CSE 601: Data Mining and Bioinformatics

Project 2: Clustering Algorithms

Team Members:

Name	UBIT Name	Person Number
Sivasathya Pradha Balamurugan	sivasath	50169073
Dnyanada Arjunwadkar	dnyanada	50169676
Ajinkya Thorve	ajinkyay	50167854

Task I:

1.1 Algorithm Implementation Details:

1. K-means

i. Ask for the number of clusters

ii. Pick random points as initial clusters

iii. Assign each point to a cluster based on which cluster it is closest to.

iv. After all points are assigned to clusters, recompute the cluster centroids.

v. Repeat steps (iii) and (iv) recursively until there is no difference between the clusters formed in the current round versus the previous round.

2. Hierarchical

i. Read the input data from the csv file and form a data matrix

ii. Calculate the distance matrix. Distance calculated is the Euclidean distance. Distance indicates similarity i.e. lower distance indicates more similarity.

iii. All the diagonal entries in the distance matrix are set to zero.

iv. Look for entries which have distance less than the minimum distance in the lower triangular part of the distance matrix.

v. Form a cluster of these entries. Remove the corresponding rows and columns by setting them to infinity.

vi. update a new row and new column for the new cluster

vii. Repeat steps (iv), (v) and (vi) until the entire distance matrix is filled with infinity for all the points.

viii. Visualize the clustering formed using a dendogram using dendroscope.

3. Density-based

i. Read the input data from the csv file and form a data matrix

ii. Calculate the distance matrix. Distance calculated is the Euclidean distance. Distance indicates similarity i.e. lower distance indicates more similarity.

iii. All the diagonal entries in the distance matrix are set to zero.

iv. Specify different values for e and the min points.

v. Start with a random point and look for the neighbour points (the points within the e distance).

vi. If the number of neighbour points is more than min points, start the process by marking it as visited otherwise mark as outlier which can be considered for later clustering.

vii. If the neighbour points belong to a cluster then the points that lie within e distance of neighbour points also belong to the cluster.

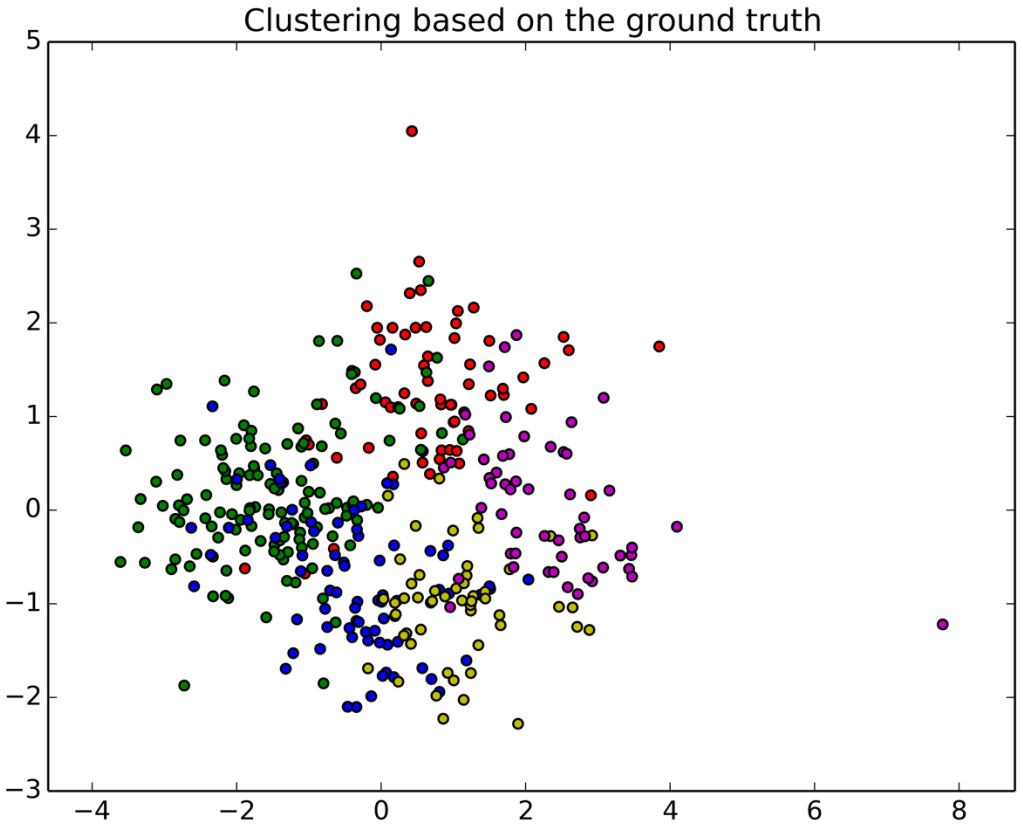
viii. Continue until all the points are marked as visited.

