STATE UNIVERSITY AT BUFFALO

Project 2 : Clustering Algorithm

Yuze Liu 50207903
Luting Chen 50133507
Vicky Zheng 50037709

11/04/2016

1 K-MEANS ALGORITHM

1.1 ALGORITHM DESCRIPTION

K Means clustering is a method of vector quantization. K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

1.2 IMPLEMENTATION

In the clustering problem, we are given a data set $x^{(1)}, ..., x^{(m)}$, and want to group the data into a few cohesive "clusters." Here, we are given feature vectors for each data point $x^{(i)} \in Dataset$; Our goal is to predict k centroids and a label $c^{(i)}$ for each data point.
The pseudo code for K-Means Algorithm is as follows:

1. Initialize cluster centroid $\mu_1, \mu_2, \mu_3, ...\mu_k \in DataSet$
2. Repeat until converge: {
   for every i, set:
   $c^{(i)} := \arg\min ||x^{(i)} - \mu_j||^2$
   for each j, set :
   $\mu_j := \frac{\Sigma_{m-1}^{m}\{c^{(i)}=j\}x^{(i)}}{\Sigma_{m-1}^{m}\{c^{(i)}=j\}}$
   }

So for each iteration, we assign the data point to the nearest center, this way, a new cluster will be formed. For each cluster, we then calculate the new center for them, we repeat this procedure until the cluster is no longer changed.
In the program, we first randomly picked K points from the dataset as the initial center. We used build in random function from python, each time we run the code, the initial center will be different, this will have a effect with the result. But to compare the result from K-Means and MapReduce K-Means, we used random.seed() function, to make sure the initial center is the same for both program for this single run.

1.3 Result Visualization

We have been provided with 4 dataset,'cho.txt', 'iyer.txt', 'new_dataset_1.txt', and 'new_dataset_2.txt'. Here is the result for each dataset. Since the ground truth cluster has been provided in the dataset, so we we choose how many clusters we need for this dataset, we just use the number of ground truth clusters.
For the data that has outliers, we didn't remove them before running the program. But when we calculate the external index, we have deleted the outliers from the dataset.
Rand Index = 0.7542
Jaccard Coefficient = 0.2953

