Algoritmo Smith-Waterman

- Laura D. Becerra Largo
- Jonnatan N. Hincapié Hernández
- Alejandro Valencia Ossa
- Mateo Cañavera Aluma

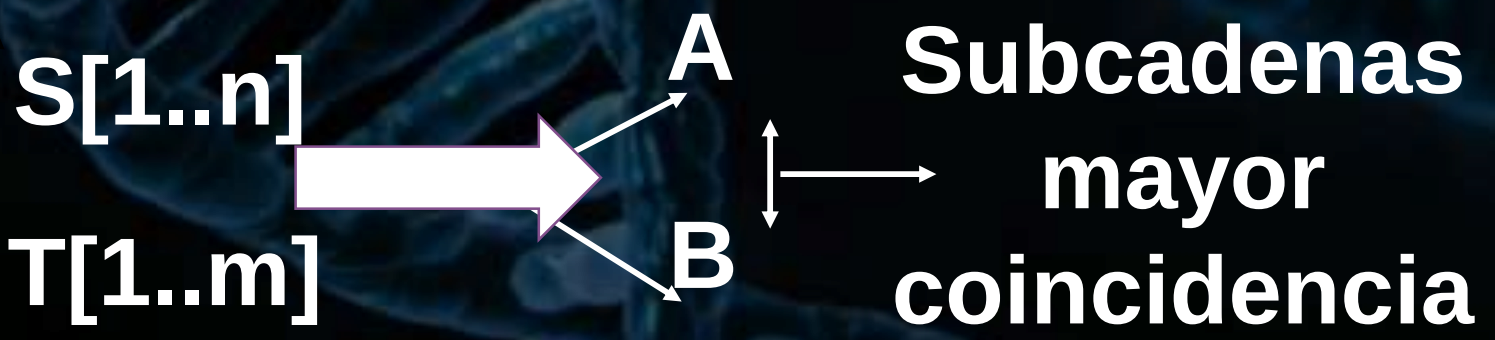
Universidad Nacional De Colombia

Contenido

- Contexto
- Problema a resolver
- Implementación del algoritmo en código
  - Lectura de secuencias
  - Matriz de puntuación
  - Matriz de recorrido
  - Alineamiento y escritura del archivo de salida
- Archivo de salida
- Otras implementaciones del algoritmo Smith-Waterman
- Conclusiones

¿Qué hace?

Alineamiento local



¿Cómo?

1)

- 1) 1)

1)

$$V(i,0) = 0 \text{ for } 0 \leq i \leq n$$
$$V(0,j) = 0 \text{ for } 0 \leq j \leq m$$

- 1)

2)

$$V(i,j) = \max \begin{cases} 0 \\ V(i-1,j-1) + \delta(S[i], T[j]) \\ V(i-1,j) + \delta(S[i], \_) \\ V(i,j-1) + \delta(\_, T[j]) \end{cases}$$

Ejemplo:

$S = ACAATCG$ $T = CTCATGC$

	_	C	T	C	A	T	G	C
_	0	0	0	0	0	0	0	0
A	0	0	0	0	2	1	0	0
C	0	2	1	2	1	1	0	2
A	0	1	1	1	4	3	2	1
A	0	0	0	0	3	3	2	1
T	0	0	2	1	2	5	4	3
C	0	2	1	4	3	4	4	6
G	0	1	1	3	3	3	6	5

CAATCG
C-AT-G

Implementación del algoritmo en código

- Leer secuencias
- Matriz de puntuación
- Matriz de recorrido
- Alineamiento y resultados
- Llamar funciones anteriores

