摘 要

我們主要是將 *Hidden Markov model* 應用在演化樹 *(phylogenetic tree)* 的結構上面， 也就是讓演化樹中的邊 *(branch)*有一個機率，我們根據這一個機率來決定我們接不接受這一個邊。這一個機率就是*Gibbs sampler method*中的*proposal distritubtion*。這一個採樣的方法會在第五章詳細說明。

因為同時執行演化樹和多重序列比對*(multiple sequence alignment)*是一件困難的電腦計算的任務。 很多方法不是限制序列的多寡就是使用非常簡單的演化樹和多重比對的模型。 我們會在第六章介紹一個方法來克服以上的種種限制。 我們在第五章介紹一個簡單的方法來讓我們可以很輕鬆去理解第六章所要介紹的方法。 在第五章有介紹到一種採樣的方法。這一個方法是奠基在假設有一個演化樹的情況之下， 我們要如何去做多重序列比對。然而再第六章卻是同時做好這兩件事。

我們首先會介紹 *Hidden Markov models* 的基本概念，再來簡單的介紹三種常用的演化樹，然後利用 *Markov chain Monte Coral* 中的 *Gibbs sampling method* 將這兩個觀念做結合。最後介紹我們的主題， 就是使用*Gibbs sampling method*配合使用*Hidden Markov models*進而達到我們的目的。

關鍵字：*Hidden Markov model*，演化樹 *(phylogenetic tree)*，*Markov chain Monte Corlo*，*tree Hidden Markov models*。

Abstract

We apply Hidden Markov Model to phylogenetic tree construction; that is, we compute the probabilities of the branches of the phylogentic tree and decide whether to accept it or not. The probabilities are the proposal distributions in the Gibbs sampler method. We will discuss the method in chapter 5.

Carrying out simultaneous tree-building and alignment of sequence data is a difficult computational task. Many methods are either limited to a few sequences or restricted to highly simplified models of alignment and phylogeny. A method is given in chapter 6 for overcoming these limitations. We introduce a simple method in chapter 5 in order to help us to understand the method in chapter 6. In chapter 5, we introduce a sampling method. This sampling method do a multiple sequences alignment conditioned on a phylogenetic tree. However we can do the two things simultaneously in chapter 6.

We will first introduce the basic concepts of the Hidden Markov models. Then we will discuss the phylogenetic tree and three kinds of the phylogenetic tree construction methods most commonly used. And we will use the Gibbs sampling method of the Markov chain Monte Carlo to combine the Hidden Markov models and sample the phylogenetic tree in a tree Hidden Markov model.

Keywords : Hidden Markov Model, phylogenetic tree, Markov chain Monte Carlo, tree Hidden Markov Model.

Contents

Chapter 1

Introduction

We know that all species are related because all species share similar genes and have similar gene functions. But how do biologists judge the relationships of any two species? Building a phylogenetic tree is a simplest way to find the relationships of all species.

There are many methods to build the phylogenetic trees. Most of them assume the site independence; that is, one site does not affect the evolution of its adjacent site. For instance, the Maximum parsimony, Maximum likelihood and so on have this assumption. But this assumption does not conform to the actual evolution.

All these phylogenetic tree construction methods are based on multiple sequences alignment. But the commonly used multiple sequences alignment method is actually based by a guide tree. Phylogenetic tree building by this way may be incorrect. We will follow Mitchison' paper [18] to introduce a method which can carry out phylogenetic tree and multiple alignment at the same time. This method was first introduced by Mitchison and Durbin (1995) [17].

This does not need this assumption of the independence of evolution of individual site. The method uses the Hidden Markov model (HMM) to explain the relationship between one site and its adjacent sites. This model is called the tree-HMM.

The basic theory of hidden Markov models was published in a series of classic papers by Baum and his colleagues in the late 1960s and early 1970s and was implemented for speech processing applications by Baker at CMU, and by Jelinek and his colleagues at IBM in the 1970s. The Hidden Markov model is composed of a number of states, which might correspond to positions in a 3D structure or columns of a multiple alignment. Each state 'emits' symbols (residues) according to symbol emission probabilities. The movement from one state to the next state corresponds to the state-transition probability.

Starting from some initial state, a sequence of states is generated by moving from state to state according to the state-transition probabilities until an end state is reached. Then each state emits symbols according to that state's emission probability distribution, creating an observable sequence of symbols. Why are called the hidden Markov model? Because that this state sequence is usually not observed.

A phylogenetic tree is assumed to be binary here and has branches, just as in plants. Three methods which are distance methods, maximum parsimony, and maximum likelihood are commonly used. Distance methods works by clustering the sequences. At each step it combines two clusters and at the same time creating a new node on a tree. The tree can be imagined as being collected upwards, each node being added above the others.

Maximum parsimony is one of the most widely used tree building algorithms. It works by finding the tree of the observed sequences with a minimal number of substitutions. Instead of building a tree, it assigns a cost to a given tree and it is necessary to search through all topologies in order to identify the 'best' tree.

Maximum likelihood method uses probability calculations to find a tree that best accounts for the variation in a set of sequences. All possible trees are considered. It uses explicit evolutionary models such as the Jukes-Cantor and Kimura models with allowances for variations in base composition.

For the tree Hidden Markov Model, we follow the discussions in G.J. Mitchison (1999) [18] and Ian Holmes and William J. Bruno (2001) [12]. In Mitchison's paper, he developed a model: tree-HMM. The tree-HMM is similar to the profile HMM. In the tree-HMM there is no insertion states but insertions are still allowed (see Figure 1.1). Insertions occur when a sequence uses a match state at a position where its ancestor uses a delete state. Hence it is not necessary represented by a special state.

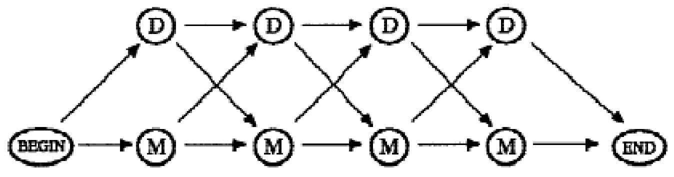


Figure 1.1: The tree-HMM; come from G.J. Mitchison (1999); Excerpt from [18]

Consider the simplest possible evolutionary tree, T, consisting of a single edge of length d, with a leaf node at one end and a root node at the other. Let the sequence of the leaf be x and that at the y, and the length of the model is 4 (see Figure 1.2).

In the alignment we can see that y go through states M, M, D, D, i.e., the symbols are represented to go through the match state and '-' is represented to go through the delete state. Hence x go through states M, D, D, M. There are transition probabilities that x is derived from y for one position to the next position. And we can see that x goes through D but y goes through M at the position 2. Then we can not regarded that x is derived from y by substitution.

To solve this problem, there is a convenient rule. This rule is namely the '* rule'. The * rule replaces the missing ancestral or descendent sequences at such positions by sum over all possible emissions or transitions. After the definition of emission and transition probabilities, we can calculate the probability of $P(x, y|T)$. Since we do not know what is the sequence, y. We must sum over all possible emissions and transitions, for y, to get the probability of the observed sequence, x.

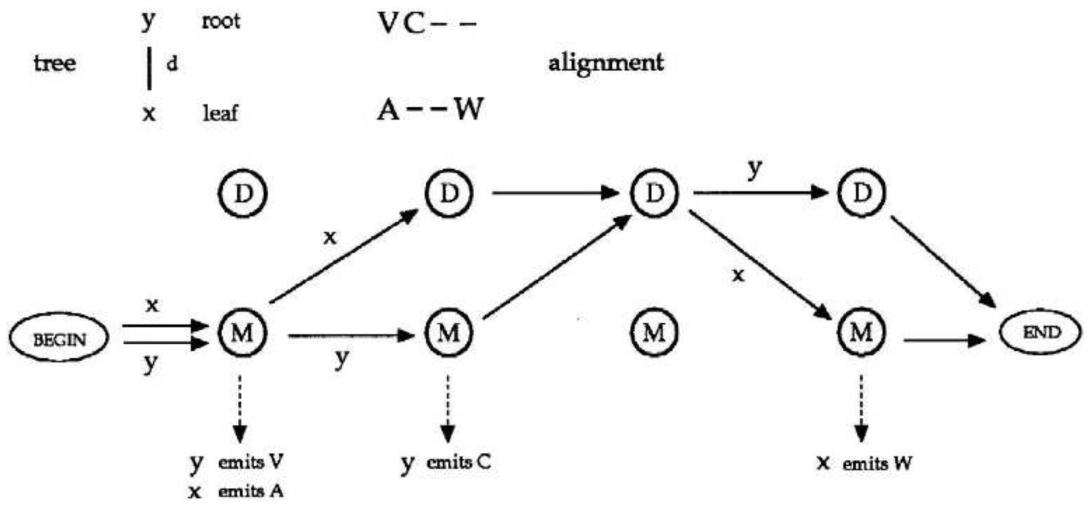


Figure 1.2: An simplest example for the tree-HMM; G.J. Mitchison (1999); Excerpt from [18]

In Holmes and Bruno's paper, they use a given tree to do multiple sequences alignment. They also use the tree-HMM, and they add the link model proposed by Thorne et al (1991, 1992). They develop a multiple alignment algorithm by Bayesian inference conditioned on a tree.

This tree-HMM has the advantage that it can have deletions that are more than one base long. But it suffers from some lack of realism. When a group of adjacent bases is deleted, the bases retain information about the base sequence. And if they are inserted again, there will be some memory of the original base sequence. Holmes and Bruno (2001) [12] point out that it is also possible that when a series of bases is reinsertied there may be 'memory' of an internal gap that was once there and that now returns with them.

Mitchison's paper is difficult to understand and has no implements and no source code. Mitchison also used Bayesian sampling to do tree-building and alignment of sequence data simultaneous. Hence we use Holmes and Bruno's paper [12] to help us to understand Mitchison's paper. The goal of this thesis is to do a simple and detailed illustration of the methods in Mitchison's paper [18].

Chapter 2

Markov chains and hidden Markov models

Our materials come from [3] and [5]. This main goal of chapter will introduce a probabilistic model for sequences of symbols, called a hidden Markov model (abbreviated HMM). The types of question we can use HMMs and their near relations, Markov Models, to consider are: 'Does this sequence belong to a particular family?' or 'Assuming the sequence does come from some family, what can we say about its internal structure?'

The hugeous majority of papers on HMMs belong to the speech recognition literature, where they were applied first in the early 1970s. Many problems in biological sequence analysis have the same structure of the speech recognition, so it is applied to biology.

Example: CpG islands

In the human genome, there is a relatively high chance of the methyl-C mutating into a T, with the result that in general CpG dinucleotides are unusual in the genome than in random. For biologically important reasons the methylation process is suppressed in short stretches of the genome, such as around the promoters or 'start' regions of many genes. In these regions we see many more CpG dinucleotides than elsewhere, and such regions are called CpG islands.

We will consider two questions: Given a short stretch of genomic sequence, how would we decide if it comes from a CpG island or not? Given a long piece of sequence, how would we find the CpG islands in it, if there are any?

2.1 Markov chains

A classical Markov chain is a model that generates sequences in which probability of a symbol depends on the previous symbol. In other words, the probabilities of each symbol x_i depends only on the value of the preceding symbol x_{i-1}, not on the entire previous sequence, i.e., $P(x_i|x_{i-1}, x_{i-2}, \ldots, x_1) = P(x_i|x_{i-1})$. We like to show a Markov chain graphically as a collection of ‘state’, each of which corresponds to a particular residue, with arrows between the states. The transition probability is the probability parameter of the arrows, which determines the probability of a certain residue following another residue, or one state following another state and we denote it by

$$a_{st} = P(x_i = t|x_{i-1} = s)$$

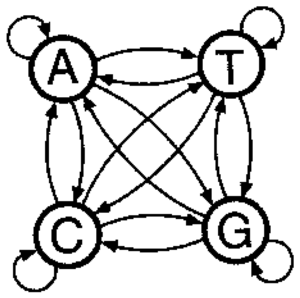


Figure 2.1: *a Markov chain graphically as a collection of ‘state’, each of which corresponds to a particular residue, with arrows between the states; Excerpt from [3].*

2.2 Hidden Markov models

Given a simple example to introduce what the Hidden Markov models are. In a casino they used a fair die most of the time, but occasionally they switch to a loaded die. The switch between dice is a Markov process. What is hidden in the model? If you can just see a sequence of rolls and you do not know which rolls used a loaded die and which used a fair one, because that is kept secret by the casino; that is, the *state sequence is hidden.*

Hence the essential difference between a Markov chain and a hidden Markov model is that for a hidden Markov model there is not a one-to-one correspondence between the states and the symbols. Let us call the state sequence the path, π. The ith state in the path is called π_i. The transition probability is the probability of the state to next state, i.e., $a_{kl} = P(\pi_i = l|\pi_{i-1} = k)$. The emission probability is the probability that a symbol b is seen when in state k, i.e., $e_k(b) = P(x_i = b|\pi_i = k)$

2.2.1 Most probable state path: the Viterbi algorithm

Although it is no longer possible to tell what state the system is in by looking at the corresponding symbol, it is often the sequence of underlying states that we are interested in. There are several approaches to find out what the observation sequence of underlying states. Here we will describe the most common one, called the Viterbi algorithm.

In general, many state sequences might give raise the same symbol sequence. For example, in the CpG model the state sequences (C_+, G_+, C_+, G_+), (C_-, G_-, C_-, G_-), (C_+, G_-, C_+, G_-) would generate the symbol sequence CGCG. The third is the product of multiple small probabilities of switching CpG islands and non CpG islands between the components and it is much smaller than others. The second is much smaller than the first because it contains two C to G transitions which are less probable in the '-' component than in the '+' component. Hence it is most likely that the sequence CGCG came from a set of '+' states.

If we choose just one path for our prediction, perhaps the one with the highest probability should be chosen,

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\ P(x, \pi)$$

The most probable path π^* can be found recursively. The probability $v_k(i)$ of the most probable path ending in state k with observation i is know for all the states k. Then

$$v_l(i+1) = e_l(x_{i+1})\max_k(v_k(i)a_{al})$$

The full algorithm is:

Algorithm: Viterbi algorithm

1. Initialization ($i = L$): $v_0(0) = 1,\ v_k(0) = 0$ for $k > 0$
2. Recursion ($i = 1 \ldots L$): $v_l(i) = e_l(x_i)\max_k(v_k(i-1)a_{kl})$
   $\text{ptr}_i(l) = \text{argmax}_k(v_k(i-1)a_{kl})$
3. Termination: $P(x, \pi^*) = \max_k(v_k(L)a_{k0})$
   $\pi^*_L = \text{argmax}_k(v_k(L)a_{k0})$
4. Termination: $\pi^*_{i-1} = \text{ptr}_i(\pi^*_i)$

2.2.2 The forward algorithm

For Markov chain, we can use the probability of a sequence, $P(x)$, to distinguish between CpG islands and other DNA. But for Hidden Markov model, there are many different state path can give the same sequence, we must add the probabilities for all possible paths to obtain the full probability,

$$P(x) = \sum_{\pi} P(x, \pi)$$

The number of possible paths increases exponentially with the length of the sequence, so we need a clever method to solve this problem. That is the forward algorithm.

$$f_k(i) = P(x_1, \ldots, x_i, \pi_i = k),$$

which is the probability of the observed sequence up to and including x_i, requiring that $\pi_i = k$. The recursion equation is

$$f_l(i+1) = e_l(x_{i+1}) \sum_k f_k(i) a_{kl}.$$

The full algorithm is:

Algorithm: Forward algorithm

1. Initialization ($i = L$): $f_0(0) = 1,\ f_k(0) = 0$ for $k > 0$
2. Recursion ($i = 1 \ldots L$): $f_l(i) = e_l(x_i) \sum_k f_k(i-1) a_{kl}$
3. Termination: $P(x) = \sum_k f_k(L) a_{k0}$

2.2.3 The backward algorithm

We might not want to know the most probable path of the observed sequence, but we might interest in the probability that observation x_i come from state k given the observed sequence, $P(\pi_i = k|x)$.