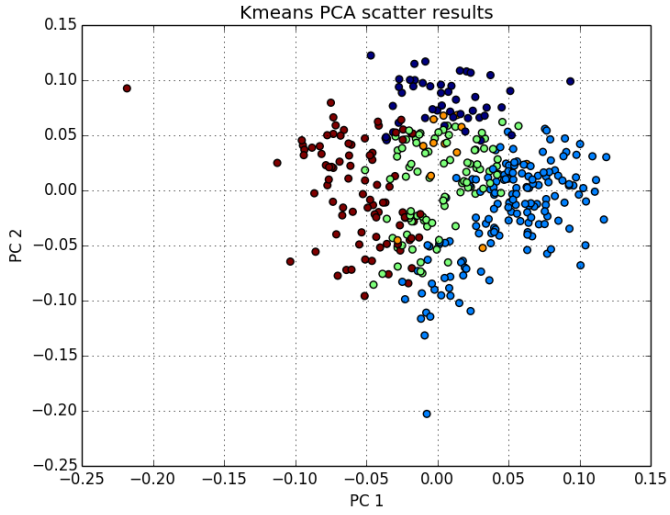

Figure 1.1: Result of K_Means Algorithm on Cho DataSet 2D Version, K=5

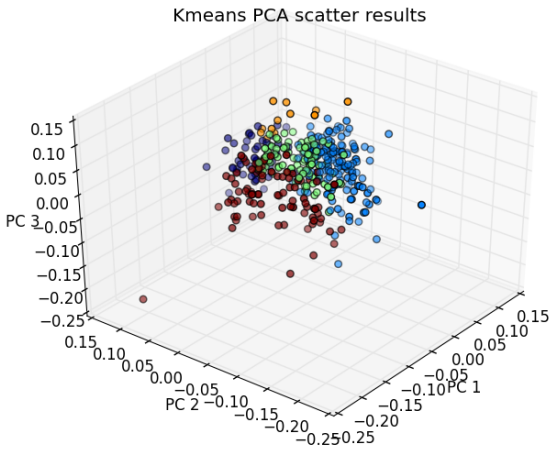

Figure 1.2: Result of K_Means Algorithm on Cho DataSet 3D Version, K=5

Rand Index = 0.8140
Jaccard Coefficient = 0.2543

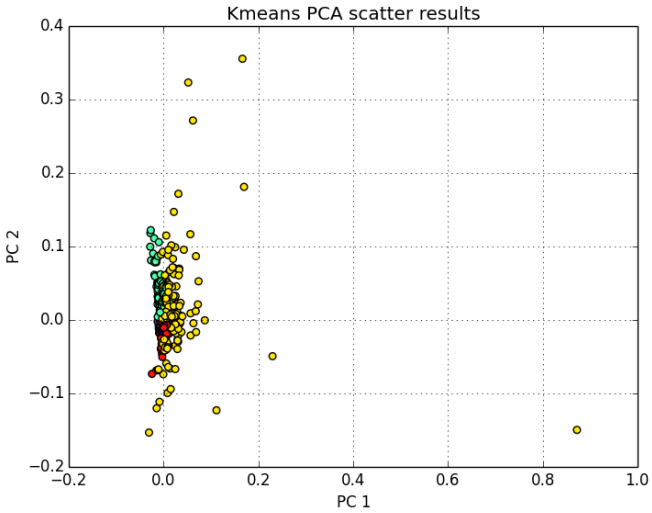

Figure 1.3: Result of K_Means Algorithm on iyer DataSet 2D Version,K=10

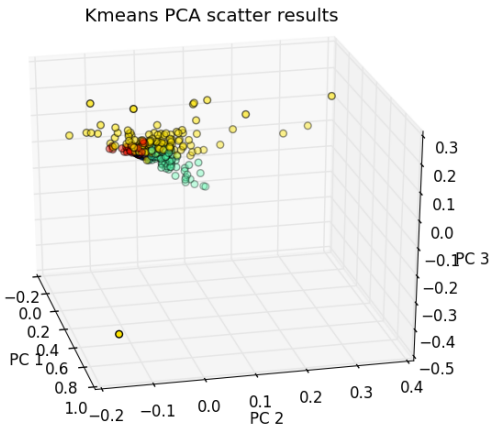

Figure 1.4: Result of K_Means Algorithm on iyer DataSet 3D Version,K=10

Rand Index = 0.7330
Jaccard Coefficient = 0.5194

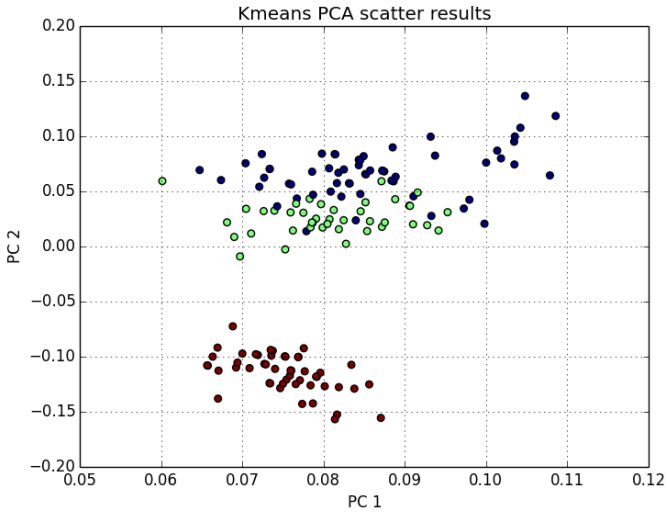

Figure 1.5: Result of K_Means Algorithm on new dataset1 2D Version,K=3

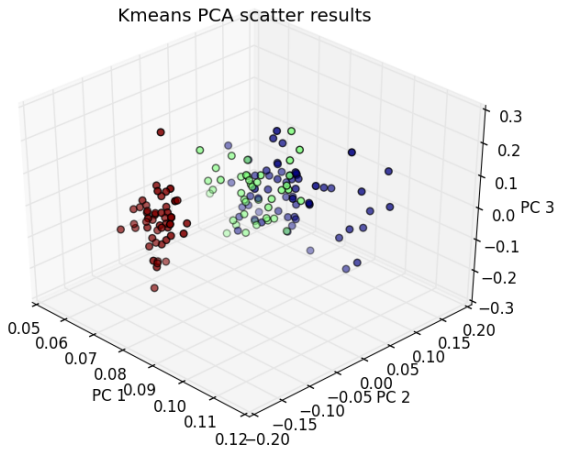

Figure 1.6: Result of K_Means Algorithm on new dataset1 3D Version,K=3

Rand Index = 1
Jaccard Coefficient = 1

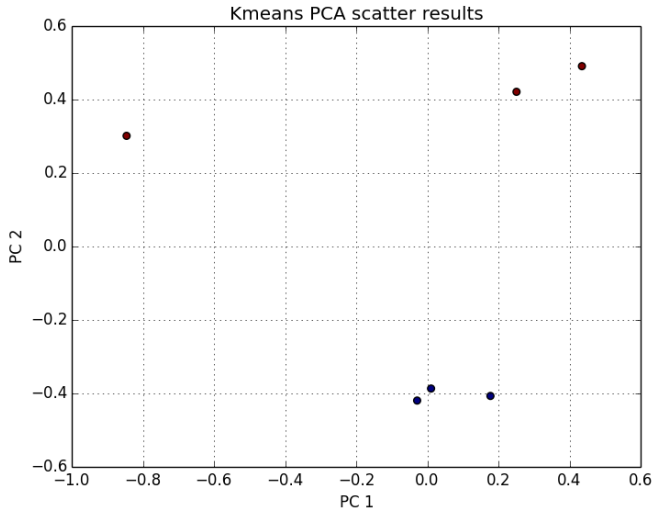

Figure 1.7: Result of K_Means Algorithm on new dataset2 3D Version,K=2

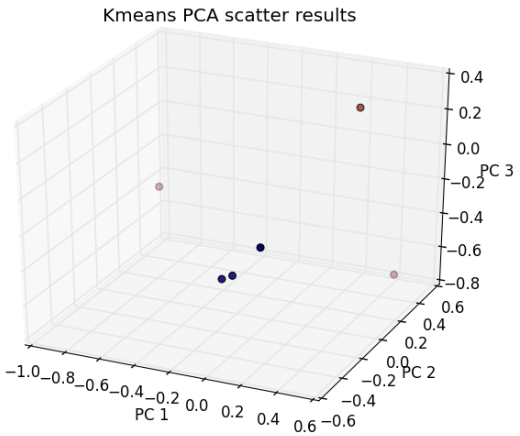

Figure 1.8: Result of K_Means Algorithm on new dataset2 3D Version,K=2

1.4 Result Evaluation

The result of K-Means is highly related to the initial K centers we chose.
For the dataset2, it's the simplest dataset among these 4 datasets. The result is good, both the rand index and the Jaccard Coefficient are both 1. But during several runs, it still has the choice to get lower value because the selection of the initial centers are random.
The advantage of the K-means:
1. Compare to other algorithm, the K-Means runs really fast and can be guaranteed to converge. Especially when the dataset is huge and we keep the k in a reasonable amount, the algorithm runs very fast.
2. K-Means produce tighter clusters than Hierarchical clustering. The disadvantage of the K-means:
1. It's difficult to predict K value. For this project, since we already know the ground truth result, so we can choose the right K for each dataset. But in the reality, for a given dataset, we can't know the proper value of K, so we need to predict K, which makes it hard.
2. Different initial centers can result in different final clusters
3. If the dataset has outliers, it will be hard to get the outliers out of the data. And if one of the outliers has been chosen as the initial center, then it will yield bad result.

2 HIERARCHICAL AGGLOMERATIVE CLUSTERING

2.1 ALGORITHM DESCRIPTION

In hierarchical agglomerative clustering, we build a hierarchy of clusters in a bottom-up fashion. To begin with, each observation is treated as a cluster, giving us N clusters if we have N observations at hand. Then we merge pairs of clusters while moving up the hierarchy, in the end we have the root of the hierarchy, which contains all observations in one cluster. The merging policy is greedy: each time, we only merge the pair of clusters that has the smallest inter-cluster distance according to some dissimilarity function.

2.2 IMPLEMENTATION

Some important variables and data structures:
ids : the list of observation IDs;
gts: the list of observation ground truth;
ges: the matrix of observation features;
cluster_list: the list of all appearing clusters in the hierarchy;
p_dist: the matrix of pairwise Euclidean distance for observations

We write 2 functions for the purpose of hierarchical clustering:

1. res = hierarchical_clustering(p_dist)
2. clustering = hc_cluster_n(cluster_num,res)

Function hierarchical_clustering(p_dist) returns the whole hierarchy using global variables ids, ges, clustering_list and p_dist. Function hc_cluster_n(cluster_num,res) returns and prints the list of cluster labels after passing cluster_num(the number of clusters) and res(the whole cluster hierarchy)

For cluster dissimilarity, we use Euclidean distance to compute the dissimilarity between observations and use Single linkage as our linkage criteria.

