TP MCS 1 à 3 : Reconnaissance de commandes audio par DTW

Nom du Groupe :

Noms :

Prénoms :

Ces trois séances de TP (2 séances encadrées et une non encadrée) vont vous permettre de tester l'algorithme de programmation dynamique vu en TD puis de réaliser la mise en oeuvre d'un système de reconnaissance audio de mots isolés (constituant des commandes pour les drones).

Ces séances se décomposent en 3 parties :

- Partie I : DTW et application du TD
- Partie II : Système de reconnaissance audio de mots de commande
- Partie III : Comparaison de la programmation dynamique avec une méthode de classification après prétraitement des données

Pour les **parties II et III**, vous testerez le système de reconnaissance audio sur deux corpus de voix qui serviront respectivement de base d'apprentissage (références) et de base de test (sons à reconnaître) que vous choisirez. La liste des 13 commandes au drone sont : *Atterrissage, Décollage, Avance, Tourne droite, Recule, Tourne gauche, Droite, Fais un flip, Gauche, Arrête toi, plus haut, plus bas et Etat d'urgence.*

Pour cela, vous devez par groupe de 3 étudiants (effectif **OBLIGATOIRE**):

1. **Proposer une étude** que vous détaillerez sur un rapport [par exemple, *influence voix masculines VS voix féminines, confronter vos propres voix à la base de données, tester l'impact de différents bruits de fond sur la reconnaissance...*];
2. Créer, en fonction de l'objectif de votre étude, vos propres base d'apprentissage et base de test à partir du corpus proposé et des voix et bruits que vous aurez enregistrés [*paramètres audio : 16 KHz, mono, 16 bits, format* .wav**];
3. Tester la DTW et une méthode de classification avec prétraitement par ACP;
4. Evaluer les résultats;
5. Rédiger un rapport en pdf présentant l'étude, les résultats par les 2 méthodes et vos commentaires et conclusions sur votre étude (Longeur max. : 5 pages).

In [5]:
```
import matplotlib.pyplot as plt
from numpy import array, zeros, full, argmin, inf, ndim
import scipy
import sklearn
import math
```

Partie I : Implémentation de l'algorithme de programmation dynamique

1. Ecrivez une fonction en python DTW qui implémente le calcul et l'affichage de la matrice des coûts définie en TD.
2. Afin d'adapter facilement le calcul des coûts suivant la nature des données (et donc des distances utilisées), écrivez une fonction pour chaque distance (euclidienne, lettres, sons) qui apparaîtra en paramètre de la fonction DTW.

Application aux exercices

1. Testez vos programmes sur les exercices vus en TD.
2. Modifiez les contraintes locales c'est-à-dire les pondérations suivant les directions.
3. Ajoutez la prise en compte de contraintes globales c'est-à-dire le non calcul lorsque les cases sont trop éloignées de la diagonale (cf exercice TD séquence ADN). A partir de quelle position les contraintes globales ne changent pas les résultats ?

Partie II : Système de reconnaissance audio de mots de commande

Sur l'espace partagé, vous trouverez des enregistrements audio de mots de commandes pour un drone quadricoptère constitués de plusieurs locuteurs masculins (notés M01..M13) et locutrice féminines (F01..F03).

Vous pouvez diviser ainsi l'ensemble des données en base d'apprentissage qui serviront de références et base de test pour évaluer la reconnaissance par programmation dynamique.

In [ ]:
```
import librosa
```

Les lignes de code suivantes permettent de transformer le fichier audio en matrice de paramètres appélés MFCC (Mel Frequency Cepstral Coefficient) en utilisant la librairie python *librosa*. Ces paramètres permettent d'extraire au mieux le contenu vocal fréquenciel de signal audio.

La matrice de sortie est composée d'autant de vecteurs colonnes que de trames d'analyses. Le nombre de lignes correspond à la dimension du vecteur représentatif : ici 12.

Chargement d'un fichier audio :

In [ ]:
```
y, sr = librosa.load(fichier_audio, offset=30, duration=5)
```

Calcul des MFCC

In [ ]:
```
mfcc = librosa.feature.mfcc(y=y, sr=sr, hop_length=1024, htk=True,
n_mfcc=12)
```

Application de la DTW

1. Réaliser une étude que vous détaillerez sur un rapport (par exemple, *influence voix masculines VS voix féminines, confronter votre propre voix à la base de données, tester l'impact de différents bruits de fond sur la reconnaissance...*) et créer votre propre base d'apprentissage et votre base de test à partir du corpus et des voix et bruits que vous aurez enregistrés.
2. Appliquer la DTW sur vos corpus.

Paramètres pour enregistrements audio de vos voix perso:

16 KHz, mono, 16 bits, format *.wav*

Evaluation de la reconnaissance

1. Calculer la matrice de confusion du système (en ligne les références et en colonne les sorties du système). Vous pourrez utiliser la fonction *confusion_matrix* de la librairie *sklearn*.

