K-means clustering from scratch with Python

Marcelo Fernandes Wellington, New Zealand https://marcelofern.com

I. INTRODUCTION

K-means clustering is a data-clustering algorithm where the number of clusters is pre-selected by the user. The algorithm will then find "K" different clusters that split the data based on a definition of distance.

The algorithm can be broken down into the following steps:

- 1) Select "K" random data points to be the center of the initial clusters. These centers will be called "centroids".
- 2) For each data point in the dataset, find the closest centroid based on a definition of distance. The Euclidean distance is a common choice.
- 3) Calculate the new centroids by finding the average point in a given cluster.
- 4) Repeat steps 2. and 3. until the algorithm converges, i.e., the new centroids are the same as the ones from the previous interaction.

March 28, 2021

II. EXAMPLE

In this example we will calculate and plot each step of the K-means clustering algorithm for K=3 and the following dataset:

TABLE I EXAMPLE DATASET

Datapoint	X	Y
A1	2	10
A2	2 8	5
A3		4
A4	5	8
A5	7	5
A6	6	4
A7	1	2
A8	4	9

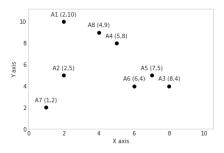


Fig. 1. Dataset in a 2d plot

III. SELECTING 3 RANDOM DATA POINTS TO BE THE INITIAL CENTROIDS

Three data points can be randomly selected using the following Python code:

```
import random
# pick 3 numbers from 1 to 9 (not inclusive)
indexes = random.sample(range(1, 9),3)
indexes
>>> [1, 4, 7]
```

This means that A1, A4, and A7 have been selected to be the initial centroids.

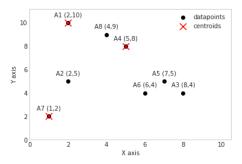


Fig. 2. Initial centroids

IV. FINDING WHICH POINTS BELONG TO EACH CLUSTER

We calculate the Euclidean distance from each data point to each centroid. A point will belong to the closest cluster.

The distance between the points and the centroids will be calculated using the Euclidean distance formula.

```
def distance(v1, v2):
    """
    Returns the Euclidean distance between two
    2d vectors.
    """
    x_distance = (v1[0] - v2[0])**2
    y_distance = (v1[1] - v2[1])**2
    return (x_distance + y_distance)**0.5
```

The initial clusters are showed in Fig.3

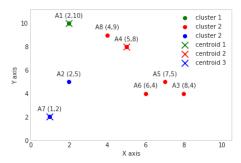


Fig. 3. Initial clusters

V. CALCULATE THE NEW CENTROIDS

New centroids are calculated by finding the average point for a given cluster:

```
def new_centroid(cluster):
    length = len(cluster)
    mean_x = sum([x for x, _ in cluster])/length
    mean_y = sum([y for _, y in cluster])/length
    return [mean_x, mean_y]
```

The new centroids will therefore be:

```
centroid_1 = [2.0, 10.0]
centroid_2 = [6.0, 6.0]
centroid_3 = [1.5, 3.5]
```

We are now going to repeat the last two steps until the algorithm reaches convergence.

VI. SECOND INTERACTION

New centroids:

```
centroid_1 = [3.0, 9.5]
centroid_2 = [6.5, 5.25]
centroid_3 = [1.5, 3.5]
```

VII. THIRD INTERACTION

New centroids:

```
centroid_1 = [3.6666666666666665, 9.0]
centroid_2 = [7.0, 4.333333333333333]
centroid_3 = [1.5, 3.5]
```

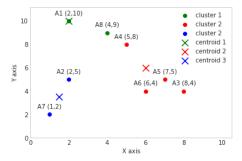


Fig. 4. New centroids, second interaction

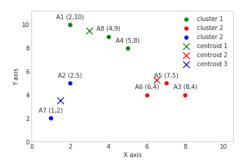


Fig. 5. New centroids, third interaction

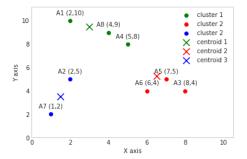


Fig. 6. Final clusters

VIII. FORTH INTERACTION (CONVERGENCE)

New centroids:

```
centroid_1 = [3.6666666666666665, 9.0]
centroid_2 = [7.0, 4.33333333333333]
centroid_3 = [1.5, 3.5]
```

