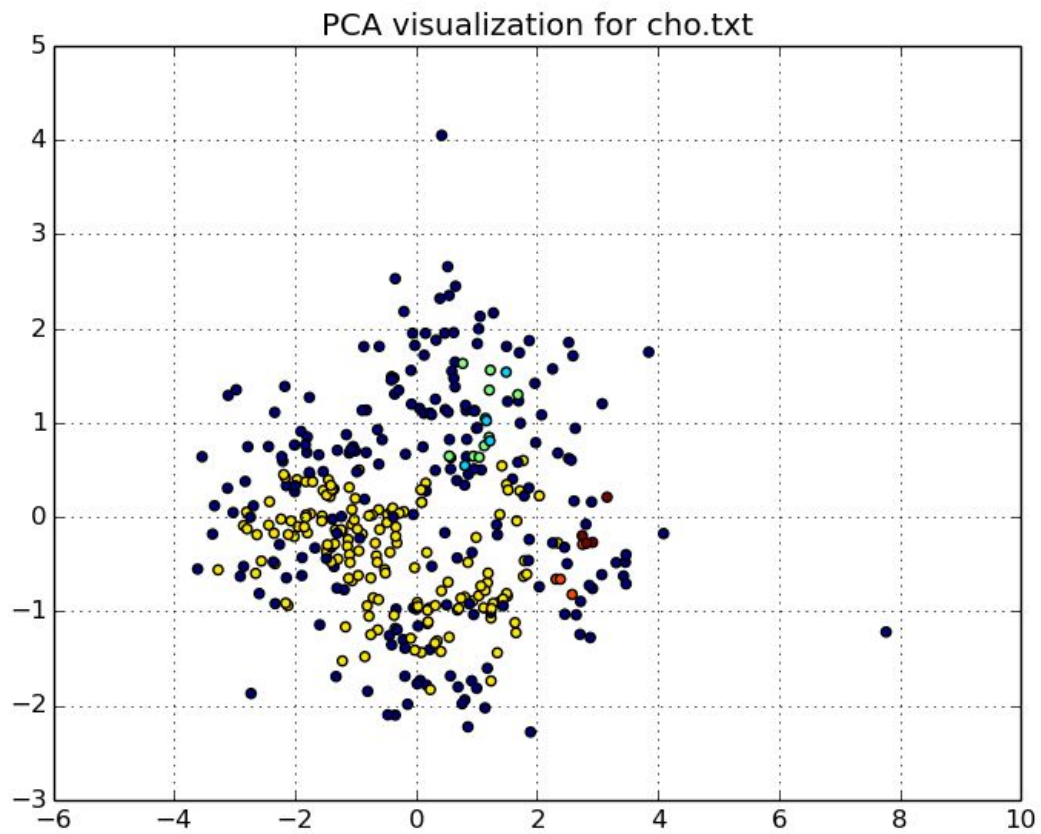


- 
1. Mark P as visited
  2. Generate a list neighbours which contains all the data points(including the point P) within the distance of eps.
  3. If the number of elements in neighbours is less than the parameter MinPts, mark P as a outlier.
  4. Else, call expand\_cluster() and increment the value of cluster\_id
- Add the point P to the current cluster and for each point P' in neighbours which has not been visited:
    1. Mark P' as visited
    2. Generate a new list new\_neighbours which contains all the data points(including the point P') within the distance of eps.
    3. If the number of elements in new\_neighbours is more than or equal to the parameter MinPts, append the elements of new\_neighbours that aren't in neighbours to neighbours
    4. If P' is not assigned to any cluster, assign it to the current cluster
  - Validate the output by computing the jaccard coefficient and rand coefficient using the ground truth specified
  - Reduce your dataset to 2 dimensions by using principal component analysis
  - Then, visualize the results using scatter plots.

## Density-based Clustering on Iyer DataSet



## Density-based Clustering on Cho DataSet



## Results

Dataset used	Eps	MinPts	Rand Coefficient	Jaccard Coefficient
Iyer	1.03	4	0.6523	0.2841
Cho	1.03	4	0.5476	0.2032

## Pros Of Density-based clustering

- Can form clusters of arbitrary shapes and sizes
- Does not require a priori specification of the number of clusters
- Detects outliers effectively (resistant to noise)

## Cons Of Density-based clustering

- Can not handle varying densities
- Does not work well in the case of high dimensional data
- Hard to determine the values for Eps and MinPts( Sensitive to parameters)

## Comparing the Clustering Algorithms

Criteria	K-means clustering	Hierarchical clustering	Density-based clustering
Time Complexity	Time complexity of K-means is linear $O(n)$	Time complexity of Hierarchical is quadratic $O(n^2)$	Time complexity of Density-based clustering is quadratic $O(n^2)$
Outliers (noise)	Sensitive to outliers and noise	Based on the distance matrix, it can be sensitive to noise	Detects outliers effectively
Number of clusters to be formed	Needs to be predefined	Needn't be predefined	Needn't be predefined
Cluster Shape	Works well with clusters of spherical shape	Can capture concentric clusters	Works well on clusters of arbitrary shapes and sizes.

