Chapter 6

Hidden Markov Models

We have been discussing conserved motifs or patterns found in multiple sequences, with a focus on three major tasks:

Discovery: Given a set of unlabeled sequences, identify an unknown pattern that is common to the input sequences.

Modeling: Given a set of sequences containing instances of a pattern, construct a compact, probabilistic representation of the pattern.

Recognition: Given a pattern model and a new unlabeled sequence, determine whether the pattern is present in the sequence and, if so, label the sites that correspond to the pattern.

In previous lectures, we considered Position Specific Scoring Matrices (PSSMs) for modeling and recognition of ungapped patterns or motifs and the Gibbs Sampler for discovering such motifs in unlabeled data. PSSMs work well for fixed length patterns in which the sites are more or less independent. However, there are other kinds of conserved patterns for which PSSMs are not well suited.

In the following lectures, we introduce *Hidden Markov models (HMMs)*, a more general and powerful formalism that can be applied to a wider variety of conserved patterns. We first examine some of the limitations of PSSMs:

1. *PSSMs cannot model positional dependencies:* Suppose, for example, that a motif has a positional dependency like this one, in which we see either `RD` or `QH` in the last two positions, but never `QD` or `RH`.

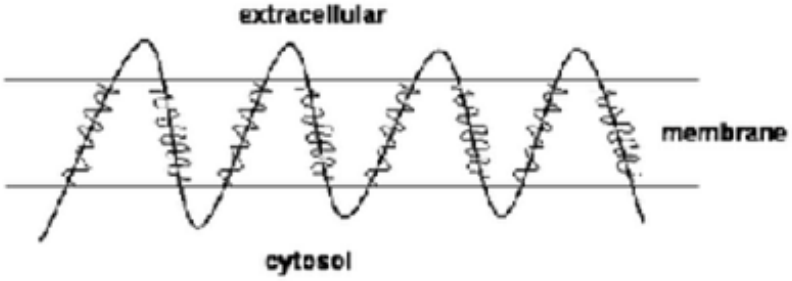


Figure 6.1: A transmembrane protein

```
WEIRD
WEIRD
WEIQH
WEIRD
WEIQH
WEIQH
```

A PSSM for this motif would give a high scores to `WEIRD`, which conforms to the positional dependency, but also to `WEIRH`, which does not.

2. *PSSMs cannot recognize motif instances containing insertions or deletions:* A PSSM for the `WEIRD` motif would not assign high scores to pattern instances with insertion and deletions, such as `WECIRD` and `WERD`.

3. *PSSMs cannot model variable length patterns:* Membrane-bound regions in a transmembrane protein (Fig. 6.1) are one example of a variable length pattern. Sequence segments in the membrane are enriched for hydrophobic residues, relative to segments in the cytosol or the extracellular matrix, providing a signal that can be exploited for transmembrane region detection. A PSSM is not suitable for recognizing this type of feature.

4. *PSSMs are not able to detect the precise boundaries of a pattern:* We are given a sequence and we wish to label each symbol in the sequence according to its class (e.g. subcellular localization; introns and exons; alpha helices and beta sheets; genes and non-coding regions). It is difficult to identify sharp boundaries between classes by scoring windows using a fixed size model, such as a PSSM, as illustrated in Fig. 6.2.

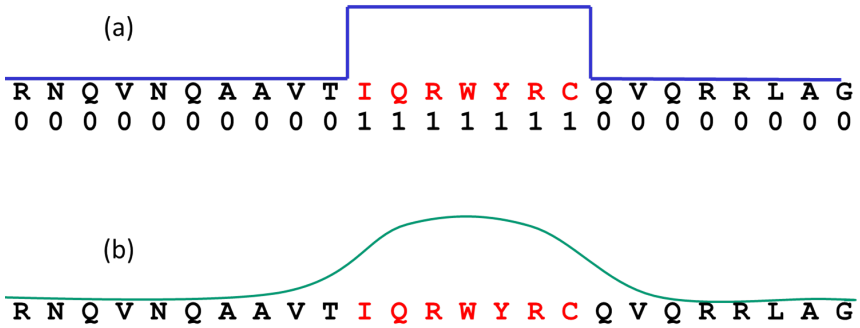


Figure 6.2: Boundary detection. (a) Perfect boundary detection identifies the exact position of a motif (`IQRWYRC`) in an amino acid sequence. (b) Scoring a sliding window identifies the general neighborhood of the motif, but cannot identify its precise position.

6.1 Modeling variable length patterns with Markov chains

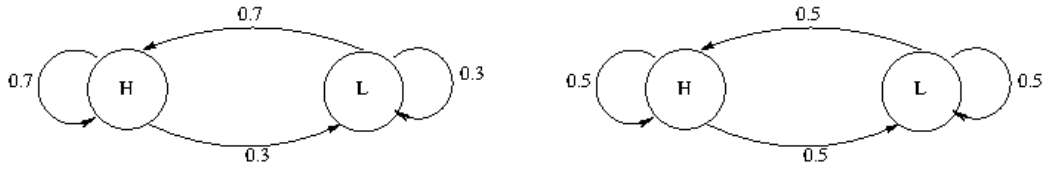
Hidden Markov models are Markov models. Since we are already familiar with Markov chains, we will start by demonstrating how a Markov chain can be used to model a variable length pattern. However, as we will see, Markov chains are not useful for boundary detection. We next introduce Hidden Markov models and show how the the additional features of HMMs allow for the inference of precise boundaries. In subsequent lectures, we will explore how HMMs can be used to model indels and, up to a point, positional dependence.

Recall that a finite, discrete-time Markov chain is defined by

- a finite set of *states*, $E_0, E_1, \ldots E_N$;
- a *transition probability*, P_{xy}, which is the probability that the chain will be in state E_y at time $t+1$ given that it was in state E_x at the time, t; and
- an *initial state probability distribution*, $\pi = (\pi_1, \pi_2 \ldots, \pi_N)$, where π_x gives the probability of being in state E_x at time $t = 0$.

Suppose we are given an unlabeled amino acid sequence, $O = O_1, O_2, \ldots O_T$, as input and we wish to assign a label to each residue, O_i, that indicates whether or not O_i is associated with a particular conserved pattern. For example, if O is a transmembrane protein sequence, we may wish to label those residues that are localized to the membrane. To further simplify the exposition, we will assume that the unlabeled sequence has been recoded in a two letter alphabet, $\Sigma_{H,L} = \{H, L\}$, in which each amino acid is replaced with an H if the residue is hydrophobic and an L if it is hydrophilic.

Let us first consider a simpler problem: given a sequence fragment that is either a subsequence of a transmembrane (TM) region or a subsequence of an extracellular or



(a) Transmembrane model ($\pi_H = 0.7, \pi_L = 0.3$) (b) Cytosol/ECM model ($\pi_H = \pi_L = 0.5$)

Figure 6.3: Markov models of sequence fragments localized to (a) the membrane or (b) the cytosol or extracellular matrix.

cytosolic (E/C) region, determine whether or not the fragment is localized to the membrane. To do this, we construct a Markov model of transmembrane sequences in which hydrophobic residues are preferred (Fig. 6.3a). Our model has two states, H and L, corresponding to hydrophobic and hydrophillic residues, respectively. Since the Markov model has one state for each symbol in Σ, a sequence of symbols corresponds to a sequence of states in the Markov chain. The probability that a given sequence $O \in \Sigma_{H,L}^*$ is localized to the membrane can be estimated by the probability of visiting the states corresponding to $O_1, O_2 \ldots$:

$$P(O|TM) = \pi_1 \prod_{i=2}^{T} P_{O_{i-1}O_i}.$$

For comparison, we also calculate the probability of O with respect to a model of extracellular and cytosolic sequences (Fig. 6.3b) in which hydrophobic and hydrophilic residues occur with equal frequencies.

Using these models, we can classify an observed sequence, $O = O_1, O_2, \ldots O_T$, by its likelihood ratio

$$\frac{P(O|TM)}{P(O|E/C)}, \tag{6.1}$$

where the probability of O with respect to a Markov model is $\pi_{O_1} \prod_2^T P_{O_{i-1}O_i}$. If the likelihood ratio is much greater than one, then we infer that O is located in the membrane.

For example, $P(\texttt{HHLHH}|TM) = 0.7 \times 0.7 \times 0.3 \times 0.7 \times 0.7 \approx 0.072$ and $P(\texttt{HHLHH}|E/C) = 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5 \approx 0.031$. The likelihood ratio is 2.3; in other words, it is a little more than twice as likely that `HHLHH` is a transmembrane sequence than an E/C sequence.

Note that the likelihood of a sequence is sensitive to the length of the sequence; the longer the sequence, the lower its probability. Intuitively, this makes sense since the number of sequences of length T grows exponentially with T. As a result, $P(\texttt{HHHHHHHHHH}|TM) = 0.03 < P(\texttt{HHLHH}|TM) = 0.07$, even though `HHHHHHHHHH` looks more like a transmembrane segment than `HHLHH`. In contrast, the likelihood ratio in Equation 6.1 is not sensitive to the length of O because $P(O|TM)$ is normalized by $P(O|E/C)$. The likelihood ratio for `HHHHHHHHHH` is roughly 29, which is much greater than 2.3, consistent with our expectations.

