Chapitre 4

Modèles de Markov cachés

Certains processus réels produisent des séquences de symboles observables, discrets (résultats du jet d'une pièce ou d'un dé, lettres, mots, nucléotides, acides aminés, ...) ou continus (signaux sonores, vecteurs, etc). On peut alors souhaiter construire un modèle de ces séquences permettant de les caractériser ou de les prédire. Les Modèles de Markov Cachés (HMM en anglais) sont l'un des formalismes les plus utilisés.

Les Modèles de Markov Cachés datent de la fin des années 1950. Ils ont d'abord été appliqués à la reconnaissance de la parole : voir par exemple : *A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition* de Rabiner (1988). De très nombreuses applications ont suivi, notamment en TAL (traitement automatique du langage naturel), bio-informatique (particulièrement en génomique) et dans tous les domaines faisant intervenir des séries temporelles.

4.1 Définition des HMMs

Les HMM décrivent un processus stochastique double dans lequel la séquence de symboles observés s'explique par un second processus stochastique qui énumère des états cachés à l'observateur.

Exemple 6 : on dispose de deux pièces de monnaie. L'une (E) est équilibrée : $P(Pile) = P(Face) = 0.5$. L'autre (B) est biaisée : $P(Pile) = 0.7, P(Face) = 0.3$. Au départ du processus, on choisit l'une d'entre elles au hasard, avec une probabilité uniforme (on ne sait pas quelle pièce est choisie). Chaque étape du processus consiste (i) à jeter la pièce choisie et noter la face apparente, (ii) à changer éventuellement de pièce, avec une probabilité de 0.1. Les symboles observés sont : $Pile$ et $Face$. Les états cachés peuvent être

représentés par E et B.

Les questions qui se posent naturellement :

— étant donnée une séquence observée

$$s = PPPFPFPPPFPPFFPPPFP,$$

quelle est la séquence d'états cachés sous-jacents la plus vraisemblable ?

— avec le modèle décrit ci-dessus, quelle est la probabilité d'observer la séquence s ?

— étant donnée une séquence observée s de longueur 1000, quel est le jeu de paramètres (d'un HMM à deux états) qui en rend le mieux compte ?

— étant donnée une séquence observée s de longueur 10000, quel est le HMM qui en rend le mieux compte ?

Définition 1 *Un HMM est un quintuplet $\langle O, S, \pi, T, E\rangle$ où*

— *O est un ensemble fini de symboles observables,*

— *S est un ensemble fini d'états,*

— *$\pi \in \mathbb{R}^{|S|}$ est le vecteur des probabilités initiales de démarrer le processus dans un état donné : $\forall i, \pi(i) \geq 0$ et $\sum \pi(i) = 1$,*

— *$T \in \mathbb{R}^{|S|\times|S|}$ est la matrice des probabilités des transitions entre états : $\forall i, j, T(i,j) \geq 0$ et $\forall i, \sum_j T(i,j) = 1$,*

— *$E \in \mathbb{R}^{|S|\times|O|}$ est la matrice des probabilités des émissions, différentes dans chaque état : $\forall i, j, E(i,j) \geq 0$ et $\forall i, \sum_j E(i,j) = 1$.*

Remarque : dans cette définition, on suppose que les symboles observables et les états sont rangés dans des listes, et on identifie ces éléments à leur rang dans la liste. Cela permet de parler de l'état (d'indice) i et de l'observation (d'indice) j.

Exemple 7 $O = \{Pile, Face\}$, $S = \{E, B\}$, $\pi = (0.5, 0.5)^\top$, $T = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$, $E = \begin{pmatrix} 0.5 & 0.5 \\ 0.7 & 0.3 \end{pmatrix}$.

Les HMMs sont des modèles probabilistes qui peuvent être utilisés pour générer des séquences d'observables de longueur n :

```
choisir un état s avec probabilité pi(s)
pour i = 1 à n faire :
  générer o avec probabilité E(s,o)
  choisir s' avec probabilité T(s,s')
  s = s'
```

Propriété : pour chaque entier $n \geq 1$, un HMM définit une distribution de probabilités sur l'ensemble O^n (suite d'observables de longueur n).

