University at Buffalo
The State University of New York

Project 2: Clustering

By:
Akshay Kumar (50169103, akumar34)
Priyanka Singh (50169994, psingh28)
Sahil Dureja (50168872, sahildur)

Agglomerative Hierarchical Clustering

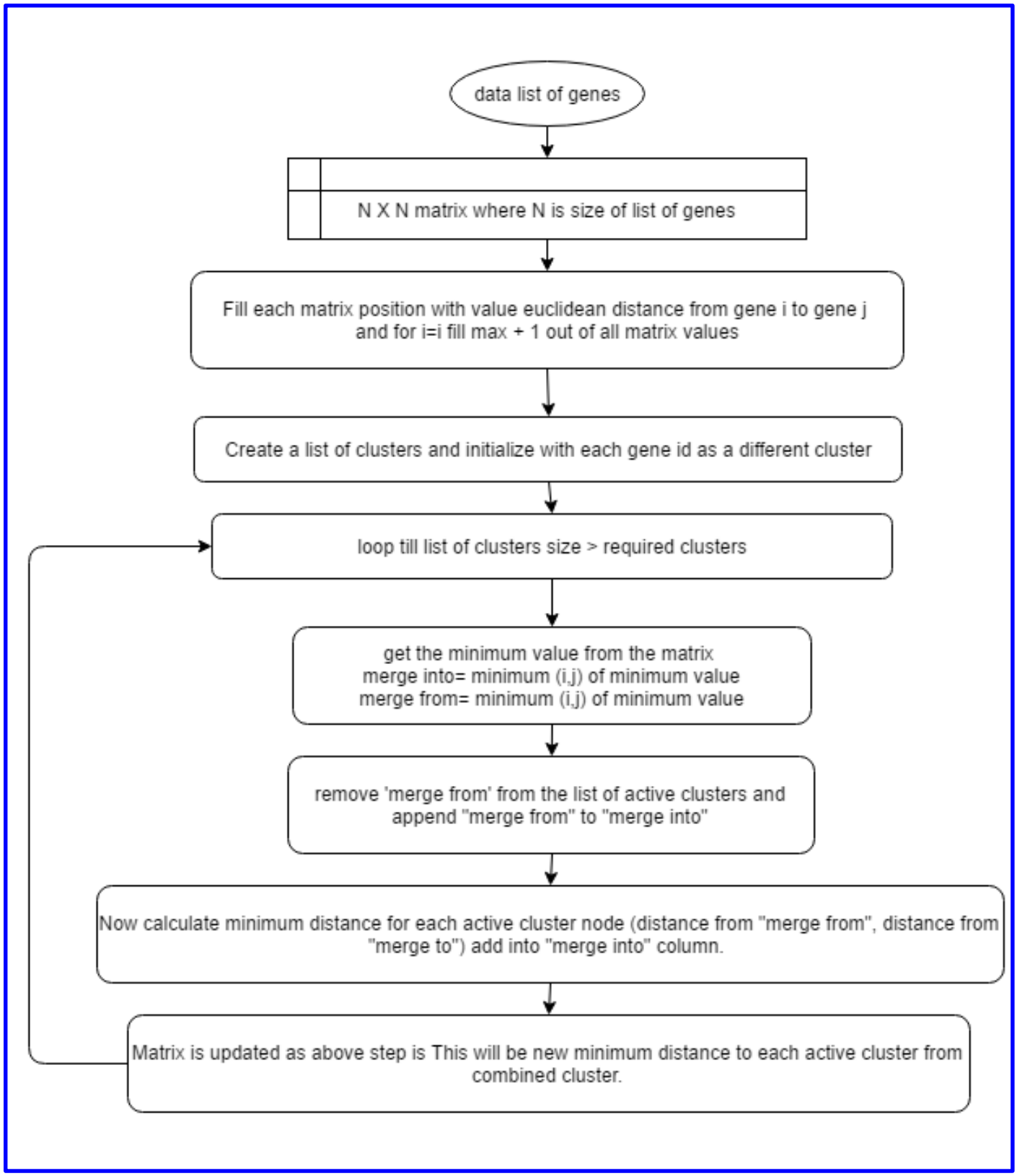
ALGORITHM DESCRIPTION

There are two types of Hierarchical clustering: Agglomerative and Divisive. In this project we have implemented the Agglomerative approach, also known as the bottom-up approach, in which every data point is treated as a singleton cluster at the beginning. Afterwards, pair of closest clusters are merged repeatedly.

The algorithm proceeds with following steps:

- Assign each object to a separate cluster.
- Evaluate all pair-wise distances between clusters.
- Construct a distance matrix using the distance values.
- Look for the pair of clusters with the shortest distance.
- Remove the pair from the matrix and merge them.
- Evaluate all distances from this new cluster to all other clusters, and update the matrix.
- Repeat until the distance matrix is reduced to a single element.

