Machine Learning

Project II

By 費蓋德 0516251

Environment used in this project

Language: Python3.7

I acknowledge that my code was a bit messy for the last assignment, now to make everything more clear, I use **JupyterLab** as my working environment.

Deepcopy, numpy, pandas, matplotlib are the packages required to run the program.

Cost function and accuracy

K-means is a clustering method, finding its accuracy is not a straightforward work. However, I calculated the **error** for each iteration and the **sum of squared distance** of the points :

As you can see from the image, the error went from 20 to 0 and the SSD is also provided.

```
centers_old = np.zeros(centers.shape) # to store old centers
centers_new = deepcopy(centers) # Store new centers

clusters = np.zeros(n)
distances = np.zeros((n,k))

error = np.linalg.norm(centers_new - centers_old)
print(error)
sum = 0
# iterate till error is null
while error != 0:
    # Measure the distance to every center
    for i in range(k):
        distances[:,i] = np.linalg.norm(data - centers[i], axis=1)
        sum += distances[:,i]*distances[:,i]
#          print(distances[:,i])
    # Assign all training data to closest center
    clusters = np.argmin(distances, axis = 1)
    centers_old = deepcopy(centers_new)
    # Calculate mean for every cluster and update the center
    for i in range(k):
        centers_new[i] = np.mean(data[clusters == i], axis=0)
    error = np.linalg.norm(centers_new - centers_old)
    print(error)
print(sum)
```

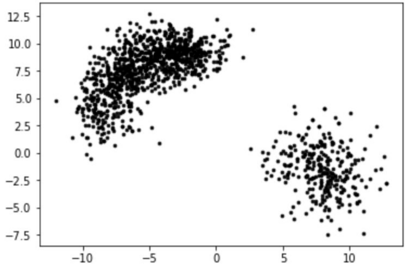
```
20.216760277218874
5.428226014063211 5
0.0
[ 179.07624185  187.69221871  181.17209647 ... 1576.24954381  156.64708247
  201.21996227]
```

Result of K-means clustering

We first plotted the raw data, categorize it into FF (green), CH(orange) and CU(blue) and converge the centroids while minimizing the error

```
[13]: # Getting the values and plotting it
      f1 = data['x'].values
      f2 = data['y'].values
      category = data['pitch_type'].values
      data = np.array(list(zip(f1, f2)))
      plt.scatter(f1, f2, c='black', s=7)

[13]: <matplotlib.collections.PathCollection at 0x11edc7588>
```



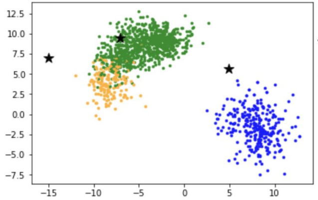
Raw data

```
# Number of clusters
k = 3
# Number of training data
n = data.shape[0]
# Number of features in the data
c = data.shape[1]
# Generate random centers
mean = np.mean(data, axis = 0)
std = np.std(data, axis = 0)
centers = np.random.randn(k,c)*std + mean

# Plot the data and the centers generated as random
colors=['orange', 'blue', 'green']

for i in range(n):
    plt.scatter(data[i, 0], data[i,1], s=7, color = colors[int(category[i])])
plt.scatter(centers[:,0], centers[:,1], marker='*', c='black', s=150)

<matplotlib.collections.PathCollection at 0x122cfb320>
```

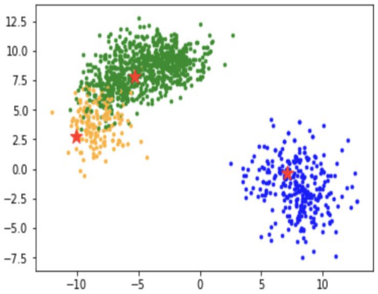


With random centroids

Random Centroids

```
# Plot the data and the centers
colors=['orange', 'blue', 'green']
for i in range(n):
    plt.scatter(data[i, 0], data[i,1], s=7, color = colors[int(category[i])])
plt.scatter(centers_new[:,0], centers_new[:,1], marker='*', c='red', s=150)

<matplotlib.collections.PathCollection at 0x1239207f0>
```

