That is, the "n-step" transition probabilities are given by the entries in the nth power of P.

4.7 Markov Chains with Absorbing States

Some Markov chains have absorbing states. These can be recognized by the appearance of one or more 1's on the main diagonal of the transition matrix. If there are no 1's on the main diagonal, then there are no absorbing states. For the Markov chains with absorbing states that we consider, sooner or later some absorbing state will be entered, never thereafter to be left. The two questions we are most interested in regarding these Markov chains are:

(i) If there are two or more absorbing states, what is the probability that a specified absorbing state is the one eventually entered?

(ii) What is the mean time until one or another absorbing state is eventually entered?

We shall address these questions in detail in Chapter 11. In the remainder of this chapter we discuss only certain aspects of the theory of Markov chains with no absorbing states, focusing on the theory needed for the construction of substitution matrices, to be discussed in more detail in Chapter 6.

4.8 Markov Chains with No Absorbing States

The questions of interest about a Markov chain with no absorbing state are quite different from those asked when there are absorbing states.

In order to simplify the discussion, we assume in the remainder of this chapter that all Markov chains discussed are *finite*, *aperiodic*, and *irreducible*.

Finiteness means that there is a finite number of possible states. The aperiodicity assumption is that there is no state such that a return to that state is possible only at t_0, $2t_0$, $3t_0, \ldots$ transitions later, where t_0 is an integer exceeding 1. If the transition matrix of a Markov chain with states E_1, E_2, E_3, E_4 is, for example,

$$P = \begin{bmatrix} 0 & 0 & 0.6 & 0.4 \\ 0 & 0 & 0.3 & 0.7 \\ 0.5 & 0.5 & 0 & 0 \\ 0.2 & 0.8 & 0 & 0 \end{bmatrix}, \tag{4.24}$$

then the Markov chain is periodic. If the Markovian random variable starts (at time 0) in E_1, then at time 1 it must be either in E_3 or E_4, at time 2 it must be in either E_1 or E_2, and in general it can visit only E_1 at times $2, 4, 6, \ldots$. It is therefore periodic. The aperiodicity assumption holds for essentially all applications of Markov chains in bioinformatics, and we often take aperiodicity for granted without any explicit statement being made.

The irreducibility assumption implies that any state can eventually be reached from any other state, if not in one step then after several steps. Except for the case of Markov chains with absorbing states, the irreducibility assumption also holds for essentially all applications in bioinformatics.

4.8.1 Stationary Distributions

Suppose that a Markov chain has transition matrix P and that at time t the probability that the process is in state E_j is φ_j, $j = 1, 2, \ldots, s$. This implies that the probability that at time $t + 1$ the process is in state j is $\sum_{k=1}^{s} \varphi_k p_{kj}$. Suppose that for every j these two probabilities are equal, so that

$$\varphi_j = \sum_{k=1}^{s} \varphi_k p_{kj}, \quad j = 1, 2, \ldots, s. \tag{4.25}$$

In this case the probability distribution $(\varphi_1, \varphi_2, \ldots, \varphi_s)$ is said to be *stationary*; that is, the probability that the process is in state E_j has not changed between times t and $t + 1$, and therefore will never change. Despite this, the state occupied by the process can of course change from one time point to the next.

It will be shown in Chapter 11 that for finite aperiodic irreducible Markov chains there is a unique distribution satisfying (4.25). This is then called the *stationary distribution* of the Markov chain. When we discuss stationary distributions in this book, we assume that they relate to finite aperiodic irreducible Markov chains, and thus exist and are unique.

If the row vector $\boldsymbol{\varphi}'$ is defined by

$$\boldsymbol{\varphi}' = (\varphi_1, \varphi_2, \ldots, \varphi_s), \tag{4.26}$$

then in matrix and vector notation, the set of equations in (4.25) becomes

$$\boldsymbol{\varphi}' = \boldsymbol{\varphi}' P. \tag{4.27}$$

The prime here is used to indicate the transposition of the row vector into a column vector. The vector $(\varphi_1, \varphi_2, \ldots, \varphi_s)$ must also satisfy the equation $\sum_k \varphi_k = 1$. In vector notation, this is the equation

$$\boldsymbol{\varphi}' \mathbf{1} = 1, \tag{4.28}$$

where $\mathbf{1} = (1, 1, \ldots, 1)'$. Equations (4.27) and (4.28) can then be used to find the stationary distribution when it exists. In this process one of the equations in (4.27) is redundant and can be omitted. An example is given in the next section.

In Chapter 11 we shall show that if the Markov chain is finite, aperiodic, and irreducible, then as n increases, $P^{(n)}$ approaches the matrix

$$\begin{bmatrix} \varphi_1 & \varphi_2 & \cdots & \varphi_s \\ \varphi_1 & \varphi_2 & \cdots & \varphi_s \\ \varphi_1 & \varphi_2 & \cdots & \varphi_s \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1 & \varphi_2 & \cdots & \varphi_s \end{bmatrix}, \tag{4.29}$$

where $(\varphi_1, \varphi_2, \ldots, \varphi_s)$ is the stationary distribution of the Markov chain.

The form of this matrix shows that no matter what the starting state was, or what was the initial probability distribution of the starting state, the probability that n time units later the process is in state j is increasingly closely approximated, as $n \to \infty$, by the value φ_j.

There is another implication, relating to long-term averages, of the calculations above. That is, if a Markov chain is observed for a very long time, then the proportion of times that it is observed to be in state E_j should be approximately φ_j, for all j.

4.8.2 Example

Consider the Markov chain with transition probability matrix given by

$$P = \begin{bmatrix} 0.6 & 0.1 & 0.2 & 0.1 \\ 0.1 & 0.7 & 0.1 & 0.1 \\ 0.2 & 0.2 & 0.5 & 0.1 \\ 0.1 & 0.3 & 0.1 & 0.5 \end{bmatrix}. \tag{4.30}$$

For this example the vector equation (4.27) consists of four separate linear equations in four unknowns. As noted above, when used jointly with (4.28) they form a redundant set of equations and any one of them can be discarded. Omission of the last equation in (4.27) leads to

$$\begin{aligned} 0.6\varphi_1 + 0.1\varphi_2 + 0.2\varphi_3 + 0.1\varphi_4 &= \varphi_1, \\ 0.1\varphi_1 + 0.7\varphi_2 + 0.2\varphi_3 + 0.3\varphi_4 &= \varphi_2, \\ 0.2\varphi_1 + 0.1\varphi_2 + 0.5\varphi_3 + 0.1\varphi_4 &= \varphi_3, \\ \varphi_1 + \varphi_2 + \varphi_3 + \varphi_4 &= 1. \end{aligned}$$

To four decimal place accuracy, these four simultaneous equations have the solution

$$\boldsymbol{\varphi}' = (0.2414, 0.3851, 0.2069, 0.1667). \tag{4.31}$$

This is the stationary distribution corresponding to the matrix P given in (4.30). In informal terms, from the point of view of long-term averages, over a long time period the random variable should spend about 24.14% of the time in state E_1, about 38.51% of the time in state E_2, and so on.

The rate at which the rows in $P^{(n)}$ approach this stationary distribution can be assessed from the following values:

$$P^{(2)} = \begin{bmatrix} 0.42 & 0.20 & 0.24 & 0.14 \\ 0.16 & 0.55 & 0.15 & 0.14 \\ 0.25 & 0.29 & 0.32 & 0.14 \\ 0.16 & 0.39 & 0.15 & 0.30 \end{bmatrix}, \tag{4.32}$$

$$P^{(4)} \cong \begin{bmatrix} 0.2908 & 0.3182 & 0.2286 & 0.1624 \\ 0.2151 & 0.4326 & 0.1899 & 0.1624 \\ 0.2538 & 0.3569 & 0.2269 & 0.1624 \\ 0.2151 & 0.4070 & 0.1899 & 0.1880 \end{bmatrix}, \tag{4.33}$$

$$P^{(8)} \cong \begin{bmatrix} 0.24596 & 0.37787 & 0.20961 & 0.16656 \\ 0.23873 & 0.38946 & 0.20525 & 0.16656 \\ 0.24309 & 0.38223 & 0.20812 & 0.16656 \\ 0.23873 & 0.38880 & 0.20525 & 0.16721 \end{bmatrix}, \tag{4.34}$$

$$P^{(16)} \cong \begin{bmatrix} 0.24142 & 0.38494 & 0.20692 & 0.16667 \\ 0.24135 & 0.38510 & 0.20688 & 0.16667 \\ 0.24140 & 0.38503 & 0.20691 & 0.16667 \\ 0.24135 & 0.38510 & 0.20688 & 0.16667 \end{bmatrix}. \tag{4.35}$$

After 16 time units, the stationary distribution has, for all practical purposes, been reached. The discussion in Chapter 11 shows how the rate at which this convergence occurs can be calculated in a more informative manner.

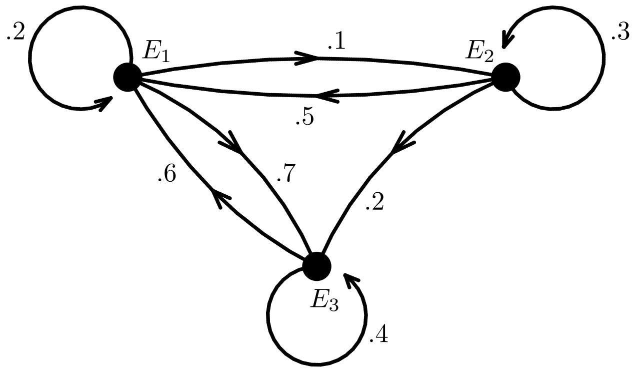
4.9 The Graphical Representation of a Markov Chain

It is often convenient to represent a Markov chain by a directed graph. A directed graph is a set of "nodes" and a set of "edges" connecting these nodes. The edges are "directed," that is, they are marked with arrows giving each edge an orientation from one node to another.

We represent a Markov chain by identifying the states with nodes and the transition probabilities with edges. Consider, for example, the Markov chain with states E_1, E_2, and E_3 and with probability transition matrix

$$\begin{bmatrix} .2 & .1 & .7 \\ .5 & .3 & .2 \\ .6 & 0 & .4 \end{bmatrix}.$$

This Markov chain is represented by the following graph:



Notice that we do not draw the edge if its corresponding transition probability is known to be zero, as is the case in this example with the transition from E_3 to E_2.

A graph helps us capture information at a glance that might not be so apparent from the transition matrix itself. Sometimes it is also convenient to include a *start state*; this is a dummy state that is visited only once, at the beginning. Therefore, all transition probabilities into the start state are zero. The transition probabilities out of the start state are given by the initial distribution of the Markov chain. If the Markov chain starts at time $t = 0$, we can think of the start state as being visited in time $t = -1$. We can further have an *end state*, which stops the Markov chain when visited.

We refer to the graph structure, without any probabilities, as the *topology* of the graph. Sometimes the topology of a model is known, but the various probabilities are unknown.

We will use these definitions when we discuss hidden Markov models in Chapter 12.

4.10 Modeling

There are many applications of the homogeneous Poisson process in bioinformatics. However, the two key assumptions made in the derivation of the Poisson distribution formula (4.10), namely homogeneity and independence, do not always hold in practice. Similarly, there are many applications of Markov chains in the literature, in particular in the evolutionary processes discussed in Chapter 14. Many of these applications also make assumptions, specifically the two Markov assumptions stated in Section 4.5. The modeling assumptions made in the evolutionary context are discussed further in Section 15.9.

In the context of DNA or protein sequence analysis, where time is replaced by position along the sequence, it is very likely that neither of these two Markov assumptions is correct. The data from chromosome 22 in humans (Dunham et al. (1999)) makes it apparent that the probability that the nucleotide a is next followed by g depends to some extent on the current location in the chromosome. Further, it is likely that the Markov chain memoryless assumption does not hold: The probability that the nucleotide a is next followed by g might well depend on the nucleotide (or nucleotides) immediately preceding a. Tests for this possibility are discussed in Section 5.2: Nevertheless these tests are often not applied, and the memoryless Markov chain theory is often assumed when its applicability is uncertain.

The construction of phylogenetic trees, both by algorithmic methods and by methods involving Markov chain evolutionary models, involves many assumptions, both explicit and implicit. An example of quite different phylogenetic trees arising from different models is given in Section 15.8. A discussion of various statistical tests for appropriate evolutionary models is discussed in Section 15.9.

Given that modeling assumptions made for both Poisson processes and Markov chains often do not hold exactly, one might ask why they are made and why there is such an extensive Poisson process and Markov chain literature. Mathematical models generally make simplifying assumptions about properties of the phenomena being modelled. This concern opens up the question of why we model natural phenomena with mathematics if we cannot do so with complete accuracy. In fact, it is not necessarily desirable that we attempt to make an extremely accurate model of reality. The more closely any phenomenon is modelled, the more complicated the model becomes. If a mathematical model becomes too complicated, then solving the equations necessary to find answers to the questions we wish to ask can become intractable. Therefore, we are almost always faced with the task of finding a middle ground between tractable simple models and intractable complex models. The key point is that a model need only capture enough of the true complexity of a situation to serve our purposes, whatever they might be.

Finding this middle ground, however, is not an easy task: Being able to extract the essence of a complex reality in a simplified model that then allows a successful mathematical analysis requires some skill and experience.

In biology it is not always possible to evaluate a model's efficacy directly. Rather, a model is often tested on how well it performs its job. Sometimes benchmarks can be well defined, but often efficacy is not easy to verify empirically, and subjectivity can enter in. This is an unfortunate but usually unavoidable problem.

To illustrate this we consider the early versions of the widely used BLAST procedure discussed in more detail in Chapter 10. One of the simplifying assumptions used is that nucleotides (or amino acids) are identically and independently distributed along a DNA (or protein) sequence. Current data show that this assumption is false. Nevertheless, this simple BLAST procedure does work, in that the model captures enough of biological reality to be effective.