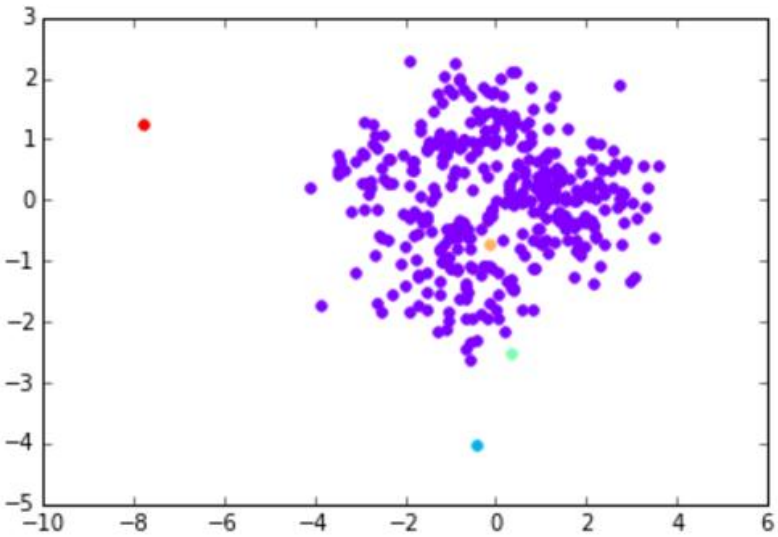
The algorithm can be further explained with the following flowchart:


data list of genes
N X N matrix where N is size of list of genes
Fill each matrix position with value euclidean distance from gene i to gene j and for i=i fill max + 1 out of all matrix values
Create a list of clusters and initialize with each gene id as a different cluster
loop till list of clusters size > required clusters
get the minimum value from the matrix
merge into= minimum (i,j) of minimum value
merge from= minimum (i,j) of minimum value
remove 'merge from' from the list of active clusters and append "merge from" to "merge into"
Now calculate minimum distance for each active cluster node (distance from "merge from", distance from "merge to") add into "merge into" column.
Matrix is updated as above step is This will be new minimum distance to each active cluster from combined cluster.

RESULT VISUALIZATION AND EVALUATION

For dataset "**cho.txt**" following is the result:

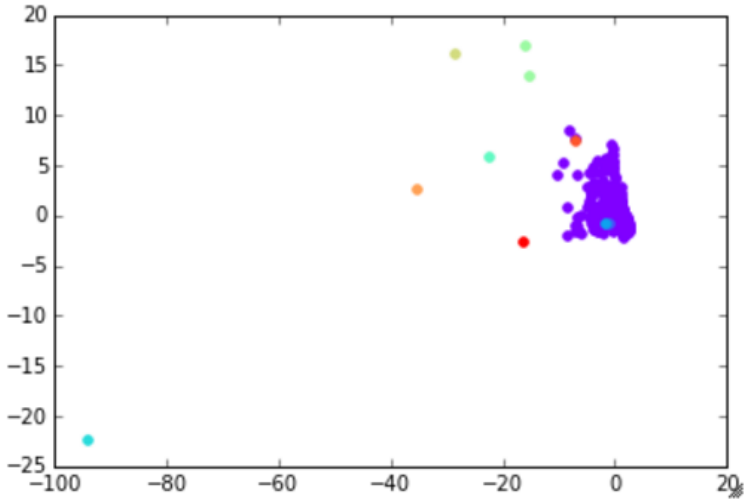
Total number of data-points in 5 clusters: 382, 1, 1, 1, 1



Jacard Coefficient: 0.228394977574
Rand: 0.240274906709

For dataset "**iyer.txt**" following is the result:

Total number of data-points in 10 clusters: 507, 1, 1, 1, 1, 2, 1, 1, 1, 1



Jacard Coefficient: 0.158243096966
Rand: 0.188286835597

Advantages of Hierarchical clustering:

- A good visualization using dendrogram is possible in this clustering.
- Does not need specific number of clusters to run the algorithm
- It can produce an ordering of the objects, which may be informative for data display.
- Smaller clusters are generated, which may be helpful for discovery.

Disadvantages of Hierarchical clustering:

- No provision can be made for a relocation of objects that may have been 'incorrectly' grouped at an early stage. The result should be examined closely to ensure it makes sense.
- Use of different distance metrics(Min, Max, Average) for measuring distances between clusters may generate different results. Performing multiple experiments and comparing the results is recommended to support the veracity of the original results.

K-means MR

ALGORITHM DESCRIPTION

class RandomMapper:
Generate random initial centroids.
Our algorithm uses a custom algorithm for random centroid selection, which ensures the centroids are selected with a degree of uniformity across the input order.

