The Neighbor Joining
Tree-Reconstruction Technique
Lecture 12

background:: Durbin et al 7.3, Gusfield 17.1-17.3, Setubal&Meidanis 6.5
©Shlomo Moran & Ilan Gronau
Recall: Distance-Based Reconstruction:

- **Input:** distances between all taxon-pairs
- **Output:** a tree (edge-weighted) best-describing the distances

\[
D = \begin{pmatrix}
0 & 9 & 21 & 19 & 15 & 16 \\
0 & 22 & 20 & 16 & 17 \\
0 & 18 & 14 & 15 \\
0 & 8 & 9 \\
0 & 3 \\
0
\end{pmatrix}
\]

Basic requirement: consistency – if the distances fit a weighted tree, return this tree.
1st case: The Tree is Ultrametric

An ultrametric tree can be reconstructed from its distance matrix by UPGMA (join closest neighbors & update matrix).

UPGMA has an optimal implementation of $O(n^2)$ time, using Nearest Neighbor Chains.

*(note: $n$ is the number of species, thus the size of the input is $\Theta(n^2)$)*
UPGMA fails on general trees

When the input distances are defined by non-ultrametric trees, UPGMA will construct (an incorrect) ultrametric tree.

Needed: Consistent reconstruction algorithms for general weighted trees.
Requirement from Distance-based Tree-Reconstruction Algorithms

1. Consistency: If the input metric is additive, i.e. fits a tree metric, the returned tree should be the (unique) tree which fits this metric.
2. Efficiency: poly-time, preferably no more than $O(n^3)$.
3. Robustness: if the input matrix is “close” to additive, the algorithm should return the correct tree. We distinguish between
   - Robust in theory (Atteson criteria – in the tutorial)
   - Robust in practice (eg in simulations)

A natural family of algorithms which satisfy 1 and 2 is called “Neighbor Joining”, and is similar in structure to the UPGMA algorithm.
The Neighbor Joining Tree-Reconstruction Scheme

Start with $n$ singletons, and each iteration join two neighboring leaves (cherries):

- Select pair $i,j$ and replace them by a new vertex $v$
- Make $v$ the parent of the cherries $i,j$
- Remove $i,j$ and insert $v$ to the distance matrix
- Method recursively applied on reduced matrix

Two issues:

- How do we find $i,j$ which are indeed cherries?
- How do we compute distances from the new vertex $v$?
Neighbor Selecting

How can we find (from distances alone) a pair of nodes which are neighboring leaves (“cherries”)?
Unlike in ultrametric trees, closest nodes aren’t necessarily cherries.

Idea: instead of using distances, use “LCA depths”
LCA Depth

Let $i,j$ be leaves in $T$, and let $r \neq i,j$ be a vertex in $T$. 
$LCA_r(i,j)$ is the Least Common Ancestor of $i$ and $j$ when $r$ is viewed as a root.
If $r$ is fixed we just write $LCA(i,j)$.
$d_T(r,LCA(i,j))$ is the “depth of $LCA_r(i,j)$”.
Matrix of LCA Depths

A weighted tree $T$ with a designated root $r$ defines a matrix of LCA depths:

$$d_T(r, LCA(A, D)) = 3$$
Let $T$ be a weighted tree, with a root $r$. For leaves $i,j \neq r$, let $L(i,j) = d_T(r, LCA(i,j))$. Then $i, j$ are cherries with parent $v$, iff:

$$\forall k \neq i, j: L(i, j) \geq L(i, k), L(j, k)$$

In other words, $i$ and $j$ are cherries iff they have the same deepest ancestor. In this case we say that $i$ and $j$ are mutual deepest neighbors. We use this to reconstruct trees from LCA matrices, defined next.
LCA Matrices

Definition: A symmetric nonnegative matrix $L$ is an LCA matrix iff

1. For all $i$, $L(i,i) = \max_j L(i,j)$.
2. It satisfies the “3 points condition”: Any subset of 3 indices can be labeled $i, j, k$ s.t. $L(i,j) = L(i,k) \leq L(j,k)$ (i.e., the minimal value appears twice)

\[
\begin{array}{ccc}
  j & i & k \\
  j & 8 & 0 & 0 & 3 & 5 \\
  & 9 & 5 & 0 & 0 \\
  & & 8 & 0 & 0 \\
  i & & & 7 & 3 \\
  k & & & & 7 \\
\end{array}
\]
LCA Matrices $\iff$ Weighted Rooted Trees

