

Phylogenetic tree construction

Nicholas Eriksson

Department of Mathematics
University of California, Berkeley

7 December, 2005

Primera Escuela Argentina de Matemática y Biología

Metrics

A **metric** is a function

$$d(i, j): \{1, \dots, n\} \times \{1, \dots, n\} \rightarrow \mathbb{R}$$

where

$$d(i, i) = 0 \quad d(i, j) \geq 0 \quad \text{and} \quad d(i, j) \leq d(i, k) + d(k, j)$$

for all $i, j, k \in \{1, \dots, n\}$.

Example

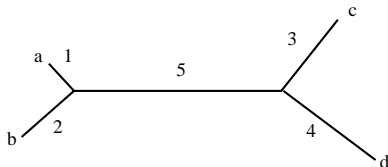
Let $n = 4$

$$(d(i, j)) = \begin{pmatrix} 0 & 3 & 9 & 10 \\ 3 & 0 & 10 & 11 \\ 9 & 10 & 0 & 7 \\ 10 & 11 & 7 & 0 \end{pmatrix}$$

Is this a metric?

A tree metric

$$\begin{pmatrix} 0 & 3 & 9 & 10 \\ 3 & 0 & 10 & 11 \\ 9 & 10 & 0 & 7 \\ 10 & 11 & 7 & 0 \end{pmatrix}$$



Tree metrics

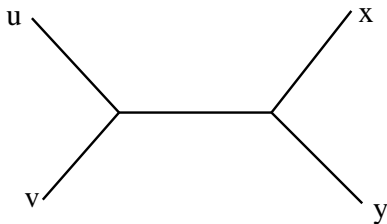
Do all metrics come from a tree?

Background

Neighbor joining

Maximum
likelihood

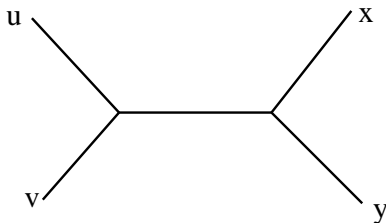
treeSVD



Do all metrics come from a tree?

Theorem (The four-point condition)

A metric d is a tree metric if and only if, for any four leaves u, v, x, y , the maximum of the three numbers $d(u, v) + d(x, y)$, $d(u, x) + d(v, y)$ and $d(u, y) + d(v, x)$ is attained at least twice.



Background

Neighbor joining

Maximum
likelihood

treeSVD

Theorem (Cherry-Picking Theorem)

If d is a tree metric on $[n]$ and

$$Z_d(i, j) = \sum_{k, l \in [n] \setminus \{i, j\}} w(ij; kl)$$

then any pair of leaves that maximizes $Z_d(i, j)$ is a cherry in the tree.

$w(ij; kl)$ is the length of the interior edge connecting i, j and k, l

$$w(ij; kl) = \frac{1}{4}(d(i, k) + d(i, l) + d(j, k) + d(j, l) - 2[d(i, j) + d(k, l)])$$

Starting with a multiple alignment, build a metric d . Then

1. Pick the maximal entry (i, j) in the matrix Z_d
2. Join leaves i and j together in the tree.
3. Replace the metric d by collapsing i and j together
4. Repeat until finished

Theorem

If d is a tree metric, then neighbor-joining constructs the correct tree.

Maximum likelihood

Recall: maximum likelihood takes the data and estimates the most likely parameters given the data.

If we fix the tree topology, maximum likelihood can estimate edge lengths and also give a likelihood for this topology. But how do we search through the space of trees?

One way:

1. Split the data into subsets.
2. For each subset, compute the maximum likelihood tree for those species.
3. Attempt to patch these trees together into a larger tree.

An algebraic algorithm

- ▶ We can use polynomials to decide if a split of the n species occurs in their phylogenetic tree.
- ▶ Our algorithm constructs a phylogenetic tree by computing only $O(n^2)$ SVD's, each of a very large but sparse matrix.
- ▶ For example, we can decide that {Rat, Mouse} is a good split and {Rat, Human} is a bad split by calculating the SVD of a 16×16 matrix.
- ▶ We only construct the tree topology. Once this is known, the branch lengths can be estimated quickly using maximum likelihood.

Assumption: Evolution follows a Markov model on a tree, with evolution at different sites occurring independently.

