Bioinformatics – Lecture Notes

Announcements

Remember: Project Proposals are due April 11.

Class 21 – April 2, 2002 -

A. Phylogenetic Trees – Maximum Likelihood Approaches continues

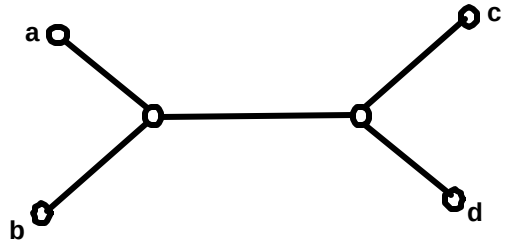
Felsenstein's method is a method that is mathematically sound, however, if the number of species (taxa) gets large, it is too computationally expensive. An alternative is *quartet puzzling* developed by Haseler. It is based on Felsenstein's method for 4-taxon trees (quartet trees).

1. Quartet Puzzling

This method can be broken down into 3 steps:

i) Given n taxa, compute $\binom{n}{4} = \frac{n!}{4!(n-4)!}$ maximum-likelihood trees for all possible quartets.
ii) Combine the quartet trees into a n-taxon tree, that preserves (or at least tries to) the neighbor relations of all quartet trees. This is the *quartet puzzling* step.
iii) Repeat steps i) and ii) many times and output the majority consensus tree. The consensus tree might be multifurcating rather than bifurcating.

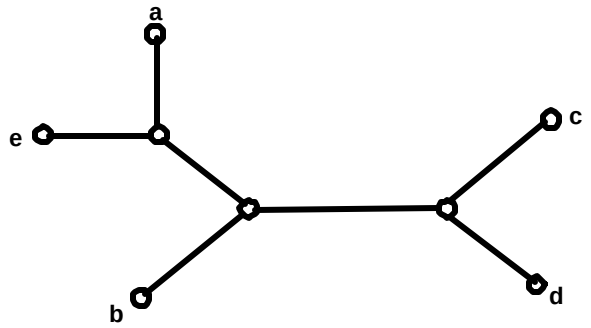
Before we discuss these steps, we should cover some preliminary discussion. Let t is an unrooted binary phylogenetic tree and let $\{a,b,c,d\}$ be any quartet tree of sublabels of t (below).



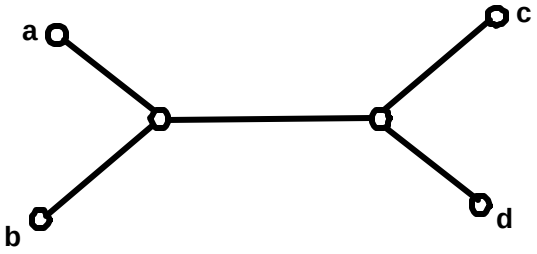
The tree t induces a neighbor relation $N(a,b;c,d)$ if a,b and c,d are found in disjoint subtrees.

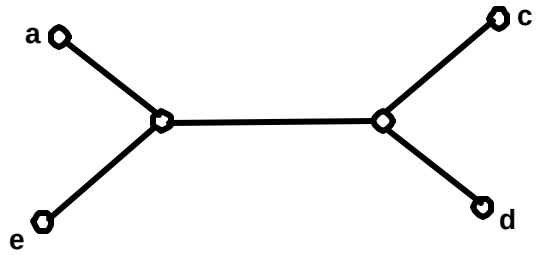
We will now discuss these steps in some detail.

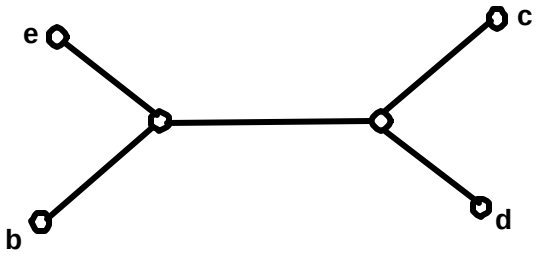
i) Computing the maximum likelihood trees.
Consider if we have tree with the topology given below.

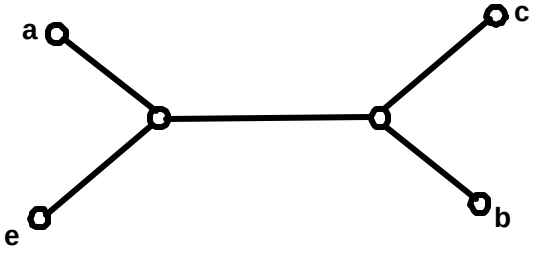


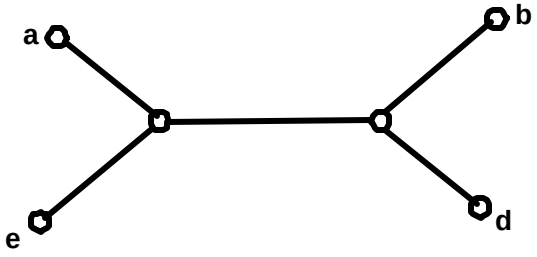
If we take the quartet $\{a,b,c,d\}$, then the neighbor relation $N(a,b;c,d)$ is shown above. The quartet trees induced by t are $\binom{5}{4} = 5$ in number and are shown below.







