The Geometry of the Neighbor-Joining Algorithm for Small Trees

Kord Eickmeyer[1] and Ruriko Yoshida[2]

[1] Institut für Informatik, Humboldt-Universität zu Berlin, Berlin, Germany
[2] University of Kentucky, Lexington, KY, USA

Abstract. In 2007, Eickmeyer et al. showed that the tree topologies outputted by the Neighbor-Joining (NJ) algorithm and the balanced minimum evolution (BME) method for phylogenetic reconstruction are each determined by a polyhedral subdivision of the space of dissimilarity maps $\mathbb{R}^{\binom{n}{2}}$, where n is the number of taxa. In this paper, we will analyze the behavior of the Neighbor-Joining algorithm on five and six taxa and study the geometry and combinatorics of the polyhedral subdivision of the space of dissimilarity maps for six taxa as well as hyperplane representations of each polyhedral subdivision. We also study simulations for one of the questions stated by Eickmeyer et al., that is, the robustness of the NJ algorithm to small perturbations of tree metrics, with tree models which are known to be hard to be reconstructed via the NJ algorithm.

1 Introduction

The Neighbor-Joining (NJ) algorithm was introduced by Saitou and Nei [14] and is widely used to reconstruct a phylogenetic tree from an alignment of DNA sequences because of its accuracy and computational speed. The NJ algorithm is a distance-based method which takes all pairwise distances computed from the data as its input, and outputs a tree topology which realizes these pairwise distances, if there is such a topology (see Fig. 1). Note that the NJ algorithm is consistent, i.e., it returns the additive tree if the input distance matrix is a tree metric. If the input distance matrix is not a tree metric, then the NJ algorithm returns a tree topology which induces a tree metric that is "close" to the input. Since it is one of the most popular methods for reconstructing a tree among biologists, it is important to study how the NJ algorithm works.

	a	**b**	**c**	**d**
a	-	3	1.8	2.5
b		-	2.8	3.5
c			-	1.3
d				-

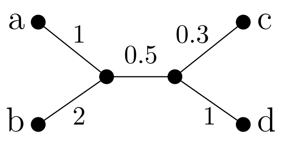


Fig. 1. The NJ algorithm takes a matrix of pairwise distances (left) as input and computes a binary tree (right). If there is a tree such that the distance matrix can be obtained by taking the length of the unique path between two nodes, NJ outputs that tree.

A number of attempts have been made to understand the good results obtained with the NJ algorithm, especially given the problems with the inference procedures used for estimating pairwise distances. For example, Bryant showed that the *Q-criterion* (defined in (Q) in Section 2.2) is in fact the unique selection criterion which is linear, permutation invariant, and *consistent*, i.e., it correctly finds the tree corresponding to a tree metric [2]. Gascuel and Steel gave a nice review of how the NJ algorithm works [15].

One of the most important questions in studying the behavior of the NJ algorithm is to analyze its performance with pairwise distances that are not tree metrics, especially when all pairwise distances are estimated via the maximum likelihood estimation (MLE). In 1999, Atteson showed that if the distance estimates are at most half the minimal edge length of the tree away from their true value then the NJ algorithm will reconstruct the correct tree [1]. However, Levy et al. noted that Atteson's criterion frequently fails to be satisfied even though the NJ algorithm returns the true tree topology [10]. Recent work of [11] extended Atteson's work. Mihaescu et al. showed that the NJ algorithm returns the true tree topology when it works locally for the quartets in the tree [11]. This result gives another criterion for when then NJ algorithm returns the correct tree topology and Atteson's theorem is a special case of Mihaescu et al.'s theorem.

For every input distance matrix, the NJ algorithm returns a certain tree topology. It may happen that the minimum Q-criterion is taken by more than one pair of taxa at some step. In practice, the NJ algorithm will then have to choose one cherry in order to return a definite result, but for our analysis we assume that in those cases the NJ algorithm will return a set containing all tree topologies resulting from picking optimal cherries. There are only finitely many tree topologies, and for every topology t we get a subset D_t of the sample space (input space) such that for all distance matrices in D_t one possible answer of the NJ algorithm is t. We aim at describing these sets D_t and the relation between them. One notices that the Q-criteria are all linear in the pairwise distances. The NJ algorithm will pick cherries in a particular order and output a particular tree t if and only if the pairwise distances satisfy a system of linear inequalities, whose solution set forms a polyhedral cone in $\mathbb{R}^{\binom{n}{2}}$. Eickmeyer et al. called such a cone a *Neighbor-Joining cone,* or *NJ cone*. Thus the NJ algorithm will output a particular tree t if and only if the distance data lies in a union of NJ cones [3].

In [3], Eickmeyer et al. studied the optimality of the NJ algorithm compared to the balanced minimum evolution (BME) method and focused on polyhedral subdivisions of the space of dissimilarity maps for the BME method and the NJ algorithm. Eickmeyer et al. also studied the geometry and combinatorics of the NJ cones for $n = 5$ in addition to the *BME cones* for $n \leq 8$. Using geometry of the NJ cones for $n = 5$, they showed that the polyhedral subdivision of the space of dissimilarity maps with the NJ algorithm does not form a fan for $n \geq 5$ and that the union of the NJ cones for a particular tree topology is not convex. This means that the NJ algorithm is not convex, i.e., there are distance matrices D, D', such that NJ produces the same tree t_1 on both inputs D and D', but it produces a different tree $t_2 \neq t_1$ on the input $(D+D')/2$ (see [3] for an example).

