TP2: Les modèles HMC pour la segmentation de texte

MA202 Modèles de Markov
27 avril 2020

Corentin Soubeiran

Chargé de TD: E. Azeraf
Professeur: W. Pieczynski

Question 1

Démontrons l'écriture du Forward et backward.

Forward

En adoptant les notations du TD nous avons:

$$\alpha_i(t) = \mathbb{P}(X_t = i, y_{1:t})$$

Ainsi:

$$\alpha_i(1) = \mathbb{P}(X_1 = i, y_{1:1})$$

$$= \mathbb{P}(X_1 = i, y_1) = \mathbb{P}(y_1|X_1 = i)\mathbb{P}(X_1 = i) = b_i(y_1)\pi_i$$

donc : $\boxed{\alpha_i(1) = b_i(y_1)\pi_i}$

De plus:

$$\alpha_i(t+1) = \mathbb{P}(X_{t+1} = i, y_{1:t+1})$$

Par les probas totales:

$\mathbb{P}(X_{t+1} = i, y_{1:t+1}) = \sum_{j=1:T}[\mathbb{P}(X_t = j, X_{t+1} = i, y_{1:t}, y_{t+1})]$

(1): $\alpha_i(t+1) = \sum_{j=1:T}[\mathbb{P}(X_t = j, y_{1:t})\mathbb{P}(X_{t+1} = i, y_{t+1}|X_t = j, y_{1:t})]$

On remarque alors que:

- ***(2)*** : $\mathbb{P}(X_t = j, y_{1:t}) = \alpha_j(t)$
- Grâce aux hypothèses de CMC on a:

$$\mathbb{P}(X_t = j, X_{t+1} = i, y_{1:t}, y_{t+1}) = \mathbb{P}(X_t = j)\mathbb{P}(X_{t+1} = i|X_t = j)\mathbb{P}(y_t|X_t = j)\mathbb{P}(y_{t+1}|X_{t+1} = i)$$

$$= \mathbb{P}(X_{t+1} = i, y_{t+1} | X_t = j, y_{1:t})] \mathbb{P}(X_t = j, y_{1:t})$$

or:

$$\mathbb{P}(X_t = j, y_{1:t}) = \mathbb{P}(X_t = j) \mathbb{P}(y_t | X_t = j)$$

d'où:

$$\mathbb{P}(X_{t+1} = i, y_{t+1} | X_t = j, y_{1:t})] = \mathbb{P}(X_{t+1} = i | X_t = j) \mathbb{P}(y_{t+1} | X_t = j)$$

- Avec la matrice de transition on a ***(3)***:

$$\mathbb{P}(X_{t+1} = i | X_t = j) = A'_{(i,j)} = A_{(j,i)} = a_{ji}$$

- Et la probabilité d'émission ***(4)***:

$$\mathbb{P}(y_{t+1} | X_t = j) = b_j(y_{t+1})$$

En réunissant les formules ***3*** et ***4*** dans ***1*** avec ***2***, on a enfin:

$$\boxed{\alpha_i(t+1) = \sum_{j=1:T} [\alpha_j(t) a_{ji} b_j(y_{t+1})] = b_j(y_{t+1}) \sum_{j=1:T} [\alpha_j(t) a_{ji}]}$$

Backward

On a:

$$\beta_i(t) = \mathbb{P}(y_{t+1:T} | X_t = i)$$

Pour l'initialisation:

$$\boxed{\beta_i(T) = \mathbb{P}(y_{T+1:T} | X_T = i) := 1}$$

Pour la récurence par les proba totales ***(5)***:

$$\beta_i(t) = \sum_{j=1:T} [\mathbb{P}(X_{t+1}, y_{t+1:T} | X_t = i)$$

On remarque que:

