Part I

Studies in Probabilistic Sequence Alignment

Chapter 2

Bayesian Methods for Hidden Markov Models in Biological Sequence Analysis

2.1 Introduction

A wide variety of score-based dynamic programming algorithms are commonly used for sequence alignment [AG96, PL88, BS87, LAK89, THG94a]. As early as 1992, Anders Krogh pointed out that the dynamic programming methods being used could be viewed as special cases of the Viterbi algorithm, widely used in speech recognition. The premise of this algorithm is that the sequences were generated by a probabilistic Markov model and that the exact state path is hidden from view, but can be reconstructed by inference. The recursive algorithm for performing this inference is an example of dynamic programming [Kro94].

Casting sequence alignment as an HMM problem does not avoid the question of what scores are significant. However, it does connect sequence alignment to a large published literature on HMM methods [DEKM98, Rab89, Mac96b]. This research puts the scores into context, explores how to choose the best scores for a particular problem, demonstrates how scores can be combined and opens up a wide range of new algorithms. Suddenly bioinformatics has a solid link to machine learning.

This chapter is a review of hidden Markov models in bioinformatics, of the main algorithms and techniques that can be used for HMMs and of certain properties they have. Sections 2.2 to 2.4 introduce notation and concepts that are used throughout the dissertation. Sections 2.5 to 2.7 are more speculative and less relevant to the rest of the dissertation.

2.2 Notation

In this section, hidden Markov models will be treated as machines that generate a single sequence, though it is only slightly more complicated to write down a definition of a "pair HMM" that generates a pair of sequences, and by extension, a "multiple HMM" that generates a whole set of sequences (this latter would be suitable for multiple alignment) [DEKM98].

An HMM has S states. The transition from state a to state b, labelled with residue X (with $X \in \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ for DNA, for example), has probability t_{abX}. (It is conventional to talk of the transition "emitting" residue X and this convention will be used from now on.) The probabilities of all the transitions leaving a particular state must sum to 1. Two states have special names: the begin state $\mathcal{B}$ and the end state $\mathcal{E}$.

Denote the set of all the t_{abX} values by $\mathbf{t}$. This set $\mathbf{t}$ is often called the parameterisation of the HMM, or equivalently a point in the parameter space of the HMM.

Suppose that $\mathbf{X}$ is a sequence with L residues, whose i'th residue is X_i (the bold typeface $\mathbf{X}$ indicates the entire sequence and the light typeface X_i indicates an individual residue in the sequence). It is possible to trace a path of L steps through the HMM so that the i'th step uses a transition with residue label X_i. Such a path can also be called an *alignment* of the sequence to the HMM, because it aligns each residue in the sequence to a transition in the HMM. If a path begins in the begin state $\mathcal{B}$ and winds up in the end state $\mathcal{E}$, it will be called a *complete* path.

Call the alignment path $\mathbf{a}$, and suppose that at the i'th step the path is in state a_i (the path starts in state a_0). The i'th step in the path thus uses a transition from state a_{i-1} to state a_i and, to be consistent with the sequence $\mathbf{X}$, this transition must emit residue X_i. The corresponding transition probability is $t_{a_{i-1}a_iX_i}$. The joint likelihood of the sequence and the alignment is defined to be the product of all the transition probabilities along the path:

$$\Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}] = \prod_{i=1}^{L} t_{a_{i-1}a_iX_i} \tag{2.1}$$

The likelihood of the sequence is the sum of the joint likelihoods of all complete paths of the same length as the sequence:

$$\Pr[\mathbf{X}|\mathbf{t}] = \sum_{\mathbf{a}:|\mathbf{a}|=L} \Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}] \tag{2.2}$$

The model is "hidden" because one typically knows the sequence **X** but not the alignment **a**. The main HMM algorithms address the problem of dealing with the missing information and these are reviewed below.

2.2.1 Other formulations of HMMs

Pair HMMs require a little more flexibility in that some transitions only emit residues for one of the two sequences. The most common type of pair HMM is the model for Needleman-Wunsch global alignment with affine gap penalties [NW70], which has three states (in addition to the start and end states). This model is shown in Figure 2.1. The three states include a match state and two indel states. Transitions into the match state emit paired residues in both sequences, whereas transitions into the indel states only emit residues in one or other of the two sequences. The indel states are often called "insert" and "delete" to distinguish each other. Transitions from the match to either of the indel states corresponds to opening a gap, so their probabilities are associated with the gap-opening penalty; looping transitions within the indel states correspond to gap-extension penalties. The probability distribution for paired match emissions corresponds to the substitution matrix. There is a more detailed discussion of the Needleman-Wunsch model in Chapter 3.

Alignments of pairs of sequences to pair HMMs specify residue→transition mappings for *both* the sequences. They therefore also specify which pairs of residues in the two sequences are aligned together. This is the commonly understood definition of sequence alignment.

A variant of the Needleman-Wunsch HMM used for local alignment - corresponding to the Smith-Waterman algorithm [SW81] - is actually a generalised HMM. Generalised HMMs are discussed in more detail in Section 2.8.

The pair HMM software described in Appendix A implements the kinds of

HMM described above, together with a limited class of generalised HMMs (including the Smith-Waterman model and the Bayesian block aligner mentioned in Section 2.8 of this chapter).

2.3 Aligning sequences to HMMs

Usually the sequence $\mathbf{X}$ is known and the alignment $\mathbf{a}$ is "missing information". Two useful tricks are: (i) to find the most likely alignment $\mathbf{a}$; (ii) to find the sum of the likelihoods of all alignments $\mathbf{a}$ (the sequence likelihood $\Pr[\mathbf{X}|\mathbf{t}]$ defined in (2.2)). (i) is a classic "maximum likelihood" approach, whereas (ii) is necessary if Bayes' rule is to be applied.

These tasks are accomplished using the Viterbi and Forward algorithms, respectively. Both are dynamic programming algorithms.

2.3.1 Maximising the alignment likelihood: the Viterbi algorithm

The Viterbi algorithm finds the most likely alignment $\mathbf{a}$ consistent with an observed sequence $\mathbf{X}$ [Vit67]. It works by building up the sequence $\mathbf{X}$ one residue at a time, so that there is a series of subsequences starting with nothing at $i = 0$ and ending up with the full sequence when $i = L$. The i'th subsequence corresponds to the first i residues of $\mathbf{X}$.

The optimal path $\mathbf{a}$ is built up step-by-step at the same time as the sequence, but the "missing information" problem is addressed by keeping track of S different optimal paths (one for each state of the model) at each value of i. It is not necessary to keep track of any more paths than this because the Markov nature of the model means that best path of length i for some state b contains the best path of length $i-1$ for some state a; and so on down to $i = 0$.

