WIKIPEDIA

Neighbor joining

In bioinformatics, **neighbor joining** is a bottom-up (agglomerative) clustering method for the creation of phylogenetic trees, created by Naruya Saitou and Masatoshi Nei in 1987.[1] Usually used for trees based on DNA or protein sequence data, the algorithm requires knowledge of the distance between each pair of taxa (e.g., species or sequences) to form the tree.[2]

Contents

The algorithm

Neighbor joining takes as input a distance matrix specifying the distance between each pair of taxa. The algorithm starts with a completely unresolved tree, whose topology corresponds to that of a star network, and iterates over the following steps until the tree is completely resolved and all branch lengths are known:

1. Based on the current distance matrix calculate the matrix Q (defined below).
2. Find the pair of distinct taxa i and j (i.e. with $i \neq j$) for which $Q(i,j)$ has its lowest value. These taxa are joined to a newly created node, which is connected to the central node. In the figure at right, f and g are joined to the new node u.
3. Calculate the distance from each of the taxa in the pair to this new node.
4. Calculate the distance from each of the taxa outside of this pair to the new node.
5. Start the algorithm again, replacing the pair of joined neighbors with the new node and using the distances calculated in the previous step.

The Q-matrix

Based on a distance matrix relating the n taxa, calculate Q as follows:

$$Q(i,j) = (n-2)d(i,j) - \sum_{k=1}^{n} d(i,k) - \sum_{k=1}^{n} d(j,k) \qquad (1)$$

where $d(i,j)$ is the distance between taxa i and j.

Distance from the pair members to the new node

For each of the taxa in the pair being joined, use the following formula to calculate the distance to the new node:

$$\delta(f,u) = \frac{1}{2}d(f,g) + \frac{1}{2(n-2)}\left[\sum_{k=1}^{n} d(f,k) - \sum_{k=1}^{n} d(g,k)\right] \qquad (2)$$

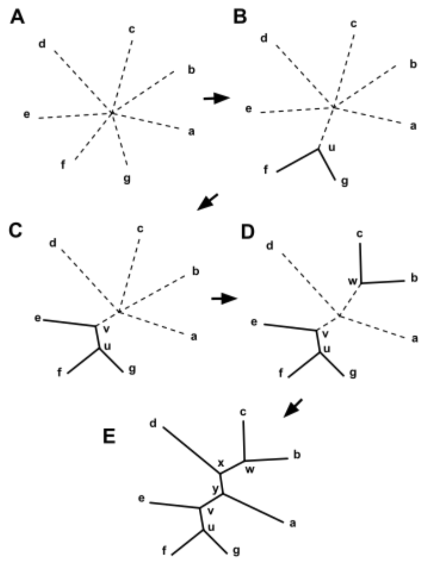
and:

$$\delta(g,u) = d(f,g) - \delta(f,u)$$

Taxa f and g are the paired taxa and u is the newly created node. The branches joining f and u and g and u, and their lengths, $\delta(f,u)$ and $\delta(g,u)$ are part of the tree which is gradually being created; they neither affect nor are affected by later neighbor-joining steps.

Distance of the other taxa from the new node

For each taxon not considered in the previous step, we calculate the distance to the new node as follows:



Starting with a star tree (A), the Q matrix is calculated and used to choose a pair of nodes for joining, in this case f and g. These are joined to a newly created node, u, as shown in (B). The part of the tree shown as solid lines is now fixed and will not be changed in subsequent joining steps. The distances from node u to the nodes a-e are computed from equation (3). This process is then repeated, using a matrix of just the distances between the nodes, a,b,c,d,e, and u, and a Q matrix derived from it. In this case u and e are joined to the newly created v, as shown in (C). Two more iterations lead first to (D), and then to (E), at which point the algorithm is done, as the tree is fully resolved.

$$d(u,k) = \frac{1}{2}[d(f,k) + d(g,k) - d(f,g)] \quad (3)$$

where u is the new node, k is the node which we want to calculate the distance to and f and g are the members of the pair just joined.

Complexity

Neighbor joining on a set of n taxa requires $n-3$ iterations. At each step one has to build and search a Q matrix. Initially the Q matrix is size $n \times n$, then the next step it is $(n-1) \times (n-1)$, etc. Implementing this in a straightforward way leads to an algorithm with a time complexity of $O(n^3)$; implementations exist which use heuristics to do much better than this on average.

