

---

# *Bayesian Methods in Biological Sequence Analysis*

---

**J.S. Liu and T. Logvinenko**

*Department of Statistics, Harvard University, Cambridge, MA, USA*

Hidden Markov models, the expectation–maximization algorithm, and the Gibbs sampler were introduced for biological sequence analysis in early 1990s. Since then the use of formal statistical models and inference procedures has revolutionized the field of computational biology. This chapter reviews the hidden Markov and related models, as well as their Bayesian inference procedures and algorithms, for sequence alignments and gene regulatory binding motif discoveries. We emphasize that the combination of Markov chain Monte Carlo and dynamic-programming techniques often results in effective algorithms for NP-hard problems in sequence analysis.

## 3.1 INTRODUCTION

In the past decade, we have witnessed the development of the likelihood approach to pairwise sequence alignments (Bishop and Thompson, 1986; Thorne *et al.*, 1991); probabilistic models for RNA secondary structure (Zuker, 1989; Lowe and Eddy, 1997); the expectation–maximization (EM) algorithm for finding regulatory binding motifs (Lawrence and Reilly, 1990; Cardon and Stormo, 1992); Gibbs sampling strategies for detecting subtle sequence similarities (Lawrence *et al.*, 1993; Liu, 1994; Neuwald *et al.*, 1997); hidden Markov models for DNA composition analysis and multiple alignments (Churchill 1989; Krogh *et al.*, 1994a; Baldi *et al.*, 1994); and the hidden semi-Markov model for gene prediction and protein secondary structure prediction (Burge and Karlin, 1997; Schmidler *et al.*, 2000; **Chapters 4 and 7**). All these developments show that algorithms resulting from statistical modeling efforts constitute a significant portion of today’s bioinformatics toolbox. This chapter aims to introduce readers to these modeling techniques and related Bayesian methodologies useful for biological sequence analysis.

xref

Section 3.2 gives an overview of the Bayesian inference procedure, including model building, prior specification, model selection and Bayesian computation. Section 3.3 introduces the general hidden Markov model framework, with an example on DNA compositional heterogeneity. Section 3.4 reviews Bayesian pairwise alignment methods. Section 3.5 demonstrates how hidden Markov models are used in multiple sequence alignment and how Bayesian inferences can be made on model parameters. Section 3.6 outlines the Bayesian methods for finding subtle repetitive motifs in a DNA sequence. Section 3.7 concludes the chapter with a brief discussion. We emphasize in this chapter the usefulness of dynamic-programming-like recursive algorithms in Bayesian and likelihood-based inference, and the importance of combining these efficient computational techniques with more flexible Markov chain Monte Carlo (MCMC) tools for bioinformatics problems.

## 3.2 OVERVIEW OF THE BAYESIAN METHODOLOGY

In Bayesian analysis, a *comprehensive* probabilistic model  $f(\mathbf{y} | \boldsymbol{\theta}, \boldsymbol{\tau})$  is employed to describe relationships among all variables under consideration: those that we observe (data and knowledge,  $\mathbf{y}$ ), those about which we wish to learn (scientific hypotheses,  $\boldsymbol{\theta}$ ), and those that are needed in order to construct a proper model (missing data or nuisance parameters,  $\boldsymbol{\tau}$ ). With this Bayesian model, the basic probability theory can lead us automatically to an efficient use of the available information and to a precise numerical quantification of uncertainties in estimation and prediction (Gelman *et al.*, 1995). In general, the Bayesian approach has the following advantages: its explicit use of probabilistic models to formulate scientific problems (i.e., a quantitative story-telling); its coherent way of incorporating all sources of information and of treating nuisance parameters and missing data; and its ability to quantify uncertainties in all estimates. Other aspects of the Bayesian method are discussed in the **Chapters 21, 12, 16, 17** and **29**.

xref

### 3.2.1 The Procedure

Instead of treating  $\boldsymbol{\theta}$  and  $\boldsymbol{\tau}$  as unknown constants as in a frequentist approach, Bayesian analysis treats them as realized values of random variables that follow a *prior distribution*  $f_0(\boldsymbol{\theta}, \boldsymbol{\tau})$ , which is typically regarded as known to the researcher independently of the data under analysis. The joint probability distribution can then be represented as *Joint = likelihood  $\times$  prior*, i.e.,

$$p(\mathbf{y}, \boldsymbol{\theta}, \boldsymbol{\tau}) = f(\mathbf{y} | \boldsymbol{\theta}, \boldsymbol{\tau}) f_0(\boldsymbol{\theta}, \boldsymbol{\tau}).$$

The theorem that combines the prior and the data to form the conditional distribution  $p(\boldsymbol{\theta}, \boldsymbol{\tau} | \mathbf{y})$ , also called the *posterior distribution* of  $\boldsymbol{\theta}$ , is a simple mathematical result first given by Thomas Bayes in 1763. Bayes' theorem says that

$$p(\boldsymbol{\theta}, \boldsymbol{\tau} | \mathbf{y}) = \frac{p(\mathbf{y}, \boldsymbol{\theta}, \boldsymbol{\tau})}{p(\mathbf{y})} = \frac{f(\mathbf{y} | \boldsymbol{\theta}, \boldsymbol{\tau}) f_0(\boldsymbol{\theta}, \boldsymbol{\tau})}{\int f(\mathbf{y} | \boldsymbol{\theta}, \boldsymbol{\tau}) f_0(\boldsymbol{\theta}, \boldsymbol{\tau}) d\boldsymbol{\theta} d\boldsymbol{\tau}}. \quad (3.1)$$

The denominator  $p(\mathbf{y})$ , which is a normalizing constant for the function, is sometimes called the *marginal likelihood* and can be used for model selection. If we are interested

only in  $\theta$ , we can obtain its posterior distribution as

$$p(\theta | \mathbf{y}) = \int p(\theta, \tau | \mathbf{y}) d\tau. \quad (3.2)$$

The statistical procedure based on the systematic use of this theorem is named after Bayes, but was first developed after his death by Laplace in 1812. The adjective *Bayesian* is often used for approaches in which subjective probabilities are emphasized. Despite the deceptively simple form of (3.1) and (3.2), the challenging aspects of Bayesian statistics are: the development of a model,  $f(\mathbf{y} | \theta, \tau) f_0(\theta, \tau)$ , which must effectively capture the key features of the underlying scientific problem; and the necessary computation for deriving the posterior distribution.

### 3.2.2 Model Building and Prior

It is often a useful strategy in building a complex biological model to distinguish two kinds of unknowns: population parameters and missing data (Liu and Lawrence, 1999). Although there is no absolute distinction between the two types, missing data are usually directly related to the individual observation. They can be ‘imputed’ either conceptually or computationally so as to ease the statistical analysis. On the other hand, the parameters usually characterize the entire population under study and are fixed in number. For example, in a multiple alignment problem, alignment variables that must be specified for each observed sequence can be viewed as missing data. Residue frequencies for the aligned positions or the choice of scoring matrices, which apply to all the sequences, are population parameters. Whereas this distinction is essential to the maximum likelihood method, it is employed primarily for conceptual clarity in Bayesian statistics.

A major controversial aspect of the Bayesian method is the need for *prior* distributions for the unknown parameters. Since the choice of priors injects subjective judgments into an analysis, Bayesian methods have long been regarded as less ‘objective’ than their frequentist counterparts and disfavored. However, the emotive words ‘subjective’ and ‘objective’ should not be taken too seriously since there are considerable subjective elements and personal judgments in all phases of scientific investigation. These subjective elements, if made explicit and treated with care, should not undermine the scientific results of the investigation. More importantly, it should be regarded as good scientific practice for investigators to make their subjective inputs explicit. A truly objective evaluation of any procedure is by how well it attains its stated goals.

Although it is worthwhile to think of prescribing ‘objective’ priors (usually with the adjective *noninformative*), such choices are usually unattainable in practice. We advocate the use of sensitivity analysis, i.e., an analysis of how the inferential statements vary for a reasonable range of prior distributions, to validate the conclusions of a Bayesian analysis.

### 3.2.3 Model Selection and Bayes Evidence

Classical hypothesis testing can be seen as a model selection procedure in which one chooses between the null and the alternative hypotheses based on the degree of ‘surprise’ of the observed data. In contrast, Bayesian model selection can be achieved in a coherent probabilistic framework. Firstly, all the candidate models are embedded into an aggregated model. Then, the ‘overall’ posterior probability of each candidate model is computed and used to discriminate or combine them (Kass and Raftery, 1995).

For example, let  $M = 0$  indicate the ‘null’ model, and let  $M = 1$  be the alternative. We are interested in seeing which one fits the data better. The first step is to write down the joint distribution for the *aggregated model*:  $P(\mathbf{y}, \boldsymbol{\theta}, M) = P(\mathbf{y} | \boldsymbol{\theta}, M)P(\boldsymbol{\theta}, M)$ . Under the assumption that the data depend on the models through their respective parameters, this equation can be rewritten as

$$P(\mathbf{y}, \boldsymbol{\theta}, M) = P(\mathbf{y} | \boldsymbol{\theta}_m)P(\boldsymbol{\theta}_m | M = m)P(M = m),$$

where  $P(\boldsymbol{\theta}_m | M = m)$  is the prior for the parameters in model  $m$ , and  $P(M = m)$  is the prior probability of model  $m$ . Note that the dimensionality of  $\boldsymbol{\theta}_m$  may be different for different  $m$ . The posterior probability for model  $m$  is obtained as:

$$\begin{aligned} P(M = m | \mathbf{y}) &\propto P(\mathbf{y} | M = m)P(M = m) \\ &= \left\{ \int P(\mathbf{y} | \boldsymbol{\theta}_m)P(\boldsymbol{\theta}_m | M = m) d\boldsymbol{\theta}_m \right\} P(M = m). \end{aligned}$$

Sometimes one may not want to select and use only one model, but prefer to use all the candidate models collectively. The foregoing Bayesian formulation can easily accommodate such ‘model averaging’ (Kass and Raftery, 1995). The choice of  $P(M = m)$ , which is the prior on models, is made independently of the data in study. A frequent choice is  $P(M = 0) = P(M = 1) = 0.5$ , implying that both models are equally likely a priori. But in some cases we might want to set  $P(M = 1)$  very small. For example, in the context of a database search, we might set  $P(M = 1)$  to be inversely proportional to the number of sequences in the database.

### 3.2.4 Bayesian Computation

In many applications, the required computation is the main obstacle for applying either the Bayesian or other sophisticated statistical methods. In fact, until recently these computations were often so difficult that sophisticated statistical modeling and Bayesian methods were largely for theoreticians and philosophers. The introduction of the bootstrap method (Efron, 1979), the EM algorithm (Dempster *et al.*, 1977), and MCMC methods (Gilks *et al.*, 1998; Liu, 2001) has brought many powerful models into the mainstream of statistical analysis. As we illustrate in later sections, by appealing to the rich history of computation in bioinformatics, many required optimizations and integrations can be done exactly, which gives rise to either an exact solution to the maximum likelihood estimation and the posterior distributions or an improved MCMC algorithm.

Markov chain Monte Carlo refers to a class of algorithms for simulating random variables from a target distribution,  $\pi(\mathbf{x})$ , known up to a normalizing constant. A major advantage of these algorithms is their ability to ‘divide and conquer’ a high-dimensional and complex problem. These algorithms serve our purpose well because in Bayesian analysis we want to draw random samples from the joint posterior distribution (3.1) without having to know its denominator. The basic idea behind all MCMC algorithms is the design and simulation of a Markov chain whose equilibrium distribution is exactly the target posterior distribution  $\pi(\mathbf{x})$ . Two main strategies for constructing such chains are the Metropolis–Hastings algorithm and the Gibbs sampler (Liu, 2001), both widely used in diverse fields and reviewed in the Appendix. More discussions on MCMC and their fascinating applications can be found in **Chapters 21** and **12**.

