1 Probabilistic Models of Evolution

1.1 Background

Assume we are given:

- A rooted binary tree $T$.
- The lengths of the edges in $T$: $\tau = \{ t_{ij} \}_{(i,j) \in T}$, where $t_{ij}$ is a scalar that represents the length of the edge from $i$ to $j$.
- Assume for simplicity that the length of all the sequences is 1. So we denote: $\Sigma = \{ A, C, G, T \}$.

Let $x_1, ..., x_n$ be the sequences in $T$’s leafs, $x_{n+1}, ..., x_{2n-2}$ be the sequences in the internal nodes and $x_{2n-1} = x_r$ be the sequence in the root. Each sequence is a nucleotide: $\forall i \ x_i \in \Sigma$.

We consider this model as a hidden Markov model, and the sequences in the leafs as the observations. Now, we would like to answer the following questions:

1. What is the likelihood of the observations: what is the probability of the sequences in the leafs, i.e. $P(x_1, ..., x_n)$.

2. What is the posterior of the root: what is the probability that $x_r$ is a specific sequence, given the sequences in the leafs. i.e. $P(x_r = a \mid x_1, ..., x_n)$.

3. What is the posterior of an internal node $x_i$, $i \in [n+1, 2n-2]$: what is the probability that $x_i$ is a specific sequence, given the sequences in the leafs. i.e. $P(x_i = a \mid x_1, ..., x_n)$.

2 Up Algorithm

2.1 Motivation

We will answer the first question using Up Algorithm which operates in a similar manner to Forward Algorithm and Backward Algorithm.

Applying the chain rule, we can simplify the likelihood:

$$P(x_1, ..., x_n) \overset{(1)}{=} \sum_{x_1} \sum_{x_2} ... \sum_{x_{n+1}} P(x_1, ..., x_n, x_{n+1}, ..., x_{n+1}) \overset{(2)}{=} \sum_{x_{n+1}} \sum_{x_{n+2}} ... \sum_{x_r} P(x_r) \cdot \prod_{(i,j) \in T \ i \ is \ j's \ parent} P(x_i \overset{t_{ij}}{\rightarrow} x_j)$$

(1) - rule of total probability.
(2) - chain rule and Markov property.
Recall that in previous lectures we defined the probability of $x_i$ to change to $x_j$ during the time $t_{ij}$:

$$P(x_i \xrightarrow{t_{ij}} x_j) = \left[ e^{t_{ij} \cdot R} \right]_{x_i, x_j}.$$  

Unfortunately, this formula of the likelihood is difficult to calculate since we have to sum over $n - 1$ variables, which can be exponential number. Notice that the evolutionary tree has the Markov property, meaning that a node $i$ in $T$ is independent of any other node $j$ in $T$, given its parent. Thus, we can use the Markov property (as in Forward Algorithm and Backward Algorithm) to split our likelihood problem to smaller problems, and reduce the number of calculations. That is what Up Algorithm is about.

### 2.2 Definitions

Let us define:

- $L_i$ - the leaves in the sub-tree rooted at $i$. So, $x_{L_i} = \{ x_j : j \in L_i \}$.
  
  - Note that: $L_i = L_j \cup L_k$, for $i$ parent of $j$ and $k$ only.

![Figure 1: The leaves in the sub-trees rooted at i, j and k.](image)

- We will define a matrix $U$ in the following way:
  
  - $U_i[a]$ - for each node $i$ and it’s value (sequence) $a$ we define the likelihood of the leaves in it’s sub-tree:

  $$U_i[a] \overset{def}{=} P(x_{L_i} \mid x_i = a)$$

  - By the Markov property we can further disassemble this expression:

2
\[ U_i[a] = P(x_L | x_i = a) = \sum_{b, c} P(x_L, x_j = b, x_k = c | x_i = a) = \]

\[ = \sum_{b, c} P(x_L, x_j = b, x_k = c | x_i = a) \]

\[ \text{chain rule} \]

\[ = \sum_{b, c} P(x_j = b | x_i = a) \cdot P(x_k = c | x_j = b, x_k = c | x_i = a) \]

\[ = \sum_{b} U_j[b] \cdot \sum_{c} P(x_k = c | x_j = b) \]

\[ \text{chain rule} \]

\[ = \sum_{b} U_j[b] \cdot \sum_{c} U_k[c] \cdot P(a \xrightarrow{ij} b) \cdot P(a \xrightarrow{ik} c) \]

\[ = \sum_{b} U_j[b] \cdot \sum_{c} U_k[c] \cdot \left( e^{ij \cdot R_{a,b}} \right) \cdot \left( e^{ik \cdot R_{a,c}} \right) \]

