Homework 3 - Clustering

Name: < insert name here >

Remember that you are encouraged to discuss the problems with your instructors and classmates, but **you must write all code and solutions on your own**.

The rules to be followed for the assignment are:

- Do **NOT** load additional packages beyond what we've shared in the cells below.
- Some problems with code may be autograded. If we provide a function or class API **do not** change it.
- Do not change the location of the data or data directory. Use only relative paths to access the data.

In [1]:
```
import argparse
import pandas as pd
import numpy as np
import pickle
from pathlib import Path
from collections import defaultdict
```

[10 points] Problem 1 - K Means Clustering

A sample dataset has been provided to you in the './data/sample_dataset_kmeans.pickle' path. The centroids are in './data/sample_centroids_kmeans.pickle' and the sample result is in './data/sample_result_kmeans.pickle' path. You can use these to test your code.

Here are the attributes for the dataset. Use this dataset to test your functions.

- Dataset should load the points in the form of a list of lists where each list item represents a point in the space.
- An example dataset will have the following structure. If there are 3 points in the dataset, this would appear as follows in the list of lists.

```
dataset = [
    [5,6],
    [3,5],
    [2,8]
]
```

Note:

- A sample dataset to test your code has been provided in the location "data/sample_dataset_kmeans.pickle". Please maintain this as it would be necessary while grading.
- Do not change the variable names of the returned values.
- After calculating each of those values, assign them to the corresponding value that is being returned.

In [16]:
```
def k_means_clustering(centroids, dataset):

#   Description: Perform k means clustering for 2 iterations given as inpu
t the dataset and centroids.
#   Input:
#       1. centroids - A list of lists containing the initial centroids fo
r each cluster.
#       2. dataset - A list of lists denoting points in the space.
#   Output:
#       1. results - A dictionary where the key is iteration number and st
ore the cluster assignments in the
#           appropriate clusters. Also, update the centroids list after ea
ch iteration.

    result = {
        '1': { 'cluster1': [], 'cluster2': [], 'cluster3': [], 'centroids'
: []},
        '2': { 'cluster1': [], 'cluster2': [], 'cluster3': [], 'centroids'
: []}
    }

    centroid1, centroid2, centroid3 = centroids[0], centroids[1], centroid
s[2]

    for iteration in range(2):
        # your code here
        # 1. find data points closest to initial centroids
            # i) compare difference between each datapoint and centroid
            # ii) Store min values to respective centroid group
        # 2. find mean of the data points and store into result
        cluster1 = []
        cluster2 = []
        cluster3 = []

        for coord in (dataset):

            diff1 = abs(np.linalg.norm(np.array((centroid1)) - np.array((c
oord))))
            diff2 = abs(np.linalg.norm(np.array((centroid2)) - np.array((c
oord))))
            diff3 = abs(np.linalg.norm(np.array((centroid3)) - np.array((c
oord))))

            if min(diff1, diff2, diff3) == diff1:
                cluster1.append(coord)

            elif min(diff1, diff2, diff3) == diff2:
                cluster2.append(coord)
```

```
            else:
                cluster3.append(coord)


        tempx1 = [x1[0] for x1 in cluster1]
        tempx2 = [x2[0] for x2 in cluster2]
        tempx3 = [x3[0] for x3 in cluster3]
        tempy1 = [y1[1] for y1 in cluster1]
        tempy2 = [y2[1] for y2 in cluster2]
        tempy3 = [y3[1] for y3 in cluster3]

        tempcent1 = [np.mean(tempx1), np.mean(tempy1)]
        tempcent2 = [np.mean(tempx2), np.mean(tempy2)]
        tempcent3 = [np.mean(tempx3), np.mean(tempy3)]
        tempcents = [tempcent1, tempcent2, tempcent3]

        centroid1, centroid2, centroid3 = tempcent1, tempcent2, tempcent3

        result[str(iteration + 1)]['cluster1'] = cluster1
        result[str(iteration + 1)]['cluster2'] = cluster2
        result[str(iteration + 1)]['cluster3'] = cluster3
        result[str(iteration + 1)]['centroids'] = tempcents

    return result
```

In [18]:
```
# This cell has hidden test cases that will run after you submit your assi
gnment.
```

[10 points] Problem 2 - Clustering using EM Method

A sample dataset has been provided to you in the './data/sample_dataset_em.pickle' path. The centroids are in './data/sample_centroids_em.pickle' and the sample result is in './data/sample_result_em.pickle' path. You can use these to test your code.

