ING 3 promo 2020 TD9

Projet de traitement du signal 2

Système de reconnaissance vocale automatique

DUCRUET Amandine, PAGES Hermance, TAO Tuong Vi
28/05/2018

Table des matières

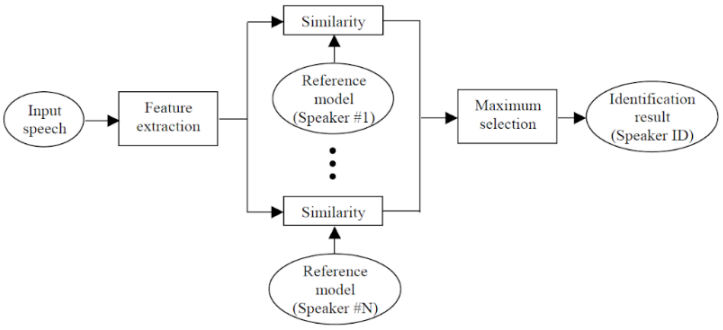
A. Le principe d'un système de reconnaissance vocale automatique

Qu'est-ce que la reconnaissance vocale automatique ?

La reconnaissance automatique vocale est une technique qui analyse la voix d'un locuteur. En décomposant et en traitant les informations incluses dans la voix, on peut vérifier l'identité du locuteur et même retranscrire ce qu'il dit. Ainsi, on peut avoir accès à des services, des données, des lieux, avec un certain degré de sécurité.

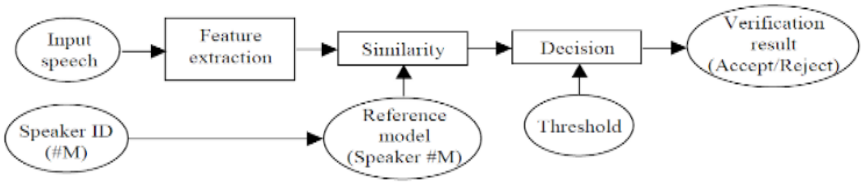
On peut diviser la reconnaissance vocale en deux types :

- Identification : qui, parmi les personnes enregistrées dans la base de données, parle ?



(a) Speaker identification

- Vérification : le locuteur est-il la personne qu'il prétend être ?



(b) Speaker verification

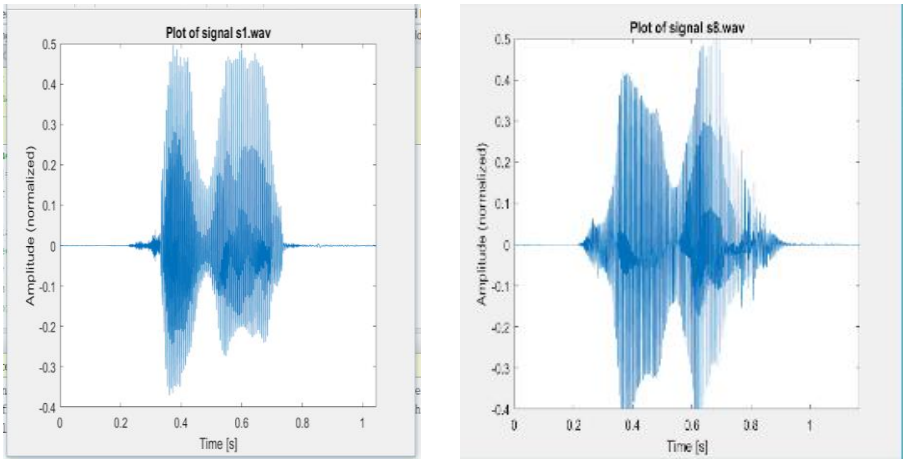
Figure 1. Basic structures of speaker recognition systems

Chaque système de reconnaissance vocale doit avoir 2 phases :

- Une « training phase », phase de catalogage, où chaque personne doit fournir des extraits de voix pour que le système puisse créer des références de voix pour chacun.
- Une « testing phase », phase de test, où on compare une voix (ou plusieurs) avec celles de la base de données pour déterminer si elle y est présente.