### 3.3 HIDDEN MARKOV MODEL: A GENERAL INTRODUCTION

A sequence of random variables  $h_1, h_2, \dots$  is said to follow an  $l$ th-order *Markov chain* if

$$P(h_k | h_1, \dots, h_{k-1}) = P(h_k | h_{k-1}, \dots, h_{k-l}).$$

For example, one may assume that an observed sequence of nucleotide base pairs forms a first-order Markov chain with transition probabilities  $P(h_{k+1} | h_k) = \theta_{h_k, h_{k+1}}$ . With an observed realization of the Markov chain (e.g., a segment of DNA sequence), one can obtain the maximum likelihood estimations of the  $\theta$ 's from counting the frequencies of dimer occurrences. For example,  $\theta_{GC}$  can be estimated by  $n_{GC}/n_G$ , where  $n_{GC}$  is the count of all the GC pairs in the sequence and  $n_G = n_{GA} + n_{GC} + n_{GT} + n_{GG}$ .

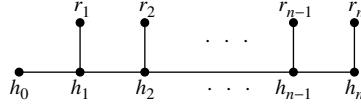
It has long been known that the simple independent and identically distributed (i.i.d.) model (which summarizes the data by the counts or frequencies of the four nucleotides) is not good enough to describe DNA sequences. In coding regions, every nonoverlapping triplet of nucleotides codes for one of the 20 amino acids or a stop signal. Thus, a second-order Markov chain is perhaps more desirable. However, correlation between the neighboring bases is still highly significant in noncoding regions. Some recent studies (Liu *et al.*, 2001; 2002; Huang *et al.*, 2002) demonstrate that using a second-order or third-order Markov chain to model the promoter regions (hundreds to thousands of bases upstream of the starting codon of the gene) can significantly improve the accuracy of gene regulatory binding motif discovery. It is clear that the genome is much more complex than a third-order Markov chain. Although it is desirable to model the genome sequences by even higher-order Markov chains, the number of unknown parameters increases too fast to make them too useful. For example, an  $l$ th-order Markov chain needs  $3 \times 4^{l-1}$  free parameters. Additionally, the Markov model cannot capture certain local structures of the sequences. For example, in order to predict where the coding regions are in a genome, where insertions and deletions occur in sequence alignments, or how to segment a protein sequence into secondary structures, it is more suitable to use a hidden Markov model (HMM).

The HMM as initially introduced in the late 1960s is a powerful statistical modeling tool and has been widely used in signal processing, speech recognition, and time series analysis (Rabiner, 1989). The method was first applied to model DNA sequences by Churchill (1989) and has recently become very popular for multiple sequence alignments due to the pioneering work of Krogh *et al.* (1994a) and Baldi *et al.* (1994). The basic form of an HMM can be written as

$$r_k \sim f_k(r | h_k, \theta); \quad h_k \sim g_k(h | h_{k-1}, \tau),$$

where  $f_k$  and  $g_k$  are probability distributions,  $\theta$  and  $\tau$  are parameters, and the  $r_k$  are observations. The  $h_i$  form a Markov chain and are often unobservable (i.e., hidden). Of interest is the inference of  $\theta$ ,  $\tau$ , and perhaps the  $h_i$ .

To be specific, let us examine an HMM that can accommodate compositional heterogeneity in DNA sequences. In particular, consider a sequence  $R$  that consists of different segments, where within each segment the sequence composition is homogeneous. We can only observe  $R$  and are interested in making inference on the locations of the segment change points and the composition parameters within each segment. A simple HMM, first proposed by Churchill (1989), is shown in Figure 3.1.



**Figure 3.1** A graphical illustration of the hidden Markov model.

In this model, we assume that the hidden layer  $\mathbf{h} = (h_0, h_1, \dots, h_n)$  is a Markov chain. Each  $h_i$  can take only two possible values:  $h_i = 0$  implies that residue  $r_i \sim \text{Multinom}(\boldsymbol{\theta}_0)$ ; and  $h_i = 1$  indicates that  $r_i \sim \text{Multinom}(\boldsymbol{\theta}_1)$ . Here  $\boldsymbol{\theta}_k = (\theta_{kA}, \theta_{kC}, \theta_{kG}, \theta_{kT})$ . A  $2 \times 2$  transition matrix,  $\boldsymbol{\tau} = (\tau_{kl})$ , where  $\tau_{kl} = P(h_i = k \rightarrow h_{i+1} = l)$ , dictates the generation of  $\mathbf{h}$ . A similar model was developed by Krogh *et al.* (1994b) to predict protein-coding regions in the *E. coli* genome.

Let  $\Theta = (\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\tau})$ . The likelihood function of  $\Theta$  can be written as

$$L(\Theta | R) = \sum_{\mathbf{h}} P(R | \mathbf{h}, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1) P(\mathbf{h} | \boldsymbol{\tau}), = \sum_{\mathbf{h}} p_0(h_0) \prod_{i=1}^n (\theta_{h_i r_i} \tau_{h_{i-1} h_i}),$$

where  $h_0$  is assumed to follow a known distribution  $p_0(h_0)$ . This function can be evaluated using a recursive summation method shown later in (3.3). With a prior distribution  $f_0(\Theta)$ , which may be a product of three independent Dirichlet distributions, we can write down the joint posterior distribution of all unknowns:

$$P(\Theta, \mathbf{h} | R) \propto P(R | \mathbf{h}, \Theta) P(\mathbf{h} | \Theta) f_0(\Theta).$$

In order to get the marginal posterior of  $\Theta$ , we may implement a data augmentation method (Tanner and Wong, 1987), which iterates the following steps:

- *Path imputation.* Draw  $\mathbf{h}^{(t+1)} \sim P(\mathbf{h} | R, \Theta^{(t)})$ .
- *Posterior sampling.* Draw  $\Theta^{(t+1)} \sim P(\Theta | R, \mathbf{h}^{(t+1)})$ .

The path imputation step samples a hidden chain, or path,  $\mathbf{h}$ , from its *posterior* distribution with given parameter value, and this task can be achieved by a recursive method (Liu, 2001). More precisely, we note that

$$\begin{aligned} P(\mathbf{h} | R, \Theta) &= cP(\mathbf{h}, R | \Theta) = cP(R | \mathbf{h}, \Theta) p(\mathbf{h} | \Theta) \\ &= cp_0(h_0) \prod_{i=1}^n \{P(r_i | h_i) P(h_i | h_{i-1})\} = cp_0(h_0) \prod_{i=1}^n (\theta_{h_i r_i} \tau_{h_{i-1} h_i}), \end{aligned}$$

where  $c^{-1} = \sum_{\mathbf{h}} \{p_0(h_0) \prod_{i=1}^n (\theta_{h_i r_i} \tau_{h_{i-1} h_i})\}$ . Define  $F_0(h) = p_0(h)$  and then, recursively, let

$$F_{k+1}(h) = \sum_{h_k=0}^1 \{F_k(h_k) \tau_{h_k h} \theta_{hr_{k+1}}\}, \quad \text{for } k = 1, \dots, n. \quad (3.3)$$

At the end of the recursion we have  $c^{-1} = F_n(0) + F_n(1)$  and

$$P(h_n | R, \Theta) = \frac{F_n(h_n)}{F_n(0) + F_n(1)}. \quad (3.4)$$

In order to sample  $\mathbf{h}$  from  $P(\mathbf{h} | R, \Theta)$ , we first draw  $h_n$  from distribution (3.4) and then draw  $h_k$  recursively backward from the distribution

$$P(h_k | h_{k+1}, R, \Theta) = \frac{F_k(h_k)\tau_{h_k h_{k+1}}}{F_k(0)\tau_{0h_{k+1}} + F_k(1)\tau_{1h_{k+1}}}. \quad (3.5)$$

The posterior sampling step in data augmentation requires one to draw from the posterior distribution of  $\Theta$  given  $\mathbf{h}$  and  $R$ . This involves only the sampling from appropriate Dirichlet distributions. For example,  $\theta_0$  should be drawn from Dirichlet  $(n_{0a} + \alpha_a, \dots, n_{0t} + \alpha_t)$ , where,  $n_{0a}$ , say, is the counts of the  $r_i$  whose type is A and whose hidden state  $h_i$  is zero. It is straightforward to extend this model to a  $k$ -state HMM so as to analyze a sequence with regions of  $k$  different compositional types.

### 3.4 PAIRWISE ALIGNMENT OF BIOLOGICAL SEQUENCES

Ever since the creation of GenBank, which grew from 680 338 base pairs in 1982 to 22 billion base pairs in 2002 (Benson *et al.*, 2002), and other related databases, sequence comparisons and sequence database searching have played a pivotal role in contemporary biological research and served as the central pillar of computational biology. By observing the sequence similarities between a new target gene (or a segment of it) and some well-studied ones, biologists can gain a very important insight into its function and structure (see also **Chapter 2**). The two best-known ‘rigorous’ pairwise alignment methods are that by Needleman and Wunsch (1970) for global alignment and that by Smith and Waterman (1981) for local alignment. They are based on dynamic programming and are guaranteed to find the global optimum of certain alignment scoring functions. Two of the most popular ‘heuristic’ pairwise alignment algorithms are BLAST (Altschul *et al.*, 1990) and FASTA (Pearson and Lipman, 1988), both of which are orders of magnitudes faster than the rigorous ones. **Chapter 2** provides detailed discussion on these algorithms and issues related to the statistical significance of the resulting scores.

This section gives a quick overview of the Smith–Waterman algorithm, and then moves on to a Bayesian version of the Needleman–Wunsch algorithm and a motif-based Bayesian pairwise alignment method. Throughout the section, the observed data consist of two DNA or protein sequences,  $R^{(1)} = (r_1^{(1)}, r_2^{(1)}, \dots, r_{n_1}^{(1)})$  and  $R^{(2)} = (r_1^{(2)}, r_2^{(2)}, \dots, r_{n_2}^{(2)})$ .

