Algorithms for Molecular Biology **Fall Semester, 1998**

Lecture 6: January 10, 1999

Lecturer: Ron Shamir *Scribe: Ron Wein and Nir Avrahami*

6.1 Hidden Markov Models

6.1.1 Preface: CpG islands

It is known that due to biochemical considerations that *CpG*, the pair of nocleotides C and G, appearing successively, in this order, along one DNA starnd, is relatively rare in DNA sequences, excluding particular sub-sequences, which are several hundreds of nucleotides long, where the couple CpG is more frequent. These sub-sequences, called *CpG islands*, are known to appear in the biologically more significant parts of the genome. The ability to identify CpG islands in the DNA will therefore help us spot the more significant regions of interest along the genome.

Problem 6.1 Identifying a CpG island.
INPUT: A short DNA sequence $X = (x_1, \ldots, x_L) \in \Sigma^*$ (where $\Sigma = \{$A,C,G,T$\}$).
QUESTION: Decide whether X is a CpG island.

We can approach such problems using a *Markov chain model.* Let us denote for each $s, t \in \Sigma$ the transition probability:

$$a_{st} \equiv P(x_i = t | x_{i-1} = s) \tag{6.1}$$

We assume that $\{x_i\}$ is a random process with a memory of length 1, i.e., the value of the random variable x_i depends only on its predecessor x_{i-1}. Formally we can write:

$$\begin{aligned} \forall s_1, \ldots, s_i \in \Sigma \quad P(x_i = s_i | x_1 = s_1, \ldots, x_{i-1} = s_{i-1}) &= \\ = P(x_1 = s_i | x_{i-1} = s_{i-1}) &= a_{s_{i-1}, s_i} \end{aligned} \tag{6.2}$$

The probability of the whole sequence X will therefore be:

$$P(X) = p(x_1) \cdot \prod_{i=2}^{L} a_{x_{i-1}, x_i} \tag{6.3}$$

+	A	C	G	T	-	A	C	G	T
A	0.180	0.274	0.426	0.120	A	0.300	0.205	0.285	0.210
C	0.171	0.368	0.274	0.188	C	0.322	0.298	0.078	0.302
G	0.161	0.339	0.375	0.125	G	0.248	0.246	0.298	0.208
T	0.079	0.355	0.384	0.182	T	0.177	0.239	0.292	0.292

Table 6.1: Transition probabilities inside/outside a CpG island

We can also add fictitious *begin* $(= x_0)$ and *end* $(= x_{L+1})$ symbols to simplify the formula, where $\forall_{s\in\Sigma}\, a_{0,s} \equiv p(s)$ is the background probability of the symbol s. Hence:

$$P(X) = \prod_{i=1}^{L} a_{x_{i-1},x_i} \tag{6.4}$$

Let a^+_{st} denote the transition probability of $s,t \in \Sigma$ inside a CpG island and let a^-_{st} denote the transition probability outside a CpG island (see table 6.1 for the values of these probabilities, taken from [4]). We can give a logarithmic likelihood score for the sequence X:

$$Score(X) = \log \frac{P(X|\text{CpG island})}{P(X|\text{non CpG island})} = \sum_{i=1}^{L} \log \frac{a^+_{x_{i-1},x_i}}{a^-_{x_{i-1},x_i}} \tag{6.5}$$

The higher this score, the more likely is that X is a CpG Island.

Problem 6.2 Locating CpG islands in a DNA sequence.
INPUT: A long DNA sequence $X = (x_1, \dots, x_L) \in \Sigma^*$.
QUESTION: Locate the CpG islands within X.

A naive approach for solving this problem will be to extract a sliding window $X^k = (x_{k+1}, \dots, x_{k+\ell})$ of a given length ℓ (where $\ell \ll L$, usually several hundreds long, and $1 \leq k \leq L - \ell$) to the sequence and calculate $Score(X^k)$ for each one of the resulting sub-sequences. Sub-sequences that receive positive scores are potential CpG islands.

The main disadvantage in this algorithm is that we have no information about the lengths of the islands, while the algorithm suggested above assumes that those islands are at least ℓ nucleotides long. Should we use a value of ℓ which is too large, the CpG islands would be short sub-strings of our windows, and the score we give those windows may not be high enough. A better approach to such problems is described in the following section.

6.1.2 Hidden Markov Models

Definition A general *Hidden Markov Model (HMM)* is a triplet $\mathcal{M} = (\Sigma, Q, \Theta)$, where:

- Σ is an alphabet of symbols.
- Q is a finite set of states capable of emitting symbols from the alphabet Σ.
- Θ is a set of probabilities, comprised of:
  - *State transition probabilities*, denoted by a_{kl} for each $k, l \in Q$.
  - *Emission probabilities*, denoted by $e_k(b)$ for each $k \in Q$ and $b \in \Sigma$.

A *path* $\Pi = (\pi_1, \ldots, \pi_L)$ in the model $\mathcal{M}$ is a sequence of states. We can now define the *state transition probabilities* and the *emission probabilities* in terms of Π given a sequence $X = (x_1, \ldots, x_L) \in \Sigma^*$:

$$\begin{aligned} a_{kl} &= P(\pi_i = l | \pi_{i-1} = k) \\ e_k(b) &= P(x_i = b | \pi_i = k) \end{aligned} \tag{6.6}$$

The probability that the sequence X was generated by the model $\mathcal{M}$ given the path Π is therefore:

$$P(X|\Pi) = a_{\pi_0,\pi_1} \cdot \prod_{i=1}^{L} e_{\pi_i}(x_i) \cdot a_{\pi_i,\pi_{i+1}} \tag{6.7}$$

Where for our convenience we denote $\pi_0 = begin$ and $\pi_{L+1} = end$.