We use a python list named cluster_list to store all clusters appearing in the hierarchy, each element in cluster_list is one cluster, which contains the IDs of observation members. The hierarchy is also stored in a list and one level of the hierarchy is one element of the list. The following is an example.

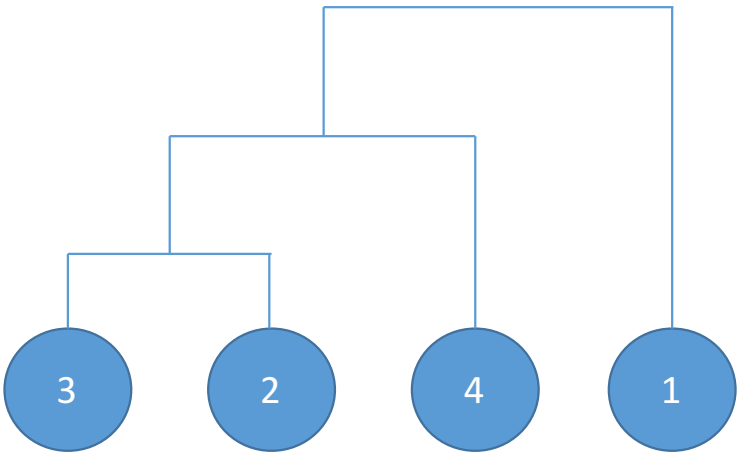

Figure 2.1: Dendrogram of a toy example with 4 points

steps	dendrogram	cluster_list (index from 0)	current level res	clusters in this level
init		[[0],[1],[2],[3]]	[0,1,2,3]	cluster_list[0]= [0] => 1 cluster_list[1]= [1] => 2 cluster_list[2]= [2] => 3 cluster_list[3]= [3] => 4
step1		[[0],[1],[2],[3],[1,2]]	[0,3,4]	cluster_list[0]= [0] => 1 cluster_list[3]= [3] => 4 cluster_list[4]= [1,2] => 2,3
step2		[[0],[1],[2],[3],[1,2],[1,2,3]]	[0,5]	cluster_list[0]= [0] => 1 cluster_list[5]= [1,2,3] => 2,3,4
step3		[[0],[1],[2],[3],[1,2],[1,2,3],[0,1,2,3]]	[6]	cluster_list[6]= [0,1,2,3] => 1,2,3,4

