Pairwise Alignment and Database Searching

Anders Gorm Pedersen
Molecular Evolution Group
Center for Biological Sequence Analysis
Sequences are related

- Darwin: all organisms are related through descent with modification
- => Sequences are related through descent with modification
- => Similar molecules have similar functions in different organisms

Phylogenetic tree based on ribosomal RNA: three domains of life
Phylogenetic tree of globin-type proteins found in humans
Why compare sequences?

- Determination of evolutionary relationships
- Prediction of protein function and structure (database searches).

Protein 1: binds oxygen

Sequence similarity

Protein 2: binds oxygen?
Dotplots: visual sequence comparison

1. Place two sequences along axes of plot

2. Place dot at grid points where two sequences have identical residues

3. Diagonals correspond to conserved regions
### Pairwise alignments

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>alpha</strong></td>
<td><strong>beta</strong></td>
<td><strong>alpha</strong></td>
<td><strong>beta</strong></td>
<td><strong>alpha</strong></td>
<td><strong>beta</strong></td>
</tr>
<tr>
<td>V-LSPADKTNVKAAWGKVGAHAGEYGAELERMFLSFPTTKTYFPHF-DLS-----HGSA</td>
<td>VHLTPEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQRFFESFGDLSSTPDAVMGNP</td>
<td>QVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAHL</td>
<td>KVKAHGKKVLGAFSDLNLKGTATLSELHCDKLHVDPMENFRLLGNVLVCVLAHHF</td>
<td>PAEFTPAVHASLDKFASVSTVLTSKYR</td>
<td>GKEFTPPVQAAYQKVAVANALAHKYH</td>
</tr>
</tbody>
</table>

43.2% identity;  
Global alignment score: 374
Pairwise alignment

100.000% identity in 3 aa overlap

SPA
::: SPA

Percent identity is not a good measure of alignment quality
Pairwise alignments: alignment score

43.2% identity;  

Global alignment score: 374

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>V-LS PADKTNVKA AWGKVGAHAGEYGAEA LERMFLS F PTTKTY FPHF-DLS-----HGSA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>beta</td>
<td>VHL TPEEKSAVTALWGKV--NVDEVGGEALGR LLVVY PWT QRF FESFGDLST PD AVMG N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>alpha</td>
<td>QVKG HGK KVADALTNAVAHVDDDMPNALSALS DLHAHKLRVDPVNFKLLLHCLLVT LA AH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>beta</td>
<td>KVKAHGKKVLGA FS DGLA HLDNLKGT FALTSELHC DL KL HQDVDPENF RLLGNVLCVL AH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>alpha</td>
<td>PAEFTP AVHASLDKFLAS VSTV LTSK YR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>beta</td>
<td>GKEFTP PVQAAYQKV VAGVANALAHKY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>130</td>
<td>140</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

120 130 140
Alignment scores: match vs. mismatch

Simple scoring scheme (too simple in fact...):

Matching amino acids: 5
Mismatch: 0

Scoring example:

\[
\begin{array}{ccccccc}
K & A & W & S & A & D & V \\
: & : & : & : & : & : & :
\end{array}
\]

\[
\begin{array}{ccccccc}
K & D & W & S & A & E & V \\
5 & +0 & +5 & +5 & +5 & +0 & +5
\end{array}
\]

\[5+0+5+5+5+0+5 = 25\]
Pairwise alignments: conservative substitutions

43.2% identity; Global alignment score: 374

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>alpha</td>
<td>V-LSPADKTNVKAAWGKVGAGHEYGAEALERMFLSFPTTKTYFPHF-DLS-----HGSA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>beta</td>
<td>VHLTPEEEKSAVTALWGKV--NVDEVGGEALGRLLVYYPWTQRFFESFGDLSTPDAVMGNP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>alpha</td>
<td>QVKHGKKSADALTNAHVDDMPNALSALSDLHAKLRVDPVNFKLSSHCLLVTLAHL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>beta</td>
<td>KVKAHGGKVLGAFSDGLAHLDNLKGTFLTSELHCDKHLHVDPPENFRLLGNVLCVLAHHF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>130</td>
<td>140</td>
<td></td>
<td></td>
</tr>
<tr>
<td>alpha</td>
<td>PAEFTPAVHASLDKFLASVSTVLSKYR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>beta</td>
<td>GKEFTPPVQAYQKVAGVANALAHKYH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>130</td>
<td>140</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Amino acid properties

