Pairwise Alignment Algorithm Using Dynamic Programming

- Initialization: Given two sequences with length m and n, create a \((m+1) \times (n+1)\) matrix \(M\). Initialize the first row and first column according to scoring matrix.

- For \(j\) in 1..n (column)
  
  for \(i\) in 1..m (row)

  \[
  M[i,j] = \max( (M[i-1,j-1]+S(i,j), M[i,j-1]+S(-,j), M[i-1, j] + S(i,-) )
  \]

  Record the selected path toward \((i,j)\)

- Report alignment score \(M[m][n]\) and trace back to \(M[0,0]\) to generate the optimal alignment.
Local Sequence Alignment Using DP

- Biological sequences usually only have local similarity. For instance: a protein sequence may consist of a few modules. Two proteins may only have one similar modules, whereas other regions are not similar at all.
- During evolution, only functional and structural important regions are highly conserved.
- Global alignment sacrifices the local similarity to maximize the global alignment score.
- We need to use alignment method to identify the local similar regions disregard of other dissimilar regions.
Local vs. Global Alignment

• Global Alignment

```
--T--CC-C-AGT--TATGT-CAGGGGACACG--A-GCATGCAGA-GAC
```

```
AATTGCCGCCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT--C
```

• Local Alignment—better alignment to find conserved segment

```
tccCAGTTATGTCAGgggacacgcagcatgcagagac
```

```
aattgcgcgcgtgcgttttcagCAGTTATGTCAGatc
```

Transcription binding site
Local Alignment Algorithm

**Goal**: find an alignment of the substrings of P and Q with maximum alignment score.

**Naïve Algorithm**:

(m+1)*m/2 substrings of P, (n+1)*n/2 substrings of Q

Using DP for each substring pairs:

m^2 * n^2 * O(mn) = O(m^3n^3)

(too slow!)
Smith-Waterman Algorithm

Same Dynamic Program algorithm as global alignment except for three differences.

1. All negative scores is converted to 0 (why?)
2. Alignment can start from anywhere in the matrix
3. Alignment can end at anywhere in the matrix
Local Alignment Algorithm

- Initialization: Given two sequences with length $m$ and $n$, create a $(m+1) \times (n+1)$ matrix $M$. Initialize the first row and first column to 0s.
- For $j$ in $1..n$ (column)
  for $i$ in $1..m$ (row)
    $M[i,j] = \max( 0, (M[i-1,j-1]+S(i,j), M[i,j-1]+S(-,j), M[i-1,j]+S(i,-) )$
    Record the selected path.
- Find elements in matrix $M$ with maximum values. Trace back till 0 and report the alignment corresponding to the path.
1. Initialization

<table>
<thead>
<tr>
<th></th>
<th>_</th>
<th>A</th>
<th>T</th>
<th>A</th>
<th>G</th>
<th>A</th>
<th>A</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 3. Trace Back

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>T</th>
<th>A</th>
<th>G</th>
<th>A</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</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>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
3. Trace Back

<table>
<thead>
<tr>
<th></th>
<th>-</th>
<th>A</th>
<th>T</th>
<th>A</th>
<th>G</th>
<th>A</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</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>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>
Two Best Local Alignments

Local Alignment 1:
- ATCAGAA
- AT-AGAA

Local Alignment 2:
- ATCAGAAAT
- AT-AGA-AT
Affine Gap Penalty

- We have treated gaps independently. However, gaps in biological sequences are dependent.
- Hard to open the first gap, and easy to extend gaps.
- First gap should be penalized more than extended gaps.
Affine Gap Function

- Penalty of gaps = Wo + We * (g-1)
- Wo: penalty of opening a gap
- We: penalty of extending a gap
- g: gap size
- Example: Wo = -2, We = -1
Example:

AGATCAGAAATG
--AT-AG-AAT--

Alignment score: -2 -1 +1 +1 – 2 +1 +1 -2 +1 +1 + 1 -2 = -2
Complexity of Affine Gap Penalty

- To compute the alignment score, we need to know the state of the previous alignment:
  - last pair matches two characters?
  - last pair matches $P[k]$ with a gap?
  - last pair matches $Q[t]$ with a gap?

So we need to introduce more matrices to record the scores in these three situations.
Dynamic programming with Affine Gap Penalty

- Algorithms proceed by aligning $P[1..i]$ with $Q[1..j]$. For these prefixes of $P$ and $Q$, define the following four matrices $V$, $G$, $F$, $E$:
  - $V[i,j]$ is the value of an optimal alignment of prefix $P[1..i]$ and prefix $Q[1..j]$.
  - $G[i,j]$ is the value of an optimal alignment of $P[1..i]$ and $Q[1..j]$ whose last pair matches $P[i]$ with $Q[j]$.
  - $F(i,j)$ is the value of an optimal alignment of $P[1..i]$ and $T[1..j]$ whose last pair matches $P[i]$ with a gap.
  - $E(i,j)$ is the value of an optimal alignment of $P[1..i]$ and $Q[1..j]$ whose last pair matches a gap with $Q[j]$. 
Initialization

- \( V(0,0) = 0 \)
- \( V(i,0) = Wo + We \times (i - 1) \)
- \( V(0,j) = Wo + We \times (j-1) \)
- \( E(i,0) = -\infty \)
- \( F(0,j) = -\infty \)
Fill Matrix

• \( V(i,j) = \max (G(i,j), F(i,j), E(i,j)) \)
• \( G(i,j) = V(i-1, j-1) + S(P[i], Q[j]) \)
• \( F(i,j) = \max(F(i-1,j)+We, G(i-1,j) + Wo, E(i-1, j) + Wo) \)
• \( E(i,j) = \max(E(i,j-1) + We, G(i,j-1) + Wo, F(i,j-1) + Wo) \)
Trace Back

• Trace within matrix and across matrices
• Time and space complexity: O(m*n)
• Comments: four matrices. 3 matrices for three different possible ending states. 1 matrix is the master matrix to record the best alignment scores from those three matrices.
Comments

• Does sequence alignment based on mathematical optimization really make biological sense?
• When alignment is good?
• When alignment is bad?
• Alignment quality and sequence identity relationship
Application Example (Alignment – Structure)

Source: A. Fisher, 2005
%Sequence Identity: percent of identical residues in alignment
RMSD: square root of average distance between predicted structure and native structure.
Global and Local Alignment Tools

- NEEDLE (global alignment)
  http://bioweb.pasteur.fr/seqanal/interfaces/needle.html

- WATER (local alignment)
  http://bioweb.pasteur.fr/seqanal/interfaces/water.html