Segmentation de l'ADN : un modèle HMM

Résumé

Ce projet propose un modèle de HMM à deux états pour modéliser les séquences ADN. Dans le but de séparer les portions codantes des portions non codantes, il présente les algorithmes de Viterbi et Forward-Backward. L'algorithme EM est introduit pour l'estimation de la matrice de transition et des lois d'émission.

1 Structure des séquences ADN

Les organismes vivants sont constitués de *cellules* qui se divisent et se spécialisent, construisant ainsi un individu presque toujours unique. Toutes ces cellules contiennent le plan de cette construction, information codée dont le support est le *génome*. Celui-ci est constitué de cellules d'*ADN* (Acide Désoxirybo-Nucléique) et d'*ARN* (acide Ribo-Nucléique), des molécules qui constituent donc le support de l'hérédité. La vie est possible grâce à leur capacité à se *répliquer* presque à l'identique.

Une molécule d'ADN est un polymère de bases désoxirybonucléiques appelées *nucléotides* disposées séquentiellement selon une structure en double hélice. Chaque nucléotide est constitué d'un groupe phosphate, d'un sucre et d'une base azotée. Il existe quatre types de bases azotées : l'Adénine, la Cytosine, la Thymine et la Guanine. Les technologies récentes permettent le séquençage du génome de plus en plus d'espèces. On dispose ainsi d'énormes bases de données : pour donner un ordre de grandeur, un chromosome humain comporte entre cinquante et deux cent cinquante millions de bases). Analyser cette masse d'information constitue donc un défi titanesque, qui ne peut être relevé que par des méthodes largement automatisées. Dans cet objectif, une première étape cruciale est d'isoler les briques d'information à l'intérieur des chromosomes.

Pour cela, on peut s'appuyer sur quelques connaissances biologiques de la structure d'une molécule d'ADN. Au niveau local, on sait qu'elle est constituée de blocs de trois bases successives, appelés *codons* parce que chacun commande la production d'un des 20 acides aminés dont sont constituées les protéines, à l'exception d'un codon stop qui commande la fin d'une protéine. Au niveau global, elle se subdivise *gènes*, séquences contigües de codons commandant à la création de protéines qui vont remplir les différentes fonctions vitales de l'organisme. Mais le chromosome n'est pas pas seulement une juxtaposition de gènes : on trouve entre deux gènes successifs des portions dites *non codantes* ou *intergéniques*, dont les fonctions sont encore mal connues.

Lors de la réplication, il arrive que des modifications aient lieu dans les séquences de nucléotides. Ces modifications ont en général des conséquences catastrophiques dans les portions codantes : quand les molécules fabriquées ne sont plus les bonnes, il y a toutes les chances pour que leur fonction ne soit plus correctement remplie et que s'ensuive la disparition de l'individu qui les porte (sauf quand, très rarement, une modification reste viable voire positive pour l'individu, et permet ainsi l'évolution de l'espèce). Par contre, dans les portions non codantes, de telles variations portent moins à conséquence, et peuvent perdurer de génération en génération.

On peut donc s'attendre à ce que les molécules ADN soient constituées de successions de portions aux propriétés statistiques distinctes. La suite de ce texte présente une méthode rudimentaire visant à utiliser ces différences pour séparer portions codantes et ADN intergénique.

2 Un modèle simple de chaînes de Markov cachées à deux états

Considérons le modèle simpliste suivant : l'information génétique contenue par un chromosome est une séquence $Y_1^n = (Y_1, \dots, Y_n) \in \mathbb{E}^n$, où n est le nombre de nucléotides et $\mathbb{E} = \{A, C, T, G\}$. Cette séquence se subdivise en une alternance de portions codantes et non codantes. On admettra que la première portion n'est jamais codante. Ensuite, on passe à chaque symbole d'une portion codante à une portion non codante avec probabilité p, et dans l'autre sens d'une portion non codante à une portion codante avec probabilité q.Dans les portions codantes, les bases Y_j sont des variables aléatoires i.i.d. dont la distribution sur $\mathbb{E}$ sera notée $g(1, \cdot)$. Dans les portions non codantes, les bases sont indépentantes et ont une loi $g(2, \cdot)$ sur l'ensemble $\mathbb{E}$.

