MINISTÈRE
DE L'ÉDUCATION
NATIONALE
ET DE LA JEUNESSE
Liberté
Égalité
Fraternité

EAE INF 2

SESSION 2024

AGREGATION
CONCOURS EXTERNE

Section : INFORMATIQUE

ÉTUDE D'UN PROBLÈME INFORMATIQUE

Durée : 6 heures

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout matériel électronique (y compris la calculatrice) est rigoureusement interdit.

Il appartient au candidat de vérifier qu'il a reçu un sujet complet et correspondant à l'épreuve à laquelle il se présente.

Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence. De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.
Le fait de rendre une copie blanche est éliminatoire.

Tournez la page S.V.P.

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie.

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

Concours	Section/option	Epreuve	Matière
EAE	6200A	102	9423

RECTIFICATIF

La formule en bas de la page 5 devrait être écrite à l'horizontale et non à la verticale.

Au lieu de :

1/4
1/4
E0= [1/4]
1/4

Lire :

E0 = [1/4 1/4 1/4 1/4]

Les modèles de Markov cachés

Dans ce sujet, nous allons aborder l'implémentation des modèles de Markov cachés. Il s'agit de modèles probabilistes qui permettent d'engendrer des mots, et plus spécifiquement des séquences d'ADN (acide désoxyribonucléique). Ils permettent de représenter des familles de séquences d'ADN proches, comme un gène codant pour une même protéine dans différents organismes. Ils sont constitués d'états dans lesquels sont émis des caractères selon une distribution probabiliste propre à chaque état, et les transitions entre les états sont également régies par une loi de probabilité. Lorsque l'on observe une séquence produite par le modèle, on ne connaît pas les états qui ont été visités pour la produire, d'où le terme *caché* dans le nom du modèle. La première partie sert à se familiariser avec cette notion de modèle, la deuxième partie s'intéresse à leur représentation informatique dans le paradigme de la programmation objet. La troisième partie s'intéresse aux algorithmes permettant de faire fonctionner ces modèles, ainsi qu'à leur implémentation. Enfin, la quatrième partie permet de raffiner le modèle pour leur donner un pouvoir de représentation plus réaliste.

Préliminaires

Programmation. Les questions de programmation doivent être traitées en langage Python. L'usage ou l'importation de tout module sera interdit. Lorsque le candidat écrira une fonction, il pourra faire appel à des fonctions définies dans les questions précédentes, même si elles n'ont pas été traitées. Il pourra également définir des fonctions ou des classes auxiliaires, mais devra préciser leurs rôles ainsi que les types et significations de leurs arguments. Les candidats sont encouragés à expliquer les choix d'implémentation de leurs fonctions lorsque ceux-ci ne découlent pas directement des spécifications de l'énoncé. Si les paramètres d'une fonction à coder sont supposés vérifier certaines hypothèses, il ne sera pas utile dans l'écriture de cette fonction de tester si les hypothèses sont bien satisfaites.

Notations. On identifiera une même grandeur écrite dans deux polices de caractères différentes, en italique du point de vue mathématique (par exemple n) et en Computer Modern à chasse fixe du point de vue informatique (par exemple `n`).

Probabilités. Il est fait mention de probabilités dans la suite. Une probabilité est notée $Prob$, et notée $Prob_H$ lorsqu'il s'agit d'une probabilité dans le modèle H.

Complexité. Sans précision supplémentaire, lorsqu'une question demande la complexité d'une fonction, il s'agira de la complexité temporelle dans le pire des cas. On considérera que toutes les opérations arithmétiques sur les entiers s'effectuent en temps constant. La complexité sera exprimée sous la forme $O(f(n, m))$ où n et m sont les tailles des arguments de la fonction, et f une expression simple. Les calculs de complexité seront justifiés succinctement. Lorsqu'une question de programmation précise qu'une complexité est attendue, sauf demande explicite, il n'est alors pas nécessaire de justifier que la fonction écrite vérifie cette contrainte.

Tournez la page S.V.P.

Dépendances. Ce sujet contient plusieurs parties. Chaque partie utilise des définitions et des résultats des parties précédentes. Sauf mention explicite du contraire, les questions restent néanmoins indépendantes, au sens où toute question peut être traitée en admettant les résultats énoncés dans les questions précédentes.

Attendus. Il est attendu des candidates et des candidats des réponses construites. Ils seront aussi évalués sur la précision, le soin et la clarté de la rédaction.