In this paper, we focus on describing geometry and combinatorics of the NJ cones for six taxa as well as some simulation study using the NJ cones for one of the questions in [3], that is, what is the robustness of the NJ algorithm to small perturbations of tree metrics for $n = 5$. This paper is organized as follows: In Section 2 we will describe the NJ algorithm and define the NJ cones. Section 3 states the hyperplane representations of the NJ cones for $n = 5$. Section 4 describes in summary the geometry and combinatorics of the NJ cones for $n = 6$. Section 5 shows some simulation studies on the robustness of the NJ algorithm to small perturbations of tree metrics for $n = 5$ with the tree models from [13]. We end by discussing some open problems in Section 6.

2 The Neighbor-Joining algorithm

2.1 Input data

The NJ algorithm is a *distance-based method* which takes a *distance matrix*, a symmetric matrix $(d_{ij})_{0\leq i,j\leq n-1}$ with $d_{ii} = 0$ representing pairwise distances of a set of n taxa $\{0, 1, \ldots, n-1\}$, as the input. Through this paper, we do not assume anything on an input data except it is symmetric and $d_{ii} = 0$. Because of symmetry, the input can be seen as a vector of dimension $m := \binom{n}{2} = \frac{1}{2}n(n-1)$. We arrange the entries row-wise. We denote row/column-indices by pairs of letters such as a, b, c, d, while denoting single indices into the "flattened" vector by letters $i, j, \ldots$. The two indexing methods are used simultaneously in the hope that no confusion will arise. Thus, in the four taxa example we have $d_{0,1} = d_{1,0} = d_0$. In general, we get $d_i = d_{a,b} = d_{b,a}$ with

$$a = \max\left\{k \mid \frac{1}{2}k(k-1) \leq i\right\} = \left\lfloor \frac{1}{2} + \sqrt{\frac{1}{4} + 2i} \right\rfloor, b = i - \frac{1}{2}(a-1)a,$$

and for $c > d$ we get

$$d_{c,d} = d_{c(c-1)/2+d}.$$

2.2 The Q-Criterion

The NJ algorithm starts by computing the so called *Q-criterion* or the *cherry picking criterion*, given by the formula

$$q_{a,b} := (n-2)d_{a,b} - \sum_{k=0}^{n-1} d_{a,k} - \sum_{k=0}^{n-1} d_{k,b}. \tag{Q}$$

This is a key of the NJ algorithm to choose which pair of taxa is a neighbor.

Theorem 1 (Saitou and Nei, 1987, Studier and Keppler, 1988 [14, 16]). *Let $d_{a,b}$ for all pair of taxa $\{a, b\}$ be the tree metric corresponding to the tree T. Then the pair $\{x, y\}$ which minimizes $q_{a,b}$ for all pair of taxa $\{a, b\}$ forms a neighbor.*

Arranging the Q-criteria for all pairs in a matrix yields again a symmetric matrix, and ignoring the diagonal entries we can see it as a vector of dimension m just like the input data. Moreover, the Q-criterion is obtained from the input data by a linear transformation:

$$\mathbf{q} = A^{(n)}\mathbf{d},$$

and the entries of the matrix $A^{(n)}$ are given by

$$A^{(n)}_{ij} = A^{(n)}_{ab,cd} = \begin{cases} n-4 & \text{if } i = j, \\ -1 & \text{if } i \neq j \text{ and } \{a,b\} \cap \{c,d\} \neq \emptyset, \\ 0 & \text{else,} \end{cases} \tag{1}$$

where $a > b$ is the row/column-index equivalent to i and likewise for $c > d$ and j. When no confusion arises about the number of taxa, we abbreviate $A^{(n)}$ to A.

After computing the Q-criterion $\mathbf{q}$, the NJ algorithm proceeds by finding the minimum entry of $\mathbf{q}$, or, equivalently, the maximum entry of $-\mathbf{q}$. The two nodes forming the chosen pair (there may be several pairs with minimal Q-criterion) are then joined ("cherry picking"), i.e., they are removed from the set of nodes and a new node is created. Suppose out of our n taxa $\{0, \dots, n-1\}$, the first cherry to be picked is $m-1$, so the taxa $n-2$ and $n-1$ are joined to form a new node, which we view as the new node number $n-2$. The reduced pairwise distance matrix is one row and one column shorter than the original one, and by our choice of which cherry we picked, only the entries in the rightmost column and bottom row differ from the original ones. Explicitly,

$$d'_i = \begin{cases} d_i & \text{for } 0 \leq i < \binom{n-2}{2} \\ \frac{1}{2}(d_i + d_{i+(n-2)} - d_{m-1}) & \text{for } \binom{n-2}{2} \leq i < \binom{n-1}{2} \end{cases}$$

and we see that the reduced distance matrix depends linearly on the original one:

$$\mathbf{d}' = R\mathbf{d},$$

with $R = (r_{ij}) \in \mathbb{R}^{(m-n+1)\times m}$, where

$$r_{ij} = \begin{cases} 1 & \text{for } 0 \leq i = j < \binom{n-2}{2} \\ 1/2 & \text{for } \binom{n-2}{2} \leq i < \binom{n-1}{2}, j = i \\ 1/2 & \text{for } \binom{n-2}{2} \leq i < \binom{n-1}{2}, j = i + n - 2 \\ -1/2 & \text{for } \binom{n-2}{2} \leq i < \binom{n-1}{2}, j = m - 1 \\ 0 & \text{else} \end{cases}$$

The process of picking cherries is repeated until there are only three taxa left, which are then joined to a single new node.

