Sequence alignments – Dynamic programming algorithms

Lecturer: Marina Alexandersson
2 September, 2005

Sequence comparisons

Sequence comparisons are used to detect evolutionary relationships between organisms, proteins or gene sequences. Sequence comparisons can also be used to discover the function of a novel gene or the structure of an unknown protein, by comparing it to an already characterized gene or protein, since sequences that are very similar often have a similar structure and/or function. If two sequences from different organisms are evolutionary related, it means that they have a common ancestor. The sequences are then said to be homologous.

Through evolution the DNA sequence of an organism accumulate mutations. Most of them make no difference or are harmful to the organism, and therefore selected against. But some of them are kept and will eventually result in a new organism.

By comparing Sequence 1 and Sequence 2, or aligning them, we may infer the evolutionary process, starting from the same ancestor sequence and then changing through mutations. A mutation is an insertion, a deletion, or a substitution.

Example 1.

Begin with ancestor sequence ACTGC and assume the following evolution

seq1: ACTGC
     ↓ substitution
     ACCGC
     ↓ deletion
     A-CGC
     ↓ insertion
     seq2: ACGTC
The true relationship, or the true alignment, between seq1 and seq2 is

\[
\text{ACTG–C} \\
\text{A–CGTC}
\]

**Dot plots**

One of the earliest methods of comparing two protein or nucleotide sequences was to create a dot plot.

**Example 2.**

Assume we want to compare the sequences

\[
\text{seq1: AGCTAGGA} \\
\text{seq2: CACTAGGC}
\]

Insert a dot in each matching cell, and then scan the resulting graphs for series of dots that form a diagonal. To maximize the number of matches the resulting alignment could then be

\[
\text{--AGCTAGGA--} \\
\text{CA-CTAGG-C}
\]
The dot matrix can reveal the presence of insertions and deletions because they shift the diagonal horizontally or vertically.

**Example 3.**
Assume we want to compare the sequences

```
seq1: AGCTAGCGA
seq2: AGCATAGGA
```

The resulting alignment could be

```
AGC–TAGCGA
AGCATAG–GA
```

Note how the alignment is represented by a path through the dot matrix, starting from the top left corner, moving to the bottom right corner, using the following steps:

- Representing a match or a substitution
- Representing either a deletion in seq1 or an insertion in seq2
- Representing either an insertion in seq1 or a deletion in seq2
Long sequences can also be compared using smaller plots. By plotting a sequence against itself it is possible to find internal repeats, reveal tandem genes, repeated domains in proteins, or regions of low complexity where the same character is repeated.

![Plot of 2-\(\alpha\) haptoglobin against itself.](image1)

![Plot of human \(\beta\)-globin vs. human myoglobin.](image2)
Sequence alignments

Although dot plots can be used to detect sequence similarity, it cannot readily resolve similarity that is interrupted by regions of low similarity or insertions or deletions. Moreover, there are often several possible alignments to choose between. Therefore one would like to device a method that can assign a measure, or a score, to each possible path in a dot matrix. The path with the highest score represents the best possible alignment, called an optimal alignment, between the two sequences.

What is an alignment?

Two sequences are placed in lines on top of each other and spaces are inserted in various numbers and places to maximize the number of identical characters in the same column.

The (true) alignment indicates the evolutionary process giving rise to the different sequences starting from the same ancestor sequence and then changing through mutations (insertions, deletions and substitutions).
Example 1. (revisited)
Begin with sequence \textbf{AGCACACA} and assume the following evolution

\begin{align*}
\text{seq1:} & \quad \text{ACTGC} \\
& \quad \downarrow \text{substitution} \\
& \quad \text{ACCGC} \\
& \quad \downarrow \text{deletion} \\
& \quad \text{A-CGC} \\
& \quad \downarrow \text{insertion} \\
\text{seq2:} & \quad \text{ACGT C}
\end{align*}

The true relationship between \text{seq1} and \text{seq2} is

\begin{align*}
\text{ACTG-} & \text{C} \\
\text{A-} & \text{CGTC}
\end{align*}

but the \textit{optimal} alignment would most likely be

\begin{align*}
\text{ACTG-} & \text{C} \\
\text{A-} & \text{GTC}
\end{align*}

What is meant by an ‘optimal’ alignment?