Partie I. Définitions et propriétés élémentaires

En bioinformatique, les modèles de Markov cachés sont utilisés pour représenter une famille de séquences d'ADN (acide désoxyribonucléique). Ce sont des modèles probabilistes qui se composent de plusieurs éléments. Un modèle de Markov caché H est un quintuplet (Σ, S, T, E, P), tel que :

— l'ensemble Σ est l'alphabet sur lequel les séquences sont construites. On pose m la taille de cet alphabet. Pour les séquences génomiques, $\Sigma = \{\text{A,C,G,T}\}$ et $m = 4$;

— l'ensemble S est un ensemble fini non vide dont les éléments sont appelés *états* du modèle. On note n la taille de S. Dans toute la suite on supposera que $S = \{1, 2, \ldots, n\}$;

— la matrice T est une matrice réelle de dimension $n \times n$, qui est la matrice des probabilités de transition $(\forall (i,j) \in \{1, \ldots n\}^2, T(i,j) \in [0,1])$;

— la matrice E est une matrice réelle de dimension $n \times m$, qui est la matrice des probabilités d'émission $(\forall (i,c) \in \{1, \ldots n\} \times \Sigma, E(i,c) \in [0,1])$;

— le vecteur P est un vecteur réel de taille n, qui est le vecteur des probabilités de départ $(\forall i \in \{1, \ldots n\}, P(i) \in [0,1])$.

Les contraintes classiques sur un modèle de Markov caché sont les suivantes :

— La somme des probabilités des états initiaux est égale à 1 :

$$\sum_{i=1}^{n} P(i) = 1 \tag{1}$$

— La somme des probabilités des transitions partant d'un état est égale à 1 :

$$\forall i \in \{1, \ldots, n\}, \sum_{j=1}^{n} T(i,j) = 1 \tag{2}$$

— La somme des probabilités d'émission dans un état est égale à 1 :

$$\forall i \in \{1, \ldots, n\}, \sum_{c \in \Sigma} E(i,c) = 1 \tag{3}$$

La représentation graphique d'un modèle de Markov caché est constituée d'un graphe dont les sommets, représentant les états, contiennent les probabilités d'émission, et les arcs représentent les probabilités de transition non nulles, qui les étiquettent. Un exemple de représentation sous

forme de graphe d'un modèle de Markov caché est donné en figure 1. Il correspond au modèle H_1 suivant : $\Sigma = \{\text{A, C, G, T}\}$, $n = 4$, et

$$P = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad T = \begin{bmatrix} 0.5 & 0 & 0.5 & 0 \\ 0 & 0.1 & 0.8 & 0.1 \\ 0 & 0.2 & 0.5 & 0.3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad E = \begin{array}{c} \begin{array}{cccc} \text{A} & \text{C} & \text{G} & \text{T} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 & 0 \\ 0.2 & 0.1 & 0.4 & 0.3 \end{bmatrix} \end{array}.$$

Les probabilités de départ ne sont pas représentées dans le graphe de la figure 1.

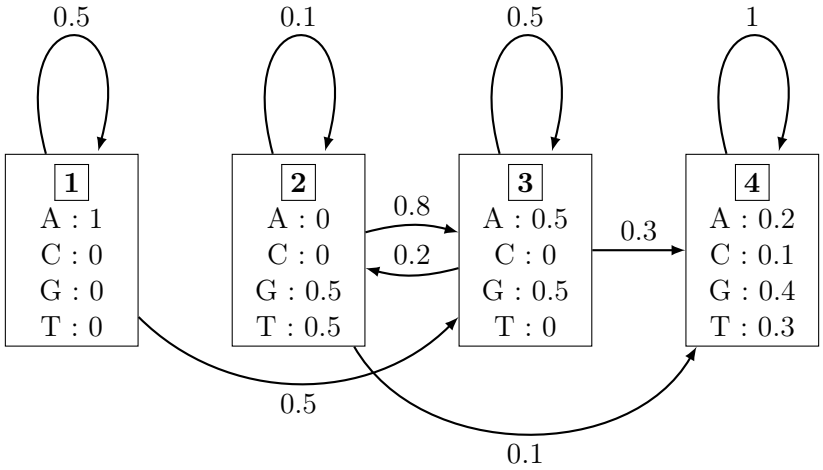

FIGURE 1 – Modèle de Markov caché H_1 représenté sous forme de graphe.

On considèrera dans toute la suite uniquement des modèles de Markov dont tous les états sont accessibles depuis un état de probabilité de départ non nulle.

1 Quelques définitions

1.1 Les marches

Une *marche* $m = m_1 m_2 \dots m_k$ dans le modèle de Markov caché H est une suite d'états de H, telle que :

- le premier état de la marche a une probabilité non nulle dans le vecteur des probabilités de départ : $P(m_1) \neq 0$;
- chaque succession d'état dans la marche a une probabilité non nulle dans la matrice des probabilités de transition : $\forall i \in \{1, \dots k-1\}$, $T(m_i, m_{i+1}) \neq 0$.

Par exemple 11332 est une marche possible dans le modèle H_1 défini ci-dessus, alors que la marche 2112 ne l'est pas.

La probabilité $Prob_H(m)$ d'une marche m dans le modèle H est définie comme le produit des probabilités :

Tournez la page S.V.P.

— de départ dans le premier état de m ;

— de transition d'un état de m au suivant.