- $$\mathbb{P}(X_t = i, X_{t+1}, y_{t+1:T}) = \mathbb{P}(X_{t+1}, y_{t+1:T} | X_t = i) \mathbb{P}(X_t = i)$$
- $$\mathbb{P}(X_t = i, X_{t+1}, y_{t+1:T}) = \mathbb{P}(X_t = i) \mathbb{P}(X_{t+1} = j | X_t = i) \mathbb{P}(y_{t+1} | X_{t+1} = j) \mathbb{P}(y_{n+2:N} | x_{n+1})$$

on en déduit que ***(6)***:

$$\mathbb{P}(X_{t+1}, y_{t+1:T} | X_t = i) = \mathbb{P}(X_{t+1} = j | X_t = i) \mathbb{P}(y_{t+1} | X_{t+1} = j) \mathbb{P}(y_{n+2:N} | X_{n+1} = j)$$

On remarque également:

- $\mathbb{P}(X_{t+1} = j | X_t = i) = a_{ij}$ (cf. forward)
- $\mathbb{P}(y_{t+1} | X_{t+1} = j) = b_j(y_{t+1})$ (probabilité d'émission)

- $\mathbb{P}(y_{n+2:N}|X_{n+1}=j)=\beta_j(t+1)$

En injectant ces trois dernière remaques dans ***6*** puis ***5***:

$$\boxed{\mathbb{P}(X_{t+1},y_{t+1:T}|X_t=i)=\sum_{j=1:T}[a_{ij}b_j(y_{t+1})\beta_j(t+1)]}$$

Question 2

1. Estimation des paramètres

La loi à priori à priori π peut être aproximé dans un premier temps la fréquence, en tant qu'application du théorème de la loi des grand nombre avec T grand.
Ainsi:

$$\pi_i=\mathbb{P}(X_t=i)$$

Est appoximé par:

$$\hat{\pi}_i=\frac{|\{t|X_t=i\}|}{T}$$

En observant le code dans la fonction \textit{HMC_Learning_parameters.py} nous remarquons que c'est ce qui est fait, on considère tou les subset du dataset et on compte le nombre d'occurance d'une catégorie de notre ensemble Ω_X avant de diviser par le nombre total de données en considerant toute les catégorie. Soit donc le pseudo code suivant:

```
- Pour tout subset du dataset
    - élement Zi du subset de catégorie x:
        Ajouter 1 à pi de la catégorie x
- Calculer sum_pi comme la somme de tous les pi
  (nombre d'élément dans tous les dataset) (cardial)
- Pour tout x catégorie dans l'ensemble des catégories
    Diviser pi de la catégorie x par sum_pi
```

Qui revient bien à ce qui est fait dans le code python, dans ce code la manipulation de dictionaire implique tout de même la création dynamique des catégories ce qui correspond aux `not in`

La procédure pour l'estimation de A (matrice de transition) et B (probabilité d'émission) suit le même principe
pour A:

$$A_{ij}=\mathbb{P}(X_t=j|X_{t-1}=i)=\frac{\mathbb{P}(X_t=j,X_{t-1}=i)}{\mathbb{P}(X_{t-1}=i)}$$

$$\hat{A_{ij}}=\frac{|\{t|X_t=j\cap X_{t-1}=i\,t\in\{2:T\}\}|}{|\{t|X_{t-1}=i,t\in\{2:T\}\}|}=\frac{nombre\ de\ transition\ de\ la\ catégorie\ i\ \text{à}\ j}{nombre\ total\ de\ transitions\ depuis\ l'état\ i}$$

pour B:

$$b_i(y) = \mathbb{P}(Y_t = y | X_t = i) = \frac{\mathbb{P}(Y_t = y, X_t = i)}{\mathbb{P}(X_t = i)}$$

$$\hat{b_i(y)} = \frac{|\{t | Y_t = y \cap X_t = i \, t \in \{1:T\}\}|}{|\{t | X_{t-1} = i, t \in \{1:T\}\}|} = \frac{occurance\ de\ l'observation\ y\ correspondant\ \text{à}\ un\ \text{éta}}{nombre\ d'observations\ correspondant\ \text{à}\ un\ \text{état}\ i}$$

