Lab 5

Distributed K-means Clustering

CS429 - Introduction to Big Data Analysis

Lab Instructor: Nguyễn Đình Thảo

Table of Contents

Pseudocode

Standard K-means

Initially choose k points to be the centroids;
While the centroids have not converged:
    For each points p in the dataset:
        Calculate the euclidean distance from the point to the k centroids;
        Assign p to the cluster containing the nearest centroid to p;
        Adjust the centroid of the cluster to account for the newly added point p;

Distributed K-means

For a naive distributed k-means, we can divide the standard k-means algorithm into two phases:

1. Find the closest centroid to each point p in the dataset and assign p to the corresponding cluster
2. Update the centroids after adding new points to the clusters

The pseudocode is as follows:
Initially choose k points to be the centroids;
While the centroids have not converged:
    For each point p in the dataset, generate a key-value pair of

$$(argmin_i(distance(p, centroid_i)),\ (p, 1))$$

    For each cluster S_i, reduce all pairs with key i to the pair $(i,\ (\sum_{p \in S_i} p,\ |S_i|))$

    Update the centroids:

$$centroid_i = \frac{1}{|S_i|} \sum_{p \in S_i} p$$

Where:

- $argmin_i(distance(p,\ centroid_i))$ returns the index i of the $centroid_i$ that is closest to point p
- $distance(p,\ centroid_i)$ function returns the euclidean distance between point p and $centroid_i$
- S_i is the cluster with index i and $centroid_i$
- $|S_i|$ denotes the total number of points assigned to cluster S_i
- $\sum_{p \in S_i} p$ is element-wise addition of all points in a cluster S_i

Lab Assignment

Implement the provided distributed k-means pseudocode with number of clusters $k = 15$. For the sake of simplicity, you can set a specific number of iterations to run k-means as the convergence condition in the pseudocode. In this assignment, the centroids should converge after a couple of dozen iterations.

Input

Sample data can be found in file ***points.txt***. Each line is a single data point in two-dimensional space.

Output

Output should be:

- Final centroid coordinates

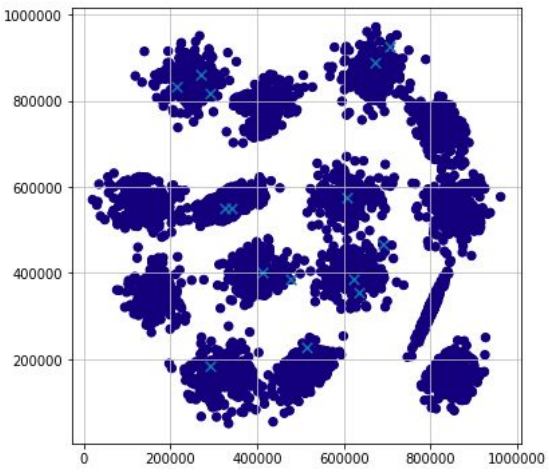
```
Out[41]: [(1, (326251, 549298)),
          (2, (271093, 861412)),
          (3, (635242, 356290)),
          (4, (411674, 402377)),
          (5, (290210, 818593)),
          (6, (477323, 384551)),
          (7, (673242, 886958)),
          (8, (513562, 229098)),
          (9, (623386, 387031)),
          (10, (705518, 925936)),
          (11, (340064, 550509)),
          (12, (689725, 465685)),
          (13, (607604, 573951)),
          (14, (213208, 833959)),
          (15, (292402, 184441))]
```

- Points with assigned cluster

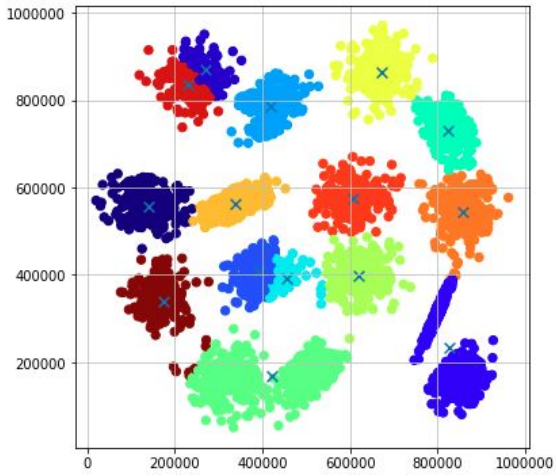
```
Out[54]: [(13, (664159, 550946)),
          (13, (665845, 557965)),
          (13, (597173, 575538)),
          (13, (618600, 551446)),
          (13, (635690, 608046)),
          (13, (588100, 557588)),
          (13, (582015, 546191)),
          (13, (604678, 574577)),
          (13, (572029, 518313)),
          (13, (604737, 574591))]
```

Visualization

Use the code provided in **Lab 5 - Visualization.ipynb** to visualize your clustering result. Below is an example of the visualization of before and after clustering, where the ‘x’ marker shows the centroids and clusters are color-coded.



Before clustering



After clustering

Submission

Submit your jupyter notebook with the naming format: **<your studentID>_lab5.ipynb**