Lecture 2
Sequence Alignment

Burr Settles
IBS Summer Research Program 2008
bsettles@cs.wisc.edu
www.cs.wisc.edu/~bsettles/ibs08/
Sequence Alignment: Task Definition

• given:
  – a pair of sequences (DNA or protein)
  – a method for scoring a candidate alignment

• do:
  – determine the correspondences between substrings in the sequences such that the similarity score is maximized
Why Do Alignment?

- *homology*: similarity due to descent from a common ancestor
- often we can infer homology from similarity
- thus we can sometimes infer structure/function from sequence similarity
Homology Example: Evolution of the Globins
Homology

• homologous sequences can be divided into two groups
  – *orthologous sequences*: sequences that differ because they are found in different species (e.g. human $\alpha$-globin and mouse $\alpha$-globin)
  – *paralogous sequences*: sequences that differ because of a gene duplication event (e.g. human $\alpha$-globin and human $\beta$-globin, various versions of both)
Issues in Sequence Alignment

• the sequences we’re comparing probably differ in length
• there may be only a relatively small region in the sequences that match
• we want to allow partial matches (i.e. some amino acid pairs are more substitutable than others)
• variable length regions may have been inserted/deleted from the common ancestral sequence
Sequence Variations

- sequences may have diverged from a common ancestor through various types of mutations:
  - substitutions (ACGA → AGGA)
  - insertions (ACGA → ACCGGAGA)
  - deletions (ACGGAGA → AGA)
- the latter two will result in gaps in alignments
Insertions, Deletions and Protein Structure

- Why is it that two “similar” sequences may have large insertions/deletions?
  - some insertions and deletions may not significantly affect the structure of a protein

*loop structures*: insertions/deletions here not so significant
Example Alignment: Globins

- figure at right shows prototypical structure of globins

- figure below shows part of alignment for 8 globins (’s indicate gaps)
Three Key Questions

• Q1: what do we want to align?

• Q2: how do we “score” an alignment?

• Q3: how do we find the “best” alignment?
Q1: What Do We Want to Align?

- **global alignment**: find best match of both sequences in their entirety
- **local alignment**: find best subsequence match
- **semi-global alignment**: find best match without penalizing gaps on the ends of the alignment
The Space of Global Alignments

• some possible global alignments for \textbf{ELV} and \textbf{VIS}

\begin{align*}
\text{ELV} & \quad \text{-ELV} & \quad \text{--ELV} & \quad \text{ELV-} \\
\text{VIS} & \quad \text{VIS-} & \quad \text{VIS--} & \quad \text{-VIS} \\
\text{E-LV} & \quad \text{ELV--} & \quad \text{EL-V} \\
\text{VIS-} & \quad \text{--VIS} & \quad \text{-VIS}
\end{align*}
Q2: How Do We Score Alignments?

- gap penalty function
  - \( w(k) \) indicates cost of a gap of length \( k \)

- substitution matrix
  - \( s(a, b) \) indicates score of aligning character \( a \) with character \( b \)
Linear Gap Penalty Function

- different gap penalty functions require somewhat different dynamic programming algorithms
- the simplest case is when a linear gap function is used

\[ w(k) = g \times k \]

where \( g \) is a constant
- we’ll start by considering this case
Scoring an Alignment

• the score of an alignment is the sum of the scores for pairs of aligned characters plus the scores for gaps
• example: given the following alignment

\[
\begin{align*}
\text{VAHV} & \quad \text{---D} \quad \text{---DMPNALSALS} \quad \text{DLHAHKL} \\
\text{AIQLQVTGVVVTDATLKNLGSVHSKG}
\end{align*}
\]

• we would score it by

\[
s(V,A) + s(A,I) + s(H,Q) + s(V,L) + 3g + s(D,G) + 2g \ldots
\]
Q3: How Do We Find the Best Alignment?

