Course Announcements

HW 3 will be released Oct 25 – due Nov 1 by 11:59pm

Project proposal due on Nov 3
(Motivation, Datasets/papers, Planned method/experiments, Timeline)

Project report due on Dec 22
Outline

• Introduction
• Hierarchical clustering
• Additive distance phylogeny
• Four point condition
• Neighbor joining

Reading:
• Chapter 10.2 and 10.5-10.8 in Jones and Pevzner
Alignments and Trees

Tree / star alignment

Guide tree in progressive alignment

Tree topology represents similarity/distance between sequences

Biological sequences typically come from the present
Evolutionary Studies and Phylogenies

• Since Darwin’s book (1859) until 1960s: Phylogeny reconstruction from anatomical features

• Subjective observations led to inconclusive/incorrect phylogenies
Evolutionary Studies and Phylogenies

• Subjective observations led to inconclusive/incorrect phylogenies

Example
• Giant pandas look like bears but have features that are unusual for bears and typical for raccoons
• In 1985, Steven O’Brien and colleagues solved the giant panda classification problem using DNA sequences and algorithms
Out of Africa Hypothesis claims that our most ancient ancestor lived in Africa roughly 200,000 years ago.

Evolutionary Tree of Humans
Evolutionary Tree of Species

http://bix.ucsd.edu/bioalgorithms/

Question: What are the evolutionary relationships between species?
Evolutionary Tree of a Tumor

Phylogenetic Tree Reconstruction

Mouse: \textbf{ACAGTGACGCACACACG}T
Gorilla: \textbf{CCTGTGACGTAACAAACG}A
Chimpanzee: \textbf{CCTGTGAGGTAAGCA}AACG
Human: \textbf{CCTGTGAGGTAAGC}ACACG

Distance Metric

\[
\begin{array}{cccc}
  v_1 & v_2 & v_3 & v_4 \\
  v_1 & - & .17 & - \\
  v_2 & .87 & .28 & - \\
  v_3 & .59 & .33 & .62 & - \\
\end{array}
\]

Question: Given sequence data, how to reconstruct tree?
Outline

• Introduction
• Hierarchical clustering
• Additive distance phylogeny
• Four point condition
• Neighbor joining

Reading:
• Chapter 10.2 and 10.5-10.8 in Jones and Pevzner
Clustering

Given:
(1) $n \times n$ matrix $D = [d_{i,j}]$

Want:
(1) Homogeneity within clusters
(2) Separation between clusters

Bad:
- Close distances from points in separate clusters
- Far distances from points in the same cluster

Good:
- Separation between clusters

Hierarchical Clustering

Organize elements into a tree such that:

• Leaves are elements
• Paths between leaves represent pairwise element distance
• Similar elements lie within same subtrees
Hierarchical Clustering

1. Hierarchical Clustering \( (D, n) \)
2. Form \( n \) clusters each with one element
3. Construct a graph \( T \) by assigning one vertex to each cluster
4. while there is more than one cluster
5. Find the two closest clusters \( C_1 \) and \( C_2 \)
6. Merge \( C_1 \) and \( C_2 \) into new cluster \( C \) with \( |C_1| + |C_2| \) elements
7. **Compute distance from \( C \) to all other clusters**
8. Add a new vertex \( C \) to \( T \) and connect to vertices \( C_1 \) and \( C_2 \)
9. Remove rows and columns of \( D \) corresponding to \( C_1 \) and \( C_2 \)
10. Add a row and column to \( D \) corresponding to the new cluster \( C \)
11. return \( T \)
Hierarchical Clustering

1. Hierarchical Clustering \((D, n)\)
2. Form \(n\) clusters each with one element
3. Construct a graph \(T\) by assigning one vertex to each cluster
4. \textbf{while} there is more than one cluster
5. Find the two closest clusters \(C_1\) and \(C_2\)
6. Merge \(C_1\) and \(C_2\) into new cluster \(C\) with \(|C_1| + |C_2|\) elements
7. \textbf{Compute distance from} \(C\) \textbf{to all other clusters}
8. Add a new vertex \(C\) to \(T\) and connect to vertices \(C_1\) and \(C_2\)
9. Remove rows and columns of \(D\) corresponding to \(C_1\) and \(C_2\)
10. Add a row and column to \(D\) corresponding to the new cluster \(C\)
11. return \(T\)
Hierarchical Clustering

