HMM Lecture Notes

October 4th

Dannie Durand and Rose Hoberman

Outline

1. Review
2. Modeling motifs using Markov chains: pros and cons
3. Hidden Markov Models
4. The Viterbi algorithm

1 Review

In the last few lectures, we have focussed on three problems related to local multiple sequence alignments:

- Discovery
- Modeling
- Recognition

We discussed Position Specific Scoring Matrices (PSSMs) for modeling and recognition of ungapped patterns or *motifs* and the Gibbs sampler for discoveriing such motifs in unlabeled data. PSSMs work well for fixed length patterns in which the sites are more or less independent. However, we saw that there are other kinds of motifs for which PSSMs are not well suited. PSSMs cannot

1. model positional dependencies,
2. recognize pattern instances containing insertions or deletions,
3. model variable length patterns
4. detect boundaries.

2 Modeling motifs using Markov chains: pros and cons

Perhaps another formalism would work better for positional dependencies, gaps, and boundaries. How about Markov chains?

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

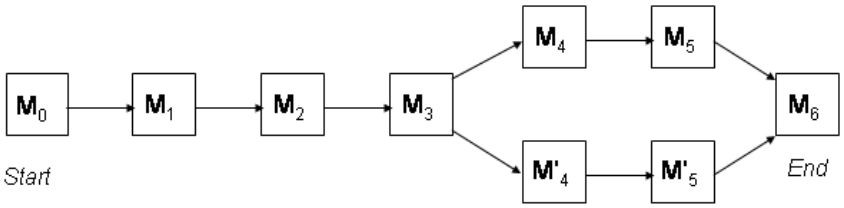
- a finite set of *states*, $S_1, S_2, \ldots S_N$;
- a *transition probability:* $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$, the probability that the chain will be in state S_j at time $t+1$ given that it was in state S_i at the previous time step, t; and
- an *initial state probability distribution*, $\pi_0 = (p(q_0 = S_1), p(q_0 = S_2), \ldots p(q_0 = S_N))$.

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 our 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`.

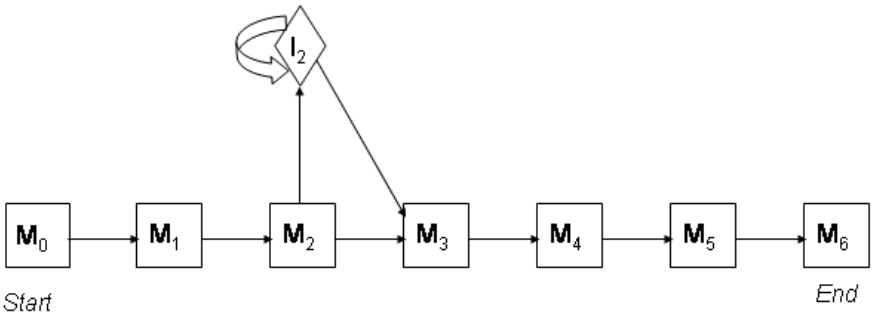
```
WEIRD
WEIRD
WEIQH
WEIRD
WEIQH
WEIQH
```

A PSSM for this motif, however, would give the sequences `WEIRD` and `WEIRH` equally good scores. We can construct a Markov chain to model this pairwise dependency like this:

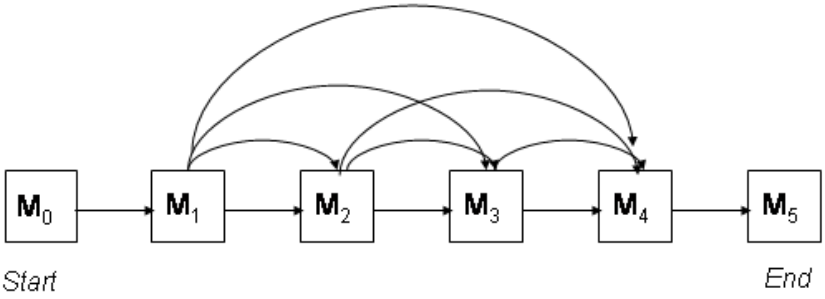


[1] Note that the notation here differs somewhat from the notation introduced earlier in the semester.