Example

Let us assume that we have five taxa (a,b,c,d,e) and the following distance matrix D:

	a	b	c	d	e
a	0	5	9	9	8
b	5	0	10	10	9
c	9	10	0	8	7
d	9	10	8	0	3
e	8	9	7	3	0

First step

First joining

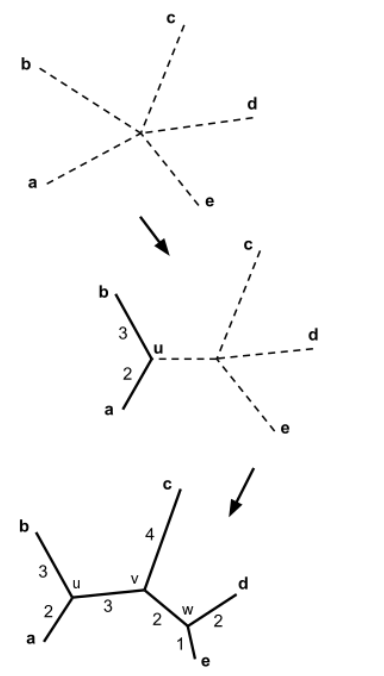
We calculate the Q_1 values by equation (1). For example:

$$Q_1(a,b) = (n-2)d(a,b) - \sum_{k=1}^{5} d(a,k) - \sum_{k=1}^{5} d(b,k)$$
$$= (5-2)\times 5 - (5+9+9+8) - (5+10+10+9) = 15 - 31 - 34 = -50$$

We obtain the following values for the Q_1 matrix (the diagonal elements of the matrix are not used and are omitted here):

	a	b	c	d	e
a		−50	−38	−34	−34
b	−50		−38	−34	−34
c	−38	−38		−40	−40
d	−34	−34	−40		−48
e	−34	−34	−40	−48	

In the example above, $Q_1(a,b) = -50$. This is the smallest value of Q_1, so we join elements a and b.



Neighbor joining with 5 taxa. In this case 2 neighbor joining steps give a tree with fully resolved topology. The branches of the resulting tree are labeled with their lengths.

First branch length estimation

Let u denote the new node. By equation (2), above, the branches joining a and b to u then have lengths:

$$\delta(a,u) = \frac{1}{2}d(a,b) + \frac{1}{2(5-2)}\left[\sum_{k=1}^{5} d(a,k) - \sum_{k=1}^{5} d(b,k)\right] \quad = \frac{5}{2} + \frac{31-34}{6} = 2$$
$$\delta(b,u) = d(a,b) - \delta(a,u) \quad = 5 - 2 = 3$$

First distance matrix update

We then proceed to update the initial distance matrix D into a new distance matrix D_1 (see below), reduced in size by one row and one column because of the joining of a with b into their neighbor u. Using equation (3) above, we compute the distance from u to each of the other nodes besides a and b. In this case, we obtain:

$$d(u,c) = \frac{1}{2}[d(a,c) + d(b,c) - d(a,b)] = \frac{9+10-5}{2} = 7$$
$$d(u,d) = \frac{1}{2}[d(a,d) + d(b,d) - d(a,b)] = \frac{9+10-5}{2} = 7$$
$$d(u,e) = \frac{1}{2}[d(a,e) + d(b,e) - d(a,b)] = \frac{8+9-5}{2} = 6$$

The resulting distance matrix D_1 is:

	u	c	d	e
u	0	**7**	**7**	**6**
c	**7**	0	*8*	*7*
d	**7**	*8*	0	*3*
e	**6**	*7*	*3*	0

Bold values in D_1 correspond to the newly calculated distances, whereas italicized values are not affected by the matrix update as they correspond to distances between elements not involved in the first joining of taxa.

Second step

Second joining

The corresponding Q_2 matrix is:

	u	c	d	e
u		−28	−24	−24
c	−28		−24	−24
d	−24	−24		−28
e	−24	−24	−28	

We may choose either to join u and c, or to join d and e; both pairs have the minimal Q_2 value of -28, and either choice leads to the same result. For concreteness, let us join u and c and call the new node v.

