Assignment 5

Data Mining

Zander Davidson

11-30-20

To run the program, type “python kmeans.py” or “python3 kmeans.py” into a command terminal with a python compiler. Requires plotly and numpy. Results from clustering can be viewed by opening the attached .html files using a modern web browser like Chrome.

The goal of this assignment was to use K-Means clustering to find clusters of similar data within a data set. I chose to use the famous Iris dataset (see the “data” folder) which contains samples of petal and sepal measurements from 3 different plant species. The data are described by 4 continuous attributes:

- **sepal_length (cm)**
- **sepal_width (cm)**
- **petal_length (cm)**
- **petal_width (cm)**

The data are classified using the **class** attribute:

- **class** - (Iris-setosa, Iris-versicolor, Iris-virginica): species of the plant measurements sample

A few rows from the data file are listed below, where the comma-delimited values correspond to the attributes above, in the order they appear. The class is the final value in each list:

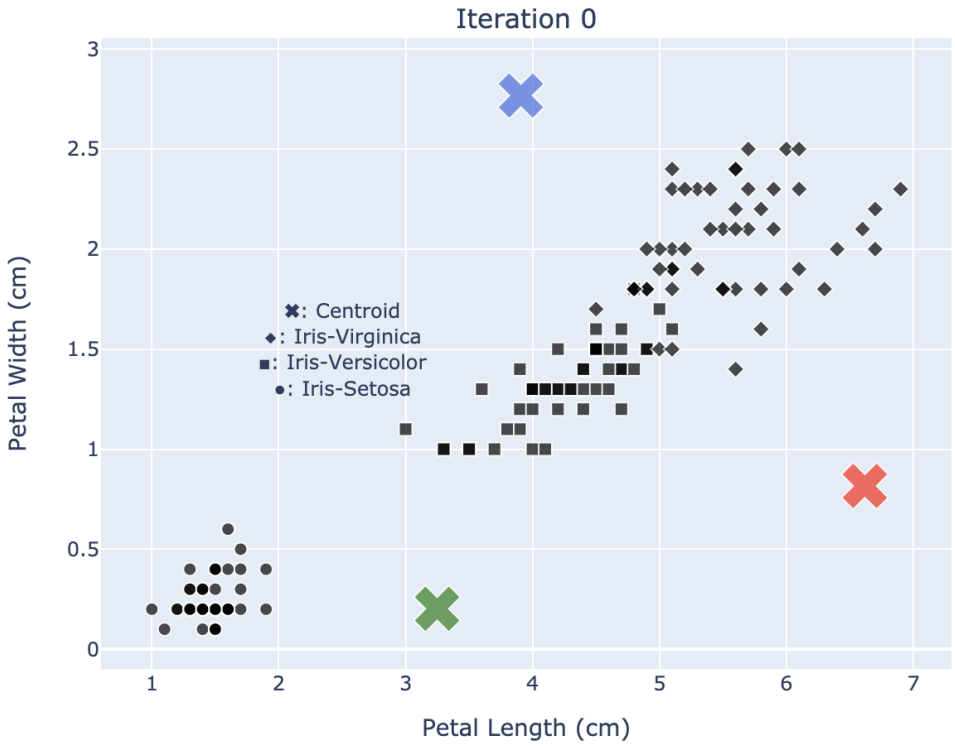
- 6.1,2.8,4.0,1.3,Iris-versicolor
- 6.2,3.4,5.4,2.3,Iris-virginica
- 4.9,3.1,1.5,0.1,Iris-setosa

