6.047/6.878 Fall 2008 - Recitation 6 - Maximum Likelihood

October 10, 2008

1 Model of evolution

Our goal in developing this maximum likelihood approach is to find the best tree (most likely) that explains our sequence data.

To represent a tree, let the leaves of a tree be numbered $1, ..., n$ and the ancestral nodes be $n+1, 2n-1$. Let the branches of the tree be numbered by the most recent of the two nodes it touches (e.g. branch i connects node i and $parent(i)$). For a tree, we have its topology T and the branch times $t_1, ..., t_{2n-2}$, where t_i is the time between nodes i and $parent(i)$.

Our sequence data can be represented as a matrix $\mathbf{x}$ (n rows, m columns), such that $x_{i,j}$ is the j^{th} character of the i^{th} sequence. We will be given sequence data for the extant (modern) sequences $\mathbf{x_1}, ...\mathbf{x_n}$, and will have to integrate over the ancestral sequences $\mathbf{x_{n+1}}, ...\mathbf{x_{2n-1}}$. Each sequence has length m.

With these definitions, our goal in the maximum likelihood method is to solve the following equation

$$\arg\max_{T,\mathbf{t}} P(\mathbf{x_1}, ..., \mathbf{x_n}|T, \mathbf{t}).$$

1.1 Defining the distributions: factoring by branches

Before we can tackle the equation above, we must first define the distributions of our variables. We will make several assumptions about the process of sequence evolution in order to make the math and algorithm tractable.

First note that the distribution above is a marginal of the joint distribution over all sequences

$$P(\mathbf{x_1}, ..., \mathbf{x_n}|T, \mathbf{t}) = \sum_{\mathbf{x_{n+1}}, ..., \mathbf{x_{2n-1}}} P(\mathbf{x_1}, ..., \mathbf{x_{2n-1}}|T, \mathbf{t})$$

The first assumption we will make is that **lineages evolve independently**. This means that a sequence $\mathbf{x_i}$ only depends on the sequence its parent $\mathbf{x_{parent(i)}}$ and the time along it branch t_i. This assumption allows us to factor the distribution as

$$\begin{aligned} P(\mathbf{x_1}, ..., \mathbf{x_{2n-1}}|T, \mathbf{t}) &= P(\mathbf{x_1}|\mathbf{x_2}, ..., \mathbf{x_{2n-1}}, T, \mathbf{t})P(\mathbf{x_2}|\mathbf{x_3}, ..., \mathbf{x_{2n-1}}, T, \mathbf{t})...P(\mathbf{x_{2n-1}}|T, \mathbf{t}) \\ &= P(\mathbf{x_1}|\mathbf{x_{parent(1)}}, t_1)P(\mathbf{x_2}|\mathbf{x_{parent(2)}}, t_2)...P(\mathbf{x_{2n-1}}) \\ &= P(\mathbf{x_{2n-1}}) \prod_{i=1}^{2n-2} P(\mathbf{x_i}|\mathbf{x_{parent(i)}}, t_i) \end{aligned}$$

Notice that this equation requires us to define only two things, a prior distribution of the root sequence $P(\mathbf{x_{2n-1}})$ and the distribution of how single branches evolve $P(\mathbf{x_i}|\mathbf{x_{parent(i)}}, \mathbf{t})$. The following sections will define these.

1.2 Modeling the evolution of a single branch

We now consider the evolution along a single branch

$$P(\mathbf{x_i}|\mathbf{x_{parent(i)}}, t_0).$$

We will make another assumption, assume **sites evolve independently**. Therefore we have

$$P(\mathbf{x_i}|\mathbf{x_{parent(i)}}, t_i) = \prod_j P(x_{i,j}|x_{parent(i),j}, t_i)$$

Previously in the presentation of the Neighbor-Joining algorithm, we saw several models for the evolution of a single site $P(x_{i,j}|x_{parent(i),j}, t_i)$. Let us now review them.

Jukes Cantor (JC) is the simplest but there are also many others: Kimura's 2 parameters (K2P), HKY, etc. All of these models make the same basic set of assumptions, and only differ in how many free parameters they have.

First they all assume that the evolutionary process is **time reversible**. Thus the probability of an "A" evolving to a "G" over time t is the same as a "G" evolving to an "A". In other words

$$P(a = G|b = A, t) = P(a = A|b = G, t).$$

You can think of this a symmetric 4x4 matrix S. They also assume that this probability matrix is **multiplicative**

$$\sum_b P(b|a, t_1)P(c|b, t_2) = P(c|a, t_1 + t_2).$$

In the Jukes Cantor model, every base mutates into every other base with an equal rate α. From these properties, our substitution probability matrix S is then

$$P(a|b, t) = S(t) = \begin{pmatrix} r_t & s_t & s_t & s_t \\ s_t & r_t & s_t & s_t \\ s_t & s_t & r_t & s_t \\ s_t & s_t & s_t & r_t \end{pmatrix},$$

where

$$r_t = \frac{1}{4}(1 + 3e^{-4\alpha t})$$
$$s_t = \frac{1}{4}(1 - e^{-4\alpha t}).$$

The derivation of these equations are given in the Durbin book (pg 194). Try plugging in a few values of t to see how this makes sense

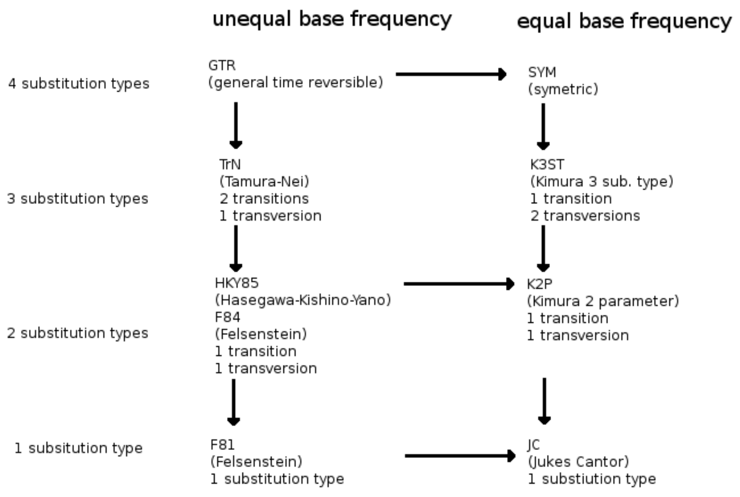

Figure 1. The hierarchy of sequence evolution models

$$t = 0$$
$$r_0 = \frac{1}{4}(1 + 3e^{-4\alpha 0}) = 1$$
$$s_0 = \frac{1}{4}(1 - e^{-4\alpha 0}) = 0,$$

which is the identity matrix (i.e. no substitutions occur in zero time). Also we have

$$t = \infty$$
$$r_\infty = \frac{1}{4}(1 + 3e^{-4\alpha\infty}) = \frac{1}{4}$$
$$s_\infty = \frac{1}{4}(1 - e^{-4\alpha\infty}) = \frac{1}{4},$$

which means in the long-run any sequence evolves towards the equilibrium of 25% frequency for each base.

See Figure 1 for more examples of sequence evolution models. For example, the K2P model allows another parameter to model the different rates of transversions and transitions. The HKY model allows three additional parameters on top of K2P to model a different equilibrium distribution than 25% for each base. The TrN model generalizes HKY by giving two parameters for the two kinds of transitions. And lastly there is the fully parametrized model General Time Reversible (GTR).

1.3 Modeling the root sequence

The last distribution that needs to be defined is the probability of the root sequence

$$P(\mathbf{x_{2n-1}})$$

We will make a very simple model for this sequence. First, we assume sites evolve independently, and second the probability of each base is modeled by a background base frequency $P(a)$. Thus we have

$$P(\mathbf{x_{2n-1}}) = \prod_{j=1}^{m} P(x_{i,j})$$

Just to be explicit, if we wanted each base to be equally likely $(1/4)$, we would have

$$P(\mathbf{x_{2n-1}}) = \left(\frac{1}{4}\right)^m$$

2 The Maximum Likelihood (ML) Algorithm

Now that we have fully defined our model we can estimate the maximum likelihood of the parameters T and $\mathbf{t}$.

First consider how many topologies there are for n leaves. For unrooted topologies we have

$$N_u = 3 * 5 * 7 * ... * (2n-5) = (2n-5)!!$$

and for rooted topologies we have

$$N_r = (2n-3) * N_u = (2n-3) * (2n-5)!!$$

There is no known algorithm for solving the general ML phylogeny problem in polynomial time. Therefore, current approaches use heuristic tree searches to explore only a fraction of these topologies and possible values for $\mathbf{t}$. There are several search methods that rely on taking a tree and performing local rearrangements to propose other trees (e.g. Nearest Neighbor Interchange (NNI) and Subtree Pruning and Regrafting (SPR)). The basic ML algorithm is then

```
while proposing values of T, t with NNI:
    Calculate likelihood of P(x_1,...,x_n|T,t)
return T,t that achieved max likelihood
```

The previous section defined how to calculate $P(\mathbf{x_1}, ..., \mathbf{x_n}|T, \mathbf{t})$, but can we do it efficiently. The answer is yes, and the solution is with dynamic programming. Remember that the full factoring of the likelihood looks like this

$$\begin{aligned} P(\mathbf{x_1}, ..., \mathbf{x_n}|T, \mathbf{t}) &= \prod_j P(x_{1,j}, ..., x_{n,j}|T, \mathbf{t}) \\ &= \prod_j \sum_{x_{2n-1,j}} ... \sum_{x_{n+1,j}} P(x_{2n-1,j}) \prod_{i=1}^{2n-2} P(x_{i,j}|x_{parent(i),j}, t_i) \end{aligned}$$

which appears to be an expensive calculation. However, many of these terms are repeated reused and can memoized with DP. The algorithm is called the "peeling" algorithm and is performed by filling the following DP table of likelihoods $P(L_{i,j}|a)$.

$$P(\mathbf{x_1}, ..., \mathbf{x_n}|T, \mathbf{t}) = \prod_j \sum_a P(L_{2n-1,j}|a)P(a)$$

$$P(L_{i,j}|a) = \begin{cases} 1 & \text{if } x_{i,j} = a, i \le n \\ 0 & \text{if } x_{i,j} \ne a, i \le n \\ \sum_{b,c} P(b|a, t_{left(i)})P(L_{left(i)}|b) & \text{if } i > n \\ P(c|a, t_{right(i)})P(L_{right(i)}|c) & \end{cases}$$

You can think of $P(L_{i,j}|a)$ as the likelihood the subtree rooted at node i given that the root contains the base a at site j. The table is computed in post-traversal order (leaves to the root) and the values at the top of the tree give the final likelihood of the whole tree.

The runtime of calculating the likelihood is thus fairly fast $O(nm)$. On the other hand, the tree search is difficult to bound and is the most expensive part of ML.