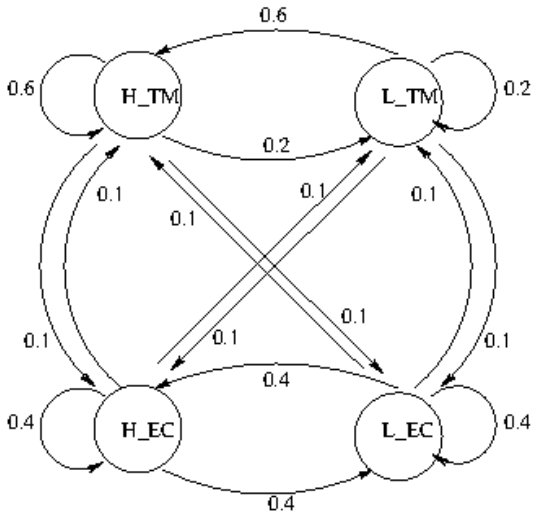


Figure 6.4: A four-state transmembrane HMM

The models (Fig. 6.3) are useful for classifying a sequence fragment where all residues are of the same class (i.e., all TM or all E/C). However, for finding boundaries in a sequence that has transitions from regions of one class to regions of another class, we would still need to score successive overlapping windows along the sequence, leading to a fuzzy boundary. For this question, the Markov chain has the same problem as the PSSM.

In order to determine the location of the transmembrane regions in an unlabeled sequence, a model that assigns a probability to a sequence is not sufficient; we need a model that explicitly labels each residue with its class (TM or E/C). For this purpose, we constructed a new model by adding transitions connecting the TM and E/C models (Fig. 6.4).

This four-state model is much better suited to the boundary detection problem: The transitions between the M states and the E/C states indicate the boundaries between membrane regions and cytosolic or extracellular regions. However, this four-state model is not a standard Markov chain; it is a Hidden Markov model. In a Markov chain, every sequence of symbols corresponds to a unique sequence of states. That is not true of the four-state model, above. In the four-state model, there are two states associated with hydrophobic residues and two states associated with hydrophilic residues. As a result, for any given sequence of `H`'s and `L`'s, there are multiple paths through the states of the model and each of these state paths is associated with a different probability. The word "hidden" refers to the fact that the true sequence of states for a given sequence of symbols is unknown or hidden. As we will see in the next section, estimating the true sequence of states requires more complex algorithms, but provides a solution to the boundary detection problem.

6.2 Hidden Markov Models

Hidden Markov models, or HMMs, are defined formally as follows:

1. N states $E_1..E_N$
2. An alphabet, $\Sigma = \{\sigma_1, \sigma_2 \ldots \sigma_M\}$
3. Transition probability matrix a_{ij}
4. Emission probabilities: $e_i(\sigma)$ is the probability that state E_i emits $\sigma \in \sum$
5. Initial distribution vector $\pi = (\pi_i \cdots \pi_N)$

We refer to the transition probabilities, the emission probabilities, and the initial distribution, collectively, as the parameters of the model, designated $\lambda = (a_{ij}, e_i(a), \pi_i)$. Following the notation used in Ewens and Grant and in Durbin, we will refer to the sequence of observed symbols as $O = O_1, O_2, O_3, ...O_T$ and the sequence of states visited as $Q = q_1, q_2, q_3, ...q_T$. To avoid confusing "sequences of symbols" with "sequences of states," from now on we will use the term *state path* to designate a sequence of states. When considering more than one sequence or state path, we will use superscripts to distinguish them: $O^d = O_1^d, O_2^d, O_3^d, ...$ and $Q^b = q_1^b, q_2^b, q_3^b, ...$

HMMs differ from Markov chains in a number of ways. First, a state in an HMM emits symbols from a fixed alphabet each time the state is visited. Emission probabilities are state-dependent, but do not change over time. Note that an HMM is a *generative* model. It gives the probability of generating a particular sequence (hence, the emission probabilities). This allows us to ask: Given an observed sequence, O, and an HMM model, what is the probability that O was generated by this HMM?

Second, unlike Markov chains, in an HMM there is no longer a one-to-one correspondence between states and symbols. In a Markov chain, for a given sequence, O, there is a unique state path corresponding to O, which determines the probability of O with respect to the model. In contrast, in an HMM, there may be more than one, and often very many, state paths associated with O. Therefore, the "true" sequence of states that generated the observed sequence is unknown, or *hidden*, hence the term, "Hidden" Markov model. This hidden sequence of states corresponds to what we want to know, namely the classification of each symbol.

In the four-state model above, we used two states to encode TM sequences, one for hydrophobic residues and one for hydrophilic residues. Another possibility is to use only one state for the transmembrane region and use the emission probabilities to distinguish between hydrophobic and hydrophilic residues as shown in Fig. 6.5. This approach is used in the following three-state HMM. Notice that this model is also a bit more sophisticated than our previous models because it distinguishes between extracellular residues and cytosolic residues using separate E and C states. The initial and emission probabilities for this model are

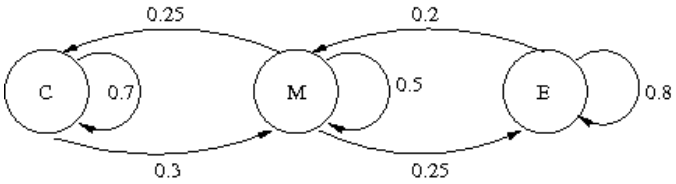


Figure 6.5: Three-state transmembrane HMM

E_i	C	M	E
π_i	0.5	0.0	0.5
$e_i(H)$	0.3	0.9	0.2
$e_i(L)$	0.7	0.1	0.8

In this example, we assume that all transmembrane sequences are equally likely to start in the cytosol or in the extracellular matrix (ECM); i.e., $\pi_C = 0.5$ and $\pi_E = 0.5$.

In this HMM model, the subcellular location of each residue is represented by the state that emitted the symbol associated with that residue. There are many state paths that can generate a given sequence of amino acids. If we are given both the observed sequence and the state path, then calculating the probability is straight-forward. Given a sequence $O = O_1, O_2, \ldots O_T$ and a state path $Q = q_1, q_2, \ldots q_T$, the probability of visiting the states in Q and emitting O is

$$P(O, Q|\lambda) = \pi_{q_1} \cdot e_{q_1}(O_1) \cdot a_{q_1 q_2} \cdot e_{q_2}(O_2) \cdot a_{q_1 q_2} \cdot e_{q_3}(O_3) \ldots a_{q_{T-1}\, q_T} \cdot e_{q_T}(O_T).$$

For example, suppose O = `LHHHL` and Q = *CMMME*, then

$$P(\texttt{ LHHHL, CMMME}|\lambda) = \pi_C \cdot e_C(\texttt{ L}) \cdot a_{CM} \cdot e_M(\texttt{ H}) \cdot a_{MM} \cdot e_M(\texttt{ H}) \cdot a_{MM} \cdot e_M(\texttt{ H}) \cdot a_{ME} \cdot e_E(\texttt{ L}).$$

In general, however, the state path that actually generated a given protein sequence is unknown. In order to infer the location of the transmembrane regions, we must infer the "true" state path that generated the protein sequence. The boundaries between the transmembrane, extracellular and cytosolic regions are precisely defined by the transitions between the C, M, and E states along this state path. The predicted subcellular location of each residue in the sequence is the state (C, M, or E) that emitted that residue.

Fig. **??** shows a cartoon of the probability distribution of all possible combinations of sequences and state paths for a hypothetical HMM. Every point on the horizontal plane corresponds to a particular sequence, O^d, and a particular state path, Q^b. The value on the vertical axis is the joint probability, $P(O^d, Q^b|\lambda)$, that the HMM will visit the states on path Q^b and emit sequence O^d. In the three-state TM model example, the set of all possible sequences, $O^1, O^2, O^3, \ldots$ corresponds to `H, L, HH, HL, LH, LL, HHH, ...` and the set of all possible state paths, $Q^1, Q^2, Q^3, \ldots$ corresponds to *C, M, E, CC, CM, CE, MC, ...*

Note that $P(O^d, Q^b) = 0$ for many (O^d, Q^b) pairs. For example, $P(O^d, Q^b) = 0$ when O^d and Q^b have different lengths. In our three-state model, $P(O^d, Q^b) = 0$ for any state path that contains C adjacent to E, because $a_{CE} = 0$.

An HMM emits each sequence $O^d \in \Sigma^*$ with probability $P(O^d) \geq 0$. Since a sequence can, potentially, be emitted from more than one state path, in order to obtain the total probability of a sequence, O, we must sum over the all possible paths:

$$P(O) = \sum_b P(O|Q^b, \lambda) \cdot P(Q^b) = \sum_b P(O, Q^b|\lambda).$$

Fig. **??** shows a cartoon representation of $P(O, Q)$ for a single sequence, O^5, for the set of all possible state paths, Q. The area under the curve is equal to $P(O)$, the total probability of sequence O.

When all possible sequences and all possible paths are considered, the probability distribution shown in Fig. **??** sums to one:

$$\sum_d \sum_b P(O^d, Q^b|\lambda) = 1.$$

6.3 Using HMMs for recognition

In this section, we focus on motif recognition using HMMs. We will discuss parameter estimation, motif discovery, and modeling using HMMs in future sections. Here, we assume that we are given an HMM with known parameter values.