Ainsi :

$$Prob_H(m) = P(m_1) \times \prod_{i=1}^{k-1} T(m_i, m_{i+1)}.$$

Par exemple, la probabilité de la marche 11324 dans le modèle H_1 ci-dessus est :

$$Prob_{H_1}(11324) = P(1) \times T(1,1) \times T(1,3) \times T(3,2) \times T(2,4) = 1 \times 0.5 \times 0.5 \times 0.2 \times 0.1 = 5 \times 10^{-3}.$$

On pourra remarquer qu'une marche m est possible dans le modèle H si et seulement si $Prob_H(m) \neq 0$.

Question 1 : Pour chaque marche suivante, indiquer si elle est possible dans le modèle H_1 et donner sa probabilité dans le modèle H_1 :

a) 13444

b) 234

c) 132323232

d) 1234

e) 13432

1.2 Les séquences

Une séquence $w = w_1 \dots w_k$ de caractères de Σ peut être *produite* dans le modèle de Markov caché H s'il existe une marche $m = m_1 \dots m_k$ de même longueur que la séquence, possible dans H, et telle que chaque caractère a une probabilité non nulle d'émission dans l'état de même indice : $\forall i \in \{1, \dots, k\}, E(m_i, w_i) \neq 0$.

Par exemple, la séquence AAGAT peut être produite dans le modèle H_1, notamment grâce à la marche 11332, mais la séquence TACCG ne peut pas être produite par le modèle H_1, car aucune marche ne commence dans un état qui permet d'émettre T avec une probabilité non nulle.

Question 2 : Pour chacune des séquences suivantes, indiquer si elle peut être produite dans le modèle H_1, et le cas échéant, indiquer une marche qui permet de la produire :

a) AATATA

b) CATGTG

c) AACGAA

d) AAGGGG

Les séquences sont des événements associés aux modèles de Markov cachés. Si l'on connaît une marche m du modèle H qui permet de produire la séquence w, on peut calculer la probabilité de produire la séquence w sachant m et H, en considérant les probabilités d'émission de caractères de w dans les états de m :

$$Prob_{m,H}(w) = \prod_{i=1}^{k} E(m_i, w_i).$$

Par exemple, la probabilité de produire la séquence AAGAT dans le modèle H_1 sachant que l'on a suivi la marche 11332 est :

$$Prob_{11332,H_1}(\text{AAGAT}) = E(1,\text{A}) \times E(1,\text{A}) \times E(3,\text{G}) \times E(3,\text{A}) \times E(2,\text{T}) = 1 \times 1 \times 0.5 \times 0.5 \times 0.5 = 0.125.$$

On va par la suite plutôt s'intéresser à la probabilité d'événements du type « la marche m a produit la séquence w, sachant le modèle H », notée $Prob_H(m, w)$, qui est calculée de la façon suivante :

$$Prob_H(m, w) = Prob_H(m) \times Prob_{m,H}(w) = P(m_1) \times \prod_{i=1}^{k-1} T(m_i, m_{i+1}) \times \prod_{i=1}^{k} E(m_i, w_i).$$

Question 3 : Pour les séquences suivantes, donner **une** marche ayant permis de la produire, avec la probabilité que cette marche produise la séquence dans le modèle H_1 :

a) AGCGAT

b) AGAAA

Question 4 : Montrer que dans H_1, il existe une marche qui peut produire plusieurs séquences. À quelle(s) condition(s) sur le modèle aurait-on une unique séquence par marche possible (sans pour autant que toutes les marches soient supposées possibles) ?

Question 5 : Montrer que dans H_1, il existe une séquence qui peut être produite par plusieurs marches.

Question 6 : Donnez toutes les marches possibles produisant la séquence AAG dans le modèle H_1. Laquelle maximise la probabilité de produire cette séquence dans le modèle H_1 ?

Question 7 : Pour une séquence w, indiquer à quelle probabilité correspond la somme pour toutes les marches possibles m des probabilités que m produise la séquence w dans un modèle de Markov H.

2 Quelques propriétés

On s'intéresse au modèle de Markov caché très particulier décrit ci-dessous, que l'on appelle le modèle nul : $H_0 = (\Sigma_0, S_0, P_0, E_0, T_0)$ avec $\Sigma_0 = \{\text{A, C, G, T }\}$, $S_0 = \{1\}$, $P_0 = [1]$,
$E_0 = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}$, et $T_0 = [1]$.

Tournez la page S.V.P.

Question 8 : Donner la représentation sous forme de graphe de ce modèle et vérifier qu'il satisfait bien aux contraintes de base des modèles de Markov cachés, données par les équations (1), (2), (3).

Question 9 : Montrer que H_0 peut produire des séquences non vides de longueur arbitraire.

Question 10 : Montrer que H_0 peut produire toutes les séquences non vides sur l'alphabet Σ_0, avec une marche que l'on précisera.