Example An HMM for detecting CpG islands in a long DNA sequence.

The model contains eight states corresponding to the four symbols of the alphabet Σ ={A,C,G,T}:

State:	A^+	C^+	G^+	T^+	A^-	C^-	G^-	T^-
Emitted Symbol:	A	C	G	T	A	C	G	T

If the probability for staying in a CpG island is p and the probability of staying outside it is q, then the transition probabilities will be as described in table 6.2 (derived from the transition probabilities given in table 6.1 under the assumption that we lose memory when moving from/into a CpG island, and that we ignore background probabilities).

$a_{\pi_i,\pi_{i+1}}$	A$^+$	C$^+$	G$^+$	T$^+$	A$^-$	C$^-$	G$^-$	T$^-$
A$^+$	$0.180p$	$0.274p$	$0.426p$	$0.120p$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$
C$^+$	$0.171p$	$0.368p$	$0.274p$	$0.188p$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$
G$^+$	$0.161p$	$0.339p$	$0.375p$	$0.125p$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$
T$^+$	$0.079p$	$0.355p$	$0.384p$	$0.182p$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$	$\frac{1-p}{4}$
A$^-$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$0.300q$	$0.205q$	$0.285q$	$0.210q$
C$^-$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$0.322q$	$0.298q$	$0.078q$	$0.302q$
G$^-$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$0.248q$	$0.246q$	$0.298q$	$0.208q$
T$^-$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$\frac{1-q}{4}$	$0.177q$	$0.239q$	$0.292q$	$0.292q$

Table 6.2: Transition probabilities in the CpG islands HMM

In this special case the emission probability of each state X^+ or X^- is exactly 1 for the symbol X and 0 for any other symbol.

Let us consider another example, where the emission probabilities will not be degenerate.

Example Suppose a dealer in a casino tosses a coin. We know the dealer may use a fair coin or a biased coin which has a probability of 0.75 to get a "head". We also know that the dealer does not tend to change coins - this happens only with a probability of 0.1. Given a sequence of coin tosses we wish to determine when did the dealer use the biased coin and when did he use a fair coin.

The corresponding HMM is:

- The states are $Q = \{F, B\}$, where F stands for "fair" and B for "biased".
- The alphabet is $\Sigma = \{h, t\}$, where h stands for "head" and t for "tails".
- The probabilities are:

$$a_{FF} = a_{BB} = 0.9 \tag{6.8}$$
$$a_{FB} = a_{BF} = 0.1 \tag{6.9}$$

$$e_F(h) = 0.5 \qquad e_F(t) = 0.5 \tag{6.10}$$
$$e_B(h) = 0.75 \qquad e_B(t) = 0.25 \tag{6.11}$$

Figure 6.1 gives a full description of the model.

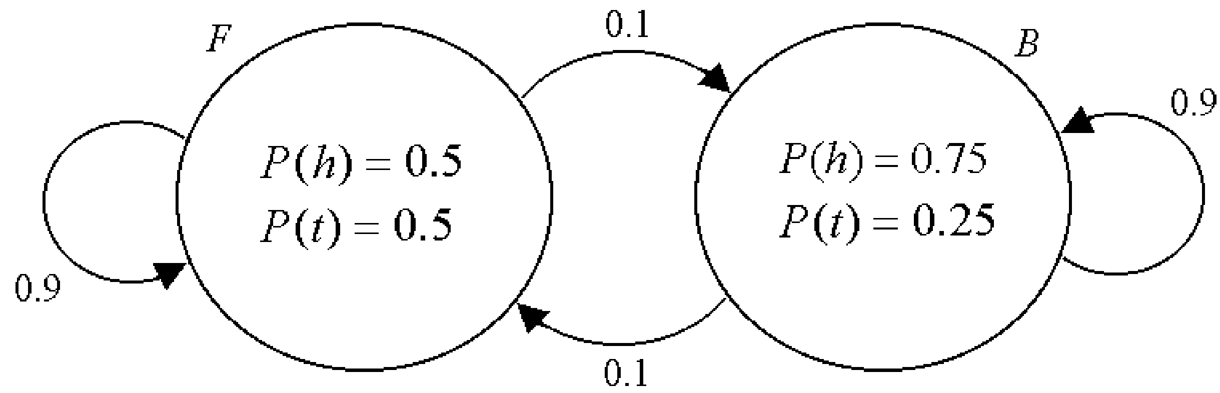

Figure 6.1: HMM for the coin tossing problem

Returning to the general case, we have defined the probability $P(X|\Pi)$ for a given sequence X and a given path Π. However, we do not know the actual sequence of states $(\pi_1, \ldots, \pi_L)$ that emitted $(x_1, \ldots, x_L)$. We therefore say that the generating path of X is *hidden*.

Problem 6.3 The decoding problem.
INPUT: A hidden Markov model $\mathcal{M} = (\Sigma, Q, \Theta)$ and a sequence $X \in \Sigma^*$.
QUESTION: Find an optimal generating path Π^* for X, such that $P(X|\Pi^*)$ is maximized. We denote this also by:

$$\Pi^* = \arg\max_{\Pi}\{P(X|\Pi)\}$$

In the CpG islands case (problem 6.2), the optimal path can help us find the location of the islands. Had we known Π^*, we could have traversed it determining that all the parts that pass through the "+" states are CpG islands.

Similarly, in the coin-tossing case (example), the parts of Π^* that pass through the B (biased) state are suspected tosses of the biased coin.

A solution for the optimal path problem is described in the next section.

6.1.3 Viterbi Algorithm

We can calculate the optimal path in a hidden Markov model using a dynamic programming algorithm. This algorithm is widely known as *Viterbi Algorithm*. Viterbi [10] devised this algorithm for the decoding problem, even though its more general description was originally given by Bellman [3].