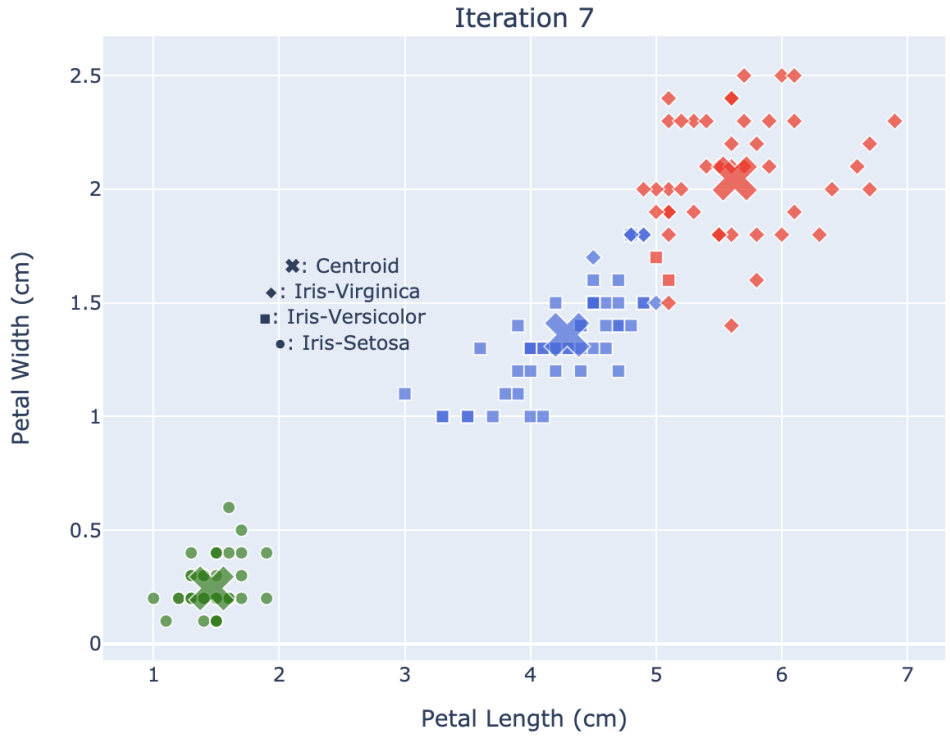
I chose to use this data set because the data were already classified into 3 types of flower species, and I thought this would be a good way to confirm the results of clustering with K=3. I decided to perform clustering on the petal_length and petal_width attributes.

The results are displayed as scatterplots for each iteration of clustering. I used the plotly python library to construct the scatterplots. The plots use symbolic data points to denote flower species (diamond: virginica, square: versicolor, circle: setosa). Colored data points indicate membership to the cluster of the centroid with the same color (red, green, or blue). There are 3 .html files attached which show the full results.

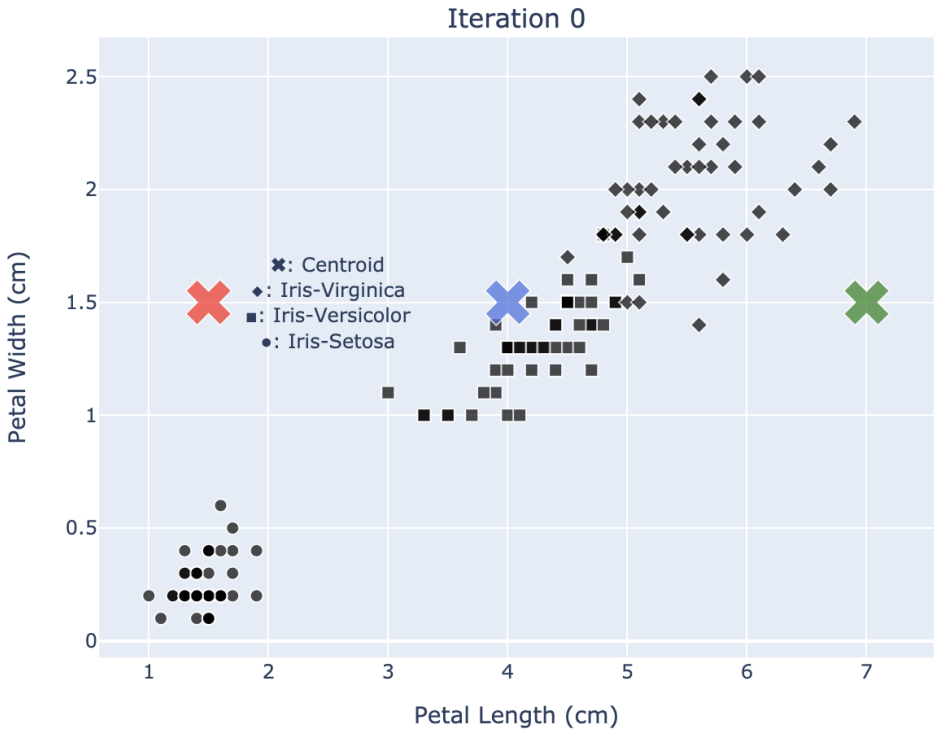
The python script uses 7 iterations of clustering. On each iteration, the data are assigned to the cluster of their nearest centroids, then the next centroids are calculated using the Euclidean midpoints of the clusters. To initialize the centroids, I tried 3 different methods:

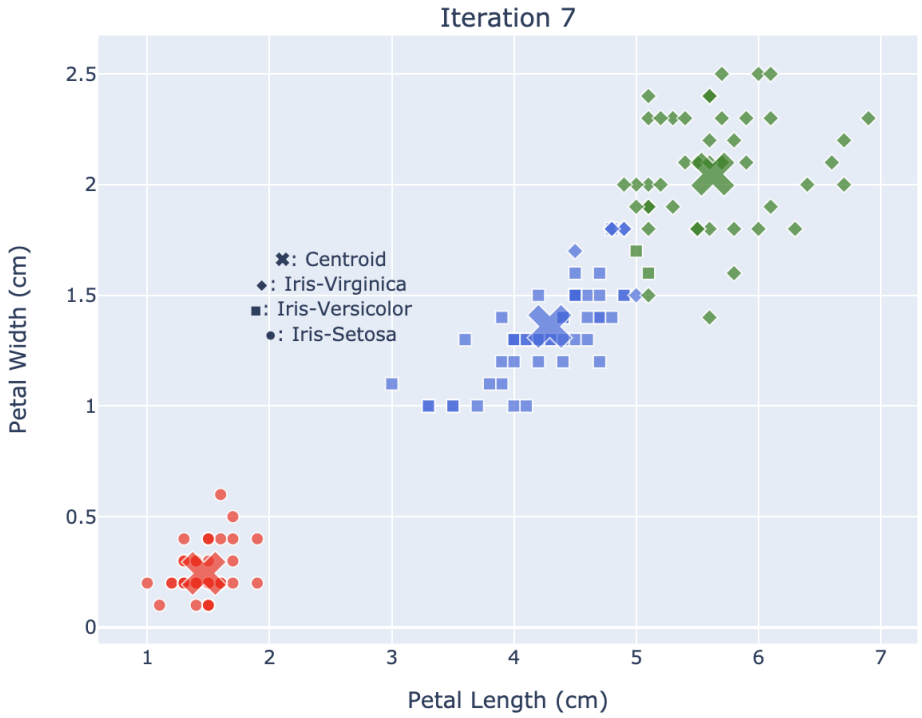
1. **Randomly initialized centroids:** the centroids (represented by X's below) were initialized using random real-valued numbers within the petal length and width ranges. Displayed below are the 0^{th} (before clustering) and 7^{th} iterations of one instance of clustering using this method (full results in random.html):