Bien sûr, ce modèle n'est pas réaliste, mais certains de ses raffinements sont utilisés par les généticiens pour l'analyse automatique des énormes séquences obtenues par séquençage ces dernières années (voir l'article de Pierre Nicolas, Laurent Bize, Florence Muri, Mark Hoebeke, François Rodolphe, S. Dusko Ehrlich, Bernard Prum, et Philippe Bessières : Mining Bacillus Subtilis Chromosome Heterogeneities Using Hidden Markov Models, Nucleic Acid Research 30, 2002, p 1418-1426).

Pour $j = 1, \dots, n$, introduisons la variable aléatoire X_j valant 1 si Y_j est dans une portion codante et 2 sinon : dans notre modèle la suite $(X_j)_j$ forme une chaîne de Markov, dont la matrice de transition est :

$$Q = \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix}.$$

Conditionnellement à $X_j = x_j \in \{1,2\}$, la loi de Y_j est $g(x_j,.)$: le processus $(X_j,Y_j)_j$ est appelé *chaîne de Markov cachée* ou *HMM* (pour Hidden Markov Model). Les variables X_j, à valeur dans l'ensemble $\mathcal{H} = \{1,2\}$, sont appelées *états cachés*, parce que l'expérimentateur n'a en général accès qu'aux variables Y_j. Remarquons que, conditionnellement à la suite des états cachés, les bases Y_j sont indépendantes. Dans le cadre de notre modèle, séparer les zones codantes et non codantes revient à estimer les états cachés X_j grâce aux variables Y_j.

Soit $y_1^n \in \mathbb{E}^n$ la séquence observée. Pour trouver la position des gènes, une première approche naturelle est de rechercher la séquence d'états cachés $\hat{X}_1^n$ *la plus probable* conditionnellement à la séquence observée Y_1^n. Mais on peut aussi être plus ambitieux et vouloir connaître la loi conditionnelle de X_j sachant $Y_1^n = y_1^n$, c'est-à-dire savoir avec quelle probabilité la j-ème base appartient à une portion codante compte tenu de la séquence tout entière.

Ces deux tâches semblent a priori complexes : en effet la probabilité d'observer y_1^n s'écrit

$$\mathbb{P}\left(Y_1^n = y_1^n\right) = \sum_{x_1^n \in \mathcal{H}^n} \delta_2(x_1) g(2,y_1) \prod_{j=2}^{n} Q_{x_{j-1},x_j} g(x_j,y_j). \tag{1}$$

Dans le premier problème, on cherche quel terme de la somme est le plus grand ; une recherche brutale est tout simplement inenvisageable dès que la taille du message grandit un peu. Pour le deuxième problème, il semble très difficile de trouver une écriture simple pour $\mathbb{P}(X_j = 1 | Y_1^n = y_1^n)$. Il est pourtant possible de résoudre très efficacement ces problèmes, grâce d'une part à l'algorithme de Viterbi, et d'autre part l'algorithme Forward-Backward.

3 L'algorithme de Viterbi

L'algorithme de Viterbi permet, pour une chaîne de Markov cachée de paramètres connus, de calculer la suite $\hat{x}_1^n$ des états cachés la plus probable ayant conduit à une observation y_1^n donnée. Il fournit un bon exemple de *programmation dynamique* : on utilise la solution de sous-problèmes pour construire la solution globale. Pour $x \in \mathcal{H}$ et i compris entre 1 et n, soit

$$M_j(x) = \max_{x_1^{j-1} \in \mathcal{H}^{j-1}} \mathbb{P}\left(X_1^{j-1} = x_1^{j-1} \cap X_j = x \cap Y_1^j = y_1^j\right).$$