Question 11 : On considère Σ_0^k l'ensemble des séquences de taille $k \geq 1$ fixée sur l'alphabet Σ_0.

a) Donner le nombre de séquences possibles dans Σ_0^k.

b) Donner la probabilité d'une séquence de Σ_0^k dans le modèle H_0.

c) Calculer la somme des probabilités des séquences de Σ_0^k dans le modèle H_0.

On s'intéresse maintenant à nouveau au modèle H_1 décrit en figure 1.

Question 12 : Donner toutes les séquences produites par H_1 de taille 1, ainsi que leurs probabilités et la somme de ces probabilités.

Question 13 : Donner toutes les séquences produites par H_1 de taille 2, ainsi que leurs probabilités et la somme de ces probabilités.

Question 14 : Justifier le fait qu'il soit possible de produire des séquences non vides de taille arbitraire dans H_1.

On revient enfin au cas le plus général d'un modèle de Markov caché H quelconque.

Question 15 : Donner une condition suffisante sur les propriétés du graphe qui le représente pour qu'un modèle de Markov caché puisse produire des séquences arbitrairement longues. Montrer que tout modèle de Markov caché satisfait cette propriété.

Question 16 : Démontrer que, dans un modèle de Markov caché H quelconque, on a la propriété suivante : « Pour tout $k \geq 1$, la somme des probabilités des séquences de taille k produites par le modèle H est égale à 1. ». On pourra procéder par récurrence sur $k \geq 1$.

Partie II. Représentation des modèles de Markov cachés

On s'intéresse dans cette partie à la représentation des modèles de Markov cachés dans le langage Python. On implémentera ceux-ci dans le paradigme objet. La classe implémentant un modèle de Markov caché s'appelle la classe `HMM` (de *Hidden Markov Model*).

3 La classe `HMM`

Question 17 : En s'appuyant sur la description des modèles de Markov cachés, décrire la structure d'un objet `HMM` représentant un modèle de Markov caché pour des séquences ADN (donc avec le même alphabet). On demande ici une description textuelle des attributs et méthodes, pas une implémentation en Python.

Question 18 : Donner une implémentation de la classe `HMM`, avec un constructeur prenant en paramètre les différents éléments du modèle, et une méthode pour afficher les paramètres du modèle. Pour implémenter les matrices en Python, qui n'a pas de type intégré pour cette structure, il est conseillé d'utiliser des listes imbriquées. On demande explicitement ici de donner le code des méthodes en Python.

Question 19 : Modifier le constructeur de cette classe pour qu'en l'absence de paramètre dans l'appel, il initialise le modèle avec le modèle nul vu dans la section 2.

Question 20 : Donner l'instruction Python permettant d'instancier le modèle H_1 vu en figure 1.

Question 21 : En Python, pour comparer l'égalité entre deux objets instances d'une même classe, on peut utiliser l'opérateur d'égalité `==`. Quand on utilise cet opérateur, on fait en réalité appel à une méthode `__eq__` propre à la classe, qui prend comme paramètres `self` et un deuxième paramètre `other`. Lorsque l'opérateur d'égalité `__eq__` n'est pas surchargé, donc si on ne le définit pas spécifiquement dans la classe, il est par défaut identique à l'opérateur d'identité qui renvoie vrai si les deux paramètres désignent la même instance d'une classe. Ainsi, si on initialise deux `HMM`, `H1` et `H2` avec les mêmes valeurs initiales, l'expression `H1 == H2` est évaluée à `False`. Ce n'est pas un comportement adéquat puisque l'on va considérer que deux modèles de Markov cachés sont égaux s'ils ont les mêmes valeurs pour les attributs correspondants aux éléments du modèle. Implémenter l'opérateur `__eq__` qui correspond à cette notion d'égalité.

Question 22 : Implémenter une méthode `estValide` qui indique si le modèle respecte les contraintes de base des modèles de Markov cachés, données par les équations (1), (2), (3). On pourra utiliser l'opérateur `sum` qui prend une liste en paramètre et renvoie la somme des éléments de la liste. Exemple : `sum([1,2,3])` renvoie la valeur 6.

4 Adaptation à une famille de séquences

Dans la section précédente, nous avons vu une façon très générique d'implémenter les modèles de Markov cachés. Mais pour s'adapter à des familles de séquences proches bien définies (par exemple, une famille de séquences représentant un même gène dans plusieurs espèces), il est indispensable d'initialiser le modèle avec des informations suffisamment pertinentes vis-à-vis de cette famille de séquences. On s'appuie pour cela sur un alignement de q séquences, défini comme un tableau de dimension $q \times p$, où q est le nombre de séquences alignées, et p la longueur de l'alignement. Dans la ligne i de ce tableau, apparaissent dans l'ordre les caractères de la $i$ème séquence, entre lesquels peuvent s'intercaler des caractères tiret - comme dans la figure 2. On va s'intéresser ici à des alignements optimaux. Nous ne détaillons pas ici comment les obtenir, mais de cette optimalité découlent quelques propriétés que nous supposerons satisfaites dans la suite :

— la longueur de l'alignement p est inférieure à la somme des longueurs des séquences ;

— la longueur de l'alignement p est supérieure ou égale à la plus petite des longueurs des séquences ;

— chaque colonne contient au moins un caractère de l'alphabet ;

— il peut y avoir des tirets dans n'importe quelle colonne.

