Métodos de alineamiento

Bioinformática, 20-2-19

Elvira Mayordomo

Motivación: 2 razones para comparar secuencias biológicas

1. Los **errores y omisiones** en los datos biológicos producidos en la extracción y “lectura” de los mismos
2. El **proceso natural de la evolución**, que produce mutaciones en forma de inserciones, borrados y sustituciones

Motivación: la redundancia

- **Redundancia**: la evolución reutiliza los patrones con éxito que se copian y modifican
- Las secuencias similares tienen **funciones o estructura similar**
- Para saber **la función** de una secuencia nueva:
  - Se compara con todas las conocidas en la base de datos
  - Es mucho más eficiente que las pruebas experimentales
  - Cambio radical en la forma de trabajar de los biólogos
- Ejemplo de éxito: **los oncogenes** (similaridad de genes encontrados en virus que causan crecimiento incontrolado con proteínas que estimulan el crecimiento celular)

Contenido

- **Alineamiento de 2 strings (global, local y semiglobal)**
- Métodos heurísticos (para búsqueda en BdD)
- Alineamientos múltiples

Requirements

¿Son estas dos secuencias parecidas?

```
A       VLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKG
B       SLSAAQKDNVKSSWAKASAAWGTAGPEFFMALFDAHDDVFAKFSGLFSGAAKGTVKN
```

Requirements

¿Son estas dos secuencias parecidas?

```
A       VLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKG
B       SLSAAQKDNVKSSWAKASAAWGTAGPEFFMALFDAHDDVFAKFSGLFSGAAKGTVKN
         !!!! ! !!!  ! !        !       !         !            !
```

14 out of 57 (25 % of) amino acids are identical

Requirements

¿Son estas dos secuencias parecidas?

```
A       VLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHFD-LSHGSAQ--VKG
B       SLSAAQKDNVKSSWAKA---SAAWGTAGPEFFMALFDAHDDVFAKFSGLFSGAAKGTVKN
         !!!! ! !!!  ! !     !  !    !     !      !  !  !  ! !   !!
```

Insertion/deletions, two evolutionary events, must be taken into account

21 out of 60 (35 % of) positions are identical

Requirements

¿Son estas dos secuencias parecidas?

```
A       VLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHFD-LSHGSAQ--VKG
B       SLSAAQKDNVKSSWAKA---SAAWGTAGPEFFMALFDAHDDVFAKFSGLFSGAAKGTVKN
         !!!!.! !!!..!.!.   .! .!. . ! ..  ! .    ! .!. !  !.!.  !!.
```

38 out of 60 (63 % of) positions have the same or similar properties

Alineamiento de 2 strings

- **Definición:** Sean $s=s_1 \ldots s_m$ y $t=t_1..t_n$ dos strings sobre Σ. Sea h: $(\Sigma \cup \{-\})^* \rightarrow \Sigma^*$ la función "quita huecos"

  ($h(-)=\lambda$, $h(a)=a$ para $a \in \Sigma$)

Un <u>alineamiento</u> de s y t es un par de strings (s',t') que cumplen

1. h(s')=s h(t')=t |s'|=|t'|
2. No hay posiciones que contengan – tanto en s' como en t'

