Chapitre II
Alignement par paires

Greg

27 septembre 2006

1 Introduction

Analyser des séquences revient principalement à déterminer si elles sont liées ou pas. Pour répondre à cette question, la première chose à faire est de les aligner puis de décider si l'alignement est biologiquement significatif (car les deux séquences sont liées) ou non (l'alignement est le produit de la chance). Il faut prendre en compte les points suivants :

- Le type d'alignement (local, global)
- Le modèle de score
- L'algorithme pour trouver l'alignement optimal (ou le meilleur)
- La méthode statistique pour évaluer la signification du score d'alignement

Rapellons encore que les paires de résidus similaires ont des valeurs positives dans les matrices de substitutions.

2 Le modèle de score

Lorsqu'une protéine diverge d'un ancêtre, il peut y avoir des substitutions, des insertions ou des deletions. Le processus natuel influence les subsitutions en en préférant certaines à d'autres.

Le score total est la somme des résidus alignés plus des termes pour chaque gaps. De plus, on s'attend à ce que des identités et des substitutions conservatives soient plus probables dans un alignement que l'on s'y attendrait par

chance ; on s'attend à voir moins fréquemment des mutations non-conservatives dans un alignement que par chance.

Tous les algorithmes d'alignement sont basés sur un système de score additif, car on suppose qu'une muation en un endroit est indépendante d'une mutation en un autre endroit de la séquence.

Notations

Soient 2 séquences $x = x_1 \dots x_i \dots x_n$ et $y = y_1 \dots y_j \dots y_m$ de longueur respectives n et m

2.1 Matrices de substitutions

Etant donné une paire de séquences alignées, on souhaite assigner un score à cette alignement qui donne une mesure de la vraissemblance relative que les 2 séquences soient liées par opposition au fait qu'elles ne soient pas liées. Il faut donc donner une probabilité à ces 2 cas.

Le modèle aléatoire ou non-lié est le plus simple : il suppose que les résidus apparaissent avec une fréquence q_a et ce independemment. La probabilité des 2 séquences vaut :

$$P(x, y|R) = \prod_i q_{x_i} \cdot \prod_j q_{y_i} \tag{1}$$

Cette probabilité est la probabilité que les séquences x et y soient alignées en sachant qu'elles ne sont pas liées (cette probabilité est faible).

Par opposition, on a le modèle match ou modèle M dans lequel les paires de résidus alignés apparaissent avec une proabbilité p_{ab} (c'est la probabilité que les résidus a et b dérivent d'un résidus c commun). Ainsi la probabilité d'un alignement vaut :

$$P(x, y|M) = \prod_i p_{x_i y_i} \tag{2}$$

Cette probabilité est la probabilité que les séquences x et y soient alignées en sachant qu'elles sont liées (cette probabilité est faible). Si on fait le rapport entre le modèle M et le modèle R, on a :

$$\text{odd ratio} = \prod_i \frac{p_{x_i y_i}}{q_{x_i} \cdot q_{y_i}} \tag{3}$$

Si ce nombre est élevé, alors c'est que les deux séquences s'alignent parce-qu'elles sont liées. S'il est faible, Si on prend le logarithme de tout cela, le produit se transforme en somme

$$S = \sum_i s(x_i, y_i) \tag{4}$$

avec

$$s(a, b) = \log\left(\frac{p_{ab}}{q_a q_b}\right) \tag{5}$$

$s(a, b)$ est le score contenu dans les matrices de substitution. Ce score est arrondi à l'entier le plus proche dans les matrices.

2.2 Pénalités de gap

On veut pénaliser les gaps. Soit g la longueur du gap, d la pénalité d'ouverture du gap et e la pénalité d'extension. Il y a alors deux méthodes pour calculer la pénalité de gap :

$$\gamma(g) = -gd \tag{6}$$

$$\gamma(g) = -d - (g-1) \cdot e \tag{7}$$

En général $e > d$.

