Advanced Sequence Alignment

CLUSTAL O(1.2.1) multiple sequence alignment

Cat
MAPWTRLLPALLSLWIPAPTRAVNVQHLCSGLHVLEALYLVCGERGFFYTPKARREAE 60
Pig
MALWTRLPLALLALALWAFAPAQAVNVQHLCSGLHVLEALYLVCGERGFFYTPKARREAE 60
Human
MALWRLLPLALLALALWAFAPAQAVNVQHLCSGLHVLEALYLVCGERGFFYTPKARREAE 60
Dog
MALWRLLPLALLALALWAFAPAQAVNVQHLCSGLHVLEALYLVCGERGFFYTPKARREAE 60
** ********:** * *: ************************:**:**:**

Cat
LQKDAElGEAPGAGGLQPSALEAPLQKRGGIEQCCASVCSLQLEHCN 110
Pig
PQAGAVELGG--GLGLQALALGETQKRGIEQCCCTICSLQLEHCN 108
Human
LQ-----GLGLQALAEQLQKRGGIEQCCCTICSLQLEHCN 98
Dog
LVQRDLAGAPEGLQQLALGALQKRGGIEQCCCTICSLQLEHCN 110

- Midterm on Wednesday
  - Covers up to and including Lecture 11
  - Online; it can be downloaded at the start of class; same Jupyter Nookbook format as Problem Sets
  - Open Computer, Open Notes
  - You can add extra cells for scratch work, but only the indicated answer cells will be graded
  - Mix of short answer, multiple choice, and writing code fragments
  - Hard deadline for submission! Bank versions as the time limit approaches.
Recall Local Alignment

![Recall Local Alignment formula]

\[ s_{i,j} = \max \begin{cases} 
0, \\
\delta(v_p \cdot, w_j) + s_{i-1,j-1}, \\
\delta(v_p , \cdot) + s_{i-1,j}, \\
\delta(\cdot , w_j) + s_{i,j-1} 
\end{cases} \]

Notice there is only this small change from the original recurrence of a Global Alignment

- The zero is our free ride that allows the node to restart with a score of 0 at any point
  - What does this imply?
- After solving for the entire score matrix, we then search for \( s_{i,j} \) with the highest score, this is \((i_2, j_2)\)
- We follow our back tracking matrix until we reach a score of 0, whose coordinate becomes \((i_1, j_1)\)
Smith-Waterman Local Alignment

Key idea: Adding "free-rides" from the source to any intersection
A Local Alignment Example

<table>
<thead>
<tr>
<th>i=</th>
<th>0</th>
<th>-</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>A</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>j=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>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

1 G 0
2 C 0
3 A 0
4 G 0
5 A 0
6 G 0
7 C 0
8 A 0
9 C 0
10 T 0

Match = 5, Mismatch = -4, Indel = -7
A Local Alignment Example - continued

\[
\begin{array}{cccccccccccccc}
\text{j}=0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\text{i}= & - & G & C & T & G & G & A & A & G & G & C & A & T \\
0 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & G & 0 & S_{1,1} & S_{1,1} & S_{0,0} + s_{G,G} = 0 + 5 = 5 & S_{1,0} + w = 0 - 7 = -7 & S_{0,1} + w = 0 - 7 = -7 & 0 \\
2 & C & 0 & S_{1,1} & S_{1,1} & S_{1,0} + w = 0 - 7 = -7 & S_{0,1} + w = 0 - 7 = -7 & 0 & 0 \\
3 & A & 0 & S_{1,1} & S_{1,1} & S_{1,0} + w = 0 - 7 = -7 & S_{0,1} + w = 0 - 7 = -7 & 0 & 0 & 0 \\
4 & G & 0 & S_{1,1} & S_{1,1} & S_{1,0} + w = 0 - 7 = -7 & S_{0,1} + w = 0 - 7 = -7 & 0 & 0 & 0 & 0 \\
5 & A & 0 & S_{1,1} & S_{1,1} & S_{1,0} + w = 0 - 7 = -7 & S_{0,1} + w = 0 - 7 = -7 & 0 & 0 & 0 & 0 & 0 \\
6 & G & 0 & S_{1,1} & S_{1,1} & S_{1,0} + w = 0 - 7 = -7 & S_{0,1} + w = 0 - 7 = -7 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & C & 0 & S_{1,1} & S_{1,1} & S_{1,0} + w = 0 - 7 = -7 & S_{0,1} + w = 0 - 7 = -7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & A & 0 & S_{1,1} & S_{1,1} & S_{1,0} + w = 0 - 7 = -7 & S_{0,1} + w = 0 - 7 = -7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
9 & C & 0 & S_{1,1} & S_{1,1} & S_{1,0} + w = 0 - 7 = -7 & S_{0,1} + w = 0 - 7 = -7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
10 & T & 0 & S_{1,1} & S_{1,1} & S_{1,0} + w = 0 - 7 = -7 & S_{0,1} + w = 0 - 7 = -7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Match = 5, Mismatch = -4, Indel = -7
A Local Alignment Example - continued