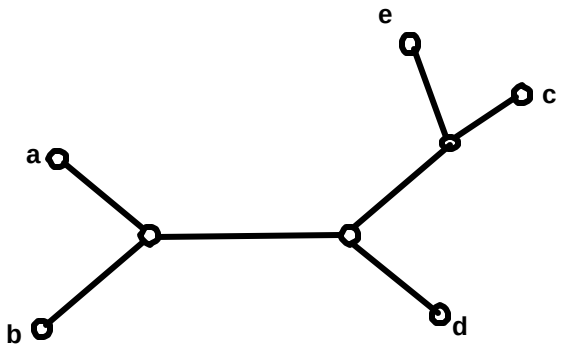




ii) Quartet Puzzling

This procedure is summarized as follows:

Starting with one of the neighbor relations, compute the cost of converting to the each of the other neighbor relation. This is done by adding a 1 to the edge with the property that if a new taxon is added at this edge, the wrong quartet topology for a given quartet would result. For example, if we are trying to convert $N_1(a,b;c,d)$ to $N_1(a,e;c,d)$, adding a taxon e to edge starting from c would yield the wrong topology (below).



This is wrong because it would induce the toplogy $N(a,d;c,e)$ which is not one of the neighbor relations we determined earlier. For this reason, a 1 is added to this edge. Similarly, a 1 is added to the edge starting from d. In other words, adding e to any edge between c and d will result in the wrong neighbor relation. Hence, 1 is added to every edge between c and d. Adding e to the other edges results in the correct neighbor relations so their weights remain at 0. (Text Fig. 4.17).

An algorithm for this exists in the book (Algorithm 4.4). You can read this at your own leisure.

iii) Determining the majority consensus tree.

Suppose that $P_1, \ldots P_m$ are trees resulting form Algorithm 4.4 above in ii). Then each of the trees have the same leaves, ie. the taxa 1,…, n. Define labels l for all the nodes of a n-taxon tree P with each label belonging to the set {1,…,n} and with the following properties

a) The label of a leaf node associated with the taxon $i \in \{1,\ldots,n\}$ is $\{i\}$
b) Suppose that the nodes $x_1,\ldots x_r \in P$ have been labeled $l_1,\ldots l_r$, respectively and that y is the barent of $x_1,\ldots x_r$. Then y is labeled by $l_1 \bigcup ,\ldots \bigcup l_r$

The collection of labels on any tree P is forms a n-tree, defined as follows

Definition – An n-tree is a tree having n leaves, for which every node has a label $l \subseteq \{1,...,n\}$ and

1. {} is not a label of any node
2. The leaves are labeled by {1},…{n}

3. If l_1 and l_2 are labels of nodes in T, and $l_1 \bigcap l_2 \neq 0$, then either $l_1 \subseteq l_2$ or $l_1 \supseteq l_2$

After labeling all the nodes of $P_1,\ldots P_m$, the *majority consensus tree* M is determined by taking those nodes whose label appears in more than half of the P_i.

B. Hidden Markov Models

We have talked about Markov chains before in which a system with multiple states undergoes a transition from state i to state j with transition probability $p_{i,j}$. If we assume that each state can no longer be directly observed and that from each state an output symbol $\sigma \in \Sigma = \{\sigma_1,...,\sigma_m\}$ is chosen, we have a *hidden Markov model*.

With the hidden Markov model, we can define the following

$\pi_i = \Pr[q_0 = i]$	initial state probabilities
$a_{i.j} = \Pr[q_t = j \mid q_{t-1} = i]$	transition probabilities between states
$b_{i,k} = \Pr[o_t = \sigma_k \mid q_t = i]$	emission probabilities (probability a certain outcome give we are in state I

Example – Consider a set of three urns that we cannot see into. Assume that each urn contains a different distribution of colored balls. If we are given a sequence of colored balls derived from drawing balls at from the urns (with replacement), we might want to determine the transition probabilities between urns and the emission probabilities.

Another Example – Assume we have three coins. The first is fair, the second has Pr[H]=0.75, and the third had Pr[H]=0.1. Assume that a coin is drawn at random from the start. After that, if the first coin is chosen we must choose either the second or third after that. If either the second or third is chosen, we can choose any coin after that.

$$\pi_i = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix} \qquad a = \begin{pmatrix} 0 & 1/2 & 1/2 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{pmatrix} \qquad b = \begin{pmatrix} 1/2 & 1/2 \\ 3/4 & 1/4 \\ 1/10 & 9/10 \end{pmatrix}$$

Definition – An *order-k (time-homogeneous) Markov chain* is a Markov chain that depends on the k previous states (instead of just the previous state).

Applications – Pattern recognition in biological systems

Ion channels – matching electrophysiological data with markov model
Speech processing
Multiple sequence alignment
Protein class recognition
Eukaryotic DNA processing
Microarray data analysis