Homework 2

David Angeles Albores, bi183

January 18, 2018

Problem 1. Compute p_{ACGT} for an HMM with 2 hidden states, 4 observable states and a sequence ACGT that is of length 4

To find the total probability of a given sequence σ we must calculate the following:

$$p(\sigma) = \sum_x p(x)p(\sigma|x),$$

where x is every possible explanation that can give rise to σ. To do this, we must implement the forward algorithm and add the entries for the last column. I am sorry I have not implemented this here, but I am in rural Mexico at the moment with limited computer access....

Problem 2. How many explanations are there for the sequence $\sigma = \{1, 1, 0\}$ in an HMM with $k = l = 2$ (hidden and observed states) with a sequence of length 3?

Assuming that all observed states are accessible from all hidden states, then there must be 2^3 possible sequences.

Problem 3. Let $S \in \{(,)\}^n$. For example $S = (()(()))(()$. Define a gene parse to be a sequence $1 \leq i_1, ..., i_{2k} \leq n$ where $S(i_{2r-1}) = ($ and $S(i_{2r}) =)$ for $1 \leq r \leq k$. In the example above (2, 6, 8, 10) is a gene parse. If S contains $n \geq 1$ open parentheses (and $m \geq 1$ close parentheses), show that the maximum number of gene parses S can contain is given by the Fibonacci number F_{n+m+1}. Note that in this enumeration the empty parse is a valid gene parse.

Note: The problem only requires us to show that the maximally parsable strings follow the Fibonacci series, not prove it from first principles. To do this, it appears to me to be sufficient to a) figure out how to build the maximally parsable strings, then b) build the first few and show the Fibonacci sequence. Maximally parsable strings containing $n = m$ number of open and closed parentheses are sequences of $() \ldots () \ldots ()$. I call these strings even. Odd maximally parsable strings where $n < m$ have sequences $() \ldots () \ldots ())) \ldots)$. See Table 1 for the Fibonacci sequence and gene parses. In the rest of the answer, I give a more thorough explanation of my thought process, but do not arrive at a formal proof.

Note 2: n is first used to refer to the total length of the string S, then is recycled to refer to the number of open parentheses. This ought to be corrected in future versions.

To answer this problem, we must figure out how to construct the string, $\hat{S}_{n,m}$ that contains the maximum possible number of gene parses. The rules on how to parse strings are given to us: A gene parse is a sequence of string indices, such that any odd index represents a '(', and any even index represents a ')'. Let the string representation of such a sequence be called a "substring". We would like to generate the string that can generate every single such possible sequence, from the shortest to the longest parse.

From the rules above that adding opening parentheses to a string S will at most increase the number of gene parses linearly for any string. By symmetry, closing parentheses will do the same. To increase growth beyond linear, we need to append *both* opening and closing parentheses to a string (call "()" a unit substring). Using this construction, we can generate the longest possible gene parse, $(1, 2, \ldots, 2n-1, 2n)$, where n is the number of open (or closed) parentheses. Since this string contains the longest possible gene parse, it also must contain the greatest number of possible gene parse combinations. Thus, to generate the set of maximally parsable strings, starting from the empty string, $\cdot$, we must append to it one parsable unit at the start or end of the string iteratively.

Because of the recipe we can use to generate $\hat{S}_n$, the gene parses it contains must be strictly related to its substring $\hat{S}_{n-2}$. Moreover, the number of gene parses must only depend on the total number $n + m$ of open and closed parentheses. Let us call the maximum number of gene parses for a string of length $n + m$, $G(n + m)$. Then, the above statement means that $G(2n) \geq G(2n - 2)$. Since a subset of the gene parses in $\hat{S}_n$ are the same, it follows that we can write $G(2n)$ recursively:

$$G(2n) \sim G(2n - 2). \tag{1}$$

However, Eq. 1 isn't complete because we know that the two sides are not trivially equal. Notice that the function above relates string of length $2n$ to the string of length $2n - 2$. Recursive functions often relate the n^{th} term to the $(n - 1)$ term as well.

We can prove that $G(2n)$ depends on $G(2n - 1)$ if we can show that the string $\hat{S}_n$ contains all of the parsable substrings in $\hat{S}_{n,n-1}$ (where I have returned to the double notation to emphasize that there are now n open parentheses and $n - 1$ closed parentheses). First, though, we must understand what $\hat{S}_{n,n-1}$ looks like. This is a string with n unit substrings and a lone closing parenthesis (call this an odd string). Odd strings like this must look like $() \cdots ())$ to maximize the number of gene parses. It follows that these odd strings have gene parses that skip an index. For example, ()) includes the gene parse (1,3), which skips an intermediate parenthesis (index 2). Even strings can be viewed as odd strings with filler in between: (...). It is these odd strings that contribute to the greater than linear increase in gene parses. Therefore,

$$G(n + m) = G(n + m - 1) + G(n + m - 2) \tag{2}$$

Since the empty string and the unit strings both contain a single gene parse (the empty parse), the above is equivalent to the Fibonacci equation.

String	Parses	No. of Parses
.	()	1
(	()	1
()	(), (1,2)	2
())	(), (1,2), (1,3)	3
()()	(),(1,2), (3,4), (1,4), (1,2,3,4)	5

Table 1. The first few maximally parsable substrings, $\hat{S}$. The number of parses in each string follows the Fibonacci series.