Computational Biology

Lecture 6
General gap penalty function

• Using a general $\gamma$ gives an $O(mn^2+nm^2)$ algorithm.

• Can we still achieve our old $O(mn)$ time bound?
What was the problem?

• For each additional gap, we have a different additional score.

• We had to accommodate for every possible gap length.

• Restrict the possibilities. Use an affine gap function to approximate the general one.
Affine gap penalty function

\[ \gamma(x) = e + d(x - 1) \quad x \geq 1. \]
How can this help

• Any gap length greater than one is penalized linearly.

• Need to distinguish only between 1 and more than 1.

• If 1
  – penalize by e.

• If $x > 1$
  – penalize linearly by increments of $d$.

• Use more than one matrix to detect that, depends on where we stopped last time.
Looking for additivity

- Break the alignment into 3 kinds of blocks:
  - Two aligned characters
  - Consecutive gaps aligned with $x$
  - Consecutive gaps aligned with $y$

\[
\begin{array}{cccccccc}
A & C & T & ACC & T & -- & C & G & C & -- \\
\end{array}
\]

- Score is additive across block boundaries, this is true regardless of the gap penalty function.
Methodology

• Instead of reasoning on the last column in an alignment, we reason on the last block.

• For each pair \((i,j)\), keep the best score for \(x_1...x_i\) and \(y_1...y_j\) that end with a particular block.

• We need three matrices
  – \(A\) for block type 1
  – \(B\) for block type 2, gaps aligned with \(x\)
  – \(C\) for block type 3, gaps aligned with \(y\)
Modified Needleman-Wunsch for affine gaps

\[ A(i, j) = s(i, j) + \max \begin{cases} 
A(i-1, j-1) \\
B(i-1, j-1) \\
C(i-1, j-1) 
\end{cases} \]

\[ B(i, j) = \max \begin{cases} 
A(i-1, j) - e \\
B(i-1, j) - d \\
C(i-1, j) - e 
\end{cases} \]

\[ C(i, j) = \max \begin{cases} 
A(i, j-1) - e \\
B(i, j-1) - e \\
C(i, j-1) - d 
\end{cases} \]

\( A(i, j) \) is the start of block of type 1, so it aligns \( x_i \) with \( y_j \) no matter what.

\( B \) for block type 2, only need to vary \( i \). First gap penalized \(-e\), i.e. after block type 1 and 3, otherwise \(-d\).

\( C \) for block type 3, only need to vary \( j \). First gap penalized \(-e\), i.e. after block type 1 and 2, otherwise \(-d\).
Initialization

• $A(0,0) = 0$
• $A(i, 0) = ? \quad -\infty$ not block type 1
• $A(0, j) = ? \quad -\infty$

• $B(0,j) = \quad -\infty$ not block type 2
• $B(i,0) = ? \quad -e - d(i - 1) \quad 1 \leq i \leq m$

• $C(0,j) = ? \quad -e - d(j - 1) \quad 1 \leq j \leq n$
• $C(i,0)\! = \! ? \quad -\infty$ not block type 3
Simplification

\[ A(i, j) = s(i, j) + \max \begin{cases} A(i - 1, j - 1) \hfill \\
B(i - 1, j - 1) \hfill \\
C(i - 1, j - 1) \hfill \end{cases} \]

\[ B(i, j) = \max \begin{cases} A(i - 1, j) - e \hfill \\
B(i - 1, j) - d \hfill \\
C(i - 1, j) - e \hfill \end{cases} \]

\[ C(i, j) = \max \begin{cases} A(i, j - 1) - e \hfill \\
B(i, j - 1) - e \hfill \\
C(i, j - 1) - d \hfill \end{cases} \]

works if \( s \leq e + d \)
Multiple sequence alignment

• Align $k$ sequences in the best way

• What is the score (assume additive)?
  – In each column, we have $k$ characters.
  – Scoring function takes $k$ arguments.
  – Need $O(2^k)$ entries (at least gap/non-gap)!

• Practical scoring function: Sum of Pairs (SP-score)
  – Let $\text{score}_{ij}$ be the score of the induced alignment for sequences $x_i$ and $x_j$, i.e. the score of the alignment obtained by isolating $x_i$ and $x_j$ and ignoring columns with only gaps.
  – $\text{SP-score} = \Sigma_{i<j} \text{score}_{ij}$
SP-score function

• Example: three sequences $x$, $y$, and $z$
  – $SP = \text{score}(x, y) + \text{score}(x, z) + \text{score}(y, z)$

• Nice properties
  – This is independent of the order of characters in a column
  – It rewards similarities and penalizes differences.

• Assume additive
  – $\text{score}(x, y) = \sum_i s(x_i, y_i), \quad s(\cdot, \cdot) = 0$
Example

- \( w = ATG, \ x = ATG, \ y = A, \ z = T \)

\[
\begin{array}{c}
ATG \\
ATG \\
ATG \\
A-- \\
- T--
\end{array}
\begin{array}{c}
A-- \\
- T--
\end{array}
\]

- Score = 3 -3 -3 -3 -3 -4 = -13

- The *induced* alignment between a pair of sequences is not necessarily an optimal one, e.g. \( y \) and \( z \).
Dynamic Programming

- $k$ sequences of length $n_i$ each.

- $k$ dimensional array $A$ of length $n_i+1$ in each direction.

- $A(i_1, \ldots, i_k)$ holds the score of the optimal alignment involving $x_1[1\ldots i_1], \ldots, x_k[1\ldots i_k]$.

- $A$ now requires $O(n^k)$ space.
Dynamic Programming (cont.)

