

Lecture 6 – Star Decomposition/Neighbor Joining

I. Introduction: So there's this vast tree space out there for phylogenetic studies of more than a few taxa. As I've said, there are phylogenetic approaches that try to search this tree space, and those that attempt simply to build a tree using some particular algorithm and hope that it's a good tree.

These are known as “Algorithmic Approaches” to tree building, and may be applied to either character data or a matrix of pairwise distances. One of the most widely used of these methods (neighbor joining) was developed by Saitou & Nei, and their justification is as such.

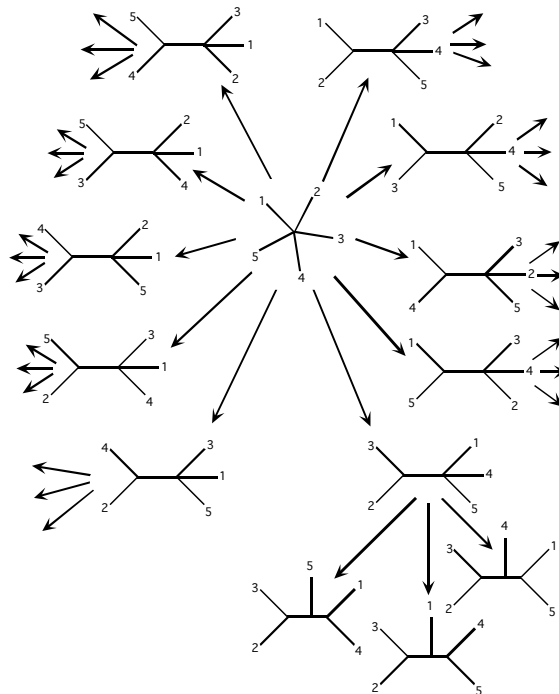
Any estimate of a phylogenetic tree has a large variance. Therefore, any tree that we can demonstrate to be optimal (under some criterion) is not guaranteed to be the true tree. A tree produced by a fast algorithm may be just as close to the true tree as the optimal tree. Therefore, it makes little sense to waste time & resources searching for the optimal tree.

This point merits discussion and we'll discuss it more when we address hypothesis testing. For now, I'm willing to concede that this is true under some limited contexts (e.g., identifying a tree for evaluation of models), but it's probably a bad idea to make it one's default position.

Nevertheless, we should have an understanding of how these methods work, if only because they can be incorporated into more rigorous tree searching.

II. Star Decomposition Methods (Still Chapter 11)

Obviously, star-decomposition methods begin with a star tree, with all taxa emanating from the central node. The star is then decomposed, usually following some optimality criterion.



The most common SD method is the **Neighbor-Joining** algorithm of Saitou and Nei (1987).

This method starts by converting the **raw distance matrix** to **corrected (or transformed) distance matrix**.

- 1) This is accomplished by first estimating the **net divergence** of each taxon **from all others**: where n is the number of nodes (on the first pass, this is equal to the number of taxa).

$$r_i = \sum_{k=1}^n d_{i,k}$$

This essentially provides normalization for unequal rates of evolution.

- 2) The **corrected matrix** is given by:

$$M_{i,j} = d_{i,j} - [r_i / (n - 2)] - [r_j / (n - 2)]$$

- 3) The minimum $M_{i,j}$ (the shortest corrected distance) identifies the two taxa to be united to an ancestral node (u).

This new node u has three branches emanating from it. One to terminal i one to terminal j and one to the rest of the tree.

The lengths of these branches are calculated by:

$$v_{i,u} = d_{i,j} / 2 + (r_i / (n-2) - r_j / (n-2)) / 2$$

and

$$v_{j,u} = d_{i,j} - v_{i,u}$$

- 4) A new matrix is calculated by replacing taxa i and j with their ancestral node u . The distances from all remaining taxa and node u are calculated as follows:

$$d_{k,u} = (d_{i,k} + d_{j,k} - d_{i,j}) / 2$$

- 5) If more than two nodes remain, reset $n = n - 1$ and return to step 1. If only two nodes remain, $v_{i,j} = d_{i,j}$.

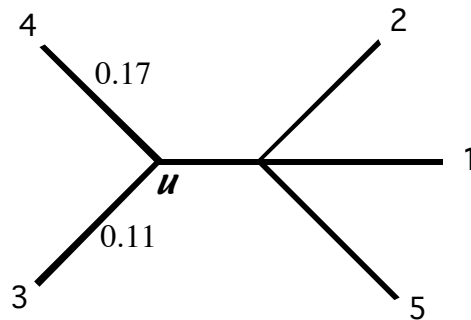
Here's a worked example.

| | OTU-1 | OTU-2 | OTU-3 | OTU-4 | OTU-5 | r_i | $r_i/(n-2)$ |
|---|--------|--------|---------------|--------|-------|-------|-------------|
| 1 | ----- | 0.17 | 0.21 | 0.31 | 0.23 | 0.92 | 0.307 |
| 2 | -0.477 | ----- | 0.30 | 0.34 | 0.21 | 1.02 | 0.340 |
| 3 | -0.490 | -0.433 | ----- | 0.28 | 0.39 | 1.18 | 0.393 |
| 4 | -0.450 | -0.453 | -0.566 | ----- | 0.43 | 1.36 | 0.453 |
| 5 | -0.497 | -0.550 | -0.533 | -0.443 | ----- | 1.26 | 0.420 |