This can be expressed more formally. Let $\{\mathbf{a}_{i,b}\}$ be the set of all the paths of length i that start in the begin state $\mathcal{B}$ and end in state b. Let $V_{i,b}$ be the maximum likelihood of all these paths, i.e.

$$V_{i,b} = \max_{\mathbf{a} \in \{\mathbf{a}_{i,b}\}} \Pr[\mathbf{a}, \mathbf{X} | \mathbf{t}] \tag{2.3}$$

(The likelihood of the complete Viterbi path is then $V_{L,\mathcal{E}}$, where $\mathcal{E}$ is the end state and L is the sequence length.)

Let the penultimate state of $V_{i,b}$ be a. The first $i-1$ steps of the $V_{i,b}$ path must also be an optimal path for some state a, so:

$$V_{i,b} = \max_a [t_{abX_i} V_{i-1,a}] \tag{2.4}$$

Equation (2.4) defines a recursion for the maximal path likelihood. Together with the boundary condition:

$$V_{a,0} = \begin{cases} 1 & \text{if } a = \mathcal{B} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

which just means "start in the start state", this recursion is the Viterbi algorithm. The algorithm only calculates the likelihoods of the paths; the paths themselves can be reconstructed by traceback. The maximal likelihoods $V_{i,b}$ form an $L \times S$ array of "cells" called the "dynamic programming matrix".

For sequence alignment, the algorithm is usually expressed in terms of the logs of the likelihoods rather than the likelihoods themselves. This is both intuitively natural (since log-likelihoods are additive, which corresponds better to the idea of scores) and more computationally well-behaved (since it avoids underflow problems).

2.3.2 Summing alignment likelihoods: the Forward algorithm

The Viterbi algorithm finds the likelihood of the most likely path consistent with the observed sequence (and by traceback, the path itself). The Forward algorithm finds the sum of the likelihoods of all paths consistent with the observed sequence (as in (2.2)) and is obtained essentially by replacing the max in equations (2.3)-(2.5) with a sum.

$$V_{i,b} = \max_{\mathbf{a} \in \{\mathbf{a}_{i,b}\}} \Pr\left[\mathbf{a}, \mathbf{X} | \mathbf{t}\right] \tag{2.3}$$

(The likelihood of the complete Viterbi path is then $V_{L,\mathcal{E}}$, where $\mathcal{E}$ is the end state and L is the sequence length.)

Let the penultimate state of $V_{i,b}$ be a. The first $i-1$ steps of the $V_{i,b}$ path must also be an optimal path for some state a, so:

$$V_{i,b} = \max_{a} \left[t_{abX_i} V_{i-1,a}\right] \tag{2.4}$$

Equation (2.4) defines a recursion for the maximal path likelihood. Together with the boundary condition:

$$V_{a,0} = \begin{cases} 1 & \text{if } a = \mathcal{B} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

which just means "start in the start state", this recursion is the Viterbi algorithm. The algorithm only calculates the likelihoods of the paths; the paths themselves can be reconstructed by traceback. The maximal likelihoods $V_{i,b}$ form an $L \times S$ array of "cells" called the "dynamic programming matrix".

For sequence alignment, the algorithm is usually expressed in terms of the logs of the likelihoods rather than the likelihoods themselves. This is both intuitively natural (since log-likelihoods are additive, which corresponds better to the idea of scores) and more computationally well-behaved (since it avoids underflow problems).

2.3.2 Summing alignment likelihoods: the Forward algorithm

The Viterbi algorithm finds the likelihood of the most likely path consistent with the observed sequence (and by traceback, the path itself). The Forward algorithm finds the sum of the likelihoods of all paths consistent with the observed sequence (as in (2.2)) and is obtained essentially by replacing the max in equations (2.3)-(2.5) with a sum.

Define $F_{i,b}$ to be the sum of the likelihoods of all the paths of length i that end in state b, i.e.:

$$F_{i,b} = \sum_{\mathbf{a} \in \{\mathbf{a}_{i,b}\}} \Pr[\mathbf{a}, \mathbf{X} | \mathbf{t}] \tag{2.6}$$

The $F_{i,b}$ form another $L \times S$ dynamic programming matrix. The Forward algorithm for calculating them is:

$$F_{i,b} = \sum_a t_{abX_i} F_{i-1,a} \tag{2.7}$$

$$F_{0,a} = \begin{cases} 1 & \text{if } a = \mathcal{B} \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

From the point of view of scores, the transition from equations (2.3)-(2.5) to equations (2.6)-(2.8) correspond to replacing the $z = \max(x, y)$ rule in the dynamic programming for choosing between two scores x and y with a modified rule $z = \max(x, y) + B(|x - y|)$, where $B = \log(1 + \exp -|x - y|)$ is a "bonus" function that rewards similar scores. When $x \simeq y$, then $B \sim \log 2 - \frac{|x-y|}{2}$, i.e. the similarity bonus directly penalises the difference in scores when the difference is small; but when $\max(x, y) \gg \min(x, y)$ then $B \sim \exp -|x - y|$, i.e. the similarity bonus decays rapidly when the difference in scores is large.

2.3.3 Posterior probabilities of alignments: the Forward-Backward algorithm

Given the joint likelihood $\Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}]$ and the sequence likelihood $\Pr[\mathbf{X}|\mathbf{t}]$ (the latter of which is calculated using the Forward algorithm), a posterior probability for the path can be calculated using Bayes' rule:

$$\Pr[\mathbf{a}|\mathbf{X}, \mathbf{t}] = \frac{\Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}]}{\Pr[\mathbf{X}|\mathbf{t}]}$$

The number of paths aligning a sequence of length L to a model of size S is $\sim S^L$. There are only $L \times S$ entries in the dynamic programming matrix, each representing the alignment of an individual residue to an individual state. It is usually sufficient to work with these rather than the entire path distribution.

Let the notation $(i \diamond b)$ mean "residue X_i is aligned to a transition that ends in state b". The posterior probability of $(i \diamond b)$ is defined as the sum of the posterior probabilities of all the paths $\mathbf{a}$ that include $(i \diamond b)$ (i.e. all the paths that align residue i to a transition that ends in b):

$$\begin{aligned}
\Pr\left[(i \diamond b)|\mathbf{X},\mathbf{t}\right] &= \sum_{\mathbf{a}:(i\diamond b)\in\mathbf{a}} \Pr\left[\mathbf{a}|\mathbf{X},\mathbf{t}\right] \\
&= \sum_{\mathbf{a}:(i\diamond b)\in\mathbf{a}} \frac{\Pr\left[\mathbf{a},\mathbf{X}|\mathbf{t}\right]}{\Pr\left[\mathbf{X}|\mathbf{t}\right]} \\
&= \sum_{\mathbf{a}_{i,b},\bar{\mathbf{a}}_{i,b}} \frac{\Pr\left[\mathbf{a}_{i,b},\mathbf{X}|\mathbf{t}\right]\Pr\left[\bar{\mathbf{a}}_{i,b},\mathbf{X}|\mathbf{t}\right]}{\Pr\left[\mathbf{X}|\mathbf{t}\right]} \\
&= \frac{\left(\sum_{\mathbf{a}_{i,b}} \Pr\left[\mathbf{a}_{i,b},\mathbf{X}|\mathbf{t}\right]\right)\left(\sum_{\bar{\mathbf{a}}_{i,b}} \Pr\left[\bar{\mathbf{a}}_{i,b},\mathbf{X}|\mathbf{t}\right]\right)}{\Pr\left[\mathbf{X}|\mathbf{t}\right]} \\
&= \frac{F_{i,b}B_{i,b}}{F_{L,\mathcal{E}}} \qquad (2.9)
\end{aligned}$$

