TP4 – Programmation dynamique (2/2)

Algorithme DTW

L'algorithme DTW (*dynamic time warping*), qui s'inspire du principe de la *programmation dynamique*, permet d'effectuer l'*alignement* de deux séquences. Vous allez l'appliquer à l'alignement de séquences d'ADN, puis à l'alignement d'enregistrements sonores, afin de réaliser une application de commande vocale.

Exercice 1 : alignement de séquences d'ADN

La fonction `alignement` est censée être générique, donc permettre d'aligner des séquences de différents types. Vous l'utiliserez donc aussi bien pour aligner des séquences d'ADN que des enregistrements sonores. En revanche, la distance entre un item de la première séquence et un item de la deuxième séquence dépend de l'application visée. C'est la raison pour laquelle, en plus des deux séquences, la fonction `alignement` comporte un troisième paramètre d'entrée appelé `distance`. Elle retourne la matrice `g` des pénalités cumulées et le `score` de l'alignement optimal (ces deux paramètres de sortie ont été décrits en cours).

Écrivez la fonction `distance_ADN`, dont le résultat vaut 1 si les deux lettres sont différentes, et 0 sinon. L'en-tête de cette fonction se déduit de son appel, dans la fonction `alignement`, par la fonction `feval` (lisez la documentation de cette fonction). Lancez le script `exercice_1`, qui compare les deux séquences d'ADN manipulées en cours. Vérifiez que votre résultat coïncide bien avec celui du cours.

Exercice 2 : alignement d'enregistrements sonores

Le répertoire WAV contient 78 enregistrements sonores correspondant aux mêmes six locutions (*Droite*, *Gauche*, *Plus bas*, *Plus haut*, *Tourne droite*, *Tourne gauche*) prononcées par treize locuteurs masculins. La fonction `parametrage` transforme un enregistrement sonore au format WAV en une matrice au format MFCC (*mel-frequency cepstral coefficients*), de taille $12 \times n$, où 12 est le nombre de « coefficients cepstraux » et n le nombre de « trames d'analyse temporelle ». L'alignement consiste donc à mettre en correspondance les colonnes de deux MFCC, à l'aide de la distance euclidienne dans $\mathbb{R}^{12}$.

Écrivez la fonction `distance_MFCC`, puis lancez le script `exercice_2`, qui compare le mot *Droite* prononcé par le premier locuteur, avec le même mot prononcé par l'ensemble des locuteurs.

Le script `exercice_2_bis` calcule la *matrice de confusion* entre les deux premiers locuteurs, le premier locuteur étant considéré comme la référence. Modifiez ce script de manière à calculer la matrice de confusion des douze locuteurs autres que le premier locuteur, considéré comme le locuteur de référence.

Exercice 3 : application à la commande vocale

Complétez le script de nom `exercice_3`, qui permet d'interpréter un enregistrement tiré au hasard (grâce à la fonction `randi` de Matlab) parmi ceux du répertoire WAV. L'enregistrement est lu à l'aide des fonctions `wavread` et `sound`, puis une action correspondant au mot reconnu est exécutée. En l'occurrence, cette action consiste à déplacer une image, conformément à la locution reconnue.