Leer Secuencias

```
#Se importa libreria Biopython, que permite leer secuencias geneticas en formato fasta
""" Este Fracmento de codigo se ejecuta solo si es en google colaboratory
try:
    import google.colab
    !pip install biopython
except ImportError:
    pass
"""

#Se importa las librerias numpy y pandas para operar matrices que se obtendran en el algoritmo, adémas de las funciones necesarias de Biopython para leer el archivo tipo
import numpy as np
import pandas as pd
from Bio import SeqIO
import math
import sys

#Se lee el archivo que contiene las secuencias a analizar y se guardan en la lista 'record'
#Este algoritmo compara la primera secuencia del archivo con la demás
record = list(SeqIO.parse(sys.argv[1], "fasta"))
"""Solo se ejecuta si es en google colaboratory, eliminando la linea anterior a este mensaje
record = list(SeqIO.parse("/content/prueba4.fasta", "fasta"))"""
```

Matriz de Puntuación

	_	A	G	C	A	T	G	C
_	0	-1	-2	-3	-4	-5	-6	-7
A	-1	2	1	0	-1	-2	-3	-4
C	-2	1	1	3	2	1	0	-1
A	-3	0	0	2	5	4	3	2
A	-4	-1	-1	1	4	4	3	2
T	-5	-2	-2	0	3	6	5	4
C	-6	-3	-3	0	2	5	5	7
C	-7	-4	-4	-1	1	4	4	7

```python
def matriz_puntuacion(record):
    ##Guarda la matriz de puntuación de la secuencia con mayor similitud
    matrix_max = 0
    ##Guarda en un diccionario el mayor coeficiente obtenido en la matriz de puntuación para cada secuencia comparada
    coef_simil={}
    ##Guarda la posición del mayor coeficiente en la matriz de puntuación de la secuencia con mayor similitud
    coef_max_posicion = [0,0]
    ##Guarda el mayor coeficiente obtenido en la matriz de puntuación de la secuencia con mayor similitud
    coef_max = 0
    ##Guarda posición de la secuencia en la lista de secuencias 'record'
    sec_pos = 0

    #para acceder a cada secuencia se hace por medio de un ciclo:
    n = 0
    for r in range(len(record)-1):
        ##Guarda longitud de la secuencia con la que se comparan las demás
        fila = len(record[0]) + 1
        ##Guarda longitud de la secuencia que se comparará con la primera
        columna = len(record[r+1]) + 1
        ##se crea matriz de puntuación con todas sus entradas en cero
        matrix = np.zeros(shape=(fila, columna), dtype=int)
        ##Guarda la posición del mayor coeficiente en la matriz de puntuación
        max_posicion = [0,0]
        ##Guarda el mayor coeficiente obtenido en la matriz de puntuación
        val_max = 0

        #Este ciclo anidado se encarga de comparar la primera secuencia con la 'r' secuencia por medio del  Algo
        ##i representará el i-ésimo nucleotido de la primera secuencia de ARN
        for i in range(1,fila):
            ##j representará el j-ésimo nucleotido de la 'r' secuencia de ARN a comparar
            for j in range(1,columna):
                ##Se compara el nucleotido i con cada nucleotido j de la 'r' secuencia y se le asigna una puntua
                if record[0][i-1] == record[r+1][j-1]:###buscar otro metodo(numpy)
                    coincidencia = 2
                else:
                    coincidencia = -1
                ##se calcula la relación de recurrencia que indica el algoritmo de Smith-Waterman
                diagonal=matrix[i-1][j-1] + coincidencia
                arriba = matrix[i-1][j] + (-1)
                izquierda = matrix[i][j-1] + (-1)
                ##se elige la puntuación de similitud con la relación de recurrencia y se guarda en la matriz creada para tal fin (matrix)
                matrix[i][j] = max(diagonal, arriba, izquierda,0)
                ##Se guarda el mayor coeficiente de puntuación y su posición en la matriz de puntuación
                if val_max <= matrix[i][j]:  (variable) j: int
                    val_max = matrix[i][j]
                    max_posicion[0]=i
                    max_posicion[1]=j
        n +=1
        ###fin del ciclo anidado###
        ##Se guarda el mayor coeficiente de puntuación para cada secuencia en un diccionario, donde el key es la identificación de la secuencia
        id = record[r+1].id
        coef_simil.setdefault(id,val_max)
        ##Se guarda el con mayor coeficiente de puntuación entre todas las secuencias con su respectiva matriz y posición del coeficiente en esta ultima
        if coef_max <= val_max:
            coef_max = val_max
            coef_max_posicion = max_posicion
            matrix_max = matrix
            sec_pos = n
    ###fin del ciclo de comparación###
    return [coef_max_posicion, sec_pos, matrix_max, coef_simil]
```