- Now each entry in $A$ depends on $2^k - 1$ entries
  (note: $k = 2 \Rightarrow 3$) (why?)
  - e.g. $A(i_1, i_2, i_3)$ depends on:

\[
\begin{align*}
A(i_1, i_2, i_3 - 1) & : 001 \\
A(i_1, i_2 - 1, i_3) & : 010 \\
A(i_1, i_2 - 1, i_3 - 1) & : 011 \\
A(i_1 - 1, i_2, i_3) & : 100 \\
A(i_1 - 1, i_2, i_3 - 1) & : 101 \\
A(i_1 - 1, i_2 - 1, i_3) & : 110 \\
A(i_1 - 1, i_2 - 1, i_3 - 1) & : 111 
\end{align*}
\]

- Computing the SP-score in each case requires $O(k^2)$ time.

- Total running time is $O(k^2 2^k n^k)$
Heuristic: Star Alignment

• Star alignment is a special case of tree alignments.

• What is a tree alignment?

• Given a tree with \( k \) nodes representing \( k \) sequences, a multiple alignment of the \( k \) sequences consistent with the tree is such that the induced alignment between \( x_i \) and \( x_j \) is optimal if there is an edge \((x_i, x_j)\).
Example

A multiple alignment consistent with the tree

\[
\begin{align*}
s_1 &= AXZ \\
s_2 &= AXZ \\
s_3 &= AXXZ \\
s_4 &= AYZ \\
s_5 &= AYXYZ
\end{align*}
\]

\[
\begin{align*}
s_1 &= A \quad X \quad - \quad - \quad Z \\
s_2 &= A \quad - \quad X \quad - \quad Z \\
s_3 &= A \quad X \quad X \quad - \quad Z \\
s_4 &= A \quad Y \quad - \quad - \quad Z \\
s_5 &= A \quad y \quad X \quad X \quad Z
\end{align*}
\]
Tree alignment

• Given a tree, is it always possible to obtain a multiple alignment consistent with the tree?

  – YES

• How?
Algorithm: step 1

- Pick \( x_i \) and \( x_j \) such that \( (x_i, x_j) \) is an edge and align them optimally.

- set \( X = \{x_i, x_j\} \), the set of aligned sequences.
Algorithm: step 2

- Pick $x_k \notin X$ and $x_i \in X$ such that $(x_k, x_i)$ is an edge.

- Align $x_k$ and $x_i$ optimally.

- **Once a gap always a gap**: For each gap added to $x_i$ in this alignment, add a corresponding gap to sequences in $X$. For each gap already in $x_i$, add a corresponding gap in $x_k$ (if needed).

- $X = X \cup \{x_k\}$
Algorithm: step 3

– Repeat step 2 until all sequences are in $X$. 
Example revisited

- \( s_1 = AXZ \)
- \( s_2 = AXZ \)
- \( s_3 = AXXZ \)
- \( s_4 = AYZ \)
- \( s_5 = AYXYZ \)

\[
\begin{align*}
\text{Join} & \quad (s_1, s_3) \\
AX-Z & \quad A-XXZ \\
AXXZ & \quad AXXZ \\
\end{align*}
\]

\[
\begin{align*}
\text{Join} & \quad (s_2, s_3) \\
A-XZ & \quad AX-Z \\
AXXXZ & \quad A-XXZ \\
\end{align*}
\]

\[
\begin{align*}
\text{Join} & \quad (s_3, s_4) \\
AXXZ & \quad AXXZ \\
AY-Z & \quad AX-Z \\
\end{align*}
\]

\[
\begin{align*}
\text{Join} & \quad (s_4, s_5) \\
AY--Z & \quad AXX-Z \\
AYXXZ & \quad AX--Z \\
\end{align*}
\]

\[
\begin{align*}
\text{Join} & \quad (s_5, s_4) \\
AYXXZ & \quad AXX-Z \\
A--Z & \quad AX-XZ \\
\end{align*}
\]
Star Alignment

• Special case where tree is a star

• Which sequence should be the center of the star?

• The sequence \( x_i \) such that

\[
M = \sum_{j \neq i} \text{OPT}(x_i, x_j)
\]

is maximized.
Star alignment algorithm

• Pick $x_i$ to maximize $M = \sum_{j \neq i} OPT(x_i, x_j)$

• Find the optimal alignments between $x_i$ and all $x_j$.

• Join the alignments using once a gap always a gap technique.

• Running time = $O(k^2n^2)$ for alignments + $O(k^2L)$ for gap updates, where $L$ is the length of the alignment
  – each time a sequence is joined, at most $k$ sequences of length at most $L$ must be updated => $O(k.kL) = O(k^2L)$
Example

\[ x_1 = \text{ATTGCCATT} \]
\[ x_2 = \text{ATGGCCATT} \]
\[ x_3 = \text{ATCCAATTTT} \]
\[ x_4 = \text{ATCTTCTT} \]
\[ x_5 = \text{ACTGACC} \]

\[ x_1 \text{ maximizes } M \]

\[
\begin{align*}
  x_1 &\quad \text{ATTGCCATT} &  x_1 &\quad \text{ATTGCCATT} &\quad \text{ATTGCCATT--} \\
  x_2 &\quad \text{ATGGCCATT} &  x_4 &\quad \text{ATCTTC-TT} &\quad \text{ATGGCCATT--} \\
  x_3 &\quad \text{ATC-CAATTTT} &  x_1 &\quad \text{ATTGCCATT} &\quad \text{ATCTTC-TT--} \\
  &\quad \text{--} &  x_3 &\quad \text{ATC-CAATTTT} &\quad \text{ACTGACC--} \\
  &\quad \text{--} &  x_5 &\quad \text{ACTGACC--} &\quad \text{ACTGACC----} \\
\end{align*}
\]