3 Algorithmes d'alignement

Etant donné un système de score, il nous faut maintenant un algorithme qui trouve un alignement optimal entre 2 séquences. Si les 2 séquences ont la même longueur, alors il n'y a qu'une seule possibilité pour un alignement global sans gap. Si on ajoute la possibilité de mettre des gaps, ou que les séquences ne sont pas de même longueur, alors il y a trop de possibilités pour toutes les tester. Nous allons alors utiliser la programmation dynamique, garante d'un alignement optimal. Pour aller encore plus vite, il y a des méthodes heuristiques, mais qui ne rendent pas forcément l'alignement optimal.

Pour trouver l'alignement optimal il est nécessaire de maximiser le score. Le but d'un algorithme d'alignement est d'incorporer autant que faire se peut des paires de résidus de score positif dans l'alignement, tout en minimisant le nombre de paires dont les résidus ne seraient pas conservés ainsi que le nombre de gaps (qui pénalisent le score).

3.1 Alignement global : Needleman-Wunsch

On veut un alignement global entre 2 séquences, en autorisant les gaps.

L'idée est de construire une matrice F avec comme index i (colonnes) et j (lignes) pour chacune des séquences et $F(i,j)$ est le score du meilleur alignement des i premiers résidus de la première séquence avec les j premiers résidus de la seconde séquence.

On commence par dire que $F(0,0) = 0$. Ensuite, on rempli la matrice avec la formule suivante :

$$F(i,j) = max \begin{cases} F(i-1,j-1) + s(x_i, y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d. \end{cases} \tag{8}$$

Si on choisi le premier cas, alors x_i et y_i sont alignés, dans le second cas, x_i est aligné avec un gap et dans le dernier cas y_i est aligné avec un gap. On garde un pointeur venant de la case qui a donné le score choisi pour $F(i,j)$. Comme conditions initiales, il faut encore dire que $F(i,0) = -id$ et $F(0,j) = -jd$ sinon on ne peut pas lancer l'algorithme (il y aurait des cases indéfinies).

Le score du meilleur alignement est alors dans la case $F(n,m)$. Pour construire l'alignement, on remonte de (n,m) à une des cases voisine $(i-1,j-1), (i-1,j)$ ou $(i,j-1)$ de laquelle a été dérivée la case (i,j). Si on choisi $(i-1,j-1)$, on ajoute x_i et y_i au début de l'alignement. Si on choisi $(i-1,j)$ on ajoute x_i et "-" et si on choisi $(i,j-1)$ on ajoute y_i et "-".

Cet algorithme a une complexité en temps et en espace de l'ordre de $O(nm)$.

3.2 Alignement local : Smith-Waterman

On cherche le meilleur alignement entre deux sous-séquences de x et y ; cela se produit quand deux séquences sont suspectées de partager un même domaine. C'est aussi utile pour trouver une relation d'évolution entre deux protéines qui ont hautement divergé (en effet, parfois il n'y a qu'une seule région qui est conservée, et donc ce n'est pas détectable par un alignement global si

beaucoup de mutations ont été accumulées dans le reste de la séquence).

$$F(i,j) = max \begin{cases} 0, \\ F(i-1,j-1) + s(x_i, y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d. \end{cases} \tag{9}$$

Il y a deux différences avec l'algorithme d'alignement global :

1. Une cellule $F(i,j)$ peut contenir 0 si toutes les autres options sont négatives.
2. Un alignement peut se terminer en n'importe quelle case de la matrice.

Assigner 0 à une case revient à débuter un nouvel alignement : aligner les paires qui donnent un score négatif est aussi bon que de recommencer un nouvel alignement en ce point. Maintenant on cherche la plus grande valeur dans la matrice F et on remonte jusqu'à trouver une valeur 0.

Pour que l'algorithme marche, il faut que les longues séquences de matches aléatoires donnent un score négatif. Cela permet à l'algorithme d'introduire un 0 et de recommencer un alignement. Si ce n'était pas le cas, alors l'alignement obtenu serait quasiement un alignement global.

3.3 Repeated matches

Il est possible que, si les séquences sont très longues, plusieurs alignements locaux différents donnent un score significatif. Dans ce cas on aimerait bien connaître tous ces alignements. Supposons que l'on soit interessé uniquement aux matches dont le score est plus grand qu'une tolérence T (cela permet d'éliminer les alignements insignifiants... il est en effet toujours possible de trouver un alignement mais qui est biologiquement insignifiant au-dessous d'un certain seuil).

