Multiple Sequence Alignment (MSA)

Yuzhen Ye (yye@indiana.edu)
School of Informatics & Computing, IUB

Outline

- Multiple sequence alignment (MSA)
- Generalize DP to 3 sequence alignment
  - Impractical
- Heuristic approaches to MSA
  - Progressive alignment – ClustalW (using substitution matrix based scoring function)
  - Consistency-based approach – T-Coffee (consistency-based scoring function)
  - MUSCLE (MUSCLE-fast, MUSCLE-prog): reduces time and space complexity

From pairwise to multiple 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

What’s multiple sequence alignment (MSA)

- A model
- Indicates relationship between residues of different sequences
- Reveals similarity/disimilarity

Why we need MSA

- MSA is central to many bioinformatics applications
- Phylogenetic tree
- Motifs
- Patterns
- Structure prediction (RNA, protein)

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
2D vs 3D alignment grid

2D vs 3D alignment grid

2D table

3D graph

DP recursion (3 edges vs 7)

Pairwise: 3 possible paths (match/mismatch, insertion, and deletion)

In 3-D, 7 edges in each unit cube

Architecture of 3D alignment cell

Multiple alignment: dynamic programming

\[ s_{i,j,k} = \max \begin{cases} s_{i-1,j-1,k-1} + \delta(v_i, w_j, u_k) \\ s_{i-1,j-1,k} + \delta(v_i, w_j, -) \\ s_{i-1,j,k-1} + \delta(-, w_j, u_k) \\ s_{i,j-1,k-1} + \delta(-, -, u_k) \\ s_{i-1,j,k} + \delta(v_i, -, u_k) \\ s_{i,j-1,k} + \delta(-, w_j, -) \end{cases} \]

\( \delta(x, y, z) \) is an entry in the 3D scoring matrix

Multiple alignment: dynamic programming

- Cube diagonal: no indels
- Face diagonal: one indel
- Edge diagonal: two indels

MSA: 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 \( (2^k-1)n^k \); \( O(2^k n^k) \)
- Conclusion: dynamic programming approach for alignment between two sequences is easily extended to \( k \) sequences (simultaneous approach) but it is impractical due to exponential running time.
- Computing exact MSA is computationally almost impossible, and in practice heuristics are used (progressive alignment)

Profile representation of multiple alignment

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>A</th>
<th>C</th>
<th>C</th>
<th>T</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>-</td>
<td>G</td>
<td>C</td>
<td>T</td>
<td>A</td>
<td>T</td>
<td>C</td>
<td>A</td>
<td>C</td>
<td>C</td>
<td>T</td>
<td>G</td>
</tr>
<tr>
<td>C</td>
<td>G</td>
<td>-</td>
<td>G</td>
<td>-</td>
<td>-</td>
<td>C</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>-</td>
<td>-</td>
<td>G</td>
</tr>
<tr>
<td>G</td>
<td>C</td>
<td>G</td>
<td>-</td>
<td>G</td>
<td>-</td>
<td>C</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>-</td>
<td>-</td>
<td>G</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>G</td>
<td>-</td>
<td>C</td>
<td>A</td>
<td>T</td>
<td>C</td>
<td>A</td>
<td>C</td>
<td>-</td>
<td>G</td>
<td>G</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>A</th>
<th>C</th>
<th>C</th>
<th>T</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>1</td>
<td>.8</td>
<td>1</td>
<td>.8</td>
<td>1</td>
<td>.8</td>
<td>1</td>
<td>.8</td>
<td>1</td>
<td>.8</td>
<td>1</td>
<td>.8</td>
</tr>
<tr>
<td>C</td>
<td>.6</td>
<td>1</td>
<td>.6</td>
<td>1</td>
<td>.6</td>
<td>1</td>
<td>.6</td>
<td>1</td>
<td>.6</td>
<td>1</td>
<td>.6</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>1</td>
<td>.6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Profile representation of multiple alignment

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>A</th>
<th>C</th>
<th>C</th>
<th>T</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>1</td>
<td>.8</td>
<td>1</td>
<td>.8</td>
<td>1</td>
<td>.8</td>
<td>1</td>
<td>.8</td>
<td>1</td>
<td>.8</td>
<td>1</td>
<td>.8</td>
</tr>
<tr>
<td>C</td>
<td>.6</td>
<td>1</td>
<td>.6</td>
<td>1</td>
<td>.6</td>
<td>1</td>
<td>.6</td>
<td>1</td>
<td>.6</td>
<td>1</td>
<td>.6</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>1</td>
<td>.6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Aligning alignments/profiles

- Given two alignments, can we align them?

