PHYLOGENETIC TREES AND EUCLIDEAN EMBEDDINGS

By

Mark Layer

RECOMMENDED: [illegible]
Dr. Elizabeth Allman

[illegible]
Dr. Jill Faudree

[illegible]
Dr. John Rhodes
Advisory Committee Chair

[illegible]
Dr. John Rhodes
Chair, Department of Mathematics and Statistics

APPROVED: Anupma Prakash
Dr. Anupma Prakash
Associate Dean, College of Natural Science and Mathematics

[illegible]
Dr. John Eichelberger
Dean of the Graduate School

4/21/14
Date

PHYLOGENETIC TREES AND EUCLIDEAN EMBEDDINGS

A

THESIS

Presented to the Faculty

of the University of Alaska Fairbanks

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

By

Mark Layer, B.A.

Fairbanks, Alaska

May 2014

Abstract

In this thesis we develop an intuitive process of encoding any phylogenetic tree and its associated tree-distance matrix as a collection of points in Euclidean space. Using this encoding, we find that information about the structure of the tree can easily be recovered by applying the inner product operation to vector combinations of the Euclidean points. By applying Classical Scaling to the tree-distance matrix, we are able to find the Euclidean points even when the phylogenetic tree is not known. We use the insight gained by encoding the tree as a collection of Euclidean points to modify the Neighbor Joining Algorithm, a method to recover an unknown phylogenetic tree from its tree-distance matrix, to be more resistant to tree-distance proportional errors.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Phylogenetics is the study of evolutionary relationships among groups of organisms. A useful way to express evolutionary relationships is through the use of phylogenetic trees, a visualization of the timeline over which different groups of organisms diverged from their common ancestor to become distinct species.

One of the challenges in phylogenetics is to infer the structure of these trees using only information from extant organisms, which are located at the leaves of the tree. Such a measure might indicate how different, or in other words, distant, any given species is from any other. Many tools have been developed to build phylogenetic trees using only the matrix of pairwise-distances of the collection of organisms. One such method is the Neighbor Joining Algorithm, which quickly and recursively constructs a tree by estimating the tree structure based on the distance matrix. When given perfect tree-distance data the Neighbor Joining algorithm will perfectly construct the tree. However in phylogenetics the difference data is found by applying a probabilistic model to sequence data, so there is error in the data due to poor model fit and the finite length of sequences. As the relative distance between different organisms grows, the accuracy of the distance measurement typically diminishes. These sources of error reduce the ability of the Neighbor Joining algorithm to produce the correct tree and impair the ability of phylogeneticists and biologists to draw accurate conclusions about the evolutionary history of life on this Earth.

All sections of this thesis are authored by Mark Layer with coauthor Dr. John A. Rhodes. In this thesis we propose a modification to the Neighbor Joining Algorithm which, it is hoped, will more accurately handle relative distance based errors in the data. Much of the work here is to develop the necessary mathematical framework behind this method, a framework which is of interest on its own. The modified algorithm is presented in detail and implemented in R, but it has not yet been extensively tested on phylogenetic data.

1.2 Euclidean Nature of Phylogenetic Distance Matrices

The main insight used to improve the Neighbor Joining algorithm is motivated by a work "Euclidean Nature of Phylogenetic Distance Matrices," which was published in 2011 by Damien de Vinne, Gabriela Aguileta and Sébastien Ollier [dVAO11]. They proposed that if T is a n-taxon metric tree with associated $n \times n$ matrix D giving pairwise tree distances between the leaves, then it is possible to find a collection of n points in Euclidean space whose pairwise distances are given by the entrywise square roots of D. However they did not offer a strong mathematical proof of this new insight, or explore what benefit this insight can produce for the field of phylogenetics. Their argument was rather indirect, and based on relating D to the variance-covariance matrix of a Brownian motion process along T. For such a simple conclusion, it seems highly desirable to find a less elaborate proof.

1.3 Chapter Overview

In chapter 2 we produce an elementary, constructive proof of the observation in [dVAO11], by directly producing, from a given metric tree T, a collection of Euclidean points whose pairwise distances are the square root of the tree distances. That many properties of the original tree are recoverable from the points then follows. Thus the construction adds additional insights to those of [dVAO11].

In chapter 3 we briefly explain how through multidimensional scaling we can find the Euclidean points corresponding directly to the tree distance matrix D without first knowing the tree.

In chapter 4 we explain how we can use the insights of chapter 2 and the ideas presented in chapter 3 to reveal tree structure information presented in the Euclidean points without ever needing to explicitly find the points. Then we use the revealed tree structure to develop a new variant of the Neighbor Joining Algorithm which will possibly perform better on inexact data sets than other versions of the Neighbor Joining Algorithm.

Chapter 2

Metric Trees and the Squareroot Map

2.1 Metric Trees

To set terminology we recall some basic notions of graph theory as used in phylogenetics [SS03].

A *(undirected simple) graph* G is an ordered pair (V, E) consisting of a nonempty set of *vertices* or *nodes* $V = V(G)$ and a set of *edges* $E = E(G)$. Each edge is a two element set $\{x, y\}$, with $x, y \in V$. We assume V, and hence E, are finite. We note that this definition precludes edges of the form $\{v, v\}$ because the set $\{v, v\}$ is the same as the set $\{v\}$ which is not a two element set and thus not an element of $E(G)$.

Two vertices x and y are *adjacent* in the graph if there is an edge $e = \{x, y\} \in E$; in this circumstance the edge e and the vertices x and y are said to be *incident*. Two edges are also considered *adjacent* if they are both incident to the same vertex. A graph is called *simple* if there is at most one edge incident to each pair of vertices. The *degree* of a vertex is the number of edges incident to that vertex.

A *path* in G is a sequence of distinct vertices $(v_0, v_1, \ldots, v_n)$ such that for all $i \in (0, 1, \ldots, n-1)$, vertex v_i is adjacent to vertex v_{i+1}. If G is a graph such that for each pair of distinct vertices $x, y \in V(G)$ there exists exactly one path whose first vertex is x and whose last vertex is y then G is called a *topological tree*, denoted by $\mathfrak{T}$. In such a tree the unique path starting at x and ending at y is denoted by $P_{x,y}$. In a topological tree, each path $(v_0, v_1, \ldots, v_n)$ determines a sequence of distinct edges $(e_1, e_2, \ldots, e_n)$ such that $e_i = \{v_{i-1}, v_i\}$. The set of edges on a path $P_{x,y}$ is denoted by $E(P_{x,y})$. We note that if $E(P_{x,y}) = E(P_{u,v})$ then either $x = u$ and $y = v$ or $x = v$ and $y = u$.

In a tree the vertices of degree one are known as *leaf nodes* or simply the *leaves* of the tree. Vertices of degree larger than one are *internal* vertices. A pair of leaf nodes which are both adjacent to the same internal vertex is known as a *cherry*. If every internal vertex is of degree exactly 3 then the tree is a *binary tree*.

Let $L = L(\mathfrak{T})$ denote an arbitrary subset of the nodes of $\mathfrak{T}$ including all leaves, called the *labeled nodes*. In the field of phylogenetics, L represents the nodes on which we have data, and is almost always the set of leaf nodes of the tree. In phylogenetics L is called the set of *taxa* on the tree $\mathfrak{T}$. If $|L| = n$ then we call the tree $\mathfrak{T}$ an *n-taxon tree*.

A *metric tree* T is a topological tree $\mathfrak{T}$ together with a function

$$w : E(\mathfrak{T}) \to \mathbb{R}^{\geq 0}.$$

For each edge $e \in E(T)$ the value $w(e)$ is called the *weight* of the edge. In phylogenetics it is common to refer to the weight of an edge as its *length*. The *length of a path* $P_{x,y}$, also known as the *tree distance* between vertices x and y, is given by the function

$$d : V(T) \times V(T) \to \mathbb{R}^{\geq 0}$$

defined by

$$d(x, y) = \sum_{e \in E(P_{x,y})} w(e).$$

We note that by this definition d is not necessarily a metric because edge lengths may be 0 and thus length zero paths may exist. This can cause a situation where x and y are different vertices but $d(x, y) = 0$. If we add the further restriction that $w(e) > 0$ for all $e \in E$ then d does define a metric. We allow length zero edges in our definition because it is sometimes useful in the field of phylogenetics.

For any set L, an *L-split* is a partition of L into two nonempty sets A and B, $L = A \sqcup B$, the disjoint union of A and B, and is denoted $A|B$. The position of A and B in this notation is arbitrary and we make no distinction between $A|B$ and $B|A$. If $A|B$ is a split with $x \in A$ and $y \in B$ then the split $A|B$ is said to *separate* the vertices x and y [SS03].

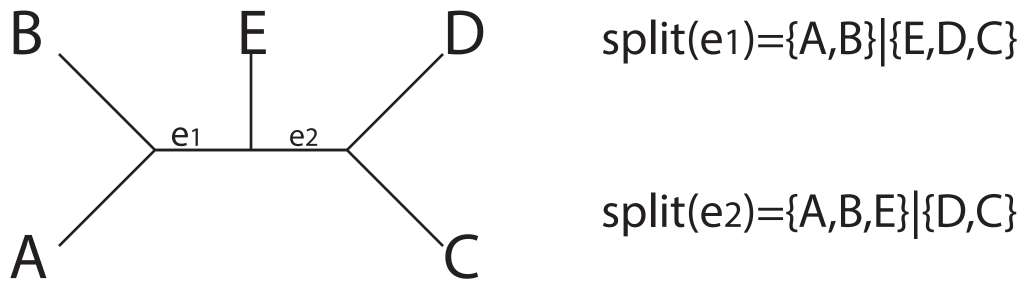

Figure 2.1: A 5-taxon tree and the splits induced by edges e1 and e2.

If $\mathfrak{T}$ is a topological tree on a set of taxa $L = L(\mathfrak{T})$ then each edge e of $\mathfrak{T}$ induces a bipartition of L, defined so that if $e \in E(P_{x,y})$ then x and y lie in different sets of the bipartition and if $e \notin E(P_{x,y})$ then x and y lie in the same set of the bipartition. We denote the split of L induced by $e \in E(\mathfrak{T})$ as $\mathit{split}(e)$. See figure 2.1 for an example on the subset $L = \{A, B, C, D, E\}$.

The following definition is not standard so we will specifically label it for future reference.

Definition 1. For $L' \subset L(\mathfrak{T})$ we define the *induced subtree* $\mathfrak{T}'$ on L' as follows. First we consider the minimal spanning tree in $\mathfrak{T}$ on L'. That is we take the collection of edges $E' \subseteq E$ and vertices $V' \subseteq V$ with $v \in V'$ if, and only if, v lies on a path $P_{x,y}$ for $x, y \in L'$, and $e \in E'$ if, and only if, $e = \{v_1, v_2\} \in E$ with $v_1, v_2 \in V'$. Then, if $v \in V' \backslash L'$ is of degree 2 with edges $e_1 = \{v_1, v\}$ and $e_2 = \{v, v_2\}$ incident to v, we delete v from V', delete e_1 and e_2 from E', and add a new edge $e_3 = \{v_1, v_2\}$ to E'. Thus we join e_1 and e_2 into a single edge. Applying this step repeatedly if necessary, V' will have no unlabeled vertices of degree 2. .

If $T = (\mathfrak{T}, w)$ is a metric tree, then on the induced tree $\mathfrak{T}'$ on L' we define a metric

w' by

$$w'(e) = \sum w(e_i), \qquad \text{if } e \text{ was found by joining edges } \{e_i\},$$
$$w'(e) = w(e) \qquad \text{otherwise.}$$

Thus $T' = (\mathfrak{T}', w')$ will have the same tree distance between elements of L' as does T.