1. On commence par remarquer que $M_1(x)$ est connu pour tout x : c'est simplement $\delta_2(x) g(x,y_1)$.
2. On peut ensuite calculer de proche en proche les $M_j(x)$ grâce à la relation de récurrence :

$$M_j(x') = \max_{x \in \mathcal{H}} M_{j-1}(x) Q(x,x') g(x',y_j). \tag{2}$$

   Au cours de ce calcul, on garde en mémoire

$$\pi_j(x') = \arg\max_{x \in \mathcal{H}} M_{j-1}(x) Q(x,x') g(x',y_j).$$

   Remarquer que pour éviter que les $M_j(x)$ ne deviennent trop petits quand j augmente, on peut les renormaliser (par exemple, en faisant en sorte que leur somme vaille 1) : cela ne modifie pas le résultat de l'algorithme, tant que pour j fixé tous les $\left(M_j(x)\right)_x$ sont multipliés par la même valeur.
3. Enfin, la séquence des états cachés les plus probables est obtenue en remontant le tableau à l'envers, grâce aux relations :

$$\left\{ \begin{array}{lcl} \hat{x}_n & = & \arg\max_{x \in \mathcal{H}} M_n(x), \text{ et} \\ \hat{x}_j & = & \pi_{j+1}\left(\hat{x}_{j+1}\right). \end{array} \right.$$

Grâce à cet algorithme, on peut traiter avec **Matlab** des séquences de plusieurs centaines de milliers de bases en quelques secondes.

4 L'algorithme Forward-Backward

L'algorithme Forward-Backward, qui a une certaine parenté avec l'algorithme de Viterbi, permet quant à lui de calculer pour tout j la loi de l'état caché X_j conditionnellement à l'observation $Y_1^n = y_1^n$. Comme son nom l'indique, il se décompose en deux étapes :

1. **Etape Forward : filtrage.** On calcule pour tous les x de $\mathcal{H}$:

$$\begin{cases} \phi_j(x) &= \mathbb{P}\left(X_j = x | Y_1^j = y_1^j\right), \text{ et} \\ c_j &= \mathbb{P}\left(Y_j = y_j | Y_1^{j-1} = y_1^{j-1}\right). \end{cases}$$

Cela se fait facilement de façon itérative : d'une part $P_1(x)$ et Q_1 sont très simples à calculer, et d'autre part on a les relations de récurrence :

$$\begin{aligned} \mathbb{P}\left(X_{j+1} = x' \cap Y_{j+1} = y_{j+1} | Y_1^j = y_1^j\right) &= \sum_{x \in \mathcal{H}} \mathbb{P}\left(X_j = x \cap X_{j+1} = x' \cap Y_{j+1} = y_{j+1} | Y_1^j = y_1^j\right) \\ &= \sum_{x \in \mathcal{H}} \mathbb{P}\left(X_j = x | Y_1^j = y_1^j\right) \times \mathbb{P}\left(X_{j+1} = x' | X_j = x\right) \times \mathbb{P}\left(Y_{j+1} = y_{j+1} | X_{j+1} = x'\right), \quad (3) \end{aligned}$$

et

$$\mathbb{P}\left(Y_{j+1} = y_{j+1} | Y_1^j = y_1^j\right) = \sum_{x' \in \mathcal{H}} \mathbb{P}\left(X_{j+1} = x' \cap Y_{j+1} = y_{j+1} | Y_1^j = y_1^j\right). \quad (4)$$

2. **Etape backward : lissage**. On calcule pour tous les x de $\mathcal{H}$:

$$\beta_j(x) = \frac{\mathbb{P}\left(Y_{j+1}^n = y_{j+1}^n | X_j = x\right)}{\mathbb{P}\left(Y_{j+1}^n = y_{j+1}^n | Y_1^j = y_1^j\right)}.$$

