CSE 601: Data Mining and Bioinformatics

Project 2: Clustering Algorithms

By,

Shri Sai Sadhana Natarajan (50247664)

Sneha Parshwanath (50248890)

Soumya Venkatesan (50246599)

K-Means Clustering

Introduction:

K-means clustering algorithm falls under the category of centroid-based clustering. A centroid is a data point at the centre of the cluster and need not be a member of the dataset. This clustering algorithm is an iterative algorithm in which the notion of similarity is derived by how close a data point is to the centroid of the cluster

Implementation:

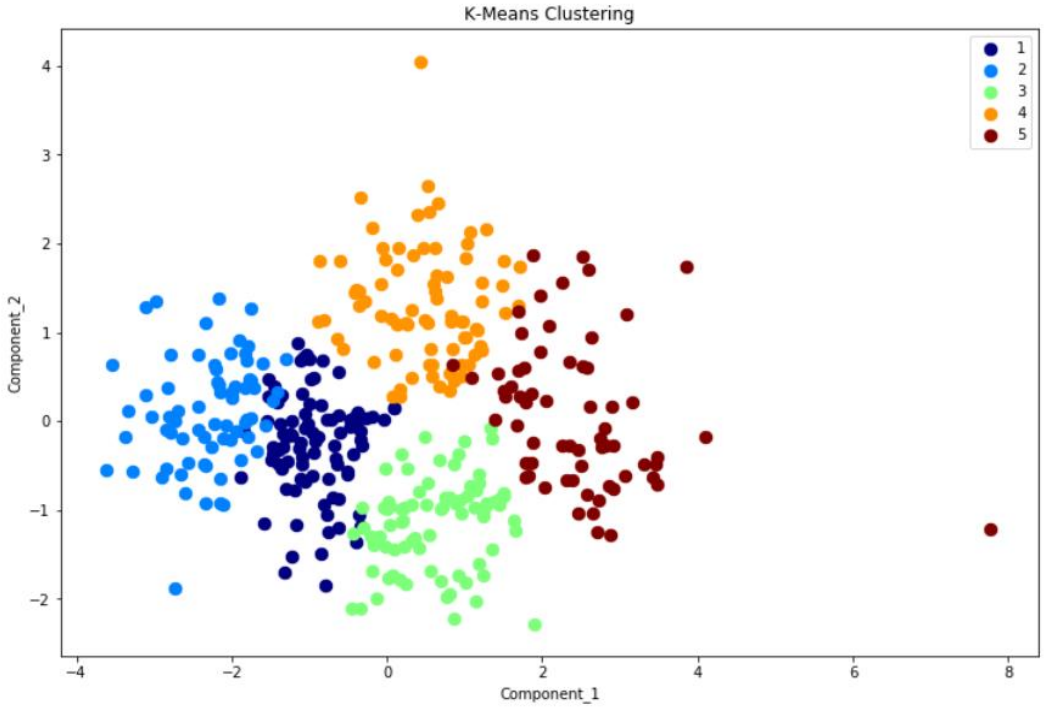
1. The input data is read from the given data file and processed as <Key,Value> pair with key being the gene_id and Value being the attributes.
2. The number of clusters is given as input. The right number of clusters for k-means can be found using Elbow method.
3. Then initially k random centroids are calculated and the distance of each data point from the centroids are calculated using Euclidean distance. The data point is assigned to its nearest cluster. Then the centroids of the new clusters are calculated and this process runs iteratively.
4. Finally, the cluster assignment stops when the data points are no longer reassigned to different custer.
5. The above steps are done iteratively for all the data points and the cluster labels are retrieved in a list.
6. External Index calculation is performed using the ground truth data and cluster labels. The values of m11, m00, m01and m10 are updated by comparing these ground truth labels and clustering labels.
7. Jaccard coefficient and rand index are calculated as per their formulas and results are displayed . Jaccard coefficient: m11/(m11+m01+m10)
   Rand Index: (m11+m00)/(m11+m00+m01+m10)
8. The original data is reduced to two principal components using dimensionality reduction algorithm principal component analysis. This is done using PCA module from sklearn.decomposition library.
9. Finally, the result is visualized using scatter plot.

