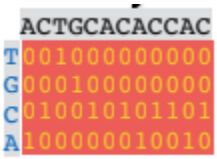
Nathan Newbury Writeup_Hw7

Recursive Formula and Base Case:

```
  # Recursive Formula
    for node in tree.traverse_postorder():
        # Iniitialize the leaves
        if node.is_leaf():
            store = np.zeros((4,len(seqs[node.get_label()])))
            # Leaf sequences and probabilities are known, so we just
generate matrix
            for i,letter in enumerate(seqs[node.get_label()]):
                if letter == 'A':
                    store[:,i] = [1,0,0,0]
                elif letter == 'C':
                    store[:,i] = [0,1,0,0]
                elif letter =='G':
                    store[:,i] = [0,0,1,0]
                elif letter == 'T':
                    store[:,i] = [0,0,0,1]
            L[node] = store


    #Main recursion
        else:
            prod = 1
            #Multiply each child node likelihood matrix together
elementwise because each event is independent
            for child in node.child_nodes():
                #calculate probability matrices
                t = child.get_edge_length()
                tranP = sp.linalg.expm(t*Rnorm)
                # find the matrix from the child node
                childprobs = L[child]

                # tranP * childprobs matrix multiplication fills out the
elements
                prod = prod *np.matmul(tranP,childprobs)
            L[node] = prod
```

```
    # for the root, we do the same with the independent probabilities pi
    Likear = np.matmul(gtr_probs,L[tree.root])

    # to find the total log likelihood, we take the log sum
    log_likelihood_score = np.sum(np.log(Likear))
```

In order to find the log likelihood of a tree topography without knowing the internal nodes, we traverse up every node of the tree to the root. At each node, we create a matrix representing an array of probabilities for each letter i in our sequence. Each element represents the likelihood of that letter transitioning into the letter of the known leaves. At the leaves, we know the sequences, and all probabilities are either 1 or 0. The matrix has rows [A,C,G,T] and columns[i…to … sequence length]). Because the sequences at the leaves are known, each column will have one 1 and three 0s (These are not random variables). Below is an example of our base case, the leaf matrix:



As we move up the tree, we fill out this same matrix of probabilities for the parent nodes. First we find the time incident on each child node and find the transition probability matrix with e^{tRnorm}. We can find the probability that a parent node sequence transitioned into a child node, store it as a likelihood matrix of the same format, and repeat the process up the tree.

The transition probability matrix multiplied with the likelihood matrix of the children nodes results in the likelihood matrix of the parent. Because we have 2 child nodes, but all events are independent, we can multiply the two child node matrices together elementwise to find the total likelihood matrix of the parent.

Matrix multiplication is a compact way to implement this recursive formula from the slides:

$$L(u,s) = \left(\sum_x P(D_v = x \mid D_u = s, t_1, \theta) L(v,x)\right) \cdot \left(\sum_y P(D_w = y \mid D_u = s, t_2, \theta) L(w,y)\right)$$

Were $P(D_v = X | D_u = s, t_1, R)$ represents our transition probability matrix (u->v) at element (s,x) and L represents our child (v) likelihood matrix at element (x,i). When finished we should have a

matrix of size (s,i) where len(x) = len(s). S is our categories [A,C,G,T] and it iterates through the letters in our sequence. We sum across all categories in x, as any letter has some probability to transition to the letter of our child node.

Finally, we multiply the root by our pi distribution (initial probabilities). Then we take the log sum of that value for our log likelihood.

Asymptotic Running time.

Our asymptotic running time is O(n*l). We loop through every node of the tree (n-1 + n). At each tree node, we multiply 2 matrices together. Multiplying a nxm matrix by a mxl matrix is of time complexity O(n*m*l). In our case, the number of categories (A,C,G,T) is fixed to 4. We multiply a 4x4 by a 4xl matrix for a time complexity of O(16*l) at each node. In the limit l and n go to infinity, we get a time complexity of **O(n*l)** where n is the number of nodes and l is the sequence length.