Soit $y = PAWHEAE$ une séquence pour laquelle on cherche les répétitions dans la séquence $x = HEAGAWGHEE$. Tout x doit être aligné avec des sous-séquences de y

```
HEAGAWGHEE
HEA.AW-HE.
```

Un "-" signifie que le x_i correspond à un gap dans y_i. Un "." signifie que le x_i correspond à aucune sous-séquence de y. Chaque sous-séquence matchée de x est séparée par une ou plusieurs sous-séquences non-matchées.

$F(i,j)$ avec $j > 0$ est la meilleure somme des scores de $x_{1...i}$ en considérant que x_i est dans une region de match. $F(i,0)$ est la meilleure somme des scores de matches complets de la sous-séquence $x_{1...i}$ en considérant que x_i est dans une region non-matchée. Exemple, pour $F(4,0)$ est le score total de la première région matchée `HEA`

$$F(i,0) = max \begin{cases} F(i-1,0), \\ F(i-1,j), \quad j = 1,\ldots,m \end{cases} \tag{10}$$

$$F(i,j) = max \begin{cases} F(i,0), \\ F(i-1,j-1) + s(x_i,y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d. \end{cases} \tag{11}$$

On rempli la matrice colonne par colonne. La première case ce chaque colonne vaut la case précédente $(i-1,0)$ s'il n'y a pas de valeur supérieure à T dans la colonne précédente $(i-1,j)$ avec $j = 1 \ldots m$. Dans le cas contraire, la case $(i,0)$ vaut le score maximum de la colonne précédente moins T.

3.4 Overlap matches

Il est possible qu'une des 2 séquences contienne entièrement l'autre séquence ou alors que les 2 séquences se chevauchent. Ici nous voulons qu'un match commence soit au sommet soit à gauche de la matrice et qu'il finisse à droite ou en bas de la matrice.

On initialise la matrice F par : $F(i,0) = 0$ et $F(0,j) = 0$ puis on utilise la formule de récurrence de l'alignement global, car c'est un alignement global que l'on veut ici. On choisi la valeur maximale parmi le bord droit et la dernière ligne de la matrice puis on remonte jusqu'à la première ligne ou la première colonne de la matrice.

La différence avec l'algorithme d'alignement global réside dans les conditions initiales. Pour un alignement global, la première ligne et la première colonne ne sont pas nulles (mais contiennent des valeurs décroissantes) et donc on finira toujours par tomber sur la case $(0,0)$ ce qui n'est pas le cas pour les overlap matches. Un alignement ne peut pas commencer ou finir à l'interieur de la matrice car on veut aligner un suffix ou un préfixe de x avec un suffix ou un préfixe de y.

On rempli la matrice colonne par colonne.

4 Programmation dynamique avec des modèles plus complexes

Nous avons considéré un modèle de gap uniquement multiple de la longueur du gap ce qui est pénalisant au niveau biologique car les gaps, quand ils apparaissent, sont souvent de longueur supérieure à un résidu. Dans ce cas, la relation de récurrence devient :

$$F(i,j) = max \begin{cases} F(i-1,j-1) + s(x_i, y_j), & \\ F(k,j) + \gamma(i-k) & k = 0, \ldots, i-1 \\ F(i,k) + \gamma(j-k) & k = 0, \ldots, j-1 \end{cases} \quad (12)$$

La complexité en temps est de $O(n^3)$ ce qui est prohibitif !

