Traitement Automatique de la Parole
M2 SID

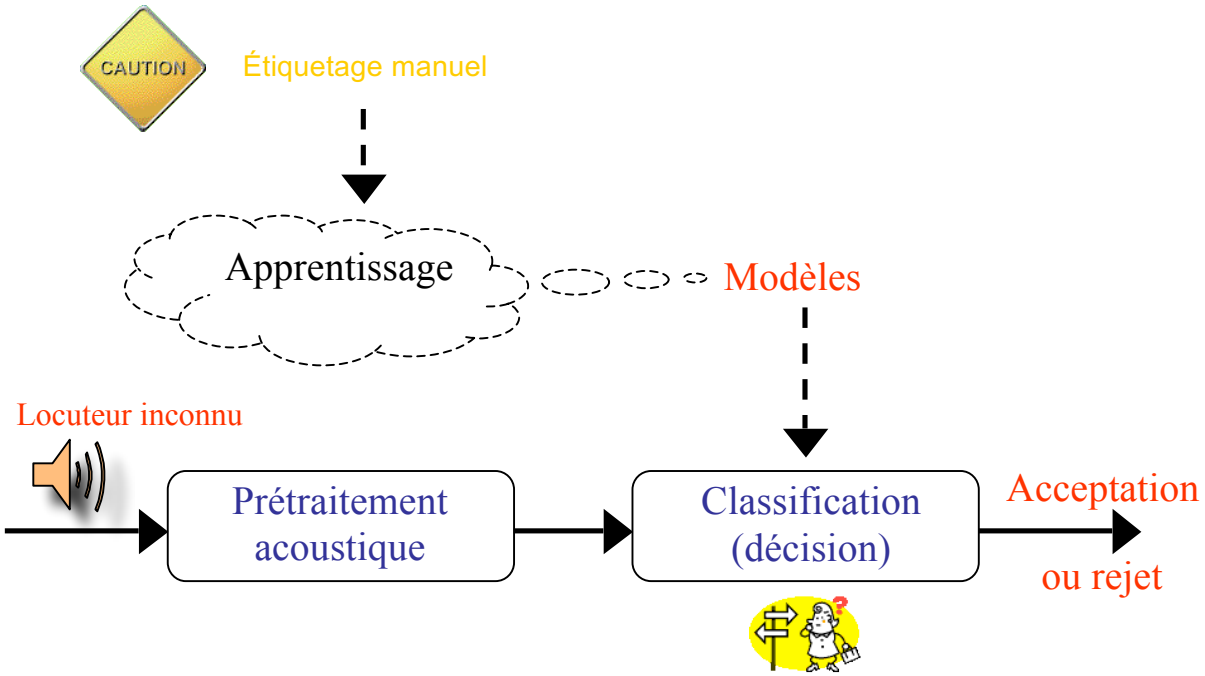
INTRODUCTION

Le but est de construire un système de vérification du locuteur et de le valider sous l'environnement Matlab.

Différentes étapes sont nécessaires à la construction et à la validation du système.

1) On utilisera tout d'abord une **base de données** (ou corpus) utile à l'apprentissage du système et ensuite à son évaluation : pour cela, on dispose de fichiers sons (ou signal).
2) Une étape de **décomposition parole/non-parole** sera nécessaire afin de ne conserver que les zones de parole (correspondant aux locuteurs).
3) Puis, on générera les fichiers acoustiques correspondants : phase de **paramétrisation**.
4) Ensuite, on fera l'**apprentissage** des modèles de mélanges de lois gaussiennes : il s'agira de construire le modèle du « monde » et le modèle du locuteur cible.
5) Enfin on passera à la phase de **reconnaissance** (décision) : le locuteur testé, est-il le locuteur cible ?

Voici le schéma général d'un système de vérification du locuteur :



1) PRESENTATION DE LA BASE DE DONNEES

La base de données est composée de 10 locuteurs francophones. Chaque locuteur possède 10 enregistrements (fichiers wave). La durée des fichiers varie de 13 à 34 secondes. Le découpage est le suivant : 8 enregistrements servent à l'apprentissage des modèles et 2 à la reconnaissance (tests).

Télécharger sur votre compte les fichiers audios présents sous Moodle.

