TÉCNICAS INTELIGENTES EN BIOINFORMÁTICA

Alineamiento de pares de secuencias de genes/proteínas

Mario de J. Pérez Jiménez
Grupo de investigación en Computación Natural
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

Máster Universitario en Lógica, Computación e Inteligencia Artificial

Curso 2014-15

Introducción (I)

Una clave de los descubrimientos en Biología:

- Comparar diferentes individuos para obtener conclusiones acerca de ellos.
- Ejemplo: si dos proteínas tienen secuencias "similares", es probable que ambas tengan funcionalidades parecidas, que hayan evolucionado dependientemente, o bien que compartan algun ancestro común.

La **técnica** más común de comparar pares de secuencias (de nucleótidos/aminoácidos/genes/proteínas) es la del **alineamiento**.

- Recuérdese que una **secuencia** es una sucesión finita ordenada (**cadena**) de elementos sobre un alfabeto finito. Se representará así $A[1, \ldots, n]$ y el elemento i-ésimo se notará $A[i]$.
- Básicamente consiste en representar el par de secuencias una debajo de la otra (pudiendo incluir huecos) e interpretando para cada posición:
  - ⋆ Si hay coincidencia de símbolos, entonces ésta se ha conservado durante la evolución.
  - ⋆ Si **no** hay coincidencia de símbolos (o existe algún hueco), entonces se ha producido una mutación puntual.

Respecto de los huecos: nunca aparecerán dos huecos en una misma posición ya que ello no proporciona información alguna de "cambio respecto del tiempo"

Introducción (II)

Alineamiento de pares de secuencias: comparación de secuencias para resaltar sus similitudes y diferencias, descubriendo relaciones de tipo:

- Funcional: dos secuencias tienen la misma función o similar.
- Estructural: dos secuencias de aminoácidos tienen la misma estructura 3D.
- Evolutiva: dos secuencias proceden de un ancestro común.

Objetivo principal:

- Determinar el grado de relación existente entre dos pares de secuencias (de nucleótidos o de aminoácidos).
- Determinar si proceden de un ancestro común (secuencias **homólogas**).

Pares de secuencias homólogas: suelen tener

- Secuencias "parecidas".
- Funcionalidades comunes.
- Estructuras 3D similares (en el caso de proteínas).

Conceptos básicos

Ortología: secuencias homólogas en diferentes especies, debido a un mecanismo evolutivo.

- El gen de la mioglobina en ratas y humanos tienen un ancestro común hace 80 millones de años (MYA).

Paralogía: Secuencias homólogas pero debido a un mecanismo distinto de la evolución.

- Por ejemplo: debido a una duplicación genética (producido en una misma especie).

Similitud: indica el grado de coincidencia entre dos pares de secuencias (suele expresarse en porcentajes).

Identidad: indica la coincidencia total entre los pares de secuencias (similitud del 100%).

Analogía: indica un alto grado de coincidencia.

- Dos pares de secuencias homólogas no tienen por qué ser análogas: la β-globina y la neuroglobina son homólogas y, en cambio, sólo comparten un 22% de sus secuencias a pesar de ser homólogas.

El problema del alineamiento (I)

El problema de alinear pares de secuencias para realizar una comparación "lo más efectiva posible" no es una tarea fácil.

- ⋆ **Ejemplo 1**: Comparamos la secuencia **MODELO** con **MUNDO**, **CORDEL** y **MODO**.

M	**O**	**D**	**E**	**L**	**O**	
M	U	N	D	O		+1
C	O	R	D	E	L	+2
M	O	D	O			+3

Se pueden introducir **huecos** en las secuencias a fin de aumentar la similitud:

M	**O**	**D**	**E**	**L**	**O**	
M	U	N	D	-	O	+2
C	O	R	D	E	L	+2
M	O	-	-	D	O	+4

O bien podemos hacer esta otra distribución

M	**O**	-	**D**	**E**	**L**	**O**	
M	U	N	D	-	-	O	+3
C	O	R	D	E	L		+5
M	O	-	D	-	-	O	+4

Los huecos pueden representar posibles mutaciones en el proceso evolutivo.

El problema del alineamiento (II)

- ⋆ **Ejemplo 2**: se trata de comparar dos secuencias de nucleótidos

$$A[1,2,3,4] = TGCG \text{ y } B[1,2,3,4] = TGCT$$

Existen diferentes formas de alinearlas para su comparación

$1^{\underline{a}}$	$2^{\underline{a}}$	$3^{\underline{a}}$	$4^{\underline{a}}$	$5^{\underline{a}}$
$T\ G\ C\ G$	$T\ G\ C\ G\ -$	$-\ T\ G\ C\ -\ G$	$TG\ -\ -\ C\ G$	$T\ -\ G\ -\ C\ -\ G$
$T\ G\ C\ T$	$T\ G\ C\ -\ T$	$T\ -\ -\ G\ C\ T$	$TG\ C\ T\ -\ -$	$-T\ -\ G\ -\ C\ T$

Interpretación: (a) Ha habido una mutación de G a T en la $2^{\underline{a}}$ secuencia respecto de la $1^{\underline{a}}$: (b) en la $1^{\underline{a}}$ secuencia respecto de la $2^{\underline{a}}$, ha habido una delección de una T y una insercción de una G; etc.