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`:



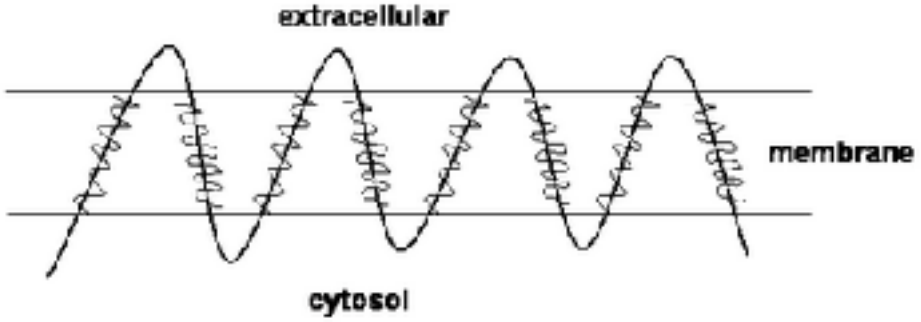
by adding an insertion state. We can also handle query sequences that have deletions, e.g., $O =$ `WERD`, by modifying the topology of the Markov chain. One approach to capturing such deletions would be to add edges allowing us to jump over any set of states:



Boundary detection:

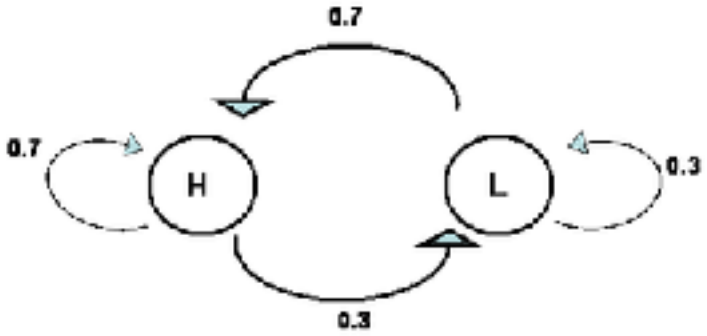
What about the fourth problem, *boundary detection*? Last week, we introduced a boundary detection problem: We are given a sequence and we wish to label each symbol in the sequence according to its class (e.g. introns and exons; alpha helices and beta sheets; genes and non-coding regions). It is difficult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

As an example, we considered the problem of recognizing transmembrane regions in a transmembrane protein. 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?

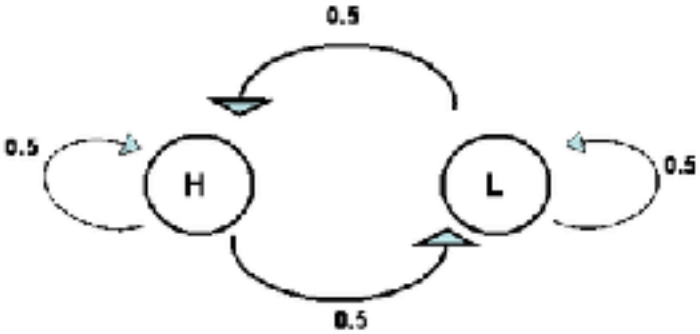


To do this we constructed two Markov models, a TM model and an E/C model. In these models, each amino acid is encoded as either hydrophobic (H) or hydrophilic (L).

Transmembrane model:



Extracellular/cytosol model:



Parameter estimation: The transition probabilities in such a model are determined from known examples of transmembrane sequences. Given labeled sequences (transmembrane or not transmembrane), we determine the transition probabilities of the two models. We use maximum likelihood to learn parameters from data. A_{ij} is the number of transitions from state i to j in the training data[2]:

$$a_{ij} = \frac{A_{ij}}{\sum_{j'} A_{ij'}}$$

In this example, we must learn four transition probabilities: a_{HH}, a_{HL}, a_{LH} and a_{LL}. Given a sequence, $HHHLLHLHLL...$ we count the number of HH pairs to determine A_{HH} and normalize by the number of pairs of the form $H*$. The other transition probabilities are estimated similarly. We estimate the initial probability π_i by counting the number of sequences that begin with residue, i. Once we have learned the parameters, we can use the model to recognize new transmembrane sequences.