Given a sequence X, denote $v_k(i)$ to be the probability of the most probable path for the prefix $(x_1, \ldots, x_i)$ that ends with the state k ($k \in Q$ and $1 \leq i \leq L$).

1. Initialize:

$$v_{begin}(0) = 1 \tag{6.12}$$
$$\forall_{k \neq begin} \quad v_k(0) = 0 \tag{6.13}$$

2. For each $i = 0, \ldots, L-1$ and for each $l \in Q$ recursively calculate:

$$v_l(i+1) = e_l(x_{i+1}) \cdot \max_{k \in Q} \{v_k(i) \cdot a_{kl}\} \tag{6.14}$$

3. Finally, the value of $P(X|\Pi^*)$ is:

$$P(X|\Pi^*) = \max_{k \in Q} \{v_k(L) \cdot a_{k,end}\} \tag{6.15}$$

   We can reconstruct the path Π^* itself by keeping back pointers during the recursive stage and tracing them.

Complexity: We calculate the values of $O(|Q| \cdot L)$ cells of the matrix V, spending $O(|Q|)$ operations per cell. The overall time complexity is therefore $O(L \cdot |Q|^2)$ and the space complexity is $O(L \cdot |Q|)$.

Since we are dealing with probabilities, the extensive multiplication operations we perform may result in an underflow. This can be avoided if we choose to work with logarithmic scores. We can therefore define $v_k(i)$ to be the logarithmic score of the most probable path for the prefix $(x_1, \ldots, x_i)$ that ends in the state k.

We shall initialize:

$$v_{begin}(0) = 0 \tag{6.16}$$
$$\forall_{k \neq begin} \quad v_k(0) = -\infty \tag{6.17}$$

The recursion will look like:

$$v_l(i+1) = \log e_l(x_{i+1}) + \max_{k \in Q} \{v_k(i) + \log(a_{kl})\} \tag{6.18}$$

Finally, the score for the best path Π^* is:

$$Score(X|\Pi^*) = \max_{k \in Q} \{v_k(L) + \log(a_{k,end})\} \tag{6.19}$$

6.1.4 Forward and Backward Probabilities

Problem 6.4 The likelihood problem.
INPUT: A hidden Markov model $\mathcal{M} = (\Sigma, Q, \Theta)$ and a sequence $X \in \Sigma^*$, for which the generating path $\Pi = (\pi_1, \dots, \pi_L)$ is unknown.
QUESTION: For each $1 \le i \le L$ and $k \in Q$, compute the probability $P(\pi_i = k|X)$.

For this we shall need some extra definitions.

Forward algorithm: Given a sequence $X = (x_1, \dots, x_L)$ let us denote by $f_k(i)$ the probability of emitting the prefix $(x_1, \dots, x_i)$ and eventually reaching $\pi_i = k$.

We use the same initial values for $f_k(0)$ as was done in the Viterbi algorithm:

$$f_{begin}(0) = 1 \tag{6.20}$$

$$\forall_{k \neq begin} \quad f_k(0) = 0 \tag{6.21}$$

In analogy to 6.14 we can use the recursive formula:

$$f_l(i+1) = e_l(x_{i+1}) \cdot \sum_{k \in Q} f_k(i) \cdot a_{kl} \tag{6.22}$$

We terminate the process by calculating:

$$P(X) = \sum_{k \in Q} f_k(L) \cdot a_{k,end} \tag{6.23}$$

Backward algorithm: In a complementary manner we denote by $b_k(i)$ the probability of the suffix $(x_{i+1}, \dots, x_L)$ given $\pi_i = k$.

In this case, we initialize:

$$\forall_{k \in Q} \quad b_k(L) = a_{k,end} \tag{6.24}$$

The recursive formula is:

$$b_k(i) = \sum_{l \in Q} a_{kl} \cdot e_l(x_{i+1}) \cdot b_l(i+1) \tag{6.25}$$

We terminate the process by calculating:

$$P(X) = \sum_{l \in Q} a_{begin,l} \cdot e_l(x_1) \cdot b_l(1) \tag{6.26}$$

Complexity: All the values of $f_k(i)$ and $b_k(i)$ can be calculated in $O(L \cdot |Q|^2)$ time and stored in $O(L \cdot |Q|)$ space, as it is the case with Viterbi algorithm.

There is however one important difference: here we cannot trivially use the logarithmic weights, since (unlike in Viterbi) we do not perform only multiplication of probabilities, but we also sum probabilities. This may lead to numeric stabilization problems, unless proper measures, such as scaling the probabilities, are taken.

Using the forward and backward probabilities we can compute the value of $P(\pi_i = k|X)$. Since the process only has memory of length 1, there is a dependency only on the last state, so we can write:

$$\begin{aligned} P(X, \pi_i = k) &= P(x_1, \ldots, x_i, \pi_i = k) \cdot P(x_{i+1}, \ldots, x_L | x_1, \ldots, x_i, \pi_i = k) = \\ &= P(x_1, \ldots, x_i, \pi_i = k) \cdot P(x_{i+1}, \ldots, x_L | \pi_i = k) = \\ &= f_k(i) \cdot b_k(i) \end{aligned} \tag{6.27}$$

Using the definition of conditional probability, we obtain the solution to the likelihood problem:

$$P(\pi_i = k|X) = \frac{P(X, \pi_i = k)}{P(X)} = \frac{f_k(i) \cdot b_k(i)}{P(X)} \tag{6.28}$$

6.1.5 Parameter Estimation for HMMs

In examples and we constructed hidden Markov models knowing the transition and emission probabilities for the problems we had to solve. In real life, this may not be the case. We may be given n strings $X^{(1)}, \ldots, X^{(n)} \in \Sigma^*$ of length $L^{(1)}, \ldots, L^{(n)}$, respectively, which were all generated from the HMM $\mathcal{M} = (\Sigma, Q, \Theta)$. The values of the probabilities in Θ, however, are unknown a-priori.

