Hes·so Master

MSE | MASTER OF SCIENCE IN ENGINEERING

Multimodal Processing, Recognition, and Interaction (MPRI)

Reconnaissance de la parole – HMMs

(Série 1)

Auteur :	Spinelli Isaia
Prof :	Stefano Carrino
Date :	26.09.2020
Salle :	A2 – Lausanne
Classe :	MPRI

Table des matières

Introduction

Le but de ce laboratoire est d'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'entrainement par chiffre).

Ce laboratoire est essentiellement consisté de quatre étapes :

1. Préparation des données
2. Extraction des paramètres
3. Entraînement des modèles
4. Test et analyses des résultats

Préparation des données

La première étape de ce laboratoire est de préparer des données afin d'entrainer et tester les HMMs. L'objectif est d'avoir 4 fichiers audio par chiffre, 3 pour entrainer et le dernier pour tester.

Les fichiers audio ont été enregistrés à l'aide de l'application « Audacity » avec tous une fréquence d'échantillonnage de 16kHz et 16 bits par échantillon.

Il faut déterminer pour chaque modèle HMM le nombre d'état en effectuant les transcriptions phonétiques de chaque chiffre. Il est important d'inclure les états non-émetteurs (Start et End) et les états réservés pour le silence. Dans mon cas, un silence est présent au début et à la fin de chaque enregistrement audio.

Voici le tableau qui contient la traduction phonétique ainsi que le nombre d'état :

Mot	Transcription phonétique	N
un	ɛ̃	5
deux	dø	6
trois	tʁwa	8
quatre	katʁ	8
cinq	sɛ̃k	7

Le paramètre « N » a été mis à jour dans le script python fourni afin de correspondre à ce tableau. Afin de comprendre le raisonnement du nombre d'état, voici un exemple :

un = ɛ̃ => 1 état + 2 (start et end) + 2 (silence avant et après) = 5

Extraction des paramètres

La deuxième étape consiste à extraire les caractéristiques des fichiers *.wav. Pour ce faire, la bibliothèque Librosa est utilisé et les coefficients MFCC [1](Mel Filtered Cepstral Coefficient) sont utilisés.

Pour ce faire, il nous est demandé d'afficher la durée des fichiers d'entrainement (en millisecondes) ainsi que le nombre de vecteurs acoustiques qui en sont extraits. Les moyennes de ces paramètres pour chaque chiffre sont affichées dans le tableau ci-dessous :

Mot	Durée (ms)	N vect acoust
un	2 240	280
deux	1 965	246
trois	2 025	253
quatre	2 265	283
cinq	2 225	278

Entraînement des modèles

Pour l'entrainement des modèles, la fonction « hmm.GaussianHMM() » est utilisée. Le nombre d´états, N, est passé en paramètre. La bibliothèque, effectue l'entrainement des HMMs à travers une réestimation des paramètres des modèles en utilisant trois fichiers pour chaque chiffre.

Topologie ergodique

Voici le code permettant l'entrainement 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 :

```
N = 5
X, lengths = concatenate_cepstrums(train_1_c)
model_1 = hmm.GaussianHMM(n_components=N, verbose=True)
print("model_1 : ")

model_1.fit(X, lengths)
```

Grâce au paramètre « verbose=True », un monitor est créé et permet observer l'évolution de la probabilité lors des itérations d'entraînement pour chaque chiffre. Voici un exemple :

[1] MFCC : http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

```
model_1 :
         1      -12535.1360             +nan
         2       -6998.1542       +5536.9819
         3       -6724.7169        +273.4372
         4       -6688.6685         +36.0484
         5       -6672.9141         +15.7545
         6       -6665.2943          +7.6198
         7       -6658.7596          +6.5346
         8       -6651.5435          +7.2161
         9       -6644.8495          +6.6940
```

On peut constater qu'au fur est à mesurer que les itérations s'effectuent, la probabilité diminue de manière inversement exponentielle. En effet, les premières itérations diminuent grandement la probabilité mais après la deuxième itération, la probabilité ne diminue plus beaucoup. Ceci est constaté pour les modèles de tous chiffres.

Topologies Left-Right (LR)

L'objectif est d'entrainer les modèles comme précédemment mais avec la topologies Left-Rifht. Pour ce faire, quelques paramètres devront être ajoutés :

- Les probabilité d'état initial (état start à 1).
- Les probabilités de transition d'état (gauche-droite uniquement).
- Les densités de probabilités d'émission seront des gaussiennes avec des matrices de covariance diagonales.

