Multiple Alignment

- Problem definition
- Can we use Dynamic Programming to solve MSA?
- Progressive Alignment
- ClustalW
- Scoring Multiple Alignments
  - Entropy
  - Sum of Pairs (SP) Score

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

Multiple alignment

- One of the most essential tools in molecular biology
  - Finding highly conserved subregions or embedded patterns of a set of biological sequences
    - Conserved regions usually are key functional regions, prime targets for drug developments
    - Estimation of evolutionary distance between sequences
    - Prediction of protein secondary/tertiary structure
  - Practically useful methods only since 1987 (D. Sankoff)
    - Before 1987 they were constructed by hand
    - Dynamic programming is expensive
Multiple Sequence Alignment (MSA)

- What is multiple sequence alignment?
- Given $k$ sequences:

\[
\begin{align*}
VTISCTGSSSNIGAG&\quad VHNYQQLPG \\
VTISCTGSSSNIG&\quad --VHNYQQLPG \\
LRLSCSSGGFYTSSYAMVWVQAPG & \\
LSLTCTVQGSFDFYSTWVQAPG & \\
FPEVTVVQSVHEDPQVVKMNLYVSDG & \\
ATLVCCLISFYFPAGVTVAWKADS & \\
AAALCCLVYKYFPEFPTVSWSNG & \\
VSLTCVLKGYFYPDIAVEWESNG & \\
\end{align*}
\]

Multiple Sequence Alignment (MSA)

- An MSA of these sequences:

\[
\begin{align*}
VTISCTGSSSNIGAG&\quad VHNYQQLPG \\
VTISCTGSSSNIG&\quad --VHNYQQLPG \\
LRLSCSSGGFYTSSYAMVWVQAPG & \\
LSLTCTVQGSFDFYSTWVQAPG & \\
FPEVTVVQSVHEDPQVVKMNLYVSDG & \\
ATLVCCLISFYFPAGVTVAWKADS & \\
AAALCCLVYKYFPEFPTVSWSNG & \\
VSLTCVLKGYFYPDIAVEWESNG & \\
\end{align*}
\]

Multiple Sequence Alignment (MSA)

- An MSA of these sequences:

\[
\begin{align*}
VTISCTGSSSNIGAG&\quad VHNYQQLPG \\
VTISCTGSSSNIG&\quad --VHNYQQLPG \\
LRLSCSSGGFYTSSYAMVWVQAPG & \\
LSLTCTVQGSFDFYSTWVQAPG & \\
FPEVTVVQSVHEDPQVVKMNLYVSDG & \\
ATLVCCLISFYFPAGVTVAWKADS & \\
AAALCCLVYKYFPEFPTVSWSNG & \\
VSLTCVLKGYFYPDIAVEWESNG & \\
\end{align*}
\]

Conserved residues

Multiple Sequence Alignment (MSA)

- An MSA of these sequences:

\[
\begin{align*}
VTISCTGSSSNIGAG&\quad VHNYQQLPG \\
VTISCTGSSSNIG&\quad --VHNYQQLPG \\
LRLSCSSGGFYTSSYAMVWVQAPG & \\
LSLTCTVQGSFDFYSTWVQAPG & \\
FPEVTVVQSVHEDPQVVKMNLYVSDG & \\
ATLVCCLISFYFPAGVTVAWKADS & \\
AAALCCLVYKYFPEFPTVSWSNG & \\
VSLTCVLKGYFYPDIAVEWESNG & \\
\end{align*}
\]

Conserved regions

Multiple Sequence Alignment (MSA)

- An MSA of these sequences:

\[
\begin{align*}
VTISCTGSSSNIGAG&\quad VHNYQQLPG \\
VTISCTGSSSNIG&\quad --VHNYQQLPG \\
LRLSCSSGGFYTSSYAMVWVQAPG & \\
LSLTCTVQGSFDFYSTWVQAPG & \\
FPEVTVVQSVHEDPQVVKMNLYVSDG & \\
ATLVCCLISFYFPAGVTVAWKADS & \\
AAALCCLVYKYFPEFPTVSWSNG & \\
VSLTCVLKGYFYPDIAVEWESNG & \\
\end{align*}
\]

Patterns? Positions 1 and 3 are hydrophobic residues

Multiple Sequence Alignment (MSA)