B. Petite explication des MFCC : extraction des caractéristiques de la voix

Voici à quoi ressemble un signal d'une voix :



Signaux générés à partir de s1 et s8

Il y a énormément d'informations. Il faut simplifier ce signal et extraire les données caractéristiques à la personne à qui appartient cette voix. Il existe de nombreuses manières de représenter le signal audio pour la tâche de reconnaissance vocale. MFCC est un des plus utilisé.

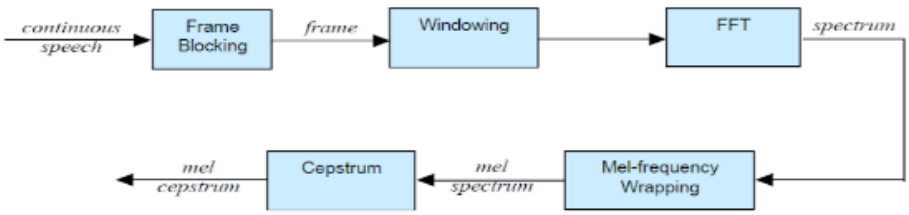


Figure 3. Block diagram of the MFCC processor

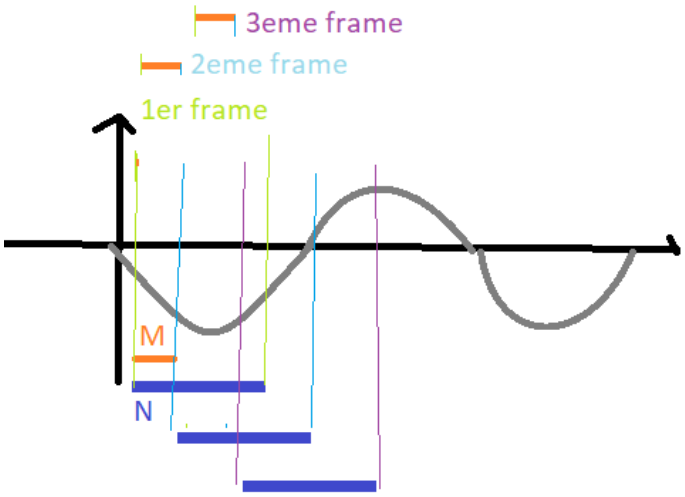
MFCC : Mel-Frequency Cepstrum Coefficients (Coefficients Cepstraux). Il s'agit d'une méthode pour représenter le signal de la voix ensuite le reconnaître. Cette méthode imite le comportement de l'oreille humaine, elle se base sur les bandes de fréquence critiques à l'oreille humaine. On exprime cela sur une échelle de mel (***mel-frequency scale***) : c'est une échelle imitant les hauteurs musicales (tons) et qui est reliée aux fréquences en Hertz, avec des filtres espacés linéairement sur les basses fréquences (en dessous de 10000 Hz) et logarithmiquement sur les hautes fréquences (au-delà de 10 000 Hz). Ainsi, elle prend en compte les caractéristiques les plus importants de la voix.

C. Procédé

La reconnaissance vocale est divisée en plusieurs étapes.

Frame blocking

Dans cette étape, on coupe le signal en plusieurs morceaux qui se superposent, avec N le nombre de points échantillonnés par morceau, et M la distance entre les 1ers points de chaque morceau.



La plupart du temps, les valeurs utilisées sont N=256 et M=100.

Windowing

Découper un signal en plusieurs morceaux crée des discontinuités au début et à la fin de chaque morceau. On souhaite alors “lisser” notre signal en appliquant, à chaque morceau, un filtre de **Hamming**, et ainsi limiter les distorsions.

FFT - Fast Fourier Transform (Transformée de Fourier Rapide)

On applique la FFT à chacun de nos morceaux pour passer du domaine temporel au domaine fréquentiel, afin de mieux traiter notre signal.
On obtient du signal x à X qui est un complexe, alors on utilise le module de ce X : cela donne spectre de fréquence d’amplitude.

Mel-frequency Wrapping

On applique l’échelle de Mel. Sur cette échelle, on peut associer à chaque tonalité réelle (en musique) avec sa fréquence associée, à une tonalité subjective mesurée sur l’échelle de Mel.

Pour ce faire, on pondère notre spectre d’amplitude en appliquant un “filter bank”, un banc de filtres (tableau) passe-bande triangulaires espacés selon l’échelle de Mel à un intervalle constant, avec K le nombre de coefficients du spectre de Mel (les coefficients sont les centres de la base du triangle).