1. Score all possible alignments according to some model.
2. Pick the alignment with the highest score.

\textbf{Align operators}

- \textbf{Gap}: represents insertions or deletions.

One usually thinks of a gap in one of the two following ways:

(i) as an insertion or deletion in the first sequence with respect to the second. An insertion results in a gap in seq2, a deletion results in a gap in seq1.

(ii) strictly as insertions in either of the sequences. An insertion in seq1 results in a gap in the second, an insertion in seq2 results in a gap in the first.

In reality insertions and deletions occur in both sequences, this is just a convention when modeling gaps and doesn’t affect the results of choosing an optimal alignment.

- \textbf{Match}: identical characters are aligned (amino acids or nucleotides) representing conserved residuals.

- \textbf{Mismatch}: two different characters are aligned, representing a substitution.
Example 4.

Scoring schemes
In order to determine which alignment is optimal we need a scoring system.

Example 5.
Align protein sequences seq1: HEAGWGHEE and seq2: PAWHEAE. Which alignment is optimal?

I. HEAGWGHEE
   PAWHEAE---

II. HEAGWGHEE
    ---PAWHEAE

III. HEAGWGHEE-E
     ---PAW-HEAE

Is it I or II, since they only have 3 gaps, or is it III, since it has the most matches? Well, it depends on the scoring scheme.

Total score of an alignment = sum of all pairwise aligned symbols + gap penalties

- Scoring models for match/mismatch: PAM, BLOSUM etc.
- Gap penalties: linear of affine.

Assumption: each aligned pair of symbols is independent of the others.
Scoring matrices

A scoring matrix gives the score for aligning two amino acids (match or mismatch) in a pairwise alignment. A scoring matrix can be considered a measure of the evolutionary change. The most widely used matrices are PAMs and BLOSUMs. Both calculate substitution frequencies between amino acids, and both are derived from known protein alignments.

**PAM** is the unit of Percent Accepted (Point) Mutations (Dayhoff et al. 1978). A PAM1 reflects an amount of evolution (an evolutionary time) producing on average one mutation per 100 amino acids.

*Note how evolutionary time differs from real time, since the “speed” of the evolution differs between different species. For instance, it takes a much longer time for the human species to accumulate on average one mutation per 100 amino acids, than it does for fruit flies."

Higher level matrices, representing a larger evolutionary distance, are extrapolated from PAM1. In practice the user selects a suitable evolutionary distance for comparing sequences and then uses the corresponding PAM matrix. PAM40 is good for sequences with small evolutionary distance, especially for short, strong local similarities. PAM250 is good for long sequences with large evolutionary distance. PAM20 is the distance thought to be most suitable for comparing human and mouse.