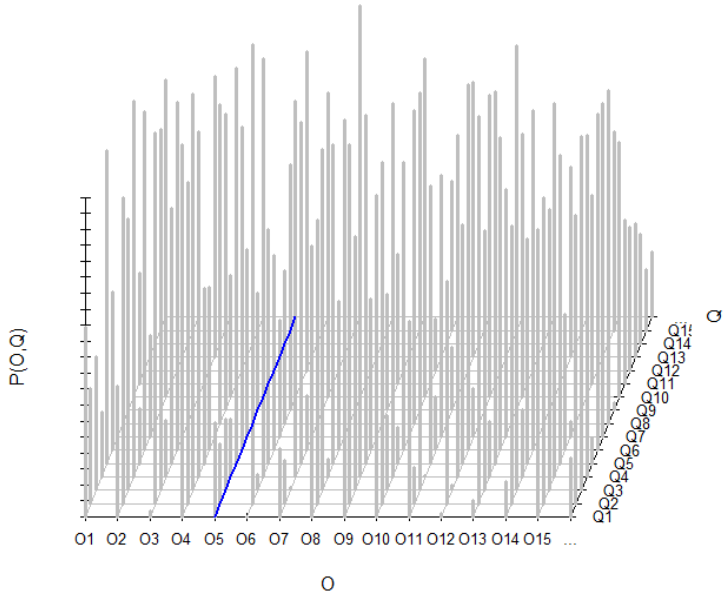
Our goal is to use the HMM to answer the various recognition questions, including:

1. What is the probability that a given sequence, O, was generated by the HMM?
   Example: Is the sequence a transmembrane protein?

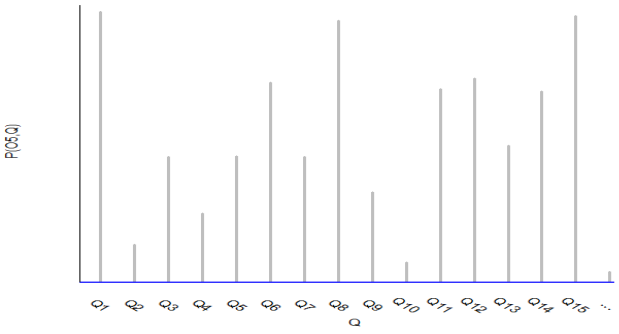
2. Given a sequence O, what is the true path? Otherwise stated, we wish to assign labels to an unlabeled sequence.
   Example: Identify the cytosolic, transmembrane, and extracellular regions in the sequence. In this case, we wish to assign the labels E, M, or C to each amino acid residue in the sequence.

3. What is the probability of being in state E_i when O_t is emitted?
   Example: Is a given residue localized to the membrane?

Since an HMM is a generative model, an HMM can also used for simulation; for example, to generate sequences with properties similar to real transmembrane sequences.



(a) The joint probability, $P(O^d, Q^b)$.



(b) $P(O^5, Q^b)$

Figure 6.6: **(a)** The joint probability $P(O^d, Q^b)$ for every sequence O^d and state path Q^b. The volume under this curve is one. **(b)**The probability of sequence $O = O^5$ for every state path $Q^1, Q^2, Q^3 \ldots$. This curve corresponds to the intersection of the probability distribution in Fig. **??** and the vertical plane at $O = O^5$ (shown as a blue line in Fig. **??**). The area under this curve is $P(O^5|\lambda)$, the probability of O^5. The maximum point on the curve is the most probable path, $Q^* = \text{argmax}_Q P(Q|O, \lambda)$ is the highest point on this curve.

6.4 Calculating the total probability of sequence O

In order to answer the first question,

1. What is the probability that a given sequence, O, was generated by the HMM?

we must calculate $P(O|\lambda)$, the total probability of the sequence given the model. The total probability is the probability of visiting states in path Q and emitting sequence O, summed over all possible state paths:

$$P(O|\lambda) = \sum_b P(O|Q^b, \lambda) \cdot P(Q^b|\lambda) = \sum_b P(O, Q^b|\lambda).$$

We could calculate this quantity by enumerating all paths, Q^b, and calculating $P(O, Q^b|\lambda)$ for each one, but this brute force approach becomes intractable as the number of states gets large, since the number of state paths grows as $O(N^T)$. Instead, we use a dynamic programming algorithm called the *Forward* algorithm, which recursively calculates the probability of emitting prefixes of O. At each step, the Forward algorithm calculates the probability of emitting the first t symbols, $O_1, O_2, \ldots O_t$, summing over all possible paths that end in state E_i. We designate this quantity

$$\alpha(t, i) = P(O_1, O_2, O_3, ...O_t, q_t = E_i).$$

The variable α is an $T \times N$ matrix, where the rows correspond to prefixes of O and the columns correspond to states. At the t^{th} iteration, the algorithm calculates the entries in the t^{th} row of the matrix, based on the entries in row $t-1$ and the parameters of the model. The entries in the final row, contain the probability of emitting the entire sequence and ending in state E_i. The probability of emitting the entire sequence, independent of the final state, is given by the summing the entries in the last row. The algorithm to calculate $\alpha(t, i)$ for all $t \in (1, T)$ proceeds as follows:

Algorithm: Forward

Initialization:

$$\alpha(1, i) = \pi_i e_i(O_1)$$

Recursion:

$$\alpha(t, i) = \sum_{j=1}^{N} \alpha(t-1, j) \cdot a_{ji} \cdot e_i(O_t)$$

Final:

$$P(O) = \sum_{i=1}^{N} \alpha(T, i)$$

The computational complexity of the Forward algorithm is $O(TN^2)$: There are $T \times N$ cells in the α matrix and the computational cost of computing each cell is $O(N)$.

In class, we worked an example based on the three-state transmembrane model shown in Fig. 6.5. A worksheet for this exercise is linked to the class syllabus page. The solution is also available. I recommend that you try to work through the Forward algorithm before looking at the solution.

6.5 Decoding

Next, we tackle the second recognition question:

2. Given a sequence O, what is the true path?

Given an unlabeled sequence, our goal is to classify or *label* each symbol in the sequence by inferring the state path. This process is called "decoding" because we decode the sequence of symbols to determine their meaning. HMMs were developed in the field of speech recognition, where recorded speech is "decoded" into words or phonemes to determine the meaning of the utterance. In our application, we decode an amino acid sequence to infer the functional role of each residue. There are two common approaches to decoding: Viterbi decoding and posterior decoding.

6.5.1 Viterbi decoding

Viterbi decoding is based on the assumption that the *most probable path*, $Q^* = \text{argmax}_Q\, P(Q|O, \lambda)$, is a good estimation of the sequence of states that generated the observed sequence O.[1] In practice, we maximize the joint probability $P(Q, O|\lambda)$, rather than the conditional $P(Q|O, \lambda)$, but this will still give us the most probable path because the path that maximizes $P(Q, O|\lambda)$ also maximizes $P(Q|O|\lambda)$. To see this, note that

$$P(Q|O, \lambda) = \frac{P(Q, O|\lambda)}{P(O|\lambda)}.$$

Since $P(O|\lambda)$ is independent of Q,

$$\underset{Q}{\text{argmax}}\, P(Q|O, \lambda) = \underset{Q}{\text{argmax}}\, P(Q, O|\lambda).$$

As in the case of the Forward algorithm, the brute approach of enumerating all paths and calculating $P(Q|O, \lambda)$ for each one is intractable, because the number of state paths

[1] Note that the most probable path is not the same as the path that maximizes the likelihood of O.

grows as $O(N^T)$. Instead, we calculate $\text{argmax}_Q P(Q, O|\lambda)$ using a dynamic programming algorithm called the *Viterbi* algorithm. Let $\delta(t, i)$ be the probability of emitting the first t residues via the most probable path that ends in E_i. We calculate $\delta(t, i)$ as follows:

Algorithm: Viterbi

Initialization:

$$\delta(1, i) = \pi_i \cdot e_i(O_1)$$

Recursion:

$$\delta(t, i) = \max_{1 \le j \le N} \delta(t-1, j) \cdot a_{ji} \cdot e_i(O_t)$$

$$j^*(t, i) = \operatorname*{argmax}_{1 \le j \le N} \delta(t-1, j) \cdot a_{ji} \cdot e_i(O_t)$$

Final:

$$P(Q^*, O|\lambda) = \max_{1 \le j \le N} \delta(T, j)$$

$$j^*(T) = \operatorname*{argmax}_{1 \le j \le N} \delta(T, j).$$

At each step in the recursion, we save $j^*(t, i)$, the value of j that maximizes $\delta(t-1) \cdot a_{ji} \cdot e_i(O_t)$. These values are used to reconstruct the most probable path. The final state on the most probable path, q_T^*, is the state that maximizes $\delta(T, j)$. The rest of the state path is reconstructed by tracing back through the dynamic programming matrix, a procedure similar to the traceback in pairwise sequence alignment.

The running time of the Viterbi algorithm is $O(TN^2)$. There are TN entries in the dynamic programming matrix. Each entry requires calculating N terms.

