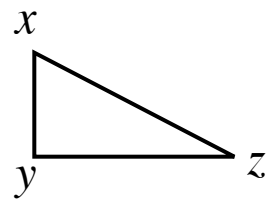
Distance d'édition

Pour comparer des séquences, on va définir une distance:

Définition: Une distance D est une relation ayant les propriétés suivantes:

$$D(x,x) = 0$$
$$D(x,y) = D(y,x)$$
$$D(x,z) \leq D(x,y) + D(y,z)$$



inégalité du triangle

Distance naturelle: compter le nombre d'insertions, de suppressions et de substitutions nécessaire pour passer d'une séquence à une autre.

Distance d'édition (suite)

Exemple: $S_1 = CATAGTG$ $S_2 = GTCAGGT$

S	Su	M	I	M	I	M	M	Su
C	A	T		A		G	T	G
G		T	C	A	G	G	T	

Distance d'édition entre S_1 et S_2 : Nombre minimal d'insertions, suppressions et substitutions nécessaire pour transformer S_1 en S_2

Une insertion/suppression est représentée par un '-':

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

Alignement global

Il existe plusieurs alignements possibles étant donné 2 séquences.

Comment trouver un alignement optimal i.e un alignement ayant une distance d'édition minimal??

Idée naïve: Énumérer tous les alignements possibles pour les 2 séquences et en choisir un dont la distance d'édition est minimal.

Exemple: $S_1 = AC$ et $S_2 = AGC$

Les alignements possibles ici sont:

$$\left\{\begin{matrix} AC- & A-C & -AC & AC-- & AC-- & A--C & A-C- & -A-C \\ AGC' & AGC' & AGC' & -AGC' & A-GC' & -AGC' & AG-C' & AGC-' \end{matrix} \cdots \right\}$$

Combien d'alignements?

Si f(n,m) est le nombre d'alignements entre une séquence $a = a_1 \ldots a_n$ de n lettres et une autre séquence $b = b_1 \ldots b_m$ de m lettres alors, on a:

$$f(0,0) = 1$$
$$f(0,m) = 1$$
$$f(n,0) = 1$$

$f(n,m) =$

Nombre d'alignements de $a_1 \ldots a_{n-1}$ avec $b_1 \ldots b_m$	a_n —
+	
Nombre d'alignements de $a_1 \ldots a_{n-1}$ avec $b_1 \ldots b_{m-1}$	a_n b_m
+	
Nombre d'alignements de $a_1 \ldots a_n$ avec $b_1 \ldots b_{m-1}$	— b_m

$$\Rightarrow f(n,m) = f(n-1,m) + f(n-1,m-1) + f(n,m-1)$$

Combien d'alignement?

$\Rightarrow$ f(n,m) = f(n-1,m) + f(n-1,m-1) + f(n,m-1)

	0	1	2	3	4	5
0	1	1	1	1	1	1
1	1	3	5	7	9	11
2	1	5	13	25	41	61
3	1	7	25	63	129	231
4	1	9	41	129	321	681

En fait, on peut montrer que

$$f(n,n) \sim (1+\sqrt{2})^{2n+1} \cdot \sqrt{n}$$

$$\Rightarrow f(1000,1000) \sim 10^{764}$$

Programmation dynamique

Étant donné 2 séquences, $S = s_1 s_2 \ldots s_m$ et $T = t_1 t_2 \ldots t_n$, on définit

D(i,j): distance d'édition entre le préfixe de taille i de S, $s_1 \ldots s_i$, et le préfixe de taille j de T, $t_1 \ldots t_j$.

