

# Pairwise Sequence Alignment (Continued)

BMI/CS 576

[www.biostat.wisc.edu/bmi576/](http://www.biostat.wisc.edu/bmi576/)

Mark Craven

[craven@biostat.wisc.edu](mailto:craven@biostat.wisc.edu)

Fall 2011

## Local alignment

- so far we have discussed *global alignment*, where we are looking for best match between sequences from one end to the other
- often we want a *local alignment*, the best match between subsequences of  $x$  and  $y$

## Example local alignment

- aligning my name against the sequence for dTDP-4-dehydrorhamnose reductase from the bacterium *opitutus terrae*

..LSGAYHLAASGHTSWHGFASAIIDLMPDARKCRAVEAIT..  
MARKCRAVEN

## Local alignment motivation

- useful for comparing protein sequences that share a common *motif* (conserved pattern) or *domain* (independently folded unit) but differ elsewhere
- useful for comparing DNA sequences that share a similar *motif* but differ elsewhere
- useful for comparing protein sequences against *genomic DNA sequences* (long stretches of uncharacterized sequence)
- more sensitive when comparing highly diverged sequences

## Local alignment DP algorithm

- original formulation: Smith & Waterman, *Journal of Molecular Biology*, 1981
- interpretation of array values is somewhat different:  
 $F(i, j)$  = score of the best alignment of a suffix of  $x[1\dots i]$  and a suffix of  $y[1\dots j]$

## Local alignment DP algorithm

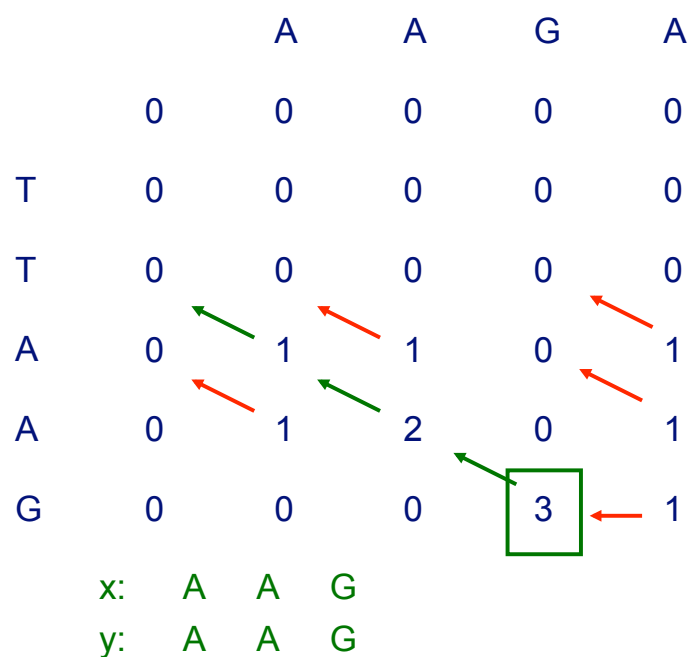
- the recurrence relation is slightly different than for global algorithm

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

# Local alignment DP algorithm

- initialization: first row and first column initialized with 0's
- traceback:
  - find maximum value of  $F(i, j)$ ; can be anywhere in matrix
  - stop when we get to a cell with value 0

## Local alignment example



## More on gap penalty functions

- a gap of length  $k$  is more probable than  $k$  gaps of length 1
  - a gap may be due to a single mutational event that inserted/deleted a stretch of characters
  - separated gaps are probably due to distinct mutational events
- a linear gap penalty function treats these cases the same
- it is more common to use gap penalty functions involving two terms
  - a penalty  $d$  associated with opening a gap
  - a smaller penalty  $e$  for extending the gap

