CSE-601: Data Mining and Bioinformatics

Project 2 Report
Clustering Algorithms

Junsong Huang(50292448)
Yi Jin(50295580)
Yin Gong(50290271)

October 31, 2019

1. K-means

1.1 Abstract

K-Means clustering is very popular. First, we need to decide how many clusters we want to get (k) and initialize k initial cluster centers. Then, for each data point, we calculate the distance from each points to the k cluster centers and find the which cluster center is the most close to the data point, then we can assign this data point to the nearest cluster center. After we finish assigning all the data points to a specific cluster, we can recalculate the cluster center by taking the average of the data points belonging to the same cluster. We will carry out the process again and again until the centroid begin to converge.In the implementation on the dataset cho.txt and iyer.txt, I randomly select the initial center

1.2 Algorithm

1. Read the datasets
2. Predefine the number of clusters K
3. Specify the initial cluster centers
4. **Repeat**
5. For each object x_i, calculate the distances between x_i and the K centroids.
6. Reassign x_i to the cluster whose centroid is the closest to x_i.
7. Update the cluster centroids based on current assignment
8. **Until** The result converges

1.3 Implementation

Fist I will decide how many clusters, we will use k to denote the number. the I randomly select k clustering centers. for each data point, we calculate the distance from each points to the k cluster centers and find the which cluster center is the most close to the data point, then I can assign this data point to the nearest cluster center. After I finish assigning all the data points to a specific cluster, my code recalculate the cluster center by taking the average of the data points belonging to the same cluster. I will carry out the process again and again until the centroid begin to converge.

1. $incidence\_mat\_gen$: The function take the result label as the only argument and output an incidence matrix based on that.
2. ja_rand_cal: Calculate rand and jaccard results. Ground truths and my results are the first and second argument respectively.

1.4 Result

File name	Rand	Jaccard
cho.txt	0.8012	0.4047
iyer.txt	0.7629	0.3524

Pros for K-Means Clustering:

1. It's easy to implement
2. The K-Means can work well on round shape clusters

Cons for K-Means Clustering:

1. K-Means is sensitive to the initial centroids
2. K-Means performs poorly on irregular shape clusters
3. Sensitivity to noise and outliers

2. Hierarchical Agglomerative Clustering with Min Approach

2.1 Abstract

The hierarchical clustering algorithm is one of the most basic clustering methods. But the idea of it is of vital importance. The main idea of Hierarchical clustering is to construct a dendrogram which is a tree that shows how clusters are merged and split hierarchically. There are two approaches to form this dendrogram, naming agglomerative approach and divisive approach. As its name suggested, the agglomerative approach starts to construct the dendrogram by merging each individual project(cluster) finally into a single cluster. While the divisive approach starts with a single cluster with all objects in it and then split it into two sub clusters until each leaf cluster contains only one object.

There are different ways to merge two clusters in hierarchical clustering. Clusters are merges by their inter-cluster distance which is represented by the distance of the pair of data objects belonging to different clusters. The most common methods are MIN, MAX, Group Average and Distance between centroids. In our case, only the closest pair of data objects from different clusters need to be considered due to MIN approach.

2.2 Algorithm

1. Read the datasets
2. Let each data point be a cluster and construct the distance matrix
3. **Repeat**
4. Merge the two closest clusters

5. Update the distance matrix

6. **Until** only a single cluster remains

2.3 Implementations

The python code takes two arguments. The first one is our dataset of txt format and the second argument is the number of cluster.

1. $dist\_cal$: it took the dataset as the only argument and output the distance matrix. And the distances are all Euclidean distance

2. min_find: The first argument is the dataset and the second argument is the index list to which we need to assign the index with minimum value. The function will return the index of the minimum value in that dataset.

3. $update\_matrix$: The main idea of this function is to merge the clusters. It will take three arguments. The first argument is the updated distance matrix. The second argument is the min index obtained from min_find function. The last one is the merged labels.

4. $merge\_list$: Take the merged labels as the only argument. Put all objects in one list.

5. $incidence\_mat\_gen$: The function take the result label as the only argument and output an incidence matrix based on that.

6. ja_rand_cal: Calculate rand and jaccard results. Ground truths and my results are the first and second argument respectively.

