TP 3 : Modèles de Markov cachés

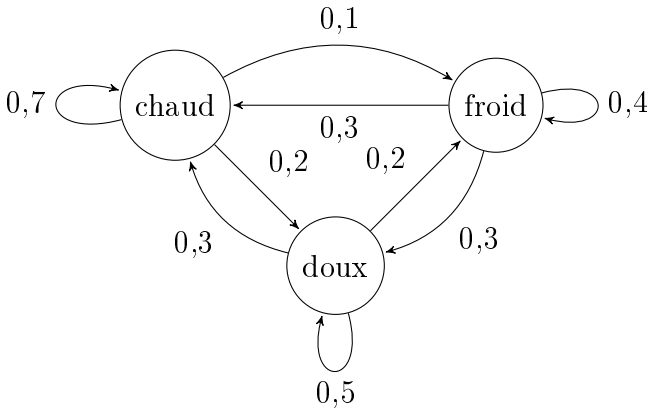
Objectifs

Le but de ce sujet est de mettre en œuvre les modèles de Markov cachés (HMM) et de les appliquer dans trois applications différentes.

Afin de gagner du temps, une partie du code (en C++) vous est fourni.

1 Prédiction de la meilleure séquence d'états

Dans cette section, on s'intéresse à un cas d'école, similaire à celui présenté en cours, où l'on chercherait à inférer la météo (chaud, doux, froid) pour chaque jour, à partir d'un journal personnel où aurait été consigné chaque jour le type de vêtement utilisé (vêtement d'été, de mi-saison, d'hiver). Les probabilités de transition A sont données par l'automate suivant :



Les paramètres B et π sont fixés comme suit : $B = \begin{pmatrix} 0.8 & 0.19 & 0.01 \\ 0.5 & 0.4 & 0.1 \\ 0.01 & 0.2 & 0.79 \end{pmatrix}$ et $\pi = \begin{pmatrix} 0.6 & 0.3 & 0.1 \end{pmatrix}$.

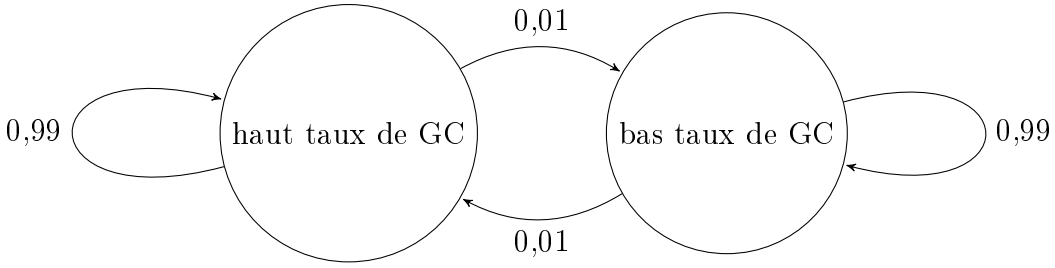
1. Compléter la fonction `viterbi(vector<int>* seq_obs)` calculant la meilleure séquence d'états cachés pour une séquence d'observations donnée ;
2. Tester votre code sur les données `data/weather/weather-test.csv` en utilisant les paramètres de modèle `data/weather/weather.true.model`. Vérifier que vous retrouvez bien les prédictions données dans `data/weather/weather-test.ref`.

2 Application en bio-informatique

Une utilisation importante des HMM dans le domaine de la bio-informatique consiste à décoder ou analyser un génome en ses composants biologiques : exons, introns, région régulatrice... Dans

cette partie, on s'intéresse à la détection spécifique de régions à haut taux et à bas taux de GC[1] de séquences d'ADN. Le taux de GC exprime le pourcentage de liaisons G-C dans la molécule d'ADN et confère à l'ADN des propriétés différentes concernant les températures de fusion, les temps de réplication dans le cycle cellulaire ou la densité de gènes. On considérera que les régions à haut taux de GC auront en moyenne 60 % de G ou de C, alors que celles à bas taux auront en moyenne 60 % de A ou de T.

