Multiple Sequence Alignment

CMSC 423
Multiple sequence alignment: find more subtle patterns & find common patterns between all sequence.
Input: Sequences: $S_1, S_2, ..., S_m$

Let $M$ be a MSA between these sequences.

Let $d_M(S_i, S_j)$ be the score of the alignment between $S_i$ and $S_j$ implied by $M$.

$$SP-Score(M) = \sum_{i,j} d_M(S_i, S_j) = \text{Sum of all pairwise alignment scores.}$$

- **Goal**: find $M$ to minimize $SP-Score(M)$.
- But this is NP-hard.
SP-Score in a Picture

\[ \text{SP-Score}(M) = \sum_{i,j} d_M(S_i, S_j) \]

= sum of all the scores of the pairwise alignments implied by M.
A multiple sequence alignment (MSA) implies a pairwise alignment between every pair of sequences. This implied alignment need not be optimal, however:

match = -1, a mismatch = 1, gap = 2
Sequences: AT, A, T, AT, AT

Optimal MSA:

\[
\begin{align*}
&A\quad T \\
&A- \\
&-T \\
&A \\
&AT \\
&AT \\
&AT
\end{align*}
\]

Optimal Alignment between A and T:

\[
\begin{align*}
&A \\
&T \\
&+1
\end{align*}
\]


\[-1 + 2 -1 -1 +2 -1 -1 +2 +2 -1 = +2\]
Slow Dynamic Programming

Suppose you had just 3 sequences.

Apply the same DP idea as sequence alignment for 2 sequences, but now with a 3-dimensional matrix
DP Recurrence for 3 sequences

\[
A[i, j, k] = \min \begin{cases} 
\text{cost}(x_i, y_j, z_k) + A[i - 1, j - 1, k - 1] \\
\text{cost}(x_i, -, -) + A[i - 1, j, k] \\
\text{cost}(x_i, y_j, -) + A[i - 1, j - 1, k] \\
\text{cost}(-, y_j, z_k) + A[i, j - 1, k - 1] \\
\text{cost}(-, y_j, -) + A[i, j - 1, k] \\
\text{cost}(x_i, -, z_k) + A[i - 1, j, k - 1] \\
\text{cost}(-, -, z_k) + A[i, j, k - 1] \\
\end{cases}
\]

Every possible \(2^k\) pattern for the gaps.
Running time

• $n^3$ subproblems, each takes $2^3$ time $\Rightarrow O(n^3)$ time.

• For $k$ sequences: $n^k$ subproblems, each takes $2^k$ time for the max and $k^2$ to compute cost() $\Rightarrow O(k^2n^k2^k)$

• Even $O(n^3)$ is often too slow for the length of sequences encountered in practice.

• One solution: approximation algorithm.
Star Alignment Approximation

SP-Score

Star-Score = \sum_i d_M(S_i, S_c)

S_i

S_j

d_M(S_i, S_j)

S_c

S_i

d_M(S_i, S_c)

Star-Score =

\sum_i d_M(S_i, S_c)
Star Alignment Algorithm

**Input**: sequences $S_1, S_2, ..., S_k$

- Build all $O(k^2)$ pairwise alignments.
- Let $S_c$ = the sequence in $S_1, S_2, ..., S_k$ that is closest to the others. That is, choose $S_c$ to minimize:
  $$\sum_{i \neq c} d(S_c, S_i)$$
- *Iteratively align* all other sequences to $S_c$. 
Iterative Alignment

- Build a multiple sequence alignment up from pairwise alignments.

Start with an alignment between $S_c$ and some other sequence:

\[
\begin{align*}
S_c & \quad \text{YFPHFDL}SHGSAQVKAHGKKVGDALT\text{LAVGHL}DDLP\text{GAL} \\
S_1 & \quad \text{YFPHFDL}SHG-AQVKG--KKVADALTNAVAHVDDMPNAL
\end{align*}
\]

Add 3rd sequence, say $S_2$, and use the $S_c - S_2$ alignment as a guide, adding spaces into the MSA as needed.

**SC - S2 alignment:**

\[
\begin{align*}
S_c & \quad \text{YFPHF-DL}S-----HGSAQVKAH\text{GKKVGDALT}\text{LAVGHL}-----\text{DDLP}\text{GAL} \\
S_2 & \quad \text{FFPKFKGLTTADQLKKSADV}R\text{WH}AER\text{II}-----\text{NAVND}AVASMD\text{D}T\text{EKMS}
\end{align*}
\]

New $\{S_c, S_1, S_2\}$ alignment (red gaps added in $S_1$):

\[
\begin{align*}
S_c & \quad \text{YFPHF-DL}S-----HGSAQVKAH\text{GKKVGDALT}\text{LAVGHL}-----\text{DDLP}\text{GAL} \\
S_1 & \quad \text{YFPHF-DL}S-----HG-AQVKG--KKVADALTNAVAHV-----DDMPNAL \\
S_2 & \quad \text{FFPKFKGLTTADQLKKSADV}R\text{WH}AER\text{II}-----\text{NAVND}AVASMD\text{D}T\text{EKMS}
\end{align*}
\]

Continue with $S_3, S_4, ...$
Performance

Assume the cost function satisfies the triangle inequality:

\[ \text{cost}(x,y) \leq \text{cost}(x, z) + \text{cost}(z,y) \]

Example: \[ \text{cost}(A, C) \leq \text{cost}(A, T) + \text{cost}(T,C) \]

- cost of 1 mutation from \( A \rightarrow C \)
- cost of a mutation from \( A \rightarrow T \) and then from \( T \rightarrow C \)

\text{STAR} = \text{cost of star alignment under SP-score}
\text{OPT} = \text{cost of optimal multiple sequence alignment (under SP-score)}