2.4 Results

File name	Rand	Jaccard
cho.txt	0.2403	0.2284
iyer.txt	0.1883	0.1582

Pros for Hierarchical Clustering:

1. The implementation is simple.

2. There's no need to assume any particular number of clusters. Because the desired number of clusters can be obtained by cutting the dendrogram at proper level.

3. The result may correspond to meaningful taxonomies. Especially useful for shopping websites

Cons for Hierarchical Clustering:

1. The algorithm is time consuming. The matrix update costs $\mathrm{O}(N^2)$. And there are N steps to merge all clusters. Thus the total time cost is $\mathrm{O}(N^3)$

2. The combined two clusters can't be undone.
3. No objective function is directly minimized
4. Sensitivity to noise and outliers
5. Difficulty handling different sized clusters and irregular shapes
6. Breaking large clusters

3. DBSCAN

3.1 Abstract

DBSCAN(Dendity-Based Spatial Clustering of Applications with Noise) is a typical density clustering algorithm. K-Means is generally only suitable for clustering of convex sample sets. In contrast, DBSCAN can be applied to both convex samples sets and non-convex sample sets.

3.2 Algorith

1. Read the datasets
2. **For** Each point o ∈ Dataset **Do**
3. **If** o is not yet classified **then**
4. **If** o is a core-object **then**
5. collect all objects density-reachable from o
6. and assign them to a new cluster
7. **else**
8. assign o to NOISE

3.3 Implementations

When I implement this algorithm, I first build a unvisited list which include all the data point and a null visited list. Then , I randomly choose the unvisited point P, get a list of its neighbors which has a distance less than from P. if the number of the list is less than the MinPts, I will mark the P as -1(NOISE). If the number >= MinPts , I will arrange a new cluster C for the point P. Then , my code traverses all the neighbor points (P') from P(dis<). If P' is not visited, we will mark P as visited and query the P' neighbors. If P' 's the number of neighbors >= MinPts, we can append P' 's neighbors to the P's neighbors. And if P"s cluster is belonging to -1, we will add the the P' to cluster C.

$find_neighbor$: Find the neighbor of a specific point in distance ϵ, and return the list of all of the neighbors.

$incidence\_mat\_gen$: The function take the result label as the only argument and output an incidence matrix based on that.

ja_rand_cal: Calculate rand and jaccard results. Ground truths and my results are the first and second argument respectively.

3.4 Result

File name	Rand	Jaccard
cho.txt	0.5444	0.2045
iyer.txt	0.6526	0.2836

Pros for DBSCAN Clustering:

1. DBSCAN can handle different shapes and sizes cluster.
2. DBSCAN can resistant to the dataset with noise

Cons for DBSCAN Clustering:

1. DBSCAN can't handle data with varying densities.
2. The parameters for DBSCAN is hard to determine.

4. Gaussian Mixture Model

4.1 Abstract

The algorithm of Gaussian mixture model is based on the probabilistic model. According to the datasets, it is assumed that the datasets are generated from a mixture of a provided number of Gaussian distributions with specific parameters.
The algorithm of gaussian mixture model implements the Expectation-Maximization algorithm.

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{(2\pi|\Sigma|)^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$$

$$L(\mu,\Sigma) = \sum_{i=1}^{n} \ln N\left(x_i|\mu,\Sigma\right) = \sum_{i=1}^{n}\left(-\frac{1}{2}\left(x_i-\mu\right)^T\Sigma^{-1}\left(x_i-\mu\right)\right) - \pi\ln|\Sigma|\Bigg)$$

In the E-step, for given parameter values we can compute the expected values of the latent variables.

$$r_{ik} \equiv E(z_{ik})$$
$$= \frac{\pi_k\mathcal{N}(x_i|u_k,\Sigma_k)}{\sum_{k=1}^{K}\pi_k\mathcal{N}(x_i|u_k,\Sigma_k)}$$

In the M-step, we maximize the expected complete log likelihood. We iterate E-step and M-step until the log likelihood of data does not increase any more.

$$\pi_k = \frac{\sum_i r_{ik}}{n} \qquad \mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}$$
$$\Sigma_k = \frac{\sum_i r_{ik} \left(x_i - \mu_k\right) \left(x_i - \mu_k\right)^T}{\sum_i r_{ik}}$$

In this way, based on the optimized parameters, we can compute posterior probability of membership and assign data points to each cluster.

