105061254 林士平 數位訊號處理實驗報告 Lab5

1. Abstract/Introduction

本次的 Lab 介紹了在語音辨識（Speech Recognition）和語者辨識（Speaker Recognition）方面，最常用到的**語音特徵**就是「梅爾倒頻譜係數」（Mel-scale Frequency Cepstral Coefficients，簡稱 **MFCC**）。此參數考慮到人耳對不同頻率的感受程度(使用 Mel-to-Hz, Hz-to-Mel nonlinear mapping)，因此特別適合用在語音辨識。

取得 MFCC 的過程就是在**去蕪存菁**：從 Pre-emphasis、STFT、Mel-scale Filtering、Log Energy 到 Discrete Cosine Tranform，經過這些程序可以將聲音的精華留下來，而這些精華是重要的**語音特徵**。本次的實驗便是要利用 python 來走一次這個流程，取得音檔的 MFCC。

2. Goals of this lab

(1) 畫出 pre-emphasis 前後的訊號並比較。
(2) 設計並畫出 Mel-scale filter bank。
(3) 錄音([a], [i], [u], [ɛ], [ɔ])然後找出音檔的 MFCC。

3. Method (取得 MFCC 的流程圖見 5. Flow chart。)

(1) pre-emphasis

pre-emphasis 將語音訊號 s(n) 通過一個 High pass filter：

$$H(z) = 1 - a * \mathrm{z}^{-1}$$

其中 a 介於 0.9 和 1.0 之間。若以 time domain 的運算式來表示，pre-emphasis 後的訊號 $s_2(n)$ 為

$$s2(z) = s(n) - a * s(\mathrm{n}-1)$$

其實就是一個 difference filter，目的之一是為了"消除發聲過程中聲帶和嘴唇的效應，來**補償**語音信號受到發音系統所壓抑的**高頻部分**"，簡單說就是一般聲音的 frequency response 中，高頻成分的強度會比低頻成分的強度小，所以藉由 high pass filter 來平衡低頻成分和高頻成分的強度，另一個目的是它可以增進訊雜比(Signal-to-noise ratio)。**附檔(SteveJobs_After_Pre-emphasis.wav)為做完 pre-emphasis 的 SteveJobs 音檔**，和原音檔比較會發現**聲音變得較尖銳且清脆**。

(2) Frame Blocking

將 N 個取樣點集合成一個觀測單位，稱為 Frame，通常 N 的值是 256 或 512，涵蓋的時間約為 20~40 ms 左右。為了避免相鄰兩音框的變化過大，

所以我們會讓兩相鄰音框之間**有一段重疊區域**，此重疊區域包含了 M 個取樣點，通常 M 的值約是 N 的一半。通常語音辨識所用的音訊的取樣頻率為 8 KHz 或 16 KHz，以 8 KHz 來說，若音框長度為 256 個取樣點，則對應的時間長度是 256/8000*1000 = 32 ms。

Frame blocking 的原因是因為**音訊的特徵(通常指頻率)會隨時改變**，這些特徵只有在短的時間區間內才能保持不變，所以做 Frame blocking 有助於我們針對這些短時間區間做傅立葉轉換，把這些特徵抓出來，最後再把這些 block 串接起來，得到特徵完整的音訊。

(3) **Windowing**

將每一個音框乘上 Hamming Window，乘上 Hamming Window 的主要目的，是要加強音框**左端和右端的連續性**。這是因為在進行 FFT 時，都是假設一個音框內的訊號代表一個週期性訊號，如果這個週期性不存在，FFT 會為了要符合左右端不連續的變化，而產生一些不存在原訊號的能量分佈，造成分析上的誤差(**spectral leakage**)。

(4) **Fast Fouier Transform**

由於訊號在 Time domaim 上的變化通常很難看出訊號的特性，所以通常將它轉換成 Frequency domain 來觀察，不同的能量分佈，就能代表不同語音的特性。所以在乘上 Hamming Window 後，每個音框還必需再經過 FFT 以得到在 Frequency domain 上的能量分佈。

