HMM Lecture Notes-Part 1

Dannie Durand

Introduction

There are three major tasks associated with local multiple alignment:

Discovery: Given a set of unlabeled sequences, identify an unknown motif that is common to the in put sequences.

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

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

We discussed 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 motifs for which PSSMs are not well suited. In particular, PSSMs cannot

1. model positional dependencies,
2. model variable length motifs,
3. recognize motif instances containing insertions or deletions,
4. detect the boundaries of a motif.

Modeling motifs using Markov chains: pros and cons

Perhaps another formalism would work better for positional dependencies, variable length patterns, and boundary detection. How about Markov chains?

Recall that a finite, discrete-time Markov chain is defined [1] by

- a finite set of *states*, $S_1, S_2, \ldots S_N$;
- a *transition probability*, a_{ij}, which is the probability that the chain will be in state S_j at time $t+1$ given that it was in state S_i at the time, t; and

[1] Note that the notation here differs somewhat from the notation introduced earlier in the semester.

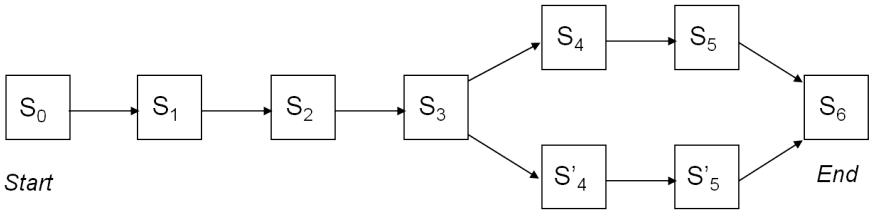
- an *initial state probability distribution*, $\pi = (\pi_1, \pi_2, \ldots, \pi_N)$, where π_j gives the probability of being in state S_j at time $t = 0$.

A Markov chain can also be represented graphically: Each state is represented by a circle or box. State S_i is connected to S_j by an arrow if $a_{i,j} > 0$.

The connectivity or *topology* of a Markov chain can be designed to capture dependencies and variable length motifs. 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`.

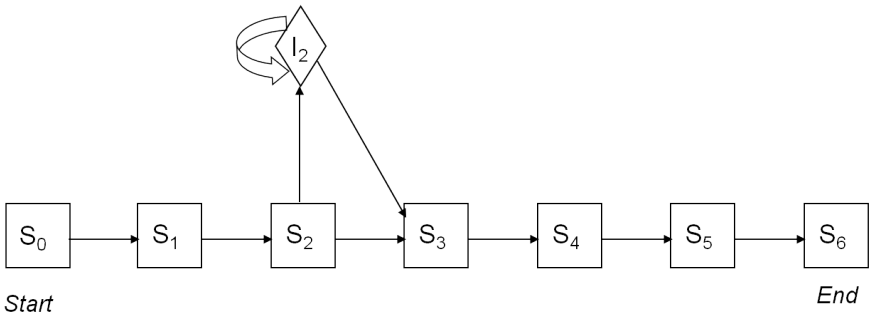
```
WEIRD
WEIRD
WEIQH
WEIRD
WEIQH
WEIQH
```

A PSSM for this motif, however, would give a high score to `WEIRD`, which conforms to the positional dependency, and to `WEIRH`, which does not. In contrast, we can construct a Markov chain to model this pairwise dependency like this:

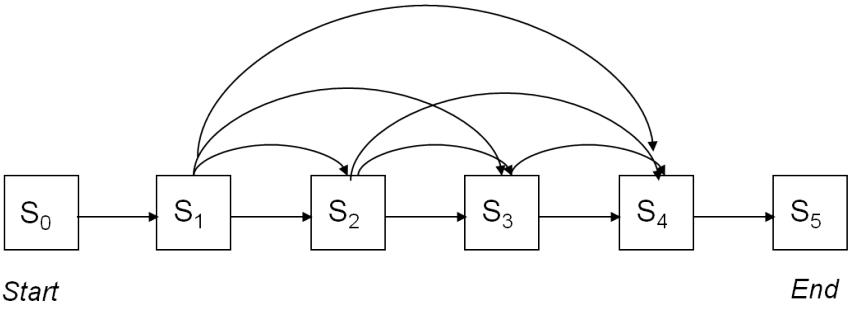


The transition probabilities are chosen so that sequences with `RD` have a high probability in states S_4 and S_5 and sequences with `QH` will have high probability in states S'_4 and S'_5.