#### 3.4.1 Dynamic Programming for Pairwise Alignment

A ‘local alignment’ of two sequences refers to the alignment of the most similar regions of the sequences without penalizing the unaligned parts from the ends of the sequences. One of its major advantages is its ability to identify conserved domains in proteins, which may not extend over the whole sequence (see also **Chapter 2**). The Smith–Waterman algorithm seeks to maximize a scoring function, which is the sum of the scores of all

aligned pairs of residues minus the number of ‘inter-alignment’ gaps times a gap penalty. To achieve this aim, it computes an alignment matrix of size  $(n_1 + 1) \times (n_2 + 1)$  whose  $(i, j)$ th element  $f(i, j)$  corresponds to the highest score of all possible alignments of the two subsequences of  $(r_1^{(1)}, r_2^{(1)}, \dots, r_i^{(1)})$  and  $(r_1^{(2)}, r_2^{(2)}, \dots, r_j^{(2)})$ . As illustrated in Figure 3.2, these entries are computed recursively via the *forward algorithm*:

- Initialization:

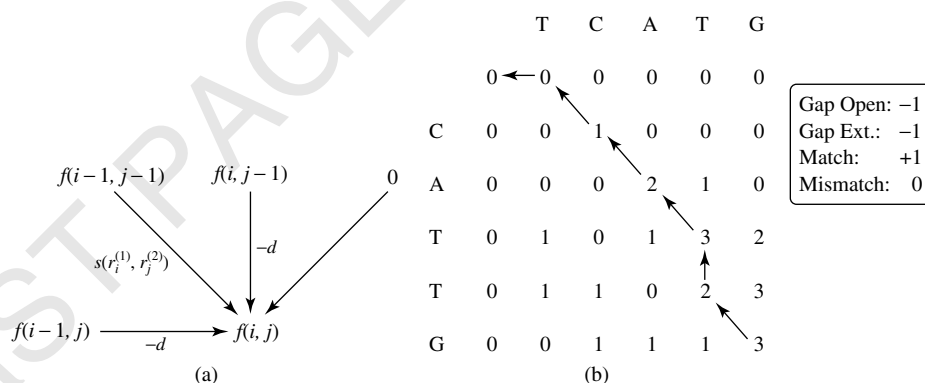
$$f(0, 0) = 0; f(0, i) = 0; f(j, 0) = 0.$$

- Recursion:

$$f(i + 1, j + 1) = \max \left\{ 0, f(i, j) + s(r_{i+1}^{(1)}, r_{j+1}^{(2)}), f(i, j + 1) - d, f(i + 1, j) - d \right\}. \quad (3.6)$$

Here  $s(\cdot, \cdot)$ , often called a *substitution matrix* and defined for all pairs of amino acid residues, measures how biologically ‘similar’ a pair of amino acids are. Two of the most widely used substitution matrices are the PAM matrix, derived from a protein evolutionary model (Dayhoff *et al.*, 1978), and the BLOSUM matrix, derived directly from large sets of conserved protein segments (Henikoff and Henikoff, 1992). **Chapter 2** gives a more detailed comparison of the two matrices. A negative score  $(-d)$ , called the gap penalty, is given to all amino acids aligned with a gap. Letting  $f(i, j)$  take an assignment score of 0 is the key to the local alignment and is equivalent to restarting the alignment from residues  $r_i^{(1)}$  and  $r_j^{(2)}$  in the two sequences, respectively. Gotoh (1982) proposed a biologically more meaningful modification of the algorithm which penalizes more heavily for starting a gap, and less for extending an already existing one.

A *backward-tracing* algorithm is used to find the best scoring alignment. It starts at the  $(n_1, n_2)$ th entry of the alignment matrix and traces back to the  $(0, 0)$ th position using arrow pointers obtained from the forward computation process. These arrow pointers, as shown in Figure 3.2(b), indicate which of the four entries on the right-hand side of (6) is the largest (ties are broken arbitrarily). For example, if  $f(i, j)$  is obtained by  $f(i, j - 1) - d$ , an arrow pointer is placed from  $(i, j)$  to  $(i, j - 1)$  and a



**Figure 3.2** (a) The dynamic recursion of the Smith–Waterman algorithm. (b) The alignment matrix and backward tracing.

gap will be introduced. At the end of the backward tracing (Figure 3.2(b)), we obtain one of the maximum-score alignments. However, this optimal result depends on the choices of a substitution matrix  $s(\cdot, \cdot)$  and gap penalties. After obtaining the optimal alignment score, it is of interest to assess its statistical significance; this is discussed in **Chapter 2**.

xref

The Needleman–Wunsch algorithm is very similar to Smith–Waterman except that it also penalizes the unaligned parts from the ends of the sequences. More precisely, its recursion satisfies

$$f(i+1, j+1) = \max \left\{ f(i, j) + s(r_{i+1}^{(1)}, r_{j+1}^{(2)}), f(i, j+1) - d, f(i+1, j) - d \right\}.$$

The resulting score is the optimal global alignment score of the two sequences.

### 3.4.2 Bayesian Pairwise Alignment

Since the traditional pairwise alignment methods are not based on any statistical models, it is difficult to judge whether they have incorporated all the relevant information, and, more importantly, whether the parameters have been tuned optimally. The use of statistical models and Bayesian methodology can help in making such judgements. A Bayesian pairwise alignment method starts with a model,  $P(R^{(1)}, R^{(2)} | \Theta, \tau)$  that describes the relationship between two sequences. To be specific, we can let  $\Theta$  be a  $20 \times 20$  joint symmetric probability matrix analogous to a scoring matrix such as the PAM or BLOSUM, describing the joint distribution of a pair of aligned amino acids. For example, according to BLOSUM62, the joint probability  $\theta_{r_1 r_2}$  of  $r_1 = \text{I}$  (isoleucine) occurring in sequence 1 and  $r_2 = \text{L}$  (leucine) in the same position of sequence 2 is about 4 times the product of their respective marginal frequencies,  $\theta_{r_1} \times \theta_{r_2}$ , where  $\theta_{r_1}$  is the sum of the entries in the row for  $r_1$  (or the column for  $r_1$ , due to symmetry) of  $\Theta$ . Generally, the  $(i, j)$ th entry of a BLOSUM $x$  matrix stores  $2 \log_2 \theta_{i,j} / \theta_i \theta_j$  for amino acid pair  $(i, j)$ . The number ‘ $x$ ’ reflects the fact that these joint frequencies are estimated from the set of protein sequences among which no pair have more than  $x\%$  alignment positions with identical residues (for further details, see **Chapter 2**).

xref

Conceptually, the alignment of  $R^{(1)}$  and  $R^{(2)}$  is characterized by an alignment matrix  $A$ , where element  $a_{i,j}$  is set to 1 if residue  $i$  of sequence 1 ‘aligns’ with residue  $j$  of sequence 2, and to 0 otherwise. A restriction is that the aligned residues have to be ‘collinear’, i.e., if  $r_{i_1}^{(1)}$  is aligned with  $r_{j_1}^{(2)}$  and  $r_{i_2}^{(1)}$  with  $r_{j_2}^{(2)}$ , then  $(i_1 - i_2)(j_1 - j_2) > 0$ .

#### Gap-based global alignment

Let  $\Lambda = (\lambda_o, \lambda_e)$  be probabilities of gap opening and gap extension, respectively, which govern the formation of the alignment matrix  $A$ . Here we show how the global alignment problem may be treated in a Bayesian way by using the statistical models pioneered by Thorne *et al.* (1991; 1992•). Firstly, the joint distribution is defined as

Q1

$$P(R^{(1)}, R^{(2)}, A, \Theta, \Lambda) = P(R^{(1)}, R^{(2)} | A, \Theta) P(\Theta) P(A | \Lambda) P(\Lambda).$$

Traditional alignment procedures can be seen as optimizing an objective function that is analogous to a log-likelihood (Holmes and Durbin, 1998). More precisely, for a set of

fixed values  $\Theta = \Theta^0$  and  $\Lambda = \Lambda^0$ , one finds  $A^*$  such that

$$\log P(R^{(1)}, R^{(2)}, A^* | \Theta^0, \Lambda^0) = \max_{\text{all } A} \{\log P(R^{(1)}, R^{(2)} | A, \Theta^0) + \log P(A | \Lambda^0)\}. \quad (3.7)$$

The need to set parameter values  $\Theta^0$  and  $\Lambda^0$  has been the subject of much discussion in the field of bioinformatics. A distinctive advantage of the Bayesian procedure is its added modeling flexibility in the specification of parameters.

Let  $A$  be the alignment indicator matrix, which can also be seen as a *path* in a dynamic programming setting. With given  $\Lambda = (\lambda_o, \lambda_e)$ , the probability of any allowable path, prior to seeing the content of the two sequences to be aligned, but conditional on their lengths  $n_1$  and  $n_2$ , is

$$P(A | \lambda_o, \lambda_e) = \frac{\lambda_o^{k_g(A)} \lambda_e^{l_g(A) - k_g(A)}}{\sum_{A'} \lambda_o^{k_g(A')} \lambda_e^{l_g(A') - k_g(A')}}, \quad (3.8)$$

where  $k_g(A)$  and  $l_g(A)$  are the total number and the total length of the gaps in  $A$ , respectively. The summation in the denominator is over all possible alignments  $A'$  of the two sequences. In the following derivation, we assume that the length information,  $n_1$  and  $n_2$ , is conditioned on implicitly.

If we let  $\Theta$  take values in a series of BLOSUM $x$  matrices (after log-odds transformations), then we can compute the marginal likelihood of  $\Lambda$  as:

$$\begin{aligned} P(R^{(1)}, R^{(2)} | \Lambda) &= \sum_{\Theta} \sum_A P(R^{(1)}, R^{(2)} | A, \Theta) P(A | \Lambda) P(\Theta), \\ &= \frac{\sum_{\Theta} \sum_A P(R^{(1)}, R^{(2)} | A, \Theta) P(\Theta) \lambda_o^{k_g(A)} \lambda_e^{l_g(A) - k_g(A)}}{\sum_{A'} \lambda_o^{k_g(A')} \lambda_e^{l_g(A') - k_g(A')}}. \end{aligned} \quad (3.9)$$

In the numerator,  $\Theta$  is marginalized out by summing over all the scoring matrices in a given set, each with a prior ‘weight’  $P(\Theta)$ . Both the numerator and the denominator of (3.8) can be computed via a recursive procedure similar to the Needleman–Wunsch algorithm (Liu and Lawrence, 1999).

As with traditional dynamic programming alignment algorithms, we can describe a path as consecutive moves of three types:  $\rightarrow$  (deletion),  $\downarrow$  (insertion), and  $\searrow$  (match). To ensure uniqueness, one often adds the restriction that an insertion cannot follow a deletion. For example, to obtain the numerator of (3.8), we start with  $p(0, 0) = p_m(0, 0) = 1$ ,  $p_i(0, 0) = p_d(0, 0) = 0$ , and compute recursively:

$$\begin{aligned} p_m(k, l) &= p(k-1, l-1) \theta_{r_k^{(1)} r_l^{(2)}}, \\ p_i(k, l) &= \{\lambda_e p_i(k-1, l) + \lambda_o p_m(k-1, l)\} \theta_{r_k^{(1)}}, \\ p_d(k, l) &= \{\lambda_e p_d(k, l-1) + \lambda_o p_m(k, l-1) + \lambda_o p_i(k, l-1)\} \theta_{r_l^{(2)}}, \\ p(k, l) &= p_m(k, l) + p_i(k, l) + p_d(k, l). \end{aligned}$$

In the equations,  $p_m(k, l)$  is the score of entry  $(k, l)$  when the last move is a ‘match’;  $p_i(k, l)$  is the score when the last move is an ‘insertion’ for sequence 1, and  $p_d(k, l)$  the score when the last move is a ‘deletion’ for sequence 1. Thus,  $P(R^{(1)}, R^{(2)} | \Lambda, \Theta) = p(n_1, n_2)$ , and the posterior distribution of  $\Lambda$  is

$$P(\Lambda | R^{(1)}, R^{(2)}) \propto P(R^{(1)}, R^{(2)} | \Lambda) f_0(\Lambda).$$

This Bayesian approach provides not only the maximum a posteriori alignment of the two sequences, but also a *distribution* of the pairwise alignments, which can be represented by a random sample from the posterior alignment distribution and used to measure uncertainty in the alignment result. A Bayesian version of the Smith–Waterman algorithm was given recently in Webb *et al.* (2002). They showed that the new method outperformed the optimally tuned Smith–Waterman algorithm in detecting distantly related proteins.

### *Motif-based local alignment*

While the gap-based approaches have dominated alignment methods for many years, Bayesian statistics opens up new directions in dealing with insertions and deletions in alignments. Zhu *et al.* (1998) attacked the Bayesian alignment problem by directly specifying a prior alignment distribution: all alignments with  $k$  gaps are equally likely, and all  $k$  in the given range are equally likely. This prior discounts alignment with many gaps by penalizing it by a factor that is inversely proportional to the number of alignments of that type. The summation over all alignments is carried out by Sankoff’s (1972) dynamic programming method. Input requirements for the scoring matrices are also more flexible in the Bayesian setting than in traditional methods. For example, Zhu *et al.* (1998) examine the use of a series of either PAM or BLOSUM matrices as prior input in which all the matrices are assigned equal probability a priori. They report that the posterior distributions of the scoring matrices are often flat and sometimes multimodal, indicating that no one matrix is clearly more preferable to others when aligning the two sequences.