IX. APPENDIX

A. Code for generating Fig. 1:

```
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
x_{coordinates} = [2, 2, 8, 5, 7, 6, 1, 4]
y_{coordinates} = [10, 5, 4, 8, 5, 4, 2, 9]
points = [
    'A1','A2','A3','A4',
    'A5','A6','A7','A8'
# graph configuration
plt.grid(b=None)
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.xlim([0,10.5])
plt.ylim([0,11.2])
# labels
iterator = zip(
    points,
    x_coordinates,
    y_coordinates
for point, x, y in iterator:
    label = f''(point) (\{x\}, \{y\})''
    plt.annotate(
         label,
         (x,y),
```

```
textcoords="offset points",
                                                               [5,8], [7,5], [6,4],
        xytext=(0,10),
                                                               [1,2], [4,9]
        ha='center'
                                                           x_coordinates = [point[0] for point in data_points]
                                                           y_coordinates = [point[1] for point in data_points]
fig = plt.plot(
    x_coordinates,
    y_coordinates, 'o', color='black'
                                                           centroid_1 = [2,10]
                                                           centroid_2 = [5,8]
                                                           centroid_3 = [1,2]
B. Code for generating Fig. 2:
                                                           centroids = [centroid_1, centroid_2, centroid_3]
import matplotlib.pyplot as plt
                                                           cluster_1 = []
%matplotlib inline
                                                           cluster_2 = []
plt.style.use('seaborn-whitegrid')
                                                           cluster_3 = []
x_{coordinates} = [2, 2, 8, 5, 7, 6, 1, 4]
                                                           def distance(v1, v2):
y_{coordinates} = [10, 5, 4, 8, 5, 4, 2, 9]
                                                               return (
centroids_x = [2,5,1]
                                                                    (v1[0] - v2[0])**2
centroids_y = [10, 8, 2]
                                                                   + (v1[1] - v2[1]) **2
points = [
                                                               ) * * 0.5
   'A1', 'A2', 'A3', 'A4',
    'A5','A6','A7','A8'
                                                           for point in data_points:
1
                                                               dist_c1 = distance(point, centroids[0])
                                                               dist_c2 = distance(point, centroids[1])
# plot!
                                                               dist_c3 = distance(point, centroids[2])
fig = plt.figure()
                                                               min_distance = min([dist_c1, dist_c2, dist_c3])
ax = fig.add_subplot(111)
                                                               if min_distance == dist_c1:
                                                                   cluster_1.append(point)
# graph configuration
                                                               elif min_distance == dist_c2:
plt.grid(b=None)
                                                                  cluster_2.append(point)
plt.xlabel('X axis')
                                                               else:
plt.ylabel('Y axis')
                                                                   cluster_3.append(point)
plt.xlim([0,10.5])
plt.ylim([0,11.2])
                                                           cluster_1_x = [point[0] for point in cluster_1]
                                                           cluster_2_x = [point[0] for point in cluster_2]
cluster_3_x = [point[0] for point in cluster_3]
iterator = zip(
    points,
    x_coordinates,
                                                           cluster_1_y = [point[1] for point in cluster_1]
    y_coordinates
                                                           cluster_2_y = [point[1] for point in cluster_2]
                                                           cluster_3_y = [point[1] for point in cluster_3]
for point, x, y in iterator:
    label = f"{point} ({x}, {y})"
                                                           # initiate the plot
    ax.annotate(
                                                           fig = plt.figure()
        label,
                                                           ax = fig.add_subplot(111)
        (x, y),
        textcoords="offset points",
                                                           # graph configuration
        xytext=(0,10),
                                                           plt.grid(b=None)
        ha='center'
                                                           plt.xlabel('X axis')
                                                           plt.ylabel('Y axis')
                                                           plt.xlim([0,10.5])
ax.scatter(
                                                           plt.ylim([0,11.2])
    x_coordinates, y_coordinates, marker='o',
    color='black', label='datapoints'
                                                           # labels
                                                           iterator = zip(
    centroids_x, centroids_y, marker='x',
color='red', label='centroids', s=120
                                                               points_names,
                                                               x_coordinates,
                                                               y_coordinates
ax.legend()
                                                           for point_name, x, y in iterator:
                                                               label = f"{point_name} ({x}, {y})"
C. Code for generating Fig. 3:
                                                               ax.annotate(
                                                                   label.
import matplotlib.pyplot as plt
%matplotlib inline
                                                                   textcoords="offset points",
plt.style.use('seaborn-whitegrid')
                                                                   xvtext=(0,10),
                                                                   ha='center'
points names = [
    'A1','A2','A3','A4',
'A5','A6','A7','A8'
                                                               )
                                                           # plot the data
                                                           ax.scatter(
                                                               cluster_1_x, cluster_1_y, marker='o',
data_points = [
                                                               color='green', label='cluster 1'
```