where $\mathbf{a}_{i,b}$ is a path of length i ending in state b as before and $\bar{\mathbf{a}}_{i,b}$ is a path of length $L-i$ that *starts* in state b, continuing on to the end state $\mathcal{E}$. The $B_{i,b}$ are called the Backward sums; they are defined as the sums of the likelihoods of all the paths $\bar{\mathbf{a}}_{i,b}$ and may be computed by flipping equations (2.6)-(2.8) in the i-direction. The algorithm for calculating the posterior probabilities $\Pr\left[(i \diamond b)|\mathbf{X},\mathbf{t}\right]$ is called the Forward-Backward algorithm [DEKM98].

2.3.4 Comparing alignments

There are various ways to compare two alignments quantitatively. Perhaps the simplest metric is the *overlap*, which counts the number of residues that both alignments agree on as being aligned to the same state of the HMM [DEKM98]. A related method just counts residues aligned to a particular subset $\{\varsigma\}$ of states. This will be referred to as the *partial overlap*.

Many pair HMMs allow pairs of residues in the two sequences to be aligned to the same state. When counting the number of residue-to-state mappings that pair HMM alignments agree on, it is common to require that *both* residues in the pair are aligned to the same state, in both alignments. This is consistent with the view that such residues are aligned to each other, rather than to a common state.

The *fractional overlap* is just the overlap divided by the total number of residues in the sequence. A *partial fractional overlap* can also be defined, by only counting residues that are aligned to a subset $\{\varsigma\}$ of states, as before. For the partial fractional overlap it is no longer unambiguous what the total number of $\{\varsigma\}$-labellings should be; a choice must be made as to which alignment is the reference alignment. The partial fractional overlap for a pair HMM, counting only match states, is called the *fidelity* [HL96].

More sophisticated measures of alignment similarity include *edit distance* [SK83] and *shift score*, which is rather like a length-normalised edit distance [CK98].

The edit distance, the overlap and the partial overlap are all additive functions. A function $F(\mathbf{a}_\alpha, \mathbf{a}_\beta)$ between two alignments $\mathbf{a}_\alpha$ and $\mathbf{a}_\beta$ of the same sequence S is additive if, when the sequence is split into subsequences S_1 and S_2 (and the alignments split into $\mathbf{a}_{\alpha 1}$, $\mathbf{a}_{\alpha 2}$, $\mathbf{a}_{\beta 1}$ and $\mathbf{a}_{\beta 2}$), then the sum of the parts $F(\mathbf{a}_{\alpha 1}, \mathbf{a}_{\beta 1}) + F(\mathbf{a}_{\alpha 2}, \mathbf{a}_{\beta 2})$ is equal to the whole $F(\mathbf{a}_\alpha, \mathbf{a}_\beta)$. Additivity is a useful property for an alignment accuracy measure since it means the alignment that optimises the measure with respect to the posterior distribution can be found using a variant of the Viterbi algorithm. This kind of "optimal accuracy" algorithm is an application of Bayesian decision theory. An example of such an algorithm that uses the fidelity as an accuracy measure has been proposed [DEKM98] and is explored further in Chapter 2.

Alignment accuracy issues are dealt with in more depth in Chapter 3, in which simulation results for the accuracy of the Viterbi algorithm for a Needleman-

Wunsch pair HMM are given. It is shown that the accuracy of the algorithm can be predicted, both for the average case (using entropy methods) and for specific cases (using posterior probabilities). The performance of the "optimal accuracy" algorithm is also assessed in this chapter. A program to calculate posterior probability tables and implement the optimal accuracy algorithm for profile HMMs is presented in Chapter 4.

2.4 Hidden Markov models in molecular evolution

The most commonly asked questions in molecular evolution involve the relative or absolute dates of divergence of biological sequences. These questions are often answered by fitting time-dependent models to alignments between the sequences (see e.g. [DEKM98]). A natural extension is to take advantage of the power of HMM algorithms to sum over all alignments by allowing the transition probabilities t_{abX} of a pair HMM to be functions of a time parameter τ, and using optimisation algorithms to find the maximum-likelihood value of τ. This approach was proposed by Thorne *et al* [TKF91, TKF92].

There are two main things that a pair HMM of this kind has to get right: the substitution probabilities and the gap probabilities. These will be covered in turn.

2.4.1 Time-dependent substitution matrices

The use of time-dependent substitution matrices predates the use of HMMs to sum over all alignments. The basic idea is that the four nucleotides (or twenty amino acids) are states in a completely interconnected Markov chain; transitions between the states correspond to residue substitutions and the more time that goes by, the more chance there is of making a substitution. In fact the PAM matrices are an example of this kind of matrix: the PAM_τ matrix is just the PAM_1 matrix raised to the τ'th power [DSO78].

A generalisation allowing τ to take continuous values (rather than just discrete ones) proceeds as follows [KT75]: let $P_{XY}(\tau)$ be the (time-dependent) probability that residue X is found to be aligned to residue Y, so that the P_{XY} matrix at time zero is the $A \times A$ identity matrix $\mathbf{P}(0) = \mathbf{I}$ (where $A = 4$ for nucleotides and $A = 20$ for amino acids). Define the rate matrix $\mathbf{R}$ by $\frac{d}{d\tau}\mathbf{P} = \mathbf{RP}$. Suppose that the eigenvalues of $\mathbf{R}$ are $\{\lambda_X\}$ and that the associated right eigenvectors are $\{\mathbf{u_X}\}$; then the solution to the ordinary differential equation can be written $\mathbf{P} = \mathbf{U}\mathbf{\Lambda}(\tau)\mathbf{U}^{-1}$, where $\mathbf{\Lambda}(\tau)$ is the diagonal matrix $\Lambda_{XY} = \delta_{XY} \exp[\lambda_X \tau]$.