For a given pair of taxa $v_1, v_2 \in L$ on a metric n-taxon tree T, the associated tree distance of the pair is given by $d(v_1, v_2)$. Arbitrarily ordering the taxa then defines an $n \times n$ matrix $D = (d_{ij})$, where $d_{ij} = d(v_i, v_j)$. We call D the *distance matrix* for T.

As $d(v_i, v_i) = 0$ and $d(v_i, v_j) = d(v_j, v_i)$ for any $v_i, v_j \in V$, then D is a symmetric matrix of nonnegative entries, which has zeros down the diagonal.

It is usually the case in phylogenetic applications that the pairwise distances between taxa can be determined experimentally, up to some error, whereas the metric tree which relates the collection remains unknown. Then D can be estimated by data, but is usually not known exactly. For this chapter however, we treat the phylogenetic tree T as fixed, and D as known and exact.

2.2 The Squareroot Map

Typically tree distances do not have a Euclidean basis. However there is a natural way to relate them to points in Euclidean space. Although this fact was first pointed out by De Vienne, et al. [dVAO11], here we develop a novel elementary approach to this fact which will lead to additional insights.

We define the squareroot map of an m-vertex metric tree T as follows.

Definition 2. Given an m-vertex metric tree T with its associated tree distance d, choose some ordering of $E(T)$ so that each edge is uniquely denoted e_i for $i \in \{1, 2, \dots, m-1\}$. Then for any arbitrary chosen $v \in V(T)$ the *squareroot map*

$$\Psi_v : V(T) \to \mathbb{R}^{m-1}$$

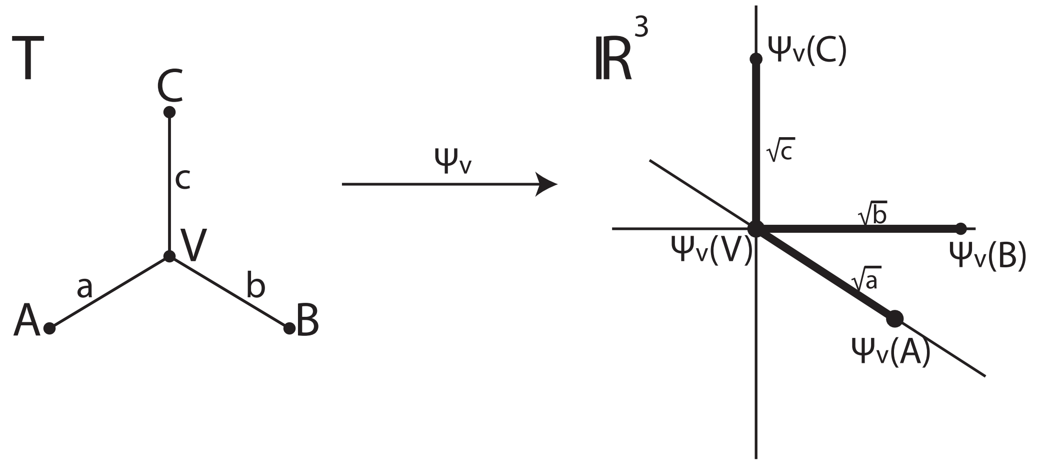

Figure 2.2: A 4-vertex tree (left) and the resulting squareroot map in $\mathbb{R}^3$ (right).

is such that $\Psi_v(v) = \mathbf{0}$, and for all $y \neq v, y \in V(T)$,

$$\Psi_v(y) = (\alpha_1, \alpha_2, \ldots, \alpha_{m-1}),$$

where

$$\alpha_i = \begin{cases} \sqrt{w(e_i)}, & \text{if } e_i \in P_{v,y}, \\ 0, & \text{otherwise.} \end{cases}$$

We refer to the ith coordinate in $\mathbb{R}^{m-1}$ as the *e_i-coordinate*. If $w(e) > 0$ for all e then Ψ_v is injective, and we refer to Ψ_v as the *squareroot embedding*.

Example 3. Figure 2.2 explicitly demonstrates how the squareroot map takes a 4-vertex tree into $\mathbb{R}^3$. The metric tree T contains vertices $\{A, B, C, V\}$ such that the edge incident to A has weight a, the edge incident to B has weight b and the edge incident to C has weight c. Note that the Euclidean distance between $\Psi_v(A)$ and $\Psi_v(B)$ is

$$\sqrt{(\sqrt{b} - 0)^2 + (0 - \sqrt{a})^2} = \sqrt{\sqrt{b}^2 + \sqrt{a}^2} = \sqrt{b + a}$$

where $b + a$ is the tree distance between the vertices A and B. In other words the Euclidean distance between the images of a pair of vertices is the square root of the tree distances between the original vertices. This fact generalizes for all vertices in the tree as Theorem 5 below will show, and is the primary motivation behind the squareroot map. To prove this we will use the following.

Lemma 4. *For any nonempty tree T and any choice of $v \in V(T)$, for all vertices v_1 and $v_2 \in V(T)$,*

$$\Psi_v(v_1) - \Psi_v(v_2) = (\gamma_1, \gamma_2, \dots, \gamma_{m-1})$$

$$\textit{where } \gamma_i = \begin{cases} \sqrt{w(e_i)} & \textit{when } e_i \in E(P_{v,v_1}) \textit{ and } e_i \notin E(P_{v,v_2}), \\ -\sqrt{w(e_i)} & \textit{when } e_i \in E(P_{v,v_2}) \textit{ and } e_i \notin E(P_{v,v_1}), \\ 0 & \textit{otherwise.} \end{cases}$$

In particular, $\gamma_i \neq 0$ only when

$$e_i \in E(P_{v_1,v_2}) \textit{ and } w(e_i) > 0.$$

Proof. To see this, we note that by the design of the squareroot map that the e-coordinate of $\Psi_v(v_1)$ is $\sqrt{w(e)}$ only when $e \in E(P_{v,v_1})$ and zero otherwise. Similarly the e-coordinate of $\Psi_v(v_2)$ is $\sqrt{w(e)}$ only when $e \in E(P_{v,v_2})$ and zero otherwise. Then it immediately follows that the e_i-coordinate of $\Psi_v(v_1) - \Psi_v(v_2)$ is given by the above formula for γ_i.

On the tree T, the edges on the path P_{v_1,v_2} are those which lie in exactly one of P_{v,v_1} and P_{v,v_2}. Therefore the nonzero coordinates of $\Psi_v(v_1) - \Psi_v(v_2)$ correspond to the edges on the path P_{v_1,v_2} of positive length. $\square$

Let

$$\rho(a, b) : \mathbb{R}^{m-1} \times \mathbb{R}^{m-1} \to \mathbb{R}^{\geq 0}$$

denote the standard Euclidean metric. We now relate the tree distance function and the Euclidean metric through the squareroot map.

Theorem 5. *For any nonempty tree T and for $v_1, v_2 \in V(T)$,*

$$\rho(\Psi_v(v_1), \Psi_v(v_2)) = \sqrt{d(v_1, v_2)}.$$

Proof. Let T be an m-vertex tree. We can apply Lemma 4 to note that

$$\rho(\Psi_v(v_1), \Psi_v(v_2)) = ||\Psi_v(v_1) - \Psi_v(v_2)|| = \sqrt{\sum_{1 \le i \le m-1} \gamma_i^2}.$$

As $\gamma_i^2 = w(e_i)$ when $e_i \in E(P_{v_1,v_2})$ and 0 otherwise, then

$$\sum_{1 \le i \le m-1} \gamma_i^2 = d(v_1, v_2).$$

Therefore

$$\rho(\Psi_v(v_1), \Psi_v(v_2)) = \sqrt{d(v_1, v_2)}.$$

□

We note that the choice of v to define Ψ_v is unimportant by the following lemma, which directly follows from Lemma 4.

Proposition 6. *Given a labeling of $E(T)$, and a pair of vertices $v_1 \neq v_2$, the maps Ψ_{v_1} and Ψ_{v_2} differ by coordinate reflections and translation. More specifically there is a reflection R in some coordinates of $\mathbb{R}^{m-1}$ and some vector $\boldsymbol{a} \in \mathbb{R}^{m-1}$ such that*

$$\Psi_{v_1}(v) = R\Psi_{v_2}(v) + \boldsymbol{a} \textit{ for all } v \textit{ in } V(T).$$

Proof. From Lemma 4 we note

$$\Psi_{v_1}(v) - \Psi_{v_1}(v_1) = R(\Psi_{v_2}(v) - \Psi_{v_2}(v_1))$$

where R is the reflection that changes sign in e-coordinates with $e \in E(P_{v_1,v_2})$.

So

$$\Psi_{v_1}(x) = R\Psi_{v_2}(x) + \mathbf{a}_{v_1,v_2}$$

where $\mathbf{a}_{v_1,v_2} = -R\Psi_{v_2}(v_1)$. □

Thus the choice of v for Ψ_v is irrelevant up to an isometry of Euclidean space. From now on we will often suppress the v in the notation so that $\Psi_v = \Psi$.

We make note that the squareroot map can be applied to *all* of the vertices of the tree, not just the leaves. Also the tree need not be a binary tree. This leads to the following lemma that will allow us to only consider trees with positive edge lengths.

Lemma 7. *Suppose for a given metric tree $T = (\mathfrak{T}, w)$ there is some edge $e = \{v_1, v_2\}$ with $w(e) = 0$. Define the metric tree $T' = (\mathfrak{T}', w')$ where $\mathfrak{T}'$ is the tree $\mathfrak{T}$ in which we delete e, identify v_1 with v_2, and let $w' = w|_{E'}$, where E' is the edge set of T'. In other words T' is the tree T in which edge e has been contracted. Then with P denoting the projection into all coordinates except the e-coordinate*

$$P(\Psi(V)) = \Psi(V').$$

Proof. If $w(e) = 0$ then the e-coordinate of $\Psi(v)$ will be zero for all $v \in V(T)$. Therefore we can safely drop the e-coordinate from $\Psi(v)$. □

Applying the above lemma repeatedly, we may assume T has no edge of length zero and thus refer to the squareroot map as the *squareroot embedding.*

We note that by repeatedly applying the above lemma, we may lose any initial assumption we may have had about working with binary trees, or about the labeled vertices L being only leaf nodes of T. We may also reduce the size of L if two of its elements are identified. However, as long as one keeps these conditions in mind, this causes no problems.

As a first step to exploring the relationship of a tree and its image under the squareroot embedding we obtain the following Proposition.

Proposition 8. *If all edge lengths of T are positive and $v_1, v_2, v_3 \in V$ are distinct then $\Psi(v_1), \Psi(v_2), \Psi(v_3)$ are not collinear.*

Proof. Suppose there exist three vertices $v_1, v_2, v_3 \in V(T)$ such that $p_1 = \Psi(v_1), p_2 = \Psi(v_2), p_3 = \Psi(v_3)$ are collinear in $\mathbb{R}^{m-1}$. Without loss of generality assume p_3 lies between p_1 and p_2.

Then p_1, p_2, p_3 must satisfy the condition

$$\rho(p_1, p_2) = \rho(p_1, p_3) + \rho(p_2, p_3)$$

where ρ is the standard Euclidean metric.

We note that $\rho(p_i, p_j) = \sqrt{d(v_i, v_j)}$ which means

$$\sqrt{d(v_1, v_2)} = \sqrt{d(v_1, v_3)} + \sqrt{d(v_2, v_3)}.$$

Thus, squaring both sides of the equality, it must be the case that

$$d(v_1, v_2) = d(v_1, v_3) + 2\sqrt{d(v_1, v_3)}\sqrt{d(v_2, v_3)} + d(v_2, v_3). \tag{2.1}$$

But as v_1, v_2, v_3 lie on T, a metric tree with no length zero edges, it is also true that

$$d(v_1, v_2) \leq d(v_1, v_3) + d(v_2, v_3) \tag{2.2}$$

with equality if, and only if, v_3 lies on the path from v_1 to v_2.