4.2 Algorithm

1. Read the datasets
2. Initiate the parameters of GMM. For means in our case, we use K-means to get the initial value. For covariance in our case, we use diagonal matrix to get the initial value.
3. **Repeat**
4. Calculate the probability of corresponding gaussian models' parameters.
5. Implement the E-step to get posterior probability.
6. Implement the M-step to optimize parameters.
7. **Until** the parameters converge to certain value.

4.3 Implementations

The python code takes two arguments. The first one is our dataset of txt format and the second argument is the number of clusters.

1. *gmm_init*: It initiates the parameters of GMM.
2. *gaussian*: It calculates the probability of corresponding gaussian models' parameters.
3. *e_step*: It implements the E-step to get posterior probability.
4. *m_step*: It implements the M-step to optimize parameters.
5. *incidence_mat_gen*: The function take the result label as the only argument and output an incidence matrix based on that.
6. *ja_rand_cal*: Calculate rand and jaccard results. Ground truths and my results are the first and second argument respectively.

4.4 Result

File name	Rand	Jaccard
cho.txt	0.7917	0.4003
iyer.txt	0.6932	0.2977

Pros for GMM Clustering:

1. It can handle clusters with varying sizes, variance.
2. It can give probabilistic cluster assignments.
3. It has probabilistic interpretation.
4. It has higher Rand index and Jaccard coefficient compared to other clustering.

Cons for GMM Clustering:

1. The GMM clustering is the most time consuming compared to other clustering algorithms.
2. The initial parameters of GMM influence the performance of clustering a lot. When initiating the inappropriate parameters for the means and covariance, it always get the wrong clustering result after too many iterations. It means that GMM clustering algorithm really relies on the initial parameters very much.

5. Spectral Clustering

5.1 Abstract

The main idea of spectral clustering is to use similarity graphs to encode local neighborhood information. Then each data points can be considered as vertices of graph. The goal is to connect points and assign similar data points to the same group. Inevitably, points with low similarity may be connected and assigned to the same group. Different graph cuts methods are helpful. However, due to the fact that identifying a minimum cut is NP-hard, efficient approximations using linear algebra based on Laplacian Matrix can help us solve this problem

By finding the eigenvalues and eigenvectors of Laplacian matrix, an insight into the connectivity of the graph can be found. In our project, K-way Spectral Clustering method was applied.

5.2 Algorithm

1. Read the datasets
2. Calculate the similarity W between each data point by Gaussian kernel and construct similarity matrix.

3. Sum up the row values of the similarity matrix and fill the diagonal elements of a n by n zero matrix D with these values.

4. Create the Laplacian matrix by D - W

5. Calculate the eigenvalues and eigenvectors of the Laplacian matrix and sort the eigenvalues and corresponding eigenvectors.

6. Find the value k which maximizes the eigen gap

7. Build embedded space from the eigenvectors corresponding to the k smallest eigenvalues

8. Apply k-means to the reduced $n \times k$ space to produce k clusters

5.3 Implementations

The python code also takes two arguments. The first one is our dataset of txt format and the second argument is the number of cluster. The sigma value can be modified in the code.

1. sim_cal: It took the dataset and sigma value as arguments and output the similarity matrix. The Gaussian kernel is in use.

2. deg_find: The function took the similarity matrix as the single argument and would return the corresponding diagonal matrix.

3. $incidence\_mat\_gen$: The function take the result label as the only argument and output an incidence matrix based on that.

4. ja_rand_cal: Calculate rand and jaccard results. Ground truths and my results are the first and second argument respectively.

5.4 Results

File name	Rand	Jaccard	Sigma
cho.txt	0.8044	0.4263	1
iyer.txt	0.8436	0.4172	3.6

Pros for Spectral Clustering:

1. Compared with the traditional K-Means algorithm, spectral clustering is more adaptable to data distribution, and the clustering result is also excellent. At the same time, the calculation is less.

2. Spectral clustering only requires a similarity matrix between data. As a result, it's effective for clustering of sparse data, which is difficult for traditional clustering algorithms such as K-Means.

3. Due to dimensional reduction, the time costs are better than that of traditional clustering algorithms when solving data with high dimensions.

Cons for Spectral Clustering:

1. If the dimension of the data is very high, due to insufficient dimensionality reduction, the time costs and the result may not good.

2. The clustering results highly depend on the similarity matrix. Thus different similarity matrices may have different results.