**Theorem.** If cost satisfies the triangle inequality, then \( \text{STAR} \leq 2 \times \text{OPT} \).

Example: if optimal alignment has cost 10, the star alignment will have cost \( \leq 20 \).
Proof (1)

**Theorem.** If cost satisfies the triangle inequality, then \( \text{STAR} \leq 2\text{OPT} \).

\[
\frac{\text{STAR}}{\text{OPT}} \leq 2
\]

For some \( B \) we will prove the 2 statements:

\[
\begin{align*}
\text{STAR} & \leq 2B \\
\text{OPT} & \geq B
\end{align*}
\]

This will imply:

\[
\Rightarrow \quad \frac{\text{STAR}}{\text{OPT}} \leq \frac{2B}{B} = 2
\]
**Proof (2)**

**Theorem.** If cost satisfies the triangle inequality, then \( \text{STAR} \leq 2\text{OPT} \).

\[ 2 \cdot \text{STAR} = \sum_{ij} d_{\text{STAR}}(S_i, S_j) \text{ defn of SP-score} \]

by triangle inequality

\[ \leq \sum_{ij}(d_{\text{STAR}}(S_i, S_c) + d_{\text{STAR}}(S_c, S_j)) \]

because \( \text{STAR} \) alignment is optimal for pairs involving \( S_c \)

distribute \( \sum \)

\[ = \sum_{ij}(d(S_i, S_c) + d(S_c, S_j)) \]

\[ = \sum_{ij} d(S_i, S_c) + \sum_{ij} d(S_c, S_j) \]

\[ \leq 2k \sum_i d(S_i, S_c) \quad \text{sums are the same and each term appears} \]

\[ \leq k \text{ (# of sequences) times}. \]
Theorem. If cost satisfies the triangle inequality, then $\text{STAR} \leq 2 \cdot \text{OPT}$.

$$2 \cdot \text{OPT} = \sum_{ij} d_{\text{OPT}}(S_i, S_j)$$

optimal pairwise alignment
is $\leq$ pairwise alignment
induced by any MSA

$$\geq \sum_{ij} d(S_i, S_j)$$

sum of all cost of all pairwise
alignments is $= \text{the sum of } k$
different stars.

We chose $S_c$ because it was
the lowest-cost star.
For some \( B \) we will prove the 2 statements:

\[
\begin{align*}
\text{STAR} & \leq 2B \\
\text{OPT} & \geq B
\end{align*}
\]

This will imply:

\[
\Rightarrow \quad \frac{\text{STAR}}{\text{OPT}} \leq \frac{2B}{B} = 2
\]

\[
\begin{align*}
2 \cdot \text{STAR} & \leq 2k \sum_i d(S_i, S_c) \\
2 \cdot \text{OPT} & \geq k \sum_i d(S_i, S_c)
\end{align*}
\]

\[
\Rightarrow \quad \frac{\text{STAR}}{\text{OPT}} \leq \frac{2k \sum_i d(S_i, S_c)}{k \sum_i d(S_i, S_c)} = 2
\]
Consensus Sequence

For every column j, choose $c \in \Sigma$ that minimizes $\sum_i \text{cost}(c, S_i[j])$.

(typically this means the most common letter)

S1 YFPHF-DLS------HGSAQVKAHGKKVG------DALTLAVAHLDLPGAL
S2 YFPHF-DLS------HG-AQVKG-GKKVA------DALTNAVAHVDDMPNAL
S3 FFPFKFKGLTTADQLKKSADVRWHAERII------NAVNDAVASMDDTEKMS
S4 LFSFLKGTSEVP--QNNPELQAHAHKVFKLVYEEAIQLQVTGVVVTDATL
CO YFPHFKDLS------HGSAQVKAHGKKVG------DALTLAVAHVDDTPGAL

- Consensus is a summarization of the whole alignment.
- Consensus sequence is sometimes used as an estimate for the ancestral sequence.
- Sometimes the MSA problem is formulated as: find MSA $M$ that minimizes:

$$\sum_i d_M(CO, S_i)$$
Profiles

- Another way to summarize an MSA:

  S1  ACG-TT-GA
  S2  ATC-GTCGA
  S3  ACGCGA-CC
  S4  ACGCGT-TA

<table>
<thead>
<tr>
<th>Character</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.25</td>
<td>0</td>
<td>0</td>
<td>0.75</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0.75</td>
<td>0.25</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0.75</td>
<td>0</td>
<td>0.75</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>0.25</td>
<td>0</td>
<td>0</td>
<td>0.25</td>
<td>0.75</td>
<td>0</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0.75</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Column in the alignment

Call this profile matrix \( R \)

Fraction of time given column had the given character
Profile-based Alignment

Score of matching character $x$ with column $j$ of the profile:

$$P(x, j) = \sum_{c \in \Sigma} \text{sim}(x, c) \times R[c, j]$$

sim($x$, $c$) = how similar character $x$ is to character $c$.

$$A[i, j] = \max \begin{cases} 
A[i - 1, j - 1] + P(x_i, j) & \text{align } x_i \text{ to column } j \\
A[i - 1, j] + \text{gap} & \text{introduce gap into profile} \\
A[i, j - 1] + P(\text{"-"}, j) & \text{introduce gap into } x
\end{cases}$$
Recap

- Multiple sequence alignments (MSAs) are a fundamental tool. They help reveal subtle patterns, compute consistent distances between sequences, etc.

- Quality of MSAs often measured using the SP-score: sum of the scores of the pairwise alignments implied by the MSA.

- Same DP idea as pairwise alignment leads to exponentially slow algorithm for MSA.

- 2-approximation obtainable via star alignments.

- MSAs often used to create profiles summarizing a family of sequences. Profile-sequence alignments solvable via dynamic programming.