Bioinformatics

Richard Bonneau
Lecture 6: Probabilistic phylogenetic trees.
Associated reading.

- **Ch. 8 BSA**

- **MAIN:**
  Large punctual contribution of speciation to evolutionary divergence. Science 314:2006, p. 119

- **Optional:**
Making trees from pairwise distances

<table>
<thead>
<tr>
<th>d</th>
<th>1:</th>
<th>2:</th>
<th>3:</th>
<th>4:</th>
<th>5:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2:</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3:</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4:</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5:</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

1: AGCTTC–TA
2: ACGTTCTTA
3: AGCTTATTA
4: TCCTATTTA
5: TCCTTATTA

Where distance is number of mismatches
Neighbor joining:

We still assume additivity, we still use a deterministic joining algorithm, but we redefine distance and the algorithms slightly to better deal with variable branch lengths.

We calc a distance $D$, where $d$ is corrected by mean path to other nodes, $r$. (where $L$ is number of leaves)

Algorithm:

0. leaf nodes $\rightarrow$ L, we’ll grow tree from this set, L
1. Pick $\text{argmin}_{ij}(D_{ij})$ and make node k joining i and j
2. Calc distance from k to all other nodes
3. Add k to growing tree
4. Remove i and j from node list (now they are represented by k
5. Rinse, lather, repeat.

\[
d_{ij} = -\frac{3}{4} \log \left(1 - \frac{4f}{3}\right)
\]

\[
D_{ij} = d_{ij} - (r_i + r_j)
\]

\[
r_i = \frac{1}{|L| - 2} \sum_{k \in L} d_{ik}
\]

\[
d_{k.} = \frac{1}{2} \left( d_{i.} + d_{j.} - d_{ij} \right)
\]
parsimony

Evaluate cost of tree: Given tree + Given alignment.

1. AGTCTACGGATTAT
2. AGTTTTACGGATTAT
3. TGTTTTACGGATTAT
4. AGTGTTACGGATTAA
5. TGTGTTACGGATTAA

1. Set $S_k(a) = 0$ for $a = x$, infinity otherwise for leaf nodes.
2. If not at leaf node (internal vertex):
   
   $S_k(a) = \min_b (S_i(b) + S(a,b)) + \min_b (S_j(b) + S(a,b))$

Here $S(a,b) = 0$ for match, 1 for mismatch.
Evaluate cost of tree: **Given tree + Given alignment.** With variable branch lengths. We aim to maximize $P(\text{tree}|\text{Data})$

1: AGTCTACGGATTAT
2: AGTTTACGGATTAT
3: TGTTTACGGATTAT

We want to calc:

$P(x'|T,t_\cdot)$

Where:

$x' = \text{sequences}$

$T = \text{the tree topology}$

$t_\cdot = \text{the branch lengths}$

$$P(x^1,x^2,x^3,x^4,x^5|T,t_\cdot) = P(x^1|x^4,t_1)P(x^2|x^4,t_2)P(x^3|x^5,t_3)P(x^4|x^5,t_4)P(x^5)$$

How do we calc:

$$P(x|y,t)$$
**S(t)**

Positional independence:

1: AGTCTACGGATTAT
2: AGTTTACGGATTAT
3: TGTTTACGGATTAT

Product over positions to calc

\[ P(x | y, t) = \prod_{i} P(x_i | y_i, t) \]

Multiplicative with respect to time:

\[ S(t_1 + t_2) = S(t_1)S(t_2) \]
\[ P(a | b, t_1)P(b | c, t_2) = P(a | c, t_1 + t_2) \]
S(t) ... rates

All substitutions equal

\[
R = \begin{pmatrix}
-3\alpha & \alpha & \alpha & \alpha \\
\alpha & -3\alpha & \alpha & \alpha \\
\alpha & \alpha & -3\alpha & \alpha \\
\alpha & \alpha & \alpha & -3\alpha
\end{pmatrix}
\]

Transitions and Transversions at different rates

\[
R = \begin{pmatrix}
-2\beta - \alpha & \alpha & \alpha & \beta \\
\alpha & -2\beta - \alpha & \beta & \alpha \\
\alpha & \beta & -2\beta - \alpha & \alpha \\
\beta & \alpha & \alpha & -2\beta - \alpha
\end{pmatrix}
\]

We need to design a substitution matrix that is time dependent.