¿Cómo decidir **cuál es el mejor alineamiento**?

El problema del alineamiento (III)

Se trata de un **problema de optimización**:

- Deberá tener una **función objetivo** que asignará valores numéricos a cada alineamiento.
  - ⋆ Necesidad de fijar un **sistema de puntuación**.

La resolución mecánica del problema mediante un algoritmo de fuerza bruta sería la siguiente:

- Considerar **TODOS** los posibles alineamientos (con la única restricción de que no pueden haber dos huecos en una misma posición).
- Asignar un valor numérico a cada uno de ellos.
- Elegir un alineamiento que sea óptimo (generalmente, máximo) para esos valores.

No existe acuerdo acerca de un sistema de puntuación "perfecto".

El problema del alineamiento (IV)

Un **ejemplo de sistema de puntuación** para pares de **secuencias de nucleótidos**:

- Los nucleótidos A y G son de la familia de las **purinas**.
- Los nucleótidos C y T son de la familia de las **pirimidinas**.
- Si dos nucleótidos **coinciden** le asignamos el valor $+3$.
- Si uno de los nucleótidos es A y el otro G le asignamos el valor $+1$.
- Si uno de los nucleótidos es C y el otro T le asignamos el valor $+1$.
- Si uno de los nucleótidos es A (resp. G) y el otro es C (resp. T), le asignamos el valor -1.
- Además, a la aparición de un **hueco** le asignaremos el valor -2.

En este caso, la matriz de similitudes $s(A[i], B[j])$ sería la siguiente:

	A	C	G	T
A	$+3$			
C	-1	$+3$		
G	$+1$	-1	$+3$	
T	-1	$+1$	-1	$+3$

El problema del alineamiento (V)

De esta manera, podemos calcular los distintos valores que la función objetivo asigna a cada uno de los posibles alineamientos:

$$F\begin{pmatrix} T & G & C & G \\ T & G & C & T \end{pmatrix} = (+3) + (+3) + (+3) + (-1) = +8$$

$$F\begin{pmatrix} T & G & C & G & - \\ T & G & C & - & T \end{pmatrix} = (+3) + (+3) + (+3) + (-2) + (-2) = +5$$

$$F\begin{pmatrix} - & T & G & C & - & G \\ T & - & - & G & C & T \end{pmatrix} = (-2) + (-2) + (-2) + (-1) + (-2) + (-1) = -10$$

Y así sucesivamente.

Algoritmos de alineamiento

Diagramas de puntos (Dot plots).

Algoritmos dinámicos.

- **Alineamiento global** (Needleman y Wunsch, 1970).
  - ⋆ Consiste en encontrar unos alineamientos de las dos secuencias completas (se introducen huecos para igualar sus longitudes) que proporciona un valor numérico global máximo (Algoritmo lento que usa mucha memoria).
  - ⋆ Ofrece buenos resultados para secuencias con alto grado de similitud, tanto en la longitud como en el contenido de éstas.
- **Alineamiento local** (Smith y Waterman, 1981).
  - ⋆ Consiste en encontrar sendos fragmentos de ambas secuencias que proporcionan alineamientos con puntuación máxima (se alinean las partes más parecidas).
  - ⋆ Suele ser una combinación de muchos alineamientos globales de secuencias cortas.
  - ⋆ Ofrece buenos resultados para secuencias con baja similitud pero que pueden contener regiones parecidas.
  - ⋆ **BLAST** (Basic Local Alignment Search Tool) es el principal programa de alineamiento local de secuencias y se puede considerar como "una evolución" del algoritmo de Smith-Waterman de alineamiento local.

Diagramas de puntos

- No es propiamente un algoritmo.
- Se puede realizar una verificación visual.
- Se coloca una secuencia en el eje de abcisas y otra en el eje de ordenadas.
- Se colocará una señal en los puntos donde haya coincidencia.
- Las secuencias similares aparecerán en forma de líneas diagonales.

Permite detectar fácilmente

- Dominios internos en una secuencia.
- Dominios conservados entre dos secuencias.
- Zonas de baja complejidad.
- Exones e intrones.
- Etc.