Le répertoire « WAV » est divisé en 2 sous répertoire « APP » pour l'apprentissage des modèles et « RECO » pour la reconnaissance (tests). Les fichiers sont nommés de la façon suivante : **L_x_fic_y.wav** avec **x** le numéro du locuteur et **y** le numéro du fichier.

Remarque : pour débuter les premières expérimentations pourront se faire sur un seul fichier, par exemple « L1_fic1.wav ».

Ecrire une fonction « lecture.m » permettant de lire (charger dans une variable) sous Matlab un fichier son et de connaître ses caractéristiques (fréquence d'échantillonnage, nombre de bits de quantification, durée).
function [signal, fe, nb_bits, duree] = lecture(fichier)

2) DECOMPOSITION PAROLE/NON-PAROLE

Chaque fichier son n'est pas uniquement composé de parole ! En effet, des zones de silence et de bruit sont également présentes. Afin d'effectuer l'apprentissage des modèles (« monde » et « locuteur cible »), il est nécessaire d'isoler les zones de parole. Ce travail pourra s'appuyer sur le calcul de l'énergie à court terme. L'énergie sera calculée sur des fenêtres de taille « taille_fenetre » avec un recouvrement de moitié. Classiquement la variable « taille_fenetre » est égale à 256, 512 ou 1024 points (échantillons).

Ecrire une première fonction « energie.m » permettant de calculer l'énergie à court terme d'un signal en fonction de la taille des fenêtres d'analyse.
function nrj = energie(signal, taille_fenetre)

Ecrire une autre fonction « etiquetage.m » (utilisant la fonction « energie ») permettant d'étiqueter un fichier son en parole et non-parole sous Matlab.
function etiq_parole = etiquetage(signal, taille_fenetre)

La variable « etiq_parole » est composée de lignes de « 0 » (en l'absence de parole) et de « 1 » (en présence de parole).

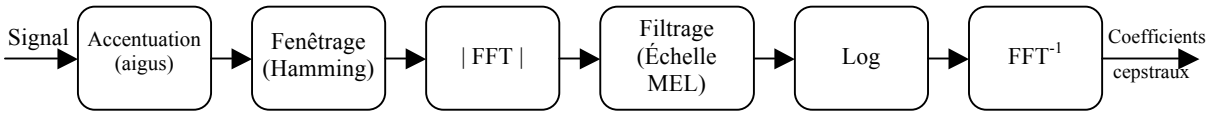
Ecrire une fonction « etiquetage_total.m » (utilisant la fonction « etiquetage ») permettant d'étiqueter en parole/non-parole l'ensemble des fichiers sonores du répertoire « WAV\APP ».
function etiquetage_total(taille_fenetre, nbe_loc, nbe_fic)

Les résultats de la fonction « etiquetage » seront enregistrés dans des fichiers texte portant le nom des fichiers son de départ avec l'extension « .lab » et seront rangés dans le répertoire « LABELS ».

3) PARAMETRISATION

Cette phase correspond à la représentation acoustique du signal. Nous utiliserons des coefficients cepstraux (MFCC). La taille de la fenêtre d'analyse « taille_fenetre » devra être la même que précédemment (cf. étiquetage) pour faciliter les traitements par la suite. Généralement le nombre de coefficients cepstraux « nbe_coef » est fixé à 8, 12 ou 16.

Voici le schéma classique d'extraction de ces paramètres :



Ecrire un programme permettant d'effectuer la paramétrisation (calcul des coefficients cepstraux).
function mfcc = parametrisation(signal, taille_fenetre, nbe_coef)