**Theorem:** The following conditions are equivalent for a symmetric matrix $L$ over a set $S$:

1. $L$ is an LCA matrix.
2. There is a weighted tree $T$ with a root $r$ and leaves-set $S$, s.t. for each $i,j$ in $S$:

   $$L(i,j) = d_T(r, LCA(i,j))$$
Weighted Tree $T$ rooted at $r \rightarrow$ LCA Matrix:

$L:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

$L(A,A) = 7 = d_T(r, A)$

$L(A,B) = 4 = d_T(r, LCA(A,B))$
Proof of this direction is identical to the proof that an ultrametric matrix corresponds to distances in an ultrametric tree (that we saw last week).

We will see another proof, by an algorithm that construct a tree from LCA matrix.
Finding Cherries in LCA Matrices

Observation: In an LCA matrix $L$:

If $i$ and $j$ are mutual deepest neighbors

Then $\forall k \neq i, j : L(i, k) = L(j, k)$

(ie all non-diagonal entries in rows $i, j$ except $L(i, j)$ are equal)

Proof: By the following 2 facts:

1. $L(i, k), L(j, k) \leq L(i, j)$,
2. By the 3PC, the two minimal values in $\{L(i, k), L(j, k), L(i, j)\}$ are equal.
DLCA algorithm for LCA matrices:

Input: an LCA matrix $L$ over a set $S$.
Output: A tree $T$ with leaves in $S \cup \{r\}$, such that $\forall i, j \in S : L(i, j) = d_r(r, LCA(i, j))$

- **Stopping condition:** If $S = \{i\}$ and $L = [w]$ return the tree:

- **Neighbor Selection:** Choose mutual deepest neighbors $i, j$,

- **Reduction:** In the matrix $L$, delete rows $i, j$ and add new row $v$, with values:
  
  $L(v,v) \leftarrow L(i,j)$;

  For $k \neq v$, $L(v,k) \leftarrow L(i,k)$ //recall that $L(i,k) = L(j,k)$//

  Recursively call DLCA on the reduced matrix

- **Neighbor connection:** In the returned tree, connect $i$ and $j$ to $v$, with edge weights:
  
  $w(v,i) \leftarrow L(i,i) - L(i,j)$
  
  $w(v,j) \leftarrow L(j,j) - L(i,j)$
One Iteration of DLCA:

Replace rows $A, E$ by $V$.

Neighbor Connection (at the end)
**Correctness of DLCA:**

**Theorem:** Let $L$ be an $LCA$ matrix over a set $S$. Then the DLCA algorithm returns a weighted tree $T$ with root $r$ ($r$ not in $S$), and leaves-set $S$, and

$$\forall i, j \in S, L(i, j) = d_T(r, LCA(i, j))$$

**Proof sketch:** Induction on the matrix size, i.e. the cardinality of $|S|$.

If $|S|=1$ then the theorem is trivial.

Induction step: Assume correctness for $n$, and consider input matrix $L$ of size $n+1$. Then after the 1st reduction step, the reduced matrix, $L'$, is also a $LCA$ matrix over $S'=S\setminus\{i,j\} \cup \{v\}$ (verify this!).

By induction, the recursive call of $DLCA$ on $L'$ returns a tree $T'$ with leaves $S'$ and root $r$, which realizes the distances in $L'$.