Cepstrum

On convertit alors tous les coefficients du spectre de Mel dans le domaine temporel en utilisant un DCT - une Transformation Discrète de Cosinus pour chaque morceau de notre signal. On convertit le log du spectre de mel en temporel. Les coefficients que l'on obtient sont appelés les MFCC, Coefficients Cepstraux de la fréquence de Mel. (Il existe une formule cf. PDF sujet) On a un ensemble de coefficients pour chaque morceau, cet ensemble est appelé "vecteur acoustique".

Finalement, on obtient une séquence de vecteurs acoustiques qui va représenter la voix de l'utilisateur.

Code

```
function c = mfcc(s, fs)

M = 100; %par defaut
N = 256; %par defaut
l = length(s); %longueur de notre echantillon

%ETAPE 1 - Frame blocking
nb_morceaux = floor((l-N)/M + 1);

%ETAPE 2 - Windowing et hamming
frames=windowing(s, N, M);


%ETAPE 3 - FFT
for i = 1:nb_morceaux
fft_frames(:,i) = fft(frames(:, i));
end

%ETAPE 4 - Mel-Frequency
disp('Coefficients de mel');

p=20;
amplitudes=melfb(p, N ,fs);

N2 = 1 + floor(N/2);
 ms = amplitudes * abs(fft_frames(1:N2, :)).^2;

%ETAPE 5 - Cepstrum
c = dct(log(ms));
c(1,:) = [];
```

D. Correspondance des caractéristiques

On peut, pour la reconnaissance vocale, parler de reconnaissance de motifs. Avec le "feature matching", la concordance des caractéristiques, on cherche à reconnaître des motifs et les classifier dans des catégories/classes. Ici, on souhaite classifier nos vecteurs acoustiques dans des catégories correspondant à un utilisateur chacune.

Dans un premier temps, on utilise des motifs (appelé “training vector set”) dont on connaît les classes (c’est à dire, des vecteurs acoustiques dont on connaît le détenteur de la voix) pour créer un algorithme de classification. Puis on utilise des vecteurs restants (“test vectorset”) pour évaluer la performance de notre algorithme.

Pour effectuer cela, nous allons utiliser l’approche de Quantification des Vecteurs (VQ- Vector Quantization). L’objectif est de cartographier des vecteurs d’un grand espace vectoriel dans un nombre fini de régions de cet espace ; ce procédé est l’approche VQ. Chaque région est appelée un “cluster”, représentées chacune par leur centre (centroïd - barycentre), appelé “codeword”. L’ensemble de ces codeword est appelé “codebook”.

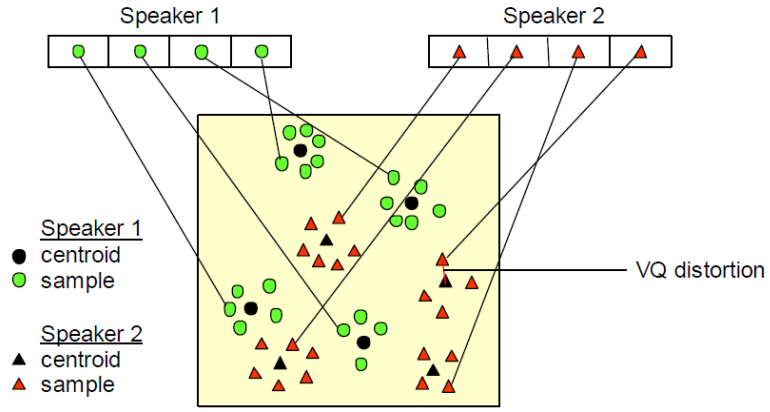


Figure 5. Conceptual diagram illustrating vector quantization codebook formation. One speaker can be discriminated from another based of the location of centroids. (Adapted from Song et al., 1987)

Sur ce schéma, chaque point/triangle représente un vecteur acoustique. Ils sont regroupés par clusters. On génère un VQ-codebook pour chaque locuteur avec les barycentres. La distance entre un vecteur et un codeword (on prend le centroïde le plus proche) est appelé VQ distorsion. On récupère le nombre total de VQ distorsions généré. La personne qui a le plus petit nombre total de VQ distorsions est alors identifié à la personne qui a généré le codebook.
Mais comment peut-on diviser notre espace vectoriel en plusieurs clusters (et ensuite utiliser un VQ codebook) ? Nous allons utiliser l’algorithme LBG (Linde, Buzo, Gray) pour séparer et regrouper un ensemble de L training vectors dans un ensemble de M codebook vectors :