We do not assume any particular model (Jukes-Cantor, Kimura, reversible, HKY, etc.) or the existence of a global rate matrix.

Our algorithm is particularly suitable for problems where one of these models is violated.

Flattenings

Flattening

Given a split (A, B) , form a matrix where the rows are indexed by possible bases for the species in A and the columns by the possibilities for B . The entries of this matrix are the joint probabilities of observing a pattern at the leaves.

Example

The flattening along $(\{1, 3\}, \{2, 4\})$:

$$\begin{array}{c}
 \text{AA} \\
 \text{AC} \\
 \text{AG} \\
 \text{AT} \\
 \text{CA} \\
 \vdots
 \end{array}
 \begin{pmatrix}
 \rho_{AAAA} & \rho_{AAAC} & \rho_{AAAG} & \rho_{AAAT} & \rho_{ACAA} & \rho_{ACAC} & \dots \\
 \rho_{AACA} & \rho_{AACC} & \rho_{AACG} & \rho_{AACT} & \rho_{ACCA} & \rho_{ACCC} & \dots \\
 \rho_{AAGA} & \rho_{AAGC} & \rho_{AAGG} & \rho_{AAGT} & \rho_{ACGA} & \rho_{ACGC} & \dots \\
 \rho_{AATA} & \rho_{AATC} & \rho_{AATG} & \rho_{AATT} & \rho_{ACTA} & \rho_{ACTC} & \dots \\
 \rho_{CAAA} & \rho_{CAAC} & \rho_{CAAG} & \rho_{CAAT} & \rho_{CCAA} & \rho_{CCAC} & \dots \\
 \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
 \end{pmatrix}.$$

Theorem

For any 4 state Markov model on a tree, the flattenings along splits in the tree must have rank 4.

Theorem

The flattenings along non-splits (almost always) have rank at least 16.

This implies that if the data comes exactly from a tree model, we can tell the difference between good and bad splits.

But what about if the data has noise?

Singular value decomposition

- ▶ The SVD can measure how close a matrix is to being rank k .
- ▶ A singular value decomposition of a $m \times n$ matrix A is a factorization $A = U\Sigma V^T$ where U , V are orthogonal and Σ is diagonal with entries $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_m$. The distance from A to the nearest rank k matrix is

$$\min_{\text{Rank}(B)=k} \|A - B\|_2 = \sigma_{k+1}.$$

- ▶ The largest singular values can be computed quickly for very large sparse matrices. Thus we can quickly estimate the distance between a flattening and rank 4.

Algorithm for tree construction

Input: multiple alignment of n species.

1. for each pair of species:
compute the SVD for the flattening of {pair},{other species}
2. Pick the pair which is closest to rank 4.
3. Join this pair together in the tree.
4. Repeat until tree is constructed.

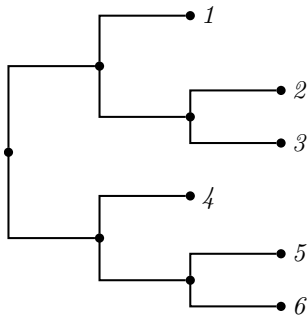
Output: tree with n leaves (without branch lengths).

Theorem

This algorithm is statistically consistent for binary trees.

Example

We begin with an alignment of DNA data of length 1000
for 6 species from the tree



Partition	Score
2 3 1 4 5 6	5.8374
5 6 1 2 3 4	6.5292
1 2 3 4 5 6	20.4385
1 3 2 4 5 6	20.5153
4 6 1 2 3 5	23.1477
4 5 1 2 3 6	23.3001
1 4 2 3 5 6	44.9313
3 4 1 2 5 6	52.1283
2 4 1 3 5 6	52.6763
1 6 2 3 4 5	52.9438
1 5 2 3 4 6	53.1727
3 6 1 2 4 5	59.5006
3 5 1 2 4 6	59.7909
2 6 1 3 4 5	59.9546
2 5 1 3 4 6	60.3253

picked split 1 4 5 6 | 2 3
tree is 1 4 5 6 (2,3)