Otros algoritmos

Objetivo último: encontrar el "mejor" alineamiento posible:

- El que proporciona mayor puntuación.

Número total de alineamientos (con huecos) de un par de secuencias de longitud n: $\binom{2n}{n}$

- Si $n = 30$ entonces el número es del orden 10^{17} (3.200 millones de años en "leerlo").

La técnica de **programación dinámica**.

Los algoritmos de alineamiento global y alineamiento local funcionan acorde a un sistema de puntuación que indica el grado de similitud entre el par de secuencias.

- Por ejemplo: (a) por cada elemento igual, **match**, ponemos +1; (b) por cada elemento desigual, **mismatch**, ponemos -1; y (c) por cada hueco introducido ponemos -1.
- Este sistema de puntuación puede ser diferente (como ya hemos hecho en otro ejemplo).

Alineamiento global: algoritmo de Needleman-Wunsch (I)

Es un ejemplo típico de algoritmo de programación dinámica.

Objetivo: dado un par de secuencias, hallar un alineamiento óptimo (generalmente, de valor o puntuación máxima).

- **Entrada del algoritmo NW**.
  - ⋆ Un par de secuencias $A[1, \ldots n]$ y $B[1, \ldots, m]$ (una secuencia FILA y una secuencia COLUMNA).
  - ⋆ Una función de similitud entre los símbolos que aparecen en las secuencias.
  - ⋆ Una constante de penalización de huecos.
  - ⋆ Un criterio de desempates (si no se usa, podrán existir distintas soluciones óptimas).
- **Estructura del algoritmo NW**.
  - **(a)** Inicialización.
  - **(b)** Rellenado de la matriz de puntuación.
  - **(c)** Realización del TRACE-BACK.
- **Salida del algoritmo NW**.
  - ⋆ Un alineamiento del par de secuencias, obtenido mediante la decodificación del TRACE-BACK.

Complejidad computacional: $O(n \cdot m)$ en tiempo y en espacio.

Alineamiento global: algoritmo de Needleman-Wunsch (II)

(a) Proceso de inicialización.

- Se parte de las dos secuencias a ordenar:
  - ⋆ La secuencia **fila** $A[1, \ldots, n]$ (proporcionará las filas de una matriz).
  - ⋆ La secuencia **columna** $B[1, \ldots, m]$ (proporcionará las columnas de una matriz).
- Se parte de una matriz de similitud **s(i, j)** entre los símbolos que aparecen en las secuencias.
- Se parte de una constante **g** de penalización de huecos.
- Se diseña una matriz $M(i, j)$ con $(n + 1)$ filas y $(m + 1)$ columnas.
  - ⋆ La primera fila y la primera columna se inicializan a 0: $M(1, j) = M(i, 1) = 0$.
  - ⋆ A la fila i-ésima $(2 \leq i \leq n + 1)$ se le asigna el nombre $A[i]$.
  - ⋆ A la columna j-ésima $(2 \leq j \leq m + 1)$ se le asigna el nombre $B[j]$.

Ilustración del proceso de inicialización

Consideremos las siguientes secuencias:

$$\begin{array}{lcll} A\,[\,i\,] & = & \mathbf{T\ G\ G\ C\ A\ T\ T\ C\ C\ G\ A} & (n{=}11) \\ B\,[\,j\,] & = & \mathbf{G\ C\ C\ A\ A\ T\ G\ A\ C} & (m{=}9) \end{array}$$

Supongamos que la matriz de similitud es la siguiente:

	A	C	G	T
A	$+3$			
C	-1	$+3$		
G	$+1$	-1	$+3$	
T	-1	$+1$	-1	$+3$

Supongamos que la penalización por huecos es $g = -2$.
La matriz $M[i, j]$ se inicializa como sigue:

		T	**G**	**G**	**C**	**A**	**T**	**T**	**C**	**C**	**G**	**A**
	0	0	0	0	0	0	0	0	0	0	0	0
G	0											
C	0											
C	0											
A	0											
A	0											
T	0											
G	0											
A	0											
C	0											

Alineamiento global: algoritmo de Needleman-Wunsch (III)

(b) Proceso de rellenado de la matriz de puntuación.

- Se parte de las valores de inicialización $M(1, j) = M(i, 1) = 0$. A partir de aquí se calculan los valores de $M(i, j)$ $(2 \leq i \leq n, 2 \leq j \leq m)$ como sigue:

$M(i-1, j-1)$	$M(i-1, j)$
$M(i, j-1)$	$M(i, j)$

$$M(i,j) = \text{máx} \begin{cases} M(i-1, j-1) + \mathbf{s(i, j)}, \\ M(i-1, j) + \mathbf{g}, \\ M(i, j-1) + \mathbf{g}. \end{cases}$$

- ⋆ Se **demuestra** que el valor $M(i, j)$ obtenido por el criterio dado, proporciona el **valor óptimo** en el alineamiento de la subsecuencia $A[1, \ldots, i]$ y la subsecuencia $B[1, \ldots, j]$.
- ⋆ Pueden existir otros criterios diferentes de puntuación o de elaboración de la matriz $M(i, j)$.