1. Prenons tout d’abord le « codebook », c’est-à-dire le barycentre de l’ensemble des « training vectors».

2. Tant que la précision n’est pas optimale, on double la taille du codebook actuel en divisant chaque barycentre.
Pour faire cela nous avons besoin de la fonction « disteu », afin de calculer les distances par paires entre chaque vecteur.
z correspondant à la distance entre le training vector et le codeword.

3. Ensuite, pour le codebook actuel, nous devons associer le training vector avec le codeword le plus proche.

Nous devons trouver le codeword le plus proche, pour chaque training vector. Nous allons ainsi utiliser la fonction « min », qui compare la distance à droite et la distance à gauche afin de retenir là où elle est la plus petite.

4. Une fois cela fait, nous mettons à jour le codeword dans chaque cellule en utilisant le barycentre du training vector assigné à cette cellule

5. Tant que la distance moyenne n'est pas en dessous d'un certain seuil, on répète les étapes 3 et 4.

6. On répète les étapes 2, 3 et 4 tant que la taille M du codebook n'est pas atteinte.

Voici le diagramme d'un algorithme LGB :

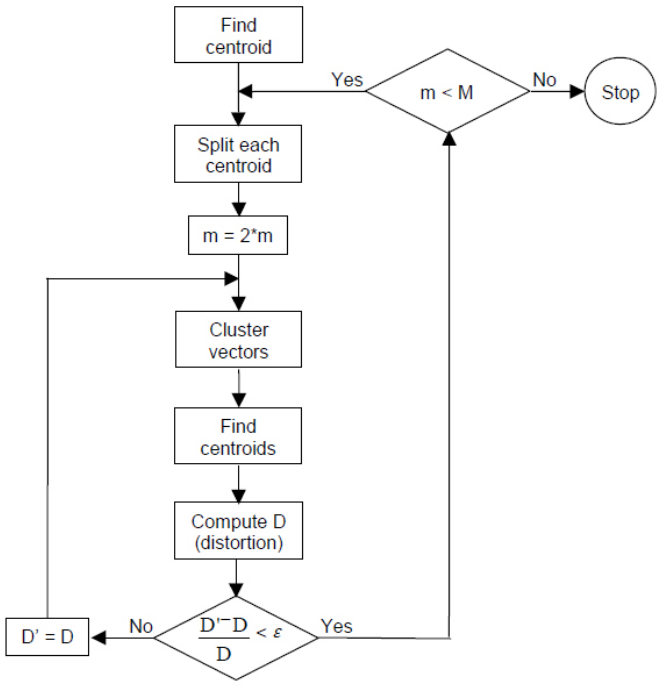


Figure 6. Flow diagram of the LBG algorithm (Adapted from Rabiner and Juang, 1993)

Code

```
function r = vqlbg(d, k)

e    = .0003; %precision des centroids/barycentres
r    = mean(d, 2); % on prend la moyenne de chaque ligne
dpr = 10000;

for i = 1:log2(k) %k nbre de centroids quon veut extraire.
   r = [r*(1+e), r*(1-e)];%pas, précis

  while (1 == 1)
```

```
        z = disteu(d, r); %distance entre vect reel d et intervalle de
recherche %distances euclidiennes entre les colonnes de deux matrices
        [m,ind] = min(z, [], 2);
         t = 0;
         for j = 1:2^i
             r(:, j) = mean(d(:, find(ind == j)), 2); %moyenne des 168x20
valeurs on prend tte la colonne dindice j, on la remplace par moyenne
               x = disteu(d(:, find(ind == j)), r(:, j)) %fait les distances
moyennes des distances au centroide/barrycentre
             for q = 1:length(x) %
                 t = t + x(q);
             end
         end
         if (((dpr - t)/t) < e) %moment ou dpr-t tend vers 1. dpr est une
valeur fixe de precision.
            break;
        else
             dpr = t;
        end
    end
end
```

E. Questions

Traitement de la parole

Question 1 :
Nous pouvons distinguer quelques voix mais pas toutes car beaucoup se ressemblent et toutes nous sont étrangères. Finalement, nous avons pu reconnaître 2 voix sur 8 grâce à l'intonation et le timbre de la voix.

