Smith-Waterman Algorithm

Local Alignment with Affine Gap Penalty

by Alyssa Richardson, Sean Marchand & Zenen Treadwell

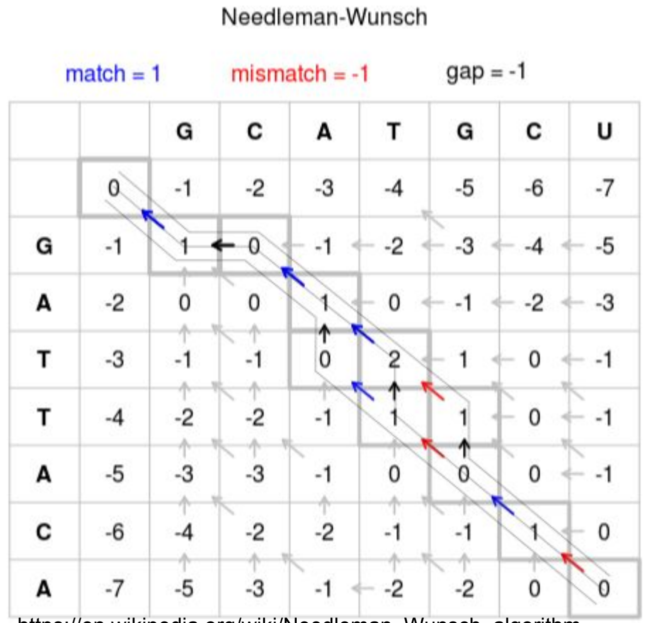
Table of Contents

Needleman-Wunsch algorithm

Created by Saul B. Needleman and Christian D. Wunsch in 1970

Goal is to find the optimal global alignment

Find the optimal path through the matrix from corner to corner



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

https://en.wikipedia.org/wiki/Needleman–Wunsch_algorithm

History of Needleman-Wunsch

Original purpose was to find the similarities in amino acid sequences between two proteins

Originally had no penalty for gaps, only based on matches and mismatches

Waterman Et al. introduced the concept of gaps in 1976

History of Smith-Waterman algorithm

Created by Temple F. Smith and Michael S. Waterman in 1981 as a variation of the Needleman-Wunsch algorithm

Genome projects generated a large amount of data that needed to be processed, sequence alignment algorithms allowed the data to be more understood

Local alignment became necessary due to mutations making DNA comparisons less meaningful

The Smith-Waterman algorithm can be made to return an expectation value, when this value is low, it could mean that two sequences share a common ancestor

Smith-Waterman

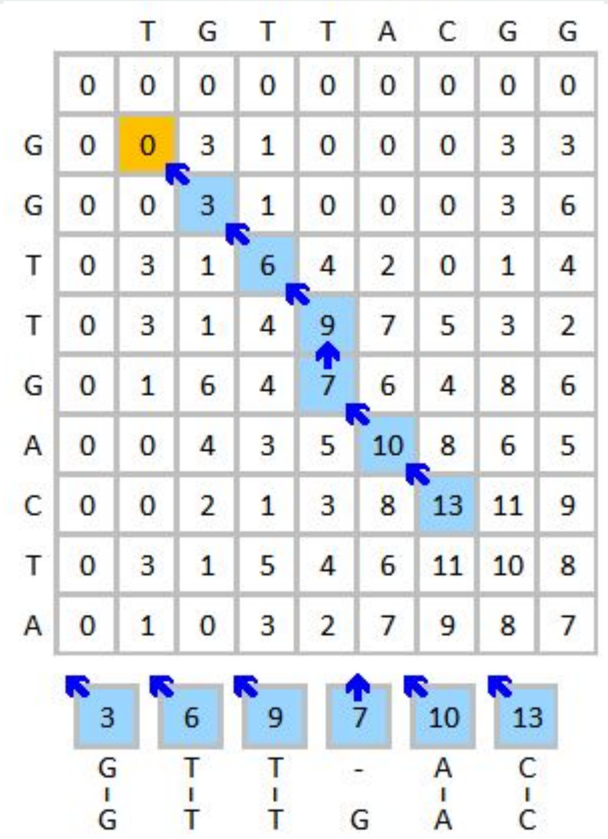
Variation of Needleman-Wunsch, for local alignment