In the transmembrane example, the amino acid sequence is known, but the subcellular location of each residue is hidden. In the HMM model, the subcellular location of each residue is represented by the (hidden) state that emitted the symbol associated with that residue. We infer the subcellular locations of the residues by inferring the sequence of states visited by the HMM. The boundaries between the transmembrane, extracellular and cytosolic regions are precisely defined by the transitions between C, M, and E states along this state path.

In class, we used the three-state HMM shown in Fig. 6.5 as an example. As an exercise, try applying the Viterbi algorithm to determine the most probable path through the three-state HMM for the sequence *HHH*. A worksheet for this exercise is linked to the class syllabus page. The solution is also available. I recommend that you try to work through the Viterbi algorithm before looking at the solution.

6.5.2 The probability that state E_i emitted O.

The third question

3. What is the probability of being in state E_i when O_t is emitted?

is a special case of the decoding problem, where the focus is on classifying one specific residue. The probability of being in state E_i when O_t is emitted is the product of two probabilities: (1) the total probability of emitting $O_1 \dots O_t$ over all paths that end in E_i and (2) the total probability emitting $O_{t+1} \dots O_T$ over all paths, given that the model was in state E_i at time t:

$$P(q_t = E_i, O) = P(O_1, O_2, O_3, ...O_t, q_t = E_i) \cdot P(O_{t+1}, O_{t+2}, ...O_T | q_t = E_i). \tag{6.2}$$

Note that the first term is just $\alpha(t, i)$, as defined in the section on the Forward algorithm. To calculate the second term, we introduce $\beta(t + i)$, the probability of emitting $O_{t+1}, O_{t+2}, ...O_T$ given that O_t was emitted from state E_i:

$$\beta(t + 1, i) = P(O_{t+1}, O_{t+2}, ...O_T | q_t = E_i).$$

Substituting α and β for the first and second terms in Equation 6.2, we obtain the following expression for the probability of emitting O_t from state E_i

$$P(q_t = E_i, O_t) = \alpha(t, i) \cdot \beta(t + 1, i). \tag{6.3}$$

The first term, $\alpha(t, i)$, is calculated using the Forward algorithm. The second term, $\beta(t+1, i)$, is calculated using an algorithm called the Backward algorithm. Like the Forward and Viterbi algorithms, the Backward algorithm is a dynamic programming algorithm. However, the Backward algorithm is different in that we start by calculating the probability of emitting the last symbol, O_T, and then work backwards from O_T to O_{t+1}.

Algorithm: Backward

Initialization:

$$\beta(T, i) = \sum_{j=1}^{N} a_{ij} \cdot e_j(O_T)$$

Recursion:

$$\beta(t, i) = \sum_{j=1}^{N} a_{ij} \cdot e_j(O_t) \cdot \beta(t + 1, j)$$

In addition to determining the probability that O_t was emitted from a given state, the Backward algorithm has several other applications. Although the Forward algorithm is usually used to calculate the probability of a sequence, O, the Backward algorithm can

also be used for this purpose. To calculate the probability of the entire sequence, we use the Backward algorithm to calculate $\beta(t, i)$ recursively, starting with $\beta(2, i)$. The total probability of O is given by:

$$P(O) = \sum_{j=1}^{N} \pi_j e_j(O_1)\beta(2, j).$$

In motif discovery, both the Forward and the Backward algorithm are needed in order to learn parameters from unlabeled data using the Baum Welch procedure, which is a form of Expectation Maximization. We will discuss this next week. The Backward algorithm is also used in another approach to inferring the true state path, called "Posterior decoding".

6.5.3 Posterior decoding

Let us revisit the question of determining the path through an HMM that corresponds to true labeling of the data. In Viterbi decoding, the *most probable path* is considered the best estimate of the true path. An alternative is to use $\hat{Q}$, the sequence of *most probable states*, as an estimate of the true path. This approach is referred to as *posterior decoding* because it is based on the posterior probability of emitting O_t from state i, when the emitted sequence is known. The most probable state at time t is the state that has the highest probability of emitting O_t when all possible state paths are considered:

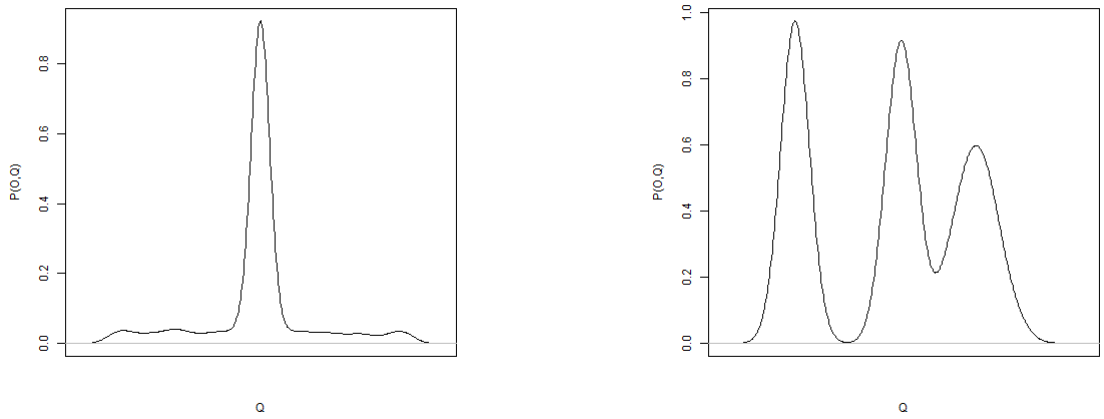
$$\begin{aligned}
\hat{q}_t &= \underset{i}{\operatorname{argmax}}\, P(q_t = E_i, O_t) \\
&= \underset{i}{\operatorname{argmax}}\, P(O_1, O_2, O_3, ...O_t, q_t = E_i) \cdot P(O_{t+1}, O_{t+2}, ...O_T | q_t = E_i) \\
&= \underset{i}{\operatorname{argmax}}\, \alpha(t, i) \cdot \beta(t + 1, i).
\end{aligned}$$

Posterior decoding may give better results in some cases, such as when suboptimal paths are almost as probable as the most probable path. If there is only one state path with high probability (e.g., Fig. 6.7a), then it is likely that Q^* and $\hat{Q}$ will represent the same sequence of states. However, when there are two or more significantly probable state paths (e.g., Fig. 6.7b), each of those paths contributes some information about the classification of each symbol, O_t. Posterior decoding takes advantage of the information encoded in all state paths.

Note that the most probable state for emitting O_t is independent of the most probable state for any other symbol in O. In fact, the sequence of most probable states, $\hat{Q} = \hat{q}_1, \hat{q}_2, \ldots\ \hat{q}_T$ may not correspond to any legitimate path through the model.

6.6 Summary

We started by introducing three recognition questions:



(a) (b)

Figure 6.7: (a) The probability distribution of paths for a given sequence of symbols, O_1, for a hypothetical hidden Markov model, HMM-1. In this hypothetical case, the probability of the most probable path is much greater than the probability of all other paths. (b) The probability distribution of paths for a given sequence of symbols, O_2, for a hypothetical hidden Markov model, HMM-2. In this hypothetical case, there are several paths with relatively high probability. One of these is almost as probable as the most probable path

1. What is the probability that a given sequence, O, was generated by the HMM?
   Example: Is the sequence a transmembrane protein?

2. Given a sequence O, what is the true path? Otherwise stated, we wish to assign labels to an unlabeled sequence.
   Example: Identify the cytosolic, transmembrane, and extracellular regions in the sequence. In this case, we wish to assign the labels E, M, or C to each amino acid residue in the sequence.

3. What is the probability of being in state E_i when O_t is emitted?
   Example: Is a given residue localized to the membrane?

We then discussed several approaches to answering these questions:

- Calculating $P(O|\lambda)$ using the Forward or Backward algorithms
- Inferring the state that emitted O_t using the Forward and Backward algorithms

- Inferring the state path that emitted O using Viterbi or Posterior decoding

These tools can be used to answer biological questions in a variety of ways. For example, one approach to predicting whether O is a transmembrane protein is to calculate $P(O|\lambda_{TM})$, the probability that O was emitted by the transmembrane model. However, the resulting probability can be difficult to interpret. How big must the probability be to convince us that O is in fact a transmembrane sequence? To answer the question, it is useful to construct a model representing a null hypothesis and to calculate $P(O|\lambda_0)$ the probability that O was emitted by this null model. If the resulting likelihood ratio

$$\frac{P(O|\lambda_{TM})}{P(O|\lambda_0)}$$

is much greater than one, then we can infer that O is a transmembrane sequence.

An alternate approach would be to infer the state path that emitted O using the Viterbi or posterior decoding. If the resulting path includes membrane states, then we can conclude that O is a transmembrane sequence. If the entire sequence is labeled with C states, then we conclude that it is not.

6.7 Designing HMMs: Motif discovery and modeling

Position Specific Scoring Matrices capture the distribution of residues observed in each position in a conserved motif, but are not a good model for variable length motifs, recognition of new instances with insertions and deletions, and positional dependencies. Moreover, PSSMs do not lend themselves to precise boundary detection. We turned to Hidden Markov models to address these limitations. HMMs provide a flexible and expressive formalism for modeling conserved sequence motifs. In addition to modeling conserved motifs like the `WEIRD` motif, HMMs can also be used to model biologically distinct regions that are characterized by a change in underlying sequence composition. Examples of these include transmembrane regions, which are enriched for hydrophobic residues, and CpG islands, which have elevated GC content.