So the star is decomposed by uniting taxa 3 & 4 and calculating their branch lengths:

$$v_{3,u} = d_{3,4} / 2 + (r_3/3 - r_4/3) / 2 = 0.28 / 2 + (0.393 - 0.453) / 2 = 0.11$$

$$v_{4,u} = 0.28 - 0.11 = 0.17$$



Now we set $n = 4$ and build a new matrix by generating distances from node u to OTU's 1, 2, and 5:

$$d_{1,u} = (d_{1,3} + d_{1,4} - d_{3,4}) / 2 = (0.21 + 0.31 - 0.28) / 2 = 0.12$$

$$d_{2,u} = (d_{2,3} + d_{2,4} - d_{3,4}) / 2 = (0.30 + 0.34 - 0.28) / 2 = 0.18$$

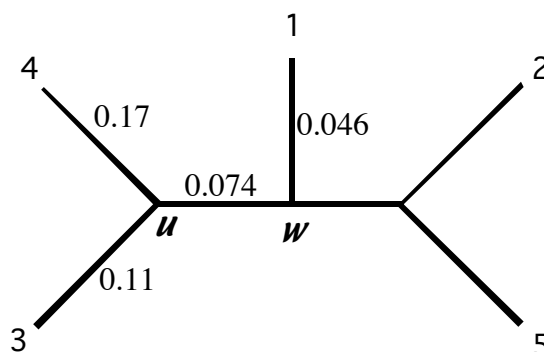
$$d_{5,u} = (d_{5,3} + d_{5,4} - d_{3,4}) / 2 = (0.39 + 0.43 - 0.28) / 2 = 0.27$$

| | OTU-1 | OTU-2 | OTU-5 | Node u | r_i | $r_i/(n-2)$ |
|-----|---------------|--------|--------|----------|-------|-------------|
| 1 | ----- | 0.17 | 0.23 | 0.12 | 0.52 | 0.260 |
| 2 | -0.370 | ----- | 0.21 | 0.18 | 0.56 | 0.280 |
| 5 | -0.385 | -0.425 | ----- | 0.27 | 0.71 | 0.355 |
| u | -0.425 | -0.385 | -0.370 | ----- | 0.57 | 0.285 |

So the star is decomposed further by uniting node u and OTU-1 with an ancestral node w , and the two branch lengths are calculated as follows:

$$v_{1,w} = d_{1,u} / 2 + (r_1/2 - r_u/2) / 2 = 0.12 / 2 + (0.260 - 0.285) / 2 = 0.046$$

$$v_{u,w} = 0.12 - 0.046 = 0.074$$



Now we set $n = 3$ and continue with a new matrix by calculating distances from node w to the remaining OTU's 2 and 5.

$$d_{2,w} = (d_{1,2} + d_{u,2} - d_{1,u}) / 2 = (0.17 + 0.18 - 0.12) / 2 = 0.115$$

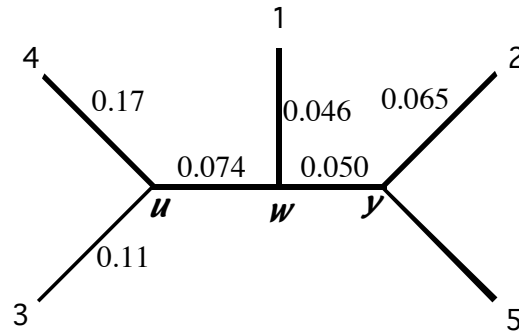
$$d_{5,w} = (d_{1,5} + d_{u,5} - d_{1,u}) / 2 = (0.23 + 0.28 - 0.12) / 2 = 0.195$$

| | OTU-2 | OTU-5 | Node w | r_i | $r_i / (n - 2)$ |
|-----|---------------|--------|----------|-------|-----------------|
| 2 | ----- | 0.21 | 0.115 | 0.325 | 0.325 |
| 5 | -0.510 | ----- | 0.195 | 0.395 | 0.395 |
| w | -0.520 | -0.510 | ----- | 0.310 | 0.310 |

Now, node y unites node w with OTU-2, and again branch lengths are calculated:

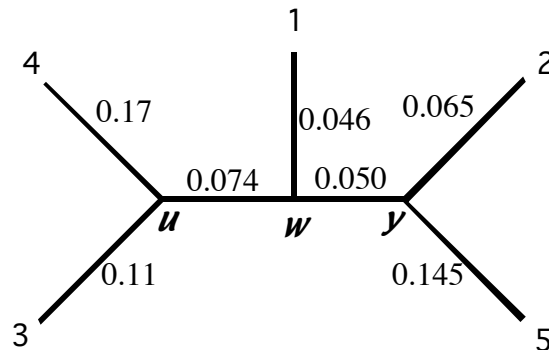
$$v_{y,2} = d_{2,w} / 2 + (r_2/1 - r_w/1) / 2 = 0.115 / 2 + (0.325 - 0.310) / 2 = 0.065$$

$$v_{y,w} = 0.115 - 0.065 = 0.050$$



Finally, we calculate the distance from node y to OTU -5:

$$d_{5,y} = (d_{w,5} + d_{2,5} - d_{2,w}) / 2 = (0.195 + 0.21 - 0.115) / 2 = 0.145 = v_{5,y}$$



Note that the number of calculations required decreases as we build the tree.

There are a couple recent modifications to the neighbor-joining algorithm, BIONJ (incorporates variances and covariances of $d_{i,j}$) and weighbor, that use different weighting factors either in Step 3 or in Steps 2 and 3.

Neighbor joining is often interpreted as an approximation to the ME solution.

Note that this is not “phenetic”. It is not attempting to cluster based on overall similarity, but attempts to parse that into primitive similarity and derived similarity (though the normalization).

In different implementations of NJ, ties are broken differently. Originally they were broken in an arbitrary but constant method. Some newer implementations (e.g., PAUP*) breaks ties randomly.