*"To call functions, tools from Library"*

from \_\_future\_\_ import print\_function

import datetime

import numpy as np

from matplotlib import cm, pyplot as plt

from hmmlearn.hmm import GaussianHMM, GMMHMM

from hmmlearn.base import \_BaseHMM

from mpl\_toolkits.mplot3d import Axes3D

import os

# from Cython.Includes.libcpp import pair

*"To call function of output Data"*

import os,sys

# import subprocess

# import glob

# from os import path

*"INPUT"*

#number of state

n = 16

#covariance type

covar\_type = *"full"*

#number of iteration

iterr = 1000

#figure name

figname1 = *"RTV600mVCurrent9n%d"* % n

# figname2 = "result\_\_foldernameanalysis1\_3d\_scatterplot"

figname3 = *"colormapplot\_1"*

# figname4 = "result\_\_analysis1\_colormapplot\_2"

script\_dir = os.path.dirname(\_\_file\_\_)

results\_dir = os.path.join(script\_dir, *'HH3/'*)

if not os.path.isdir(results\_dir):

os.makedirs(results\_dir)

# "Output Data"

# f = open(results\_dir + 'RTV600mVCurrent9;n16;output.txt','w')

# sys.stdout = f

*"Import data from excel file"*

from xlrd import open\_workbook

book = open\_workbook(*'Data2.xlsx'*)

sheet = book.sheet\_by\_index(4)

*"Input"*

#start\_time

start\_t = 0

#end\_time

end\_t = sheet.nrows

x = []

y = []

for k in range(start\_t,end\_t):

x.append(str(sheet.row\_values(k)[0])) #[k] = row, [0] = column ke nol

y.append(str(sheet.row\_values(k)[1])) #[k] = row, [1] = column ke satu

x = np.asarray(map(float, x))

y = np.asarray(map(float, y))

X = np.reshape(y,(-1,1))

*"Run Gaussian HMM"*

# Make an HMM instance and execute fit

model = GaussianHMM(n\_components=n, covariance\_type=covar\_type, n\_iter=iterr).fit(X)

# Predict the optimal sequence of internal hidden state

hidden\_states = model.predict(X)

print(*"Number of Rows"*)

print(*"--------------"*)

print(sheet.nrows)

print(*" "*)

print(*"Hidden States"*)

print(*"-------------"*)

print(hidden\_states)

*"ordered hidden states"*

#to make list of means of each hidden states

means\_state = []

means\_state1 = []

for i in range(0,n):

means\_state.append(model.means\_[i][0])

means\_state1.append(model.means\_[i][0])

#to make the hidden states mean in well ordered

ind\_mean\_of\_hiddenstates = []

for i in range(0,n):

mm = min(means\_state)

mm\_ind = means\_state1.index(mm)

means\_state.remove(mm)

ind\_mean\_of\_hiddenstates.append([mm\_ind,model.means\_[mm\_ind][0]])

*"PRINT RESULT"*

print(*"Record data for histogram"*)

print(*"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"*)

print(*"Hidden state {}th"*,*" "*,*"Mean"*)

for i in range(0,len(ind\_mean\_of\_hiddenstates)):

print(ind\_mean\_of\_hiddenstates[i][0], *" "*, ind\_mean\_of\_hiddenstates[i][1])

#hidden states in well odered

h\_s\_o = []

for i in ind\_mean\_of\_hiddenstates:

h\_s\_o.append(i[0])

print(h\_s\_o)

*"Analysis 3 : x\_i affects x\_i+1 for x\_i is hidden state in hidden\_state"*

XY = []

pairr = []

pair\_m = []

# aa = float(format(model.means\_[i][0],'.2f'))

# bb = float(format(model.means\_[j][0],'.2f'))

# pair\_d.append([aa,bb])

for i in h\_s\_o:

aa = format(model.means\_[i][0],*'.2f'*)

XY.append(aa)

for j in h\_s\_o:

pairr.append([i,j])

pair\_m.append([model.means\_[i][0],model.means\_[j][0]])

print(pairr)

print(pair\_m)

x\_i3 = hidden\_states[:]

x\_i3 = x\_i3.tolist()

x\_i3\_plus = hidden\_states[:]

x\_i3\_plus = x\_i3\_plus.tolist()

zz3 = []

print

del x\_i3[-1] #remove last element

x\_i3\_plus.pop(0) #remove first element

density3 = []

for j in range(0, len(pairr)):

density3.append([pair\_m[j],0])

for k in range(0, len(x\_i3)):

if pairr[j][0] == x\_i3[k] and pairr[j][1] == x\_i3\_plus[k]:

density3[-1][-1] += 1

zz3.append(density3[-1][-1])

print(zz3)

print(density3)