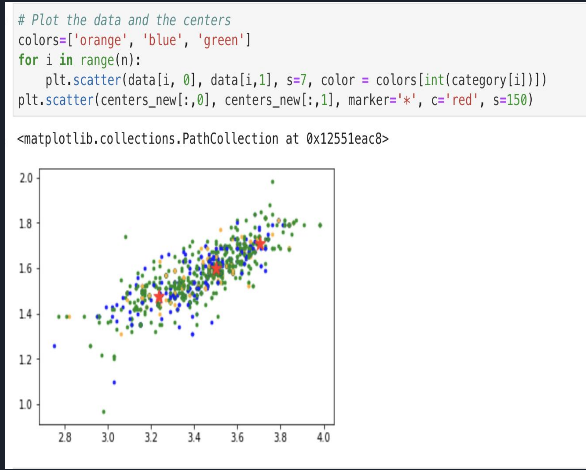


Final result (code in the previous slide)

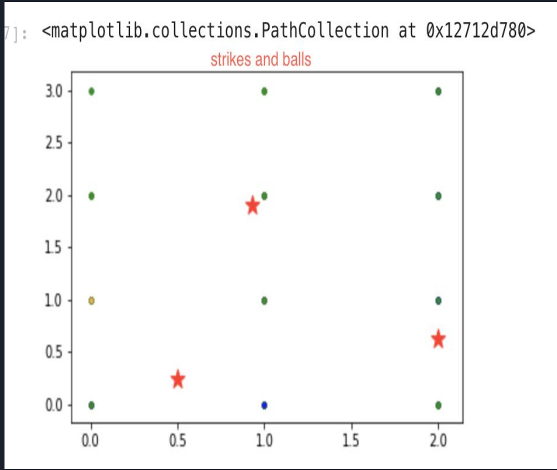
Use of other attributes and the reason of k=3

I tried using other attributes but none of them are as accurate as the attributes x and y.

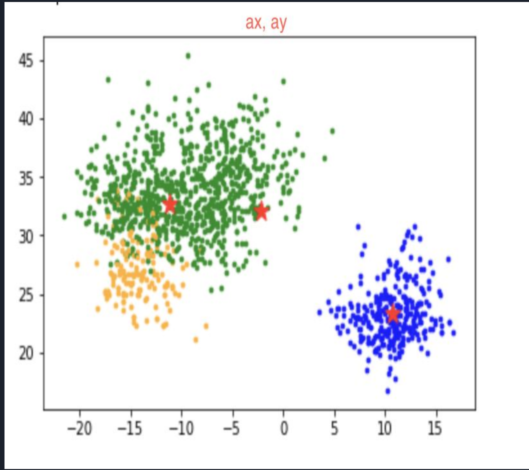
Here are some of them:



sz_top and sz_bot



strikes and balls



ax, ay (not very far)

*We use k=3 because we needed to check for 3 pitch types FF, CH and CU. So the right number of clusters should be 3. Any k less than 3 or greater than 3 would be inaccurate.

Kd-tree: code + result

Kd-tree function (wikipedia)

```
[41]: from collections import namedtuple
from operator import itemgetter
from pprint import pformat

class Node(namedtuple('Node', 'location left_child right_child')):

    def __repr__(self):
        return pformat(tuple(self))

def kdtree(point_list, depth=0):

    # assumes all points have the same dimension
    try:
        k = len(point_list[0])
    except IndexError:
        return None

    # Select axis based on depth so that axis cycles through
    # all valid values
    axis = depth % k

    # Sort point list and choose median as pivot element
    point_list.sort(key=itemgetter(axis))
    median = len(point_list) // 2         # choose median

    # Create node and construct subtrees
    return Node(
        location=point_list[median],
        left_child=kdtree(point_list[:median], depth + 1),
        right_child=kdtree(point_list[median + 1:], depth + 1)
    )
```