2. Interet des dictionnaires

Un dictionnaire est un objet python où les données sont associés à des clés contrairement au liste ou les données sont dans un ordre. Le premier avantage est il me semble algorithmique car les dictionaire sont codées avec une table de hachage, le temps de recherche d'un élément est en complexité $O(1)$ contrairement au listes qui si elles ne sont pas triés sont en $O(n)$ ou au mieux en $O(log(n))$ aprés un tri.
Dans notre cas un autre avantage est la compréhension et la facilité d'acces dans cet objet. En effet on travail ici avec des mots, si nous utilisions des listes il faudrait définir un index faisant le lien entre un mot et un indice, indice que l'on utilise ensuite pour mettre les données au bon endroit dans la liste. Ce qui n'est pas optimal car la recherche dans la liste pour l'ajout d'un index sera en $O(n_1)$ et la sauvegarde en $O(n_2)$ alors que dans un dictionaire l'absence de présence est en $O(1)$ et l'ajout d'un élement au pire en $O(n_2). Ainsi l'acces d'une donnée à un mot grace à ce mot directement dans le dictionnaire est tout de même bien plus simple non seulement d'un points de vu algorithmique que d'un point de vu de compréhension.

3. Estimation par maximum de vraissemblance (EMV)

Pour π_i

On a :

$$\pi_i = \mathbb{P}(X_t = i)$$

C'est la probabilité qu'un élément de $X = (X_1, \dots X_T$ soit dans la catégorie i. On peut donc imaginer qu'un élement X_t de X suit une loi binomiale de paramètres π_i d'être dans la catégorie i.
On a alors:

$$X_t \sim B(\pi_i)$$

On en déduit la vraissemblance:

$$L_T(X_{1:T}, \pi_i) = \prod_{t=1}^{T} \mathbb{P}_{\pi_i}(X_t = i) = \pi_i^s (1 - \pi_i)^{T-s}$$

Où s représente ici le nombre de "succes" c'est à dire de X_t de classe i, Ainsi l'estimateur du maximum de vraissemblances $\hat{\pi_i}$ est trouvé par:

$$\frac{\delta L_T}{\delta \hat{\pi_i}} = \hat{\pi_i}^{s-1} (1 - \hat{\pi_i})^{T-s-1} (s - T\hat{\pi_i}) = 0 \iff \hat{\pi_i} = \frac{s}{T}$$

Car (en passant par le log pour me simplifier les calculs):

$$\frac{\delta^2 L_T}{\delta^2 \hat{\pi_i}} = -s\frac{1}{\hat{pi}_i} - (T-s)\frac{1}{1-\hat{pi}_i} = -2T < 0$$

Conclusion l'estimateur du maximum de vraissemblance:

$$\boxed{\hat{\pi_i} = \frac{s}{T}}$$

C'est bien l'estimateur utilisé dans le code d'après ce que nous avons décrit au 2.1

Pour A_{ij}

On a:

$$A_{ij} = \mathbb{P}(X_t = j | X_{t-1} = i)$$

Il s'agit en fait de la probabilité de transition de l'état i à j en une étape. Comme précedement on peut modéliser la situation comme $Z_k(i)$ la variable aléatoire valant 1 si $X_t = j$ et 0 sinon dans $X_{t-1} = i$, loi de $X_t = j$ conditionnellement à $X_{t-1} = i$. Ici k à valeur entre 0 et le nombre de variable de X de classe i que nous notons n_i.
On a:

$$Z_k(i) \sim B(a_{ij})$$

En reprennant le calcul précédant on a:

- s que nous notons maintenant $n_{i \to j}$ le nombre de varaible X_t de catégorie j tel que $X_{t-1} = i$
- T que nous notons maintenant n_i le nombre de X_t valant i

Ainsi le résonnement précédant s'applique aussi ici et:

$$\boxed{\hat{a_{ij}} = \frac{n_{i \to j}}{n_i}}$$

C'est bien l'estimateur utilisé dans le code d'après ce que nous avons décrit au 2.1