Figure 2.2: Steps of hierarchical agglomerative clustering for the toy example

2.3 Result Visualization

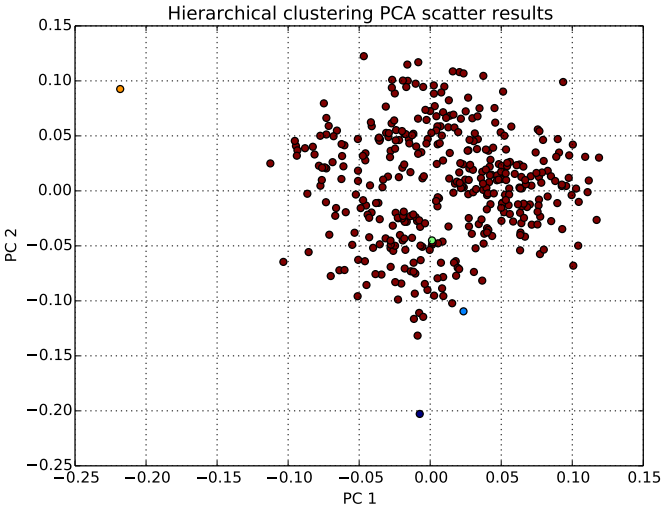

Figure 2.3: Result of hierarchical clustering on Cho DataSet 2D Version, K=5

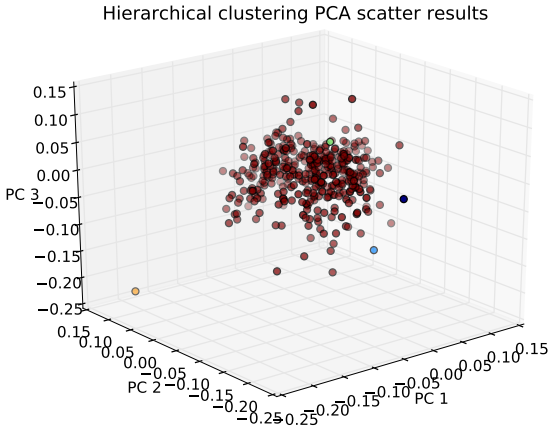

Figure 2.4: Result of hierarchical clustering on Cho DataSet 3D Version, K=5

Rand Index = 0.2403
Jaccard Coefficient = 0.2284

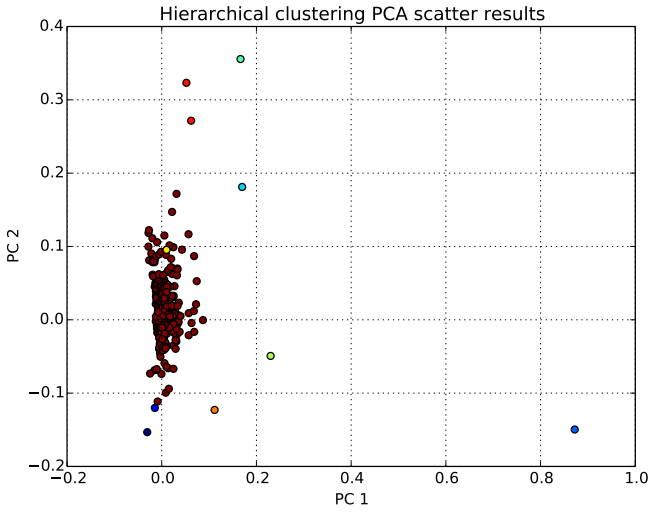

Figure 2.5: Result of hierarchical clustering on Iyer DataSet 2D Version, K=10

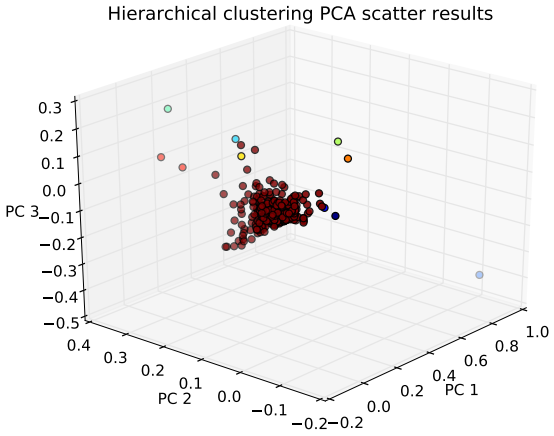

Figure 2.6: Result of hierarchical clustering on Iyer DataSet 3D Version, K=10

Rand Index = 0.2074
Jaccard Coefficient = 0.1761

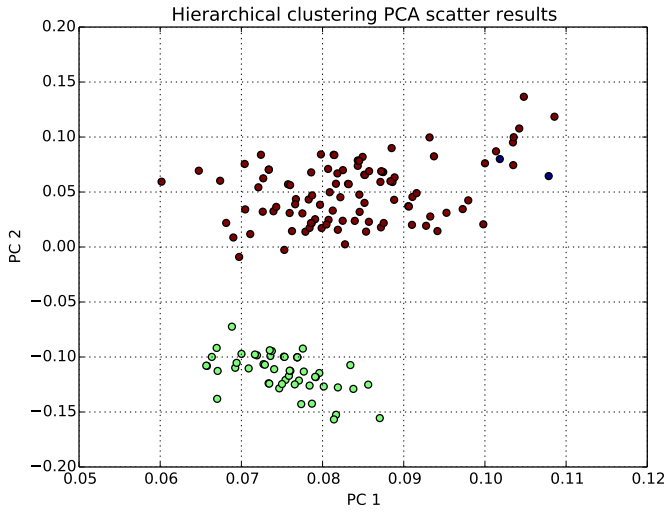

Figure 2.7: Result of hierarchical clustering on new DataSet 1 2D Version, K=3

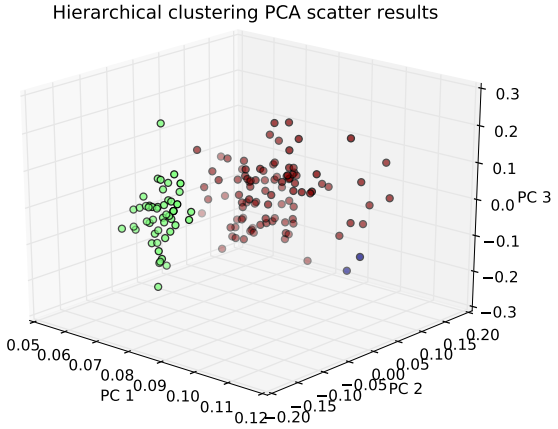

Figure 2.8: Result of hierarchical clustering on new DataSet 1 3D Version, K=3

Rand Index = 0.7781
Jaccard Coefficient = 0.5941

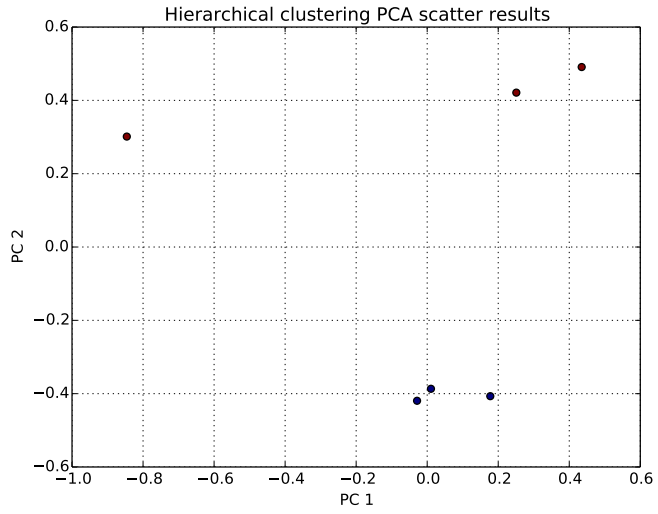

Figure 2.9: Result of hierarchical clustering on new DataSet 2 2D Version, K=2

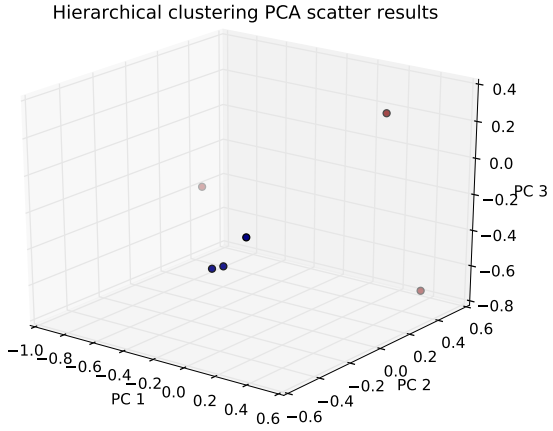

Figure 2.10: Result of hierarchical clustering on new DataSet 2 3D Version, K=2

Rand Index = 1.0000
Jaccard Coefficient = 1.0000

2.4 Result Evaluation

Here we are asked to use single linkage, so it will merge 2 clusters if they have the pair of points that has the smallest distance. One drawback of the algorithm is that it doesn't take into consideration the positions of other points in the clusters. The cho dataset shows the drawback of hierarchical clustering clearly. It contain some points that are far away from the other points, so they will be merged in the last few steps and hence more likely to become clusters with one single point. And it's also slower compared to other algorithms.

Some advantages of hierarchical clustering are it builds the dendrogram and can yield results for different cluster numbers easily; besides, it doesn't need to specify the number of clusters.

3 Spectral Algorithm

3.1 Algorithm Description

Spectral clustering computes a low-dimension embedding of the affinity matrix between samples, and then it performs K_Means in the low dimensional space and uses the resulting cluster labels as final results.

3.2 Implementation

We follow the pipeline by [1]. The following is the details.