Differences:

-Negative scores are set to zero

` -Traceback begins at the highest scoring cell

-Stop once a zero is found in the 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

https://en.wikipedia.org/wiki/Smith–Waterman_algorithm

Linear gap penalty

Whenever a gap is added, subtract a set amount from the score

The same cost is used for both opening and extending a gap

Different to constant gap penalty, which gives the same score to any gap, regardless of length

Affine Gap

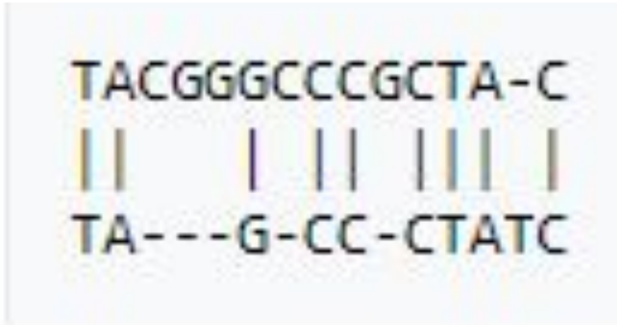
Linear penalty for gaps adds a static number each time a gap is added

Affine gap considers gap opening and extension separately

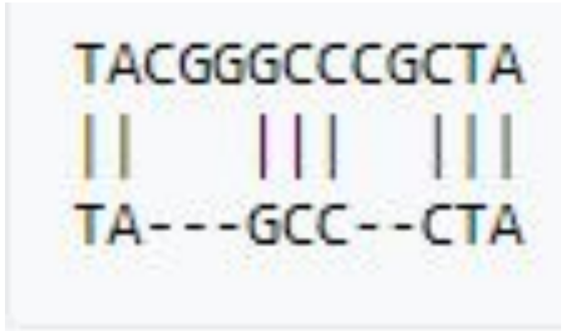
Affine gap is the more widely used version

High gap opening costs are used to discourage many small gaps, but to instead use one large

Linear Gap Penalty vs Affine Gap Penalty



```
TACGGGCCCGCTA-C
||   | || ||| |
TA---G-CC-CTATC
```



```
TACGGGCCCGCTA
||   |||  |||
TA---GCC--CTA
```

https://en.wikipedia.org/wiki/Gap_penalty

Implementation

Ask for input

Give input to makeMatrix

makeMatrix calls score do determine what to place in each square

When makeMatrix returns, perform the traceback to determine the optimal alignment

Examples of optimizations

In the original Smith-Waterman paper, runtime was O(m^2 n)

Gotoh optimized it to be O(mn), but it would only find one alignment

Altschul optimized to produce all optimal alignments, while still being O(mn)

In 1975, Myers and Miller optimized the space complexity to O(n)

Applications

Sequence alignment of proteins and nucleic acids

Levenshtein distance (spell checks, fuzzy string searching, comparing languages)

BLOSUM

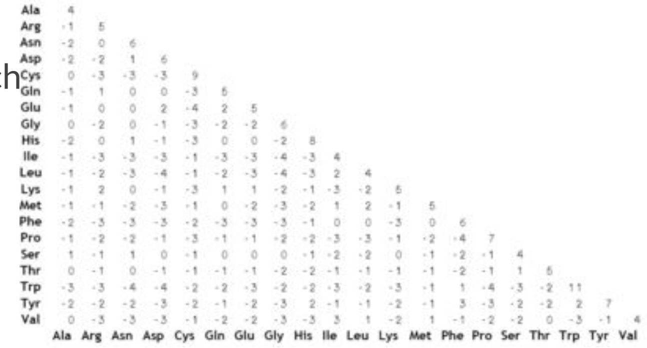
BLOSUM

BLOcks **SU**bstitution **M**atrix

substitution matrices used for alignment of evolutionarily divergent proteins

used to understand gene variance in hepatitis B virus carriers

Used in BLAST (Basic Local Alignment Search



	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

https://en.wikipedia.org/wiki/BLOSUM

Future of this research

Recent developments in this area are based around improving the time and space complexity, while keeping the same functionality

Finding new applications of the algorithm

Thank you for listening