- simple approach: compute & score all possible alignments
- but there are

\[
\binom{2n}{n} = \frac{(2n)!}{(n!)^2} \approx \frac{2^{2n}}{\sqrt{\pi n}}
\]

possible global alignments for 2 sequences of length \( n \)
- e.g. two sequences of length 100 have \( \approx 10^{77} \) possible alignments
Pairwise Alignment Via Dynamic Programming

- **dynamic programming**: solve an instance of a problem by taking advantage of solutions for subparts of the problem
  - reduce problem of best alignment of two sequences to best alignment of all prefixes of the sequences
  - avoid recalculating the scores already considered
    - example: Fibonacci sequence 1, 1, 2, 3, 5, 8, 13, 21, 34…

- first used in alignment by Needleman & Wunsch, *Journal of Molecular Biology*, 1970
Dynamic Programming Idea

- consider last step in computing alignment of **AAAC** with **AGC**
- three possible options; in each we’ll choose a different pairing for end of alignment, and add this to best alignment of previous characters

```
AAA  C
AG   C
AAA  C
AGC  -

AAA  C
AG   C
AAAC -
AG   C
```

consider best alignment of these prefixes + score of aligning this pair
Dynamic Programming Idea

- given an $n$-character sequence $x$, and an $m$-character sequence $y$
- construct an $(n+1) \times (m+1)$ matrix $F$
- $F(i, j) =$ score of the best alignment of $x[1…i]$ with $y[1…j]$

```
A   G   C
A
A
A
A
C
```

score of best alignment of AAA to AG
Needleman-Wunch Algorithm

- one way to specify the DP is in terms of its recurrence relation:

\[
F(i, j) = \max \begin{cases} 
F(i-1, j-1) + s(x_i, y_j) \\
F(i-1, j) + g \\
F(i, j-1) + g 
\end{cases}
\]
DP Algorithm Sketch: Global Alignment

- initialize first row and column of matrix
- fill in rest of matrix from top to bottom, left to right
- for each $F(i,j)$, save pointer(s) to cell(s) that resulted in best score
- $F(m,n)$ holds the optimal alignment score; trace pointers back from $F(m,n)$ to $F(0,0)$ to recover alignment
### Initializing Matrix

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>2g</td>
<td>3g</td>
</tr>
<tr>
<td>2g</td>
<td>2g</td>
<td>2g</td>
<td>2g</td>
</tr>
<tr>
<td>3g</td>
<td>3g</td>
<td>3g</td>
<td>3g</td>
</tr>
<tr>
<td>4g</td>
<td>4g</td>
<td>4g</td>
<td>4g</td>
</tr>
</tbody>
</table>
Global Alignment Example

- suppose we choose the following scoring scheme:
  \[ s(x_i, y_i) = \]
  \[ +1 \quad \text{when} \quad x_i = y_i \]
  \[ -1 \quad \text{when} \quad x_i \neq y_i \]
  \[ g \text{ (penalty for aligning with a gap) } = -2 \]
### Global Alignment Example

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The scoring function is defined as:

\[
s(x_i, y_i) = \begin{cases} 
+1 & \text{when } x_i = y_i \\
-1 & \text{when } x_i \neq y_i 
\end{cases}
\]

\[
g = -2
\]
### Global Alignment Example

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-2</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>A</td>
<td>-4</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>A</td>
<td>-6</td>
<td>-1</td>
<td>-2</td>
</tr>
<tr>
<td>C</td>
<td>-8</td>
<td>-5</td>
<td>-1</td>
</tr>
</tbody>
</table>

One optimal alignment:

- x: A A A A C C
- y: A G C C

**x:** A A A A C C

**y:** A G C C
Equally Optimal Alignments

- many optimal alignments may exist for a given pair of sequences
- can use preference ordering over paths when doing traceback

- highroad and lowroad alignments show the two most different optimal alignments
## Highroad & Lowroad Alignments

### Table

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
</tr>
<tr>
<td>A</td>
<td>-2</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>A</td>
<td>-4</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>-6</td>
<td>-3</td>
<td>-2</td>
</tr>
<tr>
<td>C</td>
<td>-8</td>
<td>-5</td>
<td>-4</td>
</tr>
</tbody>
</table>