Ejemplo de alineamiento

- **s= GACGGATTATG**
- **t= GATCGGAATAG**

- Alineamiento
- **s’= GA-CGGATTATG**
- **t’= GATCGGAATA-G**

Columnas del alineamiento

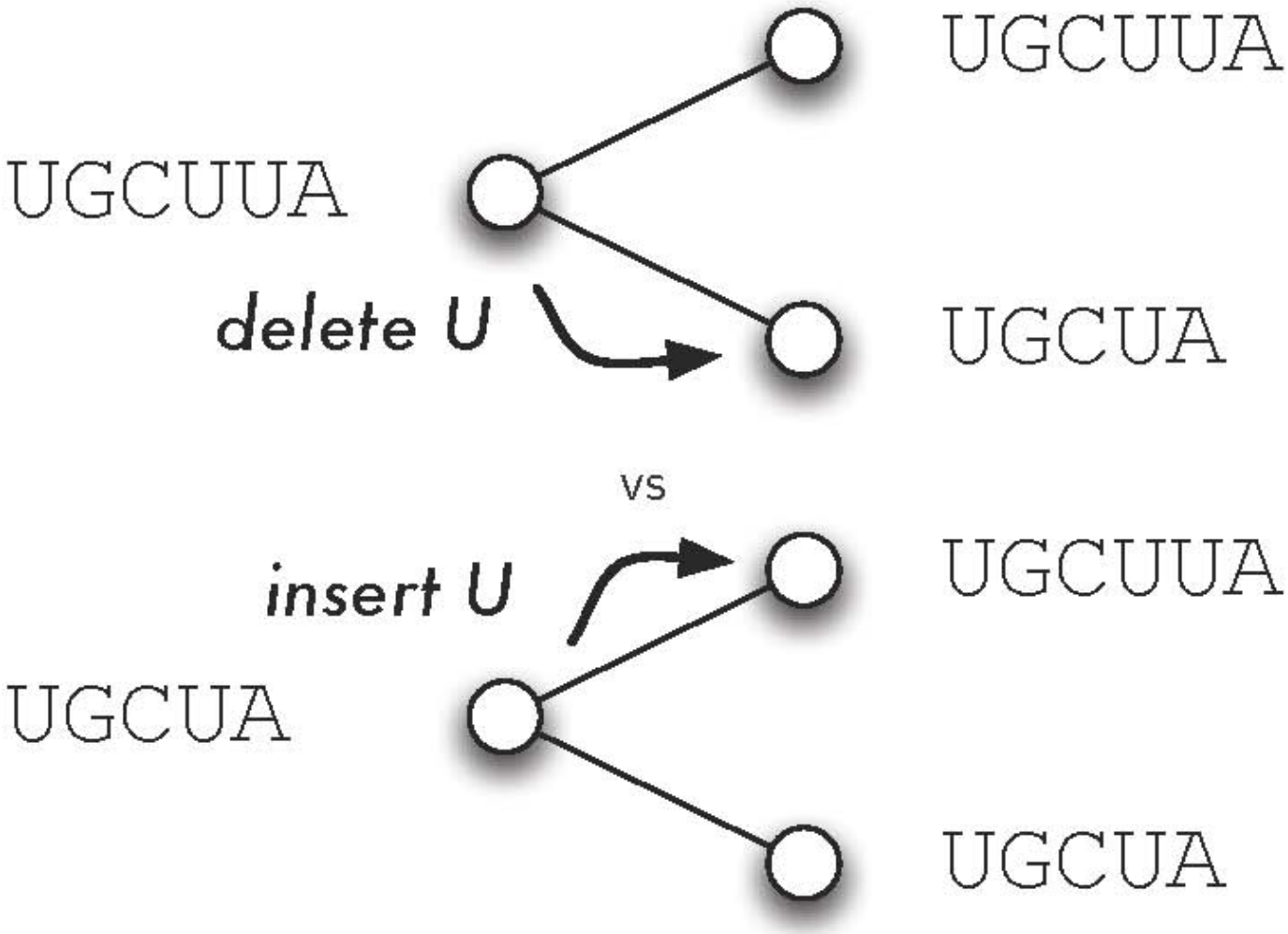
- **s′= GA-CGGATTATG**
- **t′= GATCGGAATA-G**
- Inserción
- Borrado
- Coincidencia
- Sustitución/desacuerdo

What's an indel?

Indel stands for **insertion or deletion**.
Given exactly two sequences, I am claiming that insertions cannot be distinguished from deletions, hence the use of the word indel. What do I mean?
Consider the following pairwise alignment, was the **U**, present in **S1**, deleted, to produce **S2**? Or, was is a **U** inserted into **S2** to produce **S1**?

```
S1 = UGCUUA
S2 = UGC-UA
```

What's an indel?



Puntuación de un alineamiento

- Queremos usar los alineamientos para medir la similaridad de 2 strings.
- Def. Dadas dos strings s, t.

  Dada $p(a,b) \in \mathbf{Q}$ para cada $a,b \in \Sigma$ y $g \in \mathbf{Q}$

La puntuación δ de un alineamiento (s',t') es

$\delta(a,-) = \delta(-,b) = g$, $\delta(a,b) = p(a,b)$ para $a,b \in \Sigma$

$$\delta(s',t') = \Sigma\, \delta(s'_i, t'_i)$$

Ejemplos de puntuación

- Distancia de edición o de Levenshtein

$p(a,b)=1$ si $a \neq b$, $p(a,a)=0$, $g=1$

- Muy usado: (el que usamos siempre hoy)

$p(a,b)=-1$ si $a \neq b$, $p(a,a)=1$, $g=-2$

Objetivo de optimización

- Según el significado de la puntuación anterior, los mejores alineamientos pueden ser los de puntuación grande o pequeña.
- Por defecto hoy nuestro objetivo será maximizar la puntuación

Similaridad de s y t

- Dados dos strings s, t y una puntuación de alineamiento δ,

$$sim_{\delta}(s,t) = máx \{\delta(s', t') \mid (s',t') \text{ es un alineamiento de } s, t\}$$

