CSCI1820: Sequence Alignment **Spring 2017**

Lecture 5: February 16

Lecturer: Sorin Istrail *Scribe: Pranavan Chanthrakumar*

Note: *LaTeX template courtesy of UC Berkeley EECS dept. Notes are also adapted from notes from previous years' offerings of CSCI1810 and CCSCI1820.*

5.1 Dot Plots

Here, we introduce the idea of a dot plot. Essentially, we create a table for two sequences where one sequence spans the horizontal axis and the other sequence spans the vertical axis, and fill in the table such that a cell contains a dot if the letters in the row and column that the cell corresponds to match. Below, we give two sequences and their corresponding dot plot.

$ACTAATCGA$
$ACTTAACGA$

	A	C	T	A	A	T	C	G	A
A	•			•	•				•
C		•					•		
T			•			•			
T			•			•			
A	•			•	•				•
A	•			•	•				•
C		•					•		
G								•	
A	•			•	•				•

You'll notice many different phenomena in these plots. One such phenomena are a series of dots in a diagonal. What this means is that a substring occurs in both of your strings. What about backwards diagonals (AAT and TAA from above)? This means you have the reversal of some subsequence. Perhaps most interestingly, you'll notice some diagonals are just one space off; this could very well indicate a gap in a biological context! Lastly, note how the structure of the dot plot matrix and the Smith-Waterman matrix are somewhat similar, with strings spanning the axes.

5.2 BLAST Problem Recap

Let us remind ourselves again the basic problem regarding BLAST:

Given:

- A similarity (or scoring) matrix.
- A query sequence, Q.
- A database of sequences, DB.

Compute: Hits of the query Q in the sequences of the database.

The query sequence is typically either a DNA or protein sequence. For most of our discussion of BLAST, we'll consider the query as a protein sequence.

5.3 BLAST Phase I: Seeding

When conducting the seeding phase of BLAST, we also need both a k and a T. k-mers of the both the alphabet and query are called **words**.

Now, let us consider the following query sequence:

```
Q = RGDKGDRGE
```

Words of Q are: `RGD`, `GDK`, `DKG`, `KGD`, `GDR`, `DRG`, `RGE`.

We also explore the concept of the **neighborhood** of a word, which is based on our threshold constant T. The neighborhood of a word, u, is the set of k-mers of the alphabet such that the ungapped global alignment between u and the k-mer has a score greater than (or equal to) T.

For example, let us consider the PAM 250 matrix. If we align `RGD` with `RGD`, we have a score of 15. If we align `RGD` with `RGN`, we have a score of 13. Let us use $T = 12$. Now, we can say that both `RGD` and `RGN` are in the neighborhood of the word `RGD`. (Note that, indeed, one of the words of the alphabet that are in the neighborhood of the query word is the query word itself, which makes sense considering this would most likely be your highest scoring alignment pair.)

We will next define hits a little more formally than in the last lecture. Consider all the alphabet words that meet the threshold requirement when aligned with each of our query k-mers. We list these in an abstract sense here:

$$\begin{aligned}
\texttt{RGD} &= \{\texttt{RGD}, \ldots\} \\
\texttt{GDK} &= \{\texttt{GDK}, \ldots\} \\
\ldots &= \{\ldots\} \\
\texttt{RGE} &= \{\texttt{RGE}, \ldots\}
\end{aligned}$$

Each of the above sets on the right hand side are referred to as **hits** for a certain word of the query. When we find any of the hits in a sequence of the database during seeding, we say that that sequence has a hit. All hits are in the neighborhood of their corresponding query word.

Consider creating a dot plot as we did before, except this time imagine that our query spans the horizontal axis and that our database sequence spans the vertical axis. Each time we run into a hit, we mark the amino acids involved in the hit with a dot. Concepts like this will be helpful in understanding BLAST.

Now, we'll talk about the choice of word size, k, and threshold score, T. If we choose k to be relatively small, the resulting dot plot described above would be filled with dots, and it would be hard to discern any useful information. As you increase the word size, there will be fewer and fewer dots in our dot plot, but the number of hits you have will likely get smaller and smaller, and you might miss out on important alignments. You have a tradeoff here. Now consider varying the threshold score T. If T is small, it means we tolerate a lot of mismatches, and end up with a lot of hits. If T is big, we don't allow much tolerance, and end up with less hits. We see that varying both T and k allow you to have fewer or more hits, and can ultimately lead to more significant and faster-achieved BLAST results.

There are several "tricks" researchers use when carrying out BLAST to optimize the algorithm. One example of this is the "2 hit trick." This involves looking at our dot plot matrix, and only deciding to extend a seed if you find that it has two hits in a row.

The key concept behind BLAST, in the context of dot plots, is that when you carry out the $O(N^2)$ Smith-Waterman algorithm you are essentially looking through the entire space of the dot plot. With BLAST, you only look at certain regions of the space, denoted by the actual dots. The goal of BLAST is to be extremely fast and to claim that it still does what Smith-Waterman does; BLAST approximately runs in $O(N)$ time.

5.4 BLAST Phase II: Extension

The second phase of BLAST is the extension phase. We will think of the BLAST extension phase in terms of our dot plots again. Our goal, then, is to pick some of the dot-diagonals that look promising, and then take those sections of the DB sequence and query and run ungapped alignments while extending them. To be more specific, we extend each of the database sequence seed (originally a k-mer) and the query (originally a k-mer) to the left and right one letter at a time, run ungapped global alignments on these, and repeat this process.How do you know when to stop? If there is a certain amount of decrease that occurs in the alignment score as you extend the seed, you should stop. This decrease is typically referenced as a parameter, X, known as the falloff score.

All the alignments found insofar are referred to as high scoring alignments or **HSP's** (high scoring sequence pairs).

5.5 BLAST Phase III: Evaluation

Up until now, we have obtained HSP's. We now want to trim this set using a new threshold, S, that further separates high scoring pairs from low scoring pairs. BLAST should only report statistically significant alignments, so we need a way to determine the statistical significance of the HSP's we have so far.

5.5.1 Introduction to Karlin-Atschul Statistics

We now introduce a value, E, which represents the expected number of HSPs with a score of at least S (our cutoff) given by chance. E is given by the following equation, also called the Karlin-Altschul equation:

$$E = Kmne^{-\lambda S}$$

where m is the size of our query, n is the size of our database (the sum of the lengths of each sequence in the database), and K and λ are constants. Some things to note regarding the equation above include the fact that both λ and K are normalizing constants related to the scoring system and search space, respectively.

Karlin and Altschul derived the 5 axioms of Karlin-Altschul statistics (note that we axioms 3 and 4 are approximations, while we hold the other axioms to be true):

1. The scoring matrix must have some positive entries (you can't have only penalties in the matrix)
2. The expected score of the scoring matrix needs to be negative. The intuition behind this is that if you calculate the expected number of HSP's, you don't want that number to be positive as then it would be likely that you have many alignments due to chance
3. The letters in the sequences we align are independent and identically distributed; this is obviously not true, but is an approximation that the theory is based on
4. The sequences are infinitely long
5. Alignments do not contain gaps