1. Calculer le score de reconnaissance : nombre de fichiers bien reconnus sur nombre de fichiers testés.

Vérifications :

- si vous prenez comme fichier de référence et de test M01, vous devez obtenir aucune erreur.
- si vous prenez comme fichier de référénce M01 et fichier de test M02, vous devez obtenir deux erreurs.

Partie III : Comparaison de la programmation dynamique avec une méthode de classification après prétraitement des données

Dans cette partie, nous allons comparer les résultats de la DTW avec ceux d'une méthode de classification de données : les k-plus proches voisins.

Nous utiliserons les fonctions permettant de calculer l'ACP et les kppv via la librairie python *scikit-learn*.

In [6]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from mpl_toolkits.mplot3d import Axes3D
```

Prétraitement par ACP

Pour tester une méthode de classification, il faut d'abord réduire la dimension des MFCC

1. A partir de tous les enregistrements de la base d'apprentissage, réaliser une Analyse en composantes principales (A.C.P) en utilisant la fonction *PCA* de la librairie *scikit-learn* puis projeter les données de test dans cette nouvelle base.

Remarque : vous pouvez aussi implémenter l'ACP en extrayant les 3 vecteurs propres, notés X_1, X_2, X_3, associés aux 3 plus grandes valeurs propres de la matrice de variance-covariance Σ_{App} (par les fonctions *np.cov* et *np.linalg.eig*). Ces vecteurs propres constitueront le nouveau repère P. Projetez ensuite les données de la base d'apprentissage et de test dans cette nouvelle base en multipliant chaque vecteur par la base $P = [X_1 X_2 X_3]$.

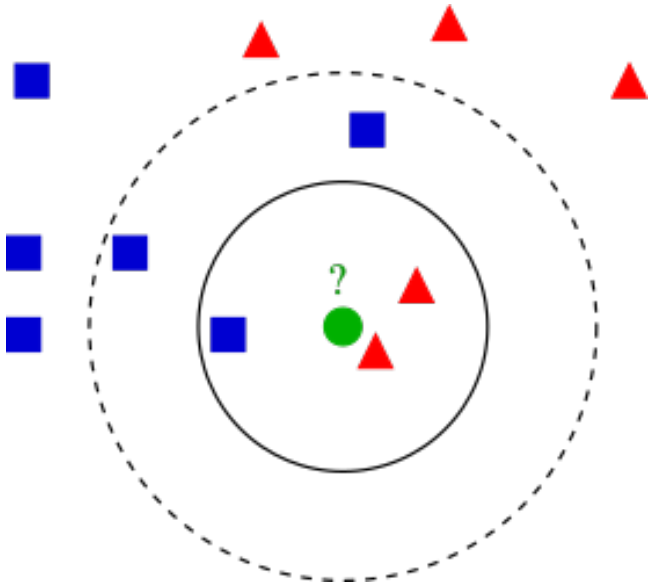
Classification par k plus proches voisins

En intelligence artificielle, la méthode des k plus proches voisins (k-ppv) est une méthode d'apprentissage supervisé. Dans ce cadre, on dispose d'une base de données d'apprentissage constituée de couples "donnée-label". Pour estimer la sortie associée à une nouvelle entrée x, la méthode des k plus proches voisins consiste à prendre en compte (de façon identique) les k échantillons d'apprentissage dont l'entrée est la plus proche de la nouvelle entrée x, selon une distance à définir. L'algorithme 1 associé et un exemple (figure 1) sont données par la suite.

Algorithm 1 Algorithme des k-ppv

Input : $Data_A$ et $Label_A$ ensemble de données et labels d'apprentissage, $Data_T$ ensemble de test.

1. Soit $x \in Data_T$ le point dont on cherche les k-ppv au sens d'une distance d.
   Calculer les distances $d(x, x_i)$, $\forall x_i \in Data_A$.
2. Trouver les k points $x_k \in Data_A$ plus proches voisins de x au sens de la distance d
3. Déterminer la classe C la plus représentée parmi les k plus proches voisins de x.
4. Assigner la classe C à la donnée x.



Exemple de classification par k-ppv. L'échantillon de test (cercle vert) doit être classé soit dans la première classe des carrés bleus, soit dans la deuxième classe des triangles rouges. Si k = 3 (cercle plein), il est assigné à la deuxième classe parce qu'il y a 2 triangles et seulement 1 carré à l'intérieur du cercle intérieur. Si k = 5 (cercle en pointillés), il est assigné à la première classe (3 carrés contre 2 triangles à l'intérieur du cercle extérieur)

1. En utilisant la fonction *KNeighborsClassifier* de la librairie *sklearn.neighbors*, réalisez une classification par k-ppv sur la base d'apprentissage et la base de test que vous avez prédéfinies (prendre $k = 1$).
2. Evaluez la méthode des k-ppv par le calcul de la matrice de confusion et du taux de reconnaissance.
3. Modifiez la valeur de k pour les k-ppv. Améliorez-vous les scores de reconnaissance ?
4. Comparez vos résultats avec ceux de la DTW.