其中，使用 N-point FFT(N 通常取 256 或 512)在每一個 frame 上計算頻譜，這個過程稱為 Short-Time Fourier-Transform (STFT)。步驟(2) ~ (4)即為 STFT 的過程。

(5) **Mel-Scale Filtering**

再來使用 triangular filter 來提取頻帶，其中要特別注意的是此 triangular filter 使用的是 Mel-Scale，也就是說這 filters_num 個 triangular filter 在「梅爾頻率」（Mel Frequency）上是平均分佈的，而梅爾頻率和一般頻率 f 的關係式如下：

$$m = 2595 * log_{10}(1 + \frac{f}{700})$$

這個 mapping 是為了模擬人耳對聲音的感受度：在較低頻率，人耳感受較為敏銳，然而越高頻人耳感受就越不敏銳。

(6) **Log Energy**

一個 frame 的音量（即能量），也是語音的重要特徵，而且非常容易計算。計算方式如下頁所示：

$$能量(dB) = 10 * log_{10}(一個音框內訊號的平方和)$$

(7) **Discrete Cosine Transform**

將上述的 Log Energy 帶入離散餘弦轉換，求出 L 階的 **Mel- scale Cepstrum** 參數，這裡 **L 通常取 12**。

4. Pseudo code

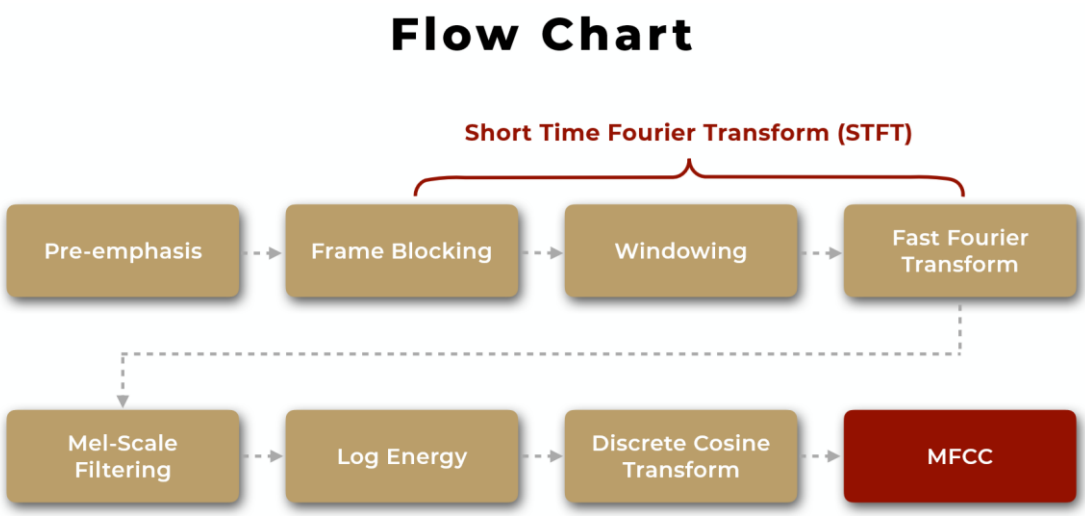
以下是本次 Lab 程式碼重點：

(0) 定義 mel 和 Hz 之間的 mapping	*def mel2hz(mel):* *'''* *mel scale to Hz scale* *'''* *hz = 700 * (10**(mel/2595) - 1)* *return hz* *def hz2mel(hz):* *'''* *hz scale to mel scale* *'''* *mel = 2595 * np.log10(1 + hz / 700)* *return mel* 說明：做 **mel** 和 **Hz** 之間的轉換，公式見上一頁。
(1) Pre-emphasis	*def pre_emphasis(signal,coefficient=0.95):* *return np.append(signal[0],signal[1:]-coefficient*signal[:-1])* *signal=pre_emphasis(signal)* 說明：做 **high pass filter(difference filter)**。
(2) Frame Blocking	*frames_num=1+int(math.ceil((1.0*signal_length-frame_length)/frame_step))* *#padding* *pad_length=int((frames_num-1)*frame_step+frame_length)* *zeros=np.zeros((pad_length-signal_length,))* *pad_signal=np.concatenate((signal,zeros))* *#split into frames* *indices=np.tile(np.arange(0,frame_length),(frames_num,1))* *+np.tile(np.arange(0,frames_num*frame_step,frame_step),(frame_length,1)).T* *indices=np.array(indices,dtype=np.int32)*

