1 Introduction

Due Date: Your kind AI suggested I give three weeks. This is a tentative due date–active participation in class, etc will keep it at three weeks. On the other hand, inattentive participation will move the due date to two weeks. Make sure you’ve finished reading Mount chps 1-3. Begin skimming Chapter Four.

Best,

Prof. Dalkilic

2 Assignment Description

The lab has several tasks:

A. Create a data exchange via the Internet that allows users to align two sequences using Smith-Waterman, opening gap penalties, gap penalties and scoring matrices.

B. Use search to discover $\lambda$ for normalizing the row score $S_{ij}$ value.

C. Modify your last program to produce ambiguous-free nucleotides in a region of genome.

D. Automate extracting promoter regions of genes.

All labs must be completed as per the last labs instructions, with READMEs, etc.

2.1 Lab 2 Part A

The first lab is meant to familiarize you with data exchange via the Internet. A user should be able to navigate this interface easily—and it should look professional. There should be, for instance, an example provided and help to FAQs if required. The user should be able to use a drop-down box for PAM or BLOSUM. Data-validation should be provided, but I’d be more interested in making the program complete than working on this aspect early on. You will write your own Smith-Waterman. The output should be easily parseable by another program, since this is one of the main methods of data exchange currently. An extra bonus will be given if you provide some semi-structured additional method of providing output. Often, you’ll be asked to provide your solution through a web portal that

1. Takes user input with parameter settings with client-side data-validation

   - two protein sequences
   - PAM250, BLOSUM82
• gap open extension penalty
• gap penalty

2. Processes the data server-side by using the Scoring Matrix, gap penalties, using Smith-Waterman you’ve written on your own.

3. Produces output presented client-side typically in an unstructured format—an alignment and a score for the alignment.

I encourage you at some time to take a software engineering class to learn about the kinds of processes to help make this process as effective for the client and you the developer as possible. We’ll speak about this during the semester, but it’s not nearly enough time.

3 Lab 2 Part B

The second lab helps you understand both about search and understand BLAST and its suit of tools better. The material is drawn from several sources. Abtractly, this is a kind of satisfaction search problem: You’ll have some equation $E$ unknown parameters $\rho_i$ that you need to provide vaues for–let’s denote this by $E(\rho_1, \rho_2, \ldots, \rho_n)$ and the some constraints involving either $E$ as a whole or some subset of unknown parameters. For some students newly introduced to symbols like $\rho_i$, what does it mean? Typically, $i$ belongs to either the natural numbers $\mathcal{N} = \{0, 1, 2, \ldots\}$ or a set of plain ordered (ad hoc usually lexicographically) symbols $S = \{s_1, s_2, s_3, \ldots\}$. The subscript acts as an argument, the larger symbol receiving the subscript a function, and this is mapped to some range. So, $\rho_2$ associated with set $S$ means first, $2 \mapsto s_2$ and that ultimately $\rho(2) \mapsto s_2$. In the case of probabilities, $p_i$ means slightly more: it means the probabilistic value $p$ that is associated with the $i$th symbol from some ordered set $S$. When ordering a set, you’re actually creating a function from $\mathcal{N}$ to some $S$. It’s a lot of work to write $F = \{(1, A), (2, C), (3, G), (4, T)\}$ then write $p(F(1))$ to meant the probability of $A$. So, you’ll often see $p_i$ to mean any one of the elements of $S$.

Example 1 Assume you have this

$$\sum_{i=1}^{n} \sum_{j=1}^{i} q_{ij} = 1 = p_{i}^{\lambda S_{ij}}$$

And this alignment:

ABC  
ACD  
BBD

to computationally find a $\lambda$ within some acceptable threshold.
You have landed on the planet Venus and have been given an alignment above. Using what you’ve learned about pairwise alignments, do the following:

- Create a log-odds model of the character set $S_{ij} = \log\left(\frac{q_{ij}}{p_i p_j}\right)$
- You’ll have to find the normalized score $\lambda S_{ij}$
- Any matching symbols have value $m$ and mismatch have value $M$.
- Two constraints must be met:
  - The Expect Score $E = \Sigma_{i=1}^{n} \Sigma_{j=i}^{i} p_i p_j S_{ij} < 0$
  - The Relative Entropy $H = -\Sigma_{i=1}^{n} \Sigma_{j=1}^{i} q_{ij} \lambda S_{ij} > 0$
- To create percent identity $\%ID = \frac{\sum S_{ii}}{\sum S_{ij}} \times 100$

Create a function $\text{Score}$ that takes as input (i) match $m$ for $p_{ii}$ and (ii) mismatch $M$ for $p_{ij}$ where $i \neq j \forall i, j \in \mathbb{N}, 1 \leq j \leq i \leq n$, (iii) an initial lambda $\lambda_{init} \in \mathbb{R}_{\geq 0}$, and (iv) a threshold $\theta \in [0, 1]$ and outputs a table with the heading:

<table>
<thead>
<tr>
<th>Match</th>
<th>Mismatch</th>
<th>$\text{S}$</th>
<th>$\text{E}$</th>
<th>$\lambda_{\text{bits}}$</th>
<th>$\text{H}$</th>
<th>$%\text{ID}$</th>
</tr>
</thead>
</table>

1. Fill out the table with varying values (at least five rows) to find some correlation among the headings.
2. Score (by hand) the sequence ABC using Smith-Waterman. Now using the $\lambda$s from above, normalize this value.
3. What conclusions can you draw between the variables—you might spend a paragraph or two on this answer.

