1 Up Algorithm

1.1 Probabilistic Model of Evolution

We would like to calculate the probability of the leaves of a given evolutionary tree. For instance for the following tree:

![Tree diagram]

\[ L = P(x_1, ..., x_5) = \sum_{x_{n+1} ... x_r} \left\{ \Pi_{i=1,...,r} P(x_i | x_{\text{parent}(i)}, t_{i,\text{parent}(i)}) \right\} \]

\[ *t_{i,\text{parent}(i)} \] denotes the time passed between a node and its parent in the tree.

Using the following diagram we’ll see how to find the above probability.

![Subtree diagram]

We’ll denote the set of leaves of the sub-tree whose root is \( x_i \) as \( \{x_{Li}\} \) and the set of \( x_i \)’s ‘children’ as \( N_i \).
We’ll notice that the above expression contains repetitive calculation and that we can simplify it by decomposing our problem into smaller ones. For each node’s value \( x_i = a \), we’ll find the probability of its leaves defined the following way:

\[
U_i[a] \triangleq P({x_{\ell_j}} | x_i = a)
\]

\[
U_i[a] = \prod_{l \in N_i} P({x_{\ell_l}} | x_i = a) = \prod_{l \in N_i} \left( \sum_{b \in \Sigma} P({x_{\ell_l}}, l = b | x_i = a) \right) = \\
= \prod_{l \in N_i} \left( \sum_{b \in \Sigma} P({x_{\ell_l}} | x_i = a) P({x_{\ell_l}}, l = b) \right) = \\
= \prod_{l \in N_i} \sum_{b \in \Sigma} U_l[b] P(\ell_l \rightarrow b) \text{ if } i \text{ has two children } j, k \\
= \left[ \sum_{b \in \Sigma} U_j[b] P(\ell_j \rightarrow b) \right] \left[ \sum_{b \in \Sigma} U_k[b] P(\ell_k \rightarrow b) \right]
\]

Notice that we can compute \( U_i[a] \) in linear time with dynamic programming.

1.2 Reminder: DFS Algorithm and Tree Traversal Order

![Image of a tree](image)

Given the above tree, let us recall three common orders for depth first search (DFS):

Post-order (left, right, root): \([4,5,2,6,7,3,1]\)
Pre-order (root, left, right): \([1,2,4,5,3,6,7]\)
In-order (left, root, right): \([4,2,5,1,6,3,7]\)

**Algorithm 1** Post Order Depth First Search

1: function DFS_Post(i):
2:     for each j in \( N_i \)
3:         DFS_Post(j)
4:     order.append(i)
1.3 The Up Algorithm

Algorithm 2 Up

1: function Up_Alg(r):
2: initialize: order = DFS_Post(r)
3: for each i in order do
4: if leaf(i):
5: for a in Σ:
6: \[ U_i[a] = (x_i = a)? 1: 0 \]
7: else:
8: for a in Σ:
9: \[ U_i[a] = \prod_{j \in N_i} \sum_{b \in \Sigma} U_j[b] P^{(e^{tL})} (a \overset{t}{\rightarrow} b) \]

*During previous lectures we saw that the probability for ‘a’ to change to ‘b’ over the period of time between the two events \( P(a \rightarrow b) = [e^{tR}]_{a,b} \)

Now we can calculate the likelihood of a tree:

\[ L = P(x_1, ..., x_n) = \sum_{b \in \Sigma} P(x_1, ..., x_n | x_r = b)P(x_r = b) = \sum_{b \in \Sigma} U_r[b]P(x_r = b) \]

Run Time Analysis: We initialize the order array with \( O(n) \). Next, we iterate over each node once \( (O(n)) \), and perform \( O(|\Sigma|^2) \) calculations during the iteration. In total - \( O(n \cdot |\Sigma|^2) + O(n) = O(n) \)

Clarification: Throughout this calculation we have assumed that the sequences are of length 1. For a calculation of sequences of greater length - m, the above expression would represent the likelihood of a single position and the overall likelihood will be provided by: \( \Pi_{j=1}^m L_j \) (this however assumes independence between the positions)

1.4 The Posterior:

We would now like to calculate the posterior probability of the tree:

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

*We’ll denote \( P(x_r = b) = \pi_b \). It is the stationary distribution over x, and we’ll discuss it further next.
2 Continuous Markov Chains (MC) Properties:

We will discuss two main properties continuous MC may possess:

- Ergodicity
- Reversibility

Let us recall the Jukes-Cantor model where:

$$R_{JC} = \begin{bmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{bmatrix}$$

$$P_{JC}(t) = \begin{cases} \frac{1}{4} (1 + 3e^{-4\alpha t}) & i = j \\ \frac{1}{4} (1 - e^{-4\alpha t}) & i \neq j \end{cases}$$

$$P_{JC}(0) = I \quad P_{JC}(\infty) = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

We will introduce two matrices properties – irreducibility and aperiodicity.