Result:

The scatter plots obtained for the datasets cho and iyer are as follows:

For Dataset: cho.txt

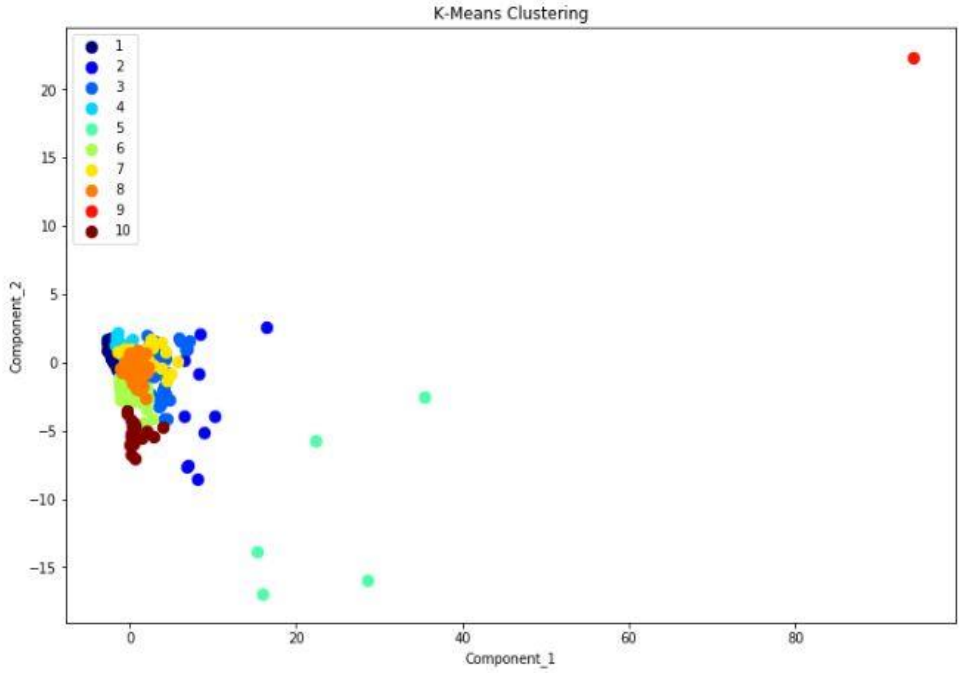
Input given for the value of k : 5




```
JACCARD COEFFICIENT : 0.336
RAND INDEX : 0.785
```

For Dataset: iyer.txt

Input given for the value of k: 10




```
JACCARD COEFFICIENT : 0.354
RAND INDEX : 0.769
```

Evaluation:

- The k-means clustering is straightforward. The data points are assigned to the nearest cluster.
- The clusters are spherical in shape.
- There are points which lie very far from the clusters but still belong to a cluster. These may be outliers and k-means is sensitive to outliers.
- Normally, Cluster size decreases as the value of k increases.

Pros:

- K-means works well with spherical shaped clusters.
- It can produce tighter clusters compared to other algorithms
- This algorithm is an efficient based on run-time and easy to implement
- It is easy to interpret the clustering results

Cons:

- K-means requires the number of clusters to be specified initially.
- Without domain knowledge or ground truth values, difficult to predict the k-value
- The order of the data has affects the final results
- It is sensitive to outliers and does not efficiently handle non spherical clusters

Hierarchical Agglomerative Clustering with Single Link

Introduction:

Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. Agglomerative Clustering is a type of hierarchical clustering which uses Bottom-Up Approach. In Hierarchical Agglomerative Clustering, each object starts in its own cluster, and pairs of clusters most similar to each other are merged as one moves up the hierarchy until all objects are merged into a single cluster.

Single Link which is one of the methods to implement hierarchical agglomerative clustering is used here. In single linkage, the distance between two clusters is defined as the minimum distance between any data point in the first cluster and any data point in the second cluster. At each stage, two clusters with the smallest single linkage distance are combined.