### PAM20

|     | A     | R     | N     | D     | C     | Q     | E     | G     | H     | I     | L     | K     | M     | F     | P     | S     | T     | W     | Y     | Y     |
|-----|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| A   | 6     | -8    | -6    | -4    | -8    | -5    | -3    | -3    | -8    | -6    | -7    | -8    | -6    | -9    | -2    | -1    | -1    | -1    | -6    | -9    | -3    |
| R   | -8    | 9     | -7    | -12   | -9    | -2    | -11   | -11   | -3    | -6    | -10   | -1    | -5    | -10   | -5    | -4    | -8    | -3    | -11   | -9    |
| N   | -5    | -7    | 8     | 1     | -13   | -5    | -3    | -4    | -1    | -5    | -4    | -2    | -11   | -10   | -7    | -1    | -3    | -9    | -5    | -9    |
| D   | -4    | -12   | 1     | 8     | -16   | -4    | 2     | -4    | -5    | -9    | -15   | -6    | -13   | -17   | -9    | -5    | -6    | -17   | -13   | -9    |
| C   | -8    | -9    | -13   | -16   | 10    | -16   | -16   | -11   | -8    | -7    | -17   | -16   | -16   | -15   | -9    | -4    | -9    | -18   | -5    | -7    |
| Q   | -5    | -2    | -5    | -4    | -16   | 9     | 0     | -8    | 0     | -9    | -6    | -4    | -5    | -15   | -4    | -6    | -7    | -15   | -14   | -8    |
| E   | -3    | -11   | -3    | 2     | -16   | 0     | 8     | -5    | -6    | -6    | -10   | -5    | -8    | -15   | -7    | -5    | -7    | -19   | -9    | -8    |
| G   | -3    | -11   | -4    | -4    | -11   | -8    | -5    | 7     | -10   | -13   | -12   | -8    | -10   | -7    | -3    | -7    | -17   | -16   | -7    |
| H   | -8    | -3    | -1    | -5    | -6    | 8     | 0     | -6    | 10    | 9     | -11   | -7    | -8    | -13   | -7    | -5    | -7    | -8    | -4    | -7    |
| I   | -6    | -6    | -6    | -9    | -7    | -9    | -6    | -13   | -11   | 9     | -2    | -7    | -2    | -3    | -10   | -3    | -16   | -7    | -1    |
| L   | -7    | -10   | -8    | -16   | -17   | -8    | -10   | -12   | -7    | -2    | -7    | -9    | 0     | -4    | -3    | -9    | -8    | -7    | -8    | -3    |
| K   | -8    | -1    | -2    | -6    | -16   | -4    | -5    | -8    | -8    | -7    | -9    | 7     | -3    | -16   | -8    | -5    | -4    | -14   | -10   | -10   |
| M   | -6    | -5    | -11   | -13   | -16   | -5    | -8    | -10   | -13   | -2    | 0     | -3    | 11    | -5    | -5    | -6    | -5    | -15   | -13   | -2    |
| F   | -9    | -10   | -10   | -17   | -15   | -15   | -16   | -10   | -7    | -3    | -4    | -16   | 5     | 9     | -11   | -7    | -10   | -6    | -1    |
| P   | -2    | -5    | -7    | -9    | -9    | -4    | -7    | -7    | -5    | -10   | -8    | -8    | -9    | 11    | 8     | -3    | -5    | -16   | -16   | -7    |
| S   | -1    | -4    | -1    | -5    | -4    | -6    | -5    | -3    | -7    | -8    | -9    | -5    | -6    | -7    | 3     | 7     | 0     | -6    | -8    | -8    |
| T   | -1    | -8    | -3    | -6    | -9    | -7    | -7    | -8    | -3    | -8    | -4    | -5    | -10   | -5    | 0     | 7     | -15   | -7    | -4    |
| W   | -16   | -3    | -9    | -17   | -18   | -15   | -19   | -17   | -8    | -16   | -7    | -14   | -15   | -6    | -16   | -5    | -15   | -13   | -6    |
| Y   | -9    | -11   | -5    | -13   | -5    | -14   | -9    | -16   | -4    | -7    | -8    | -10   | -13   | 1     | -15   | -8    | -7    | -10   | -8    |
| V   | -3    | -9    | -9    | -9    | -7    | -8    | -8    | -7    | -7    | 1     | -3    | -10   | -2    | -9    | -7    | -8    | -4    | -18   | -8    | -7    |
The **BLOSUM** (BLOcks SUbsitution Matrix) scoring matrix (Henikoff & Henikoff, 1992) is similar to PAM. BLOSUM50 and BLOSUM62 are the ones most widely used. BLOSUM50 is the default for the FASTA sequence analysis program. The BLOSUM62 is most effective detecting known members of a protein family from a database when searching with the BLAST local alignment program.

PAMs are based on an explicit evolutionary model and represent a specific evolutionary distance. They are derived from sequences with 85% similarity or higher. BLOSUMs are based on empirical frequencies from relatively distant proteins. It is hard to say which one is the best, it depends on the situation.

**BLOSUM62**

```
<table>
<thead>
<tr>
<th>C</th>
<th>-1</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>-1</td>
<td>5</td>
</tr>
<tr>
<td>T</td>
<td>-3</td>
<td>-1</td>
</tr>
<tr>
<td>P</td>
<td>-3</td>
<td>-2</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>G</td>
<td>-3</td>
<td>0</td>
</tr>
<tr>
<td>N</td>
<td>-3</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>-3</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>-4</td>
<td>0</td>
</tr>
<tr>
<td>Q</td>
<td>-3</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>-3</td>
<td>2</td>
</tr>
<tr>
<td>R</td>
<td>-3</td>
<td>1</td>
</tr>
<tr>
<td>K</td>
<td>-3</td>
<td>0</td>
</tr>
<tr>
<td>M</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>I</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>L</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>V</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>Y</td>
<td>-2</td>
<td>2</td>
</tr>
<tr>
<td>W</td>
<td>-2</td>
<td>3</td>
</tr>
</tbody>
</table>
```

Note that the upper half of the matrix is identical to the lower half and can therefore be omitted (check in the PAM20 matrix above, e.g. AC = CA).
**Gap penalties**

Gap penalties are often composed of two parts: a gap opening penalty $d$ and a gap extension penalty $e$, (both $d$, $e < 0$). The gap extension penalty is usually much smaller ($e < d$) to reflect that for instance 10 insertions of one nucleotide each should be harder than one insertion of 10 nucleotides.

