**本份作業為多媒體系統與應用的作業三，使用python以jupyter notebook撰寫**

**（建議使用anaconda執行）**

**作業包含內容：**

**- Project3-Speaker Diarization.docx:程式碼與說明word檔**

**- 輸入:**

**註冊語者01.wav**

**註冊語者02.wav**

**註冊語者03.wav**

**註冊語者04.wav**

**對話內容.wav**

**- 輸出圖檔:**

**output.png:尚未smoothing的輸出**

**output2.png: smoothing後的輸出**

**- hw3.ipynb:jupyter notebook**

**- hw3.py:jupyter notebook輸出的可執行腳本檔**

**- hw3.html:jupyter notebook輸出的執行紀錄網頁檔**

**Author**

**姓名：丁國騰**

**學號：E94036209**

**程式碼與解說(以水平線分隔出不同部分的區塊以作說明，水平線和說明文字都不是程式碼的一部分)：**

**from** **scipy.cluster.vq** **import** kmeans, vq, kmeans2, whiten

**from** **numpy** **import** array, reshape, zeros

**from** **python\_speech\_features** **import** mfcc, delta, logfbank

**from** **scipy.spatial** **import** distance

**from** **scipy.spatial.distance** **import** pdist

**import** **scipy.io.wavfile** **as** **wav**

**import** **wave**

**import** **matplotlib.pyplot** **as** **plt**

**import** **numpy** **as** **np**

**import** **math**

**import** **IPython.display** **as** **ipd**

SIZE = 512

先確認是否能夠讀入語音檔

ipd.Audio('對話內容.wav')

ipd.Audio('註冊語者01.wav')

使用python\_speech\_features來計算mfcc和delta、double-delta

(rate,sig) = wav.read("註冊語者01.wav")

mfcc\_feat = mfcc(sig,rate)

d\_mfcc\_feat = delta(mfcc\_feat, 2)

d2\_mfcc\_feat = delta(d\_mfcc\_feat, 2)

測試看看結果，證實正確，最後將這些數值合併成39個特徵值的矩陣

print(d2\_mfcc\_feat.shape)

print(d\_mfcc\_feat.shape)

*#combine\_feat = np.append(np.append(mfcc\_feat, d\_mfcc\_feat, axis=1), d2\_mfcc\_feat, axis=1)*

combine\_feat = np.concatenate((mfcc\_feat,d\_mfcc\_feat,d2\_mfcc\_feat), axis=1)

code\_books = []

print(combine\_feat.shape)

印出:

(999, 13)

(999, 13)

(999, 39)

對此矩陣進行vector quantization，在此假設code book size為512 其他speaker的語音檔也如法炮製，並將結果存於code\_books中，應該是一個4個元素的list

codebook, labeled\_obs = kmeans2(data=combine\_feat, k=SIZE)

print(codebook.shape)

code\_books.append(codebook)

**for** speaker **in** '234':

(rate,sig) = wav.read("註冊語者0" + speaker + ".wav")

mfcc\_feat = mfcc(sig,rate)

d\_mfcc\_feat = delta(mfcc\_feat, 2)

d2\_mfcc\_feat = delta(d\_mfcc\_feat, 2)

combine\_feat = np.concatenate((mfcc\_feat,d\_mfcc\_feat,d2\_mfcc\_feat), axis=1)

*#combine\_feat = np.append(np.append(mfcc\_feat, d\_mfcc\_feat, axis=1), d2\_mfcc\_feat, axis=1)*

codebook, labeled\_obs = kmeans2(data=combine\_feat, k=SIZE)

code\_books.append(codebook)

印出:

(512, 39)

/Users/kuoteng/.pyenv/versions/3.6.4/envs/NCKU-MMA/lib/python3.6/site-packages/scipy/cluster/vq.py:525: UserWarning: One of the clusters is empty. Re-run kmeans with a different initialization.

warnings.warn("One of the clusters is empty. "

讀入對話內容語音檔，並且使用歐幾里德距離來計算每個訊號特徵和參考特徵之間的差異，將結果存入result

(rate,sig) = wav.read("對話內容.wav")

mfcc\_feat = mfcc(sig,rate)

d\_mfcc\_feat = delta(mfcc\_feat, 2)

d2\_mfcc\_feat = delta(d\_mfcc\_feat, 2)