\[
\begin{array}{cccccccccccccc}
  & j=0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
  i= & - & G & C & T & G & G & A & A & G & G & C & A & T \\
  0 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  1 & G & 0 & 5 & S_{1,2} & & & & & & & & & \\
  2 & C & 0 & & & & & & & & & & & \\
  3 & A & 0 & & & & & & & & & & & \\
  4 & G & 0 & & & & & & & & & & & \\
  5 & A & 0 & & & & & & & & & & & \\
  6 & G & 0 & & & & & & & & & & & \\
  7 & C & 0 & & & & & & & & & & & \\
  8 & A & 0 & & & & & & & & & & & \\
  9 & C & 0 & & & & & & & & & & & \\
 10 & T & 0 & & & & & & & & & & & \\
\end{array}
\]

\[
S_{1,2} = \max \begin{cases} 
S_{0,1} + s_{G,C} = 0 - 4 = -4 \\
S_{1,2} + w = 5 - 7 = -2 \\
S_{0,2} + w = 0 - 7 = -7 \end{cases} = 0
\]

Match = 5, Mismatch = -4, Indel = -7
A Local Alignment Example - continued

\[
\begin{array}{cccccccccccccc}
& 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
i= & - & G & C & T & G & G & A & A & G & G & C & A & T \\
0 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & G & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & C & 0 & 0 & S_{2,2} \\
3 & A & 0 \\
4 & G & 0 \\
5 & A & 0 \\
6 & G & 0 \\
7 & C & 0 \\
8 & A & 0 \\
9 & C & 0 \\
10 & T & 0 \\
\end{array}
\]

\[
S_{2,2} = \max \begin{cases}
S_{1,1} + s_{C,C} = 5 + 5 = 10 \\
S_{2,1} + w = 0 - 7 = -7 \\
S_{1,2} + w = 0 - 7 = -7 \\
0
\end{cases} = 10
\]

Match = 5, Mismatch = -4, Indel = -7
## A Local Alignment Example - continued

<table>
<thead>
<tr>
<th>0</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>A</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<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>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>11</td>
<td>5</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>10</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>9</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>9</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>7</td>
<td>19</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

Match = 5, Mismatch = -4, Indel = -7
Once the score table is completed, find the largest score attained, then backtrack from there to find the alignment.
A Local Alignment Example - continued

G C T G G A A G - G C A T
|   |   |   |   |
G C A G A G C A C T

6 matches: $6 \times 5 = 30$
1 mismatch: -4
1 indel: -7
Total: 19
Scoring Indels: Naive Approach

ATCTTCAGCCATAAAAAGATGAAGTT Reference
ATCTTCAGCCAAAGATGAAGTT 3 base deletion relative to the reference
ATCTTCAGCCATAAAAGATGAAGTT version 1
ATCTTCAGCCATAAAAGATGAAGTT version 2
ATCTTCAGCCATAAAAGATGAAGTT version 3
ATCTTCAGCCATAAAAGATGAAGTT version 4
ATCTTCAGCCATAAAAGATGAAGTT version 5
ATCTTCAGCCATAAGTGTAAGAAAGATGAAGTT 4 base insertion