Now the equality (2.1) and inequality (2.2) imply that $2\sqrt{d(v_1, v_3)}\sqrt{d(v_2, v_3)}$ must be zero. Therefore at least one of $d(v_1, v_3)$ or $d(v_2, v_3)$ is zero. Now as T has no length zero edges, then $d(v_i, v_j) = 0$ if, and only if, $v_i = v_j$. Thus if any three points in $\Psi(V)$ are collinear, then at least two of them are the same point. □

Example 9. We note that Proposition 8 implies that even trees which one can naively embed in Euclidean space so that every point is collinear, such as a path with no edges of length zero, "crumple" when embedded under the squareroot embedding. Paths that could nicely embed in 1-dimensional space become contorted as each vertex in the path is forced into a different spacial dimension. This is demonstrated in Figure 2.3.

Although the squareroot embedding places every vertex of the tree into Euclidean space, we can choose only to consider the affine subspace spanned by the squareroot embedding applied to the taxa of the tree, i.e., $\Psi(L)$, where L is the labeled vertices of the tree. This is more relevant to phylogenetic applications, as one generally has data only relating a subset of the vertices in a metric tree.

Theorem 10. *Let T be a tree with positive edge lengths. Let $|L| = n$. Then $\Psi(L)$ spans an affine space of dimension exactly $n - 1$.*

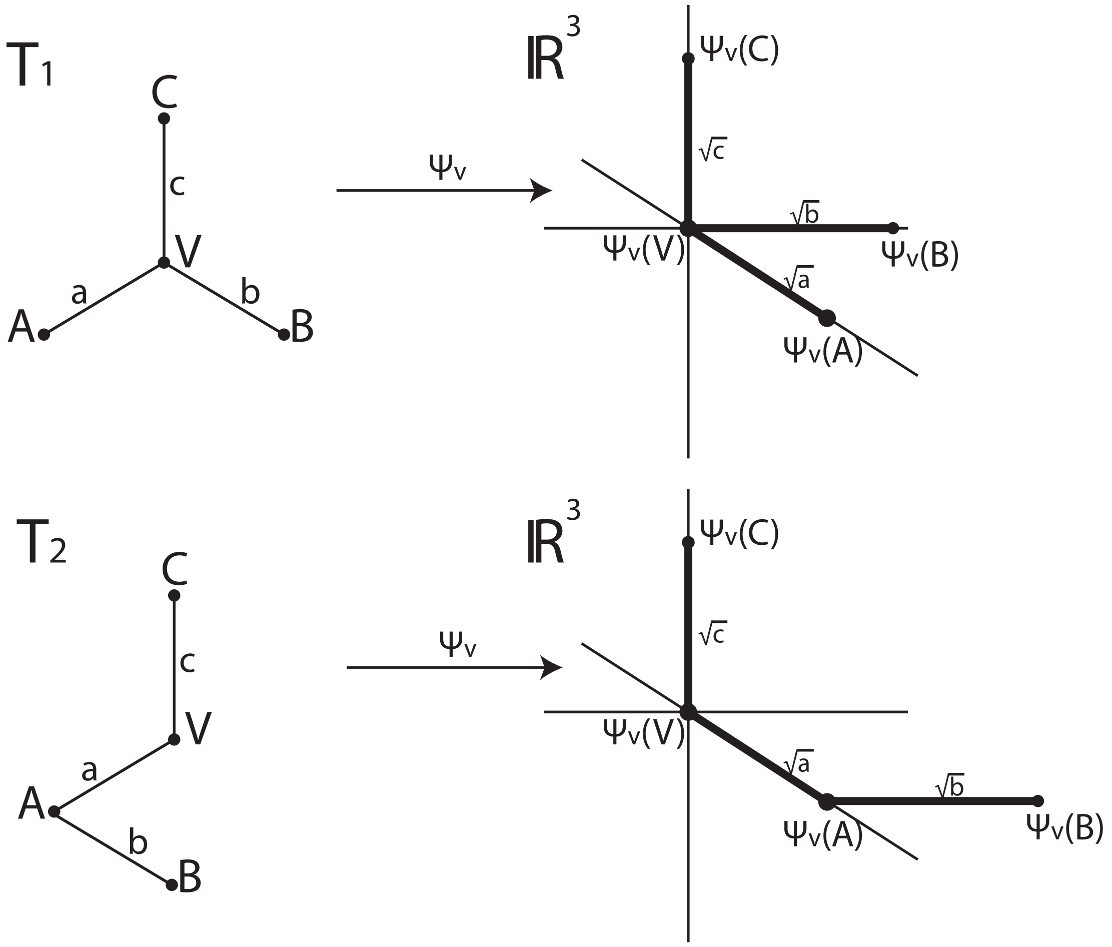

Figure 2.3: A 4-vertex tree T_1 (top left) and its image under the squareroot embedding (top right), and a 4 vertex path T_2 (bottom left) and its image under the squareroot embedding (bottom right).

The Euclidean distances between the points corresponding to the taxa of T under the embedding are given by the entry-wise square roots of the distance matrix D, which we denote $\sqrt{D}$.

Proof. The set $\Psi(L)$ is a collection of n distinct points in Euclidean space. To show that the dimension of the affine span of that set can not be less than $n-1$ we will proceed by induction on n.

Suppose $n = 1$. Then $\Psi(L)$ is a single point which defines a zero dimension affine space.

Now suppose that for any subset L' of V with $1 \leq |L'| < n$ that the dimension of the affine space spanned by $\Psi(L')$ is $|L'| - 1$. Consider L a set of labeled nodes such that $|L| = n$. If L does not contain a leaf then we can prune leaves and terminal branches of the tree T, until L does contain a leaf, without changing how L is embedded under the squareroot embedding. Thus we may assume that L contains at least one leaf w and we may also assume the squareroot embedding on L is given by Ψ_v for $v \neq w$.

We note that as $|L \backslash \{w\}| = n - 1$ then by the induction hypothesis $\Psi(L\backslash\{w\}))$ spans a $n-2$ dimension space. As w is a leaf then, if e is the edge incident to w, the e-coordinate of all points in $\Psi_v(L\backslash\{w\}))$ is 0. However the e-coordinate of $\Psi_v(w)$ is not zero, and so $\Psi_v(w)$ is not in the affine span of $\Psi_v(L\backslash\{w\}))$. Therefore the dimension of the span $\Psi_v(L)$ is 1 greater than the dimension of $\Psi_v(L\backslash\{w\}))$. In other words the dimension of the span of $\Psi_v(L)$ is $n - 2 + 1 = n - 1$. □

Remark. Our notation $\sqrt{D}$ for the entry-wise square roots of D should not be confused with other standard uses of this notation to denote a matrix whose square is D.

The squareroot embedding leads to some interesting parallels between the geometric properties of $\Psi(V)$ and the properties of T.

For the following lemma, let Ψ_v^T refer to the squareroot embedding Ψ_v associated to the tree T.

Lemma 11. *If $V' \subset V$, T' is the induced subtree, as defined in Definition 1, on V', and $v \in V'$ then there is a unique Euclidean isometry $i : \mathbb{R}^{|V'|-1} \to \mathbb{R}^{|V|-1}$ such that*

$$i(\Psi_v^{T'}(x)) = \Psi_v^T(x)$$

for all $x \in V'$.

Proof. We may assume T and then T' have positive edge lengths. By Theorems 5 and 10, the affine span of both $\Psi_v^{T'}(V')$ and $\Psi_v^{T}(V')$ produce $|V'| - 1$ dimensional affine spaces.

As passing to the induced subtree on V' preserves path distances between any $v_1, v_2 \in V'$ then, if D is the distance matrix for V' in T and D' is the distance matrix for V' in T', $D = D'$. Thus Euclidean distances between points in $\Psi_v^{T'}(V')$ agree with those for $\Psi_v^{T}(V)$, as both are given by $\sqrt{D} = \sqrt{D'}$.

Then there is an isometry i determined by mapping $\Psi_v^{T'}(w)$ to $\Psi_v^{T}(w)$ for all $w \in V'$, and extending the mapping to the affine spaces spanned by $\Psi_v^{T'}(w)$ and $\Psi_v^{T}(w)$. $\square$

Proposition 12. *Suppose T has all positive edge lengths. Given distinct $x_1, x_2, y_1, y_2 \in V(T)$, let $\boldsymbol{a} = \Psi(x_1) - \Psi(x_2)$ and $\boldsymbol{b} = \Psi(y_1) - \Psi(y_2)$. Then*

$$\boldsymbol{a} \cdot \boldsymbol{b} = \pm \sum_{e_i \in M} w(e_i) \text{ where } M = E(P_{x_2,x_1}) \cap E(P_{y_2,y_1}).$$

That is, $\boldsymbol{a} \cdot \boldsymbol{b}$ is, up to sign, the length of the subpath common to P_{x_2,x_1} and P_{y_2,y_1}. The sign is positive if the subpath is oriented in the same direction in both paths and is negative if oppositely oriented.

Proof. By Lemma 11 it is enough to prove this for the induced subtree on $\{x_1, x_2, y_1, y_2\}$.

Recall that in general the inner product of $\mathbf{a}$ and $\mathbf{b}$, is defined by

$$\mathbf{a} \cdot \mathbf{b} = \sum a_i b_i.$$

Note from Lemma 4 that the e-coordinate of $\mathbf{a}$ is nonzero only when e lies on the path P_{x_1,x_2}, in which case the e-coordinate is $\pm\sqrt{w(e_i)}$. This is similarly true for e-coordinate of $\mathbf{b}$. Thus when e_i lies on both P_{x_1,x_2} and P_{y_1,y_2}

$$a_i b_i = \pm(\sqrt{w(e_i)}^2) = \pm w(e_i).$$

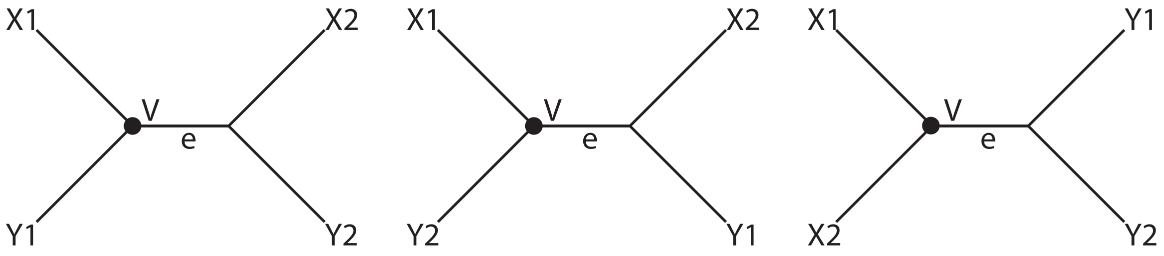

Figure 2.4: The three cases of T' with $X1, X2, Y1, Y2$ leaves.

When e_i does not lie in both P_{x_1,x_2} and P_{y_1,y_2} then

$$a_i b_i = 0.$$

Thus the only non-zero summands in $\mathbf{a} \cdot \mathbf{b}$ are $\pm w(e_i)$ for

$$e_i \in M = E(P_{x_2,x_1}) \cap E(P_{y_2,y_1}).$$

Now using Lemma 11, if T' is the induced subtree on $V' = \{x_1, x_2, y_1, y_2\}$ then $\Psi_v^{T'}(V')$ is isometric to $\Psi_v^T(V')$. Thus if $\mathbf{a}' = \Psi_v^{T'}(x_1) - \Psi_v^{T'}(x_2)$ and $\mathbf{b}' = \Psi_v^{T'}(y_1) - \Psi_v^{T'}(y_2)$, then

$$\mathbf{a}' \cdot \mathbf{b}' = \mathbf{a} \cdot \mathbf{b}.$$

The arrangement of the four vertices x_1, x_2, y_1, y_2 in the induced subtree T' are important for determining the value of $\mathbf{a}' \cdot \mathbf{b}'$, we will provide a detailed proof only of the cases shown in Figure 2.4 where x_1, x_2, y_1, y_2 are four leaves on T'. The cases where one or more of the labeled vertices are not leaves of T', or where the tree is not binary, are handled similarly.

