Finding Communities within the Citibike Network

Shay Lehmann and Michelle Ho

CUSP-GX-5006
Prof. Gustavo Nonato
Machine Learning Assignment # 4

Abstract

In this assignment, the authors explore different clustering algorithms as a way to find communities within the Citibike network. In this directed network, the nodes are station locations and the edge weights are the count of trips over the given time period of data. The clustering techniques used are K-means, Spectral Clustering, and Hierarchical Clustering.

1 Introduction

Detecting communities within networks is one application of machine learning clustering algorithms. From making recommendations within a social networks to predicting the spread of disease, there are many applications for community detection.

2 Methods and Data Sets

The dataset chosen for this assignment is Citibike trip data from July 2013 to February 2014, as well as the station locations for that same time period. The variables available from Citibike System Data are:

- 'tripduration', 'starttime', 'stoptime', 'start station id', 'start station name', 'start station latitude', 'start station longitude', 'end station id', 'end station name', 'end station latitude', 'end station longitude', 'bikeid', 'usertype', 'birth year', 'gender'

The steps taken for this assignment:

1. A k-means clustering algorithm is used to cluster the stations based on weighted distance matrix. The weighted distance matrix is calculated as the shortest path of all stations to each other, weighted by the inverse of the sum of trips between each station during the study period. The process is repeated for 2 - 7 clusters (Figure 1).

2. A directed network graph is then created using the stations as nodes and the edges as trips between stations. The edges are weighted by the same inverse sum of trips.

3. The network graph is then used to generate spectral clusters with a RBF (Gaussian) kernel. First, the laplacian matrix is calculated for the directed network using the python library networkx.

4. Then, the python library sci-kit learn is used to generate 2 - 7 spectral clusters with the laplacian matrix (Figure 2)

5. Hierarchal clusters are also generated based on the network graph. First, the shortest path

length for all pairs in the network graph are calculated, weighting the edges by the inverse weight. Smaller values indicate "closer" stations, in other words, stations that have many trips between them. These shortest path lengths are used to generate a condensed distance matrix in the python library SciPy, and this matrix was finally used to generate hierarchy trees with single and complete linkage hierarchies using SciPy.

6. Dendrograms are generated for the two hierarchy trees (Figure 3)

7. The hierarchies are visualized on a map by coloring the station locations by their cluster labels for varying cuts of the trees that generate 2 - 7 clusters (Figures 4 and 5)

8. Special attention is paid to n clusters = 4 for the four clustering methods (k-means, spectral, and hierarchy with single and complete linkage) for discussion (Figure 6).

3 Results

The results show that spectral clustering generates clusters within the network graph have a discernible pattern that aligns somewhat with our expectations. Our expectation was that the Citibike network would show greater connectivity within boroughs. With spectral clustering in Figure 6, we see a cluster in areas where people tend to work (cyan dots in downtown and midtown Manhattan, and downtown Brooklyn and parts of Williamsburg). Other clusters tend to fall where there is higher residential population and along the east and west sides of Manhattan (green and purple). A single station in red indicates a very unique cluster. This station is situated between South Brooklyn, North Brooklyn, and Manhattan. Its uniqueness in "pull" from all directions may have caused Spectral Clustering to cluster this station on its own.

The single linkage hierarchical clustering produced the strangest results. Nearly every station was classified into the same cluster, except for three unique stations that were each in their own cluster. Looking at the dendrogram (Figure 3), this corresponds to what is happening in the tree. Single linkage merges clusters that minimize the distance between the clusters closest points. Complete linkage, on the other hand, merges clusters that minimize the distances between the clusters farthest points. With single linkage, a network graph can be quickly clustered together because if several stations share a similar "trip profile".

The complete linkage hierarchical clustering and kmeans clustering appear to create evenly spread clusters throughout the network, without any pattern to discern. Again, this is probably because stations may share very similar "trip profiles".

