Travaux Pratiques – Chaînes de Markov-HMMs

MDI 341

Mars 2019

Laurence Likforman-Sulem, Hamid Jalalzai, Magdalena Fuentes, Thomas Moreau

TELECOM ParisTech

Ce TP se déroule en deux parties. La première partie concerne les chaînes de Markov et est à finir absolument. Les plus rapides d'entre vous pourront aborder la deuxième partie qui traite des HMMs. Les ressources (matrices de transition, modèles HMMs…) sont sur :
Partie I : http://perso.telecom-paristech.fr/~lauli/TP_Chaine_Markov/
Partie II : http://perso.telecom-paristech.fr/~lauli/TPHMM_MDI343/

I.1 Fonctions dont vous pouvez avoir besoin

cumsum : fonction de répartition : somme cumulative des termes du vecteur *p*

```
import numpy as np
np.cumsum(p)
```

random : génère un nombre aléatoire entre [0 1] selon une loi uniforme.

```
import numpy as np
np.random.random()
```

load : chargement d'un fichier

```
filename_A= 'bigramenglish.txt'
np.loadtxt(filename_A)
```

Création du dictionnaire clé-valeur associant états et caractères :

```
dic={'1' : ' ', '2' : 'a', '3' : 'b', '4': 'c', '5' : 'd', '6' : 'e', '7': 'f',  '8' : 'g', '9' : 'h', '10': 'i', '11': 'j', '12' : 'k', '13' : 'l', '14': 'm', '15' : 'n', '16' : 'o', '17': 'p', '18' : 'q', '19' : 'r' , '20': 's',  '21' : 't', '22' : 'u', '23': 'v',  '24' : 'w', '25' : 'x' , '26': 'y',  '27' : 'z', '28' : ' ' }
```

Inverser le dictionnaire (les valeurs deviennent les clés et réciproquement)

```
dic_inv = {v: k for k, v in dic.items()}
```

I.2 Chaîne de Markov

On veut générer des mots dans une langue donnée en modélisant la formation de ces mots par une chaîne de Markov. Les 28 états du modèle correspondent aux 26 lettres de l'alphabet auxquelles et on ajoute un état 'espace initial' (état 1) et un état 'espace final' (état 28) qui sert à terminer les mots.

La correspondance entre la valeur numérique d'un état et un caractère est la suivante : l'état 1 correspond à un espace (avant le début d'un mot) et l'état 28 à celui d'un espace en fin de

mot. Les états 2 à 27 correspondent aux caractères de a à z. On pourra utiliser une structure de dictionnaire en python pour faire cette correspondance.

On utilisera une chaîne de Markov ergodique entre les 26 états correspondants aux lettres de l'alphabet.

I.2.a Matrice de transitions

'bigramenglish.txt' contient la matrice des transitions pour l'anglais (bigrams) entre deux symboles (caractères ou espaces). Le terme générique (i,j) de la matrice de transition correspond à la probabilité de transiter vers l'état j à partir de l'état i.
A quelles probabilités correspond la première ligne de la matrice de transition ? et celles de la dernière colonne ?
Pour chaque lettre de l'alphabet, indiquer la transition la plus fréquente depuis cette lettre.

I.2.b Générer un mot

On veut générer un mot à partir de l'état initial 1 (espace de début de mot).

Ecrire une fonction *etat_suivant* qui génère un état (à t+1) à partir de l'état courant (à t) et à l'aide de la matrice de transitions et de la fonction de répartition.
Afficher sur un graphique la fonction de répartition pour une ligne de la matrice de transition et expliquer son rôle pour la génération de l'état à t+1.

Utiliser cette fonction pour écrire la fonction *genere_state_seq* qui génère une séquence d'états jusqu'à aboutir à l'état final (28). Ecrire une fonction *display_seq* qui transforme une séquence d'états en séquence de caractères, à l'aide d'un dictionnaire. Utiliser ces fonctions pour générer des mots et donner des exemples de mots générés.

I.2.c Générer une phrase

On veut générer une suite de mots (phrase). Créer un état final de phrase (état 29, correspondant au caractère .) dont la probabilité de transition vers cet état depuis un état final de mot est 0.1. Ecrire une fonction *modifie_mat_dic* qui modifie la matrice de transition et le dictionnaire en conséquence. Donner des exemples de phrases générées.

I.3. Reconnaissance de la langue

Charger la matrice des transitions entre caractères pour le français. Ecrire une fonction *calc_vraisemblance* qui calcule la vraisemblance du modèle français pour une phrase donnée en multipliant les probabilités de transition. Pour tenir compte de toutes les transitions (notamment celles entre espaces de fin et de début de mots et vers la fin de phrase) on pourra transformer une séquence « mot1 mot2.» par la séquence « -mot1+-mot2+.», les signes - , + et . représentant l'état initial de mot, l'état final de mot et l'état final de phrase, respectivement.
Calculer la vraisemblance des modèles français et anglais pour la phrase « to be or not to be ».
De même calculer la vraisemblance des modèles français et anglais pour la phrase « etre ou ne pas etre ».

Fin de la Partie I

II. 1. Introduction

L'objectif de cette partie est de générer des séquences d'observations suivant un modèle de Markov Caché donné, puis de calculer la vraisemblance d'une séquence d'observations suivant un modèle de Markov Caché donné.

