HES-SO
Engineering and Architecture
Faculty
MSE

Enseignant: Stefano Carrino
Assistant: Jérôme Moret

Multimodal Processing, Recognition and Interaction

Série 1 du 22 Septembre - 05 Octobre 2020
Reconnaissance de la parole – HMMs

Pour s'échauffer

a) On dispose de trois pièces de monnaies. L'une est normale et les deux autres sont truquées. Pour un observateur est impossible de savoir quelle pièce a été lancée (normale ou trouquée), on peut juste observer si la sortie a été pile ou face. L'expérimentateur change de pièces de façon équiprobable. Le tableau suivant résume les probabilités d'émission de chaque pièce :

	Pièce 1	Pièce 2	Pièce 3
P(face)	0.5	0.75	0.25
P(pile)	0.5	0.25	0.75

b) Donner la définition d'état et d'observation pour ce problème.

c) Dessiner le graphe représentant le modèle de Markov.

d) Si maintenant l'experimentateur a le 50% de probabilité de lancer encore une fois la même pièce qui vient de lancer, comment le graphe précedent changera ?

Contexte

Nous allons illustrer les cours sur la classification de séries temporelles avec les HMM par un petit exemple de construction de modèles pour les chiffres de 1 à 5 (avec trois exemples d'entraînement par chiffre). Pour ce faire, nous vous donnons un notebook Jupyter que vous allez utiliser et compléter pour entraîner et tester les modèles. Dans ce script, les HMMs ont été implémentés de la façon suivante (les notations correspondent aux noms des variables) :

— Le nombre d'états N est défini comme le nombre de phonèmes dans chaque chiffre plus les deux états de silence plus les deux états non-émetteurs de début et de fin (à déterminer).

— Les probabilités d'émission sont calculées par des fonctions de densité de probabilités modélisées par une Gaussienne de dimension P :

$$b_j[\mathbf{o}(t)] = \mathcal{N}(\mathbf{o}(t); \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \frac{1}{\sqrt{(2\pi)^P |\boldsymbol{\Sigma}_j|}} e^{-\frac{1}{2}(\mathbf{o}(t)-\boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{o}(t)-\boldsymbol{\mu}_j)}, \tag{1}$$

où $\boldsymbol{\mu}_j$ représente la moyenne, $\boldsymbol{\Sigma}_j$ est la matrice de covariance (ici forcée à être diagonale pour simplifier les calculs) et $|\boldsymbol{\Sigma}_j|$ dénote son déterminant.
Un modèle HMM est défini par trois matrices :

a) **A** (N x N) la matrice des transitions entre les états.

b) **MI** (P x N) la matrice des moyennes de tous les états pour les probabilités d'émission. La première et la dernière colonne de cette matrice sont vides vu que le premier et le dernier états ne sont pas émetteurs.

c) **SIGMA** (P x N) la matrice des écarts types (standard deviation) de tous les états pour les probabilités d'emission.

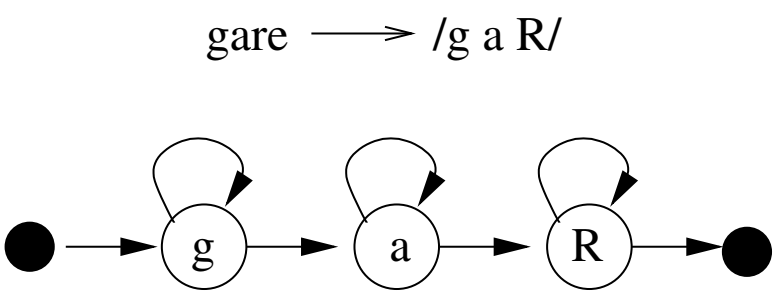

FIGURE 1 – Exemple d'un HMM pour le mot *gare*. Le mot est traduit en sa séquence de phonèmes et chaque phonème est associé à un état du HMM. Le premier et dernier état du HMM (ronds en noir) sont des états "non-émetteurs", cad qu'aucune probabilité d'émission ne leur est associée. La séquence de phonèmes est précédée et suivie par un état associé au silence.