A general $A \times A$ rate matrix can have $A^2 - A$ free parameters, but usually a smaller parameter set is used. The simplest model would use one parameter (apart from the time τ), corresponding to the substitution rate - essentially a choice of scale for the time parameter. For nucleotide substitutions, this is called the Jukes-Cantor model [JC69]. The next simplest is Kimura's two-parameter model [Kim80], which allows different rates for "transitions" (substitutions within the purine (A,G) and pyrimidine (C,T) groups) and "transversions" (substitutions between those groups). These are observed to occur at different rates in nature, with transitions being more common.

Both the Kimura and Jukes-Cantor models assume a uniform background distribution over nucleotides. This is not the case in real organisms. For the work described in this dissertation, a modified model due to Hasegawa *et al* was used [HKY85]. This model allows for a non-uniform background nucleotide distribution. The equations for $P_{AX}(t)$ under the Hasegawa model are:

$$P_{AA} = f_A(1 + \frac{f_Y}{f_R}\exp[-s_2 t]) + \frac{f_G}{f_R}\exp[-(f_Y s_2 + f_R s_1)t]$$

$$P_{AG} = f_G(1 + \frac{f_Y}{f_R}\exp[-s_2 t]) - \frac{f_G}{f_R}\exp[-(f_Y s_2 + f_R s_1)t]$$

$$P_{AC} = f_C(1 - \exp[-s_2 t]$$

$$P_{AT} = f_T(1 - \exp[-s_2 t]$$

where f_X is the background frequency of nucleotide X, $f_Y = f_A + f_G$ and $f_R = f_C + f_T$ are (respectively) the purine and pyrimidine frequencies, s_1 is the transition rate and s_2 is the transversion rate. The other probabilities may be obtained by rotating the $\{A, C, G, T\}$.

None of the above models account for the correlations between neighbouring bases that are observed in nature. In this dissertation these effects are ignored, although they are certainly non-negligible in reality [Bul86].

2.4.2 Time-dependent gap probabilities

Thorne *et al* [TKF91, TKF92] proposed a birth-death process of fragment insertion and deletion that supplies transition probabilities for a six-state pair HMM (in the collapsed state space, where each state is allowed a probability distribution over the residues it emits) in terms of a birth and a death rate.

Although the birth-death model is simple, an even simpler model was used for the work described in this dissertation. The HMM used is depicted in Figure 2.1. It is essentially the model for Needleman-Wunsch global alignment with affine gaps; there are three (collapsed) states, one for matches and two for indels. Gaps occur with frequency p_G per residue per strand and their length l is geometrically distributed with parameter p_E: $\Pr[l = l'] = p_E^{l'}(1 - p_E)$ (so that the mean length $\langle l \rangle = (1 - p_E)^{-1}$). The dependence of the gap frequency p_G on the time parameter τ is $p_G = 1 - \exp[-g\tau]$, where g plays the rôle of a gap-open rate. The gap extension parameter p_E does not depend on τ. The substitution matrix in the match state is also time-dependent, as described in Section 2.4.1 of this chapter.

Although the HMM in Figure 2.1 appears asymmetric (there is a transition from *Insert* to *Delete*, but not from *Delete* to *Insert*) the gap length distributions for the two strands are identical and independent (in fact, it is the asymmetry

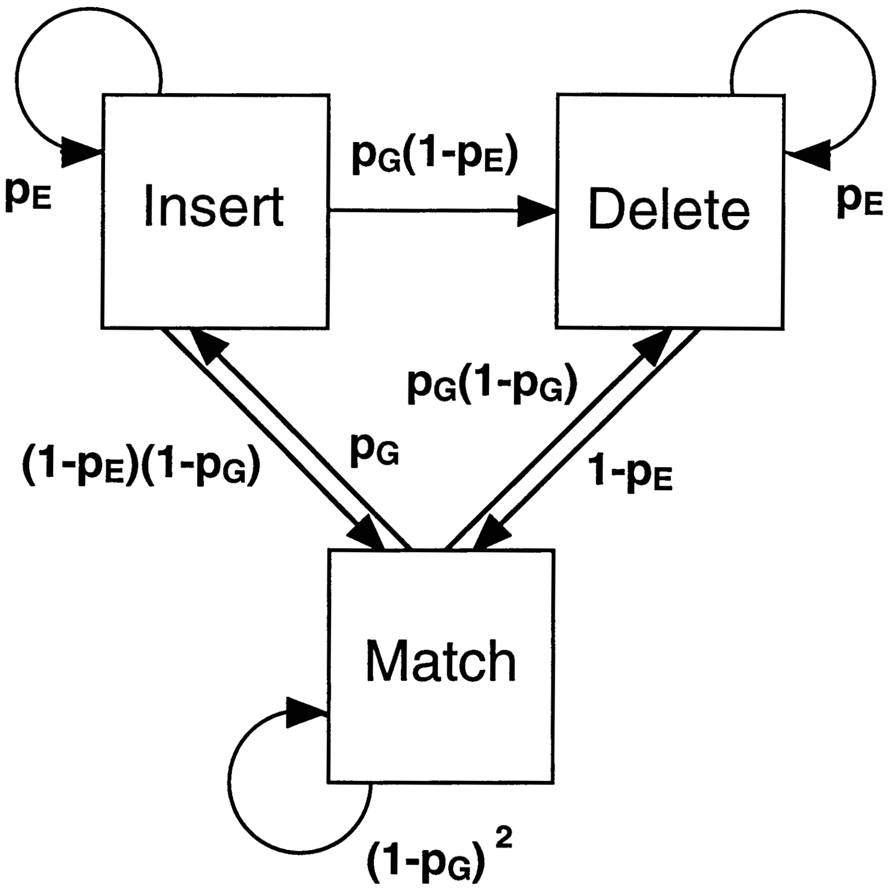

Figure 2.1: Hidden Markov model for Needleman-Wunsch global pairwise alignment with affine gaps. The start and end states are not shown. The gap penalty is determined by the gap frequency p_G (per residue per strand) and the gap extension probability p_E. The mean length of a gap is $(1 - p_E)^{-1}$.

of the model that ensures independence). For a fuller explanation of how the evolutionary model leads to the transition probabilities shown in Figure 2.1, see Chapter 3.

2.5 Likelihood derivatives and Fisher scores

There is a considerable amount of information in the posterior distribution that gets thrown away when an alignment is chosen, even if an optimal accuracy algorithm is used. This section looks at some of the ways that this information can be usefully digested.

Potentially the most useful result is that the derivatives of the sequence likelihood $\Pr[\mathbf{X}|\mathbf{t}]$ with respect to the parameters $\mathbf{t} = \{t_{abX}\}$ can be computed from the information in the Forward-Backward matrix. To see this, first write the path likelihood (2.1) as:

$$\Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}] = \prod_{a,b,X} t_{abX}^{n_{abX}} \tag{2.10}$$

where n_{abX} is the number of times that the alignment uses the transition t_{abX}. A more formal definition of n_{abX} is:

$$n_{abX}(\mathbf{a}) = \sum_{\substack{i\,:\,a_{i-1} = a \\ \text{and} \quad a_i = b \\ \text{and} \quad X_i = X}} 1 \tag{2.11}$$

which just says "to find n_{abX}, count the number of times that the i'th step of the path is from state a to state b and the i'th residue of the sequence is X".