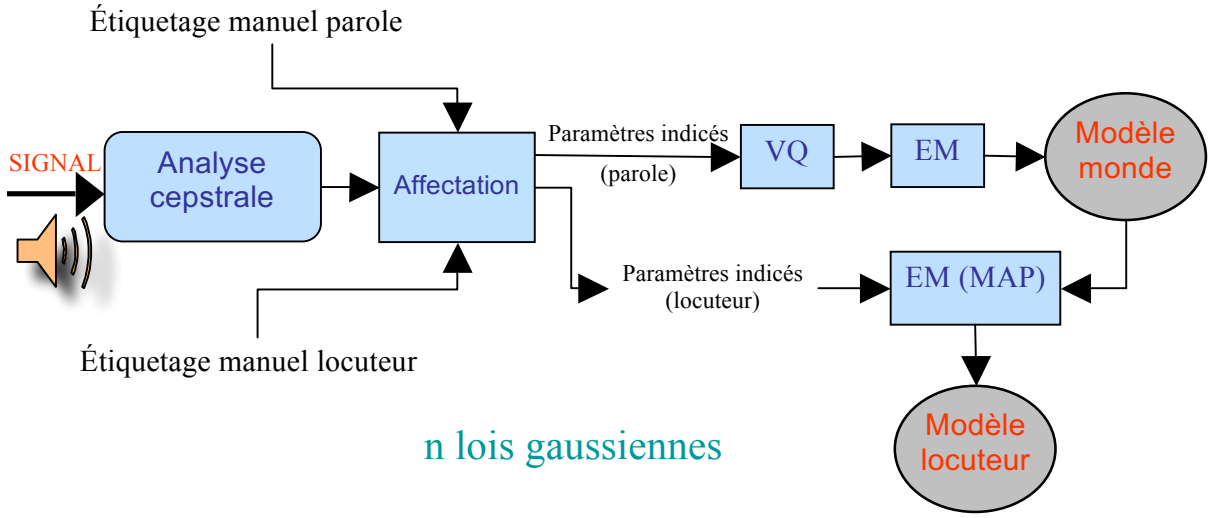
Le résultat de cette paramétrisation est un fichier composé d'une suite de valeurs telles :
$Coef_1$ $Coef_2$ … $Coef_{nbe\_coef}$

Ecrire une fonction « parametrisation_total.m » (utilisant la fonction « parametrisation ») permettant de calculer les MFCC pour l'ensemble des fichiers du répertoire « WAV\APP ».
function parametrisation_total(taille_fenetre, nbe_coef, nbe_loc, nbe_fic)

Les résultats de « etiquetage » seront enregistrés dans des fichiers texte portant le nom des fichiers de départ avec l'extension « .mfcc » et seront rangés dans le répertoire « MFCC ».

4) APPRENTISSAGE

Voici le schéma général permettant de créer les modèles du monde et du locuteur cible :



Afin d'effectuer l'apprentissage, il convient d'utiliser les vecteurs MFCC correspondant aux modèles. Pour cela, lors d'une phase d'affectation, les étiquetages en parole et en locuteur vont nous permettre de choisir les paramètres associés à chacun des modèles.

Utiliser la fonction « affectation.m » permettant de créer deux fichiers contenant l'ensemble des paramètres (MFCC) indicés nécessaire à l'apprentissage des modèles du « monde » et du « locuteur ». Le choix (numéro) du locuteur est passé en paramètres.
function affectation(repertoire_labels, repertoire_mfcc, nb_locuteurs, nb_fichiers, locuteur)

La première étape de l'apprentissage des modèles correspond à l'initialisation du modèle du « monde ». En général, une quantification vectorielle (VQ) est réalisée sur l'ensemble des paramètres (MFCC) correspondant à de la parole. Nous utiliserons, lors de ce TP, l'algorithme des « k-means ». Le nombre de centres (ou nombre de gaussiennes) est fixé à une puissance de 2 : nous prendrons ici 1, 2, 4, 8, ou 16. Nous utiliserons la fonction Matlab « gaussmix.m » pour faire l'apprentissage des modèles du monde et du locuteur choisit. Cette fonction fait appel à « kmeans.m » pour initialiser les modèles et elle fait ensuite la ré-estimation des modèles en utilisant l'algorithme d'optimisation EM.

Remarque : observer les entrées/sorties possibles et nécessaires à la fonction « gaussmix.m ».

