Dynamic Programming- continued

Semi-Global Alignment: Dealing with Overhanging Termini: Overlapping Alignments

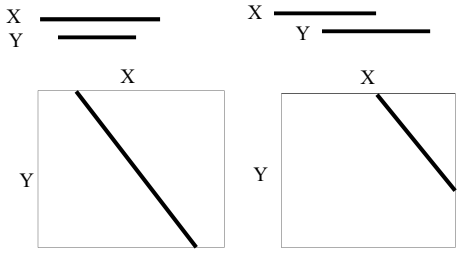
In the previous *global* alignment the whole of sequence X is aligned to the whole of sequence Y.

If one sequence is longer than the other then gaps must be inserted either within the sequence of the shorter one or at the termini.

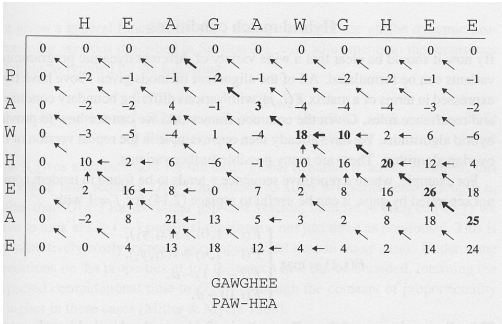
"gaps" at the beginning and end of the alignment were scored as gap penalties.

Overhanging termini should really be penalized less than true gaps.

The following algorithm doesn't penalize overhanging ends. This algorithm can be used in those cases when one sequence is contained in the other or they overlap.



- F(i,0)=0 and F(0,j)=0 so the first row and column are filled with 0's not with an increasing gap penalty.
- The algorithm is the same as previously but traceback starts from the maximum score along the last row or column, and ends on the first row or column reached.



This is dynamic programming matrix gives the score as 25 compared to 1 for the algorithm where overhanging ends are scored as gaps.

This is global alignment as we still consider the whole sequences even if overhanging ends are scored 0.

What if we just want to find the optimal alignment of subsequences within both sequences? This is called “local alignment” and we use the so-called Smith-Waterman algorithm.

Local Alignment: The Smith-Waterman Algorithm

The algorithm finds a pair of segments from each of two long sequences with the maximum score.

$$F(i,j)= \max \begin{cases} F(i-1,j-1)+s(x_i,y_j) \\ F(i-1,j)+g \\ F(i,j-1)+g \\ 0 \end{cases}$$

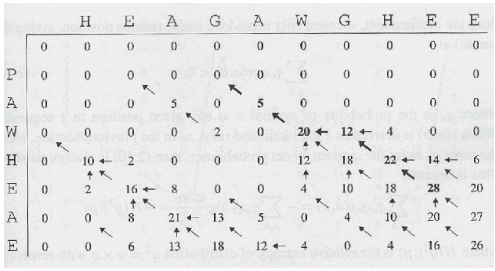
This means F(i,j)=0 is chosen if the other 3 choices are negative. Again the top row and left column are filled with 0’s.

Traceback starts from the cell with the highest score (which can be anywhere within the DP matrix) and ends at the first cell with 0.

Note: for algorithm to work expected score for a random match must be negative. This is the case.

Smith and Waterman. 1981. Identification of common molecular subsequences, J. Mol. Biol,**147**, 195-197

Example, again using the sequences: HEAGAWGHEE and PAWHEAE



		H	E	A	G	A	W	G	H	E	E
	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	5	0	**5**	0	0	0	0	0
W	0	0	0	0	2	0	**20**	**12**	4	0	0
H	0	10	2	0	0	0	12	18	**22**	14	6
E	0	2	16	8	0	0	4	10	18	**28**	20
A	0	0	8	21	13	5	0	4	10	20	27
E	0	0	6	13	18	12	4	0	4	16	26

Exercise: What is the maximum local alignment and its score? 28, AWGHE / AW-HE

This gives just a *single* best local match, but there may be many local alignments that have significant scores. For example X may be a long sequence within which a domain or motif is repeated and within Y a similar motif occurs.