Since $P(\pi_i = k|x) = \frac{P(\pi_i = k, x)}{P(x)}$, we start the probability, $P(\pi_i = k, x)$.

$$\begin{aligned} P(x, \pi_i = k) &= P(x_1, \ldots, x_i, \pi_i = k) P(x_{i+1}, \ldots, x_L | x_1, \ldots, x_i, \pi_i = k) \\ &= P(x_1, \ldots, x_i, \pi_i = k) P(x_{i+1}, \ldots, x_l | \pi_i = k) \end{aligned}$$

the second row following because everything after k only depends on the state at k. The first term in this is $f_k(i)$. The second term is called $b_k(i)$,

$$b_k(i) = P(x_{i+1}, \ldots, x_l | \pi_i = k).$$

It is analogous to the forward variable, but obtained by a backward recursion from the end of the sequence.

Algorithm: Backward algorithm

1. Initialization ($i = L$): $b_k(L) = a_k 0$ for all $k > 0$
2. Recursion ($i = L - 1 \dots 1$): $b_k(i) = \sum_l a_{kl} e_l(x_{i+1}) b_l k(i+1)$
3. Termination: $P(x) = \sum_l a_{0l} e_l(x_1) b_l(1)$

2.3 Parameter estimation for HMMs

2.3.1 Estimation when the state sequence is known

When all paths are known, we can count the number of times each particular transition or emission is used in the set of training sequences. Let these be A_{kl} and $E_k(b)$. And using the maximum likelihood estimators for a_{kl} and $e_k(b)$ are given by

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \quad \text{and} \quad e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')} \tag{2.1}$$

There is a defect for maximum likelihood estimators if there are insufficient data. Then some transitions or emissions do not find out in the insufficient data which might have value zero. To avoid this it is preferable to add predetermined pseudocounts to the A_{kl} and $E_k(b)$.

2.3.2 Estimation when paths are unknown: Baum-Welch

When the paths are unknown for the training sequences, there is no longer direct to estimate parameter value. There is a algorithm that is used most frequently, known as the Baum-Welch algorithm [Baum 1972].

It first estimates the A_{kl} and $E_k(b)$ by considering probable path for the training sequences using the current values of a_{kl} and $e_k(b)$. Then (2.1) is used to derive new values of the as and es. This process is iterated until some stopping criterion is reached.

The Baum-Welch algorithm calculates A_{kl} and $E_k(b)$ as the expected number of times each transition or emission is used, given the training sequence. The probability that a_{kl} is used at position i in sequence x is

$$P(\pi_i = k, \pi_{i+1} = l | x, \theta) = \frac{f_k(i) a_{kl} e_l(x_{i+1}) b_l(i+1)}{P(x)}.$$

Then A_{kl} is calculated by summing over all positions and over all training sequences,

$$A_{kl} = \sum_j \frac{1}{P(x^j)} \sum_i f_k^j(i) a_{kl} e_l(x_{i+1}^j) b_l^j(i+1) \tag{2.2}$$

where $f_k^j(i)$ is the forward variable calculated for the sequence j and $b_l^j(i+1)$ is the backward variable calculated for the sequence j. Similarly, we can find the expected number of times that character b appears in state k,

$$E_k(b) = \sum_j \frac{1}{P(x^j)} \sum_{\{i | x_i^j = b\}} f_k^j(i) b_k^j(i) \tag{2.3}$$

The full algorithm is:

Algorithm: Baum-Welch algorithm

```
1. Initialization: Pick arbitrary model parameters.

2. Recursion: Set all the $A$ and $E$ variables to their pseudocount values $r$ (or
   to zero).
   for each sequence $j = 1 \ldots n$

      Calculate $f_k(i)$ for sequence $j$ using the forward algorithm.

      Calculate $b_k(i)$ for sequence $j$ using the backward algorithm.

      Add the contribution of sequence $j$ to $A$ (2.2) and $E$ (2.3)

   Calculate the new model parameters using (2.1).
   Calculate the new log likelihood of the model.
```

3. Termination: Stop if the change in log likelihood is less than some predefined threshold or the maximum number of iterations is exceeded.

Here the log likelihood of the model is

$$l(x^1, \dots, x^n|\theta) = \log P(x^1, \dots, x^n|\theta) = \sum_{j=1}^{n} \log P(x^j|\theta)$$

Chapter 3

Multiple alignment using HMMs

Since functionally similar biological sequences typically come in families, many methods can help us to identify that a sequence belongs to a family by aligning it to the other members, and therefore allows inferences about its functions. How do we identify such features? A multiple alignment can show how the sequences in a family to relate to each other. You can refer to [3] for detail.

3.1 Pairwise alignment using HMMs

In the previous chapter, we introduced several kinds of methods for HMMs. Now, we will apply those to pairwise alignment.

We required three states; one corresponding to match that is named M, and two states corresponding to insert that are named X and Y, as shown Figure 3.1. We must give probabilities both for emissions of symbols from the states, and for transitions between states. State M has the emission probability distribution p_{ab} for emitting an aligned pair $a : b$. State X has a distribution q_{x_i} for emitting x_i from sequence x against a gap. Similarly, state Y has a distribution q_{y_i}. We denote the transition from M to an insert state (X or Y) by δ, and the probability of staying in an insert state by ε. Let τ be the transition into the End state. These probabilities must satisfy the requirement that the probabilities for all the transitions leaving each state sum to one.

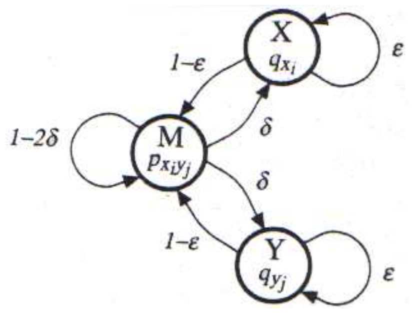


Figure 3.1: *A finite state diagram for the corresponding probabilistic model; Excerpt from [3].*

This model is similar to a hidden Markov model and the difference is that instead of emitting a single sequence it emits a pairwise alignment. When starting in the Begin state, we obey the following two steps: (1) pick the next state according to the distribution of transition probabilities leaving the current state; (2) pick a symbol pair to be added to the alignment according to the emission distribution in the new state.

3.1.1 The most probable path

The essential concept is the same. Since there are three states, we will specify the recurrence equations. $v^M(i,j)$ is that x_i is aligned to y_i, so the value of $v^M(i,j)$ is derived from the maximum value of three states. $v^X(i,j)$ is that x_i is aligned a gap. Since we use affine gap penalties, we do not allow that a gap appears to one sequence and a gap appears to the other sequence at once. Hence $v^X(i,j)$ is derived from the maximum value of two states except state Y. $v^Y(i,j)$ is similar to $v^X(i,j)$.

Algorithm: Viterbi algorithm for pair HMMs

1. Initialization: $v^M(0,0) = 1$. All other $v^\bullet(i,0)$, $v^\bullet(0,j)$.

2. Recurrence: $i = 1, \ldots, n,\ j = 1, \ldots, m$

$$\begin{aligned}
v^M(i,j) = &\ p_{x_i y_j} \max \begin{cases} (1-2\delta-\tau)v^M(i-1,j-1) \\ (1-\varepsilon-\tau)v^X(i-1,j-1) \\ (1-\varepsilon-\tau)v^Y(i-1,j-1) \end{cases} \\
(= &\ \text{emission max (transition} \times \text{ the last one))} \\
v^X(i,j) = &\ q_{x_i} \max \begin{cases} \delta v^M(i-1,j) \\ \varepsilon v^X(i-1,j) \end{cases} \\
v^Y(i,j) = &\ q_{y_j} \max \begin{cases} \delta v^M(i,j-1) \\ \varepsilon v^Y(i,j-1) \end{cases}
\end{aligned}$$

3. Termination: $v^E = \tau \max(v^M(n,m), v^X(n,m), v^Y(n,m))$

3.1.2 The full probability of x and y, summing over all paths

In the previous chapter, we use the forward algorithm to solve the equation, $P(x) = \sum_\pi P(x,\pi)$. We will consider the similar problem,

$$P(x,y) = \sum_{\text{alignments } \pi} P(x,y,\pi).$$

Here, the recurrence relations are similar to the variables of the Viterbi algorithm. The difference is the max converting to the sum.

Algorithm: Forward calculation for pair HMMs

1. Initialization: $f^M(0,0) = 1, f^X(0,0) = 0, f^Y(0,0) = 0.$

   All $f^\bullet(i,-1), f^\bullet(-1,j)$ are set to 0.

   (Why is -1? Because we will calculate $f^\bullet(i,-1)$ and $f^\bullet(-1,j)$)

2. Recursion: $i = 0, \ldots, n$ $j = 0, \ldots, m$ except $(0,0)$ (because (0,0) is calculated in the initialization.)

$$\begin{aligned} f^M(i,j) = &\ p_{x_i y_j} \times [(1-2\delta-\tau) f^M(i-1,j-1) \\ &+(1-\varepsilon-\tau) f^X(i-1,j-1) + (1-\varepsilon-\tau) v^Y(i-1,j-1)] \\ (= &\ \text{emission} \times \textstyle\sum(\text{transition} \times \text{the last one}).) \\ f^X(i,j) = &\ q_{x_i}[\delta f^M(i-1,j) + \varepsilon f^X(i-1,j)] \\ f^Y(i,j) = &\ q_{y_j}[\delta f^M(i,j-1) + \varepsilon f^Y(i,j-1)] \end{aligned}$$

3. Termination: $f^E(n,m) = \tau(f^M(n,m) + f^X(n,m) + f^Y(n,m))$.

3.1.3 The backward algorithm for pair HMMs

The degree of conservations varies depending on structural and functional constraints, so that the core sequences may be well conserved, while loop regions are not reliably alignable. In order to accomplish this goal, we will use the backward algorithm for pair HMMs similar to the backward algorithm in the previous chapter. That is, we will calculate the value of $P(x_i \diamond y_j | x, y)$.

The new notation $x_i \diamond y_j$ means that x_i is aligned to y_j. Then using the conditional probability theory we have

$$\begin{aligned} P(x, y, x_i \diamond y_j) &= P(x_{1\ldots i}, y_{1\ldots j}, x_i \diamond y_j) P(x_{i+1\ldots n}, y_{j+1\ldots m} | \, x_{1\ldots i}, y_{1\ldots j} x_i \diamond y_j) \\ &= P(x_{1\ldots i}, y_{1\ldots j}, x_i \diamond y_j) P(x_{i+1\ldots n}, y_{j+1\ldots m} | \, x_i \diamond y_j) \end{aligned}$$

Then we can use the Bayes' rule to obtain

$$P(x_i \diamond y_j | x, y) = \frac{P(x, y, x_i \diamond y_j)}{P(x,y)},$$

Algorithm: The backward algorithm for pair HMMs

1. Initialization: $b^M(n,m) = b^X(n,m) = f^Y(n,m) = \tau$.

All $b^\bullet(i, m+1)$, $b^\bullet(n+1, j)$ are set to 0.

2. Recurrence: $i = n, \cdots, 1, j = m, \cdots, 1$ except (n, m)

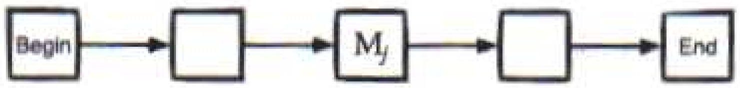
$$\begin{aligned}
b^M(i,j) = & \quad (1-2\delta-\tau)p_{x_{i+1}y_{j+1}}b^M(i+1,j+1) \\
& +\delta(q_{x_{i+1}}b^X(i+1,j) + q_{y_{j+1}}b^Y(i,j+1)) \\
b^X(i,j) = & \quad (1-\delta-\tau)p_{x_{i+1}y_{j+1}}b^M(i+1,j+1) + \varepsilon q_{x_{i+1}}b^X(i+1,j) \\
b^Y(i,j) = & \quad (1-\delta-\tau)p_{x_{i+1}y_{j+1}}b^M(i+1,j+1) + \varepsilon q_{y_{j+1}}b^Y(i,j+1))
\end{aligned}$$

3.2 Profile HMMs for sequence families

It is clear that some positions in the globin alignment are more conserved than others. For example, the helices are more conserved than the loop regions. The profile HMMs can help us to obtain these features.

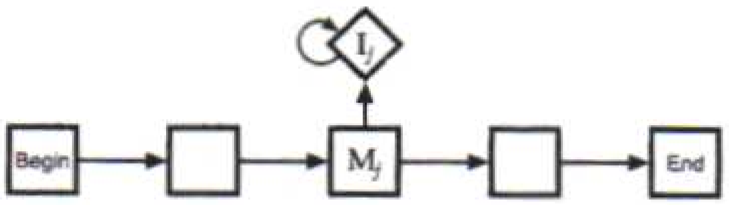
3.2.1 Adding insert and delete states to obtain profile HMMs

Before we start to introduce this model, we want to introduce a 'PSSM'. PSSM is the abbreviation of 'position specific score matrix'. A PSSM is a scoring matrix and depends on the position of alignment, i.e. $S = \sum_{i=1}^{L} \log \frac{e_i(x_i)}{q_{x_i}}$, where $e_i(x_i)$ is the probability of observing amino acid x_i in position i and q_{x_i} is the probability of generating x_i randomly. A PSSM captures some conservation information, but it is not sufficient to represent all of the information in a multiple alignment of a protein family. So we will modify the PSSM. We will call a series of identical states that using the PSSM as *match* states.



Now we will deal with gaps. First we consider the insert states. Portions of the whole alignment do not match anything in the model. Hence we need another state to specify that part, i.e. this state is called insert state I_i, where I_i

will be used to match insertions after the residue matching the ith column of the multiple alignment. The I_i have emission distribution $e_{I_i}(a)$. We need transitions from M_i to I_i, a loop transition from I_i to itself, to allow multi-residue insertions, and a transition back from I_i to M_{i+1}. We denote insert state in our graph by diamonds.



Next step we will handle the deletions. Segments of the multiple alignment that are not matched by any residue. Deletions could be handled by forward 'jump' transitions between non-neighboring match states. The deletion states do not emit any residue.

