Alex Kitsul
230134210
CPSC 450
Assignment 6 - Report
April 1

Alex Kitsul
230134210
April 1, 2022

Psedocode

```
main:
    k <- from file
    m <- from file
    points <- from file as format (x, y)

    centers <- LloydsAlgorithm(points, k)
    print(centers)

LloydsAlgorithm(data, k):
    centers = random k centers
    clusters = dict() with center indexes as keys and empty lists as values

    while(!converged):
        for point in data:
            closest_center = None
            for center in centers:
                if(distance(point, center) < distance(point, closest_center)):
                    closest_center = center

            clusters[closest_center].append(point)

        for center in clusters:
            new_center = mean of all points in cluster
            if centers are the same:
                converged = True
            centers[center] = new_center
            clusters[center] = []

    return centers
```

Program Code

LloydsAlgorithm.py

```
from math import dist
from numpy import mean, round_

def main():
    """
        Main driver method

        Parameters:

            None

        Returns:

            None. Side effect.
    """
    file_name = input("Enter the file name: ")

    with open(file_name, "r") as file:
        first_line = file.readline().split(" ")

        k = int(first_line[0])
        m = int(first_line[1])
        points = []
        for i in file:
            coords = tuple(float(j) for j in i.split())
            points.append(coords)

        centers = LloydsAlgorithm(points, k)
        for i in centers:
            print(i)

def LloydsAlgorithm(data, k):
    """
        Execute the Lloyds Algorithm to find k centers

        Parameters:

            data (list[tuple]): A list of tuples representing the data points
            k (int): The number of centers to find

        Returns:

            centers (list[tuple]): A list of tuples representing the centers
    """
```

```
    flag = True
    # Initialize centers
    centers = []
    clusters = dict()
    # Choose random points as first centers
    for i in range(k):
        centers.append(data[i])
        clusters[i] = []

    # While we haven't yet converged
    while(flag):
        # For each point
        for i in range(len(data)):
            closest_center_index = None
            closest_center_dist = None
            # For each center
            for j in range(len(centers)):
                # Calculate distance from point to center, if the center is closer, update
                if (closest_center_dist is None or dist(data[i], centers[j]) < \
                closest_center_dist):
                    closest_center_index = j
                    closest_center_dist = dist(data[i], centers[j])

            # Add point to closest center index
            clusters[closest_center_index].append(data[i])

        # Update centers
        flag = False
        for i in clusters:
            # Calculate the mean of the points in the cluster
            new_center = tuple(round_(mean(clusters[i], axis=0), decimals=3))
            # If it's different, then we need to continue
            if (new_center != centers[i]):
                flag = True
            # Update the center
            centers[i] = new_center
            # Clear the clusters
            clusters[i] = []

    return centers

if __name__ == "__main__":
    main()
```

Examples with Output