Ilustración del proceso de rellenado

La matriz $M[i,j]$ debidamente rellena es la siguiente:

		T	G	G	C	A	T	T	C	C	G	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	−1	+3	+3	+1	+1	−1	−1	−1	−1	+3	+1
C	0	+1	+1	+2	+6	+4	+2	0	+2	+2	+1	+2
C	0	+1	0	0	+5	+5	+5	+3	+3	+5	+3	+1
A	0	−1	+2	+1	+3	+8	+6	+4	+2	+3	+6	+6
A	0	−1	0	+3	+1	+6	+7	+5	+3	+1	+4	+9
T	0	+3	+1	+1	+4	+4	+9	+10	+8	+6	+4	+7
G	0	+1	+6	+4	+2	+5	+7	+8	+9	+7	+9	+7
A	0	−1	+4	+7	+5	+5	+5	+6	+7	+8	+8	+12
C	0	+1	+2	+5	+10	+8	+6	+6	+9	+10	+8	+10

Alineamiento global: algoritmo de Needleman-Wunsch (IV)

(c) Proceso de realización del TRACE-BACK.

- A partir de la matriz $M(i, j)$ rellenada, se elabora una sucesión ordenada de pares (celda , valor), denominada TRACE-BACK, construida como sigue:
  - ⋆ El primer par está compuesto por la celda $(n + 1, m + 1)$ y su valor correspondiente en la matriz $M(i, j)$ (extremo inferior derecho).
  - ⋆ A partir de ese par, se irá seleccionando los pares formados por las celdas y sus correspondientes valores, de acuerdo con la dirección que indique de dónde procedía el valor obtenido en la misma (puede existir más de un TRACE-BACK).

Ilustración del proceso de realización del TRACE-BACK

		T	G	G	C	A	T	T	C	C	G	A
	0	← 0	0	0	0	0	0	0	0	0	0	0
G	0	-1	↖ +3	+3	+1	+1	-1	-1	-1	-1	+3	+1
C	0	+1	+1	↖ +2	+6	+4	+2	0	+2	+2	+1	+2
C	0	+1	0	0	↖ +5	+5	+5	+3	+3	+5	+3	+1
A	0	-1	+2	+1	+3	↖ +8	+6	+4	+2	+3	+6	+6
A	0	-1	0	+3	+1	+6	↖ +7	+5	+3	+1	+4	+9
T	0	+3	+1	+1	+4	+4	+9	↖ +10	← +8	← +6	+4	+7
G	0	+1	+6	+4	+2	+5	+7	+8	+9	+7	↖ +9	+7
A	0	-1	+4	+7	+5	+5	+5	+6	+7	+8	+8	↖ +12
C	0	+1	+2	+5	+10	+8	+6	+6	+9	+10	+8	↑ +10

Alineamiento global: algoritmo de Needleman-Wunsch (V)

∗ **Salida del algoritmo NW**.

- Se construye un alineamiento del par de secuencias (en la parte superior se colocará la secuencia FILA y en la parte inferior se colocará la secuencia COLUMNA) **decodificando** el TRACE-BACK. Para ello:

  ⋆ Se recorre el TRACE-BACK de izquierda a derecha y en función del sentido del recorrido se escribirá en la parte superior e inferior los símbolos que corresponden a las secuencias filas y columnas de acuerdo con el siguiente criterio:

  – Si se produce un avance hacia la **diagonal izquierda** $\nwarrow$:
  Parte superior: **símbolo** de la secuencia **FILA**.
  Parte inferior: **símbolo** de la secuencia **COLUMNA**.

  – Si se produce un avance hacia la **izquierda** $\leftarrow$:
  Parte superior: **hueco** en la secuencia **FILA**.
  Parte inferior: **símbolo** de la secuencia **COLUMNA**.

  – Si se produce un avance hacia **arriba** $\uparrow$:
  Parte superior: **símbolo** de la secuencia **FILA**.
  Parte inferior: **hueco** en la secuencia **COLUMNA**.

