Proyecto Bioinformática (Programación en Matlab)

Implementación del algoritmo de Needleman-Wunsch en Matlab

1.- Importancia del alineamiento de secuencias

En biología molecular, nos encontramos constantemente con secuencias tanto de proteínas como de ADN. Estas secuencias llevan "encriptados" miles de años de evolución, y por ello resulta fundamental comprender las herramientas básicas que se utilizan en bioinformática a nivel de secuencia, pues nos permiten inferir qué pasó, cuándo pasó y, con esa información, entender, por ejemplo, cómo funcionan las proteínas que las células sintetizan.

Para este proyecto, vamos a simplificar una serie de conceptos, de modo que el problema que tenemos entre manos pueda ser tratable de un modo razonablemente sencillo. Vamos a considerar para ello el término de "evolución" como simple cambio a lo largo del tiempo. Las secuencias nucleotídicas (y por tanto las proteicas) van cambiando a lo largo de la historia evolutiva. Estos cambios (que consideraremos, para simplificar, aleatorios) normalmente no confieren ninguna ventaja adaptativa al portador de la secuencia mutada. En pocas ocasiones, es posible que una mutación en el ADN simplemente no afecte al individuo o bien afecte negativamente muy poco. En esos casos podemos considerar que la mutación tiene posibilidades de extenderse en una población.

La teoría del "reloj molecular" viene a decir que estos cambios neutrales a nivel de secuencia se van a cumulando a una tasa constante a lo largo de la evolución. Lo que significa que especies que divergieron hace mucho, tienen secuencias muy diferentes, mientras que especies evolutivamente más cercanas (como el hombre y el orangután) tienen secuencias más parecidas. Eso sucede en secuencias homólogas, es decir, secuencias que provienen de un antecesor común que en algún momento dio lugar a las dos especies que están siendo comparadas. Cuanto más lejos esté (desde un punto de vista evolutivo/temporal), más tiempo ha transcurrido desde la secuencia "ancestral", que era idéntica para hombre y orangután, hasta las secuencias actuales del hombre y del orangután.

Para poder comparar dos secuencias es importante el poder determinar de una manera cuantitativa dos secuencias, y determinar si son homólogas o no. El umbral para decidir si una secuencia es homóloga o no es complicado, y no vamos a entrar en detalles para este proyecto. Sin embargo, lo que resulta evidente es que necesitamos una herramienta capaz de comparar dos secuencias, y el mejor modo de hacerlo, es conseguir alinearlas.

2.- Alinear secuencias no es trivial

Las mutaciones que van acumulándose a lo largo del tiempo pueden ser (de nuevo, simplificando en la medida de lo posible) de distintos tipos. Sin embargo, si comparamos dos residuos (i=1 y j=1) entre las dos secuencias a examen (secuencia s y secuencia t), podemos encontrarnos con las siguientes posibilidades:

- $s(i=1) == t(j=1)$

- $s(i=1) \sim= t(j=1)$

Sin embargo, estamos dejando de lado la posibilidad de que una mutación haya dado lugar a una deleción, es decir, que uno o varios residuos se hayan perdido a lo largo de la evolución. Las combinaciones son ahora tres:

a) Alinear dos residuos equivalentes

b) Alinear dos residuos diferentes

c) Alinear un residuo con un "agujero" (que llamaremos *gap* a partir de ahora)

Ahora, vamos a alinear dos secuencias de ADN:

s=ACGC

t=ACGA

Aunque parezca un problema sencillo, no lo es. Existen muchas maneras de alinear esas dos secuencias, éstos son unos ejemplos:

ACGC ACGA	ACGC- ACG-A	-ACG-C A--CGA	AC--GC ACGA--	A-C-G-C -A-C-GA

Nótese que no hay en ninguno de estos ejemplos un *gap* (representado con el símbolo "-") alineado con un *gap*, pues no aporta ninguna información de "cambio con respecto al tiempo". ¿Cuál de estos alineamientos es el correcto? En realidad, no se trata de cuál es el correcto, pues lo que estamos buscando es retratar de la mejor manera posible lo que les ha sucedido a las secuencias a lo largo de la evolución. Estamos interesados en representar información biológica, eso es algo que no debemos olvidar en ningún momento. Desde ese punto de vista, los ejemplos anteriores nos dan información muy diferente: el primer alineamiento dice "ha habido una mutación de C a A en la segunda secuencia con respecto a la primera"; el segundo nos dice "ha habido una deleción de una A y una inserción de una C en la primera secuencia con respecto a la segunda"; etc…