Ce calcul peut s'effectuer par récurrence descendante, en partant de $j = n$, si l'on utilise les c_j calculés précédemment et le fait que :

$$\begin{aligned} \mathbb{P}\left(Y_{j+1}^n = y_{j+1}^n | X_j = x\right) &= \sum_{x' \in \mathcal{H}} \mathbb{P}\left(Y_{j+1}^n = y_{j+1}^n \cap X_{j+1} = x' | X_j = x\right) \\ &= \sum_{x' \in \mathcal{H}} \mathbb{P}\left(X_{j+1} = x' | X_j = x\right) \times \mathbb{P}\left(Y_{j+1} = y_{j+1} | X_{j+1} = x'\right) \times \mathbb{P}\left(Y_{j+2}^n = y_{j+2}^n | X_{j+1} = x'\right). \quad (5) \end{aligned}$$

Il reste seulement à remarquer que les probabilités recherchées se déduisent immédiatement des valeurs ainsi calculées, par la relation :

$$\mathbb{P}\left(X_j = x | Y_1^n = y_1^n\right) = \phi_j(x) \times \beta_j(x). \quad (6)$$

Là encore, une machine peu puissante suffit pour traiter des séquences ADN de plusieurs milliers voire millions de bases.

5 L'algorithme EM

Notons $\theta = (A, g)$ le vecteur regroupant tous les paramètres définissant la chaîne de Markov cachée discrète. L'idée de l'algorithme EM est, à partir d'une estimation $\hat{\theta}^0$ (idée que l'on a a priori du paramètre θ inconnu), de construire par récurrence une suite d'estimateurs $(\hat{\theta}^j)_j$ par une suite d'itérations se décomposant en deux phases :

1. **Phase E** (Expectation)

$$f(\theta | \hat{\theta}^n) = \mathbb{E}_{P_{\hat{\theta}^j}(\cdot|\mathbf{y})}\left[\log P_\theta(\mathbf{X})\right].$$

2. **Phase M** (Maximisation)

$$\hat{\theta}^{j+1} = \arg\max_\theta f(\theta | \hat{\theta}^j).$$

Pour notre problème, les espérances conditionnelles $\mathbb{E}_{P_{\hat{\theta}^j}(\cdot|\mathbf{y})}\left[\log P_\theta(\mathbf{X})\right]$ s'expriment simplement en fonction des quantités calculées par les algorithmes de filtrage/lissage décrit plus haut. En particulier, on remarquera que

$$\mathbb{P}(X_j = x, X_{j+1} = x' | Y_1^n = y_1^n) = \phi_j(x) Q(x, x') g(x', y_{j+1}) \beta_{j+1}(x'). \quad (7)$$

6 Etude abstraite de la convergence

Pour comprendre l'intérêt de l'algorithme EM, introduisons l'*information de Küllback-Leibler* $D(q|r)$ entre les deux lois de probabilités q et r définies sur l'ensemble $\{1,\dots,k\}$:

$$D(q|r) = \mathbb{E}_q\left[\log \frac{q(X)}{r(X)}\right] = \sum_{j=1}^{k} q(j)\log\frac{q(j)}{r(j)}.$$

On prouve aisément que $D(q|r) \geqslant 0$, et que l'on a égalité si et seulement si $q = r$.

Remarquons que, quel que soit $\theta' \in [0,1]$, la log-vraisemblance du paramètre θ pour l'observation $\mathbf{y}$ s'écrit :

$$\log P_\theta(Y=y) = \mathbb{E}_{P_{\theta'}(\cdot|y)}\left[\log P_\theta(X)\right] - \mathbb{E}_{P_{\theta'}(\cdot|y)}\left[\log P_\theta(X|y)\right].$$