Un modèle HMM simple, à deux états, est construit pour permettre de détecter les deux types de régions. La transition entre les deux états est définie par l'automate suivant.



De plus, on considère que la probabilité qu'une séquence d'ADN commence par l'un des deux types de régions est égale, tandis que la probabilité d'émission des bases en fonction de la région est définie par le tableau qui suit.

	$v_t = A$	$v_t = G$	$v_t = C$	$v_t = T$
$P(o_t = v_t \mid q_t = \text{haut taux de GC})$	0,20	0,30	0,30	0,20
$P(o_t = v_t \mid q_t = \text{bas taux de GC})$	0,30	0,20	0,20	0,30

1. Appliquer le programme avec votre code Viterbi sur le fichier data/genetics/CG-resgion.csv. En utilisant le script src/compErrors.py et les données de référence data/genetics/CG-region.ref, donner la précision du HMM sur ces données.

3 Apprentissage par maximum de vraisemblance

Jusqu'à maintenant les HMM sont définis à la main, que ce soit au niveau de la structure ou des paramètres. Dans cette section, on souhaite enrichir les capacités du programme pour lui permettre d'apprendre de manière supervisée les paramètres.

1. Définir le calcul des probabilités a_{ij}, b_{jk} et π_i en fonction des comptes déterminés sur un ensemble d'apprentissage où chaque observation est annotée par son état caché, et en suivant le critère du maximum de vraisemblance.
2. Compléter la fonction train_MLE(vector<vector<int>*>* data_obs, vector<vector<int>*>* data_state, string filename) calculant les paramètres du HMM à partir des observations et des états des données d'apprentissage. Les vecteurs (*data_obs)[i] et (*data_state)[i] donnent ainsi respectivement les observations et les états de la séquence i des données. Les paramètres appris du HMM devront être sauvegardés dans le fichier filename.
3. Utiliser votre fonction pour entraîner un HMM sur les données météo data/weather/weather-500seq.csv. Appliquer le modèle ainsi construit sur les séquences de data/weather/weather-test.csv et calculer le taux de classification en comparant avec le fichier *.ref.

1. Les séquences d'ADN sont séquencées avec 4 bases : la guanine (G), la cytosine (C), l'adénine (A) et la thymine (T).

4 Application en traitement automatique des langues

Dans cette section, on s'intéresse à l'étiquetage par les parties du discours (PoS, *Part of Speech* en anglais) qui est souvent utile comme étape de pré-traitement dans un bon nombre d'applications du traitement automatique des langues. Une partie du discours est une classe grammaticale (nom, verbe, adjectif, préposition...) pouvant être associée à chaque mot d'une phrase en fonction de son contexte d'utilisation. Les catégories grammaticales peuvent être définies de manière plus précise, en prenant en compte par exemple le genre, le nombre, le cas, la conjugaison. Nous prendrons ici un ensemble fixe de 17 catégories, dont une description vous est fournie dans le fichier data/POS-tagging/README.

L'étiquetage par les parties du discours peut être modélisé par un HMM en assimilant les PoS aux états cachés et les mots aux observations. La recherche de la meilleure séquence de PoS t_1^n pour la séquence de mots w_1^n est :

$$\begin{aligned}\hat{t}_1^n &= \arg\max_{t_1^n} P(t_1^n|w_1^n) \\ &= \arg\max_{t_1^n} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)} \\ &= \arg\max_{t_1^n} P(w_1^n|t_1^n)P(t_1^n)\end{aligned}$$

Les étiqueteurs basés sur les HMM font deux hypothèses. La première est que la probabilité qu'un mot apparaisse ne dépend que de sa partie du discours et est indépendante des mots et des étiquettes voisines :

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

La seconde hypothèse est que la probabilité d'une étiquette dépend seulement de l'étiquette précédente, plutôt que de la séquence entière d'étiquettes :

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

L'étiquetage revient ainsi au calcul de :

$$\hat{t}_1^n \approx \arg\max_{t_1^n} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

où $(w_i|t_i)$ est la probabilité d'émission B et $P(t_i|t_{i-1})$ la probabilité de transition A.