	frames=pad_signal[indices] 說明：先做 **zero padding**，**zero padding** 是為了讓每個 **frame** 有相同數量個 **sample**，避免把原始訊號的 **sample** 丟失。然後將得到的訊號分割成 **frames**，特別注意 **frame** 和 **frame** 之間訊號必須要有重疊。
(3)Window	*frames *= np.hamming(frame_length)*
(4) FFT	*complex_spectrum=np.fft.rfft(frames,NFFT).T* *absolute_complex_spectrum=np.abs(complex_spectrum)*
(5) Mel-Scale Filtering(part 1)	*def get_filter_banks(filters_num,NFFT,samplerate,low_freq=0,high_freq=None):* *''' Mel Bank* *filers_num: filter numbers* *NFFT:points of your FFT* *samplerate:sample rate* *low_freq: the lowest frequency that mel frequency include* *high_freq:the Highest frequency that mel frequency include* *'''* *#turn the hz scale into mel scale* *low_mel=hz2mel(low_freq)* *high_mel=hz2mel(high_freq)* *#in the mel scale, you should put the position of your filter number* *mel_points=np.linspace(low_mel,high_mel,filters_num+2)* *#get back the hzscale of your filter position* *hz_points=mel2hz(mel_points)* *#Mel triangle bank design* *bin=np.floor((NFFT+1)*hz_points/samplerate)* *fbank=np.zeros([filters_num,int(NFFT/2+1)])* *for j in range(0, filters_num):* *for i in range(int(bin[j]), int(bin[j+1])):* *fbank[j, i] = (i - bin[j])/(bin[j+1] - bin[j])* *for i in range(int(bin[j+1]), int(bin[j+2])):* *fbank[j, i] = (bin[j+2] - i)/(bin[j+2] - bin[j+1])* *return fbank* 說明：首先要得到 **mel-scale filter bank**，我們先在 **mel domain** 上做等分點，然後轉換回 **Hz domain**，所以就得到 **filter number** 個 **triangular filter** 的初始與結束頻率，然後做 **filter_num** 個 **trianguler filter(triangular function)**，公式如下：

	$$\mathrm{tri}_j(x) = \begin{cases} (x - x_{j-1})/(x_j - x_{j-1}) & x_{j-1} \le x < x_j \\ (x_{j+1} - x)/(x_{j+1} - x_j) & x_j \le x < x_{j+1} \\ 0 & \text{otherwise} \end{cases}$$
(5) Mel-Scale Filtering(part 2)	*fb=get_filter_banks(filters_num,NFFT,fs,low_freq,high_freq)* *filter_banks = np.dot(absolute_complex_spectrum.T, fb.T)* *filter_banks = np.where(filter_banks == 0, np.finfo(float).eps, filter_banks)* 說明：然後將 **filter bank** 和步驟 **4.**得到的頻譜做內積**(dot product)**，特別注意 **np.where(filter_banks == 0, np.finfo(float).eps, filter_banks)**是為了避免 **filter_banks = 0** 造成下一步不能取 **log**，所以如果 **= 0** 的話要轉換成和 **0** 很接近的數 **(np.finfo(float).eps)**。
(6)Log Energy	*feat = 10 * np.log10(filter_banks)*
(7) DCT	*feat=dct(feat, norm='ortho')[:,:filters_num]*
(8) 印出結果	*#Print triangular band-pass filter* *xaxis = np.arange(0, len(fb[0]))*(1.0*fs/len(fb))* *plt.figure(3)* *for i in range(len(fb)):* *plt.plot(xaxis,fb[i])* *plt.show* *#Print MFCC* *plt.figure(4)* *for i in range(1, len(feat)):* *plt.plot(feat[i])* *plt.show*