- A fixed penalty $\sigma$ is given to every indel:
  - $-\sigma$ for 1 indel,
  - $-2\sigma$ for 2 consecutive indels
  - $-3\sigma$ for 3 consecutive indels, etc.
- Can be too severe penalty for a series of 100 consecutive indels
  - Large insertions or deletions might result from a single event
Affine Gap Penalties

• In nature, a series of $k$ indels often come as a single event rather than a series of $k$ single nucleotide events:

```
AT___GC
ATTGAGC
```

This is more likely. Explained by one event

```
A_TG__C
ATTGAGC
```

Normal scoring would give the same score for both alignments

This is less likely. Requires 2 events.
Gaps - contiguous sequence of indels in one of the rows
Modify the scoring for a gap of length $x$ to be:

$$-(\rho + \sigma x)$$

where $\rho + \sigma > 0$ is the penalty for introducing a gap:

$$\rho = \text{gap opening penalty}$$

and $\sigma$ is the cost of extending it further ($\rho + \sigma >> \sigma$):

$$\sigma = \text{gap extension penalty}$$

because you do not want to add too much of a penalty for further extending the gap, once it is opened.
Affine Gap Penalties

- Gap penalties:
  - $\cdot \rho - \sigma$ when there is 1 indel
  - $\cdot \rho - 2\sigma$ when there are 2 indels
  - $\cdot \rho - 3\sigma$ when there are 3 indels, etc.
  - $\cdot \rho - x\cdot\sigma$ (gap opening - x gap extensions)

- Somehow reduced penalties (as compared to naïve scoring) are given to runs of horizontal and vertical edges
Adding Affine Gap Penalties to our Graph

- To reflect affine gap penalties we have to add “long” horizontal and vertical edges to the edit graph.
- Each such edge of length x should have weight $-\rho - x\cdot\sigma$
- There are many such edges!
- Adding them to the graph increases the running time of the alignment algorithm by a factor of $n$ (where $n$ is the number of vertices)
- So the complexity increases from $O(n^2)$ to $O(n^3)$
Adding Two More Tables

- Affine Gap penalties can be expressed in terms of 3 recurrences

\[ t_{i,j} = \max \begin{cases} t_{i-1,j} - \sigma \\ s_{i-1,j} - (\rho + \sigma) \end{cases} \quad \text{Continue Gap in } w \text{ (deletion)} \]
\[ u_{i,j} = \max \begin{cases} u_{i,j-1} - \sigma \\ s_{i,j-1} - (\rho + \sigma) \end{cases} \quad \text{Start Gap in } w \text{ (deletion): from middle} \]
\[ s_{i,j} = \max \begin{cases} s_{i-1,j-1} + \delta(v_i, w_j) \\ t_{i,j} \\ u_{i,j} \end{cases} \quad \text{Match or Mismatch} \]
\[ \text{Continue Gap in } v \text{ (insertion)} \]
\[ \text{Start Gap in } v \text{ (insertion): from middle} \]
\[ \text{End deletion: from top} \]
\[ \text{End insertion: from left} \]
A 3-level Manhattan Grid

- The three recurrences for the scoring algorithm creates a 3-layered graph.
- The top level creates/extends gaps in the sequence $w$.
- The bottom level creates/extends gaps in sequence $v$.
- The middle level extends matches and mismatches.

Gaps in $w$ (t-table)

Matches/Mismatches (s-table)

Gaps in $v$ (u-table)
Switching between 3 Layers

- Levels:
  - The main level is for diagonal edges
  - The lower level is for horizontal edges
  - The upper level is for vertical edges
- A jumping penalty is assigned to moving from the main level to either the upper level or the lower level (-ρ - σ)
- There is a gap extension penalty for each continuation on a level other than the main level (-σ)
Multiple Alignment versus Pairwise Alignment

- Up until now we have only tried to align two sequences.
- What about more than two? And what for?
- A faint similarity between two sequences becomes significant if present in many
- Multiple alignments can reveal subtle similarities that pairwise alignments do not reveal
Generalizing Pairwise Alignment