Implementation:

1. The data is read from the given input file and split into two lists one containing the genes data (all columns in the dataset starting from column three) on which the hierarchical clustering is performed and other containing the ground truth data (the second column in the dataset).
2. The number of clusters into which the data should be partitioned is also got as input.
3. The distance matrix is calculated using euclidean_distances module from sklearn.metrics.pairwise library.
4. Hierarchical agglomerative clustering is performed using the following steps:
    a. Initially each point is stored as a single cluster.
    b. The two points separated by a minimum distance that is greater than zero is identified from the distance matrix.
    c. The indices corresponding to this pair is merged into one row and column in the distance matrix.
    d. The distance matrix is updated by re-computing the distance values for all other data points based on their minimum distance to the merged points.
    e. The entry for one of the merged points is deleted from the distance matrix so that it will not be considered again in the next iteration.
    f. The merged data are added to the same cluster.
    g. This process is repeated until the desired number of clusters are obtained.
5. An array called final_clusters is computed with indices as each data point and same value is assigned for all data points belonging to the same cluster. For example: all values belonging to cluster 1 will be assigned the value 1, those belonging to cluster 2 will be assigned value 2 and so on.

6. External Index calculation is performed using the following steps:
    a. The ground truth data and final_clusters array is taken as input.
    b. The values for m11, m00, m01and m10 are updated by comparing the ground truth labels and clustering labels.
    c. Jaccard coefficient and rand index are calculated as per their formulas and results are displayed

        Jaccard coefficient: m11/(m11+m01+m10)

        Rand Index: (m11+m00)/(m11+m00+m01+m10)
7. The original data is reduced to two principal components using dimensionality reduction algorithm principal component analysis. This is done using PCA module from sklearn.decomposition library.
8. Finally, the result is visualised using scatter plot.

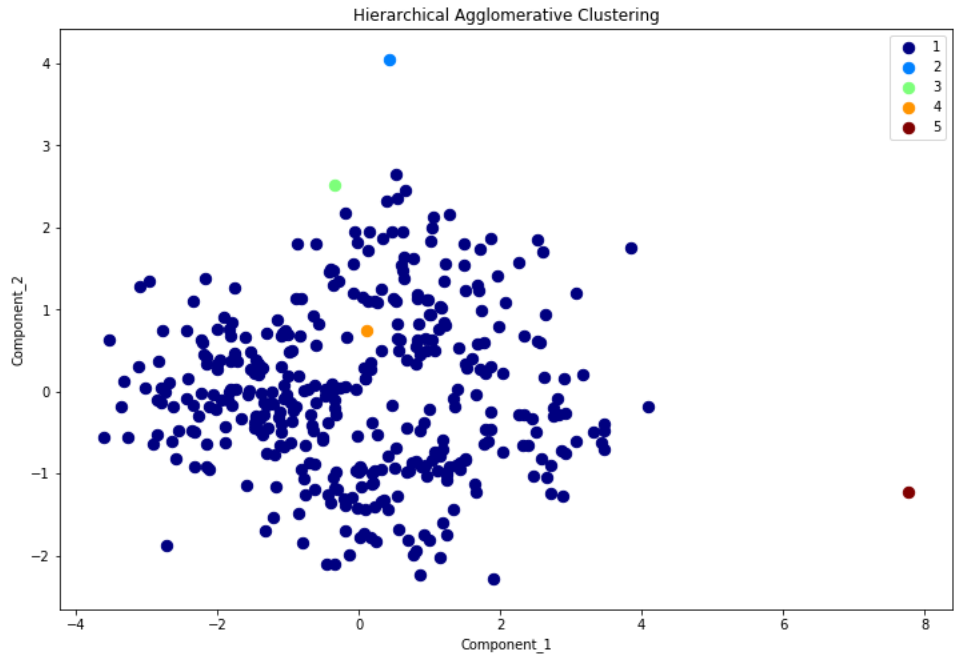
Result:

The scatter plots obtained for the datasets cho and iyer are as follows:

For Dataset: cho.txt

Input given for number of clusters: 5

```
HIERARCHICAL AGGLOMERATIVE CLUSTERING :  cho.txt
```



