BE – Systèmes simples de reconnaissance de la parole

L'objectif de ce TD est de développer un système simple de reconnaissance de mots isolés. Deux types de paramétrisation du signal seront considérés (coefficients LPC et MFCC), ainsi que deux modèles acoustiques (classification des k plus proches voisins se basant sur la distance élastique et Modèles de Markov Cachés). Ces systèmes seront développés en langage Python.

0. Présentation des données fournies

Les données disponibles (fichier digit_dataset.zip), que vous pourrez compléter avec vos propres enregistrements, consistent en des enregistrements audio de 4 personnes prononçant en anglais des chiffres de 0 à 9.

Les noms des fichiers sont sous la forme 'chiffre_nom_index.wav', par exemple '1_theo_47.wav'.

Voici un code utilisant la librairie *librosa* pour charger en mémoire les données audio sous forme d'un vecteur à partir d'un fichier *wav* :

```
import librosa
audio, Fe = librosa.load('1_theo_47.wav')
```

Dans ce cas, *audio* est un vecteur numpy contenant les valeurs des échantillons audio et *Fe* est la fréquence d'échantillonnage.

Voici un autre exemple de code permettant de parcourir une arborescence afin d'accéder aux fichiers audio (.wav) :

```
import os
input_folder='./digit_dataset'
for dirname in os.listdir(input_folder):
    subfolder = os.path.join(input_folder, dirname)
    if not os.path.isdir(subfolder):
         continue
    for filename in [x for x in os.listdir(subfolder) if x.endswith('.wav')]:
         filepath = os.path.join(subfolder, filename)
         print(filepath)
```

Partie 1 : Développement d'un système de reconnaissance de la parole basé sur les coefficients LPC et la classification des k plus proches voisins utilisant la distance élastique.

1.1 Paramétrisation des données audio à partir des coefficients LPC

Afin de représenter les signaux audio, une paramétrisation sera réalisée, consistant à associer à chaque trame successive du signal (fenêtre), un vecteur des coefficients LPC (cf cours).

La méthode considérée ici pour le calcul des coefficients LPC repose sur l'équation de Yule-Walker exprimant la relation entre les coefficients LPC et les covariances du signal considéré :

$$\begin{bmatrix} R(0) & R(1) & \cdots & R(N) \\ R(-1) & R(0) & \ddots & \vdots \\ \vdots & \ddots & \ddots & R(1) \\ R(-N) & \cdots & R(-1) & R(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} \sigma^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Dans cette équation, les a_i sont les coefficients LPC d'un modèle d'ordre N, σ^2 est la puissance de l'erreur de prédiction et $R(i)$ est la fonction d'autocovariance définie par $R(i) = E(s(t+i)s(t))$. Comme le signal s est supposé stationnaire sur la durée de trame considérée (20 à 30 ms), $R(i) = R(-i)$ et donc la matrice de covariance est symétrique.

1.2 Calcul de la distance entre deux signaux paramétrisés

Afin de permettre la reconnaissance d'un mot, une fois le signal audio correspondant paramétrisé, il est nécessaire de procéder à une phase de classification.

La première technique de classification considérée dans le cadre de ce TD, est une classification par recherche des formes les plus similaires (algorithme des k plus proches voisins). Pour cela, la définition d'une distance appropriée doit être réalisée. Afin de prendre en compte la variabilité de prononciation des mots, nous allons considérer une distance élastique, fondée sur la programmation dynamique. Le principe a été présenté en cours, et consiste à mettre en correspondance les échelles temporelles des formes à reconnaître à l'aide de transformations non linéaires.

1.3 Classification par l'algorithme des k plus proches voisins

Le principe de l'algorithme de classification des k plus proches voisins est le suivant :

- Considérer une base de référence (ensemble de données dont on connait la classe)
- Pour une nouvelle donnée à classer, déterminer les k exemplaires de la base de référence les plus proches au sens d'une certaine distance (distance élastique dans notre cas) et affecter à cette donnée la classe majoritaire parmi ses k voisins.