- Alignment of 2 sequences is represented as a 2-row matrix
- In a similar way, we represent alignment of 3 sequences as a 3-row matrix

```
A  T  _  G  C  G  _
A  _  C  G  T  _  A
A  T  C  A  C  _  A
```

- Score: more conserved columns, better alignment
Three-D Alignment Paths

- An alignment of 3 sequences: ATGC, AATC, ATGC

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>x coordinate</td>
<td>A</td>
<td>--</td>
<td>T</td>
<td>G</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>y coordinate</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>z coordinate</td>
<td>A</td>
<td>A</td>
<td>T</td>
<td>--</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>--</td>
<td>A</td>
<td>T</td>
<td>G</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

- Resulting path in (x,y,z) space:
  (0,0,0) → (1,1,0) → (1,2,1) → (2,3,2) → (3,3,3) → (4,4,4)

- Is there a better one?
Aligning Three Sequences

- Same strategy as aligning two sequences
- Use a 3-D “Manhattan Cube”, with each axis representing a sequence to align
- For global alignments, go from source to sink
2-sequence vs 3-sequence Alignment

2-D edit graph

3-D edit graph
A 2-D cell versus a 3-D Alignment Cell

- 2-D $[(i-1,j-1), (i-1,j), (i,j-1)] \rightarrow (i,j)$
- 3-D $[(i-1,j-1,k-1), (i-1,j,k), (i,j-1,k), (i,j,k-1), (i-1,j,k-1), (i-1,j-1,k),] \rightarrow (i,j,k)$
Structure of a 3-D Alignment Cell

1- Match/Mismatch Path in 3 seqs

6 – indels paths
3 in one seq
3 in two seqs

(i-1,j-1,k-1) (i-1,j,k-1) (i-1,j,k)

(i,j-1,k-1) (i,j,k-1) (i,j,k)
Multiple Alignment: Recursion Relation

\[ s_{i,j,k} = \max \begin{cases} 
  s_{i-1,j-1,k-1} + \delta (v_i, w_j, u_k) & \text{cube diagonal: no indels} \\
  s_{i,j-1,k-1} + \delta (v_i, w_j, -) & \text{face diagonal: one indel} \\
  s_{i-1,j,k-1} + \delta (v_i, -, u_k) & \text{Lattice edge: two indels} \\
  s_{i,j,k-1} + \delta (-, w_j, u_k) & \\
  s_{i-1,j,k} + \delta (v_i, -, -) & \\
  s_{i,j-1,k} + \delta (-, w_j, -) & \\
  s_{i-1,j,k-1} + \delta (-, -, u_k) & 
\end{cases} \]

\[ \delta (x, y, z) \text{ is an entry in the 3-D scoring matrix} \]
Multiple Alignment: Running Time

- For 3 sequences of length n, the run time is $7n^3; O(n^3)$
- For k sequences, build a k-dimensional Manhattan, with run time $(2^k - 1)(n^k); O(2^kn^k)$
- Conclusion: dynamic programming approach for alignment between two sequences is easily extended to k sequences but it is impractical due to exponential running time
Multiple Alignment Induces Pairwise Alignments

Every multiple alignment induces pairwise alignments

\[
\begin{align*}
x & : \text{AC-GCGG-C} \\
y & : \text{AC-GC-GAG} \\
z & : \text{GCCGC-GAG}
\end{align*}
\]

Induces:

\[
\begin{align*}
x & : \text{ACGCGG-C;} \quad x : \text{AC-GCGG-C;} \quad y : \text{AC-GCGAG} \\
y & : \text{ACGC-GAC;} \quad z : \text{GCCGC-GAG;} \quad z : \text{GCCGCGAG}
\end{align*}
\]
Inverse Problem

Do Pairwise Alignments imply a Multiple Alignment?