D définit une matrice de taille (m+1) X (n+1) qu'on appelle la matrice de programmation dynamique

L'idée est alors d'exprimer D(i,j) en fonction des valeurs de D pour des paires d'indices plus petits que (i,j)

Calculer D(i,j) à partir des 3 cases (i-1,j), (i, j-1) et (i-1,j-1):

1. L'alignement se termine par la suppression de s_i

Alignement optimal de $s_1 \ldots s_{i-1}$ avec $t_1 \ldots t_j$		s_i
		—
D(i-1,j)	+	1

2. L'alignement se termine par l'insertion de t_j

Alignement optimal de $s_1 \ldots s_i$ avec $t_1 \ldots t_{j-1}$		—
		t_j
D(i,j-1)	+	1

3. L'alignement se termine par l'alignement de s_i avec t_j

Alignement optimal de $s_1 \ldots s_{i-1}$ avec $t_1 \ldots t_{j-1}$		s_i
		t_j
D(i-1,j-1)	+	1 si $s_i \neq t_j$
		0 sinon

Remplissage de la table

Conditions initiales: $D(i,0) = i, \quad \forall i \quad 0 \leq i \leq m$

$D(0,j) = j, \quad \forall j \quad 0 \leq j \leq n$

Relation de récurrence pour i, j > 0:

$$D(i,j) = \min \begin{cases} D(i-1,j) & +1 \\ D(i,j-1) & +1 \\ D(i-1,j-1) & +\delta(i,j) \end{cases},$$

Où $\delta(i,j) = 0$ si $x_i = y_j$ et 1 sinon.

Complexité: Pour remplir chaque case de la table, on examine 3 cases.
Il y a $O(nm)$ cases et donc complexité en temps de $O(nm)$

Trouver un alignement optimal

Au cours du remplissage de la table, garder des pointeurs:

- de (i-1,j) à (i,j) si D(i,j) = D(i-1,j) + 1
- de (i,j-1) à (i,j) si D(i,j) = D(i,j-1) + 1
- de (i-1,j-1) à (i,j) si D(i,j) = D(i-1,j-1) + $\delta(i, j)$

Un alignement optimal: Commencer à la case (m,n) et suivre des pointeurs jusqu'à la case (0,0).

Une case peut contenir plusieurs pointeurs: plusieurs alignements optimaux possibles

Distance versus similarité

Plutôt que de mesurer la différence entre 2 séquences, mesurer leur degré de similarité

P(a,b): score de l'appariement (a,b). Positif si a=b et $\leq$ 0 sinon

V(i,j): valeur de l'alignement optimal entre $s_1 \ldots s_i$ et $t_1 \ldots t_j$

Conditions initiales: $V(i,0) = \sum_{1 \leq k \leq i} P(s_k, -), \quad V(0,j) = \sum_{1 \leq k \leq j} P(-, t_k),$

Relation de récurrence:

$$V(i,j) = \max \begin{cases} V(i, j-1) & + P(-, t_j) \\ V(i-1, j) & + P(s_i, -) \\ V(i-1, j-1) & + P(s_i, t_j) \end{cases},$$

Algorithme de Needleman-Wunch

Recherche approché d'un motif

Problème: On a un "petit" motif P de taille m et une "longue" séquence T de taille n et on veut trouver toutes les occurrences approximatives de P dans T (moins de k erreurs).

		G	T	C	A	G	G	…
	0	0	0	0	0	0	0	…
C	1							
A	2							
T	3							

- initialiser la première ligne à 0
- même relations de récurrence que pour l'alignement global de séquences
- rechercher à la ligne m toutes les cases contenant des valeurs plus petites ou égales à k
- pour trouver un alignement, suivre les pointeurs jusqu'à la première ligne

Alignement local - Algorithme de Smith-Waterman

Relations de récurrence:

$$V(0,j) = 0$$
$$V(i,0) = 0$$
$$V(i,j) = \max \begin{cases} 0 \\ V(i-1,j) & +p(s_i,-) \\ V(i,j-1) & +p(-,t_j) \\ V(i-1,j-1) & +p(s_i,t_j) \end{cases}$$

- Le 0 dans la récurrence permet d'ignorer un nombre quelconque de caractères en début de séquence

- Pour trouver un alignement local de score maximal:
  - On remplie la table
  - On recherche une case c contenant la valeur maximale de la table
  - De cette case c, on suit les pointeurs jusqu'à une case contenant la valeur 0