

# Lecture 12: Molecular Evolution and Phylogenetic Tree Reconstruction

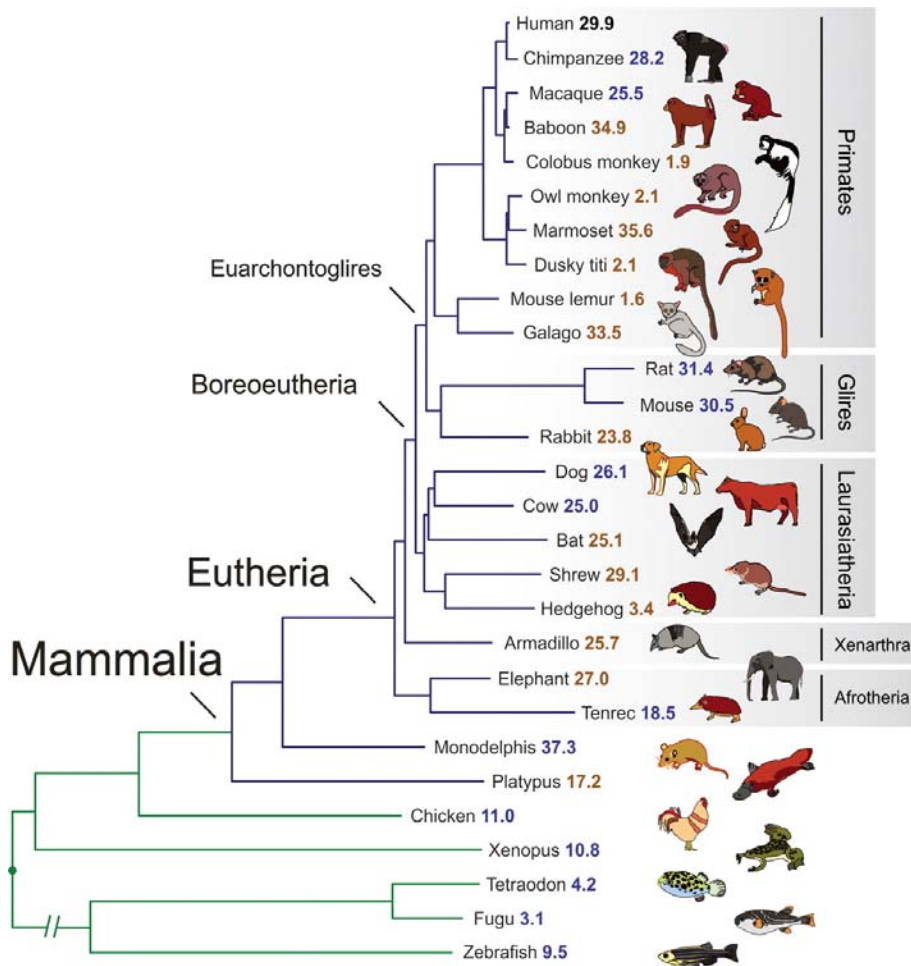
2-18-2009

Scribe: Linda Liu

We now discuss modeling sequence evolution at the nucleotide level in order to address problems in comparative genomics and evolution. The motivation for such studies includes questions like “how closely related are two species” and “how long ago did two species diverge?” Answers to these questions allow for usage of other genomes (e.g. mouse or drosophila) to analyze the human genome as well as to reconstruct the evolutionary tree of all species.

## A. Phylogenetic Trees

A phylogenetic tree is a representation of evolutionary relationships between a set of organisms. The nodes of the tree represent species, and the edges represent the time of independent evolution. The length of the edges thus represents the amount of evolutionary time between species, or how closely related two organisms are. This may not necessarily be chronological time, but is a proxy for “genetic distance”.