6. Appendix

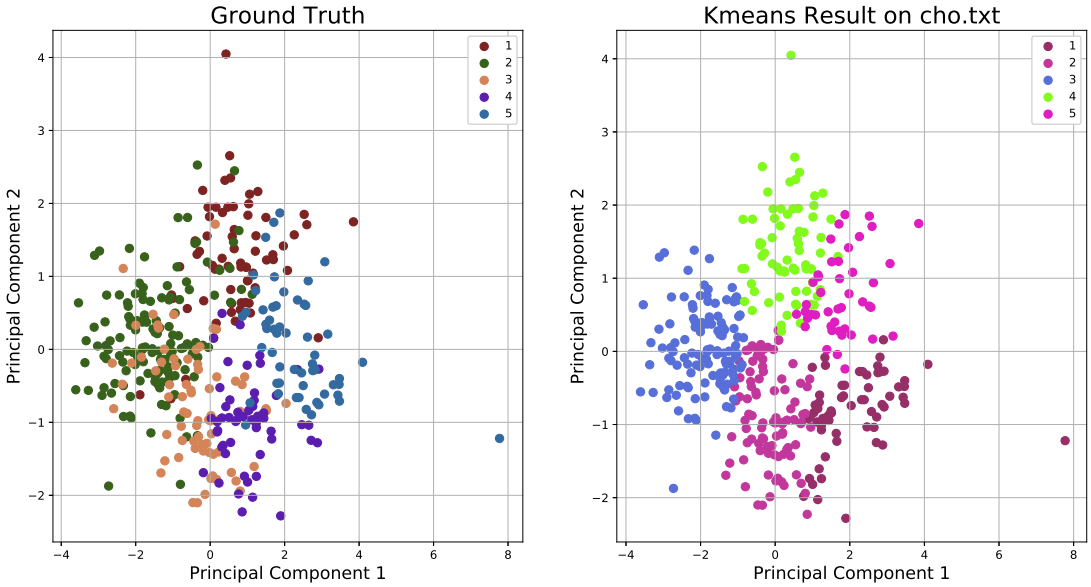

Figure 1: Kmeans clustering on cho.txt

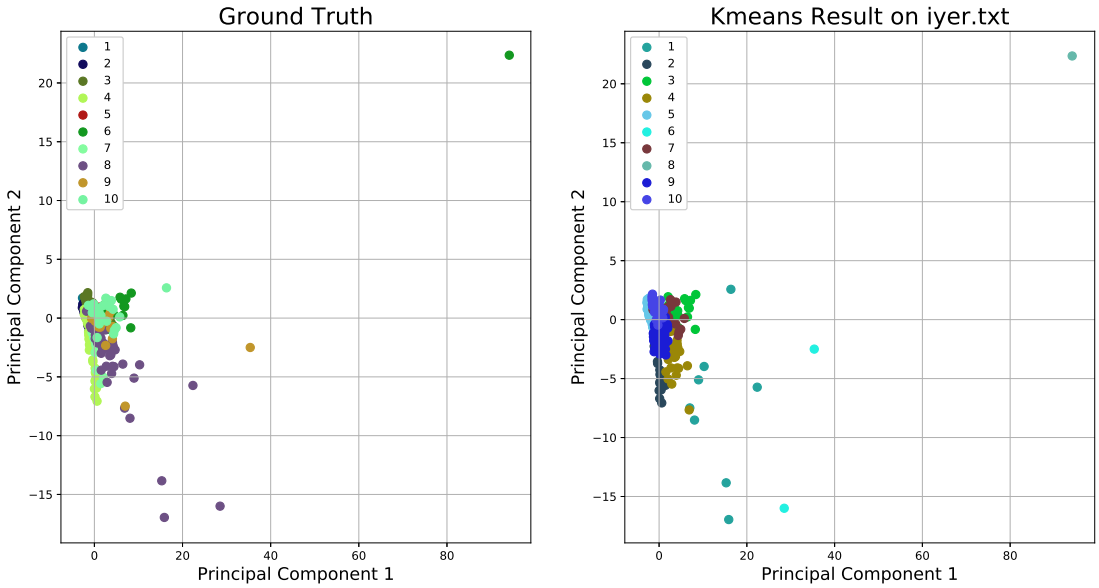

Figure 2: Kmeans clustering on iyer.txt