Alineamiento global

- Problema de determinar el alineamiento global óptimo de dos strings

Entrada: s,t strings y puntuación de alineamiento δ

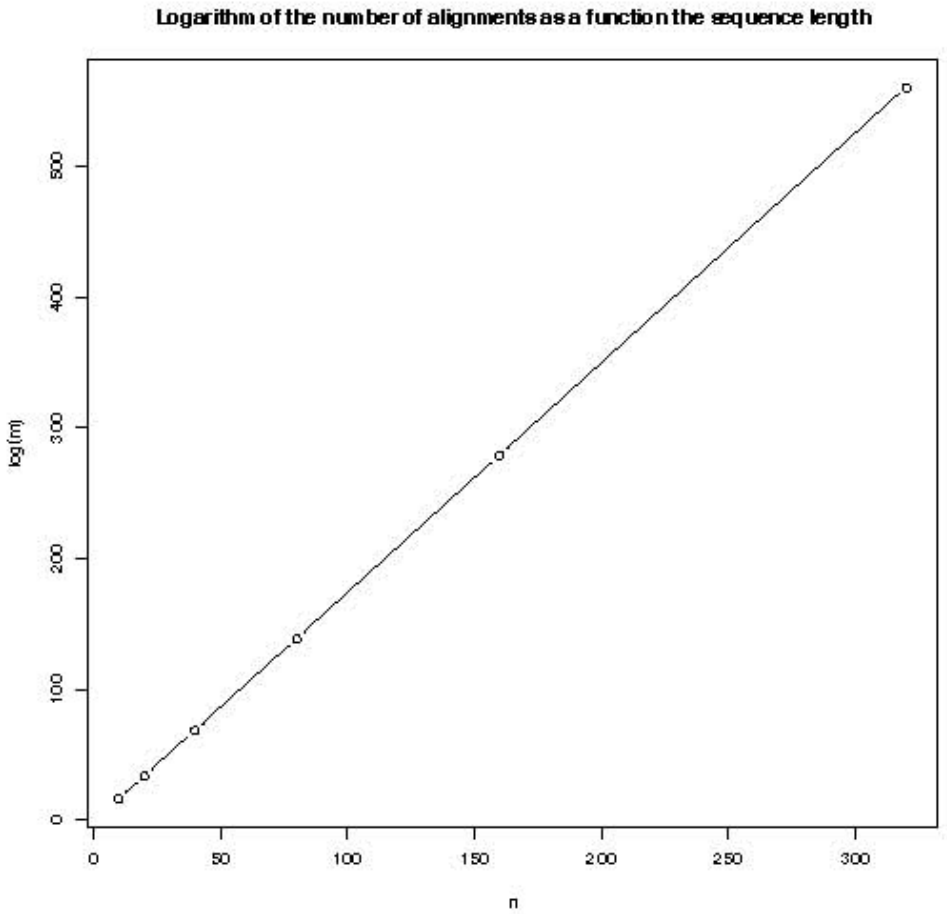
Solución: Calcular la puntuación del alineamiento óptimo $sim_{\delta}(s,t)$ y/o el alineamiento óptimo (s',t')

Ejemplo

- **s= AAAT**
- **t= AGT**

Size of the search space

Length	# alignments
10	8.097453e+06
20	2.605438e+14
40	3.781502e+29
80	1.121607e+60
160	1.392368e+121
320	3.031221e+243



Solución con programación dinámica

- $s=s_1 \ldots s_m$ y $t=t_1..t_n$
- Usamos la matriz de similaridad M que es (m+1)x(n+1)
- M(i,j) es la puntuación de una alineación óptima de $s_1 \ldots s_i$ y $t_1..t_j$
- M(m,n) es el objetivo ($sim_\delta(s,t)$)