- Then, defining \( U_{ij}[a] = P(x_L | x_i = a) = \sum_{b} U_j[b] \cdot P(a \xrightarrow{ij} b) \), we have:

\[ U_i[a] = U_{ij}[a] \cdot U_{ik}[a] \]

\[ \bullet \] Notice, that if a node \( i \) is a leaf in \( T \), i.e. \( i \in \{1, \ldots, n\} = L_r \), then:

\[ U_i[a] = \begin{cases} 1 & \text{if } x_i = a \\ 0 & \text{else} \end{cases} \]

### 2.3 Reminder: Tree Traversal & Post-Order DFS

Tree traversal is the process of visiting each node in a tree data structure, exactly once. Such traversals are classified by the order in which the nodes are visited. We will use the **Post-order** (left, right, root). For example, given the following tree -

![Figure 2: A binary tree](image)

we will get the list: [3,4,2,6,7,5,1].
Algorithm 1 Post Order Depth First Search

0 order = []
1 function DFS_Post(i)
2 for j in children(i)
3 do DFS_Post(j)
4 order += i

2.4 The Up Algorithm

We build $U$ with dynamic programming:

Algorithm 2 Up

1 Initialization: order = DFS_post(r)
2 Loop: for i in order:
3 if leaf(i):
4 for a in $\Sigma$:
5 $U_i[a] = \begin{cases} 1 & x_i = a \\ 0 & \text{else} \end{cases}$
6 else:
7 for $a$ in $\Sigma$:
8 $U_i[a] = 1$
9 for $j$ in children(i):
10 $U_{ij}[a] = \sum_{b \in \Sigma} U_j[b] \cdot P(a \xrightarrow{t_{ij}} b)$
11 $U_i[a] = U_{ij}[a]$

Notes:

- Regarding the shape of $U_i$, it depends on the number of possible sequences $x_i$. Since we consider only sequences of length 1 ($x_i \in \Sigma$), then the length of $U_i$ is $|\Sigma| = 4$. Also, there are $2n - 1$ nodes in the tree, so we have $2n - 1$ rows and $|\Sigma|$ columns in $U$.

- Note that if we ignore the assumption that all the sequences are of length 1, we will have to consider each letter in the sequence, and calculate the likelihood of a single position. In such case, we denote by $P(x_k \xrightarrow{t} x_j)$ the probability of the $k'th$ letter in the $i'th$ sequence to change to the $k'th$ letter in the $j'th$ sequence, assuming independence between the positions.

- Using the Post-order traversal guarantees that when we get to the $i'th$ node, the values of it’s children, i.e. $U_j[b]$ and $U_k[b]$ for $b \in \Sigma$, were already calculated.

Run Time Analysis:

The initialization phase includes iterating over each node in the tree and therefore takes $O(n)$.
During the algorithm, we fill a matrix of size $O(n \cdot |\Sigma|)$. In addition, filling each cell takes $|\Sigma|$ steps, where the children’s cells were already assigned. So in total the algorithm’s run time is $O(n \cdot |\Sigma|^2) + O(n) = O(n)$, assuming $|\Sigma|$ is a constant.

2.4.1 The Likelihood

Now, we can calculate the likelihood of the leafs in the tree:

$$P(x_1, ..., x_n) = \sum_a P(x_1, ..., x_n, x_r = a) = \sum_a P(x_1, ..., x_n \mid x_r = a) \cdot P(x_r = a) = \sum_a U_r[a] \cdot \pi_a$$

where $\pi_a$ is the stationary distribution of $a$. 
2.4.2 The Posterior

Using Bayes’ theorem we can easily calculate the Posterior of the root:

\[
P(x_r = a \mid x_1, ..., x_n) = \frac{P(x_1, ..., x_n \mid x_r = a)}{P(x_1, ..., x_n)} \cdot P(x_r = a) = \frac{U_r[a]}{\sum_{b \in \Sigma} U_r[b] \cdot \pi_b} \cdot \pi_a
\]

Next class we will learn how to calculate the posterior of internal nodes, i.e., \( P(x_i = a \mid x_1, ..., x_n) \).

References