Alignement multiple de séquences

Thierry Lecroq
Université de Rouen

Problème

Aligner k séquences x_0, x_1, …, x_{k-1} de manière optimale.

Interêts biologiques

- Inférer des relations d'évolution entre espèces
- Déduire des similarités de fonction ou de structure entre des séquences de nucléotides ou d'acides aminés
- Trouver des caractéristiques communes à une famille de protéines ou de séquences d'ARN.

Représentation de familles de séquences

- Séquence consensus : séquence de longueur n contenant, à chaque position, le symbole le plus fréquent à la même position dans l'alignement
- Matrice consensus : matrice card $A \times n$ contenant la fréquence d'apparition de chaque symbole à chaque position de l'alignement
- Motif : expression rationnelle décrivant l'ensemble des séquences ou une partie particulièrement conservée

Matrice consensus

alignement

```
a c g – t
a c a c t
a g g c –
g c – c g
```

	C_0	C_1	C_2	C_3	C_4
a	0,75	0	0,25	0	0
c	0	0,75	0	0,75	0
g	0,25	0,25	0,5	0	0,25
t	0	0	0	0	0,5
-	0	0	0,25	0,25	0,25

Alignement d'une séquence avec une matrice consensus

Généralisation de l'alignement de 2 séquences

Exemple

a	**a**	**c**	**-**	**c**	**g**
C_0	**-**	C_1	C_2	C_3	C_4

Soient *Sub* une matrice de substitution et *M* une matrice consensus.

Le score de l'appariement entre le symbole $x[i]$ de la séquence x et de la colonne C_j de la matrice M est :

$$score(x[i],C_j) = \sum_a Sub(x[i],a) \times M(a,C_j)$$

La valeur d'un alignement est alors égal à la somme des scores des appariements :

$$T(x,M) = \sum_{i,j} score(x[i],C_j)$$

Exemple

Sub	**a**	**c**	**g**	**t**	**-**
a	2	-3	-1	-3	-1
c	-3	2	-3	-1	-1
g	-1	-3	2	-3	-1
t	-3	-1	-3	2	-1
-	-1	-1	-1	-1	0

M	C_0	C_1	C_2	C_3	C_4
a	0.75	0	0,25	0	0
c	0	0,75	0	0,75	0
g	0,25	0,25	0,5	0	0,25
t	0	0	0	0	0,5
-	0	0	0,25	0,25	0,25

a	**a**	**c**	**-**	**c**	**g**
C_0	**-**	C_1	C_2	C_3	C_4

Programmation dynamique

$T(i,j)$ = valeur de l'alignement optimal entre $x[0..i]$ et $C[0..j]$:

$$T(i,-1) = \Sigma\, score(x[k], \text{-})$$

$$T(-1,j) = \Sigma\, score(\text{-}, C_k)$$

$$T(i,j) = \max \begin{cases} T(i\text{-}1,j\text{-}1) + score(x[i], C_j) \\ T(i\text{-}1,j) + score(x[i], \text{-}) \\ T(i,j\text{-}1) + score(\text{-}, C_j) \end{cases}$$

Complexité

- calcul d'une case de la table de programmation dynamique (hors initialisations marginales) : $O(\text{card } A)$
- calcul de la table : $O(\text{card } A \times |x| \times n)$

Score SP

Étant donné un alignement multiple A, l'alignement **induit** pour deux séquences x_i et x_j est la restriction de l'alignement à ces 2 séquences.

Les appariements de 2 **–** sont ignorés.

Fonction de score SP

SP : *sum of pairs*

Le score d'un alignement A est la somme des scores des alignements induits pour toutes les paires de séquences.

Exemple

x_1 **a a g a a – a**

x_2 **a t – a a t g**

x_3 **c t g – g – g**

En considérant la distance d'édition :

$d(x_1,x_2) = 4$, $d(x_1,x_3) = 5$, $d(x_2,x_3) = 5$

donc $SP(A) = 14$

Complexité

Pb : étant donné k séquences, trouver un alignement multiple avec un score SP minimal.