### Diagram

- **Highroad Alignment**
  - x: A A A A C C
  - y: A G - C

- **Lowroad Alignment**
  - x: A A A A C C
  - y: - A G C
DP Comments

- works for either DNA or protein sequences, although the substitution matrices used differ
- finds an optimal alignment
- the exact algorithm (and computational complexity) depends on gap penalty function (we’ll come back to this)
Local Alignment

• so far we have discussed *global alignment*, where we are looking for best match between sequences from one end to the other

• more commonly, we will want a *local alignment*, the best match between subsequences of $x$ and $y$
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


- interpretation of array values is somewhat different
  - $F(i, j) =$ score of the best alignment of a suffix of $x[1…i]$ and a suffix of $y[1…j]$
Local Alignment DP Algorithm

- the recurrence relation is slightly different than for global algorithm

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

\[
s(x_i, y_i) = \\
+1 \text{ when } x_i = y_i \\
-1 \text{ when } x_i \neq y_i
\]

\[g = -2\]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A</th>
<th>G</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Local Alignment Example

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A</th>
<th>G</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

x: A A G
y: A A G
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 an affine gap penalty function, which involves two terms:
  - a penalty $h$ associated with opening a gap
  - a smaller penalty $g$ for extending the gap
Gap Penalty Functions

- **linear**

  \[ w(k) = gk \]

- **affine**

  \[ w(k) = \begin{cases} 
  h + gk, & k \geq 1 \\
  0, & k = 0 
\end{cases} \]
Dynamic Programming for the Affine Gap Penalty Case

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

\begin{align*}
M(i, j) & \quad \text{best score given that } x[i] \text{ is aligned to } y[j] \\
I_x(i, j) & \quad \text{best score given that } x[i] \text{ is aligned to a gap} \\
I_y(i, j) & \quad \text{best score given that } y[j] \text{ is aligned to a gap}
\end{align*}
Global Alignment DP for the
Affine Gap Penalty Case

\[ M(i, j) = \max \begin{cases} 
M(i - 1, j - 1) + s(x_i, y_j) & \text{match } x_i \text{ with } y_j \\
I_x(i - 1, j - 1) + s(x_i, y_j) & \text{insertion in } x \\
I_y(i - 1, j - 1) + s(x_i, y_j) & \text{insertion in } y 
\end{cases} \]

\[ I_x(i, j) = \max \begin{cases} 
M(i - 1, j) + h + g & \text{open gap in } x \\
I_x(i - 1, j) + g & \text{extend gap in } x 
\end{cases} \]

\[ I_y(i, j) = \max \begin{cases} 
M(i, j - 1) + h + g & \text{open gap in } y \\
I_y(i, j - 1) + g & \text{extend gap in } y 
\end{cases} \]
Global Alignment DP for the Affine Gap Penalty Case

- initialization
  \[ M(0,0) = 0 \]
  \[ I_x(i,0) = h + g \times i \]
  \[ I_y(0,j) = h + g \times j \]
  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 any of \( M(0,0), I_x(0,0), I_y(0,0) \)
  - note that pointers may traverse all three matrices
## Global Alignment Example

(Affine Gap Penalty)