There are three major computational tasks associated with conserved motifs found in multiple sequences: Discovery, modeling, and recognition. In the past two weeks, we have discussed the recognition problem: Given an HMM, how do we use it to ask questions about patterns in a new, unlabeled sequences? Here, we consider modeling and discovery. For HMMs, modeling and discovery are closely coupled. There are two major issues to consider: designing the HMM topology and estimating the parameters of the model. A fundamental tradeoff drives HMM design: On the one hand, more complex models, with more parameters, can yield more accurate and biologically realistic models. On the other hand, as the number of parameters increases, so does the amount of data needed to estimate parameters without overfitting.

6.7.1 HMM topology

The topology of an HMM is determined by the set of states, $E_1, \ldots, E_N$, and how they are connected; in other words, we must specify which pairs of states will be connected by edges with non-zero transition probabilities. We could just choose a fully connected graph, but typically this has too many parameters to estimate. Instead, we can exploit biological knowledge. The goal is to choose a topology that limits the number of states and edges, while still being expressive enough to represent the structure of the biological pattern of interest.

Model topology strongly influences the properties of the patterns we will discover. The generative aspect of HMMs provides an improved model for boundary detection: HMMs map symbols to states; changes in state define boundaries. Inter-site dependencies and flexibility in pattern length can also be encoded in the topology of an HMM.

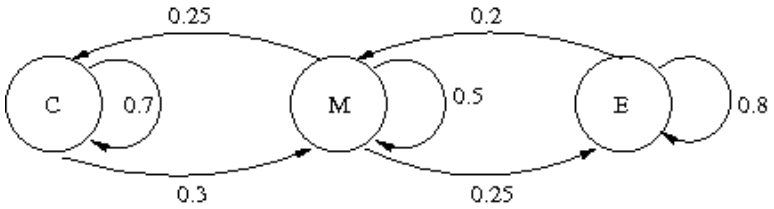


Figure 6.8: A three state Hidden Markov model of a transmembrane sequence.

The transmembrane models we discussed in class illustrate some of the issues to consider. For example, the three state model, Fig. 6.8, is not sufficiently restrictive to emit only transmembrane protein sequences. It can emit sequences that are entirely cytosolic or sequences that pass from the cytosol into the membrane and back to the cytosol, without ever passing through the extracellular region.

We can impose additional order dependencies on sequences generated by the model by modifying the topology. Suppose the goal is to generate sequences that always start and end in the cytosol, with multiple passes through the cell membrane into the extracellular matrix, and back through membrane to the cytosol. By adding additional states, we can obtain a model that only emits sequences that satisfy these conditions (Fig. 6.9). Note that this model has silent *Start* and *End* states, which we have not encountered before. These states do not emit symbols. They serve to ensure that the entry and exit from the model occur in specific states.

The topology of the model also influences the distribution of the lengths of sequences that the model can emit. For example, a simple self loop with probability p (Fig. 6.10) results in sequences with lengths that follow an exponentially decaying (geometric) distribution. The probability that this model will emit a sequence of length l is $(1-p)p^{l-1}$. This does not correspond to the length distribution of real amino acid sequences. More realistic length

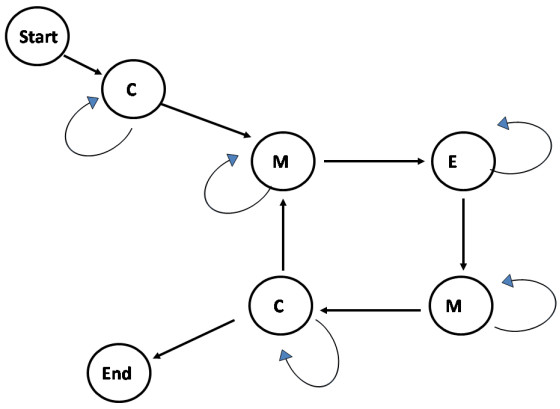


Figure 6.9: An HMM that emits transmembrane sequences that start and end in the cytosol and make at least one pass through the membrane to the extracellular matrix and back through the membrane to the cytosol. Note the use of silent Start and End states ensures that sequences start and end in the cytosol.

distributions can be obtained with more complex topologies. Some of these are described in Durbin, section 3.4.

Finally, we must also choose the alphabet and decide which states will emit which symbols. The larger the alphabet, the greater the number of emission probabilities that must be estimated. The transmembrane models we discussed in class used a two letter alphabet of hydrophobic (`H`) and hydrophilic (`L`) residues to represent sequences, instead of the full 20 letter alphabet for amino acids. This not only gives a simpler representation, it also requires fewer training sequences to learn the parameters since all hydrophobic (resp., hydrophilic) residues contribute to the estimation of a single parameter.

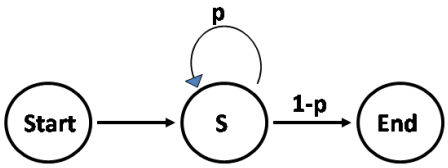


Figure 6.10: An HMM that emits a single symbol, σ, with unit emission probability ($e_s(\sigma) = 1$). The probability that a sequence is emitted by this HMM decreases exponentially with the length of the sequence.

6.7.2 Parameter estimation

Once the states and connectivity have been chosen, the parameters of an HMM are estimated from training data. We are given observed sequences, $O^1, O^2, ..., O^k$, and wish to construct an HMM with parameters, λ, to model these sequences. If the sequences are labeled, the transition and emission probabilities can be estimated easily from the observed transition and emission frequencies. If the sequences are unlabeled, we must first discover the conserved pattern using a machine learning algorithm.

Labeled sequences:

When we are given labeled sequences in which every symbol O_t^d is associated with a state, $q_t^d = E_i$, the parameters can be estimated by tabulating the emission and transition frequencies in the data. Note that inferring parameters from counts in labeled data is a form of maximum likelihood estimation; we are assuming that the emission and transition probabilities that best model the motif of interest are those that maximize the probability of the observed symbols and states in the training data.

The transition probabilities are calculated by tabulating the number of observed state changes in the data:

$$a_{ij} = \frac{\sum_{d=1}^{k} A_{ij}^d}{\sum_{d=1}^{k} \sum_{j\prime} A_{ij\prime}^d},$$

where A_{ij}^d is the number of pairs of adjacent symbols, $O_t^d\ O_{t+1}^d$, that are labeled $E_i E_j$. The emission probabilities are given by

$$e_i(\sigma) = \frac{\sum_{d=1}^{k} \mathcal{E}_i^d(\sigma)}{\sum_{d=1}^{k} \sum_{\alpha \in \Sigma} \mathcal{E}_i^d(\alpha)}$$

where $\mathcal{E}_i^d(\sigma)$ is the number of instances in O^d where the symbol σ is labeled with state E_i. Finally, the initial probability π_i is given by

$$\pi_i = \frac{1}{k} \sum_{d=1}^{k} I^d(i), \tag{6.4}$$

where

$$I^d(i) = \begin{cases} 1, & \text{if } q_1^d = E_i \\ 0, & \text{otherwise} \end{cases} \tag{6.5}$$

is an indicator variable that is equal to one when the first symbol in O^d is labeled with state E_i and zero otherwise.

We may wish to include pseudocounts to account for cases not observed in the training data. Pseudocounts are incorporated into the emission probabilities in the same way that

we used pseudocounts in the definition of the frequency matrix for PSSMs. The probability of emitting σ from state E_i is

$$e_i(\sigma) = \frac{\sum_{d=1}^{k} \mathcal{E}_i^d(\sigma) + b}{\sum_{\alpha \in \Sigma} \left(\sum_{d=1}^{k} e_i^d(\alpha) + b \right)}$$

$$= \frac{\sum_{d=1}^{k} \mathcal{E}_i^d(\sigma) + b}{\left(\sum_{\alpha \in \Sigma} \sum_{d=1}^{k} e_i^d(\alpha) \right) + |\sum| b} \tag{6.6}$$

where b is a pseudocount. In class, we have used $b = 1$ as a pseudocount. There are more sophisticated approaches to selecting a pseudocount. We will not cover them in this course; for those interested in learning more about this on their own, Durbin's discussion of Dirichlet mixtures in Section 11.5 of his book provides a starting point.

