Extraction de MFCCs

La séance de TD a pour but de vous permettre de réaliser l'implémentation des MFCCs par vous-même. Cette séance est notée, c'est-à-dire que j'attends de vous **un code détaillé et commenté** dans lequel vous ajouterez les éléments de réponses aux différentes questions ainsi que les figures obtenues.

L'ensemble des documents est à rendre pour **vendredi 18 septembre 2020 à 00h**.

On utilisera les paquets Python `numpy, scipy` et `librosa` pour la comparaison.

Exercice I - Segmentation d'un signal audio en trames

Dans un premier temps nous allons segmenter le signal audio `exemple.wav` en trames, c'est-à-dire que nous allons récupérer N trames d'une durée de 30 ms chacune. Deux trames consécutives sont espacées d'une durée de 10 ms. Soit `win = 0.03` et `step = 0.01`.

1. Récupérer l'amplitude et la fréquence d'échantillonnage du signal audio `exemple.wav` à l'aide de la fonction `scipy.io.wavfile.read()`. La tracer en fonction du temps.
2. Découper ce signal en trames mono canal qui pourront être stockées dans un tableau de type `np.array` (au moins dans un premier temps). Le fenêtrage se fera avec une fenêtre de Hamming (`np.hamming(n)`).

Exercice II - Calcul des filtres de Mel

Dans un second temps, nous allons calculer les filtres de Mel adaptés à notre contexte.

Pour cela on utilisera la fonction de conversion des fréquences f en Hertz vers les fréquences m en Mel, suivante:

$$m = \begin{cases} f & \text{if} \quad f < 1000 \\ 1000 \log_2 \left(1 + \frac{f}{1000}\right) & \text{else} \end{cases} \tag{1}$$

Sur la plage de fréquence $[f_{min}; f_{max}]$ on définira R filtres Mel_r tels que $r \in [1; R]$. Ces filtres sont triangulaires et centrés en f_r: ils sont nuls presque partout sauf sur l'intervalle $[f_{r-1}; f_{r+1}]$. Les fréquences f_r sont linéairement réparties dans l'espace des Mel, i.e. $m_{r+1} - m_r = m_r - m_{r-1}$. On normalisera chaque filtre afin que $\sum_{k=1}^{N} Mel_r(k)^2 = 1$.

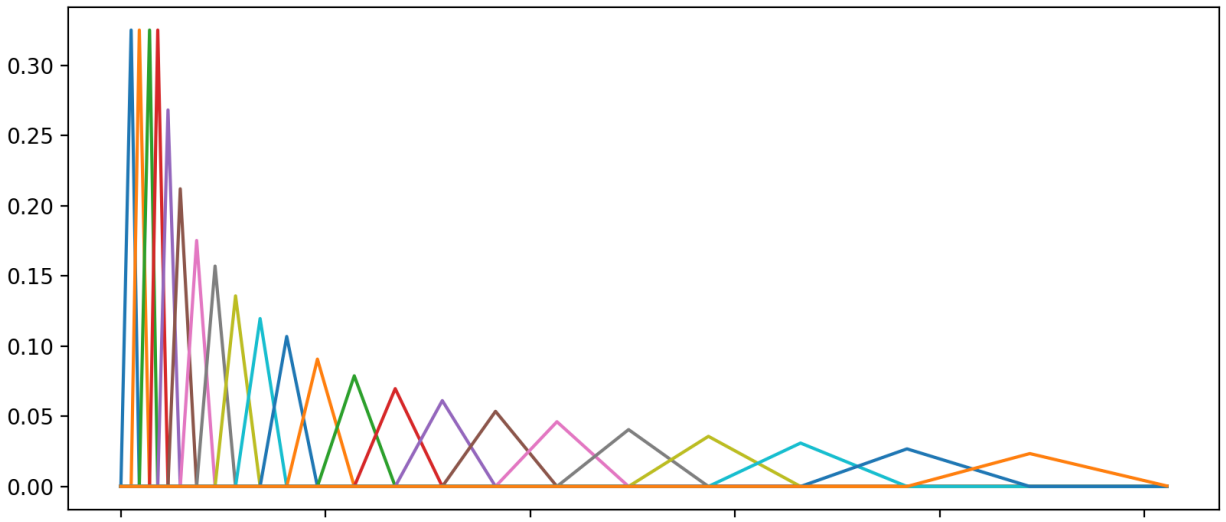
Les paramètres que nous utiliserons sont `fmin = 20`, `fmax = fs/2` et `R = 22`.

1. Calculer les fréquences f_r
2. Construire les R filtres de Mel et les tracer.

Exercice III - Calcul des MFCCs

On rappelle ici la chaîne de traitement pour le calcul des coefficients cepstraux (MFCCs).

a) Calculer le module du spectre $S[k]$ pour chaque trame temporelle $s[n]$ (`scipy.fft.fft`).



b) Prendre l'énergie du module $P[k]$ sur les fréquences réelles: $P[k] = S[k]^2$.

c) Calculer le cepstre MF (mel-frequency cepstrum) à l'aide des filtres de Mel: $MF[r] = \sum_k Mel_r[k] \cdot P[k]$.

d) Prendre le log: $logMF[r] = 10 \log_{10}(MF[r])$.

e) Calculer enfin la transformée cosinus discrète (DCT) pour un coefficient $m \in [0, N_{mfcc}]$ donné:

$$MFCC[m] = A_m \sum_r logMF[r] \cdot \cos\left(\frac{2\pi}{R}\left(r + \frac{1}{2}\right) m\right)$$

Le coefficient A_m est un coefficient de normalisation qui vaut $A_0 = 1/\sqrt{4R}$ ou $A_m = 1/\sqrt{2R}$ pour $m > 0$

1. Pour chaque trame, calculer $N_{mfcc} = 12$ coefficients.
2. Afin de vérifier l'évolution temporelle de ces coefficients, on pourra comparer avec ceux extraits par la librairie `librosa`.

```
#use a precomputed spectrogram, y is the full waveform before segmentation
S = librosa.feature.melspectrogram(y, fs, n_mels=R, fmin=20, fmax=fs/2, hop_length=p)
#extract mfcc
X_librosa = librosa.feature.mfcc(S=librosa.power_to_db(S), n_mfcc=N_mfcc, dct_type=2,
    norm='ortho')

#or else, without precomputed spectrogramm:
X_librosa0 = librosa.feature.mfcc(y=y, sr=fs, hop_length=p, n_mfcc=N_mfcc, dct_type=2,
    norm='ortho')

#plot MFCC[1] according to time samples.
plt.stem(X[:,1]) #N*N_mfcc matrix you have extracted
plt.stem(X_librosa[1,:], 'g', markerfmt='go') ##N_mfcc*N matrix extracted by librosa.
plt.show()
```

3. Si vous ne trouvez pas exactement les mêmes valeurs, c'est normal, les méthodes de calcul ne sont pas strictement identiques. Donner quelques éléments qui vous semblent différents entre les 2 méthodes.