To address pattern instances with gaps and and variable length motifs, we can construct a Markov chain to recognize query sequences with insertions, such as $O =$ `WECIRD`, by adding an insertion state:



We can also handle query sequences that have deletions, e.g., $O =$ `WERD`, by modifying the topology of the Markov chain. One approach to capturing such deletions would be to add edges allowing us to jump over any set of states:



What about the fourth problem, *boundary detection*? We are given a sequence and we wish to label each symbol in the sequence according to its class (e.g. introns and exons; alpha helices and beta sheets; genes and non-coding regions). It is difficult to identify sharp boundaries between classes using by scoring windows using a fixed size model, such as a PSSM (Fig. 1).

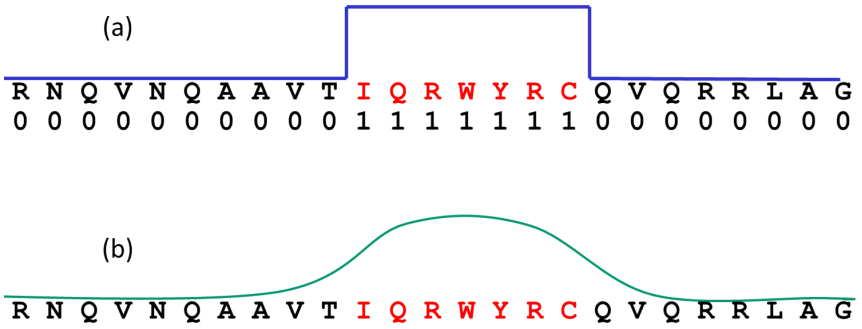


Figure 1: Boundary detection. (a) Perfect boundary detection identifies the exact position of a motif (`IQRWYRC`) in an amino acid sequence. (b) Scoring a sliding window identifies the general neighborhood of the motif, but cannot identify its precise position.

As an example, we considered the problem of recognizing membrane-bound regions in a transmembrane protein, such as the one shown in Fig. 2. We are given an unlabeled amino acid sequence as input. The goal is to label each residue with the cellular location in which it is found.

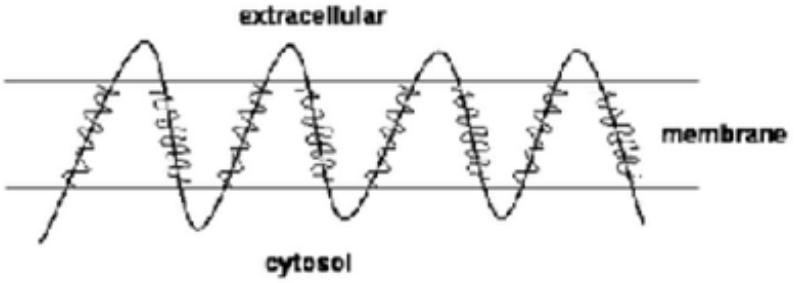


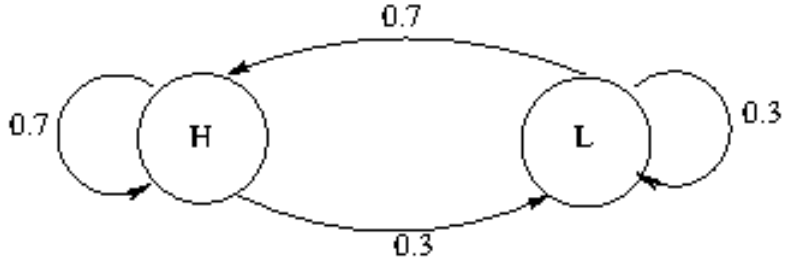
Figure 2:

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. To further simplify the problem, we will assume that the unlabeled sequence has been recoded in a two letter alphabet $\Sigma = \{H, L\}$, in which each amino acid is replaced with an H if the residue is hydrophobic and an L if it is hydrophilic.