Question 2 :
Pour obtenir le taux d'échantillonnage, on utilise la fonction ci-dessous sur Matlab :

```
[y, Fs] = audioread(filename)
```

Audioread retourne en y le signal échantillonné et en Fs le taux d'échantillonnage. Pour jouer le fichier sur Matlab, on écrit la ligne de commande :

```
sound (y, Fs)
```

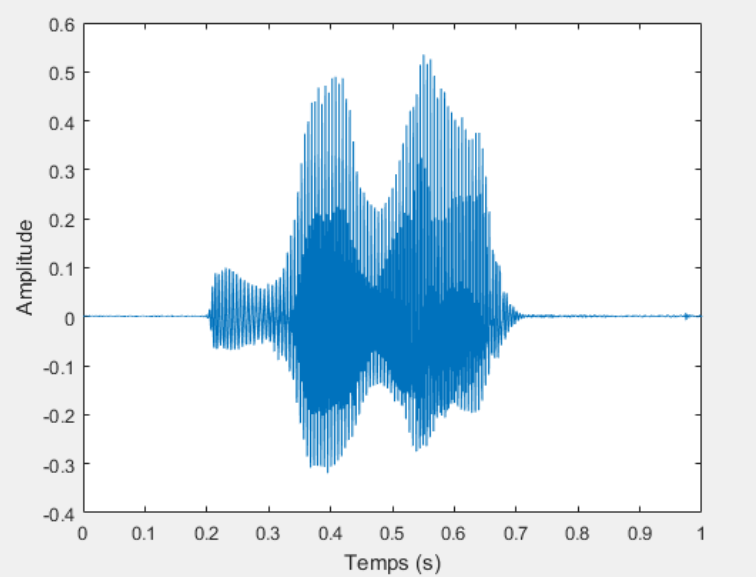
Le taux d'échantillonnage pour tous les fichiers audio vaut 12500 Hz.

Pour 256 extraits, la durée de la voix vaut 256/12500 = 20.48 msec.

Nous allons maintenant afficher le signal dans le domaine temporel. Pour cela, on utilise le code Matlab ci-dessous :

```
clear all
[y,Fs]=audioread('data\test\s1.wav');
audio2=audioread('data\test\s1.wav');
fs=length(audio2);
t=0:1/fs:1-1/fs;
plot(t,audio2)
xlabel('Temps (s)');
ylabel('Amplitude');
```

On obtient le graphe suivant :



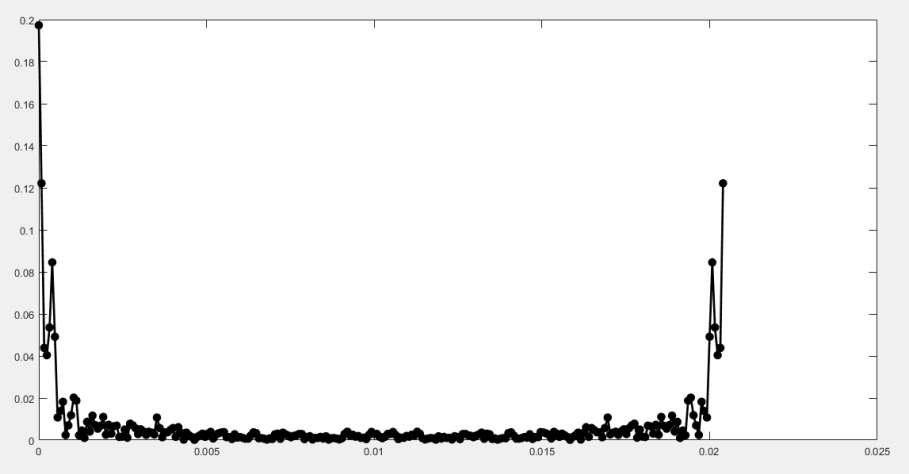
Il est très difficile d'analyser ce signal. Ce qu'il faut faire tout d'abord est transformer ce signal continu en signal discret, en utilisant la FFT (Fast Fourier Transform).
Pour garder également la bonne qualité du son, on fait un fenêtrage.