Consider the expression for the posterior distribution of the scoring matrix,

$$P(\Theta | R^{(1)}, R^{(2)}) = \frac{1}{P(R)} \sum_k \sum_A P(R^{(1)}, R^{(2)} | A, \Theta) P(\Theta) P(A | \Lambda = k) P(\Lambda = k),$$

where  $\Theta$  indicates one of the series of scoring matrices (PAM or BLOSUM), reflecting to a certain extent the distance between the two sequences, and  $\Lambda$  denotes the number of gaps allowed in the alignment. Since this posterior is obtained by averaging over all alignments, a ‘good’ alignment is not required for assessing the distance between the two sequences. This feature may be of value to distance methods employed in molecular evolution studies, since the requirement that a pair of sequences must be sufficiently close to permit a good alignment is removed.

## 3.5 MULTIPLE SEQUENCE ALIGNMENT

Although pairwise search methods have been tremendously successful in modern biological research, they treat all the positions of the query sequence as equally important,

whereas in biological reality many ‘unimportant’ regions of a protein (such as those segments corresponding to structural loops) can tolerate severe distortions and are not well conserved even for those within a specialized protein family. A more attractive approach for detecting remotely related proteins is to use features common to a set of related proteins to perform the search. These common features can be most effectively represented by a position-specific consensus ‘profile’ extracted by a comparative study of all the protein sequences under consideration, i.e., multiple sequence alignment (MSA).

Currently, the most widely used MSA method is ClustalW (Thompson *et al.*, 1994a). The algorithm starts by comparing all pairs of sequences in the set using a dynamic programming algorithm (similar to Smith–Waterman). The resulting pairwise comparison scores are transformed into evolutionary distances (Kimura, 1983). Then, a phylogenetic tree is constructed using the neighbor-joining method (Saitou and Nei, 1987). Following the branching order of the phylogenetic tree constructed, ClustalW progressively performs pairwise alignments of sequences or the consensus profile matrix at each node. To improve the quality of the alignments, ClustalW uses a sequence weighting strategy to avoid overrepresentation of closely related homologs (Thompson *et al.*, 1994b). At the alignment stage, a variety of substitution matrices (chosen depending on the closeness of the relationship between sequences/profiles to be aligned) are used and position-specific gap penalties are incorporated.

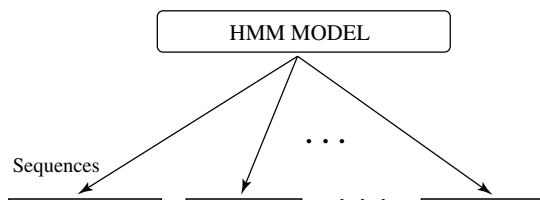
Another widely used method for constructing MSA is PSI-BLAST (Altschul *et al.*, 1997). It first uses BLAST (Altschul *et al.*, 1990) to collect all the sequences in the database searched that are significantly related to the query sequence. Then an MSA is created by ‘anchoring’ selected pairwise alignments to the query sequence (to avoid overrepresentation of closely related proteins, sequences with more than 98 % identity are thrown out). Based on the weighted counts of the residues in columns of the MSA – the weighting procedure of Henikoff and Henikoff (1994) is used – a position-specific profile is constructed. Using a modification of BLAST, another iteration of database search is performed using the profile constructed and the relatives of the next order are collected. These are anchored in turn to the current MSA and used to update the profile. The procedure is repeated until no more relatives can be found from the database.

### 3.5.1 The Rationale of Using Hidden Markov Models in Sequence Alignment

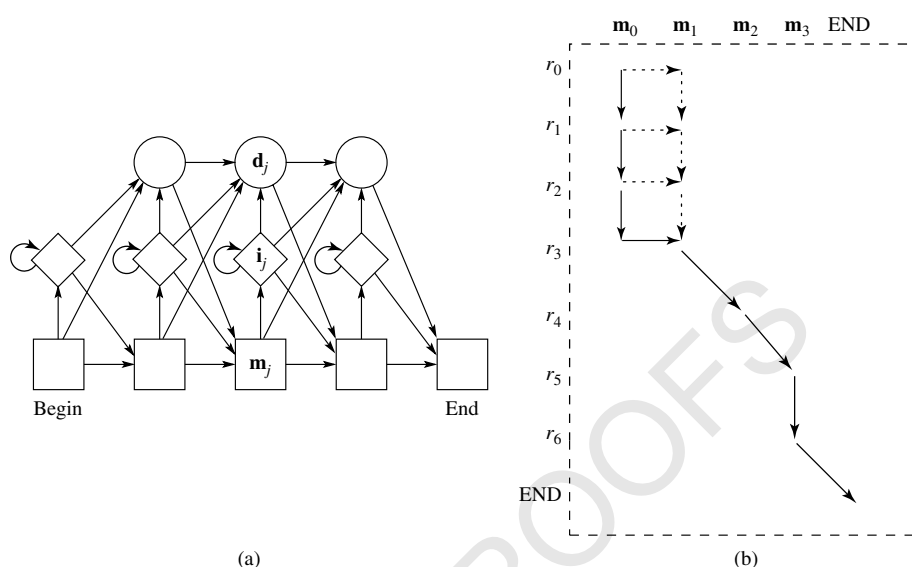
The heuristics used in both PSI-BLAST and ClustalW incorporate a significant amount of biological knowledge. However, both methods lack principled ways for synthesizing more delicate biological information and for tuning parameters. In addition, the MSAs resulting from these methods tend to be heavily influenced by either the query sequence or the sequence order. The introduction of the profile HMM for the sequence alignment problem in early 1990s (Krogh *et al.*, 1994a; Baldi *et al.*, 1994) revolutionized the field and gave the scientist fresh ways of looking at many biological problems including MSA.

In the evolution of protein sequences, segment transpositions are rare so that we can safely assume in most cases that the discrepancy between two homologous sequences is caused by insertions/deletions (indels) and point mutations. Thus, although the sequences are misaligned via indels, conserved residues remain in order. By capturing this characteristic, the HMM not only captures an important feature of protein evolution, but does so in an effective algorithm.

As shown in Figure 3.3, in the HMM framework one treats the sequences to be aligned as i.i.d. observations from a probabilistic mechanism (i.e., ‘insert’, ‘delete’, and ‘match’)



**Figure 3.3** Independent generation of protein sequences from an HMM.



**Figure 3.4** (a) A profile HMM model for sequence alignment. ‘Match’, ‘insert’, and ‘delete’ states are represented by squares, diamonds and circles, respectively. (b) A ‘path’ representing the alignment of a sequence to a profile HMM.

that perturbs a hypothetical common ‘ancestral’ model sequence (called a ‘model’), denoted as  $M = (\mathbf{m}_1, \dots, \mathbf{m}_L)$ . The ‘match’ state can be intuitively understood as that current residue at this position has evolved from the ‘ancestral’ residue via mutations, not indels. Two major distinctions between this HMM framework and the standard evolutionary model are noteworthy. Firstly, since the observed sequences are i.i.d. given the ancestral model, this HMM does not capture the important tree structure of a realistic evolutionary process, which is often essential to reflect correlations among the observed sequences. Secondly, the ancestral model is not meant to be an ancestral sequence. In other words, each model position  $\mathbf{m}_j$  is not meant to recover what the ancestral residue most likely is, but is used to model this position’s residue preference, which results from both the evolutionary force and functional constraints (selection). Although one does not observe the ancestral model, with principled statistical inference methods, one can estimate optimally all the parameters in this model, and consequently the ancestral profile model, based on the observed sequences.

The diagram in Figure 3.4(a) describes the generative procedure for the underlying (hidden) Markov chain. In this model, each  $\mathbf{m}_j$  is regarded as an abstract residue and is

associated with an ‘emission’ probability vector  $\theta_l$  of length  $p$  ( $p = 4$  for DNA sequences, and  $p = 20$  for proteins). For simplicity, we assume that all ‘insert’ states are associated with a common ‘emission’ probability  $\theta_0$ . When generating biological sequences, the types of perturbations allowed are point mutations, insertions, and deletions. A residue in an observed sequence is generated either by a ‘match’ state,  $\mathbf{m}_j$  say, or by an ‘insert’ state,  $\mathbf{i}_k$  say. Another set of transition probabilities, one associated with each arrow in Figure 3.4(a), e.g.,  $\tau_{\mathbf{m}_j\mathbf{m}_{j+1}}$ ,  $\tau_{\mathbf{d}_j\mathbf{m}_{j+1}}$ ,  $\tau_{\mathbf{i}_j\mathbf{i}_j}$ , etc., are needed to describe the underlying Markov chain. Note that the probabilities coming out of a node have to sum to one, and they can be either given in advance or estimated (‘trained’) directly from the set of unaligned sequences.

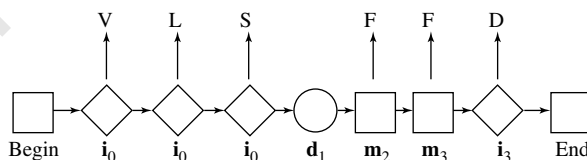
Starting from the ‘Begin’ state, a protein sequence is ‘produced’ from the profile HMM as follows. According to the transition probabilities, a series of hidden states is generated until the ‘End’ state is reached. In turn, each of the ‘match’ and ‘insert’ states generates an amino acid based on its state-specific ‘emission’ probability distribution. Figure 3.5 shows how the sequence  $R = (r_1, r_2, \dots, r_6) \equiv \text{VLSFFD}$  is generated by a profile HMM with three match states.

One can also think of the process of generating sequence  $R$  from the profile HMM as choosing a path through an  $(n + 1) \times (L + 1)$  table starting from the upper left corner and ending with the lower right corner (Figure 3.4(b)), very much similar to the diagram used to illustrate the Smith–Waterman and Needleman–Wunsch algorithms. The columns for this table are denoted by  $\mathbf{m}_0, \dots, \mathbf{m}_L$ , which correspond to a void start position and  $L$  model positions. The rows,  $r_0, \dots, r_n$ , correspond to a void starting residue and the  $n$  observed sequence residues. At any position  $(k, j)$  of this table, the next step allowed is: (a) position  $(k, j + 1)$ , implying that a deletion of model position  $\mathbf{m}_{j+1}$  has occurred ( $\rightarrow$ ); (b) position  $(k + 1, j)$ , implying that  $r_{k+1}$  has occurred ( $\downarrow$ ), or (c) position  $(k + 1, j + 1)$ , implying that  $r_{k+1}$  is generated by model position  $\mathbf{m}_{j+1}$  ( $\searrow$ ). Thus the path depicted by the solid arrows in Figure 3.4(b) corresponds to

$$\mathbf{m}_0 \rightarrow \mathbf{i}_0 \rightarrow \mathbf{i}_0 \rightarrow \mathbf{i}_0 \rightarrow \mathbf{d}_1 \rightarrow \mathbf{m}_2 \rightarrow \mathbf{m}_3 \rightarrow \mathbf{i}_3 \rightarrow \text{END},$$

which is exactly the same as in Figure 3.5.

It is worthwhile to note that the hidden states for this HMM are *not* the  $\mathbf{m}_j$ s, but the allowable ‘paths’ that traverse the  $(n + 1) \times (L + 1)$  table. Liu *et al.* (1999) provided a slightly different formulation of this model so as to make it conform with the conventional HMM shown in Figure 3.1. In other words, they constructed a hidden chain,  $h_0 \rightarrow h_1 \rightarrow \dots \rightarrow h_n$ , that generates the observed sequence  $R$ , where each  $h_k$  records the number of deletions that have occurred up to residue  $r_k$  and whether  $r_k$  is generated by an insert or a match state.