## Gap penalty functions

linear

$$w(g) = -g \times d$$

affine

$$w(g) = \begin{cases} -d - (g - 1)e, & g \geq 1 \\ 0, & g = 0 \end{cases}$$

## Dynamic programming for the affine gap penalty case

- to do in  $O(n^2)$  time, need 3 matrices instead of 1

$M(i, j)$       best score given that  $x[i]$  is aligned to  $y[j]$

$I_x(i, j)$       best score given that  $x[i]$  is aligned to a gap

$I_y(i, j)$       best score given that  $y[j]$  is aligned to a gap

## Global alignment DP for the affine gap penalty case

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

$$I_x(i, j) = \max \begin{cases} M(i-1, j) - d \\ I_x(i-1, j) - e \end{cases}$$

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

# Global alignment DP for the affine gap penalty case

- initialization

$$M(0,0) = 0$$

$$I_x(i, 0) = -d - (i-1)e \quad \text{for } i > 0$$

$$I_y(0, j) = -d - (j-1)e \quad \text{for } j > 0$$

other cells in top row and leftmost column =  $-\infty$

- traceback

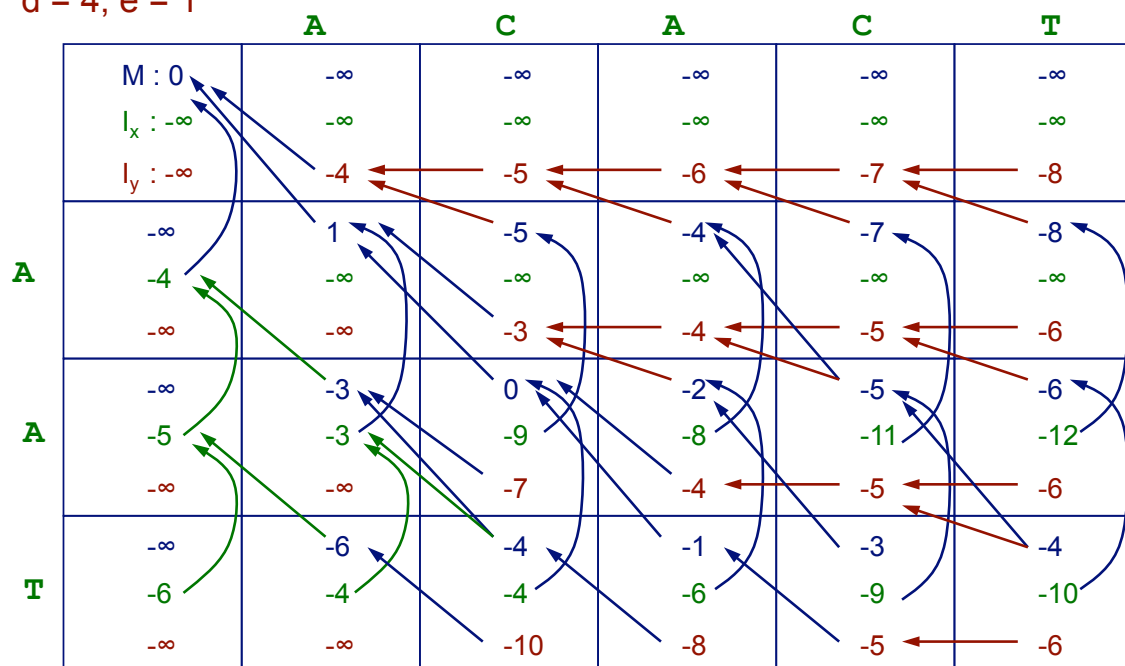
- start at largest of  $M(m, n), I_x(m, n), I_y(m, n)$

- stop at  $M(0,0)$

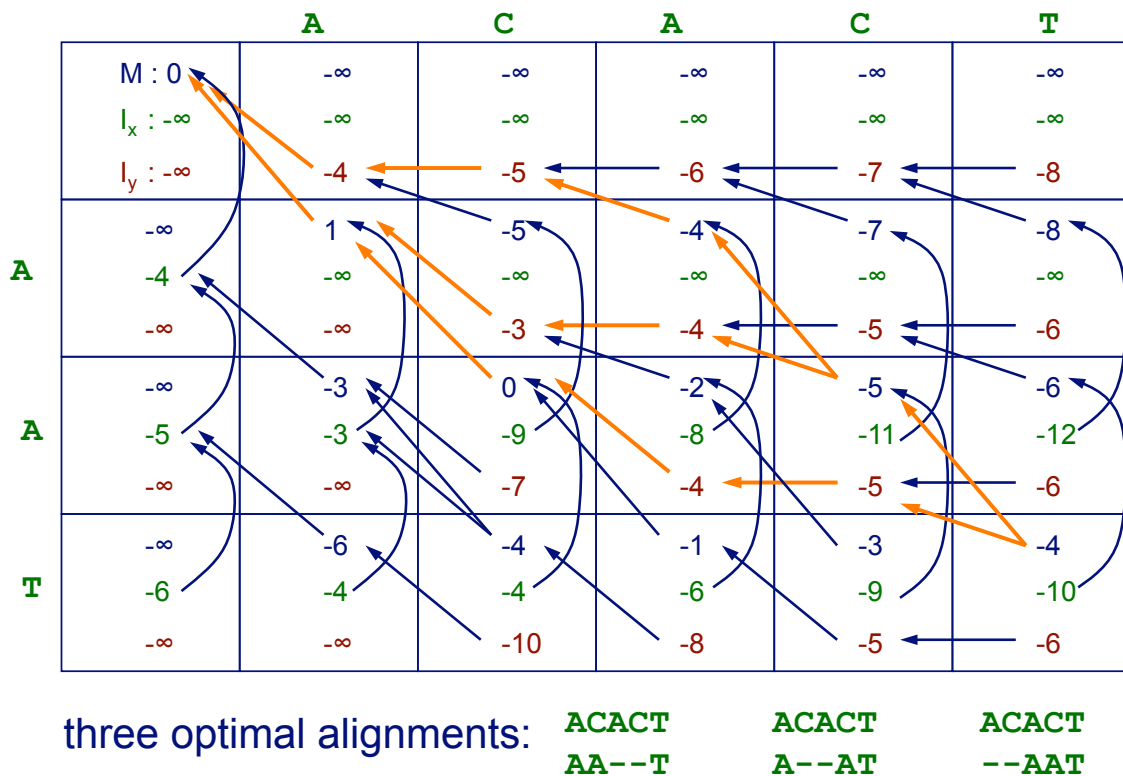
- note that pointers may traverse all three matrices