The derivatives of the sequence likelihood $\Pr[\mathbf{X}|\mathbf{t}]$ are then given by:

$$\begin{aligned}
\frac{\partial}{\partial t_{abX}} \Pr[\mathbf{X}|\mathbf{t}] &= \frac{\partial}{\partial t_{abX}} \sum_{\mathbf{a}} \Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}] \\
&= \sum_{\mathbf{a}} n_{abX} \frac{\Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}]}{t_{abX}}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{\mathbf{a}} \sum_{\substack{i\,:\,a_{i-1}=a \\ \text{and}\ \ a_i=b \\ \text{and}\ \ X_i=X}} \frac{\Pr[\mathbf{a},\mathbf{X}|\mathbf{t}]}{t_{abX}} \\
&= \sum_{i} \sum_{\substack{\mathbf{a}\,:\,a_{i-1}=a \\ \text{and}\ \ a_i=b \\ \text{and}\ \ X_i=X}} \frac{\Pr[\mathbf{a},\mathbf{X}|\mathbf{t}]}{t_{abX}} \\
&= \sum_{i} \sum_{\mathbf{a}_{i-1,a}} \sum_{\bar{\mathbf{a}}_{i,b}} \Pr[\mathbf{a}_{i-1,a},\mathbf{X}|\mathbf{t}] \Pr[\bar{\mathbf{a}}_{i,b},\mathbf{X}|\mathbf{t}] \\
&= \sum_{i} \left(\sum_{\mathbf{a}_{i-1,a}} \Pr[\mathbf{a}_{i-1,a},\mathbf{X}|\mathbf{t}] \right) \left(\sum_{\bar{\mathbf{a}}_{i,b}} \Pr[\bar{\mathbf{a}}_{i,b},\mathbf{X}|\mathbf{t}] \right) \\
&= \sum_{i} F_{i-1,a} B_{i,b} \qquad (2.12)
\end{aligned}
$$

i.e. the derivatives can be calculated directly from the Forward-Backward matrix.

The expectations $E[n_{abX}|\mathbf{X},\mathbf{t}]$ of the counts n_{abX} over the posterior path distribution may be related to the derivatives of the sequence likelihood. First, note that differentiating (2.10) with respect to t_{abX} gives:

$$
\frac{\partial}{\partial t_{abX}} \Pr[\mathbf{a},\mathbf{X}|\mathbf{t}] = \frac{n_{abX}}{t_{abX}} \Pr[\mathbf{a},\mathbf{X}|\mathbf{t}] \qquad (2.13)
$$

Therefore the posterior expectations of the n_{abX} are given by:

$$
\begin{aligned}
E[n_{abX}|\mathbf{X},\mathbf{t}] &= \sum_{\mathbf{a}} n_{abX}(\mathbf{a}) \Pr[\mathbf{a}|\mathbf{X},\mathbf{t}] \\
&= \sum_{\mathbf{a}} \frac{n_{abX}(\mathbf{a}) \Pr[\mathbf{a},\mathbf{X}|\mathbf{t}]}{\Pr[\mathbf{X}|\mathbf{t}]} \\
&= \sum_{\mathbf{a}} \frac{t_{abX} \frac{\partial}{\partial t_{abX}} \Pr[\mathbf{a},\mathbf{X}|\mathbf{t}]}{\Pr[\mathbf{X}|\mathbf{t}]} \\
&= \frac{t_{abX} \frac{\partial}{\partial t_{abX}} (\sum_{\mathbf{a}} \Pr[\mathbf{a},\mathbf{X}|\mathbf{t}])}{\Pr[\mathbf{X}|\mathbf{t}]}
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{t_{abX} \frac{\partial}{\partial t_{abX}} \Pr[\mathbf{X}|\mathbf{t}]}{\Pr[\mathbf{X}|\mathbf{t}]} \\
&= t_{abX} \frac{\partial}{\partial t_{abX}} \log \Pr[\mathbf{X}|\mathbf{t}] \qquad (2.14)
\end{aligned}
$$

The final term on the right - the derivative of the sequence log-likelihood - is called the "Fisher score" [JH98].

An interesting use of HMMs is as a pre-processing step to kernel-based methods such as Support Vector Machines [JH98, Bur98, Mac97]. Very crudely, this method feeds the Fisher scores into a perceptron which then attempts to discriminate between sequences from the family that the HMM was trained to model and other sequences. It seems that this is a more discriminative measure than simply looking at the likelihood - which agrees with intuition, in that there should be more information in the derivatives of the likelihood than in the raw Forward score. Another view of this "Fisher kernel" is that the derivatives give the perceptron an idea of the n_{abX} and hence of the most likely alignment. A theoretical justification of Fisher kernels is offered in [JH98].

2.6 Fitting parameters to HMMs

Two standard tasks in Bayesian analysis using generative models are *training* and (less commonly in biological sequence analysis - though the principle is good) *model comparison.* Both involve exploration of the parameter space $\{\mathbf{t}\}$ of the model. Analogous to the Viterbi and Forward algorithms, training involves finding the most likely parameterisation of the model, whereas model comparison involves integrating (or summing) over all possible values of the parameters.

The motivation for training is easy: one wants to find the best set of parameters for modelling sequences and thus recognising homologies. The motivation for integrating across the parameter space is perhaps less obvious; the basic idea is to put a fair penalty on models with more parameters (the principle of "Occam's Razor" [Mac92b]). Optimising over the parameter space would not necessarily be fair, even if a prior distribution over parameters were used, because models with more parameters also have more uncertainty in the maximum-likelihood value of those parameters. A fair way to deal with parameter spaces of differing dimensionality is to integrate over the lot. An example of a comparison between hypotheses with different numbers of parameters may be found in Chapter 6, where time-dependent models are fitted to divergent intron sequences.

A prior distribution over the parameter space $\Pr[\mathbf{t}]$ is required if different parameter values are to be compared. It will be assumed in this section that this prior is adequately modelled by a Dirichlet distribution with pseudocounts $\alpha = \{\alpha_{abX}\}$ corresponding to the transition probabilities $\mathbf{t} = \{t_{abX}\}$ (see e.g. [DEKM98]). This distribution will be written $\mathcal{D}(\mathbf{t}|\alpha)$.