1.4 Mise en œuvre

Le travail à réaliser consiste à écrire un programme Python permettant la réalisation d'un système simple de reconnaissance de mots isolés en se basant sur une paramétrisation du signal de parole à l'aide des coefficients LPC, et d'une reconnaissance par l'algorithme des k plus proches voisins, basé sur la distance élastique entre signaux paramétrisés.

Le programme doit donc :

- lire les fichiers audio
- calculer leur matrice des coefficients LPC
- calculer et afficher la matrice des distances entre les signaux audio
- réaliser la classification par l'algorithme des k plus proches voisins

Partie 2 : Développement d'un système de reconnaissance de la parole basé sur les coefficients MFCC et les Modèles de Markov Cachés.

2.1 Présentation et exemples de codes

Dans cette partie, la paramétrisation des données audio se fera à l'aide de la libraire *librosa* qui va permettre d'obtenir les coefficients MFCC, comme illustré dans le code suivant :

```
import librosa
from librosa.feature import mfcc

audio, Fe = librosa.load('1_theo_47.wav')
mfcc_features = mfcc(y=audio, sr=Fe, n_mfcc=15, win_length=512,
hop_length=512//2 )
```

La documentation est donnée ici :
https://librosa.org/doc/main/generated/librosa.feature.mfcc.html

- n_mfcc correspond au nombre de coefficients MFCC calculés dans chaque fenêtre
- win_length correspond à la longueur des fenêtres
- hop_length correspond au nombre d'échantillons entre deux fenêtres successives

Les modèles acoustiques seront des Modèles de Markov Cachés (un par élément à reconnaître). La librairie considérée ici permettant de les mettre en œuvre est *hmmlearn* :
https://hmmlearn.readthedocs.io/en/latest/

Le code suivant permet l'initialisation puis l'apprentissage d'un modèle. La matrice *X* de dimension (*nb_fenetres, nb_coeffs*) contient les données d'apprentissage pour le modèle (chaque ligne correspond aux coefficients MFCC calculés dans une fenêtre), et le vecteur *lengths* contient le nombre de fenêtres dans *X* pour chacun des fichiers audio (la somme des valeurs de *lengths* correspond au nombre de lignes dans *X*). *n_components* correspond au nombre d'états cachés et *n_iter* au nombre maximum d'itérations pour l'apprentissage.

```
from hmmlearn import hmm
model = hmm.GaussianHMM(n_components=4,n_iter=1000)
model.fit(X, lengths)
```

Le code suivant permet d'obtenir la probabilité qu'un modèle ait généré un signal audio caractérisé par ses coefficients :

```
score = model.score(mfcc_features)
```

En calculant cette probabilité pour tous les modèles, il est ainsi possible d'identifier la classe du signal, correspondant au modèle ayant la plus forte probabilité.

2.2 Mise en œuvre

Le travail à réaliser consiste à s'inspirer du programme de la partie 1 et des exemples ci-dessus pour mettre en œuvre un système simple de reconnaissance de mots isolés en se basant sur une paramétrisation du signal de parole à l'aide des coefficients MFCC, et d'une reconnaissance par les Modèles de Markov Cachés.

3 Travail à rendre

Ce travail (partie 1 et partie 2) peut être réalisé par monôme ou par binôme, et un compte-rendu numérique devra être produit, soit sous forme d'un notebook Python, soit sous forme d'une archive « .zip » contenant le code Python et le rapport au format pdf. Le compte-rendu devra contenir notamment les explications concernant le principe des fonctions que vous avez programmées ainsi que leur code commenté. Les expérimentations que vous aurez réalisées devront être décrites et les résultats également commentés. Ce travail devra être déposée sur le site « moodle » dans la rubrique « Rendus BE » -> « Rendu BE reconnaissance de la parole » pour le mercredi 8 janvier 2025.