Serine (S) and Threonine (T) have similar physicochemical properties
Aspartic acid (D) and Glutamic acid (E) have similar properties

=> Substitution of S/T or E/D occurs relatively often during evolution

=> Substitution of S/T or E/D should result in scores that are only moderately lower than identities
Protein substitution matrices

BLOSUM50 matrix:

- Positive scores on diagonal (identities)
- Similar residues get higher (positive) scores
- Dissimilar residues get smaller (negative) scores

|     | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V |
| A   | 5 | -2| -1| -2| -1| -1| 0 | 0 | -3| -1| -4| -2| -3| -4| -2| -4| -3| -4| -4| -4| -2| 4 |
| R   | -2| 7 | 7 | 2 | 2 | 0 | 0 | 0 | -3| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N   | -1| -1| 7 | 8 | 6 | 4 | 2 | 7 | 1 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| D   | -2| -2| 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C   | -1| -4| -2| -2| -2| -2| -2| -3| -2| -2| -2| -3| -3| -2| -3| -3| -3| -3| -3| -3| -3| -3 |
| Q   | -1| 1 | 0 | 0 | 0 | 0 | -3| -1| -2| -2| -2| -3| -3| -2| -3| -3| -3| -3| -3| -3| -3| -3 |
| E   | -1| 0 | 0 | 2 | -3| 2 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| G   | 0 | -3| 0 | -1| -3| -2| -3| 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| H   | -2| 0 | 1 | -1| -3| 1 | 0 | -2| 10| 10| 10| 10| 10| 10| 10| 10| 10| 10| 10| 10| 10| 10| 10|
| I   | -1| -4| -3| -4| -2| -3| -4| -4| -4| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| L   | -2| -3| -4| -4| -2| -2| -3| -3| 2 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| K   | -1| 3 | 0 | -1| -3| 2 | 1 | -2| 0 | -3| 3 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| M   | -1| -2| -2| -4| -2| 0 | -2| -3| -1| 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| F   | -3| -3| -4| -5| -2| -4| -3| -4| -1| 0 | 1 | -4| 0 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| P   | -1| -3| -2| -1| -1| -2| -3| -4| -1| -3| -4| 10| 10| 10| 10| 10| 10| 10| 10| 10| 10| 10| 10|
| S   | 1 | -1| 1 | 0 | -1| 0 | -1| -1| -3| -3| 0 | -2| -3| -1| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| T   | 0 | -1| 0 | -1| -1| -1| -2| -2| -1| -1| -1| -1| -2| -1| 2 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| W   | -3| -3| -4| -5| -5| -1| -3| -3| -3| -2| -3| -1| 1 | -4| -4| -3| 15| 15| 15| 15| 15| 15| 15| 15|
| Y   | -2| -1| -2| -1| -2| -1| -1| -1| -1| -1| -2| 3 | 0 | 4 | -3| -2| -2| 2 | 2 | 2 | 2 | 2 | 2 |
| V   | 0 | -3| -3| -4| -1| -3| -3| -4| -4| 4 | 1 | -3| 1 | -1| -3| -2| 0 | -3| -1| 5 | 5 | 5 | 5 |

A R N D C Q E G H I L K M F P S T W Y V
Pairwise alignments: insertions/deletions

43.2% identity;  Global alignment score: 374

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>V-LSPADKTNVKAAWGKVGAHAGEYGAEEALERMFLSFPTTKTYFPFH-DLS-----HGSA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>beta</td>
<td>VHLTPEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMSGN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
<th>110</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>QVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAAHKLVRDPVNFKLASSCLVLVTLAAAHL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>beta</td>
<td>KVKAHGKKVLGAFSDGLAHLDNKLGLTFATLSELHCDDKLHDPENFRLLGNVVLVCVLAHHF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>120</th>
<th>130</th>
<th>140</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>PAEFTPAVHASLDKFLASVSTVLTFSKVR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>beta</td>
<td>GKEFTPPVQARYQKVVAGVANALAHKYH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Alignment scores: insertions/deletions

Affine gap penalties:
Multiple insertions/deletions may be one evolutionary event =>
Separate penalties for gap opening and gap elongation
Compute 4 alignment scores: two different alignments using two different alignment matrices (and the same gap penalty system)