Solución con programación dinámica

- La primera fila y columna son fáciles
- M(0,j)=gj, M(i,0)=gi

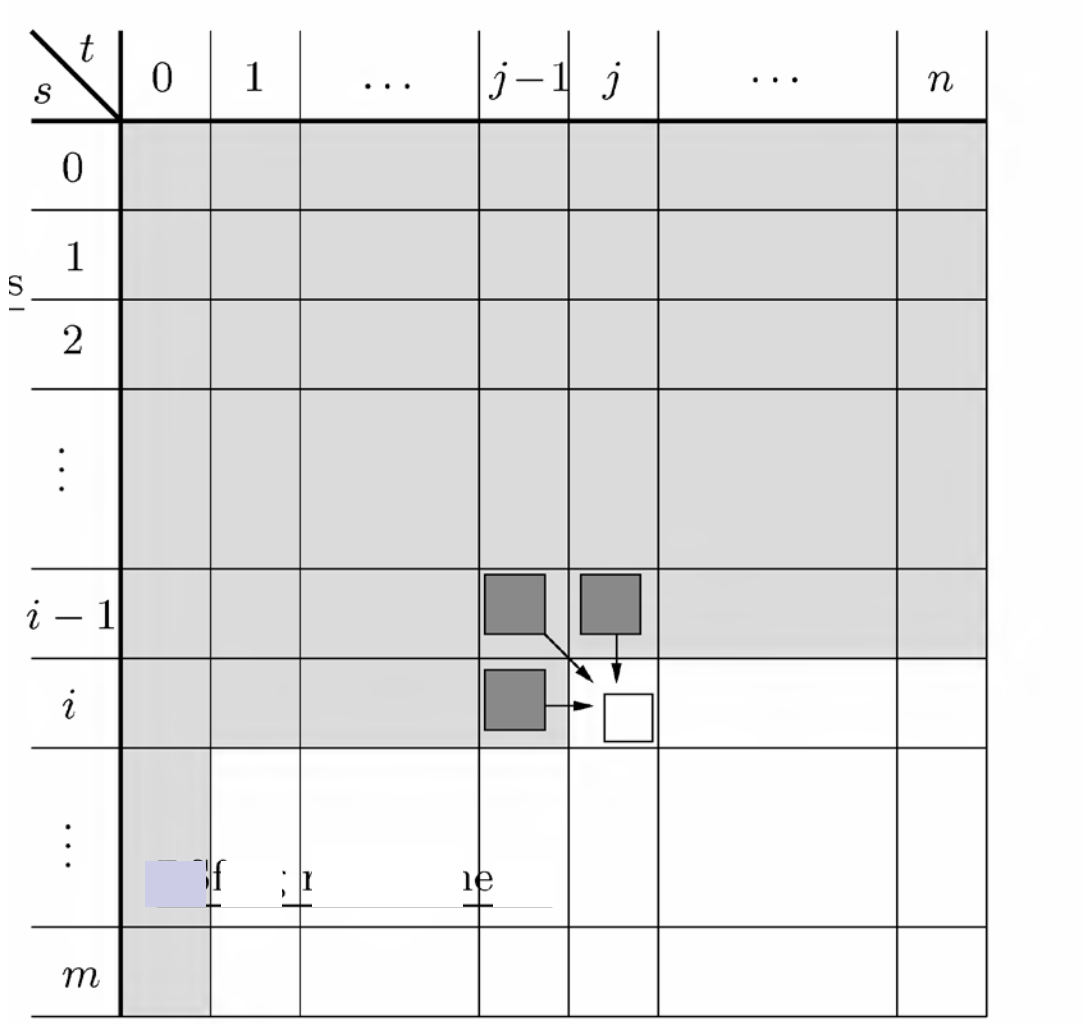
Porque sim($s_1 \ldots s_i, \lambda$) = gi …

- Falta ver cómo calculamos M(i,j) a partir de M(i-1,j-1), M(i-1,j), M(i, j-1)

Solución con programación dinámica

- $M(i,j) = \max \begin{cases} M(i-1,j)+g & \text{(inserción)} \\ M(i, j-1) +g & \text{(borrado)} \\ M(i-1,j-1)+p(s_i, t_j) & \end{cases}$

Programación dinámica



Ejemplo

- s=AAAT, t=AGT
- p(a,a)=1, p(a,b)=-1 (a≠b), g=-2

- Cada camino de la matriz correspondiente empezando en (0,0) y terminando en (m,n) es un alineamiento

- $M(i,j) = \max \begin{cases} M(i-1,j)-2 \text{ (inserción)} \\ M(i, j-1)-2 \text{ (borrado)} \\ M(i-1,j-1)+p(s_i, t_j) \end{cases}$

Ejemplo

s \ t		0	A 1	G 2	T 3
	0	0	−2	−4	−6
A	1	−2	1	−1	−3
A	2	−4	−1	0	−2
A	3	−6	−3	−2	−1
T	4	−8	−5	−4	−1

)

s \ t		0	A 1	G 2	T 3
	0	0	−2	−4	−6
A	1	−2	1	−1	−3
A	2	−4	−1	0	−2
A	3	−6	−3	−2	−1
T	4	−8	−5	−4	−1

Algorithm 5.1 Computation of the similarity

Input: Two strings $s = s_1 \dots s_m$ and $t = t_1 \dots t_n$.

1. Initialization:

   for $i = 0$ **to** m **do**
   for $j = 0$ **to** n **do**
   $M(i, j) := 0$

2. Initialization of the borders:

   for $i = 0$ **to** m **do**
   $M(i, 0) = i \cdot g$
   for $j = 0$ **to** n **do**
   $M(0, j) = j \cdot g$

3. Filling the matrix:

   for $i = 1$ **to** m **do**
   for $j = 1$ **to** n **do**
   $M(i, j) := \max\{M(i-1, j) + g,$
   $M(i, j-1) + g,$
   $M(i-1, j-1) + p(s_i, s_j)\}$

Output: $sim(s, t) = M(m, n)$.

¿Cómo calcular el alineamiento óptimo?