Decidir entre todas las posibles maneras de alinear dos secuencias cuál es la biológicamente relevante es muy complicado. En lo que a nosotros nos respecta, vamos a imaginar que conseguimos obtener todos los alineamientos posibles por ahora. Lo que ahora nos preocupa es cómo puntuar los diferentes alineamientos de una manera lo más "biológica" y parsimoniosa posible. Esta cuestión es muy compleja, y no es lo que se pretende en este proyecto. Se sabe que son más frecuentes las transiciones (mutaciones entre bases nucleotídicas del mismo tipo) que las transversiones. A y G son purinas, C y T son pirimidinas. Las transiciones son, por tanto, mutaciones A/G, G/A, C/T y T/C. Las transversiones son las otras posibilidades: C/A, A/C, C/G, G/C, T/A, A/T, T/G, G/T. Con esta idea (entre otras, que no vamos a comentar aquí), podemos "puntuar" los alineamientos: en cada posición alineada vamos a puntuar positivamente (+2) el hecho de que los residuos sean iguales, con +1 a transiciones y negativamente las transversiones (-1). Además, alinear un residuo con un *gap* tendrá también una puntuación negativa (-2). Estos son los parámetros:

$$(1) S = \begin{array}{c|cccc} & A & C & G & T \\ A & +2 & -1 & +1 & -1 \\ C & -1 & +2 & -1 & +1 \\ G & +1 & -1 & +2 & -1 \\ T & -1 & +1 & -1 & +2 \end{array}$$

$$(2) g = -2$$

Ahora podemos puntuar los alineamientos que hemos propuesto anteriormente.

	ACGC ACGA	ACGC- ACG-A	-ACG-C A--CGA	AC--GC ACGA--	A-C-G-C -A-C-GA
Score	2+2+2-1= **+5**	2+2+2-2-2= +2	-2-2-2-1-2-1= -10	2-2-2-1-2-1= -6	-2-2-2-2-2-2-1= -12

3.- Needleman-Wunsch

Por fin entramos en materia… Hemos dejado de lado en el apartado anterior el cómo obtener todos los alineamientos posibles. Y es una cosa que no es sencilla en absoluto. De hecho, este proyecto consiste en encontrar el alineamiento óptimo de entre todos los posibles (dada la matriz de sustituciones y la penalización por *gap*) implementando el algoritmo de Needleman-Wunsch.

Este algoritmo lo propusieron Saul Needleman y Christian Wunsch en 1970 [1]. Es un algoritmo que resuelve el problema de optimización almacenando los *scores* máximos de las soluciones de cada *subproblema* en lugar de recalcularlos.

El algoritmo no es excesivamente complicado, vamos a ir viendo por pasos cómo funciona. En primer lugar vamos a necesitar crear una matriz D de tamaño $(m+1)\text{x}(n+1)$, donde m y n son las longitudes de las secuencias s y t respectivamente, donde vamos a ir almacenando los *scores* de los fragmentos (subproblemas de alineamiento).

Atención: vamos a usar los mismos subíndices (i y j) tanto para movernos por las secuencias $s(1 \le i \le m)$ y $t(1 \le j \le n)$ como para movernos en la matriz $D(0 \le i \le m, 0 \le j \le n)$. Esto es muy feo desde un punto de vista matemático-formal, pero espero que sirva para entender mejor cómo implementar el algoritmo.