We note that since new distances d' are always linear combinations of the previous distances, all Q-criteria computed throughout the NJ algorithm are linear combinations of the original pairwise distances. Thus, for every possible tree topology t outputted by the NJ algorithm (and every possible ordering σ of picked cherries that results in topology t), there is a polyhedral cone $C_{T,\sigma} \subset \mathbb{R}^{\binom{n}{2}}$ of dissimilarity maps. The NJ algorithm will output t and pick cherries in the order σ iff the input lies in the cone $C_{T,\sigma}$. We call the cones $C_{T,\sigma}$ *Neighbor-Joining cones*, or *NJ cones*.

2.3 The shifting lemma

We first note that there is an n-dimensional linear subspace of $\mathbb{R}^m$ which does not affect the outcome of the NJ algorithm (see [11]). For a node a we define its *shift vector* $\mathbf{s}_a$ by

$$(\mathbf{s}_a)_{b,c} := \begin{cases} 1 & \text{if } a \in \{b, c\} \\ 0 & \text{else} \end{cases}$$

which represents a tree where the leaf a has distance 1 from all other leaves and all other distances are zero. The Q-criterion of any such vector is -2 for all pairs, so adding any linear combination of shift vectors to an input vector does not change the relative values of the Q-criteria. Also, regardless of which pair of nodes we join, the reduced distance matrix of a shift vector is again a shift vector (of lower dimension), whose Q-criterion will also be constant. Thus, for any input vector $\mathbf{d}$, the behavior of the NJ algorithm on $\mathbf{d}$ will be the same as on $\mathbf{d} + \mathbf{s}$ if $\mathbf{s}$ is any linear combination of shift vectors. We call the subspace generated by shift vectors S.

We note that the difference of any two shift vectors is in the kernel of A, and the sum of all shift vectors is the constant vector with all entries equal to n. If we fix a node a then the set

$$\{\mathbf{s}_a - \mathbf{s}_b \mid b \neq a\}$$

is linearly independent.

2.4 The first step in cherry picking

After computing the Q-criterion $\mathbf{q}$, the NJ algorithm proceeds by finding the minimum entry of it, or, equivalently, the maximum entry of $-\mathbf{q}$. The set $cq_i \subset \mathbb{R}^m$ of all q-vectors for which q_i is minimal is given by the normal cone at the vertex $-e_i$ to the (negative) simplex

$$\Delta_{m-1} = \operatorname{conv}\{-e_i \mid 0 \leq i \leq m-1\} \subset \mathbb{R}^m,$$

where $e_0, \ldots, e_{m-1}$ are the unit vectors in $\mathbb{R}^m$. The normal cone is defined in the usual way by

$$\begin{aligned} \mathcal{N}_{\Delta_{m-1}}(-e_i) := &\{\mathbf{x} \in \mathbb{R}^m \mid (-e_i, \mathbf{x}) \geq (\mathbf{p}, \mathbf{x}) \text{ for } \mathbf{p} \in \Delta_{m-1}\} \\ = &\{\mathbf{x} \in \mathbb{R}^m \mid (-e_i, \mathbf{x}) \geq (-e_j, \mathbf{x}) \text{ for } 0 \leq j \leq m-1\}, \end{aligned} \tag{2}$$

with $(\cdot, \cdot)$ denoting the inner product in $\mathbb{R}^m$.

Substituting $\mathbf{q} = A\mathbf{d}$ into (2) gives

$$\begin{aligned} \mathbf{q} \in cq_i \quad &\Leftrightarrow \quad i = \arg\max(-e_j, A\mathbf{d}) \\ &\Leftrightarrow \quad i = \arg\max(-A^T e_j, \mathbf{d}) \\ &\Leftrightarrow \quad i = \arg\max(-Ae_j, \mathbf{d}) \qquad \text{because } A \text{ is symmetric.} \end{aligned} \tag{3}$$

Therefore the set cd_i of all *parameter* vectors $\mathbf{d}$ for which the NJ algorithm will select cherry i in the first step is the normal cone at $-Ae_i$ to the polytope

$$P_n := \operatorname{conv}\{-Ae_0, \ldots, -Ae_{m-1}\}. \tag{4}$$

The shifting lemma implies that the affine dimension of the polytope P_n is at most $m - n$. Computations using `polymake` show that this upper bound gives the true value.

If equality holds for one of the inner products in this formula, then there are two cherries with the same Q-criterion.

As the number of taxa increases, the resulting polytope gets more complicated very quickly. By symmetry, the number of facets adjacent to a vertex is the same for every vertex, but this number grows following a strange pattern. See Table 1 for some calculated values. We also computed f-vectors for P_n via `polymake`. With $n = 5$, we have $(1, 10, 45, 90, 75, 22, 1)$, with $n = 6$, $(1, 15, 105, 435, 1095, 1657, 1470, 735, 195, 25, 1)$, and with $n \geq 7$ we ran `polymake` over several hours and it took more than 9GB RAM. Therefore, we could not compute them.

no. of taxa	**no. of vertices**	**dimension**	**facets through vertex**	**no. of facets**
4	3	2	2	3
5	10	5	12	22
6	15	9	18	25
7	21	14	500	717
8	28	20	780	1,057
9	36	27	30,114	39,196
10	45	35	77,924	98,829

Table 1. The polytopes P_n for some small numbers of taxa n.