- Given a data set $S = \{s_1, \ldots s_n\}$ to be clustered

1. Calculate the affinity matrix $\boldsymbol{A_{ij} = \exp(-\|s_i - s_j\|^2/2\sigma^2)}$, if $i \neq j$ and $A_{ii} = 0$ where σ^2 is the scaling parameter
2. Define $\boldsymbol{D}$ to be the diagonal matrix whose (i,i)-element is the sum of A's i-th row, and construct the matrix $\boldsymbol{L = D^{-1/2} A D^{-1/2}}$
3. Find k largest eigenvectors of L and form the matrix $\boldsymbol{X = [x_1\, x_2 \ldots x_k]}$
4. Form the matrix Y from X by normalizing each of X's rows to have unit length, $\boldsymbol{Y_{ij} = X_{ij} / (\Sigma_j X_{ij}^2)^{1/2}}$
5. Treating each row of Y as a point, cluster them into k clusters via K-means or any other algorithm
6. Assign the original point s_i to cluster j if and only if row i of the matrix Y was assigned to cluster j

Figure 3.1: Pipeline for spectral clustering

To notice, laplace matrix L has many forms. Instead of using L defined in class slides, we follow the definition of [1], where $L = D^{-0.5} A D^{-0.5}$ instead of using $L_{slide} = D^{-0.5}(D - A)D^{-0.5}$. Their relationship is easy to get: $L_{slide} = D^{-0.5}(D - A)D^{-0.5} =$

$I - D^{-0.5}AD^{-0.5} = I - L$ and eigen values of our L equals to 1 minus the corresponding eigen values of L_{slide}. So here we choose the k eigen vectors corresponding to the k largest eigen values. In our program, we can specify the value of σ, or we can choose optimal σ automatically by searching over candidates of σs and choose the one that yields highest external index value. In our implementation, we use rand index for default to judge if one σ is good.

3.3 Result Visualization

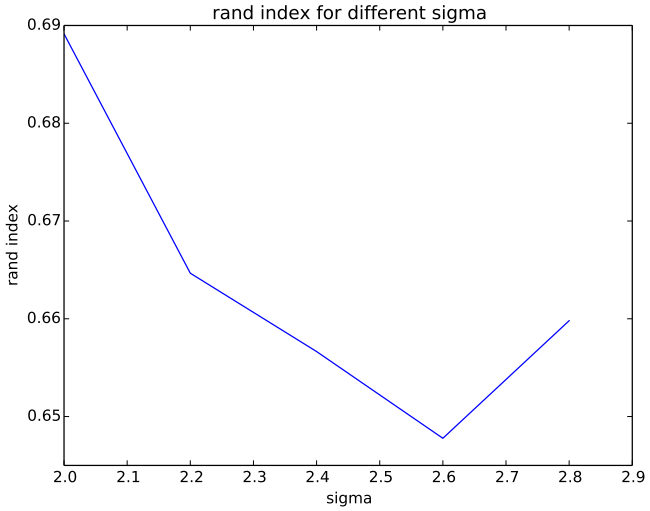

Figure 3.2: Choosing σ for Cho DataSet

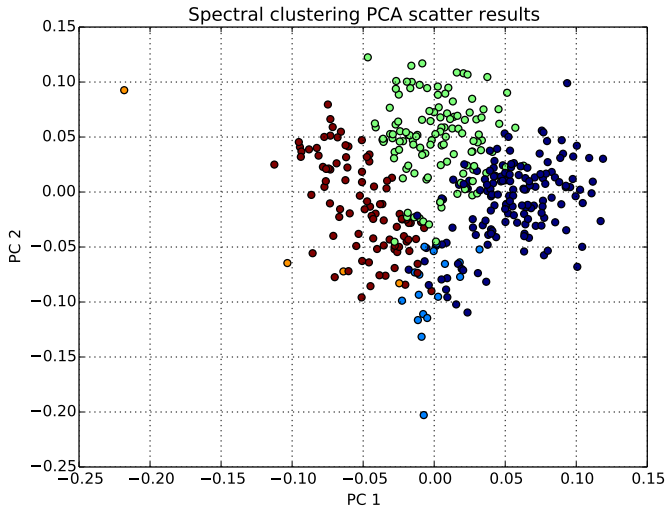

Figure 3.3: Result of spectral clustering on Cho DataSet 2D Version, K=5, $\sigma = 2$

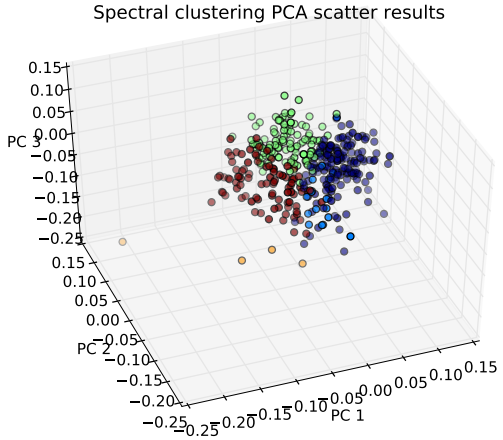

Figure 3.4: Result of spectral clustering on Cho DataSet 3D Version, K=5, $\sigma = 2$

Rand Index = 0.7570
Jaccard Coefficient = 0.3817

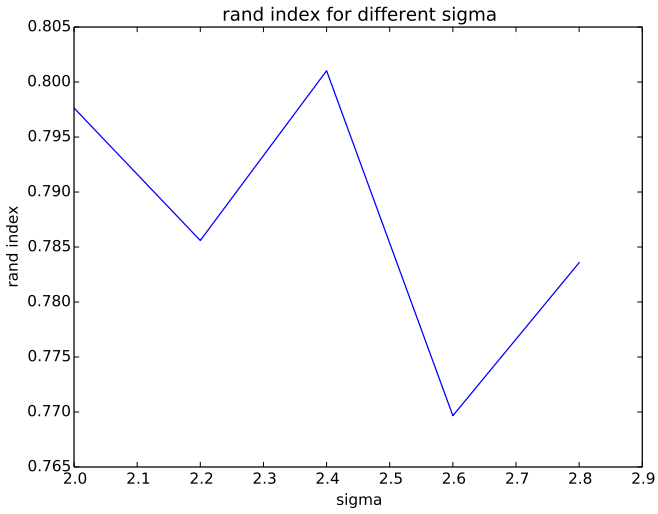

Figure 3.5: Choosing σ for Iyer DataSet

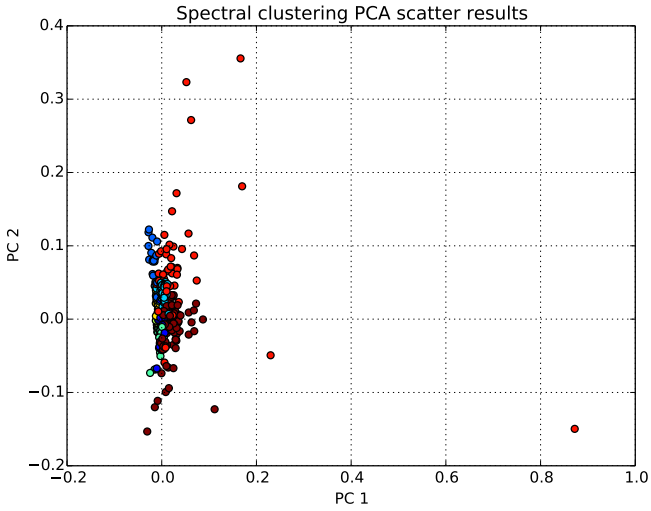

Figure 3.6: Result of spectral clustering on iyer DataSet 2D Version, K=10, σ = 2.4