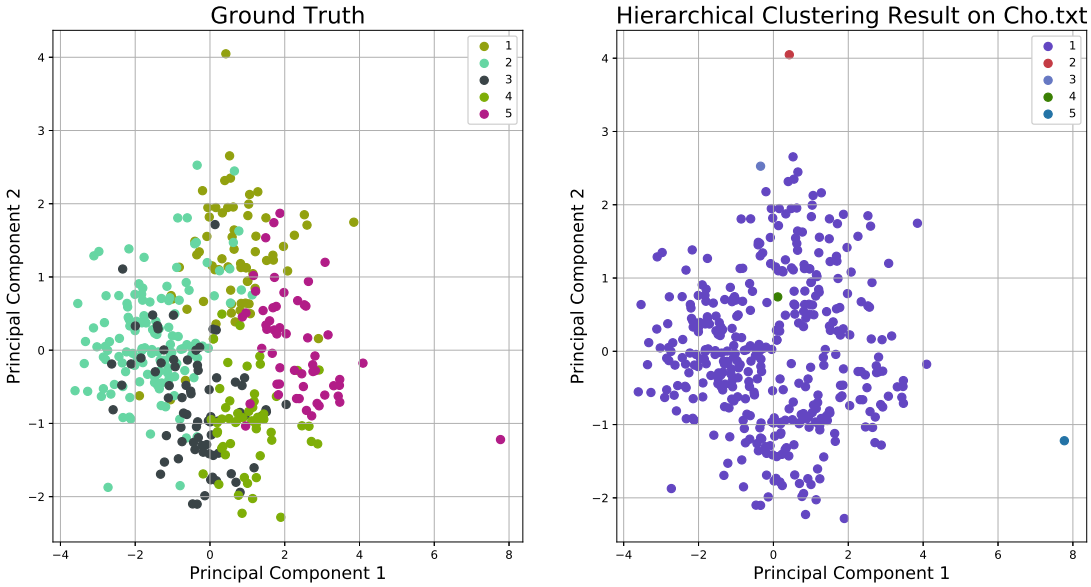

Figure 3: Hierarchical Clustering on cho.txt

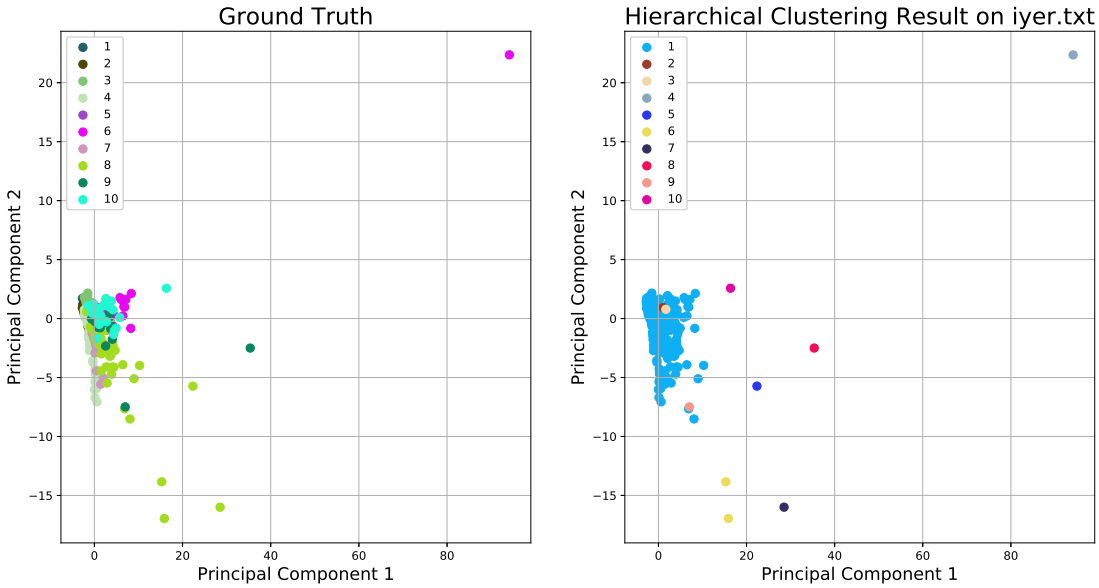

Figure 4: Hierarchical Clustering on iyer.txt