## Global alignment example (affine gap penalty)

$d = 4, e = 1$



## Global alignment example (continued)

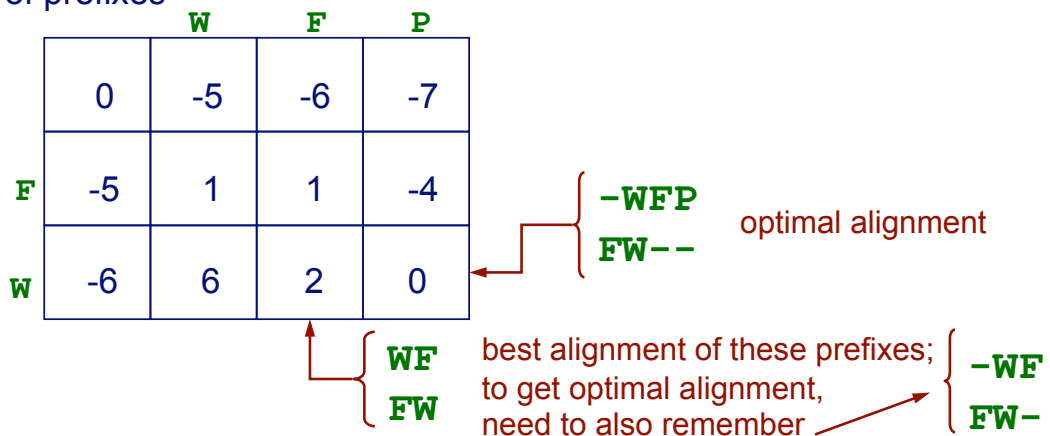


## Why three matrices are needed

- consider aligning the sequences **WFP** and **FW** using  $d = 5$ ,  $e = 1$  and the following values from the BLOSUM-62 substitution matrix:

$$\begin{aligned} s(\mathbf{F}, \mathbf{W}) &= 1 & s(\mathbf{W}, \mathbf{W}) &= 11 \\ s(\mathbf{F}, \mathbf{F}) &= 6 & s(\mathbf{W}, \mathbf{P}) &= -4 \\ s(\mathbf{F}, \mathbf{P}) &= -4 \end{aligned}$$

- the matrix shows the highest-scoring partial alignment for each pair of prefixes





## Local alignment DP for the affine gap penalty case

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

$$I_x(i, j) = \max \begin{cases} M(i-1, j) - d \\ I_x(i-1, j) - e \end{cases}$$

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

## Local alignment DP for the affine gap penalty case

- initialization

$$M(0,0) = 0$$

$$M(i,0) = 0$$

$$M(0, j) = 0$$

cells in top row and leftmost column of  $I_x, I_y = -\infty$

- traceback

- start at largest  $M(i, j)$

- stop at  $M(i, j) = 0$

## Gap penalty functions

- linear:  $w(g) = -g \times d$
- affine: 
$$w(g) = \begin{cases} -d - (g-1)e, & g \geq 1 \\ 0, & g = 0 \end{cases}$$
- convex: as gap length increases, magnitude of penalty for each additional character decreases

e.g.  $w(g) = -d - \log(g) \times e$

## Computational complexity and gap penalty functions

linear:	$O(n^2)$
affine:	$O(n^2)$
convex:	$O(n^2 \log n)$
general:	$O(n^3)$

\* assuming two sequences of length  $n$

# Alignment (global) with general gap penalty function

why the general case has time complexity  $O(n^3)$

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

$k$  ranges over previous coordinates

consider every previous element in the row

consider every previous element in the column

## Pairwise alignment summary

- the number of possible alignments is exponential in the length of sequences being aligned
- dynamic programming can find optimal-scoring alignments in polynomial time
- the specifics of the DP depend on
  - local vs. global alignment
  - gap penalty function
- affine penalty functions are most commonly used