Aperitifs

From previous lesson

Run time complexity of the Up algorithm: $O(n \cdot |\Sigma|^2)$
DFS runs in linear time

The DFS post-order algorithm

\begin{algorithm}
\caption{Depth First Search post order}
\begin{algorithmic}[1]
\Function{DFS\_POST}{i} \Comment{operates on a global variable \textit{order}}
\For{each $j \in \text{children}(i)$}
\State \text{DFS\_POST}(j)
\EndFor
\State \text{order} $\gets$ \{\text{order}, $i$\} \Comment{adds the current node at the end of the list.}
\State \Return
\EndFunction
\end{algorithmic}
\end{algorithm}

In this lesson

We continue the search for a good algorithm that performs ancestral reconstruction in reasonable time.

Reminder:

\[ U_i[a] = Pr(X_{L_i} | X_i = a) \]
is the likelihood of the subset of leaves $X_{L_i}$ given that the ancestor $i$ (an internal node) will obtain the value $a$. See figure 1.

\[
ri = 2n - 1
\]

Total probability on the tree

\[
P(X_1 \ldots X_n) = \sum_{a \in \Sigma} P(X_r = a) \cdot U_r[a]
\]

where $r$ is the element at the root, $\Sigma$ is the alphabet, and $\Pi_a$ is the prior probability distribution.

**Posterior probability and the Up-Down algorithm**

We have $U_r[a] = \Pr(X_r | X_r = a)$, and we’d like to calculate the probability that the root node contains letter $a$:

\[
\Pr(X_r = a | \overline{X}) = \frac{U_r[a] \cdot \Pi_a}{P(\overline{X})} = \frac{U_r[a] \cdot \Pi_a}{\sum_b U_r[b] \cdot \Pi_b}
\]
Working through the joint probability

\[
\Pr(\vec{X}, X_r = a) = \underbrace{\Pr(\vec{X} | X_r = a)}_{U_r[a]} \cdot \Pr(X_r = a) \underbrace{\Pi_a}_{\Pi_a}
\]

Is the posterior probability good for any internal node in the tree? In order to work this out, we note that the tree could be rooted at any internal node, and then we could perform the posterior calculation with the new position of \( r \). The shift of \( r \) is illustrated in figure 2. The reader may appreciate that in order to recalculate the posterior probability, the time dimension may be reversed in some of the nodes: on panel 2(a) the time counts forward as we descend the tree, and on panel 2(b) it is not.

![Figure 2: A tree “hung” at two different roots. The broken line on the left panel marks the element and two immediate children that transform to the new root. The blue arrows show paths that counter the downward time direction.](image)

**Reversibility of a Markov Process**

The reversibility is an abstract property of the Markov process in the sense that it cannot exist in real life situations. The most intuitive explanation is that reversing the time direction may not be plausible for certain species that underwent an evolutionary modification following a specific change in environmental conditions.

In order to reposition the root of the tree (“hang” it from an internal node), we must relax the requirement that time must advance in one direction. Consider the following situation of a three-node tree. The leaves \( a \) and \( b \) under the root \( c \). The distance between \( a \) and \( b \) is accumulated as we traverse the edges \((a, c)\) and \((c, b)\). If the time between \( c \) and \( a \) was \( t_1 \), and the time \( t((c, b)) = t_2 \)
then \( t_1 + t_2 = t \) is the distance between \( a \) and \( b \). Formally, the reversibility property can be written as

\[
\begin{align*}
\Pi_a \cdot \Pr(a \xrightarrow{t} b) &= \Pi_b \cdot \Pr(b \xrightarrow{t} a) = \\
[\exp[tR]]_{ab} &= [\exp[tR]]_{ba} \\
&= \sum_{c \in \Sigma} \Pi_c \cdot \Pr(c \xrightarrow{t_1} a) \cdot \Pr(c \xrightarrow{t_2} b)
\end{align*}
\]

The equal sign here is due to the reversibility property of the Markov process, that allows us to flip the direction of the transition.

This is equivalent to the following visual transformation:

![Figure 3: display of the rearrangement procedure of the tree in figure 1. In this procedure we “hang” the tree by the edge between the nodes \( i \) and \( l \). \( l \) being the former parent of \( i \) (cf. figure 1). The labeled nodes retain their respective positions and the black, blue and red boxes give the same subsets \( X_{Li}, X_{Lj} \) and \( X_{Lk} \).](image)

**Detailed Balance Property**

We consider the equality of the likelihood and its reverse along the time axis:

\[
\Pi_a \Pr(a \xrightarrow{t} b) = \Pi_a \cdot \Pr(a \xrightarrow{t} b) \quad \forall a, b, t
\]

A matrix \( R \) will define a reversible Markov process if it satisfies the condition (1) above. Let us write the Taylor expansion of this expression for \( a \neq b \) and an infinitesimally small time step \( t = \mathcal{E} \), up to the second term:

\[
\Pr(\mathcal{E}) \approx I + \mathcal{E}R + o(\mathcal{E})
\]
Applying the constraint $a \neq b$ (the elements on the diagonal do not affect the process) we get

$$\Pi_a \mathcal{E} R_{ab} = \Pi_b \mathcal{E} R_{ba}$$

$$\Rightarrow \frac{\Pi_a}{\Pi_b} = \frac{R_{ba}}{R_{ab}}$$

(3)

this way we converted the detailed balance property to a measurable that could be verified by inspecting matrix $R$.

**Putting it all together**

Using the detailed balance property we will now calculate the posterior probability, which is the likelihood of encountering the set of leaves $L_i$ under ancestor $i$ given that the letter at $i$, is $a$. Formally

$$\Pr \left( \bar{X} | X_i = a \right) = \prod_{j \in N(i)} U_{ij}[a]$$

where $N(i)$ are the neighbors of $i$ and $U_{ij} = \Pr(X_{L_j} | X_i = a)$. This may be achieved by implementing an Up-Down algorithm. A visual sketch of the ‘Up-Down’ workings can be found in figure 4.

Figure 4: The Up-Down algorithm starts at node $i$ and recursively collects the likelihoods in subtrees. The first collected element is marked ‘1’, second collected element is ‘2’. Blue color is the progress in the ‘Down’ direction, while the ‘Up’ direction is in red. Some red arrows were not plotted to prevent overcrowding of the plot.

The algorithm starts off by ordering the nodes in a pre-order DFS (descending the tree: the ‘Down’ part). Therefore the leaves are listed first. Then, it recursively traverses upwards toward the (new) root $i$, while accumulating the likelihoods of the leaves (the data) in all subtrees under any internal
node along the way to $i$. This is in the blue arrows. When the upward traversal ends, all the likelihoods of any subtree under $i$ are collected. Now, the traversal continues downstream from $i$, thus collecting likelihoods of the data under any subtree of the full graph (that may also contain the subtrees of $i$). The downstream movement is marked with red arrows.

The algorithm listing will be given in the beginning of the next lesson.