Second branch length estimation

The lengths of the branches joining u and c to v can be calculated:

$$\delta(u,v) = \frac{1}{2}d(u,c) + \frac{1}{2(4-2)}\left[\sum_{k=1}^{4} d(u,k) - \sum_{k=1}^{4} d(c,k)\right] \quad = \frac{7}{2} + \frac{20-22}{4} = 3$$
$$\delta(c,v) = d(u,c) - \delta(u,v) \quad = 7 - 3 = 4$$

The joining of the elements and the branch length calculation help drawing the neighbor joining tree as shown in the figure.

Second distance matrix update

The updated distance matrix D_2 for the remaining 3 nodes, v, d, and e, is now computed:

$$d(v,d) = \frac{1}{2}[d(u,d) + d(c,d) - d(u,c)] = \frac{7+8-7}{2} = 4$$
$$d(v,e) = \frac{1}{2}[d(u,e) + d(c,e) - d(u,c)] = \frac{6+7-7}{2} = 3$$

	v	**d**	**e**
v	0	**4**	**3**
d	**4**	0	*3*
e	**3**	*3*	0

Final step

The tree topology is fully resolved at this point. However, for clarity, we can calculate the Q_3 matrix. For example:

$$Q_3(v,e) = (3-2)d(v,e) - \sum_{k=1}^{3} d(v,k) - \sum_{k=1}^{3} d(e,k) = 3 - 7 - 6 = -10$$

	v	**d**	**e**
v		−10	−10
d	−10		−10
e	−10	−10	

For concreteness, let us join v and d and call the last node w. The lengths of the three remaining branches can be calculated:

$$\delta(v,w) = \frac{1}{2}d(v,d) + \frac{1}{2(3-2)}\left[\sum_{k=1}^{3} d(v,k) - \sum_{k=1}^{3} d(d,k)\right] \quad = \frac{4}{2} + \frac{7-7}{2} = 2$$
$$\delta(w,d) = d(v,d) - \delta(v,w) = 4 - 2 = 2$$
$$\delta(w,e) = d(v,e) - \delta(v,w) = 3 - 2 = 1$$

The neighbor joining tree is now complete, as shown in the figure.

Conclusion: additive distances

This example represents an idealized case: note that if we move from any taxon to any other along the branches of the tree, and sum the lengths of the branches traversed, the result is equal to the distance between those taxa in the input distance matrix. For example, going from d to b we have $2+2+3+3=10$. A distance matrix whose distances agree in this way with some tree is said to be 'additive', a property which is rare in practice. Nonetheless it is important to note that, given an additive distance matrix as input, neighbor joining is guaranteed to find the tree whose distances between taxa agree with it.

Neighbor joining as minimum evolution

Neighbor joining may be viewed as a greedy heuristic for the Balanced Minimum Evolution[3] (BME) criterion. For each topology, BME defines the tree length (sum of branch lengths) to be a particular weighted sum of the distances in the distance matrix, with the weights depending on the topology. The BME optimal topology is the one which minimizes this tree length. Neighbor joining at each step greedily joins that pair of taxa which will give the greatest decrease in the estimated tree length. This procedure does not guarantee to find the optimum for the BME criterion, although it often does and is usually quite close.

Advantages and disadvantages

The main virtue of NJ is that it is fast[4]:466 as compared to least squares, maximum parsimony and maximum likelihood methods.[4] This makes it practical for analyzing large data sets (hundreds or thousands of taxa) and for bootstrapping, for which purposes other means of analysis (e.g. maximum parsimony, maximum likelihood) may be computationally prohibitive.

Neighbor joining has the property that if the input distance matrix is correct, then the output tree will be correct. Furthermore, the correctness of the output tree topology is guaranteed as long as the distance matrix is 'nearly additive', specifically if each entry in the distance matrix differs from the true distance by less than half of the shortest branch length in the tree.[5] In practice the distance matrix rarely satisfies this condition, but neighbor joining often constructs the correct tree topology anyway.[6] The correctness of neighbor joining for nearly additive distance matrices implies that it is statistically consistent under many models of evolution; given data of sufficient length, neighbor joining will reconstruct the true tree with high probability. Compared with UPGMA and WPGMA, neighbor joining has the advantage that it does not assume all lineages evolve at the same rate (molecular clock hypothesis).

Nevertheless, neighbor joining has been largely superseded by phylogenetic methods that do not rely on distance measures and offer superior accuracy under most conditions. Neighbor joining has the undesirable feature that it often assigns negative lengths to some of the branches.

