19/20

William Argus
A12802324
ECE 208

Homework 7 Writeup

Asymptotic running time for estimate_gtr_pair.py:

Here the function will be described and the runtime will be given. First each of the sequences are looped through. This makes this part *O(2l)*, where *l* is the length of the longest sequence. Next is a list comprehension of *O(2l)*, to get the gtr probs, and then variable definitions. Next is a for loop that is order *O(l)* and more variable definitions and single line computations to get the gtr rates. In total the runtime is *O(2l) + O(2l) + O(l),* so the asymptotic runtime is *O(l).*

Asymptotic running time for estimate_gtr.py:

Here the function will be described and the runtime will be given. First there is 2 nested for-loops that loop through all the leaf nodes. In the inner for loop, the **estimate_gtr_pair** function is called. So the total runtime is *O(n^2l), where l is the length of longest sequence and n is the number of leaf nodes.* Following this are single line computations to find the average gtr probs and gtr rates.

Next is the optimization. This is a for loop that runs numSteps (an integer parameter), a list of length 6 for each of the 6 parameters, and then for each of the parameters, the function **likelihood** is called at most 1 + countSteps (an integer parameter) times. The runtime of likelihood is described below (taken from homework):

The main loop goes through all *n* nodes. Each time it loops it either sends the current node to the function *popleafs* (if it is a leaf) or the function *updateMatrix* (if it is not a leaf). Within the function *popleafs*, the loop runs *l* times. Within the function *updateMatrix*, there is a nested for-loop, with the outer loop going through *l* times, the inner loop going through 4 times. The runtime of this main loop is then written as *O(nl)*, since the linear 4 in the *updateMatrix* runtime is not considered in big-O notation, making that function and *popleafs* have runtime *l*, with both inside a for loop that runs *n* times.

The final loop runs *l* times in order to find the likelihood of the tree, making its runtime *O(l)*. Combined, this results in a runtime for the function of *O(nl) + O(l)*.

Accordingly, the asymptotic runtime of the function likelihood is *O(nl),* where *n* is the number of nodes in the tree and *l* is the length of the longest sequence.

Accordingly, the runtime of this optimization loop is:

*O(numSteps*6*(1+countSteps)*nl)*, which gives the runtime of the optimization loop *O(nl).*

Therefore, the final asymptotic runtime is given by *O(n^2l) + O(nl), so* ***the asymptotic runtime of estimate_gtr is O(n^2l).***

Equations for Estimating from a pair of Sequences:

In order to estimate the GTR probabilities, $\pi_A, \pi_G, \pi_C, \pi_T$, the approach to find the maximum likelihood is as follows: Sum the number of occurrences of each letter between the two sequences and then divide by the total number of letters between the two sequences. This will give the probabilities of each letter occurring, which is the values we are looking for.
For example: $\pi_A = (total\ number\ of\ A's\ in\ sequences\ 1\ and\ 2)/(length\ of\ seqence\ \ 1 + length\ of\ sequence\ 2)$.

In order to estimate the GTR rates, first the probability matrix, *P*, must be found. It is found by counting the number of each type of transition from sequence 1 to sequence 2, and the number of each type of transition from sequence 2 to sequence 1. So the number of *A->G* transitions is the number of times a site in sequence 1 is *A* and the corresponding site in sequence 2 is *G* plus the number of times a site in sequence 2 is *A* and the corresponding site in sequence 1 is *G.*

In order to generate the probability matrix *P*, assuming the rows and columns go in the order (starting from the upper left corner of the matrix) A,G,C,T, with the row being the starting site and the column being the ending site. For example, the probability of going from A to G is given by the cell in the matrix located in the 1st row, 2nd column. In order the find the probabilities in these cells, the number of transitions found above is divided by the GTR probability (found previously as well) that corresponds to whichever letter the row is (which is whichever letter the starting site is). For example, to find the value of the cell in the 1st row, 2nd column, it is done by: $(\text{the number of } A\!-\!> G \text{ transitions})/\pi_A$. Keeping in mind that as mentioned above, the number of A -> G transitions is the number from sequence 1 to sequence 2, as well as the number from sequence 2 sequence 1. Following this format, the probability matrix *P* is filled in.

Once *P* is found, the *R* matrix can be found by the following equation:
$R = {}^{\log(\mathrm{P})}/_{t}$, where *t* is the branch length between the nodes from which the sequences were taken, the log is done with base e (otherwise known as ln). *R* will be in the following format:

$$
\begin{array}{c|cccc|}
 & A & G & C & T \\
A & -(\alpha\pi_G+\beta\pi_C+\gamma\pi_T) & \alpha\pi_G & \beta\pi_C & \gamma\pi_T \\
G & \alpha\pi_A & -(\alpha\pi_A+\delta\pi_C+\varepsilon\pi_T) & \delta\pi_C & \varepsilon\pi_T \\
C & \beta\pi_A & \delta\pi_G & -(\beta\pi_A+\delta\pi_G+\eta\pi_T) & \eta\pi_T \\
T & \gamma\pi_A & \varepsilon\pi_G & \eta\pi_C & -(\gamma\pi_A+\varepsilon\pi_G+\eta\pi_C)
\end{array}
$$

Now, the desired parameters can be solved for by taking each of the two cells that contain a respective parameter, dividing the value in that cell by the π_i value shown in the picture of matrix *R* above, and then taking the average of those two results to get that rate.
For example, once the *R* matrix is found, the procedure to find the parameter α, which is the r(A->G) and r(G->A) parameter, is as follows:
The cell is the first row, second column of this matrix will be designated q_{AG} and the cell in the second row, first column of this matrix will be designated q_{GA}.

$$\alpha = \frac{q_{AG}/\pi_G + q_{GA}/\pi_A}{2}$$

The other parameters are found accordingly. If it is desired to normalize to compare answer's with the given answers, one needs to only divide by whichever of the rates is normalized to 1.0 (this normalizing rate varies depending on if the multi or the pair example data is used.)

prove that your approach gives ML estimates!
-1 pt

Algorithm for Estimating Multiple Sequences:

In order to estimate multiple sequences, the following algorithm is used.

First, there are 2 nested for-loops looping through all the leaves, with the inner for loop sending the set of two sequences given by the current two leaves to the function gtr_params_pair to computer the gtr_probs and gtr_rates between those two sequences. Since the 2 for loops are nested, every combination of 2 leaves (permitted they aren't the same sequence) is sent to the gtr_params_pair function. Each time this function is called, the results are stored in a list. There are 4 lists for the gtr probs (one for each) and 6 lists for the gtr rates (one for each). Once these nested for loops have been run through, there is now a list for each of desired quantities that has the calculation of that quantity for each of the pairs of non-equal sequences.

By averaging these lists, the maximum likelihood estimate for the gtr probs is given, and a good starting point

No, it is not the ML estimate!

for the optimization of the gtr rates is given. It should be noted that one could stop here and still have reasonable results for the gtr rates, but the optimization provides even better results.

To optimize, a for-loop loops through the desired number of optimization steps (ultimately set to 1 since it gave good results and cut down the runtime). In each loop, the 6 gtr rates are looped though; for each rate, a step is taken in increasing the rate, decreasing the rate, and keeping it the same. Whichever step has the best likelihood, that step is repeated in order to continually modify that gtr rate until the likelihood of the tree, rate, and sequences (given by the function **likelihood**) stops increasing, or the number of improvements (given by countSteps) is reached. At that point that rate is saved and the next gtr is moved on to. Once all 6 rates have been improved, the new rates are applied to the gtr rates matrix and the for-loop running the desired number of optimization steps repeats. This has given very good results when compared to the given values for gtr probs and gtr rates.