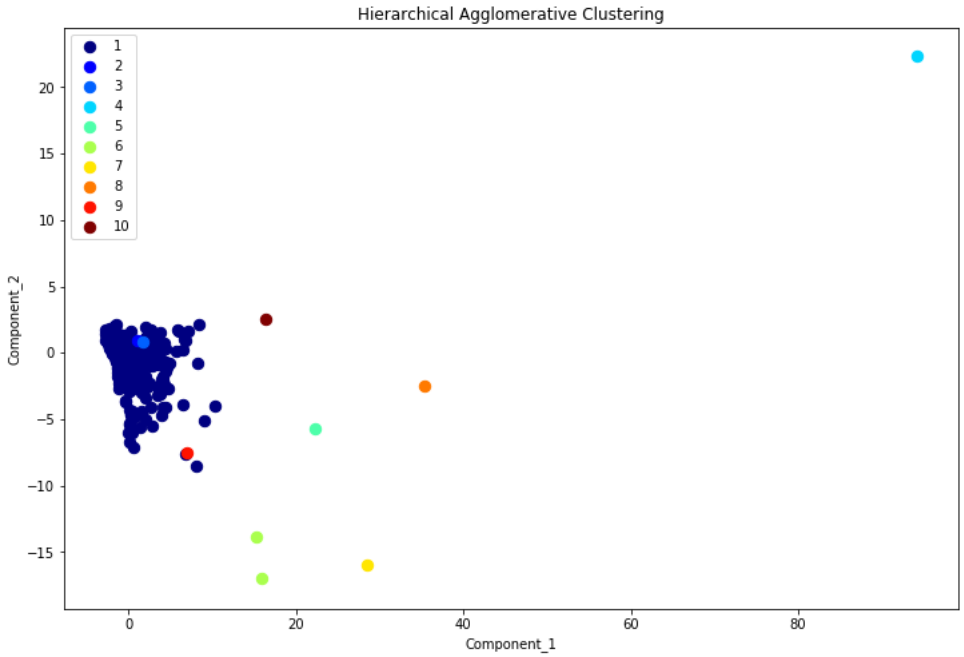

```
JACCARD COEFFICIENT :  0.22839497757358454

RAND INDEX :  0.24027490670890495
```

For Dataset: iyer.txt

Input given for number of clusters: 10

```
HIERARCHICAL AGGLOMERATIVE CLUSTERING :  iyer.txt
```




```
JACCARD COEFFICIENT :  0.15824309696642858

RAND INDEX :  0.1882868355974245
```

Evaluation:

- Hierarchical clustering with single linkage is implemented. So, the points that are closest to each other join first to form a cluster. This leads to the formation of non-elliptical shaped clusters which can be observed from the results.
- It can also be noticed from the results that the dendrogram is cut at a very high point to create the limited number of clusters requested for. So this leads to forming one dominant cluster that contains most of the data points and only the points that are farther apart form small separate clusters.
- It can be seen from the results that few points present inside a large cluster don't belong to that cluster. This might be a consequence of dimensionality reduction, those points might in reality be at a very far distance from the large cluster but due to dimensionality reduction that might not appear so.
- It can also be observed that hierarchical clustering doesn't provide evenly distributed clusters and it is adversely influenced by outliers resulting in low external index values (Jaccard Coefficient and Rand Index) compared to the other algorithms.

Pros:

- It does not require any prior information about number of clusters.
- Any desired number of clusters can be obtained by ‘cutting’ the dendrogram at the proper level
- Produces meaningful taxonomies
- It produces an ordering of objects that is informative for data display(Dendrogram)

Cons:

- It is sensitive to noise and outliers
- It can’t undo any previously made decisions such as combining two clusters
- It is difficult to handle clusters with different sizes and convex shapes
- It has a high time complexity

Density Based Spatial Clustering of Algorithms with Noise (DBSCAN)

Introduction:

DBSCAN is a clustering algorithm that groups together points that are close to each other based on a distance measurement (Euclidean distance) and a minimum number of points. It also marks as outliers the points that are in low-density regions. The algorithm requires two parameters:

- 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.
- minPts: the minimum number of points to form a dense region. For example, if we set the minPts parameter as 5, then we need atleast 5 points to form a dense region.