For HMMs, both training and model comparison are trivial if the alignment $\mathbf{a}$ is known. In this case, the likelihood $\Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}]$ is just the multinomial distribution (2.10) whose coefficients n_{abX} may be obtained from the alignment as in equation (2.11). The posterior distribution is a Dirichlet with parameters $n_{abX} + \alpha_{abX}$. The t_{abX} that maximise this probability are given by:

$$t_{abX} = \frac{n_{abX} + \alpha_{abX}}{\sum_{b',X'} n_{ab'X'} + \alpha_{ab'X'}} \tag{2.15}$$

The integral of the likelihood over parameter space is the normalising factor for the Dirichlet posterior:

$$\Pr[\mathbf{X}] = \int \Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}] \Pr[\mathbf{t}] dt = \frac{\prod_{a,b,X} \Gamma(\alpha_{abX} + n_{abX})}{\prod_a \Gamma(\sum_{b,X} \alpha_{abX} + n_{abX})} \tag{2.16}$$

where $\Gamma(x)$ is the gamma function. The extension of equations (2.15) and (2.16) to the case where the alignment $\mathbf{a}$ is unknown will be tackled below; essentially, the Dirichlet posterior becomes a mixture of $\sim N^L$ Dirichlets and the maximisation/integration is most easily handled approximately.

2.6.1 Maximising the likelihood in parameter space: training

When the alignment is unknown, the counts n_{abX} must be treated as missing data. A powerful algorithm for maximising the likelihood of a model with missing data is the expectation-maximization (EM) algorithm; the application of this algorithm to HMMs is called the Baum-Welch algorithm.

The basic idea of Baum-Welch is to calculate expected values $E[n_{abX}|\mathbf{X}, \mathbf{t}]$ for the counts n_{abX} given a particular value of the parameters $\mathbf{t}$ using equation (2.14) (the *expectation* step), then to maximize the sequence likelihood with respect to the parameters by plugging these expected counts back into equation (2.15) to give new values for the parameters (the *maximization* step).

It can be shown that both steps of this procedure increase the sum of the sequence log-likelihood and the Kullback-Leibler divergence between successive posterior distributions for the n_{abX} [NH93]. This sum is analogous to a variational free energy in statistical mechanics.

A problem with EM is that it can get stuck in local maxima of the likelihood. Modifications of Baum-Welch that attempt to address this problem include noise injection during the estimation of the n_{abX} [KBM$^+$94, HK96] (or the related technique of sampling the t_{abX} from the tempered Dirichlet posterior rather than taking the mean), Gibbs sampling [LAB$^+$93] and simulated Viterbi annealing [Edd95].

If the t_{abX} are not independent variables but can be expressed parametrically in terms of independent variables (such as a time parameter), then equations (2.14) and (2.15) will no longer apply. In this case it should still be possible to find the derivatives of the sequence likelihood with respect to the independent variables, by applying the chain rule to (2.12). Standard gradient-ascent algorithms can then be applied; for examples of such algorithms see [Bis95].

2.6.2 Integrating the likelihood over parameter space: model comparison

If the t_{abX} are independent variables, one way to estimate the integral $\Pr[\mathbf{X}] = \int \Pr[\mathbf{X}|\mathbf{t}]\mathcal{D}(\mathbf{t}|\alpha)dt$ is by importance sampling [Nea98]. Roughly speaking, this works as follows: sample points in parameter space $\mathbf{t}$ are generated from the Dirichlet prior distribution $\mathcal{D}(\mathbf{t}|\alpha)$, as described in [DEKM98]; the likelihoods $\Pr[\mathbf{X}|\mathbf{t}]$ are calculated for each point; these likelihoods are then averaged to estimate $\Pr[\mathbf{X}]$:

$$\Pr[\mathbf{X}] = \lim_{N\to\infty} \frac{1}{N} \sum_{i}^{N} \Pr[\mathbf{X}|\mathbf{t}_i] \quad \text{where } \Pr[\mathbf{t}_i = \mathbf{t}] \equiv \mathcal{D}(\mathbf{t}|\alpha) \tag{2.17}$$

The iteration can be stopped when, for example, the change in the estimate for $\Pr[\mathbf{X}]$ becomes sufficiently small.

Various modifications to this procedure might improve its efficiency. For example, it is possible to adapt the prior on-the-fly as more data points are seen; this is known as annealed importance sampling [Nea98]. An alternative approximate procedure is Markov chain Monte Carlo sampling [Nea96].

Another improvement to the importance sampling method would be to replace the $\Pr[\mathbf{X}|\mathbf{t}_i]$ term on the right-hand side of (2.17) with a more sophisticated estimate for $\Pr[\mathbf{X}]$. Speculatively speaking, it might be possible to derive this estimate by calculating expectation values for the counts n_{abX} using equation (2.14) and plugging these expectations into equation (2.16). By analogy with the Baum-Welch algorithm, this might be hoped to speed up convergence as it seems to take more account of the nature of the likelihood distribution.

If the t_{abX} are not independent, but instead are parametric functions of a set of independent variables, then importance sampling can proceed by sampling from a prior over this independent variable set. If there are just a few independent variables (such as a time parameter) it may be more convenient just to take a fixed set of sample points from a regularly spaced grid, though in general this can be expected to perform slightly worse than random sampling [Nea]. Notwithstanding, this is the method used in Chapter 6.

2.6.3 Incremental Baum-Welch and sparse envelopes

The view of the EM algorithm outlined in Section 2.6.1 of this chapter and presented in full in [NH93] suggests that, since the E and M steps may both be viewed as incremental maximizations of the same "variational free energy" function, even more incremental variants of the algorithm (where the variational free energy is improved, but not quite optimised, with respect to the probability distribution over the n_{abX} at each M-step) may speed up computation and hence convergence.

There are at least two ways this could be applied to the Baum-Welch algorithm for HMM training. One way would be if a set of K sequences $\{\mathbf{X}_k\}_{k=1}^{K}$ were being used to train the HMM, and the counts n_{abX} were obtained by summing the individual sequence counts, i.e. $n_{abX} = \sum_{k=1}^{K} n_{abX}(k)$; in this case, the individual sequence counts $n_{abX}(k)$ could be updated one at a time, and the t_{abX} re-estimated after the Forward-Backward algorithm was performed on each sequence, so that the dynamic programming M-step was only performed on each sequence every K'th E-step of the iteration. This corresponds to the incremental algorithm described in [NH93].

The second proposed optimisation to Baum-Welch can work on just one sequence. It can be applied not just to Baum-Welch but also to approximate numerical integration over the parameter space; it also works when the t_{abX} are not independent but are parametrically dependent on a reduced independent variable set. The optimisation corresponds loosely to the sparse algorithm described in [NH93]; here, it is called the "sparse envelopes" method and may be explained as follows.

The basic idea is that after the first run of the Forward-Backward algorithm, it should be obvious which alignments are the most probable, since these alignments will lie in the regions of the dynamic programming matrix where most of the probability distribution $\Pr[(i \diamond a)|\mathbf{X}, \mathbf{t}]$ is concentrated. Accordingly, cells that have extremely low probability can be "frozen" at their low-probability levels and not updated in subsequent runs. In practise it is often easier to freeze the likelihoods rather than the probabilities of these cells; alternatively, they can just be set to zero.