![Diagram showing gap penalties](chart)

Standard gap penalties for a gap of length $g$:

- **Linear gap penalties:**
  \[
  
  \gamma(g) = -gd
  
  \]
  i.e. each gap residue is independent with gap score $-d$.

- **Affine:**
  \[
  \gamma(g) = -d - e(g - 1)
  
  \]
  i.e. opening a gap gets score $-d$ and then every succeeding extending gap gets score $-e$.

Note that the linear gap is just a special case of the affine model where $d = e$.

**Example 5. (revisited)**

Using a linear gap penalty of $d = -8$ and the PAM20 scoring matrix we get the following scores for the alignments:

1. **HEAGAWGHEE**
   
   **PAWHEAE---**
   
   $= (3*(-8)) - 5 - 3 - 16 - 10 - 3 - 16 - 5 = -82$

2. **HEAGAWGHEE**
   
   **---PAWHEAE**
   
   $= (3*(-8)) - 7 - 6 + 13 - 10 - 6 - 3 + 8 = -35$

3. **HEAGAWGHEE**
   
   **---PAW-HEAE**
   
   $= (5*(-8)) - 7 + 6 + 13 + 9 + 8 + 8 = -3$

Thus, using this model, alignment III is optimal.
**Total number of alignments**

Thus, we can determine the optimal alignment by calculating the score of each possible alignment. But usually the total number of alignments is too high to perform this calculation by hand.

**Example 6.**

Align seq1: **ATAAGC** and seq2: **AAAACG**.

Let $L$ be the alignment length.

\[
\begin{array}{c}
\text{ATAAGC}\
\text{AAAA-CG}
\end{array}
\]

$L = 7$

We know that $L$ is at least 6 and at most 12.

\[
\begin{array}{c}
\text{ATAAGC}\
\text{AAAAACG}
\end{array}
\]

$L = 6:$

\[
\begin{array}{c}
\text{ATAAGC}\
\text{AAAAACG}
\end{array}
\]

$L = 12:$

\[
\begin{array}{c}
\text{ATAAGC}\
\text{AAAAACG}
\end{array}
\]

But the total number of possible alignments is $924!!$

A more realistic example: two protein sequences of 300 amino acids each have $10^{88}$ possible alignments. To compare, in universe there are approximately $10^{80}$ elementary particles.

(The calculations for this are not part of the scope of this course, but can be found in Waterman’s *Introduction to Computational Biology.*)
Dynamic programming

Dynamic programming (DP) is an efficient recursive method to search through all possible alignments and finding the one with the optimal score. Dynamic programming usually consists of three components.

- Recursive relation
- Tabular computation
- Traceback

Example 7.

An alignment can be represented as a path through a matrix:

\[
\begin{array}{cccccc}
\text{A} & \text{T} & \text{C} & \text{T} & \text{C} & \text{A} \\
\text{T} & & & & & \\
\text{G} & & & & & \\
\text{A} & & & & & \\
\text{C} & & & & & \\
\text{T} & & & & & \\
\text{A} & & & & & \\
\end{array}
\]

To search through the matrix of all possible paths and find the optimal path DP is used.

Needleman-Wunsch (global alignment)

We want to align two sequences \(x_1x_2...x_n\) and \(y_1y_2...y_m\) and create an \(n \times m\) matrix \(F\) where

\[
F(i, j) = \text{score of optimal path of subsequences } x_1...x_i \text{ and } y_1...y_j
\]

Assume a linear gap penalty \(d\).

\[
F(i, j) = \max\left\{ \begin{array}{ll}
F(i-1, j-1) + s(x_i, y_j) & \text{(match/mismatch)} \\
F(i-1, j) - d & \text{(gap in y)} \\
F(i, j-1) - d & \text{(gap in x)}
\end{array} \right.
\]

with \(F(0,0) = 0\), \(F(i, 0) = -id\), \(F(0, j) = -jd\).