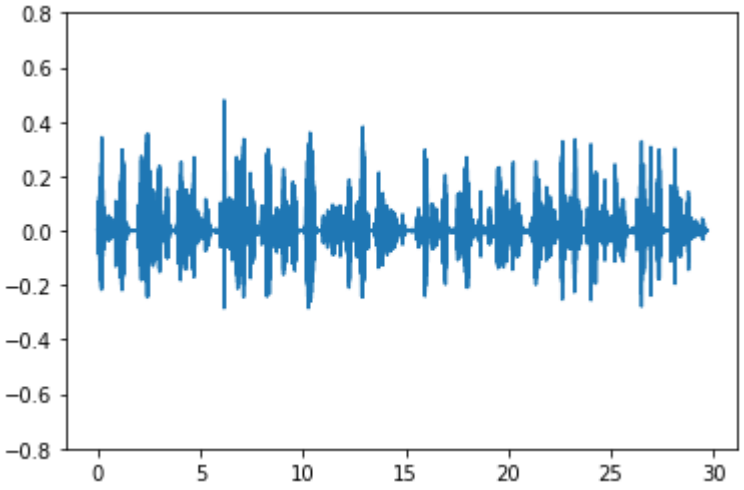
5. Flow chart



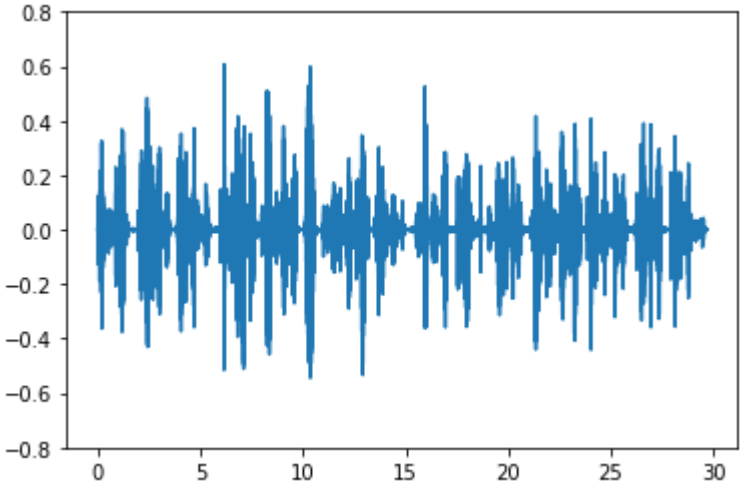
6. Results

(1) SteveJobs

a. Signal and Pre-emphasis signal

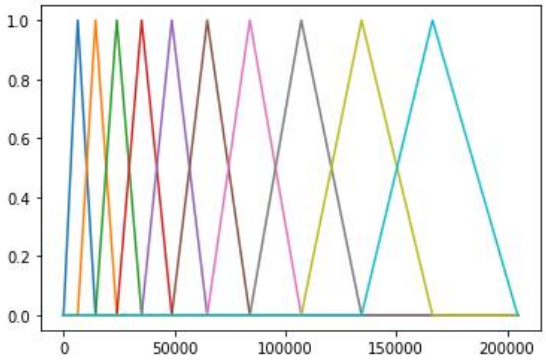


↑原始音檔訊

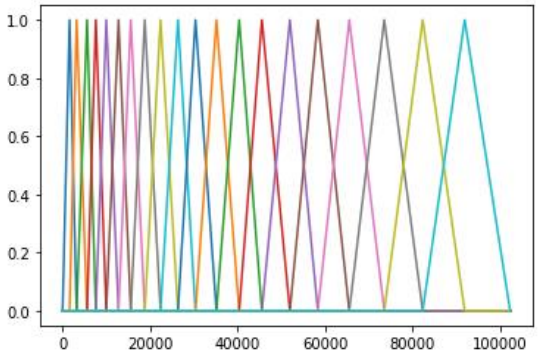


↑pre-emphasis 後的音檔訊號，可以大略看出相較原始音檔，尖端變得更明顯，由附檔(SteveJobs_After_Pre-emphasis.wav)更可以明顯聽出與原始音檔之間的差異。

