Algorithmic Bioinformatics DD2450, spring 2010, Lecture 12

Lecturer Jens Lagergren
Several current and previous students
will be acknowledged in a separate document.

May 22, 2010

Chapter 1

Lecture 10

1.1 Algorithm for the Small ML Problem

We abstract "away" the model and use trees with edges labeled with transition matrices. Following is an example of a transition matrix

$$\begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} 1/2 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/2 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/2 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/2 \end{pmatrix} \end{array}$$

Input a leaf labeled tree T, l, a root distinction ρ and for each edge e a transition matrix $M(e)$

Output the probability that T, ρ, M generates l, i.e. $Pr[l|T, \rho, M]$

We have

$$Pr[l|T, \rho, M] = \sum_{\substack{l' = U(T) \to \Sigma \\ l'|L(T) = l}} Pr[l'|T, \rho, M]$$

Idea conditioning and dynamic programming

In this problem positions are independent and identically distributed, so we can compute the probability of one position at a time and then multiply those

$$P(T, M, \sigma, e) = \sum_{\substack{l' = V(T) \to \Sigma \\ l'|L(T) = l \\ l'(\text{root}(T)) = \sigma}} Pr[l'|T, \rho, M]$$

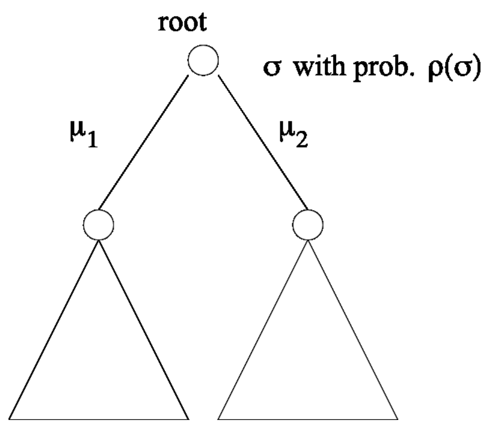

Figure 1.1: A tree

Let us define the counter

$$c(u, \sigma) = P(T_u, M|_{E(T_u)}, \sigma, l|_{L(T_u)})$$

Now we do a recursion for c

- For $V \in L(T)$ (base case)

$$c(u, \sigma) = \begin{cases} 1 & \text{if } \sigma = \rho(u) \\ 0 & \text{otherwise} \end{cases}$$

- For $u \in U(T) \backslash L(T)$ (internal vertex u), summing over mutually exclusive events that cover the entire space

$$c(u, \sigma) = \sum_{\sigma_v, \sigma_w \in \Sigma} [M(u, v)_{\sigma\sigma_v} c(v, \sigma_v) M(u, w)_{\sigma\sigma_w} c(w, \sigma_w)]$$

  We write this as separate sum for faster computation

$$c(u, \sigma) = \left(\sum_{\sigma' \in \Sigma} M(u, v)_{\sigma\sigma'} c(v, \sigma') \right) \left(\sum_{\sigma' \in \Sigma} M(u, w)_{\sigma\sigma'} c(w, \sigma') \right)$$

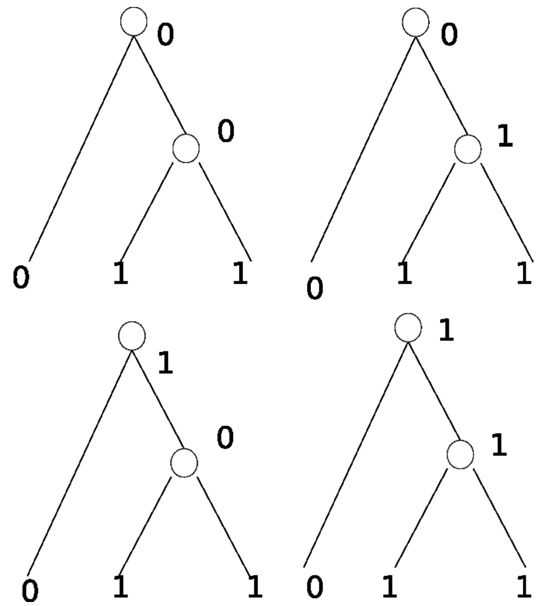

Figure 1.2: Probability computations

Final answer is given by

$$Pr[l|T, \rho, M] = \sum_{\sigma \in \Sigma} \rho(\sigma) - c(\text{root}(T), \sigma)$$