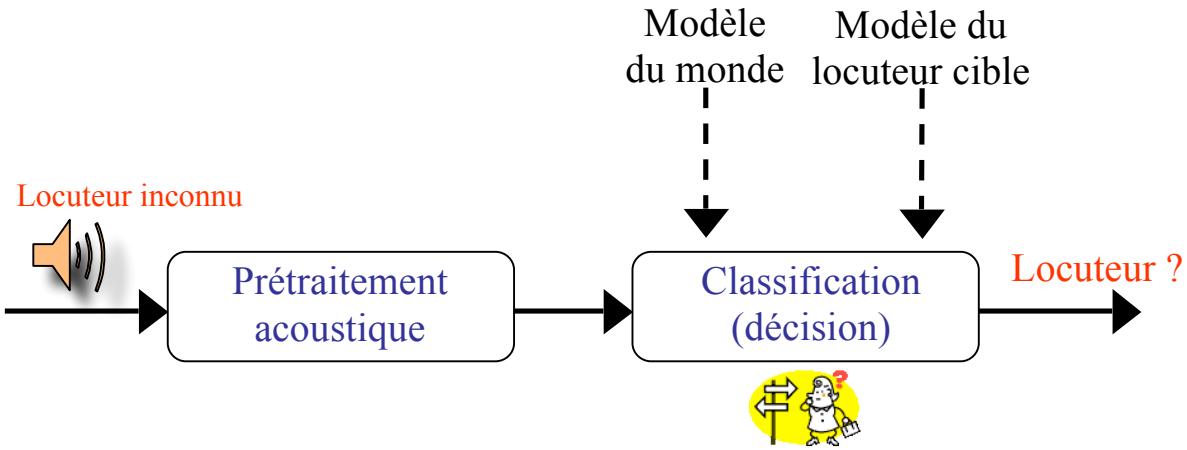
Ecrire la fonction « apprentissage.m » permettant d'effectuer l'apprentissage des modèles. Cette fonction prendra en entrée les vecteurs de données à traiter (fichiers « monde » et « locuteur ») ainsi que le nombre de lois gaussiennes « n ». Elle fournira en sortie les modèles du monde et du locuteur représentés par les poids (w), les moyennes (m) et les variances (v) de chacune des lois gaussiennes.

```
function [m, v, w, m_l, v_l, w_l] = apprentissage(fic_monde,fic_locuteur,n)
```

Remarque : le nombre de lois gaussiennes « n » de la fonction **apprentissage.m** est égal au nombre de centres « k » de la fonction **kmeans.m**.

5) RECONNAISSANCE

Voici le schéma général permettant d'effectuer la reconnaissance.



Afin d'effectuer cette phase, il convient de calculer les vecteurs MFCC, issus du prétraitement acoustique, du locuteur inconnu : pour cela, vous utiliserez les 20 fichiers sons contenus dans le répertoire « WAV/RECO ». Vous utiliserez également les 2 modèles issus de l'apprentissage : « monde » et « locuteur cible ». Votre système fonctionnera parfaitement si les 2 fichiers du locuteur cible sont acceptés par le système et que les 18 autres sont rejetés.

Nous utiliserons la fonction Matlab « gmmlpdf.m » pour calculer la vraisemblance (probabilité) des vecteurs acoustiques par rapport aux 2 modèles. Nous considérerons que la décision (acceptation/rejet du locuteur) s'effectue par un vote majoritaire (sur chacune des vraisemblances).

Ecrire la fonction « tests_total.m » permettant d'effectuer la reconnaissance des fichiers sons inconnus situés dans le répertoire « WAV/RECO ». Cette fonction prendra en entrée la taille de la fenêtre d'analyse, le nombre de coefficients cepstraux ainsi que les paramètres (poids, moyennes et matrices de covariance) de chacun des modèles, composés de mélanges de lois gaussiennes. Elle fournira en sortie un vecteur (ou matrice) du taux de reconnaissance par fichier.

```
function taux_reco = tests_total(taille_fenetre, nbe_coef, w, m, v, w_l, m_l, v_l)
```

Vous pourrez modifier, améliorer vos résultats en faisant varier l'ensemble des constantes/valeurs que vous avez utilisé pour construire ce système de vérification du locuteur, notamment :

- le nombre de lois gaussiennes : n,
- le nombre de paramètres (coefficients cepstraux) : nbe_coef,
- la taille de la fenêtre d'analyse : taille_fenetre,
- le nombre d'itérations et le seuil d'apprentissage,
- le seuil de l'énergie,
- le type de paramètres (en utilisant « rceps.m » par exemple),
- le nombre de fichiers d'apprentissage et de tests.