We define expected rates as matrix R.
For small $t$

$$S(t) \approx (I + R\varepsilon) =$$

$$
\begin{pmatrix}
1 - 3\alpha \varepsilon & \alpha \varepsilon & \alpha \varepsilon & \alpha \varepsilon \\
\alpha \varepsilon & 1 - 3\alpha \varepsilon & \alpha \varepsilon & \alpha \varepsilon \\
\alpha \varepsilon & \alpha \varepsilon & 1 - 3\alpha \varepsilon & \alpha \varepsilon \\
\alpha \varepsilon & \alpha \varepsilon & \alpha \varepsilon & 1 - 3\alpha \varepsilon \\
\end{pmatrix}
$$
\[ S(t) \]

Solving for \( s(t) \)

\[
\frac{\partial(S(t))}{\partial t} = S(t)R
\]

\[
S(t) = \begin{pmatrix}
  r_t & s_t & s_t & s_t \\
  s_t & r_t & s_t & s_t \\
  s_t & s_t & r_t & s_t \\
  s_t & s_t & s_t & r_t
\end{pmatrix}
\]

\[ r_t = \frac{1}{4} \left( 1 + 3e^{-4\alpha t} \right) \]

\[ s_t = \frac{1}{4} \left( 1 + e^{-4\alpha t} \right) \]

As \( t \to \infty \), \( r_t = s_t = 0.25 \)
Solving for $s(t)$

$$S(t) = \begin{pmatrix} r_t & s_t & s_t & s_t \\ s_t & r_t & s_t & s_t \\ s_t & s_t & r_t & s_t \\ s_t & s_t & s_t & r_t \end{pmatrix}$$

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

$$s_t = \frac{1}{4} (1 + e^{-4\alpha t})$$
Two sequences

The $P(x|T,t)$ is root invariant for two gene case.

\[
\begin{align*}
P(x_u^1, x_u^2, a | T, t_1, t_2) &= q_a P(x_u^1 | a, t_1) P(x_u^2 | a, t_2) \\
P(x_u^1, x_u^2 | T, t_1, t_2) &= \sum_a q_a P(x_u^1 | a, t_1) P(x_u^2 | a, t_2) \\
P(x^1, x^2 | T, t_1, t_2) &= \prod_u P(x_u^1, x_u^2 | T, t_1, t_2) \\
P(x^1 = C, x^2 = C | T, t_1, t_2) &= q_C r_{t_1} r_{t_2} + q_A s_{t_1} s_{t_2} + q_G s_{t_1} s_{t_2} + q_T s_{t_1} s_{t_2} \\
P(C, C | T, t_1, t_2) &= \frac{1}{16} (1 + 3e^{-4\alpha(t_1 + t_2)}) \\
P(C, G | T, t_1, t_2) &= \frac{1}{16} (1 - e^{-4\alpha(t_1 + t_2)})
\end{align*}
\]
Two sequences

\[ P(C, C \mid T, t_1, t_2) = \frac{1}{16} (1 + 3e^{-4\alpha(t_1 + t_2)}) \]
\[ P(C, G \mid T, t_1, t_2) = \frac{1}{16} (1 - e^{-4\alpha(t_1 + t_2)}) \]
\[ P(x_1, x_2 \mid T, t_1, t_2) = \frac{1}{16^{n_1 + n_2}} (1 + 3e^{-4\alpha(t_1 + t_2)})^{n_1} (1 - e^{-4\alpha(t_1 + t_2)})^{n_2} \]

The \( P(x \mid T, t) \) is root invariant for two gene case. If we carry the product over all sequence positions and make \( n_1 = \) number of matches and \( n_2 = \) number of mismatches.

We show \( P \) as a function of \( t_1 + t_2 \) (alpha = 0.01) for various values of \( n_1 \) and \( n_2 \).
n sequences

\[ P(x_u^1, ..., x_u^n \mid T, t^\star) = \]

\[ \sum_{a^{n+1}, a^{n+2}, ..., a^{2n-1}} q_{a^{2n-1}} \prod_{i=n+1}^{2n-2} P(a^i \mid a^{\alpha(i)}, t_i) \prod_{i=1}^{n} P(x_u^i \mid a^{\alpha(i)}, t_i) \]

Given T and t.
\( \alpha(i) \) is immediate ancestor to node i.
X’s represent sequence positions
a’s represent sequence possibilities at internal nodes.