Affine Gap Penalty

```
GAAGAATTCCACA
|||     ||||
GAA----TCCA--
```

```
GAAGAATTCCACA
|  |  |  |  |   ||
G-A-A-T-C--CA
```

Up to here we have used a linear gap penalty where:

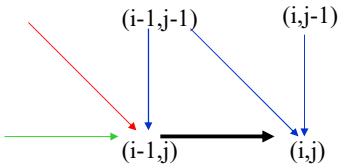
$$\gamma(n) = ng$$

The affine gap penalty punishes the opening of a gap more than the extension of an already opened gap.

$$\gamma(n) = g + (n-1)e$$

g gap opening penalty (-ve)
e gap extension penalty (-ve)
n gap length
$|e| < |g|$

The score will now depend on the route into the previous cell. For example:



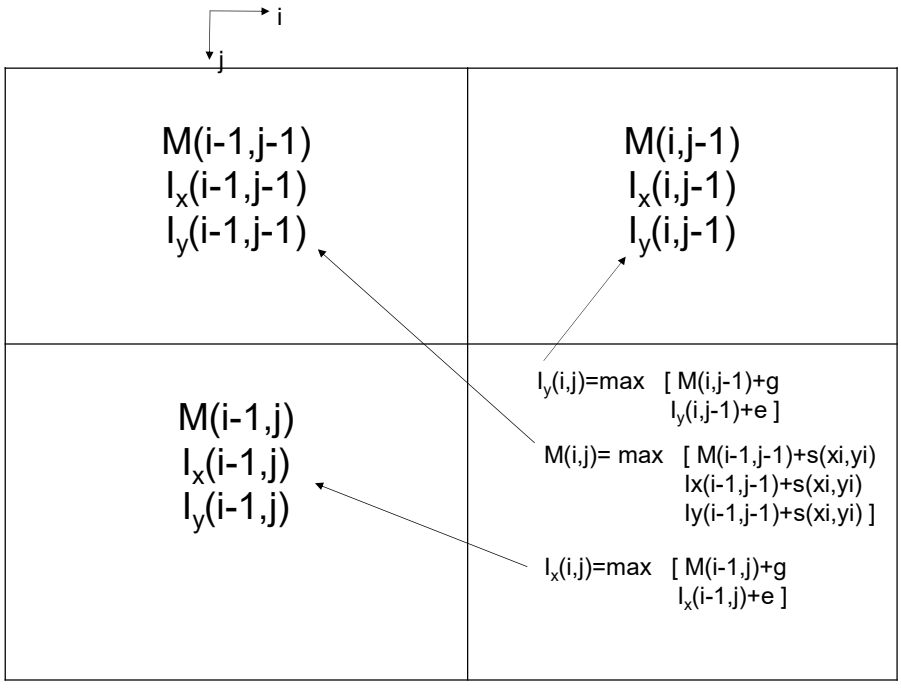
In going from (i-1,j) to (i,j), if via red add g, if via green add e. So each cell needs to store 3 scores each derived from the 3 directions into the cell.

$$M(i,j) = \max \begin{cases} M(i-1,j-1)+s(x_i,y_j) \\ I_x(i-1,j-1)+ s(x_i,y_j) \\ I_y(i-1,j-1)+ s(x_i,y_j) \end{cases}$$

$$I_x(i,j) = \max \begin{cases} M(i-1,j)+g & \text{red path} \\ I_x(i-1,j)+e & \text{green path} \end{cases}$$

$$I_y(i,j) = \max \begin{cases} M(i,j-1)+g \\ I_y(i,j-1)+e \end{cases}$$

The traceback principle applies.



Substitution matrix: Identity

g=-4
e=-1

ACACT
--AAT score = -3

ACACT
A--AT score = -3

ACACT
AA--T score= -3

		A	C	A	C	T
	M=0 I_x=0 I_y=0	M=-∞ I_x=-4 I_y=-∞	M=-∞ I_x=-5 I_y=-∞	M==∞ I_x=-6 I_y=-∞	M=-∞ I_x=-7 I_y=-∞	M=-∞ I_x=-8 I_y=-∞
A	M=-∞ I_x=-∞ I_y=-4	M=1 I_x=-∞ I_y=-∞	M=-4 I_x=-3 I_y=-∞	M=-4 I_x=-4 I_y=-∞	M=-6 I_x=-5 I_y=-∞	M=-7 I_x=-6 I_y=-∞
A	M=-∞ I_x=-∞ I_y=-5	M=-3 I_x=-∞ I_y=-3	M=1 I_x=-7 I_y=-8	M=-2 I_x=-3 I_y=-8	M=-4 I_x=-4 I_y=-10	M=-5 I_x=-5 I_y=-11
T	M=-∞ I_x=-∞ I_y=-6	M=-5 I_x=-∞ I_y=-4	M=-3 I_x=-9 I_y=-3	M=1 I_x=-7 I_y=-6	M=-2 I_x=-3 I_y=-8	M=-3 I_x=-4 I_y=-9