We can also use pseudocounts to account for state transitions that are allowed, but not observed in the training data. Let $\mathcal{N}(i)$, the neighborhood of state E_i, be the set of states that can be reached from E_i in a single transition. In other words, $\mathcal{N}(i)$ is the set of states, E_j such that a_{ij} has not been explicitly defined to be zero in the design of the topology. Then,

$$a_{ij} = \frac{\left(\sum_{d=1}^{k} A_{ij}^d \right) + b}{\sum_{j' \in \mathcal{N}(i)} \left((\sum_{d=1}^{k} A_{ij'}^d) + b \right)}$$

$$= \frac{\left(\sum_{d=1}^{k} A_{ij}^d \right) + b}{\left(\sum_{j' \in \mathcal{N}(i)} \sum_{d=1}^{k} A_{ij'}^d \right) + |\mathcal{N}(i)| b} \tag{6.7}$$

As an example, consider the three-state transmembrane model in Fig. 6.8. For this model, seven transition probabilities must be inferred: a_{CC}, a_{CM}, a_{MC}, a_{MM}, a_{ME}, a_{EM}, and a_{EE}. To estimate A_{CC}, for example, given the following labeled sequence

H	H	H	L	L	H	L	H	L	L	H	H	H	H	H
C	C	C	C	C	C	C	C	C	C	M	M	M	M	M

we count the number of CC pairs and normalize by the number of pairs of the form $C*$ where $*$ can be any state. Since there are nine pairs of adjacent symbols labeled CC and one pair labeled CM, for this sequence $a_{CC} = 0.9$ without pseudocounts. With pseudocounts,

$$a_{CC} = \frac{A_{CC} + b}{\sum_{j' \in \mathcal{N}(C)} [A_{Cj'} + b]} ,$$

where $\mathcal{N}(C) = \{C, M\}$. With a pseudocount of $b = 1$, we obtain

$$\begin{aligned} a_{CC} &= \frac{9+1}{(9+1)+(1+1)}\,, \\ &= \frac{10}{12}. \end{aligned}$$

To obtain the emission probabilities from state C, note that C is associated with five hydrophobic and five hydrophilic residues. Thus,

$$\begin{aligned} e_C(\texttt{ H}) &= \frac{\mathcal{E}_C(\texttt{ H}) + b}{\sum_{\alpha \in \{\texttt{ H, L}\}} \mathcal{E}_C(\alpha) + 2b}\,, \\ &= \frac{5+1}{10+2} \end{aligned}$$

or $e_C(\texttt{H}) = 0.5$, again assuming a pseudocount of $b = 1$.

The other transition and emission probabilities are estimated similarly. We estimate the initial probability π_C by counting the number of sequences that begin in the cytosol, and normalizing by the total number of sequences.

Unlabeled sequences:

If the sequences are *unlabeled*, then it is necessary to both discover the motif and learn the model parameters. The motif is discovered automatically, but implicitly, through the process of parameter inference. Once the parameters have inferred, the parameters are used to obtain an explicit model of the motif via Viterbi or posterior decoding.

Parameters are inferred using maximum likelihood estimation. Given sequences $O^1, O^2, \ldots O^k$, we seek $\lambda_l = \{a_{ij}, e_i(\cdot), \pi_i\}$ such that the probability of observing the input sequences is maximized. Stated formally,

$$\begin{aligned} \lambda &= \operatorname*{argmax}_{\lambda_l} \mathcal{L}(\lambda_l) \\ &= \operatorname*{argmax}_{\lambda_l} \sum_d P(O^d|\lambda_l) \\ &= \operatorname*{argmax}_{\lambda_l} \sum_d \sum_Q P(O^d|\lambda_l, Q)\,. \end{aligned}$$

Except for very small problem instances, finding a global maximum is intractable. We would have to calculate $\mathcal{L}(\lambda_l)$ for all possible combinations of parameters, λ_l, to find the parameters that maximize $P(O^1 \ldots O^k|\lambda_l)$. Instead, heuristics are used. These are typically guaranteed to find at least a local maximum. Since these are heuristics, evaluation is usually done empirically by withholding some of the training data for testing, but we will not discuss this further.

The *Baum-Welch* algorithm (Algorithm 3) is used to estimate the parameters of a Hidden Markov model from unlabeled training data. Baum-Welch belongs to a family of algorithms, called Expectation Maximization (EM) algorithms, that work by alternating between estimating the likelihood of the data, given the current estimate of the parameters and re-estimating the parameters from the current likelihoods. Baum-Welch is based on algorithms that we have already encountered: Given labeled data, we can estimate the model parameters using Equations 6.4-6.7, as described in the previous section. Given a model with parameters, we can label unlabeled sequences using Viterbi or posterior decoding.

Informally, Baum-Welch is an iterative algorithm that alternately applies these two procedures. First, an initial estimate of the parameter values is required, for example, based on prior knowledge of the biology underlying the model or on a uniform prior. With this initial estimate, the model is used to label the training data, typically using posterior decoding with the Forward and Backward algorithm. Once the sequences have been labeled, the parameters are re-estimated from the labeled data. The training sequences are then re-labeled using this new estimate of the parameters. The algorithm iterates, alternately labeling the data with the current estimate of the parameter values and then re-estimating the parameters from the labeled data. At each iteration, the likelihood is guaranteed to remain unchanged or increase. This iterative process terminates when the likelihood ceases to improve.

It is instructive to note the similarities and differences between the Baum-Welch algorithm and the Gibbs sampler. Like Baum-Welch, the Gibbs sampler alternates between re-estimating parameters (i.e., a PSSM) from the current estimate of the motif and inferring a new instance of the motif from the updated parameters. However, unlike Baum-Welch, where every training sequence is relabeled at each iteration, in the Gibbs sampler, only one sequence is relabeled at each iteration. A second major difference between the two methods is that the Gibbs sampler is guaranteed to converge to a global optimum given enough time. In contrast, the Baum-Welch algorithm is only guaranteed to find a local optimum.

Given the observed, unlabeled sequences, the parameters are re-estimated in the inner loop of the algorithm. A_{ij} is the expected number of transitions from E_i to E_j. For a given sequence, O^d, probability of transiting from state E_i to E_j at time t is $P(q_t^d = i, q_{t+1}^d = j|O^d, \lambda)$. The number of total transitions from E_i to E_j can be obtained by summing over all time steps, $t = 1$ to T and all input sequences:

$$A_{ij} = \sum_d \sum_t P(q_t^d = i, q_{t+1}^d = j|O^d, \lambda) \tag{6.8}$$

To facilitate the calculation, we again use the trick of converting the conditional probability

Algorithm 1: Baum-Welch

Input:

A set of observed sequences, $O^1, O^2, \ldots O^k$

Initialization:

Select arbitrary model parameters, $\lambda = (a_{ij}, e_i(\cdot), \pi_i)$.

Iteration:

Repeat

{

For each sequence, O^d,

{

Calculate $\alpha(t,i)$, $\beta(t,i)$ and $P(O_d)$ using Forward and Backward algorithms.

$A^d_{ij} = \frac{1}{P(O^d)} \cdot \sum_t \alpha(t,i) a_{ij} e_j(O^d_{t+1}) \beta(t+2,j)$

$\mathcal{E}^d_i(\sigma) = \frac{1}{P(O^d)} \sum_{\{t|O^d_t=\sigma\}} \alpha(t,i)\, \beta(t+1,i).$

}

$a_{ij} = \frac{\sum_d A^d_{ij}}{\sum_d \sum_l A^d_{il}}$

$e_i(\sigma) = \frac{\sum_d \mathcal{E}^d_i(\sigma)}{\sum_d \sum_\alpha \mathcal{E}^d_i(\alpha)}$

$\pi_i = \sum_{d=1}^k \frac{I^d(i)}{P(O^d)}$

$\lambda = (a_{ij}, e_i(\cdot), \pi_i)$.

$\mathcal{L}(\lambda) = \prod_d P(O^d|\lambda)$

}

Until ($\mathcal{L}(\lambda)$ stops changing.)

Algorithm 3: **Baum Welch**

into a joint probability:

$$\begin{aligned} P(q_t^d = i, q_{t+1}^d = j | O^d, \lambda) &= \frac{P(q_t^d = i, q_{t+1}^d = j, O^d)}{P(O^d)} \\ &= \frac{\alpha(t,i) a_{ij} e_j(O_{t+1}^d) \beta(t+2, j)}{P(O^d)} . \end{aligned}$$

The term $\alpha(t,i)$ is the probability that the model has emitted symbols $O_1^d \ldots O_t^d$ and is in state E_i at time t. This probability can be obtained using the Forward algorithm. The term in the denominator, $P(O^d)$, is also calculated with the Forward Algorithm. The terms a_{ij} and $e_j(O_{t+1}^d)$ give the probability of making the transition from E_i to E_j and emitting O_{t+1}. The Backward algorithm yields $\beta_{t+2}(j)$, the probability of emitting the rest of the sequence if the model was in state E_j at time $t+1$. From this we can estimate

$$A_{ij} = \sum_d \frac{\sum_t \alpha(t,i)\, a_{ij}\, e_j(O_{t+1}^d)\, \beta(t+2,j)}{P(O^d)} \tag{6.9}$$

Note that instead of explicitly labeling the data and then counting state transitions as we do with labeled data, the association of symbols and states is implicit in the re-estimation process in the inner loop of the algorithm.

$\mathcal{E}_i(\sigma)$ is the expected number of times that σ is emitted from state E_i:

$$\mathcal{E}_i(\sigma) = \sum_d P(q_t^d = E_i, O_t^d = \sigma | O^d, \lambda) \tag{6.10}$$

$$= \sum_d \frac{\sum_{\{t | O_t^d = \sigma\}} \alpha(t,i)\, \beta(t+1,i)}{P(O^d)} . \tag{6.11}$$

Again, the quantities on the right hand side can be calculated using the Forward and Backward algorithms. Finally, the initial probability π_i is given by

$$\pi_i = \sum_{d=1}^{k} p(q_1^d = S_i | O_d, \lambda) \tag{6.12}$$

$$= \sum_{d=1}^{k} \frac{I^d(i)}{P(O^d)} \tag{6.13}$$

It can be proven that if current estimate is replaced by these new estimates then the likelihood of the data will not decrease (i.e. will increase unless already at a local maximum/critical point). See Durbin, Section 11.6 for discussion of avoiding local maxima and other typical pitfalls with this algorithm.

