

## Lecture 9, April 19, 2002

### 1 Longest Common Subsequence and Longest Common Substring

Definitions and computation via dynamic programming (LCsubsequence) and local alignment (LCsubstring), which is also via dynamic programming.

### 2 Introduction to Statistical and Probability Issues in Sequence Analysis and Database Search

When we use a target sequence to search a database of sequences, either by local or global alignment, we want to know if the high scoring alignments are biologically meaningful. Can we learn something useful about our target sequence from the high scoring alignments and what is known about the sequences involved in those alignments? This is of course an empirical question about the validity of the whole approach of trying to expose important biology from comparing sequences by aligning them under some objective function. The answer has been Yes (at least often enough), or otherwise we would not be so invested in database search and sequence alignment. However, it is not always true that the database sequence that gives the best alignment to the target sequence is actually related to the target sequence in any meaningful way. Since we often have no good intuition about whether an alignment is meaningful or not (that is, whether it reveals significant biology), we often use probability and statistics to help identify alignments that might be biologically important, or more often, to exclude those that are not.

Suppose you align a target sequence  $S_1$  to every sequence in a database  $D$  and the best similarity value found is  $S$ , and that is obtained by aligning  $S_1$  with  $S_2$ . How high does  $S$  have to be before you should get excited? Is the function, structure, or other properties of  $S_2$  likely to be carry over to  $S_1$ ? Only additional work (perhaps lab work) will resolve this for sure. But we would like some cheap computational or mathematical way to tell whether additional work is likely to be profitable.

The rough philosophy that is followed is to think about aligning  $S_1$  to each sequence in a database the size of  $D$  consisting of *randomly* generated strings, where the probability of any particular character is its frequency in  $D$ . Let  $D'$  denote this randomly generated database and let  $S'$  be the largest similarity value of  $S_1$  to the strings in  $D'$ . If  $S'$  is "much smaller" than  $S$ , then one feels more confident that the alignment of  $S_1$  to  $S_2$  has some true biological meaning, and not just something that would be seen in random strings, and hence it is worth investing additional effort to confirm this. Conversely, if  $S'$  is larger than  $S$ , then one feels much less confident (at least without some additional evidence or intuition) that  $S_2$  has any valuable relationship to  $S_1$ . (There really is a philosophical point here that runs all through science and statistics, but I am having a hard time making it explicit: Phenomena that could easily occur as the result of random acts, should not be taken as evidence that a non-random process is being observed, even though that may be the truth. This is actually a large debate, but we will leave it here – lacking any other reasons to believe that  $S_1$  and  $S_2$  are related in a meaningful way,  $S$  must be "close to" or larger than  $S'$  to be considered as good evidence of a relationship.)

So much for philosophy. How do we make this precise? One approach is to take every sequence in  $D$  and randomly permute its characters. The resulting database is taken to be  $D'$ . Then align  $S_1$  to every string in  $D'$ . Because we compute the similarity of  $S_1$  to every string in  $D'$ , we have the complete distribution of the similarity values, so we can see where  $S$  falls in that distribution. Let  $f$  be fraction of similarity values that are above  $S$ . We might pick some small cutoff  $C$ , say .001, and use the test that if  $f > C$ , then  $S$  is not large enough to be considered significant. That is, if one in one-thousand of the strings in  $D'$  are more similar (have a higher similarity value) to  $S_1$  than does  $S_2$ , then a similarity value of  $S$  seems like weak evidence for a true relationship between  $S_1$  and  $S_2$ .

An alternative is to randomly permute  $S_1$  and then align that string to  $D$ . What are the advantages and disadvantages of doing that?

Now what actual cutoff  $C$  should be used? That depends entirely on one's tolerance for false-positive results versus false-negative results. What are you most concerned about, wasting time and money checking out a sequence that is not actually related to  $S_1$  or missing an important clue to solving some important biological problem? If you have cheap ways to check out the reported sequences then you can set  $C$  high, but if not, you set  $C$  low. In

the end you decide if you want to use database search as a *filter* or an *oracle*.

Now what I've outlined above doesn't change if we use a score other than similarity (local or global), as long as we can get an empirical distribution of those scores when comparing  $S_1$  to every sequence in  $D'$  (or a large enough sample of  $D'$ ). And this kind of "Jumbling" or "Shuffling" test is actually used in some search applications. But this approach is not feasible for Blast or Fasta, because so many sequence are filtered out before their score is computed. In that case, analytical results are needed. For concreteness, let  $S$  now denote the best Blast score obtained by Blasting  $S_1$  against  $D$ . We want a *formula* for the probability that a score of  $S$  or better would be obtained from Blasting  $S_1$  against  $D'$ , i.e. a database of random sequences, where the size of that database is the same as  $D$ , and where its character composition is the same as  $D$ . Later, we will discuss such a formula.