Voici le nouveau code pour l'entraînement des modèles :

```
N = 5
X, lengths = concatenate_cepstrums(train_1_c)
model_1_lr = hmm.GaussianHMM(n_components=N, covariance_type="diag", init_params="cm", params="cmt", verbose=True)
print("model_1_lr : ")

model_1_lr.startprob_ = np.array(   [1.0, 0.0, 0.0, 0.0, 0.0])
model_1_lr.transmat_ = np.array([   [0.5, 0.5, 0.0, 0.0, 0.0],
                                    [0.0, 0.5, 0.5, 0.0, 0.0],
                                    [0.0, 0.0, 0.5, 0.5, 0.0],
                                    [0.0, 0.0, 0.0, 0.5, 0.5],
                                    [0.0, 0.0, 0.0, 0.0, 1.0]])
model_1_lr.fit(X, lengths)
```

Le paramètre « init_params="cm" » permet d'indiquer quels paramètres sont à initialiser avant l'entraînement.

Le paramètre « params="cmt" » permet d'indiquer quels paramètres sont mis à jour dans le processus d'entraînement.

Les paramètres possibles :

- «s» pour startprob
- «t» pour transmat
- «m» pour means
- «c» pour covars.

Comme on peut le voir, un tableau contenant les probabilités de départ « startprob » est indiqué afin de commencer par l'état « Start ». Ainsi qu'une matrix contenant les probabilités de transition « transmat » afin d'aller uniquement de gauche à droite.

Voici l'évolution de la probabilité lors des itérations d'entrainement :

```
model_1_lr :
         1      -14258.6836            +nan
         2       -9534.1291       +4724.5545
         3       -9385.7367        +148.3925
         4       -9354.0453         +31.6913
         5       -9337.8716         +16.1738
         6       -9324.2252         +13.6463
         7       -9314.7995          +9.4257
         8       -9309.9792          +4.8204
         9       -9305.2203          +4.7588
        10       -9301.5394          +3.6809
```

On peut constater la même type d'évolution que pour les modèles ergodique. Cependant, il arrive parfois pour un modèle une évolution étrange :

```
         1      -15223.3648            +nan
         2      -13937.5315       +1285.8333
         3      -13869.3081         +68.2234
         4      -13863.0934          +6.2147
         5      -13599.2154        +263.8780
         6      -11481.1968       +2118.0186
         7       -9236.7914       +2244.4054
         8       -9077.5329        +159.2584
         9       -9076.6930          +0.8399
        10       -9076.6323          +0.0608
```

On peut voir ici que l'évolution de la probabilité varie un peu comme un signal sinusoïdal. Toutefois, dans tous les cas, la probabilité diminue.

Test et analyses des résultats

Dans ce chapitre, nous allons tester et analyser les modèles précédemment présenté. Pour ce faire, nous allons utiliser des observations de tests personnels et venant d'un collègue (*Sampiemon Thibault*).

Modèle ergodique / observations personnelles

Voici un tableau qui résume les valeurs de probabilités obtenues pour les observations de test personnelles étant donné les cinq modèles HMM en mode ergodique :

Mot testé	Modèle 1	Modèle 2	Modèle 3	Modèle 4	Modèle 5
1	**-2475.14**	-5148.17	-4149.59	-5101.39	-6537.38
2	-3822.51	**-1590.95**	-4626.29	-4255.37	-5781.09
3	-3421.94	-5662.76	**-2115.7**	-5352.25	-6679.67
4	-4856.2	-5189.69	-5393.22	**-2896.6**	-5301.96
5	-4454.77	-5794.03	-5571.79	-6273.31	**-2515.06**
peu	-4372.1	**-3345.09**	-5745.83	-4949.06	-4557.8

En gras, on peut voir les probabilités les plus grandes pour chaque mot testé. Donc, le modèle reconnus.

Ici, pour tous les mots, tous les modèles reconnus correspondent bien.

Pour le mot « peu », comme attendu, le modèle 2 est reconnu. Cependant, la probabilité est plus petite que pour le réel mot « deux ».

Modèle LR / observations personnelles

Voici un tableau qui résume les valeurs de probabilités obtenues pour les observations de test personnelles étant donné les cinq modèles HMM en mode LR:

Mot testé	Modèle 1	Modèle 2	Modèle 3	Modèle 4	Modèle 5
1	**-3227.08**	-5025.97	-4026.96	-4879.53	-5909.71
2	-4081.25	**-2348.39**	-4026	-4346.16	-4495.92
3	-4653.16	-5260.15	**-3388.93**	-4945.11	-5302.33
4	-6434.26	-5738.22	-5430.56	**-3548.24**	-5499.64
5	-5438.55	-5551.22	-4862.96	-4798.87	**-2918.24**
peu	-5006.05	**-3737.97**	-5926.13	-4205.26	-5962.95

Exactement comme pour les modèles ergodique, les modèles LR ont été reconnus correctement par tous les mots. Le mot « peu » a encore été reconnu comme le modèle 2 aussi avec une probabilité plus faible.

De manière général, les probabilités sont plus petites pour les modèles LR que les modèles ergodique. Ce qui veut dire que dans ce cas, le modèle ergodique fonctionne mieux. Cela est surement dû au fait que nous n'avons pas beaucoup de donnée à disposition.

