Algoritmo Smith-Waterman

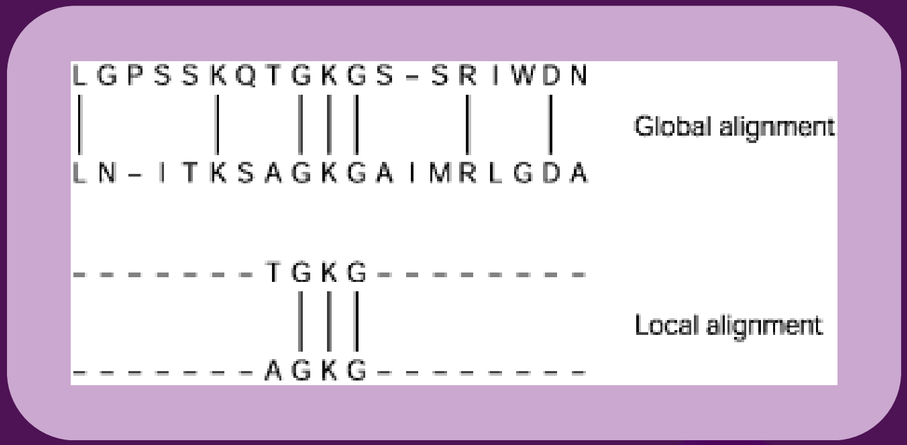
Alineamiento Local de Secuencias Biológicas

Mayra Oropeza

01

Alineamiento Local de Secuencias Biológicas

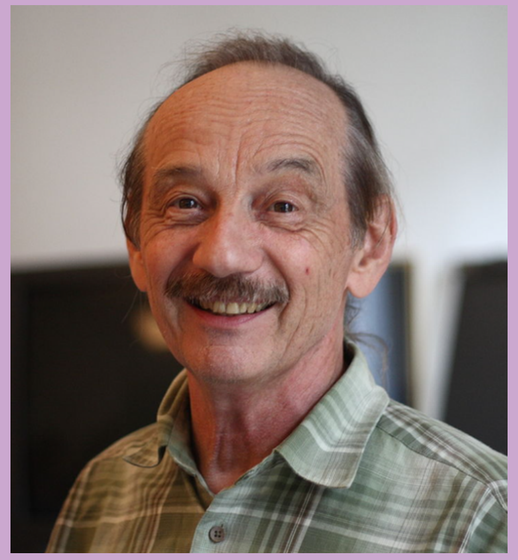
- El alineamiento local de secuencias biológicas es un método fundamental en bioinformática que se utiliza para comparar y encontrar regiones de similitud entre dos secuencias de ácidos nucleicos (como ADN o ARN) o de aminoácidos (como proteínas).
- A diferencia del alineamiento global, que busca alinear secuencias enteras, el alineamiento local se centra en identificar regiones específicas dentro de esas secuencias que tienen una similitud significativa.



02

Algoritmo Smith-Waterman

- Propuesto por Smith y Waterman en 1981, resuelve el problema de alineamiento local, enfocándose en el uso de una matriz de puntuación dinámica.



Temple F. Smith, biólogo computacional

Michael S. Waterman, biólogo computacional

02

Algoritmo Smith-Waterman

Sean $A = a_1 a_2 \dots a_n$ y $B = b_1 b_2 \dots b_m$ las dos secuencias biológicas a alinear, cuyas longitudes son n y m respectivamente. La puntuación de similitud entre dos elementos a y b esta dada por $s(a, b)$. A cada eliminación de longitud k se le asigna una penalización W_k.

1. Para encontrar un par de segmentos con una gran similitud, se construye una matriz H de $(n+1)\times(m+1)$, inicializando la primera columna y primer fila con valores de 0.

   $$H_{k0} = H_{0l} = 0 \quad para \quad 0 \le k \le n \quad y \quad 0 \le l \le m.$$

2. Cada valor H_{ij} representa la máxima similitud entre dos segmentos que terminan en a_i y b_j respectivamente. Dichos valores se obtienen de la siguiente relación de recurrencia:

   $$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ \max_{k \ge 1}\{H_{i-k,j} - W_k\}, \\ \max_{l \ge 1}\{H_{i,j-l} - W_l\}, \\ 0 \end{cases} \qquad (1 \le i \le n, 1 \le j \le m)$$

   donde

   $H_{i-1,j-1} + s(a_i, b_j)$ es la puntuación de alinear a_i y b_j,

   $H_{i-k,j} - W_k$ es la puntuación si a_i se encuentra al final de una eliminación de longitud k,

   $H_{i,j-l} - W_l$ es la puntuación si b_j se encuentra al final de una eliminación de longitud l,

   0 indica que no existe alguna similitud entre a_i y b_j, se añade este valor para evitar valores negativos.

3. Para recuperar el par de segmentos con máxima similitud, se lleva a cabo un rastreo reverso a partir del máximo elemento de H hasta terminar en un elemento cuyo valor sea igual a 0, siendo este el inicio de la alineación local óptima.

