COS 597c: Topics in Computational Molecular Biology
Lecture 11: October 25, 1999
Lecturer: Mona Singh
Scribes: Jordan Parker[1]

Profile Hidden Markov Models

In the previous lecture, we began our discussion of profiles, and today we will talk about how to use hidden Markov models to build profiles. One of the advantages of using hidden Markov models for profile analysis is that they provide a better method for dealing with gaps found in protein families. We will begin by an introduction of hidden Markov models. HMMs have also been used in many other areas of computational biology, including for gene finding as well as construction of genetic linkage maps and of physical maps.

1 Introduction to Hidden Markov Models

A hidden Markov model is defined by specifying five things:

Q = the set of states = $\{q_1, q_2, ..., q_n\}$

V = the output alphabet = $\{v_1, v_2, ..., v_m\}$

$\pi(i)$ = probability of being in state q_i at time $t = 0$ (i.e., in initial states)

A = transition probabilities = $\{a_{ij}\}$,
where $a_{ij} = Pr[\text{entering state } q_j \text{ at time } t+1 \mid \text{in state } q_i \text{ at time } t]$. Note that the probability of going from state i to state j does not depend on the previous states at earlier times; this is the Markov property.

B = output probabilites = $\{b_j(k)\}$,
where $b_j(k) = Pr[\text{producing } v_k \text{ at time } t \mid \text{in state } q_j \text{ at time } t]$

[1]Portions of the notes are adapted from lecture notes originally scribed by Xuxia Kuang in Fall 1998. Last edited October 2002.

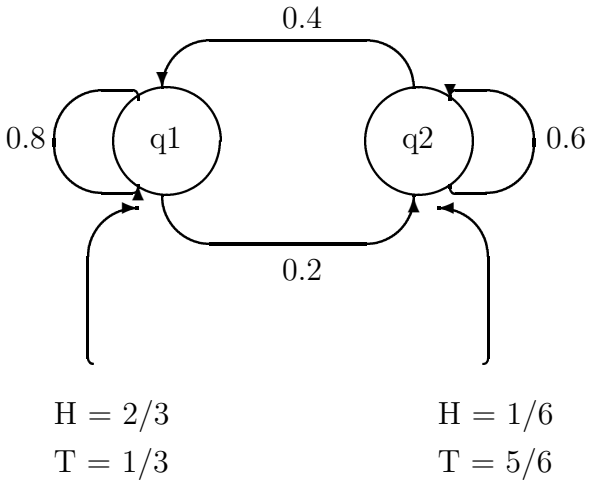

Figure 1: A Hidden Markov Model

As an example, let's say that we have two biased coins, which we are flipping, and an observer is seeing the results of our coin flips (not which coin we're flipping). In fact, suppose that what we are actually doing can be described by Figure 1. Here, the states of the HMM are q_1 and q_2 (the coins), the output alphabet is $\{H, T\}$, and the transition and output probabilities are as labelled. If we let $\pi(q_1) = 1$ and $\pi(q_2) = 0$ then the following is a example of a possible transition sequence and output sequence for the HMM in Figure 1:

$$\begin{array}{ccccccccccccc} & & 0.8 & & 0.8 & & 0.2 & & 0.6 & & 0.4 & & 0.8 & \\ q_1 & \longrightarrow & q_1 & \longrightarrow & q_1 & \longrightarrow & q_2 & \longrightarrow & q_2 & \longrightarrow & q_1 & \longrightarrow & q_1 \\ \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow \\ \mathrm{H} & & \mathrm{H} & & \mathrm{T} & & \mathrm{T} & & \mathrm{T} & & \mathrm{T} & & \mathrm{H} \end{array}$$

We can easily calculate probabilities for the following events.