Score 1: Alignment 1 + BLOSUM-50 matrix + gaps
Score 2: Alignment 1 + BLOSUM-Trp matrix + gaps
Score 3: Alignment 2 + BLOSUM-50 matrix + gaps
Score 4: Alignment 2 + BLOSUM-Trp matrix + gaps

Note: fake matrix constructed for pedagogic purposes.
### Handout: summary of results

<table>
<thead>
<tr>
<th></th>
<th>Alignment 1</th>
<th>Alignment 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BLOSUM-50</strong></td>
<td>38</td>
<td>51</td>
</tr>
<tr>
<td><strong>BLOSUM-Trp</strong></td>
<td>118</td>
<td>91</td>
</tr>
</tbody>
</table>
### Protein substitution matrices

**BLOSUM50 matrix:**

- Positive scores on diagonal (identities)
- Similar residues get higher (positive) scores
- Dissimilar residues get smaller (negative) scores

|     | A   | R   | N   | D   | C   | Q   | E   | G   | H   | I   | L   | K   | M   | F   | P   | S   | T   | W   | Y   | V   |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| A   | 5   | -2  | -1  | -2  | -1  | 1   | 0   | 0   | -3  | -1  | 3   | 0   | -1  | 3   | 6   | 0   | -1  | 0   | -1  | 1   | -1  | -1  | -1  | -1  | -1  | -1  | -1  | -1  | -1  | -1  | -1  |
| R   | -2  | 7   | -1  | -1  | 7   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| N   | -1  | -1  | 7   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| D   | -2  | -2  | 2   | 8   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| C   | -1  | -4  | -2  | -4  | 13  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| Q   | -1  | 1   | 0   | 0   | -3  | 7   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| E   | -1  | 0   | 0   | 2   | -3  | 2   | 6   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| G   | 0   | -3  | 0   | -1  | -3  | -2  | -3  | 8   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| H   | -2  | 0   | 1   | -1  | -3  | 1   | 0   | -2  | 10  |     |     |     |     |     |     |     |     |     |     |     |     |     |
| I   | -1  | -4  | -3  | -4  | -2  | -3  | -4  | -4  | -4  | 5   |     |     |     |     |     |     |     |     |     |     |     |     |
| L   | -2  | -3  | -4  | -2  | -2  | -3  | -4  | -3  | 2   | 5   |     |     |     |     |     |     |     |     |     |     |     |
| K   | -1  | 3   | 0   | -1  | -3  | 2   | 1   | -2  | 0   | -3  | -3  | 6   |     |     |     |     |     |     |     |
| M   | -1  | -2  | -2  | -4  | -2  | 0   | -2  | -3  | -1  | 2   | 3   | -2  | 7   |     |     |     |     |     |     |
| F   | -3  | -3  | -4  | -5  | -2  | -4  | -3  | -4  | -4  | -1  | 0   | 1   | -4  | 0   | 8   |     |     |     |     |     |
| P   | -1  | -3  | -2  | -1  | -4  | -1  | -1  | -2  | -2  | -3  | -4  | -1  | -3  | -4  | 10  |     |     |     |     |     |     |
| S   | 1   | -1  | 1   | 0   | -1  | 0   | -1  | 0   | -1  | -3  | -3  | 0   | -2  | -3  | -1  | 5   |     |     |     |     |     |
| T   | 0   | -1  | 0   | -1  | -1  | -1  | -1  | -2  | -2  | -1  | -1  | -1  | -1  | -2  | -1  | 2   | 5   |     |     |     |     |     |
| W   | -3  | -3  | -4  | -5  | -5  | -1  | -3  | -3  | -3  | -2  | -3  | -3  | -1  | 1   | -4  | -4  | -3  | 15  |     |     |     |     |     |
| Y   | -2  | -1  | -2  | -3  | -3  | -1  | -2  | -3  | 2   | -1  | -1  | -2  | 0   | 4   | -3  | -2  | -2  | 2   | 8   |     |     |     |     |
| V   | 0   | -3  | -3  | -4  | -1  | -3  | -3  | -4  | -4  | 4   | 1   | -3  | 1   | -1  | -3  | -2  | 0   | -3  | -1  | 5   |     |     |     |     |

**A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V**
Protein substitution matrices: different types

- **Identity matrix**
  (match vs. mismatch)

- **Genetic code matrix**
  (how similar are the codons?)