In order to construct the HMM that will best characterize $X^{(1)}, \ldots, X^{(n)}$, we will have to assign values to Θ that will maximize the probabilities of our strings according to the model. Since all strings are assumed to be generated independently, we can write:

$$P(X^{(1)}, \ldots, X^{(n)}|\Theta) = \prod_{j=1}^{n} P(X^{(i)}|\Theta) \tag{6.29}$$

Using the logarithmic score, our goal is to find Θ^* such that

$$\Theta^* = \arg\max_{\Theta}\{Score(X^{(1)}, \ldots, X^{(n)}|\Theta)\} \tag{6.30}$$

where:

$$Score(X^{(1)}, \ldots, X^{(n)}|\Theta) = \log P(X^{(1)}, \ldots, X^{(n)}|\Theta) = \sum_{j=1}^{n} \log(P(X^{(i)}|\Theta)) \tag{6.31}$$

The strings $X^{(1)}, \ldots, X^{(n)}$ are usually called the *training sequences.*

Case 1: Assuming we know the state sequences $\Pi^{(1)}, \ldots, \Pi^{(n)}$ corresponding to $X^{(1)}, \ldots, X^{(n)}$, respectively. We can scan these sequences and compute:

- A_{kl} - the number of transitions from the state k to l.
- $E_k(b)$ - the number of times that an emission of the symbol b occurred in state k.

The maximum likelihood estimators will be:

$$a_{kl} = \frac{A_{kl}}{\sum_{q \in Q} A_{kq}} \tag{6.32}$$

$$e_k(b) = \frac{E_k(b)}{\sum_{\sigma \in \Sigma} E_k(\sigma)} \tag{6.33}$$

To avoid zero probabilities, when working with a small amount of samples, it is recommended to work with A'_{kl} and $E'_k(b)$, where:

$$A'_{kl} = A_{kl} + r_{kl} \tag{6.34}$$

$$E'_k(b) = E_k(b) + r_k(b) \tag{6.35}$$

Usually the *Laplace correction*, where all r_{kl} and $r_k(b)$ values equal 1, is applied, having an intuitive interpretation of a-priori assumed uniform distribution. However, it may be beneficial in some cases to use other values for the correction (e.g. when having some prior information about the transition or emission probabilities).

Case 2: Usually, the state sequences $\Pi^{(1)}, \ldots, \Pi^{(n)}$ are not known. In this case, the problem of finding the optimal set of parameters Θ^* is known to be NP-complete. The *Baum-Welch algorithm* [2], which is a special case of the *EM technique (Expectation and Maximization)*, can be used for heuristically finding a solution to the problem.

1. *Initialization*: Assign values to Θ.

2. *Expectation*:

   (a) Compute the expected number of state transitions from state k to state l. Using the same arguments we used for computing $P(X, \pi_i = k)$ (see 6.27), we get:

   $$P(\pi_i = k, \pi_{i+1} = l | X, \Theta) = \frac{f_k(i) \cdot a_{kl} \cdot e_l(x_{i+1}) \cdot b_l(i+1)}{P(X)} \tag{6.36}$$

   Hence, we can compute the expectation:

   $$A_{kl} = \sum_{j=1}^{n} \frac{1}{P(X^{(j)})} \cdot \sum_{i=1}^{L^{(j)}} f_k^{(j)}(i) \cdot a_{kl} \cdot e_l(x_{i+1}^{(j)}) \cdot b_l^{(j)}(i+1) \tag{6.37}$$

   (b) Compute the expected number of emissions of the symbol b that occurred at the state k (using the value of $P(\pi_i = k|X)$ as calculated in 6.28):

   $$E_k(b) = \sum_{j=1}^{n} \frac{1}{P(X^{(j)})} \cdot \sum_{\{i | x_i^{(j)} = b\}} f_k^{(j)}(i) \cdot b_k^{(j)}(i) \tag{6.38}$$

3. *Maximization*: Re-compute the new values for Θ from A_{kl} and $E_k(b)$, as explained above (in case 1).

4. Repeat steps 2 and 3 until the improvement of $Score(X^{(1)}, \ldots, X^{(n)}|\Theta)$ is less then a given parameter ϵ.

The EM-algorithm guarantees that the values of the target function $Score(X^{(1)}, \ldots, X^{(n)}|\Theta)$ are monotonically increasing, and as logarithms of probabilities are certainly bounded by 0, the algorithm is guaranteed to converge. It is important to notice that the convergence is of the target function and not in the Θ space: the values of Θ may change drastically even for almost equal values of the target function, which may imply that the obtained solution is not stable.

The main problem with the Baum-Welch algorithm is that there may exist several local maxima of the target function and it is not guaranteed that we reach the global maximum: the convergence may lead to a local maximum. A useful way to circumvent this pitfall is to run the algorithm several times, each time with different initial values for Θ. If we reach the same maximum most of the times, it is highly probable that this is indeed the global maximum.

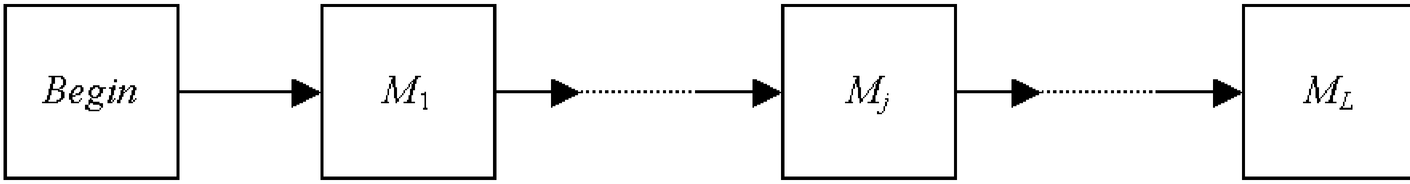

Figure 6.2: Match states in a profile HMM

6.2 Profile Alignment

6.2.1 Profile HMMs

HMMs can be used for aligning a string versus a given profile, thus helping us to solve the multiple alignment problem.