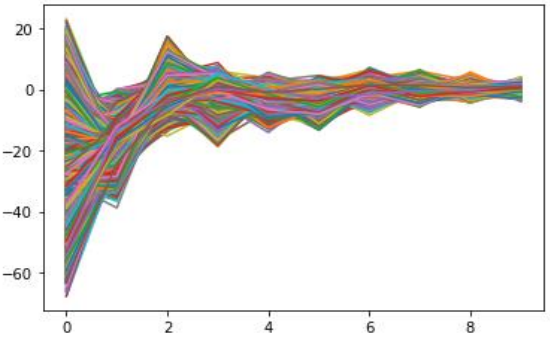
b. Mel-scale filter bank and MFCC



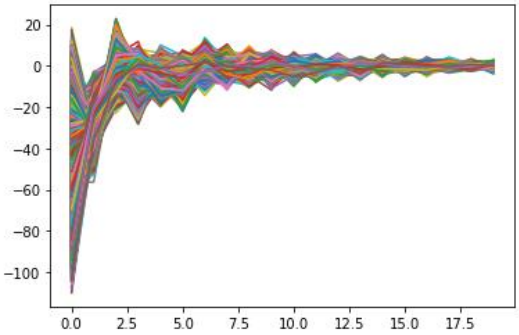
↑ Mel-scale filter bank(filters_num=10)



↑Mel-scale filter bank(filters_num=20)



↑MFCC for SteveJobs.wav
(filters_num=10)

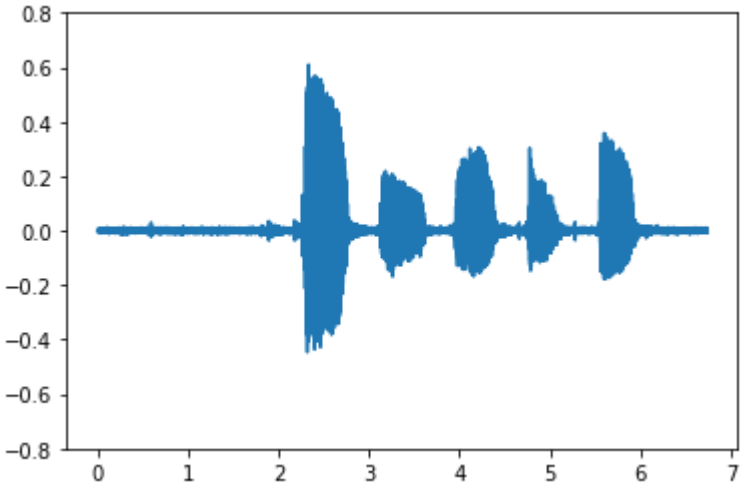


↑MFCC for SteveJobs.wav
(filters_num=20)

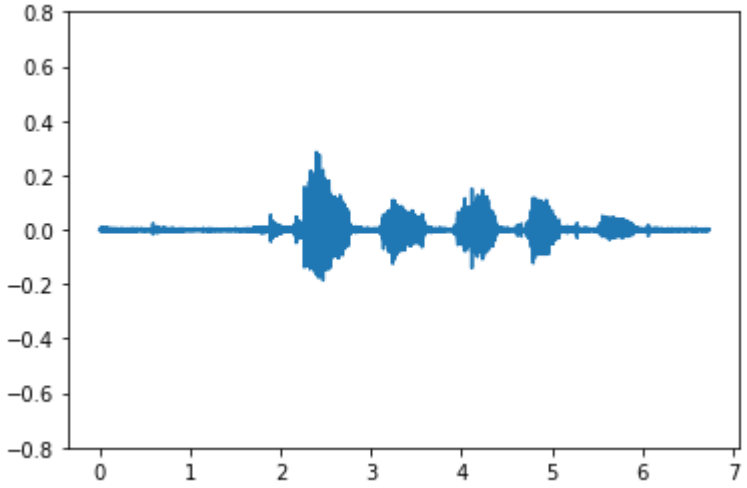
由此結果可見：filter numbers 越多，其 quefrency domain 上點的連線也會較平滑，這是因為用越多的 filter numbers，代表使用越多數量的 triangular filter 做 convolution。