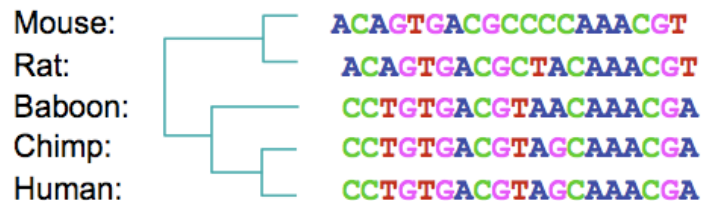
## B. Inferring Phylogenetic Trees

### 1. Morphology

A naïve way to differentiate species is by using morphological traits such as body size, number of legs, presence of fins, etc. However, this is not a reliable method of inferring evolutionary relationships as there is the possibility of convergent evolution (two different species evolving the same trait independently of each other due to similar environmental pressures).

### 2. Sequence comparison

A more reliable method of inferring phylogenetic trees is through sequence comparison. The reasoning behind these methods is that all organisms evolved from a common ancestor and that changes (mutations) in the DNA sequence over many generations led to the differences between different species that we see today.



The two main sequence comparison methods are:

- Distance-based methods
- Sequence-based methods

## C. Molecular Evolution

Before we go into sequence comparison methods, let's briefly discuss what we mean by molecular evolution. Here we are interested in how to model evolution on the nucleotide level, i.e. how nucleotide differences (called substitutions) accumulate over time.

Let's consider what happens at a specific nucleotide over a small time interval,  $\Delta t$ . We will estimate the rate of substitutions.

Let  $P(t)$  = vector of probabilities that represent seeing the nucleotides A,C,T or G at time  $t$ .

Let  $\mu_{XY}$  be the rate of transition from base X to base Y per unit time.

Then if a certain base is A,  $\mu_A = \mu_{AC} + \mu_{AG} + \mu_{AT}$  is the rate of transition out of base A.

The probability that a given base is A in time  $t+\Delta t$  is

$$p_A(t + \Delta t) = [p_A(t) - p_A(t)\mu_A\Delta t] + p_C(t)\mu_{CA}\Delta t + p_G(t)\mu_{GA}\Delta t + p_T(t)\mu_{TA}\Delta t$$

where the first term represents the probability that the base was previous A minus the probability that it transitioned out of A.

In matrix notation, we get  $P(t+\Delta t) = P(t) + Q P(t)\Delta t$  where  $Q$  is the substitution rate matrix:

$$Q = \begin{pmatrix} -\mu_A & \mu_{GA} & \mu_{CA} & \mu_{TA} \\ \mu_{AG} & -\mu_G & \mu_{CG} & \mu_{TG} \\ \mu_{AC} & \mu_{GC} & -\mu_C & \mu_{TC} \\ \mu_{AT} & \mu_{GT} & \mu_{CT} & -\mu_T \end{pmatrix}$$

$Q$  implies a probability distribution over  $\{A,C,G,T\}$  at each position, including transition probabilities and stationary (equilibrium) frequencies of bases  $\pi_A, \pi_C, \pi_G, \pi_T$ .

The above equation turns into the differential equation  $P'(t) = Q P(t)$ , and different solutions  $Q$  to this equation exist. Each  $Q$  gives a distance metric, and is a different model for evolution. There are four common models in practice:

### 1. Jukes-Cantor

$$Q = \begin{pmatrix} * & \frac{\mu}{4} & \frac{\mu}{4} & \frac{\mu}{4} \\ \frac{\mu}{4} & * & \frac{\mu}{4} & \frac{\mu}{4} \\ \frac{\mu}{4} & \frac{\mu}{4} & * & \frac{\mu}{4} \\ \frac{\mu}{4} & \frac{\mu}{4} & \frac{\mu}{4} & * \end{pmatrix}$$

This is the simplest model since it assumes that the substitution frequencies for all bases are equal and only has one parameter,  $\mu$ .

### 2. Kimura

$$Q = \begin{pmatrix} * & \kappa & 1 & 1 \\ \kappa & * & 1 & 1 \\ 1 & 1 & * & \kappa \\ 1 & 1 & \kappa & * \end{pmatrix}$$

This model uses different frequencies for transitions (purine to purine or pyrimidine to pyrimidine) and transversions (purine to pyrimidine or vice versa) and assumes that transitions are more likely than transversions.