1. The probability of the above state transition sequence:

$$Pr[q_1q_1q_1q_2q_2q_1q_1] = \pi(q_1)a_{11}a_{11}a_{12}a_{22}a_{21}a_{11} \approx 0.025$$

2. The probability of the above output sequence given the above transition sequence:

$$Pr[(HHTTTTH)|(q_1q_1q_1q_2q_2q_1q_1)] = \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{1}{3} \cdot \frac{5}{6} \cdot \frac{5}{6} \cdot \frac{1}{3} \cdot \frac{2}{3} \approx 0.23$$

3. The probability of the above output sequence and the above transition sequence:

$$Pr[(HHTTTTH) \wedge (q_1q_1q_1q_2q_2q_1q_1)] \approx (0.025) \cdot (0.023) \approx 5.7 \times 10^{-4}$$

While we just considered the case where we knew both the results of the coin flips, and which coin was being flipped, in general, we consider the case where we do *not* know which coins are being flipped. That is, while we know the underlying model is as described in Figure 1, and we observe the output symbol sequence, the state sequence is "hidden" from us. In this case, we can also figure out the answers to the following questions:

1. What is the probability of the observed data $O_1, O_2, \ldots O_T$ given the model? That is, calculate $\Pr(O_1, O_2, \ldots O_T|\text{model})$.

2. At each time step, what state is most likely? It is important to note that the sequence of states computed by this criterion might be impossible. Thus more often we are interested in what single sequence of states has the largest probability. That is, find the state sequence $q_1, q_2, \ldots, q_T$ such that $\Pr(q_1, q_2, \ldots, q_T|O_1, O_2, \ldots O_T,$ model) is maximized.

3. Given some data, how do we "learn" a good hidden Markov model to describe the data? That is, given the topology of a HMM, and observed data, how do we find the model which maximizes Pr(observations|model)?

To answer the first two questions, we can use dynamic programming techniques. To compute the answer to the last question, we can use the Baum-Welch method. We will briefly discuss the general methods to solve of these questions (see the appendix). More details can be found in Rabiner and Juang's tutorial paper on hidden Markov models [1]. We will discuss in more detail the methods used to answer these questions in the context of HMM profiles.

2 Building HMM-Profiles

How does the above relate to profiles? Let's see how we can use HMMs to model them. In the last lecture, we built a profile for the alignment:

```
LEVK
LDIR
LEIK
LDVE
```

Ignoring the "background" frequencies for now, a profile for this alignment can be viewed as trivial HMMs with one "match" state for each column, where consecutive match states are separated by transitions of probability 1. We also need to define output probabilities for each of these match states; these come from the probability of observing a particular amino acid in the corresponding column (i.e., these are identical to the probabilities we compute per column for the original profile method). We also introduce "dummy" begin and end states which emit no output symbols. This trivial HMM is illustrated in Figure 2.

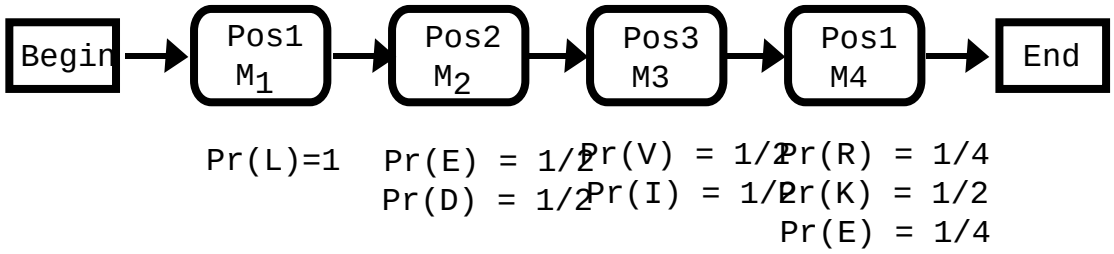

Figure 2: A profile-HMM

Now let's extend our model to handle insertions. Insertions are portions of sequences that do not match anything in the above model. We will introduce insert states I_j, which will model inserts after j^{th} column in our alignment. See Figure 3. Typically, the output probabilities for insert states are set equal to the background probabilities. Note that we can have different probabilities for entering different insert states, and this models the fact that insertions may be less well-tolerated in certain portions of the alignment. Also, for any particular insert state, we may have different transition probabilities for entering it for the first time vs. staying in the insert state; this models *affine* gap penalities.