1. **Hierarchical Clustering** \((D, n)\)
2. Form \(n\) clusters each with one element
3. Construct a graph \(T\) by assigning one vertex to each cluster
4. **while** there is more than one cluster
5. Find the two closest clusters \(C_1\) and \(C_2\)
6. Merge \(C_1\) and \(C_2\) into new cluster \(C\) with \(|C_1| + |C_2|\) elements
7. **Compute distance from** \(C\) **to all other clusters**
8. Add a new vertex \(C\) to \(T\) and connect to vertices \(C_1\) and \(C_2\)
9. Remove rows and columns of \(D\) corresponding to \(C_1\) and \(C_2\)
10. Add a row and column to \(D\) corresponding to the new cluster \(C\)
11. return \(T\)
Hierarchical Clustering

1. Hierarchical Clustering \((D, n)\)
2. Form \(n\) clusters each with one element
3. Construct a graph \(T\) by assigning one vertex to each cluster
4. while there is more than one cluster
5. Find the two closest clusters \(C_1\) and \(C_2\)
6. Merge \(C_1\) and \(C_2\) into new cluster \(C\) with \(|C_1| + |C_2|\) elements
7. Compute distance from \(C\) to all other clusters
8. Add a new vertex \(C\) to \(T\) and connect to vertices \(C_1\) and \(C_2\)
9. Remove rows and columns of \(D\) corresponding to \(C_1\) and \(C_2\)
10. Add a row and column to \(D\) corresponding to the new cluster \(C\)
11. return \(T\)
Hierarchical Clustering

1. Hierarchical Clustering \((D, n)\)
2. Form \(n\) clusters each with one element
3. Construct a graph \(T\) by assigning one vertex to each cluster
4. while there is more than one cluster
5. Find the two closest clusters \(C_1\) and \(C_2\)
6. Merge \(C_1\) and \(C_2\) into new cluster \(C\) with \(|C_1| + |C_2|\) elements
7. Compute distance from \(C\) to all other clusters
8. Add a new vertex \(C\) to \(T\) and connect to vertices \(C_1\) and \(C_2\)
9. Remove rows and columns of \(D\) corresponding to \(C_1\) and \(C_2\)
10. Add a row and column to \(D\) corresponding to the new cluster \(C\)
11. return \(T\)
Hierarchical Clustering

1. Hierarchical Clustering \((D, n)\)
2. Form \(n\) clusters each with one element
3. Construct a graph \(T\) by assigning one vertex to each cluster
4. while there is more than one cluster
5. Find the two closest clusters \(C_1\) and \(C_2\)
6. Merge \(C_1\) and \(C_2\) into new cluster \(C\) with \(|C_1| + |C_2|\) elements
7. **Compute distance from \(C\) to all other clusters**
8. Add a new vertex \(C\) to \(T\) and connect to vertices \(C_1\) and \(C_2\)
9. Remove rows and columns of \(D\) corresponding to \(C_1\) and \(C_2\)
10. Add a row and column to \(D\) corresponding to the new cluster \(C\)
11. return \(T\)

Definition of distance between clusters affects clustering!
Hierarchical Clustering – Linkage Criteria

<table>
<thead>
<tr>
<th>Names</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum or <strong>complete-linkage clustering</strong></td>
<td>[ \max { d(a, b) : a \in A, b \in B } ].</td>
</tr>
<tr>
<td>Minimum or <strong>single-linkage clustering</strong></td>
<td>[ \min { d(a, b) : a \in A, b \in B } ].</td>
</tr>
<tr>
<td>Mean or average linkage clustering, or UPGMA</td>
<td>[ \frac{1}{</td>
</tr>
<tr>
<td>Centroid linkage clustering, or UPGMC</td>
<td>[ | c_s - c_t | ] where ( c_s ) and ( c_t ) are the centroids of clusters ( s ) and ( t ), respectively.</td>
</tr>
<tr>
<td><strong>Minimum energy clustering</strong></td>
<td>[ \frac{2}{nm} \sum_{i,j=1}^{n,m} | a_i - b_j |<em>2 - \frac{1}{n^2} \sum</em>{i,j=1}^{n} | a_i - a_j |<em>2 - \frac{1}{m^2} \sum</em>{i,j=1}^{m} | b_i - b_j |_2 ]</td>
</tr>
</tbody>
</table>