Ilustración del proceso de decodificación del TRACE-BACK

↖ **símbolo** FILA **símbolo** COLUMNA	← **hueco** FILA **símbolo** COLUMNA	↑ **símbolo** FILA **hueco** COLUMNA

Decodificando el TRACE-BACK se obtiene el siguiente alineamiento del par de secuencias:

←	↖	↖	↖	↖	↖	↖	←	←	↖	↖	↑
0	3	2	5	8	7	10	8	6	9	12	10
–	**G**	**C**	**C**	**A**	**A**	**T**	–	–	**G**	**A**	**C**
T	**G**	**G**	**C**	**A**	**T**	**T**	**C**	**C**	**G**	**A**	–

Valor óptimo del alineamiento: $6 \cdot (+3) + 4 \cdot (-2) + 2 \cdot (-1) = +10$.

Ilustración del algoritmo de Needleman-Wunsch (I)

Consideremos las siguientes secuencias:

T G G C A T T C C G A

G C C A A T G A C

Vamos a determinar un alineamiento óptimo aplicando el algoritmo de Needleman-Wunsch.

Supongamos que la matriz de similitud es la siguiente:

	A	C	G	T
A	$+3$			
C	-1	$+3$		
G	$+1$	-1	$+3$	
T	-1	$+1$	-1	$+3$

Supongamos que la penalización por huecos es $g = -2$.

Ilustración del algoritmo de Needleman-Wunsch (II)

La matriz correspondiente debidamente rellena es la siguiente:

		T	G	G	C	A	T	T	C	C	G	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	−1	+3	+3	+1	+1	−1	−1	−1	−1	+3	+1
C	0	+1	+1	+2	+6	+4	+2	0	+2	+2	+1	+2
C	0	+1	0	0	+5	+5	+5	+3	+3	+5	+3	+1
A	0	−1	+2	+1	+3	+8	+6	+4	+2	+3	+6	+6
A	0	−1	0	+3	+1	+6	+7	+5	+3	+1	+4	+9
T	0	+3	+1	+1	+4	+4	+9	+10	+8	+6	+4	+7
G	0	+1	+6	+4	+2	+5	+7	+8	+9	+7	+9	+7
A	0	−1	+4	+7	+5	+5	+5	+6	+7	+8	+8	+12
C	0	+1	+2	+5	+10	+8	+6	+6	+9	+10	+8	+10

Ilustración del algoritmo de Needleman-Wunsch (III)

		T	G	G	C	A	T	T	C	C	G	A
	0	← 0	0	0	0	0	0	0	0	0	0	0
G	0	-1	↖ +3	+3	+1	+1	-1	-1	-1	-1	+3	+1
C	0	+1	+1	↖ +2	+6	+4	+2	0	+2	+2	+1	+2
C	0	+1	0	0	↖ +5	+5	+5	+3	+3	+5	+3	+1
A	0	-1	+2	+1	+3	↖ +8	+6	+4	+2	+3	+6	+6
A	0	-1	0	+3	+1	+6	↖ +7	+5	+3	+1	+4	+9
T	0	+3	+1	+1	+4	+4	+9	↖ +10	← +8	← +6	+4	+7
G	0	+1	+6	+4	+2	+5	+7	+8	+9	+7	↖ +9	+7
A	0	-1	+4	+7	+5	+5	+5	+6	+7	+8	+8	↖ +12
C	0	+1	+2	+5	+10	+8	+6	+6	+9	+10	+8	↑ +10

Ilustración del algoritmo de Needleman-Wunsch (IV)

Decodificando el TRACE-BACK se obtiene el siguiente alineamiento del par de secuencias:

←	↖	↖	↖	↖	↖	↖	←	←	↖	↖	↑
0	3	2	5	8	7	10	8	6	9	12	10
–	**G**	**C**	**C**	**A**	**A**	**T**	–	–	**G**	**A**	**C**
T	**G**	**G**	**C**	**A**	**T**	**T**	**C**	**C**	**G**	**A**	–

Valor óptimo del alineamiento: $6 \cdot (+3) + 4 \cdot (-2) + 2 \cdot (-1) = +10$.

Otra aproximación del algoritmo de Needleman-Wunsch (I)

Se parte de una secuencia fila (de longitud n) y una secuencia columna (de longitud m).
Se construye una matriz $M(i, j)$:

- (a) en las **filas**, una secuencia $A[1, \ldots, n]$ (FILA);
- (b) en las **columnas**, la otra secuencia $B[1, \ldots, m]$ (COLUMNA).

No obstante, ahora:

- La **primera fila** contiene los valores de las distancias al "origen" respectivo (gap scores), acorde con el valor del hueco (penalización $g = -1$).
- La primera **columna** contiene los valores de las distancias al "origen" respectivo (gap scores).
- En cada celda de esa primera fila y primera columna aparece una flecha indicando el sentido del origen

		E	M	B	C	F	E	F	G	O	N
	0	← -1	← -2	← -3	← -4	← -5	← -6	← -7	← -8	← -9	← -10
A	↑ -1										
B	↑ -2										
C	↑ -3										
D	↑ -4										
E	↑ -5										
F	↑ -6										
G	↑ -7										

Otra aproximación del algoritmo de Needleman-Wunsch (II)

Supongamos que la matriz de similitud está definida como sigue:

$$s(i,j) = \begin{cases} 1 & \text{si } A[i] = B[j] \\ -1 & \text{si } A[i] \neq B[j] \end{cases}$$

y que cada hueco se penaliza con una puntuación $g = -1$.