**Figure 3.5** A toy sequence of length 6 generated by a profile HMM with three match states. Note that  $\mathbf{m}_1$  is replaced by state  $\mathbf{d}_1$ , implying that first match state is deleted in generating the observed sequence.

### 3.5.2 Bayesian Estimation of HMM Parameters

Let  $\mathbf{R} = (R_1, \dots, R_m)$  be the set of sequences to be aligned, let  $\Theta$  denote the collection of all parameters including the transition probabilities  $\tau$  and the emission probabilities  $\theta_j$ , and let  $\mathcal{A} = (A_1, \dots, A_m)$  denote the unobserved alignment variable (i.e., for each of the sequences an alignment path as shown in Figure 3.4(b)). For simplicity, we assume that the emission probabilities from the ‘insert’ state are the same for all positions and denoted as  $\theta_0$ . It is observed that once the sequences are aligned (i.e., given  $\mathcal{A}$ ), the transition probabilities,  $\tau_{k,l} = P(s_k \rightarrow s_l)$  (from any state  $s_k$  to any state  $s_l$ ), model emission probabilities,  $\theta_j(r) = P(\text{residue } r | s_k = \mathbf{m}_j)$ , and insertion emission probabilities  $\theta_0$  are easy to estimate:

$$\tau_{k,l} = \frac{T_{k,l}}{\sum_{\text{states } l'} T_{k,l'}},$$

$$\theta_j(r) = \frac{E_j(r)}{\sum_{\text{residues } r'} E_j(r')},$$

where  $T_{k,l}$  is the number of transitions from state  $s_k$  to  $s_l$ , and  $E_j(r)$  is the number of residues of type  $r$  emitted from match state  $\mathbf{m}_j$  in the sequence alignment. With Dirichlet priors – or Dirichlet mixture priors (Sjölander *et al.*, 1998) – a Bayesian estimate of these parameters is also easy. On the other hand, once the parameters of the HMM are given, it is also feasible to find either the optimal or the average alignment of each sequence to the model by using a dynamic programming technique shown earlier (i.e., finding the optimal path in Figure 3.4(b)). Based on these insights, Krogh *et al.* (1994a) modified a Baum–Welch algorithm – an earlier version of the EM algorithm (Baum 1972) – for training the profile HMM. Baldi *et al.* (1994) developed a gradient descent method for finding the maximum likelihood estimate of  $\Theta$ .

The foregoing heuristics according to which it is ‘easy’ to estimate  $\Theta$  given  $\mathcal{A}$  and to handle  $\mathcal{A}$  given  $\Theta$  also facilitate a Bayesian MCMC approach (Churchill and Lazareva, 1999). The general procedure is outlined below.

- *Initialization.* Choose model length  $L$  (the number of ‘match’ states) and set initial parameter values. In the absence of prior knowledge, one may let  $L$  be the average length of all the sequences.
- *Data augmentation.* Starting with  $\Theta^{(0)}$ , we proceed for  $t = 1, \dots, N$ .
  - *Alignment imputation.* Sample  $\mathcal{A}^{(t+1)}$  from  $P(\mathcal{A} | \mathbf{R}, \Theta^{(t)})$ .
  - *Parameter simulation.* Draw  $\Theta^{(t+1)}$  from  $P(\Theta | \mathbf{R}, \mathcal{A}^{(t+1)})$ .

The number of iterations  $N$  may be determined dynamically based on certain convergence criteria (Liu, 2001).

- The final alignment model can be estimated in one of the following ways.
  - Use the average of the  $\Theta^{(t)}$  in the last  $K$  iterations ( $K = 500$ , say).
  - Estimate  $\Theta$  by the one corresponding to the posterior mode.

- One can also find an optimal alignment  $\mathcal{A}^*$ , and then estimate  $\Theta$  as  $E(\Theta | \mathbf{R}, \mathcal{A}^*)$ .

Figure 3.6 shows a simple alignment of three sequences, from which the HMM parameters can be estimated.

In most applications, the number of match states  $L$  is not known and it is desirable to let the data speak for themselves. In some available packages for HMM-based multiple sequence alignment,  $L$  is updated iteratively using heuristic rules. For example, once an alignment like the one shown in Figure 3.6 is produced, one can either remove certain ‘match’ positions if the corresponding column contains more than  $x\%$  deletions or add a new ‘match’ position if more than  $y\%$  of the sequences have insertions of similar types at the same place. It is also possible to treat  $L$  as a new parameter and infer the size of HMM in a coherent Bayesian framework. Operationally, one can insert between the alignment imputation and parameter simulation steps of data augmentation a ‘match’ state updating step to either add or delete a match column using the Metropolis–Hastings rule (see the Appendix).

A key step in implementing the data augmentation procedure is the computation of the likelihood

$$P(\mathbf{R} | \Theta) = \sum_{\mathcal{A}} P(\mathbf{R} | \Theta, \mathcal{A})P(\mathcal{A} | \Theta),$$

because this is needed in imputing  $\mathcal{A}$  from  $P(\mathcal{A} | \mathbf{R}, \Theta) = P(\mathcal{A}, \mathbf{R} | \Theta)/P(\mathbf{R} | \Theta)$ . Dynamic programming recursions can be applied to complete the task with complexity  $O(nmL)$ , where  $m$  is the number of sequences,  $n$  is their average length, and  $L$  is the number of match states. Since all the sequences in consideration are mutually independent conditional on  $\Theta$ , we may focus only on one sequence, i.e., the computation of  $P(R | \Theta)$ . The imputation of  $\mathcal{A}$  for this sequence can be achieved by the following two-stage algorithm, which is analogous to the forward–backward algorithm in Section 3.3. The first stage, forward summation proceeds as follows:

- Initialization with  $f_m(0, 0) = 1$  and  $f_i(0, 0) = f_d(0, 0) = 0$ .
- Recursion:

$$f_m(j+1, k+1) = \theta_k(j+1)[f_m(j, k)\tau_{m_k m_{k+1}} + f_i(j, k)\tau_{i_k m_{k+1}} + f_d(j, k)\tau_{d_k m_{k+1}}];$$

$$f_i(j+1, k+1) = \theta_0(j+1)[f_m(j, k+1)\tau_{m_{k+1} i_{k+1}} + f_i(j, k+1)\tau_{i_{k+1} i_{k+1}}];$$

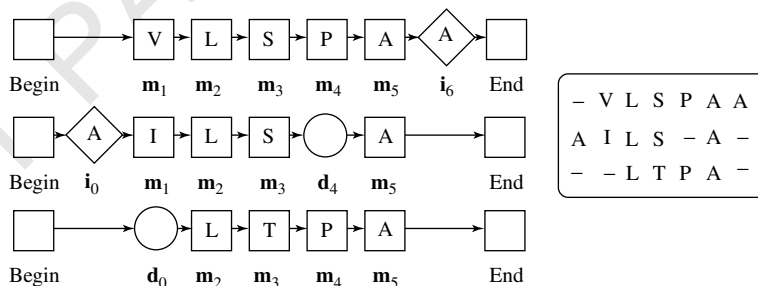


Figure 3.6 The alignment of three sequences to the profile HMM.

$$f_d(j+1, k+1) = f_m(j+1, k)\tau_{\mathbf{m}_k \mathbf{d}_{k+1}} + f_i(j+1, k)\tau_{\mathbf{i}_k \mathbf{d}_{k+1}} + f_d(i+1, k)\tau_{\mathbf{d}_k \mathbf{d}_{k+1}}.$$

- The algorithm terminates when the  $(L, n)$ th entry is reached.

At the end, we have  $P(R | \Theta) = f_m(n, L) + f_i(n, L) + f_d(n, L)$ .

After the forward summation, one can carry out the backward-sampling step to impute the alignment path  $A$ . For a simple description, we let  $v()$  be a backward-tracing function of the three types of states:  $v(\mathbf{m}) = (-1, -1)$ ,  $v(\mathbf{i}) = (-1, 0)$  and  $v(\mathbf{d}) = (0, -1)$ .

- Start from position  $(n, L)$  and proceed as follows.
- Suppose we are at a position  $(x, y)$ . Sample  $B$  from one of the three symbols  $\mathbf{m}$ ,  $\mathbf{i}$ , and  $\mathbf{d}$  with probabilities proportional to  $f_m(x, y)$ ,  $f_i(x, y)$ , and  $f_d(x, y)$ , respectively; set an arrow from  $(x, y)$  to  $(x, y) + v(B)$ .
- Terminate when the  $(0, 0)$ th entry is reached. The path indicated by the collection of arrows is a sample of  $A$  from distribution  $P(A | R, \Theta)$ .

Several HMM-based software packages for producing multiple protein sequence alignments are available, although none of them is Bayesian. HMMER was developed by Eddy (1998) and is available at <http://HMMer.wustl.edu/>. SAM was developed by the UC Santa Cruz group and is available at <http://www.soe.ucsc.edu/projects/compbio/HMM-apps/HMM-applications.html>. It uses Dirichlet mixture priors on amino acid emission probabilities.

### 3.5.3 PROBE: A Motif-Based Multiple Alignment Method

Although the HMM is a very flexible and powerful tool for modeling multiple related sequences, a large number (over 100) of unaligned sequences is necessary to train an HMM due to the large number of parameters present in the model. This requirement in effect limits the HMM-based algorithms to families containing many proteins. In order to detect more remote homologs and to align subtly related proteins, one has to further constrain the profile HMM model in a biologically meaningful way. The propagation model utilized in PROBE (Neuwald *et al.*, 1997; Liu *et al.*, 1999) achieves this type of parameter reduction by focusing on the alignment composed of block motifs.

A block motif is a special HMM that allows no insertions or deletions among the match states. The propagation model consists of a number of block motifs as shown in Figure 3.7. The gaps between blocks, corresponding to gaps in an HMM, can be modeled by a probability distribution other than the geometric one implied by the standard HMM. PROBE also uses a Bayesian procedure for selecting the sizes of the blocks and the number of blocks.

Similar to HMM, the propagation model also assumes that there are  $L$  conserved model positions for each sequence to be aligned, with the only difference being that each model position represents a block of residues of width  $w_l$ . We can imagine that



**Figure 3.7** An illustration of the propagation model in Liu *et al.* (1999).

$L$  motif elements propagate along a sequence. Insertions are reflected by gaps between adjacent motif elements. No deletions are allowed, but it is possible to relax this constraint by treating each block as a mini-HMM. Again, let  $\Theta$  be the collection of all parameters needed in the propagation model (e.g., the motif profile matrices, background frequencies, and parameters characterizing the distribution of gaps between motif blocks) and let the alignment variable  $\mathcal{A} = (A_1, \dots, A_L) = (a_{k,l})_{K \times L}$  be a matrix with  $a_{k,l}$  indicating the starting position of the  $l$ th motif element in sequence  $k$  (Figure 3.7). Note that both  $\Theta$  and  $\mathcal{A}$  in this model are of much smaller dimensions than their counterparts in the HMM because of the block-motif constraint.

The alignment variable  $\mathcal{A}$  as represented by the locations of the motif elements is not observable. But once it is known, we can write down the likelihood  $P(\mathbf{R} | \mathcal{A}, \Theta)$  (details can be found in Liu *et al.*, 1999), from which we can make Bayesian inference on  $\Theta$  easily. On the other hand, once  $\Theta$  were given, we could impute  $\mathcal{A}$  by a draw from  $P(\mathcal{A} | \mathbf{R}, \Theta)$ , which can be achieved using a forward–backward method similar to that in Section 3.3. Thus, we can again implement a data augmentation strategy to iterate between sampling of  $\mathcal{A}$  and  $\Theta$ . In PROBE, a ‘collapsing’ technique (Liu, 1994) is employed to further improve the efficiency of the computation.