2.5 The cone cd_i

Equation (3) allows us to write cd_i as an intersection of half-spaces as follows:

$$\begin{aligned} cd_i &= \{\mathbf{x} \mid (-A\mathbf{e}_i, \mathbf{x}) \geq (-A\mathbf{e}_j, \mathbf{x}) \text{ for } j \neq i\} \\ &= \{\mathbf{x} \mid (-A(\mathbf{e}_i - \mathbf{e}_j), \mathbf{x}) \geq 0 \text{ for } j \neq i\} \\ &= \bigcap_{j \neq i} \{\mathbf{x} \mid (-A(\mathbf{e}_i - \mathbf{e}_j), \mathbf{x}) \geq 0\} \end{aligned} \tag{5}$$

We name the half-spaces, their interiors and the normal vectors defining them as follows:

$$\begin{aligned}
\mathbf{h}_{ij}^{(n)} &:= -A^{(n)}(\mathbf{e}_i - \mathbf{e}_j), \\
H_{ij}^{(n)} &:= \left\{\mathbf{x} \in \mathbb{R}^m \mid (\mathbf{h}_{ij}^{(n)}, \mathbf{x}) \geq 0\right\}, \\
\mathring{H}_{ij}^{(n)} &:= \left\{\mathbf{x} \in \mathbb{R}^m \mid (\mathbf{h}_{ij}^{(n)}, \mathbf{x}) > 0\right\},
\end{aligned} \tag{6}$$

where again we omit the superscript (n) if the number of taxa is clear.

If there are more than four taxa, then this representation is not redundant: For any pair i and j of cherries, we can find a parameter vector $\mathbf{d}$ lying on the border of H_{ij} (i.e., $(\mathbf{h}_{ij}, \mathbf{d}) = 0$) but in the interior $\mathring{H}_{ik}$ of the other half-spaces for $k \neq i, j$. One such $\mathbf{d}$ is given by

$$d_k := \begin{cases} 2 & \text{if } k = i \text{ or } k = j, \\ 4 & \text{else.} \end{cases}$$

Thus we have found an $\mathcal{H}$-representation of the polyhedron cd_i consisting of only $m-1$ inequalities. Note that Table 1 implies that a $\mathcal{V}$-representation of the same cone would be much more complicated, as the number of vectors spanning it is equal to the number of facets incident at the vertex $-A\mathbf{e}_i$.

Example 1. The normal vectors to the 22 facets of P_5, and thus the rays of the normal cones to P_5, form two classes (see Fig. 2). The first class contains a total of 12 vectors (as there are 12 assignments of nodes 0 to 4 to the labels a to e which yield nonisomorphic labelings), and every normal cone contains six of them. The second class contains 10 vectors, and again every normal cone has six of them as rays. For each class there are two diagrams in Fig. 2, and we obtain

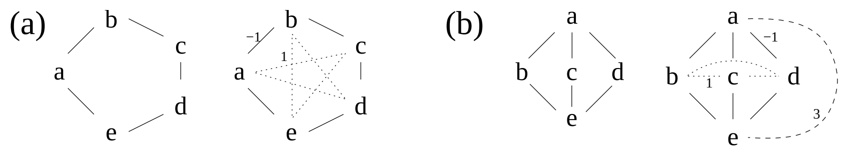


Fig. 2. Diagrams describing the facet-normals of P_5.

a normal vector to one of the facets of P_5 by assigning nodes from $\{0, \ldots, 4\}$ to the labels $a, \ldots, e$. The left diagram tells which vertices of P_5 belong to the facet defined by that normal vector: Two nodes in the diagram are connected by an edge iff the vertex belonging to that pair of nodes is in the facet. The edges in the right diagram are labeled with the distance between the corresponding pair of nodes in the normal vector. This calls for an example: Setting $a = 0, \ldots,$ $e = 4$, Fig. 2(a) gives a distance vector

$$(d_{01}, d_{02}, \ldots, d_{24}, d_{34})^{\mathrm{T}} = (-1, 1, 1, -1, -1, 1, 1, -1, 1, -1)^{\mathrm{T}},$$

which is a common ray of the cones cd_{01}, cd_{12}, cd_{23}, cd_{34} and cd_{04}. Thus of the 22 facets of P_5, 12 have five vertices and 10 have six vertices.

3 The NJ cones for five taxa

In the case of five taxa there is just one unlabeled tree topology (cf. Fig. 3) and there are 15 distinct labeled trees: We have five choices for the leaf which is not part of a cherry and then three choices how to group the remaining four leaves into two pairs. For each of these labeled topologies, there are two ways in which they might be reconstructed by the NJ algorithm: There are two pairs, any one of which might be chosen in the first step of the NJ algorithm.

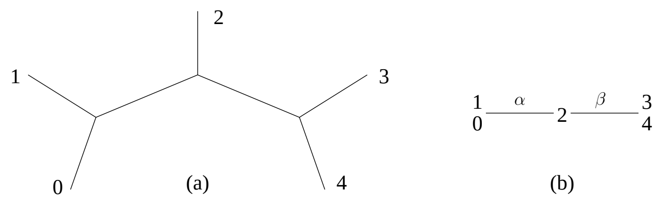


Fig. 3. (a) A tree with five taxa (b) The same tree with all edges adjacent to leaves reduced to length zero. The remaining two edges have lengths α and β.

For distinct leaf labels a, b and $c \in \{0, 1, 2, 3, 4\}$ we define $C_{ab,c}$ to be the set of all input vectors for which the cherry a-b is picked in the first step and c remains as single node not part of a cherry after the second step. For example, the tree in Fig. 3(a) is the result for all vectors in $C_{10,2} \cup C_{43,2}$. Since for each tree topology $((a, b), c, (d, e))$ (this tree topology is written in the Newick format) for distinct taxa $a, b, c, d, e \in \{0, 1, 2, 3, 4\}$, the NJ algorithm returns the same tree topology with any vector in the union of two cones $C_{ab,c} \cup C_{de,c}$, there are 30 such cones in total, and we call the set of these cones $\mathcal{C}$.