Le modèle de Markov est de type discret. Les classes de caractères (classes 0, 1, 7) sont modélisées chacune par un modèle à Q=5 états de type gauche-droite. Les états 1 et 5 correspondent à des colonnes de pixels de type fond de l'image (niveau 0). Les états 2, 3 et 4 correspondent au début, milieu et fin du caractère respectivement. Les transitions entre états sont indiquées dans la matrice de transitions A de taille QxQ. Les vecteurs π sont tous égaux à π=(1 0 0 0 0). Les séquences d'états commencent donc toujours par l'état q_1=1.

Les séquences d'observations sont discrètes et issues d'images de chiffres de la base MNIST. Les séquences d'observations consistent en séquences d'index (symboles) des éléments du dictionnaire. Ce dictionnaire est stocké sous forme matricielle (matrice v) dans le fichier matrice_symboles. L'élément numéro i d'une séquence d'observations correspond au symbole i et donc à la colonne i de la matrice v. Un symbole correspond à une configuration de colonne de 5 pixels (binaires : noir/blanc). Il y a 2^5=32 configurations, et donc symboles possibles.
Une séquence d'observations correspondant à l'image simplifiée de la fig. 1 est :
[1 1 1 1 1 1 14 23 23 27 18 18 18 12 12 12 12 12 12 12 23 23 23 14 4 1 1 1]
La concaténation des éléments du dictionnaire correspondant aux index de la séquence d'observations peut être visualisée sous forme d'image en remplaçant chaque index par le vecteur de pixels correspondant dans le dictionnaire (Fig. 2).

Les probabilités des observations dans chaque état sont indiquées dans la matrice B (32 lignes, 5 colonnes).

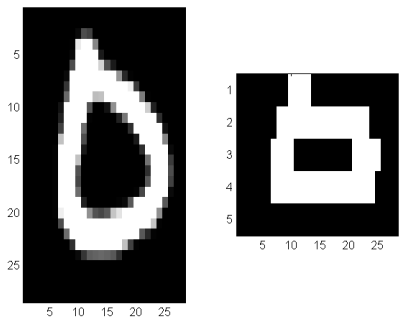

Fig. 1 : (gauche) image de chiffre (base MNIST, taille 28x28) (droite) image simplifiée (taille 5x28).

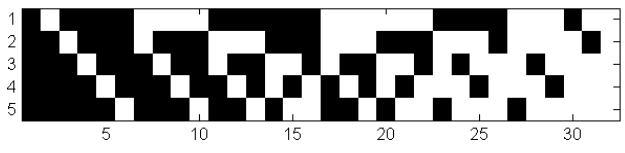

Fig. 2 : Chaque colonne est associée à un symbole du dictionnaire (32 symboles en tout).

II. 2. Génération de séquences d'observations

Les matrices A0.txt, B0.txt, vect_p0.txt contiennent les matrices A, B, et le vecteur π correspondant au modèle de Markov caché du chiffre 0. Le fichier matrice_symboles.txt contient le dictionnaire. Charger ces fichiers depuis le site pédagogique.

Le dictionnaire de symboles se trouve dans la matrice *v* qui se charge en utilisant :

```
import numpy as np
filename='matrice_symboles.txt'
v=np.loadtxt(filename)
```

II. 2.1 A quoi correspondent les zéros de la matrice B ? et ceux de la matrice A et du vecteur π ?

II.2.2 Ecrire une fonction *etat_suivant* qui génère un état q_{t+1}(à t+1) à partir de l'état courant q_t (à t) à l'aide de la matrice de transitions et de la fonction de répartition *cumsum*.
Afficher la fonction de répartition pour une ligne de la matrice de transition et expliquer son rôle pour la génération de l'état à t+1.

II.2.3 Générer une séquence d'observations suivant le modèle de Markov Caché du chiffre 0. On commencera par générer une séquence d'états suivant ce modèle à l'aide de la fonction *etat_suivant*. Puis on générera la séquence d'observations par le même procédé.

II.2.4 Visualiser le résultat sous forme d'image. Générer des séquences pour le chiffre 7 et le chiffre 1 (matrices B1.txt, B7.txt, etc…)

```
import matplotlib.pyplot as plt
im=[]
# les x contiennent les index dans le dictionnaire
for t in range (0,len(stateSeq)) :
        im_col=v[:, x[t]-1]
        im.append(im_col)

im = np.array(im).T  # now make an array
plt.imshow(im*255, cmap='Greys', interpolation='none', aspect='auto')
```

II.3. Calcul de la vraisemblance de séquences d'observations

Les fichiers SeqTest0.txt, SeqTest1.txt, SeqTest7.txt contiennent chacun 10 séquences d'observations de chiffres des 3 classes 0, 1 et 7, disposés en ligne. Le script suivant extrait la 5ème observation de la 3ème séquence des chiffres 0.

```
filename='SeqTest0.txt'
TestChiffres=np.loadtxt(filename)
```

```
nex=2
seq= TestChiffres[nex, : ]
seq[4]
```

II.3.1 Calculer la vraisemblance de ces séquences suivant chacun des modèles (0, 1 et 7) par l'algorithme de Viterbi (on pourra implémenter la version logarithmique de cet algorithme). Pour cela les matrices A, B et π seront converties en logarithmes (utiliser np.log).

II.3.2 Donner le résultat de la classification des images de test en considérant un problème à trois classes : 0, 1 et 7.