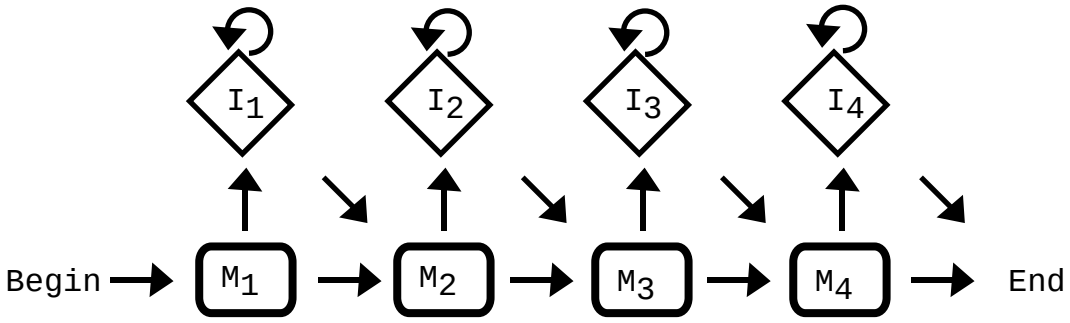

Figure 3: Insert States

One could model deletions is as in Figure 4. However, arbitrarily long gaps introduces lots of transitions in the model. Instead, we will introduce "delete" states that do not emit any symbols (see Figure 5). The structure of the complete HMM with both inserts and deletes is shown in Figure 6.