Soit $o_1 o_2 \dots o_n$ une suite de symboles observés, et $s_1 s_2 \dots s_n$ une suite d'états, on a :

$$P(s_1, s_2, \dots, s_n) = \pi(s_1) \prod_{i=1}^{n-1} T(s_i, s_{i+1}) \tag{4.1}$$

$$P(o_1 o_2 \dots o_n | s_1 s_2 \dots s_n) = \prod_{i=1}^{n} E(s_i, o_i) \tag{4.2}$$

et

$$P(o_1 o_2 \dots o_n) = \sum_{s_1 s_2 \dots s_n} \pi(s_1) \prod_{i=1}^{n-1} T(s_i, s_{i+1}) \prod_{i=1}^{n} E(s_i, o_i). \tag{4.3}$$

Remarque : ce n'est évidemment pas en programmant directement ces formules qu'on peut espérer calculer la probabilité d'une séquence d'observations (pourquoi ?).

On définit parfois une variante des HMMs permettant de définir une distribution de probabilités sur l'ensemble O^* des mots construits sur l'alphabet O.

Définition 2 *Un HMM généralisé est un sextuplet $\langle O, S, \pi, T, E, F \rangle$ où*

- *O est un ensemble fini de symboles observables,*
- *S est un ensemble fini d'états,*
- *$\pi \in \mathbb{R}^{|S|}$ est le vecteur des probabilités initiales : $\forall i, \pi(i) \geq 0$ et $\sum \pi(i) = 1$,*
- *$T \in \mathbb{R}^{|S| \times |S|}$ est la matrice des probabilités des transitions : $\forall i, j, T(i,j) \geq 0$ et $\forall i, \sum_j T(i,j) \leq 1$,*
- *$E \in \mathbb{R}^{|S| \times |O|}$ est la matrice des probabilités des émissions : $\forall i, j, E(i,j) \geq 0$ et $\forall i, \sum_j E(i,j) = 1$,*
- *$F \in \mathbb{R}^{|S|}$ est le vecteur des probabilités finales : $\forall i, F(i) \geq 0$ et $\forall i, \sum_j T(i,j) + F(i) = 1$.*

Propriété : un HMM généralisé définit une distribution de probabilités sur l'ensemble O^* (suite d'observables de longueur finie).

$$P(o_1 o_2 \dots o_n) = \sum_{s_1 s_2 \dots s_n} \pi(s_1) \prod_{i=1}^{n-1} T(s_i, s_{i+1}) \prod_{i=1}^{n} E(s_i, o_i) F(s_n). \tag{4.4}$$

Exercice 11

1. Implémentez une classe `HMM_Class.py` en prévoyant les contrôles de saisie adaptés.

```
class HMM:
    """ Define an HMM
    """

    def __init__(self, nbL, nbS, initial, transitions, emissions):
        # The number of letters
        self.nbL = nbL
        # The number of states
        self.nbS = nbS
        # The vector defining the initial weights
        self.initial = initial
        # The array defining the transitions
        self.transitions = transitions
        # The list of vectors defining the emissions
        self.emissions = emissions
```

   Indications :
   — le vecteur initial et les matrices d'émissions et de transitions seront représentés par des `numpy array`
   — on peut utiliser la primitive `isinstance(x, np.ndarray)` contrôler la saisie
   — les dimensions de ces paramètres doivent être compatibles : utiliser `numpy.shape`
   — pour vérifier qu'un réel est égal à 1, aux erreurs d'arrondi près, on peut utiliser `numpy.isclose(x,1.0)`
   — pour tester que deux tableaux `numpy` sont égaux : `np.testing.assert_array_equal(x,y)`. D'autres tests sont disponibles dans le module `testing`.

2. On définit un format de fichier texte pour mémoriser des HMMs :

```
# The number of letters
2
# The number of states
2
# The initial transitions
0.5
0.5
# The internal transitions
```

```
0.9 0.1
0.1 0.9
# The emissions
0.5 0.5
0.7 0.3
```

Écrivez la méthode statique `load(adr)` qui permet de charger un HMM à partir d'un fichier texte au format ci-dessus. On pourra supposer que toutes les lignes commençant par le caractère `#` sont ignorées.

3. Écrivez la méthode `gen_rand(self, n)` qui génère aléatoirement une séquence de longueur n à partir d'un HMM. Indications : on pourra utiliser la méthode `random` du module `random` qui génère aléatoirement un nombre réel compris entre 0 et 1 ; ne pas oublier d'initialiser (une fois seulement) le générateur aléatoire à l'aide de la méthode `seed()` ; on pourra écrire une méthode statique `draw_multinomial(L)` qui prend en entrée une liste (ou un np.array) dont les éléments sont positifs et somment à 1, et retourne un indice de la liste (ou du tableau) selon le modèle multinomial correspondant.
4. Écrivez la méthode `save(self, adr):` qui permet de sauver un HMM dans un fichier texte selon le format ci-dessus.
5. Proposez une implémentation permettant de faire cohabiter les HMMs et les HMMs généralisés.