- **Chemical properties matrix**
  (use knowledge of physicochemical properties to design matrix)

- **Empirical matrices**
  (based on observed pair-frequencies in hand-made alignments)
  - PAM series
  - BLOSUM series
  - Gonnet
Estimation of the BLOSUM 50 matrix

- For each alignment in the BLOCKS database the sequences are grouped into clusters with at least 50% identical residues (for BLOSUM 50).

- All pairs of sequences are compared, and the observed pair frequencies are noted (e.g., A aligned with A makes up 1.5% of all pairs. A aligned with C makes up 0.01% of all pairs, etc.)

- Expected pair frequencies are computed from single amino acid frequencies. (e.g., $f_{A,C} = f_A \times f_C = 7\% \times 3\% = 0.21\%$).

- For each amino acid pair the substitution scores are essentially computed as:

  $$S_{A,C} = \log \frac{\text{Pair-freq}(\text{obs})}{\text{Pair-freq}(\text{expected})} = \log \frac{0.01\%}{0.21\%} = -1.3$$
Pairwise alignment

Optimal alignment:

alignment having the highest possible score given a substitution matrix and a set of gap penalties
Pairwise alignment: the problem

The number of possible pairwise alignments increases explosively with the length of the sequences:

Two protein sequences of length 100 amino acids can be aligned in approximately $10^{60}$ different ways.

Time needed to test all possibilities is same order of magnitude as the entire lifetime of the universe.
Pairwise alignment: the solution

"Dynamic programming"
(the Needleman-Wunsch algorithm)
Alignment depicted as path in matrix

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

\[
\begin{array}{cccccc}
T & C & G & C & A \\
T & & & & \rightarrow TC\text{--}CA \\
C & & & & \rightarrow TC\text{--}CA \\
C & & & & \rightarrow TC\text{--}CA \\
A & & & & \rightarrow TC\text{--}CA \\
\end{array}
\]
Dynamic programming: computation of scores

Any given point in matrix can only be reached from three possible previous positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.
Dynamic programming: computation of scores

Any given point in matrix can only be reached from three possible positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

\[
score(x, y) = \max \left\{ \text{score}(x, y-1) - \text{gap-penalty} \right\}
\]
Dynamic programming: computation of scores

Any given point in matrix can only be reached from three possible positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

\[
score(x,y) = \max \begin{cases} 
    score(x,y-1) - \text{gap-penalty} \\
    score(x-1,y-1) + \text{substitution-score}(x,y)
\end{cases}
\]
Dynamic programming: computation of scores

Any given point in matrix can only be reached from three possible positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

\[
\text{score}(x, y) = \max \left\{ \begin{array}{l}
\text{score}(x, y-1) - \text{gap-penalty} \\
\text{score}(x-1, y-1) + \text{substitution-score}(x, y) \\
\text{score}(x-1, y) - \text{gap-penalty}
\end{array} \right\}
\]
Dynamic programming: computation of scores

Any given point in matrix can only be reached from three possible positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

Each new score is found by choosing the maximum of three possibilities. For each square in matrix: keep track of where best score came from.

Fill in scores one row at a time, starting in upper left corner of matrix, ending in lower right corner.

\[
\text{score}(x,y) = \max \begin{cases} 
\text{score}(x,y-1) - \text{gap-penalty} \\
\text{score}(x-1,y-1) + \text{substitution-score}(x,y) \\
\text{score}(x-1,y) - \text{gap-penalty}
\end{cases}
\]
Dynamic programming: example

\[
a[i,j] = \max \begin{cases} 
a[i,j-1] - 2 \\
a[i-1,j-1] + p(i,j) \\
a[i-1,j] - 2 \end{cases}
\]

\[
\begin{array}{cccccc}
A & C & G & T \\
A & 1 & -1 & -1 & -1 \\
C & -1 & 1 & -1 & -1 \\
G & -1 & -1 & 1 & -1 \\
T & -1 & -1 & -1 & 1 \\
\end{array}
\]