A further aspect of the modeling process is that we do not expect any model to be the final one used. Any given process is often initially modelled using several simplifying assumptions, and then more refined models are introduced as time goes on. Indeed, applications of the simple models often indicate those areas in which more precise modeling is needed. Various updates of the BLAST procedure exemplify this. Recent versions of BLAST remove some of the simplifying assumptions made in earlier versions and provide an example of the joint evolution of models and data analysis. Unfortunately, the mathematical theory involved in these more sophisticated versions is far more complicated than that in the simpler BLAST theory, and we shall only outline it in this book.

Not every problem we might wish to solve with a model has a happy middle ground where our assumptions find a workable balance between tractability and reality. Thus while we should be willing to accept simplifying assumptions, we should always be on the lookout for oversimplifications, especially those that are not sufficiently backed up by testing for the efficacy of the model used. Model testing is an active area of statistical research in bioinformatics, and aspects of model testing, especially in the evolutionary and phylogenetic tree contexts, are discussed further in Section 15.9.

Problems

4.1. Prove (4.14) by repeated integration by parts of the right-hand side.

4.2. Events occur in a Poisson process with parameter λ. Given that 10 events occur in the time period $[0, 2]$, what is the probability that 6 of these events occur in the time period $[0, 1]$? Given that 6 of these events occur in the time period $[0, 1]$, what is the probability that 10 of these events occur in the time period $[0, 2]$?

4.3. ("Competing Poissons.") Suppose that events occur as described in Section 4.1, but that now each event is of one of k types, labeled types $1, 2, \ldots, k$. The type of any event is independent of the type of any other event. The probability that any event is of type i is p_i. Equation (4.10) continues to give the probability that exactly j events occur in the time period $(0, t)$. Assuming this,

(i) Find the (marginal) probability that j_i events of type i occur in the time period $(0, t)$.

(ii) Find the probability that j_i $(i = 1, \ldots, k)$ events of type i occur occur in the time period $(0, t)$.

(iii) Find the joint conditional probability that j_i $(i = 1, \ldots, k)$ events of type i occur in the time period $(0, t)$, given that j events in total occur in this time period. Relate your answer to expression (2.45).

4.4. The transition matrix of a Markov chain is

$$\begin{bmatrix} .7 & .3 \\ .4 & .6 \end{bmatrix}.$$

Find the stationary distribution of this Markov chain.

4.5. *Continuation.* If the initial probability distribution (at time 0) is $(.8, .2)$, what is the probability that at time 3 the state occupied is E_1?

4.6. The transition matrix of a Markov chain is

$$\begin{bmatrix} 1-a & a \\ b & 1-b \end{bmatrix}.$$

Find the stationary distribution of this Markov chain in terms of a and b, and interpret your result.

4.7. The transition matrix of a Markov chain is

$$\begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{bmatrix}.$$

Use induction on n (see Section B.18) to show that the probability that the Markov chain revisits the initial state at the nth transition is

$$p_{ii}^{(n)} = \frac{1}{4} + \frac{3}{4}(-\frac{1}{3})^n.$$

(This result is needed for Problem 14.7.)

4.8. Use equations (4.27) and (4.23) to show that the stationary distribution $\boldsymbol{\varphi}'$ satisfies the equation

$$\boldsymbol{\varphi}' = \boldsymbol{\varphi}' P^{(n)}, \tag{4.36}$$

for any positive integer n. For the numerical example in Section 4.8.2, use the expression for $P^{(2)}$ given in equation (4.32) and the expression for $\boldsymbol{\varphi}'$ given in (4.31) to check this claim for the case $n = 2$.

Why does equation (4.36) "make sense"?

4.9. Show that if the transition matrix of an irreducible, aperiodic, finite Markov chain is symmetric, then the stationary distribution is a (discrete) uniform distribution.

4.10. Show that if the transition matrix P of a Markov chain is of the circulant form

$$P = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & \cdots & a_{s-3} & a_{s-2} & a_{s-1} & a_s \\ a_s & a_1 & a_2 & a_3 & \cdots & a_{s-4} & a_{s-3} & a_{s-2} & a_{s-1} \\ a_{s-1} & a_s & a_1 & a_2 & \cdots & a_{s-5} & a_{s-4} & a_{s-3} & a_{s-2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ a_4 & a_5 & a_6 & a_7 & \cdots & a_s & a_1 & a_2 & a_3 \\ a_3 & a_4 & a_5 & a_6 & \cdots & a_{s-1} & a_s & a_1 & a_2 \\ a_2 & a_3 & a_4 & a_5 & \cdots & a_{s-2} & a_{s-1} & a_s & a_1 \end{bmatrix}, \tag{4.37}$$

where $a_j > 0$ for all j, then the stationary distribution is a (discrete) uniform distribution.

4.11. Suppose that the transition matrix of a Markov chain is

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ q & 0 & p & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & q & 0 & p & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & q & 0 & p & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & q & 0 & p \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{4.38}$$

Show that this Markov chain is periodic. Despite this fact, solve equations (4.27) and (4.28) for the case $p = q$.

4.12. Suppose that a transition matrix P is of size $2s \times 2s$ and can be written in the partitioned form

$$P = \left[\begin{array}{c:c} 0 & A \\ \hdashline B & 0 \end{array}\right],$$

where A and B are both $s \times s$ matrices. Use this expression to find formulae for (i) $P^{(2n)}$, (ii) $P^{(2n+1)}$ in terms of the matrices A and B, and interpret your results.

4.13. Suppose that a finite Markov chain is irreducible and that there exists at least one state E_i such that $p_{ii} > 0$. Show that the Markov chain is aperiodic.

4.14. (More difficult). Any $s \times s$ matrix of non-negative numbers for which all rows sum to 1 can be regarded as the transition matrix of some Markov chain. Is it true that any such matrix can be the two-step transition matrix of some Markov chain? *Hint:* Consider the case $s = 2$. Write down the general form of a 2×2 Markov chain matrix and find when the sum of the diagonal terms is minimized.

(11.22) give

$$t_1 = p_a t_1 + p_t p_a t_2 + 1, \quad t_2 = p_a^2 t_1 + p_t p_a t_2 + 1. \tag{11.27}$$

These equations give

$$t_1 = \frac{1}{1 - p_a - p_a p_t + p_a^2 p_t - p_a^3 p_t}, \quad t_2 = \frac{1 - p_a + p_a^2}{1 - p_a - p_a p_t + p_a^2 p_t - p_a^3 p_t}.$$

Given that a clump occurs, it is natural to assume that the first word is *aaa* with probability $p_a/(p_a + p_t)$ and is *ata* with probability $p_t/(p_a + p_t)$. In this case the unconditional mean number of words in the clump is $(t_1 p_a + t_2 p_t)/(p_a + p_t)$.

11.7 Continuous-Time Markov Chains

11.7.1 Definitions

Markov processes can be in either discrete or continuous time, and in either discrete or continuous space; that is to say, there are four types of Markov processes. The Markov chains considered in Chapter 4 and so far in this chapter are in discrete time and discrete space. In this section we consider a random variable Y that takes values in some discrete space but whose values can change in continuous time. The value of Y at time t is denoted by $Y(t)$.

The "Markov" and the "time homogeneity" assumptions are defined for the continuous-time process as follows. The *Markov* assumption is that, given that $Y = i$ at any time u, the probability that $Y = j$ at any future time $u + t$ does not depend further on the values before time u. The *time homogeneity* assumption is that the conditional probability

$$\text{Prob}(Y(u + t) = j \mid Y(u) = i) \tag{11.28}$$

is independent of u, so we write it as $P_{ij}(t)$.

We focus on the case where the possible values of $Y(t)$ are 1, 2, …, s and where it is assumed that the transition probability equations (11.28) take the form

$$P_{ij}(h) = q_{ij} h + o(h), \quad j \neq i, \tag{11.29}$$

$$P_{ii}(h) = 1 - q_i h + o(h), \tag{11.30}$$

as $h \to 0$, with q_i defined by

$$q_i = \sum_{j \neq i} q_{ij}. \tag{11.31}$$

We shall call the q_{ij} the *instantaneous transition rates* of the process. If q_i is independent of i, the expression (11.30) becomes identical in form to (4.1). The justification for the choice of the mathematical forms on the right-hand sides of (11.29) and (11.30) is the same as that given in Chapter 4 leading to the expression (4.1).

In Section 14.3.1 we consider evolutionary models in which equations of the form (11.29) and (11.30) are assumed. The justification for this is, in effect, that a change at a position in a population from i to j in a small time interval of length $2h$ is approximately twice the probability that this happens in a time interval of length h. Therefore, a generalization of the argument leading to the expression (4.1) leads to equations (11.29) and (11.30). In some evolutionary models that we consider it happens that q_i is independent of i, so that the Poisson process theory of Chapter 4 may be used directly for them. When q_i depends on i we may regard the process governing the number of transitions as a generalization of the Poisson process of Chapter 4.

11.7.2 Time-Dependent Solutions

By the time-homogeneity property of the process of interest,

$$P_{ij}(t+h) = \sum_k P_{ik}(t)P_{kj}(h),$$

where the sum is taken over all possible states. From this, equations (11.29)–(11.31) imply that for fixed i,

$$P_{ij}(t+h) = P_{ij}(t)(1-q_jh) + h\sum_{k\neq j} P_{ik}(t)q_{kj} + o(h). \tag{11.32}$$

From this equation and the assumptions (11.29)–(11.31) we arrive at the following system of differential equations:

$$\frac{d}{dt}P_{ij}(t) = -q_jP_{ij}(t) + \sum_{k\neq j} P_{ik}(t)q_{kj}, \quad j = 1,2,\ldots,s. \tag{11.33}$$

These are called the *forward Kolmogorov equations* of the system. They can be solved explicitly in cases where the q_{kj} take simple forms. An example is given in Problem 11.10. Some applications in the evolutionary context where these equations can be solved are given in Sections 14.3.1, 14.3.2, and 14.3.3.

11.7.3 The Stationary Distribution

A stationary distribution $\{\varphi_j\}$ has the property that if at any time t $\text{Prob}(Y(t)=j) = \varphi_j$ for all j, then for all j, $\text{Prob}(Y(u)=j) = \varphi_j$ for all $u > t$. Thus a stationary distribution, if it exists, can be found by replacing the derivatives on the left-hand sides of the system of equations (11.33) by zero and replacing $P_{ij}(t)$ and $P_{ik}(t)$ by φ_j and φ_k, respectively, to get

$$q_j \varphi_j = \sum_{k \neq j} \varphi_k q_{kj}, \quad j = 1, 2, \ldots, s. \tag{11.34}$$

This equation is used in an evolutionary context in Sections 14.3.1, 14.3.2, and 14.3.3.

11.7.4 Detailed Balance

When a stationary distribution $\{\varphi_j\}$ exists, the detailed balance conditions analogous to the discrete-time conditions (11.5) are that for all i, j, and t,

$$\varphi_i P_{ij}(t) = \varphi_j P_{ji}(t). \tag{11.35}$$

A simpler form of this criterion is that

$$\varphi_i q_{ij} = \varphi_j q_{ji}, \tag{11.36}$$

where the q_{ij} are defined in equation (11.29).

11.7.5 Exponential Holding Times

Suppose that $Y(t) = j$. Then $Y(\cdot)$ will remain at the value j for some length of time until it changes to some value other than j, and we now find the probability density function of the time until such a change occurs.

Let T be the (random) time until the value of $Y(\cdot)$ moves to some value different from j. Then

$$\text{Prob}(T \geq t + h) = \text{Prob}(T \geq t) \cdot \text{Prob}(T \geq t + h \,|\, T \geq t).$$

From the Markov and time homogeneity properties of the process, this is

$$\text{Prob}(T \geq t + h) = \text{Prob}(T \geq t) \cdot \text{Prob}(T \geq h). \tag{11.37}$$

The probability that $T \geq h$ is the probability that the random variable has not moved from the value j before time h, and if terms of order $o(h)$ are ignored, this is the probability $1 - q_j h + o(h)$ given in equation (11.30). Thus

$$\text{Prob}(T \geq t + h) = \text{Prob}(T \geq t) \cdot (1 - q_j h) + o(h). \tag{11.38}$$

Rearrangement of terms gives

$$\frac{\text{Prob}(T \geq t + h) - \text{Prob}(T \geq t)}{h} = -q_j \,\text{Prob}(T \geq t) + \frac{o(h)}{h},$$

and the limiting operation $h \to 0$ gives

$$\frac{d}{dt} \text{Prob}(T \geq t) = -q_j \,\text{Prob}(T \geq t). \tag{11.39}$$

Standard differential equation calculations show that

$$\text{Prob}(T \geq t) = C \cdot e^{-q_j t},$$

for some constant C. The case $t = 0$ shows that $C = 1$, so that

$$\text{Prob}(T \geq t) = e^{-q_j t}. \tag{11.40}$$

Allowing for a change in notation, this is identical to equation (1.67), and thus T has an exponential distribution. From (1.54), the density function of T is

$$f(t) = q_j e^{-q_j t}, \quad t > 0. \tag{11.41}$$

To summarize, if $Y(\cdot)$ has just arrived at the value j, it next moves to some other value after a random length of time having the exponential distribution given in equation (11.41). This implies that having just arrived at the value j, the mean time spent at this value before moving to some other value is q_j^{-1}.

11.7.6 The Embedded Chain

In some applications we might not be interested in the time spent by $Y(\cdot)$ at any value but only in the sequence of values that $Y(\cdot)$ assumes. In other words we are interested only in the so-called *embedded chain* of the process. This embedded chain is a discrete-time Markov chain whose transition probabilities are

$$p_{jk} = \frac{q_{jk}}{q_j}. \tag{11.42}$$

These are conditional probabilities, derived from (1.101); given that a change occurs, the probability that it is to k is given by (11.42). Markov chain theory can be used to find properties of this embedded process. These properties can provide information about the original time-dependent process. For example, if a time-dependent process has absorbing states, the probability that the process enters a specific absorbing state is the same as the corresponding probability in the embedded chain. The latter probability might be found more easily using discrete time Markov chain theory than by the continuous time theory of this section. On the other hand it is not possible to find absorption time properties for the continuous time process from the embedded chain.