We define a profile $\mathcal{P}$ of length L, as a set of probabilities, consisting of, for each $b \in \Sigma$ and $1 \leq i \leq L$, the probability $e_i(b)$ of observing the symbol b at the i^{th} position. In such a case the probability of a string $X = (x_1, \ldots, x_L)$ given the profile $\mathcal{P}$ will be:

$$P(X|\mathcal{P}) = \prod_{i=1}^{L} e_i(x_i) \tag{6.39}$$

We can calculate a likelihood score for the ungapped alignment of X against the profile $\mathcal{P}$:

$$Score(X|\mathcal{P}) = \sum_{i=1}^{L} \log \frac{e_i(x_i)}{p(x_i)} \tag{6.40}$$

where $p(b)$ is the background frequency of occurrences of the symbol b.

This leads to a definition of the following HMM, with the *match states* $M_1, \ldots, M_L$ which correspond to matches with the profile. All those states are sequentially linked (i.e., each match state M_j is linked to its successor M_{j+1}) as shown in figure 6.2. The emission probability of the symbol b from the state M_j is of course $e_j(b)$.

To allow insertions, we will add the *insertion states* $I_0, \ldots, I_L$ to the model. We shall assume that:

$$\forall_{b\in\Sigma} \quad e_{I_j}(b) = p(b)$$

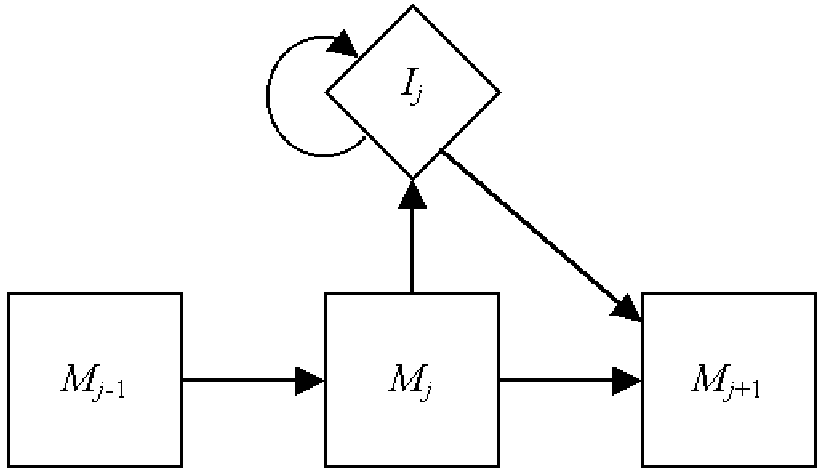

Figure 6.3: Profile HMM with an insertion state

Each insertion state I_j has an link entering from the corresponding match state M_j, a leaving link towards the next match state M_{j+1} and also has a self-loop (see figure 6.3). Assigning the appropriate probabilities for those transitions corresponds to the application of affine gap penalties, since the overall contribution of a gap of length h to the logarithmic likelihood score is:

$$\underbrace{\log(a_{M_jI_j}) + \log(a_{I_jM_{j+1}})}_{\text{gap creation}} + \underbrace{(h-1)\cdot\log(a_{I_jI_j})}_{\text{gap extension}}$$

To allow deletions as well, we add the *deletion states* $D_1, \ldots, D_L$. These states cannot emit any symbol and are therefore called *silent* (Note that the *begin/end* states are silent as well). The deletion states are sequentially linked, in a similar manner to the match states and they are also interleaved with the match states (see figure 6.4).

To model both insertions and deletions, we have to add a link from D_j to I_j and a link from I_j to D_{j+1}.

The full HMM for modeling the profile $\mathcal{P}$ of length L is comprised of L layers, each layer has three states M_j, I_j and D_j. To complete the model, we add *begin* and *end* states, connected to the layers as shown in figure 6.5. This model is due to Haussler et al [5].