4 Conclusions

In summary, spectral clustering yields results that aligned most with our expectations. We were able to discern a pattern in how people used Citibike in each cluster based on our knowledge of Manhattan and Brooklyn, and their corresponding residential and commercial areas. However, the interpretation of these clusters is difficult given that the Citibike network is not a typical social network. Calculating the "shortest path" between nodes has no real-life interpretation for Citibike. For example, if the shortest path between Stations A and B is through a more popular station C, this tells us that A and B are not very directly connected, but may fall into the same cluster anyway. Further examination of communities within the Citibike network need to consider different distance metrics in addition to the shortest path.

Python code used to generate the results and figures for this assignment can be found at `https://github.com/michellemho/machine_learning_for_cities`.

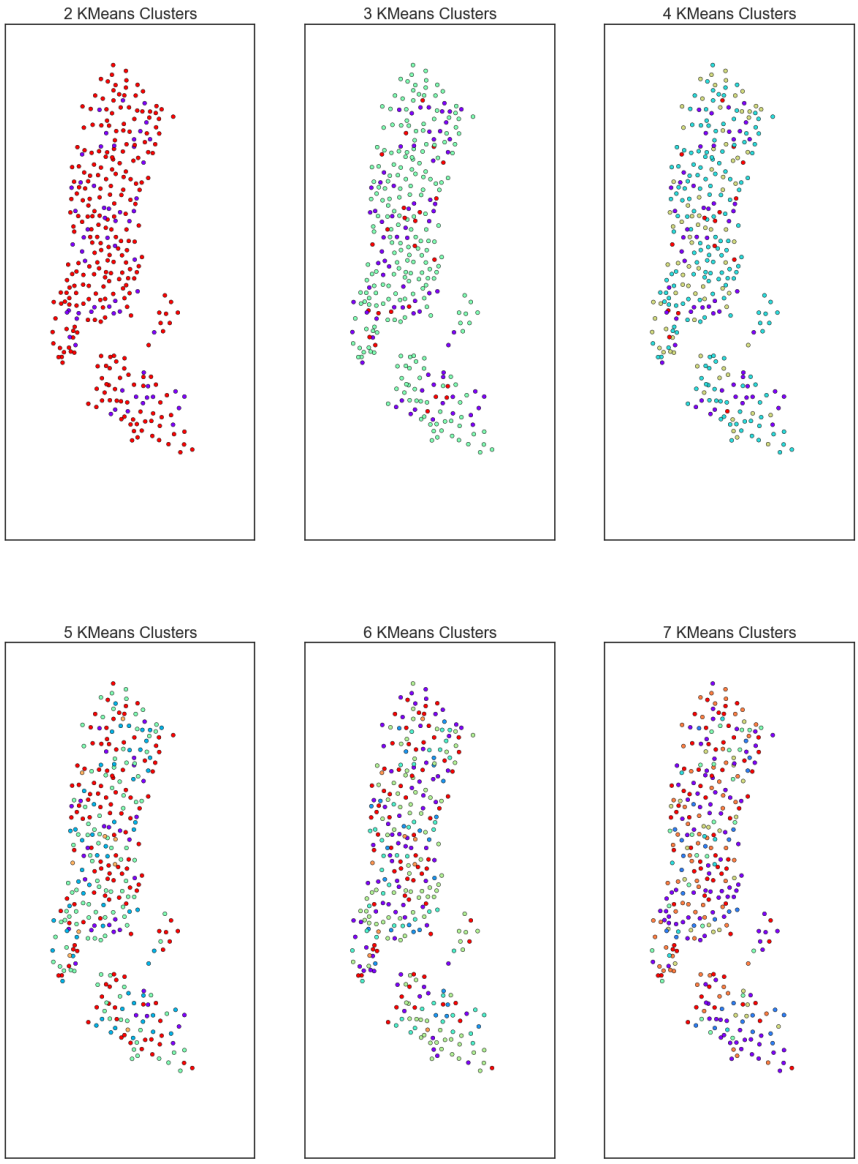

Figure 1: K-Means Clustering

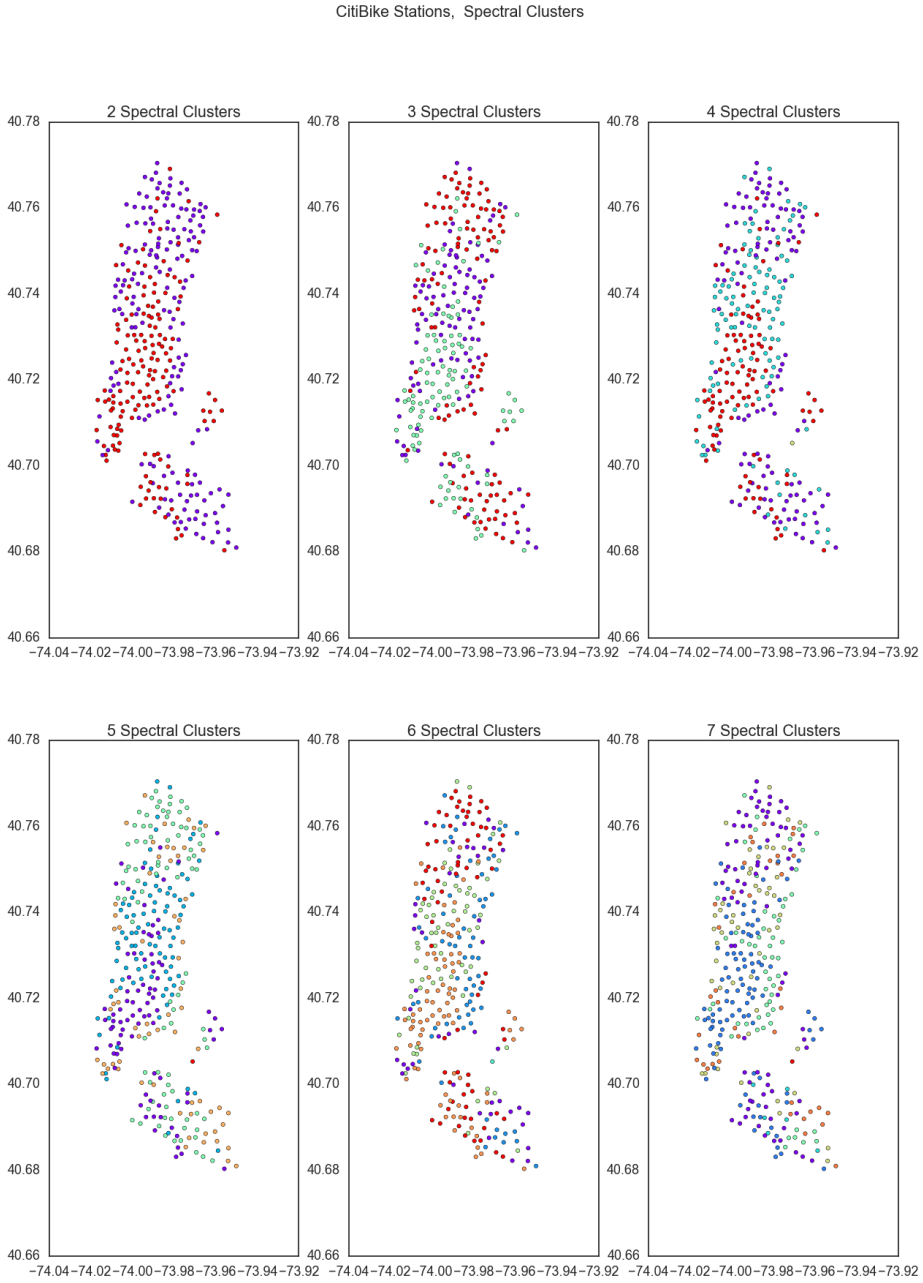

Figure 2: Spectral Clustering