4.1 Alignement avec un modèle de score de gap affiné

Le but est d'utiliser la fonction $\gamma(g) = -d-(g-1)\cdot e$ car cela nous ramène à deux cas possibles : soit on a une ouverture de gap, on la pénalise par d, sinon on pénalise par e. Il faut alors être capable de se souvenir où est l'ouverture du gap ! Il faut alors être capable de distinguer 3 types d'alignements : les alignements qui se terminent sans gap, les alignements qui se terminent avec un gap aligné à x_i et les alignements qui se terminent avec un gap aligné à y_j. Raison pour laquelle on aura besoin de 3 matrices (respectivement M, I_x et I_y). Chaque entrée de ces matrices va déprendre des entrées précédentes dans les autres matrices. Par exemple, le score de l'alignement optimal entre $x_1 \ldots x_i$ et $y_1 \ldots y_j$ qui se termine par un gap aligné avec x_i (se trouve donc dans la matrice $I_x(i,j)$) est obtenu soit en ouvrant un gap (donc le gap serait le premier en réalité) et donc prend la valeur $M(i-1,j)-d$ ou $I_y(i-1,j)-d$, soit en continuant un gap existant $I_x(i-1,j)-e$

$$M(i,j) = max \begin{cases} M(i-1,j-1) + s(x_i, y_j), & \text{précédemment un match} \\ I_x(i-1,j-1) + s(x_i, y_j) & \text{précédemment un gap en } y_i \\ I_y(i-1,j-1) + s(x_i, y_j) & \text{précédemment un gap en } x_i \end{cases} \quad (13)$$

$$I_x(i,j) = max \begin{cases} M(i-1,j) - d, & \text{précédemment un match, donc on ouvre un gap} \\ I_x(i-1,j) - e & \text{précédemment un gap en } y_i\text{, donc on le continue} \end{cases}$$

$$I_y(i,j) = max \begin{cases} M(i,j-1) - d, & \text{précédemment un match, donc on ouvre un gap} \\ I_y(i,j-1) - e & \text{précédemment un gap en } y_i\text{, donc on le continue} \end{cases}$$

Pour remonter, on met des pointeurs comme précédemment (les pointeurs peuvent sauter d'une matrice à l'autre) et on commence à remonter depuis la plus grande valeur, toutes matices confondues.

Les équations ci-dessus peuvent être représentées par un automate à états finis (FSA pour finite state automaton). Cela montre un état pour chacune des 3 matrices avec, en guise de transition l'incrémentation du score lors du passage d'un état à un autre. Un alignement correspond à un chemin au travers des états. La Figure 2.10 du livre montre un chemin pour une partie d'un alignement. Le passage de l'état M à l'état I_x correspond à l'ouverture d'un gap puis on reste dans l'état I_x ce qui signifie qu'on étend le gap.

5 Algorithmes d'alignement heurustiques

Les algorithmes s'alignements considérés jusqu'à présents sont garants de trouver le meilleur alignement étant donné un système de score. Cependant ils ne sous pas rapides. C'est pourquoi nous avons développé des algorithmes heuristiques qui sont plus rapides, mais moins sensibles et ne trouvent pas toujours la solution optimale.

5.1 BLAST

Pour une séquence donnée, on découpe cette séquence en petits mots m_i de taille fixe. On cherche, pour chacun de ces mots, quels sont les mots voisins. Pour se faire on compare tous les mots m_i à tous les mots possibles de même longueur (générés théoriquement) au moyen d'une matrice de substitution. On ne garde que les mots voisins ayant un score supérieur à une certaine tolérence T. Ensuite on regarde dans une base de données quelles sont les séquences contenant les mots voisins. On essaye ensuite d'aligner la séquence trouvée avec la séquence de départ.

5.2 FASTA

Permet de trouver un alignement local entre 2 séquences. La première étape consiste à localiser tous les mots de longueur l qui matchent parfaitement entre les 2 séquences. Ensuite, on se réfère à la Fig. 1.