Implementations and variants

There are many programs available implementing neighbor joining. RapidNJ (http://birc.au.dk/Software/RapidNJ/) and NINJA (http://wheelerlab.org/software/ninja/index.html) are fast implementations with typical run times proportional to approximately the square of the number of taxa. BIONJ (http://www.atgc-montpellier.fr/bionj/) and Weighbor (https://web.archive.org/web/20150305041147/http://www.t6.lanl.gov/billb/weighbor/) are variants of neighbor joining which improve on its accuracy by making use of the fact that the shorter distances in the distance matrix are generally better known than the longer distances. FastME (https://www.ncbi.nlm.nih.gov/CBBresearch/Desper/FastME.html) is an implementation of the closely related balanced minimum evolution method.

See also

- Nearest neighbor search
- UPGMA and WPGMA
- Minimum Evolution

References

1. Saitou, N.; Nei, M. (1 July 1987). "The neighbor-joining method: a new method for reconstructing phylogenetic trees" (https://doi.org/10.1093/oxfordjournals.molbev.a040454). *Molecular Biology and Evolution*. **4** (4): 406–425. doi:10.1093/oxfordjournals.molbev.a040454 (https://doi.org/10.1093%2Foxfordjournals.molbev.a040454). PMID 3447015 (https://pubmed.ncbi.nlm.nih.gov/3447015).
2. Xavier Didelot (2010). "Sequence-Based Analysis of Bacterial Population Structures" (https://books.google.com/books?id=gPVjfsWnGCcC&pg=PA46). In D. Ashley Robinson; Daniel Falush; Edward J. Feil (eds.). *Bacterial Population Genetics in Infectious Disease*. John Wiley and Sons. pp. 46–47. ISBN 978-0-470-42474-2.
3. Gascuel O, Steel M (2006). "Neighbor-joining revealed" (https://hal-lirmm.ccsd.cnrs.fr/lirmm-00136653/document). *Mol Biol Evol*. **23** (11): 1997–2000. doi:10.1093/molbev/msl072 (https://doi.org/10.1093%2Fmolbev%2Fmsl072). PMID 16877499 (https://pubmed.ncbi.nlm.nih.gov/16877499).
4. Kuhner, M. K.; Felsenstein, J. (1994-05-01). "A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates" (https://doi.org/10.1093/oxfordjournals.molbev.a040126). *Molecular Biology and Evolution*. **11** (3): 459–468. doi:10.1093/oxfordjournals.molbev.a040126 (https://doi.org/10.1093%2Foxfordjournals.molbev.a040126). ISSN 0737-4038 (https://www.worldcat.org/issn/0737-4038). PMID 8015439 (https://pubmed.ncbi.nlm.nih.gov/8015439).
5. Atteson K (1997). "The performance of neighbor-joining algorithms of phylogeny reconstruction", pp. 101–110. *In* Jiang, T., and Lee, D., eds., *Lecture Notes in Computer Science, 1276*, Springer-Verlag, Berlin. COCOON '97.
6. Mihaescu R, Levy D, Pachter L (2009). "Why neighbor-joining works". *Algorithmica*. **54** (1): 1–24. arXiv:cs/0602041 (https://arxiv.org/abs/cs/0602041). doi:10.1007/s00453-007-9116-4 (https://doi.org/10.1007%2Fs00453-007-9116-4).

Other sources

- Studier JA, Keppler KJ (1988). "A note on the Neighbor-Joining algorithm of Saitou and Nei" (https://doi.org/10.1093/oxfordjournals.molbev.a040527). *Mol Biol Evol*. **5** (6): 729–731. doi:10.1093/oxfordjournals.molbev.a040527 (https://doi.org/10.1093%2Foxfordjournals.molbev.a040527). PMID 3221794 (https://pubmed.ncbi.nlm.nih.gov/3221794).
- Martin Simonsen; Thomas Mailund; Christian N. S. Pedersen (2008). *Rapid Neighbour Joining* (http://birc.au.dk/fileadmin/uploaded/rapidNJ.pdf) (PDF). *Proceedings of WABI*. Lecture Notes in Computer Science. **5251**. pp. 113–122. CiteSeerX 10.1.1.218.2078 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.218.2078). doi:10.1007/978-3-540-87361-7_10 (https://doi.org/10.1007%2F978-3-540-87361-7_10). ISBN 978-3-540-87360-0.

External links

- The Neighbor-Joining Method (http://www.deduveinstitute.be/~opperd/private/neighbor.html) — a tutorial