In the leftmost case where x_1, y_1 form a cherry, edge e is on both paths P_{v,x_2} and P_{v,y_2} and does not lie on either path P_{v,x_1} nor P_{v,y_1}. Then the e-coordinate of $\mathbf{a}' \cdot \mathbf{b}'$ is

$$-\sqrt{w'(e)} * -\sqrt{w'(e)} = +w'(e).$$

In the middle case, x_1 and y_2 form a cherry. Then edge e lies on the paths P_{v,x_2} and P_{v,y_1} and does not lie on either path P_{v,x_1} nor P_{v,y_2}. Then the e-coordinate of

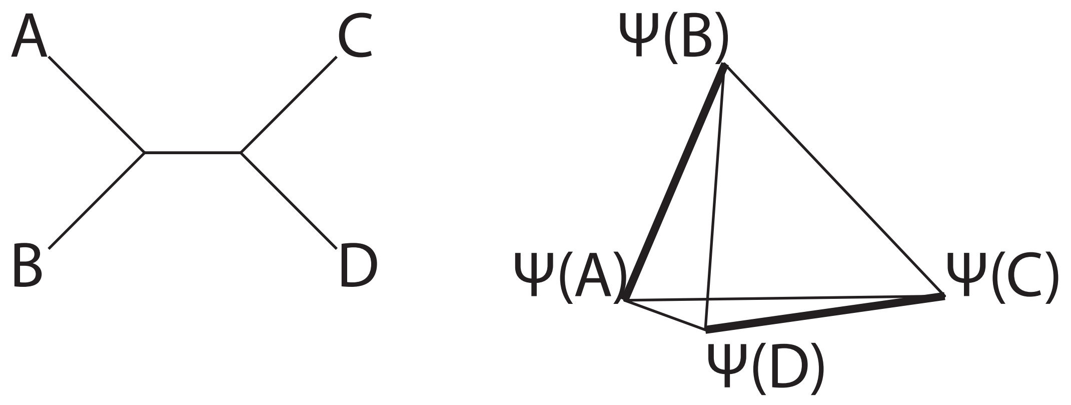

Figure 2.5: A 4-taxon tree (left) and the image of the 4 taxa image under the square-root embedding (right).

$\mathbf{a}' \cdot \mathbf{b}'$ is

$$-\sqrt{w'(e)} * \sqrt{w'(e)} = -w'(e).$$

In the rightmost case,, P_{x_1,x_2} and P_{y_1,y_2} have no edges in common and so $\mathbf{a}' \cdot \mathbf{b}' = 0$.

Because of how $w'(e)$ is defined in the formation of the induced subtree T', $w'(e)$ is the length of the sub-path common to both P_{x_1,x_2} and P_{y_1,y_2}. Thus $\mathbf{a} \cdot \mathbf{b}$ is up to sign, the length of the subpath common to P_{x_2,x_1} and P_{y_2,y_1}. The sign is positive if the subpath is oriented in the same direction in both paths and is negative if oppositely oriented. □

Example 13. One of the motivating images for many of the proofs in this chapter, Figure 2.5 shows a 4-taxon tree and the tetrahedron formed by the image of the taxa under the squareroot embedding. The edge $\Psi(A) - \Psi(B)$ is orthogonal to the edge $\Psi(C) - \Psi(D)$, according to Proposition 12. Additionally all triangles of the tetrahedron are acute, as Corollary 14 below states.

This example motivates the following Corollary.

Corollary 14. *For a tree with nonzero edge lengths, any three points in $\Psi(L)$ form either an acute or a right triangle.*

Proof. With $\mathbf{a} = \Psi(v_1) - \Psi(v_3), \mathbf{b} = \Psi(v_2) - \Psi(v_3)$ the common subpath to P_{v_1,v_3} and P_{v_2,v_3} is either empty or oriented in the same direction, so $\mathbf{a} \cdot \mathbf{b} \geq 0$. ☐

The following Corollaries are very important for using the squareroot map to reconstruct the tree. We will use these extensively in Chapter 4.

Corollary 15. *Suppose T is a binary tree with no length zero edges, then distinct v_1 and $v_2 \in V$ form a cherry in T if, and only if*

$$(\Psi(v_1) - \Psi(v_2)) \cdot (\Psi(w_i) - \Psi(w_j)) = 0$$

for all leaves $w_i, w_j \in V \backslash \{v_1, v_2\}$.

Proof. Suppose v_1, v_2 form a cherry. Then the path P_{v_1,v_2} consists of two edges e_1 and e_2, where e_1 is incident to v_1 and e_2 is incident to v_2. But then $E(P_{w_i,w_j})$ will not contain e_1 nor e_2. By Proposition 12

$$(\Psi(v_1) - \Psi(v_2)) \cdot (\Psi(w_i) - \Psi(w_j)) = 0$$

for all $w_i, w_j \in V \backslash \{v_1, v_2\}$

Now suppose $(\Psi(v_1) - \Psi(v_2)) \cdot (\Psi(w_1) - \Psi(w_2)) = 0$ for all leaves $w_1, w_2 \in V \backslash \{v_1, v_2\}$. Then by Proposition 12

$$E(P_{v_1,v_2}) \cap E(P_{w_1,w_2}) = \emptyset.$$

Thus the only paths between two leaves on which any edge $e \in E(P_{v_1,v_2})$ lies has either v_1 or v_2 as an endpoint. As T is a binary tree then this implies v_1 and v_2 form a cherry in T. ☐

Corollary 16. *Suppose T has no length zero edges and suppose T is a binary metric tree. Let $v_1, v_2, w_1, w_2 \in L$. Then*

$$(\Psi(v_1) - \Psi(v_2)) \cdot (\Psi(w_1) - \Psi(w_2)) = 0$$

if, and only if there exists an edge $e \in E$ with split(e) separating v_1, v_2 from w_1, w_2.

Proof. To prove the forward implication, first note that by Proposition 12 the above inner product is $\pm$ the sum of the edge weights of the edges which lie on the path common to both paths P_{v_1,v_2} and P_{w_1,w_2}. If the sum of the edge weights is zero and if every edge weight on the tree is positive then P_{v_1,v_2} and P_{w_1,w_2} must have no edges in common. As v_1, v_2, w_1, w_2 are in the vertex set of T and as T is a binary tree then there must exist an edge e in $E(T)$ such that cutting the edge e from T will form two trees, one of which contains v_1 and v_2 and the other of which contains w_1 and w_2. By definition this is a split separating v_1, v_2 from w_1, w_2.

The converse holds since if an edge e separating the taxa exists, the paths P_{v_1,v_2} and P_{w_1,w_2} cannot have any edge in common. $\square$

So far each of these insights are based on the squareroot map which requires knowledge of the original tree T and so is not directly useful for learning the tree's structure. However as we shall see in the following chapter, there are ways to find a collection of Euclidean points, using only the tree-distance matrix D, which are isometric to the points found with the squareroot embedding. This will allow us to use the insights of chapter 2 even when the original tree is not known.

Chapter 3

Finding Euclidean Points From the Tree Distance Matrix

In chapter 2 we proved that $\sqrt{D}$ is a Euclidean distance matrix, so now we can use standard techniques of multidimensional scaling to find Euclidean coordinates from the tree-distance matrix D directly. What follows will be a brief tangent into classical scaling as described in [CC94].

3.1 Classical Scaling

Classical scaling originated in the 1930's from the work of Eckart and Young (1936), and Young and Householder (1938). The technique is designed so that starting with a matrix of precise Euclidean distances, one may find the coordinates for points such that distances between them match those given in the matrix.

The process for finding the coordinates given an Euclidean distance matrix E is motivated as follows.

Let the coordinates of n points in a p dimensional Euclidean space be given by the rows of the $n \times p$ matrix X where the ith point, $\mathbf{x}_i$, is the ith row of X. Then the squared Euclidean distance between the ith and jth points is given by $E = (e_{ij})$ where

$$e_{ij}^2 = \rho(\mathbf{x}_i, \mathbf{x}_j)^2 = (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T. \tag{3.1}$$

Let the inner product matrix be $H = (h_{ij})$ where

$$h_{ij} = \mathbf{x}_i \mathbf{x}_j^T = \mathbf{x}_i \cdot \mathbf{x}_j.$$

Our plan is to use E to find H and then from H find X.

First, as the desired Euclidean points specified by X are only determined up to isometries of Euclidean space, we choose to make it so that the center of mass for the points specified by X is at the origin. That is, we require

$$\sum \mathbf{x}_i = \mathbf{0}.$$

This still allows for arbitrary rotations and reflection about the origin, but that ambiguity will not impact the following calculations.

To find H we note from equation (3.1)

$$e_{ij}^2 = \mathbf{x}_i \cdot \mathbf{x}_i + \mathbf{x}_j \cdot \mathbf{x}_j - 2\mathbf{x}_i \cdot \mathbf{x}_j,$$

and so

$$h_{ij} = -\frac{1}{2}(e_{ij}^2 - \mathbf{x}_i \cdot \mathbf{x}_i - \mathbf{x}_j \cdot \mathbf{x}_j). \tag{3.2}$$

But

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n} e_{ij}^2 &= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i \cdot \mathbf{x}_i) + \frac{n}{n}(\mathbf{x}_j \cdot \mathbf{x}_j) - \frac{2}{n}\sum_{i=1}^{n}\mathbf{x}_i \cdot \mathbf{x}_j \\
&= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i \cdot \mathbf{x}_i) + \frac{n}{n}(\mathbf{x}_j \cdot \mathbf{x}_j) - \frac{2}{n}(\sum_{i=1}^{n}\mathbf{x}_i) \cdot \mathbf{x}_j \\
&= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i \cdot \mathbf{x}_i) + (\mathbf{x}_j \cdot \mathbf{x}_j) - \frac{2}{n}\mathbf{0} \cdot \mathbf{x}_j \\
&= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i \cdot \mathbf{x}_i) + (\mathbf{x}_j \cdot \mathbf{x}_j),
\end{aligned}$$

so

$$\mathbf{x}_j \cdot \mathbf{x}_j = \frac{1}{n}\sum_{i=1}^{n} e_{ij}^2 - \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \cdot \mathbf{x}_i. \tag{3.3}$$

Similarly

$$\mathbf{x}_i \cdot \mathbf{x}_i = \frac{1}{n}\sum_{j=1}^{n} e_{ij}^2 - \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j \cdot \mathbf{x}_j. \tag{3.4}$$

Summing equation (3.3) over $j = 1, \ldots, n$ shows

$$\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} e_{ij}^2 = \frac{2}{n}\sum_{i=1}^{n} \mathbf{x}_i \cdot \mathbf{x}_i. \tag{3.5}$$

Substituting equation (3.3), (3.4), and (3.5) into (3.2) gives

$$h_{ij} = -\frac{1}{2}\left(e_{ij}^2 - \frac{1}{n}\sum_{i=1}^{n} e_{ij}^2 - \frac{1}{n}\sum_{j=1}^{n} e_{ij}^2 + \frac{1}{n^2}\sum_{j=1}^{n}\sum_{i=1}^{n} e_{ij}^2\right). \tag{3.6}$$

Defining $A = (-\frac{1}{2}e_{ij}^2)$, then equation (3.6) can be expressed as

$$H = FAF$$

where F is the *centering matrix*

$$F = I - n^{-1}\mathbf{1}\mathbf{1}^T,$$

with $\mathbf{1} = (1, 1, \ldots, 1)$, a column vector of n ones.

With a single formula for H found, we turn to recovering X.

