CS1820 Algorithmic Foundations of Computational Biology Sorin Istrail

Homework 8

Due: May 12th at 11:59pm

1 BLAST (100 pts)

In this problem you will be implementing the BLAST algorithm, and then testing your code against a series of databases.

The BLAST Algorithm The BLAST algorithm is a heuristic search method that seeks words of length k that score at least T when aligned with the query and scored with a substitution matrix. Words in the database that score T or greater are extended in both directions in an attempt to find a locally optimal ungapped alignment or MSP (maximum segment pair) with a score of at least S. MSPs that meet these criteria will be reported.

The Substitution Matrix We will use the following DNA substitution matrix:

$$\delta(x_1, x_2) = \begin{cases} +5, & x_1 = x_2 \\ -4, & x_1 \neq x_2 \end{cases}$$

Roughly, the algorithm is as follows:

1. Break down the query string into all possible **words** of length k. Now consider each word, w, individually.
2. Create a list of **search terms**, comprised of w and all other words that, when compared against w, have a score of at least T.
3. Identify all the places in the database where one of search terms appears, and extend in each direction to find the maximum segment pair (MSP).
4. If the MSP has a score of at least S, report it as a match.

You will need to complete the following:

1. Implement the BLAST algorithm, which will accept query strings and search for matches in the database, using the index you constructed. Your BLAST program should take the following three parameters as input, as well as an input search string and query string. The default parameters should be as follows:

$$\begin{aligned} k &= 4 \\ T &= 10 \\ S &= 0.60 * 5 * |query\_string| \end{aligned}$$

Your output should be the BLAST score, the offset index of the match for both sequences, and the two statistical quantities which we describe below. We (and you) would probably appreciate pretty-printing the alignment too.

2. We would like you to calculate the probability of the MSP match and the quantity E of matches with scores at least S that you would expect for BLASTing two sequences with lengths m and n. Please do your reading, but we will attempt to describe the theory here. Given two sequences, the probability of finding a segment pair with a score $\geq S$ is

$$p_{S,m,n}(1) = 1 - e^{-(Kmne^{-\lambda S})} \tag{1}$$

$$p_{S,m,n}(c) = 1 - e^{-(Kmne^{-\lambda S})} \sum_{i=0}^{c-1} \frac{\left[Kmne^{-\lambda S}\right]^i}{i!} \tag{2}$$

where K and λ are the Karlin-Altschul parameters. (1) is for the case of a single MSP of the query string, (2) applies in the general case of c MSP matches.

Unfortunately, K and λ depend on the input parameters; typically they are precomputed for the given scoring matrix. Let K^* and λ^* be the Karlin-Altschul parameters for our scoring matrix. Also assume that the probability of each base is $p_i = .25$ for all bases. Then λ^* is the solution to the equation

$$1 = \sum_{(i,j)} p_i p_j e^{-\lambda^* \delta(i,j)} \tag{3}$$

where $\delta(i,j)$ is the scoring matrix we have defined. Please compute λ^*. K^* is more difficult to compute, and any student who solves for K^* earns 10 extra-credit points (see last page).

Then the expected quantity E of matches with scores at least S is:

$$E = Kmne^{-\lambda^* S} \tag{4}$$

For each alignment you return, please give the quantities E and $p_{S,m,n}(\#\ of\ matches)$ **in terms of K^***. If you do not wish to hard code an exponentiation function into your algorithm, feel free to have the algorithm output Matlab or similar code for this.

Stats are not always a BLAST. If you do further research on BLAST, please note that you are not implementing query length correction or the $E_{database}$ E-value which is the output of real BLAST. Some things are best left to BLAST magic.

3. You will test your BLAST on a practical hunt for Mammalian alignments of segments of the BRCA1 gene. Recall that in HW4 you aligned the BRCA1 – a cancer-associated gene – human sequence to its cDNA. I have extracted four highly conserved BRCA1 exons from four other mammalian species: mouse, dog, cow, and rat. For each of these exons, perform a BLAST alignment with the exon as query string and the BRCA1 human DNA as search string, and use the default parameters above.

For each of the 4 exons, output all MSPs (ie. all contiguous matches of score at least S) or return "None" if there exists no match of at least score S. For each MSP provide the BLAST score, the offset index of the match for both sequences, and the quantities E and $p_{S,m,n}(\#\ of\ matches)$ **in terms of** K^*.

Also, for just one of the exons please compare the computation time of your BLAST program to a global or local alignment program you have already written for the class. We hope you observe that BLAST is much faster. On Unices you can easily get the runtime by typing "time" in front of your command.

4. **Extra-extra credit (25 pts):** Compared to the frequency of searches, the underlying database changes relatively infrequently. So it can be extremely useful to create an index of the BRCA1 human DNA database beforehand, which the BLAST algorithm can reference to look up word locations. Write a program that constructs an index of the database.

Appendix

To give a general formula for the parameter K^*, we need to develop some notation. Let S_k be a random variable representing the sum of the scores of k independently chosen letters. Let $E(X)$ be the expected value of the random variable X—i.e., the sum of x Prob$(X = x)$ over all values x that X can attain. Let $E(X; X > 0)$ denote the same sum, but taken only over possible values of X that are greater than 0.

We need to define the constant C^*.

$$C^* = \frac{\exp\left\{-2\sum_{k=1}^{\infty}\frac{1}{k}\left(E[e^{\lambda^* S_k}; S_k < 0] + \mathrm{Prob}(S_k \geq 0)\right)\right\}}{\lambda^* E[S_1 e^{\lambda^* S_1}]}.$$

Then the parameter K^* of *Theorem 1* in the text is bounded between

$$K^- = C^*\left(\frac{\lambda^*\delta}{e^{\lambda^*\delta} - 1}\right),\ K^+ = C^*\left(\frac{\lambda^*\delta}{1 - e^{-\lambda^*\delta}}\right),$$

where δ is the smallest span of score values. When all scores are integers with greatest common divisor 1, then $\delta = 1$. A rigorous statement of Eq. **2** is that for n large (in practical terms $n \geq 150$ suffices)

$$1 - \exp\{-K^- e^{-\lambda^* x}\} \leq \mathrm{Prob}\left\{M(n) > \frac{\ln n}{\lambda^*} + x\right\} \leq 1 - \exp\{-K^+ e^{-\lambda^* x}\}. \quad \textbf{[6]}$$

Using K^+ for K^* always provides a conservative estimate of statistical significance. The infinite series for C^* converges geometrically fast, so that only a small number of terms are needed to get a good estimate of K^* (i.e., K^+).

For certain sets of scores the formulas above can be simplified to give closed-form expressions for λ^* and K^*. For example, if score 1 occurs with probability p, score 0 with probability r, and score -1 with probability q where $q > p$, then $\lambda^* = \ln(q/p)$ and $K^* = (q - p)^2/q$.

Theorem 1 allows explicit calculation of the probability that some segment from a random sequence has score less than or equal to any given value. For example, consider a specific protein of length n and set of amino acid scores. We wish to calculate the level below which 99% of the maximal segment scores for random sequences with similar composition and length will fall. First, we take the amino acid probabilities for a random protein model directly from the protein at hand. From these probabilities and the given scores, we can calculate the parameters K^* and λ^* as described above. Solving the equation $\exp\{-e^{-\lambda^* x}\} = 0.99$ for x yields $x = [-\ln \ln(1/0.99)]/\lambda^*$. Any segment with score greater than $[(\ln n + \ln K^*)/\lambda^*] + x$ is then considered significant at the 99% level.