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

Progressive alignment

- *Progressive alignment* uses guide tree
- Sequence weighting & scoring scheme and gap penalties
- Progressive alignment works well for close sequences, but deteriorates for distant sequences
  - Gaps in consensus string are permanent
  - Use profiles to compare sequences

ClustalW

- Popular multiple alignment tool today
- 'W' stands for 'weighted' (sequences are weighted differently).
- Three-step process
  1.) Construct pairwise alignments
  2.) Build guide tree
  3.) Progressive alignment guided by the tree

ClustalW -- a diagram

Dynamic Programming Using A Substitution Matrix

Step 1: Pairwise alignment

- Aligns each sequence against each other giving a similarity matrix
- Similarity = exact matches / sequence length (percent identity)
  \[
  \begin{bmatrix}
  v_1 & v_2 & v_3 & v_4 \\
  v_1 & .17 & - & - \\
  v_2 & .28 & - & - \\
  v_3 & .59 & .33 & .62 & - \\
  \end{bmatrix}
  \]
  (.17 means 17% identical)
Step 2: Guide tree

![Guide tree diagram]

Calculate:

- \( v_{1,3} = \text{alignment}(v_1, v_3) \)
- \( v_{1,3,4} = \text{alignment}((v_{1,3}), v_4) \)
- \( v_{1,2,3,4} = \text{alignment}(((v_{1,3}), v_4), v_2) \)

ClustalW uses NJ to build guide tree;

Guide tree roughly reflects evolutionary relations

Step 3: Tree based recursion

![Tree based recursion diagram]

```
Align (Node N) {
  if (N->left_child is a Node)
    A1 = Align (N->left_child)
  else if (N->left_child is a Sequence)
    A1 = N->left_child
  if (N->right_child is a node)
    A2 = Align (N->right_child)
  else if (N->right_child is a Sequence)
    A2 = N->right_child
  Return dp_alignment(A1, A2)
}
```

Progressive alignment: Scoring scheme

- Scoring scheme is arguably the most influential component of the progressive algorithm
- Matrix-based algorithms
  - ClustalW, MUSCLE, Kalign
  - Use a substitution matrix to assess the cost of matching two symbols or two profiled columns
  - Once a gap, always a gap
- Consistency-based schemes
  - T-Coffee, Dialign
  - Compile a collection of pairwise global and local alignments (primary library) and to use this collection as a position-specific substitution matrix

Substitution matrix based scoring

- Sum of pairs (SP score)
- Tree based scoring
- Entropy score

Sum of pairs score (SP score)

```
<table>
<thead>
<tr>
<th>Seq</th>
<th>Column A-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N-N-N-N-N</td>
</tr>
<tr>
<td>2</td>
<td>N-N-N-N-N</td>
</tr>
<tr>
<td>3</td>
<td>N-N-N-N-N</td>
</tr>
<tr>
<td>4</td>
<td>N-N-N-N-N</td>
</tr>
<tr>
<td>5</td>
<td>N-N-N-N-N</td>
</tr>
</tbody>
</table>
```

Score:

- \( 10 \times S(N,N) = 60 \)
- \( 4 \times S(N,N) + 6 \times S(N,C) = 6 \times 6 + 6 = 66 \) (BLOSUM62)

Problem: over-estimation of the mutation costs (assuming each sequence is the ancestor of itself; requires a weighting scheme

Tree-based scoring

```
```

“Real” tree:
Cost = 1
But we do not know the tree!