4 **Lab 2 C**

Rewrite your initial program from two weeks ago to produce a nucleotide sequence free of ambiguous nucleotides. You should prove the user with one of two options:

- Pick deterministically (This means the choice can be determined $a \ priori$ to the run).
- Pick probabilistically (This means the choice cannot be determined $a \ priori$ to the run).

I suggest implementing more than the minimum of two. The switches are $-d1, -d2, \ldots, -d n$ for options for the first kind of switch, and $-p1, -p2, \ldots, -pn$ for the second. You must implement at least $-d1$ and $-p1$. Additional switches will be rewarded. Please explain your rationale for your choices. This last lab is to help you understand how important thinking carefully about the implementation so that subtle changes don’t force a cascade of changes that take nearly as long as rewriting the application from scratch. What I’d like you to do think deeply about what you could have changes in your design to make this conceptual change affect the logical change have as little aspect as possible. Again, you should produce FASTA format with some kind of meaningful preamble. I suggest taking a window approach that makes distributions sensitive to neighborhoods of nucleotides.
5 Lab 2 Part D

The fourth lab also is meant to familiarize yourself with elements of genomic data processing. You’ll be asked to create a FASTA file of promoters $x$ bases upstream of genes in *Drosophila m.* Regulatory genomics has become very important lately and this is the region one would typically to examine.

I am providing two links: one to the smallest chromosome of *Drosophila m.* and one to the annotated genes. Write a program `pfind` that takes four inputs (i) `file1` is a FASTA chromosome file and (ii) `file2` a single file of annotated genes in FASTA format and (iii) the number of basepairs upstream to the beginning (1-5000) of the gene and (iv) produces a single FASTA file `file3` with each single promoter receiving a preamble that includes the gene name, its start location, and how far upstream from the gene.

Here is the chromosome:

ftp://flybase.net/genomes/Drosophila_melanogaster/current/fasta/dmel-4-chromosome-r4.3.fasta

Here are all the genes on this chromosome:

ftp://flybase.net/genomes/Drosophila_melanogaster/current/fasta/dmel-all-gene-r4.3.fasta
6 Notes & Peculiarities & Some Examples

Because I grew-up in Computer Science, I don’t see any difference between $T$, $\text{Truth}$, truth, 1, or non-zero. I think using $C$, $C++$, and maybe even Scheme sort of brainwashed me. So, I’ll use them interchangeably. I know there is a pervasive philosophy of using objects and types and so forth, and I’ll try to adhere to some of it; but like all philosophies, we each have our own recipe of how we use it. I’m writing these notes–hopefully to compile in a book–to help computer scientists become bioinformaticians by developing a better understanding the algorithms and implementations of these various algorithms and then judging these efficiency and effectiveness–typically, one can simply begin with

- What is the general problem you’re solving? Why is important? Why would answering the problem be significant?
- Does it scale?
- Does it perform better than existing solutions?

Also, I’ll probably swap sentence with expression without much compunction–the former is from logic, the latter from programming (probably algebraists would lay claim to this though too), but to me, they’re pretty much the same. These notes are rough too–and I’m trying to produce them with an occasionally uncooperative LaTeX. Also, we assume we’re moving from Left to Right over sequences–that $Left = 5'$ and $Right = 3'$. 

7 Probability–just another function

We’ll be working in a finite set of symbols. Usually, when this is associated with probabilities, the symbol $\Omega$ is used to represent this set and is called the sample space. We place a further restriction on $\Omega$ that, unlike sets in general, each element $\omega \in \Omega$ appears only once, i.e., it’s unique. Each $\omega$ is called an elementary event to emphasize this point. An event $A$ is a subset of the sample space, written in symbols $A \subseteq \Omega$. With each elementary event we can associate a real number on the interval $[0,1]$. This means we provide a mapping or function $P : \Omega \rightarrow [0,1]$. But we can constrain $P$ such that

$$\sum_{\omega \in \Omega} P(\omega) = 1$$

Definition 1 Given a sample space $\Omega$ and a function $P : \Omega \rightarrow [0,1]$ such that $\sum_{\omega \in \Omega} P(\omega) = 1$, then $P$ is called a probability function.

An event $A$ is a subset of the sample space; in symbols $A \subseteq \Omega$. We can find the probability of $A$ by simply restricting $\Omega$ in Eq. 1 to $A$ and write $P(A)$ to mean

$$\sum_{\omega \in A} P(\omega) = 1$$

There many questions we can pose using $\Omega$ and $P$ tying them together through relying on

- $\Omega$ being set and, therefore, set operators $\cup, \cap, \sim$ ($\sim$ is relative complement with respect to $\Omega$).
- $P$ being a non-negative real associated uniquely with each elementary event.