3.1 Permuting leaf labels

Because there is only one unlabeled tree topology, we can map any labeled topology to any other labeled topology by only changing the labels of the leafs. Such a change of labels also permutes the entries of the distance matrix. In this way, we get an action of the symmetric group S_5 on the input space R^{10}, and the permutation $\sigma \in S_5$ maps the cone $C_{ab,c}$ linearly to the cone $C_{\sigma(a)\sigma(b),\sigma(c)}$. Therefore any property of the cone $C_{ab,c}$ which is preserved by unitary linear transformations must be the same for all cones in $\mathcal{C}$, and it suffices to determine it for just one cone.

The action of S_5 on R^{10} decomposes into irreducible representations by

$$\underbrace{\mathbb{R} \oplus \mathbb{R}^4}_{=S} \oplus \underbrace{\mathbb{R}^5}_{=:W},$$

where the first summand is the subspace of all constant vectors and the second one is the kernel of $A^{(5)}$. The sum of these two subspaces is exactly the space S generated by the shift vectors. The third summand, which we call W, is the orthogonal complement of S and it is spanned by vectors $w_{ab,cd}$ in W with

$$(w_{ab,cd})_{xy} := \begin{cases} 1 & \text{if } xy = ab \text{ or } xy = cd \\ -1 & \text{if } xy = ac \text{ or } xy = bd \\ 0 & \text{else} \end{cases}$$

where a, b, c and d are pairwise distinct taxa in $\{0, 1, 2, 3, 4\}$ and $(w_{ab,cd})_{xy}$ is the x–yth coordinate of the vector $w_{ab,cd}$. One linearly independent subset of this is

$$w_1 := w_{01,34}, \quad w_2 := w_{12,40}, \quad w_3 := w_{23,01}, \quad w_4 := w_{34,12}, \quad w_5 := w_{40,23}.$$

Note that the 5-cycle (01234) of leaf labels cyclically permutes these basis vectors, whereas the transposition (01) acts via the matrix

$$T := \frac{1}{2}\begin{pmatrix} 2 & 1 & 1 & 1 & 1 \\ 0 & 1 & -1 & -1 & -1 \\ 0 & -1 & -1 & 1 & -1 \\ 0 & -1 & 1 & -1 & -1 \\ 0 & -1 & -1 & -1 & 1 \end{pmatrix}.$$

Because a five-cycle and a transposition generate S_5, in principle this gives us complete information about the operation.

3.2 The cone $C_{43,2}$

Since we can apply a permutation $\sigma \in S_5$, without loss of generality, we suppose that the first cherry to be picked is cherry 9, which is the cherry with leaves 3 and 4. This is true for all input vectors $\mathbf{d}$ which satisfy

$$(\mathbf{h}_{9,i}, \mathbf{d}) \geq 0 \text{ for } i = 0, \ldots, 8,$$

where the vector

$$\mathbf{h}_{ij}^{(n)} := -A^{(n)}(\mathbf{e}_i - \mathbf{e}_j)$$

is perpendicular to the hyperplane of input vector for which cherries i and j have the same Q-criterion, pointing into the direction of vectors for which the Q-criterion of cherry i is lower.

We let $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ be the first three rows of $-A^{(4)}R^{(5)}$. If $(\mathbf{r}_1, \mathbf{d})$ is maximal then the second cherry to be picked is 0-1, leaving 2 as the non-cherry node, and similarly $\mathbf{r}_2$ and $\mathbf{r}_3$ lead to non-cherry nodes 1 and 0. This allows us to define the set of all input vectors $\mathbf{d}$ for which the first picked cherry is 3-4 and the second one is 0-1:

$$C_{34,2} := \{\mathbf{d} \,|\, (\mathbf{h}_{9,i}, \mathbf{d}) \geq 0 \text{ for } i = 0, \ldots, 8, \text{ and } (\mathbf{r}_1 - \mathbf{r}_2, \mathbf{d}) \geq 0, (\mathbf{r}_1 - \mathbf{r}_3, \mathbf{d}) \geq 0\}. \tag{7}$$

We have defined this set by 11 bounding hyperplanes. However, in fact, the resulting cone has only nine facets. A computation using `polymake` [6] reveals that the two hyperplanes $\mathbf{h}_{9,1}$ and $\mathbf{h}_{9,2}$ are no longer faces of the cone, while the other nine hyperplanes in (7) give exactly the facets of the cone. That means that, while we can find arbitrarily close input vectors $\mathbf{d}$ and $\mathbf{d}'$ such that with an input $\mathbf{d}$ the NJ algorithm will first pick cherry 3-4 and with an input $\mathbf{d}'$ the NJ algorithm will first pick cherry 1-2 (or 0-2), we cannot do this in such a way that $\mathbf{d}$ will result in the labeled tree topology of Fig. 3, where 2 is the lonely leaf.

Also note that $\mathcal{C}$, the set of NJ cones which the NJ algorithm returns the same tree topology with any vector in the union of two cones $C_{ab,c} \cup C_{de,c}$, is not convex which is shown in [3]. For details of geometry and combinatorics of the NJ cones for $n = 5$, see [3].

4 The six taxa case

Note that since each of the NJ cones includes constraints for five taxa, the union of the NJ cones which gives the same tree topology is not convex. To analyze the behavior of NJ on distance matrices for six taxa, we use the action of the symmetric group as much as possible. However, in this case we get three different classes of cones which cannot be mapped onto each other by this action. We assume the cherry which is picked in the first step to consist of the nodes 4 and 5. Picking this cherry replaces these two nodes by a newly created node 45, and we have to distinguish two different cases in the second step (see Fig. 4):

- If the cherry in the second step does not contain the new node 45, we may assume the cherry to be 01. For the third step, we again get two possibilities:
  - The two nodes 45 and 01 get joined in the third step. We call the cone of input vectors for which this happens C_{I}.
  - The node 45 is joined to one of the nodes 2 and 3, without loss of generality, to 3. We call the resulting cone C_{II}.
- If the cherry in the second step contains the new node 45, we may assume the other node of this cherry to be 3, creating a new node $45-3$. In the third step, all that matters is which of the three nodes 0, 1 and 2 is joined to the node $45-3$, and we may, without loss of generality, assume this to be node 2. This gives the third type of cone, C_{III}.