Il a été prouvé que ce problème est **NP-dur**.

Généralisation de l'algorithme de programmation dynamique pour l'alignement de 2 séquences : avec k séquences de longueur commune n, algorithme en $O(n^k)$: inapplicable dès que $k > 5$ et $n \approx 100$.

Heuristique pour l'alignement multiple avec score SP

But : produire un « bon » alignement multiple en temps polynomial.

Bon alignement : indiquer le taux d'erreur avec un alignement optimal.

Soit X un ensemble de séquences et T un arbre dont chaque nœud est étiqueté par une séquence de X.

Un alignement A de X est dit consistant avec T si, pour tout couple de séquences (x_i,x_j) reliées par une arête de T, l'alignement induit pour (x_i,x_j) a un score optimal.

Alignement consistant avec un arbre

La notion d'alignement consistant est très liée à l'arbre considéré.

Deux séquences non adjacentes ont, en général, un score induit supérieur à $d(x_i,x_j)$.

Théorème

Un alignement A de X consistant avec un arbre T peut être produit en temps polynomial.

Méthode

Étape 0

Choisir 2 séquences quelconques adjacentes dans l'arbre et former un alignement de score $d(x_i,x_j)$.

Étape 1

Choisir une séquence x non encore alignée, adjacente à une séquence x_i déjà alignée.

Soit x'_i la séquence correspondant à x_i dans l'alignement (avec des **-**).

Étape 1 (suite)

Former un alignement optimal entre x et x'_i, avec la règle supplémentaire qu'un appariement formé de deux trous a un score nul.

Soit x' la séquence correspondant à x dans l'alignement de x avec x'_i.

Étape 1 (suite)

- Si aucun nouveau trou n'est inséré dans x'_i alors rajouter x' à l'alignement déjà existant.
- Si un ou plusieurs trous ont été rajoutés aux positions j dans x'_i alors rajouter des trous aux positions j dans toutes les séquences de l'alignement.

Répéter l'étape 1 jusqu'à ce que l'alignement contienne toutes les séquences de X.

Complexité

On calcule k-1 alignements optimaux de 2 séquences.

Si les séquences sont de longueur m : $O(km^2)$

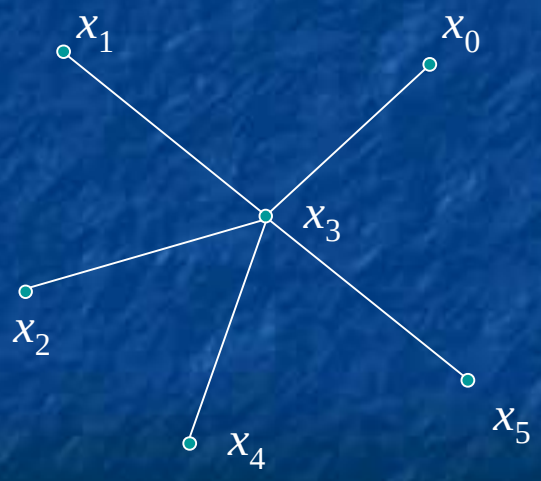
Alignement SP par la méthode de « l'étoile centrale »

Séquence centrale

Une séquence $x_c \in X$ est dite centrale si la valeur $M = \Sigma_i \, d(x_c,x_i)$ est minimale parmi toutes les séquences de X.

Arbre étoile (*center star*)

Arbre T_c de k nœuds, chaque nœud est étiqueté par une séquence distincte de X, la racine de l'arbre est x_c, et toutes les arêtes de T_c sont adjacentes à x_c.