Modèle ergodique / observations autres

Voici un tableau qui résume les valeurs de probabilités obtenues pour les observations de test effectué par un collègue pour les cinq modèles HMM en mode ergodique :

Mot testé	Modèle 1	Modèle 2	Modèle 3	Modèle 4	Modèle 5
1	**-4745.18**	-6375.65	-7292.62	-5563.04	-7841.74
2	-7046.24	**-6207.22**	-8773.01	-7749.08	-11626.9
3	**-2453.94**	-3891.52	-3295.27	-4081.05	-4954.3
4	-4022.95	-4778.74	-5138.68	**-3626.47**	-6349.58
5	**-4356.94**	-4433.14	-5998.07	-4849.57	-5664.94

On peut voir que les mots « un », « deux » et « quatre » ont bien reconnu leur modèle respectif. Cependant, les mots « trois » et « cinq » ont reconnu le modèle 1.

En relançant plusieurs fois le programme, les modèles reconnus étaient toujours identiques. De ce fait, les résultats sont plutôt stables.

Modèle LR / observations autres

Voici un tableau qui résume les valeurs de probabilités obtenues pour les observations de test effectué par un collègue pour les cinq modèles HMM en mode LR :

Mot testé	Modèle 1	Modèle 2	Modèle 3	Modèle 4	Modèle 5
1	-5943.56	-5435.45	-5315.68	-6541.49	**-4330.46**
2	-9832.28	**-5895.3**	-9323.6	-8515.06	-6197.12
3	-3243.21	-3552.97	-3577.07	-4721.67	**-3157**
4	-5082.07	**-4121.64**	-5604.74	-4497.29	-4368.12
5	-5853.49	**-4180.33**	-6009.54	-6030.04	-4240.81

Dans ce cas, uniquement le mot « deux » a correctement reconnu son modèle. Encore une fois, on peut constater que les modèles LR offre de moins bon résultat.

De plus, en relançant plusieurs fois le programme, les modèles reconnus variaient beaucoup. De ce fait, les résultats ne sont pas très stables.

Classification binaire biométrique

Il nous est demandé d'expliquer comment créer un système de classification binaire biométrique capable de dire si OUI ou NON, un enregistrement à été créé par un utilisateur.

Dans le cas demandé, il existe que deux modèles :

1. L'enregistrement audio a été créé par l'utilisateur (Correct)
2. L'enregistrement audio n'a pas été créé par l'utilisateur (Faux)

Il faudrait entraîner le premier modèle avec tous les enregistrement personnel et le deuxième modèle avec tous les autres fichiers. Ceci est une manière simple de faire un système de classification binaire.

Sinon il est possible d'envisager une autre méthode comprenant deux sortes de modèles HMM créés pour chaque chiffre.

Le principe est le même que la première méthode mais au lieu de mélanger tous les chiffres, on crée les deux modèles (Correct / Faux) pour chaque chiffre. Voici par exemple comment il faudrait entrainer les modèles :

Model_1_correcte <= /1_1.wav + /1_2.wav + /1_3.wav *(Enregistrement de l'utilisateur)*

Model_1_faux <= /colleg1/1_1.wav + /colleg1/1_2.wav + /colleg1/1_3.wav +
/colleg2/1_1.wav + /colleg2/1_2.wav + /colleg2/1_3.wav + …

Model_2_correcte <= /2_1.wav + /2_2.wav + /2_3.wav *(Enregistrement de l'utilisateur)*

Model_2_faux <= /colleg1/2_1.wav + /colleg1/2_2.wav + /colleg1/2_3.wav +
/colleg2/2_1.wav + /colleg2/2_2.wav + /colleg2/2_3.wav + …

…

Finalement, si la probabilité la plus haute correspond à un modèle_x (1 - 5) correct, l'enregistrement provient surement de l'utilisateur dédié. Dans le cas où la probabilité la plus haute correspond à un modèle_x (1 - 5) faux, l'enregistrement ne provient surement pas de l'utilisateur.

Conclusion

Difficultés rencontrées

- Prise en main et installation de l'environnement
- Prise en main du script python
- Compréhension des différentes fonctions utilisées

Compétences acquises

- Installation de l'environnement (IDE / Python / librairies)
- Familiarisation des librairies
- Comprendre et mettre en pratique les différentes topologies HMM

Résultats obtenus

Finalement, différents modèles ont pu être mis en place (Ergodique et Gauche-Droite) de manière pratique. Les différents fichiers de test enregistrés personnellement ont tous correctement reconnu leur modèle respectif. De plus, 60% (3/5) des fichiers de test de mon collègue ont reconnu correctement leur modèle ergodique de manière stable. Malheureusement, les modèles gauche-droite ont fourni des résultats moins généreux.

Date : 05.10.20

Nom de l'étudiant : Spinelli Isaia