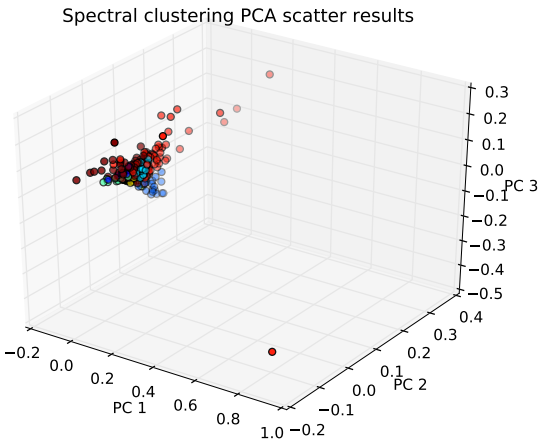

Figure 3.7: Result of spectral clustering on iyer DataSet 3D Version, K=10, σ = 2.4

Rand Index = 0.8241
Jaccard Coefficient = 0.2742

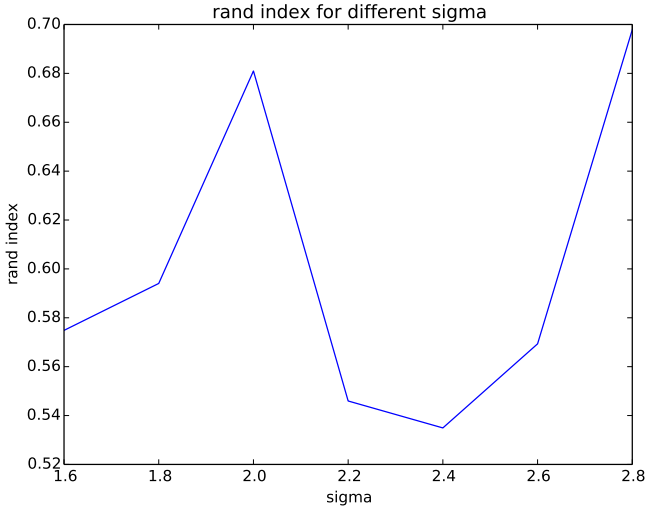

Figure 3.8: Choosing σ for new DataSet 1

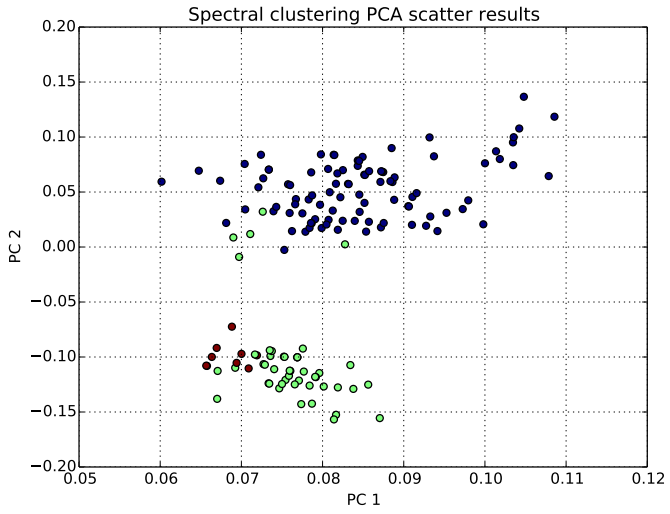

Figure 3.9: Result of spectral clustering on new DataSet 1 2D Version, K=3, σ = 2.8

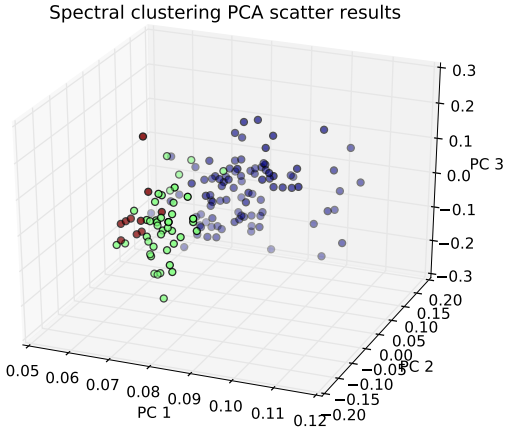

Figure 3.10: Result of spectral clustering on new DataSet 1 3D Version, K=3, σ = 2.8

Rand Index = 0.7267
Jaccard Coefficient = 0.5040

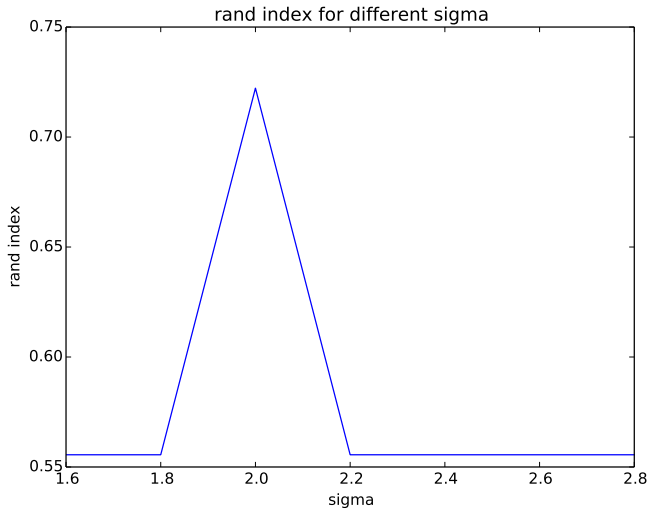

Figure 3.11: Choosing σ for new DataSet 2

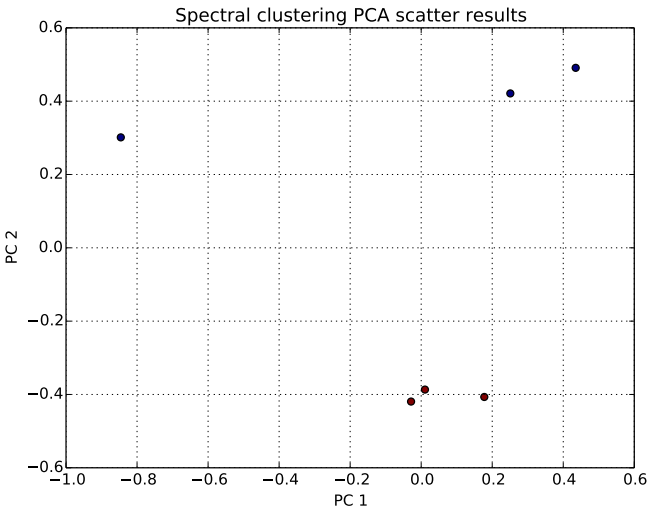

Figure 3.12: Result of spectral clustering on new DataSet 2 2D Version, K=2, $\sigma = 2$

