Modèles de Markov Cachés [1]

Objectif : la segmentation automatique de mails

Ce TP vise à construire un outil de segmentation de e-mails, visant à séparer l'en-tête du mail de son corps. On propose pour réaliser cette tâche d'apprendre un HMM à deux états, l'un (l'état 0) pour l'en-tête, l'autre (état 1) pour le corps. Dans ce modèle, on suppose que chaque mail contient effectivement un en-tête : le décodage débute nécessairement dans l'état 0. Le vecteur des probabilités initiales est donc [1, 0]. Sachant que chaque mail contient exactement un en-tête et un corps, chaque mail emprunte une fois unique la transition de 0 vers 1. La matrice de transition ($A(i,j) = P(j|i)$) estimée sur un petit corpus étiqueté à donc l'allure suivante : $A = [0.999218078035812, 0.000781921964187974; 0, 1]$.

Dans l'état 1 (en-tête), on peut passer avec une faible probabilité dans l'état 2 ; en revanche, dans l'état 2, on ne peut que rester dans l'état 2. Chaque partie du mail est caractérisée par une distribution de probabilité discrète sur les caractères $P(c|s)$, avec $s = 0$ ou $s = 1$.

Matériel et documentation

Pour la première partie du TP, on travaille avec les distributions qui sont fournies dans le fichier P.dat. Chacune des colonnes de ce fichier contient la distribution des probabilités d'occurrence de chacun des codes ASCII respectivement dans l'en-tête et dans le corps. Ces distributions ont été apprises sur un petit corpus étiqueté de 10 mails ; elles présentent des différences manifestes, surtout dans les zones où les codes ASCII correspondent aux caractères alphabétiques, comme vous pourrez vous en apercevoir en visualisant ces distributions.

Codage des mails

Pour coder les mails, on les représente comme des vecteurs de caractères ASCII. Un codeur et un décodeur sont fournis respectivement dans les deux scripts codeur et decodeur. Pour les utiliser :

```
perl codeur mail.txt
```

transforme mail.txt en un vecteur de chiffres (1 par ligne).

```
perl decodeur mail.dat
```

transforme un vecteur de chiffres (entre 0 et 255) en un texte ; Les fichiers de la forme `dat/*.dat` contiennent les versions déjà codées des mails correspondants. La liste figure dans `mail.lst`. Enfin, l'utilitaire `segment.pl` permet de visualiser une segmentation produite par votre segmenteur sous la forme du meilleur chemin trouvé par l'algorithme de Viterbi (dans un vecteur de 1 et de 2). Pour l'utiliser :

```
perl segment.pl mail.txt path.dat
```

produit un fichier dans lequel la segmentation est visualisée par

```
"===== coupez ici".
```

1. TP d'origine défini par François Yvon http://perso.limsi.fr/yvon/mysite/mysite.php?n=Main.HomePage

Travail à réaliser

L'ensemble du travail est à réaliser sous MatLab. En utilisant les valeurs données précédemment pour les paramètres du modèle, implémentez l'algorithme de Viterbi et testez le sur quelques mails qui sont donnés dans le répertoire dat (en particulier `mail11.txt` à `mail30.txt`). Concrètement, il s'agit de coder une fonction matlab qui prend comme argument un vecteur d'observations et les paramètres du modèle, et retourne un vecteur d'états représentant la séquence la plus probable.

Pour aller plus loin

1. Comment modéliseriez vous le problème si vous deviez segmenter les mails en plus de deux parties (par exemple : en-tête, corps, signature) ?
2. Comment modéliseriez vous le problème de séparer les portions de mail inclus en sachant qu'elles débutent toujours par le caractère "$>$".
3. On se propose maintenant d'implémenter les briques nécessaires à l'apprentissage non supervisé des paramètres du modèle.
   (a) Programmez le calcul des alpha. Vous pouvez si vous en sentez le besoin utiliser la fonction logsum, qui calcule des sommes de log. Si $x = log(y)$ et $x' = log(y')$, $logsum(x, x') = log(exp(x) + exp(x'))$.
   (b) Programmez le calcul des beta. On rappelle que $beta(t, i) = P(X_{t+1}...X_T|S_t = i)$.
   (c) A l'aide des formules de réestimation étudiées en cours, achevez de programmer une étape de l'algorithme EM. Ceci demande de calculer le vecteur des gammas ($gamma(t, i)P(S_t = i|X_1...X_T)$), puis d'utiliser ce vecteur pour mettre à jour les matrices A et P.
4. Lancez l'apprentissage en initialisant les paramètres aléatoirement, puis en initialisant avec des distributions de la question 1. Que concluez-vous ?
5. Il est en fait possible de faire un apprentissage "approché" en remplaçant le calcul des alphas/betas par un appel à viterbi. Le vecteur gamma des probabilités d'occupation des états est dans ce cas déterministe. à cette différence près, la mise à jour des paramètres reste la même. Implémentez cette variante et comparez les résultats obtenus avec la précédente. Conclusions ?