TP Détecteur de parole (et de silence)

Objectif : construire un détecteur de parole et de silence en utilisant un modèle multi gaussien (GMM) appris avec l'algorithme EM.

Étape 1 : Préparation des données.

On extrait le logarithme de l'énergie à court terme (sur 30 ms) d'un enregistrement audio toutes les 10 ms. L'énergie à l'instant *t* est la suivante (Wikipédia pour plus d'information) ou *d*'un nombre d'échantillons correspondant à 30ms.

$$e_t = log(\sum_{i=t}^{t+d} x_i^2)$$

Hypothèse : Si la valeur de l'énergie à l'instant *t* est faible, nous supposerons que nous sommes en présence de silence. Au contraire, si elle est forte, nous supposerons que nous sommes en présence de paroles.

On va travailler avec un fichier audio d'une émission de radio.

a) La lecture du fichier se fait au moyen de la fonction scipy.io.wavfile.read(). Elle retourne la fréquence d'échantillonnage et le signal dans une matrice. Le fichier d'exemple a une fréquence de 16000 Hz, il y a donc 16000 échantillons en une seconde.

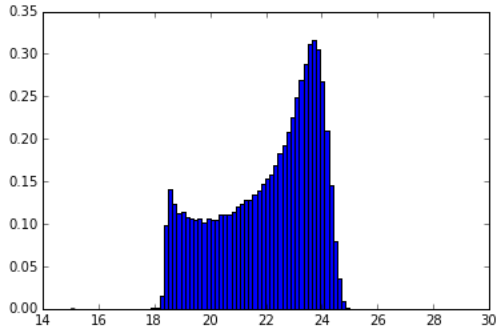
   Question 1 : quelle est la durée du fichier en secondes ?

b) Le signal va être "découpé" en fenêtres de 30 ms avec un pas de 10 ms. Il y a donc un recouvrement entre les fenêtres. La première fenêtre contient les échantillons de [0, 30[, la deuxième fenêtre couvre l'intervalle [10, 40[, le troisième intervalle de [20, 50[, etc.

   Question 2 : combien d'énergies seront extraites ?
   Question 3 : À combien d'échantillons correspondent respectivement 30 et 10 ms ?

c) Calculer l'énergie à court terme du signal de chaque fenêtre et stocker les valeurs dans une matrice colonne (ou dans une liste qui sera convertie en matrice à la fin, attention on veut un vecteur colonne). On note X, le vecteur des énergies.

   Question 4 : À partir de l'histogramme des énergies, que remarquez-vous (modes principales et secondaires de la distribution) ?



Note : le calcul des énergies étant relativement long, travailler dans un premier temps avec les 10000 échantillons.

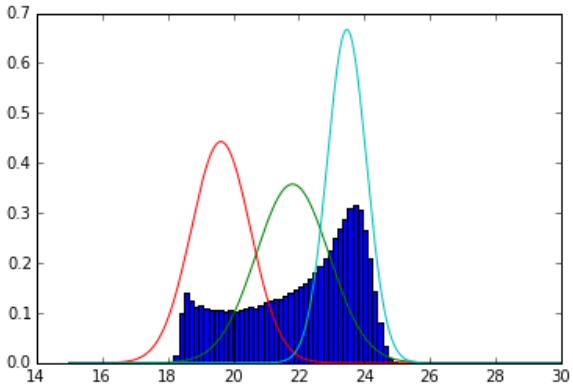
Étape 2 : Apprendre un GMM

La classe GMM est disponible dans *sklearn.mixture.* À la création de l'instance GMM, vous fixerez le nombre de composantes à 3 avec 10 itérations d'apprentissage, la forme de la covariance. Mettre aussi *verbose* à 10 pour obtenir un retour sur le déroulement de l'apprentissage.

L'apprentissage se fait avec la méthode *gmm.fit(X).* Le corpus d'apprentissage (les énergies) doit être un vecteur colonne.

Question 5 : quelle sera la dimension des gaussiennes ?
Question 6 : quelle est la méthode d'initialisation du modèle ?
Question 7 : À quoi correspondent les valeurs "change" dans la sortie de l'apprentissage ?
Question 8 : est-il important de fixer à *diag* ou *full* la forme des matrices de covariances ?
Question 9 : quelle est la vraisemblance du modèle sur les données d'apprentissage ?

Les attributs *means_* et *covariances_* vous donnent respectivement la moyenne et les covariances du modèle. Afficher chaque gaussienne en utilisant la fonction *scipy.stats.norm.pdf*. Attention, nous avons besoin des écarts-types ! Ajouter aussi l'histogramme des énergies (normaliser les valeurs : *density=True* de la fonction *hist*)




```
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
import math

mu = 0
variance = 1
sigma = math.sqrt(variance)
x = np.linspace(mu - 3*sigma, mu + 3*sigma, 100)
plt.plot(x, stats.norm.pdf(x, mu, sigma))
plt.show()
```

Étape 3 : Détection de parole

On va supposer que les 2 gaussiens avec les moyennes les plus grandes modélisent la parole. On va travailler avec la gaussienne du milieu (utiliser la fonction médian). On suppose que les valeurs d'énergies inférieures à la médiane moins 2 fois son écart-type sont du silence.

Sauver dans un fichier Wave les échantillons de silence (attention aux fenêtres) à partir de la valeur des énergies. Utiliser *wavfile.write.*

Construire un vecteur *E* de même dimension que *X* qui contient la décision 0 pour silence, 1 pour parole. Ce vecteur *E* est utile dans la partie 2 du TP.

Étape 4 : Algorithme EM

Implémenter votre propre algorithme EM. Utiliser le cours pour les formules.