Etapes

a) **Préparation des données :**
Vous devez enregistrer les données qui vont servir à entraîner et à tester les HMMs. Pour

ce faire, utilisez un outil de votre choix qui vous permet d'enregistrer des fichiers audio (format Microsoft wav, 16kHz, 16 bits). Vous avez besoin de 4 fichiers audio par chiffre, 3 pour entraîner, le dernier pour tester. Veuillez nommer les fichiers audio *'1_1.wav' '1_2.wav' '1_3.wav' '2_1.wav' '2_2.wav'... '5_3.wav'* pour les fichiers d'entraînement et *'1t.wav' '2t.wav' ... '5t.wav'* pour les fichiers de test. Enregistrer les fichier audio dans un répertoire 'wave'. Vous devez ensuite déterminer le nombre d'états de chaque modèle HMM en effectuant les transcriptions phonètiques de chaque chiffre (complétez le tableau ci-après). Modifiez le script en fonction des différentes valeurs de N (n'oubliez pas d'inclure les états non-émetteurs et les états réservés pour le silence).

b) **Extraction des paramètres :**
La paramétrisation des fichiers *.wav est faite via la bibliothèque *librosa* qui est directement appelée par le script principal. Nous utiliserons pour l'exercice les coefficients MFCC (Mel Filtered Cepstral Coefficient). Vous devriez modifier le notebook Jupyter pour afficher la durée des fichiers d'entraînement (en millisecondes) ainsi que le nombre de vecteurs acoustiques qui en sont extraits (compléter le tableau ci-dessous avec les valeurs moyennes des 3 fichiers d'entraînement). L'écart entre deux fenêtres d'analyse est de 8 ms. Vérifier que le nombre de vecteurs acoustiques correspond bien à ce qui est attendu étant donné l'écart entre deux fenêtres d'analyse.

c) **Entraînement des modèles :**
Pour l'entraînement des modèles, la fonction *hmm.GaussianHMM()* est utilisée. Le nombre d'états, N, est passé en paramptre. La bibliothèque, effectue l'entraînement des HMMs à travers une réestimation des paramètres des modèles en utilisant trois fichiers pour chaque chiffre. Par défaut, l'entraînement s'effectue avec le critère de Viterbi : on cherche l'alignement optimal en fonction des anciennes valeurs des gaussiennes et on calcule des nouvelles valeurs des gaussiennes sur base des nouveaux alignements. Les "Inputs" du script sont : les trois séquences d'observations et les anciennes matrices **A**, **MI** et **SIGMA**. Les Outputs sont : les nouvelles matrices **A**, **MI** et **SIGMA**, ainsi que la probabilité moyenne P_{tot} obtenues lors de la réestimation.

Points à résoudre

TABLE 1 – Veuillez compléter les transcriptions phonètiques, le nombre d'états du HMM, la durée moyenne des 3 fichiers wav ainsi que le nombre moyen de vecteurs acoustiques pour les 3 fichiers.

Mot	Transcription phonètique	N	durée (ms)	n vect acoust
un				
deux				
trois				
quatre				
cinq				

a) Copier le fichier "T1_HMMs.zip" depuis le Moodle.

b) Enregistrer votre propre voix comme indiqué dans les sections précédentes.

c) Compléter le tableau ci-dessus et effectuer les modifications des scripts comme indiqué.

d) Entraîner les modèles des 5 chiffres (1-5). Utiliser le bon N pour chaque modèle.

e) Créer un monitor et observer l'évolution de la probabilité lors des itérations d'entraînement pour chaque chiffre. Que pouvez-vous conclure de cette évolution ?

f) Répéter les 2 points précedents, mais cette fois utiliser les bons paramètres dans la méthode GaussianHMM afin que :
   — Le modèle de Markov caché est du type strictement gauche droite avec $a_{ij} \neq 0$ pour $j = i$ et $j = i + 1$, comme illustré sur la figure relative au mot gare (ci-dessus).
   — Les densités de probabilités d'émission seront des gaussiennes avec des matrices de covariance diagonales.

g) Construisez un tableau 5×5 avec les valeurs de probabilités obtenues pour les observations de test étant donné les cinq modèles HMM. Quelles sont les modèles reconnus pour vos 5 fichiers de test ?

h) Enregistrer le mot "peu" et faire le test de reconnaissance. Quelles valeurs de probabilités obtenez-vous dans ce cas ? Quel est le modèle gagnant ?

i) Demander à un collègue de vous fournir ses enregistrements de test et faites la reconnaissance sur ces fichiers avec vos modèles entraînés. Comme au point précédent, construisez également un tableau 5×5 récapitulant les probabilités obtenues. Que pouvez-vous conclure de ces résultats ?

j) Expliquer comment créer un système de classification binaire biométrique capable de dire si OUI ou NON, un enregistrement à été crée par un utilisateur. Par exemple, le système devra vous dire si oui ou non un fichier contient la voix de votre collegue. Comme fichiers d'entrainement vous avez tous les fichiers crées dans les points précédents par toute la classe.

Que faut-il rendre ? Dans une archive zip :

— **un** document pdf contenant les réponses aux questions
— les fichiers sons *.wav
— le notebook Jupyter (ou le code python) modifié