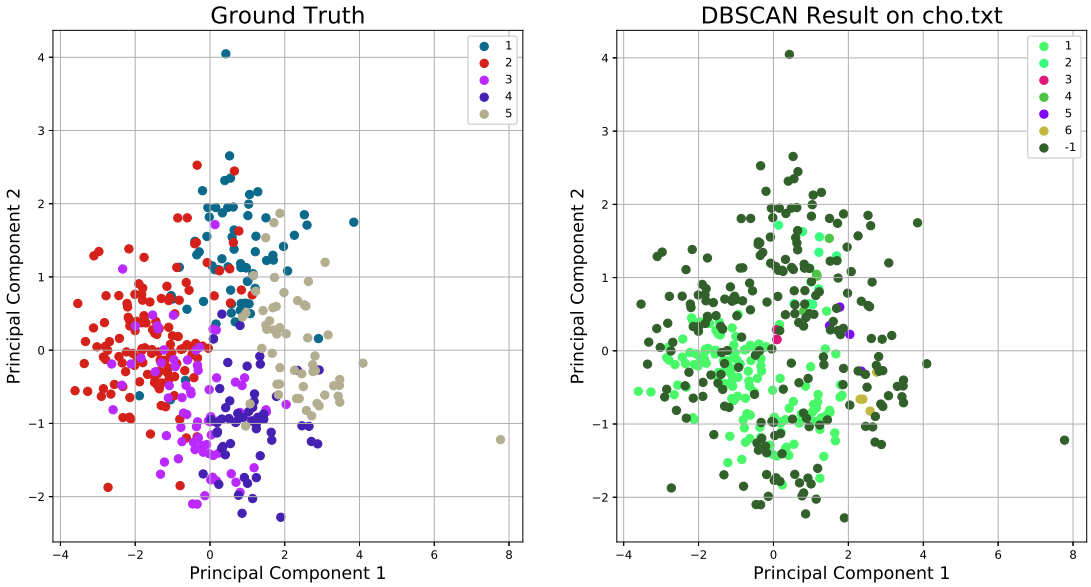

Figure 5: DBSCAN Clustering on cho.txt

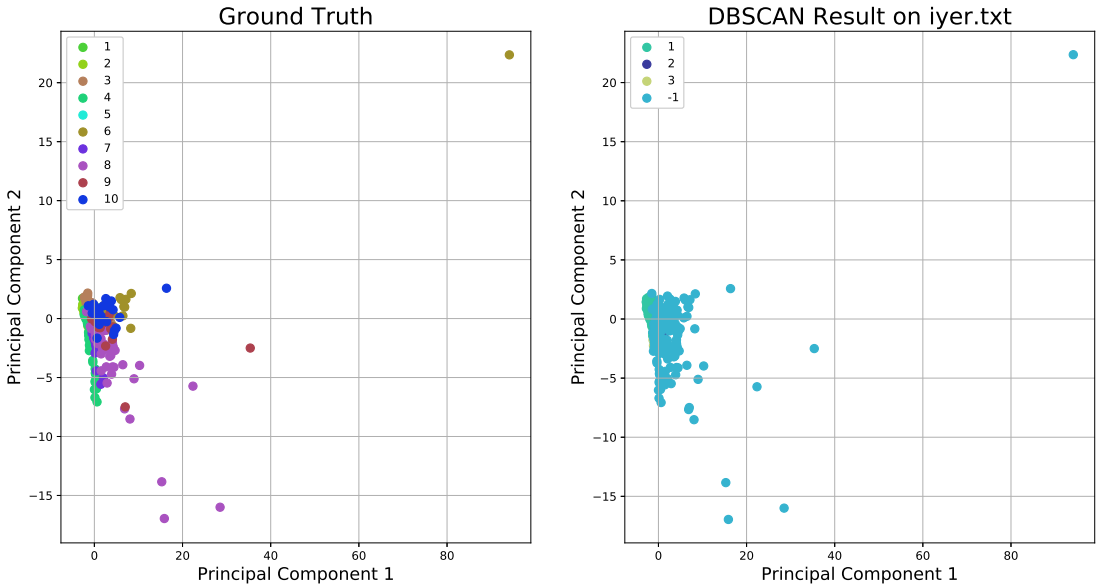

Figure 6: DBSCAN Clustering on iyer.txt