Pour $b_i(y)$

On a:

$$b_i(y) = \mathbb{P}(Y_t = y | X_t = i)$$

Il s'agit de la probabilité qu'un observation prenne la valeur y sachant que la catégorie qui lui est associé vaut i. Ici y est un mot de l'espace Ω_y fini (l'ensemble des mots d'une langue est un espace fini). En notant $W_m(i)$ la variable aléatoire valant 1 si $Y_t = y$ et 0 sinon dans $X_t = i$, loi

de $Y_t = y$ conditionnellement à $X_t = i$. Ici m à valeur entre 0 et le nombre de mot de catégorie i noté n_i.
On a:

$$W_m(i) \sim B(b_i(y))$$

En reprennant le calcul précédant on a:

- s que nous notons maintenant n_y le nombre de varaible Y_t valant le mot y de catégorie $X_t = i$
- T que nous notons maintenant n_i le nombre de X_t valant i

Ainsi le résonnement précédant s'applique aussi ici et:

$$\boxed{\hat{b_i(y)} = \frac{n_y}{n_i}}$$

C'est bien l'estimateur utilisé dans le code d'après ce que nous avons décrit au 2.1

Question 3

Algorithmique

Question 4

Chunking

Le Chunking permet la décomposition syntaxique.

Chunking	**Dataset 2000**	**Dataset 2003**
Accuracy	7.19%	5.62%
KW	6.68%	5.21%
UW	14.05%	8.93%
Time	0.17s	0.34s

POS

Le Part Of Speech consiste à labeliser les mots par leur fonction gamaticale

POS	**Dataset 2000**	**Dataset 2003**	**Dataset UD English**
Accuracy	5.36%	9.91%	11.16%
KW	1.94%	3.98%	6.22%
UW	50.94%	57.09%	65.27%
Time	0.18s	0.21s*	0.27s

*: le code donné ne calculait pas ce temps, je me suis permis de rajouter les lignes necessaire dans *hmc_pos_conll2003.py*.

NER

La Named entity recognition correspond à la reconaissance des noms propres

NER dataset 2003	**Précision**	**Recall**	**F1**
Résultat	0.62	0.86	0.72
KW	0.87	0.87	0.87
UW	0.006	0.52	0.01

Remarque sur le F1 score

Le F1 score est définit comme:

$$F1 = \frac{2}{recall^{-1} + precision^{-1}}$$

où:

$$recall = \frac{\text{vrai positif}}{\text{faux négatif} + \text{vrai positif}} \text{ (sensibilité ou puissance en français)}$$

$$precision = \frac{\text{vrai positif}}{\text{positif}}$$

Le recall *(true posotive rate)* permet de représenter la proportion d'item positif corectement classé comme positif.
La précision *(positive predictive value)* représente la proportion d'item classé positif à raison (parmis les positifs).

Le score *F1* permet donc de quantifier la qualité d'un test, sa maximisation permet d'avoir un compromis entre sa capacité à bien identifier un cas positif (précision) tout en restant sensible à un cas (recall).

En effet la maximisation du *recall* ou de la *precision* seul n'est pas viable, pour avoir un *recall=1* il suffit de classer tous les cas comme positif on a alors aucun faux négatif, mais notre test ne sers à rien puisque par exemple dans un contexte comme celui actuel il correspond à dire que tous le monde est malade. A l'inverse avoir *precision=1* correspond à tendre vers le cas limite ou le nombre de faux positifs est nul et donc dans notre cas à ne considerer personne comme malade. On voit que le compromis entre les deux est indispensable.

Ce qui est appellé *accuracy* pour le POS et le Chunking correspond au taux d'erreur de classification (à plusieurs niveau). De plus précisons que le un vrai positif pour le score *F1* correspond à classer un terme à raison dans la catégorie *i* autre que *O*. On ne peut pas comparer le score *F1*, le *recall* ou la *precision* à l'*accuracy* puisque:

$$accuracy = \frac{total - (\text{Vrai Positif} + \text{Vrai négatif})}{total} = \frac{\text{faux positif} + \text{faux négatifs}}{total}$$

$$(\text{où } total = VP + VF + FP + FN)$$

Conclusion

Nous remarquons déjà que pour le Chunking comme pour le POS l'erreur de classification est plus importante sur les mots inconnus que connus ce qui parait logique puisque il n'y a pas eu d'apprentissage sur les mots inconnus. Cette erreur est jusqu'à 25 fois plus importante sur les mots inconnus dans le POS et 2.5 fois plus importante pour le Chunking. On remarque ainsi que le **POS est bien plus intolérant aux mots inconnus** que le Chunking.
Néamoins on remarque que l'erreur globale est inférieur à 10%, une phrase est en moyenne composé de 10 mots, on peut donc considerer que l'on commet une erreur sur la fonction gramaticale.