The sum is over all possible assignments of a at each non-leaf node... this could mean a big computation per evaluation of each T,t over X ...
n sequences

\[ P(x_u^1, ..., x_u^n \mid T, t_i) = \sum_{a^{n+1}, a^{n+2}, ..., a^{2n-1}} q_{a^{2n-1}} \prod_{i=n+1}^{2n-2} P(a^i \mid a^{\alpha(i)}, t_i) \prod_{i=1}^{n} P(x_u^i \mid a^{\alpha(i)}, t_i) \]

init:

\( k = 2n - 1 \)

recursion(\( P(L_k \mid a) \)):

if \( k = \text{leaf} \):

\( P(L_k \mid a = x_u^k) = 1; P(L_k \mid a \neq x_u^k) = 0 \)

if \( k > n \):

\( P(L_k \mid a) = \sum_{b, c} P(b \mid a, t_i)P(L_i \mid b)P(c \mid a, t_j)P(L_j \mid c) \)

term:

\( l_u = P(x_u^* \mid T, t_i) = \sum_a P(L_{2n-1} \mid a)q_a \)
Finding most probable trees

For small trees numerically solve for maximum likelihood tree

Or, maximum likelihood algorithm proposed by Felsenstein.

Conjugate gradient.
Finding most probable trees

For small trees numerically solve for maximum likelihood tree

Or, maximum likelihood algorithm proposed by Felsenstein.

Conjugate gradient.

We can also use Monte Carlo to sample from

\[ P(T, t_\star | x_\star) = \frac{P(x_\star | T, t_\star)P(T, t_\star)}{P(x_\star)} \]

\[ P_1 = P(T, t_\star | x_\star) \]

\[ P_2 = P(\tilde{T}, \tilde{t}_\star | x_\star) \]

If \( P_2 \leq P_1 \) accept move
If \( P_2 > P_1 \) accept move with \( P \sim P_2/P_1 \)

Moves are defined by a so-called proposal distribution. Possible moves to change one tree into another:

1. Change node height
2. Reordering leaves / branch switching

- Still a very difficult search.
Parsimony and Felsenstein algorithm

We can relate the weighted parsimony algorithm to the ML algorithm of Felsenstein.

Score from parsimony can be related to $P$:

$$S(a,b) = -\log P(b \mid a)$$

We see that parsimony uses:

$$\min(S(b) + S(a,b)) \approx \max(P(b)P(b \mid a))$$

While maximum likelihood algorithm uses:

$$\sum_b P(b)P(b \mid a)$$

Thus we can think of the weighted parsimony algorithm as a Viterbi approximation of the of the ML result with fixed branch length given the tree.

Problems:

No branch length optimization, so several cases where parsimony does quite poorly.
Neighbor joining
-> Maximum Likelihood

With multiplicative and reversible P’s for substitutions we can show neighbor joining correctly reconstructs tree:

\[ P(a^1 | a^8, t_1 + t_6) = \sum_{a_6} P(a^1 | a^6, t_1)P(a^6 | a^8, t_6) \]

\[ \sum_{a_8} P(a^1 | a^8, t_1 + t_6)P(a^3 | a^8, t_7 + t_3)q_{a^8} = \]

\[ P(a^1 | a^3, t_1 + t_6 + t_3 + t_7)q_{a^3} \]

\[ P(x^i_u, x^j_u | T, t) = q^i_u P(x^i_u | x^j_u, t_{k1} + t_{k2} + \ldots + t_{kr}) \]

\[ d_{ij}^{ML} = \arg \max_t (\prod_u q^j_u P(x^i_u | x^j_u, t)) \]

\[ d_{ij}^{ML} \approx t_{k1} + t_{k2} + \ldots + t_{kr} \]
Next week’s reading

• Ch. 9 BSA : Preparing for RNA structure prediction

• Berezikov, Cuppen & Plasterk. Approaches to microRNA discovery. NATURE GENETICS SUPPLEMENT. S2 VOLUME 38.JUNE 2006