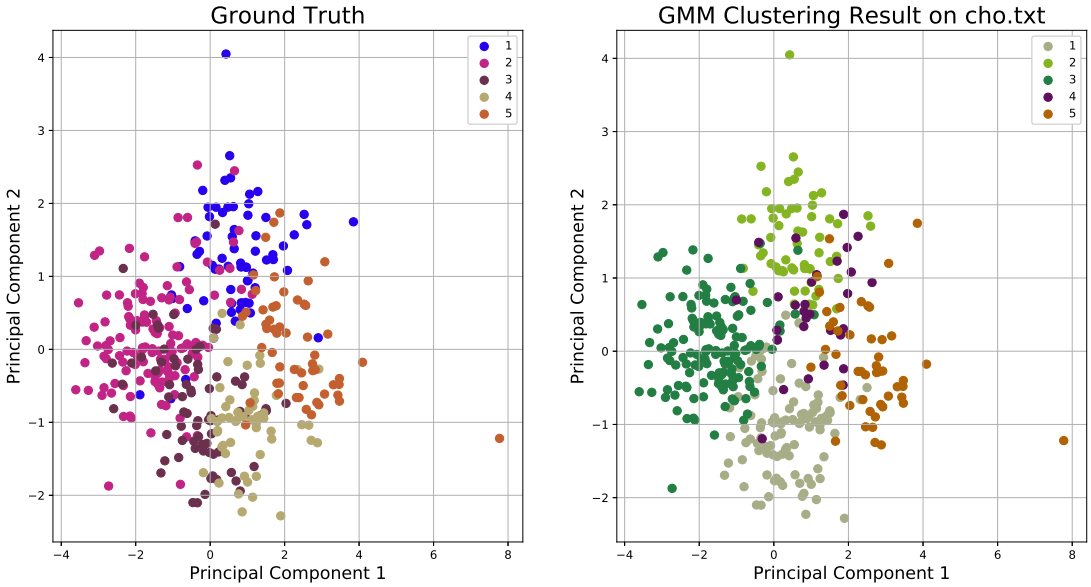

Figure 7: GMM Clustering on cho.txt

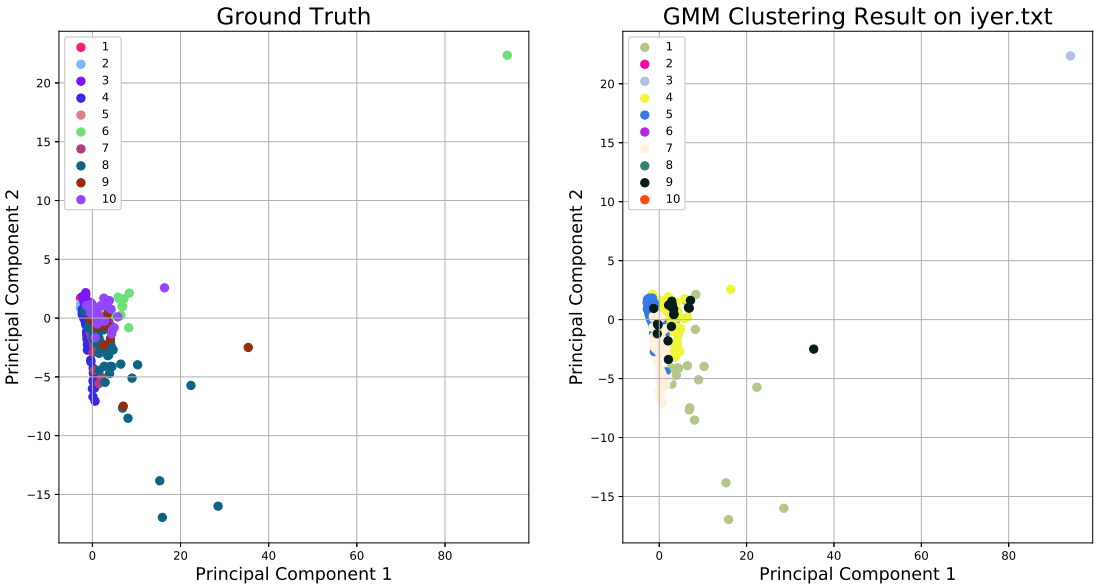

Figure 8: GMM Clustering on iyer.txt