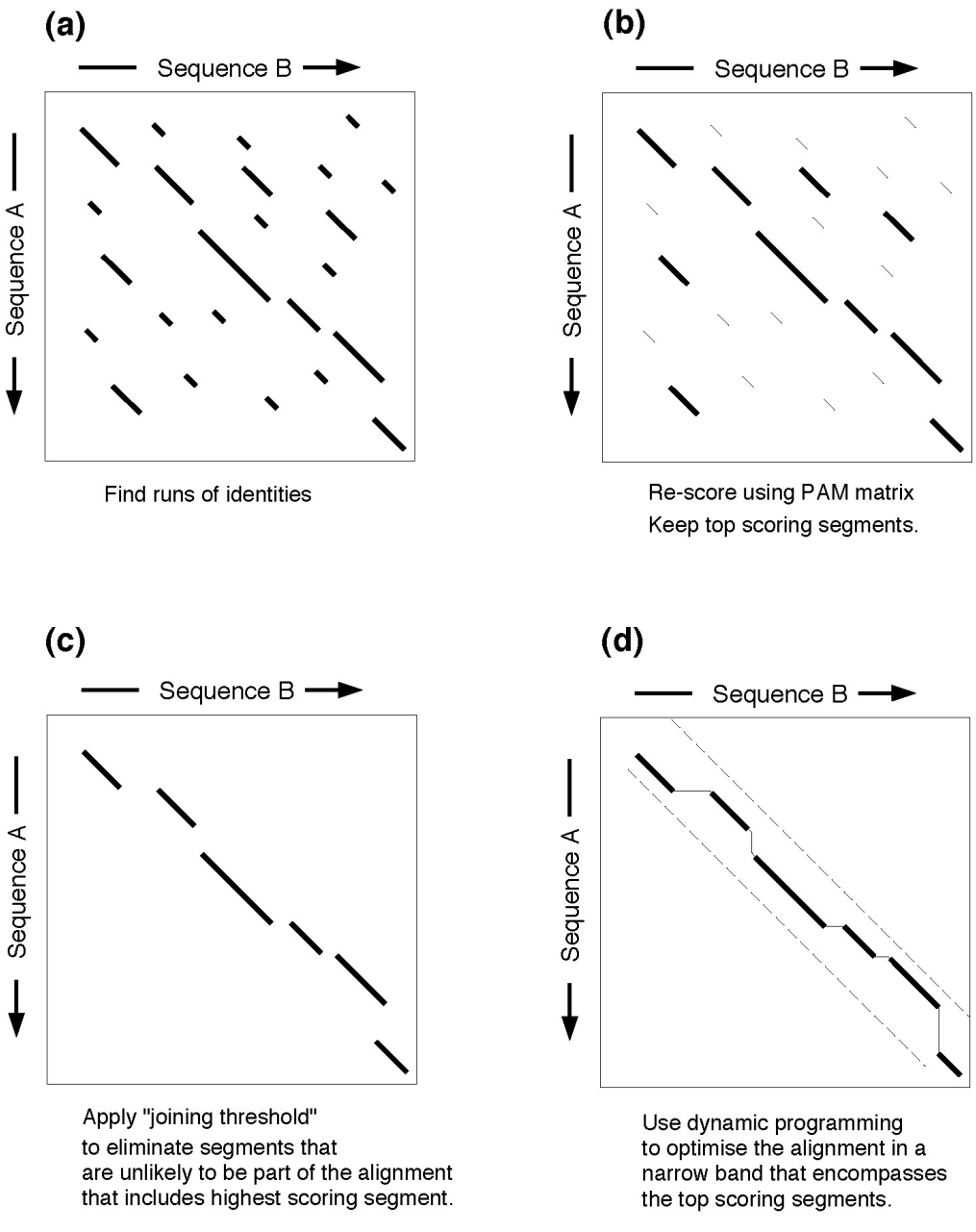

FIG. 1 – Détail de l'algorithme FASTA

6 Alignements en espace linéaire

Tous les algorithmes vus jusqu'à présent ont une complexité en espace de $O(mn)$. Cela peut être problématique si une (ou les deux) séquence(s) font plusieurs miliers de résidus. Heureusement il est possible de réduire la complexité en esapce $O(m+n)$ en ne faisant que doubler le temps de calcul. C'est ce qu'on appele la méthode d'*esapce linéaire*.

Comme la récurrence sur $F(i,j)$ est locale (on a besoin de que cases voisines), on peut jetter les cases de distance supérieure à 1. Cependant il ne sera plus possible de remonter dans la matrice. Il nous faut une autre approche : Diviser pour conquérir.