Custom Random Algorithm:

- Choose the first point.
- Choose the next point after skipping s number of data points.
- s is the seed provided to the algorithm.

seed constraints: $1 \leq s \leq n/k$

where n is the number of data points, and k is the number of clusters

class CentroidMapper Algorithm:
input: key<object>, value<text>

- Assign each data point to nearest cluster.
- Emit (nearest cluster id, data point)

output: key<IntWritable>, value<text>
data point is a vector of f features. the vector is converted to text and emitted.

class CentroidReducer Algorithm:
input: key<IntWritable>, value<iterable<text>>

- Calculate the median point of the list of points.
- Assign calculated median as the new centroid of corresponding centroid id.
- If centroid is changed, set flag for convergence true.

flag of convergence ensures kmeans iteration continues till any of the centroid is changed.

output: nothing written to file. Shared centroid list is maintained at master node. Less file I/O → Faster Execution

main() Algorithm:
Run RandomMapper

Loop below 2, till convergence flag is true and max iterations are not reached

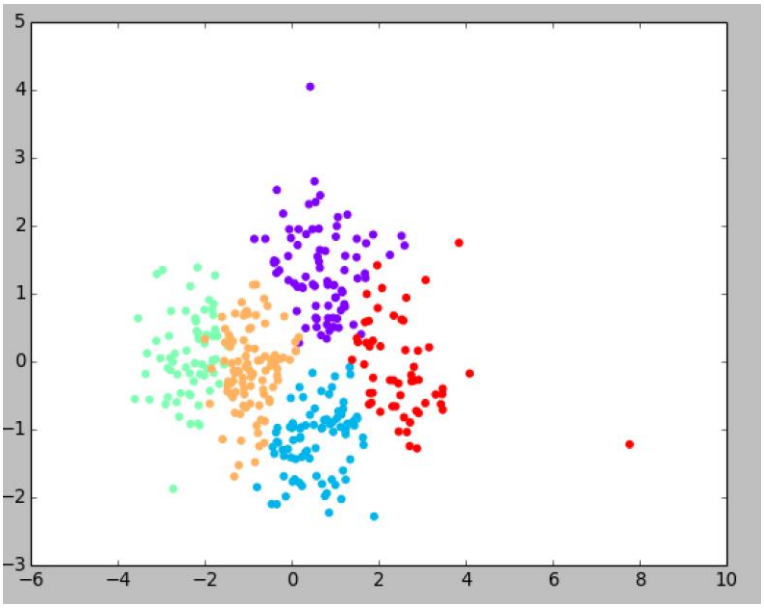
- Run CentroidMapper
- Run CentroidReducer

Print the centroid labels for all data points to file.

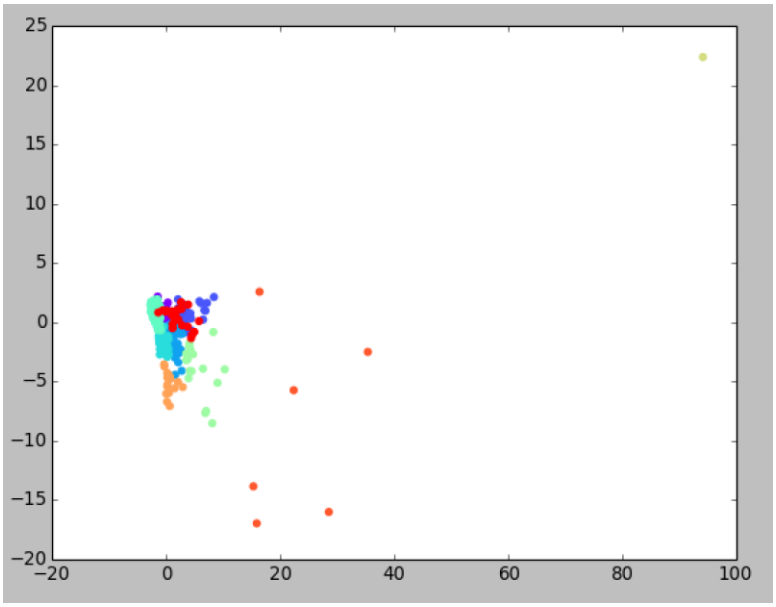
RESULT VISUALIZATION- kmeans Map Reduce

---

cho.txt



iyer.txt



RESULT EVALUATION - kmeans Map Reduce

---

cho.txt
Seed: 50
Clusters: 5
Time taken: ~23s
Iterations: 13
Jaccard: 0.336
Rand: 0.782

iyer.txt
Seed: 50
Clusters: 10
Time taken: ~50s
Iterations: 34
Jaccard: .35
Rand: .755

Steps Taken To Improve Efficiency:

Chose initial centroids spread across input data.
Motivation- The data in this project, cho.txt and iyer.txt, have data points in partial order of their clusters. Meaning, data points close together in the file are also close in the dimension space and have high probability of being in the same cluster.

Minimized file input output.
Motivation- File input output is a time consuming task. Final output is written to disk at the end of all iterations.