Here are some examples.

Example 2 Recall that the emptyset $\emptyset$ is always a subset of any set; hence $\emptyset \subseteq \Omega$. Thus, we can find the probability of $\emptyset$ in the spirit of Eq. 2:

$$\sum_{\omega \in \emptyset} P(\omega) = 0$$
Example 3  Given two events $X, Y \subseteq \Omega$, we can find $X \cup Y$, $X \cap Y$, and $\sim X$. Using $P$ we can find the probability of these events:

$$
\Sigma_{\omega \in X \cup Y} P(\omega) = \Sigma_{\omega \in X} P(\omega) + \Sigma_{\omega \in Y} P(\omega) - \Sigma_{\omega \in X \cap Y} P(\omega) \quad (4)
$$

It’s really a lot of symbols to write for each probability the $\omega \in \text{some-set}$ and the $\Sigma$ sometimes too–so we’ll agree here to keep the clutter to a minimum.

Suppose you have a genomic alphabet $\Sigma = \{A, C, T, C\}$. You can find an infinite number of probability functions $P$. One is where $P(A) = P(C) = P(T) = P(G) = \frac{1}{16}$. The $|\cdot|$ function takes an event and returns the number of elements. In this case, $|\Sigma|$ would return 4. If we’re interested in a particular substring—the number of times $x = x_0x_1 \cdots x_k$ occurs in $y = y_0y_1 \cdots y_n$ where $x, y \in \Sigma^*$, we’ll write $|x|_y$. What about a word, like a promoter or gene, over $\Sigma^*$ called Kleene Closure?

Example 4  What if you have AATTAGCTCT and want to know the frequency of CT. You can approach this a number of ways.

- You can assume uniform probability (as in the previous problem and independence). Then $P(CT) = P(C)P(T) = \frac{1}{16}$.

- You can simply find the relative frequency of CT. The relative frequency is (this is easiest done in verbatim mode):

  AATTAGCTCT
  CT
  CT
  CT
  CT
  CT
  CT
  CT
  CT
  +
  CT
  CT
  +

  How many pairs?

  AA, AT, TT, TA, AG, GC, CT, TC, CT

  So, $P(CT) = \frac{2}{5}$. You could have used Bayes too:

  $$
P(CT) = P(T|C)P(C) = 1 \times \frac{2}{10}
$$

  Why the discrepancy? Because of counts–but $\frac{2}{9} - \frac{2}{10} = .2 - .2 \approx .02 = 2\%$ difference. What about alignments? Let’s work through two examples

Example 5

AATTAGCTCT
ACCTAGCCTC
+ + + +

So, $\frac{4}{10} = \frac{2}{5}$

AATTAGCTCT row (1,2) row(1,3) row(2,3)
ACCTAGCCTC + + + + + + + + + + + +
ATTTGCTCT

So, $\frac{10}{30} = \frac{1}{3}$
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>\neg(A \land B)</th>
<th>\leftrightarrow</th>
<th>\neg(A \lor \neg B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 1: Showing two sentences (or expressions) are equivalent.

8 What the Heck does that symbol mean?

**Definition 2** \([x, y]\) means the set of reals greater than or equal to \(x\) and less than or equal to \(y\). In symbols we’d write for \(x, y, a \in \mathbb{R}\), \([x, y] = \{a|x \leq a \leq y \land x \leq y\} \),

**Definition 3** De Morgan’s Law really is about the relationship among operators or symbols. \(\neg(A \land B) \equiv (\neg A \lor \neg B)\) and \(\neg(A \lor B) \equiv (\neg A \land B)\). The *logical equivalence* relation means the two sentences when connected with \(\leftrightarrow\) produces a *tautology*: the outcome is *always True* (or 1) As an example in Fig.8, I’ve shown the first De Morgan’s Law is true.

Notice that the values to the left and right of \(\leftrightarrow\) are *identical*. Sometimes, \(\leftrightarrow\) is called *coincidence*. If there is a true associated with all possible inputs, then the expressions are *logically* equivalent. You use De Morgan’s Laws to get you out of little problems. For example, it’s a common problem to ask whether a regular language is *closed* under intersection—that is, does it still remain a regular language. So you’re trying to find whether \(L_1 \cap L_2\) is regular. If you can prove

- \(\sim L_2\) (the relative complement of the language—just swap final accepting states in the DFA with those that aren’t and it’ll be the set \(\Sigma^* - L_2\))
- The union of two regular languages is regular \(L_1 \cup L_2\) (you can make, for example, an \(\epsilon\) transition to either DFA)

you can then use the fact that

\[
L_1 \cap L_2 = \sim \sim L_1 \cap \sim \sim L_2 = \sim (\sim L_1 \cup \sim L_2)
\]  

(5)

**Definition 4** For a set \(\Sigma\), \(\Sigma^*\) called *Kleene Closure* is

\[
\Sigma^* = \Sigma \cup \{xy | x, y \in \Sigma\} \cup \{xyz | x, y, z, \in \Sigma\} \cup \ldots
\]  

(6)