- A partir de la matriz reconstruir cómo se ha llegado de (0,0) a (m,n)

Ejemplo

s \ t		0	A 1	G 2	T 3
	0	0	−2	−4	−6
A	1	−2	1	−1	−3
A	2	−4	−1	0	−2
A	3	−6	−3	−2	−1
T	4	−8	−5	−4	−1

AAAT
AG-T

AAAT
-AGT

AAAT
A-GT

Algorithm 5.2 Computation of an optimal alignment

Input: A similarity matrix M for two strings $s = s_1 \dots s_m$ and $t = t_1 \dots t_n$.
Call the recursive procedure Align(m, n).

Output: The alignment (s', t') of s and t.

Procedure Align(i, j):

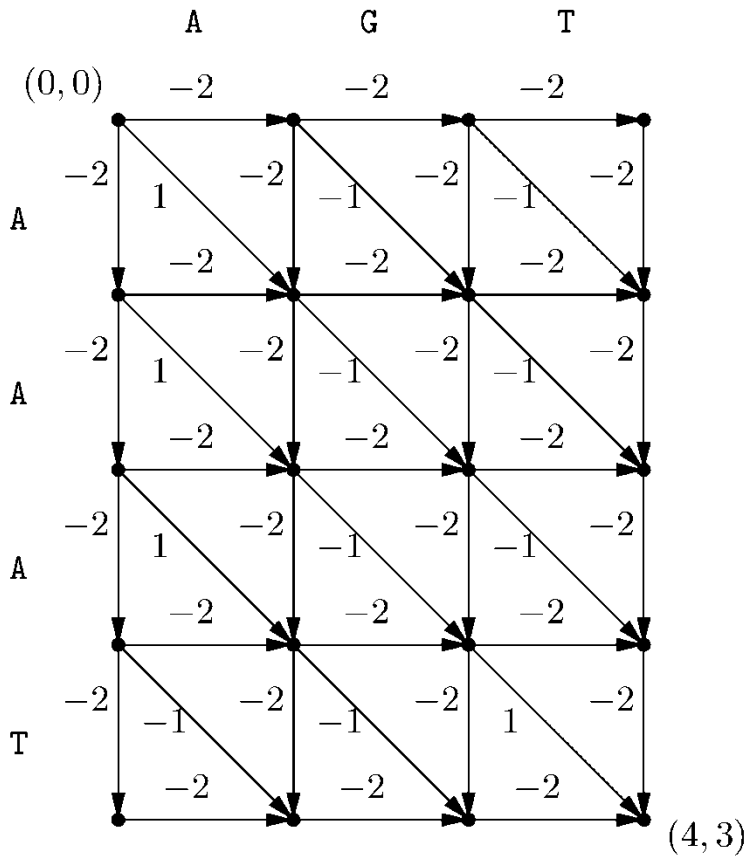
if $i = 0$ and $j = 0$ **then**
  $s' := \lambda$
  $t' := \lambda$
else
  if $M(i,j) = M(i-1,j) + g$ **then**
    $(\overline{s}, \overline{t}) :=$ Align$(i-1, j)$
    $s' := \overline{s} \cdot s_i$
    $t' := \overline{t} \cdot -$
  else if $M(i,j) = M(i,j-1) + g$ **then**
    $(\overline{s}, \overline{t}) :=$ Align$(i, j-1)$
    $s' := \overline{s} \cdot -$
    $t' := \overline{t} \cdot t_j$
  else $\{M(i,j) = M(i-1,j-1) + p(s_i, t_j)\}$
    $(\overline{s}, \overline{t}) :=$ Align$(i-1, j-1)$
    $s' := \overline{s} \cdot s_i$
    $t' := \overline{t} \cdot t_j$
return (s', t')

Complejidad …

- Calcular M cuesta O(nm)
- Una vez calculada, un alineamiento óptimo cuesta O(n+m)

- Por supuesto puede haber muchos alineamientos óptimos, sacarlos todos puede ser exponencial ($s=A^{2n}$ $t=A^{n}$)

Otra posibilidad: el grafo de edición



Buscamos caminos de (0,0) a (m,n) de peso máximo

Alineamientos locales y semiglobales

- Se trata de generalizar el alineamiento global a que sólo se alinee un substring de s con uno de t, o un prefijo de s con uno de t, etc

¿Por qué alineamiento local?

- El alineamiento global sirve para comparar proteínas de la misma familia, por ejemplo (misma longitud, propiedades similares, etc)
- Cuando comparamos **fragmentos anónimos de DNA**, algunas secciones serán similares
- Cuando comparamos **proteínas de distintas familias**, encontraremos patrones separados por zonas muy distintas

Ejemplos muy concretos

- **Genes homeobox**: presentes en muchas especies, regulan el desarrollo embrionario, las mutaciones pequeñas son muy relevantes
  - Sólo la región homeodomain, de 60 aminoácidos, es casi idéntica
- Algunas familias de proteínas (**serine proteases**) tienen zonas aisladas muy conservadas