Outline

• Introduction
• Hierarchical clustering
• Additive distance phylogeny
• Four point condition
• Neighbor joining

Reading:
• Chapter 10.2 and 10.5-10.8 in Jones and Pevzner
Phylogenetic Tree Reconstruction

**Mouse:** 
ACAGTGACGGCCACACACG

**Gorilla:** 
CCGTGACGTAACAAACGA

**Chimpanzee:** 
CCGTGAGGTAGCAAACGA

**Human:** 
CCGTGAGGTAGCACAACGA

**Distance Metric**

<table>
<thead>
<tr>
<th></th>
<th>v1</th>
<th>v2</th>
<th>v3</th>
<th>v4</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v2</td>
<td>.17</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>v3</td>
<td>.87</td>
<td>.28</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>v4</td>
<td>.59</td>
<td>.33</td>
<td>.62</td>
<td>-</td>
</tr>
</tbody>
</table>

**Question:** Given sequence data, how to reconstruct tree?
A **distance** (metric) on a set $X$ is a function $d : X \times X \rightarrow \mathbb{R}$ s.t. for all $x, y, z \in X$:

i. $d(x, y) \geq 0$ [non-negativity]
ii. $d(x, y) = 0$ if and only if $x = y$ [identity of indiscernibles]
iii. $d(x, y) = d(y, x)$ [symmetry]
iv. $d(x, y) \leq d(x, z) + d(z, y)$ [triangle inequality]

**Examples:**

- $X = \mathbb{R}$ and $d(x, y) = |x - y|
- $X = \Sigma^*$ and $d$ is Hamming distance
- $X = \Sigma^*$ and $d$ is edit distance
Alignment vs. Distance Matrices

Mouse: \textcolor{red}{A}C\textcolor{orange}{A}\textcolor{green}{G}\textcolor{blue}{T}\textcolor{red}{G}\textcolor{orange}{A}\textcolor{green}{C}\textcolor{blue}{G}\textcolor{red}{T}\textcolor{orange}{G}\textcolor{green}{A}\textcolor{blue}{C}\textcolor{red}{A}\textcolor{orange}{G}\textcolor{green}{C}\textcolor{blue}{A}\textcolor{red}{C}\textcolor{orange}{G}\textcolor{green}{T}

Gorilla: \textcolor{red}{C}C\textcolor{orange}{T}G\textcolor{green}{T}G\textcolor{blue}{A}\textcolor{red}{C}\textcolor{orange}{G}\textcolor{green}{T}A\textcolor{blue}{A}\textcolor{red}{C}\textcolor{orange}{A}\textcolor{green}{A}\textcolor{blue}{A}\textcolor{red}{A}\textcolor{orange}{C}\textcolor{green}{GA}\textcolor{blue}{A}\textcolor{red}{C}\textcolor{orange}{G}\textcolor{green}{T}

Chimpanzee: \textcolor{red}{C}C\textcolor{orange}{T}G\textcolor{green}{T}G\textcolor{blue}{A}\textcolor{red}{C}\textcolor{orange}{G}\textcolor{green}{T}A\textcolor{blue}{G}\textcolor{red}{C}\textcolor{orange}{A}\textcolor{green}{A}\textcolor{blue}{A}\textcolor{red}{A}\textcolor{orange}{C}\textcolor{green}{GA}\textcolor{blue}{A}\textcolor{red}{C}\textcolor{orange}{G}\textcolor{green}{T}