## MapReduce K-means on Hadoop cluster

- Hadoop is an open-source software framework used for distributed storage and processing of dataset of big data using the MapReduce programming model.
- The MapReduce algorithm contains two important tasks, namely Map and Reduce.

- Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).
- Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples.
- The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes.

## Implementation Details

### *Mapper*

- The input to the mapper is all the data points obtained from the input dataset.
- Initially we maintain our initial k centroids in a cache file.
- The mapper reads the values from this file and then compares the Euclidean distance from each data point to all the centroids and assigns the data points to the centroid it is closest to.
- Finally we maintain a dictionary containing all the cluster numbers as the keys and all the points present in that cluster as the value.
- Next, the mapper maps each cluster to the sum of all the values in the cluster along with the number of elements present in the cluster.
- This mapping in the form of a key-value pair is sent to the reducer, where the key is the cluster number and the value contains the sum and the total number of elements.

### *Reducer*

- The reducer upon receiving the cluster number with its respective sum and number of elements, computes the new centroids.
- The output from the mapper is sorted before it is sent to the reducer.
- The reducer will receive the cluster number, sum and total for different clusters from different mappers as the work is split amongst the mappers.
- All values with the same key(i.e, cluster number) will be sent to the same reducer. This is how the reducer computes the centroids for each cluster.
- This now produces k new centroids, one for each cluster.

### *Main Script*

- This main script ensures that the mapper and reducer continue running until the centroids converge i.e all the centroids for each cluster remain the same. Thus having achieved the best possible result.

- After every iteration, the script compares the new centroids given by the reducer to the centroids created from the last iteration.
- If the centroids match, The main script stops executing.
- Else, the old centroids in the cache file are replaced with the new centroids.

## Pros

- It is less time consuming when the dataset is large as it can run multiple jobs in parallel.
- Scalability - This is largely because of its ability to store as well as distribute large data sets across plenty of servers.
- Security and Authentication - MapReduce works with HDFS and HBase security that allows only approved users to operate on data stored in the system.

## Cons

- If the map phase generates too many keys, The sorting of the keys takes a long time.
- Doesn't work too well when you need a fast response.
- Not suitable for real-time processing.
- Hadoop does not have any type of abstraction so, MapReduce developers need to hand code for each and every operation which makes it very difficult to work.

## Improvement of performance of k-means

- Instead of using the non-cached file, We used the cached file in the Hadoop File System (HDFS) to store the current centroid information.

## Comparing the parallel and nonparallel approach to K-means

K-means (Without MapReduce)	K-means (With MapReduce)
Better when dataset is small in size.	Better when dataset is large as jobs can run in parallel.
It works better than MapReduce when the number of iterations are large.	It doesn't work too well when the number of iterations are large.

With our dataset, This took lesser time to execute.	With our dataset, This took longer to execute. As our dataset was relatively small, the overhead of running the hadoop protocols is greater than the computation time.
When we took the initial centroids as random points, We observed that the results of both the approaches were quite similar.	


## Result Summary

We have come up with the following observations by experimenting with the input parameters and comparing the jaccard coefficients:

- The results of the k-means algorithm depends on the choice of initial centroids.
- K-means algorithm had the best jaccard coefficient value of all the algorithm. This signifies that K-means clustering algorithm produced the results closest to the ground truth.
- We observed K-means produced faster results than K-means using MapReduce for our dataset. This shows that MapReduce would produce better results if the dataset we used was larger.
- Hierarchical agglomerative clustering produced the least similarity to the ground truth. This is because the ground truth produces clusters that are more balanced than the hierarchical clustering. When  $k=5$ , hierarchical clustering produces clusters of size 382, 1, 1, 1 and 1 (For cho.txt, the points are not equally distributed like the ground truth).
- The results of the hierarchical clustering depends on the number of iterations.
- The jaccard value for density-based clustering is quite low which signifies that the results are not too similar to those in the ground truth. It could be due to the reason that clusters are formed based on the parameters eps and MinPts.
- This signifies that the results of the density-based clustering are sensitive to the initial parameters.
- Overall, on the given data, K-means performed the best when compared using the external index.

Based on the pca plots for the Iyer and Cho dataset:

- From the plot for Cho we can see that it has somewhat spherical data. When the data is spherical and the number of clusters are predefined, k-means is able to produce more distinct clusters than the other two algorithms.
- DBSCAN works better with Iyer dataset than in Cho, as in Iyer dataset there are more regions of high density.

- 
- Hierarchical clustering works better when the number of clusters are not known beforehand.

## References

- Lecture Slides
- Wikipedia
- Quora