Initially, we considered a simpler problem: Supposing we are given a sequence fragment that is either a transmembrane (TM) region or an extracellular/cytosolic region (E/C), can we determine which it is?

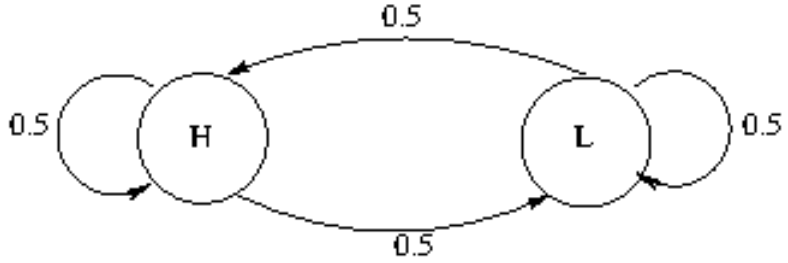
To do this, we constructed a Markov model of transmembrane sequences, in which hydrophobic residues are preferred. The initial state probabilities are $\pi_H = 0.7$ and $\pi_L = 0.3$.

Transmembrane model:



We compared the probability of a sequence with respect to this model with the probability of the same sequence with respect to a model of extracellular and cytosolic sequences in which hydrophobic and hydrophilic residues occur with equal frequencies ($\pi_H = \pi_L = 0.5$).

Extracellular/cytosol model:



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

$$\frac{P(O|TM)}{P(O|EC)}, \tag{1}$$

where the probability of O with respect to a Markov model is $\pi_{O_1} \prod_2^T a_{O_{i-1}O_i}$. If the likelihood ratio is much greater than one, 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 $P(\texttt{HHLHH}|EC) = 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5 \approx 0.031$. 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.

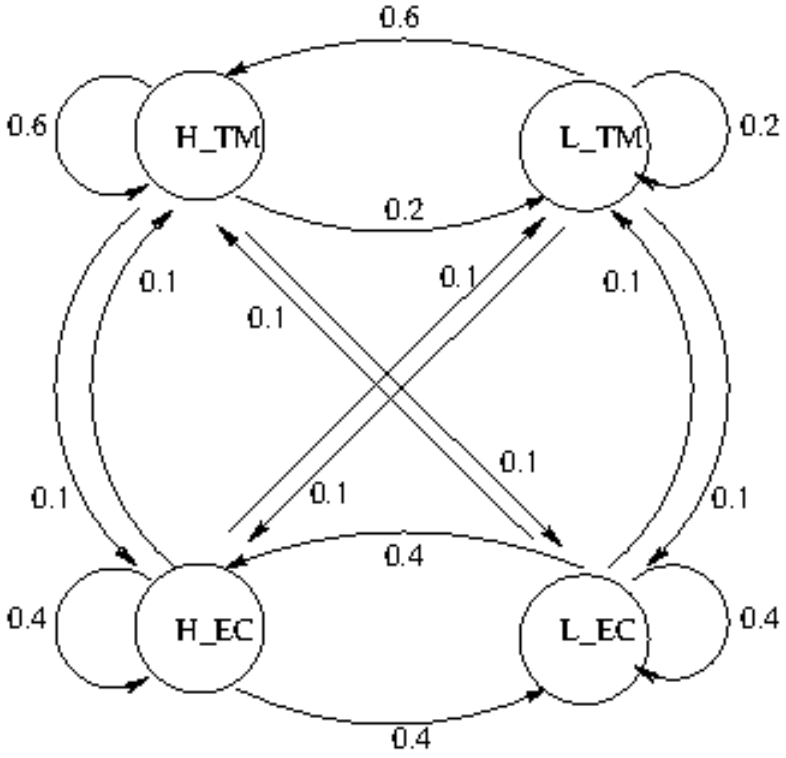
Note that the likelihood of a sequence is sensitive to the length of the sequence; the longer the sequence, the lower its probability. Intuitively, this makes sense since the number of sequences of length T grows exponentially with T. As a result, $P(\texttt{HHHHHHHHHH}|TM) = 0.04 < P(\texttt{HHLHH}|TM) = 0.07$, even though `HHHHHHHHHH` looks more like a transmembrane segment than `HHLHH`. In contrast, the likelihood ratio in Equation 1 is not sensitive to the length of O because $P(O|TM)$ is normalized