- Given 3 arbitrary pairwise alignments:

  \[ x: \text{ACGCTGG-C}; \quad x: \text{AC-GCTGG-C}; \quad y: \text{AC-GC-GAG} \]
  \[ y: \text{ACGC-GAC}; \quad z: \text{GCCGCA-GAG}; \quad z: \text{GCCGCAGAG} \]

- Can we construct a multiple alignment that induces them?

  NOT ALWAYS

- Why? Because pairwise alignments may be arbitrarily inconsistent
Combining Optimal Pairwise Alignments

- In some cases we can combine pairwise alignments into a single multiple alignment
- But, in others we cannot because one alignment makes a choice that is inconsistent with the overall best choice

```
AAAATTTT - - - - - - - - - - - - AAAATTTT - - - - - - -
- - - - TTTTGGGG - - - - - OR - - - - - - - - TTTTGGGG
- - - - - - GGGGAAAAA GGGGAAAAA - - - - - - - -
```

- Is there another way?
Multiple Alignment from Pairwise Alignments

- From an optimal multiple alignment, we can infer pairwise alignments between all pairs of sequences, but they are not necessarily optimal
- It is difficult to infer a “good” multiple alignment from optimal pairwise alignments between all sequences
- Are we stuck, or is there some other trick?
Multiple Alignment using a Profile Scores

- We used profile scores earlier when we discussed Motif finding

- A G G C T A T C A C C T G
  T A G - C T A C C A - - - G
  C A G - C T A C C A - - - G
  C A G - C T A T C A C - G G
  C A G - C T A T C G C - G G

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

- Thus far we have aligned sequences against other sequences
- Can we align a sequence against a profile?
- Can we align a profile against a profile?
Aligning Alignments

A more general version of the multi-alignment problem:

- Given two alignments, can we align them?

  x:  GGGCACTGCAT
  y:  GGTTACGTC--    Alignment 1
  z:  GGGAACTGCAG

  w:  GGACGTACC--    Alignment 2
  v:  GGACCT------

- Idea: don’t use the sequences, but align their profiles

  x:  GGGCAC=TGCAT
  y:  GGTTAC=GTC--
  z:  GGGAAAC=TGCAG    Combined Alignment
      ||   ||   ||
  w:  GG==ACGTACC--
  v:  GG==ACCT------
Profile-Based Multiple Alignment: A Greedy Approach

- Choose the most similar pair of strings and combine them into a profile, thereby reducing alignment of \( k \) sequences to an alignment of of \( k-1 \) sequences/profiles. **Repeat**
- This is a heuristic **greedy** method

\[
\begin{align*}
\{ & u_1 = \text{ACGTACGTACGT} \ldots \\
& u_2 = \text{TTAATTAATTA} \ldots \\
& u_3 = \text{ACTACTACTACT} \ldots \\
& \vdots \\
& u_k = \text{CCGGCGGCGCGG} \\
\} & \quad \rightarrow \quad \{ & u_1 = \text{ACg/tTAcg/tTAcg/cT} \ldots \\
& u_2 = \text{TTAATTAATTA} \ldots \\
& \vdots \\
& u_k = \text{CCGGCGGCGCGG} \ldots \\
\} \\
\end{align*}
\]
Example

- Consider these 4 sequences

\[ s_1: \text{GATTCA} \]
\[ s_2: \text{GTCTGA} \]
\[ s_3: \text{GATATT} \]
\[ s_4: \text{GTCAGC} \]

- with the scoring matrix: \{Match = 1, Mismatch = -1, Indel = -1\}
There are $\binom{4}{2} = 6$ possible pairwise alignments

\begin{align*}
S_2 & : \text{GTCTGA} & S_1 & : \text{GATTCA--} \\
S_4 & : \text{GTCAGC (score = 2)} & S_4 & : \text{G-T-CAGC (score = 0)} \\
S_1 & : \text{GAT-TCA} & S_2 & : \text{G-TCTGA} & S_3 & : \text{GATAT-T (score = -1)} \\
S_2 & : \text{G-TCTGA (score = 1)} & S_3 & : \text{GAT-ATT} & S_4 & : \text{G-TCAGC (score = -1)} \\
S_1 & : \text{GAT-TCA} & S_3 & : \text{GATAT-T (score = 1)} & S_4 & : \text{G-TCAGC (score = -1)}
\end{align*}