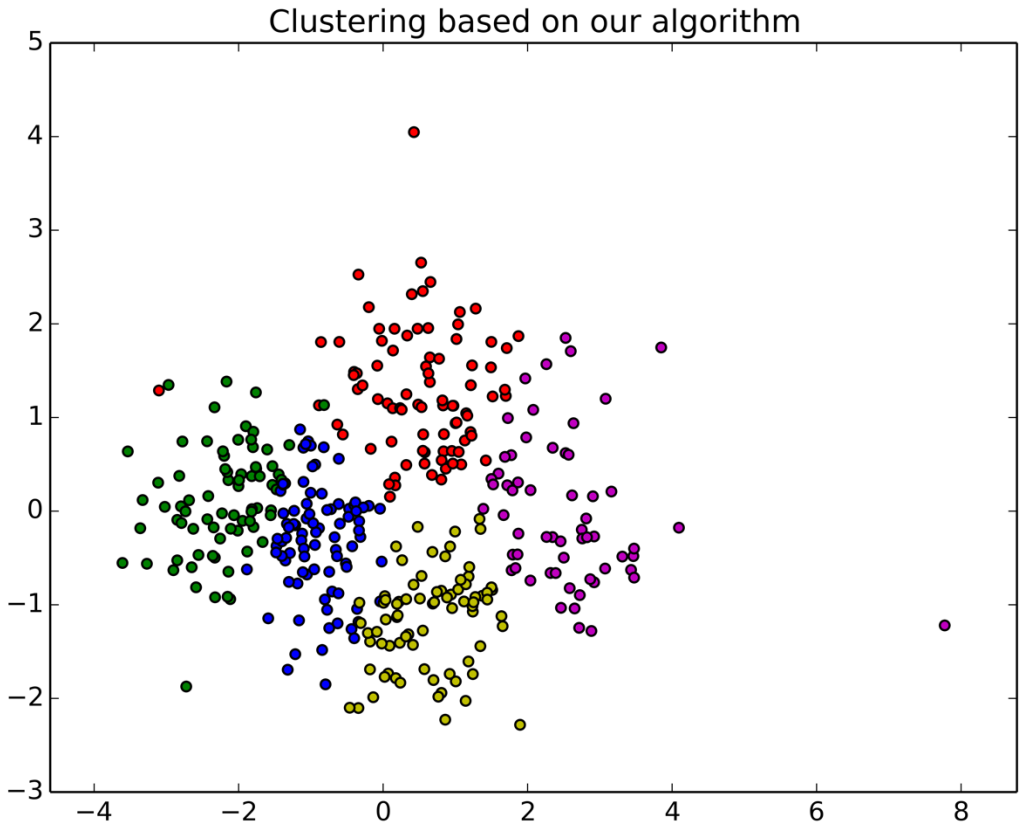
1.2 Result visualization:

For visualization, first we need to reduce the dimensionality of the given data and project it to a 2-D space. This can be done if we implement PCA on the given data. For implementing PCA we have used an existing Python library provided by scikit-learn (sklearn.decomposition.PCA). After executing our clustering algorithms, clusters are assigned to each gene. Then we plot the genes in the 2-D space and represent each assigned cluster using a different colour.