The number of motifs  $L$  and the total number of motif columns  $W = w_1 + \dots + w_L$  are treated as random variables and selected according to an approximate Bayesian criterion. PROBE is also one of the earliest algorithms implementing the iterative database search process. Similar to PSI-BLAST, PROBE first uses BLAST to get the first-order relatives of the query sequence and aligns them to produce a block-motif profile. Then, it uses the constructed profile to find more relatives and further refine the alignment. It stops when no new relatives are found. It has been shown that PROBE is a very sensitive method for detecting distantly related proteins and capturing very weak similarities (Hudak and McClure, 1999; Neuwald *et al.*, 1997). The program is available at <ftp://ftp.ncbi.nih.gov/pub/neuwald/probe1.0/>.

### 3.5.4 Bayesian Progressive Alignment

Due to its use of Gibbs sampling in profile estimation, PROBE is very slow. Another limitation of PROBE is its inflexibility in introducing gaps within block motifs. As mentioned before, the HMM-based model can allow for flexible gaps, but it needs many sequences in order to estimate well the excessive number of parameters. In contrast, PSI-BLAST produces the multiple alignment progressively without having to iterate and can accommodate gaps in motifs in a biologically meaningful fashion because of their use of BLAST (a motif-based approach). To combine the attractive features of PSI-BLAST and PROBE, Logvinenko (2002) proposed a Bayesian progressive alignment procedure based on a sequential Monte Carlo principle (Liu, 2001).

The algorithm starts by aligning the query sequence  $R^{(1)}$  to each of the other sequences in a set,  $R^{(2)}, \dots, R^{(n)}$ , by a gap-based Bayesian local alignment method (similar to the one described in Section 3.4.2). The set of sequences can be either given in advance or obtained by an iterative BLAST search as in PSI-BLAST. The sequences are then sorted according to their distances from  $R^{(1)}$  derived based on the pairwise alignments. A position-specific profile matrix  $\Theta = (\hat{\theta}_{al})_{20,l}$  is constructed based on the alignment of  $R^{(1)}$  to its closest relative,  $R^{(2)}$ , and then to  $R^{(3)}$  and so on. Each time a new sequence is introduced, the profile is updated based on the new alignment. As in PSI-BLAST, Henikoff

and Henikoff's (1994) sequence-weighting strategy is used in profile computation to avoid overrepresentation of closely related sequences.

In producing the progressive alignment, a Dirichlet mixture prior distribution (Sjolander *et al.*, 1996) is used for amino acid emission probabilities:

$$P(\boldsymbol{\theta}) \sim \sum_{k=1}^K p_k \frac{\Gamma(\alpha_1^{(k)} + \dots + \alpha_{20}^{(k)})}{\Gamma(\alpha_1^{(k)}) \dots \Gamma(\alpha_{20}^{(k)})} \theta_1^{\alpha_1^{(k)}-1} \dots \theta_{20}^{\alpha_{20}^{(k)}-1}.$$

For every column  $l$  of the alignment residue counts  $\mathbf{m}^{(l)} = (m_1^{(l)}, \dots, m_{20}^{(l)})$  follow a Multinom( $\boldsymbol{\theta}_l$ ) distribution. Profile entries are set to be predictive probabilities of the residues:

$$\hat{\theta}_{d,l} = \int_{\boldsymbol{\theta}_l} \theta_{d,l} P(\boldsymbol{\theta}_l | \mathbf{m}_l) d\boldsymbol{\theta}_l \propto \int_{\boldsymbol{\theta}_l} \theta_{d,l} P(\mathbf{m} | \boldsymbol{\theta}_l) P(\boldsymbol{\theta}_l) d\boldsymbol{\theta}_l.$$

As in Section 3.4.2 a dynamic programming type recursion is used to compute  $P(R, \boldsymbol{\Theta} | \Lambda) = p(n, L)$ :

$$\begin{aligned} p_m(k, l) &= p(k-1, l-1) \hat{\theta}_{r_k l}, \\ p_i(k, l) &= \{\lambda_e p_i(k-1, l) + \lambda_o p_m(k-1, l)\}, \\ p_d(k, l) &= \{\lambda_e p_d(k, l-1) + \lambda_o p_m(k, l-1) + \lambda_o p_i(k, l-1)\}, \\ p(k, l) &= p_m(k, l) + p_i(k, l) + p_d(k, l); \end{aligned}$$

where  $\Lambda = (\lambda_o, \lambda_e)$  are the gap opening and extension probabilities;  $L$  is the number of positions in the profile; and  $\hat{\theta}_{r_k l}$  is a profile entry corresponding to residue  $r_k$  in the  $l$ th column of the alignment. A backward-tracing procedure analogous to that in Section 3.4.1 is used to obtain an alignment.

After all the sequences are incorporated into an alignment, a Gibbs sampling step is used to improve on its quality. Each sequence  $R^{(i)}$  is removed in turn and realigned to the others. The profile is updated accordingly. Although this additional step makes the procedure relatively slow, the resulting alignment is less likely to be only locally optimal.

### 3.6 FINDING RECURRING PATTERNS IN BIOLOGICAL SEQUENCES

Our focus here is the discovery of repetitive motif elements in a given set of sequences. The main motivation for this task is that these recurring patterns in biopolymer sequences often correspond to the functionally or structurally important parts of these molecules. For example, repetitive patterns in noncoding regions of DNA sequences may correspond to a 'regulatory motif' to which a certain regulatory protein binds so as to control gene expressions (Stormo and Hartzell, 1989, Lawrence and Reilly, 1990, Liu, 1994, Roth *et al.*, 1998). Without loss of generality, we can imagine concatenating all the sequences in the data set to form one 'supersequence'  $R$ . The multiple occurrences of a regulatory motif in  $R$  is thus analogous to the multiple occurrences of a *word* in a long sentence. It is of interest to find out what this motif is and where it has occurred. What makes this problem challenging is that even though the motif occurs in the supersequence multiple times, these occurrences are not necessarily identical to each other. In other words, there

are often some ‘typos’ in each occurrence of the word. It is therefore quite natural for us to employ probabilistic models to handle this problem.

### 3.6.1 Block-Motif Model with i.i.d. Background

A simple model conveying the basic idea of a motif that repeats itself with random variations is the block-motif model shown in Figure 3.8. It was first developed by Liu *et al.*, (1995) and has been employed to find subtle repetitive patterns, such as helix-turn-helix structural motifs (Neuwald *et al.*, 1995) or gene regulation motifs (Roth *et al.*, 1998; Liu *et al.* 2001; 2002), in both protein and DNA sequences.

The model says that at unknown locations  $A = (a_1, \dots, a_K)$  there are repeated occurrences of a motif. So the sequence segments at these locations should look similar to each other. In other parts of the sequence, the residues (or base pairs) follow a ‘background model’ modeled now as i.i.d. observations from a multinomial distribution. Since the motif region is a very small fraction of the whole sequence, it is not unreasonable to assume that the background frequency  $\theta_0 = (\theta_{0,a}, \dots, \theta_{0,t})$  is known in advance. For the motif of width  $w$ , we let  $\Theta = [\theta_1, \dots, \theta_w]$ , where each  $\theta_j$  describes the base frequency at position  $j$  of the motif. The matrix  $\Theta$  is simply the profile matrix for the motif.

To facilitate analysis, we introduce an indicator vector  $\mathbf{I} = (I_1, \dots, I_n)$ , where  $n$  is the length of  $R$ . We let  $\mathbf{I}_{[-i]}$  be the vector of  $I$ s without  $I_i$ . Here,  $I_i = 1$  means that position  $i$  is the start of a motif pattern, and  $I_i = 0$  means otherwise. We assume a priori that each  $I_i$  has a small probability  $p_0$  of being equal to 1. With this set-up, we can write down the joint posterior distribution:

$$P(\Theta, \mathbf{I} | R) \propto P(R | \mathbf{I}, \Theta) P(\mathbf{I} | \Theta) f_0(\theta), \quad (3.10)$$

where

$$P(\mathbf{I} | \Theta) \propto \prod_{i=1}^n p_0^{I_i} (1 - p_0)^{1-I_i}.$$

If we do not allow overlapping motifs, we need the restriction that in  $\mathbf{I}$  there is no pair  $I_i = 1$  and  $I_j = 1$  with  $|i - j| < w$ .

With a prior Dirichlet( $\alpha$ ) for each  $\theta_j$ , we can easily obtain the posterior distribution of the  $\theta_j$  if we know the positions of the motif (Liu *et al.*, 1995). Thus, a simple Gibbs sampling algorithm can be designed to draw  $\Theta$  and  $\mathbf{I}$  from (3.10):

- For a current realization of  $\Theta$ , we update each  $I_i$ ,  $i = 1, \dots, n$ , by a random draw from its conditional distribution,  $P(I_i | \mathbf{I}_{[-i]}, R, \Theta)$ , where

$$\frac{P(I_i = 1 | \mathbf{I}_{[-i]}, R, \Theta)}{P(I_i = 0 | \mathbf{I}_{[-i]}, R, \Theta)} = \frac{p_0}{1 - p_0} \prod_{k=1}^w \left( \frac{\theta_{k, r_i+k-1}}{\theta_{0, r_i+k-1}} \right). \quad (3.11)$$

Intuitively, this odds ratio is simply the ‘signal-to-noise’ ratio.



Figure 3.8 A graphical illustration of the repetitive motif model.

- Based on the current value of  $\mathbf{I}$ , we update the profile matrix  $\Theta$  column by column. That is, each  $\theta_j$ ,  $j = 1, \dots, w$ , is drawn from an appropriate posterior Dirichlet distribution determined by  $\mathbf{I}$  and  $R$ .

After a burn-in period (until the sampler stabilizes), we continue to run the sampler for  $m$  iterations and use the average of the  $\Theta$ s to estimate the profile matrix. The estimated  $\Theta$  can then be used to scan the sequence to find the locations of the motif.

### 3.6.2 Block-Motif Model with a Markovian Background

It is well known that the i.i.d. multinomial distribution cannot model a DNA or protein sequence (background) well. In particular, Liu *et al.* (2001; 2002) showed that using a second- or third-order Markov model for the background can significantly improve the motif-finding capability of the method. For simplicity, we describe here only the first-order Markov background model, where a  $4 \times 4$  transition matrix,  $B_0 = (\beta_{jj'})$ , needs to be estimated. Since the total number of base pairs occupied by motif sites is a very small fraction of all the base pairs in  $R$ , we may estimate  $B_0$  from the raw data directly, pretending that the whole sequence of  $R$  is homogeneous. With this approximation, the transition probabilities can be estimated as  $\hat{\beta}_{j_1 j_2} = n_{j_1 j_2} / n_{j_1 \cdot}$ , similarly to Section 3.3. We may then treat  $B_0$  as a known parameter.

The joint posterior distribution of  $(\Theta, \mathbf{I})$  in this case differs from (3.10) only in the description of the residues in the background. The Gibbs sampler can also be implemented similarly, with a slight modification in  $P(I_i | \mathbf{I}_{[-i]}, R, \Theta)$ . That is, conditional on  $R$ , we slide  $\Theta$  through the whole sequence, position by position, to update  $I_i$  according to a random draw from  $P(I_i | \mathbf{I}_{[-i]}, R, \Theta)$ , which satisfies

$$\frac{P(I_i = 1 | \mathbf{I}_{[-i]}, R, \Theta)}{P(I_i = 0 | \mathbf{I}_{[-i]}, R, \Theta)} = \frac{p_0}{1 - p_0} \prod_{k=1}^w \left( \frac{\theta_{k, r_i+k-1}}{\hat{\beta}_{r_i+k-2, r_i+k-1}} \right).$$

For given  $\mathbf{I}$ , we update the profile matrix  $\Theta$  in the same way as in Section 3.6.1.