Pour tester les capacités des HMM dans ce cadre applicatif, nous prendrons les données disponibles pour le français sur le site `https://universaldependencies.org/#download`. Ces données ont été reformatées et mises à disposition dans le répertoire data/POS-tagging.

1. Contrairement aux deux applications vues précédemment où le vocabulaire des observations était fermé, il est possible (pour ne pas dire très probable) de rencontrer ici des mots qui n'ont pas été vus par le modèle dans son apprentissage. À partir de l'examen des fonctions read_vocab et read_sequences, expliquer comment le programme traite les mots inconnus ;
2. Le fichier fr-train.word-pos concatène l'ensemble des fichiers des sous-corpus *train.word-pos. Construire un HMM sur ce fichier en utilisant le critère du maximum de vraisemblance. Étiqueter avec l'algorithme de Viterbi chacun des corpus de test *test.word. Donner un tableau synthétique donnant le pourcentage total d'étiquettes erronées pour chaque corpus de test. Quelles sont les étiquettes les plus difficiles à prédire ?

5 Calcul de scores de confiance

Jusqu'à présent, le programme est capable de donner l'étiquette la plus probable pour un HMM donné. Nous souhaitons ajouter ici une nouvelle fonctionnalité qui indique la confiance qu'a le modèle dans la prédiction de chaque étiquette t_i. Pour ce faire, nous utiliserons la probabilité *a posteriori* $P(t_i|w_1^n)$. Cette probabilité correspond à la fonction $\gamma_t^k(i)$ du cours, donnant la probabilité que l'observation au temps t d'une séquence $O^k = w_1^n$ soit émise par l'état s_i (ici l'étiquette t_i). Vous pourrez utiliser le calcul suivant ;

$$\gamma_t^k(i) = \frac{\alpha_t^k(i)\beta_t^k(i)}{P(O^k|\mathcal{M})}$$

Le calcul de la confiance s'appuie ainsi sur l'algorithme *forward-backward*.

1. Coder les fonctions alpha(vector<int>* seq_obs) et beta(vector<int>* seq_obs), donnant respectivement les valeurs $\alpha_t^k(i)$ et $\beta_t^k(i)$ pour chaque séquence d'observations O^k associée à la variable seq_obs. NB : Ces fonctions doivent faire des sommes de probabilités, ce qui peut conduire à manipuler des probabilités très faibles. Il est impératif d'employer les fonctions ln_of_sum qui vous sont données pour faire la somme de deux probabilités et en récupérer le logarithme, sous peine de se retrouver avec des logarithmes valant $-\infty$;
2. Coder la fonction pgen(vector<int>* seq_obs, double** alpha) calculant la probabilité que le HMM génère la séquence O^k, en utilisant comme suit la fonction alpha :

   $$P(O^k|\mathcal{M}) = \sum_{i=1}^{n} \alpha_T^k(i)$$

   où T est la longueur de la séquence d'observations O^k ;
3. Coder la fonction posterior(vector<int>* seq_obs, vector<int>* seq_state) en réutilisant les trois fonctions que vous venez de définir ;
4. Afficher les probabilités a posteriori du HMM en étiquetant le fichier fr-test.word concaténant tous les fichiers tests du sous-corpus. Pour réaliser cet affichage lors de l'étiquetage, décommenter la ligne faisant appel à la fonction posterior dans la fonction write_predict de use_hmm.cpp. Vous pourrez vérifier le calcul des probabilités *a posteriori* sur le jeu de données data/weather/weather-test.ref.posteriors.

6 Rendu

Ce TP est à faire de manière individuelle.

Il vous est demandé de déposer un rapport répondant aux questions, le fichier fr-test.word étiqueté, ainsi que le code source C++ modifié.