Human: \textcolor{red}{C}C\textcolor{orange}{T}G\textcolor{green}{T}G\textcolor{blue}{A}\textcolor{red}{C}\textcolor{orange}{G}\textcolor{green}{T}A\textcolor{blue}{G}\textcolor{red}{C}\textcolor{orange}{A}\textcolor{green}{A}\textcolor{blue}{A}\textcolor{red}{A}\textcolor{orange}{C}\textcolor{green}{GA}\textcolor{blue}{A}\textcolor{red}{C}\textcolor{orange}{G}\textcolor{green}{T}

Genes of length $m$ in $n$ species

Easy: use (weighted) edit distance

$n \times n$ distance matrix

Reverse transformation not possible due to loss of information
Distances in Trees

Given a tree $T$ with positive edge weights $w(e)$, the **tree distance** $d_T(i, j)$ between two leaves $i$ and $j$ is the sum of weights of edges on the unique path from $i$ to $j$.

$$d_T(1, 4) = 12 + 13 + 14 + 17 + 13 = 69$$
General Distance vs. Tree Distance

Tree distance $d_T(i, j)$ not necessarily equal to $d_{i,j}$ as given by distance matrix obtained from alignment
Fitting a Tree to a Given Distance Matrix

• Given $n$ species, we can compute $n \times n$ distance matrix $D = [d_{i,j}]$
• Evolution of these $n$ species is described by an unknown tree
• We need an algorithm to construct tree $T$ that best fits $D$
Fitting a Tree to a Given Distance Matrix

• Given $n$ species, we can compute $n \times n$ distance matrix $D = [d_{i,j}]$
• Evolution of these $n$ species is described by an unknown tree
• We need an algorithm to construct tree $T$ that best fits $D$

Distance-Based Phylogeny: Given $n \times n$ distance matrix $D = [d_{i,j}]$, find edge-weighted tree $T$ with $n$ leaves that best fits $D$

Question: How to define ‘best fit’?
Additive Distance Matrices

Matrix $D$ is ADDITIVE if there exists a tree $T$ with $d_{ij}(T) = D_{ij}$

<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>0</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Diagram: A connected tree with distances one, two, and one.
Additive Distance Matrices

Matrix $D$ is ADDITIVE if there exists a tree $T$ with $d_{ij}(T) = D_{ij}$

NON-ADDITIVE otherwise

This is a constructive definition
A Small and a Large Problem

Small Additive Distance Phylogeny Problem:
Given $n \times n$ distance matrix $D = [d_{i,j}]$ and unweighted tree $T$ with $n$ leaves, determine edge weights such that $d_T(i, j) = d_{i,j}$
A Small and a Large Problem

Small Additive Distance Phylogeny Problem:
Given $n \times n$ distance matrix $D = [d_{i,j}]$ and unweighted tree $T$ with $n$ leaves, determine edge weights such that $d_T(i, j) = d_{i,j}$

Large Additive Distance Phylogeny Problem:
Given $n \times n$ distance matrix $D = [d_{i,j}]$, find tree $T$ with $n$ leaves and edge weights such that $d_T(i, j) = d_{i,j}$
A Small and a Large Problem

**Small Additive Distance Phylogeny Problem:**
Given $n \times n$ distance matrix $D = [d_{i,j}]$ and unweighted tree $T$ with $n$ leaves, determine edge weights such that $d_T(i, j) = d_{i,j}$

**Large Additive Distance Phylogeny Problem:**
Given $n \times n$ distance matrix $D = [d_{i,j}]$, find tree $T$ with $n$ leaves and edge weights such that $d_T(i, j) = d_{i,j}$

Both problems can be solved in polynomial time
Additive Distance Problem with \( n = 3 \) Sequences
Additive Distance Problem with $n > 3$ Sequences

Unrooted binary tree with $n$ leaves has $2n - 3$ edges and $\binom{n}{2}$ pairwise distances:

- $2n - 3$ variables
- $\binom{n}{2}$ equations

Solution not always possible for $n > 3$

<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>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
Small Additive Distance Phylogeny Problem:

Given $n \times n$ distance matrix $D = [d_{i,j}]$ and unweighted tree $T$ with $n$ leaves, determine edge weights such that $d_T(i, j) = d_{i,j}$.
## Small Additive Distance Problem