### 3.6.3 Block-Motif Model with Inhomogeneous Background

It has long been noticed that DNA sequences contain regions of distinctive compositions. As discussed in Section 3.3, an HMM can be employed to delineate a sequence with  $k$  types of regions. Suppose we decide to use an HMM to model sequence inhomogeneity. As already mentioned, we may estimate the background model parameters using all the sequences by the methods in Section 3.3, pretending that  $R$  does not contain any motifs. Then, we can treat these parameters as known at the estimated values, and use one of the following two strategies to modify the odds-ratio formula (3.11).

In the first strategy, we treat each position in the sequence as a ‘probabilistic base pair’ (i.e., having probabilities of being one of the four letters) and derive its frequency. That is, we need to find  $\theta_{ij}^* = p(r_i^* = j | R)$  for a future  $r_i^*$  and then treat residue  $r_i$  in the background as an independent observation from  $\text{Multinom}(\theta_i^*)$ , with  $\theta_i^* = (\theta_{ia}^*, \dots, \theta_{it}^*)$ . But this computation is nontrivial because

$$\theta_{ij}^* = P(r_i^* = j | R) = \theta_{0j} P(h_i = 0 | R) + \theta_{1j} P(h_i = 1 | R), \quad (3.12)$$

where  $P(h_i)$  can be computed via a recursive procedure similar to (3.3). More precisely, in addition to the series of forward functions  $F_i$ , we can define the backward functions  $B_i$ . Let  $B_n(h) = \sum_{h_n} \tau_{hh_n} \theta_{h_n r_n}$ , and let

$$B_k(h) = \sum_{h_k} \{ \tau_{hh_k} \theta_{h_k r_k} B_{k+1}(h_k) \}, \quad \text{for } k = n-1, \dots, 1. \quad (3.13)$$

Then we have

$$P(h_i = 1 | R) = \frac{F_i(1)B_{i+1}(1)}{F_i(1)B_{i+1}(1) + F_i(0)B_{i+1}(0)}. \quad (3.14)$$

This is the *marginal* posterior distribution of  $h_i$  and can be used to predict whether position  $i$  is in state 1 or 0. Thus, in the Gibbs sampling algorithm we only need to modify the denominator of the right-hand side of (3.11) to  $\prod_{k=i}^{i+w-1} \theta_{k,r_k}^*$ .

In the second strategy, we seek to obtain the joint probability of a whole segment,  $R_{[i:i+w-1]} \equiv (r_i, \dots, r_{i+w-1})$ , conditional on the remaining part of the sequence. Then we modify (3.11) accordingly. Clearly, compared with the first strategy, this second one is more faithful to the compositional HMM assumption. The required probability evaluation can be achieved as follows:

$$\begin{aligned} P(R_{[i:i+w-1]} | R_{[1:i-1]}, R_{[i+w:n]}) &= \frac{P(R)}{P(R_{[1:i-1]}, R_{[i+w:n]})} = \frac{P(R)}{\sum_{\mathbf{h}} P(R_{[1:i-1]}, R_{[i+w:n]}, \mathbf{h})} \\ &= \frac{F_n(0) + F_n(1)}{\sum_{h_1, \dots, h_w} F_i(h_1) \tau_{h_1 h_2} \cdots \tau_{h_{w-1} h_w} B_{i+w}(h_w)}, \end{aligned} \quad (3.15)$$

where the denominator can also be obtained via recursions.

### 3.6.4 Extension to Multiple Motifs

Previously, we have assumed that there is only one kind of motif in the sequence and the prior probability for each  $I_i = 1$  is known as  $p_0$ . Both of these assumptions can be relaxed. Suppose we want to detect and align  $m$  different types of motifs of lengths  $w_1, \dots, w_m$ , respectively, and each occurring an unknown number of times in  $R$ . We can similarly introduce the indicator vector  $\mathbf{I}$ , where  $I_i = j$  indicates that an element from motif  $j$  starts at position  $i$ , and  $I_i = 0$  means that no elements start from position  $i$ . For simplicity, we consider only the independent background model.

Let  $P(I_i = j) = \varepsilon_j$ , where  $\varepsilon_0 + \dots + \varepsilon_m = 1$ , be an unknown probability vector. Given what is known about the biology of the sequences being analyzed, a crude guess  $k_j$  for the number of elements for motif  $j$  is usually possible. Let  $k_0 = n - k_1 - \dots - k_m$ . We can represent this prior opinion about the number of occurrences of each type of element by a Dirichlet distribution on  $\boldsymbol{\varepsilon} = (\varepsilon_0, \dots, \varepsilon_m)$ , which has the form  $\text{Dirichlet}(b_0, \dots, b_m)$  with  $b_j = J_0(k_j/n)$ , where  $J_0$  represents the ‘weight’ (or ‘pseudo-counts’) to be put on this prior belief. Then the same predictive updating approach as illustrated in Section 3.6.1

can be applied. To be precise, the update formula (3.11) for  $\mathbf{I}$  is changed to

$$\frac{P(I_i = j \mid \mathbf{I}_{[-i]}, R)}{P(I_i = 0 \mid \mathbf{I}_{[-i]}, R)} = \frac{\varepsilon_j}{\varepsilon_0} \prod_{k=1}^{w_j} \left( \frac{\theta_{k, r_{i+k-1}}^{(j)}}{\theta_{0, r_{i+k-1}}} \right),$$

where  $\Theta^{(j)} = [\theta_1^{(j)}, \dots, \theta_{w_j}^{(j)}]$  is the profile matrix for the  $j$ th motif. Conditional on  $\mathbf{I}$ , we can then update  $\boldsymbol{\varepsilon}$  by a random sample from  $\text{Dirichlet}(b_0 + n_0, \dots, b_m + n_m)$ , where  $n_j$  ( $j > 0$ ) is the number of motif type  $j$  found in the sequence, i.e., the total number of  $i$  such that  $I_i = j$ , and  $n_0 = n - \sum n_j$ .

### 3.7 DISCUSSION

This chapter has reviewed some of the statistical models used in biological sequence analysis, the corresponding Bayesian formulations, and related computational strategies. As in classical statistics, optimization has been the primary tool in computational biology, in which point estimates of very high-dimensional objects obtained by dynamic programming or other clever computational methods are used. Characterizations of uncertainty in these estimates are mostly limited to simple significance tests or completely ignored. The removal of nuisance parameters is also problematic, most frequently being the *profile likelihood* method in which the nuisance parameters are fixed at their best estimates. In comparison, the Bayesian method has no difficulties in these important aspects: the uncertainty in estimation is addressed by posterior calculations and the nuisance parameters are removed by summation and integration. In exchange for these advantages, however, one needs to set prior distributions and overcome computational hurdles, none of which are trivial in practice. Recursion-based Bayesian algorithms generally have time and space requirements of the same order as their dynamic programming counterparts. For those problems where there is no polynomial time solution, MCMC methods (and other Monte Carlo methods) provide an effective means to implement Bayesian analysis.

#### Acknowledgments

This work was supported in part by the National Science Foundation grant DMS-0204674 and the National Institutes of Health grant R01 HG02518-01.

## APPENDIX: MARKOV CHAIN MONTE CARLO METHODS

### Metropolis–Hastings Algorithm

Let  $\pi(\mathbf{x}) = c \exp\{-h(\mathbf{x})\}$  be the target distribution with unknown constant  $c$ . Metropolis *et al.* (1953) introduced the fundamental idea of Markov chain sampling and prescribed the first general construction of such a chain. Hastings (1970) later provided an important generalization. Starting with any configuration  $\mathbf{x}^{(0)}$ , the Metropolis–Hastings algorithm evolves from the current state  $\mathbf{x}^{(t)} = \mathbf{x}$  to the next state  $\mathbf{x}^{(t+1)}$  as follows:

- Propose a new state  $\mathbf{x}'$  which can be viewed as a small and random ‘perturbation’ of the current state. More precisely,  $\mathbf{x}'$  is generated from a *proposal* function  $T(\mathbf{x}^{(t)} \rightarrow \mathbf{x}')$  (i.e., it is required that  $T \geq 0$  and  $\sum_{\text{all } \mathbf{y}} T(\mathbf{x} \rightarrow \mathbf{y}) = 1$  for all  $\mathbf{x}$ ) determined by the user.
- Compute the Metropolis ratio

$$r(\mathbf{x}, \mathbf{x}') = \frac{\pi(\mathbf{x}')T(\mathbf{x}' \rightarrow \mathbf{x})}{\pi(\mathbf{x})T(\mathbf{x} \rightarrow \mathbf{x}')}. \quad (3.16)$$

- Generate a random number  $u \sim \text{Unif}[0,1]$ .
  - Let  $\mathbf{x}^{(t+1)} = \mathbf{x}'$  if  $u \leq r(\mathbf{x}, \mathbf{x}')$ .
  - Let  $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)}$  otherwise.

A better-known form of the Metropolis algorithm is described as iterating the following steps: (a) a small random perturbation of the current configuration is made; (b) the ‘gain’ (or loss) in an objective function (i.e.,  $-h(\mathbf{x})$ ) resulting from this perturbation is computed; (c) a random number  $U$  is generated independently; and (d) the new configuration is accepted if  $\log(U)$  is smaller than or equal to the ‘gain’, and is rejected otherwise. The well-known *simulated annealing* algorithm (Kirkpatrick *et al.*, 1983) is built upon this basic Metropolis iteration with the additional twist of including an adjustable exponential scaling parameter in the objective function (i.e.,  $\pi(\mathbf{x})$  is scaled to  $\pi^\alpha(\mathbf{x})$  and  $\alpha \rightarrow 0$ ). Metropolis *et al.* (1953) restricted their choices of the ‘perturbation’ function to the symmetric ones, i.e.,  $T(\mathbf{x} \rightarrow \mathbf{x}') = T(\mathbf{x}' \rightarrow \mathbf{x})$ . Hastings (1970) generalized the choice of  $T$  to all those that satisfy the property  $T(\mathbf{x} \rightarrow \mathbf{x}') > 0$  if and only if  $T(\mathbf{x}' \rightarrow \mathbf{x}) > 0$ .

Heuristically,  $\pi$  can be seen as a ‘fixed point’ under the Metropolis–Hastings operation in the space of all distributions. It follows from the standard Markov chain theory that if the chain is *irreducible* (i.e., it is possible to go from anywhere to anywhere else in a finite number of steps), *aperiodic* (i.e., there is no parity problem), and not drifting away, then in the long run the chain will settle in its invariant distribution (Liu, 2001). The random samples so obtained are eventually like those drawn directly from  $\pi$ .

### Gibbs Sampler

Suppose  $\mathbf{x} = (x_1, \dots, x_d)$ . In the Gibbs sampler, one randomly or systematically chooses a coordinate, say  $x_1$ , and then updates its value with a new sample  $x'_1$  drawn from the conditional distribution  $\pi(\cdot | \mathbf{x}_{[-1]})$ , where  $\mathbf{x}_{[-A]}$  refers to  $\{x_j, j \in A^c\}$ . Algorithmically, the Gibbs sampler can be implemented as follows.

For the *random scan* Gibbs sampler, suppose that currently  $\mathbf{x}^{(t)} = (x_1^{(t)}, \dots, x_d^{(t)})$ .

- Randomly select  $i$  from  $\{1, \dots, d\}$  according to a given probability vector  $(\alpha_1, \dots, \alpha_d)$ .
- Let  $x_i^{(t+1)}$  be drawn from the conditional distribution  $\pi(\cdot | \mathbf{x}_{[-i]}^{(t)})$ , and let  $x_{[-i]}^{(t+1)} = \mathbf{x}_{[-i]}^{(t)}$ .