Pour cela, on découpe le signal en fenêtres qui ont chacune N=256 échantillons. Chaque fenêtre est séparée par M points, avec M=100 (M<N). La deuxième fenêtre commence M points après la première, donc elles se chevauchent de N-M points. Ce processus continue jusqu'à ce que tout le signal soit contenu dans une ou plusieurs fenêtres.

Pour obtenir une représentation discrète du signal audio en temporel, on utilise le code Matlab suivant:

```
clear all
[y,Fs]=audioread('data\test\s1.wav');
audio1=audioread('data\test\s1.wav');
fs=length(audio1);
l = length(audio1);
n = 256;
m = 100;
blockFrames = floor((l-n)/m) + 1;
for i = 1:n
for j = 1:blockFrames
M1(i,j) = audio1(((j-1) * m) + i);
end
end
M1
h=hamming(n);
M2= diag(h)*M1;
M2
for i = 1:blockFrames
 M3(:,1) = fft(M2(:,i));
end
```

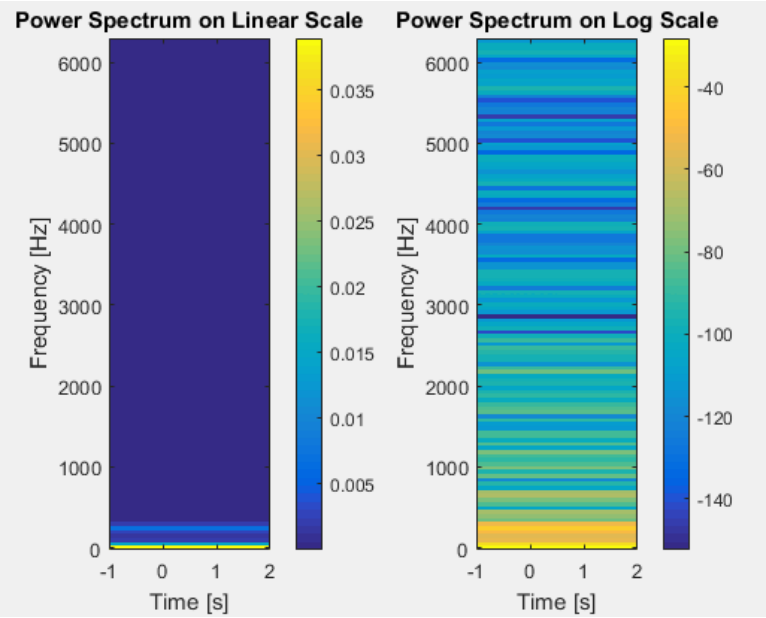
```
M3
tspan=(0:n-1)/Fs;
plot(tspan,abs(M3),'-ok','linewidth',2,'MarkerFaceColor','black');
```



Question 3 :

Pour obtenir le spectre de puissance, on utilise le code suivant :

```
Fs=12544
r = n/2;
rt = l/Fs;
subplot(121);
imagesc([0 rt],[0 Fs/2],abs(M3(1: r, :)).^2),axis xy;
title('Power Spectrum on Linear Scale');
xlabel('Time [s]');
ylabel('Frequency [Hz]');
colorbar;
%log scale
subplot(122);
imagesc([0 rt],[0 Fs/2] ,20*log10(abs(M3(1: r, :)).^2)), axis xy;
title('Power Spectrum on Log Scale');
xlabel('Time [s]');
ylabel('Frequency [Hz]');
colorbar;
```



Le nombre de fenêtres permet de déterminer la perception du son aux oreilles humaines. L'intensité en puissance permet de savoir si le son est fort ou pas.