Convergence Flag
Instead of checking the state of all points, or centroids after every iteration, we keep track if a change was made in any of the reducer.
Explanation: Whenever a reducer recomputes a new centroid, if the new centroid is different from the old one, a flag is set to make sure next iteration takes place.
Benefit: Instead of having a separate loop construct in main, the comparisons are made in the reducers.

K-means

ALGORITHM DESCRIPTION

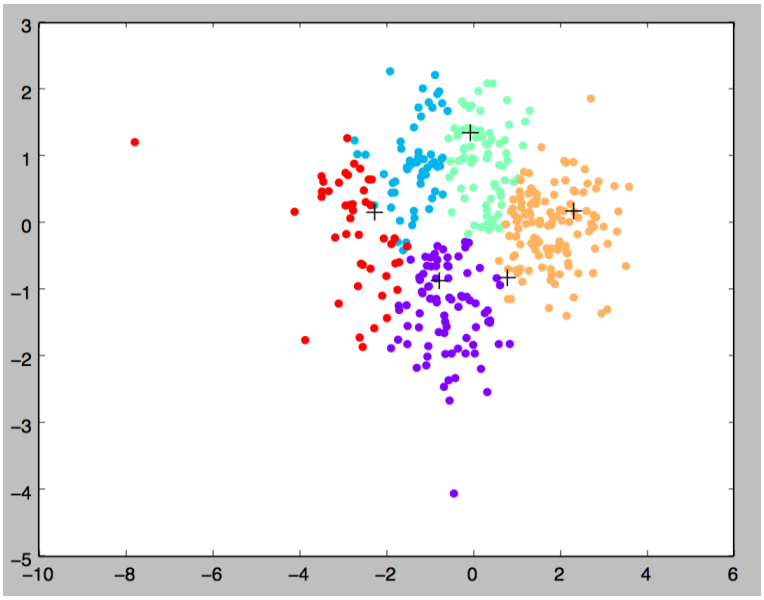
1. Randomly choose k centroids from dataset.
2. For each point calculate its Euclidean distance to each centroid.
3. Assign the point to the minimum distance centroid cluster.
4. Recompute the centroids by taking mean of data points in that cluster.
5. Repeat 2-5 steps till there is no change in centroids.

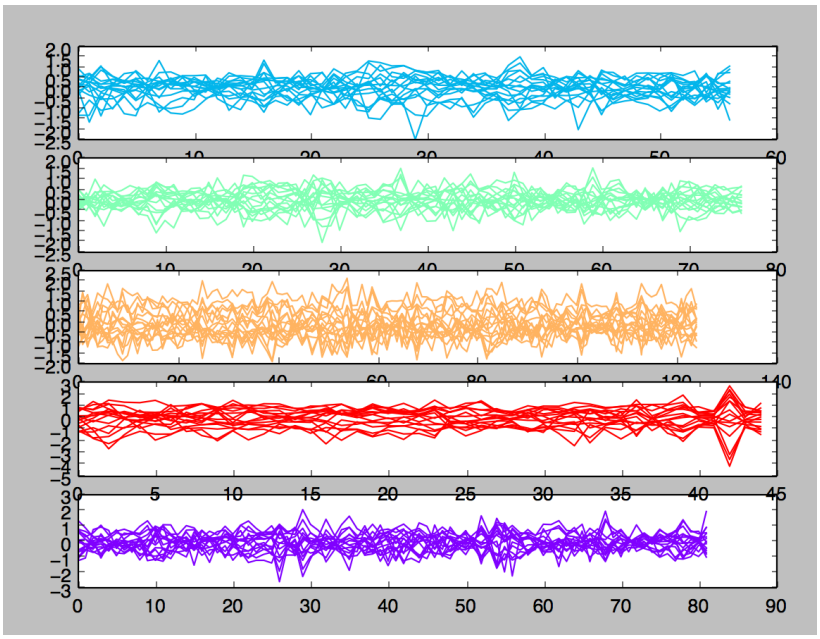
Initial centroid selection has a significant effect on the runtime and accuracy of k-means clustering. Kmeans++ offers a way to optimize this. In Kmeans++, initial centroids are selected based on a point's distance to other selected centroids. Further the distance, more the probability of it being selected.

K-means is generally run multiple times with different initial centroids, and the result with the minimum error is chosen.