Pasos

02

Initialize the scoring matrix

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

Substitution matrix: $S(a_i, b_j) = \begin{cases} +3, & a_i = b_j \\ -3, & a_i \neq b_j \end{cases}$

Gap penalty: $W_k = kW_1$, $W_1 = 2$

Fill the scoring matrix

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3 → 1	0	0	0	3	3	
G	0	0	3 → 1	0	0	0	3	6	
T	0								
T	0								
G	0								
A	0								
C	0								
T	0								
A	0								

Substitution matrix: $S(a_i, b_j) = \begin{cases} +3, & a_i = b_j \\ -3, & a_i \neq b_j \end{cases}$

Gap penalty: $W_k = kW_1$, $W_1 = 2$

Fill the scoring matrix

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

Substitution matrix: $S(a_i, b_j) = \begin{cases} +3, & a_i = b_j \\ -3, & a_i \neq b_j \end{cases}$

Gap penalty: $W_k = kW_1$, $W_1 = 2$

Pasos

02

Identify the highest score

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

Traceback

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

3

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

Result

Sequences

Sequence 1: T G T T A C G G

Sequence 2: G G T T G A C T A

Parameters:

Substitution matrix: $S(a_i, b_j) = \begin{cases} +3, & a_i = b_j \\ -3, & a_i \neq b_j \end{cases}$

Gap penalty: $W_k = kW_1$, $W_1 = 2$

Result:

Sequence 1 G T T - A C

| | | | | | |

Sequence 2 G T T G A C

03

El algoritmo en Haskell

```haskell
type ScoreMatrix = Array (Int, Int) Int
type Traceback = Array (Int, Int) Direction

data Direction = Stop
               | LeftDir
               | Up
               | Diag
               deriving (Eq)
```

```haskell
matchScore, mismatchScore, gapScore :: Int
matchScore = 3
mismatchScore = -3
gapScore = -2
```

```haskell
-- Alineamiento
tracebackAlignment :: ScoreMatrix -> Traceback -> String -> String -> (String, String)
tracebackAlignment scoreMatrix tb s1 s2 = go (maxIndices scoreMatrix) ([], [])
  where
    go (0, 0) al = al
    go (i, j) (as1, as2) =
      case tb!(i, j) of
        Stop    -> (as1, as2)
        Diag    -> go (i-1, j-1) (s1!!(i-1) : as1, s2!!(j-1) : as2)
        Up      -> go (i-1, j) (s1!!(i-1) : as1, '-' : as2)
        LeftDir -> go (i, j-1) ('-' : as1, s2!!(j-1) : as2)

    maxIndices :: ScoreMatrix -> (Int, Int)
    maxIndices sm = fst (maximumBy (comparing snd) (assocs sm))
```

```haskell
-- Smith-Waterman
smithWaterman :: String -> String -> (ScoreMatrix, Traceback)
smithWaterman s1 s2 = (scoreMatrix, traceback)
  where
    m = length s1
    n = length s2
    bounds = ((0, 0), (m, n))
    scoreMatrix = array bounds [((i, j), score i j) | (i, j) <- range bounds]
    traceback = array bounds [((i, j), trace i j) | (i, j) <- range bounds]

    score i j
      | i == 0 || j == 0 = 0
      | otherwise = maximum [0,
                             scoreMatrix!(i-1, j-1) + delta (s1!!(i-1)) (s2!!(j-1)),
                             scoreMatrix!(i-1, j)   + gapScore,
                             scoreMatrix!(i, j-1)   + gapScore]

    trace i j
      | i == 0 || j == 0 = Stop
      | score i j == scoreMatrix!(i-1, j-1) + delta (s1!!(i-1)) (s2!!(j-1)) = Diag
      | score i j == scoreMatrix!(i-1, j) + gapScore = Up
      | score i j == scoreMatrix!(i, j-1) + gapScore = LeftDir
      | otherwise = Stop

    delta a b = if a == b then matchScore else mismatchScore
```

```haskell
-- Imprimir matriz de puntuación
print2DScoreMatrix :: ScoreMatrix -> IO ()
print2DScoreMatrix sm = mapM_ putStrLn [intercalate " " (map (show . (sm!)) [(i, j) | j <- [0..n]]) | i <- [0..m]]
  where
    ((_, _), (m, n)) = bounds sm
```

03

Resultado

```
*Main> main
Ingrese la primera secuencia:
TGTTACGG
Ingrese la segunda secuencia:
GGTTGACTA
Matriz de puntuación:
0 0 0 0 0 0 0 0 0 0
0 0 0 3 3 1 0 0 3 1
0 3 3 1 1 6 4 2 1 0
0 1 1 6 4 4 3 1 5 3
0 0 0 4 9 7 5 3 4 2
0 0 0 2 7 6 10 8 6 7
0 0 0 0 5 4 8 13 11 9
0 3 3 1 3 8 6 11 10 8
0 3 6 4 2 6 5 9 8 7

Alineamientos:
GTT-AC
GTTGAC
```

Traceback

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

3

```
G T T - A C
| | |   | |
G T T G A C
```

¡Gracias!