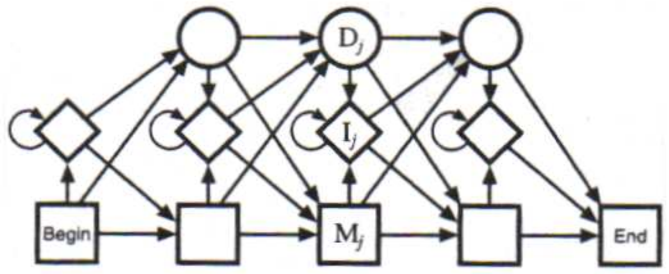


Figure 3.2: *The transition structure of a profile HMM. We use diamonds to indicate the insert states and circles for the delete states; Excerpt from [3].*

3.2.2 Basic profile HMM parameterization

The aim of the parameterization processes it to make the distribution maximum around members of the family. Assuming that the emission and transition probabilities are nonzero, a profile HMM can model any possible sequence of residues from the given alphabet.

The choice of length of the model corresponds more precisely to a decision on which multiple alignment columns to assign to match states, and which to assign to insert states. A simple rule to decide which columns should correspond to match states or insert states, i.e. that column are more than half gap characters should be modeled by inserts.

A problem is how to assign the probability parameters. When the alignment is given, we just count up the times of each transition or emission and assign probabilities according to

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \quad \text{and} \quad e_k(a) = \frac{E_k(a)}{\sum_{a'} E_k(a')}$$

where k and l are indices over states, and a_{kl} and $e_k(a)$ are the transition and emission probabilities, and A_{kl} and E_k are the corresponding frequencies.

In the few training sequences, a major difficult is some transition probabilities or emission probabilities that are not seen in training alignment. To avoid zero probabilities, we will add pseudo counts to the observed frequencies. The simplest pseudo count method is Laplace's rule that to add one to each frequency.

3.2.3 Searching with profile HMMs

One of the main purposes of developing profile HMMs is to use them to find out potential membership in a family by obtaining important matches of a sequence to the profile HMM. We either use the Viterbi equation to give the most probable alignment π^* of a sequence x together with its probability $P(x, \pi^*|M)$, or the forward equations to calculate the full probability of x summed over all possible paths $P(x|M)$.

Viterbi equations

We will consider the log-odds ratio of the resulting probability to the probability of x given our standard random model because we can avoid problems of underflow when working with raw probability. Before introducing the algorithm, we need to introduce the notations.$V_j^M(i)$ is the log-odds score of the best path ending with x_i being emitted by state M_j. $V_j^I(i)$ is the score of the best path ending in x_i being emitted by state I_j. $V_j^D(i)$ is similar to $V_j^I(i)$. Then the recurrence relations are:

$$V_j^M(i) = \log \underbrace{\frac{e_{M_j}(x_i)}{q_{x_i}}}_{q_{x_i}\text{:random}} + \max \begin{cases} V_{j-1}^M(i-1) + \log a_{M_{j-1}M_j} \\ V_{j-1}^I(i-1) + \log a_{I_{j-1}M_j} \\ V_{j-1}^D(i-1) + \log a_{D_{j-1}M_j} \end{cases}$$

$$(= \text{ emission} + \max(\text{the last one} + \text{transition}))$$

$$V_j^M(i) = \log \frac{e_{M_j}(x_i)}{q_{x_i}} + \max \begin{cases} V_j^M(i-1) + \log a_{M_jI_j} \\ V_j^I(i-1) + \log a_{I_jI_j} \\ V_j^D(i-1) + \log a_{D_jI_j} \end{cases}$$

(it is similar to state X in pair HMM that i of $V_j^M(i)$ is similar to i of $X(i,j)$)

$$V_j^M(i) = \max \begin{cases} V_{j-1}^M(i) + \log a_{M_{j-1}D_j} \\ V_{j-1}^I(i) + \log a_{I_{j-1}D_j} \\ V_{j-1}^D(i) + \log a_{D_{j-1}D_j} \end{cases}$$

In a typical case, $e_{M_j}(x_i) = a_{ID} = a_{DI} = 0$.

Since the beginning or end of sequence doesn't match the first or the last match state of the model, we could rename the Begin state as M_0 and set $V_0^M(0) = 0$. Similarly, we can collect possible paths ending in insert or delete states by renaming the End state to M_{L+1} and using above relation to calculate $V_{L+1}^M(n)$ as the final score.

Forward algorithm

We define variables $F_j^M(i)$, $F_j^I(i)$ and $F_j^D(i)$ for the partial full log-odds ratios, corresponding to $V_j^M(i)$, $V_j^I(i)$ and $V_j^D(i)$. Then we can write:

$$\begin{aligned}
F_j^M(i) = {} & \log \frac{e_{M_j}(x_i)}{q_{x_i}} + \log[a_{M_{j-1}M_j} \exp(F_{j-1}^M(i-1)) \\
& +a_{I_{j-1}M_j} \exp(F_{j-1}^I(i-1)) + a_{D_{j-1}M_j} \exp(F_{j-1}^D(i-1))] \\
F_j^I(i) = {} & \log \frac{e_{I_j}(x_i)}{q_{x_i}} + \log[a_{M_jI_j} \exp(F_j^M(i-1)) + \\
& a_{I_jI_j} \exp(F_j^I(i-1)) + a_{D_jI_j} \exp(F_j^D(i-1))] \\
F_j^D(i) = {} & \log[a_{M_{j-1}D_j} \exp(F_{j-1}^M(i)) + a_{I_{j-1}D_j} \exp(F_{j-1}^I(i)) \\
& +a_{D_{j-1}D_j} \exp(F_{j-1}^D(i))]
\end{aligned}$$

Initialization and termination conditions are handled as for the Viterbi case, with $F_0^M(0)$ setting to 0. And $\log(e^x + e^y)$ can be performed efficiently in a practical implementation. Assume we want to calculate $\tilde{r} = \log(p+q)$ from the log of the probabilities, $\tilde{p} = \log p$ and $\tilde{q} = \log q$. The direct way is to do $\tilde{r} = \log(\exp(\tilde{p}) + \exp(\tilde{q}))$. By pulling out $\tilde{p}$, we can write this as

$$\tilde{r} = \tilde{p} + \log(1 + \exp(\tilde{q} - \tilde{p})).$$

It is possible to approximate the function $\log(1 + \exp(x))$ by interpolation from a table. For a reasonable level of accuracy, the table can actually by quite small, assuming we always pull out the largest of $\tilde{p}$ and $\tilde{q}$, because $\exp(\tilde{q} - \tilde{p})$ rapidly approaches zero for large $(\tilde{p} - \tilde{q})$.

3.2.4 Multiple alignment by profile HMM training

The goal of this section will introduce how to estimate the parameters in this model and when we got new sequence how to use the model to align it. We use simplest sentence to explain these as follows:

1. The model is initialized with estimates of transition probabilities and amino acid composition for each match and insert state.

2. All possible paths through the model for generating each sequence in turn are examined. This calculation provides a probability of the sequence, given

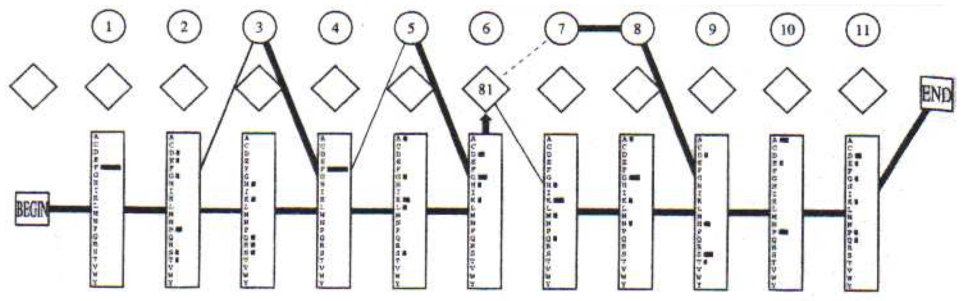
all possible paths through the model, and, from this value, the probability of any particular path may be found. Another algorithm, the Baum-Welch algorithm, then counts the number of times a particular state-to-state transition is used and a particular amino acid is required by a particular match state to generate the corresponding sequence position.

3. A new version of the HMM is produced that uses the results found in step 2 to generate new transition probabilities and match-insert state compositions.

4. Step 3 and 4 are repeated up to 10 more times until the parameters do not change significantly.

5. The trained model is used to provide the most likely path for each sequence. The Viterbi algorithm is used for this purpose.

6. The HMM may be used to search a sequence database for additional sequences that share the same sequence variation. Hence the sum of the probabilities of all possible sequence alignments to the model is obtained. These probabilities are calculated by the forward algorithm. This analysis gives a type of distance score of the sequence from the model, thus providing an indication of how well a new sequence fits the model and whether the sequence may be related to the sequences used to train the model.



```
FPHF-DLS-----HGSAQ
FESFGDLSTPDAVMGNPK
FDRFKHLKTEAEMKASED
FTQFAG-KDLESIKGTAP
FPKFKGLTTADQLKKSAD
FS-FLK-GTSEVPQNNPE
FG-FSG----AS---DPG
```

Figure 3.3: *A model (top) estimated form an alignment (bottom). The residues in the shaded area of the alignment were treated as inserts. Emission probabilities are shown as bars opposite the different amino acids for each match state and transition probabilities are indicated by the thickness of the lines. The $I \rightarrow I$ transition probabilities times 100 are shown in the insert states; Excerpt from [3].*

Chapter 4

Phylogenetic trees

4.1 Background on the phylogenetic trees

Our materials come from these [3], [11] and [20]. The similarities of molecular mechanisms of the organisms have been strongly suggested that all organisms on Earth had a common ancestor. A phylogenetic analysis of a family of related nucleic acid or protein sequences is a determination of how the family members might have been derived during evolution. The evolutionary relationships among the sequences are described by using a graph called a tree.

The tree is assumed to be binary here and has branches, just as in plants, with the outer branches representing the currently existed sequences and the inner branches representing common ancestor sequences. We use the general term 'length' or 'branch length' here, and represents this by the lengths of branches. The branches are joined through nodes that represent relationship among current and ancestor sequences. Finally, we will reach a main branch with a root. More often, however, the phylogenetic tree is left without a root because we do not care which species is the ultimate ancestor.

The goal of a phelogenetic analysis of nucleic acid or protein sequences is to analyze the relationships among a group of sequences that can be aligned into a multiple sequence alignment. Three methods-distance methods, maximum parsimony, and maximum likelihood-are used for predicting such trees. We will concentrate these in the next section.

4.2 Three kinds of phylogenetic trees

4.2.1 Distance methods

Distance methods are based on genetic distances between sequence pairs in a multiple sequence alignment. The genetic distance between two sequences is the fraction of aligned positions in which the sequence has been changed. Sequence pairs that have the smallest distances are 'neighbors'. On a tree, these sequences share a node or common ancestor position and are each joined to that node by a branch.

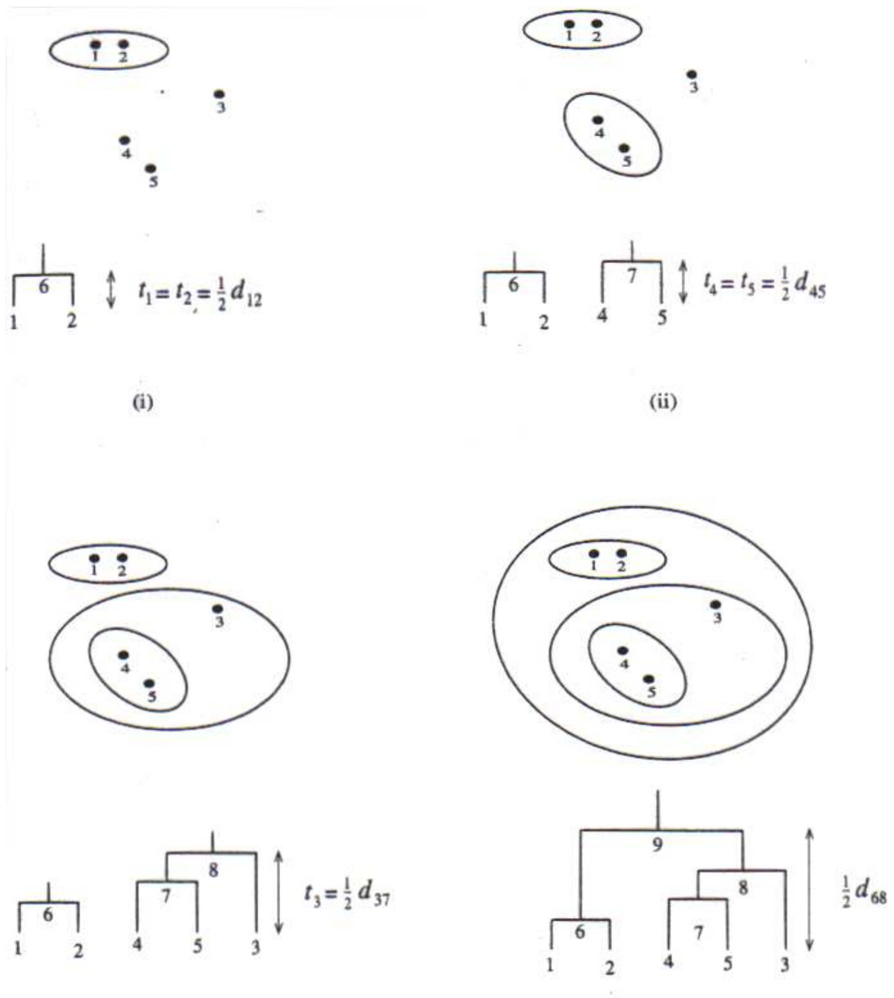
We begin with a clustering procedure called UPGMA. It works by clustering the sequences, at each stage combining two clusters and at the same time creating a new node on a tree. The tree can be imagined as being collected upwards, each node being added above the others, and the edge lengths being determined by the difference in the heights of the nodes at the top and bottom of an edge.

We defined the distance d_{ij} between two clusters C_i and C_j to be the average distance between pairs of sequences from each cluster:

$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{p \text{ in } C_i, q \text{ in } C_j} d_{pq}$$

where $|C_i$ and $|C_j|$ denote the number of sequences in the clusters i and j, respectively. If C_k is the union of the two clusters C_i and C_j, i.e., $C_k = C_i \cup C_j$, and if C_l is any other cluster, then:

$$d_{kl} = \frac{d_{il}|C_i| + d_{jl}|C_j|}{|C_i| + |C_j|} \tag{4.1}$$



1
2
3
4
5
6
$t_1 = t_2 = \frac{1}{2} d_{12}$
(i)
7
$t_4 = t_5 = \frac{1}{2} d_{45}$
(ii)
8
$t_3 = \frac{1}{2} d_{37}$
(iii)
9
$\frac{1}{2} d_{68}$
(iv)

Figure 4.1: *A example of hoe UPGMA produces a rooted tree by successively clustering sequences, in this case a set of five sequence whose distances can be represented by points in the plane; Excerpt from [3].*

Algorithm: UPGMA

1. Initialization:

   Assign each sequence i to its own cluster C_i,

   Define on leaf of T for each sequence, and place at height zero.

2. Iteration:

   Determine the two clusters i, j for which $d_{ij}$is minimal. (If there are several equidistant minimal pairs, pick one randomly.)

   Define a new cluster k by $C_k = C_i \cup C_j$, and define d_{kl} for all l by (4.1).