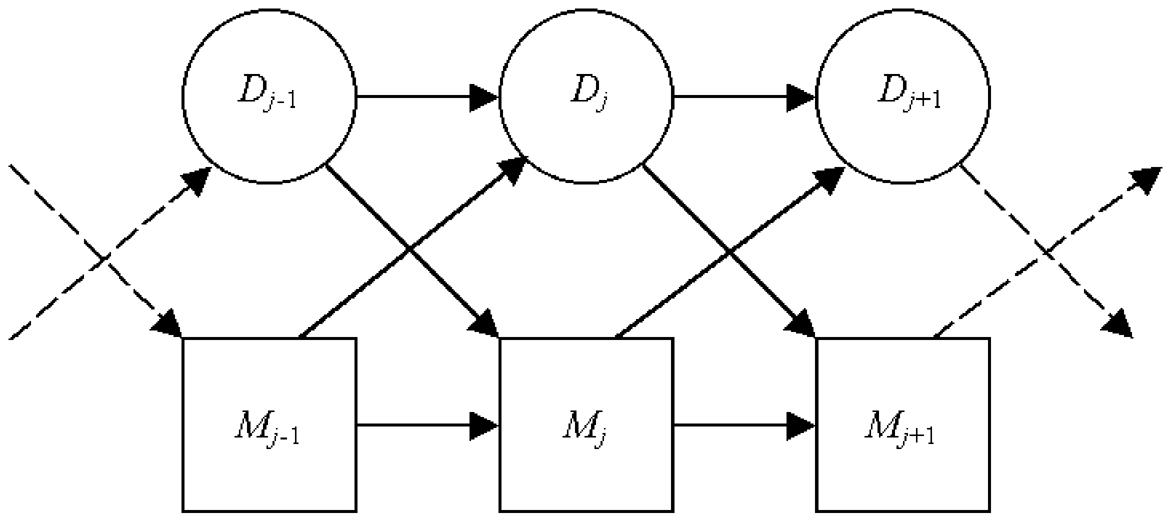

Figure 6.4: Profile HMM with deletion states

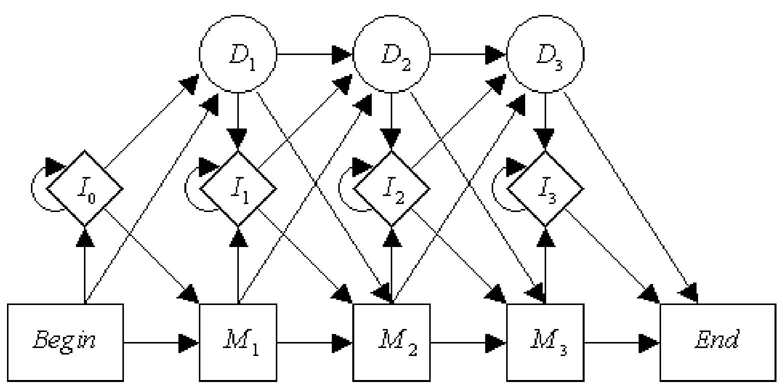

Figure 6.5: Profile HMM for global alignment

6.2.2 Aligning Sequences to a Profile HMM

To align the string $X = (x_1, \dots, x_m)$ against a profile $\mathcal{P}$ of length L, we will use a variant of the Viterbi algorithm. For each $1 \leq j \leq L$ and $1 \leq i \leq m$ we use the following definitions:

- Let $v_j^M(i)$ be the logarithmic likelihood score of the best path for matching $X = (x_1, \dots, x_i)$ to the profile HMM $\mathcal{P}$, ending with x_i emitted by the state M_j.
- Let $v_j^I(i)$ be the logarithmic likelihood score of the best path for matching $X = (x_1, \dots, x_i)$ to the profile HMM $\mathcal{P}$, ending with x_i emitted by the state I_j.
- Let $v_j^D(i)$ be the logarithmic likelihood score of the best path for matching $X = (x_1, \dots, x_i)$ to the profile HMM $\mathcal{P}$, ending with the state D_j (without emitting any symbol).

The initial value of the special *begin* state is:

$$v_{begin}(0) = 0 \tag{6.41}$$

To calculate the values of $v_j^M(i)$, $v_j^I(i)$ and $v_j^D(i)$ we use the same technique as in the Viterbi algorithm. There are however two major differences:

- Each state in the model has at most three entering links (see figure 6.5).
- The deletion states are silent - they cannot emit any symbol.

The three predecessors of the match state M_j are the three states of the previous layer, $j-1$:

$$v_j^M(i) = \log \frac{e_{M_j}(x_i)}{p(x_i)} + \max \begin{cases} v_{j-1}^M(i-1) + \log(a_{M_{j-1},M_j}) \\ v_{j-1}^I(i-1) + \log(a_{I_{j-1},M_j}) \\ v_{j-1}^D(i-1) + \log(a_{D_{j-1},M_j}) \end{cases} \tag{6.42}$$

The three predecessors of the insertion state I_j are the three states of the same layer, j:

$$v_j^I(i) = \log \frac{e_{I_j}(x_i)}{p(x_i)} + \max \begin{cases} v_j^M(i-1) + \log(a_{M_j,I_j}) \\ v_j^I(i-1) + \log(a_{I_j,I_j}) \\ v_j^D(i-1) + \log(a_{D_j,I_j}) \end{cases} \tag{6.43}$$

The three predecessors of the deletion state D_j are the three states of the layer $j-1$. Since D_j is a silent state, we should not consider the emission likelihood score for x_i in this case:

$$v_j^D(i) = \max \begin{cases} v_{j-1}^M(i) + \log(a_{M_{j-1},D_j}) \\ v_{j-1}^I(i) + \log(a_{I_{j-1},D_j}) \\ v_{j-1}^D(i) + \log(a_{D_{j-1},D_j}) \end{cases} \tag{6.44}$$