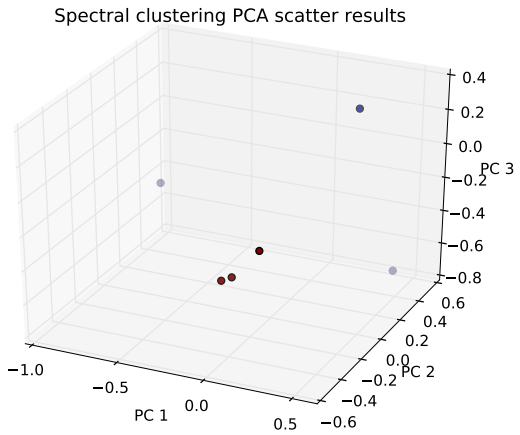

Figure 3.13: Result of spectral clustering on new DataSet 2 3D Version, K=2, $\sigma = 2$

Rand Index = 1.0000
Jaccard Coefficient = 1.0000

3.4 Result Evaluation

Spectral clustering is built on beautiful mathematics and the solution is elegant, which has a closed form. But it's not very robust to noise, like for iyer dataset, which has some outliers, its performance is badly affected. Another drawback is it will become computationally expensive if the input dimension gets large because of the computation for eigen values and eigen vectors.

3.5 Algorithm Description

Kmeans is a nice algorithm to implement with map reduce because it is embarrassingly parallel. Kmeans consists of calculating the distance of a node with every centroid, finding the centroid which has the minimum distance, assigning each entry to a cluster and then updating the centroids. It is clear that calculating the distances between nodes and centroids can be done independently from other nodes. Therefore, this can be our map function that can be performed in parallel. Our reduce function would contain finding the minimum, assigning the node to a new centroid and calculating the new centroid. This is identical to the Kmeans described above, but in this part of the project we take advantage of the fact that kmeans is embarrassingly parallel.

3.6 IMPLEMENTATION

For this project, we used Hadoop to implement MapReduce Kmeans. We used hadoop streaming to read lines in from our data file to our mapper file, and then also used streaming to read in the outputs from the mapper file to the reducer program. In the mapper program, we calculated the distance of a node x and all k centroids. We then printed x's ID and its distances. In the sequential implementation of kmeans, we are not concerned with the ID because there is only one process running and you know exactly what order things are being run in. However, in the parallel version of kmeans, we have to keep track of x's ID because there are multiple distances being calculated at the same time. Simply printing out the distances would not be sufficient because we would not know which node this corresponds to.

In our reduce, we take in these values and also read in from a cached file which tells us the current centroids. We calculate the minimum distance becaused on the distances already given to us, and then assign each node x to a new centroid (if necessary). We know that we have finished assigning all nodes when we stop receiving inputs from the mapper program. Once this happens, we calculate the new centroids by calculating the average distance between each node and assigning that as the centroid. This is then updated in our cache file. We are also constantly updating another file which stores all current assignments of the nodes.

For this project, I wrote a script to repeatly call map and reduce. In the script, I specify the number of iterations of map and reduce to be performed. We can do multiple iterations because in each iteration, the centroids are updated. In the ideal scenerio, our program would converge within the number of iterations we allow it. Fortunately for this project, the datasets were relatively small so we are typically able to converge within 20 iterations. However, if we don't converge within the given iterations, we still have the assignments for the first 20 iterations. With this, we can visualize the state of our program at the 20th iteration.

3.7 Result Visualization

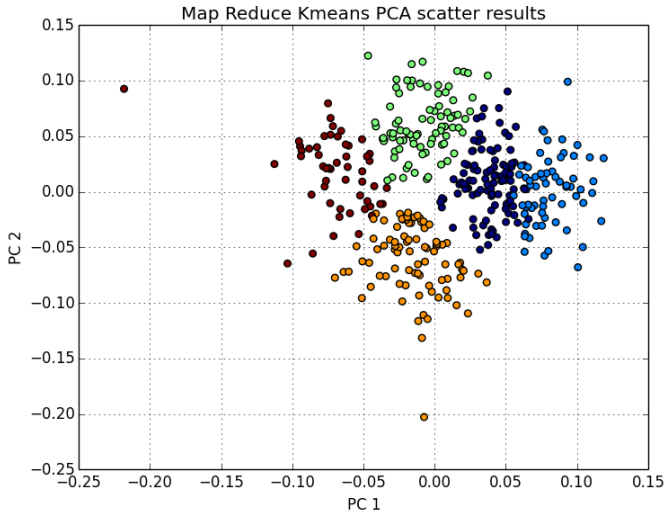

Figure 3.14: Result of K_Means Map Reduce Algorithm on Cho DataSet 2D Version, K=5

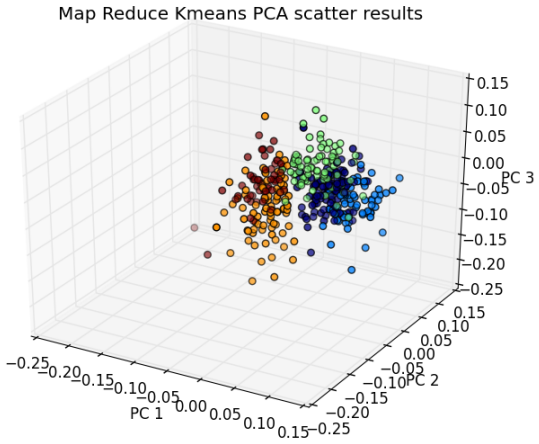

Figure 3.15: Result of K_Means Map Reduce Algorithm on Cho DataSet 3D Version, K=5