<table>
<thead>
<tr>
<th></th>
<th>v</th>
<th>w</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>0</td>
<td>10</td>
<td>17</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>w</td>
<td>0</td>
<td>15</td>
<td>14</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>0</td>
<td>9</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram:

- Vertices: v, w, x, y, z
- Edges: v -> w, w -> x, x -> y, y -> z
### Small Additive Distance Problem

#### Distance Matrix $D$

<table>
<thead>
<tr>
<th></th>
<th>$v$</th>
<th>$w$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v$</td>
<td>0</td>
<td>10</td>
<td>17</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>$w$</td>
<td>0</td>
<td>14</td>
<td>15</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>$x$</td>
<td>0</td>
<td>9</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y$</td>
<td>0</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$z$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

#### Distance Matrix $D_1$

<table>
<thead>
<tr>
<th></th>
<th>$a$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>0</td>
<td>11</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>$x$</td>
<td>0</td>
<td>9</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>$y$</td>
<td>0</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$z$</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Find neighbors $v$, $w$ (common parent)

Differences:

$$d_{ax} = \frac{1}{2} (d_{vx} + d_{wx} - d_{vw})$$

$$d_{ay} = \frac{1}{2} (d_{vy} + d_{wy} - d_{vw})$$

$$d_{az} = \frac{1}{2} (d_{vz} + d_{wz} - d_{vw})$$
Small Additive Distance Problem

**D_1**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>11</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>9</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td></td>
<td>0</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>z</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**D_2**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>b</td>
<td></td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>z</td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Neighbors x, y (common parent)
Small Additive Distance Problem

\[
d(a, c) = 3 \\
d(b, c) = d(a, b) - d(a, c) = 3 \\
d(c, z) = d(a, z) - d(a, c) = 7 \\
d(b, x) = d(a, x) - d(a, b) = 5 \\
d(b, y) = d(a, y) - d(a, b) = 4 \\
d(a, w) = d(z, w) - d(a, z) = 4 \\
d(a, v) = d(z, v) - d(a, z) = 6
\]
Correct!!!
Small Additive Distance Problem

1. Find neighboring leaves $i$ and $j$ with parent $k$

2. Remove the rows and columns of $i$ and $j$

3. Add a new row and column corresponding to $k$, where the distance from $k$ to any other leaf $m$ is computed as

$$d_{k,m} = \frac{(d_{i,m}+d_{j,m}-d_{i,j})}{2}$$

4. Repeat steps 1-3 until tree has only two vertices
A Small and a Large Problem

Small Additive Distance Phylogeny Problem:
Given $n \times n$ distance matrix $D = [d_{i,j}]$ and unweighted tree $T$ with $n$ leaves, determine edge weights such that $d_T(i, j) = d_{i,j}$

Large Additive Distance Phylogeny Problem:
Given $n \times n$ distance matrix $D = [d_{i,j}]$, find tree $T$ with $n$ leaves and edge weights such that $d_T(i, j) = d_{i,j}$

Both problems can be solved in polynomial time
Large Additive Distance Phylogeny Problem

Idea: find neighboring leaves by simply selecting pair of closest leaves

<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>13</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>j</td>
<td>0</td>
<td>12</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td>0</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>l</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

WRONG!

Finding a pair of neighboring leaves is a nontrivial problem!

\[ d_{ij} = 13 > d_{jk} = 12. \]
A degenerate triple is a set of three distinct elements $i, j, k \in [n]$ such that $d_{i,j} + d_{j,k} = d_{i,k}$

Element $j$ in a degenerate triple $(i, j, k)$ lies* on the evolutionary path from $i$ to $k$

*or is attached to this path by an edge of length 0
**Degenerate Triples can be Removed**

A **degenerate triple** is a set of three distinct elements \( i, j, k \in [n] \) such that \( d_{i,j} + d_{j,k} = d_{i,k} \)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Element** \( j \) **in a degenerate triple** \( (i, j, k) \) **lies*** on the **evolutionary path** from \( i \) to \( k \)

*or is attached to this path by an edge of length 0
Looking for Degenerate Triples

If distance matrix $D$ does not have a degenerate triple, one can create one by shortening all hanging edges.