Star tree
Cost = 2
But the tree is wrong!
Entropy-based scoring

**Column entropy:**

\[-(p_A \log p_A + p_C \log p_C + p_G \log p_G + p_T \log p_T)\]

- Column 1 = \(-[1 \cdot \log(1) + 0 \cdot \log(0) + 0 \cdot \log(0) + 0 \cdot \log(0)]\)
  = 0

- Column 2 = \(-[(1/4) \cdot \log(1/4) + (3/4) \cdot \log(3/4) + 0 \cdot \log(0) + 0 \cdot \log(0)]\)
  = \(-[(1/4) \cdot (-2) + (3/4) \cdot (-0.415)]\)
  = +0.811

- Column 3 = \(-[(1/4) \cdot \log(1/4) + (1/4) \cdot \log(1/4) + (1/4) \cdot \log(1/4) + (1/4) \cdot \log(1/4)]\)
  = +2.0

- Alignment Entropy = 0 + 0.811 + 2.0 = +2.811

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)
  - \(p_A = 0.50, p_T = 0.25, p_G = 0.25, p_C = 0\) (3rd column)

- Compute entropy of each column

\[ - \sum_{i=0,A,C} p_i \log p_i \]

Consistency-based approaches

- T-Coffee
  - M-Coffee & 3D-Coffee (Expresso)
- Principle
  - Primary library
  - Library extension

### T-Coffee: Primary library

Input sequences:

- SeqA: GARFIELD THE LAST FAT CAT
- SeqB: GARFIELD THE FAST CAT
- SeqC: GARFIELD THE VERY FAST CAT
- SeqD: THE FAT CAT

Primary library: collection of global/local pairwise alignments

- SeqA GARFIELD THE LAST FAT CAT
- SeqB GARFIELD THE FAST CAT
- SeqC: GARFIELD THE VERY FAST CAT
- SeqD: THE FAT CAT

### T-Coffee: Library extension

Extended library: new pairwise alignments

- SeqA GARFIELD THE LAST FAT CAT
- SeqB GARFIELD THE FAST CAT
- SeqC: GARFIELD THE VERY FAST CAT
- SeqD: THE FAT CAT

Different “weights”
T-Coffee uses progressive strategy to derive multiple alignment

- Guide tree
- First align the closest two sequences (DP using the weights derived from the extended library)
- Align two "alignments" (using the weights from the extended library – average over each column)
- No additional parameters (gaps etc)
  - The substitution values (weights) are derived from extended library which already considered gaps
  - High scoring segments (consistent segments) enhanced by the data set to the point that they are insensitive to the gap penalties

MUSCLE: a tool for fast MSA

- Initial progressive alignment followed by horizontal refinement (stochastic search for a maximum objective score)
  - Step 1: draft progressive (using k-mer counting for fast computation of pairwise distance; tree building using UPGMA or NJ)
  - Step 2: Improved progressive to improve the tree and builds a new progressive alignment according to this tree (can be iterated).
  - Step 3: Refinement using tree-dependent restricted partitioning (each edge is deleted from the tree to divide the sequences into two disjoint subsets, from each a profile is built; the profile-profile alignment is computed, and if the score improves, retain the new alignment).

Multiple alignment: History

- 1975 Sankoff: Formulated multiple alignment problem and gave DP solution
- 1988 Carrillo-Lipman: Branch and Bound approach for MSA
- 1990 Feng-Doolittle: Progressive alignment
- 1994 Thompson-Higgins-Gibson-ClustalW: Most popular multiple alignment program
- 2000 T-coffee (consensus-based)
- 2004 MUSCLE
- 2005 ProbCons (uses Bayesian consistency)
- 2006 M-Coffee (consensus meta-approach)
- 2006 Expresso (3D-Coffee; use structural template)
- 2007 PROMALS (profile-profile alignment)

Summary & References

- "A majority of studies indicate that consistency-based methods are more accurate than their matrix-based counterparts, although they typically require an amount of CPU time N times higher than simpler methods (N being the number of sequences)"
- http://tcoffee.vital-it.ch/cgi-bin/Tcoffee/tcoffee_cgi/index.cgi (try it!!)
- Cedric Notredame. Recent evolutions of multiple sequence alignment algorithms. 2007, 3(8):e123