### 3. Felsenstein

$$Q = \begin{pmatrix} * & \pi_T & \pi_T & \pi_T \\ \pi_C & * & \pi_C & \pi_C \\ \pi_A & \pi_A & * & \pi_A \\ \pi_G & \pi_G & \pi_G & * \end{pmatrix}$$

This model assumes that all substitutions are equally likely but that the equilibrium frequencies of the four nucleotides might not be the same.

### 4. HKY

$$Q = \begin{pmatrix} * & \kappa\pi_T & \pi_T & \pi_T \\ \kappa\pi_C & * & \pi_C & \pi_C \\ \pi_A & \pi_A & * & \kappa\pi_A \\ \pi_G & \pi_G & \kappa\pi_G & * \end{pmatrix}$$

This model is a combination of Kimura and Felsenstein.

## D. Distance-based Methods for Reconstructing Phylogenetic Trees

The basic principles behind distance-based methods include:

1. The degree of sequence difference between two species is proportional to the length of independent sequence evolution
2. Only use positions where the alignment is certain (avoid areas with gaps)

It will be useful to define what we mean by evolutionary distance.

Given sequences  $x^i, x^j$  and a trustworthy alignment between them, we can define

$d_{ij}$  = distance between the two sequences.

One possible definition:  $d_{ij}$  = fraction of sites  $u$  where  $x^i[u] \neq x^j[u]$ .

There are two common clustering methods for building trees

### 1. Average Linkage Method

(UPGMA - Unweighted Pair Group Method Using Arithmetic Averages)

This is the simplest method for constructing phylogenetic trees.

Given two disjoint clusters  $C_i, C_j$  of sequences, define a distance metric

$d_{ij} = \frac{1}{|C_i| \cdot |C_j|} \sum_{\{p \in C_i, q \in C_j\}} d_{pq}$ . The distance between two clusters is the sum of the distances

between the sequences in the clusters, normalized by the size of the clusters.

We then claim that if  $C_k = C_i \cup C_j$ , then distance to another cluster  $C_l$  is:  $d_{kl} = \frac{d_{il}|C_i| + d_{jl}|C_j|}{|C_i| + |C_j|}$ .

The basic idea is that given two clusters, we can merge them and then compute the distance from the merged cluster to any other cluster based on the distances between sequences in the clusters.

The algorithm for average linkage is as follows. It runs in quadratic time.

#### **Initialization:**

Assign each  $x_i$  into its own cluster  $C_i$

Define one leaf per sequence, height 0

#### **Iteration:**

Find two clusters  $C_i, C_j$  such that  $d_{ij}$  is minimal

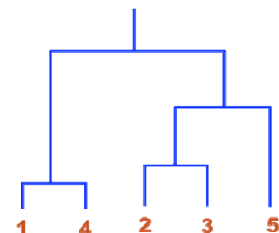
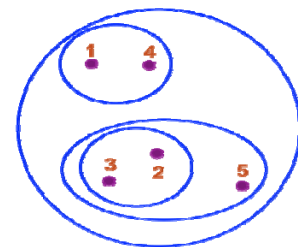
Let  $C_k = C_i \cup C_j$

Define a node connecting  $C_i, C_j$ , and place it at height  $d_{ij}/2$

Delete  $C_i, C_j$

#### **Termination:**

When two clusters  $i, j$  remain, place the root of the tree at height  $d_{ij}/2$



**Example:** each subsequent matrix corresponds to merging nodes and recomputing distances to remaining nodes.

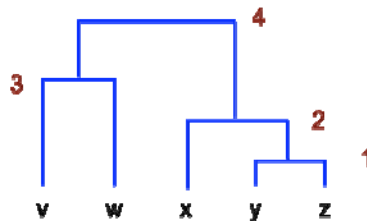
	v	w	x	y	z
v	0	6	8	8	8
w		0	8	8	8
x			0	4	4
y				0	2
z					0

	v	w	x	yz
v	0	6	8	8
w		0	8	8
x			0	4
yz				0

	v	w	xyz
v	0	6	8
w		0	8
xyz			0

	vw	xyz
vw	0	8
xyz		0

After we are down to two nodes, we can draw the corresponding tree:



**Ultrametric distances and the molecular clock:**

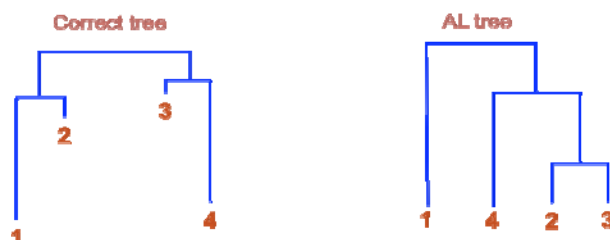
The Average Linkage method is guaranteed to correctly reconstruct a tree only if the distances are ultrametric.