   Define a node k with daughter nodes i and j, and place it at height $d_{ij}/2$.

   Add k to the current clusters and remove i and j.

3. Termination: When only two clusters i, j remain, place the root at height $d_{ij}/2$.

4.2.2 Maximum parsimony

Maximum parsimony is one of the most widely used of all tree building algorithms. It works by finding the tree of the observed sequences with a minimal number of substitutions. Instead of building a tree, it assigns a cost to a given tree and it is necessary to search through all topologies in order to identify the 'best' tree.

Parsimony treats each site independently. Hence we calculate the minimal number of substitutions column by column. And then we sum the substitutions for all columns. Given a example, there are four aligned nucleotide sequences:

x_1: AAG

x_2: AAA

x_3: GGA

x_4: AGA

There are three topologies of four sequences. The three topologies of the first column of the multiple sequence alignment are shown as Figure 4.2. We can see that the cost of the tree topologies of the first column is 1. By the similar procedure, the cost of the tree topologies of the second column and third column are shown as Figure 4.3 and Figure 4.4, respectively. And then we sum the cost of all columns of the three topologies, respectively. We find out the maximal parsimony of this multiple sequence alignment that is the first topology tree.

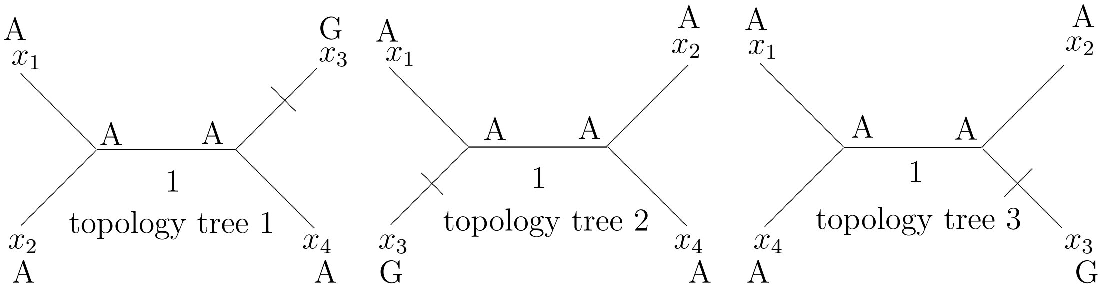


Figure 4.2: *The three topologies of the first column of the multiple sequence alignment*

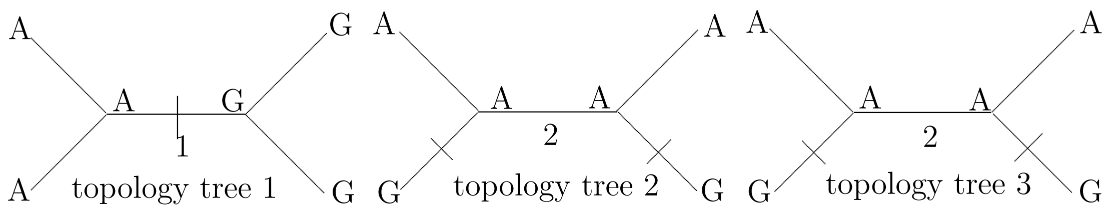


Figure 4.3: *The three topologies of the second column of the multiple sequence alignment*

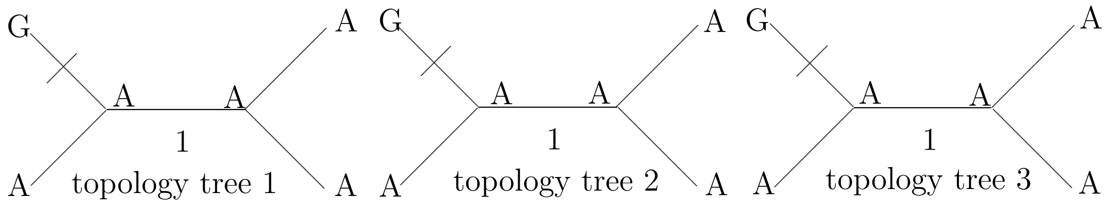


Figure 4.4: *The three topologies of the third column of the multiple sequence alignment*

Fitch's algorithm-step 1

1. Do a post-order (from leaves to root) traversal of tree.

2. Initialization: Set $C = 0$ and $k = 2n - 1$.

3. Recursion: To obtain the set R_k

   If k is leaf node:

      Set $R_k = x_u^k$ (x_u^k is the character of the column u of the node k)

   If k is not a leaf node:

      Compute R_i, R_j for the daughter nodes i, j of k,

      and set $R_k = R_i \cap R_j$ if this intersection is no empty,

      or else set $R_k = R_i \cup R_j$ and increment C.

4. Minimal cost of tree $= C$.

Fitch's algorithm-step 2

1. Do a pre-order (from root to leaves) traversal of tree.

2. Select state r_j of internal node j with parent i

   Set $r_j = r_i$ if $r_i \in R_j$,

   or else set $r_j =$ *arbitrary state* $\in R_j$.

The weighted parsimony does not just count the number of substitutions but adds costs $S(a, b)$ for each substitution of a by b. The aim is now to minimize this cost. When set $S(a, a)$ to 0 for all a and set $S(a, b)$ to 1 for all $a \neq b$. Let $S_k(a)$ denoted the minimal cost of the assignment of a to node k.

Weighted parsimony-step 1

1. Do a post-order (from leaves to root) traversal of tree.

2. Initialization: Set $k = 2n - 1$, the number of the root node.

3. Recursion: Compute $S_k(a)$ for all a as follows:

   If k is leaf node:

   Set $S_k(a) = 0$ for $a = x_u^k$, $S_k(a) = \infty$, otherwise.

   If k is not a leaf node:

   Compute $S_i(a)$, $S_j(a)$ for all a at the daughter nodes i, j, and define $S_k(a) = \min_b(S_i(a) + S(a,b)) + \min_b(S_j(a) + S(a,b))$.

4. Termination: Minimal cost of tree $= \min_a S_{2n-1}(a)$.

Weighted parsimony-step 2

1. Do a pre-order (from root to leaves) traversal of tree.
2. Select minimal cost character for root.
3. For each internal node i, select character that produced minimal cost at parent k.

4.2.3 Maximum likelihood

Parsimony can give misleading information. The rates of sequence change vary in the different branches of a tree that are represented by the sequence data shown in the left of Figure 4.5. These variations produce a range of branch lengths, with long ones representing more extended periods of time and short ones representing short times. Because in parsimony analysis rates of change along all branches of the tree are assumed to be equal, the tree predicted by parsimony and shown in the right of Figure 4.5 will not be correct. However the maximum likelihood can avoid this problem.

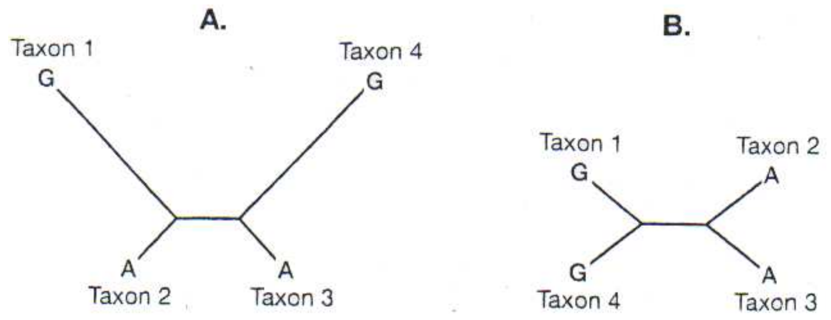


Figure 4.5: *Type of sequence variation that leads to an incorrect prediction by the maximum parsimony method; Excerpt from [3].*

Let us look at possible forms for the substitution probabilities $P(b|a,t)$, for a residue a having being substituted by a residue b over a branch length t. Given a residue alphabet of size K, we can write these as a $K \times K$ matrix that depends on t, and which we denote by $S(t)$:

$$S(t) = \begin{pmatrix} P(A_1|A_1,t) & P(A_2|A_1,t) & \dots & P(A_K|A_1,t) \\ P(A_1|A_2,t) & P(A_2|A_2,t) & \dots & P(A_K|A_2,t) \\ \dots & \dots & \dots & \dots \\ P(A_1|A_K,t) & P(A_2|A_K,t) & \dots & P(A_K|A_K,t) \end{pmatrix}$$

We assume the matrix is multiplicative in the sense that

$$S(t)S(s) = S(t+s)$$

for any tie lengths s or t. This is equivalent to saying that the substitution probabilities satisfy

$$\sum_b P(a|b,t)P(b|c,s) = P(a|c,t+s)$$

for all a, c, s, and t.

One of models for evolutionary mutations is Jukes & Cantor [1969]. Jukes-Cantor assumes equal rate of change:

R=

	A	C	G	T
A	-3α	α	α	α
C	α	-3α	α	α
G	α	α	-3α	α
T	α	α	α	-3α

For a short time period ε, we write:

$$S(\varepsilon) \cong I + R\varepsilon = \begin{pmatrix} 1-3\alpha & \alpha & \alpha & \alpha \\ \alpha & 1-3\alpha & \alpha & \alpha \\ \alpha & \alpha & 1-3\alpha & \alpha \\ \alpha & \alpha & \alpha & 1-3\alpha \end{pmatrix}$$

where I is the identity matrix with ones down the diagonal and zero elsewhere. By multiplicatively,

$$S(t+s) = S(t)S(s) \cong S(t)(I + R\varepsilon).$$

Hence

$$(S(t+s) - S(t))/\varepsilon \cong S(t)R.$$

Leading to the linear differential equation:

$$S'(t) \cong S(t)R$$

We give $S(t)$ the following form:

$$S(t) = \begin{pmatrix} r_t & s_t & s_t & s_t \\ s_t & r_t & s_t & s_t \\ s_t & s_t & r_t & s_t \\ s_t & s_t & s_t & r_t \end{pmatrix}$$

With the additional condition that the limit as t goes to infinity:

$$r_t = s_t = \frac{1}{4}$$

Hence we get the equations

$$\begin{aligned} \dot{r} &= -3\alpha r + 3\alpha s, \\ \dot{s} &= -\alpha s + \alpha r, \end{aligned}$$

and yield the unique solution which is known as the Jukes-Cantor model:

$$\begin{aligned} r_t &= \tfrac{1}{4}(1 + 3e^{-4\alpha t}), \\ s_t &= \tfrac{1}{4}(1 - e^{-4\alpha t}). \end{aligned}$$

The Jukes-Cantor model does not capture some important features of nucleotide substitution. For instance, transitions are more common than transversions. The transitions (between purine) are $A \leftrightarrow G$, $C \leftrightarrow T$. and the transversions (between pyrimidine) are $A \leftrightarrow T$, $A \leftrightarrow C$, $G \leftrightarrow T$, $G \leftrightarrow C$. To obtain these features, Kimura [1980] proposed a model with the rate matrix

R=

	A	C	G	T
A	$-2\beta-\alpha$	β	α	β
C	β	$-2\beta-\alpha$	β	α
G	α	β	$-2\beta-\alpha$	β
T	β	α	β	$-2\beta-\alpha$

By the similar procedure, we get

$$S(t) = \begin{pmatrix} r_t & s_t & u_t & s_t \\ s_t & r_t & s_t & u_t \\ u_t & s_t & r_t & s_t \\ s_t & u_t & s_t & r_t \end{pmatrix}$$

where

$$\begin{aligned} s_t &= \tfrac{1}{4}(1 - e^{-4\beta t}), \\ u_t &= \tfrac{1}{4}(1 + e^{-4\beta t} - 2e^{-2(\alpha+\beta)t}), \\ r_t &= 1 - 2s_t - u_t. \end{aligned}$$

Just like the parsimony method, we only need to search over unrooted tree topologies. Hence, two assumptions suffice which are reversibility and lacking of memory. The reversibility is

$$P(b|a,t)P(a) = P(a|b,t)P(b)$$

, where $P(a)$ denotes the probability of a occurring at the root of the tree, for all a, b, and t. The lacking of memory is

$$P(c|b, t_i + t_j) = \sum_a P(c|a, t_j)P(a|b, t_i).$$

Given the tree topology T and branch lengths t (we write $t_1 \ldots$ compactly as t), we can compute the probability of T with a specific set of ancestors assigned to its nodes by multiplying all the evolutionary probabilities, one for each branch of the tree. For example, for the tree shown in Figure 4.6 the probability would be

$$P(x^1, \ldots, x^5|T,t) = P(x^5)P(x^4|x^5, t_4)P(x^3|x^5, t_3)P(x^1|x^4, t_1)P(x^2|x^4, t_2).$$

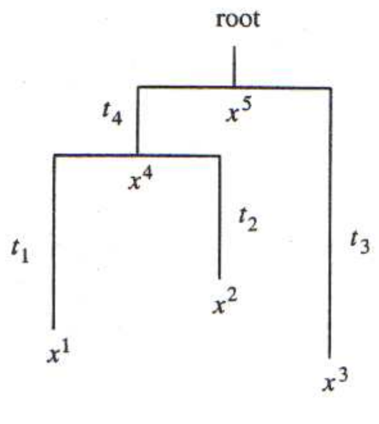


Figure 4.6: *An example of a tree with three sequences; Excerpt from [3].*

We are interested in the probability of observed sequences given tree and branch lengths, we sum over all the possible ancestors.

$$P(x^1, \ldots, x^3|T,t) = \sum_{x^4, x^5} P(x^1, \ldots, x^5|T,t).$$

We make some basic simplifying assumptions that every site of the given data sequences can be treated as independent and that deletions and insertions do not occur. Then this assumption implies

$$\begin{aligned} P(x^1,\ldots,x^3|T,t) &= \prod_u P(x_u^1,\ldots,x_u^3|T,t) \\ &= \prod_u (\sum_{x_u^4,x_u^5} P(x_u^1,\ldots,x_u^5|T,t)), \end{aligned}$$

where u indexes columns in the alignment. There are many possible candidates which we must to sum over. This can be done efficiently using a tree upward traversal pass.

Let $P(L_k|a)$ denote the probability of all leaves below node k given that the residue at k is a. Then we calculate $P(L_k|a)$ form the probabilities $P(L_i|b)$ and $P(L_j|c)$ for all b and c, where i and j are the daughter nodes of k. (Figure 4.7)

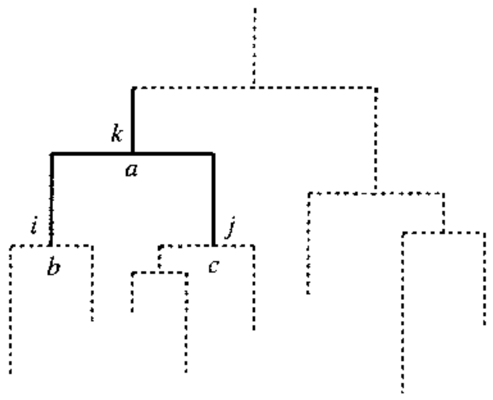


Figure 4.7: *Labelling at a branch in a tree; Excerpt from [3].*

Algorithm: Felsenstein's algorithm for likelihood

1. Initialization: Set $k = 2n - 1$.

2. Recurrence: Compute $P(L_k|a)$ for all a as follows:

    If k is a leaf node:

        Set $P(L_k|a) = 1$ if $a = x_u^k$, $P(L_k|a) = 0$ if $a \neq x_u^k$.