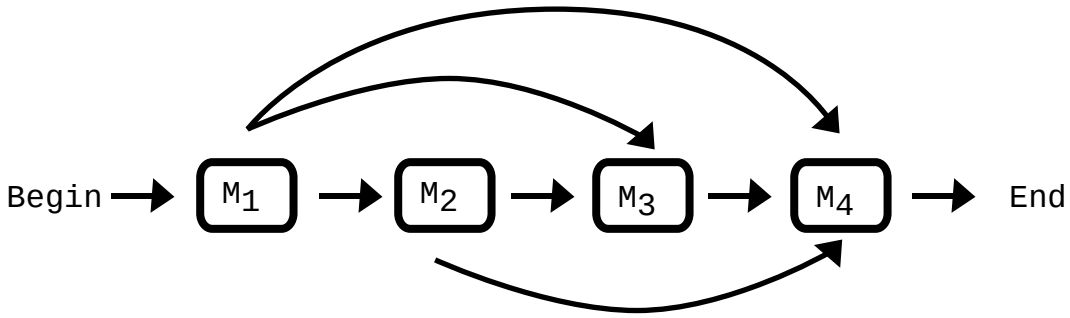

Figure 4: Possible deletions

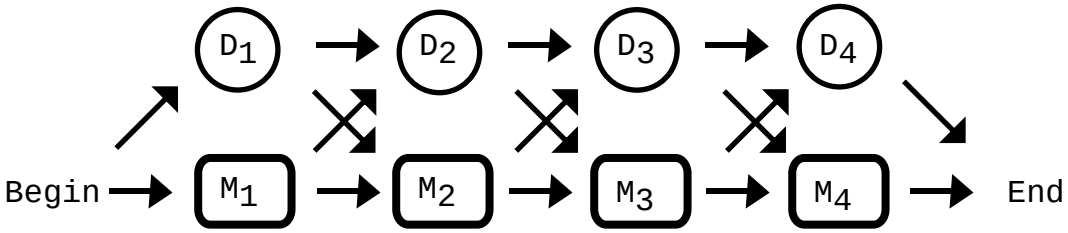

Figure 5: Deletions

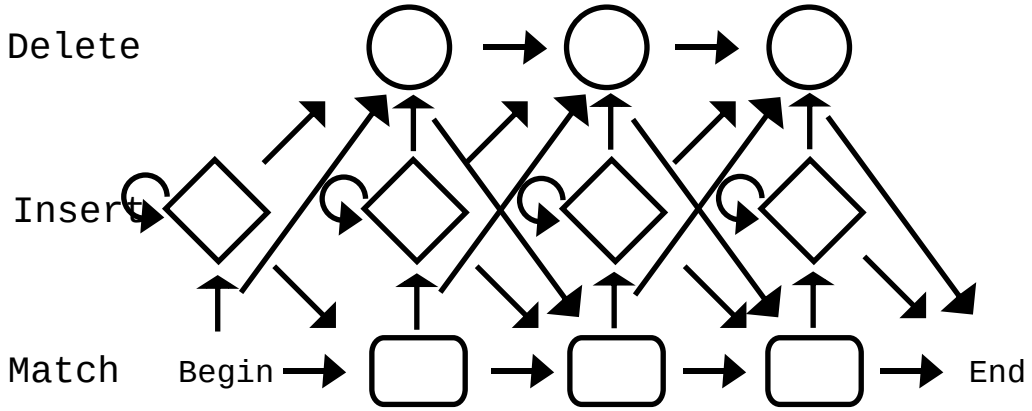

Figure 6: The complete HMM formation

We have just given the overall topology of a profile-HMM, but we still need to decide how many states our HMM has, what the transition probabilities are, etc. As an example, let's build an HMM profile for following multiple sequence alignment:

```
VGA--HAGEY
V----NVDEV
VEA--DVAGH
VKG------D
VYS--TYETS
FNA--NIPKH
IAGADNGAGY
```

How do we pick the length of the HMM (i.e., how many match states do we have in the profile?). One heuristic is to only include those columns that have amino acids

in at least half of the sequences. For example, in the above alignment, there would be match states for each column except for the fourth and fifth columns.

How do we pick output probabilities for match states? We will use the same technique as was used with building our non-HMM profiles. For example, for the first match state we have:

$$\begin{aligned} b_{m_1 V} &= \tfrac{5}{7} \\ b_{m_1 F} &= \tfrac{1}{7} \\ b_{m_1 I} &= \tfrac{1}{7} \end{aligned}$$

In reality, we must connect for zero frequency case. As mentioned in the last lecture, a common way of doing this by adding a small amount to every frequency (e.g., the "add-one" rule). Using the add-one rule, our probabilities would be:

$$\begin{aligned} b_{m_1 V} &= \tfrac{5+1}{7+20} = \tfrac{6}{27} \\ b_{m_1 F} &= \tfrac{2}{27} \\ b_{m_1 I} &= \tfrac{2}{27} \\ \text{all else} &= \tfrac{1}{27} \end{aligned}$$

How do we pick transition probabilities? We let the transition probability of going from state k to state l a_{kl} be equal to:

$$\frac{\text{number of times go from state } k \text{ to state } l}{\text{the number of times go from state } k \text{ to any other state}}$$

So, to calculate the probability of transition from match state 1 to match state 2, we count the number of times we get a match (=6) in the second column, as well as the number of gaps (=1). (Note, using the initial alignment and our model, we only have insertions after the third match state.)

$$\begin{aligned} a_{M_1 M_2} &= \tfrac{6}{7} \\ a_{M_1 D_1} &= \tfrac{1}{7} \\ a_{M_1 I_1} &= \tfrac{0}{7} \end{aligned}$$

Again, using the add-one rule, we correct our probabilites to be:

$$a_{M_1M_2} = \frac{6+1}{7+3} = \frac{7}{10}$$
$$a_{M_1D_1} = \frac{2}{10}$$
$$a_{M_1I_1} = \frac{1}{10}$$

The rest of the parameters are calculated analagously.

In the next lecture, we will address the question of how we use a given profile model of a protein family to figure out whether a new protein sequence is a member of that family. We will also show how to find the most likely alignment of this protein sequence to the family.

We can actually build profiles from *unaligned* sequences using the Baum-Welch procedure, but we won't have time to go over this. Note, however, the topology of the HMM is fixed before learning. We might not always know the number of relevant positions in the family (i.e., the number of conserved positions). One heuristic to get around this is as follows. First, guess the number of states by choosing an initial length. Then, after learning, if more than half of the paths of sequences choose the *delete* state at a particular position, do "model surgery" and remove the whole position. If more than half of the paths choose an *insert* state in a position, then add insert states at this position, with the number of new states equal to the average number of inserts. In practice, methods for learning HMMs are prone to getting caught in local minima, and, so it is best to start with a good initial guess (in our case, an aligment), and to start with different starting parameters. In fact, mostly HMM-profiles are built just from alignments, similar to the way we have just described via our example (i.e., without trying to learn the parameters).

References

[1] L. R. Rabiner and B. H. Juang. "An Introduction to Hidden Markov Models." *IEEE ASSP Magazine*, 3(1):4-16, January 1986.

[2] R. Durbin, S. Eddy, A. Krogh and G. Mitchison. *Biological Sequence Analysis.* Cambridge University Press, 1998.