Since the inner product matrix H, can be expressed as

$$H = XX^T,$$

and is thus symmetric, applying the singular value decomposition to H gives

$$H = U\Sigma U^T,$$

where Σ is a $n \times n$ diagonal matrix with all real entries and U is an $n \times n$ orthogonal matrix. In fact H must be positive semi-definite as for any column vector c

$$\begin{aligned} c^T H c &= c^T X X^T c \\ &= (c^T X)(c^T X)^T \\ &= ||cX||^2 \geq 0. \end{aligned}$$

Thus the entries of Σ are nonnegative and

$$H = \left(U\sqrt{\Sigma}\right)\left(U\sqrt{\Sigma}\right)^T.$$

(Note the "$\sqrt{\ }$" notation here is consistent both with the standard usage for matrices, and with our usage in Chapter 2.)

Thus to find X from H, we may take

$$X = U\sqrt{\Sigma},$$

[CC94]. The ambiguity in X up to rotation and reflection is now apparent, as $X = U\sqrt{\Sigma}Q$ for any orthogonal matrix Q would also give us a solution to $H = XX^T$.

Remark. Note that since $\sum \mathbf{x}_i = \mathbf{0}$ we have $\mathbf{1}^T X = \mathbf{0}$ so $\mathbf{1}^T H\mathbf{1} = 0$. Thus

$$0 = \mathbf{1}^T H\mathbf{1} = \mathbf{1}^T U\Sigma U^T\mathbf{1} = (\mathbf{1}^T U)\Sigma(\mathbf{1}^T U)^T,$$

where $\mathbf{1}^T U \neq \mathbf{0}$, because U has orthogonal rows. Since Σ has non-negative diagonal entries, this implies at least one diagonal entry of Σ is 0. Thus we can always assume Σ is $(n-1)\times(n-1)$, and U is $n\times(n-1)$, so X is $n\times(n-1)$. Thus the points $\mathbf{x}_i$ specified by the rows of X live in $\mathbb{R}^{n-1}$.

3.2 Classical Scaling Applied to the Tree

When the above approach is applied to the distances from a tree metric, these are given in a matrix D where $\sqrt{D}$ is Euclidean. Then the matrix A in the above explanation is simply

$$A = -\frac{1}{2}D.$$

To apply classical scaling to D we first find H where

$$H = -\frac{1}{2}FDF.$$

As F is the centering matrix, then the process of multiplying D on both sides by F is called "double centering" D. Then to explicitly find coordinates of n Euclidean points we apply the singular value decomposition to H to get

$$H = U\Sigma U^T.$$

Thus we can find a collection of Euclidean points as the rows of Y by

$$Y = U\sqrt{\Sigma}.$$

One of the motivations for this thesis is to find a way to infer the tree structure of T based on a known associated tree distance matrix D. For this reason we would like to be able to apply our observations from Chapter 2 to the collection of points given by Y gained through classical scaling.

In Chapter 2 we proved there exists a collection $\Psi(L)$ of points in a $(n-1)$ dimensional affine space whose pairwise distances are given by $\sqrt{D}$. If X is the matrix whose rows are the points of $\Psi(L)$ translated to be centered at the origin, then we have already shown that $XX^T = H$. In order to apply all the observations from Chapter 2 to the $n \times n-1$ matrix Y recovered from the SVD procedure, we will prove the following theorem.

Theorem 17. *$X = YQ$ for some $(n-1) \times (n-1)$ orthogonal matrix Q.*

Proof. To begin, we know the rows of X span an $n-1$ dimensional affine space, and so X has rank $(n-1)$ which implies XX^T also is of rank $(n-1)$.

Then as

$$XX^T = H = U\Sigma U^T$$

where U is an $n \times n$ orthogonal matrix, this implies Σ has exactly $n-1$ nonzero diagonal entries and one zero diagonal entry. Thus

$$\Sigma = \begin{pmatrix} \sigma_1 & & & & \\ & \sigma_2 & & \text{\Large 0} & \\ & & \ddots & & \\ & \text{\Large 0} & & \sigma_{n-1} & \\ & & & & 0 \end{pmatrix}_{n\times n}$$

and we can define $\widetilde{\Sigma}$ as the invertible $(n-1)\times(n-1)$ matrix

$$\widetilde{\Sigma} = \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \text{\Large 0} \\ & & \ddots & \\ & \text{\Large 0} & & \sigma_{n-1} \end{pmatrix}_{(n-1)\times(n-1)} .$$

In a similar way, the nth column of U can be deleted to define $\widetilde{U}$ by

$$U = \begin{pmatrix} & | \\ \widetilde{\text{U}} & \mathbf{u}_n \\ & | \end{pmatrix}_{n\times n}$$

where $\widetilde{U}$ is an $n\times(n-1)$ matrix and $\mathbf{u}_n$ is a column vector.

Now as

$$H = U\Sigma U^T = \widetilde{U}\widetilde{\Sigma}\widetilde{U}^T$$

we have that $XX^T = U\Sigma U^T$ and $Y = \widetilde{U}\sqrt{\widetilde{\Sigma}}$.

Now because U is an orthogonal matrix then $U^{-1} = U^T$ and so

$$XX^T = U\Sigma U^T$$

implies

$$U^T XX^T U = \Sigma.$$

Then, with $I_{(n-1)}$ denoting the identity matrix of dimension $(n-1)\times(n-1)$,

$$\begin{pmatrix} \sqrt{\widetilde{\Sigma}} & 0 \\ 0 & 1 \end{pmatrix}^{-1} U^T XX^T U^T \begin{pmatrix} \sqrt{\widetilde{\Sigma}} & 0 \\ 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} \sqrt{\widetilde{\Sigma}} & 0 \\ 0 & 1 \end{pmatrix}^{-1} \Sigma \begin{pmatrix} \sqrt{\widetilde{\Sigma}} & 0 \\ 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} I_{(n-1)} & 0 \\ 0 & 0 \end{pmatrix}. \tag{3.7}$$

Let

$$Z = \begin{pmatrix} \sqrt{\widetilde{\Sigma}} & 0 \\ 0 & 1 \end{pmatrix}^{-1} U^T X, \tag{3.8}$$

we note Z is a $n \times (n-1)$ matrix. Then

$$Z^T = X^T U^T \begin{pmatrix} \sqrt{\widetilde{\Sigma}} & 0 \\ 0 & 1 \end{pmatrix}^{-1},$$

and so by equation (3.7)

$$ZZ^T = \begin{pmatrix} I_{(n-1)} & 0 \\ 0 & 0 \end{pmatrix}.$$

That the bottom right entry of the matrix ZZ^T is zero implies the bottom row of the matrix Z is a vector with zero norm, thus the bottom row of Z is the zero vector. That the first $n-1$ rows and first $n-1$ columns of ZZ^T form the identity matrix implies the first $n-1$ rows of the $n \times (n-1)$ dimension matrix Z form an orthogonal matrix Q, i.e. $QQ^T = I_{(n-1)}$.

So equation (3.8) becomes

$$\begin{pmatrix} Q \\ 0 \ldots 0 \end{pmatrix} = \begin{pmatrix} \sqrt{\widetilde{\Sigma}} & 0 \\ 0 & 1 \end{pmatrix}^{-1} U^T X,$$

$$U \begin{pmatrix} \sqrt{\widetilde{\Sigma}} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} Q \\ 0 \ldots 0 \end{pmatrix} = X.$$

As the bottom row of the matrix Z is all zeros then

$$U \begin{pmatrix} \sqrt{\widetilde{\Sigma}} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} Q \\ 0 \ldots 0 \end{pmatrix} = \widetilde{U}\sqrt{\widetilde{\Sigma}}Q = YQ.$$

Therefore

$$YQ = X.$$

$\square$

Thus all observations from chapter 2 can be applied to the points found by Classical Scaling.

As an aside a natural corollary neatly summarizes some of the results from this chapter.

Corollary 18. *If D is an exact tree distance matrix, then H is symmetric positive semidefinite with exactly one eigenvalue of 0.*

Chapter 4

The ENJ Algorithm

In chapter 2 we proved that if D is a matrix of exact tree distances then $\sqrt{D}$ is a Euclidean distance matrix, and the collection of Euclidean points whose distance matrix is $\sqrt{D}$ exhibits several properties of the original tree. In chapter 3 we provided a method for finding a collection of Euclidean points whose distance matrix is $\sqrt{D}$. In this chapter we use this earlier work to produce a new variant of the neighbor joining algorithm which aims to be more reliable in the face of proportional errors in distance data. A surprising feature of this method is that we can actually avoid the SVD calculation of chapter 3, so that the entire procedure should be computationally fast.

4.1 The Neighbor Joining Algorithm

Throughout this paper we have referred to a metric tree T and its associated tree distance matrix D. The *Neighbor Joining Algorithm*, referred to as the *NJ* algorithm, is designed to recover T from a given D, even if D has some error in its entries. The NJ algorithm is a recursive algorithm whose inductive step is to find a cherry in the tree and replace it with the vertex adjacent to both vertices of the cherry [SS03]. Briefly we outline its operation in Algorithm 1.

The *Neighbor Joining Criterion* is as follows. It can be shown that picking i and j so that the quantity defined by

$$\sum_r d_{ri} + \sum_r d_{rj} - (n-2)d_{ij}, \tag{4.2}$$

Algorithm 1 The Neighbor Joining Algorithm.

0. The NJ algorithm acts on a collection of leaves L and their pairwise distance matrix D. Set $n = |L|$, then D is a $n \times n$ matrix.

1. If $|L| > 3$ go to step 2. Otherwise $L = \{1, 2, 3\}$ and a new vertex c is defined such that

$$\begin{aligned} d(1,c) &= (d(1,2) + d(1,3) - d(2,3))/2 \\ d(2,c) &= (d(1,2) - d(1,3) + d(2,3))/2 \\ d(3,c) &= (-d(1,2) + d(1,3) + d(2,3))/2. \end{aligned} \tag{4.1}$$

   Then the tree is completely resolved. Stop.

2. Using the *NJ criterion*, choose a pair of taxa v_i and v_j to be joined as a cherry with both v_i and v_j adjacent to a new vertex v.

3. Produce estimates of $d(v, v_i), d(v, v_j)$ and for all $v_r \in L$ with $r \neq i, j$, estimates of $d(v, v_r)$, using equations (4.3).

4. The distances $d(v, v_i), d(v, v_j)$ are lengths of resolved edges in the final tree T, and can be stored for later presentation. Append an $n+1$st row and column to the matrix D so that $d_{r,n+1} = d_{n+1,r} = d(v, v_r)$, $d_{n+1,n+1} = d(v, v) = 0$. Then remove from D the rows and columns corresponding to v_i and v_j, so now D is a $(n-1) \times (n-1)$ dimension matrix. Then v is added to L and v_i and v_j are removed from L, so now $|L| = n - 1$.

5. Loop to step 1.

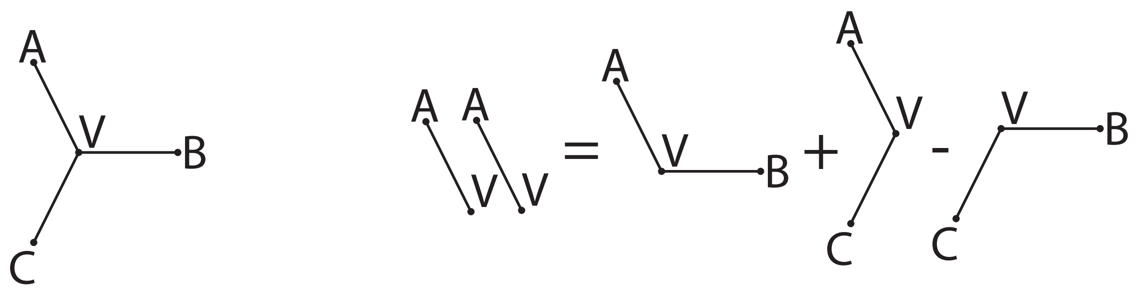

Figure 4.1: A 3 taxa tree (left) and a graphical interpretation of equations (4.1) (right).

is maximized will identify two vertices v_i,v_j in a cherry. Note that these sums are over all r including i and j. Although it is not obvious that this will correctly identify cherries given perfect tree data, it can be proved [SK88]. Methods have been developed to confront the problem of distance based errors in choosing cherries to make the algorithm more robust [BSH00], however we will not be modifying the Neighbor Joining Criterion in our forthcoming algorithm.