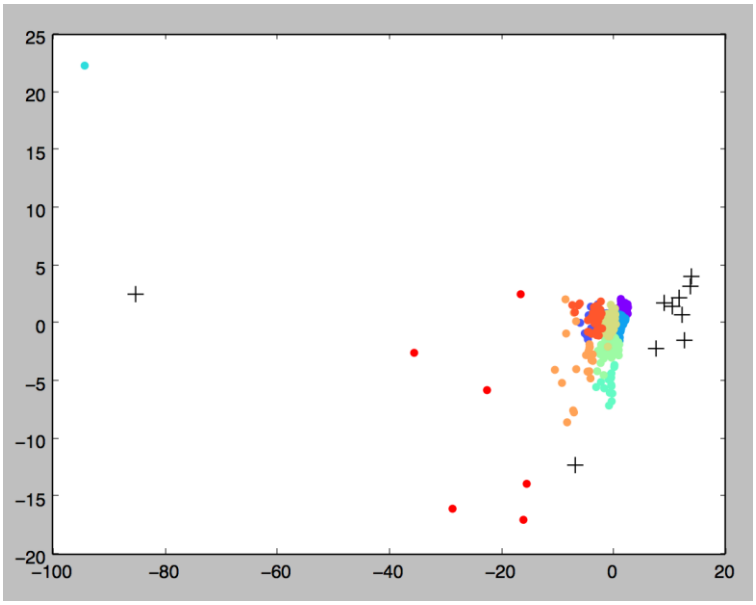
RESULT VISUALIZATION- kmeans

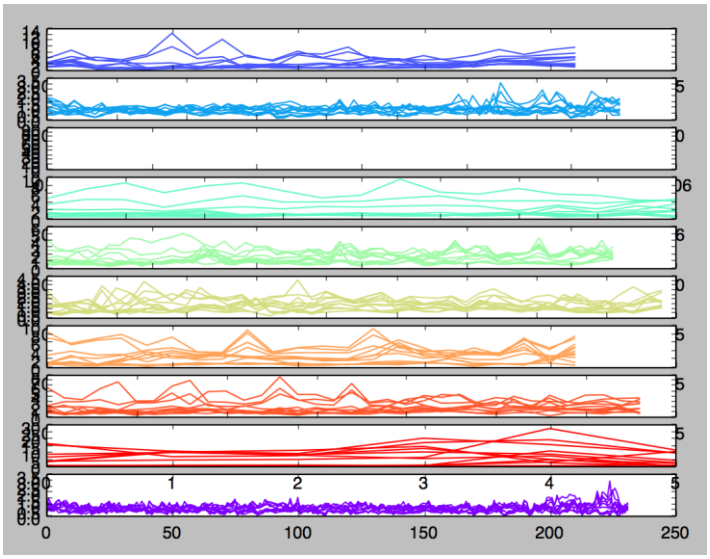
cho.txt





iyer.txt





RESULT EVALUATION - kmeans

---

cho.txt
Clusters: 5 (random)
Iterations: 13
Jaccard: ~0.40
Rand: ~0.80

iyer.txt
Clusters: 10 (random)
Iterations: 34
Jaccard: ~0.30
Rand: ~0.77

K-MEANS CHARACTERISTICS:

Advantages:

- K-means is simple to implement
- K-means works good when clusters are circular
- K-means is a fast algorithm
- Good for pre-clustering, i.e. K-means is often used before using other more expensive clustering algorithms

Disadvantages:

- Not good for complicated clusters, which are not simply circular. For eg, concentric circular clusters, intertwined clusters

Sensitive to initial cluster selection

Density based Clustering

ALGORITHM DESCRIPTION

For density based clustering we have used DBSCAN algorithm. In this method clustering is based on density such as density connected point. Each cluster has a considerable higher density of points than outside of the cluster. Basic idea is if an object *p* is density connected to *q*, then *p* and *q* belong to the same cluster. However, if an object is not density connected to any other object it is considered noise.

We have two global parameters:

- **Eps:** Maximum radius of the neighborhood
- **MinPts:** Minimum number of points in an Eps-neighborhood of that point

According to the parameters we have two types of data points in a cluster:

- **Core Object:** object with at least MinPts objects within a radius 'Eps-neighborhood'
- **Border Object:** object that on the border of a cluster

Steps of DBSCAN are given below:

- Arbitrary select a point p
- Retrieve all points density-reachable from p wrt Eps and MinPts.
- If p is a core point, a cluster is formed.
- If p is a border point, no points are density reachable from p and DBSCAN visits the next point of the database.
- Continue the process until all of the points have been processed.

Following is the pseudo code:

```
DBSCAN(D, eps, MinPts) {
   C = 0
   for each point P in dataset D {
      if P is visited
         continue next point
      mark P as visited
      NeighborPts = regionQuery(P, eps)
      if sizeof(NeighborPts) < MinPts
         mark P as NOISE
      else {
         C = next cluster
         expandCluster(P, NeighborPts, C, eps, MinPts)
      }
```

```
    }
}


expandCluster(P, NeighborPts, C, eps, MinPts) {
    add P to cluster C
    for each point P' in NeighborPts {
        if P' is not visited {
            mark P' as visited
            NeighborPts' = regionQuery(P', eps)
            if sizeof(NeighborPts') >= MinPts
                NeighborPts = NeighborPts joined with NeighborPts'
        }
        if P' is not yet member of any cluster
            add P' to cluster C
    }
}

regionQuery(P, eps)
    return all points within P's eps-neighborhood (including P)
```

