4.4 Algorithme de Baum-Welch

4.4.1 Introduction

Étant donné un modèle probabiliste M_θ et une observation w, on appelle vraisemblance de w dans le modèle M_θ la probabilité $P(w|M_\theta)$.

Considérons par exemple une pièce de monnaie vérifiant $P(Pile) = \theta$ et $P(Face) = 1-\theta$, une expérience consistant à jeter 10 fois la pièce et à noter la face apparente. Soit $w = PPFPFFPPFP$. La vraisemblance de w dans le modèle est $L(w;\theta) = \theta^6(1-\theta)^4$.

En statistique inférentielle et en apprentissage automatique, on cherche à inférer un modèle à partir d'observations. Une stratégie classique consiste à choisir le modèle qui maximise la vraisemblance de l'observation. Quel est le paramètre θ qui maximise $L(w;\theta)$? Pour le calculer, on utilise le logarithme de la vraisemblance $\log L(w;\theta)$ puisque ces fonctions ont la même variation. On a $\log L(w;\theta) = 6\log\theta + 4\log(1-\theta)$. La dérivée de cette fonction est $\frac{6}{\theta} - \frac{4}{1-\theta} = \frac{6-10\theta}{\theta(1-\theta)}$ qui s'annule pour $\theta = 6/10$. C'est cette valeur qui maximise la vraisemblance de l'observation.

Pour expliquer une séquence de symboles observables w, et donc éventuellement pour prédire de nouvelles observations, on peut souhaiter rechercher le HMM qui maximise l'observation. Si l'on n'introduit aucune contrainte sur la taille du HMM recherché, cette stratégie conduira à proposer un HMM ayant le même nombre d'états que l'observation et pouvant la générer avec une probabilité de 1 ! Ce HMM a toutes les chances de n'avoir aucune capacité prédictive, et donc de constituer une très mauvaise explication de l'observation. Il est nécessaire de se donner une contrainte supplémentaire sur la taille du HMM considéré, ou selon les problèmes d'autres contraintes plus sophistiquées. On recherchera par exemple le HMM à deux états maximisant la vraisemblance de l'observation. On peut montrer qu'il n'y a pas de solution analytique à ce problème et qu'il est même NP-difficile - et donc sans espoir d'une solution efficace et exacte. L'algorithme de Baum-Welch décrit dans la section suivante propose une heuristique possédant de bonnes garanties théoriques - mais pas celle de converger vers la solution dans tous les cas.

4.4.2 Algorithme de Baum-Welch : principes

L'algorithme de Baum-Welch peut-être décrit de la manière suivante :

— un HMM M_0 est généré aléatoirement ; on suppose que l'observation w a été générée à l'aide de ce modèle

— pour tous les états k, on évalue la probabilité que le processus ait démarré dans cet état ; pour tout couple d'états (k, l), on évalue le nombre moyen de fois que la transitions (k, l) a été empruntée lors de la génération de w ; pour tout couple (k, o), on évalue le nombre moyen de fois que le symbole o a été émis à partir de l'état k lors de la génération de w
— une fois ces calculs faits, on réévalue les paramètres de M_0 de façon à augmenter la vraisemblance de l'observation : cela définit un modèle M_1,
— on recommence le processus avec M_1.

On peut démontrer que

— à chaque itération, on obtient un modèle qui accroît la vraisemblance de l'observation
— le processus converge (le plus souvent) vers un maximum *local* de la vraisemblance
— le plus souvent, ce maximum local n'est pas le maximum global.

En pratique :

— on arrête le processus soit lorsque la vraisemblance de l'observation n'augmente plus que très faiblement, ou après un nombre fixe d'itérations
— on relance le processus plusieurs fois après réinitialisation aléatoire du HMM M_0. On garde celui qui assure la meilleure vraisemblance.

Cet algorithme n'apporte donc pas de garanties théoriques très fortes - en pratique, il s'avère un des meilleurs dans un très grand nombre de problèmes.

Exemple 9 Reprenons le HMM de l'exemple 4.1. Supposons que l'observation soit $\{PF\}$. Sa vraisemblance dans le modèle est égale à $0.5 \times 0.5 \times (0.9 \times 0.5 + 0.1 \times 0.3) + 0.5 \times 0.7 \times (0.9 \times 0.3 + 0.1 \times 0.5) = 0.12 + 0.112 = 0.232$.

— Le premier état a été choisi comme état initial 0.12 fois. Le second état a été choisi comme état initial 0.112 fois. On redéfinit les probabilités initiales par $[0.517, 0.483]$.
— Pile a été émis dans le premier état en moyenne 0.12 fois. Face a été émis dans le premier état en moyenne $0.5 \times 0.5 \times 0.9 \times 0.5 + 0.5 \times 0.7 \times 0.1 \times 0.5 = 0.13$ fois. On redéfinit les probabilités d'émissions dans le premier état par $[0.12/0.25, 0.13/0.25 = [0.48, 0.52]$.
— La transition du premier état vers lui-même a été empruntée en moyenne $0.5 \times 0.5 \times 0.9 \times 0.5 = 0.1125$ fois. La transition du premier état vers le second a été empruntée en moyenne $0.5 \times 0.5 \times 0.1 \times 0.3 = 0.0075$ fois. On redéfinit ces probabilités de transitions

par $[0.9375, 0.0625]$.
La vraisemblance de l'observation dans le modèle devient 0.2506. Si l'on recommence l'opération, on obtient un modèle de vraisemblance 0.2508.