À partir de cet alignement, on calcule les fréquences d'apparition des différents caractères de l'alphabet $\Sigma = \{A,C,G,T\}$ dans les colonnes, comme sur la figure 2. Les tirets - dans l'alignement

Tournez la page S.V.P.

indiquent que la séquence s'aligne avant et après mais qu'il n'y a pas de caractère dans la séquence à mettre en correspondance dans la colonne où figure le tiret. Les successions de tirets s'appellent des *gaps*. Ainsi, dans la figure 2, la première séquence GATTCA est sans *gap*, la deuxième séquence GCTA présente deux *gaps* de taille 1, la troisième séquence GATTT présente un *gap* de taille 1, et la quatrième séquence GTCG présente un *gap* de taille 2.

colonne n°	1	2	3	4	5	6
séq. 1	G	A	T	T	C	A
séq. 2	G	-	C	T	-	A
séq. 3	G	A	T	T	-	T
séq. 4	G	-	-	T	C	G

A	0	1	0	0	0	$\frac{1}{2}$
C	0	0	$\frac{1}{3}$	0	1	0
G	1	0	0	0	0	$\frac{1}{4}$
T	0	0	$\frac{2}{3}$	1	0	$\frac{1}{4}$

FIGURE 2 – En haut, l'alignement des séquences GATTCA, GCTA, GATTT et GTCG. En bas, les fréquences d'apparition des caractères dans l'alignement. Le modèle de Markov caché généré à partir de cet alignement est noté H_2.

Le modèle de Markov caché correspondant à un alignement est alors défini comme suit :

— il est composé d'autant d'états que la longueur p de l'alignement,

— le départ se fait de façon sûre (*i.e.* avec probabilité égale à 1) dans le premier état,

— les probabilités d'émission dans l'état i correspondent aux fréquences des caractères apparaissant dans la colonne i de l'alignement,

— les probabilités de transition sont calculées de la façon suivante : une transition passe forcément d'un état où l'on émet vers un état où l'on émet également, elle ne peut pas passer par un *gap*. Ainsi, considérant un état i, pour chaque ligne ℓ où un caractère différent de `'-'` apparaît (donc pour lequel on émet dans l'état i), on va regarder dans quelle colonne $j_{i,\ell}$ apparaît le prochain caractère émis (donc différent de `'-'`). S'il y a c_i caractères de l'alphabet Σ apparaissant dans la colonne i, la probabilité de transition de l'état i vers l'état j est le nombre de lignes ℓ telles que $j_{i,\ell} = j$, divisé par c_i, et le dernier état transitionne de façon sûre (*i.e.* avec une probabilité égale à 1) vers lui-même.

Ainsi, dans la figure 2 par exemple, pour calculer les probabilités de transition de l'état $i = 1$ vers les autres états, on va considérer le nombre de caractères différents de - dans la colonne 1, c'est-à-dire $c_1 = 4$.

À la ligne 1, le caractère émis après celui de la colonne 1 ('G') est celui de la colonne 2 ('A') donc $j_{1,1} = 2$. À la ligne 2, le caractère émis après celui de la colonne 1 ('G') est celui de la colonne 3 ('C') donc $j_{1,2} = 3$. À la ligne 3, le caractère émis après celui de la colonne 1 ('G') est celui de la colonne 2 ('A') donc $j_{1,3} = 2$. À la ligne 4, le caractère émis après celui de la colonne 1 ('G') est celui de la colonne 4 ('T') donc $j_{1,4} = 4$.

La probabilité de transition de l'état $i = 1$ vers l'état $j = 2$ correspond à la fréquence des lignes ℓ telles que $j_{1,\ell} = 2$, ce qui donne une probabilité égale à $\frac{2}{4} = 0.5$. La probabilité de transition de l'état $i = 1$ vers l'état $j = 3$ correspond à la fréquence des lignes ℓ telles que $j_{1,\ell} = 3$, ce qui donne une probabilité de $\frac{1}{4} = 0.25$. La probabilité de transition de l'état $i = 1$ vers l'état $j = 4$ correspond à la fréquence des lignes ℓ telles que $j_{1,\ell} = 4$, ce qui donne une probabilité de $\frac{1}{4} = 0.25$. Toutes les autres probabilités de transition de l'état $i = 1$ vers les autres états sont nulles.

Question 23 : Représenter sous forme de graphe le modèle de Markov caché H_2 correspondant à l'alignement de la figure 2. Préciser à part le vecteur des probabilités de départ.