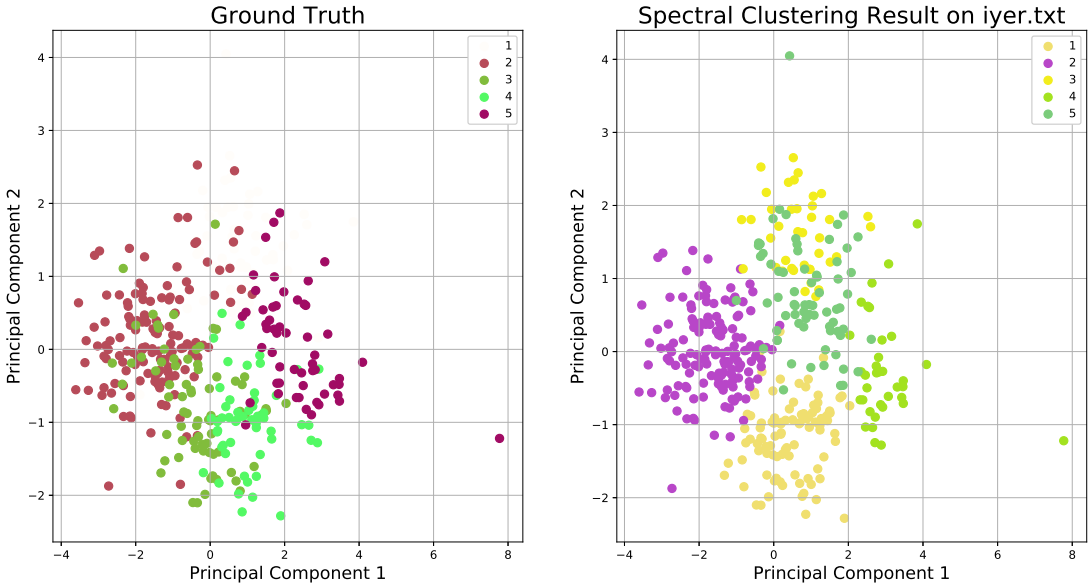

Figure 9: Spectral Clustering on cho.txt

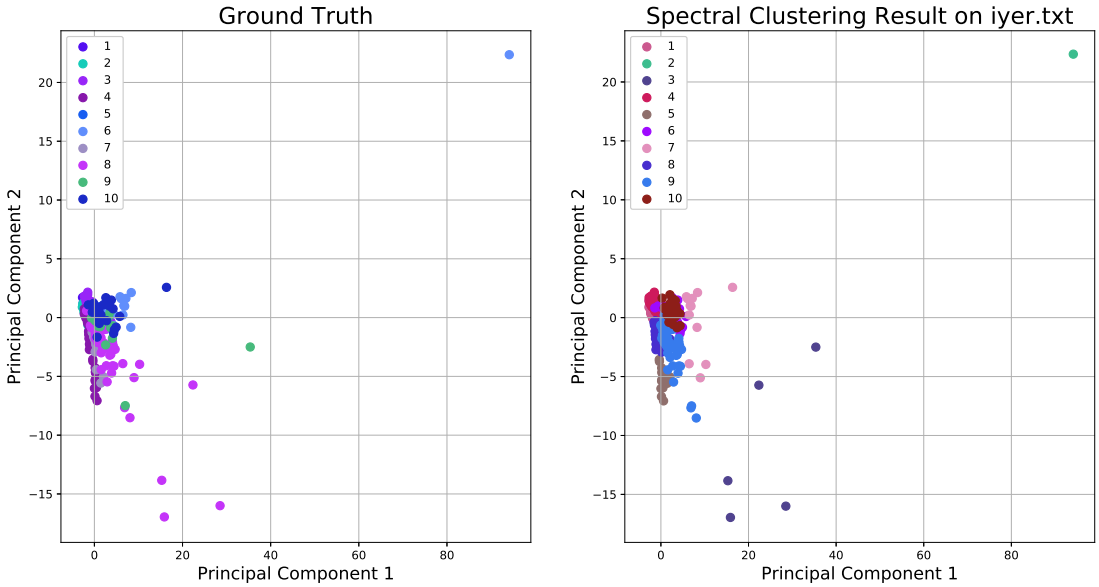

Figure 10: Spectral Clustering on iyer.txt