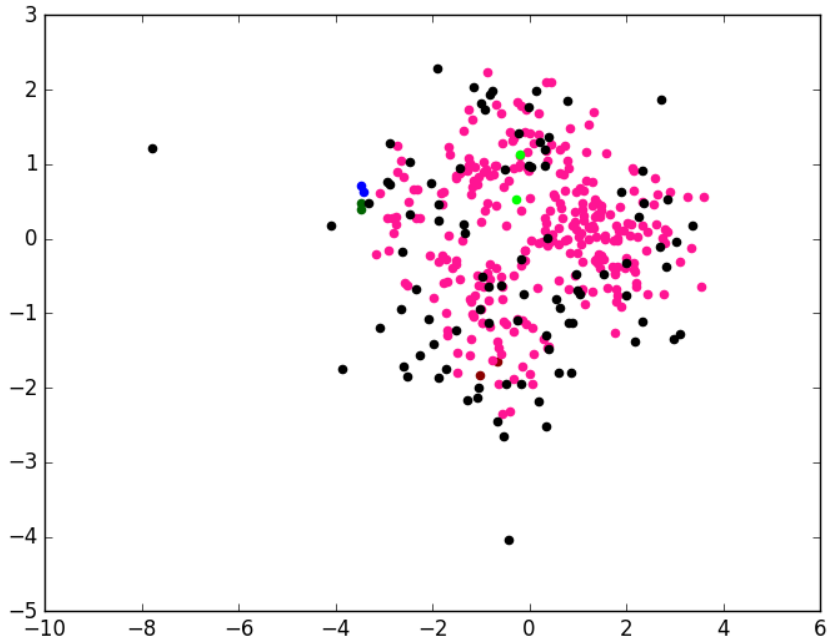
RESULT VISUALIZATION AND EVALUATION

For the data sets given after some trials we have taken the values of parameters to be:

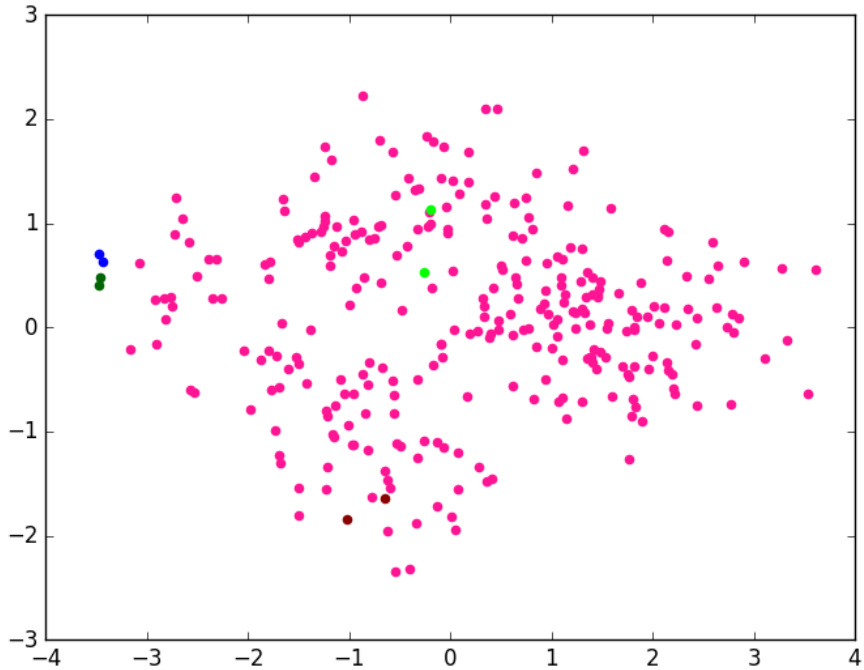
- Eps: 1.25
- Minpts: 2

And the clusters are as below:

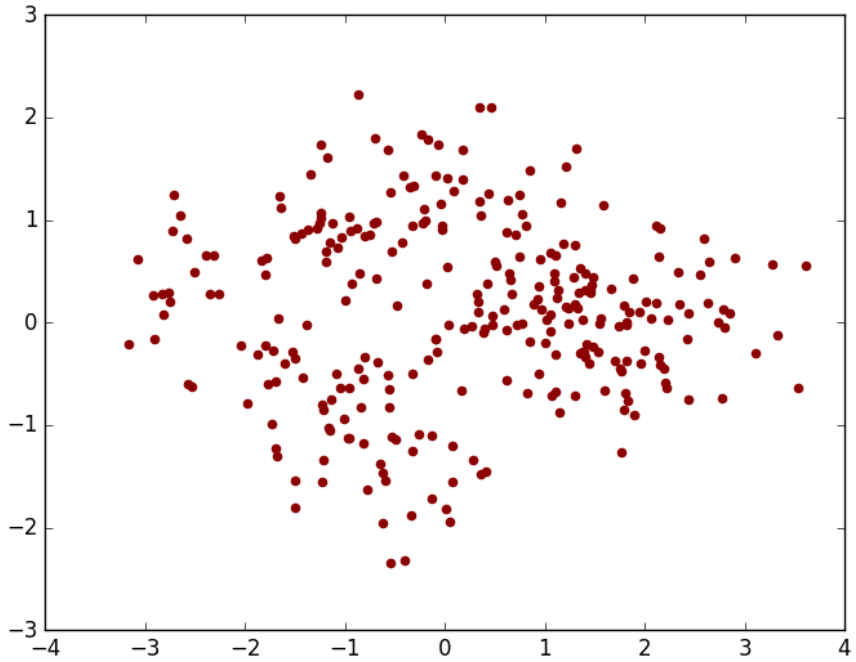
For “**cho.txt**” data set:



With outliers



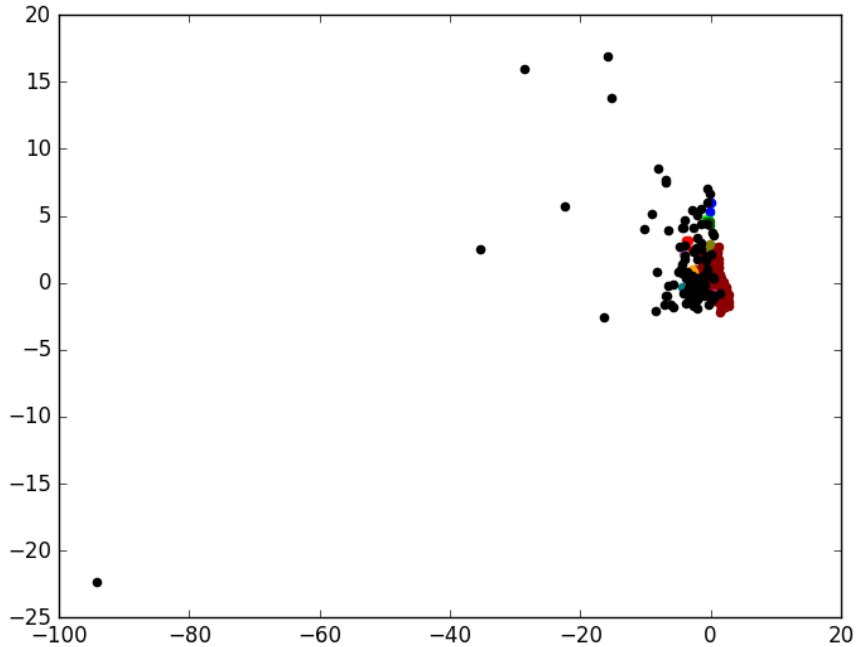
Without outliers



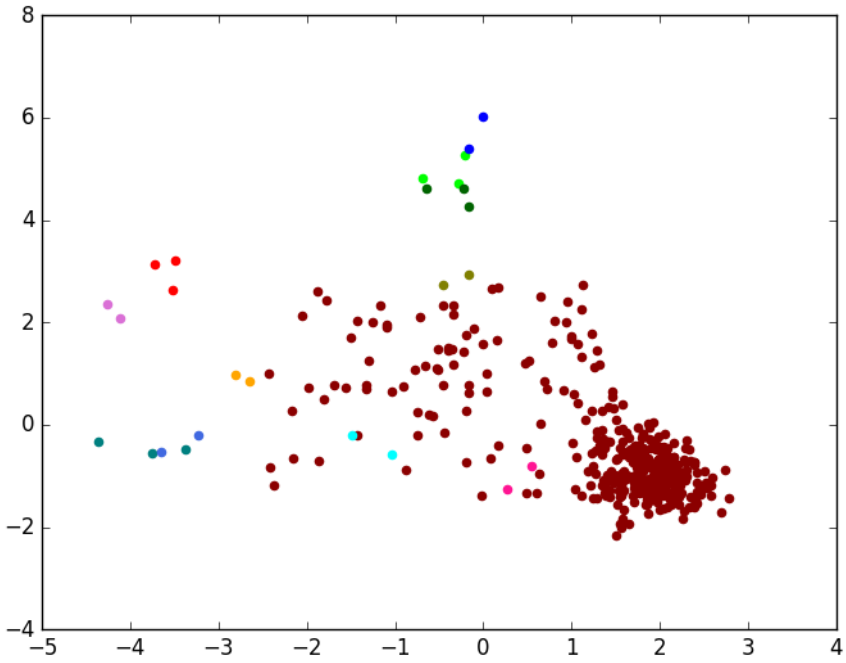
With Minpts.=3

Jaccard coefficient = 0.22017578125
Rand = 0.464052726248

For “**iyer.txt**” dataset:



With outliers



Without outliers

Jaccard coefficient = 0.216727311836
Rand = 0.50954210611

Advantages of DBSCAN are given below:

- DBSCAN does not require you to know the number of clusters in the data a priori, as opposed to k-means.
- DBSCAN can find arbitrarily shaped clusters. It can even find clusters completely surrounded by (but not connected to) a different cluster.
- DBSCAN has a notion of noise.