Ainsi,

$$\begin{aligned}
\log P_{\hat\theta^{j+1}}(Y=y) - \log P_{\hat\theta^j}(Y=y) &= \mathbb{E}_{P_{\hat\theta^j}(\cdot|y)}\left[\log P_{\hat\theta^{j+1}}(X)\right] - \mathbb{E}_{P_{\hat\theta^j}(\cdot|y)}\left[\log P_{\hat\theta^{j+1}}(X|y)\right] \\
&\quad -\mathbb{E}_{P_{\hat\theta^j}(\cdot|y)}\left[\log P_{\hat\theta^j}(X)\right] + \mathbb{E}_{P_{\hat\theta^j}(\cdot|y)}\left[\log P_{\hat\theta^j}(X|y)\right] \\
&= \mathbb{E}_{P_{\hat\theta^j}(\cdot|y)}\left[\log P_{\hat\theta^{j+1}}(X)\right] - \mathbb{E}_{P_{\hat\theta^j}(\cdot|y)}\left[\log P_{\hat\theta^j}(X)\right] \qquad (8) \\
&\quad +\mathbb{E}_{P_{\hat\theta^j}(\cdot|y)}\left[\log P_{\hat\theta^j}(X|y)\right] - \mathbb{E}_{P_{\hat\theta^j}(\cdot|y)}\left[\log P_{\hat\theta^{j+1}}(X|y)\right] \qquad (9) \\
&\geqslant 0.
\end{aligned}$$

En effet, la première différence (8) est positive par définition de $\hat\theta^{j+1}$. D'autre part, la différence (9) est égale à $D\left(P_{\hat\theta^n}(\cdot|y)|P_{\hat\theta^{j+1}}(\cdot|y)\right)$. Cela suffit à montrer que la suite des vraisemblances est croissante. On voit aisément que la limite est nécessairement un maximum local de la fonction de vraisemblance.

7 Suggestions et développements

1. Discuter le modèle proposé dans ce texte. Quelles lois suivent les tailles des portions codantes et non codantes dans notre modèle ? Que dire de la proportion de l'espace pris par les portions codantes dans des séquences suffisamment longues ?
2. Prouver les égalités (numérotées) et les affirmations énoncées dans le texte.
3. Quelle est la complexité des algorithmes présentés ici ?
4. Simuler des séquences ADN suivant le modèle présenté dans le texte. On pourra prendre par exemple $p = 1/1000$, $q = 1/300$, $g(2,\cdot)$ uniforme sur $\mathbb{E}$ et $g(1,\cdot)$ est donnée par

$$\left\{\begin{array}{lcl} g(1,A) &=& 0.31, \\ g(1,C) &=& 0.19, \\ g(1,T) &=& 0.27, \\ g(1,G) &=& 0.23. \end{array}\right.$$

5. Implémenter l'algorithme de Viterbi et/ou l'algorithme Forward-Backward. Expérimenter sur les séquences simulées, et visualiser les résultats (on pourra utiliser la fonction `imagesc`). Estimer expérimentalement la probabilité que $\hat X_j$ soit égal à x_j. Même question avec l'estimateur $\tilde X_j$ qui maximise la vraisemblance a posteriori :
$$\tilde x_j = \arg\max_{x\in\mathcal{H}} \mathbb{P}\left(X_j = x | Y_1^n = y_1^n\right).$$
6. Programmer et illustrer le fonctionnement de l'algorithme EM présenté dans la section 5.
7. Prouver les affirmations contenues dans les sections 5 et 6.
8. Le fichier 'Bsub.mat' contient un morceau commenté de l'ADN de Bacillius Subtilis, une bactérie dont le génome a été beaucoup étudié. Grâce à la commande `load 'Bsub'`, vous récupérer les variables `y` et `x`, deux vecteurs de taille $n = 200000$ contenant respectivement la succession des bases (avec le codage : 1 pour A, 2 pour C, 3 pour T, 4 pour G) et la région correspondante (1 pour 'codant', 2 pour 'non codant'). Tester les méthodes de segmentation par les algorithmes de Viterbi et Forward-Backward. Visualiser et commenter les résultats obtenus.