- An MSA of these sequences:

\[
\begin{align*}
VTISCTGSSSNIGAG&\quad VHNYQQLPG \\
VTISCTGSSSNIG&\quad --VHNYQQLPG \\
LRLSCSSGGFYTSSYAMVWVQAPG & \\
LSLTCTVQGSFDFYSTWVQAPG & \\
FPEVTVVQSVHEDPQVVKMNLYVSDG & \\
ATLVCCLISFYFPAGVTVAWKADS & \\
AAALCCLVYKYFPEFPTVSWSNG & \\
VSLTCVLKGYFYPDIAVEWESNG & \\
\end{align*}
\]

Patterns? Positions 1 and 3 are hydrophobic residues

Multiple Sequence Alignment (MSA)

- An MSA of these sequences:

\[
\begin{align*}
VTISCTGSSSNIGAG&\quad VHNYQQLPG \\
VTISCTGSSSNIG&\quad --VHNYQQLPG \\
LRLSCSSGGFYTSSYAMVWVQAPG & \\
LSLTCTVQGSFDFYSTWVQAPG & \\
FPEVTVVQSVHEDPQVVKMNLYVSDG & \\
ATLVCCLISFYFPAGVTVAWKADS & \\
AAALCCLVYKYFPEFPTVSWSNG & \\
VSLTCVLKGYFYPDIAVEWESNG & \\
\end{align*}
\]

Patterns? Positions 1 and 3 are hydrophobic residues

MSA Warnings

- MSA algorithms work under the assumption that they are aligning related sequences
- They will align ANYTHING they are given, even if unrelated
- If it just "looks wrong" it probably is
Generalizing the Notion of 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

\[
\begin{align*}
A & \_ G C G _ \\
A & _ C G T _ A \\
A & T C A C _ A \\
\end{align*}
\]

- Score: more conserved columns, better alignment

Alignments = Paths in \( k \) dimensional grids

- Align 3 sequences: ATGC, AATC, ATGC

\[
\begin{align*}
& \begin{array}{c}
A - T G C \\
A A T - C \\
- A T G C \\
\end{array} \\
& \begin{array}{c}
A - T G C \\
A A T - C \\
- A T G C \\
\end{array}
\end{align*}
\]

Alignment Paths

- 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

\[
\begin{align*}
0 & 1 1 2 3 4 \\
A & - T G C \\
A & A T - C \\
- & A T G C \\
\end{align*}
\]

- Score: more conserved columns, better alignment

Alignment Paths

- 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

\[
\begin{align*}
0 & 1 1 2 3 4 \\
A & - T G C \\
0 & 1 2 3 3 4 \\
A & A T - C \\
0 & 0 1 2 3 4 \\
- & A T G C \\
\end{align*}
\]

- Resulting path in (x,y,z) space:

\[(0,0,0) \rightarrow (1,1,0) \rightarrow (1,2,1) \rightarrow (2,3,2) \rightarrow (3,3,3) \rightarrow (4,4,4)\]

Aligning Three Sequences

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

2-D cell versus 3-D Alignment Cell

Architecture of 3-D Alignment Cell

Multiple Alignment: Dynamic Programming

Multiple Alignment: Running Time

Multiple Alignment Induces Pairwise Alignments

- For 3 sequences of length $n$, the run time is $7n^3$; $O(n^3)$

- For $k$ sequences, build a $k$-dimensional matrix, 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

Every multiple alignment induces pairwise alignments

$\begin{align*}
x & : ACGCGG-C \\
y & : AC-CCG-C \\
z & : GC-GAG-C
\end{align*}$

Induces:

$\begin{align*}
x & : AC-GCGG-C \\
y & : ACGC-GAG \\
z & : GCCGC-GAG
\end{align*}$
Reverse Problem: Constructing Multiple Alignment from Pairwise Alignments

Given 3 arbitrary pairwise alignments:

$x$: ACCTGG-C;  $x$: AC-GCTGG-C;  $y$: AC-GC-GAG
$y$: ACCC--GAC;  $z$: GCCGCA-GAG;  $z$: GCCGCAGAG

can we construct a multiple alignment that induces them?