Trimming Parameter

Decrease entries in matrix $D$ by $2\delta$
Additive Phylogeny

• If there is no degenerative triple:
  • Reduce all hanging edges by the same amount $\delta$, so that all pairwise distances in the matrix are reduced by $2\delta$.

• This process will eventually collapse one of the leaves (when $\delta$ equals the length of the shortest hanging edge), forming a degenerate triple $(i, j, k)$ and reducing the size of the distance matrix $D$.

• The attachment point for $j$ can be recovered in the reverse transformations by saving $d_{i,j}$ for each collapsed leaf.
Additive Phylogeny

AdditivePhylogeny(D)
if D is a 2 x 2 matrix
    T = tree of a single edge of length D_{1,2}
return T
if D is non-degenerate
    Compute trimming parameter \( \delta \)
    Trim(D, \( \delta \))
Find a triple \( i, j, k \) in D such that \( D_{ij} + D_{jk} = D_{ik} \)
    \( x = D_{ij} \)
Remove \( j^{th} \) row and \( j^{th} \) column from D
T = AdditivePhylogeny(D).
Add a new vertex \( v \) to T at distance \( x \) from \( i \) to \( k \)
Add \( j \) back to T by creating an edge \((v,j)\) of length 0
for every leaf \( l \) in T
    if distance from \( l \) to \( v \) in the tree \( \neq D_{lj} \)
        output “matrix is not additive”
return
Extend all “hanging” edges by length \( \delta \)
return T
Outline

• Introduction
• Hierarchical clustering
• Additive distance phylogeny
• Four point condition
• Neighbor joining

Reading:
• Chapter 10.2 and 10.5-10.8 in Jones and Pevzner
Additive Distance Matrices

Matrix $D$ is ADDITIVE if there exists a tree $T$ with $d_{ij}(T) = D_{ij}$

<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>0</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

NON-ADDITIVE otherwise

This is a constructive definition

\[ \begin{array}{c|cccc}
   & A & B & C & D \\
\hline
   A & 0 & 2 & 2 & 2 \\
   B & 2 & 0 & 3 & 2 \\
   C & 2 & 3 & 0 & 2 \\
   D & 2 & 2 & 2 & 0 \\
\end{array} \]

Question: Can we characterize set of additive matrices?
Four Point Condition (Zaretskii 1965, Buneman 1971)

Four point condition of matrix \( D = [d_{i,j}] \):
Every four leaves (quartet) can be labeled as \((i, j, k, l)\) such that
\[
d_{i,j} + d_{k,l} \leq d_{i,k} + d_{j,l} = d_{i,l} + d_{j,k}
\]

Three sums:
1. \( d_{i,j} + d_{k,l} \)
2. \( d_{i,k} + d_{j,l} \)
3. \( d_{i,l} + d_{j,k} \)

2 and 3 represent the same number: (length of all edges) + 2 * (length middle edge)

1 represents a smaller number: (length of all edges) – (length middle edge)
Four Point Condition

**Four point condition of matrix** $D = [d_{i,j}]$: Every four leaves (quartet) can be labeled as $(i, j, k, l)$ such that

$$d_{i,j} + d_{k,l} \leq d_{i,k} + d_{j,l} = d_{i,l} + d_{j,k}$$

If two leaves are the same, four point condition is triangle inequality (e.g. set $l = j$)

Four point condition generalizes triangle inequality and defines a subset of distances, namely additive distances
Four Point Condition: Theorem

Every four leaves (quartet) can be labeled as \((i, j, k, l)\) such that
\[d_{i,j} + d_{k,l} \leq d_{i,k} + d_{j,l} = d_{i,l} + d_{j,k}\]

**Theorem:** An \(n \times n\) matrix \(D\) is additive if and only if the for point condition holds for every quartet \((i, j, k, l) \in [n]^4\)
Four Point Condition: Theorem