Gaps: -2
Dynamic programming: example

\[ a[i, j] = \max \begin{cases} a[i, j-1] - 2 \\ a[i-1, j-1] + p(i, j) \\ a[i-1, j] - 2 \end{cases} \]
Dynamic programming: example

\[
a[i, j] = \max \begin{cases} 
  a[i, j-1] - 2 \\
  a[i-1, j-1] + p(i, j) \\
  a[i-1, j] - 2
\end{cases}
\]
Dynamic programming: example
Dynamic programming: example
Dynamic programming: example

\[
\begin{align*}
\text{a}[i,j] &= \max \left\{ a[i-1,j-1] - 2, a[i-1,j] + p(i,j), a[i,j-1] - 2 \right\}
\end{align*}
\]
Dynamic programming: example

Diagram showing a grid of numbers with arrows indicating the movement from one number to another.
Dynamic programming: example

\[
\begin{array}{cccccc}
& T & C & G & C & A \\
0 & 0 & -2 & -4 & -6 & -8 & -10 \\
1 & -2 & 1 & -1 & -3 & -5 & -7 \\
2 & -4 & -1 & 2 & 0 & -2 & -4 \\
3 & -6 & -3 & 0 & 1 & 1 & -1 \\
4 & -8 & -5 & -2 & -1 & 0 & 2 \\
\end{array}
\]

\[
\begin{align*}
T & C & G & C & A \\
& : & : & : & : \\
T & C & C & A \\
\frac{1+1-2+1+1}{1+1-2+1+1} &= 2
\end{align*}
\]
Global versus local alignments

Global alignment: align full length of both sequences. (The “Needleman-Wunsch” algorithm).

Local alignment: find best partial alignment of two sequences (the “Smith-Waterman” algorithm).
Local alignment overview

- The recursive formula is changed by adding a fourth possibility: zero. This means local alignment scores are never negative.

\[
\text{score}(x,y) = \max \begin{cases} 
\text{score}(x,y-1) - \text{gap-penalty} \\
\text{score}(x-1,y-1) + \text{substitution-score}(x,y) \\
\text{score}(x-1,y) - \text{gap-penalty} \\
0
\end{cases}
\]

- Trace-back is started at the highest value rather than in lower right corner
- Trace-back is stopped as soon as a zero is encountered
**Local alignment: example**

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>A</th>
<th>W</th>
<th>H</th>
<th>E</th>
<th>G</th>
<th>A</th>
<th>W</th>
<th>G</th>
<th>H</th>
<th>E</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>10</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>18</td>
<td>22</td>
<td>14</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>2</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>10</td>
<td>18</td>
<td>28</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>8</td>
<td>21</td>
<td>13</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>10</td>
<td>20</td>
<td>27</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>6</td>
<td>13</td>
<td>18</td>
<td>12</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>16</td>
<td>26</td>
</tr>
</tbody>
</table>

**AWGHE**

**AW-HE**
Substitution matrices and sequence similarity

- Substitution matrices come as series of matrices calculated for different degrees of sequence similarity (different evolutionary distances).

- "Hard" matrices are designed for similar sequences
  - Hard matrices are designated by high numbers in the BLOSUM series (e.g., BLOSUM80)
  - Hard matrices yield short, highly conserved alignments

- "Soft" matrices are designed for less similar sequences
  - Soft matrices have low BLOSUM values (45)
  - Soft matrices yield longer, less well conserved alignments
Alignments: things to keep in mind

“Optimal alignment” means “having the highest possible score, given substitution matrix and set of gap penalties”.

This is NOT necessarily the biologically most meaningful alignment.

Specifically, the underlying assumptions are often wrong: substitutions are not equally frequent at all positions, affine gap penalties do not model insertion/deletion well, etc.

Pairwise alignment programs always produce an alignment - even when it does not make sense to align sequences.
BLAST

Anders Gorm Pedersen
&
Rasmus Wernersson
Database searching

Using pairwise alignments to search databases for similar sequences

Query sequence

Database
Database searching

Most common use of pairwise sequence alignments is to search databases for related sequences. For instance: find probable function of newly isolated protein by identifying similar proteins with known function.

Most often, *local* alignment ("Smith-Waterman") is used for database searching: you are interested in finding out if ANY domain in your protein looks like something that is known.