Time complexity is given as below

- Time complexity for one position $O(|v(T)||\Sigma|^2)$
- Time complexity for m positions $O(|v(T)||\Sigma|^2 m)$

1.2 Algorithm for the Medium ML Problem

Framework

- A rooted tree T with edge lengths λ and leaf labeling l.
- An alphabet Σ.
- A model gives a mapping $M : R^+$ to $|\Sigma| \times |\Sigma|$ -matrices (i.e. transitions matrices)

so for an edge e, $M(\lambda(e))$ is its transition matrix.
We know how to compute $\Pr[l|T, \lambda]$

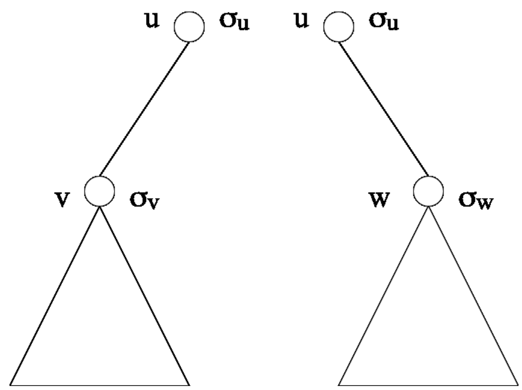

Figure 1.3: Rooted subtrees

Medium ML-problem

Input a leaf labeled tree T, l.

Output edge lengths λ that maximize $\Pr[l|T, \lambda]$.

Heuristic

1. Pick reasonable or random initial edge lengths
2. Until no edge length is altered
   (a) pick an edge e
   (b) Modify $\lambda(e)$ such that

$$\Pr[l|T, \lambda] \quad \text{is maximized}$$

How do we perform (b)? Notice our models are reversible and, therefore, we can use any vertex as the root (see figures 1.4, 1.5 and 1.6).

Assume that $e = (u, v)$. Then:

1. Make u the root. Let T^u be $T \setminus T_v$ (see 1.6).
2. For each position i and $\sigma \in \Sigma$ compute the probability for σ in position i in T^u and σ in position i in T_v (see 1.6).
3. Optimize $\lambda(e)$ without recomputing anything in T^u or T_v, using a more or less advanced numerical optimization procedure.

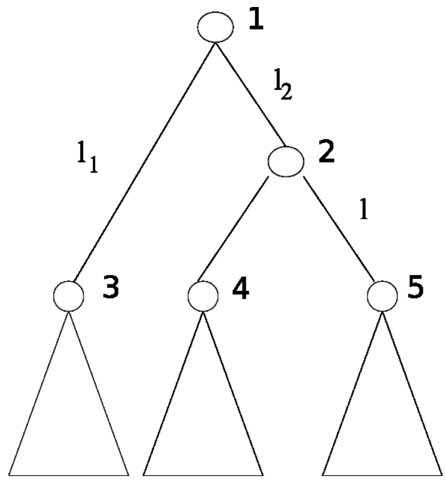

Figure 1.4: Tree with node no. 1 as root (see figure 1.5)

1.3 Algorithm for the Big ML Problem

Input Sequences $s_1, \ldots, s_n$ all of length m.

Output A leaf labeled tree T, l (labeled with $s_1, \ldots, s_n$) and edge lengths λ that maximize $\Pr[l|T, \lambda]$ among all such T, l, λ.

Heuristic

1. Let T, l, λ be the result of applying "the medium algorithm" to the NI tree or a random tree.
2. Until no better trees are found:

   For every edge e:

   (a) Do a branch swap (see figure 1.7) to get a possibly better tree.