Evaluating a new sequence: Using this model, we can classify an observed sequence, $O = O_1, O_2, \ldots$, by its log odds ratio

$$S(O) = log \frac{P(O|TM)}{P(O|EC)}$$

where $P(O|TM) = \pi_{O_1} \prod_{i=1}^{T-1} a_{O_{i-1}O_i}$ is the probability of the observed sequence given the TM model. $P(O|EC)$ is defined similarly.

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 $\frac{0.072}{0.031} = 2.3$; in other words, it is a little more than twice as likely that `HHLHH` is a transmembrane sequence than an E/C sequence. The log-odds score is $\log_2(2.3) \approx 1.2$.

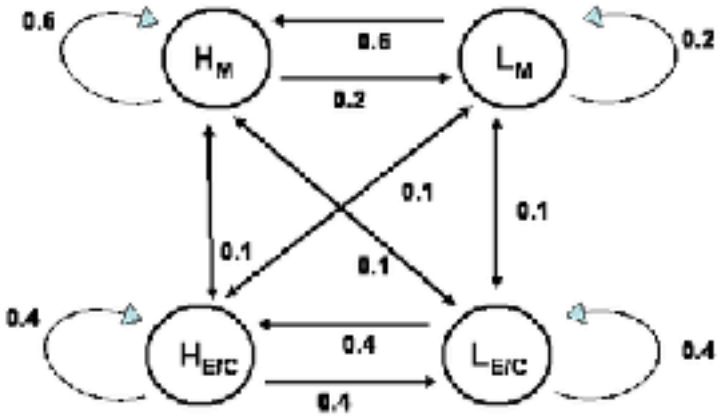
The above models are useful for classifying a sequence fragment where all residues are of the same class (i.e., all TM or all E/C). However, for finding boundaries in a sequence that has transitions from regions of one class to regions of another class, we would still need to score successive overlaping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

[2]Note that for some models, we may want to incorporate pseudocounts in the calculation of the parameters.

3 Hidden Markov Models

We now consider the following, more difficult, problem: Given a transmembrane protein sequence with TM regions interspersed with E/C regions, label each residue with its class (TM or E/C). To do this, we construct a new model by adding transitions connecting the TM and E/C models.

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 no longer a standard Markov chain. In a Markov chain, there a one-to-one correspondance between symbols and states, which is not true of the above model. For example, both H_M and $H_{E/C}$ are associated with hydrophilic residues. This four-state transmembrane model is a Hidden Markov model.

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

1. N states $S_1..S_N$
2. M symbols in alphabet
3. Transition probability matrix a_{ij}
4. Emission probabilities $e_i(a)$ probability state i emits character a.
5. Initial distribution vector $\pi = (\pi_i)$

We refer to the emission probabilities, the emission probabilities and the initial distribution, collectively as the parameters, designated $\lambda = (a_{ij}, e_i(a), \pi)$. Following the notation used in Durbin, we will refer to the sequence of observed symbols as $O = O_1, O_2, O_3, ...$ and the sequence of states visited as $Q = q_1, q_2, q_3, ...$ (the "state path".)

HMMs differ from Markov chains in a number of ways:

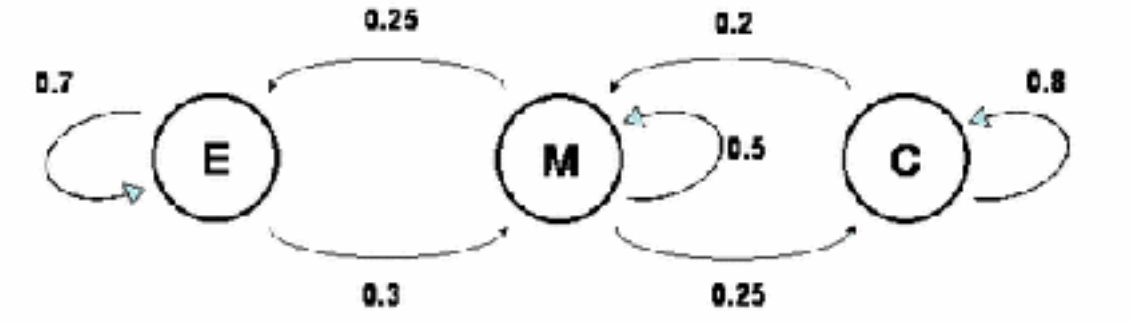
- Each state emits symbols from a fixed alphabet each time a state is visited. Emission probabilities are state-dependent, but not time-dependent.
- A symbol may be emitted by more than one state. For example, in the four-state HMM above, "H" is emitted by state H_M and by state $H_{E/C}$.
- Similarly, a state can emit more than one symbol. An example of this can be seen in the three-state HMM below.

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?

Unlike Markov chains, in an HMM there is no longer a one-to-one correspondence between states and output symbols. For a given sequence, O, in a Markov chain 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. The sequences are hidden because it is not possible to tell the state merely by the output symbol. 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 different states to distinguish between hydrophobic and hydrophilic residues in the TM region. Another possibility is to use only one state for the transmembrane model and use the emission probabilities distinguish between hydrophobic and hydrophilic residues. This approach is used in the following three-state HMM. Notice that we also now distinguish between extracellular sequences and cytosolic sequences by using separate E and C states.

Three-state transmembrane HMM:



As in the case of Markov chains, the parameters of an HMM can be learned from labeled data:

$$a_{ij} = \frac{A_i}{\sum_{j'} A_{ij'}} \qquad e_i(x) = \frac{E_i(x)}{\sum'_x E_i(x')}$$

Note that we now have to learn the initial probabilities, the transition probabilities and the emission probabilities. The parameters for this model are

i	E	M	C
π_i	0	0	1
$e_i(H)$	0.2	0.9	0.3
$e_i(L)$	0.8	0.1	0.7

In this example, we assume that all transmembrane sequences start in the cytosol; i.e., $\pi(C) = 1.0$.

4 The Viterbi algorithm:

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 will 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 defined by the transitions between C, M, and E states along this state path.

We approach this problem by assuming that most likely path, $Q^*(0) = \text{argmax}_Q\, P(Q|O)$, is a good estimation of the sequence of states that generated a given observed sequence O. This process is called "decoding" because we decode the sequence of symbols to determine the hidden sequence of states. 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.

We could use brute force by calculating $P(Q|O)$ for all paths. However, this becomes **intractable** as soon as number of states gets larger, since the number of state sequences grows exponentially (N^T). Instead, we calculate $\text{argmax}_Q\, P(Q, O)$ using a dynamic programming algorithm called the *Viterbi* algorithm. Note, that this will still give us the most probable path because the path that maximizes $P(Q, O)$ also maximizes $P(Q|O)$:

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

Let $\delta_t(i)$ be the probability of observing the first t residues and ending up in state S_i via *the most probable path*. We calculate $\delta_t(i)$ as follows:

Initialization:

$$\delta_1(i) = \pi_i e_i(O_1)$$

Recursion:

$$\delta_t(i) = \max_{1<=j<=N} \delta_{t-1}(j) \cdot a_{ji} \cdot e_i(O_t)$$

Final: The probability of the most probable path

$$P(Q^*) = \max_{1<=i<=N} \delta_T(i)$$

The final state, q_T is the state that maximizes $\delta_T(i)$. The state path can be reconstructed by tracing back through the dynamic programming matrix, similar to the traceback in pairwise sequence alignment.

Running time: $O(TN^2)$

There are TN entries in the dynamic programming matrix. Each entry requires calculating N terms.

An example: As an exercise, try applying the Viterbi algorithm to determine the most likely path through the three-state HMM above for for the sequence *LHHL*. You will need to fill out a matrix of values of $\delta_t(i)$ of this form:

		C	M	E
L	$t = 0$			
H	$t = 1$			
H	$t = 2$			
L	$t = 4$			

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.