Inferring 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

Combining Optimal Pairwise Alignments into Multiple Alignment

Can combine pairwise alignments into multiple alignment

Can not combine pairwise alignments into multiple alignment

Profile Representation of Multiple Alignment

Earlier, we were aligning a sequence against a sequence
Can we align a sequence against a profile?
Can we align a profile against a profile?
Aligning alignments

- Given two alignments, can we align them?

\[
\begin{align*}
\text{x} & \text{ GGGCACTGCAT} \\
\text{y} & \text{ GGTTACGTC--} \quad \text{Alignment 1} \\
\text{z} & \text{ GGGAACTGCAG} \\
\text{w} & \text{ GGACGTACC--} \quad \text{Alignment 2} \\
\text{v} & \text{ GGACCT-----}
\end{align*}
\]

- Hint: use alignment of corresponding profiles

\[
\begin{align*}
\text{x} & \text{ GGGCACTGCAT} \\
\text{y} & \text{ GGTTACGTC--} \quad \text{Combined Alignment} \\
\text{z} & \text{ GGGAACTGCAG} \\
\text{w} & \text{ GGACGTACC--} \\
\text{v} & \text{ GGACCT-----}
\end{align*}
\]

Multiple Alignment: Greedy Approach

- Choose most similar pair of strings and combine into a profile, thereby reducing alignment of \( k \) sequences to an alignment of \( k-1 \) sequences/profiles. Repeat

- This is a heuristic greedy method

\[
\begin{align*}
\text{u}_1 & = \text{ACGTACGTACGT...} \\
\text{u}_2 & = \text{TTAATTAATTAA...} \\
\text{u}_3 & = \text{GCGCGCGCGCG...} \\
\vdots & \quad \quad \quad \quad \quad \quad \quad \quad \vdots \\
\text{u}_k & = \text{CGCGCGCGCGCG...}
\end{align*}
\]

Greedy Approach: Example

- Consider these 4 sequences

\[
\begin{align*}
\text{s}_1 & \quad \text{GATTCA} \\
\text{s}_2 & \quad \text{GTCTGA} \\
\text{s}_3 & \quad \text{GATATT} \\
\text{s}_4 & \quad \text{GTCAGC}
\end{align*}
\]

- There are \( \binom{4}{2} = 6 \) possible alignments

\[
\begin{align*}
\text{s}_2 & \quad \text{GCTCTG} \\
\text{s}_4 & \quad \text{GCTAGC \ (score = 2)} \\
\text{s}_1 & \quad \text{G-ATTCA--} \\
\text{s}_4 & \quad \text{G-T-CAGC \ (score = 0)} \\
\text{s}_1 & \quad \text{GAT-TCA} \quad \text{(score = 1)} \\
\text{s}_2 & \quad \text{G-TCTGA} \quad \text{(score = -1)} \\
\text{s}_1 & \quad \text{GAT-TCA} \\
\text{s}_3 & \quad \text{GATATT \ (score = 1)} \\
\text{s}_4 & \quad \text{GTCAGC \ (score = -1)}
\end{align*}
\]

- \( s_2 \) and \( s_4 \) are closest; combine:

\[
\begin{align*}
\text{s}_2 & \quad \text{GTCTGA} \\
\text{s}_4 & \quad \text{GTCAGC \ (profile)} \\
\text{s}_{2,4} & \quad \text{GTCt/aGa/cA}
\end{align*}
\]

new set of 3 sequences:

\[
\begin{align*}
\text{s}_1 & \quad \text{GATTCA} \\
\text{s}_2 & \quad \text{GATATT} \\
\text{s}_{2,4} & \quad \text{GTCl/aGa/c}
\end{align*}
\]
Progressive Alignment

- **Progressive alignment** is a variation of greedy 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.
  - Gaps in consensus string are permanent.
  - Use profiles to compare sequences.

Star alignment

- Heuristic method for multiple sequence alignments.
- Select a sequence $c$ as the center of the star.
- For each sequence $x_i, \ldots, x_k$ such that index $i \neq c$, perform a Needleman-Wunsch global alignment.
- Aggregate alignments with the principle "once a gap, always a gap."