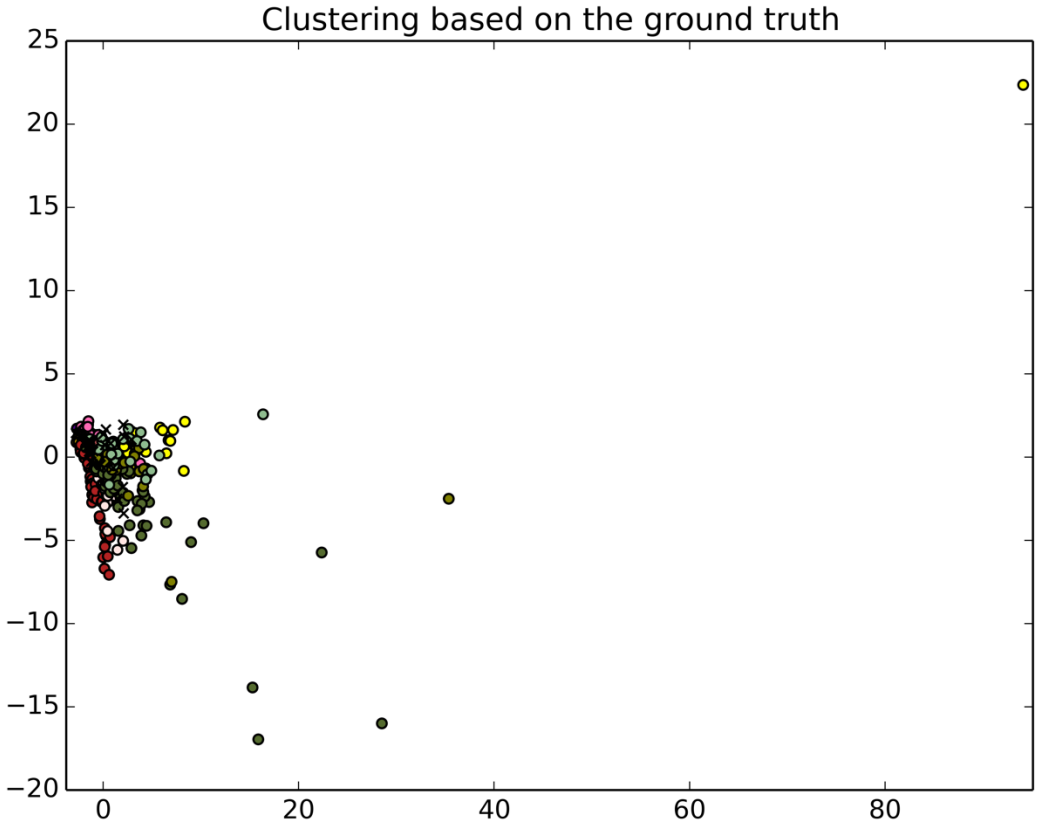
1. K-means

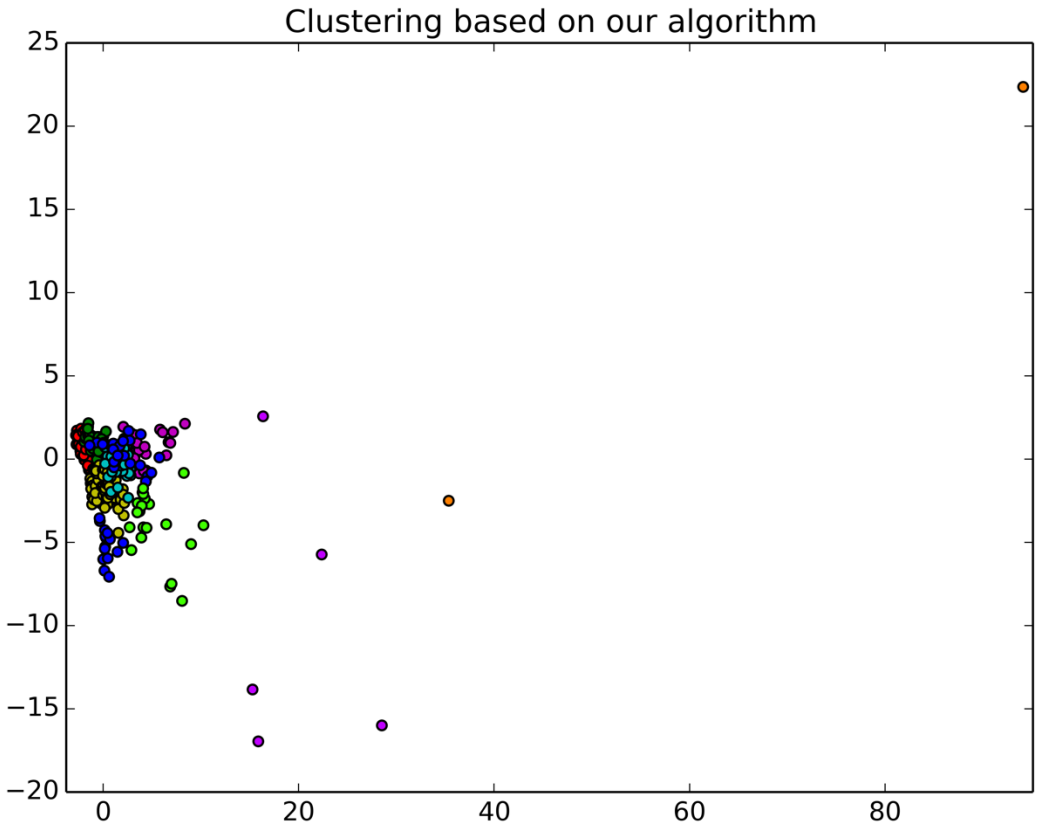
For cho dataset:




Clustering based on our algorithm
5
4
3
2
1
0
−1
−2
−3
−4
−2
0
2
4
6
8

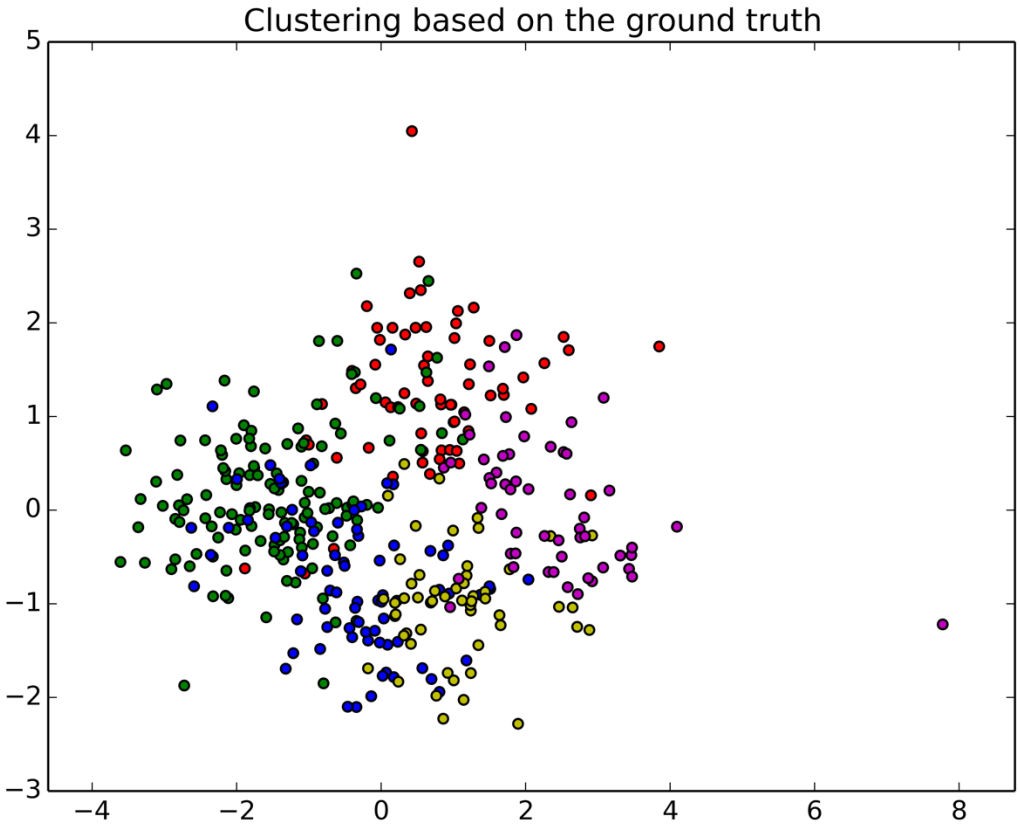
For iyer dataset:

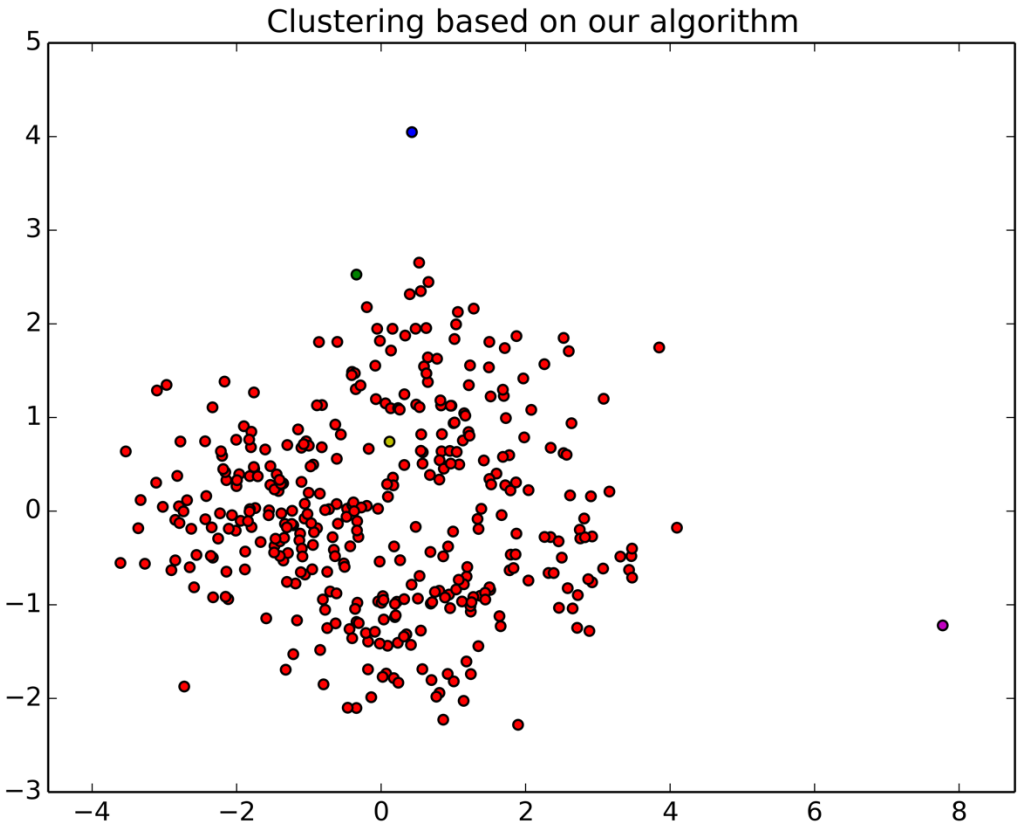


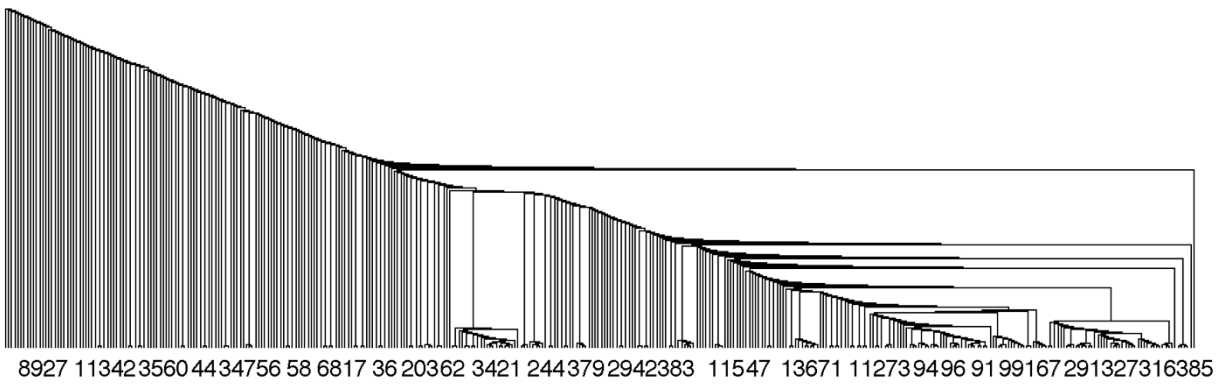

Clustering based on our algorithm
25
20
15
10
5
0
−5
−10
−15
−20
0
20
40
60
80

2. Hierarchical

For cho dataset:

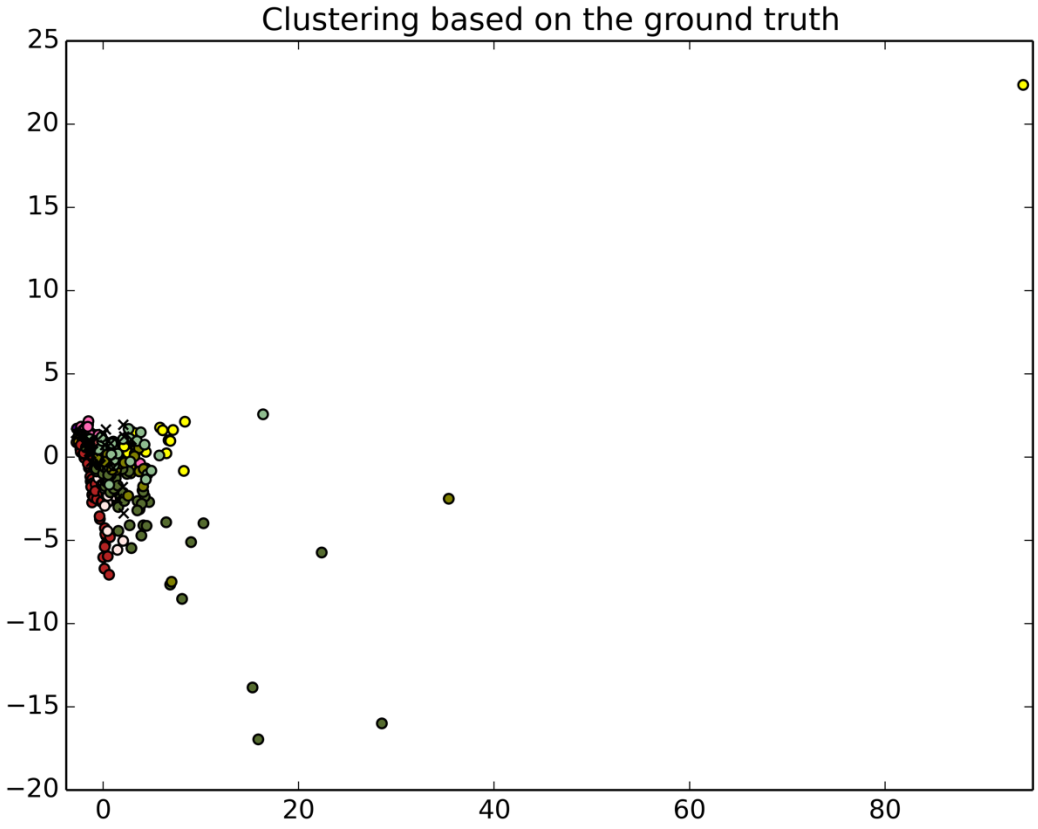


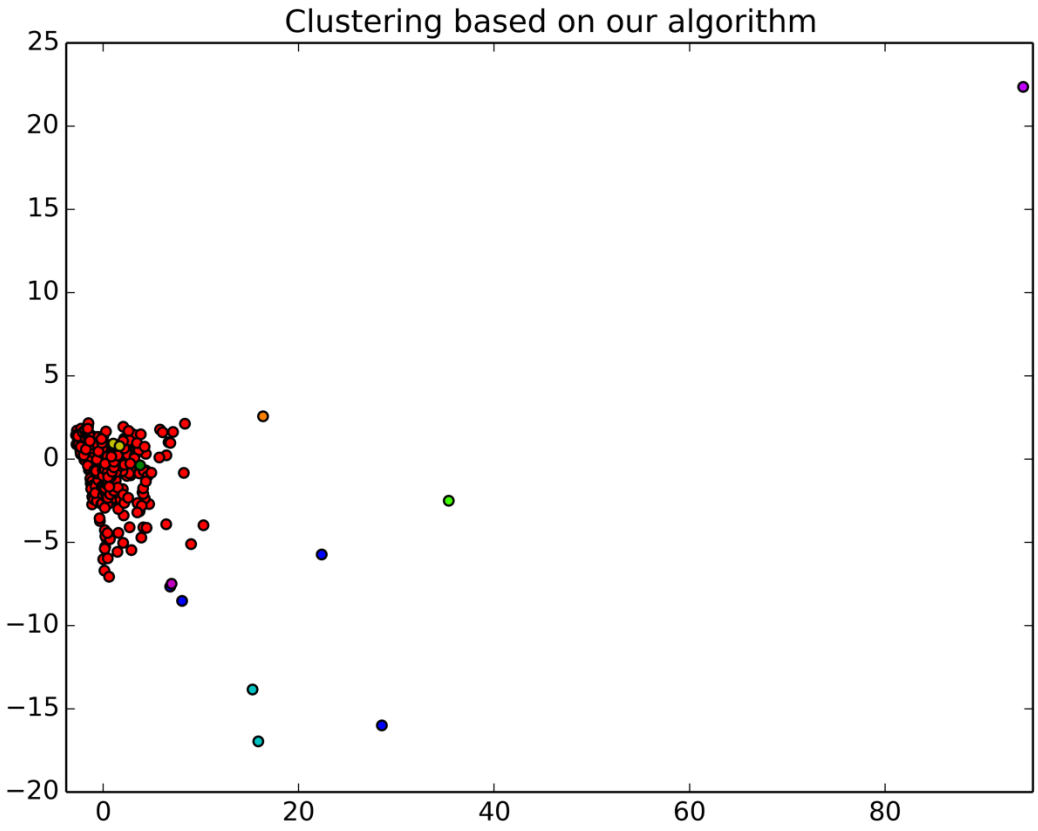


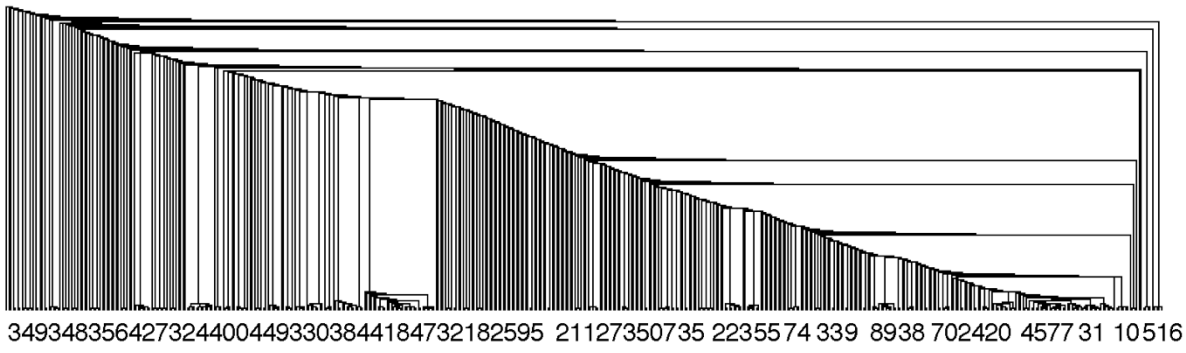


Dendogram for cho dataset

For iyer dataset:



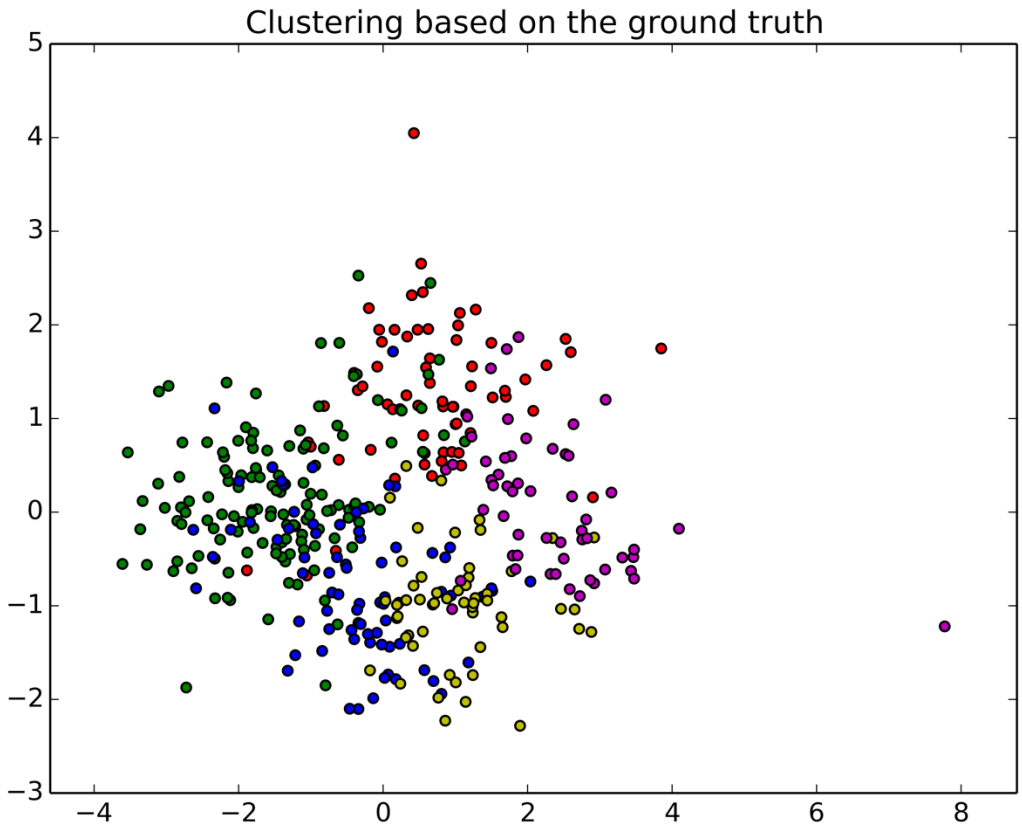


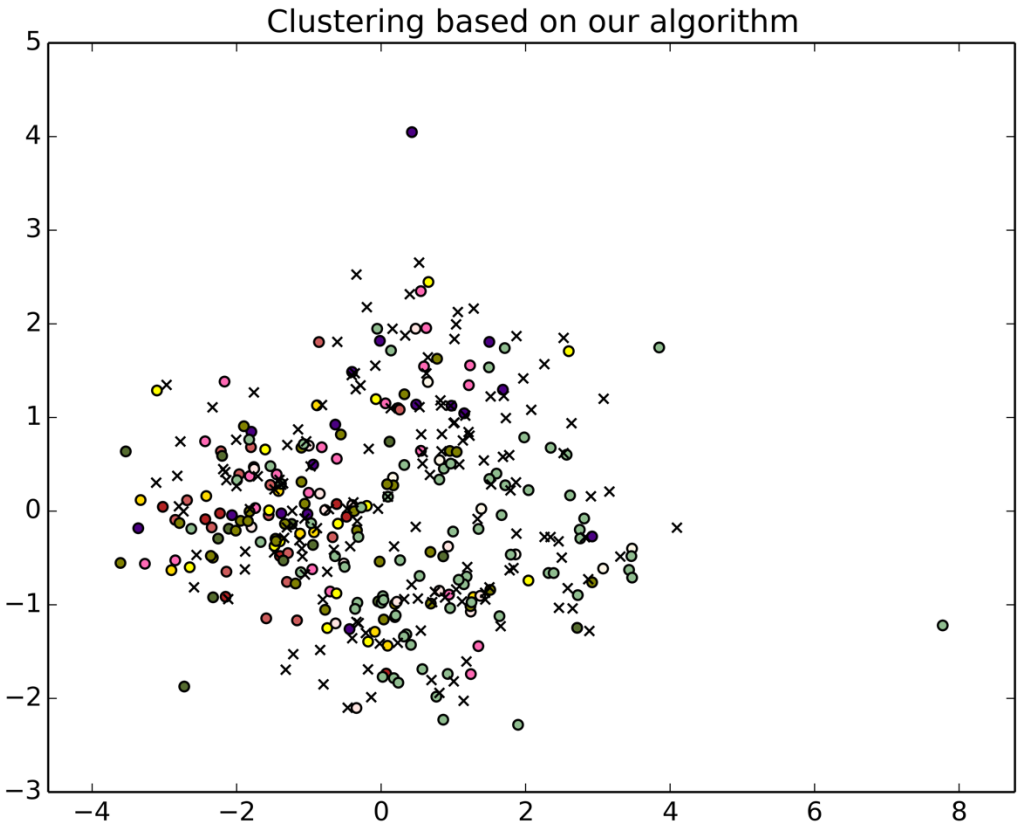


Dendogram for cho dataset