The set of cells that is chosen for inclusion in future updates is called the *envelope*. The choice of envelope can be managed as follows. For each residue i in the sequence, the posterior probabilities $\Pr[(i \diamond a)|\mathbf{X}, \mathbf{t}]$ (corresponding to a column in the dynamic programming matrix) must sum to unity. Choose some threshold $\epsilon \ll 1$ and find the lowest value of p such that the all the posterior probabilities $\Pr[(i \diamond a)|\mathbf{X}, \mathbf{t}]$ that are greater than p add up to more than $1 - \epsilon$, i.e.:

$$p(i) = \min\{p' : \left(\sum_{j:\Pr[(i\diamond a)|\mathbf{X},\mathbf{t}] \geq p'} \Pr[(i \diamond a)|\mathbf{X}, \mathbf{t}] \right) \geq 1 - \epsilon\}$$

The cells to be masked out for this value of i are those cells whose posterior probability is lower than $p(i)$. For pair HMMs, the dynamic programming matrix becomes a cube and the co-ordinates of a cell are $(i_1 \diamond i_2 \diamond a)$ rather than $(i \diamond a)$: each value of i_1 corresponds to a "slice" of the dynamic programming matrix since there are two parameters to sum over (i_2 and a) rather than just one (i). It may be convenient not to mask out quite all of the cells whose probability is lower than p', either to ensure that there is always a valid path through the matrix, or to avoid storing complicated masks (a convenient alternative for pair HMMs is just to keep track of an interval $i_2^{\min} - i_2^{\max}$ for each value of i_1, and mask out cells that fall outside this interval).

The approximation of the sparse envelopes method is rather similar to conditioning the integral over parameter space on the Viterbi alignment as in (2.16) in that, if the Viterbi alignment has probability greater than $1 - \epsilon$, the envelope will contain just the Viterbi path. However, sparse envelopes seem slightly more principled, as they allow a variable tradeoff between summation over suboptimal alignments and high computation time.

The incremental and the sparse variants of Baum-Welch are not guaranteed to converge on the same local minimum that the standard EM algorithm is guaranteed to find; indeed, the incremental algorithm can fail to converge at all, instead oscillating between results. However, there are plausible arguments that these algorithms may find the neighbourhood of the minimum more quickly. A sensible strategy might be to run several iterations of the approximate algorithm, then return to the standard EM algorithm for the last few iterations. This should combine the speed advantages of the approximate algorithms with the accuracy advantages of the exact algorithm.

2.7 Score and length distributions of an HMM

It may be useful to know the probability distribution of the scores of the paths that an HMM emits. For example, if the HMM is being used to search for instances of a sequence family, the score distribution can be used to estimate the probability that a true family member will score below the cutoff. An elementary result from the theory of Markov chains allows calculation of any number of moments of the score distribution.

Note that the following derivation assumes there is at most one transition between any pair of states. The HMMs considered so far have allowed multiple transitions between a pair of states, so long as they each emit a different residue. However, any HMM with multiple transitions can be converted to an HMM with single transitions by simply augmenting the state space, so no generality is lost in this assumption.

Suppose that the score of the transition from state a to state b is σ_{ab} (if the score is a straightforward log-likelihood, then $\sigma_{ab} = \log t_{ab}$ and the expected score will be the entropy of the model). Let $f_a(s)$ be the probability density function of the score s starting from state a. Then:

$$f_a(s) = \sum_b t_{ab} f_b(s - \sigma_{ab}) \tag{2.18}$$

Let $\phi_a(k)$ be the Fourier transform of s, i.e. $\phi_a(k) = E_a[\exp[\imath ks]]$ where $\imath = \sqrt{-1}$. The equivalent of equation (2.18) for $\phi_a(k)$ is:

$$\phi_a(k) = \sum_b t_{ab} \exp[\imath k \sigma_{ab}] \phi_b(k) \tag{2.19}$$

which is a matrix equation (although the entries in the matrix are functions of k).

Since $\phi_a(k)$ is the characteristic function of $f_a(s)$, the n'th moment of s can be evaluated by taking the value of the n'th derivative of $\phi_a(k)$ at $k = 0$. Differentiating equation (2.19) n times and setting $k = 0$ gives:

$$\begin{aligned} E_a[s^n] &= \frac{d^n \phi_a}{d(\imath k)^n}(0) \\ &= \sum_b t_{ab}(\sigma_{ab}^n + \frac{d^n \phi_b}{d(\imath k)^n}(0)) \\ &= (\sum_b t_{ab} E_j[s^n]) + (\sum_b t_{ab} \sigma_{ab}^n) \end{aligned} \tag{2.20}$$

This is a matrix equation which may be solved for any n by inverting the transition matrix t_{ab} and setting $E_{\mathcal{E}}[s^n] = 0$. It is thus possible to calculate any number of moments of the score distribution from any state of the model. Calculating the first two moments is sufficient to approximate the distribution with a Gaussian [KT75].

The result can be generalised to the case where the σ_{ab} are themselves random variables by replacing the n'th power of σ on the right-hand side of (2.20) with the n'th moment $\langle \sigma_{ab}^n \rangle$.

By setting all the σ_{ab} to 1 and identifying the score distribution from a state with the waiting time from that state to the end state, equation (2.20) can be used to derive constraints for modelling (sensibly shaped) sequence length distributions to arbitrary precision.

It was mentioned above that if $\sigma_{ab} = \log t_{ab}$, then the expected score of a path is the entropy of the HMM. More correctly, this score is the entropy of the joint distribution over paths and sequences $\Pr[\mathbf{a}, \mathbf{X}]$. The variance of the score is analogous to fluctuations in the statistical mechanical entropy. To find the entropy of the marginal distribution over sequences $\Pr[\mathbf{X}]$ is more difficult, but more relevant to the question of whether a sequence will score high enough to be observed, since the sequence likelihood $\Pr[\mathbf{X}]$ is the score returned by the Forward algorithm (and will be close to the Viterbi score for many cases of interest; see the comment at the end of Section 2.3.2 of this chapter). A similarly difficult problem is to find the relative entropy $D(\tilde{P}_1 || \tilde{P}_2)$ between sequence probability distributions $\tilde{P}_n \equiv \Pr[\mathbf{X}|M_n]$ generated by two alternative models M_1 and M_2 (for example, a null model and a model of a protein domain). Implicitly, when programs such as HMMER [Edd96] fit extreme-value or other distributions to the observed scores from a database search, they are heuristically estimating the fluctuative behaviour of the relative entropy. A very simple approximation towards calculating the relative entropy fluctuations is described in Chapter 3. A more sophisticated approach has been developed by Hwa and Lässig [HL96], who apply renormalisation group techniques from the theory of critical phenomena in statistical physics to find the scaling behaviour of various properties of the Viterbi path.