Estamos hablando de alineamientos "globales", es decir, alineamientos de las secuencias s y t completas. El mejor *score* entre las secuencias s y t va a ser el valor de $D(m,n)$, el último valor de la tabla. Cada valor $D(i,j)$ va a ser la puntuación máxima entre el fragmento $s(1...i)$ y el fragmento $t(1...j)$. Es decir, en cada posición de la matriz D, vamos a tener el máximo *score* de los posibles alineamientos locales hasta las posiciones i,j.

$$(3) D(i,j) = max \begin{Bmatrix} D(i-1,j-1) + S(i,j) \\ D(i-1,j) + g \\ D(i,j-1) + g \end{Bmatrix}$$

Dicho de otra manera, si tenemos un alineamiento óptimo hasta $D(i\text{-}1,j\text{-}1)$, sólo hay tres opciones para lo que va a pasar en la siguiente posición:

1. Los residuos para $s(i)$ y $t(j)$ coinciden[1]
2. Se introduce un *gap* en t
3. Se introduce un *gap* en s

		i=0	i=1	i=2	i=3	i=m
		-	**A**	**C**	**G**	**C**
j=0	**-**	0	?			
j=1	**A**	?	?			
j=2	**C**					
j=3	**G**					
j=n	**A**					

Nótese que se han añadido dos *gaps* en las secuencias al principio de cada una (de ahí que el tamaño de la matriz sea de $(m+1)\text{x}(n+1)$). Esto es por que para calcular los valores en las posiciones marcadas con el signo "?" en la tabla anterior se necesita algún valor para i=0 y j=0.

Utilizando los parámetros descritos anteriormente (1) y (2) y la fórmula (3), puede rellenarse toda la matriz D. Consejo práctico 1: puede ser cómodo inicializar la matriz con ceros. Consejo práctico 2: un bucle para rellenar la primera columna y la primera fila puede no estar de más.

4.- El "traceback"

Obtenida la matriz, sólo queda reconstruir el camino desde la posición $D(m,n)$ hasta la $D(0,0)$. Para ello, habrá que recalcular los valores de $D(i\text{-}1,j\text{-}1)$, $D(i\text{-}1,j)$ y $D(i,j\text{-}1)$, para que "recordemos" de qué casilla venía el máximo correspondiente a $D(i,j)$.

- Si habíamos venido de $D(i\text{-}1,j\text{-}1)$, los residuos i y j están alineados

[1] Que "coincidan" no significa necesariamente que sean iguales, sino que coincidan en el alineamiento (que no haya un *gap*). El hecho de que sean iguales o diferentes es algo que no tiene importancia a nivel algorítmico, pues esa diferencia está incluída en la matriz *S*.

- Si habíamos venido de $D(i\text{-}1,j)$, hay que añadir un *gap* a la secuencia s
- Si habíamos venido de $D(i,j\text{-}1)$, hay que añadir un *gap* a la secuencia t

Básicamente, esto es todo. Por supuesto, puede haber más de un camino óptimo, para este proyecto es suficiente con devolver uno de ellos.

5.- ¿Qué debe hacer mi script, concretamente?

En primer lugar, lo ideal sería programar una función, no un script. Esta función debería requerir como *inputs* dos vectores con las secuencias nucleotídicas s y t

$$s(i), t(i) \in X$$
$$X = \{A, C, G, T\}$$

y tendría que devolver como *output* dos vectores y un valor escalar. Los vectores tendrían cada uno las secuencias s' y t', en este caso

$$s'(i), t'(i) \in X'$$
$$X' = \{A, C, G, T, '-'\}$$

y el escalar, debería contener el valor del *score* de ese alineamiento óptimo.

6.- Estoy perdido o atrancado. ¿Qué debo hacer?

Si has empezado con este proyecto, te aconsejo que no te rindas. Al principio puede parecer algo complicado. En cierto modo así es. Si crees que estás perdido, mándame un email y te contestaré lo más pronto posible. Piensa que si has entendido bien la base lógica del algoritmo, el implementarlo es cuestión de sintaxis. Notarás que has comprehendido la lógica que subyace a este tipo de algoritmos (llamados algoritmos de programación dinámica) cuando de pronto te digas a ti mismo, “Eh, hay algo que trasciende las letras y los números en el algoritmo... ¡Hay belleza detrás de todo esto!”.

Dudas, sugerencias y comentarios, jklett@cbm.uam.es and acortes@cbm.uam.es .

[1] Needleman SB, Wunsch CD. (1970). “A general method applicable to the search for similarities in the amino acid sequence of two proteins". *J Mol Biol* **48** (3): 443–53