Vendredi 30 janvier 2004

Démo 3 – Programmation Dynamique 3

Jean-Eudes Duchesne

1. Retour matrices de pondérations : Modèle probabiliste

Représentation de l'ADN : Longue séquence de caractères aléatoires, indépendants et identiquement distribués.

→ Probabilité de trouver le caractère a en position i : $P\{S[i] = a\} = \pi_a$

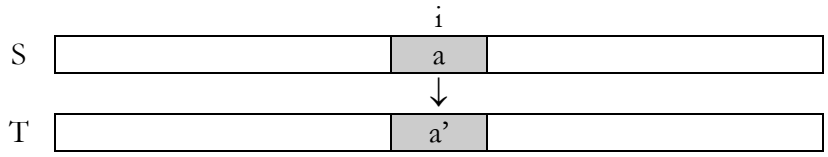
Modèle évolutif (pour passer de S à T) : Substitutions sont aléatoires et indépendantes

→ Probabilité de passer du caractère a dans S à a' dans T : $P\{T[i]=a' \mid S[i] = a\} = p_{a->a'}$

Une matrice de substitution est une matrice qui contient toutes les valeurs $p_{a->a'}$

Maintenant, quelle est la probabilité qu'un alignement observé est un « vrai » alignement ?

Considérons un alignement fictif :



La probabilité d'avoir un vrai alignement entre S et T à la position i dépend d'abord de la probabilité d'obtenir le caractère a à la position i de S et que par la suite, ce caractère se soit muté en a', donc :

$$\pi_a * p_{a->a'}$$

À quoi correspond un alignement dû au hasard ? Si l'on considère l'alignement de S et T, l'alignement de S[i] et T[i], s'il est dû au hasard, c'est la probabilité d'avoir le caractère a en S[i] et le caractère a' en T[i], donc :

$$\pi_a * \pi_{a'}$$

Ainsi, si nous sommes intéressés par la « chance » d'avoir un alignement significatif entre S[i] et T[i], il suffit de diviser la première probabilité par la deuxième ! (pour obtenir le « log odds » ou le « log des chances » il suffit de faire le log du résultat ci-dessous.

$$\frac{\pi_a p_{a\to a'}}{\pi_a \pi_{a'}} = \frac{p_{a\to a'}}{\pi_{a'}}$$

2. BLOSUM (Blocks Substitution Matrix)

BLOSUM correspond à une mesure de similarité entre les résidus de séquences. Par exemple BLOSUM62 correspond à 62% d'identités entre les séquences.

1) Utilise des alignements sans gaps entre des blocks conservés plutôt que des séquences complètes. (utilise une base de données de blocks alignés).
2) Pour chaque couple (A_i, A_j) calculer f(i, j) = nb (A_i, A_j) / nb total d'appariements
3) On arrange les blocks en groupes similaires (cluster) avec un seuil prédéterminé. Si le seuil est fixé à 62% alors les éléments des groupes auront au moins 62% d'homologie et la matrice finale sera appelée BLOSUM 62. Les fréquences obtenues pour les paires observées ne comptent que pour une fraction du groupe qui les contiennent. Ensuite, les groupes sont comptabilisés entre eux.

Avantages :

1) Moins dépendant du modèle d'évolution « aléatoire constant »
2) Plus précis que PAM. La paire remplacée la moins fréquente a été observée 2369 fois sur un total de 1.25×10^6 paires qui ont contribué aux résultats dans la matrice.

3. Chiaromonte (2002)

Le but est de créer une matrice de pondération qui peut bien représenter les nucléotides comme PAM et BLOSUM le font pour les acides aminés.

Motivation ? Dans l'ARN, d'autres paires que les paires canoniques sont observables (ex : paire G-U ou Wobble). Dans l'ADN, une matrice asymétrique peut représenter des différences dans le temps d'évolution de génomes (par exemple, le génome de la souris évolue plus rapidement, ce qui implique qu'un A humain apparié a un C de souris peut ne pas avoir la même contribution que le cas inverse).