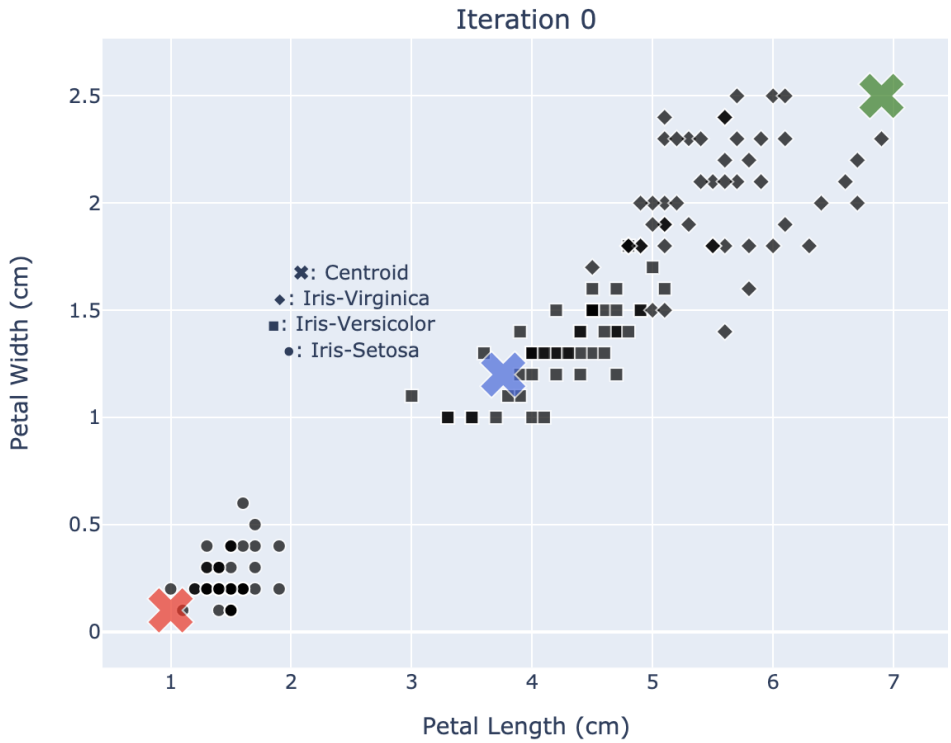


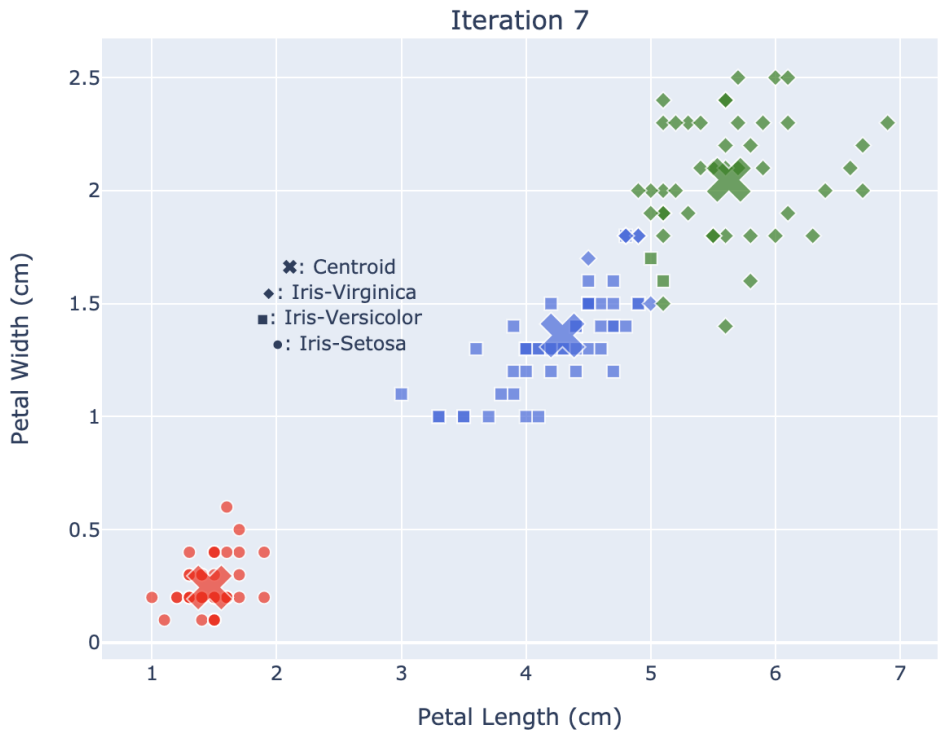
2. **Guess initialized centroids:** the centroids were initialized using a lazy guess (full results in guess.html):





3. **Smart initialized centroids:** the centroids were initialized using the minimum, average, and maximum values of the petal lengths and widths. This was only an option because the data were classified into 3 categories, but the clustering was effective (full results in smart.html):





Each method of clustering led to similar results in the final clusters.

My goal was to design a clustering algorithm that clustered flower samples according to their species, and the results show that the algorithm was extremely effective. For each method described above, less than 1% of the data were placed in clusters in which their species was the minority of the cluster (for example, in the plot immediately above, there are 4 blue diamonds; these represent Iris-Virginica samples (diamonds) that were erroneously clustered with Iris-Versicolor samples (squares)).