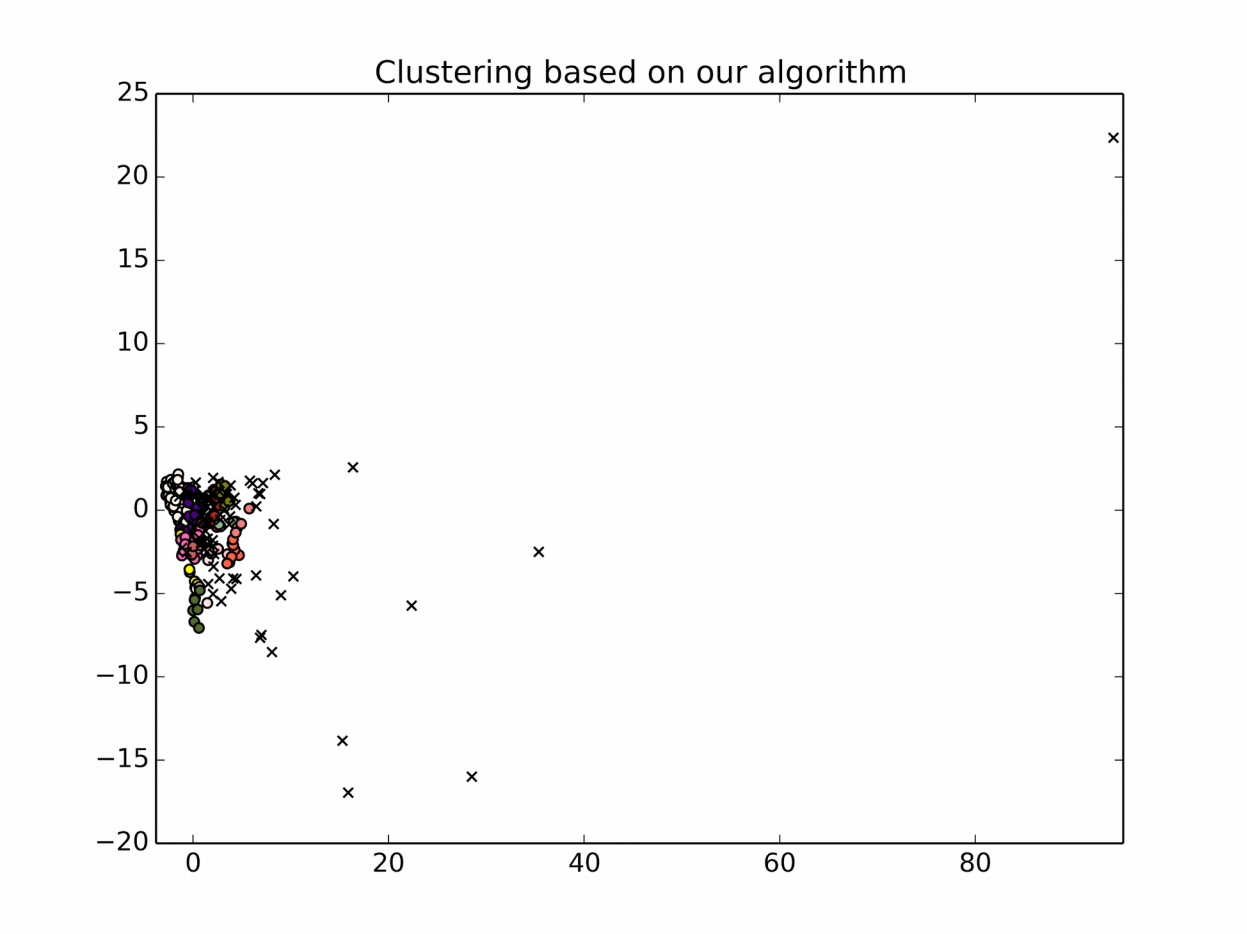
3. Density-based

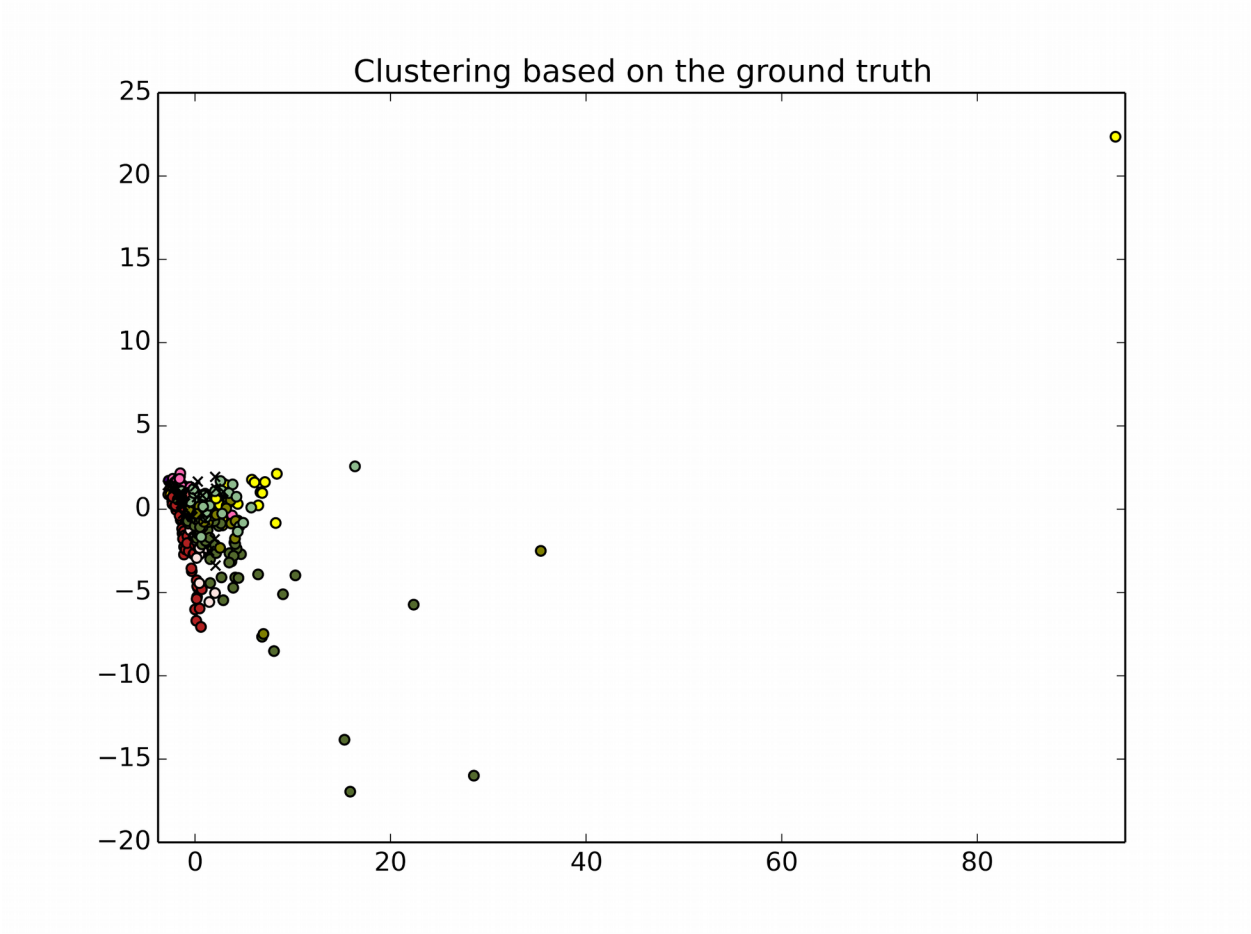
For cho dataset:




Clustering based on our algorithm
5
4
3
2
1
0
−1
−2
−3
−4
−2
0
2
4
6
8

For iyer dataset:




Clustering based on the ground truth
25
20
15
10
5
0
−5
−10
−15
−20
0
20
40
60
80

1.3 Result validation using external index:

We need to perform validation to test the correctness or goodness of our clustering algorithms. After executing all the three clustering algorithms, we validate our clustering results by using external index. For this we use a Python program which computes a matrix containing the P_{ij} values, C_{ij} values, different values like M_{11}, M_{00}, M_{10}, M_{01} and finally the Jaccard coefficient. We calculate the Jaccard coefficient as:

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

Algorithm	**Jaccard**	
	cho	**iyer**
K-means	0.336467566896 (#clusters:5)	0.356451769477 (#clusters:10)
Hierarchical	0.228394977574	0.158802423387
Density-based	eps=2, minPts=4 Ans:**0.175** eps=3, minPts=3 Ans: **0.269**	eps=3, minPts=3 Ans**: 0.292** eps=2, minPts=3 Ans: **0.317**

Algorithm	Rand	
	cho	iyer
K-means	0.784974093264	0.776780937487
Hierarchical	0.240274906709	0.197430496579

1.4 Result evaluation, time complexity, pros and cons:

Result evaluation: Looking at the results, we can say that K-means works the best among the three clustering algorithms on these given datasets. This is evident from the visualization results and external index values. Density-based performs moderately well, while hierarchical performs the worst on these datasets.

Time complexity: K-means is linear in the number of data objects i.e. $O(n)$, where n is the number of data objects. Hierarchical clustering algorithms is quadratic i.e. $O(n^2)$. Thus, K-means is more suitable for bigger datasets, and Hierarchical is more suitable for smaller datasets.

Pros and cons:

1. K-means

Pros: More efficient run-time wise, works well when the shape of clusters is spherical or circular, results may not be repeatable and lack consistency.

Cons: Produces a single partitioning, needs the number of clusters to be specified, performs badly if the clusters have non-spherical or non-circular shapes.

2. Hierarchical

Pros: Does not need the number of clusters as input, can give different partitions depending on the level of resolution we need.

Cons: Hierarchical clustering can be slow on very large datasets.

3. Density-based

Pros: The number of clusters need not be known in advance, it is not affected much by noise, complexity is better than the others, and it can work on arbitrary dataset

Cons: Not suitable for high dimensional data set, setting the correct parameters is difficult as it has to be done by trial and error.

Task II:

MapReduce K-means

2.1 Algorithm Description:

i. The non-parallel K-means is executed in a distributed file system making it a parallel k-means.

ii. This part of the algorithm involves two functions Mapper and Reducer.

Mapper:

i. Read the input file and store it in a NumPy array

ii. Calculate the distance matrix which gives the distance between points

iii. Pick the number of random centroids as mentioned and calculate the distance of each data point from the centroids and allocate to the one with the minimum distance

iv. Write it as key-value pairs and print it as input to reducer

Reducer:

i. Read the key-value pair from the mapper's output and compute the new centroid for each cluster by taking the average of attributes of all nodes in the cluster

ii. Compare the current cluster with the previous cluster and check if it's similar

iii. If similar end the iteration or call the mapper again by giving the Reducer function's output

Non-parallel K-means is linear in the number of data objects i.e. $O(n)$, where n is the number of calculation involved in calculating the similarity of the data point to the centroids to find the closest one. Parallel k-means also involves this calculation but distance has been pre-calculated and stored in a matrix to avoid recomputation, and also the input file is split into smaller files and processed simultaneously which decreases the complexity and the running time. Iterators and hashmaps are used to read the input file and store it to reduce the running time.

References:

1. http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

2. http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html#sphx-glr-auto-examples-cluster-plot-kmeans-digits-py

3. https://www.quora.com/What-are-the-pros-and-cons-of-kmeans-vs-hierarchical-clustering