La méthode combine les idées de PAM et BLOSUM :

- Utilise une méthode hybride entre les blocks et les séquences. Utilise une séquence où toutes les répétitions sont épurées (humain) et l'aligne localement à une autre séquence (souris). Les résultats avec un haut score sont utilisés pour les calculs subséquents.
- Utilise un seuil MAIS contrairement à BLOSUM seulement les alignements en dessous du seuil sont conservés. L'idée est de créer une matrice qui est capable de reconnaître les séquences moyennement conservées plutôt que les séquences très conservées. L'idée est que des séquences très conservées peuvent être trouvées par d'autres algorithmes tel que BLAST.
- Calcul d'un « log odd ratio » qui est ensuite normalisé pour que la plus grande valeur soit 100.

Notes :

- L'algorithme peut être modifié pour créer des matrices symétriques ou asymétriques.
- Résultat de la matrice dépend de la constitution en G+C.

4. BLAST (Basic Local Alignment Search Tool)

BLAST est une heuristique qui approche l'algorithme d'alignement local de programmation dynamique. Comme les bases de données contiennent des données chiffrées en milliards, la programmation dynamique avec une complexité en temps de O(nm) est trop lente pour parcourir des bases de données complètes (GenBank). Qui plus est, les ordinateurs normaux n'ont pas assez de mémoire pour contenir la table de programmation dynamique nécessaire à l'algorithme. Ainsi, il est nécessaire de faire des recherches dans les bases de données qui sont moins voraces. Idée ? Commencer par trouver des petites régions conservées et les agrandir.

BLAST en général :

1) Choisir un mot de taille w > 0. Ex : 7 ou 11 (11 est la valeur utilisée par défaut dans l'algorithme)
2) Comparer chaque sous mot de P avec ceux de T pour trouver les « hits » (petites régions similaires).
3) Étendre les « hits » pour trouver les alignements.

Choix de la taille de w :

Ceci représente un dilemme puisque plus w est grand, plus la chance de manquer des occurrences augmente. Par exemple, considérons l'alignement suivant :

S	a	c	c	g	a	t	t	t	a	c	g	t	a	a	c	c	g	g	c
	|	|	|		|	|	|	|	|	|	|	|	|	|		|	|	|	|
T	a	c	c	c	a	t	t	t	a	c	g	t	a	a	t	c	g	g	c

Un tel alignement, jugé très homologue à l'œil nu, ne peut être trouvé quand w = 11 puisque que l'alignement de contient pas 11 matchs consécutifs. Par contre, w = 7 nous permettrais de le trouver. Par contre si l'on réduit la taille de w alors on ralentit l'algorithme. En somme un petit w donne un algorithme plus lent mais plus sensible tendis qu'on s'attend à gagner en vitesse tout en perdant en sensibilité quand w augmente.

Méthode de comparaison :

Comme les comparaisons se font à une base de données, c'est donc dire que l'on possède T préalablement à la phase d'alignement. Nous profitons de ce fait pour préparer T à la recherche :

i. Pour tous les mots de taille w dans T, insérer le mot dans une table de hachage avec sont indice (début de la séquence).
ii. Pour tous les mots de taille w dans P, utiliser la fonction de hachage pour trouver les indices correspondants aux « hits » de taille w entre P et T.

Méthodes d'extension :

Extension rapide : pour chaque hit trouvé, étendre à gauche et à droite du hit en utilisant seulement les identités et les substitutions jusqu'à ce que le score tombe à 0. Ensuite calculer le meilleur alignement (PD) sur la région trouvée.

X-Drop : Calculer la PD mais seulement sur une région supérieure à A* - X où A* est la valeur du meilleur alignement obtenu jusqu'à présent.

PD dans une bande : Calculer la PD mais seulement dans une région définie par la diagonale sur laquelle se trouve le hit. Si on nomme cette diagonale D, alors la programmation dynamique sera effectuée dans la bande D-k à D+k ou k est arbitraire.