For the *systematic scan* Gibbs sampler, again let the current state be  $\mathbf{x}^{(t)} = (x_1^{(t)}, \dots, x_d^{(t)})$ . Then for  $i = 1, \dots, d$ , we draw  $x_i^{(t+1)}$  from the conditional distribution

$$\pi(x_i | x_1^{(t+1)}, \dots, x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, \dots, x_d^{(t)}).$$

The Gibbs sampler's popularity in the statistics community stems from its extensive use of *conditional distributions* in each iteration. Tanner and Wong's (1987) data augmentation first linked the Gibbs sampling structure with missing-data problems and the EM algorithm. Gelfand and Smith (1990) further popularized the method by pointing out that the conditionals needed in Gibbs iterations are commonly available in many Bayesian and likelihood computations.

### Other Techniques

A major problem with all MCMC algorithms is that they may, for some problems, move very slowly in the configuration space or may be trapped in the region of a local mode. This phenomenon is generally called *slow-mixing* of the chain. When a chain is slow-mixing, estimation based on the resulting Monte Carlo samples becomes very inaccurate. Some recent techniques suitable for designing more efficient MCMC samplers in bioinformatics applications include simulated tempering, parallel tempering, multicanonical sampling, the multiple-try method, and evolutionary Monte Carlo. These and some other techniques are summarized in Liu (2001). Some more discussions and applications of MCMC can also be found in **Chapters 21** and **12**.

xref

## REFERENCES

- Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. (1990). Basic local alignment search tool. *Journal of Molecular Biology* **215**, 403–410.
- Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D.J. (1997). Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research* **25**, 3389–3402.
- Baldi, P., Chauvin, Y., McClure, M. and Hunkapiller, T. (1994). Hidden Markov models of biological primary sequence information. *Proceedings of the National Academy of Sciences (USA)* **91**, 1059–1063.
- Baum, L.E. (1972). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics* **41**, 164–171.
- Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Rapp, B.A. and Wheeler, D.L. (2002). GenBank. *Nucleic Acids Research* **30**, 17–20.
- Bishop, M.J. and Thompson, E.A. (1986). Maximum likelihood alignment of DNA sequences. *Journal of Molecular Biology* **190**, 159–165.
- Burge, C. and Karlin, S. (1997). Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology* **268**, 78–94.
- Cardon, L.R. and Stormo, G.D. (1992). Expectation maximization algorithm for identifying protein-binding sites with variable lengths from unaligned DNA fragments. *Journal of Molecular Biology* **223**, 159–170.
- Churchill, G.A. (1989). Stochastic models for heterogeneous DNA sequences. *Bulletin of Mathematical Biology* **51**, 79–94.
- Churchill, G.A. and Lazareva, B. (1999). Bayesian restoration of a hidden Markov chain with applications to DNA sequencing. *Journal of Computational Biology* **6**, 261–277.
- Dayhoff, M., Eck, R.V. and Park, C.M. (1972). In *Atlas of Protein Sequence and Structure*, Vol. 5, M. Dayhoff, ed. *National Biomedical Research Foundation*, Washington, DC, pp. 89–99.
- Dempster, A.P., Laird, N.M. and Rubin, D.B. (1977). Maximum likelihood estimation from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B* **39**, 1–38.

- Efron, B. (1979). Bootstrap methods: another look at the jackknife. *Annals of Statistics* **7**, 1–26.
- Eddy, S. (1998). Profile hidden Markov models. *Bioinformatics* **14**, 755–763.
- Gelfand, A.E. and Smith, A.F.M. (1990). Sampling-based approach to calculating marginal densities. *Journal of the American Statistical Association* **85**, 398–409.
- Gelman, A., Carlin, J.B., Stern, H.S. and Rubin, D.B. (1995). *Bayesian Data Analysis*. Chapman & Hall, London.
- Gilks, W.R., Richardson, S. and Spiegelhalter, D.J. (1996). *Markov Chain Monte Carlo in Practice*. Chapman & Hall, Boca Raton, Fla.
- Gotoh, O. (1982). An improved algorithm for matching biological sequences. *Journal of Molecular Biology* **162**, 705–708.
- Hastings, W.K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57**, 97–109.
- Henikoff, S. and Henikoff, J.G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences (USA)* **89**, 10915–10919.
- Henikoff, S. and Henikoff, J.G. (1994). Position-based sequence weights. *Journal of Molecular Biology* **243**, 574–578.
- Holmes, I. and Durbin, R. (1998). In *Proceedings of the Second Annual International Conference on Computational Molecular Biology*. S. Istrail, P. Pevzner and M. Waterman, eds. Association for Computing Machinery, New York, pp. 102–108.
- Huang, H., Kao, M.J., Zhou, X., Liu, J.S. and Wong, W.H. (2002). *Identification of transcription factor binding sites using local Markov models*. Technical Report, Department of Statistics, Harvard University.
- Hudak, J. and McClure, M.A. (1999). In *Proceedings of the Pacific Symposium on Biocomputing '99*, R.B. Altman, A.K. Dunker, L. Hunter, T.E. Klein and K. Laudesdale, eds. World Scientific, River Edge, NJ, pp. 138–149.
- Kass, R.E. and Raftery, A.E. (1995). Bayes factors. *Journal of the American Statistical Association* **90**, 773–795.
- Kimura, M. (1983). *The Neutral Theory of Molecular Evolution*. Cambridge University Press, Cambridge.
- Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983). Optimization by simulated annealing. *Science* **220**, 671–680.
- Krogh, A., Brown, M., Mian, S., Sjolander, K. and Haussler, D. (1994a). Protein modeling using hidden Markov models. *Journal of Molecular Biology* **235**, 1501–1531.
- Krogh, A., Mian, I.S. and Haussler, D. (1994b). A hidden Markov model that finds genes in *E. coli* DNA. *Nucleic Acids Research* **22**, 4768–4778.
- Lawrence, C.E. and Reilly, A.A. (1990). An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins* **7**, 41–51.
- Lawrence C.E., Altschul, S.F., Boguski, M.S., Liu, J.S., Neuwald, A.F. and Wootton, J.C. (1993). Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science* **262**, 208–214.
- Liu, J.S. (1994). The collapsed Gibbs sampler with applications to a gene regulation problem. *Journal of the American Statistical Association* **89**, 958–966.
- Liu, J.S. (2001). *Monte Carlo Methods for Scientific Computing*. Springer-Verlag, New York.
- Liu, J.S. and Lawrence, C.E. (1999). Bayesian inference on biopolymer models. *Bioinformatics* **15**, 38–52.
- Liu, J.S., Neuwald, A.F. and Lawrence, C.E. (1995). Bayesian models for multiple local sequence alignment and Gibbs sampling strategies. *Journal of the American Statistical Association* **90**, 1156–1170.
- Liu, J.S., Neuwald, A.F. and Lawrence, C.E. (1999). Markovian structures in biological sequence alignments. *Journal of the American Statistical Association* **94**, 1–15.

- Liu, X.S., Brutlag, D.L. and Liu, J.S. (2001). In *Proceedings of the Pacific Symposium on Biocomputing*, R.B. Altman, A.K. Dunker, L. Hunter, K. Lauderdale and T.E. Klein, eds. World Scientific, Singapore, pp. 127–138.
- Liu, X.S., Brutlag, D.L. and Liu, J.S. (2002). An algorithm for finding protein–DNA binding sites with applications to chromatin immunoprecipitation microarray experiments. *Nature Biotechnology* **20**, 835–839.
- Logvinenko, T. (2002). Sequential Monte Carlo and Dirichlet mixtures for extracting protein alignment models. PhD thesis, Stanford University.
- Lowe, T.M. and Eddy, S.R. (1997). tRNA-scan-SE: A program for improved detection of transfer RNA genes in genomic sequence. *Nucleic Acids Research* **5**, 955–964.
- Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics* **21**, 1087–1091.
- Needleman, S.B. and Wunsch, C.D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* **48**, 443–453.
- Neuwald, A.F., Liu, J.S. and Lawrence, C.E. (1995). Gibbs motif sampling: detection of bacterial outer membrane protein repeats. *Protein Science* **4**, 1618–1632.
- Neuwald, A.F., Liu, J.S., Lipman, D.J. and Lawrence, C.E. (1997). Extracting protein alignment models from the sequence database. *Nucleic Acids Research* **25**, 1665–1677.
- Pearson, W.R. and Lipman, D.J. (1988). Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences (USA)* **85**, 2444–2448.
- Rabiner, L.R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* **77**, 257–286.
- Roth, F.P., Hughes, J.D., Estep, P.W. and Church, G.M. (1998). Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nature Biotechnology* **16**, 939–945.
- Saitou, N. and Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution* **4**, 406–425.
- Sankoff, D. (1972). Matching sequences under deletion/insertion constraints. *Proceedings of the National Academy of Sciences (USA)* **69**, 4–6.
- Schmidler, S.C., Liu, J.S. and Brutlag, D.L. (2000). Bayesian segmentation of protein secondary structure. *Journal of Computational Biology* **7**, 233–248.
- Sjolander, K., Karplus, K., Brown, M., Hughey, R., Krogh, A., Mian, I.S. and Haussler, D. (1996). Dirichlet mixtures: A method for improving detection of weak but significant protein sequence homology. *Computer Applications in the Biosciences* **12**, 327–345.
- Smith, T.F. and Waterman, M.S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology* **147**, 195–197.
- Stormo, G.D. and Hartzell, G.W. (1989). Identifying protein-binding sites from unaligned DNA fragments. *Proceedings of the National Academy of Sciences (USA)* **86**, 1183–1187.
- Tanner, M.A. and Wong, W.H. (1987). The calculation of posterior distributions by data augmentation (with discussion). *Journal of the American Statistical Association* **82**, 528–550.
- Thompson, J.D., Higgins, D.G. and Gibson, T.J. (1994a). CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acids Research* **22**, 4673–4680.
- Thompson, J.D., Higgins, D.G. and Gibson, T.J. (1994b). Improved sensitivity of profile searches through the use of sequence weights and gap excision. *Computer Applications in the Biosciences* **10**, 19–29.
- Thorne, J.L., Kishino, H. and Felsenstein, J. (1991). An evolutionary model for maximum likelihood alignment of DNA sequences. *Journal of Molecular Evolution* **33**, 114–124.
- Webb, B.M., Liu, J.S. and Lawrence, C.E. (2002). BALSAs: Bayesian algorithm for local sequence alignment. *Nucleic Acids Research* **30**, 1268–1277.

- Zhu, J., Liu, J.S. and Lawrence, C.E. (1998). Bayesian adaptive sequence alignment algorithms. *Bioinformatics* **14**, 25–31.
- Zuker, M. (1989). Computer prediction of RNA structure. *Methods in Enzymology* **180**, 262–288.

FIRST PAGE PROOFS

**QUERIES TO BE ANSWERED BY AUTHOR (SEE MARGINAL MARKS)**

**IMPORTANT NOTE: Please mark your corrections and answers to these queries directly onto the proof at the relevant place. Do NOT mark your corrections on this query sheet.**

---

| Query No. | Query   |
|-----------|---|
| Q1        | Throne <i>et al.</i> (1992) is not listed in the References.  |
| Q2        | You have Kimura (1980) here but Kimura (1983) in the References. The Cambridge University Press book is cited elsewhere in the Handbook as 1983, and other authors cite a journal paper by Kimura from 1980 (A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. <i>Journal of Molecular Evolution</i> <b>16</b> , 111–120). |
| Q3        | Sjolander <i>et al.</i> (1998) is not listed in the References.   |

---

FIRST PAGE PROOFS