(2) 我的[a], [i], [u], [ɛ], [ɔ]

a. Signal and Pre-emphasis signal

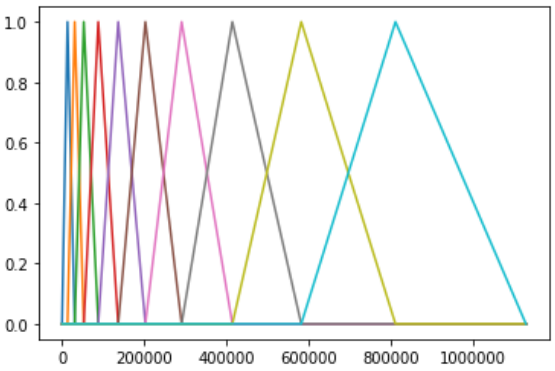


↑原始音檔訊

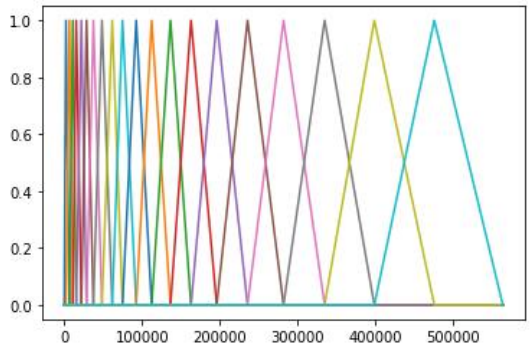


↑ pre-emphasis 後的音檔訊號，可以大略看出相較原始音檔，尖端變得更明顯，而且 y 值變小了，由附檔(my_a_3_After_Pre-emphasis.wav)更可以明顯聽出與原始音檔之間的差異。

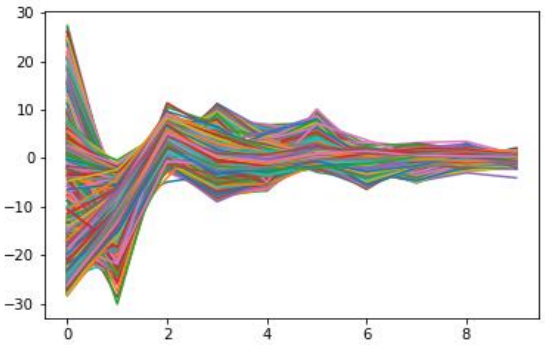
b. Mel-scale filter bank and MFCC



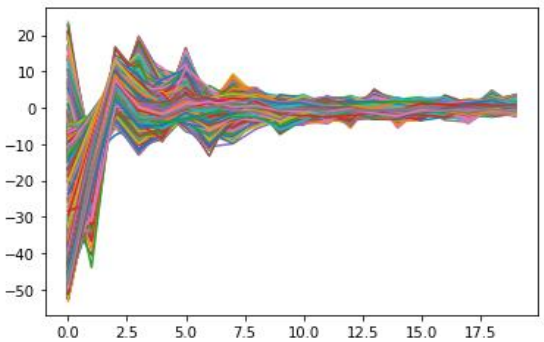
↑ Mel-scale filter bank(filters_num=10)



↑Mel-scale filter bank(filters_num=20)



↑MFCC for my_a_3.wav
(filters_num=10)



↑MFCC for my_a_3.wav
(filters_num=20)

7. Discussion

(1) Why not using rectangular filters for “energy” calculation? Why should mel-filters be overlapping? Compare the results of different filter banks?