   (b) Optimize edge lengths. First for a, b, c, d and e, then the rest of the edges.
3. We let the best tree T_e, l_e, λ_e be the current tree and continue.

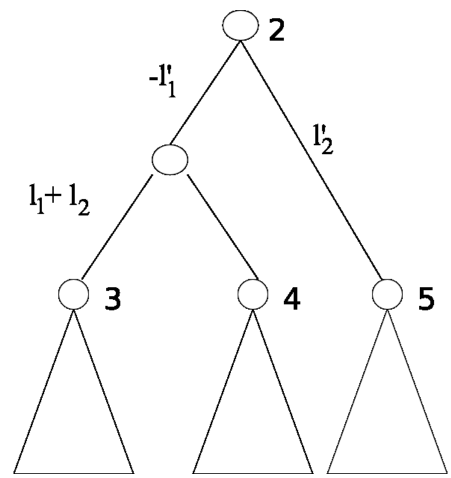

Figure 1.5: Tree with node no. 2 as root (see figure 1.4)

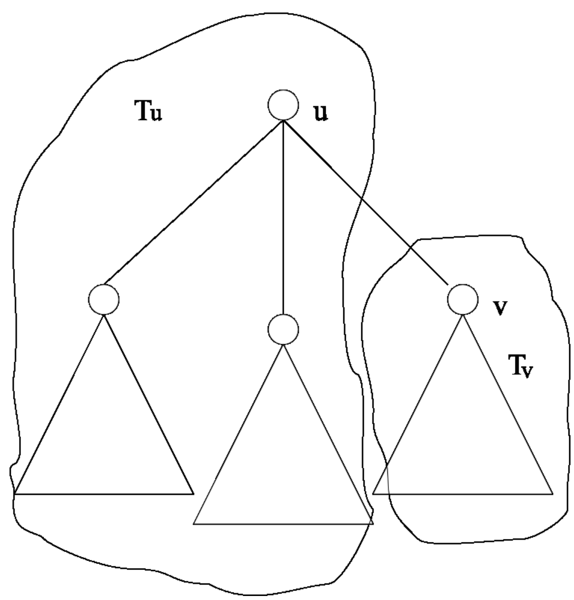

Figure 1.6: Changing root: tree with root u and a subtree with root v, we may now select v as root for the tree when u will become root of a subtree (see figures 1.4 and 1.5)

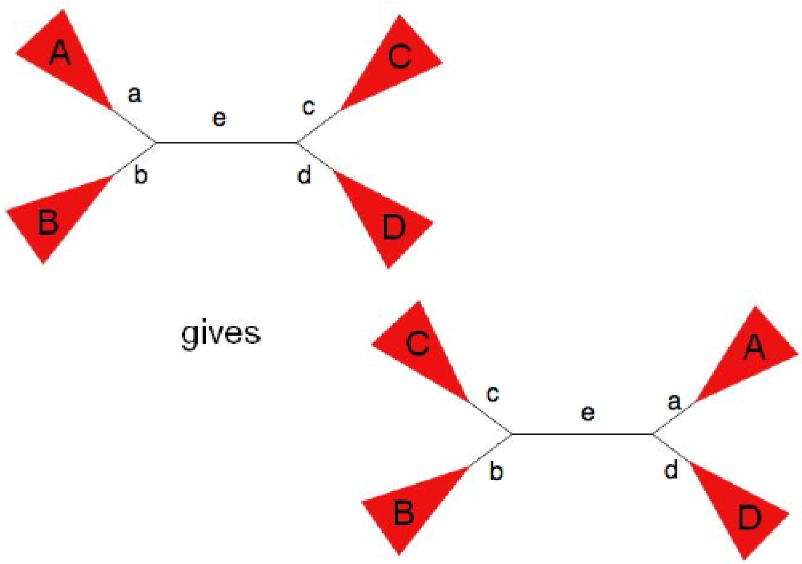

Figure 1.7: Branch Swap