A distance function  $d$  is **ultrametric** if for any three distances  $d_{ij} \leq d_{ik} \leq d_{kj}$ , it is true that  $d_{ij} \leq d_{ik} = d_{kj}$ .

This means that for two species  $i, j$  and their common ancestor  $k$ , the evolutionary distance between  $i$  and  $k$  is equal to the distance between  $j$  and  $k$  and greater than the distance between  $i$  and  $j$ .

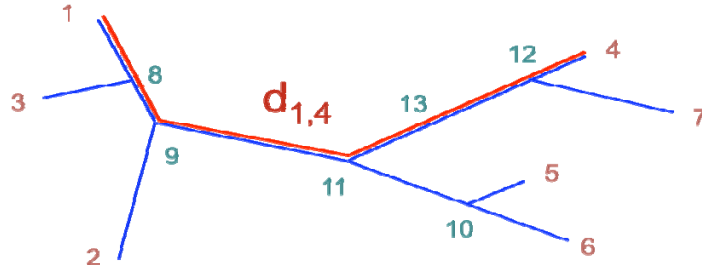
This is a result of the **molecular clock** concept, which says that the evolutionary distance between species  $x$  and  $y$  is  $2 \times$  the Earth time to reach the nearest common ancestor. That is, the molecular clock has a constant rate in all species – all species evolve at the same rate.

Unfortunately this is not true – certain species evolve faster than others due to shorter generation times and greater numbers of offspring. This can lead to the Average Linkage method reconstructing the wrong tree:



A solution to this problem is using additive distances instead of ultrametric ones.

Given a tree, a distance measure is **additive** if the distance between any pair of leaves is the sum of lengths of the edges connecting them.

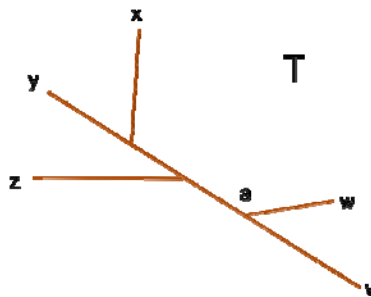


Given a tree  $T$  and additive distances  $d_{ij}$ , we can uniquely reconstruct edge lengths:

- Find two neighboring leaves  $i, j$ , with common parent  $k$
- Place parent node  $k$  at distance  $d_{km} = \frac{1}{2} (d_{im} + d_{jm} - d_{ij})$  from any node  $m \neq i, j$

**Example:** We know the tree  $T$  and matrix  $D$  of pairwise distances between nodes, but don't know the lengths of each leaf. We can use the following equations to reconstruct the leaves:

	v	w	x	y	z
v	0	10	17	16	16
w		0	15	14	14
x			0	9	15
y				0	14
z					0



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

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

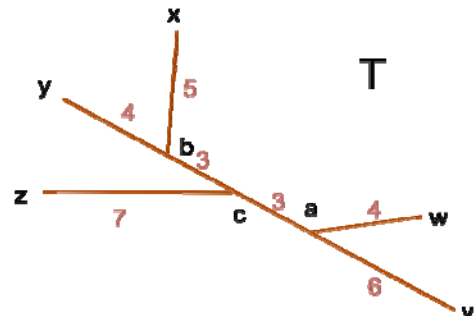
$$d_{az} = \frac{1}{2} (d_{vz} + d_{wz} - d_{vw})$$

Using the same idea of merging nodes, we get:

	a	x	y	z
a	0	11	10	10
x		0	9	15
y			0	14
z				0

	a	b	z
a	0	6	10
b		0	10
z			0

	a	c
a	0	3
c		0



We have successfully reconstructed all the internal nodes of the tree.