Supposons que l'on cherche un alignement global sans gap. Suppsons que l'on puisse connaître un v tel que (u,v) soit sur l'alignement optimal. On peut alors séparer le problème en $(0,0)$ à (u,v) et (u,v) à (m,n). L'alignement optimal sera alors la concaténation des 2 alignements des 2 sous-matrices.

7 Signification des scores

Comment peut-on vérifier la pertinence des scores associés aux alignements ? Comment décider si l'alignement est biologiquement significatif ou non étant donné une évidence d'homologie, ou est-ce juste un alignement entre deux séquences totalement non-liées ?

7.1 Signification des scores

Ce que nous voulons réellement c'est la probabilité que les séquences soient liées par opposition au fait qu'elles ne le soient pas, ce qui représente $P(M|x,y)$ plutôt que $P(x,y|M)$. $P(M|x,y)$ peut être calculé par le théorème de Bayes.

Nous nous attendons à ce que les les séquences soient liées avant de les avoir vues. Cela se traduit par $P(M)$, la probabilité à priori que les séquence soient liées et donc que le modèle match soit correct et $P(R) = 1 - P(M)$ la probabilité à priori que le modèle aléatoire soit correct.

Soit

$$S' = S + \log\left(\frac{P(M)}{P(R)}\right)$$

où

$$S = \log\left(\frac{P(x, y|M)}{P(x, y|R)}\right)$$

est le log-odds score de l'alignement. Ensuite

$$P(M|x, y) = \sigma(S')$$

où

$$\sigma(x) = \frac{e^x}{1 + e^x}$$

qui est la fonction logistique. Voir Fig 2.12 du livre.

On sait que $S' = S + \log\left(\frac{P(M)}{P(R)}\right)$ et que $S = \log\left(\frac{P(x,y|M)}{P(x,y|R)}\right)$ qui est le score standard de l'alignement (log-odds ratio). Nous avons des nouvelles données ($P(M)$ et $P(R)$) et donc il faut ajouter au score standard le $\log\left(\frac{P(M)}{P(R)}\right)$ que l'on appelle prior-log-odds-ratio, d'où la première équation. On a donc :

$$S' = \log\left(\frac{P(x, y|M)}{P(x, y|R)}\right) + \log\left(\frac{P(M)}{P(R)}\right) = \log\left(\frac{P(x, y|M) \cdot P(M)}{P(x, y|R) \cdot P(R)}\right)$$

En voyant le graphique 2.12, on a que $\lim_{x\to\infty} \sigma(x) = 1$ et $\lim_{x\to-\infty} \sigma(x) = 0$ et que la probabilité que le match modèle soit correcte (donc que les séquences soient liées en sachant que x et y sont alignés) est fonction du score S'. On voit donc que le graphique nous donne la probabilité que le match modèle soit correct.

Dans le cas où on cherche un grand nombre d'alignements possibles pour un match significatif (recherche dans une base de données par exemple), il est clair que si le prior-odd-ratio est fixe, alors, même si toutes les séquences de la base de donées ne sont pas liées et que l'on cherche le plus grand nombre de matches, le nombre de matches significatifs par chance augmentera aussi ! Si on veut rester fixé, il faut que le prior-odd-ratio soit inversement proportionnel au nombre de séquences dans la base de données N. Cela a pour effet que pour maintenir un nombre fixe de faux positifs il faut comparer le score non plus avec 0, mais avec $\log(N)$.

7.2 Approche classique

$$P(M_N \leq x) \approx e^{-KNe^{\lambda(x-\mu)}}$$

où M_N est le score maximum sur une série de N variables normales indépendantes. K et λ sont des constantes. Cette équation est utile pour calculer la probabilité que le meilleur match d'une recherche sur un nombre N de séquences non liées soit plus grand que le score maximal observé S.

Pour un alignement local, le nombre de matches non liés avec un score plus grand que S est approximativement donné par :

$$E(S) = Kmne^{-\lambda S}$$

où λ vaut la racine positive de

$$\sum_{a,b} q_a q_b e^{\lambda s(a,b)} = 1$$

La probabilité qu'il y ait un match de score plus grand que S est alors :

$$P(x > S) = 1 - e^{-E(S)}$$