The best pairwise score, 2, is between $S_2$ and $S_4$
Example (continued)

• Combine $s_2$ and $s_4$:

\[
\begin{aligned}
  s_2: & \quad G \ T \ C \ T \ G \ A \\
  \quad \mid \mid \mid \mid \mid \quad \quad \to \quad s_{2,4}: & \quad G \ T \ C \ t/a \ G \ a/c \\
  s_4: & \quad G \ T \ C \ A \ G \ C
\end{aligned}
\]

• Giving a set of three sequences:

\[
\begin{aligned}
  s_1 : & \quad G \ A \ T \ T \ C \ A \\
  s_3 : & \quad G \ A \ T \ A \ T \ T \\
  s_{2,4} : & \quad G \ T \ C \ t/a \ G \ a/c
\end{aligned}
\]

• **Repeat** for $\binom{3}{2} = 3$ possible pairwise alignments

\[
\begin{aligned}
  s_1 : & \quad \text{GAT-TCA} \\
  s_3 : & \quad \text{GATAT-T (score } = 1 + 1 + 1 - 1 + 1 - 1 - 1 = 1) \\
  s_1 : & \quad \text{GAT-TCA} \\
  s_{2,4} : & \quad \text{G-TctGa (score } = 2 - 2 + 2 - 2 + 1 - 1 + 1 = 1) \\
  s_3 : & \quad \text{GATAT-T} \\
  s_{2,4} : & \quad \text{G-TctGa (score } = 2 - 2 + 2 - 2 + 1 - 1 - 1 = -1)
\end{aligned}
\]
Progressive Alignment

- Progressive alignment is a variation of a greedy profile alignment algorithm with a somewhat more intelligent strategy for choosing the order of alignments.
- Progressive alignment works well for close sequences, but deteriorates for distant sequences
  - Once a gap appears in a consensus string it is permanent
  - Uses profiles to compare sequences
- CLUSTAL OMEGA
Clustal Omega

• A popular multiple alignment tool commonly used today
• ‘W’ stands for ‘weighted’ (different parts of alignment are weighted differently).
• Three-step process

1. Construct pairwise alignments
2. Build Guide Tree
3. Progressive Alignment guided by the tree
Clustal Omega's First Step

Pairwise alignment

- Align each sequence against all others giving a similarity matrix
- Similarity = exact matches / sequence length (percent identity)

\[
\begin{array}{cccc}
  & v_1 & v_2 & v_3 & v_4 \\
v_1 & - & .17 & - & - \\
v_2 & .17 & - & - & - \\
v_3 & .87 & .28 & - & - \\
v_4 & .59 & .33 & .62 & - \\
\end{array}
\]

(.17 means 17 % identical)
ClustalW's Second Step

- Create Guide Tree using the similarity matrix
  - ClustalW uses the neighbor-joining method
    (we will discuss this later in the course, in the section on clustering)
  - Guide tree roughly reflects evolutionary relations

<table>
<thead>
<tr>
<th></th>
<th>(v_1)</th>
<th>(v_2)</th>
<th>(v_3)</th>
<th>(v_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(v_1)</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(v_2)</td>
<td>.17</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(v_3)</td>
<td>.87</td>
<td>.28</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>(v_4)</td>
<td>.59</td>
<td>.33</td>
<td>.62</td>
<td>-</td>
</tr>
</tbody>
</table>

Calculate:
- \(v_{1,3} = \text{alignment}(v_1, v_3)\)
- \(v_{1,3,4} = \text{alignment}(v_{1,3}, v_4)\)
- \(v_{1,2,3,4} = \text{alignment}(v_{1,3,4}, v_2)\)
ClustalW's Third Step

- Start by aligning the two most similar sequences
- Following the guide tree, add in the next sequences, aligning to the existing alignment
- Insert gaps as necessary

Dots and stars show how well-conserved a column is.
Next Time

- Other approaches to sequence alignment
- Midterm next Wednesday
- Covers material up to Lecture 13