- En cada celda restante (recorrida en orden lexicográfico) se anota el máximo de los tres valores siguientes:
  - MM + el valor de la celda adyacente superior izquierda (**diagonal izquierda**).
  - MM + el valor de la celda adyacente izquierda (**izquierda**).
  - MM + el valor de la celda adyacente superior (**arriba**).

  En donde MM (match/mismatch) vale +1 (respectivamente, -1) si los símbolos asociados a la celda coinciden (respect. son distintos).
- Se fijará un criterio de desempate para el caso de que el máximo sea alcanzado por más de uno de los valores para indicar la celda de la que procede (a fin de hacer **determinista** el algoritmo).
- Además, en cada celda se pondrá una flecha que indica la procedencia del valor obtenido.

Otra aproximación del algoritmo de Needleman-Wunsch (III)

Hallemos el valor numérico y la flecha que aparecerá en la celda indicada.

		E	M	B	C	F	E	F	G	O	N
	0	← -1	← -2	← -3	← -4	← -5	← -6	← -7	← -8	← -9	← -10
A	↑ -1	? ?									
B	↑ -2										
C	↑ -3										
D	↑ -4										
E	↑ -5										
F	↑ -6										
G	↑ -7										

Otra aproximación del algoritmo de Needleman-Wunsch (IV)

En este caso, los símbolos **A** y **E** son distintos: Por tanto, el valor de MM será -1.

Por otra parte,

- El valor de la celda adyacente superior izquierda es 0; por tanto, el primer valor será: MM+0 $= -1$.
- El valor de la celda adyacente superior es -1; por tanto, el segundo valor será: MM+$(-1) = -2$.
- El valor de la celda adyacente izquierda es -1; por tanto, el tercer valor será: MM+$(-1) = -2$.

En consecuencia, el máximo de esos tres valores es -1 y su dirección debe marcar la celda adyacente superior izquierda.

		E	M	B	C	F	E	F	G	O	N
	0	← -1	← -2	← -3	← -4	← -5	← -6	← -7	← -8	← -9	← -10
A	↑ -1	↖ -1									
B	↑ -2										
C	↑ -3										
D	↑ -4										
E	↑ -5										
F	↑ -6										
G	↑ -7										

Otra aproximación del algoritmo de Needleman-Wunsch (V)

Procediendo acorde lo indicado anteriormente se obtiene la siguiente tabla:

		E	M	B	C	F	E	F	G	O	N
	0	← -1	← -2	← -3	← -4	← -5	← -6	← -7	← -8	← -9	← -10
A	↑ -1	↖ -1	↖ -2	↖ -3	↖ -4	↖ -5	↖ -6	↖ -7	↖ -8	↖ -9	↖ -10
B	↑ -2	↖ -2	↖ -2	↖ -1	← -2	← -3	← -4	← -5	← -6	← -7	← -8
C	↑ -3	↖ -3	↖ -3	↑ -2	↖ 0	← -1	← -2	← -3	← -4	← -5	← -6
D	↑ -4	↖ -4	↖ -4	↑ -3	↑ -1	↖ -1	↖ -2	↖ -3	↖ -4	↖ -5	↖ -6
E	↑ -5	↖ -3	← -4	↑ -4	↑ -2	↖ -2	↖ 0	← -1	← -2	← -3	← -4
F	↑ -6	↑ -4	↖ -4	↖ -5	↑ -3	↖ -1	↑ -1	↖ 1	← 0	← -1	← -2
G	↑ -7	↑ -5	↖ -5	↖ -5	↑ -4	↑ -2	↖ -2	↑ 0	↖ 2	← 1	← 0

Otra aproximación del algoritmo de Needleman-Wunsch (VI)

Seguidamente se obtiene el TRACE-BACK: comenzando por la parte inferior derecha, se marca la ruta que tiene mejores puntuaciones.