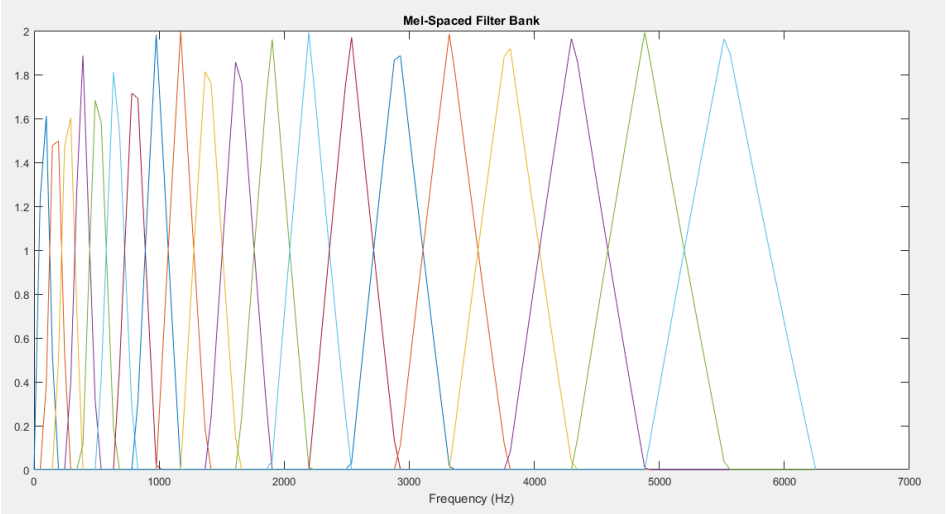
Question 5 :

On limite le spectre à 20 coefficients. Pour chaque signal entré, seulement 20 sinusoïdes seront considérées filtrées et traitées. En théorie, quand il y a plus de coefficients, les positions des triangles changent et varient, donc il y a plus de sinusoïdes qui peuvent être traitées.

On écrit le code suivant dans le main :

```
plot(linspace(0, (Fs/2), 129), ( ( melfb(20,256,Fs) ) ) );
title('Mel-Spaced Filter Bank') ;
xlabel('Frequency (Hz)');
```

L'échelle de Mel obtenue est la figure ci-dessous :

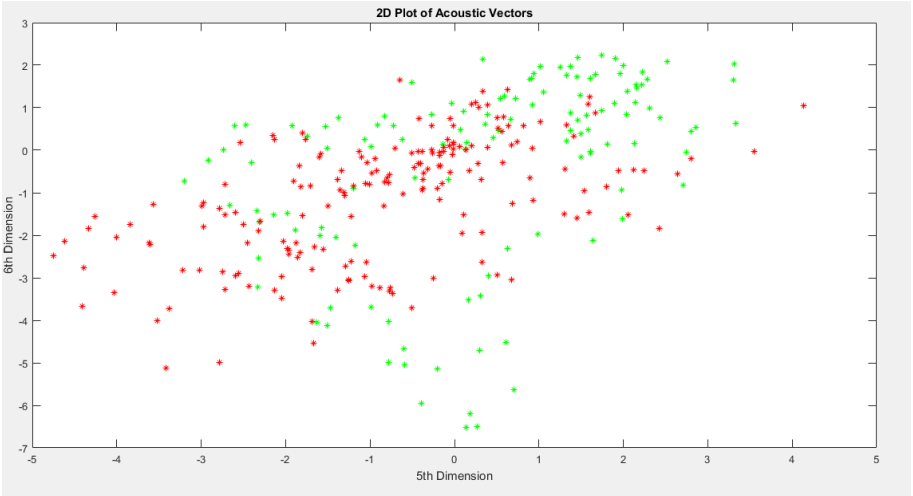


Question 6 :
Avant d'appliquer le programme melfb, l'énergie du signal son est dans la zone des basses fréquences. Après avoir appliqué le programme melfb, le niveau d'énergie est amplifié, ce qui permet d'améliorer la qualité du signal.

Quantification vectorielle

Nous avons transformé le signal son en vecteurs. Nous allons maintenant utiliser la technique de reconnaissance de forme basée sur la quantification vectorielle. Cela va permettre de construire des modèles de référence dans la phase "training", et puis identifier des séquences de vecteurs de n'importe quelle personne.