Matriz de recorrido

```python
def matriz_recorrido(record, matrix_max, sec_pos):
    ##se crea matriz que indica el camino de similitud para la secuencia con mayor similitud respecto a la primera secuencia
    matrix2ruta = np.zeros(shape=(len(matrix_max),len(matrix_max[0]) ), dtype=int)
    ##se recorre la matriz de maxima similitud siguendo los parametros descritos en el algoritmo de Smith-Waterman para crear
    # La nueva matriz que indica el recorrido
    for i in range(1,len(matrix_max)):
        for j in range(1,len(matrix_max[0])):
            ##se calcula nuevamente la puntuación de coincidencia (solo para la matriz de maxima similitud)
            if record[0][i-1] == record[sec_pos][j-1]:
                coincidencia = 2
            else:
                coincidencia = -1
            ##se calcula la relación de recurrencia que indica el algoritmo de Smith-Waterman
            diagonal=matrix_max[i-1][j-1] + coincidencia
            arriba = matrix_max[i-1][j] + (-1)
            izquierda = matrix_max[i][j-1] + (-1)
            ##se llena la matriz de recorrido
            if matrix_max[i][j] == 0 :
                matrix2ruta[i][j]=0
            elif matrix_max[i][j] == diagonal:
                matrix2ruta[i][j]=1
            elif matrix_max[i][j] == arriba:
                matrix2ruta[i][j]=2
            elif matrix_max[i][j] == izquierda:
                matrix2ruta[i][j]=3
    ###fin del ciclo para llenado de matriz de recorrido###
    return matrix2ruta
```

	_	A	G	C	A	T	G	C
_	0	-1	-2	-3	-4	-5	-6	-7
A	-1	2	1	0	-1	-2	-3	-4
C	-2	1	1	3	2	1	0	-1
A	-3	0	0	2	5	4	3	2
A	-4	-1	-1	1	4	4	3	2
T	-5	-2	-2	0	3	6	5	4
C	-6	-3	-3	0	2	5	5	7
C	-7	-4	-4	-1	1	4	4	7

Alineamiento y resultados

```
##se crea el string que guardará los simbolos correspondiente al alineamiento entre las secuencias
alin=""
##Se crea contador que permite saber el total de nucleotidos que se alinean entre ambas secuencias
favor=0
##Se invierte los strings que indican las secuencias comparadas (al revés)
comparacion=comparacion[::-1]
principal=principal[::-1]

##se llena la linea que indica el alineamiento entre secuencias comparando las secuencias por medio de un ciclo
for i in range(len(comparacion)):
  if principal[i]==comparacion[i]:
    ##'|'indica que en ambas cadenas de ARN se encuentra el mismo nucleotido en la misma posición
    alin=alin+"|"
    favor = favor + 1
  else:
    ##' ' indica que no hay similitud en la i-ésima posición de las secuencias
    alin=alin+" "

##se convierte el diccionario con los coeficientes de similitud de cada secuencia en un DataFrame
dataf = pd.DataFrame([[key, coef_simil[key]] for key in coef_simil.keys()], columns=['Id de la secuencia', 'Coef. de similitud'])
##Se calcula el porcentaje de similitus entre las secuencias
P = (favor/len(record[0]))*100
per = round(P,3)
###dividir el alineamiento de la secuenciencia#####
M=math.ceil(len(principal)/100)
##se crea y se escribe en un archivo de texto toda la información analizada en el algoritmo
doc = open('comparacion_similitud.txt','w')
doc.write(f'La secuencia principal, con la que se compararon las demás: {record[0].id}')
doc.write(f'\n')
doc.write(f'La secuencia con mayor similitud a la secuencia principar fue {record[sec_pos].id} con un')
doc.write(f'\n')
```