The *Baum-Welch* algorithm estimates the values of the parameters from training data and, thus, implicitly discovers the motif. *Baum-Welch* does output an explicit description of the motif. To determine the motif explicitly, the Viterbi or posterior decoding are used to label each of the input sequences.

6.8 Profile HMMs

In 1994, Krogh, Haussler[2] and colleagues introduced a flexible HMM topology specifically to model conserved sequence motifs. It captures the propensity to observe specific amino acids or nucleotides at each position in a pattern and allows for insertions and deletions. This topology, called a *Profile HMM*, can be customized for a broad range of conserved motifs by selecting the appropriate length for a given motif and initializing the parameters to capture the specific properties of the motif.

Here, we introduce the features of the Profile HMM model by showing how it could be used to model the `WEIRD` motif based on the following alignment, which has no gaps and no positional dependencies:

```
WEIRD
WEIRD
WEIRE
WEIQH
```

We can recognize the `WEIRD` motif using an HMM with the simple topology shown in Fig. 6.11, where the transitions probabilities are

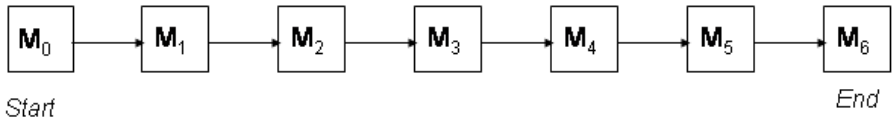


Figure 6.11: A HMM for modeling fixed length motifs. Note that this HMM is equivalent to a Position Specific Scoring Matrix.

$$a_{M_i,M_j} \begin{cases} 1, & \text{if } j = i+1 \\ 0, & \text{otherwise} \end{cases}$$

[2]Krogh, A., Brown, M., Mian, I. S., Sjolander, K., and Haussler, D. (1994). Hidden Markov models in computational biology: Applications to protein modeling. J. Mol. Biol., 235:1501-1531.

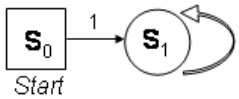


Figure 6.12: An HMM representing the null model. Each symbol, σ, is emitted with probability $p(\sigma)$, the background frequency of σ.

The emission probabilities can be estimated from labeled training sequences. Given an ungapped multiple alignment of k sequences, the emission probabilities are

$$e_{M_i}(\sigma) = \frac{c[\sigma, i] + b}{k + b|\Sigma|}, \tag{6.14}$$

where $c[\sigma, i]$ is the number of σ's at position i in the training motif and b is a pseudocount. The Start and End states (M_0 and M_6 in Fig. 6.11) are silent. Note that when $\mathcal{E}_{M_i}(\sigma) = c[\sigma, i]$, Equation 6.14 is equivalent to the general definition of emission probabilities given in Equation 6.6. Thus, $e_{M_i}(\sigma)$ is equivalent to $q[\sigma, i]$, the frequency matrix that we derived for the PSSM example using pseudocounts.

In order to assess whether a new sequence, O, contains an instance of the `WEIRD` motif, we calculate a likelihood ratio:

$$\frac{P(O|H_A)}{P(O|H_O)}.$$

Our alternate hypothesis, H_A, is that O is an instance of the motif represented by the HMM in Fig. 6.11. In order to obtain a likelihood ratio, we also need a model of the null hypothesis, H_0, that the amino acids in O were sampled from the background distribution. Fig. 6.12 shows a very simple null model. In this model, all transition probabilities are equal to one. The emission probabilities are $e(\sigma) = p(\sigma)$, where $p(\sigma)$ is the background frequency of residue σ.

Given these two models, we can score O by calculating the probability that O was emitted by the Profile HMM in Fig. 6.11 and comparing it with the probability that O was emitted by the background model (Fig. 6.12). For example, if $O = O_1O_2O_3O_4O_5$, then $P(O|H_A)$ is equal to

$$\pi_{M_O} \cdot e_{M_1}(O_1) \cdot a_{M_O M_1} \cdot e_{M_2}(O_2) \cdot a_{M_2 M_3} \cdot e_{M_3}(O_3) \cdot a_{M_3 M_4} \cdot e_{M_4}(O_4) \cdot a_{M_4 M_5} \cdot e_{M_5}(O_5) \cdot a_{M_5 M_6}.$$

Since the initial and transition probabilities are equal to one ($\pi_{M_O} = 1; a_{M_i M_1} = 1, 0 \leq i \leq 5$), this reduces to

$$P(O|H_A) = e_{M_1}(O_1) \cdot e_{M_2}(O_2) \cdot e_{M_3}(O_3) \cdot e_{M_4}(O_4) \cdot e_{M_5}(O_5)$$

or

$$P(O|H_A) = \prod_{t=1}^{5} e_{M_t}(O_t) \ .$$

The probability that O was emitted by the background model is $\prod_t^5 p(O_t)$. The score of sequence O is the log likelihood ratio

$$\sum_{t=1}^{5} \log \frac{e_{M_t}(O_t)}{p(O_t)} \ ,$$

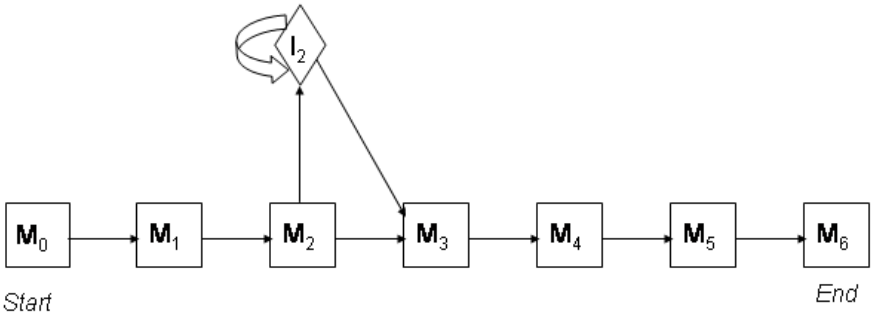
which is equivalent to $\sum_{t=1}^{5} S[O_t, i]$, the score we would have obtained with the PSSM for the `WEIRD` motif.

We now have an HMM that is equivalent to a PSSM for a conserved motif. It can be used to identify motifs of a fixed size, but not cannot handle variations in length. We next extend this model to accommodate insertions and deletions. We can modify the basic HMM to recognize motif instances with insertions, such as $O =$ `WECIRD`, by adding an insertion state between any two match states, M_i and M_{i+1}, as shown in Fig. 6.13a. The emission probabilities for the insertion state are the background frequencies.

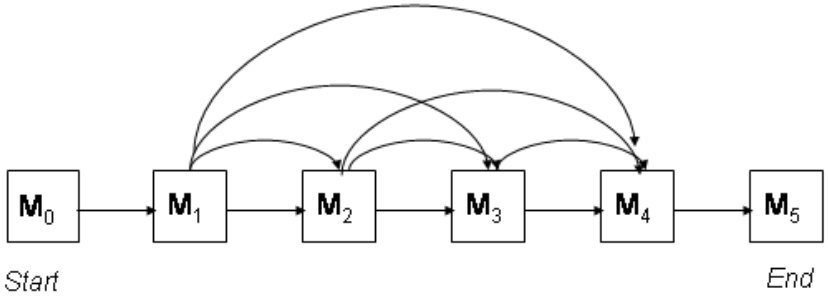
We also wish to recognize motif instances with deletions, e.g., $O =$ `WERD`. One approach to capturing such deletions would be to add edges allowing us to jump over any set of match states. An example of this is shown in Fig. 6.13b. The disadvantage to this approach is that the number of transitions grows rapidly as the number of match states increases. To infer the transition probabilities, we would need a very large set of training data, in which all deletions of all possible sizes were represented. An alternative approach that requires fewer parameters is to model long deletions as sequences of short ones, as seen in the HMM in Fig. 6.13c.

The canonical Profile HMM, shown in Fig. 6.14, combines these features. A Profile HMM has a column containing a Match, an Insertion, and a Deletion state for each position in the conserved pattern. States M_i, I_i, and D_i correspond to the ith position in the pattern. We refer to the number Match states, not including the silent Start and End states, as the *length* of a Profile HMM. A leading insertion state, I_0, allows for patterns that occur in the middle of a longer sequence. If the pattern ends before the end of the sequence, the remaining sequence is emitted by the insertion state I_n, where n is the last position in the pattern.