Exemple

$X = \{ x_0, x_1, x_2, x_3, x_4, x_5 \}$

Notations

- A_c : alignement consistant avec T_c
- $s(x_i,x_j)$: valeur de l'alignement de x_i et x_j induit par l'alignement A_c ($s(x_i,x_j) \geq d(x_i,x_j)$ et $s(x_c,x_i) = d(x_c,x_i)$)
- $s(A)$: valeur de l'alignement A ($s(A_c) = \Sigma s(x_i,x_j)$)
- A* : un alignement optimal de X
- $d^*(x_i,x_j)$: valeur de l'alignement de x_i et x_j induit par l'alignement A* ($d(A^*) = \Sigma d^*(x_i,x_j)$)
- On suppose que d vérifie l'inégalité triangulaire

On veut montrer que $s(A_c)$ est inférieur au double de la valeur d'un alignement optimal :

$$s(A_c) \leq 2s(A^*)$$

Lemme 1

$s(x_i,x_j) \leq s(x_i,x_c) + s(x_c,x_j)$.

Preuve

$s(x_i,x_j) \leq s(x_i,x_c) + s(x_c,x_j) = d(x_i,x_c) + d(x_c,x_j) \leq d(x_i,x_j)$ par l'inégalité triangulaire. □

Théorème 2

$s(A_c)/s(A^*) \leq 2(k\text{-}1)/k < 2.$ □

Pour $k = 3$: $s(A_c) \leq 4/3\ s(A^*)$

Pour $k = 6$: $s(A_c) \leq 1{,}666\ s(A^*)$

Complexité

- Trouver la séquence centrale : $O(k^2m^2)$
- Produire un alignement A_c consistant avec T_c : $O(km^2)$

Programmation dynamique avec scrutage en avant

Pour l'alignement de 2 séquences : dès que $T(i,j)$ est calculé, les valeurs de $T(i,j+1)$, $T(i+1,j)$ et $T(i+1,j+1)$ sont mises à jour.

Graphe d'édition : les sommets du graphe sont toutes les paires (i,j). Les arcs relient (i,j) à $(i,j+1)$, $(i+1,j)$ et $(i+1,j+1)$. Le poids $p(v,w)$ d'un arc (v,w) est le coût de l'opération d'édition (insertion, suppression ou substitution) qui fait passer de v à w.

Notons $r(w)$ la valeur provisoire de $T(w)$.

Après le calcul de $T(v)$ on met à jour $r(w)$ avec $\min\{ r(w), T(v) + p(v,w) \}$.

La valeur $T(w)$ est la valeur $r(w)$ obtenue après que tous les voisins de w aient été considérés.

algo Carillo-Lipman(x,m,y,n)

$f \leftarrow \emptyset$

ENFILER(f, (0,0))

tantque $f \neq \emptyset$ **faire**

$(i,j) \leftarrow$ DÉFILER(f)

$T(i,j) \leftarrow r(i,j)$

si $j+1 < n$ **alors**

ENFILER(f, $(i,j+1)$)

$r(i,j+1) \leftarrow \min \{ r(i,j+1), T(i,j) + Ins(y[j]) \}$

si $i+1 < m$ **alors**

ENFILER(f, $(i+1,j)$)

$r(i+1,j) \leftarrow \min \{ r(i+1,j), T(i,j) + Sup(x[i]) \}$

si $i+1 < m$ et $j+1 < n$ **alors**

ENFILER(f, $(i+1,j+1)$)

$r(i+1,j+1) \leftarrow \min \{ r(i+1,j+1), T(i,j) + Sub(x[i],y[j]) \}$

Pour 3 séquences x, y et z de longueur n, le graphe d'édition a n^3 sommets et $7n^3$ arcs.

On veut éviter un certain nombre d'opérations par case, et surtout, éviter de considérer toutes les cases.

On note $d_{xy}(i,j)$ la valeur de l'alignement optimal de $x[i..n\text{-}1]$ et $y[j..n\text{-}1]$.

On définit de façon analogue $d_{xz}(i,\ell)$ et $d_{yz}(j,\ell)$.

Toutes ces valeurs peuvent être calculées en $O(n^2)$.