Alineamiento local

- Un alineamiento local de s y t es un alineamiento global de dos substrings $\underline{s}=s_i \ldots s_{i+k}$ y $\underline{t}= t_j \ldots t_{j+l}$
- Un alineamiento local $(\underline{s'}, \underline{t'})$ es óptimo si

$$\delta(\underline{s'}, \underline{t'}) = \max \{sim_\delta (\underline{s}, \underline{t}) \mid \underline{s} \text{ es un substring de } s, \ \underline{t} \text{ es un substring de } t\}$$

Problema del alineamiento local óptimo

- Muy usado para comparar secuencias de DNA desconocidas

- **s= AAAAACTCTCTCT**
  t= GCGCGCGCAAAAA
- Local óptimo (puntuación 5)
  AAAAACTCTCTCT
  GCGCGCGCAAAAA
- Global óptimo (puntuación -11)
  AAAAACTCTCTCT
  GCGCGCGCAAAAA

Cálculo del alineamiento local óptimo

- $s=s_1 \ldots s_m$ y $t=t_1..t_n$
- Usamos la matriz de similaridad M que es (m+1)x(n+1)
- M(i,j) es la puntuación de una alineación óptima de **un sufijo de** $s_1 \ldots s_i$ y **un sufijo de** $t_1..t_j$
- El objetivo es el máximo valor que ocurra en la matriz

Solución con programación dinámica

- La primera fila y columna son fáciles
- M(0,j)=0, M(i,0)=0

- Falta ver cómo calculamos M(i,j) a partir de M(i-1,j-1), M(i-1,j), M(i, j-1)

Solución con programación dinámica

Sigue siendo …

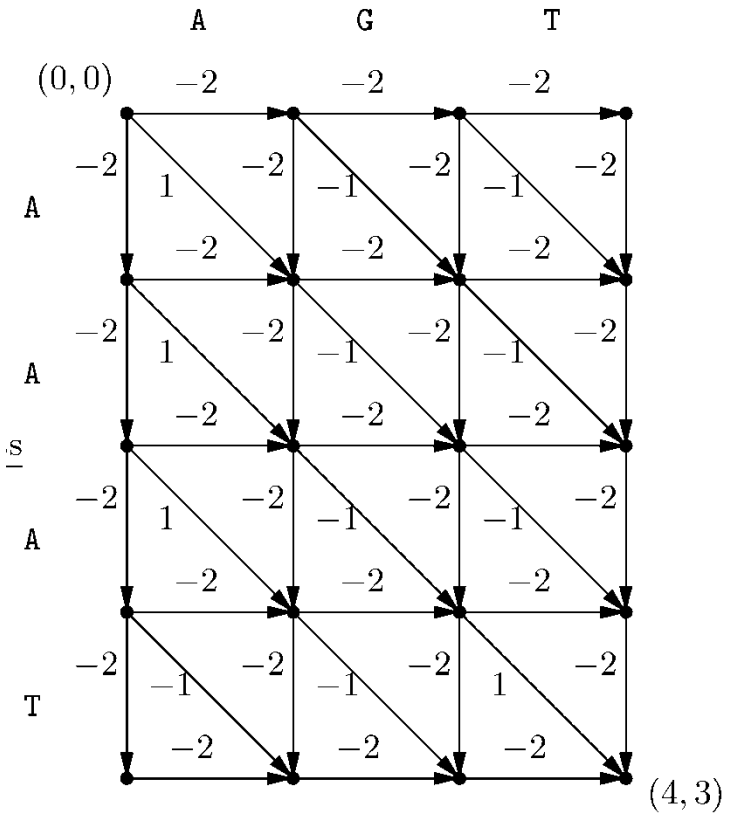
- $M(i,j) = \max \begin{cases} M(i-1,j)+g & \text{(inserción)} \\ M(i, j-1) +g & \text{(borrado)} \\ M(i-1,j-1)+p(s_i, t_j) & \\ 0 & \text{(empezar)} \end{cases}$

Solución con programación dinámica

- Guardamos en cada posición de la matriz cuál de los 4 casos da el máximo
- Un alineamiento local óptimo es el camino de vuelta de una posición de la matriz con valor máximo a una con valor 0

También se pueden usar grafos de edición …

- Añadir al grafo de edición anterior arcos de peso 0 desde (0,0) a los demás y desde cualquier vértice a (m,n)
- Buscamos caminos de (0,0) a (m,n) de peso máximo



Alineamientos semiglobales …

- Se trata de casos intermedios, alineamos todo el string pero ignoramos algunos huecos en los extremos
- Por ejemplo cuando se alinea un string largo s con uno corto t no se tienen en cuenta los – que se añaden al principio y al final de t

Ejemplo

s=ACTTTATGCCTGCT
t=ACAGGCT

- Global con puntuación -7
  ACTTTATGCCTGCT
  AC---A-G---GCT
- Con puntuación -16
  ACTTTAT-GCCTGCT
  -----ACAGGCT---
- Si no consideramos los – antes y después de t, puntuación 0
- Alineamiento local óptimo
  (ACTTTATGCCT)GCT
  (ACAG)GCT