by $P(O|EC)$. The likelihood ratio for `HHHHHHHHH` is roughly 20.7, which is much greater than 2.3, consistent with our expectations.

The above models 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.

A four-state transmembrane HMM:



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 is not a standard Markov chain. 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. This four-state model is a *Hidden Markov model*. 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.

Hidden Markov Models

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

1. N states $S_1..S_N$
2. An alphabet, $\Sigma = \{\sigma_1, \sigma_2 \dots \sigma_M\}$
3. Transition probability matrix a_{ij}
4. Emission probabilities: $e_i(\sigma)$ is the probability that state S_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 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 $Q = q_1, q_2, q_3, ...q_T$ (the "state path"). 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. 3. 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.

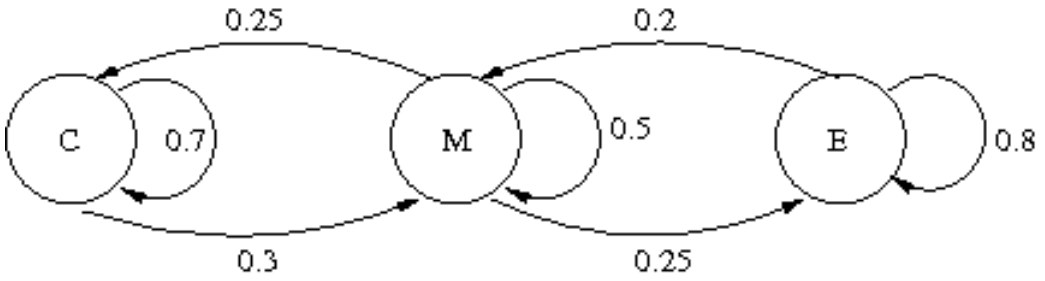


Figure 3: Three-state transmembrane HMM

The parameters for this model are

S_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

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

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. 4 shows a cartoon of the probability distribution of sequence, state pairs 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, $P(O^d, Q^b)$, that the HMM will visit the states on path Q^b and emit sequence O^d. In the three-state TM model example,