### 3 Initial probabilistic analysis of random sequence alignment

In order to develop some intuition about probability and statistics of alignment values, we will examine two simple cases. Here is the first one:

#### A first simple case

Consider the alignment objective function of Maximizing the number of matches. This occurs either under global or local alignment when the mismatch, indel and gap costs are set to zero. The maximum value is also called the length of the *longest common subsequence*. To generate a random string, consider an alphabet of  $c$  characters, and for each position in the string, pick one of the  $c$  characters with equal probability  $p = 1/c$ . So for DNA,  $p = 1/4$ . Now consider two such random sequences of length  $n$  each, and let  $LCA$  denote the maximum alignment value under this objective function. What  $LCA$  value should one get excited about?

We will show that the expected value of  $LCA$  is at least  $n/c$ . Let's just take  $c = 4$  for concreteness. Here is the argument for  $n/4$ . Instead of computing a longest common subsequence, consider the longest prefix of the first string that also occurs as a subsequence in the second string. Let

$L$  denote the length of that longest prefix. For example, with *acctaca* and *catcaca*,  $LCA = 5$  (*ataca* for example), but  $L = 3$  (*acc*). In general,  $LCA \geq L$  for every pair of strings, so the expected value of  $LCA$ ,  $E(LCA) \geq E(L)$ , the expected value of  $L$ .

We will show that  $E(L) \geq n/4$ ? The analysis uses a simple algorithm: Start at the first character of the first string and scan the second string until the first point where you find a matching character or hit the end of the string; then move on to the second character of the first string and scan the second string forward from where you stopped, until you find a matching character or hit the end; etc. This algorithm certainly finds a prefix of the first string that matches a subsequence of the second string. Let's call its length  $P$ . By definition of  $L$ ,  $L \geq P$  and  $E(L) \geq E(P)$ . Actually,  $L = P$  always (and you are encouraged to figure out why), but we won't even need that fact. What is  $E(P)$ ? In finding  $P$ , each character of the second string is scanned, and if it matches the currently examined character of the first string, then the length of the matching prefix is extended by one. The probability that any *specific* character of the second string matches the current character of the first string is  $1/4$ , so  $E(P) = n/4$ .

In case you don't follow that last step, but assuming you have had a course in probability (which is a required prerequisite), here is a full demonstration:

Let  $X_i$  be a random variable which takes on the value 1 if the character in position  $i$  of  $S_2$  is part of the common subsequence found by the algorithm, and takes on a value of 0 otherwise. Then,

$$P = \sum_{i=1}^{i=n} X_i$$

and

$$E(P) = E\left(\sum_{i=1}^{i=n} X_i\right) = \sum_{i=1}^{i=n} E(X_i).$$

The last step is by the *linearity of expectation*. Now

$$E(X_i) = 0 \times 3/4 + 1 \times 1/4 = 1/4$$

so

$$\sum_{i=1}^{i=n} E(X_i) = n/4.$$

In summary  $E(LCA) \geq E(L) \geq E(P) = n/4$ . This is what we can establish with a simple analysis. But, clearly one should think that  $E(LCA)$  is much bigger than  $E(P)$  because of the constrained way we found  $P$ . In actuality, it is known that  $E(LCA)$  converges to  $n \times p_c$  as  $n$  grows, where  $p_c$  is some constant number which depends on  $c$ . The exact formula for  $p_c$  is unknown, but for  $c = 4$ , it is known that  $p_c$  is somewhere between 0.54 and 0.72. So for DNA, as the length of the string increases,  $E(LCA)$  grows to be larger than  $n/2$ , but not more than  $0.72 \times n$ .

The take-home lesson is that if we just use the simple alignment objective function of maximizing the number of matches (i.e. longest common subsequence), it will take a relatively large number of matches before we see a phenomena that can't be explained by random action alone.

The other interesting point is that  $E(LCA)$  grows linearly with  $n$ . That is, if the two strings both double in length, then the expected  $LCA$  also doubles in length. So in the context of local alignment, if we were to use such a simple objective function (maximizing the number of matches), the lengths of substrings selected by Smith-Waterman, would grow linearly with  $n$ . Later, we will look at the opposite extreme, where the objective function is to maximize matches, but mismatches and spaces have infinite cost. Then in the context of local alignment we will see that the length of the optimal local alignment grows logarithmically with  $n$ .