Here are the attributes for the dataset. Use this dataset to test your functions.

- Dataset should load the points in the form of a list of lists where each list item represents a point in the space.
- An example dataset will have the following structure. If there are 3 points in the dataset, this would appear as follows in the list of lists.

```
dataset = [5,7,9]
```

Note:

- A sample dataset to test your code has been provided in the location "data/em_dataset.pickle". Please maintain this as it would be necessary while grading.
- Do not change the variable names of the returned values.
- After calculating each of those values, assign them to the corresponding value that is being returned.

In [32]:
```
import numpy as np
import math

def em_clustering(centroids, dataset):

#   Input:
#       1. centroids - A list of lists with each value representing the me
an and standard deviation values picked from a gausian distribution.
#       2. dataset - A list of points randomly picked.
#   Output:
#       1. results - Return the updated centroids(updated mean and std val
ues after the EM step) after the first iteration.

    new_centroids = list()

    # your code here
    # E-step
    p_temp_n = []
    p1 = []
    p2 = []
    mu1 = []
    mu2 = []
    sigma1 = []
    sigma2 = []
    for idx, x in enumerate(dataset):
        p_temp = []


        for stat in centroids:
            mu, sigma = stat[0], stat[1]
            p = (1 / (sigma * np.sqrt(2 * 3.14))) * math.e**(-0.5 * ((x -
mu)/sigma)**2)
            p_temp.append(p)

        p1.append(p_temp[0] / sum(p_temp))   # normalized p values for eac
h cluster
        p2.append(p_temp[1] / sum(p_temp))
    # M-step

    # for loop if there are more than two centroids
        mu1.append(p1[idx] * x)
        mu2.append(p2[idx] * x)

    mu1_n = sum(mu1) / sum(p1)
    mu2_n = sum(mu2) / sum(p2)
    print('mu1_n', mu1_n)
    print('mu2_n', mu2_n)

    for i, point in enumerate(dataset):
        s1 = p1[i]*(point-mu1_n)**2
        s2 = p2[i]*(point-mu2_n)**2
        sigma1.append(s1)
        sigma2.append(s2)

    sigma1_n = np.sqrt((sum(sigma1)) / sum(p1))
    sigma2_n = np.sqrt((sum(sigma2)) / sum(p2))
    print('sigma2_n', sigma2_n)
```

```
        #mu1_n =
        #sigma1_n.append(np.sqrt((p1[idx] * (x-mu1_n)

    new_centroids = [[mu1_n, sigma1_n], [mu2_n, sigma2_n]]

    return new_centroids
```

In [34]:
```
dataset = pd.read_pickle( './data/sample_dataset_em.pickle')
centroids = pd.read_pickle('./data/sample_centroids_em.pickle')
result = pd.read_pickle( './data/sample_result_em.pickle')

print('dataset', dataset, '\ncentroids', centroids, '\result', result)
em_clustering(centroids, dataset)
```

```
dataset [8, 16, 13, 9, 18, 15, 7, 5, 7, 3]
esult [[13.346550530668159, 3.236599802533008], [7.9971108077796735, 4.473
417525043109]]
mu1_n 13.346550530668159
mu2_n 7.997110807779673
sigma2_n 4.473417525043108
```

Out[34]:
```
[[13.346550530668159, 3.2365998025330085],
 [7.997110807779673, 4.473417525043108]]
```

In [ ]:
```
# This cell has hidden test cases that will run after you submit your assi
gnment.
```