Result of the 2-d tree

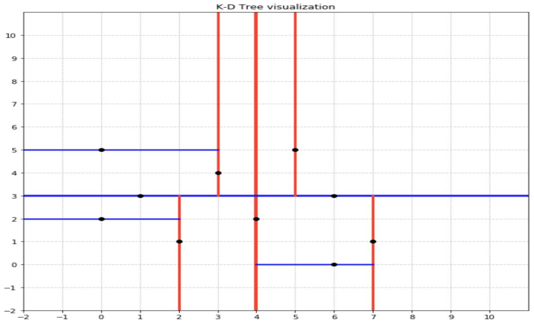
```
import pandas as pd
import random
import numpy as np

min_val = 0   # minimal coordinate value
max_val = 9  # maximal coordinate value

df = pd.read_csv('points.txt', header=None)
#     points = points.apply(pd.to_numeric)
df.columns = ['xy']
df['x'], df['y'] = df['xy'].str.split(' ', 1).str
df = df.drop(['xy'], axis=1)
df = df.apply(pd.to_numeric)

#     print(df)
#     print(df.shape)
point_list = [tuple(x) for x in df.values]
# construct a K-D tree
tree = kdtree(point_list)
print(tree)
```

```
((4, 2),
 ((1, 3),
 ((2, 1), ((0, 2), None, None), None),
 ((3, 4), ((0, 5), None, None), None)),
 ((6, 3), ((7, 1), ((6, 0), None, None), None), ((5, 5), None, None)))
```



Function to visualize

```
import matplotlib.pyplot as plt

# line width for visualization of K-D tree
line_width = [4., 3.5, 3., 2.5, 2., 1.5, 1., .5, 0.3]
delta = 2
def plot_tree(tree, min_x, max_x, min_y, max_y, prev_node, branch, depth=0):
    cur_node = tree.location         # current tree's node
    left_branch = tree.left_child    # its left branch
    right_branch = tree.right_child  # its right branch

    # set line's width depending on tree's depth
    if depth > len(line_width)-1:
        ln_width = line_width[len(line_width)-1]
    else:
        ln_width = line_width[depth]

    k = len(cur_node)
    axis = depth % k

    # draw a vertical splitting line
    if axis == 0:

        if branch is not None and prev_node is not None:

            if branch:
                max_y = prev_node[1]
            else:
                min_y = prev_node[1]

        plt.plot([cur_node[0],cur_node[0]], [min_y,max_y], linestyle='-', color='red', linewidth=ln_width)

    # draw a horizontal splitting line
    elif axis == 1:

        if branch is not None and prev_node is not None:

            if branch:
                max_x = prev_node[0]
            else:
                min_x = prev_node[0]

        plt.plot([min_x,max_x], [cur_node[1],cur_node[1]], linestyle='-', color='blue', linewidth=ln_width)

    # draw the current node
    plt.plot(cur_node[0], cur_node[1], 'ko')

    # draw left and right branches of the current node
    if left_branch is not None:
        plot_tree(left_branch, min_x, max_x, min_y, max_y, cur_node, True, depth+1)

    if right_branch is not None:
        plot_tree(right_branch, min_x, max_x, min_y, max_y, cur_node, False, depth+1)

plt.figure("K-d Tree", figsize=(10., 10.))
plt.axis( [min_val-delta, max_val+delta, min_val-delta, max_val+delta] )

plt.grid(b=True, which='major', color='0.75', linestyle='--')
plt.xticks([i for i in range(min_val-delta, max_val+delta, 1)])
plt.yticks([i for i in range(min_val-delta, max_val+delta, 1)])

# draw the tree
plot_tree(tree, min_val-delta, max_val+delta, min_val-delta, max_val+delta, None, None)

plt.title('K-D Tree visualization')
plt.show()
plt.close()
```

Conclusion

In this assignment, we understand and learn how to implement k-means and kd-tree algorithms. K-means is a clustering method (unsupervised learning) where the task is finding label in unlabelled data and classify the data into clusters. We found out that the data we were dealing with required that we use 3 clusters. This task is for the data scientist to decide and it can be tricky. A k-d tree, or k-dimensional tree, is a data structure for organizing some number of points in a space with k dimensions. It is a binary search tree with other constraints imposed on it. K-d trees are very useful for range and nearest neighbor searches. We constructed a 2-d tree for the points given and plotted the result.

I am really sorry for the fact I have to do individual work. I know you are busy and you would have less homeworks to review if it was a group. But I had no choice since me and my teammates could not find a common ground to work together.

Thanks again for your understanding

The zip file includes 0516251_kmeans.ipynb, 0516251_kdtree.ipynb and the datasets