The resulting tree topology for the cone C_{I} is shown in Fig. 5(a), while both C_{II} and C_{III} give the topology shown in 5(b). We now determine which elements of S_6 leave these cones fixed (stabilizer) and how many copies of each cone give the same labeled tree topology:

	C_{I}	C_{II}	C_{III}
stabilizer	$\langle(01),(23),(45)\rangle$	$\langle(01),(45)\rangle$	$\langle(01),(45)\rangle$
size of stabilizer	8	4	4
number of cones	90	180	180
cones giving same labeled topology	6	2	2
solid angle (approx.)	$2.888 \cdot 10^{-3}$	$1.848 \cdot 10^{-3}$	$2.266 \cdot 10^{-3}$

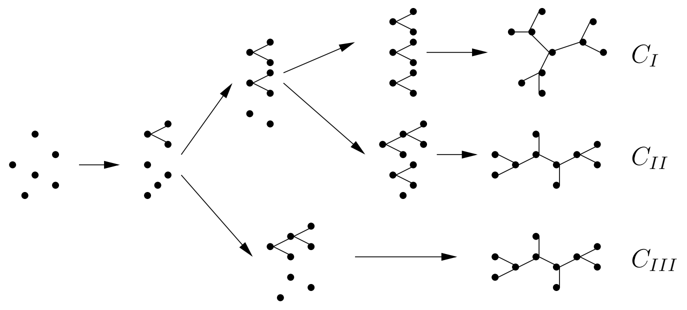


Fig. 4. The three ways of picking cherries in the six taxa case.

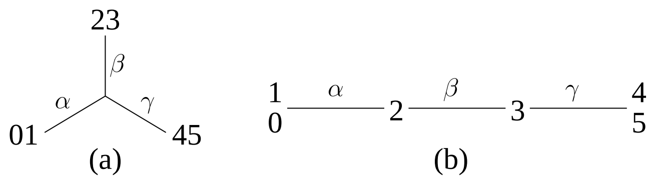


Fig. 5. The two possible topologies for trees with six leaves, with edges connecting to leaves shrunk to zero.

Thus, the input space $\mathbb{R}^{15}$ is divided into 450 cones, 90 of type I and 180 each of types II and III. There are 15 different ways of assigning labels to the tree topology in Fig. 5(a), and for each of these there are six copies of C_{I} whose union describes the set of input vectors resulting in that topology. For the topology in Fig. 5(b) we get 90 ways of assigning labels to the leaves, each corresponding to a union of two copies of C_{II} and two copies of C_{III}.

The above table also gives the solid angles of the three cones. In the five taxa case, any two cones can be mapped onto one another by the action of the symmetric group, which is unitary. Therefore all thirty cones have the same solid angle, which must be 1/30. However, in the six taxa case, we get different solid angles, and we see that about three thirds of the solid angle at the origin are taken by the cones of types II and III. Thus, on a random vector chosen according to any probability law which is symmetric around the origin (e.g., standard normal distribution), NJ will output the tree topology of Fig. 5(b) with probability about 3/4.

On the other hand, any labeled topology of the type in Fig. 5(a) belongs to six cones of type I, giving a total solid angle of $\approx 1.73 \cdot 10^{-2}$, whereas any labeled topology of the type in Fig. 5(b) belongs to two cones each of type II and III, giving a total solid angle of only $\approx 0.82 \cdot 10^{-2}$, which is half as much. This suggests that reconstructing trees of the latter topology is less robust against noisy input data.

5 Simulation results

In this section we will analyze how the tree metric for a tree and pairwise distances estimated via the maximum likelihood estimation lie in the polyhedral subdivision of the sample space. In particular, we analyze subtrees of the two parameter family of trees described by [13]. These are trees for which the NJ algorithm has difficulty in resolving the correct topology. In order to understand which cones the data lies in, we simulated 10,000 data sets on each of the two tree shapes, T_1 and T_2 (Fig. 6) at the edge length ratio, a/b = 0.03/0.42 for sequences of length 500BP under the Jukes-Cantor model [9]. We also repeated the runs with the Kimura 2-parameter model [7]. They are the cases (on eight taxa) in [13] that the NJ algorithm had most difficulties in their simulation study (also the same as in [10]). Each set of 5 sequences are generated via `evolver` from `PAML` package [17] under the given model. `evolver` generates a set of sequences given the model and tree topology using the birth-and-death process. For each set of 5 sequences, we compute first pairwise distances via the heuristic MLE method using a software `fastDNAml` [12]. To compute cones, we used `MAPLE` and `polymake`.

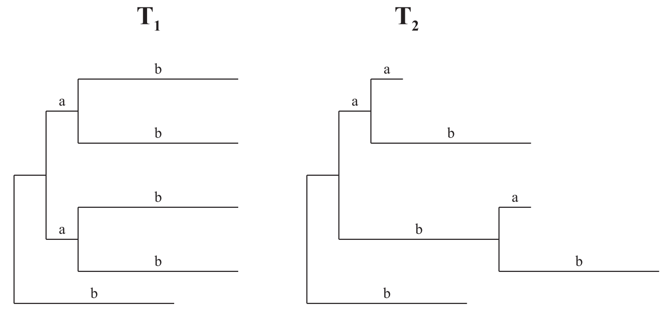


Fig. 6. T_1 and T_2 tree models which are subtrees of the tree models in [13].