Problems

11.1. Suppose that the transition matrix P of a Markov chain is given by

$$P = \begin{bmatrix} 0.6 & 0.4 \\ 0.3 & 0.7 \end{bmatrix}. \tag{11.43}$$

Use the definitions (B.45) and (B.46) to find the (two) eigenvalues and the (two pairs of) corresponding left and right eigenvectors of P.

11.2. Use your answer to Problem 11.1 to check that equation (B.48) holds.

11.3. For the matrix P given in (11.43), find P^2 by direct matrix multiplication. Then find P^2 by using the eigenvalues and eigenvectors calculated in Problem 11.1, together with the right-hand side of equation (B.49).

11.4. Find the stationary distribution of the Markov chain whose transition matrix is given in (11.43). Find P^3 and P^4 using the spectral expansion, and thus check that P^n is approaching the matrix defined through the stationary distribution.

11.5. Suppose that a reversible Markov chain with transition matrix $P = \{p_{ij}\}$ and stationary distribution φ whose typical element is φ_i. Show that for such a matrix $\varphi_i p_{ij}^n = \varphi_j p_{ji}^n$. (Hint: this equation is true for $n = 1$: see equation (11.5)). Now use induction and the fact that

$$p_{ij}^{(n+1)} = \sum_k p_{ik}^{(n)} p_{kj} = \sum_k p_{ik} p_{kj}^{(n)}.$$

Assuming that the matrix M_1 defined in Section 6.5.3 is reversible, use this result to show that the expression in (6.24) is unchanged by reversing the roles of j and k (as is desired in the context of PAM matrices).

11.6. Use equation (11.5) to show that if the transition matrix of a finite irreducible aperiodic Markov chain is symmetric, then that Markov chain is reversible.

11.7. Suppose that a finite aperiodic irreducible Markov chain has transition probability matrix P and stationary distribution $\boldsymbol{\varphi}'$. Show that if k is any constant, $0 < k < 1$, then the Markov chain with transition probability matrix $P^* = kP + (1 - k)I$ also has stationary distribution $\boldsymbol{\varphi}'$. What interpretation or explanation can you give for this result?

11.8. Prove the assertion made below equation (11.10), that $p_{ij} > 0$ for all (i, j) including the case $i = j$.

11.9. Calculate the value of P in equation (11.26).

11.10. A simple Markov chain has two states, called here "0" and "1," with instantaneous transition rates $q_{01} = q_{10} = \alpha$. Write down the differential equation (11.33) for this case, and show that the solution is

$$P_{ii}(t) = \frac{1}{2} + \frac{1}{2}e^{-2\alpha t}, \quad P_{ij}(t) = \frac{1}{2} - \frac{1}{2}e^{-2\alpha t}, \quad (i \neq j, i, j = 0, 1). \quad (11.44)$$

Let $t \to +\infty$ to find the stationary distribution of this Markov process, and show that this distribution satisfies (11.34).

12
Hidden Markov Models

We divide this brief account of hidden Markov models into three sections: (i) a description of the properties of these models, (ii) the three main algorithms of the models, and (iii) applications. For a more complete account of these models, see Rabiner (1989).

12.1 What is a Hidden Markov Model?

A hidden Markov model (HMM) is similar to a Markov chain, but is more general, and hence more flexible, allowing us to model phenomena that we cannot model sufficiently well with a regular Markov chain model. An HMM is a discrete-time Markov model with some extra features. The main addition is that when a state is visited by the Markov chain, the state "emits" a letter from a fixed time-independent alphabet. Letters are emitted via a time-independent, but usually state-dependent, probability distribution over the alphabet. When the HMM runs there is, first, a sequence of states visited, which we denote by $q_1, q_2, q_3, \ldots$, and second, a sequence of emitted symbols, denoted by $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3, \ldots$. Their generation can be visualized as a two-step process as follows:

$$\underset{q_1}{\text{initial}} \rightarrow \underset{\mathcal{O}_1}{\text{emission}} \rightarrow \underset{\text{to } q_2}{\text{transition}} \rightarrow \underset{\mathcal{O}_2}{\text{emission}} \rightarrow \underset{\text{to } q_3}{\text{transition}} \rightarrow \underset{\mathcal{O}_3}{\text{emission}} \rightarrow \cdots$$

We denote the entire sequence of q_i's by Q and the entire sequence of $\mathcal{O}_i$'s by $\mathcal{O}$, and we write "the observed sequence $\mathcal{O} = \mathcal{O}_1, \mathcal{O}_2, \ldots$" and "the state sequence $Q = q_1, q_2, \ldots$."

Often we know the sequence $\mathcal{O}$ but do not know the sequence Q. In such a case the sequence Q is called "hidden." An important feature of HMMs is that we can efficiently answer several questions about $\mathcal{O}$ and Q.

One of these questions concerns the estimation of the hidden state sequence that has the highest probability given the observed sequence. We illustrate this with a simple example. Consider the Markov chain with two states S_1 and S_2, with uniform initial distribution and transition matrix

$$\begin{bmatrix} .9 & .1 \\ .8 & .2 \end{bmatrix}.$$

Let A be an alphabet consisting only of the numbers 1 and 2. State S_1 emits a 1 or 2 with equal probability $\frac{1}{2}$, state S_2 emits a 1 with probability $\frac{1}{4}$ and a 2 with probability $\frac{3}{4}$. Suppose the observed sequence is $\mathcal{O} = 2, 2, 2$. What sequence of states $Q = q_1, q_2, q_3$ has the highest probability given $\mathcal{O}$? In other words, what is

$$\operatorname*{argmax}_{Q} \operatorname{Prob}(Q \mid \mathcal{O})?$$

There are eight possibilities for Q. Each of these can be written down and its probability calculated, and from this it is found that the answer to the above question is $Q = S_2, S_1, S_1$. The sequence Q contains more S_1's, even though S_2 is more likely to produce a 2 when visited (probability $\frac{3}{4}$) than S_1 (probability $\frac{1}{2}$). The reason is because S_1 is much more likely to be visited than S_2 ($p_{11} = .9$ and $p_{21} = .8$).

We can also calculate

$$\operatorname{Prob}(\mathcal{O}) = \sum_{Q} \operatorname{Prob}(\mathcal{O} \mid Q) \cdot \operatorname{Prob}(Q). \tag{12.1}$$

This calculation is useful in distinguishing which of several models is most likely to have produced $\mathcal{O}$.

In the above example all of these calculations can be done by hand. However, models arising in practice have many states, sometimes hundreds, and an alphabet with many symbols (often 20, one for each amino acid). In these cases, calculation of the quantities above by exhaustive methods becomes impossible even for the fastest computers. Fortunately, there are dynamic programming approaches that overcome this problem, which we discuss in detail below. Before turning to the algorithms, however, it is necessary to introduce some specific notation. An HMM will consist of the following five components:

(1) A set of N states $S_1, S_2, \ldots, S_N$.

(2) An alphabet of M distinct observation symbols $A = \{a_1, a_2, \ldots, a_M\}$.

(3) The transition probability matrix $P = (p_{ij})$, where

$$p_{ij} = \operatorname{Prob}(q_{t+1} = S_j \mid q_t = S_i)$$

(4) The emission probabilities: For each state S_i and a in A,

$$b_i(a) = \text{Prob}(S_i \text{ emits symbol } a).$$

The probabilities $b_i(a)$ form the elements in an $N \times M$ matrix $B = (b_i(a))$.

(5) An initial distribution vector $\pi = (\pi_i)$, where $\pi_i = \text{Prob}(q_1 = S_i)$.

Components 1 and 2 describe the structure of the model, and 3–5 describe the parameters. It is convenient to let $\lambda = (P, B, \pi)$ represent the full set of parameters. We can now describe the main algorithms.

12.2 Three Algorithms

There are three calculations that are frequently required in HMM theory. Given some observed output sequence $\mathcal{O} = \mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_T$, these are:

(i) Given the parameters λ, efficiently calculate

$$\text{Prob}(\mathcal{O} \mid \lambda).$$

That is, efficiently calculate the probability of some given sequence of observed outputs.

(ii) Efficiently calculate the hidden sequence $Q = q_1, q_2, \ldots, q_T$ of states that is most likely to have occurred, given $\mathcal{O}$. That is, calculate

$$\underset{Q}{\text{argmax}} \ \text{Prob}(Q \mid \mathcal{O}).$$

(iii) Assuming a fixed topology of the model (i.e., a fixed graph structure of the underlying Markov chain, as defined in 4.9), find the parameters $\lambda = (P, B, \pi)$ that maximize $\text{Prob}(\mathcal{O} \mid \lambda)$.

We address these problems in turn.

12.2.1 The Forward and Backward Algorithms

We first consider problem (i). The naive way of calculating $\text{Prob}(\mathcal{O})$ is to use formula (12.1). This calculation involves the sum of N^T multiplications, each being a multiplication of $2T$ terms. The total number of operations is thus on the order of $2T \cdot N^T$.

Unless T is quite small, this calculation is computationally infeasible. For example, if $N = 4$, $T = 100$, the number of calculations is on the order of 10^{60}. It would take the life of the universe to make such a calculation. Fortunately, there is a much more efficient and computationally feasible procedure, called the *forward algorithm.*

The forward algorithm focuses on the calculation of the quantity

$$\alpha\,(t,i) = \text{Prob}(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3, \ldots, \mathcal{O}_t, q_t = S_i), \tag{12.2}$$

which is the joint probability that the sequence of observations seen up to and including time t is $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3, \ldots, \mathcal{O}_t$, and that the state of the HMM at time t is S_i. The $\alpha(t,i)$ are called the *forwards* variables.

Once we know $\alpha(T,i)$ for all i, then $\text{Prob}(\mathcal{O})$ can be calculated as

$$\text{Prob}(\mathcal{O}) = \sum_{i=1}^{N} \alpha(T,i). \tag{12.3}$$

We calculate the $\alpha(t,i)$'s inductively on t. The first calculation is of the *initialization* step, and uses the obvious result

$$\alpha(1,i) = \pi_i b_i(\mathcal{O}_1). \tag{12.4}$$

Next, the equation

$$\alpha(t+1,i) = \sum_{j=1}^{N} \text{Prob}(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3, \ldots, \mathcal{O}_{t+1}, q_{t+1} = S_i \text{ and } q_t = S_j)$$

leads to the *induction* step

$$\alpha(t+1,i) = \sum_{j=1}^{N} \alpha\,(t,j) p_{ji} b_i(\mathcal{O}_{t+1}). \tag{12.5}$$

This equation gives $\alpha(t+1,i)$ in terms of the $\alpha(t,j)$, so that $\alpha(t+1,i)$ can be calculated quickly once the $\alpha(t,j)$ are known. We use (12.4) to calculate $\alpha(1,i)$ for all i; then we use (12.5) to calculate $\alpha(2,i)$ for all i and again to calculate $\alpha(3,i)$ for all i, and so on, until we have obtained the $\alpha(T,i)$ for all i, needed in (12.3).

This procedure provides an algorithm for the solution to problem (i). The algorithm requires on the order of TN^2 computations, and thus is feasible in practice, even for very large models.

Before going on to problem (ii), we consider briefly the *backward* part of the forward–backward algorithm. This provides another approach to solving problem (i), but we introduced it because we will use the "backwards" variables when we discuss problem (iii).

In the above, we calculated successively $\alpha(1,\cdot), \alpha(2,\cdot), \ldots, \alpha(T,\cdot)$, that is, we calculated forward in time. In the backward algorithm we calculate another quantity backwards in time, as the name suggests. We do not use these quantities to solve (12.3): instead, we shall need them for a later calculation. The goal of the backwards algorithm is to calculate the probability $\beta(t,i)$, defined by

$$\beta(t,i) = \text{Prob}(\mathcal{O}_{t+1}, \mathcal{O}_{t+2}, \ldots, \mathcal{O}_T \mid q_t = S_i), \tag{12.6}$$

for $1 \leq t \leq T-1$, and for convenience we define $\beta(T, j)$ to be 1 for all j. We then calculate (12.6) working backwards from $t = T-1$. The relevant equations for this procedure are

$$\beta(t-1, i) = \sum_{j=1}^{N} p_{ij}\, b_j(\mathcal{O}_t)\beta(t, j). \tag{12.7}$$

Using these equations we can successively calculate $\beta(T-1, i)$ for all i, $\beta(T-2, i)$ for all $i, \ldots,$ and $\beta(1, i)$ for all i.

12.2.2 The Viterbi Algorithm

Given some observed sequence $\mathcal{O} = \mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3, \ldots, \mathcal{O}_T$ of outputs, we want to compute efficiently a state sequence $Q = q_1, q_2, q_3, \ldots, q_T$ that has the highest conditional probability given $\mathcal{O}$. In other words, we want to find a Q that makes $\text{Prob}(Q \,|\, \mathcal{O})$ maximal, that is, we want to calculate

$$\operatorname*{argmax}_{Q}\ \text{Prob}(Q \,|\, \mathcal{O}). \tag{12.8}$$

There may be many Q's that maximize $\text{Prob}(Q \,|\, \mathcal{O})$. We give an algorithm that finds one of them. It can easily be generalized to find them all. However, for our applications this generalization will not be necessary.

The Viterbi algorithm carries out the efficient computation of (12.8). The algorithm is divided into two parts. It first finds $\max_Q \text{Prob}(Q \,|\, \mathcal{O})$, and then "backtracks" to find a Q that realizes this maximum. This is another dynamic programming algorithm.