2.8 Generalised HMMs

A useful generalisation of the HMMs described here is to relax the idea that each transition t_{abX} emits a single residue X and allow each transition to have a probability distribution over the length and content of the sequence it emits. This is called a "generalised HMM" after [KHRE96].

The simplest example of this kind of system keeps the basic structure of the HMMs described above, but instead of the length of the sequence emitted by each state being geometrically distributed like a Markov waiting time [KT75], a flat distribution for the length of sequence emitted by each state is used. The total emitted sequence length is conditioned on as a separate constraint. A prior can be put on the total sequence length, although it is usually an observed quantity and so the prior is only relevant during model comparison.

This kind of generalised HMM can be obtained from the HMMs described in Section 2.2 of this chapter as follows. Consider the simple two-state (*Loop*,*End*) model shown at the top of Figure 2.2. The model starts in the *Loop* state and at each step it either returns to the *Loop* state with probability $1 - \varepsilon$ or it moves on to the *End* state (and stops there) with probability ε. On every transition a residue is emitted. The probability that the model emits L residues is $\Pr[L] = (1-\varepsilon)^{L-1}\varepsilon$. Consider what happens as ε becomes small. The model will tend to stay in the *Loop* state for longer and longer times and the probability distribution of the emitted sequence length will get flatter and flatter. Formally, as $\varepsilon \to 0$, terms of $O(\varepsilon^2)$ become negligible and $\Pr[L] \to \varepsilon$.

So it is possible to design a very simple HMM where the probability of getting a sequence of a particular length L is almost independent of L, but the price one pays is that the likelihood of any individual alignment is very small - virtually zero, in fact. This is because L could be huge, and the probabilities of all possible values of L have to add up to one. However, given a sequence of a particular length L, Bayes' rule specifies how to work out the likelihood of an alignment $\mathbf{a}$ given the length L by *conditioning* on L: one simply has to divide the joint likelihood by the marginal length likelihood, i.e. $\Pr[\mathbf{a}|L] = \Pr[\mathbf{a}, L]/\Pr[L]$. Since $\Pr[L] \simeq \varepsilon$, this corresponds to cancelling out the final ε from the alignment likelihood. This can be woven seamlessly into the dynamic programming algorithm by pretending that the *Loop*$\to$*Loop* and *Loop*$\to$*End* transitions both have a "probability" of 1.

Now consider the three-state (*Loop*$_1$, *Loop*$_2$, *End*) model shown at the bottom of Figure 2.2. The same trick can be done to flatten the length distribution from each state by letting $\varepsilon \to 0$. However, the probability of getting a particular

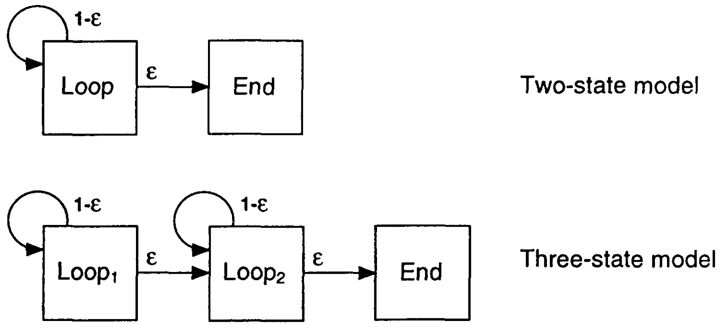

Figure 2.2: Looping models that tend towards flat length distributions as $\varepsilon \to 0$.

sequence length L is now $\Pr[L] = (L-1)(1-\varepsilon)^{L-1}\varepsilon^2$ and as $\varepsilon \to 0$ then $\Pr[L] \to (L-1)\varepsilon^2$. The extra factor of ε arises because there each path now has to make two low-probability transitions to reach the *End* state, rather than one. The extra factor of $L-1$ arises because there are $L-1$ different ways that a path can get to the *End* state in L steps, depending on when it chooses to move from $Loop_1$ to $Loop_2$.

In general, for a k-state model ($Loop_1 \ldots Loop_{k-1}$, *End*) there will be $k-2$ such choices and $\begin{pmatrix} L-1 \\ k-2 \end{pmatrix} = \frac{(L-1)!}{(k-2)!(L-k+2)!}$ ways of getting to the *End* state in L steps. The correct length-conditioned alignment likelihood $\Pr[\mathbf{a}|L]$ can be computed by doing dynamic programming with all the ε-transitions artificially set to 1 and dividing the result by $\begin{pmatrix} L-1 \\ k-2 \end{pmatrix}$. Once the conditional distribution $\Pr[\mathbf{a}|L]$ has been obtained, it can be multiplied by a more realistic prior distribution for L if this is desired.

The model underlying the Smith-Waterman algorithm with affine gaps is an example of a generalised HMM, since there is no prior length distribution on the flanking states which distinguish it from the Needleman-Wunsch model [SW81]. However, it is not a linear architecture like the HMMs in Figure 2.2 since the reciprocal transitions between the match and indel states form an internal cycle. Also, the affine gap costs put implicit priors on both the length and the number of gaps. A natural extension is to roll out the model, expanding the state

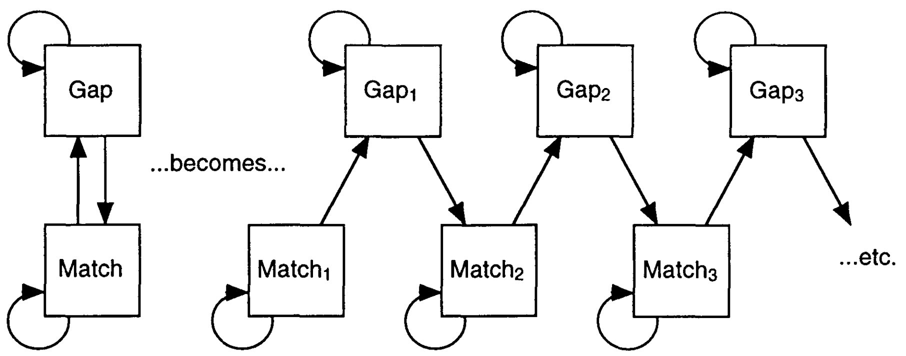

Figure 2.3: Unrolling the states of the Smith-Waterman model leads to the Bayes block aligner.

space up to a size proportional the maximum number of gaps as illustrated in Figure 2.3. Prior distributions can then be placed on the sequence lengths and the number of gaps. This is called the "Bayesian block aligner" [ZLL98].

The probability distribution of the sequences emitted by a state of a generalised HMM does not have to come from a length-conditioned HMM; more complex probabilistic models can be used, such as neural networks. A review of the use of generalised HMMs in gene-finding is provided in [Hau98].