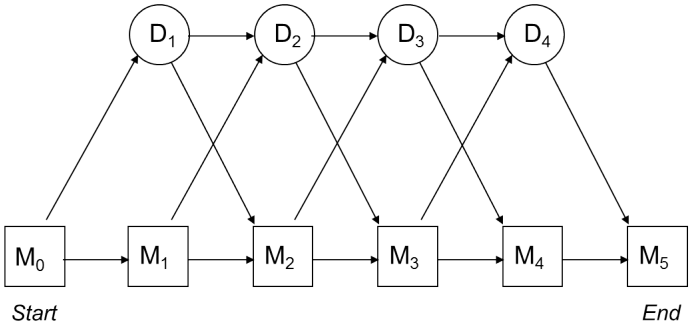
Note that in a Profile HMM there is a path from the Start state, M_0, to the End state, M_{n+1}, that passes only through Insertion and Deletion states. Thus, a Profile HMM can emit a sequence that does not contain an instance of the pattern. Such a sequence would have a low probability, compared with a sequence generated by the Match states.



(a) Insertion model



(b) Deletion model with many transitions



(c) Deletion model with fewer transitions

Figure 6.13: (a) Additional insertion states enable recognition of pattern instances with insertions. This example allows for the insertion of one or more symbols between positions 2 and 3 in the pattern. (b) Adding an arc between every pair of sequences allows for deletions, but the number of transitions grows rapidly with the number of Match states. (c) In this topology, the number of transitions grows linearly with the number of Match states.

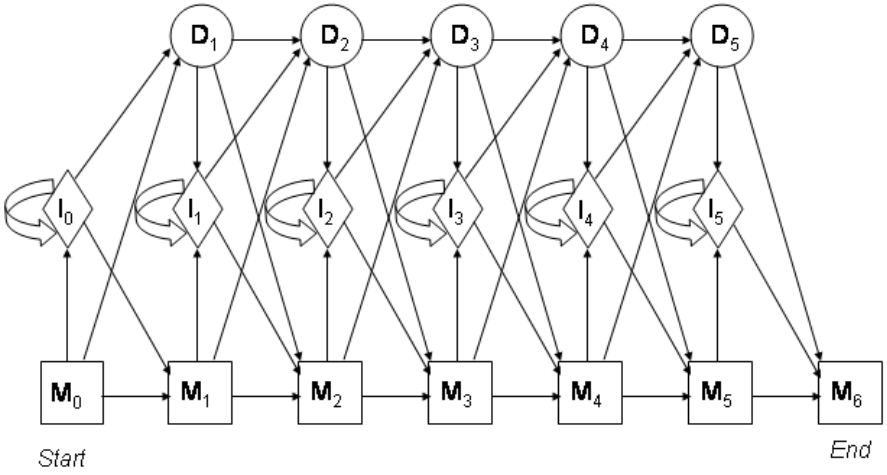


Figure 6.14: A profile HMM of length 5

Paremeter estimation

The emission and transition probabilities of a Profile HMM must be estimated from data. If the sequences are already aligned, then we have labeled training data. In other words, it is possible determine from the alignment which state is associated with each symbol in each sequence. In that case, all we need to do is determine the number of Match states in the Profile HMM, set up the topology, and calculate the parameter values from the labeled data.

Given unlabeled sequences that are known to have a common pattern, we can use the Profile HMM to discover the pattern, infer the values of the parameters, label the data, and construct a multiple sequence alignment. We give an example of each case below.

Constructing a profile HMM for a variable length motif with labeled data: Profile HMMs, like the one show in Fig. 6.14, can be used to model variable length motifs. We illustrate this process with this example:

```
VG--H
V---N
VE--D
IAADN
```

The length of the Profile HMM should correspond to the typical length of the motif. A rule of thumb is to use the average of the length of the training sequences. The lengths of the above sequences are 3, 2, 3, and 5 (before the gaps were inserted), respectively, yielding an average length of 3.25. This suggests that a Profile HMM of length 3 is appropriate for modeling this pattern. Our HMM will have a silent Start state M_0, Match states M_1, M_2, M_3, Insertion states, I_0, I_1, I_2, I_3, Deletion states, D_1, D_2, D_3, and a silent End state, M_4.

The Insertion and Match states emit the 20 amino acids (for protein motifs), or the four nucleotides (for DNA and RNA motifs). Deletion states emit the indel symbol, e.g. "–". For our Profile HMM, the emission probabilities of the Insertion and Deletion states might look like this:

$$e_{D_j}(\sigma) = 0, \quad e_{D_j}(-) = 1$$

$$e_{I_j}(\sigma) = p(\sigma), \ \forall I_j$$

where $p(\sigma)$ is the background probability of residue σ. In order to estimate the parameters for the Match states, we assign labels to the data using the multiple alignment as a guide. Columns in the alignment that have gaps in less than half of the rows correspond to Match states. Those with more gaps in than half of the rows correspond to Insertion states:

V	G	-	-	H
V	-	-	-	N
V	E	-	-	D
I	A	A	D	N
M_1	M_2	I_2	I_2	M_3

This yields the following labeled sequences:

V	G	H		
M_1	M_2	M_3		
V	_	H		
M_1	D_2	M_3		
V	E	D		
M_1	M_2	M_3		
I	A	A	D	N
M_1	M_2	I_2	I_2	M_3

From these labeled sequences, we can estimate the emission and transition probabilities from Equations 6.6 and 6.7. For example, using $b = 1$ as a pseudocount, we obtain

$$e_{M_1}(V) = \frac{3+1}{4+20}.$$

Similarly, the probability of a transition from M_2 to I_2 is

$$a_{M_2 I_2} = \frac{1+1}{(2+1)+(1+1)+(0+1)}\ .$$

The three sums in the denominator correspond to all possible transitions out of state M_2. The first term in each sum in the number of transitions observed in the training data; the second term is a pseudocount. In the training sequences in our example, there are two transitions from M_2 to M_3, one transition from M_2 to I_2 and no transitions from M_2 to D_3. The other emission and transition probabilities are calculated the same way. The model always starts in M_0, so $\pi_{M_j} = 0$, when j>0.

Modeling unlabeled data with a Profile HMM: To discover a pattern in unlabeled data requires the following steps:

1. **Estimating the length:** Given a set of unaligned sequences, where each sequence is an instance of the pattern, we set the length of the HMM (i.e., the number of non-silent Match states) to L, where L is the average sequence length. An example of this type of input would be sequences approximately 50 residues long, where each sequence corresponds to an instance of the Ig domain.

   Given sequences that contain a pattern, but are much longer than the pattern, we must rely on biological knowledge to obtain an initial estimate of the pattern length. The initial estimate of the pattern length can be adjusted later using model surgery (Step 6).
   An example of this type of input would be a set of protein sequences, typically several hundred residues in length, each of which contains an instance of an unknown domain. In this case, you might estimate the length of the pattern to be approximately 100, since that is the length of a typical protein domain.

2. **The topology:** Construct a Profile HMM with $L + 2$ Match states, $L + 1$ Insertion states, and L Deletion states. M_0 and M_{L+1} are silent states corresponding to the Start state and the End state.

3. **Learn parameters:** Guess "good" initial parameters (e.g., $a_{M_i\, M_j} \gg a_{M_i\, I_j}$ and $a_{M_i\, M_j} \gg a_{M_i D_j}$) and train the model using the Baum Welch algorithm.

4. **Determining the motif:** Use the Viterbi algorithm or posterior decoding to infer the state path that emitted each sequence. The Viterbi recurrence can be greatly simplified and expressed in terms of log odds for the special case of Profile HMMs (Durbin, pp 108-110). The log odds formulation avoids underflow and reduces length effects. This was not covered in class. Note the similarity to the dynamic programming algorithm for pairwise alignment.

5. **Multiple Sequence Alignment:** The most likely paths for each sequence obtained from decoding can be used to obtain a multiple alignment of the input sequences. If symbols O_t^c and O_u^d were emitted by same Match state, then align positions t in sequence O^c with position u in sequence O^d. See Ewens and Grant, p 337 - 339 for a discussion and example of multiple sequence alignment using Profile HMMs.

6. **Model surgery:** The topology of the model can be iteratively refined. If more than half of the sequences enter the Delete state, D_j, then remove M_j, D_j, and I_j from the

topology. If more than half of the sequences enter the Insertion state, I_j, then add Match, Insertion and Deletion states between positions j and $j+1$.

7. **Re-estimate the parameters:** If the states change due to model surgery, the parameters must be re-estimated. Label the multiple alignment with the new states and calculate the transition and emission probabilities as described above for labeled data. If the number of states that changed is a substantial fraction of the entire HMM, then you may obtain better results by retraining with the Baum Welch algorithm.

Compared with the exact dynamic programming algorithm for multiple sequence alignment, which runs in exponential time, this approach can align many sequences quickly.

Pattern recognition with profile HMMs: Once you have constructed your Profile HMM, how do you determine whether a new, unlabeled sequence, O, contains the motif?

If you have a model for a suitable null hypothesis, H_0, you can obtain a log odds ratio,

$$\log \frac{P(O|H_A)}{P(O|H_0)} ,$$

using the Forward algorithm to determine the probability of the sequence for each model. Typically, H_A would be represented by a profile HMM and H_0 by a background model such as the one shown in Fig. 6.12. This gives a score, but does not infer the location of the motif.

Alternatively, you can find the most likely path using the Viterbi algorithm or posterior decoding. The location of the motif corresponds to the symbols emitted by the Match states. If no symbols were emitted by Match states, then the motif is not present in O.