The NJ-algorithm also repeatedly uses the resolution of the 3-taxon tree as given by the equations (4.1). The graphical interpretation of these equations is given in Figure 4.1. In addition to using these equations in step 1 of the algorithm, they are used in step 3. There the NJ algorithm calculates the new distances by the following formulas

$$\begin{aligned} d(v,v_i) &= \frac{1}{2}d(v_i,v_j) + \frac{1}{2(n-2)}\left[\sum_{r\neq i,j} d(v_i,v_r) - \sum_{r\neq i,j} d(v_j,v_r)\right], \\ d(v,v_j) &= \frac{1}{2}d(v_i,v_j) + \frac{1}{2(n-2)}\left[\sum_{r\neq i,j} d(v_j,v_r) - \sum_{r\neq i,j} d(v_i,v_r)\right], \\ d(v,v_r) &= \frac{1}{2}[d(v_i,v_r) + d(v_j,v_r) - d(v_i,v_j)]. \end{aligned} \tag{4.3}$$

The first two of these are simply the 3-taxon formulas averaged over all $r \neq i,j$.

For those unfamiliar with how the NJ algorithm functions, please refer to the following example.

Example 19. We will apply the NJ algorithm to the given distance matrix

$$D = \left[\begin{array}{cccccc} & a & b & c & d & e \\ a & 0 & 2 & 3 & 4 & 4 \\ b & 2 & 0 & 3 & 4 & 4 \\ c & 3 & 3 & 0 & 3 & 3 \\ d & 4 & 4 & 3 & 0 & 2 \\ e & 4 & 4 & 3 & 2 & 0 \end{array}\right].$$

Begin Algorithm Step 1: $|L| = 5 > 3$ so we proceed to step 2.

Step 2: The typical way to do the this step of the algorithm is to calculate a matrix $Q = (q_{ij})$ such that the entries of Q are given by equation (4.2), i.e.

$$q_{ij} = \sum_r d_{ri} + \sum_r d_{rj} - (n-2)d_{ij}.$$

Then the Q matrix is

$$Q = \left[\begin{array}{cccccc} & a & b & c & d & e \\ a & 0 & 20 & 16 & 14 & 14 \\ b & 20 & 0 & 16 & 14 & 14 \\ c & 16 & 16 & 0 & 16 & 16 \\ d & 14 & 14 & 16 & 0 & 20 \\ e & 14 & 14 & 16 & 20 & 0 \end{array}\right].$$

As a, b and c, d are pairs of taxa tied for highest Q score of 20 we can pick either one of them. We will select a, b as the pair to be joined in a cherry.

Step 3: We label the new vertex to replace the cherry formed by a and b as vertex ab. Then we produce estimates of $d(a, ab)$ and $d(b, ab)$ using the first two equations of (4.3).

$$\begin{aligned}
d(ab,a) &= \frac{1}{2}d(a,b) + \frac{1}{2(n-2)}\left[\sum_{r\neq a,b} d(a,v_r) - \sum_{r\neq i,j} d(b,v_r)\right], \\
&= \frac{1}{2}2 + \frac{1}{2(3)}\left[3+4+4-(3+4+4)\right], \\
&= 1 + \frac{1}{6}\left[0\right], \\
&= 1.
\end{aligned}$$

$$\begin{aligned}
d(ab,b) &= \frac{1}{2}d(a,b) + \frac{1}{2(n-2)}\left[\sum_{r\neq a,b} d(b,v_r) - \sum_{r\neq i,j} d(a,v_r)\right], \\
&= \frac{1}{2}2 + \frac{1}{2(3)}\left[3+4+4-(3+4+4)\right], \\
&= 1 + \frac{1}{6}\left[0\right], \\
&= 1.
\end{aligned}$$

Then using the third equation of (4.3) we calculate

$$\begin{aligned}
d(ab,c) &= \frac{1}{2}[d(a,c)+d(b,c)-d(a,b)] = \frac{1}{2}(3+3-2) = 2, \\
d(ab,d) &= \frac{1}{2}[d(a,d)+d(b,d)-d(a,b)] = \frac{1}{2}(4+4-2) = 3, \\
d(ab,e) &= \frac{1}{2}[d(a,e)+d(b,e)-d(a,b)] = \frac{1}{2}(4+4-2) = 3.
\end{aligned}$$

Step 4: The calculation of step 3 are compiled into a new distance matrix

$$D_1 = \left[\begin{array}{ccccc}
 & c & d & e & ab \\
c & 0 & 3 & 3 & 2 \\
d & 3 & 0 & 2 & 3 \\
e & 3 & 2 & 0 & 3 \\
ab & 2 & 3 & 3 & 0
\end{array}\right].$$

Loop to step 1.

Step 1: $|L| = 4 > 3$ so go to step 2:

Step 2: New Q matrix is

$$Q_1 = \left[\begin{array}{ccccc} & c & d & e & ab \\ c & 0 & 10 & 10 & 12 \\ d & 10 & 0 & 12 & 10 \\ e & 10 & 12 & 0 & 10 \\ ab & 12 & 10 & 10 & 0 \end{array}\right].$$

We choose ab, c as the pair to be joined in a cherry.

Step 3: New vertex will be named abc. Then

$$\begin{aligned} d(abc, ab) &= \frac{1}{2}d(ab,c) + \frac{1}{2(n-2)}\left[\sum_{r\neq a,b} d(ab, v_r) - \sum_{r\neq a,b} d(c, v_r)\right], \\ &= \frac{1}{2}2 + \frac{1}{2(2)}\left[3+3-(3+3)\right], \\ &= 1. \end{aligned}$$

Similarly

$$d(abc, c) = 1.$$

Then

$$\begin{aligned} d(abc, d) &= \frac{1}{2}[d(ab,d) + d(c,d) - d(ab,c)] = \frac{1}{2}(3+3-2) = 2, \\ d(abc, e) &= \frac{1}{2}[d(ab,e) + d(c,e) - d(ab,c)] = \frac{1}{2}(3+3-2) = 2. \end{aligned}$$

Step 4: New matrix:

$$D_2 = \left[\begin{array}{cccc} & d & e & abc \\ d & 0 & 2 & 2 \\ e & 2 & 0 & 2 \\ abc & 2 & 2 & 0 \end{array}\right].$$

Loop to step 1.

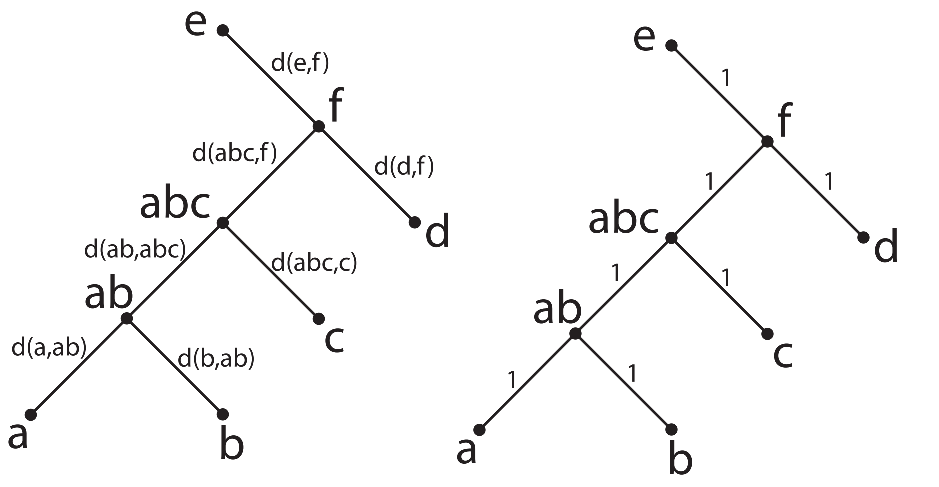

Figure 4.2: The resolved tree from Example 19.

Step 1: $|L| = 3$, so for the new vertex which we will label f:

$$\begin{aligned} d(abc, f) &= (d(abc, d) + d(abc, e) - d(d, e))/2 = (2 + 2 - 2)/2 && = 1 \\ d(d, f) &= (d(abc, d) - d(abc, e) + d(d, e))/2 = (2 - 2 + 2)/2 && = 1 \\ d(e, f) &= (-d(abc, d) + d(abc, e) + d(d, e))/2 = (-2 + 2 + 2)/2 && = 1. \end{aligned}$$

Then the tree is resolved and given in Figure 4.2.

Essentially in each iteration the NJ algorithm finds the placement of the new vertex v in the tree by considering each induced 3-leaf tree using taxa v_i, v_j, v_r and averaging the edge-lengths from them. This method assumes that all these known estimates are equally accurate and weights them equally in determining the position of v in the tree. It is theorized that in an experimental setting the error of measured tree distances is proportional to the size of the distance. In other words one can assume that smaller tree distances are more accurate than larger tree distances. Ideally we want an algorithm which will give more weight to nearby vertices and less to distant ones when placing the new vertex v. However there are also unknown correlations in this distances due to the underlying tree structure, so ways of weighting to improve general performance are not clear.

In 1997 Gascuel et.al. developed the BIONJ algorithm as an attempt to address the issue of distance based error. The BIONJ algorithm works by adopting weights on v_i and v_j where the weights are chosen to minimize the sampling variance of the new distance matrix of estimates. This serves to minimize the topological variance of the resulting tree and produce a more accurate metric tree than basic NJ on specific types of trees with distance based error. The BIONJ algorithm very specifically does not change the weights of the vertices that are not to be joined in the cherry and so suffers undue influence from exceptionally erroneous distant taxa [Gas97].

Using the squareroot embedding, we propose a method for implicitly assigning weights to the $n-2$ taxa not in the cherry for the purpose of estimating the distances from the new vertex v to the remaining taxa of L, as well as for estimating the lengths of edges in a cherry.

4.2 Weights From the Squareroot Embedding

The squareroot embedding for an n-taxon tree defines n points in $(n-1)$-dimensional Euclidean space. By Corollary 15 if v_i and v_j form a cherry in the tree then the vector $\Psi(v_i) - \Psi(v_j)$ lies perpendicular to all other vectors $\Psi(v_r) - \Psi(v_s)$ for any $r, s \neq i, j$. In other words, if V is the 1-dimensional affine space spanned by $\Psi(v_i), \Psi(v_j)$ and W is the $(n-3)$-dimensional affine space spanned by $\Psi(v_r)$ for all $r \neq i, j$, then W and V are orthogonal. For the following, $W, V, \mathbf{p}, \mathbf{q}, \Psi(v_r), \Psi(v_j), \Psi(v_i)$ are arranged as in Figure 4.3.

If $\mathbf{p}$ is in V and $\mathbf{q}$ is in W such that the Euclidean distance $\rho(\mathbf{p}, \mathbf{q})$ is minimized, then the vector $\mathbf{p} - \mathbf{q}$ is orthogonal to both V and W. As $\mathbf{q}$ is in the affine space W then $\mathbf{q}$ is an affine combination of $\Psi(v_r), r \neq i, j$, and $\mathbf{q}$ can be identified with the $(n-2)$-dimensional column vector $\mathbf{t} = (t_r)$, with $\sum_r t_r = 1$, such that

$$\mathbf{q} = \sum_{\Psi(v_r) \in W} t_r \Psi(v_r).$$

We propose the vector $\mathbf{t}$ as the vector of assigned weights to the $n-2$ taxa for the purpose of estimating the distances from the new vertex v to the remaining taxa of L.