[2,10], [2,5], [8,4],

```
cluster_2_x, cluster_2_y, marker='\circ',
                                                       # initiate the plot
    color='red', label='cluster 2'
                                                       fig = plt.figure()
                                                       ax = fig.add_subplot(111)
ax.scatter(
   cluster_3_x, cluster_3_y, marker='o',
                                                       # graph configuration
   color='blue', label='cluster 2'
                                                       plt.grid(b=None)
                                                       plt.xlabel('X axis')
                                                       plt.ylabel('Y axis')
ax.scatter(
   centroid_1[0], centroid_1[1], marker='x',
                                                       plt.xlim([0,10.5])
    color='green', label='centroid 1', s=120
                                                       plt.ylim([0,11.2])
)
ax.scatter(
   centroid_2[0], centroid_2[1], marker='x',
                                                       # labels
   color='red', label='centroid 2', s=120
                                                       iterator = zip(
                                                           points_names,
                                                           x_coordinates,
ax.scatter(
   centroid_3[0], centroid_3[1], marker='x',
                                                           y_coordinates
   color='blue', label='centroid 3', s=120
                                                        for point_name, x, y in iterator:
ax.legend()
                                                           label = f''{point_name} ({x},{y})"
                                                           ax.annotate(
                                                               label.
D. Code for generating Fig. 6:
                                                               (x,y),
                                                               textcoords="offset points",
                                                               xytext=(0,10),
import matplotlib.pyplot as plt
                                                               ha='center'
%matplotlib inline
plt.style.use('seaborn-whitegrid')
                                                       # plot the data
points_names = [
   'A1','A2','A3','A4',
'A5','A6','A7','A8'
                                                       ax.scatter(
                                                           cluster_1_x, cluster_1_y, marker='o',
                                                           color='green', label='cluster 1'
1
                                                       ax.scatter(
data_points = [
                                                           cluster_2_x, cluster_2_y, marker='o',
   [2,10], [2,5], [8,4], [5,8],
                                                           color='red', label='cluster 2'
    [7,5], [6,4], [1,2], [4,9]
                                                       )
                                                       ax.scatter(
x_coordinates = [point[0] for point in data_points]
                                                           cluster_3_x, cluster_3_y, marker='o',
y_coordinates = [point[1] for point in data_points]
                                                           color='blue', label='cluster 2'
centroid_1 = [3.6666666666666666, 9.0]
                                                       ax.scatter(
centroid_1[0], centroid_1[1], marker='x',
centroid_3 = [1.5, 3.5]
                                                           color='green', label='centroid 1', s=120
centroids = [centroid_1, centroid_2, centroid_3]
                                                       ax.scatter(
                                                           centroid_2[0], centroid_2[1], marker='x',
cluster_1 = []
                                                           color='red', label='centroid 2', s=120
cluster_2 = []
                                                       )
cluster_3 = []
                                                        ax.scatter(
                                                           centroid_3[0], centroid_3[1], marker='x',
def distance(v1, v2):
                                                           color='blue', label='centroid 3', s=120
   return ( (v1[0] - v2[0]) **2 + (v1[1] - v2[1]) **2
    ) * * 0.5
                                                       ax.legend()
for point in data_points:
                                                       def new_centroid(cluster):
    dist_c1 = distance(point, centroids[0])
                                                           length = len(cluster)
   dist_c2 = distance(point, centroids[1])
                                                           mean_x = sum([x for x, _ in cluster])/length
   dist_c3 = distance(point, centroids[2])
                                                           mean_y = sum([y for _, y in cluster])/length
   min_distance = min([dist_c1, dist_c2, dist_c3])
                                                           return [mean_x, mean_y]
    if min_distance == dist_c1:
       cluster_1.append(point)
                                                       print (new_centroid(cluster_1))
    elif min_distance == dist_c2:
                                                       print (new_centroid(cluster_2))
       cluster_2.append(point)
                                                       print (new_centroid(cluster_3))
    else:
       cluster_3.append(point)
cluster_1_x = [point[0] for point in cluster_1]
cluster_2_x = [point[0] for point in cluster_2]
cluster_3_x = [point[0] for point in cluster_3]
cluster_1_y = [point[1] for point in cluster_1]
cluster_2_y = [point[1] for point in cluster_2]
```

)

cluster_3_y = [point[1] for point in cluster_3]