First define, for arbitrary t and i,

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} \text{Prob}(q_1, q_2, \ldots, q_{t-1}, q_t = S_i \text{ and } \mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3, \ldots, \mathcal{O}_t)$$

($\delta_1(i) = \text{Prob}(q_1 = S_i \text{ and } \mathcal{O}_1)$). In words, $\delta_t(i)$ is the maximum probability of all ways to end in state S_i at time t and have observed sequence $\mathcal{O}_1$, $\mathcal{O}_2, \ldots,$ $\mathcal{O}_t$. Then

$$\max_{Q}\ \text{Prob}(Q \text{ and } \mathcal{O}) = \max_{i}\ \delta_T(i). \tag{12.9}$$

The probability in this expression is the joint probability of Q and $\mathcal{O}$, not a conditional probability. Our aim is to find a sequence Q for which the maximum conditional probability (12.8) is achieved. Since

$$\max_{Q}\ \text{Prob}(Q \,|\, \mathcal{O}) = \max_{Q} \frac{\text{Prob}(Q \text{ and } \mathcal{O})}{\text{Prob}(\mathcal{O})},$$

and since the denominator on the right-hand side does not depend on Q,

$$\operatorname*{argmax}_{Q}\ \text{Prob}(Q \,|\, \mathcal{O}) = \operatorname*{argmax}_{Q} \frac{\text{Prob}(Q \text{ and } \mathcal{O})}{\text{Prob}(\mathcal{O})} = \operatorname*{argmax}_{Q}\ \text{Prob}(Q \text{ and } \mathcal{O}).$$

The first step is to calculate the $\delta_t(i)$'s inductively. Then we will "backtrack" and recover the sequence that gives the largest $\delta_T(i)$. The *initialization* step is

$$\delta_1(i) = \pi_i b_i(\mathcal{O}_1), \quad 1 \leq i \leq N. \tag{12.10}$$

The induction step is

$$\delta_t(j) = \max_{1\leq i\leq N} \delta_{t-1}(i) p_{ij} b_j(\mathcal{O}_t), \quad 2 \leq t \leq T,\ 1 \leq j \leq N. \tag{12.11}$$

We recover the q_i's as follows. Define

$$\psi_T = \operatorname*{argmax}_{1\leq i\leq N} \delta_T(i),$$

and put $q_T = S_{\psi_T}$. Then q_T is the final state in the state sequence required. The remaining q_t for $t \leq T-1$ are found recursively by first defining

$$\psi_t = \operatorname*{argmax}_{1\leq i\leq N} \delta_t(i) p_{i\psi_{t+1}},$$

and then putting $q_t = S_{\psi_t}$. If the argmax is not unique, we arbitrarily take one value of i giving the maximum.

12.2.3 The Estimation Algorithms

We now address problem (iii). Suppose we are given a set of observed data from an HMM for which the topology is known (by topology we mean the graph structure of the underlying Markov model). We wish to try to estimate the parameters in that HMM. The parameter space is usually far too large to allow exact calculation of a set of parameter estimates that maximizes the probability of the data. Instead, we employ algorithms that find "locally" best sets of parameters. This partial solution to the problem has proven to be useful in many applications.

The focus on local estimation means that the procedure is heuristic. Therefore, the efficacy of the procedure must evaluated empirically by using benchmarks and test sets for which there are known outcomes. This matter is discussed further below.

Some additional comments are in order. It is not necessary to assume that the data come from an HMM. Instead, it is usually more accurate to assume that the data are generated by some random process that we try to "fit" with an HMM. Sometimes it might be possible to achieve a tight fit with an HMM and sometimes it might not.

The discussion above shows that we should use the term "estimation" of parameters cautiously in this section. Our aim is to "set" parameters at values providing a good fit to data rather than to estimate parameters in the sense of Chapter 8.

We now describe the Baum–Welch method of parameter estimation. This is a difficult algorithm, so we do not provide proofs of the claims made but instead indicate the intuition behind the method.

We assume that the alphabet A and number of states N is fixed at the outset, and that the parameters π_i, p_{jk}, and $b_i(a)$ are unknown and are to be "estimated." The data we use to estimate the parameters constitute a set of observed sequences $\{\mathcal{O}^{(d)}\}$. Each observed sequence $\mathcal{O}^{(d)} = \mathcal{O}_1^{(d)}, \mathcal{O}_2^{(d)}, \ldots$ has a corresponding hidden state sequence $Q^{(d)} = q_1^{(d)}, q_2^{(d)}, \ldots$.

The procedure starts by setting the parameters π_i, p_{jk}, and $b_i(a)$ at some initial values. These can be chosen from some uniform distribution or can be chosen to incorporate prior knowledge about them. We then calculate, using these initial parameter values,

$$\overline{\pi}_i = \text{the expected proportion of times in state } S_i \text{ at the first time point, given } \{\mathcal{O}^{(d)}\}, \tag{12.12}$$

$$\overline{p}_{jk} = \frac{E(N_{jk} \mid \{\mathcal{O}^{(d)}\})}{E(N_j \mid \{\mathcal{O}^{(d)}\})}, \tag{12.13}$$

$$\overline{b}_i(a) = \frac{E(N_i(a) \mid \{\mathcal{O}^{(d)}\})}{E(N_i \mid \{\mathcal{O}^{(d)}\})}, \tag{12.14}$$

where N_{jk} is the (random) number of times $q_t^{(d)} = S_j$ and $q_{t+1}^{(d)} = S_k$ for some d and t; N_i is the (random) number of times $q_t^{(d)} = S_i$ for some d and t; and $N_i(a)$ equals the (random) number of times $q_t^{(d)} = S_i$ and it emits symbol a, for some d and t. The expected values in (12.13) and (12.14) are conditional expected values, as defined in (2.59).

We show how to calculate these efficiently below. These are the "re-estimation" parameter values that then replace π_i, p_{jk}, and $b_i(a)$. These values follow the form of estimation used, for example, in equation (3.10). The algorithm proceeds by iterating this step.

It can be shown that if $\lambda = (\pi_i,\ p_{jk},\ b_i(a))$ is replaced by $\overline{\lambda} = (\overline{\pi}_i,\ \overline{p}_{jk},\ \overline{b}_i(a))$, then $\text{Prob}(\{\mathcal{O}^{(d)}\} \mid \overline{\lambda}) \geq \text{Prob}(\{\mathcal{O}^{(d)}\} \mid \lambda)$, with equality holding if and only if $\overline{\lambda} = \lambda$. Thus successive iterations continually increase the probability of the data, given the model. Iterations continue until either a local maximum of the probability is reached or until the change in the probability becomes negligible.

In order to discuss the calculations needed for (12.13)–(12.12), define $\xi_t^{(d)}(i,j)$ by

$$\xi_t^{(d)}(i,j) = \text{Prob}(q_t^{(d)} = S_i, q_{t+1}^{(d)} = S_j \mid \mathcal{O}^{(d)}), \tag{12.15}$$

where $i, j = 1, \ldots, N$, and $t \geq 1$. The conditional probability formula (1.101) shows that this is equal to

$$\frac{\text{Prob}(q_t^{(d)} = S_i, q_{t+1}^{(d)} = S_j, \mathcal{O}^{(d)})}{\text{Prob}(\mathcal{O}^{(d)})}.$$

The denominator is $\text{Prob}(\mathcal{O}^{(d)})$ and is thus calculated efficiently using the methods of Section 12.2.1. The numerator is calculated efficiently by writing it in terms of the forwards and backwards variables discussed in Section 12.2.1,

$$\text{Prob}(q_t^{(d)} = S_i, q_{t+1}^{(d)} = S_j, \mathcal{O}^{(d)}) = \alpha_t(i) p_{ij} b_j(\mathcal{O}_{t+1}^{(d)}) \beta_{t+1}(j). \tag{12.16}$$

Let $I_t^{(d)}(i)$ be the indicator variables defined by

$$I_t^{(d)}(i) = \begin{cases} 1, & \text{if } q_t^{(d)} = S_i, \\ 0, & \text{otherwise.} \end{cases}$$

The number of times S_i is visited is then $\sum_d \sum_t I_t^{(d)}(i)$. The expected number of times S_i is visited, given $\{\mathcal{O}^{(d)}\}$, is then

$$\sum_d \sum_t E(I_t^{(d)}(i) \,|\, \mathcal{O}^{(d)}). \tag{12.17}$$

Now $E(I_t^{(d)}(i) \,|\, \mathcal{O}^{(d)})$ is $\text{Prob}(q_t^{(d)} = S_i \,|\, \mathcal{O}^{(d)})$, which is

$$\sum_{j=1}^{N} \xi_t^{(d)}(i,j). \tag{12.18}$$

Thus the expected number of times S_i is visited, given $\{\mathcal{O}^{(d)}\}$, is

$$\sum_d \sum_t \sum_{j=1}^{N} \xi_t^{(d)}(i,j).$$

Similarly, the expected number of transitions from S_i to S_j given $\{\mathcal{O}^{(d)}\}$ is

$$\sum_d \sum_t \xi_t^{(d)}(i,j).$$

These expressions give efficient formulae to calculate all the quantities in equations (12.12)–(12.14) except the numerator of (12.14). This is calculated as follows.

Define the indicator random variables $I_t^{(d)}(i,a)$ by

$$I_t^{(d)}(i,a) = \begin{cases} 1, & \text{if } q_t^{(d)} = S_i \text{ and } \mathcal{O}_t^{(d)} = a, \\ 0, & \text{otherwise.} \end{cases}$$

Then $E(I_t^{(d)}(i,a) \,|\, \mathcal{O}^{(d)})$ is the expected number of times the dth process is in state S_i at time t and emits symbol a, given $\mathcal{O}^{(d)}$. The numerator of (12.14) is equal to $\sum_d \sum_t E(I_t^{(d)}(i,a) \,|\, \mathcal{O}^{(d)})$, which is

$$\sum_d \sum_t \sum_{\mathcal{O}_t^{(d)}=a} \sum_{j=1}^{N} \xi_t^{(d)}(i,j).$$

12.3 Applications

We sketch here the applications of HMMs in several different areas of computational biology. Only a brief outline of each application is given: further details may be found in the references provided.

12.3.1 Modeling Protein Families

In this section we develop an HMM to model protein families, and we shall use the model for two purposes: to construct multiple sequence alignments and to determine the family of a query sequence. These applications were first presented in Krogh et al. (1994). In order to present the main ideas we simplify many of the details.

Figure 12.1 gives an example of the basic type of HMM we shall use. This example has "length" five; any length is possible. The underlying Markov model is presented in graphical form (as in Section 4.9). The states are

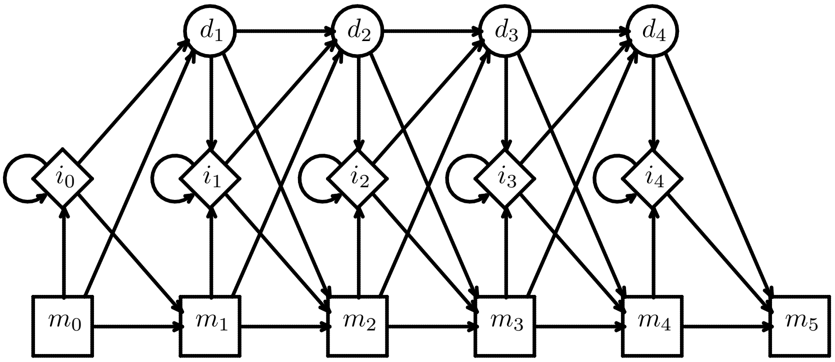


Figure 12.1. Hidden Markov model for a protein family.

the squares, diamonds, and circles labeled m_0, $m_1, \ldots,$ m_5, i_0, $i_1, \ldots,$ i_4, and d_1, $d_2, \ldots,$ d_4, respectively. The squares are called the *match* states, the diamonds the *insert* states, and the circles the *delete* states. The edges not shown have transition probability zero. State m_0 is the *start* state, so that the process always starts in state m_0. A transition never moves to the left, so that as time progresses the current state gradually moves to the right, eventually ending in match state m_5, the *end* state. When this state is reached the process ends. A match or delete state is never visited more than once.

The alphabet A consists of the twenty amino acids together with one "dummy" symbol representing "delete" (denoted δ). Delete states output

δ with probability one. Each insert and match state has its own distribution over the 20 amino acids, and cannot emit a δ. That is, only a delete state can emit a δ, and each delete state emits *only* δ.

If the emission probabilities for the match and insert states are uniform over the 20 amino acids, the model will produce random sequences that do not have much in common except possibly their lengths. At the other extreme, if each state emits one specific amino acid with probability one, and if further the transitions from m_i to m_{i+1} have probability one, then the model will always produce the same sequence. Somewhere in between these two extremes the parameters of the model can be set so that it produces sequences that are similar, thus producing what can be thought of as a "family" of sequences. Each choice of parameters produces a different family. This family can be rather "tight," meaning all sequences in it are very similar, or can be "loose," so that there is little similarity between the sequences produced. It is also possible that the similarity is high in some positions of the sequences produced and low in others. This will happen if some match states have distributions concentrated on a few amino acids while the others have distributions in which all amino acids are approximately equally likely. By contrast, the dynamic programming sequence alignment algorithms and BLAST allow one gap open penalty and use one substitution matrix uniformly across the entire length of the sequences compared. Allowing gap penalties and substitution probabilities to vary along the sequences reflects biological reality better. Alignments of related proteins generally have regions of higher conservation and regions of lower conservation. The regions of higher conservation are called functional domains, because their resistance to change indicates that they serve some critical function. Dynamic programming alignment and BLAST are essential for certain applications, such as pairwise alignments, or aligning a small number of sequences. But for modeling large families of sequences, or constructing alignments of many sequences, HMMs allow for efficiency, and at the same time exploit the larger data sets to increase flexibility.