Choosing a center

- Try them all and pick the one which is most similar to all of the sequences.
- Let $S(x_i, x_j)$ be the optimal score between sequences $x_i$ and $x_j$.
- Calculate all $O(k^2)$ alignments, and choose as $x_c$ the sequence $x_i$ that maximizes the following:
  $$\sum_{j \neq i} S(x_i, x_j)$$

Star alignment example

- Assuming all sequences have length $n$.
- $O(k^2n^2)$ to calculate center.
- Step $i$ of iterative pairwise alignment takes $O((i \cdot n) \cdot n)$ time.
  - two strings of length $n$ and $i \cdot n$.
  - $O(k^2n^2)$ overall cost.

ClustalW

- Popular multiple alignment tool today.
- 'W' stands for 'weighted' (different parts of alignment are weighted differently).
- Three-step process:
  1.) Construct pairwise alignments.
  2.) Build Guide Tree (by Neighbor Joining method).
  3.) Progressive Alignment guided by the tree.
    - The sequences are aligned progressively according to the branching order in the guide tree.
Step 1: Pairwise Alignment

- Aligns each sequence against each other giving a similarity matrix
- Similarity = exact matches / sequence length (percent identity)

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

(.17 means 17% identical)

Step 2: Guide Tree

- Create Guide Tree using the similarity matrix
- ClustalW uses the neighbor-joining method
- Guide tree roughly reflects evolutionary relations

Step 2: Guide Tree (cont’d)

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

Calculate:

\[
\begin{align*}
& v_{1,3} = \text{alignment } (v_1, v_3) \\
& v_{1,3,4} = \text{alignment } ((v_1, v_3), v_4) \\
& v_{1,2,3,4} = \text{alignment } ((v_1, v_3, v_4), v_2)
\end{align*}
\]

Step 3: Progressive Alignment

- 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

ClustalW: another example

<table>
<thead>
<tr>
<th>S_1</th>
<th>ALSK</th>
<th>TNSD</th>
<th>NASK</th>
<th>NTSD</th>
</tr>
</thead>
</table>

ClustalW example

<table>
<thead>
<tr>
<th>S_1</th>
<th>S_2</th>
<th>S_3</th>
<th>S_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Distance Matrix
### ClustalW example

<table>
<thead>
<tr>
<th></th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
<th>S₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>0</td>
<td>9</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>S₂</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>S₃</td>
<td>0</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S₄</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Distance Matrix**

**Neighbor Joining**

**Rooted Tree**

#### Multiple Alignment Steps
1. Align S₁ with S₃
2. Align S₂ with S₄
3. Align (S₁, S₃) with (S₂, S₄)

### Other progressive approaches

- **PILEUP**
  - Similar to CLUSTALW
  - Uses UPGMA to produce tree

### Problems with progressive alignments

- Depend on pairwise alignments
- If sequences are very distantly related, much higher likelihood of errors
- Care must be made in choosing scoring matrices and penalties
Iterative refinement in progressive alignment

Another problem of progressive alignment:
- Initial alignments are "frozen" even when new evidence comes

Example:

- **x**: GAAGTT
- **y**: GAC-TT  \(\rightarrow\) Frozen!
- **z**: GACTG
- **w**: GTACTG  \(\rightarrow\) Now clear that correct **y** = GA-CTT

Evaluating multiple alignments

- Balibase benchmark (Thompson, 1999)
- De-facto standard for assessing the quality of a multiple alignment tool
- Manually refined multiple sequence alignments
- Quality measured by how good it matches the core blocks
- Another benchmark: SABmark benchmark
  - Based on protein structural families

Scoring multiple alignments

- Ideally, a scoring scheme should
  - Penalize variations in conserved positions higher
  - Relate sequences by a phylogenetic tree
    - Tree alignment
  - Usually assume
    - Independence of columns
    - Quality computation
      - Entropy-based scoring
        - Compute the Shannon entropy of each column
      - Sum-of-pairs (SP) score

Multiple Alignments: Scoring

- Number of matches (multiple longest common subsequence score)
- Entropy score
- Sum of pairs (SP-Score)

Multiple LCS Score

- A column is a "match" if all the letters in the column are the same
  - AAA
  - AAT
  - ATC
- Only good for very similar sequences

Entropy

- Define frequencies for the occurrence of each letter in each column of multiple alignment
  - \(p_A = 1, p_T = p_G = p_C = 0\) (1st column)
  - \(p_A = 0.75, p_T = 0.25, p_G = p_C = 0\) (2nd column)
  - \(p_A = 0.50, p_T = 0.25, p_C = 0.25, p_G = 0\) (3rd column)
- Compute entropy of each column
  \[
  - \sum_{x = A, T, C} \frac{p_x}{\log p_x} \quad \text{for } x \neq A, T, C
  \]