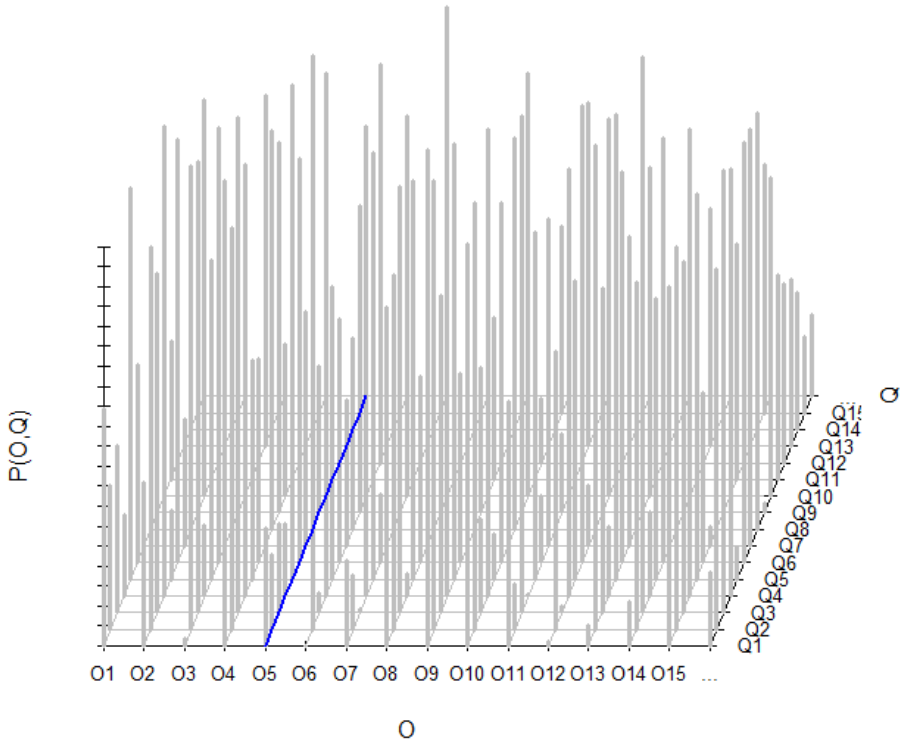


Figure 4: The joint probability $P(O^d, Q^b)$ for every sequence O^d and state path Q^b. The volume under this curve is one.

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:

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

Fig. 5 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. 4 sums to one:

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

Using HMM's for recognition

In this lecture, we focus on motif recognition using HMM's. We will discuss parameter estimation, motif discovery, and modeling using HMM's in future lectures. 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 S_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.

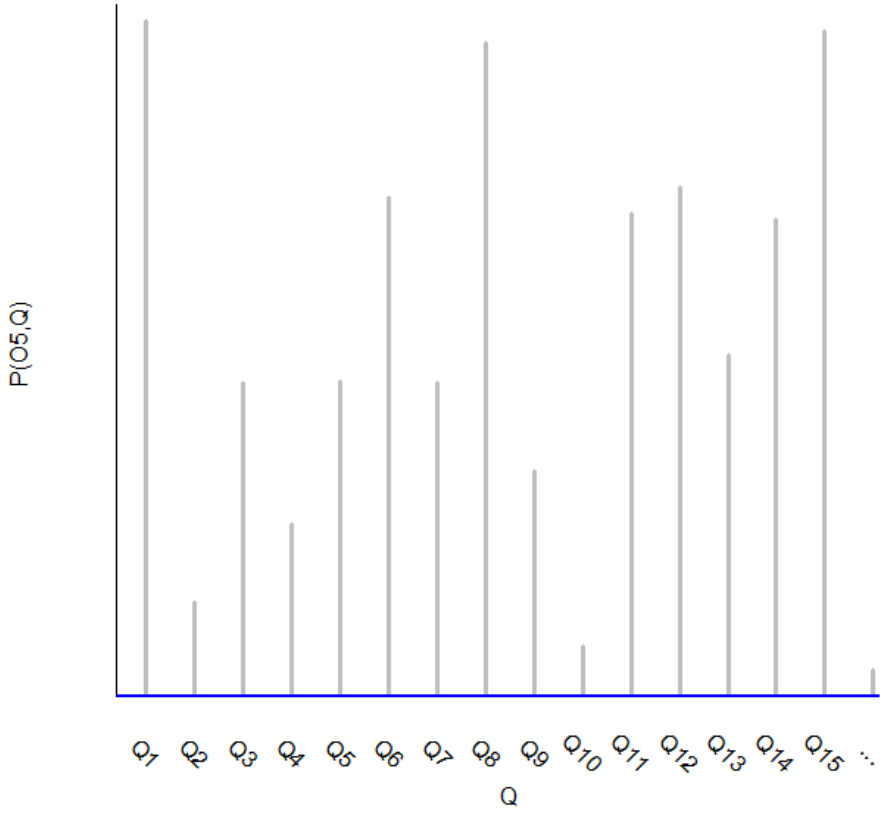


Figure 5: The probability of sequence $O = O_5$ for every state path $Q^1, Q^2, Q^3 \ldots$. This curve corresponds to the intersection of the probability distribution in Fig. 4 and the vertical plane at $O = O_5$ (shown as a blue line in Fig. 4). The area under this curve is $P(O_5)$, the probability of O_5. The maximum point on the curve is the most probable path, $Q^* = \text{argmax}_Q P(Q|O)$ is the highest point on this curve.