In the HMM model of a protein family the transition (arrow) from a match state to an insert state corresponds to the gap open penalty, and the arrow from an insert state to itself corresponds to the gap extension penalty. Loosely speaking, the distribution over the amino acids for any state takes the place of a substitution matrix. The probabilities in the model can differ from position to position in the sequence, since each arrow has its own probability and each match and insert state has its own distribution. Thus the HMM model is sufficiently flexible to model the varying features of a protein along its length. While the model can be made even more flexible by adding further parameters, more data are needed to estimate these parameters effectively. The model described has proven in practice to provide a good compromise between flexibility and tractability. Such HMM models of are called *profile* HMMs.

All applications start with *training*, or estimating, the parameters of the model using a set of training sequences chosen from a protein family, such as the set of all globins in GenBank. This estimation procedure uses the Baum–Welch algorithm. The model is chosen to have length equal to the average length of a sequence in the training set, and all parameters are initialized by using uniform distributions (i.e., amino acids are given probability $\frac{1}{20}$, and transitions of the same type are given $\frac{1}{2}$ equal probabilities).[1]

12.3.2 Multiple Sequence Alignments

In this section we describe how to use the theory described above to compute multiple sequence alignments for a family of sequences. The sequences to be aligned are used as the training data, to train the parameters of the model. For each sequence the Viterbi algorithm is then used to determine a path most likely to have produced that sequence. These paths can then be used to construct an alignment. Amino acids are aligned if both are produced by the same match state in their paths. Indels are then inserted appropriately for insertions and deletions.

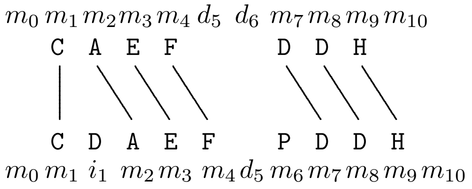
We illustrate this with an example. Consider the sequences `CAEFDDH` and `CDAEFPDDH`. Suppose the model has length 10 and their most likely paths through the model are

$$m_0 m_1 m_2 m_3 m_4 d_5 d_6 m_7 m_8 m_9 m_{10}$$

and

$$m_0 m_1 i_1 m_2 m_3 m_4 d_5 m_6 m_7 m_8 m_9 m_{10},$$

respectively. Then the alignment induced is found by aligning positions that were generated by the same match state:


$m_0\ m_1\ m_2 m_3\ m_4\ d_5\ \ d_6\ \ m_7\ m_8\ m_9\ m_{10}$
C A E F D D H
C D A E F P D D H
$m_0\ m_1\ i_1\ \ m_2\ m_3\ \ m_4\ d_5\ m_6\ m_7\ m_8\ m_9\ m_{10}$

This leads to the alignment

```
C–AEF–DDH
CDAEFPDDH .
```

[1]What Krogh et al. (1994) do is somewhat more complicated. They allow the length of the model to change along with the parameters after each iteration of the re-estimation algorithm. They also must adjust the Baum–Welch algorithm from how we have described it, in order to handle the delete states. The reader interested in implementing these applications should refer to the literature for further details.

More generally, suppose we have the five sequences

```
CAEFTPAVH,
CKETTPADH,
CAETPDDH,
CAEFDDH,
CDAEFPDDH,
```

and the corresponding paths returned by the Viterbi algorithm are

$$\begin{gathered} m_0m_1m_2m_3m_4m_5m_6m_7m_8m_9m_{10}, \\ m_0m_1m_2m_3m_4m_5m_6m_7m_8m_9m_{10}, \\ m_0m_1m_2m_3d_4m_5m_6m_7m_8m_9m_{10}, \\ m_0m_1m_2m_3m_4d_5d_6m_7m_8m_9m_{10}, \\ m_0m_1i_1m_2m_3m_4d_5m_6m_7m_8m_9m_{10}. \end{gathered}$$

Then the induced alignment is

```
C-AEFTPAVH
C-KETTPADH
C-AE-TPDDH .
C-AEF--DDH
CDAEF-PDDH
```

This technique can give ambiguous results in some cases. For example, if the model has length two and the sequences ABAC and ABBAC had

$$m_0m_1i_1i_1m_2m_3 \text{ and } m_0m_1i_1i_1i_1m_2m_3$$

as paths, then the leading A's and trailing C's will be aligned, but it is not clear how to align the BA from the first sequence to the BBA from the second. In such cases Krogh et al. (1994) represent the ambiguous symbols with lowercase letters and do not attempt to give alignments of these regions.

This technique can be used to align many sequences with relatively little computing power. By contrast, dynamic programming algorithms cannot in practice align 50 or 100 long sequences. This is the value of a heuristic approach. Another advantage of the method is that it allows the sequences themselves to guide the alignment, rather than having a precomputed substitution matrix and gap penalties. Thus less bias should be introduced.

Krogh et al. (1994) tested this method on a family of 625 globin sequences. They used a published alignment of seven of these sequences constructed using knowledge of the three-dimensional structure of the sequences (Bashford et al. (1987)). (An alignment using a three-dimensional structure is considered reliable and so serves as a benchmark for testing multiple alignment algorithms.) They chose 400 of the 625 globins to train the model and then used the Viterbi algorithm to align all 625. These alignments were then compared to the induced alignment on the seven sequences from the Bashford alignment. The alignments agreed extremely well.

12.3.3 Pfam

Pfam is a web-based resource maintained by the Sanger Center (web URL (`http://www.sanger.ac.uk/Pfam/`). Pfam uses the basic theory described above to determine protein domains in a query sequence. A protein usually has one or more functional domains, namely portions of the protein that have essential function and thus have low tolerance for amino acid substitutions. Proteins in different families often share high homology in one or more domains. Entire protein families can be characterized by HMMs, as in the previous section, or one can characterize just functional domains. Pfam focuses on the latter.

Suppose that a new protein is obtained for which no information is available except the raw sequence. We wish to "annotate" this sequence. Annotation is the process of assigning to a sequence biologically relevant information, such as where the functional domains are, what their homology is to known domains, and what their function is. The typical starting point is a BLAST search. This will return all sequences in the chosen databases that have significant similarity to the query sequence. BLAST can return many such sequences. Though this is an important step in the annotation process, it is also desirable to have a database not of protein sequences themselves, but of protein domains. Pfam is not the first such database; however, previous domain databases do not use methods as flexible as HMMs and consequently tend not to model entire domains, but rather only the most highly conserved "motifs" that can be put in *ungapped* multiple sequence alignments. The use of HMMs allows for more effective characterization of full domains.

The domains in Pfam are determined based on expert knowledge, sequence similarity, and other protein family databases. Currently, Pfam contains 2,008 protein domains. For each domain a set of examples of this domain is selected. The sequences representing each domain are put into an alignment, and the alignments themselves are used to set the parameters; that is, Baum–Welch is not used. Recall that an alignment implies for each sequence in the alignment a path through the HMM, as described in the previous section. The proportion of times these paths take a given transition is used to estimate the transition probabilities, and likewise for the emission probabilities. These alignments are called "seed alignments" and are stored in the database. Given the HMMs for all of the domains, a query sequence is then run past each one using the forward algorithm. When a portion of the query sequence has probability of having been produced by an HMM above a certain cutoff, the domain corresponding to that HMM is reported. Furthermore, the sequence can be aligned to the seed alignment using the Viterbi algorithm as described above. For more details, see the Pfam web site.

12.3.4 Gene Finding

Genomic sequences with lengths on the order of many millions of bases are now being produced, and the sequences of entire chromosomes are becoming available. Such sequences consist of a collection of genes separated from each other by long stretches of nonfunctional sequence. It is of central importance to find where the genes are in the sequence. Therefore, computational methods that quickly identify a large proportion of the genes are very useful. The problem involves bringing together a large amount of diverse information, and there have been many approaches to doing this. Currently, a popular and successful gene finder for *human* DNA sequences is GENSCAN (Burge et al. (1997)), which is based on a generalization of hidden Markov models. We sketch below an algorithm similar in spirit to that in GENSCAN in order to illustrate the basic concept of an HMM human gene finder. To increase the accuracy of the procedure it is necessary to introduce many details that we do not describe here. The interested reader is encouraged to read Burge et al. (1997) and Burge (1997).

Semihidden Markov Models

Suppose that, in an HMM, p is the probability of the transition from any state to itself. The probability that the process stays in this state for n steps is $p^{n-1}(1-p)$, so that the length of time the process stays in that state follows a geometric distribution. For the gene model we construct, it is necessary to allow other distributions for this length. In a *semi-hidden Markov Model* (semiHMM, more logically called a hidden semi-Markov model) all transition probabilities from a state to itself are zero, and when the process visits a state it produces not just a single symbol from the alphabet but rather an entire sequence. The length of the sequence can follow any distribution, and the model generating the sequence of that length can be any distribution. The positions in the sequences emitted from a state need not be iid.

The model is formulated more precisely as follows. Each state S has associated with it a random variable L_S (L for "length") whose range is a subset of $0, 1, 2, \ldots$, and for each observable value ℓ of L_S there is a random variable $Y_{S,\ell}$ whose range consists of all sequences of length ℓ. When state S is visited a length ℓ is determined randomly from the distribution for L_S. Then the distribution for $Y_{S,\ell}$ is used to determine a sequence of length ℓ. Then a transition is taken to a new state and the process is repeated, generating another sequence. These sequences are concatenated to create the final output sequence of the semiHMM.

The algorithms involved in this model are an order of magnitude more complex than for a regular HMM, since given an observed output sequence not only do we not know the path of states that produced it, we also do not know the division points in the sequence indicating where a transition was made to a new state. The gene-finding application requires a generalization of the Viterbi algorithm. There is a natural generalization. However, since one is generally working with very long sequences, the natural generalization does not run in reasonable time. In practice, further assumptions must be made. Burge (1997) observed that if the lengths of the long intergenic regions can be taken as having geometric distributions, and if these lengths generate sequences in a relatively iid fashion, then the algorithm can be adjusted so that practical running times can be obtained. These assumptions are not unreasonable in our case, and so they should not greatly affect the accuracy of the predictions. We shall omit the technical details surrounding this issue. Our goal is to convey the main idea of how an HMM gene finder works.

A *parse* ϕ is a sequence of states $q_1, q_2, \ldots, q_r$ and a sequence of lengths $d_1, d_2, \ldots, d_r$. Given an observed sequence s from a semiHMM, the Viterbi algorithm finds an optimal parse ϕ_{opt} such that $\text{Prob}(\phi_{\text{opt}} \mid s) \geq \text{Prob}(\phi \mid s)$ for all parses ϕ. In other words, ϕ_{opt} is a parse that is most likely to have given rise to the sequence s. As we will see, the optimal parse gives the gene predictions.

Gene Structure

We now outline the basic properties of human genes that are to be captured in the model. The statistical aspects arise because (1) characteristics shared by genes have similar but not identical properties and (2) signals that genes share can also exist randomly in the non-gene sequence. This issue has also been discussed in Section 5.3.

A gene consists mainly of a continuous sequence of the DNA that is copied, or "transcribed," into RNA, called "premessenger" RNA or pre-mRNA. This pre-mRNA consists of an alternating sequence of exons and introns. After transcription the introns are edited out of the pre-mRNA, and the final molecule, called "messenger RNA" or mRNA, is translated into protein. There can be some other editing and processing of an mRNA before translation. However, that will not be important for our purposes.

The region of the DNA before the start of the transcribed region is called the *upstream region*. This is where the *promoter* of the gene is, the region where certain specialized proteins bind and initiate transcription. There are different definitions of what constitutes the promoter region; often it is taken to be the 500 bases before the start of transcription. Here we shall be interested in only about 40 bases upstream from the start of transcription, since specific signals in the promoter region are extremely complex and are not well characterized. Our model, and the model used by Burge (1997), uses the so-called TATA box, which is a fairly common signal (approximately 70% of genes contain this signal), which is located 28–34 bases upstream from the start of transcription. We do not try to capture any other signal in the promoter region. For those genes without a TATA box, we rely on identifying the gene by the other signals in its transcribed region. The $5'$ *untranslated region* ($5'$UTR) follows the promoter. This is a stretch of DNA that does not get translated into protein. We call the first 8 bases of this region the *cap end* of the $5'$UTR. Near the other end of the $5'$UTR, just before the start codon in the first exon, is a signal that indicates the start of translation, called the translation initiation signal. We shall refer to the 18 bases just before the start codon as the *translation initiation end* (TIE) of the $5'$UTR. This is followed either by a single exon or by a sequence of exons separated by introns. An intron may break a codon anywhere between its three nucleotides. Each intron has signals indicating its beginning and end. Modeling these signals well is crucial for correctly predicting the intron/exon structure. Following the final exon is the $3'$ *untranslated region* ($3'$UTR), which is another stretch of sequence that is transcribed but not translated. Near the end of the $3'$UTR are one or more *Poly–A* signals signaling the end of transcription. A Poly–A signal is 6 bases long with the typical sequence AATAAA.

The Training Data

Each state of the model we construct is a model in its own right. It is necessary to train each state to produce sequence that models the corresponding part of an actual gene. To do this we start with a large set of training data consisting of long stretches of DNA where the gene structures have been completely characterized. Burge et al. (1997) compiled 2.5 million bases (Mb) of human DNA with 380 genes, consisting of 142 single-exon genes and a total of 1,492 exons and 1,254 introns. Many of these are complete genes consisting of both the upstream and downstream regions. In addition to this they included the coding region only (no introns) of 1,619 human genes.

The Model

We model a $5'$ to $3'$ oriented gene with a 13 state semiHMM as shown in Figure 12.2. The first row represents the intergenic region. The second row represents the promoter. The third row is the $5'$UTR. The fourth row of five states represents the introns and exons. The final row is the $3'$UTR and the Poly–A signal.