Entropy: Example

\[
\text{Best case: } \text{entropy } \begin{pmatrix} A \\ A \\ A \end{pmatrix} = 0
\]

\[
\text{Worst case: } \text{entropy } \begin{pmatrix} A \\ T \\ G \\ C \end{pmatrix} = - \sum \frac{1}{4} \log \frac{1}{4} - 4 \left( \frac{1}{4} \right) = 2
\]

Multiple Alignment: Entropy Score

Entropy for a multiple alignment is the sum of entropies of its columns:

\[
\sum \text{ over all columns } - \sum_{X=A,T,G,C} p_X \log p_X
\]

Entropy of an Alignment: Example

\[
\text{column entropy: } -(p_A \log p_A + p_C \log p_C + p_G \log p_G + p_T \log p_T)
\]

<table>
<thead>
<tr>
<th>Column</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>A A A</td>
<td>0</td>
</tr>
<tr>
<td>A C C</td>
<td>-0.811</td>
</tr>
<tr>
<td>A C G</td>
<td>+2.0</td>
</tr>
<tr>
<td>A C T</td>
<td>+2.811</td>
</tr>
</tbody>
</table>

Alignment Entropy = 0 + (−0.811) + 2.0 = +2.811

Multiple Alignment Induces Pairwise Alignments

Every multiple alignment induces pairwise alignments

\[
x: \text{AC-GCGG-C} \\
y: \text{AC-GCGG-C} \\
z: \text{GCCGC-GAG}
\]

Induces:

\[
x: \text{ACCGG-C}; \quad x: \text{AC-GCGG-C}; \quad y: \text{AC-GCGAG} \\
y: \text{ACGAC-GAC}; \quad z: \text{GCCGC-GAG}; \quad z: \text{GCCGCGAG}
\]

Sum of Pairs (SP) Scoring

- SP scoring is the standard method for scoring multiple sequence alignments.
- Columns are scored by a ‘sum of pairs’ function using a substitution matrix (PAM or BLOSUM)
- Assumes statistical independence for the columns, does not use a phylogenetic tree.

Sum of Pairs Score (SP-Score)

- Consider pairwise alignment of sequences \(a_i\) and \(a_j\) imposed by a multiple alignment of \(k\) sequences
- Denote the score of this suboptimal (not necessarily optimal) pairwise alignment as \(s^*(a_i, a_j)\)
- Sum up the pairwise scores for a multiple alignment:

\[
s(a_1, \ldots, a_k) = \sum_{ij} s^*(a_i, a_j)
\]
Computing SP-Score

Aligning 4 sequences: 6 pairwise alignments

Given \( a_1, a_2, a_3, a_4 \):
\[
s(a_1 \ldots a_4) = \sum s^*(a_i, a_j) = s^*(a_1, a_2) + s^*(a_1, a_3)\]
\[+ s^*(a_1, a_4) + s^*(a_2, a_3) + s^*(a_2, a_4) + s^*(a_3, a_4)\]

SP-Score: Example

\[
s(a_1 \ldots a_4) = \sum_{i<j}^n s^*(a_i, a_j) = \binom{n}{2}\text{ Pairs of Sequences}\]

May also calculate the scores column by column:

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Score</th>
<th>Column 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>1</td>
<td>Score = 3</td>
<td>Score = 1 - 2\mu</td>
</tr>
</tbody>
</table>

Multiple alignment tools

- Clustal W (Thompson, 1994)
- Most popular
- PRRP (Gotoh, 1993)
- HMMT (Eddy, 1995)
- DIALIGN (Morgenstern, 1998)
- T-Coffee (Notredame, 2000)
- MUSCLE (Edgar, 2004)
- Align-m (Walle, 2004)
- PROBCONS (Do, 2004)

Useful links

http://cnx.org/content/m11036/latest/
http://www.biokemi.uu.se/Utbildning/Exercises/ClustalX/index.shtml
http://bioinformatics.weizmann.ac.il/~pietro/Making_and_using_protein_MA/
http://homepage.usask.ca/~ctl271/857/paper1_overview.shtml