If k is not a leaf node:

Compute $P(L_i|b)$ and $P(L_j|c)$ for all a at the daughter nodes i, j, and set $P(L_k|a) = \sum_{b,c} P(b|a, t_i)P(L_i|b)P(c|a, t_j)P(L_j|c)$

3. Termination: Likelihood at column $u = P(x_u^\bullet|T, t_\bullet) = \sum_a P(L_{2n-1}|a)P(a)$

where we write $x_u^1, \ldots$ compactly as $x_u^\bullet$. Then we compare all values for all possible topology trees and all branch lengths and find out the maximum value of these values.

Chapter 5

Multiple sequence alignment and evolutionary HMM

Here we follow Holmes and Bruno's paper [12]. They did multiple sequences alignment under a given tree. Most sequence profiling tools which use sequence weighting to correct for phylogenetic bias in the training set are a shortcut compared to a full phylogenetic model and it may miss potentially important clues on the sequence family. The phylogenetic context of mutation events is significant in molecular evolution while an lack of mutation on short branches tell virtually no information (Figure 5.1).

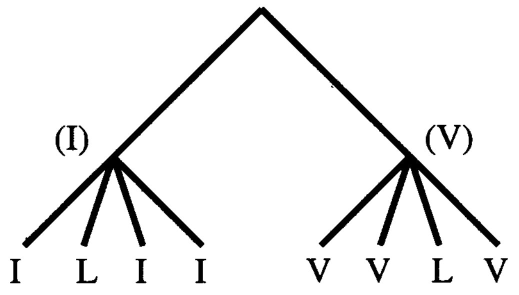


Figure 5.1: *The phylogenetic context of mutation events is significant in molecular evolution while an lack of mutation on short branches tell virtually no information; Excerpt from [12].*

As an improvement on weighted training, one may consider evolutionary models of biological sequences, such as profile hidden Markov models. Evolutionary models give a joint distribution for all the sequences in a family at once, conditioned on the tree that relates them. Thus correlations between related sequences are built into the model. Holmes and Bruno did multiple sequences alignment that allowed gaps.

5.1 The links model

A more gap-sensitive model was proposed by Thorne *et al.*(1991, 1992) [24] and [25]. Their 'links' model is a birth-death process with immigration, another canonical stochastic system. We simplest introduce the concept of the birth-death process and you can refer to [6], [10],[15] and [21] for detail.

A birth-death process is a special type of Markov process. As the name implies, birth-death processes were originally used to describe populations that were increased by births and decreased by deaths. The birth-death processes usually arise when there is a group of entities that are increased by births or arrivals and decreased by deaths or departures. The birth-death processes are a powerful tool to analyze queues. The birth-death process is a process in which changes of state are only to adjacent states (Figure 5.2), i.e. the state space will be $\{0, 1, 2, \ldots\}$, and changes of state will always be from n to $n+1$ or n to $n-1$.

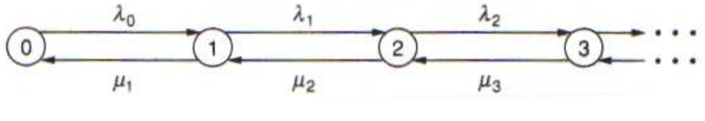


Figure 5.2: *State diagram for a birth-death process; Excerpt from [6]*

To describe the chain, we give birth rates λ_n, $n = 0, 1, 2, \ldots$ and death rates μ_n, $n = 1, 2, 3, \ldots$. If the population is currently n, then new individuals arrive at rate λ_n and individuals leave at rate μ_n (note if the population is 0 there can be no deaths, so $\mu_0 = 0$). If we let X_t denote the state of the chain at time t, then

$$\begin{aligned} P\{X_{t+\Delta t} = n|X_t = n\} = & \ 1 - (\mu_n + \lambda_n)\Delta t + o(\Delta t), \\ P\{X_{t+\Delta t} = n|X_t = n-1\} = & \ \lambda_{n-1}\Delta t + o(\Delta t), \\ P\{X_{t+\Delta t} = n|X_t = n+1\} = & \ \mu_{n+1}\Delta t + o(\Delta t). \end{aligned}$$

where $o(\Delta t)$ represents some function that is much smaller than Δt for Δt small, i.e.,

$$\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0.$$

Let $P_n(t) = P\{X_t = n\}$.

$$\begin{aligned} P\{X_{t+\Delta t} = n\} = & \ P\{X_t = n\}P\{X_{t+\Delta t} = n|X_t = n\} \\ & + P\{X_t = n-1\}P\{X_{t+\Delta t} = n|X_t = n-1\} \\ & + P\{X_t = n+1\}P\{X_{t+\Delta t} = n|X_t = n+1\}. \end{aligned}$$

And $\dot{P}_n(t) = \lim_{\Delta t \to 0} \frac{1}{\Delta t}(P\{X_{t+\Delta t} = n\} - P\{X_t = n\})$. Therefore,

$$\dot{P}_n(t) = \mu_{n+1}P_{n+1}(t) + \lambda_{n-1}P_{n-1}(t) - (\mu_n + \lambda_n)P_n(t).$$

with initial conditions,

$$\begin{aligned} & P_1(t = 0) = 1 \\ & P_n(t = 0) = 0 \text{ for } n > 1 \end{aligned}$$

Return to our subject. At any instant, a single residue many spawn a new child or it may die; the former (birth) event happens with rate λ, the latter (death) with rate μ. Child residues are inserted adjacent to the parent residue on the right-hand side. New residues are also injected into the sequence at the left-hand end of the sequence with rate λ (the immigration of the classical process; Thorne *et al.* ascribe this to an 'immortal link').

5.1.1 Model

Let us repeat the links model by Thorne *et al.*(1991, 1992) [24], [25]. Consider an individual residue. Let $p_n(t)$ be the probability that, at time t, it has survived, spawning n descendants (including itself, its children, its grandchildren and so on). Since the insertion rate is $\lambda = \frac{1}{n}\lambda_n$ and the deletion rate is $\mu = \frac{1}{n}\,\mu_n$, the time-evolution of $p_n(t)$ is described by

$$\dot{p}_n = \lambda(n-1)p_{n-1} + \mu(n+1)p_{n+1} - (\mu+\lambda)np_n.$$

with $p_1(t=0) = 1,\ p_n(t=0) = 0$ for $n > 1$, and taking $p_n(t) = 0$ for $n \leq 0$ at all t.

The other eventuality is that, by time t, the residue has died leaving n descendants. Call the probability of this event, $q_n(t)$. It evolves as follows:

$$\dot{q}_n = \begin{cases} \lambda(n-1)q_{n-1} + \mu(n+1)q_{n+1} + \mu p_{n+1} - (\lambda+\mu)nq_n & \text{for } n > 0 \\ \mu(q_1 + p_1) & \text{for } n = 0 \end{cases}$$

with $q_n(t=0) = 0$ for all n.

We must also consider descendants of the immortal link at the left-hand end of the sequence. Let $r_n(t)$ be the probability that there are n such residues at time t, then

$$\dot{r}_n = \begin{cases} \lambda n r_{n-1} + \mu(n+1)r_{n+1} - \lambda(n+1)r_n + \mu n r_n & \text{for } n > 0 \\ \mu r_1 - \lambda r_0 & \text{for } n = 0 \end{cases}$$

The solutions to the above equations are

$$\begin{aligned} p_n = {} & \alpha\beta^{n-1}(1-\beta) & \\ q_n = {} & (1-\alpha)(1-\gamma) & \text{for } n = 0 \\ = {} & (1-\alpha)\gamma\beta^{n-1}(1-\beta) & \text{for } n > 0 \\ r_n = {} & \beta^n(1-\beta) & \end{aligned}$$

where

$$\begin{aligned} \alpha(t) &= e^{-\mu t} \\ \beta(t) &= \frac{\lambda(1-e^{(\lambda-\mu)t})}{\mu - \lambda e^{(\lambda-\mu)t}} \\ \gamma(t) &= 1 - \frac{\mu(1-e^{(\lambda-\mu)t})}{(1-e^{-\mu t})(\mu - \lambda e^{(\lambda-\mu)t})} \end{aligned} \tag{5.1}$$

Conceptually, α is the probability of ancestral residue survival, β is the probability of more insertions given one or more extant descendants and γ is the probability of insertions given that the ancestral residue did not survive.

5.2 The link model as a pair HMM

In the 2.1 section, we introduced what is a pair HMM. Here, we will combine the link model and a pair HMM. We will introduce 'null' (non-emitting) states to simplify the model. The pair HMM for the link model is shown in Figure 5.3 (the tree on which this model is based is shown in Figure 5.4). The central recurrent loop of this model uses all three types of state and describes the fate of an individual ancestral residue. Either the residue lives (*math* state) or dies (*delete* state). In each case it spawns a geometrically distributed number of ancestor residues (*insert* state) although the geometric distribution is subtly altered if the ancestor dies (the *match* $\rightarrow$ *insert* and *delete* $\rightarrow$ *insert* transitions have different probabilities).

Note that a rough examination of Figure 5.3 reveals nod direct *delete* $\rightarrow$ *delete* transition. This is because a deleted ancestral link may have given birth to orphaned descendant links. If transitions via null states are considered, however, there is a direct self-transition from the *delete* state with probability $\frac{\lambda}{\mu}(1-\gamma)(1-\alpha)$.

Inference of the alignment of the two sequences, π, employs dynamic programming. (Recall that π describes the evolutionary relationship between the sequences and π is taken to be the path through the Markov model). The optimal value of π may be obtained using the Viterbi algorithm; alternatively, the Forward algorithm can be used to calculate the sum-over-alignments likelihood $P(D|A) = \sum_{\pi} P(D|A,\pi)$ (A for ancestor and D for descendant) or to sample an alignment from the posterior distribution $P(\pi, d|A)$. These algorithms are described in previous sections.

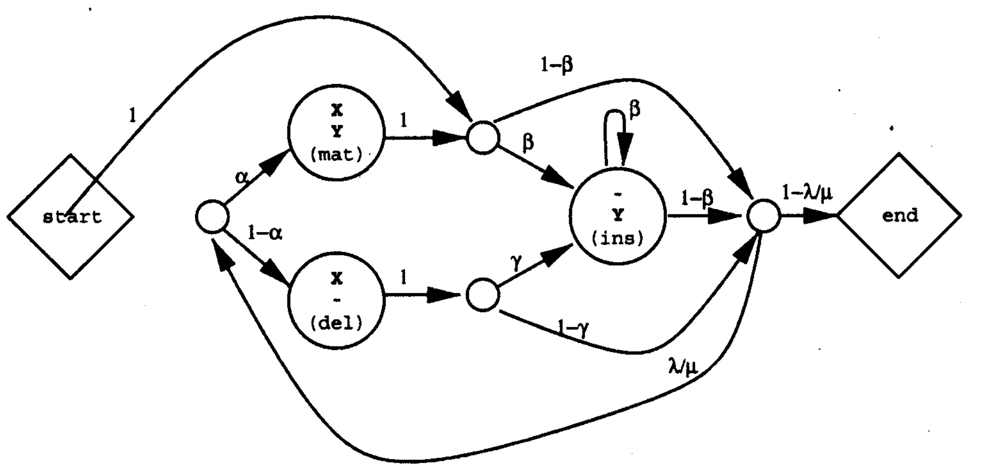


Figure 5.3: *The pair HMM for the links model on the single-branch tree shown in Figure 5.4. Null states are shown as small circles. The parameters α, β and γ are related to the branch length t as described in equation (5.1); Excerpt from [12].*

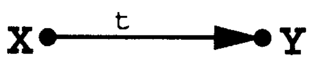


Figure 5.4: *A single-branch tree for an ancestor (X) and a descendant (Y). The branch length is t; Excerpt from [12].*

5.2.1 From pair HMMs to multiple HMMs

A multiple alignment is a composition of pairwise branch alignments. Given an evolutionary tree relating N sequences (including the sequences at internal tree nodes), one can construct a composite multiple HMM analogous to the pair HMM shown in Figure 5.3 by considering the ancestor-descendant relationship of every branch. For example, the three-node tree in Figure 5.5 has the HMM shown in Figure 5.6, and the four-node tree in Figure 5.7 has the HMM shown in Figure 5.8. Note how the HMM structure of Figure 5.3 is inset in Figure 5.6; likewise, Figure 5.6 is inset in Figure 5.8.

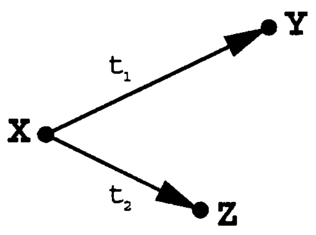


Figure 5.5: *A simple binary tree. The root node is X and the two children are Y and Z; Excerpt from [12].*

It is useful to image composite multiple HMMs and their relationship to the factorization of equation

$$P(\{\pi\}, S|T, \Theta) = P(\text{root}|\Theta) \prod_b P(\pi_b, D_b|A_b, t_b, \Theta) \tag{5.2}$$

where S is a set included all sequences at internal nodes of the tree. There is an algorithm to construct a composite multiple HMM for any evolutionary pair HMM and tree T, such that the likelihood function for this multiple HMM is equal to $P(\{\pi\}, S|T, \Theta)$ of equation (5.2). The multiple alignment represents the complete evolutionary history of all the sequence, whereas the pairwise alignments represent the individual historical accounts of each branch.

By constraining the pairwise alignments along subsets of all tree branches (and the inferred sequences at subsets of all tree nodes), Holmes and Bruno provide a *Gibbs sampler* for the likelihood function described in equation (5.2). Progressive alignment and refinement algorithms are obtained by replacing the construct 'sample an alignment π using the Forward algorithm' with 'find the optimal alignment π using the Viterbi algorithm' in the Gibbs sampler code.

For their alignment algorithm, they need a well-defined and decided way of decomposing a multiple alignment into a set of pairwise branch alignments for neighboring nodes. More generally, it is useful to obtain the pairwise alignment of any two nodes of the tree not just neighboring nodes. Conversely, they need a way of composing a multiple alignment from a complete set of pairwise alignments.