Similarly $\mathbf{p}$ is the affine combination of $\Psi(v_i)$ and $\Psi(v_j)$ and thus $\mathbf{p}$ can be iden-

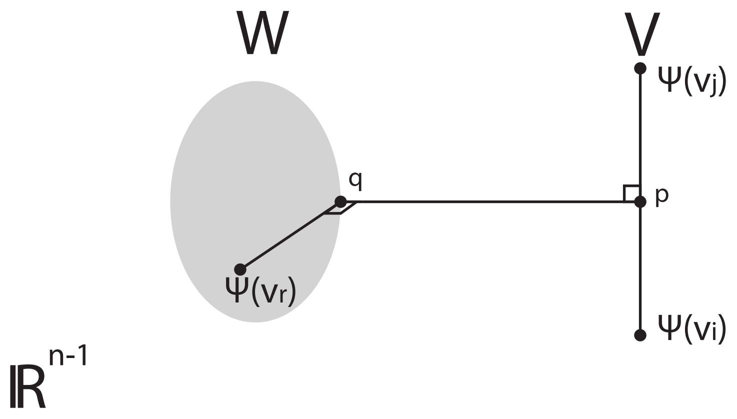

Figure 4.3: The $(n-3)$ dimension affine space spanned by $\Psi(v_r), r \neq i, j$ (left), and the 1-dimension affine space spanned by $\Psi(v_i)$, and $\Psi(v_j)$ (right).

tified with a 2-dimensional column vector $\mathbf{s}$, with $\sum s_i = 1$, and

$$\mathbf{p} = s_1\Psi(v_i) + s_2\Psi(v_j).$$

A great advantage of these weights, as we will see, is they both reflect distances in the tree and they are easily computable through linear algebra.

4.3 Finding p and q

We will now walk through the process to calculate and use $\mathbf{s}$ and $\mathbf{t}$ to estimate the new tree distances for the algorithm.

To calculate $\mathbf{s}$ and $\mathbf{t}$ we need to minimize the function giving the distance between a point in W and a point in V. Let $X_1 = \begin{pmatrix} \Psi(v_i) \\ \Psi(v_j) \end{pmatrix}$ and let $X_2 = \begin{pmatrix} \Psi(v_r) \\ \vdots \end{pmatrix}$ given $r \neq i, j$. Reordering our taxa if necessary, we have

$$X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}. \tag{4.4}$$

Now to find $\mathbf{s}$ and $\mathbf{t}$ is to solve the optimization problem of minimizing the objective function

$$f(\mathbf{s},\mathbf{t}) = ||\mathbf{s}X_1 - \mathbf{t}X_2||^2$$

subject to the constraints

$$s_1 + s_2 = 1 \text{ and } t_1 + t_2 + \cdots + t_{n-2} = 1.$$

Letting $J = \begin{pmatrix} I_2 & 0 \\ 0 & -I_{n-2} \end{pmatrix}$, since

$$\mathbf{s}X_1 - \mathbf{t}X_2 = \begin{pmatrix} \mathbf{s}^T & \mathbf{t}^T \end{pmatrix} JX,$$

the objective function can be expanded as

$$f(\mathbf{s},\mathbf{t}) = \begin{pmatrix} \mathbf{s}^T & \mathbf{t}^T \end{pmatrix} JXX^TJ^T \begin{pmatrix} \mathbf{s} \\ \mathbf{t} \end{pmatrix},$$

or as $XX^T = H$,

$$f(\mathbf{s},\mathbf{t}) = \begin{pmatrix} \mathbf{s}^T & \mathbf{t}^T \end{pmatrix} JHJ^T \begin{pmatrix} \mathbf{s} \\ \mathbf{t} \end{pmatrix}.$$

Let $\mathbf{1}_n = (1, 1, 1, \ldots, 1)$ denote the column vector with n entries of 1, the problem is now to minimize

$$f(\mathbf{s},\mathbf{t}) = \begin{pmatrix} \mathbf{s}^T & \mathbf{t}^T \end{pmatrix} JHJ^T \begin{pmatrix} \mathbf{s} \\ \mathbf{t} \end{pmatrix},$$

subject to

$$c_1(\mathbf{s},\mathbf{t}) = \mathbf{1}_2^T\mathbf{s} = 1,$$

$$c_2(\mathbf{s},\mathbf{t}) = \mathbf{1}_{n-2}^T\mathbf{t} = 1.$$

By the method of Lagrange multipliers, the solution must satisfy the system of

$n+2$ equations in $n+2$ unknowns

$$\begin{aligned}
\nabla_{\mathbf{s},\mathbf{t}} f(\mathbf{s},\mathbf{t}) &= -\lambda_1 \nabla_{\mathbf{s},\mathbf{t}} c_1(\mathbf{s},\mathbf{t}) - \lambda_2 \nabla_{\mathbf{s},\mathbf{t}} c_2(\mathbf{s},\mathbf{t}),\\
c_1(\mathbf{s},\mathbf{t}) &= \begin{pmatrix} \mathbf{1}_2^T & \mathbf{0}_{n-2}^T \end{pmatrix} \begin{pmatrix} \mathbf{s} \\ \mathbf{t} \end{pmatrix} = 1,\\
c_2(\mathbf{s},\mathbf{t}) &= \begin{pmatrix} \mathbf{0}_2^T & \mathbf{1}_{n-2}^T \end{pmatrix} \begin{pmatrix} \mathbf{s} \\ \mathbf{t} \end{pmatrix} = 1.
\end{aligned} \tag{4.5}$$

Now

$$\begin{aligned}
\nabla_{\mathbf{s},\mathbf{t}} f(\mathbf{s},\mathbf{t}) &= 2JHJ^T \begin{pmatrix} \mathbf{s} \\ \mathbf{t} \end{pmatrix},\\
\nabla c_1(\mathbf{s},\mathbf{t}) &= \begin{pmatrix} \mathbf{1}_2 \\ \mathbf{0}_{n-2} \end{pmatrix},\\
\nabla c_2(\mathbf{s},\mathbf{t}) &= \begin{pmatrix} \mathbf{0}_2 \\ \mathbf{1}_{n-2} \end{pmatrix}.
\end{aligned}$$

With $K = \begin{pmatrix} \mathbf{1}_2^T & \mathbf{0}_{n-2}^T \\ \mathbf{0}_2^T & \mathbf{1}_{n-2}^T \end{pmatrix}$ a $2 \times n$ matrix, we can express the system of equations (4.5) in block matrix form as

$$\begin{bmatrix} 2JHJ^T & K^T \\ K & \mathbf{0}_{2\times 2} \end{bmatrix} \begin{pmatrix} \mathbf{s} \\ \mathbf{t} \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} \mathbf{0}_n \\ \mathbf{1}_2 \end{pmatrix}. \tag{4.6}$$

Thus **s** and **t** can be found by solving this matrix equation by standard techniques.

Note the pleasant surprise here that we do not need to actually find X from H to solve for **s** and **t**; knowing H is sufficient. Thus the SVD computation of chapter 3 need not be performed explicitly.

4.4 Using p and q to Calculate Distances.

For ease of notation, let $\mathbf{e}_i$ denote the $n \times 1$ vector with 1 in the ith position and zeros in all other positions, i.e. the ith standard basis vector, as a column. Then, for instance

$$\Psi(v_i) = \mathbf{e}_i^T X.$$

Referring again to Figure 4.3,

$$\begin{aligned}\rho^2(\Psi(v_i), \mathbf{q}) &= \left(\begin{pmatrix}\mathbf{0}_2^T & \mathbf{t}^T\end{pmatrix} - \mathbf{e}_i^T\right) XX^T \left(\begin{pmatrix}\mathbf{0}_2 \\ \mathbf{t}\end{pmatrix} - \mathbf{e}_i\right) \\ &= \left(\begin{pmatrix}\mathbf{0}_2^T & \mathbf{t}^T\end{pmatrix} - \mathbf{e}_i^T\right) H \left(\begin{pmatrix}\mathbf{0}_2 \\ \mathbf{t}\end{pmatrix} - \mathbf{e}_i\right).\end{aligned}$$

Similarly

$$\rho^2(\Psi(v_j), \mathbf{q}) = \left(\begin{pmatrix}\mathbf{0}_2^T & \mathbf{t}^T\end{pmatrix} - \mathbf{e}_j^T\right) H \left(\begin{pmatrix}\mathbf{0}_2 \\ \mathbf{t}\end{pmatrix} - \mathbf{e}_j\right).$$

We could estimate $d(v, v_i)$ and $d(v, v_j)$ using 3-taxon formulas like those in equation (4.1) applied to $d(v_r, v_i), d(v_r, v_j)$ and $d(v_i, v_j)$ for any r. For perfect data we would get the same result regardless of our choice of r. For instance, we have

$$\begin{aligned}d(v, v_i) &= \frac{d(v_r, v_i) - d(v_r, v_j) + d(v_j, v_i)}{2} \\ &= \frac{\rho^2(\Psi(v_r), \Psi(v_i)) - \rho^2(\Psi(v_r), \Psi(v_j)) + d(v_i, v_j)}{2}.\end{aligned} \tag{4.7}$$

But for any $\mathbf{m} \in W$ the Pythagorean Theorem and orthogonality depicted in Figure 4.3 show

$$\rho^2(\mathbf{m}, \Psi(v_i)) - \rho^2(\mathbf{m}, \Psi(v_j)) = \rho^2(\mathbf{m}, \mathbf{q}) + \rho^2(\mathbf{q}, \Psi(v_i)) - \left(\rho^2(\mathbf{m}, \mathbf{q}) + \rho^2(\mathbf{q}, \Psi(v_j))\right).$$

Thus in equation (4.7) we may replace the two occurrence of $\Psi(v_r)$ with any $\mathbf{m} \in W$, We choose to do this using $\mathbf{m} = \mathbf{q}$ giving us equations (4.8) and (4.9).

$$d(v, v_i) = \frac{\rho^2(\Psi(v_i), \mathbf{q}) - \rho^2(\Psi(v_j), \mathbf{q}) + d(v_i, v_j)}{2} \tag{4.8}$$

$$d(v, v_j) = \frac{-\rho^2(\Psi(v_i), \mathbf{q}) + \rho^2(\Psi(v_j), \mathbf{q}) + d(v_i, v_j)}{2}. \tag{4.9}$$

Now to calculate $d(v, v_r)$ for all $r \neq i, j$ we note that for perfect tree distances

$$\begin{aligned}
d(v_r, v) &= d(v_r, v_i) - d(v, v_i) \\
&= \rho^2(\Psi(v_r), \Psi(v_i)) - d(v, v_i) \\
&= \rho^2(\Psi(v_r), \mathbf{q}) + \rho^2(\mathbf{q}, \Psi(v_i)) - d(v, v_i) \text{ by Pythagorean Theorem.} \\
&= \rho^2(\Psi(v_r), \mathbf{q}) + \rho^2(\mathbf{q}, \Psi(v_i)) - \left(\frac{\rho^2(\Psi(v_i), \mathbf{q}) - \rho^2(\Psi(v_j), \mathbf{q}) + d(v_i, v_j)}{2} \right) \text{ using (4.8).} \\
&= \rho^2(\Psi(v_r), \mathbf{q}) + \frac{\rho^2(\Psi(v_i), \mathbf{q}) + \rho^2(\Psi(v_j), \mathbf{q}) - d(v_i, v_j)}{2}. \qquad (4.10)
\end{aligned}$$

We adapt this last formula for our modified NJ algorithm, notice that it is symmetric in v_i and v_j.