Question 24 : Un alignement est composé de séquences de même longueur (en comptant les *gaps*). On supposera que ces séquences ne comportent que les caractères `'A'`, `'C'`, `'G'`, `'T'` et `'-'` en majuscules. Choisir en justifiant une structure de données pour représenter un alignement.

Question 25 : Implémenter un constructeur `fromAlignement` de la classe `HMM` qui prend en paramètre un alignement et initialise les paramètres du modèle.

Question 26 : Montrer qu'un modèle de Markov caché initialisé avec un alignement satisfait nécessairement les contraintes de base des modèles de Markov cachés, données par les équations (1), (2) et (3).

5 Sur la représentation du modèle

Nous avons admis que les paramètres probabilistes du modèle de Markov caché sont représentés sous forme de matrices (implémentées par des listes de listes en Python). On s'intéresse à la matrice de probabilités de transition, dans le cas où le modèle est construit à partir d'un alignement. Il est admis qu'un alignement de bonne qualité se caractérise par des *gaps* de longueur faible.

Question 27 : Majorer le nombre d'entrées non nulles dans la matrice de probabilités de transition lorsque le modèle est construit à partir d'un alignement de longueur p et dont les *gaps* sont de longueur maximale g.

Question 28 : La densité d'une matrice est calculée comme étant le nombre d'éléments non nuls de la matrice divisé par le nombre total d'éléments dans la matrice. Donner un majorant de la densité de la matrice des probabilités de transition, en fonction de p.

Question 29 : La matrice des probabilités de transition peut être vue comme la matrice d'adjacence du graphe représentant le modèle de Markov. Donner une autre représentation possible de ce graphe, plus adaptée à la densité du graphe (correspondant à la densité de sa matrice d'adjacence). Appliquer cette représentation à la matrice de probabilités de transition du modèle H_2.

Question 30 : Donner de façon textuelle (on ne demande pas d'implémenter) les grandes modifications à réaliser pour implémenter la structure de données choisie. On indiquera notamment quelles sont les méthodes impactées et les changements éventuels de la complexité de l'accès aux éléments des matrices.

Tournez la page S.V.P.

Partie III. Algorithmes

6 L'algorithme de décodage

Le problème du décodage consiste à fournir, étant donnés une séquence w et un modèle de Markov caché H, une marche de probabilité maximale dans H qui produit w.

Question 31 : Montrer que, pour une séquence donnée $w_1 \dots w_k$, si $m_1 \dots m_k$ est une marche de probabilité maximale qui produit w, alors pour tout $k' \leq k$, $m_1, \dots m_{k'}$ est, parmi tous les préfixes de taille k' de marches produisant w se terminant en l'état m'_k, une marche qui produit $w_1 \dots w_{k'}$ avec une probabilité maximale.

Question 32 : Soit $w = w_1 \dots w_k$ une séquence de caractères de Σ. On définit le tableau suivant δ, de taille $k \times n$, avec $\delta[i, j]$ qui contient la probabilité maximale qu'une marche se terminant en l'état j produise le début de séquence $w_1 \dots w_i$.

a) Que contient la première ligne de ce tableau ?

b) En supposant les $i - 1$ premières lignes de ce tableau remplies, comment remplit-on la $i-$ième ligne ? Au besoin, faire un schéma pour appuyer l'explication.

c) Dans quelle case du tableau se trouve la probabilité maximale qu'une marche produise w ?

L'algorithme de Viterbi calcule une marche la plus probable pour une séquence $w = w_1 \dots w_k$ donnée, dans un modèle de Markov caché $H = (\Sigma, S, T, E, P)$. Il utilise le principe de la programmation dynamique, et comporte une phase de remplissage du tableau δ permettant de trouver la probabilité maximale qu'une marche produise w, puis une phase de retour arrière (*backtracking*) permettant d'identifier une marche qui atteint cette probabilité. La sortie de l'algorithme est cette marche la plus probable. On donne cet algorithme en figure 3.

Question 33 : Donner le résultat de cet algorithme sur la séquence AGG et le modèle H_1 de la figure 1. On indiquera également les tableaux δ et b remplis.

Question 34 : Donner une preuve de la correction de cet algorithme.

Question 35 : Donner la complexité de cet algorithme en temps dans le pire des cas.