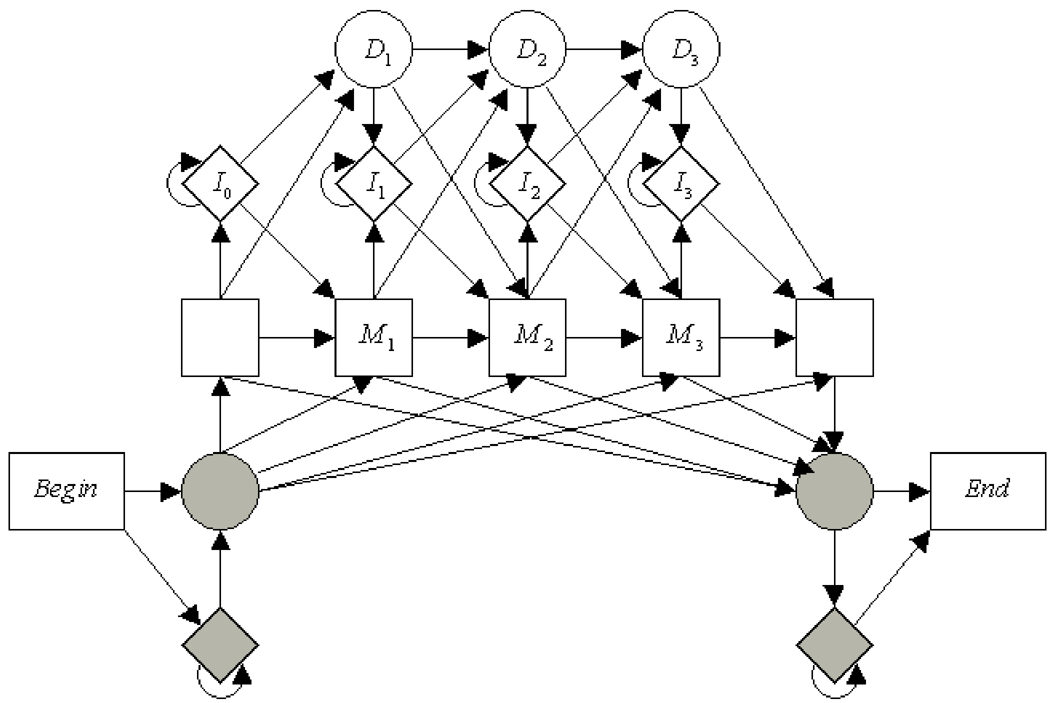

Figure 6.6: Profile HMM for local alignment

We conclude by calculating the optimal score:

$$Score(X|\Pi^*) = \max \begin{cases} v_L^M(m) + \log(a_{M_L,end}) \\ v_L^I(m) + \log(a_{I_L,end}) \\ v_L^D(m) + \log(a_{D_L,end}) \end{cases} \tag{6.45}$$

Complexity: We have to calculate $O(L \cdot m)$ values, while calculating each value takes $O(1)$ operations (since we only need to consider the scores at most three predecessors). We therefore need $O(L \cdot m)$ time and $O(L \cdot m)$ space.

We can use a similar approach for the problem of local alignment of a sequence versus a profile HMM. This is achieved by adding four additional states (the lightly shaded states in figure 6.6) corresponding to the alignment of a sub-string of X to a part of the profile.

6.2.3 Forward and Backward Probabilities for a Profile HMM

In the previous section we modeled the problem of aligning a string to a profile. As with general HMMs, the main problem is to assign meaningful values to the transition and emission probabilities to a profile HMM. It is possible to use the Baum-Welch algorithm for training the model probabilities, but it remains to show how to compute the forward and backward probabilities needed for the algorithm.

Given a string $X = (x_1, \ldots, x_m)$ we define:

- The forward probabilities:

$$f_j^M(i) = P(x_1, \ldots, x_i \text{ ending at } M_j) \tag{6.46}$$
$$f_j^I(i) = P(x_1, \ldots, x_i \text{ ending at } I_j) \tag{6.47}$$
$$f_j^D(i) = P(x_1, \ldots, x_i \text{ ending at } D_j) \tag{6.48}$$

- The backward probabilities:

$$b_j^M(i) = P(x_{i+1}, \ldots, x_m \text{ beginning from } M_j) \tag{6.49}$$
$$b_j^I(i) = P(x_{i+1}, \ldots, x_m \text{ beginning from } I_j) \tag{6.50}$$
$$b_j^D(i) = P(x_{i+1}, \ldots, x_m \text{ beginning from } D_j) \tag{6.51}$$

Computing the Forward Probabilities:

1. Initialization:

$$f_{begin}(0) = 1 \tag{6.52}$$

2. Recursion:

$$\begin{aligned} f_j^M(i) = e_{M_j}(x_i) \cdot [&f_{j-1}^M(i-1) \cdot a_{M_{j-1},M_j} + \\ &f_{j-1}^I(i-1) \cdot a_{I_{j-1},M_j} + \\ &f_{j-1}^D(i-1) \cdot a_{D_{j-1},M_j}] \end{aligned} \tag{6.53}$$

$$\begin{aligned} f_j^I(i) = e_{I_j}(x_i) \cdot [&f_j^M(i-1) \cdot a_{M_j,I_j} + \\ &f_j^I(i-1) \cdot a_{I_j,I_j} + \\ &f_j^D(i-1) \cdot a_{D_j,I_j}] \end{aligned} \tag{6.54}$$

$$\begin{aligned} f_j^D(i) = &f_{j-1}^M(i) \cdot a_{M_{j-1},D_j} + \\ &f_{j-1}^I(i) \cdot a_{I_{j-1},D_j} + \\ &f_{j-1}^D(i) \cdot a_{D_{j-1},D_j} \end{aligned} \tag{6.55}$$

Computing the Backward Probabilities:

1. Initialization:

$$b_L^M(m) = a_{M_L,end} \tag{6.56}$$

$$b_L^I(m) = a_{I_L,end} \tag{6.57}$$

$$b_L^D(m) = a_{D_L,end} \tag{6.58}$$

2. Recursion:

$$\begin{aligned} b_j^M(i) = {} & b_{j+1}^M(i+1) \cdot a_{M_j,M_{j+1}} \cdot e_{M_{j+1}}(x_{i+1}) + \\ & b_j^I(i+1) \cdot a_{M_j,I_j} \cdot e_{I_j}(x_{i+1}) + \\ & b_{j+1}^D(i) \cdot a_{M_j,D_{j+1}} \end{aligned} \tag{6.59}$$

$$\begin{aligned} b_j^I(i) = {} & b_{j+1}^M(i+1) \cdot a_{I_j,M_{j+1}} \cdot e_{M_{j+1}}(x_{i+1}) + \\ & b_j^I(i+1) \cdot a_{I_j,I_j} \cdot e_{I_j}(x_{i+1}) + \\ & b_{j+1}^D(i) \cdot a_{I_j,D_{j+1}} \end{aligned} \tag{6.60}$$

$$\begin{aligned} b_j^D(i) = {} & b_{j+1}^M(i+1) \cdot a_{D_j,M_{j+1}} \cdot e_{M_{j+1}}(x_{i+1}) + \\ & b_j^I(i+1) \cdot a_{D_j,I_j} \cdot e_{I_j}(x_{i+1}) + \\ & b_{j+1}^D(i) \cdot a_{D_j,D_{j+1}} \end{aligned} \tag{6.61}$$