由北京清華發表的文章：COMPARISON OF DIFFERENT IMPLEMENTATIONS OF MFCC(參考資料(5))，其中他們利用不同的 filter shape，包含 rectangular curve、triangular curve 和 Schroeder curve，並且考慮有無 overlapped 的 rectangular curve 和 triangular curve，得到的結果如下表：

TABLE 2. DIFFERENT FILTER SHAPES AND FREQUENCY WARPING

(In this table, XTRI stands for crossed/overlapped triangular filters while TRI for non-overlapped, and XRECT for overlapped rectangular filters while RECT non-overlapped.)

Features (&aRa)		Top 1	Top 5	Top 10
Warping	*Filter Shape*			
MEL	XTRI	68.01	91.79	95.77
MEL	TRI	66.35	91.21	95.42
MEL	**XRECT**	**68.38**	**92.14**	**95.91**
MEL	RECT	66.36	91.18	95.37
BARK	XTRI	67.61	91.56	95.57
BARK	TRI	66.99	91.38	95.43
BARK	XRECT	67.59	91.53	95.57
BARK	RECT	67.00	91.35	95.53
BARK	SCHROEDER	67.25	91.53	95.51

結論是：**不同的 filter shape 對 performance 的影響不大，然而有 overlapped 的 filter 可以有較好的 performance**。

(2) What is the best filter numbers?

同樣由參考資料(5)，該研究分析了不同 filter number 的 **Overlapped Triangular Filter**(即本次 Lab 所使用的)的 performance，結果如下：

TABLE 1. DIFFERENT NUMBERS OF OVERLAPPED TRIANGULAR FILTERS

# of filters (MFCC0^&aRa)	Top 1	Top 5	Top 10
25	67.39	91.56	95.57
30	67.73	91.72	95.66
35	**68.01**	**91.79**	**95.77**
40	67.84	91.92	95.82
45	67.86	91.81	95.74

由表中可見 filter number = **35**，有最好的 performance。

(3) Why are high-quefrency MFCCs usually abandoned when doing speech recognition?

由參考資料(6)，低 quefrency 的 MFCC 幾乎已經包含了所有聲音的特徵，其中 zero order coefficient 代表音檔的平均功率，first-order coefficient 代表高頻和低頻之間的頻譜能量分布。因此固然增加更多的 coefficient 可以更詳細的描述聲音的細節，但是考量到如此一來會增加複雜度，且低 quefrency 的 MFCC 幾乎已經包含了重要的聲音特徵，所以通常會取 **12-20** 個 MFCC，high-quefrency 便予以捨棄。

(4) Do you think MFCC is good for speaker identification purposes? If two sounds have similar MFCCs, does that imply they sound similar to our ears?

我認為 MFCC 是好的語音辨識參數，原因是我們將 MFCC 還原為音檔後，仍然可以聽得出來原音檔的一些特徵，還是可以知道這個是從原音檔得到的。不過還原後的音檔和原音檔有一個很大的不同，就是**音高**不見了，所以如果兩個聲音有相同的 MFCC，並不能表示他們對人耳來說聽起來是一樣的，最起碼音高可能就不一樣。

8. Reference

(1) PySoundFile
https://pysoundfile.readthedocs.io/en/0.9.0/
(2)Audio Signal Processing and Recognition(音訊與辨識)：C12-2 MFCC ,張智星教授
http://mirlab.org/jang/books/audioSignalProcessing/
(3) 數位語音處理概論：7.0 Speech Signal and Front-end Processing ,李琳山教授
http://ocw.aca.ntu.edu.tw/ocw_files/104S204/104S204_AA07L01.pptx
(4) Speech Processing for Machin Learning: Filter banks, Mel-Frequency Cepstral Coefficients(MFCCs) and What's In-Between
https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html
(5) COMPARISON OF DIFFERENT IMPLEMENTATIONS OF MFCC
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.6868&rep=rep1&type=pdf
(6) Why we take only 12-13 MFCC coefficients in feature extraction?
https://www.researchgate.net/post/Why_we_take_only_12-13_MFCC_coefficients_in_feature_extraction