Multiple Sequence Alignment

If we have more than 2 sequences what would an alignment of these sequences look like? The following shows the alignment of 8 fragments from immunoglobulin sequences. The alignment is a table where each row is a sequence and each column a base or residue (or gap) in the alignment. If sequences come from the same family mulitple alignments can reveal conserved regions (perhaps due to structural or functional constraints) and evolutionary relationships to a much greater degree than simple pairwise alignments.

```
VTISCTGSSSNIGAG-NHVKWYQQLPG
VTISCTGTSSNIGS--ITVNWYQQLPG
LRLSCSSSGFIFSS--YAMYWVRQAPG
LSLTCTVSGTSFDD--YYSTWVRQPGG
PEVTCVVVDVSHEDPQVKFNWYVDG--
ATLVCLISDFYPGA--VTVAWKADS--
AALGCLVKDYFPEP--VTVSWNSG---
VSLTCLVKGFYPSD--IAVEWESNG--
```

Scoring Multiple Sequence Alignments

Before methods to do multiple alignments can be introduced a scoring system is required. The algorithms that perform multiple alignments aim to determine the alignment with the maximum score. There are a number of scoring methods, but the standard one is calculated like this:

Consider one column of the table:

L
L
A
P
G
S
-
G

Then the scoring system employed scores all possible pairs of letters. So the total cost for this column is given as:

S=s(1,2)+s(1,3)+s(1,4)+s(1,5)+s(1,6)+s(1,7)+s(1,8)+s(2,3)+s(2,4)+….+s(7,8)

Scoring Multiple Sequence Alignments

To get the total score this procedure has to done for all columns. Formally, the score S is:

$$S = \sum_i S(m_i)$$

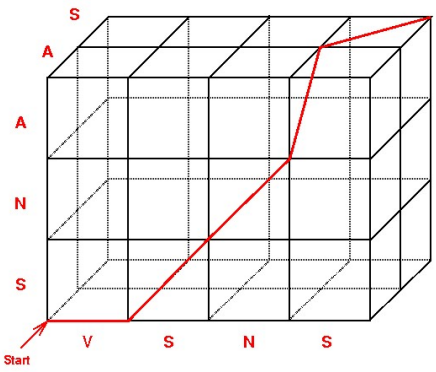
$$S(m_i) = \sum_{k<l} s(m_i^k, m_i^l)$$

This is known as the, Sum of Pairs score, “SP-score.”

Note that in this approach can only be used with the linear gap score, whereby:

s(-,a)=g
s(a,-)=g
s(-,-)=0

Alignment of three sequences **VSNS, SNA, AS**



Exercise: What is the alignment of the 3 sequences?

VSN-S
-SNA-
---AS

Alignment of 3 Sequences

The algorithm to align 3 sequences is just an extension of the pairwise alignment algorithm of Needleman and Wunsch:

$$F(i,j,k) = \max \begin{cases} F(i-1,j-1,k-1) + S(x_i, y_j, z_k) \\ F(i,j-1,k-1) + S(-, y_j, z_k) \\ F(i-1,j,k-1) + S(x_i, -, z_k) \\ F(i-1,j-1,k) + S(x_i, y_j, -) \\ F(i,j,k-1) + S(-,-,z_k) \\ F(i,j-1,k) + S(-, y_j, -) \\ F(i-1,j,k) + S(x_i, -, -) \end{cases}$$

Just like in for two sequences the optimal alignment can be viewed as a path through a matrix except now it is a 3D matrix rather than a 2D one. In the 2D case there were 3 choices in this case there are 7.

Exercise: Satisfy yourself that in the case of a cube results in 7 choices.

Alignment of N Sequences

Alignment of N sequences is just a generalization of the DP algorithm to N-dimensions. The algorithm is:

$$F(i_1,i_2,...,i_N) = \max\begin{cases} F(i_1-1,i_2-1,i_3-1,...,i_N-1)+S(x_{i_1},x_{i2},x_{i_3},..,x_{i_N}) \\ F(i_1,i_2-1,i_3-1,...,i_N-1)+S(-,x_{i_2},x_{i_3},..,x_{i_N}) \\ F(i_1-1,i_2,i_3-1,...,i_N-1)+S(x_{i_1},-,x_{i_3},..,x_{i_N}) \\ \vdots \\ F(i_1-1,i_2-1,i_3-1,...,i_N)+S(x_{i_1},x_{i2},x_{i_3},..,-) \\ F(i_1,i_2,i_3-1,...,i_N-1)+S(-,-,x_{i_3},..,x_{i_N}) \\ \vdots \\ F(i_1,i_2-1,i_3-1,...,i_N)+S(-,x_{i2},x_{i_3},..,-) \\ \vdots \end{cases}$$

Exercise: How many choices are there? 2^N-1

The algorithm is exact, but is enormously expensive in both memory, $O(L^N)$, and time, $O(2^N L^N)$. Alternatives must be sought to make multiple alignment feasible for all but a few sequences.