Le score SP pour les séquences $x[i..n\text{-}1]$, $y[j..n\text{-}1]$ et $z[\ell..n\text{-}1]$ est inférieur à $d_{xy}(i,j)+d_{xz}(i,\ell)+d_{yz}(j,\ell)$.

Supposons que l'on connaisse la valeur t d'un alignement multiple « suffisamment bon » (par exemple, l'alignement A_c consistant avec l'arbre étoile).

Observation clef : pour une case (i,j,ℓ), si $T(i,j,\ell)+d_{xy}(i,j)+d_{xz}(i,\ell)+d_{yz}(j,\ell) > t$ alors (i,j,ℓ) ne peut appartenir à aucun chemin optimal.

Aucun scrutage en avant n'est nécessaire pour une telle case.

Plus important, certaines cases ne sont jamais introduites dans la file *f*, et ne sont donc jamais considérées.

Cet algorithme réduit considérablement le nombre de cases à considérer.

Séquences consensus

Mot de Steiner

Étant donné un ensemble X de k séquences et une autre séquence y (appartenant ou non à X), l'erreur consensus de y relativement à X est la valeur $E_X(y) = \Sigma_i d(x_i,y)$.

Un mot de Steiner est un mot s^* qui minimise la valeur $E_X(s^*)$.

Un mot de Steiner reflète les caractéristiques communes aux séquences de X : séquence consensus.

Il n'y a pas de méthode exacte de calcul du mot de Steiner mais il existe des méthodes d'approximation.

Lemme 2

Il existe $x \in X$ tel que $E(x)/E(s^*) \leq 2 - 2/k < 2$. □

Théorème 3

$E(x_c)/E(s^*) \leq 2\text{-}2/k$.

Preuve

Lemme 2 et $E(x_c) = \Sigma\, d(x_c,x_i) \leq E(y)$ par définition de x_c. □

Étant donné un alignement A de X, le caractère consensus de A à la colonne j est le caractère a qui minimise la somme des distances de a à chaque caractère de la colonne. On note $g(j)$ cette somme minimale à la colonne j.

La séquence consensus de A est la séquence x_A obtenue en concaténant les caractères consensus de chaque colonne de A.

Pour la distance d'édition, le caractère consensus est le caractère le plus fréquent de la colonne (peut être un trou).

Une façon naturelle d'évaluer la pertinence d'un alignement A est de considérer sa séquence consensus x_A et de calculer l'erreur consensus $\Sigma_i\, d(x_A,x_i)$.

Erreur d'alignement de $x_A = \Sigma_j\, g(j)$.

Alignement multiple optimal relativement à la séquence consensus : alignement A dont la séquence consensus présente une erreur d'alignement minimale.

Méthode itérative

À chaque étape, on rajoute de nouvelles séquences à l'alignement courant : on crée un nouvel alignement en fusionnant 2 alignements de sous-ensembles X_1 et X_2 de X.

La différence entre les méthodes dépend de l'ordre d'insertion des séquences, et de la façon d'effectuer l'alignement.

Exemple

- Calculer tous les alignements 2 à 2
- Choisir l'alignement formé des 2 séquences dont la distance d'édition est minimale
- À chaque étape, choisir la séquence (non encore alignée) dont la distance d'édition avec une des séquences de l'alignement courant est minimale
- Rajouter cette séquence à l'alignement

Cette méthode peut être vue comme un alignement consistant avec un arbre de recouvrement minimal (*minimal spanning tree*, arbre formé du sous-ensemble d'arcs le moins coûteux maintenant une seule composante connexe).

Il y a plusieurs arbres de recouvrement possibles donc plusieurs alignements.

Prim ou Kruskal

Clustering

Cette méthode d'alignement multiple est très liée à la façon de choisir les alignements à fusionner. Dans quel ordre choisir les sous-alignements pour former des alignements plus grands ? Les méthodes classiques de clustering sont toutes utiles pour résoudre un tel problème. La méthode à utiliser dépend des données biologiques.