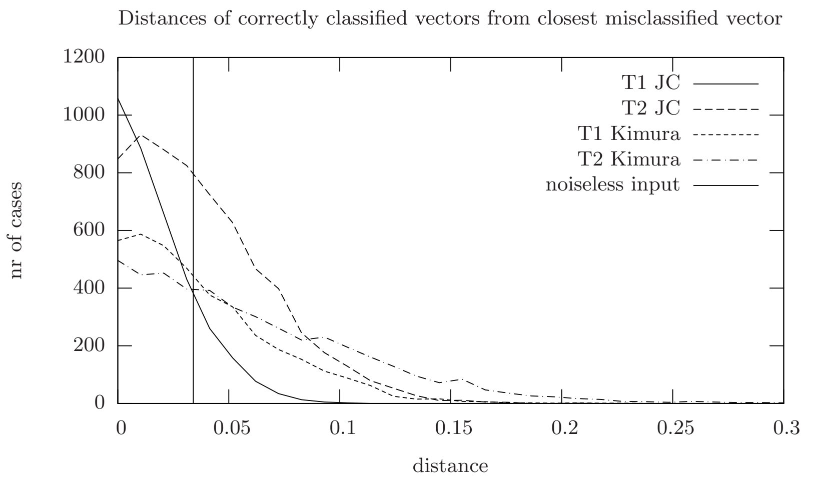
To study how far each set of pairwise distances estimated via the maximum likelihood estimation (which is a vector $\mathbf{y}$ in $\mathbb{R}^5$) lies from the cone, where the additive tree metric lies, in the sample space, we calculated the ℓ_2-distance between the cone and a vector $\mathbf{y}$.

Suppose we have a cone C defined by hyperplanes $\mathbf{n}_1, \dots, \mathbf{n}_r$, i.e.,

$$C = \{\mathbf{x} \mid (\mathbf{n}_i, \mathbf{x}) \geq 0 \text{ for } i = 1, \dots, r\},$$

and we want to find the closest point in C from some given point $\mathbf{v}$. Because C is convex, for ℓ_2-norm there is only one such point, which we call $\mathbf{u}$. If $\mathbf{v} \in C$ then $\mathbf{u} = \mathbf{v}$ and we are done. If not, there is at least one $\mathbf{n}_i$ with $(\mathbf{n}_i, \mathbf{v}) < 0$, and $\mathbf{u}$ must satisfy $(\mathbf{n}_i, \mathbf{u}) = 0$.

Now the problem reduces to a lower dimensional problem of the same kind: We project $\mathbf{v}$ orthogonally into the hyperplane H defined by $(\mathbf{n}_i, \mathbf{x}) = 0$ and call the new vector $\tilde{\mathbf{v}}$. Also, $C \cap H$ is a facet of C, and in particular a cone, so we proceed by finding the closest point in this cone from $\tilde{\mathbf{v}}$.



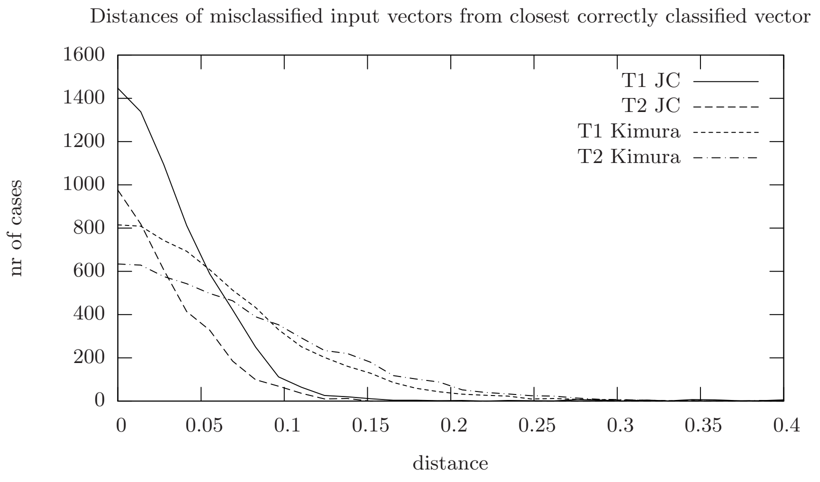


Fig. 7. Distances of correctly (top) and incorrectly (bottom) classified input vectors to the closest incorrectly/correctly classified vector.

We say an input vector (distance matrix) is *correctly classified* if the vector is in one of the cones where the vector representation of the tree metric (noiseless input) is. We say an input vector is *incorrectly classified* if the vector is in the complement of the cones where the vector representation of the tree metric is. For input vectors (distance matrices) which are correctly classified by the NJ algorithm, we compute the minimum distance to any cone giving a different tree topology. This distance gives a measure of robustness or confidence in the result, with bigger distances meaning greater reliability. The results are plotted in the left half of Fig. 7 and in Fig. 8. Note that the distance of the noiseless input, i.e., the tree metric from the tree we used for generating the data samples, gives an indication of what order of magnitude to expect with these values.

	JC		Kimura2	
	T1	T2	T1	T2
# of cases	3,581	6,441	3,795	4,467
Mean	0.0221	0.0421	0.0415	0.0629
Variance	$2.996 \cdot 10^{-4}$	$9.032 \cdot 10^{-4}$	$1.034 \cdot 10^{-3}$	$2.471 \cdot 10^{-3}$

Fig. 8. Mean and variance of the distances of correctly classified vectors from the nearest misclassified vector.