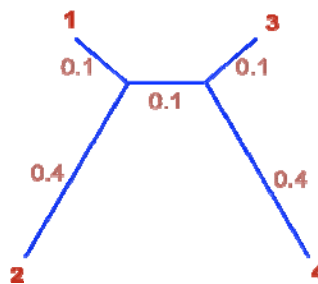
## 2. Neighbor Joining

The second distance-based method of tree construction takes advantage of additive distances. It is guaranteed to reconstruct the correct tree if distances are additive, and may construct a good tree even if distances are not additive.

Given the set of species and the pairwise distances between their sequences (calculated using Jukes-Cantor or another method), we can reconstruct the tree by finding neighboring species (leaves) and joining them until we are down to two nodes.

To find neighboring leaves, we define  $D_{ij} = (N - 2)d_{ij} - \sum_{k \neq i} d_{ik} - \sum_{k \neq j} d_{kj}$  where  $N$  is the number of leaf nodes (species) in the tree. This definition ensures that  $D_{ij}$  is minimal if and only if  $i$  and  $j$  are neighbors. We can't just find the two nodes that are closest together – they may not be neighbors. This definition ensures that the two nodes are close to each other and far from all other nodes.

In this example, nodes 1 and 3 are the closest two leaf nodes, but they are not direct neighbors. 1 and 2 are neighbors, and 3 and 4 are neighbors.



The algorithm for neighbor joining is as follows. It runs in cubic time. The time-consuming step is finding leaf nodes that are neighbors (minimizing  $D_{ij}$ ).

### **Initialization:**

Define  $T$  to be the set of leaf nodes, one per sequence  
Let  $L = T$

### **Iteration:**

Pick  $i, j$  such that  $D_{ij}$  is minimal  
Define a new node  $k$ , and set  $d_{km} = \frac{1}{2} (d_{im} + d_{jm} - d_{ij})$  for all  $m \in L$   
Add  $k$  to  $T$ , with edges of lengths  $d_{ik} = \frac{1}{2} (d_{ij} + r_i - r_j)$ ,  $d_{jk} = d_{ij} - d_{ik}$   
where  $r_i = (N - 2)^{-1} \sum_{k \neq i} d_{ik}$   
Remove  $i, j$  from  $L$   
Add  $k$  to  $L$

### **Termination:**

When  $L$  consists of two nodes,  $i, j$ , and the edge between them of length  $d_{ij}$

## E. Sequence-based Methods for Reconstructing Phylogenetic Trees

There are two main sequence-based approaches: Deterministic and Probabilistic

### 1. Deterministic - Parsimony

The basic idea behind parsimony is that given a set of species with multiple alignment information, we want to reconstruct the tree that explains the observed sequences with the fewest possible substitutions. The premise is that during the course of evolution, it is more likely that substitutions at particular nucleotides only occurred once or a very few times.

Two computational subproblems are involved in finding the most parsimonious tree:

- Find the parsimony cost of a given tree (easy)
- Search through all tree topologies if not given a tree (hard)

The algorithm for finding the parsimony cost for a tree (**parsimony scoring**) is as follows.

Given a tree, and an alignment column  $u$ , label the internal nodes to minimize the number of required substitutions.

#### **Initialization:**

Set cost  $C = 0$ ; node  $k = 2N - 1$  (last leaf)

#### **Iteration:**

If  $k$  is a leaf, set  $R_k = \{ x^k[u] \}$  //  $R_k$  is simply the character of  $k^{\text{th}}$  species

If  $k$  is not a leaf,

Let  $i, j$  be the daughter nodes;

Set  $R_k = R_i \cap R_j$  if intersection is nonempty

Set  $R_k = R_i \cup R_j$ , and increment  $C$  if intersection is empty

**Termination:** Minimal cost of tree for column  $u = C$

The basic idea of this algorithm is to start from the bottom up, labeling the leaf nodes with their own letters. At each node that is not a leaf, look at its children nodes and label with the intersection if it is nonempty. This does not increase the cost associated with that node. If the children have no labels in common, we must take the union of their labels. This increases the cost. This algorithm assumes all substitutions are equally bad.

