MASSACHVSETTS INSTITVTE OF TECHNOLOGY

Problem Set 1
Due February 10

Assume, for both problems, that we are dealing with two DNA sequences of length n and m respectively: $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_m$.

Problem 1

An *intercalation* of two sequences is a list (without gaps) combining both their elements without gaps; for example, $x_1, x_2, y_1, x_3, y_2, y_3, y_4, \ldots$. The number of ways of intercalating two sequences of lengths n and m (you may assume that $m < n$) to give a single sequence of length $m + n$, while preserving the order of the symbols in each, is $\binom{m+n}{m}$.

a. By taking alternating symbols from the upper and lower sequences in an alignment, then discarding the gap characters, show that there is a one-to-one correspondence between gapped alignments of the two sequences and intercalated sequences of the type described in the previous problem.

(Thus, by a simple argument, the number of potential alignments is $\binom{m+n}{m} = \frac{(m+n)!}{m!n!}$. This is number is approximated in the lecture notes, using Stirling's formula, by $\frac{2^{2n}}{\sqrt{\pi n}}$.)

Problem 2

The approximation above suggests that a "brute force" search through all $\binom{m+n}{m}$ alignments would require searching a prohibitively large number of possible alignments. How can we avoid this problem?

In class (Lecture 2) we saw a tractable algorithm for finding the optimal *global alignment* of two sequences. This algorithm, called Needleman-Wunsch, relies on a score function F of two arguments. The value of $F(i, j)$ is the score of the *optimal sub-alignment* between sequences $x_1, \ldots, x_i$ and $y_1, \ldots, y_j$. Therefore, by definition, $F(n, m)$ is the score of the optimal alignment between the two original sequences. (This explains why we started our track-back for Needleman-Wunsch in the lower-right corner of the matrix in the lecture notes).

The function F depends, first, on the substitution matrix s and gap penalty d. The substitution matrix s is a simple function that scores both matches and mismatches. The value of $s(i, j)$ is the score that results from matching x_i to y_j. In a very simple example, we might simply give one score if $x_i = y_j$, and another if $x_i \neq y_j$: our "match" and "mismatch" scores. The BLOSUM and PAM matrices are more complex examples of the same kind of function, used to align proteins.

The function F is also interesting because it has a recursive structure: its value for any (i, j) pair depends on its value for some other (i', j') pair(s).

$$\begin{aligned} F(0,0) &= 0 \\ F(i,j) &= \max \left\{ \begin{array}{l} F(i-1, j-1) + s(i,j), \\ F(i-1, j) - d, \\ F(i, j-1) - d \end{array} \right\} \end{aligned} \tag{1}$$

This function has a particularly nice recursive structure: values of $F(i, j)$ depend on values of F for other, *smaller*, i and j. In other words, the optimal solution to the problem depends on the optimal solutions to one or more sub-problems. Problems with solutions structured in this way can be solved by a technique known as *dynamic programming*.

Dynamic programming builds a table, from the bottom up, of the solutions to *all* the possible sub-problems. Since any sub-problem in this case corresponds to a pair $i \leq n$ and $j \leq m$, there are at most n^2 distinct sub-problems – a significant improvement over $\binom{m+n}{m}$! We built this kind of table in the lecture notes

using a matrix of entries, one for each (i, j) pair. The value of an entry in the matrix depends on the values of up to three of its neighbors, reflecting the three terms in the maximization above (eq 1). Replacing the $\max$ with an $\arg\max$ (represented by an $\rightarrow$ in the lecture notes) allows us to trace-back the *actual alignment* which produced the optimal score.

Our use of a single gap parameter d here implies a simple model of gap scoring: "linear gap scoring." We also covered a more complex score function, "affine gap scoring," in class. Affine gap scores take the form $-e-(g-1)d$, as discussed in the notes; dynamic programming can also be used for affine gap scores, although that requires an additional level of bookkeeping than we require here. One can also imagine even more complex gap-scoring models, although not all of them are compatible with a dynamic programming approach.

For this problem, please:

- Calculate the alignment-score matrix for the optimal alignment of the DNA sequences `GAATACGGAT` and `GTAATAGGT`, scoring $+2$ for a match, -1 for a mismatch, and with a linear gap penalty of $d = 2$.
  - Report the matrix of scores,
  - ... the optimal score,
  - ... and *an* alignment of the two sequences with that score.
  - Draw arrows in your matrix to represent the route taken by $\arg\max$ for tracing back the optimal alignment.

  As an alternative, you may (if you feel comfortable) code the algorithm in Matlab (or a reasonable environment, negotiated with the TA) and give both the code and its output as an answer to this problem.
- In general, it may be that the $\arg\max$ for eq. 1 is not unique; you may represent this in your table with a double-arrow. What phenomenon does this reflect? Is this necessarily a problem with the alignment algorithm, that it sometimes allows these degeneracies?
- How sensitive is the result of your algorithm to the alignment scoring parameters s and d? (Try the same alignment with one or two other values of the parameters – how hard is it to find different "optimal" alignments?) What does this tell you about the choice of parameters when doing this kind of simple alignment?