	JC		Kimura2	
	T1	T2	T1	T2
# of cases	6,419	3,559	6,205	5,533
Mean	0.0594	0.0331	0.0951	0.0761
Variance	0.0203	$7.39 \cdot 10^{-4}$	0.0411	$3.481 \cdot 10^{-3}$

Fig. 9. Mean and variance of the distances of misclassified vectors to the nearest correctly classified vector.

For input vectors to which the NJ algorithm returns with a tree topology different from the correct tree topology, we compute the distances to the two cones for which the correct answer is given and take the minimum of the two. The bigger this distance is, the further we are off. The results are shown in the right half of Fig. 7 and in Fig. 9.

From our results in Fig. 8 and Fig. 9, one notices that the NJ algorithm returns the correct tree more often with T_2 than with T_1. These results are consistent with the results in [15, 11]. Note that any possible quartet in T_1 has a smaller (or equal) length of its internal edge than in T_2 (see Fig. 6). Gascuel and Steel defined this measure as *neighborliness* [15]. Mihaescu et al. showed that the NJ algorithm returns the correct tree if it works correctly locally for the quartets in the tree [11]. The neighborliness of a quartet is one of the most important factors to reconstruct the quartet correctly, i.e., the shorter it is the more difficult the NJ algorithm returns the correct quartet. Also Fig. 7 shows that most of the input vectors lie around the boundary of cones, including the noiseless input vector (the tree metric). This shows that the tree models T_1 and T_2 are difficult for the NJ algorithms to reconstruct the correct trees. All source code for these simulations described in this paper will be available at authors' websites.

6 Open problems

Question 1. Can we use the NJ cones for analyzing how the NJ algorithm works if each pairwise distance is assumed to be of the form $D_0 + \epsilon$ where D_0 is the unknown true tree metric, and ϵ is a collection of independent normally distributed random variables? We think this would be very interesting and relevant.

Question 2. With any n, is there an efficient method for computing (or approximating) the distance between a given pairwise distance vector and the boundary of the NJ optimality region which contains it? This problem is equivalent to projecting a point inside a *polytopal complex* P onto the boundary of P. Note that the size of the complex grows very fast with n. How fast does the number of the complex grow? This would allow assigning a confidence score to the tree topology computed by the NJ algorithm.

References

1. Atteson, K.: The performance of neighbor-joining methods of phylogenetic reconstruction. Algorithmica, **25** (1999) 251–278.
2. Bryant, D.: On the uniqueness of the selection criterion in neighbor-joining. J. Classif. **22** (2005) 3–15.
3. Eickmeyer, K., Huggins, P., Pachter, L. and Yoshida, R.: On the optimality of the neighbor-joining algorithm. To appear in Algorithms in Molecular Biology.
4. Felsenstein, J.: Evolutionary trees from DNA sequences: a maximum likelihood approach. Journal of Molecular Evolution **17** (1981) 368–376.
5. Galtier, N., Gascuel, O., and Jean-Marie, A.: Markov models in molecular evolution. In *Statistical Methods in Molecular Evolution* edited by Nielsen, R., (2005) 3–24.
6. Gawrilow, E. and Joswig, M.: polymake: a framework for analyzing convex polytopes. in *Polytopes — Combinatorics and Computation*, edited by G Kalai and GM Ziegler (2000) 43–74.
7. Kimura, M.: A simple method for estimating evolutionary rates of base substitution through comparative studies of nucleotide sequences. Journal of Molecular Evolution **16** (1980) 111–120.
8. Neyman, J.: Molecular studies of evolution: a source of novel statistical problems. In *Statistical decision theory and related topics* edited by Gupta, S., Yackel, J., New York Academic Press, (1971) 1–27.
9. Jukes, H.T. and Cantor, C.: Evolution of protein molecules. In *Mammalian Protein Metabolism* edited by HN Munro, New York Academic Press, (1969) 21–32.
10. Levy, D., Yoshida, R. and Pachter, L.: Neighbor-joining with phylogenetic diversity estimates. Molecular Biology and Evolution **23** (2006) 491–498.
11. Mihaescu, R., Levy, D., and Pachter, L.: Why Neighbor-Joining Works. (2006) arXiv:cs.DS/0602041.
12. Olsen, G.J., Matsuda, H., Hagstrom, R., and Overbeek, R.: fastDNAml: A tool for construction of phylogenetic trees of DNA sequences using maximum likelihood. Comput. Appl. Biosci. **10** (1994) 41–48.
13. Ota, S. and Li, WH.: NJML: A Hybrid algorithm for the neighbor-joining and maximum likelihood methods. Molecular Biology and Evolution **17** 9 (2000) 1401–1409.
14. Saitou, N. and Nei, M.: The neighbor joining method: a new method for reconstructing phylogenetic trees. Molecular Biology and Evolution **4** (1987) 406–425.
15. Gascuel, O. and Steel, M.: Neighbor-joining revealed. Molecular Biology and Evolution **23** (2006) 1997-2000.
16. Studier, J.A. and Keppler, K.J.: A note on the neighbor-joining method of Saitou and Nei. Molecular Biology and Evolution **5** (1988) 729–731.
17. Yang, Z: PAML: A program package for phylogenetic analysis by maximum likelihood. *CABIOS* **15** (1997) 555–556.
18. Yang, Z.: Complexity of the simplest phylogenetic estimation problem. Proceedings of the Royal Society B: Biological Sciences **267** (2000) 109-116.
19. Ziegler, G.: *Lectures on Polytopes.* Springer-Verlag 1995.