We first describe how each state is trained by giving the distributions of L_S and $Y_{S,\ell}$. We then discuss how the transition probabilities are set. The intergenic region between genes is labeled N in Figure 12.2. For completely uncharacterized sequence it is reasonable to assume that genes are randomly distributed. The number of genes in any given stretch of this uncharacterized sequence is therefore modeled by a Poisson distribution,

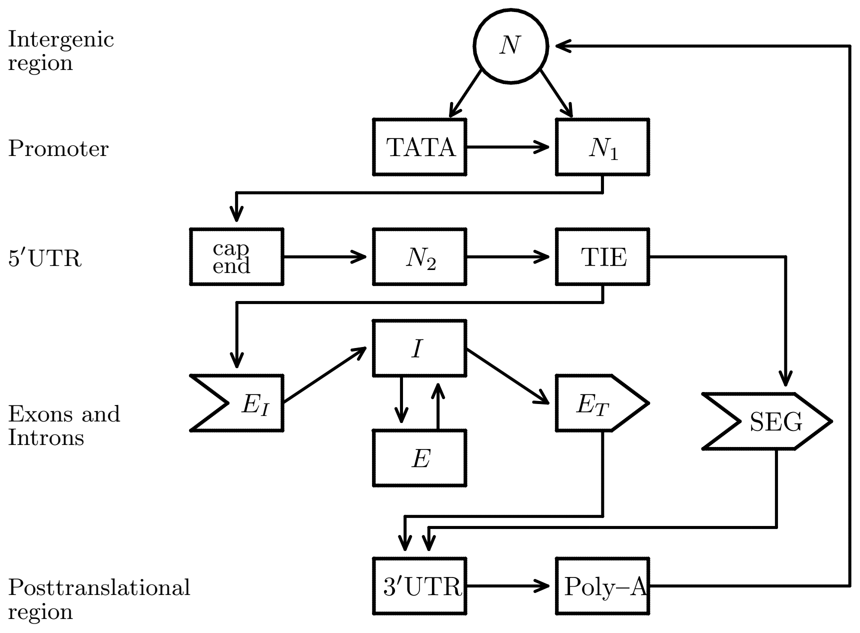


Figure 12.2.

and the distance between genes is modeled by an exponential distribution. Hence we model the length L_N with a geometric distribution (since it is the discrete analogue of the exponential) with mean equal to the size of the human genome (approximately 3 billion nucleotides) divided by the number of genes (approximately 60,000). Given an observed value ℓ of L_N, the sequence of length ℓ is generated by a fifth-order Markov model (as described in Section 11.3). The parameters of this model are set using the noncoding portions of the genes in the training set. This fifth-order model has $3 \cdot 4^5 = 3072$ parameters. Training so many parameters is possible only because there are millions of bases of DNA in the training set. There is known to be at least fifth-order Markov dependence in such noncoding DNA. A fifth-order Markov chain model is used in particular because it provides improved performance over lower-order models and because a higher-order model cannot be trained with the current training sets available. This model for producing sequence will be used for several of the states (those denoted by N_1 and N_2, as well as the entire 3′UTR), and will be referred to as the *intergenic null model.*

The TATA box is modeled with a 15-base weight matrix similar to that described in Section 5.3.2. The path can bypass the TATA box, which is necessary because only about 70% of genes have them.

The state following the TATA box produces a sequence following the null intergenic model. The length of the sequence L_{N_1} follows a uniform distribution from 28 to 34 bases.

The cap end state models the signal at the start of transcription. There has been shown to be information content in at least the first few bases of this sequence, and it is modeled with an 8-base weight matrix derived from the training data.

State N_2 is modeled with the intergenic null model with length given by a geometric distribution with mean 735 bases. This state is the main part of the 5′UTR.

The state labeled TIE contains the translation initiation signal. This state is modeled by a 18-base weight matrix derived from the training data.

The states on the next row are the exons and introns. There are two paths that can be taken through this row. The state on the right corresponds to single exon genes (SEG). It is necessary to give single exon genes their own state because the distribution of their lengths is quite different from the lengths of the exons of multiexon genes. The generation of codons is not, however, different from the multiexon genes. They both use a fifth-order Markov model, but in contrast to the intergenic region, this Markov model is nonhomogeneous. For each position in the codon there is a separate model. So in effect there are three times as many parameters as in the homogeneous intergenic model. This is partly why the training set was supplemented with the coding region of 1619 complete genes (with no introns), which gave approximately 2.3 million bases of coding sequence. The SEG generates a sequence starting with the start codon *atg* and ending with one of the three stop codons *taa, tag, tga*, chosen in accordance with their observed frequencies in the training set. The intervening codons are produced according to the fifth-order nonhomogeneous Markov model. The length of the single-exon gene is taken from an empirical distribution from the training set.

The situation with multiexon genes is more complicated. One complication arises from the fact that a single codon may be split between two exons. To handle this, the exon states in the model only produce a sequence that has length that is a multiple of three. The intron state produces one codon and then a further random number, either 0, 1, or 2, with probability $\frac{1}{3}$ of each. If this random number is 1, it places the first nucleotide at the beginning and the last two at the end; if 2, it places the first two nucleotides at the beginning and the last one at the end; and if it is 0, it does not include the codon at all.[2]

[2]Burge handles this in a different way, by creating three internal exon and three intron states, each corresponding to a different "phase." Due to the technical nature of the algorithms, that method may be more effective than the one described here. This method does, however, require that the model be complicated substantially in order to

Empirical distributions from the training set are used for the lengths of the initial, internal, and terminal exons. Intron length is modeled with a geometric distribution. The codons in the exons are modeled with the same fifth-order nonhomogeneous Markov model as the single-exon genes. The intron sequence is generated by the intergenic null model, except at the ends. At the beginning of an intron is a signal called the *donor splice signal*, and at the end is the *acceptor splice signal*. Each of these tells the machinery in the cell where to splice out the intron during the editing process. It is important to model these signals well in order to predict intron/exon structure correctly. The donor signal is taken to be the first six nucleotides of the intron, and the acceptor signal is the last 20. A weight matrix or first-order Markov approach is generally insufficient to capture these signals effectively. Ideally, we would like to have a complete joint probability distribution for these sequences. However, there is insufficient data to do this. For the donor signal we instead use the maximal dependence decomposition discussed in Section 5.3.4, and for the acceptor signal we use a second-order Markov model.[3] All of this is incorporated into the single intron state I.

The $3'$UTR is modeled with the intergenic null model, with geometric length of mean approximately 450. The Poly–A signal has constant length 6 bases and is modeled with a weight matrix.

We now turn to the issue of assigning probabilities to the transitions. Most of the transitions have probability one. The exception is the transitions from N, from TIE, and from I. Since approximately 70% of genes have a TATA box, the transition probability from N to TATA is 0.7 and to N_1 is 0.3. The transition probability from the state labeled TIE to the state labeled SEG is taken from the proportion of single-exon genes. The transitions from the intron state I are also taken from the appropriate proportions observed in the training data.

With the model so defined, given an uncharacterized sequence of DNA, we apply the Viterbi algorithm to obtain an optimal parse. The parse gives a list of the states visited and the lengths of the sequences generated at those states. We thus get a decomposition of the original sequence into gene predictions, as well as predictions of complete gene structure for each predicted gene.

In practice, there are many more considerations in optimizing the performance. Perhaps one of the most important is that many of the probabilities that have been estimated depend on the *cg* content of the region of the DNA

avoid technically violating the semiHMM assumption. We made an effort here to give a model that satisfies the definition of a semiHMM, at the likely cost of some degree of accuracy.

[3]Burge models the donor splice signal also as dependent on the last three bases of the exon. Again, this violates the definition of semiHMM, so we have not included these dependencies in our model.

being searched. For example, regions with high *cg* content tend to contain a significantly higher density of genes. Burge's model takes this and several other factors into careful consideration, and the model continues to be refined as more and different types of data become available.

Problems

12.1 Define an HMM λ with the following parameters:

Three states, S_1, S_2, S_3, alphabet $A = \{1, 2, 3\}$,

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

$$\pi = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix},$$

$b_1(1) = \frac{1}{2}$, $b_1(2) = \frac{1}{2}$, $b_1(3) = 0$,
$b_2(1) = \frac{1}{2}$, $b_2(2) = 0$, $b_2(3) = \frac{1}{2}$,
$b_3(1) = 0$, $b_3(2) = \frac{1}{2}$, $b_3(3) = \frac{1}{2}$.

What are all possible state sequences for the following observed sequences $\mathcal{O}$, and what is $p(\mathcal{O} \mid \lambda)$?

(a) $\mathcal{O} = 1, 2, 3$.
(b) $\mathcal{O} = 1, 3, 1$.

12.2 Given an HMM λ, suppose $\mathcal{O} = \mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_T$ is an observed sequence with hidden state sequence $q_1, q_2, \ldots, q_T$. For $t = 1, 2, \ldots, T$, let $\sigma_t = S_j$m where

$$j = \underset{i}{\text{argmax}}\ \text{Prob}(\mathcal{O}_t \mid q_t = S_i).$$

In other words, σ_t is the state most likely to produce symbol $\mathcal{O}_t$. Construct an HMM λ with uniform initial distribution, three states, and an alphabet of size three, and give an observed sequence of length two, $\mathcal{O} = \mathcal{O}_1\mathcal{O}_2$, for which

$$\text{Prob}(\mathcal{O} \mid \lambda) > 0, \text{ but } \text{Prob}(q_1 = \sigma_1, q_2 = \sigma_2 \mid \lambda) = 0.$$

12.3 Given an observed sequence $\mathcal{O}$, we have given an efficient method for calculating

$$\operatorname*{argmax}_{Q} \ \text{Prob}(Q \mid \mathcal{O}).$$

One might ask why we are interested in this Q and not

$$\operatorname*{argmax}_{Q} \ \text{Prob}(\mathcal{O} \mid Q) \tag{12.19}$$

instead. The latter state sequence Q, after all, is the one that if given has the highest probability of producing $\mathcal{O}$. To illustrate why this is the wrong Q to find, construct an HMM where the Q given by (12.19) is such that $\text{Prob}(Q) = 0$, so could not possibly have produced $\mathcal{O}$.

12.4 Prove that the reestimation parameters (12.12)–(12.14) do indeed satisfy $\sum_i \overline{\pi}_i = 1$, $\sum_k \overline{p}_{jk} = 1$, and $\sum_a \overline{b}_i(a) = 1$.

12.5 Consider the five amino acid sequences

`WRCCTGC, WCCGGCC, WCGCC, WCCCGCC,  WCCGC.`

Suppose their respective paths through a protein model HMM of length 8 are

$$\begin{gathered}m_0m_1i_1m_2m_3m_4m_5d_6m_7m_8,\\ m_0m_1m_2m_3m_4m_5m_6m_7m_8,\\ m_0m_1m_2d_3d_4m_5m_6m_7m_8,\\ m_0m_1m_2m_3m_4m_5m_6m_7m_8,\\ m_0m_1m_2m_3d_4m_5d_6m_7m_8.\end{gathered}$$

Using the theory of Section 12.3.2, give the alignment of the sequences that these paths determine.

The necessary and sufficient condition for the Markov chain with transition matrix (14.23) to be reversible is that the equations

$$A = D, \quad B = G, \quad C = J, \quad E = H, \quad F = K, \quad I = L \tag{14.24}$$

are all satisfied. When this condition holds, we call the model (14.23) the *general reversible process model*, following the terminology of Yang (1994) in the analogous continuous-time model. The general reversible model has six free parameters, which can be taken as A, B, C, E, F, and I, so one can think of paying for reversibility by losing the choice of six further parameters, or by losing six degrees of freedom.

It is easily checked that when the conditions (14.24) hold, (φ_a, φ_g, φ_c, φ_t) is indeed the stationary distribution of the model (14.23) (see Problem 14.4).

The Felsenstein F81 model (14.18) is the particular case of (14.23) with $A = B = \cdots = L = 1$, the Felsenstein F84 model defined by (14.18), (14.19) and (14.20) is the particular case of (14.23) with $B = C = E = F = G = H = J = K = 1$, $A = D$, $I = L$, and the HKY model is a special case of (14.23) with $B = C = E = F = G = H = J = K$ and $A = D = I = L$. All three models are thus reversible, as claimed above. The choice of one of these models instead of the general reversible model may also be thought of as involving a loss of degrees of freedom. These degrees of freedom are relevant to the test of one these models against the general reversible model, a matter discussed in Section 15.9.4.

14.2.8 The Simple Symmetric Amino Acid Model

The simple symmetric matrix M_1 defined in in Section 6.5.4 is the "amino acid" 20×20 matrix generalization of the "nucleotide" Jukes–Cantor matrix (14.1). Like that model it is reversible. The n-step transition probabilities $\{m_{jk}^{(n)}\}$ given in equations (6.28) and (6.29) define the spectral expansion of this matrix, which is a direct analogue spectral expansion of the Jukes–Cantor matrix (14.1). As with the Jukes–Cantor model, the simple symmetric model is not realistic. We have analyzed it in some detail in Chapters 6 and 10 because it illuminates various properties of more realistic PAM models.

14.3 Continuous-Time Models

As justified in Section 11.7.1, it is assumed for all continuous-time evolutionary models considered that the transition probabilities are of the form described in (11.29) and (11.30).

14.3.1 The Continuous-Time Jukes–Cantor Model

The Jukes–Cantor and Kimura models were first proposed as continuous-time models, and in their continuous-time versions can be analyzed by the methods of Section 11.7. It is convenient to continue to label the states as a, g, c, and t, as in Section 14.1, and to introduce the further notation that i, j, and k are arbitrary members of the set $\{a, g, c, t\}$.