4.5 The ENJ Algorithm

The **ENJ**, or *Embedded Neighbor Joining* algorithm is the regular NJ algorithm as outlined in Algorithm 1 except for that, in step 3, equations (4.8), (4.9) and (4.10) are used to produce estimates of $d(v, v_i), d(v, v_j)$ and, for all $v_r \in L$ with $r \neq i, j$, estimates of $d(v, v_r)$. As these involve the use of H, that must be calculated as well for each iteration of the algorithm. We also must use H to find $\mathbf{q}$, by solving the system of equations (4.6). The full procedure is given as Algorithm 2

4.6 R code for the ENJ Algorithm

We have implemented the algorithm in R [R C13] to test its performance compared to various other NJ algorithms. We do this using the APE [PCS04] package for phylogenetic tree analysis.

The algorithm is implemented as two functions in R, first is the function called

Algorithm 2 The Embedded Neighbor Joining Algorithm.

0. The ENJ algorithm acts on a collection of leaves L and their pairwise distance matrix D. Set $n = |L|$, then D is a $n \times n$ matrix.

1. If $|L| > 3$ go to step 2. Otherwise $L = \{1, 2, 3\}$ and a new vertex c is defined such that

$$\begin{aligned} d(1, c) &= (d(1, 2) + d(1, 3) - d(2, 3))/2 \\ d(2, c) &= (d(1, 2) - d(1, 3) + d(2, 3))/2 \\ d(3, c) &= (-d(1, 2) + d(1, 3) + d(2, 3))/2. \end{aligned}$$

   Then the tree is completely resolved. Stop.

2. Using the *NJ criterion*, choose a pair of taxa v_i and v_j to be joined as a cherry with both v_i and v_j adjacent to a new vertex v.

3. Calculate H by

$$H = -\frac{1}{2} FDF$$

   where F is the centering matrix described in chapter 3. Then calculate $\mathbf{s}$ and $\mathbf{t}$ by solving the matrix equation (4.6).

4. Produce estimates of $d(v, v_i), d(v, v_j)$ and for all $v_r \in L$ with $r \neq i, j$, estimates of $d(v, v_r)$, using equations (4.8), (4.9) and (4.10).

5. The distances $d(v, v_i), d(v, v_j)$ are lengths of resolved edges in the final tree T, and can be stored for later presentation. Append an $n + 1$st row and column to the matrix D so that $d_{r,n+1} = d_{n+1,r} = d(v, v_r)$, $d_{n+1,n+1} = d(v, v) = 0$. Then remove from D the rows and columns corresponding to v_i and v_j, so now D is a $(n - 1) \times (n - 1)$ dimension matrix. Then v is added to L and v_i and v_j are removed from L, so now $|L| = n - 1$.

6. Loop to step 1.

`otherst`, which takes as its argument the tree distance matrix D and returns a list $[\mathbf{s}, \mathbf{t}, \mathbf{H}]$

```
otherst <- function(D){
#returns a list of the form [s,t,H]
#double centering the matrix
temp<-dim(D);
n=temp[1];
n2=temp[2];
F=diag(n)-(1/n)*matrix(c(rep(1,n*n)),nrow=n);
H=-.5*F%*%D%*%F;
#double centering the matrixn

J=-diag(n);
J[1,1]=1;
J[2,2]=1;
A=2*J%*%H%*%J
C=matrix(c(rep(0,4)),nrow=2,ncol=2);
B=matrix(c(rep(1,2),rep(0,n),rep(1,n-2)) ,nrow=2,ncol=n,byrow=TRUE);
W=cbind(A,t(B));
P=cbind(B,C);
M=rbind(W,P);
b=matrix(c(rep(0,n),rep(1,2)) ,nrow=n+2,ncol=1);
position1=solve(M,b);
s=matrix(c(position1[1:2,1],rep(0,n-2)),nrow=n,ncol=1);
t=matrix(c(rep(0,2),position1[3:n,1]),nrow=n,ncol=1);
newlist<- list(s,t,H);
return(newlist);
}
```

Using the function `otherst`, the implementation of the ENJ algorithm is below. In its current form it builds a Newick format tree as it progresses and at the end passes it to a native APE function to convert the Newick tree into an APE object of class “phylo”, the APE packages format for phylogenetic trees.

```
enj<-function(D,varargin){
# enj.r
#
# usage: enj(Distarray, Names{:})
#    or: enj(Distarray,’Name1’,’Name2’,...,’Namen’)
#    or: enj(Distarray)
#    or: enj(D,rownames(D))
#
# Performs neighbor joining on distance data.
#
# Distarray should be a square matrix with distances in
# upper triangle (all other entries are ignored);
# Names should be a cell array with names of taxa.
# Otherwise names of taxa can be listed, or omitted.
# If Names is omitted, taxa are called S1, S2, etc.
#
# 8/2/03
#Modified into R with Newick output Mark Layer 3/3/2014


temp=dim(D);
n=temp[1];
if(temp[1]!=temp[2]){
stop("Matrix is not square.")
}else if(n<3){
stop("Must have at least 3 taxa.")
}else{
N=temp[1];
if (missing(varargin)){
for(i in 1:N){
names[i]=paste("S",as.character(i),sep="");
}}else{
names=varargin;}

if(length(names)!= N){
stop("Incorrect number of taxon names.");
}

D=upper.tri(D)*D;
D=D+t(D);

while(n>3){
#step 1 %%Choose which two taxa get to be joined%%
R=as.matrix(colSums(D));
M=(n-2)*D-matrix(c(rep(1,n)),nrow=n)%*%t(R)-R%*%matrix(c(rep(1,n)),ncol=n);
M=M+diag(rep(Inf,n));
inds <- arrayInd(which.min(M), dim(M));
rowind=inds[1];
colind=inds[2]; #taxa rowind  and colind to be joined

#Swap rows and columns of D so that taxa 1 and 2 now to be joined
D[,c(1,rowind)]<- D[,c(rowind,1)];
D[c(1,rowind), ]<- D[c(rowind,1), ];
names[c(1,rowind)] <- names[c(rowind,1)] ;
if (colind==1){
          colind = rowind;}
D[,c(2,colind)]<- D[,c(colind,2)];
D[c(2,colind), ]<- D[c(colind,2), ];
names[c(2,colind)] <- names[c(colind,2)] ;

#idea, build new vertex V="(s1:d(s1,v),s2:d(s2,v))"
#then for the final step, concactinate the strings into
# a single newick formatted tree
```

```
K=otherst(D);
s=K[[1]];
t=K[[2]];
H=K[[3]];

A=matrix(c(1,rep(0,n-1)),ncol=n);
B=matrix(c(0,1,rep(0,n-2)),ncol=n);

rhosqAT=(t(t)-A)%*%H%*%t(t(t)-A);
rhosqBT=(t(t)-B)%*%H%*%t(t(t)-B);
rhosqVA=(D[1,2]+rhosqAT-rhosqBT)/2
rhosqVB=(D[1,2]-rhosqAT+rhosqBT)/2;
rhosqVT=(rhosqAT+rhosqBT-D[1,2])/2;

V=paste("(",names[2],":",as.character(rhosqVB),",",names[1],":",as.character(rhosqVA),")


names=names[-c(1,2)];
names[n-1]<- V;  #at end of algorithm names will have one entry which
#is the newick tree

newD=D[-c(1,2),];
newD=newD[,-c(1,2)];

newColumn=matrix(c(rep(0,n-2)),nrow=n-2);
for(i in 1:n-2){
E=matrix(c(rep(0,i-1+2),1,rep(0,n-2-i)));
newColumn[i]=t(s-E)%*%H%*%(s-E)+rhosqVT;
}

newD=cbind(newD,newColumn);
D=rbind(newD,cbind(t(newColumn),0));
n=n-1;

}
#now only 3 taxa remain, names is a list of three objects

   d1=(D[1,2]+D[1,3]-D[2,3])/2;
   d2=(D[2,1]+D[2,3]-D[1,3])/2;
   d3=(D[3,1]+D[3,2]-D[1,2])/2;
Tree=paste("(",names[1],":",as.character(d1),",",names[2],":",as.character(d2),","

Tree3<-read.tree(text=Tree);
return(Tree3);
}
}
```

Chapter 5

Conclusions

In this thesis we have only covered a few possibilities for analyzing metric trees using the squareroot map. In our research we explored many possible avenues to analyze and visualize the Euclidean points which didn't progress far enough to include in the main body of this thesis.

One idea was to consider lower dimensional projections of the Euclidean points, for instance low dimensional projections, with the hope that pertinent tree structure information would become visually apparent. The idea behind this is that 1,2, and 3 dimensional projections can be drawn on a piece of paper and directly presented to readers to see the conclusions for themselves.

Another idea was to create a list of all possible combinations of pairs of points $(\Psi(v_a), \Psi(v_b))$ in the image of the squareroot map. Then from this list produce a matrix whose ij entry corresponds to the inner product of the ith entry of the list with the jth entry. The resulting matrix contains a lot of structure which can possibly be analyzed to detect tree splits, cherries, and other features of the data.

A third idea is that it may be possible to devise a new way to compare two trees using the Euclidean points of their respective squareroot maps. This may be a useful direction of inquiry as it is currently unclear as to how to resolve topological and metric differences between pairs of metric trees.

The thesis ultimately focused on developing the ENJ algorithm. However the ENJ algorithm has not yet been tested for its resilience in the face of distance-proportional errors. This is partly because it remains unclear how to best test the algorithm. One idea is to start with a phylogenetic tree T and its associated tree-distance matrix D, then add to D a symmetric matrix E where the entries of E are the entries of

D multiplied by some random number. The hope is that when E is added to D it simulates measurement error which is proportional to the entries of D. There are several problems with this approach. For one thing it is unclear as to what range of random numbers would best simulate experimental phylogenetic error. For another it is unclear that merely adding proportional error to the distance matrix would accurately simulate experimental sampling, and tree error in the first place.

Ideally testing the effectiveness of the ENJ algorithm will require producing simulated evolutionary data using some model of evolution. Then we would know the true tree but have error filled data that may not necessarily reflect it. On this simulated data we apply various Neighbor Joining algorithms and compare the produced trees to the known true tree. Then we can judge the effectiveness of the ENJ algorithm by comparing, on average, if it produced a more accurate tree than the other Neighbor Joining algorithms. The accuracy of the algorithms can be tested by applying one of several methods to measure how different two trees are from one another. We believe that the topology of the resulting trees is more important than the metric of the trees and so we will want to use a method that counts topological differences in trees.

To construct the simulations will require making several arbitrary decisions such as which model of evolution to simulate, what parameters should be used in the simulated evolution, what metric to use to judge differences in trees, what shape of the true tree should we focus our efforts on, and so on. On top of this there is the issue of which batch of simulations to include as it is probable that there are at least some of them which will heavily favor the ENJ algorithm and others which will heavily punish it. To answer these concerns will require further research and experimentation.

References

[BSH00] William J Bruno, Nicholas D Socci, and Aaron L Halpern. Weighted neighbor joining: a likelihood-based approach to distance-based phylogeny reconstruction. *Molecular Biology and Evolution*, 17(1):189–197, 2000.

[CC94] Trevor F Cox and Michael AA Cox. *Multidimensional scaling.* Chapman & Hall, 1994.

[dVAO11] Damien M de Vienne, Gabriela Aguileta, and Sébastien Ollier. Euclidean nature of phylogenetic distance matrices. *Systematic biology*, 60(6):826–832, 2011.

[Gas97] Olivier Gascuel. BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Molecular biology and evolution*, 14(7):685–695, 1997.

[PCS04] E. Paradis, J. Claude, and K. Strimmer. APE: analyses of phylogenetics and evolution in R language. *Bioinformatics*, 20:289–290, 2004.

[R C13] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2013.

[SK88] J. Studier and K. Keppler. A Note on the Neighbor-Joining Algorithm of Saitou and Nei. 5:729–731, 1988.

[SS03] Charles Semple and Mike A Steel. *Phylogenetics*, volume 24. Oxford University Press, 2003.