\[ h = -3, \ g = -1 \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>C</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M : 0</strong></td>
<td>-\infty</td>
<td>-\infty</td>
<td>-\infty</td>
<td>-\infty</td>
<td>-\infty</td>
</tr>
<tr>
<td><strong>I_x : -3</strong></td>
<td>-\infty</td>
<td>-\infty</td>
<td>-\infty</td>
<td>-\infty</td>
<td>-\infty</td>
</tr>
<tr>
<td><strong>I_y : -3</strong></td>
<td>-4</td>
<td>-5</td>
<td>-6</td>
<td>-7</td>
<td>-8</td>
</tr>
<tr>
<td>A</td>
<td>-\infty</td>
<td>1</td>
<td>-5</td>
<td>-4</td>
<td>-7</td>
</tr>
<tr>
<td></td>
<td>-4</td>
<td>-\infty</td>
<td>-\infty</td>
<td>-\infty</td>
<td>-\infty</td>
</tr>
<tr>
<td></td>
<td>-\infty</td>
<td>-\infty</td>
<td>-3</td>
<td>-4</td>
<td>-5</td>
</tr>
<tr>
<td>A</td>
<td>-\infty</td>
<td>-3</td>
<td>0</td>
<td>-2</td>
<td>-5</td>
</tr>
<tr>
<td></td>
<td>-5</td>
<td>-3</td>
<td>-9</td>
<td>-8</td>
<td>-11</td>
</tr>
<tr>
<td></td>
<td>-\infty</td>
<td>-\infty</td>
<td>-7</td>
<td>-4</td>
<td>-5</td>
</tr>
<tr>
<td>T</td>
<td>-\infty</td>
<td>-6</td>
<td>-4</td>
<td>-1</td>
<td>-3</td>
</tr>
<tr>
<td></td>
<td>-6</td>
<td>-4</td>
<td>-4</td>
<td>-6</td>
<td>-9</td>
</tr>
<tr>
<td></td>
<td>-\infty</td>
<td>-\infty</td>
<td>-10</td>
<td>-8</td>
<td>-5</td>
</tr>
</tbody>
</table>
Global Alignment Example (Continued)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>C</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>M : 0</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
<tr>
<td>I_x : -3</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
<tr>
<td>I_y : -3</td>
<td>-4</td>
<td>-5</td>
<td>-6</td>
<td>-7</td>
<td>-8</td>
</tr>
<tr>
<td>A</td>
<td>-∞</td>
<td>1</td>
<td>-5</td>
<td>-4</td>
<td>-7</td>
</tr>
<tr>
<td></td>
<td>-4</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
<tr>
<td></td>
<td>-∞</td>
<td>-∞</td>
<td>-3</td>
<td>-4</td>
<td>-5</td>
</tr>
<tr>
<td>A</td>
<td>-∞</td>
<td>-3</td>
<td>0</td>
<td>-2</td>
<td>-5</td>
</tr>
<tr>
<td></td>
<td>-5</td>
<td>-3</td>
<td>-9</td>
<td>-8</td>
<td>-11</td>
</tr>
<tr>
<td></td>
<td>-∞</td>
<td>-∞</td>
<td>-7</td>
<td>-4</td>
<td>-5</td>
</tr>
<tr>
<td>T</td>
<td>-∞</td>
<td>-6</td>
<td>-4</td>
<td>-1</td>
<td>-3</td>
</tr>
<tr>
<td></td>
<td>-6</td>
<td>-4</td>
<td>-10</td>
<td>-8</td>
<td>-5</td>
</tr>
<tr>
<td></td>
<td>-∞</td>
<td>-∞</td>
<td>-10</td>
<td>-5</td>
<td>-6</td>
</tr>
</tbody>
</table>

three optimal alignments:  
ACACT
AA--T
ACACT
A--AT
ACACT
--AAT
Local Alignment DP for the Affine Gap Penalty Case

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

\[
I_x(i, j) = \max \begin{cases} 
M(i - 1, j) + h + g \\
I_x(i - 1, j) + g
\end{cases}
\]

\[
I_y(i, j) = \max \begin{cases} 
M(i, j - 1) + h + g \\
I_y(i, j - 1) + g
\end{cases}
\]
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:** \( w(k) = gk \)

- **affine:**
  \[
  w(k) = \begin{cases} 
  h + gk, & k \geq 1 \\
  0, & k = 0 
  \end{cases}
  \]

- **concave:** a function for which the following holds for all \( k, l, m \geq 0 \)
  \[
  w(k + m + l) - w(k + m) \leq w(k + l) - w(k)
  \]
  e.g. \( w(k) = h + g \times \log(k) \)
Concave Gap Penalty Functions

\[ w(k + m + l) - w(k + m) \leq w(k + l) - w(k) \]
More On Scoring Matches

• so far, we’ve discussed multiple gap penalty functions, but only one match-scoring scheme:

\[ s(x_i, y_i) = \]

\[ +1 \text{ when } x_i = y_i \]

\[ -1 \text{ when } x_i \neq y_i \]