Question 7 :
Voici le graphe obtenu en affichant les points de données de la 5e et 6e dimension (sachant qu'il y en a 20)



Le code utilisé est :

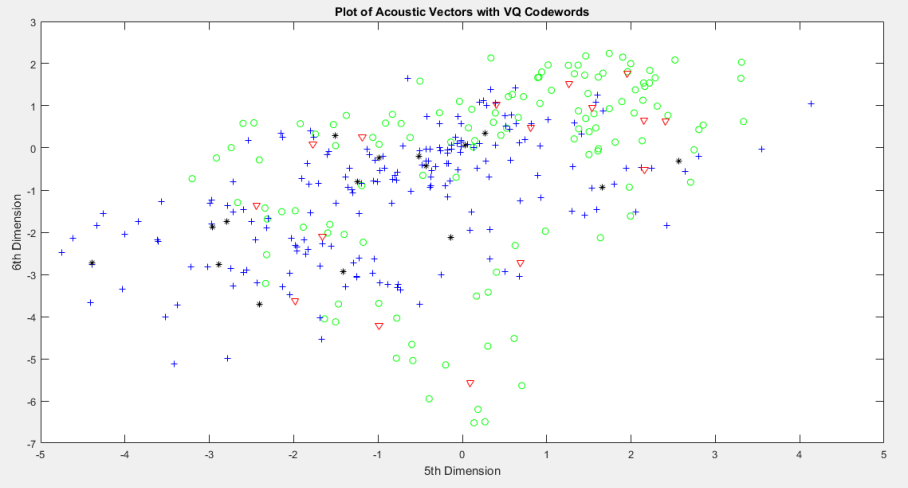
```
[ya1,Fa1]=audioread('s1.wav');
[ya3,Fa3]=audioread('s3.wav');
vec1 = mfcc(ya1,Fa1);
vec2 = mfcc(ya3,Fa3);
plot(vec1(5, :), vec1(6, :), '*g');
hold on;
plot(vec2(5, :), vec2(6, :), '*r');
title('2D Plot of Acoustic Vectors');
xlabel('5th Dimension');
ylabel('6th Dimension');
```

On s'aperçoit que certains points se chevauchent, on peut en déduire que les deux personnes disent le même mot.

Cependant, il n'y a pas de cluster, ce qui veut dire que ce n'est pas la même voix ; donc les deux personnes doivent être différentes.

Question 8 :

La figure ci-dessous représente les vecteurs acoustiques de s1.wav et s3.wav. Les "o" verts sont les vecteurs acoustiques de s1.wav et les "+" bleus sont de s3.wav. dans la phase "training", un dictionnaire (codebook) est généré pour chaque personne. Les barycentres obtenus sont les triangles rouges et noirs pour s1.wav et s3.wav respectivement.



Le code utilisé est :

```
[ya1,Fa1]=audioread('data\test\s1.wav');
[ya3,Fa3]=audioread('data\test\s3.wav');
vec1 = mfcc(ya1,Fa1);
vec2 = mfcc(ya3,Fa3);
vq1 = vqlbg(vec1, 16);
vq2 = vqlbg(vec2, 16);
plot(vec1(5, :), vec1(6, :), 'og');
hold on;
plot(vec2(5, :), vec2(6, :), '+b');
hold on;
plot(vq1(5, :), vq1(6, :), 'vr');
hold on;
plot(vq2(5, :), vq2(6, :), '*k');
title('Plot of Acoustic Vectors with VQ Codewords');
xlabel('5th Dimension');
ylabel('6th Dimension');
```

Simulation et Evaluation

Question 9 :

En écrivant ce code ci-dessous dans le main :

```
code = train('data\train\', 8);
test('data\test\', 8, code);
```

On obtient le résultat suivant :

```
Speaker 1 matches with Speaker 1
Speaker 2 matches with Speaker 2
Speaker 3 matches with Speaker 7
Speaker 4 matches with Speaker 4
Speaker 5 matches with Speaker 5
Speaker 6 matches with Speaker 6
Speaker 7 matches with Speaker 7
Speaker 8 matches with Speaker 8
```

On peut voir que seul le fichier s3.wav ne concorde avec aucun des fichiers. Mais, si on compare l'ordinateur avec la reconnaissance par l'oreille humaine, le programme arrive à faire un meilleur travail.

F. Bibliographie

- Mahima Garg, Omar Razi, Supriya Phutela, Vaibhav Kapoor, Varun Chopra

- https://github.com/gauravshelangia/Speaker-Recognition

- https://github.com/madanjhawar/Speaker-Recognition

- https://github.com/Jeevan-J/Speaker-voice-Recognition-using-MFCC-algorithm-in-matlab

- NEU Near East University

- Sujet 2 ECE Paris