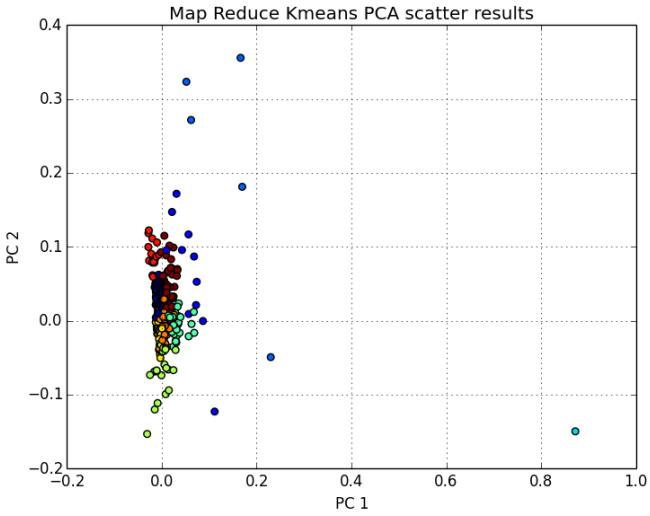

Figure 3.16: Result of K_Means Map Reduce Algorithm on Iyer DataSet 2D Version,K=10

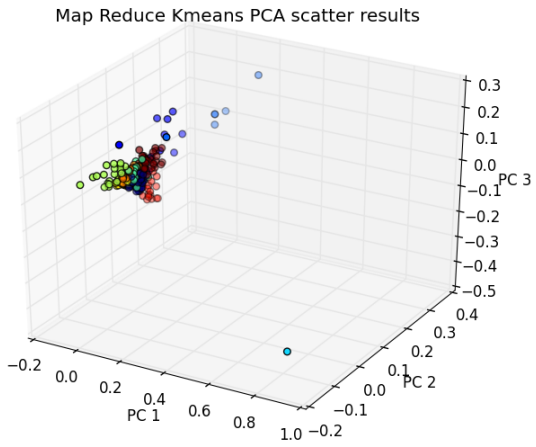

Figure 3.17: Result of K_Means Map Reduce Algorithm on Iyer DataSet 3D Version,K=10

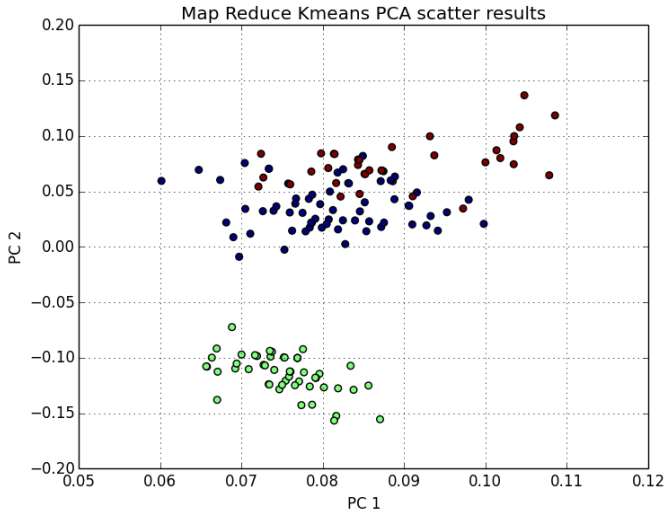

Figure 3.18: Result of K_Means Map Reduce Algorithm on new dataset 1 DataSet 2D Version,K=3

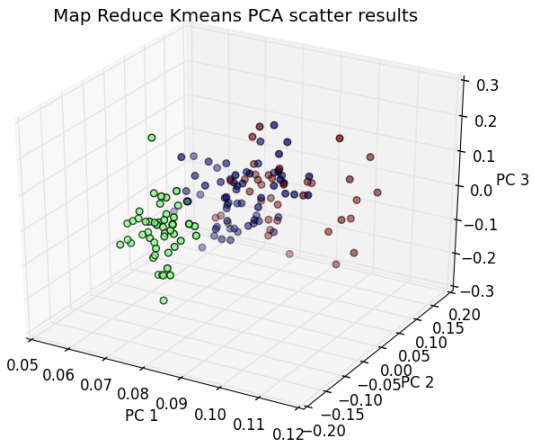

Figure 3.19: Result of K_Means Map Reduce Algorithm on new dataset 1 DataSet 3D Version,K=3

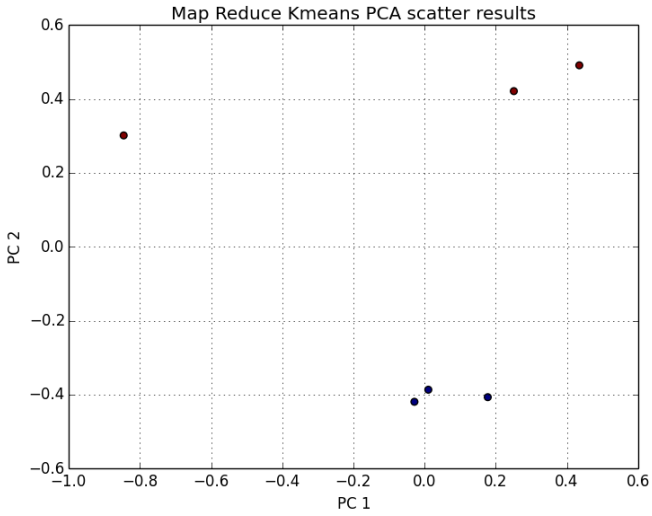

Figure 3.20: Result of K_Means Map Reduce Algorithm on new dataset 2 DataSet 2D Version,K=2

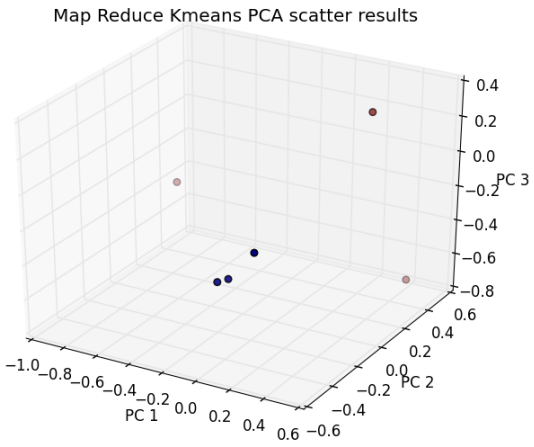

Figure 3.21: Result of K_Means Map Reduce Algorithm on new dataset 2 DataSet 3D Version,K=2

3.8 Result Evaluation

The map reduce kmeans has the same strong and weak points as the sequentially implemented kmeans described earlier. An additional downside to mapreduce kmeans, however, is that it runs more slowly than the sequentially kmeans. This is because of the overhead cost of using hadoop and setting up our node. The map reduce kmeans would likely outperform the sequential kmeans for large datasets, but the datasets we were given for this assignment were relatively small.

References

[1] Ng, Andrew Y., Michael I. Jordan, and Yair Weiss. "On spectral clustering: Analysis and an algorithm." Advances in neural information processing systems 2 (2002): 849-856.