7 Les algorithmes d'évaluation

L'algorithme de Viterbi permet d'identifier une marche qui produit une séquence de façon la plus probable dans un modèle de Markov caché. Une autre question, plus simple, consiste à calculer la probabilité d'émission d'une séquence donnée dans un modèle de Markov caché. On utilise également le principe de la programmation dynamique pour réaliser ce calcul, en remplissant un tableau α de taille $k \times n$. Ce tableau contient les probabilités $\alpha(i, j)$ de produire le préfixe $w_1 \dots w_i$ dans le modèle H en arrivant dans l'état j.

```
1  VITERBI(H , w)
2  Début
3      Soit k la taille de w et n la taille de S
4      Soient δ[1..k, 1..n] et b[1..k, 1..n] deux nouveaux tableaux bidimensionnels
5      Soit z un tableau d'entiers de taille k
6      Pour j = 1 à n
7          δ[1, j] = P(j) × E(j, w_1)
8          b[1, j] = 0
9      Fin Pour
10     Pour i = 2 à k
11         Pour j = 1 à n
12             δ[i, j] = E(j, w_i) × max_{ℓ∈S} {δ[i − 1, ℓ] × T(ℓ, j)}
13             b[i, j] = arg max_{ℓ∈S} {δ[i − 1, ℓ] × T(ℓ, j)}
14         Fin Pour
15     Fin Pour
16     z[k] = arg max_{ℓ∈S} {δ[k, ℓ]}
17     Pour i = k à 2
18         z[i − 1] = b[i, z[i]]
19     Fin Pour
20     Renvoyer z
21 Fin
```

FIGURE 3 – Algorithme de Viterbi pour le décodage des séquences dans les modèles de Markov cachés. La notation $\arg\max_{\ell \in S}\{f(l)\}$ signifie « une valeur de ℓ dans S qui maximise la fonction $f(l)$ ».

Question 36 : En vous appuyant sur la question 6, donner la probabilité que le modèle H_1 émette la séquence AAG.

Question 37 : Proposer des modifications de l'algorithme de Viterbi pour calculer le tableau α. Indiquer notamment s'il faut enlever ou rajouter des structures de données ou des phases. Justifier.

Question 38 : Comment trouver la probabilité de la séquence w à partir du tableau α renvoyé ?

L'algorithme modifié s'appelle l'algorithme *Forward*. Il existe un autre algorithme permettant de calculer la probabilité d'une séquence produite par un modèle de Markov caché en remplissant un tableau avec des probabilités partielles. Il s'agit de l'algorithme *Backward*, donné en figure 4.

Question 39 : Préciser les différences entre l'algorithme *Forward* et l'algorithme *Backward*. Notamment, que contient le tableau β ?

Question 40 : Indiquer comment trouver la probabilité de la séquence w produite par le modèle H à partir de la sortie de l'algorithme *Backward*.

Question 41 : Donner la complexité en temps et en espace de l'algorithme *Forward* et de l'algorithme *Backward*.

Question 42 : Donner l'implémentation d'une méthode `backward` correspondant à l'algorithme ci-dessus.

Tournez la page S.V.P.

```
BACKWARD(H , w)
Début
    Soit k la taille de w et n la taille de S
    Soit β[1..k, 1..n] un nouveau tableau bidimensionnel
    Pour j = 1 à n
        β[k, j] = E(j, w_k)
    Fin Pour
    Pour i = k − 1 à 1
        Pour j = 1 à n
            β[i, j] = 0
            Pour ℓ = 1 à n
                β(i, j) = β(i, j) + E(j, w_i) × β(i + 1, ℓ) × T(j, ℓ)
            Fin Pour
        Fin Pour
    Fin Pour
    Pour j = 1 à n
        β(1, j) = P(j) × β(1, j)
    Fin Pour
    Renvoyer β
Fin
```

FIGURE 4 – Algorithme Backward permettant l'évaluation des séquences dans un modèle de Markov caché.

Partie IV. Pour aller plus loin

Le modèle de Markov caché généré à partir d'un alignement permet de bien capturer la structure globale des séquences qui composent l'alignement, mais ne permet pas nécessairement de représenter des séquences proches de façon satisfaisante. Cette partie a pour objectif d'améliorer la génération du modèle de Markov caché à partir d'un alignement.

8 Les *pseudo-counts*

Un modèle de Markov caché construit à partir d'un alignement est souvent trop restreint dans son expressivité, et des séquences proches de celles de la famille ne peuvent pas être produites, à cause de sa structure. Ainsi, dans le modèle H_2 produit à partir de l'alignement de la figure 2, la séquence AATTCA, très proche de la première séquence, ne pourrait être produite car la probabilité d'émission de A en l'état 1 est nulle, du fait de n'avoir observé que des G dans cette colonne. Pour pallier ce problème, on introduit de légères perturbations dans les probabilités d'émission, les *pseudo-counts*, définis de la façon suivante, avec n le nombre d'états du modèle et pour tout $i \in \{1, \ldots, n\}$, en notant n_i le nombre de caractères c tels que $E(i, c) = 0$. Alors, pour tout $c \in \Sigma$:

— si $E(i, c) = 0$, remplacer $E(i, c)$ par $\frac{1}{n \times n_i}$,

— sinon, remplacer $E(i, c)$ par $\frac{n-1}{n} E(i, c)$.

Question 43 : Démontrer que les contraintes classiques sur les modèles de Markov restent respectées.

Question 44 : Que devient le modèle H_2 lorsque l'on introduit les *pseudo-counts* ? On demande uniquement la matrice de probabilités d'émission modifiée.