Pour ce qui est de la reconaissance des noms propres (NER) on remarque que la sensibilité pour les mots inconnus est de 0.5, cela signifie que 1 mot sur deux considéré comme un nom propre en est en fait pas un. De plus la sensibilité de l'algorithme (*recall*) est quasi nulle, c'est la double peine, en plus de se tromper une fois sur deux dans la consideration d'un nom propre, un recall faible correspond à un faible nombre de vrai positif: et donc peut de mots sont classés positifs Autrement dit quand le nom propre n'a pas été appris, il n'a quasiment aucune chance d'être reconnu en tant que tel ou dans la bonne catégorie.

Question 5

Dans les fonctions de la question 3, la normalisatio de α et β s'appelle de *rescaling* il a pour but d'assurer la convergence algorithmique. En effet α et β peuvent atteindre des valeurs très faible qui par l'ordinateurs peuvant alors être considéré comme 0, là ou mathématiquement ils ne valent pas 0. Cela entraine alors des divisions par 0 ou des arrondis à 0 non désirables.

Le *rescaling* permet donc d'éviter de se retrouver dans cette situation d'autans plus qu'il ne modifie par le résultat de l'algothme de *Forward / Backward*. En effet la methode MPM consite à selectionner la catégorie i qui maximise la probabilité:

$$(R): \quad \mathbb{P}(X_t = i | y_{1:T}) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j \in \Omega_X} \alpha_j(t)\beta_j(t)}$$

Considérons alors:

$$\tilde{\alpha}_i(t) = \frac{\alpha_i(t)}{\sum_{k \in \Omega_X} \alpha_k(t)}$$

$$\tilde{\beta}_i(t) = \frac{\beta_i(t)}{\sum_{k \in \Omega_X} \beta_k(t)}$$

Notons:

$$\Sigma_\alpha := \sum_{k \in \Omega_X} \alpha_k(t)$$

$$\Sigma_\beta := \sum_{k \in \Omega_X} \beta_k(t)$$

Qui ne dépendent pas de la catégorie i.

Le rapport $(\tilde{R})$ s'écrit:

$$\frac{\tilde{\alpha}_i(t)\tilde{\beta}_i(t)}{\sum_{j \in \Omega_X} \tilde{\alpha}_j(t)\tilde{\beta}_j(t)}$$

$$= \frac{\alpha_i(t)\beta_i(t)}{\Sigma_\alpha \Sigma_\beta \sum_{j \in \Omega_X} \frac{\alpha_j(t)}{\Sigma_\alpha} \frac{\beta_j(t)}{\Sigma_\beta}} = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j \in \Omega_X} \alpha_j(t)\beta_j(t)}$$

$$= \mathbb{P}(X_t = i | y_{1:T})$$

La normalisation des coefficients α et β n'a donc aucun impact sur maximisation et donc sur le résulat de l'algorithme.

Question 6

Dans les algorithme on voit apparaitre un paramètre $epsilon\_laplace$ (defaut) dans la normalisation.
Remarquons tout d'abord que cette modification ne modifie par la notion d'ordre.

Soit: $f(x) = \frac{x+\epsilon}{\Sigma' + x + \epsilon}$ qui modélise la fonction de normalisation où x correspond à $\alpha_i(t)$ ou $\beta_i(t)$ et $\Sigma' = \sum_{k \in \Omega_X} \alpha_k(t) - \alpha_i(t)$ ou $\Sigma' = \sum_{k \in \Omega_X} \beta_k(t) - \beta_i(t)$. On a donc $x \in \mathbb{R}^+$ et $\Sigma' \in \mathbb{R}^+$.