Varios casos

- Que sean gratis los huecos al principio, al final o en los dos casos
- Por ejemplo para el overlap aproximado de s y t consideramos alineamientos sin contar los huecos a la derecha de s ni a la izquierda de t

Ignorando huecos a la derecha de s

- Si tenemos la matriz de los alineamientos globales de s y t
- Buscamos el valor máximo de la última fila: el mejor alineamiento entre s y un prefijo de t

Ignorando huecos a la derecha de t

- Si tenemos la matriz de los alineamientos globales de s y t
- Buscamos el valor máximo de la última columna: el mejor alineamiento entre un prefijo de s y t

Ignorando huecos a la izquierda de s

- Ignoramos los huecos antes de s
- Basta con inicializar la primera fila con 0 en el alineamiento global
- Corresponde a alinear s con un sufijo de t

Ignorando huecos a la izquierda de t

- Ignoramos los huecos antes de t
- Basta con inicializar la primera columna con 0 en el alineamiento global

Combinando

- Los cuatro casos anteriores se pueden combinar
- Por ejemplo el overlap aproximado de s y t es ignorar los huecos antes de t y después de s …
- Es decir, primera columna 0 y buscamos el máximo de la última fila

Overlap aproximado

primera columna 0 y buscamos el máximo de la última fila = sufijo de s con prefijo de t

- $M(i,j) = \max \begin{cases} M(i-1,j)+g & \text{(inserción)} \\ M(i, j-1) +g & \text{(borrado)} \\ M(i-1,j-1)+p(s_i, t_j) & \\ M(0,j) & \text{(empezar s)} \end{cases}$

Overlap aproximado

s \ t		0	A 1	G 2	T 3	A 4
	0	0	−2	−4	−6	−8
A	1	0	1	−1	−3	−5
A	2	0	1	0	−2	−2
A	3	0	1	0	−1	−1
T	4	0	−1	0	1	−1

AAAT-
-AGTA

Generalizando las puntuaciones

- Podemos penalizar los huecos largos utilizando en lugar de siempre la misma puntuación g para –
  - Una puntuación $-(\rho+\sigma k)$ para cuando encontremos k huecos seguidos $-^k$
- Se puede utilizar programación dinámica algo más sofisticada

Métodos de puntuación sofisticados

- A veces (proteínas) algunas sustituciones son más usuales que otras, ya que el resultado tiene propiedades similares (un aminoácido se sustituye por otro con propiedades similares)
- Utilizamos matrices de puntuación para asignar distintos valores p(a,b) según si la sustitución es usual o no (En el caso de las proteínas una matriz 20x20)

Mutaciones aceptadas

- Una mutación aceptada de una proteína es una mutación que prácticamente no afecta la función biológica de la proteína y puede ser pasada a la siguiente generación
- Dos secuencias s y t están a una unidad PAM si s se transforma en t por mutaciones aceptadas que ocurren en media una vez por cada 100 aminoácidos

Matriz k-PAM

- Son matrices para puntuar alineamientos entre secuencias de proteínas que están a k unidades PAM
- Es complicado:
  - Saber qué matriz usar
  - Calcularla
- ¿Cómo se calcula una matriz k-PAM? A base de datos experimentales …

Cálculo de una matriz k-PAM

- Contamos con pares de secuencias a k unidades PAM que están alineados
- Sea A el conjunto de esos alineamientos de pares de secuencias
- Col(A) es el conjunto de columnas de esos alineamientos que no tienen hueco

U

G

Definiendo la matriz k-PAM …

$$\text{freq}(a,b) = \frac{\text{número de } (a,b) \text{ ó } (b,a) \text{ en Col}(A)}{2\,|\text{Col}(A)|}$$

$$\text{freq}(a) = \frac{\text{número de ocurrencias de } a \text{ en } A}{\text{longitud total de todas las secuencias}}$$

Definiendo la matriz k-PAM …

$$PAM_k(i,j) = \log \frac{freq(a_i, a_j)}{freq(a_i)\ freq\ (a_j)}$$

- **Significado**: prob de que a_i se transforme en a_j por mutaciones aceptadas / prob ocurra por casualidad
- Es fácil multiplicar estos valores …
- En la práctica se multiplican por 10 y se redondean al siguiente entero

Definiendo la matriz k-PAM …

- En la práctica no es fácil contar con los datos anteriores (pares de secuencias a distancia k PAM)
- El **método que se usa** trabaja primero en calcular la matriz 1-PAM y a partir de ella saca las otras
- Es fácil encontrar secuencias que están a 1 unidad PAM a partir de secuencias muy similares con un antepasado común …

