Assignment 4

I have used the following simplification:

- y__2n = 1-y__1n
- pi__1 = sum(y__1n)/sum(y__1n+y__2n) = sum(y__1n)/sum(y__1n+1-y__1n) =sum(y__1n)/n
- Took my initial guess of mu as mean of the data generated for faster convergence.
- Similarly took the initial variance as the variance of the data.

Sample output:

True Parameters: pi__1 = 0.7, mu__1 = 1.0, mu__2 = 4.0, v = 0.5

Estimated Parameters: pi__1 = 0.7036487636639586, mu__1 = 1.0110600112242416, mu__2 = 4.02350646764783, v = 0.5017586475461915

%Error in estimation : pi__1 =0.5212519519940871 %, mu__1 = 1.1060011224241606 %, mu__2 = 0.5876616911957511 %, v =0.3517295092382966 %

```
#Importing library
import numpy as np; #For mathematical operations
from tqdm import tqdm; # Progress bar

def generate_data(n, pi_1, mu_1, mu_2, v): #Generating gaussian random data.
    data = np.zeros(n) # Store the random data here
    for i in range(n): # loop from 0 to n-1 to generate n random gaussian mixture da
ta.
        if np.random.rand() < pi_1: # Binary choice
            data[i] = np.random.normal(mu_1, np.sqrt(v)) # Generate the data for fir
st gaussian
        else: # if 1-pi_1 generate the second gaussian
            data[i] = np.random.normal(mu_2, np.sqrt(v)) # Generate the 2nd gaussian
    return data # Return the data generated

# Initial guess
pi_1 = 0.7
mu_1 = 1.0
mu_2 = 4.0
v = 0.5
n=10000;

#Generate random data
data = generate_data(n, pi_1, mu_1, mu_2, v)

pi_1_hat = 0.5; #some random guess
mu_1_hat = np.mean(data) #taking mu_1_old close to the mean so that we get the conve
rgence faster.
mu_2_hat = np.mean(data)+np.random.rand() #adding some random number from the mean f
or faster convergence.
v_hat = np.var(data) # obtaining the variance of the data again for the faster conve
rgence.

for i in tqdm(range(100)): #Looping through 100 steps
    # E step
    # Calculating the y1_old as z using equation from example 3.7
    z = pi_1_hat * np.exp(-0.5 * (data - mu_1_hat)**2 / v_hat) / np.sqrt(2*np.pi*v_h
at) \
    / ((1-pi_1_hat) * np.exp(-0.5 * (data - mu_2_hat)**2 / v_hat) / np.sqrt(2*np.pi*
v_hat) \
        + pi_1_hat * np.exp(-0.5 * (data - mu_1_hat)**2 / v_hat) / np.sqrt(2*np.pi*v
_hat))
    # M step
    pi_1_hat = np.sum(z) / n # Calculating new p1
    mu_1_hat = np.sum(z * data) / np.sum(z) # Calculating new mu_2
    mu_2_hat = np.sum((1 - z) * data) / np.sum(1 - z) # Calculating new mu_2
    v_hat = np.sum(z * (data - mu_1_hat)**2) / np.sum(z) #Calculating new v

print("True Parameters: pi_1 = {}, mu_1 = {}, mu_2 = {}, v = {}".format(pi_1, mu_1,
mu_2, v))#Print original
print("Estimated Parameters: pi_1 = {}, mu_1 = {}, mu_2 = {}, v = {}".format(pi_1_ha
t, mu_1_hat, mu_2_hat, v_hat))#Print obtained values
print("%Error in estimation : pi_1 ={} %, mu_1 = {} %, mu_2 = {} %, v ={} %".format(
np.abs(pi_1-pi_1_hat)*100/pi_1,np.abs(mu_1-mu_1_hat)*100/mu_1,np.abs( mu_2-mu_2_hat)
*100/mu_2,np.abs(v-v_hat)*100/v))
```