This is the recursive relation in the dynamic programming algorithm.
In the tabular computation we start in cell \((0,0)\) and calculate one row at a time. In each cell \((i, j)\) we keep a pointer to the optimal previous position given the current.

![Diagram of tabular computation](image)

The last cell \(F(n,m)\) is the score of the optimal path, and in the traceback we start in cell \((n,m)\) and follow the pointers back to cell \((0,0)\) to achieve the optimal alignment.

**Example 8.**

Assume we want to align the sequences

\[
\text{seq1: ATACGT} \\
\text{seq2: ATCGAT}
\]

using the Needleman-Wunsch algorithm with the model

\[
gap \text{penalty: } d = -2 \\
s(x_i, y_j) = \begin{cases} 
2 & \text{if } x_i = y_j \\
-1 & \text{otherwise.}
\end{cases}
\]

We start by creating the Needleman-Wunsch matrix with \((n+1) \times (m+1) = 7 \times 7\) cells (one extra row and column for the \(F(i,0)\) and \(F(0,j)\) cases), and start to fill it from the top left cell \((0,0)\) to the bottom right cell \((n,m)\) (tabular computation). Remember to make a note for each cell \((i, j)\) which “it came from”, that is, which of the cells \((i-1, j-1)\), \((i-1, j)\), or \((i, j-1)\) gave rise to the maximum for the current cells. The procedure goes

1. \(F(0,0) = 0\)
2. \(F(0, i) = -2i\) (top row) – cell \((0, i)\) always “comes from” \((i-1,0)\).
3. \(F(j,0) = -2j\) (leftmost column) – cell \((j,0)\) always “comes from” \((0, j-1)\).
4. \[ F(1,1) = \max \begin{cases} F(0,0) + s(A, A) = 0 + 2 = 2 \\ F(0,1) - 2 = -4 = 2 \\ F(1,0) - 2 = -4 \end{cases} \]

5. \[ F(1,2) = \max \begin{cases} F(0,1) + s(T, A) = -2 - 1 = -3 \\ F(0,2) - 2 = -6 = 0 \\ F(1,1) - 2 = 0 \end{cases} \]

and so on until all cells are filled and point to a previous cell.

\[
F(i,j): \begin{array}{ccccccc}
\text{A} & \text{T} & \text{A} & \text{C} & \text{G} & \text{T} \\
\hline
0 & -2 & -4 & -6 & -8 & -10 & -12 \\
A & -2 & 2 & 0 & -2 & -4 & -6 & -8 \\
T & -4 & 0 & 4 & 2 & 0 & -2 & -4 \\
C & -6 & -2 & 2 & 3 & 4 & 2 & 0 \\
G & -8 & -4 & 0 & 1 & 2 & 6 & 4 \\
A & -10 & -6 & -2 & 2 & 0 & 4 & 5 \\
T & -12 & -8 & -4 & 0 & 1 & 2 & 6 \\
\end{array}
\]

In the traceback we start in cell \((n,m)\) and follow the arrows back to \((0,0)\) to achieve the optimal alignment. In this case we get
To get the actual alignment, we start from the top left corner again, and start to match the symbols. A diagonal step results in a symbol in both sequences (match or substitution), a horizontal step results in a symbol in seq1 and a gap in seq2, and a vertical step results in a gap in seq1 and a symbol in seq2.

\[
\begin{align*}
\text{ATACG-T} \\
\text{AT-CGAT}
\end{align*}
\]

Note how a cell in the tabular computation may point to one, two or all three possible previous cells, if they give rise to the same cell score. If this occurs in the optimal alignment path, it only means that we have several alternative alignments reaching the same optimal score.

**Smith-Waterman (local alignment)**

Sometimes we want to find a conserved region, e.g. a conserved protein domain, and not align the entire sequences.

\[
\begin{align*}
\text{CTCCCCCCTTCAGGCTGCAC} \\
\text{T-----CTTCAGGC-----A--}
\end{align*}
\]

The local alignment algorithm is very similar to the global; it only needs a slight modification.

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

The tabular computation is performed the same way as above.

The traceback starts in the \((i, j)\) cell with the highest score which is not necessarily (but of course still could be) cell \((n, m)\). Then we follow the pointers back until a cell with score 0 is reached or until we reach cell \((0,0)\).