By the connection formula, adding $i,j$ as children of $v$ results with a tree $T$ with leaves $S$ and root $r$ which realizes the distances in $L$. 
DLCA on “noisy” input matrices

Input: A symmetric nonnegative matrix $L$ over a set $S$.
Output: A tree with a root $r$ // $L(i,j)$ should be close to $d_T(r,LCA(i,j))$ for all $i,j$.//

- Stopping condition: If $L=[w]$ return a tree with a single edge of weight $w$
- Neighbor Selection:
  Choose mutual deepest neighbors $i,j$.
- Reduction:
  In the matrix $L$, replace rows $i,j$ by a new row $v$, with values:
  
  $L(v,v) \leftarrow L(i,j);$ 
  
  For $k \neq v$, $L(v,k) \leftarrow \alpha L(i,k)+(1-\alpha)L(j,k)$ // $0 \leq \alpha \leq 1$//
  
  Recursively call DLCA on the reduced matrix
- Neighbor connection:
  In the returned tree, connect $i$ and $j$ to $v$, with edge weights:
  
  $w(v,i) \leftarrow \max \{0, L(i,i)-L(i,j)\}$
  
  $w(v,j) \leftarrow \max \{0, L(j,j)-L(i,j)\}$
The algorithm has $n$-1 iterations
Each iteration:
1. Mutual Deepest Neighbors are selected
2. Two rows are deleted, and one row is added to the matrix.

Step 2 requires $O(n)$ operations per iteration, total $O(n^2)$. 
Step 1 (finding mutual deepest neighbors) can also be done in total $O(n^2)$, time – as in the UPGMA algorithm we saw last week. We sketch it again next:
Implementation by Deepest Neighbor Chains (cont)

Finding mutual deepest neighbors in $O(n^2)$ total time:

- Find maximal off-diagonal entry $(i_r, i_{r+1})$ in row $i_r$ (i.e., $(i_r, i_{r+1})$ are mutual deepest neighbors).
- Stop if $i_{r+1}$ is also deepest neighbor of $i_r$.
- Otherwise, continue.

Complete NN chain:

- $i_{r+1}$ is a Deepest Neighbor of $i_r$
- Final pair $(i_{r-1}, i_r)$ are mutual deepest neighbors.
Implementation of Deepest Neighbor Chains (cont.)

An $\theta(n^2)$ implementation using Nearest Neighbors Chains:

- Extend a DN chain until it is complete.
- Select final pair for joining, and remove them from chain.

Note: in the new matrix, the remaining chain is still DN chain

Total running time is $O(n^2)$, as in the NN chain of UPGMA
Running DLCA from (Additive) Distance Matrix D:

When the input is an (additive) distance matrix $D$, we apply on $D$ the following $LCA$ reduction to obtain an ($LCA$) matrix $L$:

- Choose any leaf as a root $r$
- Set for all $i,j$: $L(i,j) = \frac{1}{2}(D(r,i) + D(r,j) - D(i,j))$
- Run DLCA on $L$.

Important observation: If $D$ is an additive distance matrix corresponding to a tree $T$, then $L$ is an $LCA$ matrix in which

$L(i,j) = d_T(r, LCA(i,j))$
Example

A tree with the corresponding additive distance matrix

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>8</td>
<td>7</td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>B</td>
<td>-</td>
<td>9</td>
<td>14</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>-</td>
<td></td>
<td>11</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>-</td>
<td></td>
<td>-</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>-</td>
<td></td>
<td>-</td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>
Use $D$ to compute an LCA matrix $L$

$$L(i, j) = \frac{1}{2}(D(r, i) + D(r, j) - D(i, j))$$

$$L(A, B) = \frac{1}{2}(7 + 9 - 8) = 4$$
The relation of $L$ to the original tree:

\[ L(A,A) = 7 = d_T(r, A) \]

\[ L(A,B) = 4 = d_T(r, LCA(A,B)) \]
Discussion of DLCA

- Consistency: If the input matrix $L$ is an LCA matrix, then the output is guaranteed to be the unique weighted tree which realizes the LCA distances in $L$.
- Complexity: It can be implemented in optimal $O(n^2)$ time.
- Robustness to noise:
  - Theoretical: it has optimal robustness when $0 \leq \alpha \leq 1$.
  - Practical: it is inferior to other NJ algorithms – possibly due to the fact that its neighbor-selection criterion is biased by the selected root $r$.

Various Neighbor-Joining algorithms differ mainly by their “neighbor selection criterion”.
Next we present a neighbor selection criterion which is known to be most robust to noise in practice.
Saitou & Nei’s Neighbor Joining Algorithm (1987)

- ~13,000 citations (*Science Citation Index*)
- Implemented in numerous phylogenetic packages
- Fastest implementation - \( \theta(n^3) \)
- Usually referred to as “the NJ algorithm”
- Identified by its neighbor selection criterion

select \( i, j \) which maximize the sum

\[
Q(i, j) = \sum_r D(r, i) + \sum_r D(r, j) - (n - 2)D(i, j)
\]
Consistency of Saitou&Nei method

Let $T$ be a weighted tree.