Definiendo la matriz k-PAM …

- A partir de pares de secuencias a 1 PAM definimos primero la matriz F como

$$F(i,j)= \frac{freq(a_i, a_j)}{freq(a_i)}$$

- Calculamos F^k, el producto de F consigo misma k veces

Definiendo la matriz k-PAM …

$$PAM_k(i,j) = \log \frac{freq(a_i)\, F^k(i,j)}{freq(a_i)\, freq\,(a_j)} = \log \frac{F^k(i,j)}{freq\,(a_j)}$$

- En general no se conoce la distancia PAM, así que se usan valores estándar como k= 40, k=100 y k=250

Otras matrices de puntuación

- Las matrices BLOSUM se sacan a partir de bloques largos en alineamientos múltiples
- La base de datos de BLOCKS contiene esos bloques largos, por ejemplo un bloque de tamaño n del alineamiento de k secuencias r_1, .., r_k es una matriz B de k filas y n columnas

Matrices BLOSUM

- Para calcular la matriz BLOSUM-x nos fijamos en las filas de un bloque B que difieren en al menos x%
- Sea P el conjunto de parejas de filas de bloques conocidos que difieren en x%
- $\text{freq}(a,b) = \frac{\text{ocurrencias de } (a,b) \text{ ó } (b,a) \text{ en P}}{|P|}$
- freq(a)= frecuencia de a en P

Matrices $BLOSUM_x$

$$BLOSUM_x(i,j) = 2 \log \frac{freq(a_i, a_j)}{freq(a_i)\ freq\ (a_j)}$$

- Para calcular las BLOSUM se usan las secuencias alejadas
- Se suele usar x entre 50 y 80, sobre todo BLOSUM-62

BLOSUM50

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0	-2	-1	-1	-5
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	-3	-1	0	-1	-5
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3	4	0	-1	-5
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4	5	1	-1	-5
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1	-3	-3	-2	-5
Q	-1	1	0	0	-3	7	2	-2	1	-3	-2	2	0	-4	-1	0	-1	-1	-1	-3	0	4	-1	-5
E	-1	0	0	2	-3	2	6	-3	0	-4	-3	1	-2	-3	-1	-1	-1	-3	-2	-3	1	5	-1	-5
G	0	-3	0	-1	-3	-2	-3	8	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4	-1	-2	-2	-5
H	-2	0	1	-1	-3	1	0	-2	10	-4	-3	0	-1	-1	-2	-1	-2	-3	2	-4	0	0	-1	-5
I	-1	-4	-3	-4	-2	-3	-4	-4	-4	5	2	-3	2	0	-3	-3	-1	-3	-1	4	-4	-3	-1	-5
L	-2	-3	-4	-4	-2	-2	-3	-4	-3	2	5	-3	3	1	-4	-3	-1	-2	-1	1	-4	-3	-1	-5
K	-1	3	0	-1	-3	2	1	-2	0	-3	-3	6	-2	-4	-1	0	-1	-3	-2	-3	0	1	-1	-5
M	-1	-2	-2	-4	-2	0	-2	-3	-1	2	3	-2	7	0	-3	-2	-1	-1	0	1	-3	-1	-1	-5
F	-3	-3	-4	-5	-2	-4	-3	-4	-1	0	1	-4	0	8	-4	-3	-2	1	4	-1	-4	-4	-2	-5
P	-1	-3	-2	-1	-4	-1	-1	-2	-2	-3	-4	-1	-3	-4	10	-1	-1	-4	-3	-3	-2	-1	-2	-5
S	1	-1	1	0	-1	0	-1	0	-1	-3	-3	0	-2	-3	-1	5	2	-4	-2	-2	0	0	-1	-5
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	2	5	-3	-2	0	0	-1	0	-5
W	-3	-3	-4	-5	-5	-1	-3	-3	-3	-3	-2	-3	-1	1	-4	-4	-3	15	2	-3	-5	-2	-3	-5
Y	-2	-1	-2	-3	-3	-1	-2	-3	2	-1	-1	-2	0	4	-3	-2	-2	2	8	-1	-3	-2	-1	-5
V	0	-3	-3	-4	-1	-3	-3	-4	-4	4	1	-3	1	-1	-3	-2	0	-3	-1	5	-4	-3	-1	-5
B	-2	-1	4	5	-3	0	1	-1	0	-4	-4	0	-3	-4	-2	0	0	-5	-3	-4	5	2	-1	-5
Z	-1	0	0	1	-3	4	5	-2	0	-3	-3	1	-1	-4	-1	0	-1	-2	-2	-3	2	5	-1	-5
X	-1	-1	-1	-1	-2	-1	-1	-2	-1	-1	-1	-1	-1	-2	-2	-1	0	-3	-1	-1	-1	-1	-1	-5
*	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	1

Próximo día

- Alineamiento de 2 strings (global, local y semiglobal)
- **Métodos heurísticos (para búsqueda en BdD)**
- Alineamientos múltiples