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:

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

We could calculate this quantity by enumerating all paths, Q^b, and calculating $P(O, Q^b)$ 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 S_i. We designate this quantity

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

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 S_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 we introduced last week, shown in Fig. 3. 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.

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.

Viterbi decoding

Viterbi decoding is based on the assumption that the *most probable path*, $Q^* = \text{argmax}_Q P(Q|O)$, is a good estimation of the sequence of states that generated the observed sequence O.[2] In practice, we maximize the joint probability $P(Q, O)$, rather than the conditional $P(Q|O)$, but this will still give us the most probable path because the path that maximizes $P(Q, O)$ also maximizes $P(Q|O)$. To see this, note that

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

Since $P(O)$ is independent of Q,

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

As in the case of the Forward algorithm, the brute approach of enumerating all paths and calculating $P(Q|O)$ for each one is intractable, because the number of state paths grows as $O(N^T)$.

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

Instead, we calculate $\text{argmax}_Q P(Q, O)$ 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 S_i. We calculate $\delta(t, i)$ as follows:

Algorithm: Viterbi

Initialization:

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

Recursion:

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

Final:

$$P(Q^*) = \max_{1 \leq j \leq N} \delta(T, j).$$

At each step in the recursion, we save $j^*(t)$, 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. 3 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.

The probability that state S_i emitted O.

The third question

3. What is the probability of being in state S_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 S_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 S_i and (2) the total probability emitting $O_{t+1} \dots O_T$ over all paths, given that the model was in state S_i at time t:

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

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 i:

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

Substituting α and β for the first and second terms in Equation 2, we obtain the following expression for the probability of emitting O_t from state S_i

$$P(q_t = S_i, O_t) = \alpha(t, i) \cdot \beta(t + 1, i). \tag{3}$$

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:

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

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".

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:

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

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, left), 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, right), 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.

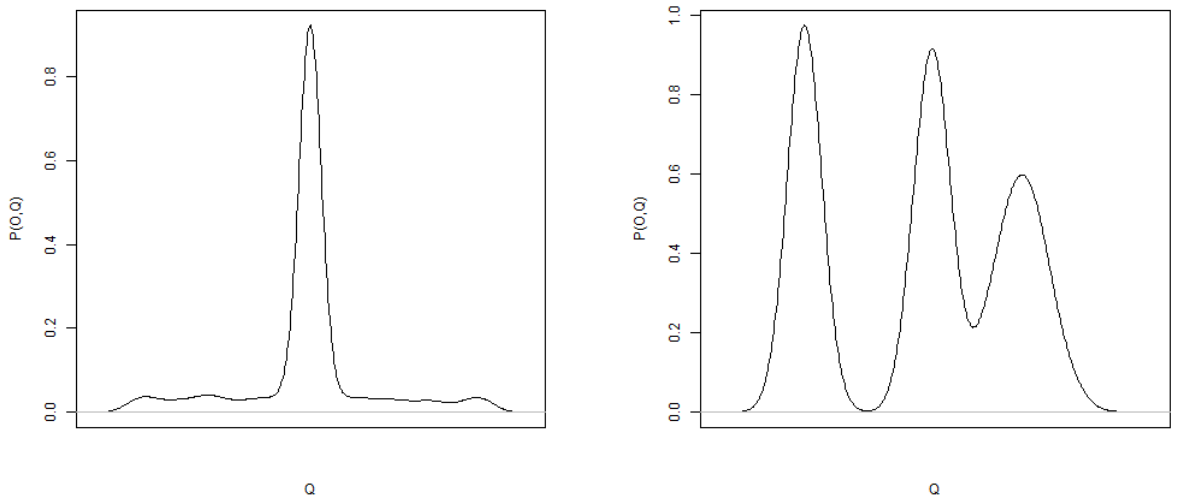


Figure 6: (Left) The probability distribution of paths for a given sequence of symbols, O_1, for a hypothetical hidden Markov model, HMM-1. In this hypothetical case, the probability of the most probable path is much greater than the probability of all other paths. (Right) The probability distribution 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

Summary

We started by introducing three recognition questions:

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 S_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.