• for protein sequence alignment, some amino acids have similar structures and can be substituted in nature:

aspartic acid (D)  glutamic acid (E)
Substitution Matrices

• two popular sets of matrices for protein sequences
  – PAM matrices [Dayhoff et al., 1978]
  – BLOSUM matrices [Henikoff & Henikoff, 1992]

• both try to capture the relative substitutability of amino acid pairs in the context of evolution
BLOSUM62 Matrix

|   | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | X |
| A |   | 4 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| R | -1|   | 5 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| N | -2| 0 | 6 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| D | -2| -2| 1 | 6 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| C | 0 | -3| -3| -3| 9 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Q | -1| 1 | 0 | 0 | -3| 5 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| E | -1| 0 | 0 | 2 | -4| 2 | 5 |   |   |   |   |   |   |   |   |   |   |   |   |   |
| G | 0 | -2| 0 | -1| -3| -2| -2| 6 |   |   |   |   |   |   |   |   |   |   |   |   |
| H | -2| 0 | 1 | -1| -3| 0 | 0 | -2| 8 |   |   |   |   |   |   |   |   |   |
| I | -1| -3| -3| -3| -1| -3| -3| -3| -4| -3 |   |   |   |   |   |   |   |
| L | -1| -2| -3| -4| -1| -2| -3| -4| -3 | 2 | 4 |   |   |   |   |
| K | -1| 2 | 0 | -1| -3| 1 | 1 | -2| -1| -3 | -2| 5 |   |   |   |
| M | -1| -1| -2| -3| -1| 0 | -2| -3| -2 | 1 | 2 | 1 | 5 |   |
| F | -2| -3| -3| -2| -3| -3| -3| -1| 0 | 0 | 0 | -3| 0 | 6 |   |
| P | -1| -2| -2| -1| -3| -1| -1| -2| -2 | -3| -3| -1| -2| -4| 7 |   |
| S | 1 | -1| 1 | 0 | -1| 0 | 0 | -1| -2| -2 | 0 | -1| -2| -1| 4 |   |
| T | 0 | -1| 0 | -1| -1| -1| -1| -2| -2 | -1| -1| -1| -2| 1 | 1 | 5 |   |
| W | -3| -3| -4| -4| -2| -2| -3| -2| -3 | -2| -3| -1| 1 | 4 | -3| -2 | 11 |
| Y | -2| -2| -2| -3| -2| -1| -2| -3| 2 | -1| -1| -2| 1 | 3 | -3| -2| -2 | 2 | 7 |   |
| V | 0 | -3| -3| -3| -1| -2| -2| -3| 3 | 1 | -2| 1 | -1| -2| -2 | 0 | -3 | 1 | 4 |   |
| X | 0 | -1| -1| -1| -1| -1| -1| -1| -1 | 0 | 0 | 0 | 2 | -1| -1| -1 |   |   |   |   |

Positive for chemically similar substitution

Common amino acids have low weights

Rare amino acids have high weights
Heuristic Methods

• the algorithms we learned today take $O(nm)$ time to align sequences, which is too slow for searching large databases
  – imagine an internet search engine, but where queries and results are protein sequences

• heuristic methods do fast approximation to dynamic programming
  – example: BLAST [Altschul et al., 1990; Altschul et al., 1997]
  – break sequence into small (e.g. 3 base pair) “words”
  – scan database for word matches
  – extend all matches to seek high-scoring alignments
  – tradeoff: sensitivity for speed
Multiple Sequence Alignment

- we’ve only discussed aligning 2 sequences, but we may want to do more
- discover common motifs in a set of sequences (e.g. DNA sequences that bind the same protein)
- characterize a set of sequences (e.g. a protein family)
- much more complex

Figure from A. Krogh, An Introduction to Hidden Markov Models for Biological Sequences
Next Time…

• basic molecular biology
• sequence alignment
• **probabilistic sequence models**
• gene expression analysis
• protein structure prediction
  – by Ameet Soni