Appendix

Given an HMM, we can figure out the answers to the following questions:

1. What is the probability of the observed data $O_1, O_2, \ldots O_T$ given the model? That is, calculate $\Pr(O_1, O_2, \ldots O_T|\text{model})$.
2. At each time step, what state is most likely? It is important to note that the sequence of states computed by this criterion might be impossible. Thus more often we are interested in what single sequence of states has the largest probability. That is, find the state sequence $q_1, q_2, \ldots, q_T$ such that $\Pr(q_1, q_2, \ldots, q_T|O_1, O_2, \ldots O_T,$ model) is maximized.
3. Given some data, how do we "learn" a good hidden Markov model to describe the data? That is, given the topology of a HMM, and observed data, how do we find the model which maximizes Pr(observations|model)?

To answer the first question we must calculate: Pr[observed data | model]
= $\sum (Pr[\text{observed data} \mid \text{state sequence, model}] \times Pr[\text{state sequence} \mid \text{model}])$
where we are summing over all possible state sequences. That is, we must calculate:

$$Pr[O_1 O_2 ... O_T|\lambda] = \sum_{q_1 ... q_T} Pr[O_1 O_2 ... O_T|q_1 ... q_T, \lambda] \times Pr[q_1 ... q_T|\lambda]$$

where λ is the model.

However, there are exponentially many possible state sequences. To solve the problem in polynomial time, we can use dynamic programming. In particular, the algorithm used is known as the *Forward Algorithm*.

Our dynamic programming recurrence is as follows. Let
$\alpha_t(i) = Pr[O_1, O_2 ... O_t, q_t = S_i|\lambda]$. That is, $\alpha_t(i)$ is the probability of observations O_1,...,O_t and being in state i at time t.

1. Initially we have:
$$\alpha_1(i) = \pi(i) b_i(O_1), \text{ for } 1 \leq i \leq n$$
2. Induction step:
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{n} \alpha_t(i) a_{ij} \right] b_j(O_{t+1})$$

3. Finally:

$$Pr(O|\lambda) = \sum_{i=1}^{n} \alpha_T(i)$$

Hence we can calculate $Pr[O_1O_2...O_T]$ in $O(N^2T)$ time.

To answer the second question we must find a state sequence q_1, $q_2 \ldots, q_T$, such that $Pr[q_1q_2...q_T|O_1O_2...O_T, \text{model}]$ is maximized. We can also use dynamic programming techniques, specifically the *Viterbi Algorithm.*

Our dynamic programming recurrence is as follows.
Let $\delta_t(i) = \max_{q_1 \ldots q_{t-1}} Pr[q_1q_2...q_t = i, O_1O_2...O_t|\lambda]$. That is, $\delta_t(i)$ is the highest probability along a single path at time t which accounts for the first t observations and ends in state S_i. We will also use $\phi_t(i)$ to help us in keeping track of the actual path; that is, $\phi_t(i)$ tells us which state at time $t-1$ "led" us to the highest probability $\delta_t(i)$ at time t.

1. Initially we have:

$$\delta_1(i) = \pi(i)b_i(O_1), \text{ for } 1 \leq i \leq n$$

$$\phi_1(i) = 0$$

2. Induction:

$$\delta_t(j) = \left(\max_{1 \leq i \leq n} [\delta_{t-1}(i)a_{ij}] \right) b_j(O_t)$$

$$\phi_t(j) = \text{argmax}_{1 \leq i \leq n} [\delta_{t-1}(i)a_{ij}]$$

3. Finally, we find the states $q_1^*, \ldots, q_T^*$ by backtracking:

$$q_T^* = \text{argmax}_{1 \leq i \leq n} \delta_T(i)$$

$$q_t^* = \phi_{t+1}(q_{t+1}^*)$$

To answer the third question, given the topology of a HMM and observed data, we must find the model that maximizes $Pr[O_1O_2 \ldots O_T|\lambda]$.

There is no optimal analytical way of doing this. However, we can choose parameters using Baum-Welch, an iterative procedure which finds a local maximum. We will not go over this algorithm, but the idea as follows. To re-estimate the transition probabilities a_{ij}, for example, we calculate:

$$\frac{\text{expected \# of transitions from } q_i \text{ to } q_j}{\text{expected \# of transitions from } q_i}$$

Other parameters are updated in similar fashion.