#### **Parsimony traceback algorithm:**

1. Choose an arbitrary nucleotide from  $R_{2N-1}$  for the root

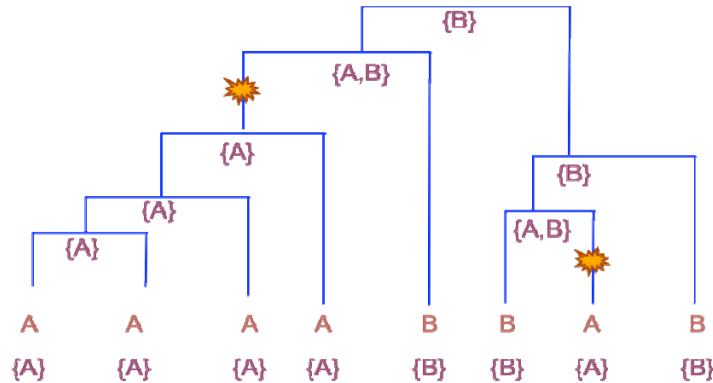
2. Having chosen nucleotide  $r$  for parent  $k$ ,

If  $r \in R_i$  choose  $r$  for daughter  $i$

Else, choose arbitrary nucleotide from  $R_i$

It's easy to see that this traceback produces some assignment of cost  $C$ , however it may not give all possible parsimonious tracebacks.

**Example:** The most parsimonious set of substitutions changes B to A twice.



**Another parsimony algorithm:**

Let  $C(v)$  be cost for subtree rooted at node  $v$

Let  $C(v,x)$  be cost for subtree rooted at  $v$  if we force  $v$  to have value  $x$

**Initialization:**

For each leaf  $v$

$$C(v) = 0$$

$C(v,x) = 0$  if  $x$  is input character that labels  $v$ ;  $C(v,x) = \infty$  otherwise

**Iteration:**

Let  $u, w$  be children of  $v$

$$C(v,x) = \min(C(u) + 1, C(u,x)) + \min(C(w) + 1, C(w,x))$$

$$C(v) = \min C(v,x)$$

**Termination:**

Minimal cost is  $C(\text{root})$

This algorithm does not assume all substitutions are equal cost so it can generalize to more situations. It is similar to Viterbi in the sense that we find the most likely (least cost) assignment to the internal nodes of the tree, given the leaves.

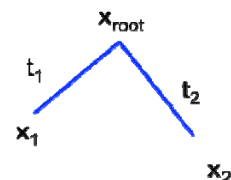
**2. Probabilistic**

Probabilistic methods are a more refined measure of evolution along a tree than parsimony.

We can define the probability of leaves and their root given priors:

$$P(x_1, x_2, x_{\text{root}} | t_1, t_2) = P(x_{\text{root}}) P(x_1 | t_1, x_{\text{root}}) P(x_2 | t_2, x_{\text{root}})$$

The priors may come from Jukes-Cantor or another model.

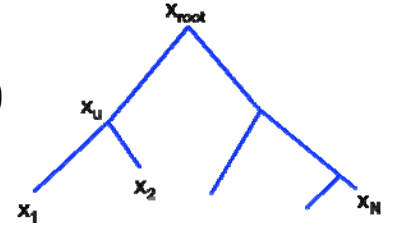


If we know all internal labels  $x_u$ ,

$$P(x_1, x_2, \dots, x_N, x_{N+1}, \dots, x_{2N-1} \mid T, t) = P(x_{\text{root}}) \prod_{j \neq \text{root}} P(x_j \mid x_{\text{parent}(j)}, t_j, \text{parent}(j))$$

Usually we don't know the internal labels, therefore

$$P(x_1, x_2, \dots, x_N \mid T, t) = \sum_{x_{N+1}} \sum_{x_{N+2}} \dots \sum_{x_{2N-1}} P(x_1, x_2, \dots, x_{2N-1} \mid T, t)$$



### Felsenstein's likelihood algorithm:

To calculate  $P(x_1, x_2, \dots, x_N \mid T, t)$ . Let  $P(L_k \mid a)$  denote the probability of all the leaves below node  $k$ , given that the residue at  $k$  is  $a$ .