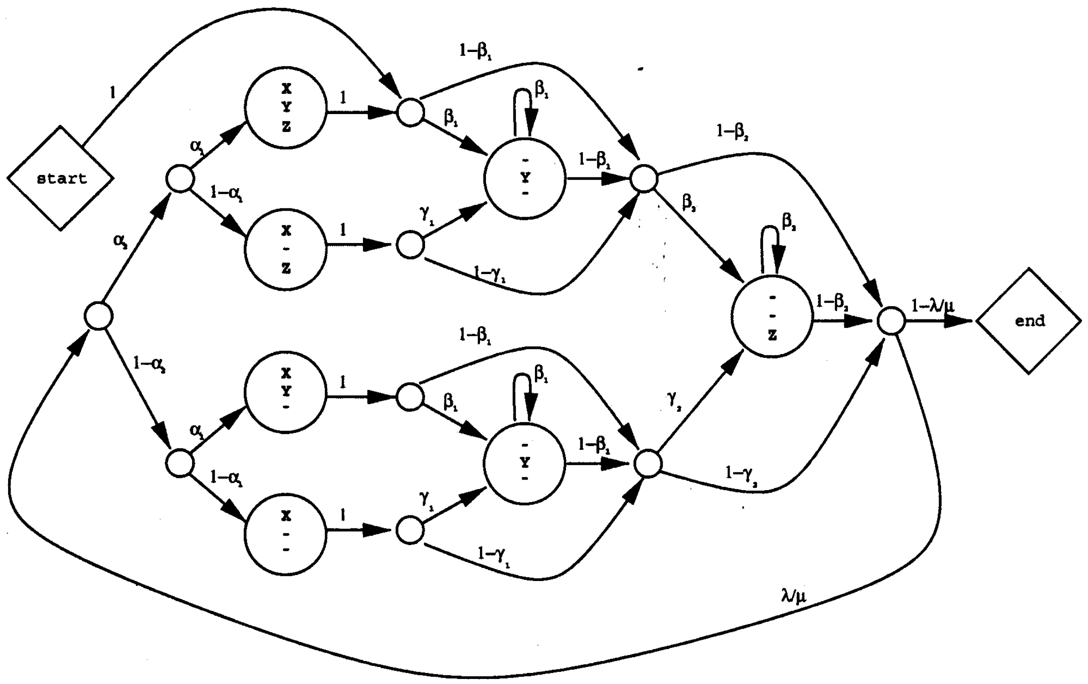


Figure 5.6: *The multiple HMM for the binary tree of Figure 5.5. Note that* $\alpha_1 = \alpha(t_1)$, $\alpha_2 = \alpha(t_2)$, *etc., according with equation (5.1); Excerpt from [12].*

A generalization of this task is to find the optimal multiple alignment given an incomplete pairwise alignment set. The algorithm to do this will not be described in full, but the essential rule is as follows: residues X_i and Y_j in a multiple alignment containing sequences X and Y are considered to be aligned if and only if *both* of the following conditions hold:

1. the residues X_i and Y_j are in the same column;

2. the column contains no gap characters for any of the sequences intermediate to X and Y on the tree.

The 'intermediate' sequences include any sequences in the lineages $A \to X$ or

$A \to Y$, where A is the most recent common ancestor of X and Y. This sentence stipulates that residue deletion followed by re-insertion at the same position does not constitute a direct evolutionary relationship under this model and there should be no correlation between X and Y. Note that Figure 5.8 contains no states that emit both Ws and Ys without emitting Xs; this is because X is intermediate to W and Y in Figure 5.7.

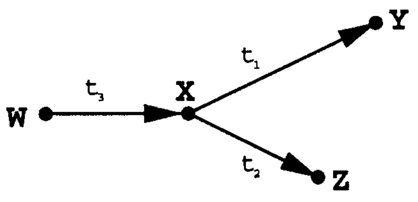


Figure 5.7: *A tree for a node (X) with a parent (W) and two children (Y and Z); Excerpt from [12].*

5.2.2 Eliminating internal nodes

Often, the actual inference of ancestral sequence is unnecessary. In Bayesian theory, these sequences are 'missing data' and the correct thing to do would be to sum over them of the likelihood function. Unfortunately, summing over the indel histories of these sequences means giving up the branch-to-branch independence that allows them to conveniently factorize the likelihood function as in equation (5.2).

Despite this problem, they could sum over all substitution histories for a given multiple alignment using the post-order traversal algorithm of Felsenstein which we introduce in the section 3.2.3. And the scores of the Felsenstein algorithm are a probabilistic summing over over all residues.

In MCMC analysis, troublesome computations like this problem are avoided by sampling extensively from the posterior distribution. This is the approach taken by *Handel.*(You can download the source code at

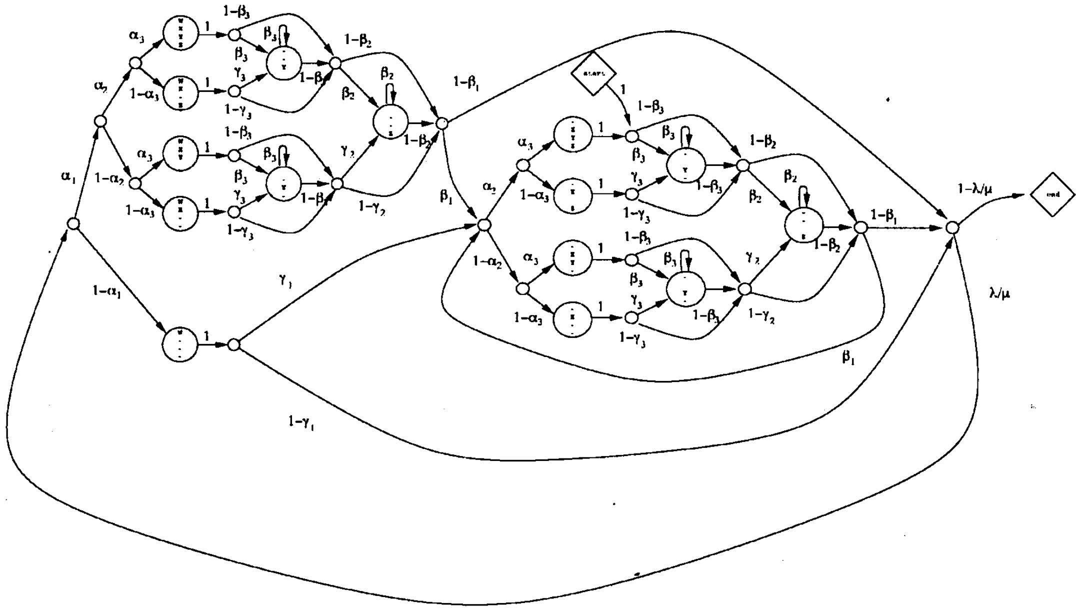


Figure 5.8: *The multiple HMM for the tree of Figure 5.7. Note that* $\alpha_1 = \alpha(t_1)$, $\alpha_2 = \alpha(t_2)$, $\alpha_3 = \alpha(t_3)$, *etc., according with equation (5.1); Excerpt from [12].*

http://sourceforge.net/projects/dart/, and you can see directly the code on the web site at http://b-src.cbrc.jp/markup/dart).

5.3 Gibbs sampling method

Before we introduce the algorithm, we see what Gibbs sampling method is and you can refer to [2], [3], [9] and [13] for detail. Markov chain Monte Carlo (MCMC) methodology provides huge range for realistic statistical modelling. MCMC is essentially Monte Carlo integration using Markov chains. Bayesians and frequentists need to integrate over possibly high-dimensional probability distributions to make inference about model parameters or to make predictions.

Bayesians need to integrate over the posterior distribution of model parameters given the data (i.e. the posterior distribution is $P(\text{model parameters}|\text{data})$). Frequentists need to integrate over the distribution of observables given parameter values (i.e. the frequentist is $P(\text{data}|\text{model parameters})$). Monte Carlo integration draws samples from the required distribution (is called the proposal distribution) and then forms sample averages to approximate expectations. MCMC draws these samples by running a cleverly constructed Markov chain for a long time. Metropolis and Gibbs samplers are included in MCMC.

The Metropolis algorithm (Metropolis *et al.*, 1953) [19] considers only symmetric proposals, having the form $q(Y|X) = q(X|Y)$ for all X and Y. Note that the proposal distribution may depend on the current point X_t. For example, when X is continuous, $q(.|X)$ might be a multivariate normal distribution with mean X. At each time t, the next state X_{t+1} is chosen by first sampling a candidate point Y from a proposal distribution $q(.|X_t)$.

The candidate point Y is then accepted with probability $\alpha(X_t, Y)$ where

$$\alpha(X, Y) = \min(1, \frac{\pi(Y)}{\pi(X)})$$

where $\pi(.)$ is the distribution of data. If the candidate point is accepted, the next state becomes $X_{t+1} = Y$. If the candidate is rejected, the chain does not move, i.e. $X_{t+1} = X_t$.

Algorithm

```
1. Initialize $X_0$; set $t = 0$.
2. Repeat:
    (1) Sample a point $Y$ from $q(.|X_t)$.
    (2) Sample a Uniform(0,1) random variable $U$.
    (3) If $U \leq \alpha(X_t, Y)$ set $X_{t+1} = Y$
        otherwise set $X_{t+1} = X_t$.
    (4) Increment $t$.
```

The 'Gibbs sampler' is a special case of Metropolis algorithm. The 'Gibbs sampler' was given its name by Geman and Geman (1984) [8], who used it for analysing Gibbs distributions on lattices. Moreover, the same method was already in use in statistical physics and was known there as the heat bath algorithm. To date, most statistical applications of MCMC have used Gibbs sampling.

For Gibbs sampler, the proposal distribution is

$$P(Y|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N)$$

for all i, cycling repeatedly through $i = 1, \ldots, N$. The candidate point Y is then accepted with probability $\alpha(X_t, Y)$ where

$$\alpha(X, Y) = \min(1, \frac{\pi(Y)q(X|Y)}{\pi(X)q(Y|X)})$$

where $\pi(.)$ is the distribution of data. If the candidate point is accepted, the next state becomes $X_{t+1} = Y$. If the candidate is rejected, the chain does not move, i.e. $X_{t+1} = X_t$.

Provided that the process is ergodic, Metropolis and Gibbs sampling will surely converge to a stationary distribution. Once a sample from the stationary distribution has been obtained, all subsequent samples will be from that distribution. When the distribution of X converge to a stationary distribution, X is ergodic. The chain needs to satisfy three important properties, the distribution of X converge to a stationary distribution.

1. *irreducible*: from all starting points, the Markov chain can reach any non-empty set with positive probability, in some number of iterations. In other words, if for all $i,\ j$, there exists a $t > 0$ such that $P_{ij}(t) > 0$ where $P_{ij}(t)$ is the transition probability, $P_{ij}(t) = P(X_t = j|X_0 = i)$.

2. *aperiodic*: this stop the Markov chain from oscillating between different sets of states in a regular periodic movement. In other words, if for some (and hence for all) i, greatest common divider $t > 0:\ P_{ii}(t) > 0$=1.

3. *positive recurrent*: the initial value X_0 is sampled from a stationary distribution, then all subsequent iterates will also distributed according to the stationary distribution. In other words, the existence of a stationary distribution for X, that is there exists $\pi(.)$ such that $\sum_i = \pi(i)P_{ij}(t) = \pi(j)$ for all j and $t \geq 0$.

5.4 Algorithm: under a given tree

The most popular kind of multiple alignment algorithms is progressive alignment, whereby profiles for missing parents are estimated by aligning relative sequences on a post-order traversal of the underlying binary tree. A third strategy is to sample from a population of alignments, exploring suboptimal alignments in anticipation that short-term sacrifices will yield long-term improvements.

Three types of 'move' are used to explore alignment space. Holmes and Bruno first discuss the motivation for each move, and then describe the move sligntly alignment more formally.

The first move mirrors the sibling alignment step of progressive alignment (building a 'guide tree'). Given two relative sequences (choosing from the 'guide tree'), their alignment is sampled. The length of the parent sequence is implicitly sampled at this stage as well.

- Move #1: parent sampling
  - The goal is to align relative nodes Y and Z and simultaneously infer their parent node X (see Figure 5.5).
  - Construct the pair HMM for X, Y and Z.
  - Realign the alignment of Y and Z using the Forward algorithm. That is, we calculate the value of $\frac{V^E(n,m)(\text{Viterbi Aigorithm})}{f^E(n,m)(\text{Forward algorithm})}$.
  - Deduce the implicit alignments XY and XZ and the sequence X.

The second move mirrors the branch alignment step of refined alignment. Given a branch, the pairwise alignment for that parent-child sequence pair is resampled, by applying the Forward algorithm to Figure 5.3. This move resamples alignments inferred during the progressive phase.

- Move #2: branch sampling
  - The goal is to realign the adjacent nodes X and Y (see Figure 5.4).
  - Fix all pairwise branch alignment except branch XY (use the Gibbs sampler method) and construct the pair HMM for X and Y.
  - Realign the alignment of X and Y using the Forward algorithm. That is, we calculate the value of $\frac{V^E(n,m)(\text{Viterbi Aigorithm})}{f^E(n,m)(\text{Forward algorithm})}$.

The third move completes the ergodicity requirement. Given any internal node, the sequence at that node is resampled by inserting or deleting residues without disturbing the pairwise alignment of adjacent nodes. This move resamples parent sequence lengths inferred during the progressing phase.

- Move #3: node sampling
  - The goal is to resample the sequence at internal node X.
  - Let the parent of X be W. Let the children of X be Y and Z (see Figure 5.7).
  - Fix all pairwise branch alignment except branch WX, XY and XZ (use the Gibbs sampler method). Construct the multiple HMM for X and its neighbors. (see Figure 5.8.)
  - Realign the sequence X, conditioned on the relative alignment of W, Y and Z. That is, we calculate the value of $\frac{V^E(n,m)(\text{Viterbi Aigorithm})}{f^E(n,m)(\text{Forward algorithm})}$. (In other words, all variants of the original multiple alignment having either a residue or a gap character at each column of row X are considered.)

Chapter 6

Tree HMM

Our materials come from Mitchison's paper [18]. Carrying out simultaneous tree-building and alignment of sequence data is a difficult computational task. The methods currently available are either limited to a few sequences or restricted to highly simplified models of alignment and phylogeny. A method given by Mitchison is to overcome these limitations by Bayesian sampling of trees and alignments simultaneously. The method uses a standard substitution matrix model for residues together with a hidden Markov model structure that allows affine gap penalties.

6.1 Modeling of phylogeny and alignment: The Tree-HMM

Alignment and phylogeny can be treated simultaneously by combining an alignment model, a profile-HMM, with a probabilistic model of phylogeny. The resulting model is called a tree-HMM. Here we follow Mitchison' paper [18] and he stressed the idea of evolutionary changes of paths through an HMM.

The basic idea of a tree-HMM is that a path through an HMM, which represents an alignment of a sequence, can change as a result of evolution and that the probability of such changes is given by substitution probabilities. With the tree-HMM, Mitchison is no longer concerned with emission and transition probabilities, as in a standard HMM described in chapter 3, but with the probabilities of an emission or transition being substituted by another emission or transition.

The HMM in question can have any architecture. Here a profile-HMM structure is assumed, with only match and delete states (Figure 6.1), as there are certain complications in the use of the insert states present in the original profile-HMM. Insertions are still allowed, however, even though they are not represented by a special state. Insertions occur when a sequence uses a match state at a position where its ancestor uses a delete state.

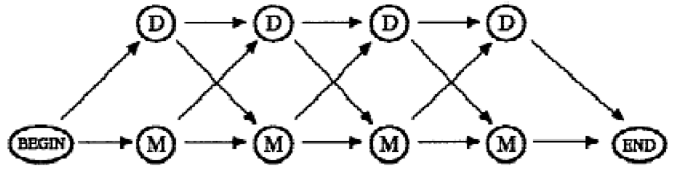


Figure 6.1: *An HMM-profile architecture that provides a convenient structure for a tree-HMM. It is simpler than the standard version, having no insert states; Excerpt from [18].*

Consider the simplest possible evolutionary tree, T, consisting of a single edge of length d, with a leaf node at one end and a root node at the other. Let the sequence of the leaf be x and that at the y. The tree means that x has evolved from y over an evolutionary distance d. Figure 6.2 shows an example of what these paths might look like for a model of length 4. Observe that the paths can differ in two ways: they can use different transitions and states, and they can emit different residues (Mitchison included emissions in the definition of a 'path').