4.4.3 L'algorithme

Soit M un HMM, $w = o_1 \dots, o_n$ une séquence générée par M et $s_1 \dots s_n$ la suite d'états associée.

Pour chaque indice $1 \leq t \leq n$ et chaque état k, on a

$$\begin{aligned} P(w, s_t = k) &= P(o_1, \dots, o_t, s_t = k, o_{t+1}, \dots, o_n) \\ &= P(o_1, \dots o_t, s_t = k) \cdot P(o_{t+1}, \dots o_n | s_t = k) \\ &= f_k(t) \cdot b_k(t). \end{aligned}$$

On en déduit en particulier que pour tout indice t,

$$P(w) = \sum_k f_k(t) \cdot b_k(t)$$

et que

$$P(s_t = k|w) = \frac{P(w, s_t = k)}{P(w)} = \frac{f_k(t) \cdot b_k(t)}{\sum_k f_k(t) \cdot b_k(t)}.$$

Pour chaque indice $1 \leq t \leq n-1$ et chaque couple d'états (k, l), on a

$$\begin{aligned} P(w, s_t = k, s_{t+1} = l) &= P(o_1, \dots, o_t, s_t = k, o_{t+1}, s_{t+1} = l, o_{t+2} \dots, o_n) \\ &= P(o_1, \dots o_t, s_t = k) \cdot P(s_{t+1} = l | s_t = k) \cdot P(o_= o_{t+1} | s_{t+1} = l) \\ &\cdot P(o_{t+2}, \dots o_n | s_{t+1} = l) \\ &= f_k(t) T(k, l) E(l, o_{t+1}) \cdot b_k(t+1). \end{aligned}$$

En particulier, pour tout indice $t \leq n-1$,

$$P(w) = \sum_k \sum_l f_k(t) T(k, l) E(l, o_{t+1}) \cdot b_l(t+1).$$

On en déduit que

$$P(s_t = k, s_{t+1} = l|w) = \frac{f_k(t) T(k, l) E(l, o_{t+1}) \cdot b_l(t+1)}{\sum_k \sum_l f_k(t) T(k, l) E(l, o_{t+1}) \cdot b_l(t+1)}.$$

Introduisons les notations suivantes :

$$\gamma_k(t) = \frac{f_k(t) \cdot b_k(t)}{\sum_k f_k(t) \cdot b_k(t)} \text{ et } \xi_{kl}(t) = \frac{f_k(t) T(k, l) E(l, o_{t+1}) \cdot b_l(t+1)}{\sum_k \sum_l f_k(t) T(k, l) E(l, o_{t+1}) \cdot b_l(t+1)}.$$

On en déduit les mises à jour suivantes :

$$\pi^*(k) = P(s_1 = k|w) = \gamma_k(1). \tag{4.5}$$

$$T^*(k,l) = \frac{1}{Z_T^k} \sum_{t=1}^{n-1} P(s_t = k, s_{t+1} = l|w) = \frac{1}{Z_T^k} \sum_{t=1}^{n-1} \xi_{kl}(t) \tag{4.6}$$

où Z_T^k est un coefficient normalisateur.

$$O^*(k,o) = \frac{1}{Z_O^k} \sum_{t=1}^{n} 1_{o_t=o} P(s_t = k|w) = \frac{1}{Z_O^k} \sum_{t=1}^{n} 1_{o_t=o} \gamma_k(t) \tag{4.7}$$

où Z_O^k est un coefficient normalisateur.

Dans la pratique, on dispose de K séquences $S = \{w_1, \ldots, w_K\}$ et l'on cherche à maximiser la vraisemblance de S. Cela revient, après avoir ajouté l'exposant j aux paramètres $\gamma_k^{(j)}$ et $\xi_{kl}^{(j)}$, à effectuer les mises à jour suivantes :

$$\pi^*(k) = P(s_1 = k|S) = \frac{1}{K} \sum_{j=1}^{K} \gamma_k^{(j)}(1). \tag{4.8}$$

$$T^*(k,l) = \frac{1}{Z_T^k} \sum_{j=1}^{K} \sum_{t=1}^{n-1} \xi_{kl}^{(j)}(t) \tag{4.9}$$

$$O^*(k,o) = \frac{1}{Z_O^k} \sum_{j=1}^{K} \sum_{t=1}^{n} 1_{o_t=o} \gamma_k^{(j)}(t) \tag{4.10}$$

où les Z_T^k et Z_O^k sont de nouveaux coefficients normalisateurs.

On en déduit les algorithmes suivants :

```
Algorithme BW1
Entrée : HMM M0, liste S de séquences
Sortie : HMM M1(pi*, T*, O*) après mise à jour.

Algorithme BW2
Entrée : entier nbS, entier nbL, liste S de séquences, entier N
Sortie : HMM M
Tirer aléatoirement un HMM M construit sur nbL lettres et nbS états
Pour i = 1 à N faire
   M = BW2(M,S)
```

```
Algorithme BW3
Entrée : entier nbS, entier nbL, séquence w, entier N, entier M
Sortie : HMM M
Pour i = 1 à M faire
   Mi = BWM2(nbS, nbL, w, N)
retourner le HMM Mi qui maximise la vraisemblance de w
```

Il faudra programmer les méthodes :

— `gen_HMM(nbL, nbS)` pour tirer un HMM au hasard,

— `logV(self,S)` qui calcule la log-vraisemblance d'un échantillon : cela permettra de vérifier que la vraisemblance des observations augmente à chaque itération.