#### Initialization:

Set  $k = 2N - 1$

#### Iteration: Compute $P(L_k \mid a)$ for all $a \in \Sigma$

If  $k$  is a leaf node:

$$\text{Set } P(L_k \mid a) = \mathbf{1}(a = x_k)$$

If  $k$  is not a leaf node:

1. Compute  $P(L_i \mid b)$ ,  $P(L_j \mid b)$  for all  $b$ , for daughter nodes  $i, j$
2. Set  $P(L_k \mid a) = \sum_{b,c} P(b \mid a, t_i) P(L_i \mid b) P(c \mid a, t_j) P(L_j \mid c)$

#### Termination:

$$\text{Likelihood at this column} = P(x_1, x_2, \dots, x_N \mid T, t) = \sum_a P(L_{2N-1} \mid a) P(a)$$

This algorithm is analogous to the Forward and Backward algorithms from HMMs.

Let  $S(i,j)$  be the subtree containing nodes  $i$  and  $j$ .

Define:

$$U_{i \rightarrow j}(a) \equiv P(\{X_k = x_k\}_{k \in S(i,j)} \mid X_i = a, T, t)$$

$$u_{i \rightarrow j}(a) \equiv P(\{X_k = x_k\}_{k \in S(i,j)} \mid X_j = a, T, t)$$

and recursively compute:

$$U_{i \rightarrow j}(a) = \begin{cases} \mathbf{1}\{x_i = a\} & i \text{ is a leaf} \\ \prod_{k \neq j: (k,i) \in T} u_{k \rightarrow i}(a) & i \text{ is an internal node} \end{cases}$$

$$u_{i \rightarrow j}(a) = \sum_b p_{a \rightarrow b}(t_{i,j}) U_{i \rightarrow j}(b)$$

Here  $U_{i \rightarrow j}(a)$  is analogous to the "forward" probability (probability of one subtree) and  $u_{i \rightarrow j}(a)$  is the "backward" probability (probability of the other subtree). For  $u_{i \rightarrow j}(a)$ , we sum over all nucleotides  $b$  that  $a$  goes to.

Then the probability of the leaves of  $S(i,j)$  given the root of that subtree is the product of the left and right subtrees.

Using  $u$  and  $U$  we can compute this product:

$$\mathbb{P}(x^{[1\dots N]} | \mathcal{L}, \mathbf{t}) = \sum_{\alpha} b^{\alpha} \Omega^{s \rightarrow \lambda}(\alpha) \Omega^{\lambda \rightarrow s}(\alpha)$$

We can also do posterior decoding using:

$$P(X_i = a | x_{[1\dots N]}, T, \mathbf{t}) = \frac{P(a)U_{i \rightarrow j}(a)u_{j \rightarrow i}(a)}{P(x_{[1\dots N]} | T, \mathbf{t})}$$

$$P(X_i = a, X_j = b | x_1, \dots, x_N, T, \mathbf{t}) = \frac{P(a)U_{i \rightarrow j}(a)p_{a \rightarrow b}(t_{i,j})U_{j \rightarrow i}(b)}{P(x_{[1\dots N]} | T, \mathbf{t})}$$

### Tree reconstruction:

Given  $M$  (ungapped) alignment columns of  $N$  sequences, we can define the likelihood of a tree:

$$L(T, \mathbf{t}) = P(\text{Data} | T, \mathbf{t}) = \prod_{m=1\dots M} P(x_{1m}, \dots, x_{nm} | T, \mathbf{t})$$

Maximum Likelihood Reconstruction:

Given data  $X = (x_{ij})$ , find a topology  $\mathbf{T}$  and length vector  $\mathbf{t}$  that maximize likelihood  $L(\mathbf{T}, \mathbf{t})$

This is not feasible in practice, and approximation methods are needed.

SEMPHY is a popular program: <http://www.cs.huji.ac.il/labs/compbio/semphy/>