Consider now the various types of differences between paths. At the M state at position 1, x emits an A, whereas y emits a V. Mitchison assigned a probability $P_d(A|V)$ to this substitution and assume henceforth that this is the

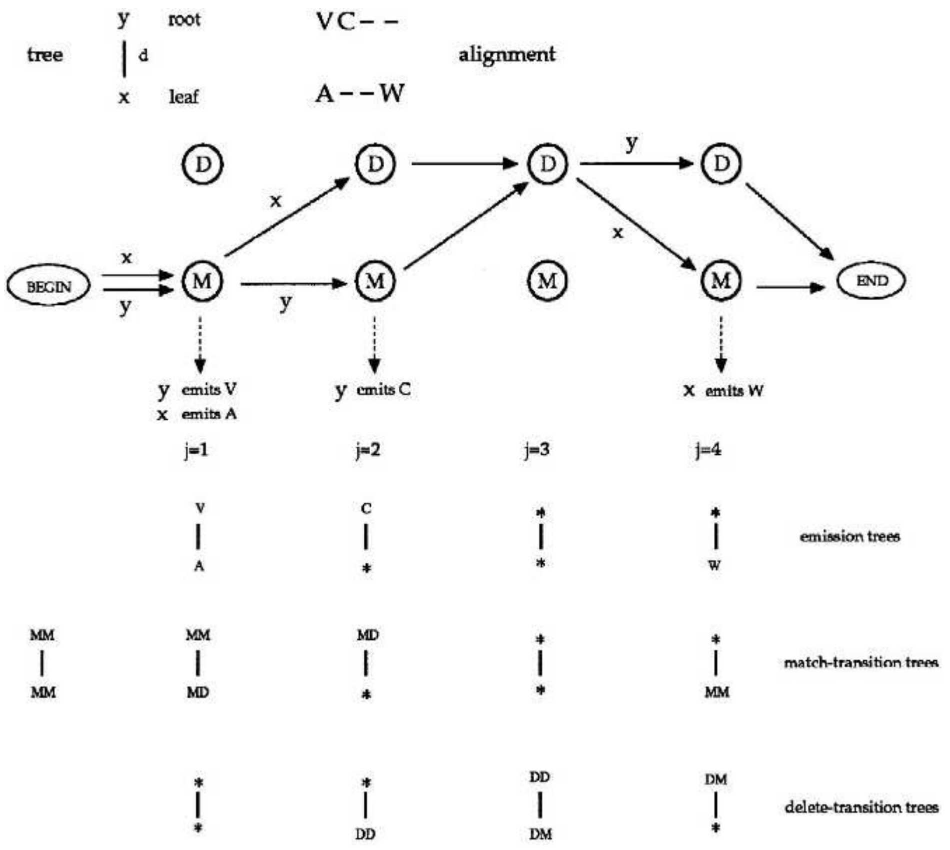


Figure 6.2: *A short tree-HMM for the simple tree with two nodes shown above. Note that there is only a match-transition tree at the BEGIN state because it is treated as a dummy match state that emits no residue; Excerpt from [18].*

familiar Dayhoff matrix element for a distance of d PAMs. Then we have

$$P(A, V|T) = \rho(V)P_d(A|V),$$

where ρ is the prior for emissions. $P(A, V|T)$ will be one of the terms whose product defines $P(x, y|T)$.

In going form position 1 to position 2, y undergoes a transition from state M to M, whereas x undergoes a transition from state M to D. We use the compact notation 'MM' for the former transition and 'MD' for the later. The substitution of MM by MD can be treated in an analogous way to substitution of emitted residues, its probability being denoted $P_d(MD|MM)$. The probability $P(MD, MM|T)$, which provides another factor in $P(x, y|T)$, is

$$P(MD, MM|T) = \rho(MM)P_d(MD|MM),$$

where $\rho(MM)$ is the prior probability for MM. There are analogous substitution probabilities for the delete state. At position 3, x undergoes a DM transition and y a DD. Then

$$P(DM, DD|T) = \rho(DD)P_d(DM|DD).$$

In positions 2 and 4 of the model they use different states. Then x cannot be regarded as evolving from y. Mitchison consider x having the missing ancestral and y having the missing descendent. Mitchison assume that their emission or transitions in the course of the separate paths they follow should be treated as independent of each other. As an attempt to capture this, Mitchison adopt a rule of replacing the missing ancestral or descendent sequences at such positions by sum over all possible emissions or transitions and use the symbol '*' to denote this sum.

At position 2, for instance, the foregoing rule means that the delete-transition tree has DD at the leaf and a * at the root, so

$$\begin{aligned} P(DD, *|T) = &\ \rho(DM)P_d(DD|DM) + \rho(DD)P_d(DD|DD) \\ = &\ \rho(DD)(P_d(DD|DD) + P_d(DM|DD)) \\ &\ \text{(the substitution probability is reversible)} \\ = &\ \rho(DD) \end{aligned} \tag{6.1}$$

Thus one gets the prior for DD. Similarly, at position 2 has a * at the leaf and MD at the root, so

$$\begin{aligned} P(*, MD|T) = &\ \rho(MD)P_d(MM|MD) + \rho(MD)P_d(MD|MD) \\ = &\ \rho(MD)(P_d(MD|MD) + P_d(DM|MD)) \\ &\ \text{(the substitution probability is reversible)} \\ = &\ \rho(MD) \end{aligned} \tag{6.2}$$

giving the prior probability of y's transition MD. Finally, note that at positions where a state is not used by any sequence, for instance, the D at position 1, the tree has *'s at both leaf and root, and the probability $P(*, *|T)$ is 1.

Then we can represent both substitutions and priors for transition in a 4×4 matrix, corresponding to the four transitions. However, this is not a standard substitution matrix because the probabilities in a row do not sum to one. Instead, it breaks up into four 2×2 blocks, determined by the state (match or delete) that the ancestral and descendant sequences begin their transition from:

	MM	MD	DM	DD
MM	$P_d(MM\|MM)$	$P_d(MD\|MM)$	$\rho(DM)$	$\rho(DD)$
MD	$P_d(MM\|MD)$	$P_d(MD\|MD)$	$\rho(DM)$	$\rho(DD)$
DM	$\rho(MM)$	$\rho(MD)$	$P_d(DM\|DM)$	$P_d(DD\|DM)$
DD	$\rho(MM)$	$\rho(MD)$	$P_d(DD\|DM)$	$P_d(DD\|DD)$

And Mitchison called

$$\begin{pmatrix} P_d(MM|MM) & P_d(MD|MM) \\ P_d(MM|MD) & P_d(MD|MD) \end{pmatrix}$$

as the match-transition matrix family and

$$\begin{pmatrix} P_d(DM|DM) & P_d(DD|DM) \\ P_d(DD|DM) & P_d(DD|DD) \end{pmatrix}$$

as the delete-transition matrix family.

Multiplying together the probabilities of all transitions and emissions in the paths x, y gives $P(x, y|T)$. To express this formally, let $M^k(x_i)$ denote the transition from the match state used by sequence x_i at position k; a * otherwise. Similarly, let $E^k(x_i)$ be the emission, and $D^k(x_i)$ the transition from the delete state, at k, either being a * if the relative state is not used. Then we have

$$P(x, y|T) = \prod_k P(M^k(x), M^k(y)|T) P(D^k(x), D^k(y)|T) P(E^k(x), E^k(y)|T) \tag{6.3}$$

Since the root sequence y is generally unknown, to get the probability of the observed sequence, $P(x|T)$, we must sum over all y. This means summing over all possible paths, including all possible emissions, for y. It is easy to see that this implies

$$P(x|T) = \prod_k P(M^k(x)|T)P(D^k(x)|T)P(E^k(x)|T) \tag{6.4}$$

where $P(E^k(x)|T)$ is the probability for the observed emission at position k obtained by summing over all possible root residues, i.e., emissions of y. $P(M^k(x)|T)$ and $P(D^k(x)|T)$ are similarly defined by summing over all root values of relevant transitions. Equation (6.3) implies Equation (6.4) because the sums over possible states and emissions of y distribute over the product. The * rule means that, when y uses an M state, all transitions from D are summed over, so all combinations of the terms in (6.4) occur.

Now it will be extended to any tree T with n leaves (Figure 6.3). Label the nodes $i = 1, \ldots, 2n-1$ with $i = 1, \ldots, n$ the leaves and $2n-1$ the root. Let d_i be the length of the edge that has node i at the bottom, and let $\alpha(i)$ denote the number of the node at the top of that edge. Suppose that the leaf sequences are $x_1, \ldots, x_n$ and the sequences at ancestral nodes are $y_{n+1}, \ldots, y_{2n-1}$. Let L be E, M or D, the probability of the specific assignments to all its nodes is given by

$$\begin{aligned}&P(L^k(x_1), \ldots, L^k(x_n), L^k(y_{n+1}), \ldots, L^k(y_{2n-1})|T)\\ &= \rho(L^k(y_{2n-1})) \prod_{i=1}^{n} P_{d_i}(L^k(x_i)|L^k(y_{\alpha(i)})) \prod_{i=n+1}^{2n-2} P_{d_i}(L^k(y_i)|L^k(y_{\alpha(i)}))\end{aligned}$$

the product being taken over all edges. The analogue of (6.3) is then

$$\begin{aligned}&P(x_1, \ldots, x_n, y_{n+1}, \ldots, y_{2n-1})\\ &= \prod_L \prod_k P(L^k(x_1), \ldots, L^k(x_n), L^k(y_{n+1}), \ldots, L^k(y_{2n-1})|T)\end{aligned}$$

As before, the sum over all nonleaf nodes, over all y_i, distributes over the product, giving

$$P(x_1, \ldots, x_n) = \prod_L \prod_k P(L^k(x_1), \ldots, L^k(x_n)|T)$$

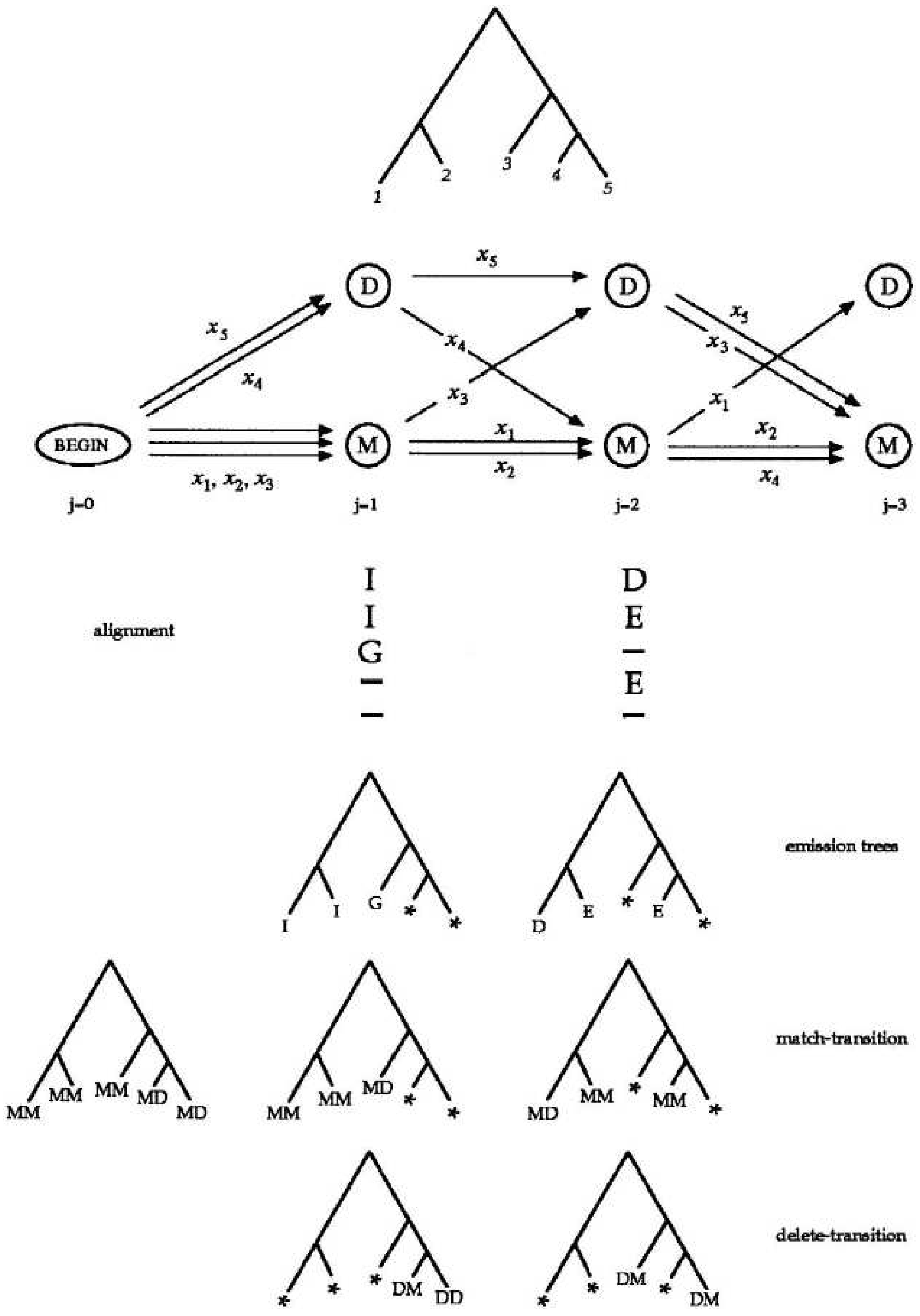


Figure 6.3: *A tree-HMM has five leaves. To compute the probabilities of data conditioned on the trees, the standard Neyman/Felsenstein algorithm is used, with the additional rule that all possible residues are summed over at a leaf with a *; Excerpt from [18].*

The existence of a fixed alignment means that the path through the tree-HMM for each sequence is fixed; the tree parameters T and d_i can then be varied to maximize the likelihood or sample from the posterior. If we have a number of data sets, each consisting of four globin sequences. Then the choosing quadruples have the effect of making one of the three possible tree topologies more likely than the other two.

To define the tree-HMM used for these data sets, we need to specify the substitution matrix families and the priors. The PAM family was used for substitution of emissions and its equilibrium probabilities for large values of d were used as the emission prior. The match-transition matrix was assumed to have the following form:

	MM	MD
MM	$a + (1-a)e^{-rd}$	$(1-a)(1-e^{-rd})$
MD	$a - ae^{-rd}$	$1 - a + ae^{-rd}$

where d is the evolutionary distance, $r \geq 0$ a rate constant, and $0 \leq a \leq 1$ determines the equilibrium probabilities for large d. If one takes the priors to be the equilibrium probabilities, one gets $\rho(MM) = a$, $\rho(MD) = 1-a$. This matrix family is reversible and multiplicative.

