Assignment4 Report

Qiu Wei, Lu Yizhou

May 18, 2017

Contents

1 Implementation

The code is seperated into two parts, *kmeans.py* and *main.py*. *main.py* parses the data and draw the graphs of different solution, and *kmeans.py* implemented the core function of k-means algorithm.

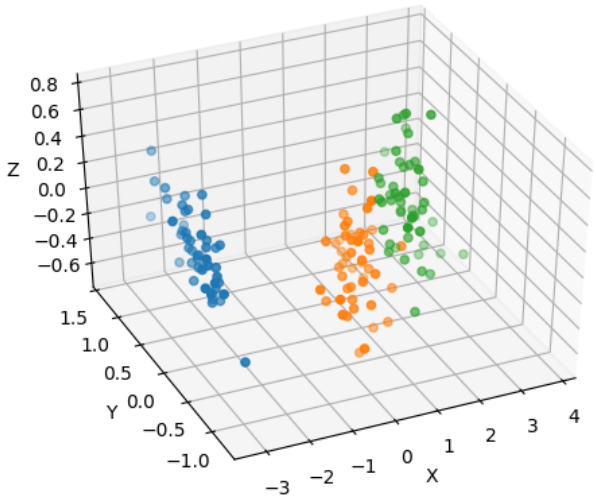
The k-means algorithm have the folloing steps:

- 1.Randomly choose k distinct points as centroids
- 2.Partition the data points into k parts surrounding the k centroids
- 3.Move the centroid of each parts to the arithmetic mean of the points in this part
- 4.If the centroids converges, halt the algorithm. Otherwise repeat step 2,3

In each case, *main.py* will run k-means algorithm for 100 times and record the maximum, minimum and average diameters.

2 Result and Analysis

The origin distribution is as following:



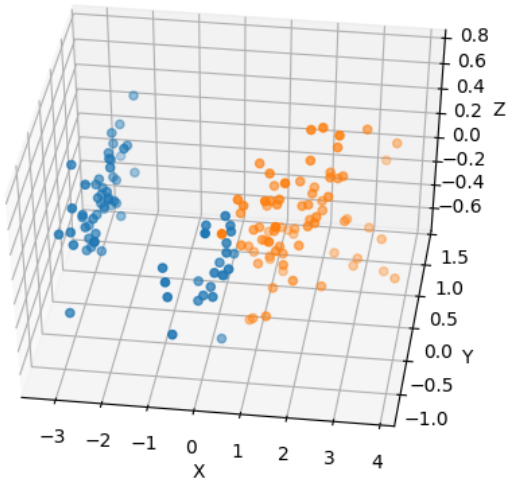
And we tried k=2,3,4 and calculated the maximum diameter of the k parts.

k	maximum diameter	minimum diameter	average diameter
2	6.91	3.91	4.72
3	4.86	2.58	3.07
4	4.70	2.41	2.52

Obviously, the diameter strictly decreases when k increase. Now I will analyse each case in the following.

2.1 k = 2

The distribution graph of the best result is as following:



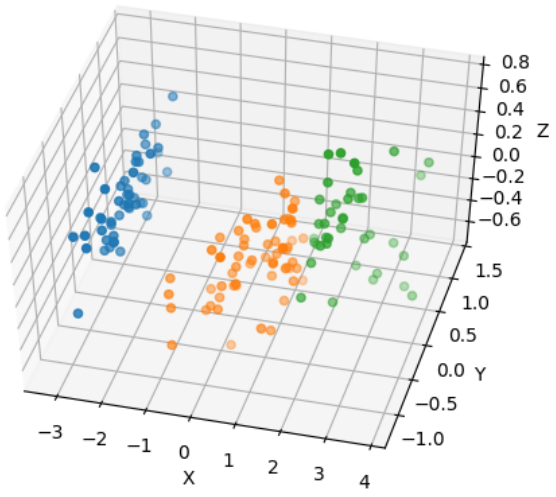
This graphs show a problem of k-means algorithm:

In the middle of the picture some points was partitioned into wrong class to reduce the maximum diameter.

Here is the question: the k-means algorithm can not tell a hollow sphere and a dense sphere, so it will classify two distinct classes as one class. Also this makes k-means algorithm sensitive with the original centroids.

2.2 k = 3

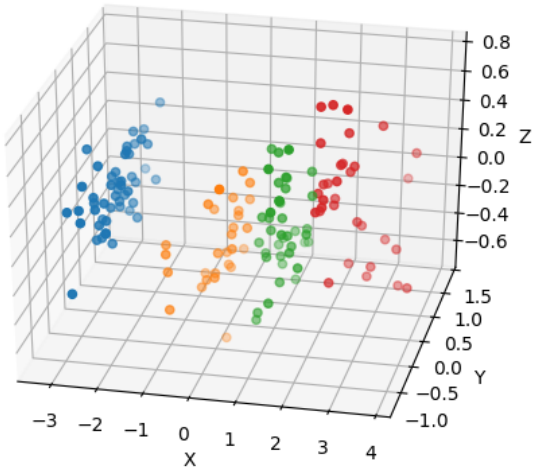
The distribution graph of the best result is as following:



Obviously this distribution is similar with the original graph.

2.3 $\mathbf{k = 4}$

The distribution graph of the best result is as following:



This distribution is just seperate two class in original distribution into three class. And it do not benefit a lot to the diameter.