For a leaf $i$, let $r_i = \sum_{u \text{ is a leaf}} d(i,u)$.

For leaves $i, j$:

$Q(i, j) = [r_i + r_j - (n-2)d(i, j)]$

**Theorem** (Saitou&Nei) Assume all edge weights of $T$ are positive. If $Q(i,j) = \max_{\{i',j'\}} Q(i',j')$, then $i$ and $j$ are cherries in the tree.
Expressing Saitou & Nei selection criterion in terms of LCA distances

Saitou & Nei’s Selection criterion:
Select $i,j$ which maximize

$$Q(i, j) = \sum_r D(r, i) + \sum_r D(r, j) - (n - 2)D(i, j)$$

$$= 2 \cdot \left[ \sum_{r \neq i, j} LCA_r(i, j) + D(i, j) \right]$$

Intuition: NJ “tries” to selects taxon-pairs with average deepest LCA

The addition of the term $D(i, j)$ is needed to make the formula consistent.

Next we prove the above equality.
Proof of equality

\[ Q(i, j) = \left( d(i, j) + \sum_{u \neq i, j} d(i, u) \right) + \left( d(j, i) + \sum_{u \neq i, j} d(j, u) \right) - (n - 2)d(i, j) \]

\[ = \sum_{u \neq i, j} [d(i, u) + d(j, u) - d(i, j)] + 2d(i, j) \]

\[ = 2 \left[ d(i, j) + \sum_{u \neq i, j} d(u, LCA_u(i, j)) \right] \]
Seitou&Nei proof (cont.)

It remains to show that
\[ Q(i, j) \left[ = d(i, j) + \sum_{u \neq i, j} d(u, LCA_u(i, j)) \right] \]

is maximized only when \( i, j \) are cherries.

For a vertex \( i \), and an edge \( e \):
\[ N_i(e) = |\{u : e \text{ is on } path(i,u)\}| \]

Then:

\[ Q(i, j) = d(i, j) + \sum_{u \neq i, j} d(u, LCA_u(i, j)) = \sum_{e \in path(i, j)} w(e) + \sum_{e \notin path(i, j)} N_i(e)w(e) \]

**Note:** If \( e' \) is a “leaf edge”, then \( w(e') \) is added exactly once to \( Q(i, j) \).
Assume for contradiction that $Q(i,j)$ is maximized for $i,j$ which are not cherries.

Let (see the figure below):
• $path(i,j) = (i,...,k,j)$.
• $T_1 = \text{the subtree rooted at } k$. WLOG that $T_1$ has at most $n/2$ leaves.
• $T_2 = T \setminus T_1$.

Let $i',j'$ be any two cherries in $T_1$. We will show that $Q(i',j') > Q(i,j)$. 
Proof that $Q(i',j') > Q(i,j)$:

$$Q(i, j) = \sum_{e \in p(i,j)} w(e) + \sum_{e \notin p(i,j)} N_i(e)w(e)$$

$$Q(i', j') = \sum_{e \in p(i',j')} w(e) + \sum_{e \notin p(i',j')} N_i'(e)w(e)$$

Each leaf edge $e$ adds $w(e)$ both to $Q(i,j)$ and to $Q(i',j')$, so we can ignore the contribution of leaf edges to both $Q(i,j)$ and $Q(i',j')$. 