These can be estimated by maximum-likelihood from a given data set, choosing a and r to maximize the product of all substitution probabilities arising in the data set. The delete-transition matrix was assumed to have a similar form, though it was not constrained to have the same parameters as match-transition matrix.

6.2 Sampling from alignments and trees

We return to the use of tree-HMMs as tools for simultaneous sampling of trees and alignments. There are two procedures to be used. To sample alignments, the first procedure is leaf-sampling used the Gibbs sampler method (This method is described in chapter 5). The second is inter-nodes-sampling. All steps are carried out under a given tree.

A tree-HMM at leaf node k is obtained by computing the probability

$$P(y \text{ at leaf } k | \{x_i\} - x_k)$$

of emissions or transitions y at k given the residues or transitions at all the leaves other than k. These probabilities supply the emission or transition probabilities for the HMM-profile that is used to sample alignments of sequence x_k at leaf k.

In the leaf sampling, it derives a guide tree with the correct leaves alignment. Now we do not change these leaves alignment and we will resample the internal nodes. The child sequences of internal node m are realigned to the ancestor sequences of internal node m. To give freedom for realignment, a special type of tree is treated as a 'blank'; this is the tree all of whole leaves use D states.

A blank tree can inserted or removed from the alignments, provided that the transitions are modified appropriately. For instance, if a leaf sequence uses M at positions k and $k+1$, so it makes an MM transitions at k. Inserting a blank tree at position $k+1$ means that the leaf uses M at k, D at $k+1$, and M at $k+2$. Hence the leaf sequence makes an MD transition at k and a DM transition at $k+1$.

Standard algorithms (Felsentein 1981; Mitchison and Durbin 1995) enable one to compute the probability $P^{\downarrow}(y)$ of emissions or transitions in the tree below m given y at node m, i.e.,

$$P^{\downarrow}(y) = P(\{L^k(x_{u_i})\}_{u_i \text{ below } m} | y \text{ at node } m, T)$$

where k is each position below node m. Similarly, one can compute the probability of y at node m given the tree above m, i.e.,

$$P^{\uparrow}(y) = P(y \text{ at node } m | \{L^k(x_{u_i})\}_{u_i \text{ above } m}, T)$$

Summing over the product of these distributions gives

$$\sum_y P^{\uparrow}(y)P^{\downarrow}(y) = \frac{P(\{L^k(x_i)\}|T)}{P(\{L^k(x_{u_i})\}_{u_i\,\text{above}\,m}|T)} \tag{6.5}$$

using the fact that the sequence above and below m are independent. The probability is similar to the proposal distribution in the Gibbs sampler method. If we are aligning the part of the tree below node m to the rest of the tree,

$$P(\{L^k(x_{u_i})\}_{u_i\,\text{above}\,m}|T)$$

remains fixed, and this constant factor has no effect on the sampling. With the distinction that the probability is evaluated by the sum in (6.5), and that the blank tree replaces the use of the D state, the sampling procedure is identical to that for the leaves.

After these preliminaries, we were ready to sample from tree parameters as well as alignments. This was achieved by randomly choosing either alignment sampling or tree sampling, the latter using the method of Mau *et al.*(1996) [16], with a flat prior on edge length.

The first one uses a representation of a tree that they call a traversal profile. In the traversal profile, a node is placed at a height corresponding to the sum of the edge lengths from the root to that node. Then beginning at the leftmost leaf, we traverse the tree depth first from left to right and assigning numbers incrementally according to the x-coordinate. (see the top of Figure 6.4). The root is taken to be the highest node. The root is taken to be the highest node. Edges are then drawn to the highest nodes to the left and the right of the root. The process stops when a leaf is reached (the leaves have been marked as hollow circles in the Figure 6.4)

Mau *et al.* take the traversal profile for the current tree and shifting the positions of nodes up and down by amount chosen from a uniform distribution in some interval $[x - \delta, x + \delta]$, where x is the height of the node. Whenever the relative heights of nodes are switched a new topology is produced (Figure 6.5).

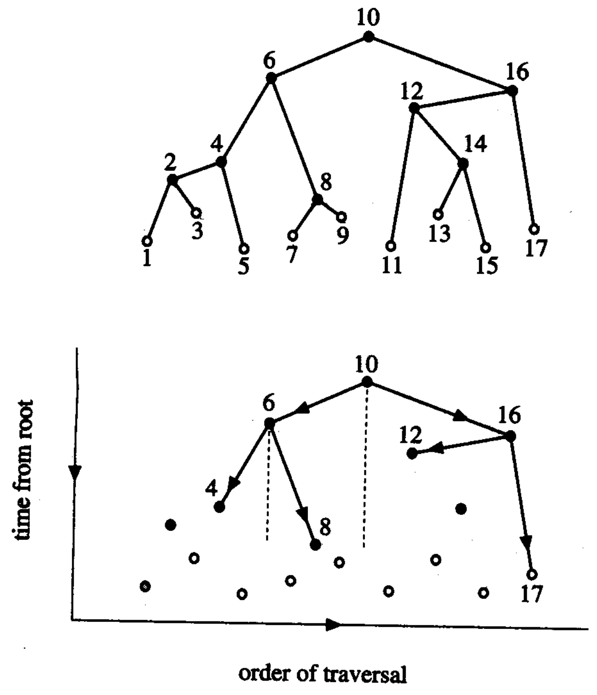


Figure 6.4: *Above: an example of a tree with its nodes numbered in the order of the traversal profile. Below: Reconstruction of the tree from the traversal profile; come from [3]; Excerpt from [3].*

When a molecular clock is assumed, the leaf nodes all lie at the same height and shifting the positions of nodes is reflected upward.

Without a molecular clock, leaf nodes are always leaf nodes. This can be achieved by two proposal mechanisms. The first displaces leaf nodes uniformly in some interval, but if the displacement increases the leaf node's height above that of the lower of the two neighboring nonleaf nodes, the displacement is reflected downward. The second displaces nonleaf nodes, but now they are reflected upward if they cross the level of either of the two neighboring leaf nodes.

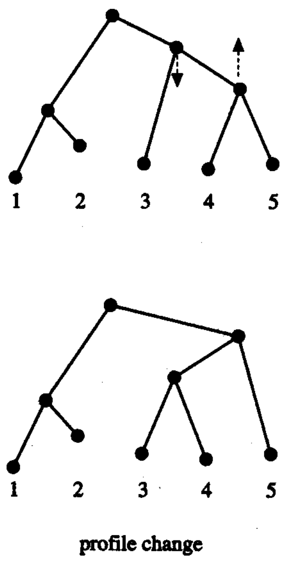


Figure 6.5: *The two parts of the proposal mechanism are changes in the height of the nodes in the profile; come from [3]; Excerpt from [3].*

Algorithm: carrying out tree-building and multiple sequences alignment simultaneously

1. tree-sampling : using the method of Mau et al. (1996)
2. multiple sequences alignment : using the tree that is chosen by 1 to do multiple sequences alignment.
3. repeat 1 and 2 until some threshold is achieved.

To assess the effectiveness of sampling, Mitchison defined the overlap of an alignment to be the fraction of individual residue pairs that were correctly aligned according to the Pfam database seed alignments of the globins (Sonnhammer et al. 1997; Bashford et al. 1987) [22] and [1].

The mean overlap is the average of this fraction over the sampling run. The mean overlap of alignments produced by Mitchison's sampling method was compared to that obtained with an efficient alignment program, CLUSTAL W

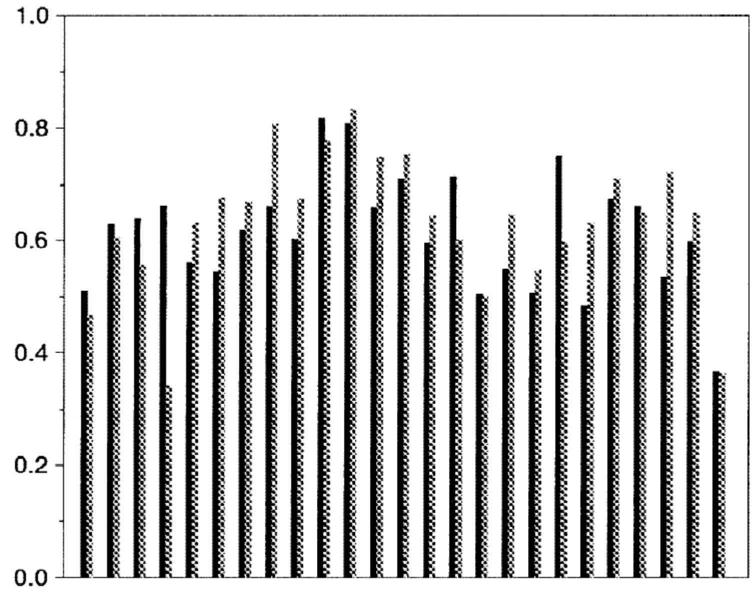


Figure 6.6: *This shows the degree to which algorithm-generated alignments of four globins agree with those in the Pfam database. The y axis gives the mean overlap for 25 sets of four globins. For each set, the mean overlap is given for an alignment generated by CLUSTAL W (gray bars). The black bars show the mean overlap for alignments generated by simultaneous sampling of trees and alignments, using the tree-HMM; Excerpt from [18].*

(Thompson et al. 1994) [23]. The latter produced alignments from the same globin data sets whose overlap varied between 0.34 and 0.83 (Figure 6.4). Note the low values; some of these data sets were not easy to align. In fact, alignment of small numbers of sequences is often particularly troublesome because of the scant statistical information they provide (Eddy 1995) [4], and practical experience suggests that profile-HHMs perform poorly at this task compared to CLUSTAL W. Mitchison's sampling procedure did only a little less well than CLUSTAL W judged by mean overlap (Figure 6.4), achieving an average value of 0.615 on 25 data sets, compared to 0.631 for CLUSTAL W. For comparison, alignment with a profile-HMM, using S. Eddy's package hmmer (simulated annealing with hmmer version 1.8.4; http://hmmer.wustl.edu/) gave a mean overlap of 0.257.

6.3 Conclusion and discussion

The tree-HMM can be used in several ways.

1. it can be used for standard phylogenetic inference, given an aligned set of sequences, but with the advantage that it treats insertions and deletions more realistically compared to simple character substitution models of gaps.

2. the tree-HMM can be used as an alignment tool that assumes a specific phylogeny. Lake (1991) [14] has pointed out that there is a danger in using certain alignment algorithms before carrying out a phylogenetic analysis, because these alignment algorithms assume a tree.

3. it is possible to combine phylogeny with alignment, by means of sampling.

Even so, it suffers from some lack of realism, because when a group of adjacent bases is deleted, the bases retain information about the base sequence, and if they are inserted again, there will be some memory of the original base sequence. Holmes and Bruno (2001) [12] point out that it is also possible that when a series of bases is reinserted there may be 'memory' of an internal gap that was once there and that now returns with them. (Excerpt from [7])

The interpretation the tree-HMM give to these events may often be biologically incorrect. Once a deletion has occurred, a subsequent insertion may have a different structural role. A more realistic model would assign new states to insertions, which is what the model of Thorne et al. (1991) [24] does. In Thorne's model, deletion followed by insertion using the same states could not occur, and the tree-HMM with this structure would behave more correctly as an evolutionary model. There is clearly range for devising new tree-HMM architectures and reason to hope that they will provide useful tools for modeling the evolution of sequence families.

References

[1] Bashford, D., Chothia, C., Lesk, A. M. 1987 Determinants of a protein fold: Unique features of the globin amino acid sequence. J Mol Biol 196:199—216

[2] Brémaud, P. 1999 *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues.* Springer-Verlag New York, Inc.

[3] Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. 1998 *Biological sequence analysis: Probabilistic models of proteins and nucleic acids* Cambridge University Press

[4] Eddy, SR 1995 Multiple alignment using hidden Markov models. In: Rawlings C, Clark D, Altman R, Hunter L, Lengauer T, Wodak S (eds) Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology. AAAI Press, Menlo Park, CA, pp 114—120.

[5] Eddy, S. R. 1996 Hidden Markov models current opinion in structural biology, 6:361-365

[6] Feldman, R. M. and Ciriaco Valdez-Flores 1996 *Applied Probability & Stochastic Processes.* PWS Publishing Company, a division of International Thomson Publishing Inc.

[7] Felsenstein, J. 2004 *Inferring Phylogenies* Ainauer Associates, Inc.

[8] Geman, S. and Geman, D. 1984 Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. IEEE Trans. Pattn. Anal. Mach. Intel. 6: 721-741.

[9] Gilks, W. R., Richardson, S. and Spiegelhalter, D. J. 1996 *Markov Chain Monte Carlo In Practice* Chapman & Hall

[10] Grassmann, W. K. 1981 *Stochastic systems for Management.* Elsevier North Holland, Inc.

[11] Gusfield, D. 1997 *Algorithms on Strings, Trees, and Sequences: computer science and computational biology.* Cambridge University Press

[12] Holmes, I and Bruno, W. J. 2001 Evolutionary HMMs: a Bayesian approach to multiple alignment. Bioinformatices Vol. 17 no. 9 2001 Pages 803-802

[13] Huelsenbeck, J. P. MrBayes: A program for the Bayesian inference of phylogeny Department of Biology, University of Rochester, Rochester, NY 14627, U.S.A

[14] Lake, J. A 1991 The order of sequence alignment can bias the selection of tree topology. Mol Biol Evol 8:378—385.

[15] Lawler, G. F. 1995 *Introduction to Stochastic Processes* Chapman & Hall

[16] Mau, B., Newton, M. A., Larget, B. 1996 Bayesian phylogenetic inference via Markov chain Monte Carlo methods, Technical Report No. 961. Statistics Department, University of Wisconsin-Madison.

[17] Mitchison, G. J. and Durbin R . M. 1995 Tree-based maximal likelihood substitution matrices and hidden Markov Models. J Mol Evol 41:1139-1151.

[18] Mitchison, G. J. 1999 A probabilistic Treatment of Phylogeney and Sequence Alignemet. Jouranl of Molecular Evolution J Mol Evol 49:11-22 Spronger-Verlag New York Inc.

[19] Metropolis, N., Roesenbluth, A. W., Roesenbluth, M. N., Teller, A. H. and Teller, E. 1953 Equations of state calculations by fast computing machine. J. Chem. Phys., 21: 1087-1091.

[20] Mount, D. W. 2004 *Bioinformatics: Sequence and Genome Analysis.* Cold Spring Harbor Laboratory Press

[21] Papoulis, A. 1984 *Porbability, Random Variables, and Stochastic Processes.* McHraw-Hill Inc.

[22] Sonnhammer, E., Eddy, S. R., Durbin, R. M. 1997 A comprehensive database of protein families based on seed alignments. Proteins 28:405—420 1997

[23] Thompson, J. D, Higgins, D. G, Gibson, T. J 1994 CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. Nucleic Acids Res 22:4673—4680.

[24] Throne, J. L., Kishino, H. and Felsenstein, J. 1991 An evolutionary model for maximum likelihood alignment of DNA sequences. J Mol Evol 33,114-124.

[25] Throne, J. L., Kishino, H. and Felsenstein, J. 1992 Inching toward relity: an improved likelihood model of squence evolution. J Mol Evol 34, 3-16.