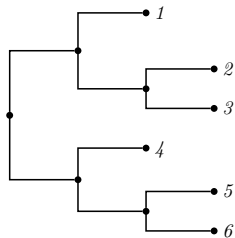
Repeat with 1 and 2 joined
together

Partition	Score
1 2 3 4 5 6	5.8534
5 6 1 2 3 4	6.5292
4 6 1 2 3 5	23.1477
4 5 1 2 3 6	23.3001
1 4 2 3 5 6	44.9313
2 3 4 1 5 6	45.1427
1 6 2 3 4 5	52.9438
2 3 6 1 4 5	53.0300
1 5 2 3 4 6	53.1727
2 3 5 1 4 6	53.3838
picked split 1 2 3 4 5 6	
tree is 4 5 6 (1,(2,3))	

Repeat with 1, 2, and 3
joined together

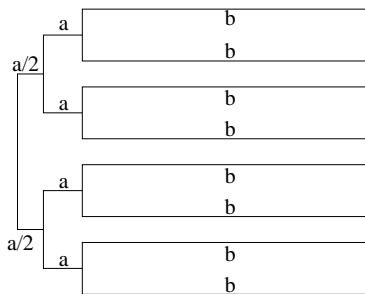
Partition	Score
5 6 1 2 3 4	6.5292
4 6 1 2 3 5	23.1477
4 5 1 2 3 6	23.3001
picked split 1 2 3 4 5 6	
tree is 4 (1,(2,3)) (5,6)	

Final tree is:
(4,(1,(2,3)),(5,6))



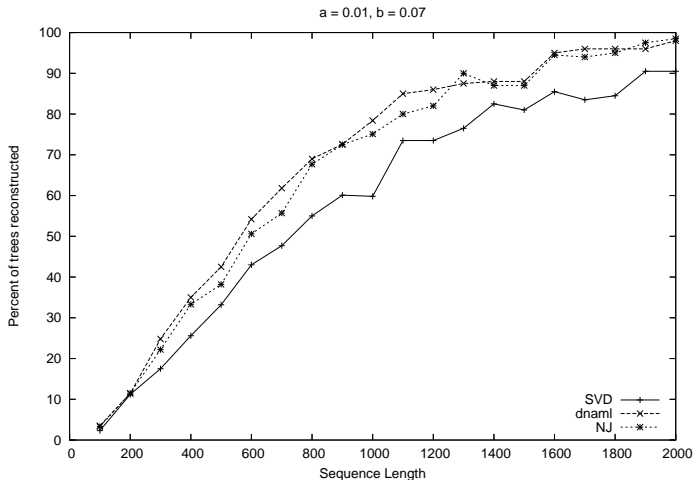
Simulation testing

We simulated *binary* data under the general reversible model with `seq-gen` and then compared our SVD method, neighbor joining, and `dnaml` for the tree:



with branch lengths $(a, b) = (0.01, 0.07)$ and $(0.02, 0.19)$.

Results



A real-life example

- ▶ 44 ENCODE regions, aligned to homologous regions in 8 mammals.
- ▶ Rodent placement is difficult with current models since rats and mice have evolved faster than other mammals.
- ▶ Tested SVD and `dnaml` on entire regions, genes only, and exons only.
- ▶ SVD produced the correct tree much more often than `dnaml` and produced a more correct tree on average.

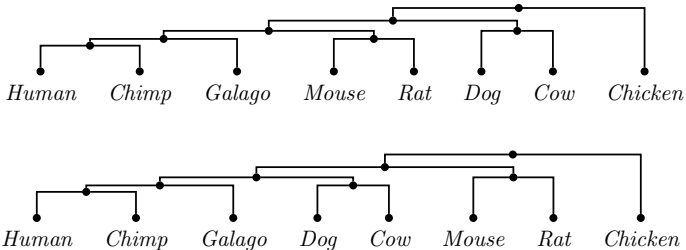


Figure 1: (Top) The accepted tree representing the evolution of our eight vertebrates. (Bottom) The tree usually constructed.

	SVD		dnaml	
	Ave. dist.	Correct	Ave. dist.	Correct
All	2.06	5.8%	3.29	2.9%
Genes	1.93	10.3%	3.21	0.0%
Exons	2.43	21.4%	3.0	3.5%

Table: Comparison of the SVD algorithm and `dnaml` on data from the ENCODE project. Distance between trees is given by the symmetric distance, percent correct gives the percentage of the regions which had the correct tree reconstructed.