		E	M	B	C	F	E	F	G	O	N
	0	← -1	← -2	← -3	← -4	← -5	← -6	← -7	← -8	← -9	← -10
A	↑ -1	↖ -1	↖ **-2**	↖ -3	↖ -4	↖ -5	↖ -6	↖ -7	↖ -8	↖ -9	↖ -10
B	↑ -2	↖ -2	↖ -2	↖ **-1**	← -2	← -3	← -4	← -5	← -6	← -7	← -8
C	↑ -3	↖ -3	↖ -3	↑ -2	↖ **0**	← -1	← -2	← -3	← -4	← -5	← -6
D	↑ -4	↖ -4	↖ -4	↑ -3	↑ -1	↖ **-1**	↖ -2	↖ -3	↖ -4	↖ -5	↖ -6
E	↑ -5	↖ -3	← -4	↑ -4	↑ -2	↖ -2	↖ **0**	← -1	← -2	← -3	← -4
F	↑ -6	↑ -4	↖ -4	↖ -5	↑ -3	↖ -1	↑ -1	↖ **1**	← 0	← -1	← -2
G	↑ -7	↑ -5	↖ -5	↖ -5	↑ -4	↑ -2	↖ -2	↑ 0	↖ **2**	← **1**	← **0**

Otra aproximación del algoritmo de Needleman-Wunsch (VII)

Decodificando el TRACE-BACK se obtiene el siguiente alineamiento del par de secuencias:

←	↖	↖	↖	↖	↖	↖	↖	←	←
-1	-2	-1	0	-1	0	1	2	1	0
–	**A**	**B**	**C**	**D**	**E**	**F**	**G**	–	–
E	**M**	**B**	**C**	**F**	**E**	**F**	**G**	**O**	**N**

Valor óptimo del alineamiento: $3 \cdot (-1) + 2 \cdot (-1) + 5 \cdot (+1) = 0$.

Alineamiento local: algoritmo de Smith-Waterman

Está inspirado el algoritmo de NW de alineamiento global.

- En este caso, la primera fila y la primera columna tiene todos sus elementos iguales a 0.
- Para calcular los valores de las siguientes celdas, se procede de manera similar a la del algoritmo de NW, con la siguiente variación:
  - ⋆ Si el valor obtenido para una celda resulta negativo, entonces se sustituirá un 0.
- El TRACE-BACK se construye ahora prodedicendo de la siguiente manera:
  - ⋆ Comienza con el valor más alto de la tabla.
  - ⋆ Continúa de la misma manera que en el algoritmo de NW, prosiguiendo por el camino (puede haber más de uno) que indica la procedencia del valor de la celda actual, hasta que se llega a un 0 en el que finaliza el proceso de obtención de un alineamiento local óptimo.

Ilustración del algoritmo de Smith-Waterman (I)

Vamos a ilustrar el algoritmo de Smith-Waterman considerando el mismo ejemplo de antes: los pares de secuencias

A B C D E F G

E M B C F E F G O N

Recordemos que la matriz de similitud era:

$$s(i,j) = \begin{cases} 1 & \text{si } A[i] = B[j] \\ -1 & \text{si } A[i] \neq B[j] \end{cases}$$

y la penalización por huecos era $g = -1$.

Ilustración del algoritmo de Smith-Waterman (II)

Procedamos a la fase de rellenado de la matriz en el caso del algoritmo local de Smith-Waterman.

		E	M	B	C	F	E	F	G	O	N
	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	1	0	0	0	0	0	0	0
C	0	0	0	0	2	1	0	0	0	0	0
D	0	0	0	0	1	1	0	0	0	0	0
E	0	1	0	0	0	0	2	1	0	0	0
F	0	0	0	0	0	0	1	3	2	1	0
G	0	0	0	0	0	0	0	2	4	3	2

Ilustración del algoritmo de Smith-Waterman (III)

A continuación, vamos a determinar el correspondiente TRACE-BACK.

		E	**M**	**B**	**C**	**F**	**E**	**F**	**G**	**O**	**N**
	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	↖ 1	0	0	0	0	0	0	0
C	0	0	0	0	↖ 2	1	0	0	0	0	0
D	0	0	0	0	1	↖ 1	0	0	0	0	0
E	0	1	0	0	0	0	↖ 2	1	0	0	0
F	0	0	0	0	0	0	1	↖ 3	2	1	0
G	0	0	0	0	0	0	0	2	↖ 4	3	2

Ilustración del algoritmo de Smith-Waterman (IV)

Finalmente decodificamos el TRACE-BACK:

↖	↖	↖	↖	↖	↖
1	2	1	2	3	4
B	**C**	**D**	**E**	**F**	**G**
B	**C**	**F**	**E**	**F**	**G**

Valor óptimo del alineamiento local: $1 \cdot (-1) + 5 \cdot (+1) = +4$.

Otro ejemplo del algoritmo de Smith-Waterman (I)

Vamos a considerar otro ejemplo de ilustración del algoritmo de Smith-Waterman.