Soit $h \in \mathbb{R}^+$, on suppose $\epsilon \in \mathbb{R}^+$ on a:

$$f(x) - f(x+h) = \frac{x+\epsilon}{\Sigma' + x + \epsilon} - \frac{x+h+\epsilon}{\Sigma' + x + h + \epsilon} < \frac{x+\epsilon}{\Sigma' + x + \epsilon} - \frac{x+h+\epsilon}{\Sigma' + x + \epsilon}$$

$$< -\frac{h}{\Sigma' + x + \epsilon} < 0$$

Ainsi:

$$\forall h \in \mathbb{R}^+ : f(x) < f(x+h)$$

D'où:

$$\forall (x_1, x_2) \in \mathbb{R}, x_1 < x_2 \Longrightarrow f(x_1) < f(x_2)$$

On en déduit donc qu'une telle modification ne change pas l'ordre et donc ne change en rien le résultat de la maximisation de l'algorithme. Par contre les valeurs du rapport sont alors "amplifiés", et le rapport n'est plus sensible à un division par 0 lorque $\epsilon > 0$.

L'objectif est donc à mes yeux que lorsque les $\alpha_i(t)$ sont très petits, leur somme $\Sigma_\alpha(t)$ l'est aussi. **Par conséquent la normalisation peu entrainer une division par zéro qui est empéchée par l'ajout de ce coefficient**.

Question 7

L'objectif est d'améliorer la classification des mots inconnus, pour cela on fonctionne de la manière suivante par exemple pour la methode forward de α_1:

```
- Si le mot est connu dans l'ensemble d'apprentissage:
    alpha_1[idi]=Pi_hmc[i]*list_B_hmc[0][i][list_Y[0][0]]
    on utilise la fomule classique comme dans la question 4
- Si le mot est inconnu de cet ensemble:
  - Si sa terminaison (suffixe) est connue
      alpha_1[idi]=Pi_hmc[i]*list_B_hmc[1][i][list_Y[1][0]
      On adapte en se plaçant sur le second ensemble d'apprentissage des terminaisons
  - Sinon:
      on fait l'approximation alpha_1[idi]=Pi_hmc[i]
```

Chunking

Le Chunking permet la décomposition syntaxique.

Chunking	**Dataset 2000**	**Dataset 2003**
Accuracy	7.09%	5.5%
KW	6.68%	5.21%
UW	12.54%	7.83%
Time	0.39s*	1.60s

POS

Le Part Of Speech consiste à labeliser les mots par leur fonction gamaticale

POS	Dataset 2000	Dataset 2003	Dataset UD English
Accuracy	3.43%	6.25%	9.00%
KW	1.94%	4.04%	6.14%
UW	23.35%	23.83%	40.30%
Time	0.83s	1.03s	1.06s

NER

La Named entity recognition correspond à la reconaissance des noms propres

NER dataset 2003	Précision	Recall	F1
Résultat	0.75	0.80	0.78
KW	0.88	0.87	0.87
UW	0.44	0.60	0.51

training time: $1.00s$

Conclusion

Pour le Chunking l'amélioration est faible, c'était à mes yeux prévisible puisqu'il s'agit de la reconnaissance des mots par syntaxe. Or la langue anglaise a peu de suffixes pour les composantss liés aux verbe ou aux noms. Je ne vois personnellement que la marque du possesif "s". Ce qui expliquerait cette faible amélioration. Je pense que si l'on travaillait sur des texte en Français l'amélioration serait notable ici avec tout les "ant", "ement", ... de la langue.

Par contre le POS bénéficie d'une bonne amélioration en tant qu'acteur de la détermination des Noms et des verbes, les noms sont souvent en début de phrase et prennent donc une majuscule même quand ce n'est pas des noms propres, pour les verbes on peut reconnaitre la terminaison "ing". Cette amélioration à multiplié par 2 les performances sur les mots inconnus des dataset 2000 et 2003.

Pour le NER la statégie des majuscules a bien fonctionné, la précision à considérablement augmenté, beaucoup plus de mots sont considérés, et en plus le recall à également augmenté, les positifs sont plus souvent (en terme de probabilité) marqué à juste titre comme nom propre dans la bonne catégorie).

Question 8

Bonus non réalisé