Seitou&Nei proof (cont.)
Contribution of \textit{internal} edges to $Q(i,j)$ and to $Q(i',j')$

<table>
<thead>
<tr>
<th>Location of internal edge $e$</th>
<th>$# w(e)$ added to $Q(i,j)$</th>
<th>$# w(e)$ added to $Q(i',j')$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e \in \text{path}(i,j)$</td>
<td>1</td>
<td>$N_i(e) \geq 2$</td>
</tr>
<tr>
<td>$e \in \text{path}(i',j)$</td>
<td>$N_i(e) &lt; n/2$</td>
<td>$N_i(e) \geq n/2$</td>
</tr>
<tr>
<td>$e \in T \setminus \text{path}(i,i')$</td>
<td>$N_i(e) = N_i(e)$</td>
<td></td>
</tr>
</tbody>
</table>

Since there is at least one internal edge $e$ in $\text{path}(i,j)$, $Q(i',j') > Q(i,j)$. QED
Complexity of Seitou&Nei NJ Algorithm

**Initialization:** $\theta(n^2)$ to compute $Q(i,j)$ for all $i,j \in L$.

**Each Iteration:**
- $O(n^2)$ to find the maximal $Q(i,j)$, and to update the values of $Q(x,y)$

**Total:** $O(n^3)$

(There is a variant [Elias&Lagergren 2005] of this algorithm of complexity $O(n^2)$. This variant uses the same selection criterion but it saves in updating. Consequently, its output on non-additive matrices can be different from the output of NJ.)
A characterization of additive metrics: the 4 points condition

Distances on 3 objects are always realizable by a (unique) tree with one internal node.

For instance

\[ c = d(k, m) = \frac{1}{2} [d(i, k) + d(j, k) - d(i, j)] \geq 0 \]
How about four objects?

L=4: Not all distance metrics on 4 objects are additive: eg, there is no tree which realizes the below distances.

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>j</th>
<th>k</th>
<th>l</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>j</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td></td>
<td>0</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>l</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
The Four Points Condition

A necessary condition for distances on four objects to be additive: its objects can be labeled \(i,j,k,l\) so that:

\[
d(i,k) + d(j,l) = d(i,l) + d(k,j) \geq d(i,j) + d(k,l)
\]

Proof: By the figure...

\{\{i,j\},\{k,l\}\} is a “split” of \{i,j,k,l\}.
Definition: A distance metric satisfies the four points condition iff any subset of four objects can be labeled $i, j, k, l$ so that:

$$d(i,k) + d(j,l) = d(i,l) + d(k,j) \geq d(i,j) + d(k,l)$$
Equivalence of the Four Points Condition and the Three Points Condition

\[ 4PC \iff d(r, k) + d(j, l) = d(r, l) + d(j, k) \geq d(r, j) + d(k, l) \]

\[ \Downarrow \]

\[ d(r, k) - d(j, k) = d(r, l) - d(j, l) \text{ and } d(r, l) - d(k, l) \geq d(r, j) - d(j, k) \]

\[ \Downarrow \]

\[ 2d(r, \text{LCA}(k,l)) = d(r, k) + d(r, l) - d(k, l) \geq d(r, k) + d(r, j) - d(j, k)[= 2d(r, \text{LCA}(j,k))] \]

\[ = d(r, j) + d(r, l) - d(j, l) = 2d(r, \text{LCA}(j,l)) \iff 3PC \]

i.e., a matrix \( D \) satisfies the 4PC on all quartets that include \( r \) iff the LCA reduction applied on \( D \) and \( r \) outputs a matrix \( L \) which satisfies the 3PC.
The Four Points Condition

**Theorem:** The following 3 conditions are equivalent for a distance matrix $D$ on a set $M$ of $L$ objects

1. $D$ is additive
2. $D$ satisfies the four points condition for all quartets in $M$.
3. There is an object $r$ in $M$, s.t. $D$ satisfies the 4 points condition for all quartets **that include** $r$. 

![Diagram]

\[ i \quad j \quad l \quad k \]
The Four Points Condition

Proof: we’ll show that $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$.

1 $\rightarrow$ 2
Additivity $\Rightarrow$ 4P Condition satisfied by all quartets: By the figure...

2 $\rightarrow$ 3: trivial
**Proof that $3 \Rightarrow 1$**

4PC on all quartets which include $r \Rightarrow$ additivity

The proof is as follows:

- All quartets in $D$ which include $r$ satisfy the 4PC $\Rightarrow$
- The matrix $L$ obtained by applying the LCA reduction on $D$
  and $r$ is an LCA matrix $\Rightarrow$
- The tree $T$ output by running DLCA on $L$ realizes the LCA
  depths in $L$ $\Rightarrow$
- $T$ realizes the distances in $D$. 