Tratemos de alinear localmente el par de secuencias

G G A T C G A

G A A T T C A G T T A

Consideremos la siguiente matriz de similitud era:

$$s(i,j) = \begin{cases} +5 & \text{si } A[i] = B[j] \\ -3 & \text{si } A[i] \neq B[j] \end{cases}$$

y la penalización por huecos es $g = -4$.

Otro ejemplo del algoritmo de Smith-Waterman (II)

Procedamos a la fase de rellenado de la matriz en el caso del algoritmo local de Smith-Waterman.

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	5	1	0	0	0	0	0	5	1	0	0
G	0	5	2	0	0	0	0	0	5	2	0	0
A	0	1	10	7	3	0	0	5	1	2	0	5
T	0	0	6	7	12	8	4	1	2	6	7	3
C	0	0	2	3	8	9	13	9	5	2	3	4
G	0	5	1	0	4	5	9	10	14	10	6	2
A	0	1	10	6	2	1	5	14	10	11	7	11

Otro ejemplo del algoritmo de Smith-Waterman (III)

A continuación, vamos a determinar el correspondiente TRACE-BACK.

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	5	1	0	0	0	0	0	5	1	0	0
G	0	5	2	0	0	0	0	0	5	2	0	0
A	0	1	10	7	3	0	0	5	1	2	0	5
T	0	0	6	7	12	8	4	1	2	6	7	3
C	0	0	2	3	8	9	13	9	5	2	3	4
G	0	5	1	0	4	5	9	10	14	10	6	2
A	0	1	10	6	2	1	5	14	10	11	7	11

Otro ejemplo del algoritmo de Smith-Waterman (IV)

Finalmente decodificamos el TRACE-BACK: en este ejemplo hemos obtenido dos alineamientos locales óptimos.

- Primer alineamiento local óptimo:

↖	↖	↖	↖	←	↖	↑	↖
5	2	7	12	8	13	9	14
G	**G**	**A**	**T**	–	**C**	**G**	**A**
G	**A**	**A**	**T**	**T**	**C**	–	**A**

Valor óptimo del alineamiento local: $5 \cdot (+5) + 1 \cdot (-3) + 2 \cdot (-4) = +14$.

- Segundo alineamiento local óptimo:

↖	↖	↖	←	↖	↖	↑	↖
5	2	7	3	8	13	9	14
G	**G**	**A**	–	**T**	**C**	**G**	**A**
G	**A**	**A**	**T**	**T**	**C**	–	**A**

Valor óptimo del alineamiento local: $5 \cdot (+5) + 1 \cdot (-3) + 2 \cdot (-4) = +14$.

BLAST (Basic Local Alignment Search Tool)

- Programa de alineamiento local de secuencias.
- Compara una secuencia objetivo con una gran cantidad de secuencias de una cierta base de datos.
- Utiliza un algoritmo heurístico y, en consecuencia, no garantiza una solución óptima.
- Proporciona un parámetro que te permite valorar los resultados obtenidos.
- Desarrollado por los Institutos Nacionales de Salud del gobierno de EE. UU.
- Es de dominio público y se puede usar gratuitamente desde el servidor del Centro Nacional para la Información Biotecnológica (NCBI).
- Está disponible para ser instalado localmente: al usar el servidor del NCBI, el usuario no tiene que mantener ni actualizar las bases de datos y la búsqueda se hace en un cluster de computadoras.

BLOSUM (BLOcks of amino acid SUbstitution Matrix)

- Matriz de sustitución o puntuación para el alineamiento de secuencias de proteínas.
- Está basada en **alineamientos observados** y se introdujo en 1992.
- Existen muchos tipos de matrices **BLOSUM**: se basan en el mínimo porcentaje de identidad de la secuencia de proteína alineada usada al calcularlas.
- El término a_{ij} de una matriz **BLOSUM** se calcula por la siguiente fórmula:

$$a_{ij} = \frac{1}{\lambda} \log \frac{p_{ij}}{q_i \cdot q_j}$$

En donde: (a) λ es un factor de escala para asegurar que la matriz contenga valores enteros dispersos y fácilmente tratables; (b) p_{ij} es la probabilidad de que dos aminoácidos i y j reemplacen uno al otro en una secuencia homóloga; y (c) q_i (resp. q_j) es la probabilidade de encontrar el aminoácido i (resp. j) en cualquier secuencia de proteína de forma aleatoria.

BLOSUM (BLOcks of amino acid SUbstitution Matrix)

- **BLOSUM k**: matriz calculada usando las sustituciones observadas entre proteínas que tienen, como mínimo, el k % de identidades.
- **BLOSUM80**: se usa para alineamientos de secuencias con "muchas" identidades.
- **BLOSUM45**: se usa para alineamientos de secuencias con "menos" identidades.
- **BLOSUM62**: es la más estándar en programas de alineamiento.

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4