A MC is **irreducible** if $$\forall a, b \exists t \text{ s.t. } P_t(a \rightarrow b) > 0$$ (e.g. $$P_{JC}$$)

An example to a reducible matrix is:

$$\begin{bmatrix} -\alpha & \alpha & 0 & 0 \\ \alpha & -\alpha & \alpha & 0 \\ 0 & -\alpha & -\alpha & \alpha \\ \alpha & \alpha & 0 & -\alpha \end{bmatrix}$$

This matrix is **not** irreducible since if for instance we start at A, we cannot reach G.

A MC is **aperiodic** if $$\forall a, b \text{ gcd}\{t | P_t(a \rightarrow b) > 0\} = 1$$ (e.g. $$P_{JC}$$)

An example to a periodic matrix is:

$$\begin{bmatrix} -\alpha & \alpha & 0 & 0 \\ 0 & -\alpha & \alpha & 0 \\ 0 & 0 & -\alpha & \alpha \\ \alpha & 0 & 0 & -\alpha \end{bmatrix}$$

This matrix is **not** aperiodic since if we start at A then only when t is a multiplicity of 4 we can reach A again.
2.1 Ergodicity

A MC is ergodic if it is irreducible and aperiodic: \( \forall a, b, t \ P_t(a \rightarrow b) > 0 \) (e.g. \( P_{JC} \))

**Theorem:**

If MC is ergodic then \( \exists \pi \) s.t. \( P(t) \xrightarrow{t \to \infty} \begin{bmatrix} \pi_b & \pi_c & \pi_d & \pi_e \\ \pi_b & \pi_c & \pi_d & \pi_e \\ \pi_b & \pi_c & \pi_d & \pi_e \\ \pi_b & \pi_c & \pi_d & \pi_e \end{bmatrix} \)

\( \pi_b \equiv P(\infty + \mathcal{E})_{a,b} = \Sigma_c P(\infty)_{a,c} P(\mathcal{E})_{c,b} = \Sigma_c \pi_c P(\mathcal{E})_{c,b} = \pi P(\mathcal{E})_{:,b} \)

Or when viewed as matrices:

\( \overline{\Pi} = \Pi P(\mathcal{E}) \) and \( \overline{\Pi}^T = P(\mathcal{E})^T \overline{\Pi} \)

Meaning that \( \overline{\Pi} \) is the eigenvector of the transitions matrix at time \( \mathcal{E} \) with eigenvalue of 1.

We can now see that:

\( \overline{\Pi}^T = P(\mathcal{E})^T \overline{\Pi} = (I + \mathcal{E}R)^T \overline{\Pi}^T = \overline{\Pi}^T + \mathcal{E}R^T \overline{\Pi} \Rightarrow R^T \overline{\Pi}^T = 0 \)

We have reached a fixed point, and \( \overline{\Pi}^T \) is indeed stationary.

The Perron-Frobenius theorem promises the existence of such \( \Pi \).

2.2 Reversibility

We would like to be able to discuss unrooted trees. In order to do so, we wish to be able to ‘lift’ any point C such that \( C = t_1 + t_2 \):

In order to do so we will require our MC matrix to be **reversible**.

**Definition:** a matrix is reversible iff

\( \forall a, b, t \ \pi_a P(t)_{a,b} = \pi_b P(t)_{b,a} \)
\[
\Sigma_c \pi_c P(t_1)_{c,a} P(t_2)_{c,b} \overset{rev}{=} \Sigma_c \pi_a P(t_1)_{a,c} P(t_2)_{c,b} = \pi_a \Sigma_c P(t_1)_{c,a} P(t_2)_{c,b} =
\]
\[
= \pi_a \Sigma_c P(t_1 + t_2)_{a,b} = \pi_a P(t)_{a,b} \overset{rev}{=} \pi_b P(t)_{b,a}
\]

Meaning it does not matter if we root at a and move towards b, root at b and move towards a or set the root as some point c that divides t to \( t_1 \) and \( t_2 \).