Question 45 : Modifier l'implémentation du constructeur `fromAlignement` de la classe `HMM`, écrit à la question 25, pour intégrer les *pseudo-counts*.

9 Les zones mal alignées

Un autre problème peut se poser avec les alignements de séquences pour initialiser les modèles de Markov cachés : certaines colonnes successives contiennent des *gaps*, ce qui indique un endroit où l'alignement est de mauvaise qualité localement à ces colonnes, que l'on va qualifier de « mal alignées ». Lorsque l'on connaît un intervalle $[i, i+t]$ de colonnes mal alignées (avec $t > 0$), on peut remplacer les t états correspondant par un seul état, appelé état d'insertion, dans lequel les probabilités de transition et d'émission sont calculées globalement. Ainsi, dans l'exemple de la figure 5, l'état d'insertion couvre les trois colonnes 4, 5 et 6, dans lesquelles la fréquence d'apparition des A est $\frac{1}{5}$, la fréquence d'apparition des C est $\frac{2}{5}$, la fréquence d'apparition des G est $\frac{1}{5}$ et la fréquence d'apparition des T est $\frac{1}{5}$. Pour ce qui est des transitions, on compte dans cet état :

— une transition de la colonne 4 à la colonne 5 (interne)

— une transition de la colonne 5 à la colonne 6 (interne)

— une transition de la colonne 6 à la colonne 7 (externe)

— deux transitions de la colonne 4 à la colonne 7 (externes),

soit des probabilités de transition de l'état d'insertion à lui-même égale à $\frac{2}{5}$ et de l'état d'insertion à l'état suivant égale à $\frac{3}{5}$.

```
A C A - - - A T G
T C A A C T A T C
A C A C - - A G C
A G A - - - A T C
A C C G - - A T C
```

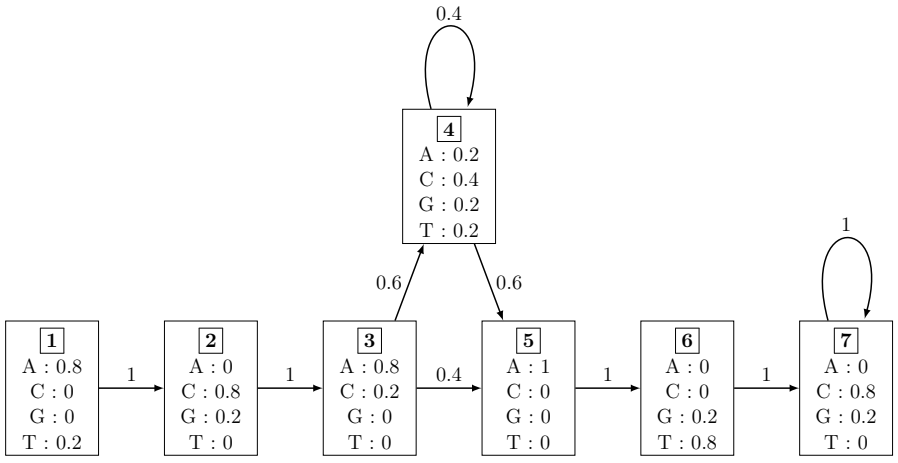

FIGURE 5 – À gauche, un alignement de séquences présentant une zone mal alignée au niveau de l'intervalle de colonnes $[4, 6]$. À droite, le modèle de Markov caché (sans *pseudo-counts*) généré à partir de cet alignement, et de l'intervalle des colonnes mal alignées.

Tournez la page S.V.P.

Remarque : une colonne isolée avec des tirets, comme la colonne 5 de l'alignement de la figure 2, n'a pas besoin d'être transformée en état d'insertion (elle est déjà un état d'insertion).

Question 46 : Si l'on part du principe qu'une colonne mal alignée est une colonne qui contient au moins un tiret, appliquer ce principe à l'alignement de la figure 2 et donner le graphe correspondant au modèle de Markov ainsi généré.

Question 47 : Donner un algorithme qui prend en entrée un alignement de séquences, et renvoie en sortie une liste d'intervalles de colonnes, dans laquelle les colonnes mal alignées ont été fusionnées. Par exemple, cet algorithme doit renvoyer la liste :

```
[(0, 0), (1, 2), (3, 3), (4, 4), (5, 5)]
```

à partir de l'alignement de la figure 2 et la liste :

```
[(0, 0), (1, 1), (2, 2), (3, 5), (6, 6), (7, 7), (8, 8)]
```

à partir de l'alignement de la figure 5. Quelle est la complexité en temps dans le pire des cas de cet algorithme ?

Question 48 : Donner l'implémentation de cet algorithme sous forme d'une ou plusieurs fonction(s).

Question 49 : Modifier le constructeur de la classe `HMM` qui prend en entrée un alignement, et une liste d'intervalles de colonnes comme construite à la question précédente. Ce constructeur doit construire le modèle en tenant compte de ces colonnes mal alignées.

* *
*