Last week we discussed dynamic programming algorithm for global pairwise alignment using a distance scoring function. Today we will discuss global alignment using similarity scoring and semi-global alignment.

Dynamic programming algorithms for sequence alignment have four components:

- **Initialization** of the first row and column of \( a[i, j] \).
- A **recurrence relation** for \( a[i, j], i, j > 1 \).
- Determination of the **optimal alignment score** from the matrix \( a[i, j] \).
- Trace back through the alignment matrix to obtain the optimal alignment.

The details of each of these steps are what differentiate global, semi-global, and local alignment.

Global alignment with similarity scoring

Last week, we considered the “edit distance” scoring function, where \( d(x, x) = 0, d(x, y) = 1 \), and \( d(x, -) = 1 \). We can also use a similarity score

\[
S(\alpha(s, t)) = \sum_{i=1}^{l} (p(s'[i], t'[i])),
\]

where \( p(x, y) \) is a score that reflects the similarity of \( x \) and \( y \) and \( p(x, -) \) is the gap score. The similarity score, like the distance score, assumes that there is no interaction between different
positions in the alignment. This assumption of site independence is widely used in computational biology, although in many cases there are, in fact, dependencies between some positions within a sequence.

In general, amino acid substitution matrices assign a different similarity score to each pair of residues. Examples include the PAM and BLOSUM matrices. We will discuss such substitution matrices later in the semester.

For now, we will consider a simple similarity scoring function treats all symbols in $\Sigma$ equally:

$$p(x, x) = M, \\
p(x, y) = m, \\
p(x, -) = g,$$

where $M > m$ and $M > g$. Note that we require that $m \geq 2g$. If we allow $m < 2g$, then we exclude the possibility of an alignment with substitutions since the score of any substitution can be improved by replacing it with two gaps.

Under this simple scoring function, the dynamic programming algorithm for global alignment has the following steps:

- **Initialization:**
  
  $$a[0, i] = a[i - 1, 0] + g \\
a[j, 0] = a[0, j - 1] + g$$

- **Recurrence relation:**
  
  $$a[i, j] = \max \begin{cases} 
  a[i, j - 1] + g \\
  a[i - 1, j - 1] + p(i, j) \\
  a[i - 1, j] + g 
  \end{cases}$$

- **Optimal alignment score:** $a[m, n]$.

- **Traceback** from $a[m, n]$ to $a[0, 0]$ to obtain the optimal alignment.

**Semi-global alignment**

Global alignment seeks the best, full length alignment; that is, the best way to match up two sequences along their entire length.

For some applications, it is desirable to relax this requirement and not penalize gaps at the beginning and/or end of an alignment. For example, for sequence assembly, we seek sequence fragments that
overlap. We expect to be able to align the end of one fragment with the beginning of another. Occasionally, we may find a pair that overlap completely, but, in general, we expect some gaps at the beginning of one sequence and gaps at the end of the other.

Another example is aligning cDNA’s or EST’s with genomic DNA to identify gene structure. The cDNA will be completely covered by the genomic DNA, so we expect gaps at both the beginning and end of the cDNA fragment.

*Semi-global alignment* is a modification of global alignment that allows the user to specify that gaps will be penalty free at the beginning of one of the sequences and/or at the end of one of the sequences.

Given sequences $s$ and $t$, there are eight possible cases to consider:

- Gaps are allowed at the beginning of $s$, but not $t$; *e.g.*
  
  $$
  s: \quad _\quad _\quad D \quad O \\
  t: \quad R \quad E \quad D \quad O
  $$

- Gaps are allowed at the beginning of $t$, but not $s$; *e.g.*
  
  $$
  s: \quad R \quad E \quad D \quad O \\
  t: \quad _\quad _\quad D \quad O
  $$

- Gaps are allowed at the end of $s$, but not $t$; *e.g.*
  
  $$
  s: \quad D \quad O \quad _\quad _ \\
  t: \quad D \quad O \quad N \quad E
  $$

- Gaps are allowed at the end of $t$, but not $s$; *e.g.*
  
  $$
  s: \quad D \quad O \quad N \quad E \\
  t: \quad _\quad _\quad D \quad O \quad _\quad _
  $$

- Gaps are allowed at the beginning and end of $s$; *e.g.*
  
  $$
  s: \quad _\quad _\quad D \quad O \quad _\quad _ \\
  t: \quad R \quad E \quad D \quad O \quad N \quad E
  $$

- Gaps are allowed at the beginning and end of $t$; *e.g.*
  
  $$
  s: \quad R \quad E \quad D \quad O \quad N \quad E \\
  t: \quad _\quad _\quad D \quad O \quad _\quad _
  $$
• Gaps are allowed at the beginning of \( s \) and at the end of \( t \); \textit{e.g.}

\begin{align*}
\text{s:} & \quad _ _ \text{D O N E} \\
\text{t:} & \quad \text{R E D O } _ _
\end{align*}

• Gaps are allowed at the beginning of \( t \) and at the end of \( s \); \textit{e.g.}

\begin{align*}
\text{s:} & \quad \text{R E D O } _ _ \\
\text{t:} & \quad _ _ \text{D O N E}
\end{align*}

In semi-global alignment, we do not allow gaps at the beginning of \( s \) and the beginning of \( t \) in the same alignment. Nor do we not allow gaps at the end of \( s \) and the end of \( t \). Why not?

The global dynamic programming algorithm can be modified for semi-global alignment as follows:

• \textit{Initialization}:
  
  – To allow gaps at the beginning of \( s \), set \( a[0, j] = 0 \); i.e., the first row is zero.
  
  – To allow gaps at the beginning of \( t \), set \( a[i, 0] = 0 \); i.e., the first column is zero.

• \textit{Recurrence relation}: Same as global.

• \textit{Optimal alignment score}:
  
  – To avoid trailing gap penalties at the end of \( t \), we define the optimal score to be \( a[i^*, n] \), where \( i^* = \arg\max_i a[i, n] \). In other words, the optimal score is \( \max_i a[i, n] \).
  
  – To avoid trailing gap penalties at the end of \( s \), we define the optimal score to be \( a[m, j^*] \), where \( j^* = \arg\max_j a[m, j] \). In other words, the optimal score is \( \max_j a[m, j] \).

• \textit{Trace back}:
  
  – To avoid trailing gap penalties at the end of \( t \), trace back from \( a[i^*, n] \). In other words, trace back from the cell(s) in the last column with maximum score.
  
  – To avoid trailing gap penalties at the end of \( s \), trace back from \( a[m, j^*] \). In other words, trace back from the cell(s) in the last row with maximum score.

Note that when the first row (or column) of the matrix is initialized to zero, the traceback will end in the first row (or column), but not necessarily in the cell \( a[0, 0] \).