2.1 Introduction to Alignment

2.1.0.1 What is sequence alignment?

We have sequences drawn from some alphabet. In biology, we are usually interested in DNA or protein sequences.

DNA sequences are made from deoxyribonucleic acids. In classical biology, we are concerned with Adenine, Thymine, Guanine, and Cytosine. Protein sequences consist of the 20 amino acids. In sequence alignment, we are concerned only with linear sequences of nucleic or amino acids.

An alignment over a pair of sequences from alphabet Σ is a sequence over pairs of elements of Σ, one of which may be replaced by the special gap character ‘-’. Taking the first member of every pair yields the first sequence, with gaps inserted, and similarly taking the second member of every pair yields the second sequence, gaps inserted. For example, given the strings CGATTAGGGCC and GTAGTACCT, we can align them as follows:

ACGATTAGGGCC-
||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| ||| |
2.1.1 Similarity Scores

In order to score an alignment, we require a scoring function. We are given a function \( \delta(x, y) \) that outputs a score for \( x, y \) members of \( \Sigma \) or \('\)'. Then, recalling an alignment is a sequence of pairs \((a_1, a_2), (b_1, b_2), \ldots, (n_1, n_2)\) we take the alignment score to be the sum \( \delta(a_1, a_2) + \delta(b_1, b_2) + \cdots + \delta(n_1, n_2) \).

The choice of \( \delta \) significantly impacts the optimal alignment. The unit score is a simple scoring technique where mismatches and gaps have cost 1, and matches have cost 0, in other words:

\[
\delta(a, b) = \begin{cases} 
  1 & a = b \\
  0 & a \neq b 
\end{cases}
\]

We can interpret similarity scores as log probabilities relative to the likelihood that the sequences are unrelated, and then the maximally scoring alignment is the maximum likelihood alignment.

2.1.2 DNA Sequence Alignment

In an alignment, there are 3 possibilities at each position. We can have a match, a mismatch, or an indel (an portmanteau of insertion and deletion). When the two bases aligned to a position are the same, they are said to match, when they differ, they are said to be a mismatch, and when one is a gap character \('\)\', they are said to be an indel.

Biologically speaking, we have gaps for various reasons, and indels for various reasons. Depending on our application, we may wish to score them differently.

2.1.3 The DNA Code and Protein Sequence Alignment

There is a genetic code that translates triplets of DNA sequences, known as codons to protein sequences.

This code is very much like the ASCII code that converts bit sequences to text sequences.

DNA code is over \( \{A, T, G, C\}^3 \), whereas ASCII is over \( \{0, 1\}^7 \). Technically, each ASCII character has unique meaning, but many are control characters not intended to be rendered and many are rarely if ever used today.

The DNA code must be able to code for each of the 20 amino acids and the stop sequence (which is in a sense a control character). Additionally, DNA must be able to represent other things, and certain regions should be able to signal a region that should be transcribed; in order to enable the synthesis of arbitrary protein strands, we need redundancy to avoid conflating a protein sequence with a control sequence. Ignoring this, we require at least \( \log_4(2.196) \approx 6 \) bases to represent individual characters; rounding up to 3 wastes \( 6 - \log_2(21) \approx 1.608 \) bits of information per codon, but allows each codon to map to a specific base. In this sense, the DNA code is optimal.

When aligning amino acid sequences, we generally want to use a more sophisticated scoring metric. Margaret Dayhoff introduced the Point Accepted Mutation (PAM) matrices in [Eck and Dayhoff 1966]. These matrices give a similarity criterion based on the relative log probability of mutation from one amino acid into another over a given length of time. PAM250 is shown in Figure 2.1.
Figure 2.1: PAM250 log odds matrix

2.1.4 Edit Graphs

The concept of an edit graph is a powerful one, because there is a 1:1 correspondence between sequence alignments and paths through the edit graph. By assigning weights to edges in the edit graph based on similarity scores, the problem of computing the maximum similarity alignment reduces to identifying the maximum-distance path in a directed acyclic graph (DAG).

An edit graph over strings $x$ and $y$ of lengths $m$ and $n$, respectively, has $(m+1)(n+1)$ nodes, each labeled $n_{a,b}$, with $a \in \{0, 1, \ldots, m\}$ and $b \in \{0, 1, \ldots, n\}$. Each node $n_{a,b}$ has edges to $n_{a+1,b}$, $n_{a,b+1}$, and $n_{a+1,b+1}$, if possible (an edge to a node that does not exist also may not exist). See Figure 2.2 for an example of an edit graph.
Each diagonal edge represents a *match* or a *mismatch*, whereas each vertical or horizontal edge represents an *indel*. Then, any path through the edit graph represents a sequence of matches, mismatches, and indels that uniquely represents one possible alignment.

### 2.1.5 Dynamic Programming

#### 2.1.5.1 Global Alignment

The Needleman-Wunsch algorithm (Needleman and Wunsch, 1970) is a dynamic programming algorithm to identify the optimal pairwise global alignment.

The algorithm works by associating a cost with each edge in the edit graph, and then calculating the highest-scoring path through the graph. We store these scores in a *score matrix* according to the following rules:

\[
S_{0,0} = 0
\]

\[
S_{i,j} = \max \begin{cases} 
S_{i-1,j-1} + \delta(x_i, y_j) \\
S_{i-1,j} + \delta(x_i, \_)
\end{cases}
\]

In this formula, the first term of the maximum corresponds to the score of a match or a mismatch (depending on \(y_i\) and \(x_j\)), and the next two terms correspond to the score of indels.

The \(S_{i,j}\) term of this matrix corresponds to the optimal alignment score of a substring of \(x\) of length \(i\) and a substring of \(y\) of length \(j\). Because the edit graph is a DAG, we can calculate the cells in an order such that all the cells required to take the maximum have already been calculated.

#### 2.1.5.2 Local Alignment

The Smith-Waterman algorithm (Smith and Waterman, 1981) is a dynamic programming algorithm to identify the optimal *local* pairwise alignment. It closely resembles the Needleman-Wunsch algorithm, but differs in that it identifies locally rather than globally optimal alignments. To accomplish this, a simple change to the definition of \(S\) is required.

\[
S_{0,0} = 0
\]

\[
S_{i,j} = \max \begin{cases} 
0 \\
S_{i-1,j-1} + \delta(x_i, y_j) \\
S_{i-1,j} + \delta(x_i, \_) \\
S_{i,j-1} + \delta(_, y_i)
\end{cases}
\]

Here the 0 term captures the fact that a local alignment can start anywhere. If a region of an alignment was known to have negative score, we could always remove it to produce a superior local alignment, so the 0 term essentially wipes the slate clean whenever the alignment score becomes negative in a local region.
References