Every four leaves (quartet) can be labeled as \((i, j, k, l)\) such that
\[d_{i,j} + d_{k,l} \leq d_{i,k} + d_{j,l} = d_{i,l} + d_{j,k}\]

**Theorem:** An \(n \times n\) matrix \(D\) is additive if and only if the four point condition holds for every quartet \((i, j, k, l) \in [n]^4\)

**Proof:** (=>) Since \(D\) is additive, there is a tree \(T\) such that \(d_{i,j} = d_T(i, j)\) for all \((i, j) \in n^2\). Let \((i, j, k, l)\) be a quartet. Assume w.l.o.g. that \(i, j\) and \(k, l\) are neighbors. Define \(\lambda_m\) as illustrated.
Four Point Condition: Theorem

Every four leaves (quartet) can be labeled as \((i, j, k, l)\) such that
\[ d_{i,j} + d_{k,l} \leq d_{i,k} + d_{j,l} = d_{i,l} + d_{j,k} \]

**Theorem:** An \(n \times n\) matrix \(D\) is additive if and only if the for point condition holds for every quartet \((i, j, k, l) \in [n]^4\)

**Proof:** \((=>)\) Since \(D\) is additive, there is a tree \(T\) such that \(d_{i,j} = d_T(i, j)\) for all \((i, j) \in n^2\). Let \((i, j, k, l)\) be a quartet. Assume w.l.o.g. that \(i, j\) and \(k, l\) are neighbors. Define \(\lambda_m\) as illustrated.

\[
\begin{align*}
\lambda_1 + \lambda_3 + \lambda_4 & = d_{i,k} + d_{j,l} \\
\lambda_2 + \lambda_3 + \lambda_5 & \geq (\lambda_1 + \lambda_2) + (\lambda_4 + \lambda_5) = d_{i,j} + d_{k,l}
\end{align*}
\]
Four Point Condition: Theorem

Every four leaves (quartet) can be labeled as \((i, j, k, l)\) such that
\[d_{i,j} + d_{k,l} \leq d_{i,k} + d_{j,l} = d_{i,l} + d_{j,k}\]

**Theorem:** An \(n \times n\) matrix \(D\) is additive if and only if the for point condition holds for every quartet \((i, j, k, l) \in [n]^4\).

**Proof:** \((\leq)\) Assume four point condition holds. Need an algorithm to construct \(T\). AdditivePhylogeny\((T)\) is one such algorithm*. Neighbor joining is another algorithm.

*we have not proved correctness nor shown how to correct \(\delta\)*
Theorem: Let $D$ be an $n \times n$ matrix. The following statements are equivalent.

1. Matrix $D$ is additive.
2. There exists a unique tree $T$ (modulo isomorphism) s.t. $d_{i,j} = d_T(i,j)$ for all $(i,j) \in n^2$.
3. Four point condition holds for every quartet $(i,j,k,l) \in [n]^4$.

Four point condition of matrix $D = [d_{i,j}]$:

Every four leaves (quartet) can be labeled as $(i,j,k,l)$ such that

\[ d_{i,j} + d_{k,l} \leq d_{i,k} + d_{j,l} = d_{i,l} + d_{j,k} \]
Outline

• Introduction
• Hierarchical clustering
• Additive distance phylogeny
• Four point condition
• Neighbor joining

Reading:

• Chapter 10.2 and 10.5-10.8 in Jones and Pevzner
Distance Based Phylogeny Problem

**Large Additive Distance Phylogeny Problem:**
Given \( n \times n \) matrix \( D = [d_{i,j}] \), find tree \( T \) with \( n \) leaves and edge weights such that
\[
\max_{(i,j) \in [n]^2} |d_T(i,j) - d_{i,j}| \text{ is minimum.}
\]

Equivalently, find additive matrix \( D' \) closest to input matrix \( D \).
Neighbor Joining Algorithm (Saitou and Nei 1987)

• Constructs binary unrooted trees.
• Recall: leaves $a$ and $b$ are neighbors if they have a common parent
• Recall: closest leaves are not necessarily neighbors
• NJ: Find pair of leaves that are “close” to each other but “far” from other leaves

Two advantages: (1) reproduces correct tree for additive matrix, and (2) otherwise gives good approximation of correct tree
Distance Trees as Hierarchical Clustering

Leaves = Data points.