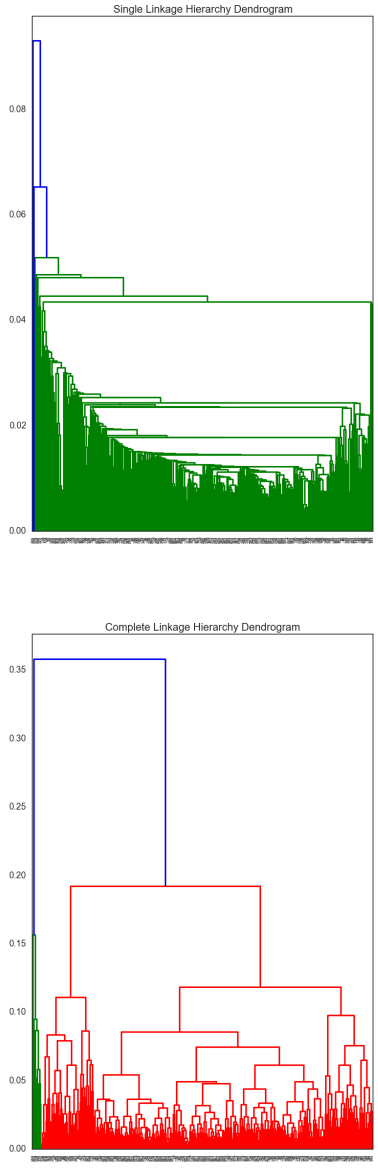

Figure 3: Dendrograms

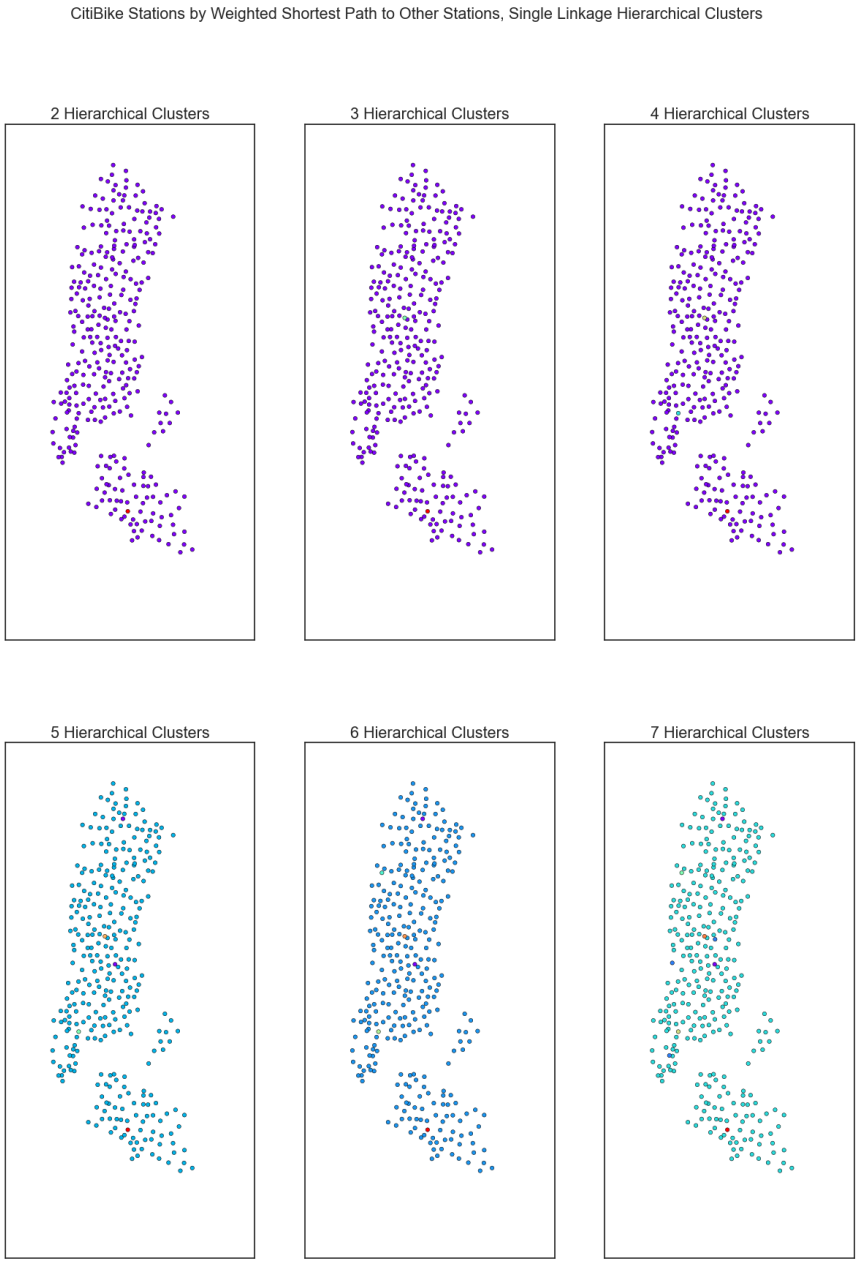

Figure 4: Hierarchy Single-Linkage

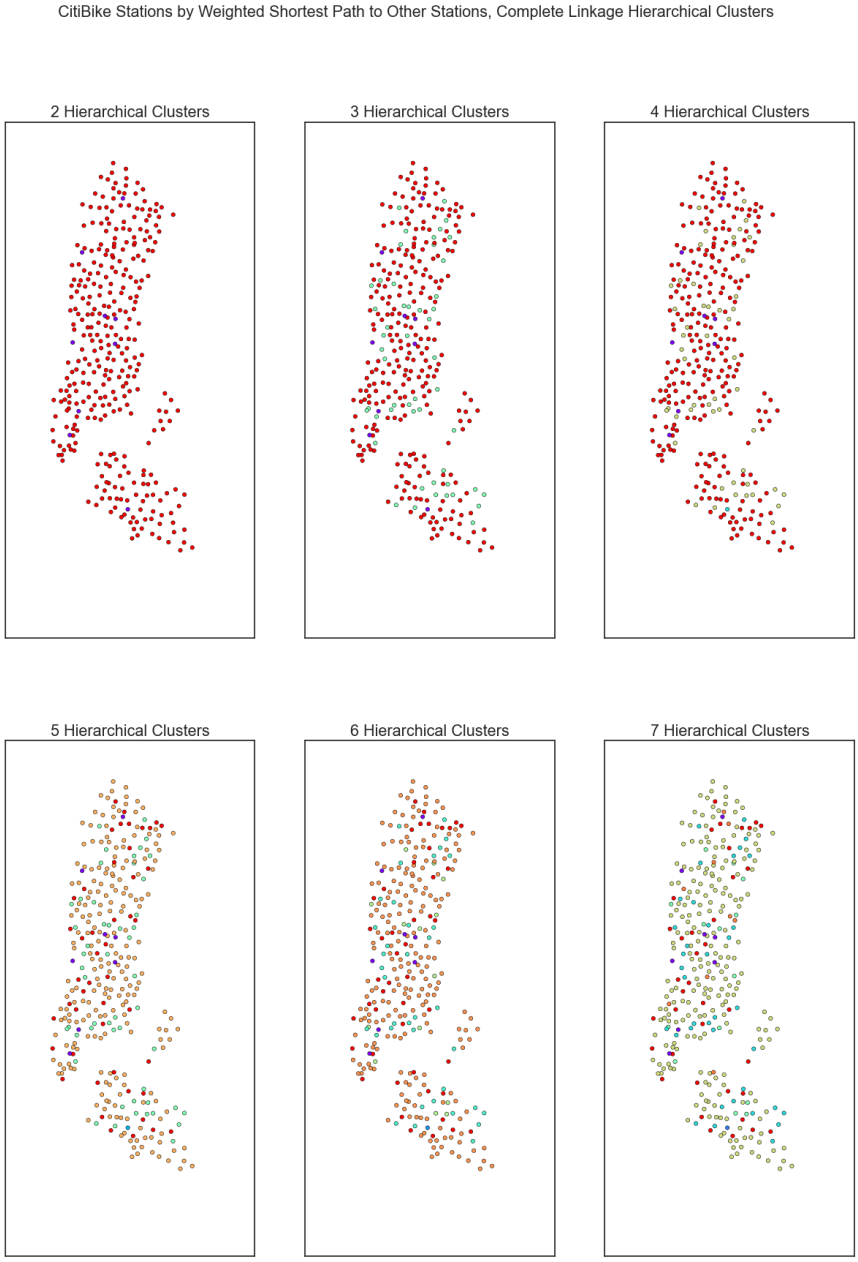

Figure 5: Hierarchy Complete-Linkage

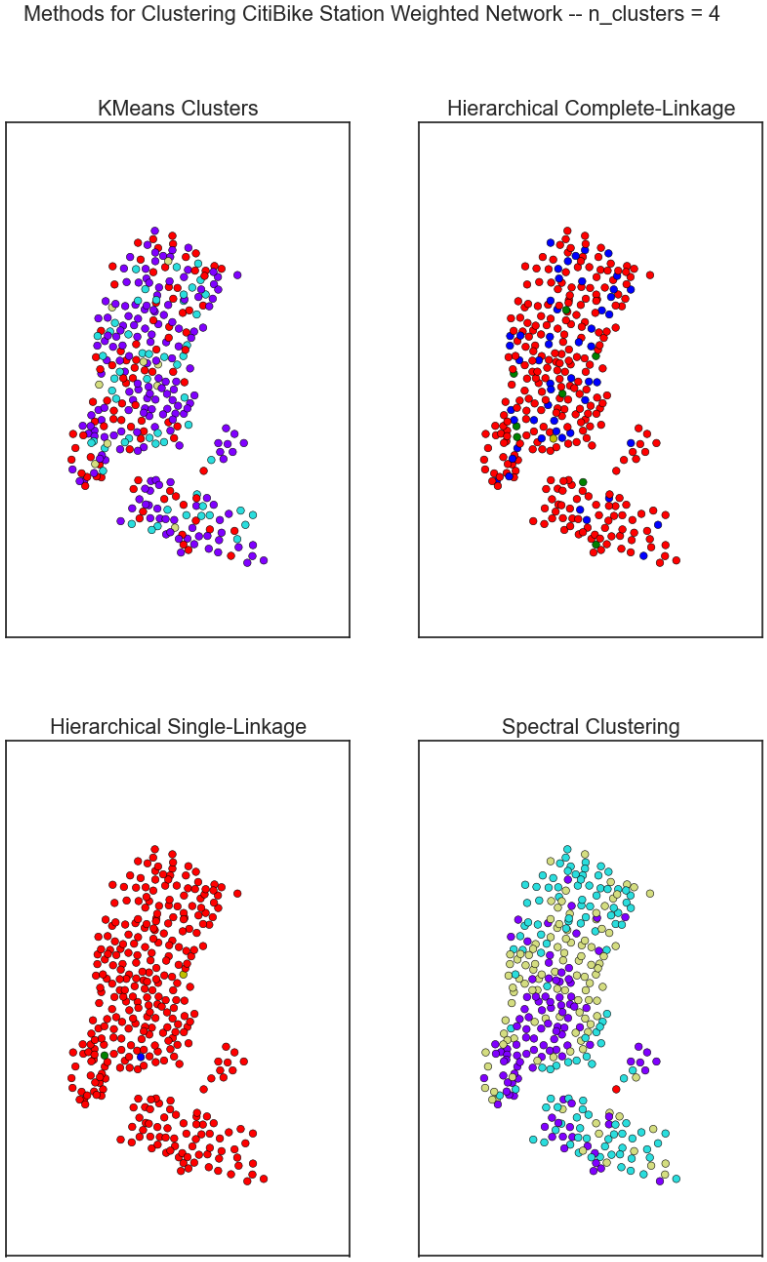

Figure 6: Multiple Clustering Methods