6.2.4 Multiple Alignment with Profile HMMs

Profile HMMs can help us to obtain an approximate solution to the multiple alignment problem. Given n sequences $S^{(1)}, \ldots, S^{(n)}$, consider the following cases:

1. If the profile HMM $\mathcal{P}$ is known, the following procedure can be performed:
   - Align each sequence $S^{(i)}$ to the profile separately.
   - Accumulate the obtained alignments to a multiple alignment.
2. If the profile HMM $\mathcal{P}$ is not known, one can use the following technique in order to obtain an HMM profile from the given sequences:

- Choose a length L for the profile HMM and initialize the transition and emission probabilities.
- Train the model using the Baum-Welch algorithm.
- Obtain the multiple alignment from the resulting profile HMM, as in the previous case.

One can use an extension of the above approach to identify similar patterns in a given set of sequences by using the profile HMM for local alignment (see figure 6.6). We now present another approach to this problem, which tackles the problem from a totally different perspective.

6.3 Gibbs Sampling

Problem 6.5 Locating a common pattern.
INPUT: A set of sequences $\mathcal{S} = S^{(1)}, \ldots, S^{(n)}$ and an integer w.
QUESTION: For each string $S^{(i)}$, find a sub-string of length at most w, so that the similarity between the n sub-strings is maximized.

Let $a^{(1)}, \ldots, a^{(n)}$ be the starting indices of the chosen sub-strings in $S^{(1)}, \ldots, S^{(n)}$, respectively. We introduce the following notations:

- Let c_{ij} be the number of occurrences of the symbol $j \in \Sigma$ among the i^{th} positions of the n sub-strings: $\{s^{(1)}_{a^{(1)}+i-1}, \ldots, s^{(n)}_{a^{(n)}+i-1}\}$.
- Let q_{ij} denote the probability of the symbol j to occur at the i^{th} position of the pattern.
- Let p_j denote the frequency of the symbol j in all sequences of $\mathcal{S}$.

We therefore wish to maximize the logarithmic likelihood score:

$$Score = \sum_{i=1}^{w} \sum_{j \in \Sigma} c_{ij} \cdot \log \frac{q_{ij}}{p_j} \tag{6.62}$$

To accomplish this task, we perform the following iterative procedure:

1. Initialization: Randomly choose $a^{(1)}, \ldots, a^{(n)}$.
2. Randomly choose $1 \leq z \leq n$ and calculate the c_{ij}, q_{ij} and p_j values for the strings in $\mathcal{S} \setminus S^{(z)}$.

3. Find the best substring of $S^{(z)}$ according to the model, and determine the new value of $a^{(z)}$. This is done by applying the algorithm for local alignment for $S^{(z)}$ against the profile of the current pattern.

4. Repeat steps 2 and 3 until the improvement of the score is less then ϵ.

Unlike the profile HMM technique, the Gibbs sampling algorithm (due to Lawrence et al. [8]) does not rely on any substantial theoretic basis. However, this method is known to work in specific cases.

Known problems:

- *Phase shift* - The algorithm may converge on an offset of the best pattern.
- The value of w is usually unknown. Choosing different values for w may significantly change the results.
- The strings may contain more than a single common pattern.
- As it is the case with the Baum-Welch algorithm, the process may converge to a local maximum.

References

- Hidden Markov models were originally invented and are commonly used for speech recognition. For additional material on this subject, see [9].
- For a more comprehensive overview on the EM technique, see [1].
- For a detailed discussion about the adaptation of HMMs to computational biology algorithms see [4] (chapters 3-6).
- Applications of profile HMMs for multiple alignment of proteins can be found in [5] and for finding genes in the DNA can be found in [7, 6].

Bibliography

[1] N. M. Laird A. P. Dempster and D. B. Rubin. Maximum likelihood estimation from incomplete data. *Journal of the Royal Statistics Society*, 39:1–38, 1977.

[2] L. E. Baum. An inequality and associated maximization technique occuring in the statistical analysis of probabilistic functions of markov chains. *Inequalities*, 3:1–8, 1972.

[3] R. Bellman. *Dynamic Programming.* Princeton University Press, Boston, 1957.

[4] R. Durbin, S. Eddy, A. Krough, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, 1998.

[5] A. Krogh, M. Brown, S. Mian, M. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. Technical Report UCSC-CRL-93-32, Department of Computer and Information Sciences, University of California at Santa Cruz, 1993.

[6] Anders Krogh, I. Saira Mian, and David Haussler. A hidden markov model that finds genes in E. coli DNA. *Nucleic Acids Research*, 22:4768–4778, 1994.

[7] D. Kulp, D. Haussler, M.G. Reese, and F.H. Eeckman. A generalized hidden Markov model for the recognition of human genes in DNA. In D. J. States, P. Agarwal, T. Gaasterland, L. Hunter, and R. Smith, editors, *Proc. Conf. On Intelligent Systems in Molecular Biology '96*, pages 134–142. AAAI/MIT Press, 1996. St. Louis, Mo.

[8] Charles E. Lawrence, Stephen F. Altschul, Mark S. Boguski, Jun S. Liu, Andrew F. Neuwald, and John C. Wootton. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262:208–214, 8 October 1993.

[9] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[10] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Information Theory*, IT-13:260–269, 1967.