Data points clustered/grouped into hierarchy according to some distance criterion.
Distance Trees as Hierarchical Clustering

Leaves = Data points.

Data points clustered/grouped into hierarchy according to some distance criterion.
Distance Trees as Hierarchical Clustering

1. Hierarchical Clustering \((D, n)\)
2. Form \(n\) clusters each with one element
3. Construct a graph \(T\) by assigning one vertex to each cluster
4. while there is more than one cluster
5. Find the two closest clusters \(C_1\) and \(C_2\)
6. Merge \(C_1\) and \(C_2\) into new cluster \(C\) with \(|C_1| + |C_2|\) elements
7. Compute distance from \(C\) to all other clusters
8. Add a new vertex \(C\) to \(T\) and connect to vertices \(C_1\) and \(C_2\)
9. Remove rows and columns of \(D\) corresponding to \(C_1\) and \(C_2\)
10. Add a row and column to \(D\) corresponding to the new cluster \(C\)
11. return \(T\)

Selection criterion: distance between clusters affects clustering!
Neighbor Joining: Selection Criterion

Let \( C = \{1, \ldots, n\} \) be current clusters/leaves.

Define: \( u_i = \sum_k D(i, k) \).

Intuitively, \( u_i \) measures separation of \( i \) from other leaves.

**Goal:** Minimize \( D(i, j) \) and maximize \( u_i + u_j \).

**Solution:** Find pair \((i, j)\) that minimizes:
\[
S_D(i, j) = (n - 2) \ D(i, j) - u_i - u_j
\]

**Claim:** Given additive matrix \( D \).
\[
S_D(x, y) = \min S_D(i, j) \text{ if and only if } x \text{ and } y \text{ are neighbors in tree } T \text{ with } d_T = D.
\]
Neighboring Joining: Algorithm

Initialization:
Form \( n \) clusters \( C_1, C_2, \ldots, C_n \), one for each leaf node.
Define tree \( T \) to be the set of leaf nodes, one per sequence.

Iteration: \( (D \text{ is } m \times m) \)
Pick \( i, j \) such that \( S_D(i,j) = (m-2)D(i,j) - u_i - u_j \) is minimal.

Merge \( i \) and \( j \) into new node \([ij]\) in \( T \).
Assign length \( \frac{1}{2} \left( D(i,j) + \frac{1}{m-2} (u_i - u_j) \right) \) to edge \((i, [ij])\)
Assign length \( \frac{1}{2} \left( D(i,j) + \frac{1}{m-2} (u_j - u_i) \right) \) to edge \((j, [ij])\)

Remove rows and columns from \( D \) corresponding to \( i \) and \( j \).
Add row and column to \( D \) for new vertex \([ij]\).
Set \( D([ij], m) = \frac{1}{2} \left[ D(i, m) + D(j, m) - D(i,j) \right] \)

Termination:
When only one cluster

Question: Does this create rooted or unrooted trees?
Advantages of Neighbor Joining

**Theorem:** Let $D$ be an $n \times n$ matrix. If matrix $D$ is additive then neighbor joining produces the unique phylogenetic tree $T$ (modulo isomorphism) such that $d_{i,j} = d_T(i,j)$ for all $(i,j) \in n^2$.

**Theorem:** Let $D$ be an $n \times n$ matrix. If there exists an additive matrix $D'$ such that $|D - D'|_\infty \leq 0.5$ then neighbor joining applied to $D$ reconstructs the unique tree $T$ (modulo isomorphism) such that $d'_{i,j} = d_T(i,j)$ for all $(i,j) \in n^2$.

Atteson 1991
Neighbor Joining tree relating copy number profiles from single cells in a tumor.

[Navin et al, Nature 2011]
Summary

- Introduction
- Hierarchical clustering
- Additive distance phylogeny
- Four point condition
- Neighbor joining

Reading:
- Chapter 10.2 and 10.5-10.8 in Jones and Pevzner