Often, full Smith-Waterman is too time-consuming for searching large databases, so heuristic methods are used (fasta, BLAST).
## Database searching: heuristic search algorithms

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses heuristics to avoid calculating the full dynamic programming matrix</td>
<td>Uses rapid word lookup methods to completely skip most of the database entries</td>
</tr>
<tr>
<td>Speed up searches by an order of magnitude compared to full Smith-Waterman</td>
<td>Extremely fast</td>
</tr>
<tr>
<td>The statistical side of FASTA is still stronger than BLAST</td>
<td>One order of magnitude faster than FASTA</td>
</tr>
<tr>
<td></td>
<td>Two orders of magnitude faster than Smith-Waterman</td>
</tr>
<tr>
<td></td>
<td>Almost as sensitive as FASTA</td>
</tr>
</tbody>
</table>
BLAST flavors

BLASTN
- Nucleotide query sequence
- Nucleotide database

BLASTP
- Protein query sequence
- Protein database

BLASTX
- Nucleotide query sequence
- Protein database
  - Compares all six reading frames with the database

TBLASTN
- Protein query sequence
- Nucleotide database
  - "On the fly" six frame translation of database

TBLASTX
- Nucleotide query sequence
- Nucleotide database
  - Compares all reading frames of query with all reading frames of the database
Searching on the web: BLAST at NCBI

Very fast computer dedicated to running BLAST searches

Many databases that are always up to date (e.g. NR and Human Genome)

Nice simple web interface

But you still need knowledge about BLAST to use it properly
When is a database hit significant?

• **Problem:**
  - Even unrelated sequences can be aligned (yielding a low score)
  - How do we know if a database hit is meaningful?
  - When is an alignment score sufficiently high?

• **Solution:**
  - Determine the range of alignment scores you would expect to get for random reasons (i.e., when aligning unrelated sequences).
  - Compare actual scores to the distribution of random scores.
  - Is the real score much higher than you’d expect by chance?
Distribution of random alignment scores

- Software simulation
Random alignment scores follow extreme value distributions

Searching a database of unrelated sequences result in scores following an extreme value distribution.

The exact shape and location of the distribution depends on the exact nature of the database and the query sequence.
Significance of a hit: one possible solution

(1) Align query sequence to all sequences in database, note scores

(2) Determine shape of background distribution (which is an extreme value distribution) from distribution of all scores

(3) Use fitted extreme-value distribution to predict how many random hits to expect for any given score (the “E-value”)

BLAST uses precomputed extreme value distributions to calculate E-values from alignment scores.

For this reason BLAST only allows certain combinations of substitution matrices and gap penalties. This also means that the fit is based on a different data set than the one you are working on.

A word of caution: BLAST tends to overestimate the significance of its matches.

E-values from BLAST are fine for identifying sure hits. One should be careful using BLAST’s E-values to judge if a marginal hit can be trusted (e.g., you may want to use E-values of $10^{-4}$ to $10^{-5}$).
BLAST heuristics

- Best possible search:
  - Do full pairwise alignment (Smith-Watermann) between the query sequence and all sequences in the database.
  - (“ssearch” does this).

- BLAST speeds up the search by at least two orders of magnitude, by pre-screening the database sequences and only performing the full Dynamic Programming on “promising” sequences.

- This is done by indexing all database sequences in a so-called suffix-tree which makes it very fast to search for perfect matching sub-strings.
  - A suffix tree is the quickest possible way (so far) to search for the longest matching sub-string between two strings.

- When a BLAST search is run, candidate sequences from the database is picked based on perfect matches to small sub-sequences in the query sequence. (BLASTN and BLASTP does this differently - more about this in a moment).
  - Full Smith-Waterman is then performed on these sequences.
BLASTN

- **Heuristics:**
  - Perfect match “word” of the size: 7, 11 (default) or 15.

- **Alignment matrix:**
  - Match: 1
  - Mismatch: -3

- **Notice:** All mismatches are equally penalized:
  - E.g. A:G == A:C == A:T
  - More advanced models for DNA evolution does exist.

Potential matches of length < word size (not seen by BLAST)

Subset to align

All sequences

Match => word size
**BLASTP**

- **Heuristics:**
  - 2 x “Near match” within a window.
  - Default word length: 3 aa
  - Default window length: 40 aa

- **Alignment matrix:**
  - PAM and BLOSUM-series (default: BLOSUM 62)

- **Notice:** These alignment matrices incorporate knowledge about protein evolution.