In the Jukes–Cantor model the instantaneous transition rates q_{ij} and q_i are defined by $q_{ij} = \alpha$ for all $i \neq j$, and $q_i = 3\alpha$. Replacing q_{kj} and q_j in equation (11.33) by α and 3α, respectively, we get

$$\frac{d}{dt}P_{ij}(t) = -3\alpha P_{ij}(t) + \alpha \sum_{k \neq j} P_{ik}(t). \tag{14.25}$$

Since $\sum_{k \neq j} P_{ik}(t) = 1 - P_{ij}(t)$, this equation simplifies to

$$\frac{d}{dt}P_{ij}(t) = \alpha - 4\alpha P_{ij}(t). \tag{14.26}$$

This is a linear differential equation and may be solved by standard methods. The solution involves a constant of integration, which is allocated by the boundary conditions $P_{ii}(0) = 1$, $P_{ij}(0) = 0$, $i \neq j$. Using these boundary conditions, the solution is found to be

$$P_{ii}(t) = .25 + .75e^{-4\alpha t}, \tag{14.27}$$

$$P_{ij}(t) = .25 - .25e^{-4\alpha t}, \quad j \neq i. \tag{14.28}$$

These equations show that as $t \to \infty$, both $P_{ii}(t)$ and $P_{ij}(t)$ approach 0.25. Thus in the stationary distribution all nucleotides have equal probability at any site, as is expected from the symmetry of the model. This stationary distribution can also be found by applying equations (11.34). The similarity of the expressions for $P_{ii}(t)$ and $P_{ij}(t)$ and the expressions given in equations (14.4) and (14.5) indicate why the present model is the continuous-time analogue of the discrete-time Jukes–Cantor model.

The fact that q_i is independent of i shows that if an event is defined as the substitution of one nucleotide for another, substitutions follow the laws of a homogeneous Poisson process as discussed in Section 4.1. The probability that j substitutions occur in time t is given by the Poisson probability distribution (4.10) with parameter 3α. It can be shown (see Problem 14.7) that if j transitions from one state to another have occurred in the continuous-time Markov process described by equation (14.26), the probability that the process has returned to its initial state after these j transitions is

$$\frac{1}{4} + \frac{3}{4}\left(-\frac{1}{3}\right)^j. \tag{14.29}$$

The probability that j transitions do occur in time t is given by (4.10) with the parameter λ replaced by 3α. Combining these results, the probability that at time t the original nucleotide is the predominant one is

$$\sum_{j=0}^{\infty} e^{-3\alpha t} \frac{(3\alpha t)^j}{j!} \left(\frac{1}{4} + \frac{3}{4} \left(-\frac{1}{3} \right)^j \right). \tag{14.30}$$

By equation (B.20), this reduces to the right-hand side in equation (14.27). Equation (14.28) is found similarly.

Suppose that two independent contemporary populations are available that descended from a common population t units of time ago, and that the evolutionary properties of these two populations since their common ancestor are described by the Jukes–Cantor model. We would like to use data from these two populations to estimate the evolutionary parameter α in this model, describing in a sense the rate at which one nucleotide replaces another in these populations. We would also like to estimate the time t, since this can be used in defining a distance between the two populations.

It is convenient to begin the analysis by considering the probability p that, at a given site, the predominant nucleotides in the two populations differ. Given two DNA sequences each consisting of N nucleotide sites, and with the assumption that the replacement processes at all sites are independent and have identical stochastic properties, the theory of Section 3.3.1 shows that $\hat{p}$, the observed proportion of sites at which the predominant nucleotides differ between the two populations considered, is an unbiased estimator of p. It is thus natural to estimate p by $\hat{p}$. Further, equation (2.77) shows that the variance of this estimate is

$$\text{variance of } \hat{p} = \frac{p(1-p)}{N}. \tag{14.31}$$

The simplicity of this calculation follows largely from the iid assumption made. This assumption is unlikely to hold in practice, and we later examine some of the consequences of its not holding.

We now use this estimate, and its variance, to discuss the estimation of t and of the parameter α in the Jukes–Cantor model. Poisson process theory shows that in this model, the mean number of substitutions, in t units of time, at any nucleotide site down the two lines of descent leading from the common founder to the two populations is $6\alpha t$. The mean number ν of substitutions down the two lines of descent at all N sites together is then $\nu = 6N\alpha t$.

The symmetry inherent in the Jukes–Cantor model implies that whatever the predominant nucleotide at time 0, the probability $I(t)$ that at time t the two descendent populations have the same predominant nucleotide is

$$I(t) = (P_{ii}(t))^2 + \sum_{j \neq i} (P_{ij}(t))^2,$$

where $P_{ii}(t)$ is given by (14.27) and $P_{ij}(t)$ is given by (14.28). These equations show that

$$I(t) = .25 + .75e^{-8\alpha t}. \tag{14.32}$$

The probability p that the nucleotides are different in the two populations is thus

$$p = 1 - I(t) = .75(1 - e^{-8\alpha t}). \tag{14.33}$$

This equation relating p to αt can be found in a second, more efficient, way. The reversibility of the Jukes–Cantor model implies that the properties of the stochastic process describing any line of descent is the same as that describing the process in reverse, that is, by considering the corresponding line of ascent. The elapsed time up the line of ascent from one of the two contemporary populations up to the founder population, and then down the line of descent from the founder population to the other contemporary population, is $2t$. Therefore, the probability that, at any nucleotide site, the same nucleotide occurs in both populations is given by equation (14.27) with t replaced by $2t$. This leads directly to the expression (14.32) and thus to (14.33).

The evolutionary parameters introduced above are all functions of the composite parameter αt. Standard practice in estimating αt is to invert the relation between p and αt in (14.33), yielding

$$\alpha t = -\frac{1}{8}\log\left(1 - \frac{4}{3}p\right),$$

and thus to estimate the composite αt by

$$\widehat{\alpha t} = -\frac{1}{8}\log\left(1 - \frac{4}{3}\hat{p}\right). \tag{14.34}$$

This right-hand side is defined only if $\hat{p} < \frac{3}{4}$. When $\hat{p} \geq \frac{3}{4}$ the two sequences are more dissimilar than two random sequences will tend to be, and it is natural to define $t = +\infty$. Even when $\hat{p} < \frac{3}{4}$ the estimator (14.34) is a biased estimator of αt, and in fact, there is no unbiased estimator of αt. We consider this "within-model" bias further in Section 14.3.6. For the moment we follow the implications of the estimation procedure in 14.34).

Equation (14.34) shows that unless t can be estimated extrinsically, the aim of estimating the parameter α cannot be achieved. Similarly, unless α can be estimated extrinsically, the aim of estimating t cannot be achieved. All that is possible is estimation of the composite parameter αt. If on the other hand the parameter α can be regarded as being the same for all populations, at all times, and at all nucleotide sites, then proportional values for t can be found, and these can be used as surrogate distances in a phylogenetic tree estimation procedure as described in Section 15.5.

The estimate $\hat{\nu}$ of $\nu = 6N\alpha t$ is

$$\hat{\nu} = -\frac{3N}{4} \log \left(1 - \frac{4}{3}\hat{p}\right). \tag{14.35}$$

The issue of bias discussed above arises also for this estimator, but again we ignore it.

The estimation formula (14.35) has several interesting consequences. Perhaps the most interesting is that if $\hat{p}$ is small, the Taylor series approximation (B.25) shows that

$$\hat{\nu} \cong N \left(\hat{p} + \frac{2}{3}\hat{p}^2\right).$$

Thus $\hat{\nu}$ is marginally larger than $N\hat{p}$, the observed number of sites at which the predominant nucleotides in the two populations differ from each other. For example, if in 300 of $N = 3{,}000$ sites the predominant nucleotide differs between the two populations, then $\hat{p} = 0.1$ and $\hat{\nu} \cong 320$. Thus it is estimated that about 20 substitutions occurred but are not observed by counting the differences between the extant sequences. This is because more than one substitution can occur in the same position. The quantity ν accounts for *all* substitutions.

The statistical differentials variance formula (B.33) can be used to approximate the variances of the estimators (14.34), and (14.35). We illustrate this by considering the variance of $\hat{\nu}$. Equation (B.33) shows that this variance is approximately

$$\left(\left.\frac{d\hat{\nu}}{d\hat{p}}\right|_{\hat{p}=p}\right)^2 (\text{variance of } \hat{p}),$$

which from (14.31) is

$$\left(\left.\frac{d\hat{\nu}}{d\hat{p}}\right|_{\hat{p}=p}\right)^2 \frac{p(1-p)}{N}.$$

By equation (14.35), this equals

$$Np(1-p)e^{8\nu/3N}. \tag{14.36}$$

Since maximum likelihood estimators have optimality properties as discussed in detail in Chapter 8, it is appropriate to find the maximum likelihood estimate of the parameter ν. Suppose that of the N sites sampled there are $j = N\hat{p}$ sites for which different nucleotides appear in the two populations. Under the iid assumption the likelihood of the data in terms of ν is, from equations (14.32) and (14.33), proportional to

$$\left(\frac{3}{4}\left(1 - e^{-4\nu/3N}\right)\right)^j \left(\frac{1}{4} + \frac{3}{4}e^{-4\nu/3N}\right)^{N-j}, \tag{14.37}$$

where $8\alpha t$ has been replaced by $4\nu/3N$. The derivative of the logarithm of this expression with respect to ν is

$$\frac{1}{N} e^{-4\nu/3N} \left(\frac{j}{p} - \frac{N-j}{1-p} \right), \tag{14.38}$$

where

$$p = \frac{3}{4} \left(1 - e^{-4\nu/3N} \right). \tag{14.39}$$

It follows from this that the expression (14.35) for $\hat{\nu}$ in is the maximum likelihood estimate of ν if $\hat{p}$ is given by j/N.

This result also follows directly from the invariance property of maximum likelihood estimators, discussed in Section 8.3. The maximum likelihood estimator of p is $\hat{p} = j/N$, and the relation (14.39) between p and ν implies that the maximum likelihood estimate $\hat{\nu}$ is given by (14.35). In the following section we use this more direct approach when analyzing the continuous-time Kimura model.

A further differentiation of the log likelihood and the asymptotic variance formula (8.18) shows that the asymptotic variance of $\hat{\nu}$ is identical to the expression in (14.36).

14.3.2 The Continuous-Time Kimura Model

The analysis of the continuous-time Kimura model (14.6) is more complex than that for the Jukes–Cantor model, and the notation must be handled more carefully. The instantaneous parameters q_{ij} and q_i, introduced in equations (11.29) and (11.30), take the values

$$\begin{aligned} q_{ij} &= \alpha \text{ for the } (i,j) \text{ pairs } (a,g), (g,a), (c,t), \text{ and } (t,c), \\ q_{ij} &= \beta \text{ for the } (i,j) \text{ pairs } (a,c), (a,t), (g,c), (g,t), \\ &\qquad\qquad\qquad\qquad (c,a), (c,g), (t,a), \text{ and } (t,g). \end{aligned}$$

In all cases,

$$q_i = \alpha + 2\beta. \tag{14.40}$$

If these choices are made, the system of equations (11.33) yields a set of four simultaneous differential equations typified by

$$\frac{d}{dt} P_{aa}(t) = -(\alpha + 2\beta) P_{aa}(t) + \alpha P_{ag}(t) + \beta \left(P_{ac}(t) + P_{at}(t) \right), \tag{14.41}$$

where the notation now indicates specific nucleotide types. The boundary conditions for these equations are $P_{ii}(0) = 1$ for all i, $P_{ij}(0) = 0$ for all $j \neq i$. This system of four differential equations can be solved, and for any i the value of $P_{ii}(t)$ is found to be

$$P_{ii}(t) = .25 + .25 e^{-4\beta t} + .5 e^{-2(\alpha+\beta)t}. \tag{14.42}$$

The expression for $P_{ij}(t)$ depends on the choice of i and j. If the initial predominant nucleotide is a specified purine (respectively pyrimidine), then the probability that at time t the predominant nucleotide is the other purine (respectively pyrimidine) is

$$.25 + .25e^{-4\beta t} - .5e^{-2(\alpha+\beta)t}. \tag{14.43}$$

If the initial predominant nucleotide is a specified purine (respectively pyrimidine), then the probability that at time t the predominant nucleotide is a pyrimidine (respectively purine) is

$$.5 - .5e^{-4\beta t}. \tag{14.44}$$

The probabilities given by these expressions reduce to the corresponding formulae for the Jukes–Cantor model when $\alpha = \beta$. They are also the continuous time analogues of the discrete-time equations (14.8), (14.9), and (14.11). A further implication of these equations is that, as expected, in the stationary distribution for this model all nucleotides have equal probability.

The detailed balance requirements (11.35) hold for the Kimura model, so that the process is reversible. This implies that the Jukes–Cantor process is also reversible.

As in the Jukes–Cantor model, Poisson process theory is relevant to the evolutionary properties of the Kimura model. For the Kimura model q_i, defined in (14.40), is independent of i. Thus if an event is defined as a substitution from one nucleotide to another at any nucleotide site in the line of descent of any population, the probability that j events occur in time t is given by equation (4.10), with the generic parameter λ now given by $\alpha + 2\beta$.

The observed number of transition substitutions and the observed number of transversion substitutions can be used to estimate the parameters of the Kimura model using the reversibility property of the model and the "line of ascent and descent" argument described in Section 14.3.1. Here t is replaced by $2t$ in the right-hand sides of equations (14.42), (14.43), and (14.44). The resulting expressions then show that if

$$\gamma = 4(\alpha+\beta)t, \quad \phi = 8\beta t,$$

then the probability p_1 that at any site one purine (pyrimidine) occurs in one sequence and the other purine (pyrimidine) in the other sequence is

$$p_1 = .25 + .25e^{-\phi} - .5e^{-\gamma}.$$

The probability p_2 that at any site a purine (pyrimidine) occurs in one sequence and a pyrimidine (purine) in the other sequence is

$$p_2 = .5 - .5e^{-\phi}.$$

Suppose that in a DNA sequence consisting of N nucleotide sites there are n_1 sites where one purine (pyrimidine) occurs in the sequence of one population and the other purine (pyrimidine) in the sequence of the other population, and n_2 sites where a purine occurs in one sequence and a pyrimidine in the other. The observed proportions $\hat{p}_1 = n_1/N$ and $\hat{p}_2 = n_2/N$ are the maximum likelihood estimators of p_1 and p_2, respectively (see Problem 8.2). The invariance property of maximum likelihood estimators discussed in Section 8.3 implies that the maximum likelihood estimates $\hat{\gamma}$ and $\hat{\phi}$ satisfy the equations

$$\hat{p}_1 = .25 + .25e^{-\hat{\phi}} - .5e^{-\hat{\gamma}},$$
$$\hat{p}_2 = .5 - .5e^{-\hat{\phi}}.$$

These equations lead to the maximum likelihood estimates

$$\hat{\gamma} = -\log(1 - 2\hat{p}_1 - \hat{p}_2), \tag{14.45}$$
$$\hat{\phi} = -\log(1 - 2\hat{p}_2). \tag{14.46}$$

In the DNA sequence considered, the mean number ν of nucleotide substitutions through the course of evolution since the original founding population is

$$\nu = 2N(\alpha + 2\beta)t = N(.5\gamma + .25\phi).$$

From equations (14.45) and (14.46), the maximum likelihood estimator $\hat{\nu}$ of ν is

$$\hat{\nu} = -N\left(.5\log(1 - 2\hat{p}_1 - \hat{p}_2) + .25\log(1 - 2\hat{p}_2)\right). \tag{14.47}$$

This estimator is subject to the same qualifications as those made for the Jukes–Cantor estimator.