Alineamiento y resultados

```
doc.write(f'porcentaje de similitud del {per}%')
doc.write(f'\n')
doc.write(f'\n')
doc.write(f'A continuación se muestra el alineamiento entre ambas secuencias:')
doc.write(f'\n')
for i in range(M):
    doc.write(f'{principal[100*i:100*(i+1)]}')
    if i == M-1:
        doc.write(f'----{record[0].id}')
    doc.write('\n')
    doc.write(f'{alin[100*i:100*(i+1)]}')
    doc.write('\n')
    doc.write(f'{comparacion[100*i:100*(i+1)]}')
    if i == M-1:
        doc.write(f'----{record[sec_pos].id}')
    doc.write('\n')
doc.write(f'\n')
doc.write(f'\n')
doc.write(f'Además se presenta la siguiente tabla que relaciona cada una de las secuencias comparadas con su')
doc.write(f'\n')
doc.write(f'coeficiente de similitud, que corresponde a la puntuación maxima de similitud que tuvo cada secuencia')
doc.write(f'\n')
doc.write(f'respecto a la secuencia principal')
doc.write(f'\n')
doc.write(f'\n')
doc.write(f'{dataf}')
doc.close()
```

Llamar funciones anteriores

```
punt = matriz_puntuacion(record)
matr = matriz_recorrido(record, punt[2], punt[1])
resultados(record, matr, punt[0], punt[3], punt[1])
```

Archivo de salida

```
La secuencia principal, con la que se compararon las demás: gi|645322056|ref|NR_118889.1|
La secuencia con mayor similitud a la secuencia principar fue gi|645322058|ref|NR_118890.1| con un
porcentaje de similitud del 93.538%

A continuación se muestra el alineamiento entre ambas secuencias:
GGTCTNATACCGGATATAACAACTCATGGCATGGTTGGTAGTGGAAAGCTCCGGCGGTACGGGATGAGCCCGCGGCCTATCAGCTTGTTGGTGGGGTAAT
|||||||||||||||||| |||||| ||||||||||||| |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||
GGTCTNATACCGGATATGACAACTGATGGCATGGTTGGTTGTGGAAAGCTCCGGCGGTCCGGGATGAGCCCGCGGCCTATCAGCTTGTTGGTGGGGTAAT
GTNAATACGTTCCCGGGCCTTGTACACACCGGNCGTTACGGCATGAAAGNCGGTAACACCCGAA-CCCATGGCCCAACCCGTAA-GGGGGGAGTNGTCGA
|||||||||||||||||||||||||||||| |||| ||| |||||||| ||||||||||||| |||||||||||||||||| ||||||||| |||||
GTNAATACGTTCCCGGGCCTTGTACACACCGCNCGTGACGTCATGAAAGTCGGTAACACCCGAAGCCCATGGCCCAACCCGTAAGGGGGGGAGTGGTCGA
AGGTGGGACTNGCGAT----gi|645322056|ref|NR_118889.1|
|||||||||| |||||
AGGTGGGACTGGCGAT----gi|645322058|ref|NR_118890.1|


Además se presenta la siguiente tabla que relaciona cada una de las secuencias comparadas con su
coeficiente de similitud, que corresponde a la puntuación maxima de similitud que tuvo cada secuencia
respecto a la secuencia principal

                Id de la secuencia  Coef. de similitud
0  gi|645322068|ref|NR_118899.1|                2051
1  gi|444303911|ref|NR_074334.1|                1415
2  gi|645322058|ref|NR_118890.1|                2332
```