Variantes

Plutôt que de se baser sur l'alignement de 2 séquences, considérer la distance d'édition « moyenne » avec toutes les séquences des alignements. Par exemple, aligner deux matrices consensus. Si plus de temps de calcul est permis, commencer par calculer les distances 2 à 2 entre tous les alignements déjà obtenus et garder le meilleur.

CLUSTALW (Thompson, Higgins & Gibson 1994)

- Calcul des scores d'alignement de chaque paire de séquences. Convertir les scores d'alignement en distance d'évolution en utilisant le modèle de Kimura.
- Construire un arbre par une méthode de distance « Neighbour-Joining ».
- Utiliser cet arbre pour choisir les séquences à incorporer dans l'alignement multiple. Choisir les plus petites distances à chaque étape. L'algorithme effectue trois types différents d'alignement : entre 2 séquences, entre une séquence et une matrice consensus ou entre 2 matrices consensus.

La matrice de substitution est choisie en fonction de la similarité entre les séquences comparées :

- séquences très proches : BLOSUM80
- séquences éloignées : BLOSUM50

Différents scores pour les brèches, selon le type des résidus supprimés et la position des brèches.

Algorithme de Barton-Stenberg

Problème des méthodes décrites : l'alignement produit est directement lié aux séquences choisies. Les alignements partiels obtenus ne sont jamais modifiés.

Amélioration : après avoir obtenu un premier alignement multiple, retirer certaines séquences de l'alignement et les réaligner avec la matrice consensus de l'alignement multiple restant.

L'algorithme

- Calculer tous les alignements 2 à 2
- Choisir l'alignement de score maximal. Considérer la matrice consensus de l'alignement obtenu.
- À chaque étape, choisir une paire de séquences x_i et x_j de score maximal, où une seule des 2 séquences est dans l'alignement courant. Aligner la nouvelle séquence avec la matrice consensus de l'alignement courant. Mettre à jour la matrice consensus. Recommencer cette étape jusqu'à ce que toutes les séquences soient dans l'alignement.

L'algorithme (suite)

- Retirer la séquence x_0 et la réaligner avec la matrice consensus de l'alignement restant. Recommencer cette étape avec toutes les autres séquences $x_1, x_2, \ldots, x_{k-1}$.
- Répéter le processus un nombre fixé de fois ou jusqu'à ce que le score de l'alignement converge.

Calcul des scores

- Pour chaque paire de séquences, les symboles sont permutés aléatoirement, les séquences ainsi obtenues sont alignées, et le score (similarité) d'alignement est calculé.
- Effectuer le processus précédant une centaine de fois par paire de séquences et calculer moyenne et déviation.
- Score $sd(x_i,x_j)$: score d'alignement des 2 séquences divisé par la déviation calculée pour les 2 séquences.

Intuitivement, si x_i et x_j contiennent des structures non-aléatoires liées à la fonction de la protéine, alors le score d'alignement de x_i et x_j doit être beaucoup plus grand que celui des permutations de x_i et x_j.

Il a été empiriquement démontré que ce score est une bonne mesure de la qualité de l'alignement.

Alignement par points d'ancrage

Basée sur la recherche de motifs communs aux séquences.

- Rechercher un motif commun à l'ensemble ou une partie des séquences (suffisamment long et fréquent)
- Décaler les séquences de telle sorte à aligner les occurrences du motif
- Le problème est alors subdivisé en 2 sous-problèmes : aligner l'ensemble des préfixes des séquences à gauche du motif d'un côté et l'ensemble des suffixes des séquences à droite du motif

- Recommencer récursivement avec chacun des 2 sous-problèmes
- Les séquences ne contenant pas le motif sont alignées séparément, par une méthode utilisant le score SP et les 2 alignements ainsi obtenus sont ensuite fusionnés
- Lorsque les séquences ne contiennent plus de « bons » motifs, elles sont alignées par une méthode utilisant le score SP.

Plusieurs méthodes existent pour la recherche (extraction) exacte ou approchée de motifs.

L’utilisation des arbres de suffixes est fréquente.