It is interesting to compare calculations derived from (14.47) with those from the parallel Jukes–Cantor model. If N =3,000 and the data yield 210 transitional and 90 transversional changes, and thus 300 changes in total, then $\hat{p}_1 = 0.07$ and $\hat{p}_2 = 0.03$. The total number of changes from one predominant nucleotide to another at some time during the evolution of the two populations since their common founder population is then estimated from equation (14.47) to be about 326. This is somewhat larger than the corresponding estimate in the Jukes–Cantor model.

This implies that if a satisfactory estimate of α and β are available extrinsically, the estimate of t differs in the Kimura and the Jukes–Cantor models given the same data in each. Thus if the true model is the Kimura model and the Jukes–Cantor equation (14.34) is used to estimate t, then a "between-models" biased estimation procedure has been used. This indicates a problem with using an estimate of t as a surrogate for a between–population distance, as discussed above. In practice, the true evolutionary procedure was far more complex than that described by the Kimura model, so that even larger biases can be expected if a simple model such as the Jukes–Cantor model or the Kimura model is used for estimation of evolutionary parameters. This is discussed further in Section 14.3.6.

All the above estimators can also be found by writing down the likelihood and maximizing its logarithm with respect to γ and ϕ. In this case the likelihood is a multinomial, being proportional to

$$\left(.25 + .25e^{-\phi} + .5e^{-\gamma}\right)^{N-n_1-n_2} \left(.25 + .25e^{-\phi} - .5e^{-\gamma}\right)^{n_1} \left(.5 - .5e^{-\phi}\right)^{n_2}. \tag{14.48}$$

Maximization of this function with respect to ϕ and γ leads to estimators identical to those in (14.45) and (14.46). If the values of α and β are known, a similar procedure may be used to find the maximum likelihood estimate of t. A procedure very similar to this is discussed below in Section 14.3.5.

While the procedure of maximizing (14.48) is unnecessarily long so far as finding estimates is concerned, there is one advantage to it. Variance approximations for $\hat{\gamma}$ and $\hat{\phi}$ can be found from a second differentiation of the logarithm of the likelihood, using the information matrix (8.31), together with an approximation for the covariance of $\hat{\gamma}$ and $\hat{\phi}$ parallel to those for the variances of $\hat{\gamma}$ and $\hat{\phi}$. The parameter ν is a linear combination of γ and ϕ, and the variance of $\hat{\nu}$ can be found by applying equation (2.62).

14.3.3 The Continuous-Time Felsenstein Model

The discrete-time Felsenstein model was discussed in Section 14.2.4. That model is the discrete-time version of a continuous-time model originally proposed to address the problem of phylogenetic tree reconstruction, and so we now describe this continuous-time model.

The essential assumption of the discrete-time Felsenstein model (14.18) is that the probability of the substitution of one nucleotide by another is proportional to the stationary probability of the substituting nucleotide. This assumption is maintained in the continuous-time version of the model. If, for example, the predominant nucleotide at any given time t is a, the model assumes that for small h, the probability that the predominant nucleotide at time $t+h$ is X is $u\varphi_X h$, where X stands for any of the nucleotides g, c, and t, while the probability that the predominant nucleotide continues to be a at time $t+h$ is $1 - u(\varphi_g + \varphi_c + \varphi_t)h$, where in all the above expressions terms of order $o(h)$ are ignored. Parallel assumptions are made if the predominant nucleotide at time t is g, c, or t.

These assumptions define the instantaneous transition rates q_{jk} and q_j in equations (11.29) and (11.30), and this leads to specific forms for four simultaneous differential equations of the form (11.33). The solution of these equations is

$$P_{ii}(t) = e^{-ut} + (1 - e^{-ut})\varphi_i, \tag{14.49}$$

$$P_{ij}(t) = (1 - e^{-ut})\varphi_j, \quad j \neq i. \tag{14.50}$$

This solution implies that the continuous-time Felsenstein model satisfies the detailed balance equations (11.35), and is thus reversible.

14.3.4 The Continuous-Time HKY Model

The continuous-time analogue of the HKY model (14.21) is discussed in detail by Hasegawa et al. (1985). Its time-dependent solution is given by the spectral expansion corresponding to the discrete-time model (14.21), with λ_j^n replaced by $e^{(\lambda_j - 1)t}$. It shares all the desirable properties of its discrete-time analogue, and is used in phylogenetic studies, as discussed in Chapter 15.

14.3.5 Continuous-Time Amino Acid Model

The simple symmetric discrete-time model of Section 14.2.8 has a direct continuous-time analogue, found from the Kolmogorov equations (11.33) by putting $q_{jk} = \alpha$. We shall choose α so that the PAM requirement that the probability that the initially predominant amino acid after one time unit is still predominant is 0.99. With these choices the solutions of equations (11.33) are

$$P_{ii}(t) = .05 + .95e^{-20\alpha t}, \tag{14.51}$$

$$P_{ij}(t) = .05 - .05e^{-20\alpha t}, \quad j \neq i. \tag{14.52}$$

These equations are similar in form to the Jukes–Cantor equations (14.27) and (14.28). The requirement $P_{ii}(1) = 0.99$ yields $\alpha = 0.00053$.

These calculations allow maximum likelihood estimation of t, the time (in "PAM model" time units) back to an assumed common ancestor of two contemporary amino acid sequences, provided that the simple symmetric model can be assumed. Suppose, for example, that two contemporary sequences of length 10,000 are compared and that in this comparison there are 1,238 matches and 8,762 mismatches. Since the simple symmetric model is reversible and the time connecting the two sequences is $2t$, the invariance property of maximum likelihood estimators and equation (14.51) show that the maximum likelihood estimate $\hat{t}$ of t satisfies the equation

$$.1238 = .05 + .95e^{-40\alpha\hat{t}}. \tag{14.53}$$

The solution of this equation (with $\alpha = 0.00053$) is $\hat{t} = 120.5$. Of course, as discussed in Section 14.2.8 with respect to its discrete-time analogue, this simple model is quite unrealistic, so that a result such as this cannot be taken as applying to real evolutionary processes.

Further "amino acid" transition matrices may be found from the "amino acid" analogues of the Kimura, Felsenstein and HKY models. For example, the continuous-time amino acid analogue of the continuous-time F81 Felsenstein model (14.50) was discussed by Hasegawa and Fujiwara (1993), and has properties which are the direct "amino acid" extensions of those of the Felsenstein model.

Models based on amino acids can be far more complicated than those based on nucleotides. For example, Muse and Gaut (1994) consider a model with a 61 $\times$ 61 transition matrix whose states correspond to the non-terminating codons. In this model the probability of a substitution of codon i by codon j in time δt is of the form $\alpha\pi_n\delta t$ for a synonymous substitution (that is, if codons i and j correspond to the same the amino acid), is of the form $\beta\pi_n\delta t$ for a non-synonymous substitution for which codons i and j differ by a single nucleotide substitution, and is 0 if codons i and j differ by more than a single nucleotide substitution. Here π_n is the population frequency of the substituting nucleotide.

Further properties of amino acid models where reversibility is assumed are developed by Müller and Vingron (2000).

14.3.6 A Remark About Bias

Three different forms of bias arise in the discussion in Sections 14.3.1 and 14.3.2, and it is important to distinguish among them.

First, in Section 14.3.1 the estimator $\widehat{\alpha t}$ given by equation (14.34) is biased: its expected value is not the true value of αt. This bias is analogous to that arising for the maximum likelihood estimate of the parameter k described below equation (8.34). Although in both cases the correct probability distribution is assumed (the probabilities defined by the Jukes–Cantor model in Section 14.3.1, the gamma distribution in the case of estimating k), a bias in the estimation of a parameter still arises. We might call this a "within-model" bias.

We could hope to amend the estimator to lower or even remove the bias. In the case of the estimation of k as discussed below equation (8.34), the bias would be removed by using the method of moments estimator, or decreased by using the maximum likelihood estimation with a large sample size. In the case of equation (14.34), a bias reduction is possible by using a Taylor series approximation to the right-hand side. The second-order Taylor approximation (B.30) shows that the right-hand side in (14.34) may be approximated by

$$-\frac{1}{8}\log\Big(1-\frac{4}{3}p\Big)+\frac{1}{6}\frac{\hat{p}-p}{1-4p/3}+\frac{8}{9}\frac{(\hat{p}-p)^2}{(1-4p/3)^2}. \tag{14.54}$$

The mean of $\hat{p}$ is p and the variance of $\hat{p}$ is as given in equation (14.31). Thus the expected value of the right-hand side in (14.54) is

$$-\frac{1}{8}\log\Big(1-\frac{4}{3}p\Big)+\frac{8}{9}\frac{p(1-p)}{N(1-4p/3)^2}. \tag{14.55}$$

This implies that the estimator $\widehat{\alpha t}^*$ of αt, defined by

$$\widehat{\alpha t}^* = \widehat{\alpha t} - \frac{8}{9}\frac{\hat{p}(1-\hat{p})}{N(1-4\hat{p}/3)^2}, \tag{14.56}$$

removes some of the bias arising for the estimator $\widehat{\alpha t}$ defined in (14.34).

Second, the bias discussed in Section 14.3.2 arises for a different reason than that discussed above. In that section the true evolutionary model is the Kimura two-parameter model, but the parameter αt in that model is estimated by the procedure which assumes the Jukes–Cantor evolutionary model, that is through equation (14.34). Thus this bias arises through model misspecification, and the bias might be called a "between-models," or "systematic," bias. This form of bias is not removed by increasing the sample size. Nor is a bias reduction possible along the lines of that leading to (14.56) when the true evolutionary model is quite unknown.

In the case of the bias discussed in Section Section 14.3.2, if the value of α is known, the model misspecification described implies a bias in the estimation of the evolutionary time t. While the numerical value of the bias is not large, the two models involved, that is the Kimura two-parameter model and the Jukes–Cantor model differ from each other only through the value of one parameter. Thus the two models are "close." One can expect a far greater bias if the Jukes–Cantor model, or any other relatively simple evolutionary model, is used for estimation of evolutionary parameters when in fact a far more complicated model is appropriate.

Finally, in using any evolutionary model to estimate properties of a phylogenetic tree, independent evolutionary processes are often assumed at the various sites, with the same stochastic process properties assumed at all sites, at all times and in all species. These assumptions are unrealistic, and using them adds a further systematic bias in estimation procedures. This bias also is not removed by increasing the sample size or by methods similar to those leading to the revised estimator (14.56).

These various biases have obvious implications concerning the reliability of any phylogenetic tree reconstruction. This matter is discussed further in Chapter 15 in the context of such a reconstruction.

Problems

14.1 Prove equation (14.3) by induction on n. (For a discussion of proofs by induction, see Section B.18.)

14.2 Check that the stationary distribution of the Markov chain (14.15) is the vector given in (14.16).

14.3 Check that the Markov chains (14.12) and (14.13) are reversible by showing that they satisfy the detailed balance conditions (11.3), but that the chain defined by (14.15) does not satisfy these conditions and is thus not reversible.

14.4 Show that when the conditions (14.24) hold, the stationary distribution of the model (14.23) is $(\varphi_a, \varphi_g, \varphi_c, \varphi_t)$.

14.5 The Jukes–Cantor model (14.1) is a special case of (that is, is *nested within*) the Kimura model (14.6). It is also special case of (that is, is *nested within*) the Felsenstein model (14.18). Show that it is the only model that is nested within both the Kimura model (14.6) and the Felsenstein model (14.18), and that neither the Kimura model nor the Felsenstein model is nested within the other.

14.6 Show that the conditions (14.24) are necessary and sufficient for the transition matrix (14.23) to be reversible.

14.7 Derive the expression (14.29). (Hint: use the result of Problem 4.7.)

14.8 Check that the continuous-time Kimura model satisfies the detailed balance requirements (11.36).

14.9 Show that the sum in the expression (14.30) does reduce, as claimed, to the expression on the right-hand side of (14.27).

14.10 The aim of this question is to compare the numerical values of the two estimators $\hat{\nu}$ given respectively in equations (14.35) for the Jukes–Cantor model and (14.47) for the Kimura model. In this comparison we identify the values of $\hat{p}$ in the Jukes–Cantor model with $\hat{p}_1 + \hat{p}_2$ in the Kimura model.

(i) Use the logarithmic approximation (B.24) to show that when $\hat{p}_1$ and $\hat{p}_2$ are both small, the two estimators are close, and differ by only a term of order $\hat{p}^2$.

(ii) Use the more accurate approximation (B.25) to compare the values of the two estimators when (a) $\hat{p}_1 = \hat{p}_2$, (b) $\hat{p}_1 = 2\hat{p}_2$. Thus show that even when $\hat{p}_1 = \hat{p}_2$ the two estimates of ν differ.

14.11 Estimate the standard deviation of the estimate $\hat{t} = 120.5$ found by solving equation (14.53).