Implementation:

1. The data is read from the given input file and extracted as two lists one containing the genes data (all columns in the dataset starting from column three) on which the DBSCAN clustering is performed and other containing the ground truth data (the second column in the dataset).
2. The values of eps and minPts are read from the user to compute dense regions.
3. As the first step, the DBSCAN method is called which finds the neighbors and checks the density.
4. If the size of the neighbors is less than minPts then it is marked as noise.
5. Otherwise it is assigned to next cluster and the expand cluster method is called.
6. In the region query method, all the points in the eps-neighborhood are retrieved by scanning all the points, computing the Euclidean distances and checking the value with the eps.
7. Finally, the expand cluster method is called when the neighbor points are density reachable and added to the same cluster.
8. Steps 2-7 are done iteratively for all data points and the cluster labels are retrieved in a list.
9. External Index calculation is performed using the ground truth data and cluster labels. The values of m11, m00, m01and m10 are updated by comparing these ground truth labels and clustering labels.
10. Jaccard coefficient and rand index are calculated and results are displayed

    Jaccard coefficient: m11/(m11+m01+m10)

    Rand Index: (m11+m00)/(m11+m00+m01+m10)
11. The original data is reduced to two principal components using dimensionality reduction algorithm principal component analysis.

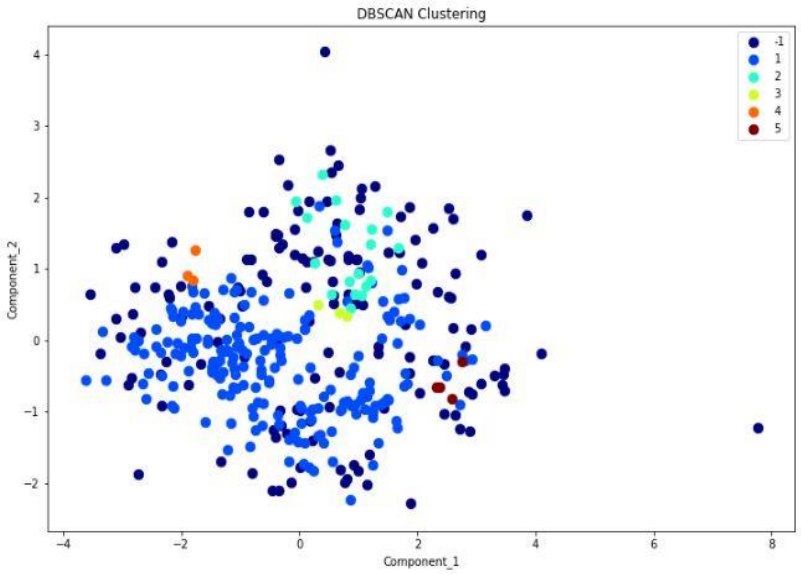
12. Finally, the result is visualized using scatter plot.

Result:

The scatter plots obtained for the datasets cho and iyer are as follows:

For Dataset: cho.txt

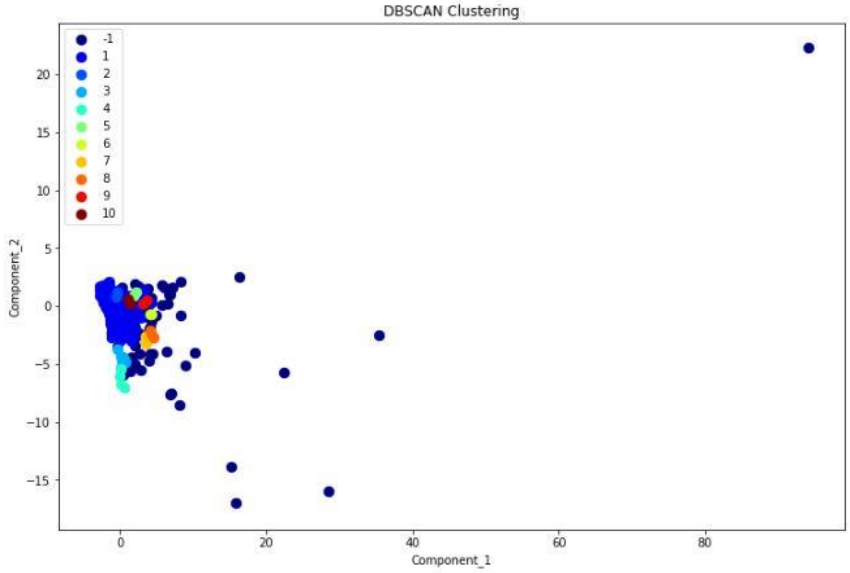
Input given for eps : 1.14 and minPts : 3



JACCARD COEFFICIENT : 0.212
RAND INDEX : 0.561

For Dataset: iyer.txt

Input given for eps : 1.42 and minPts : 2



JACCARD COEFFICIENT : 0.205
RAND INDEX : 0.469

Evaluation:

- On experimenting with different values of eps and minPts, we can see that DBSCAN is robust to outliers.
- If we choose very low eps value, then a large part of the data will be considered noise as they don't satisfy the number of points to create a dense region, but if we choose a high value then most of the data points will be in the same cluster.
- The value of minPts has to be chosen based on the size of the dataset and the dimensions in the dataset.
- With some domain knowledge, if we choose the right parameter values i.e eps and minPts we can determine any arbitrary shaped clusters and even find clusters that are surrounded by clusters.

Pros:

- DBSCAN can handle any shape of clusters.
- It does not require any prior information about number of clusters.
- It requires just two parameters and does not depend on the ordering of the points.
- It is resistant to outliers

Cons:

- DBSCAN is highly dependent on the parameters eps and minPts and it is hard to determine the correct values.
- It fails to identify clusters if density varies or if the dataset is too sparse
- It depends on the distance measure – Euclidean distance. For high dimensional data, it performs poorly due to various phenomena that arise when analyse and organize data.
- The dataset cannot be sampled as sampling would affect the density measures.

K-Means Clustering using Map Reduce:

Introduction:

Map Reduce is a programming paradigm for large datasets which can be processed fast by processing them on distributed clusters in parallel. It consists of two main steps:

- Map Step: In performs a filtering or sorting operation and outputs the result in the form of <key, value> pair.
- Reduce Step: This takes the <key, value> pair given by the map step and usually performs a summary operation.

Apache Hadoop is a popular Map-Reduce Framework.

Implementation:

- There are mainly two files that run iteratively till the required conditions are met: they are mapper.py and reducer.py.
- There is a main.py to control the number of iterations till the converge condition occurs and it calls the mapper and reducer repeatedly. The value of k is set here. The randomly initialized centroid values are passed to the mapper as input.
- In the mapper.py, the k centroids are given as input and the datapoint is given as input. For each datapoint, it finds the Euclidean distance between the datapoint and the centroid. The datapoint is assigned to the cluster to whose centroid it has the minimum distance. It outputs the cluster_id and the datapoint to the reducer.
- In the reducer.py, it takes as input the cluster_id and datapoint given to it and for each cluster_id,it computes the mean of all the datapoints in that cluster and writes the new value of the centroids to the output file.
- The main.py reads the output returned by the reducer and compares the old centroids with the new centroids. If they are not equal it calls the mapper and reducer else, the iterations stop at this point.

Evaluation:

- The given datasets are very small. Hence, running parallel k means takes longer than the serial one resulting in higher runtime. This is because of the setup overhead involved.
- Running parallel k-means is computationally intensive for smaller datasets compared to the serial k-means approach.

Pros:

- The logical partitioning of mapper and reducer functions make the code more readable.
- For very large datasets, parallel and distributed processing of k-means clustering runs significantly faster compared to the iterative serial implementation.

Cons:

- The setup overhead for the parallel k means clustering is very large for small datasets.
- For smaller datasets, serial k-means runs faster than the parallel k means.

References:

- ✓ Lecture Slide: Clustering1, Clustering2, Clustering 3, Clustering 4, Clustering 7
- ✓ http://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.euclidean_distances.html
- ✓ http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
- ✓ https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html
- ✓ https://matplotlib.org/users/colormaps.html
- ✓ https://onlinecourses.science.psu.edu/stat505/node/143/
- ✓ https://en.wikipedia.org/wiki/Hierarchical_clustering
- ✓ https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80
- ✓ https://en.wikipedia.org/wiki/DBSCAN
- ✓ https://en.wikipedia.org/wiki/K-means_clustering
- ✓ https://mubaris.com/posts/kmeans-clustering/
- ✓ https://www.datacamp.com/community/tutorials/k-means-clustering-python