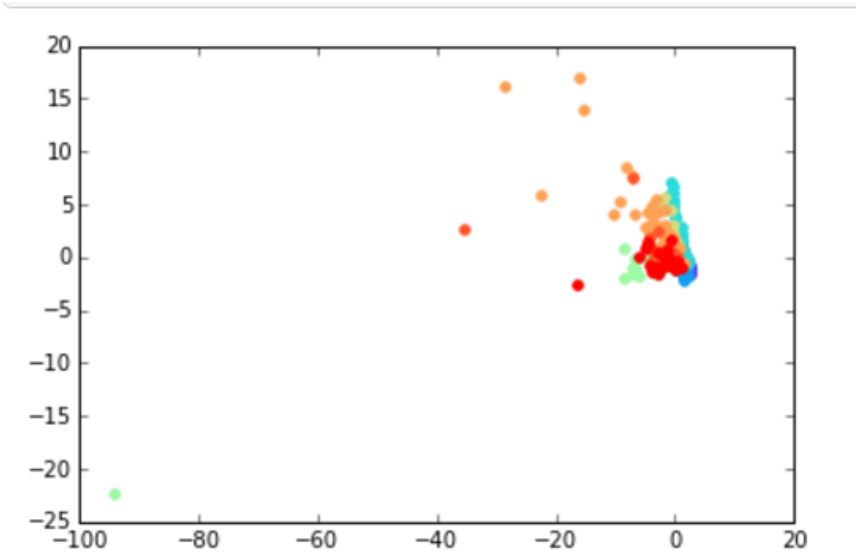
Disadvantages of DBSCAN are given below:

- DBSCAN can only result in a good clustering as good as its distance measure is. The most common distance metric used is the euclidean distance measure. Especially for high-dimensional data, rendering it hard to find an appropriate value for epsilon.
- DBSCAN cannot cluster data sets well with large differences in densities, since the minPts-epsilon combination cannot be chosen appropriately for all clusters then.
- The Algorithm is not partitionable for multi-processor systems.

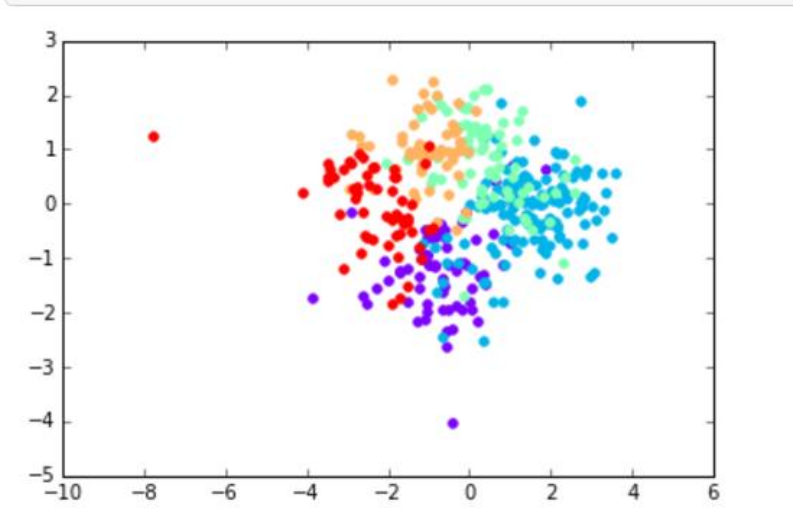
COMPARISON AMONG DIFFERENT ALGORITHMS, FINDINGS AND SELF LEARNING.

First let us look at PCA of data from the truth labels.

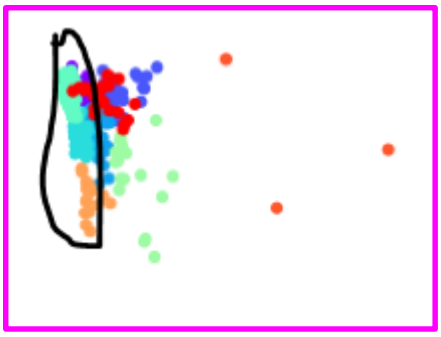
Iyer dataset:



Cho dataset:



We can see that in case of Cho the original labels have kind of spherical shaped clusters. For this purpose the k means approach is bound to give better results.
But if we look at PCA of Iyer data it has mixed of complex shapes and spherical shaped original clusters. So for all the clusters not in spherical kind of shape our K means has further broken these to give clusters in rounded shape as we can see in the screenshot below.



Now the Agglomerative Hierarchical Clustering which is more efficient when the actual clusters are far separated in plane . Which is not the case in any of the cho and Iyer. So in our case outliers(single cluster unit) are the ones to merge into big Hierarchical Cluster in the end. Leveraging these ideas and learning what we plan to do in the future for data mining projects is in the flowchart below.