*#test\_feature\_vector = np.append(np.append(mfcc\_feat, d\_mfcc\_feat, axis=1), d2\_mfcc\_feat, axis=1)*

test\_feature\_vector = np.concatenate((mfcc\_feat,d\_mfcc\_feat,d2\_mfcc\_feat), axis=1)

test\_frame\_num = test\_feature\_vector.shape[0]

print(test\_feature\_vector)

print(test\_frame\_num)

result = np.zeros(test\_frame\_num)

temp\_min = np.zeros(4)

**for** i **in** range(test\_frame\_num):

**for** j **in** range(4):

**for** k **in** range(SIZE):

x = (test\_feature\_vector[i,:], code\_books[j][k, :])

dist = sum(pdist(x, metric='euclidean'))

**if** k == 0:

temp\_min[j] = dist

**else**:

**if** dist < temp\_min[j]:

temp\_min[j] = dist

index = np.argmin(temp\_min)

result[i] = index

印出:

[[ 1.19105222e+01 -2.52417727e+01 -5.43302469e+00 ... -6.13639459e-01

-1.75801261e-01 -1.49060403e-01]

[ 1.19244191e+01 -2.59663304e+01 -5.15507087e+00 ... -6.80254987e-01

-6.74012803e-02 -1.13553311e-01]

[ 1.21900290e+01 -2.58000468e+01 -3.20824779e+00 ... -4.69117754e-01

1.10505278e-02 6.35995757e-02]

...

[ 1.32191087e+01 -2.10063391e+01 -6.40080881e+00 ... -1.04868138e+00

-6.49677731e-01 -9.55397626e-01]

[ 1.32610163e+01 -2.20521248e+01 -6.14442969e+00 ... -1.28289640e+00

-5.54965821e-01 -7.77192881e-01]

[ 1.30048586e+01 -2.13497785e+01 -3.31983820e+00 ... -6.81378744e-01

-2.33280776e-01 -2.95840185e-01]]

2094

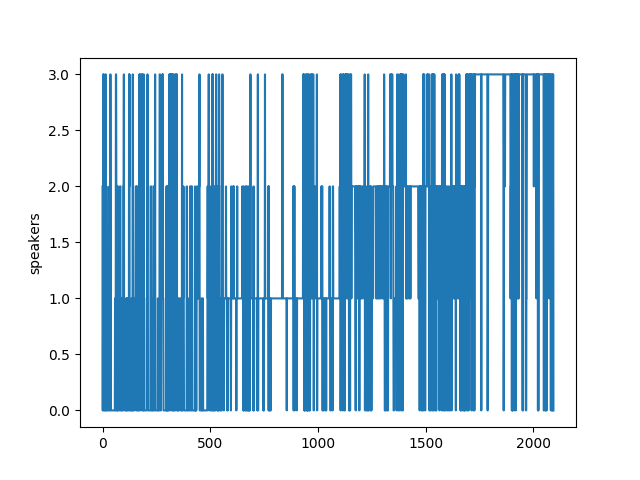
印出result，雖然可以看出大致趨勢（從區間密集程度），但無法明確辨識，故必須做smoothing

plt.plot(result)

plt.ylabel("speakers")

plt.savefig('output1.png')

印出:



實作smoothing

smooth\_result = np.zeros(test\_frame\_num)

smooth\_size = 100

overlap = smooth\_size//4

**for** i **in** range(test\_frame\_num//smooth\_size):

count = np.zeros(4)

**for** j **in** range(smooth\_size):

**if** i == 0:

count[int(result[int(i\*smooth\_size + j)])] = count[int(result[int(i\*smooth\_size + j)])] + 1

**else**:

count[int(result[int(i\*smooth\_size + j - overlap)])] = count[int(result[int(i\*smooth\_size + j - overlap)])] + 1

index = np.argmax(count)

**for** j **in** range(smooth\_size):

**if** i == 0:

smooth\_result[int(i\*smooth\_size + j)] = index

**else**:

smooth\_result[int(i\*smooth\_size + j - overlap)] = index

再一次印出result，發現這次可以看出四位speaker使用者（分別對應到index0,index1,index2,index3）

plt.plot(smooth\_result)

plt.ylabel("speaker")

plt.savefig('output2.png')

印出:

