

HMM Lecture Notes

Dannie Durand and Rose Hoberman

October 29th

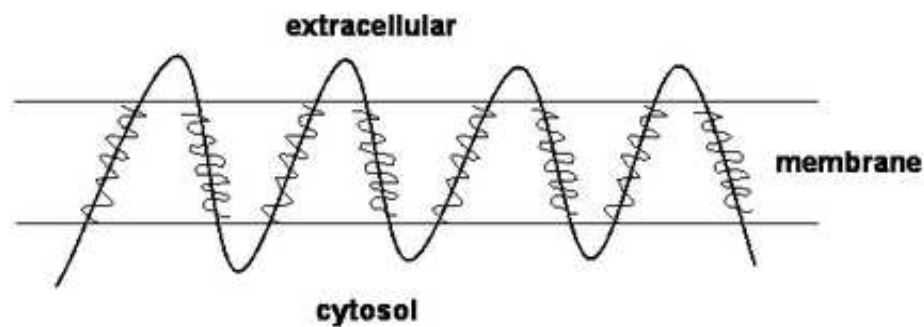
Outline

1. Review
2. Hidden Markov Models
3. The Viterbi algorithm
4. The Forward algorithm
5. The Backward algorithm

1 Review

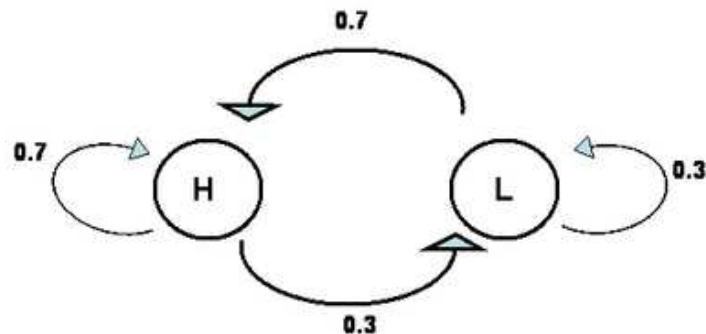
Last Tuesday, we introduced a boundary detection problem: We are given a sequence and we wish to label each symbol in the sequence according to its class (e.g. introns and exons; alpha helices and beta sheets; genes and non-coding regions). It is difficult to identify sharp boundaries between classes using by scoring windows using a fixed size model. Instead, we use Hidden Markov Models.

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?

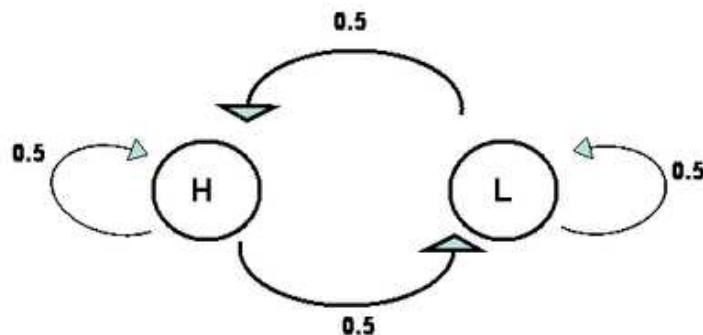


To do this we constructed two Markov models, a TM model and an E/C model. In these models, each amino acid is encoded as either hydrophobic (H) or hydrophilic (L).

Transmembrane model:



Extracellular/cytosol model:



First, 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 data¹:

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

In this example, we must learn four transition probabilities: a_{HH} , a_{HL} , a_{LH} and a_{LL} . Given a sequence, $HHHLLHLHLL\dots$ 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.

¹Note that for some models, we may want to incorporate pseudocounts in the calculation of the parameters.

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.

Using this model, we can classify an observed sequence, $O = O_1, O_2, \dots$, 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.

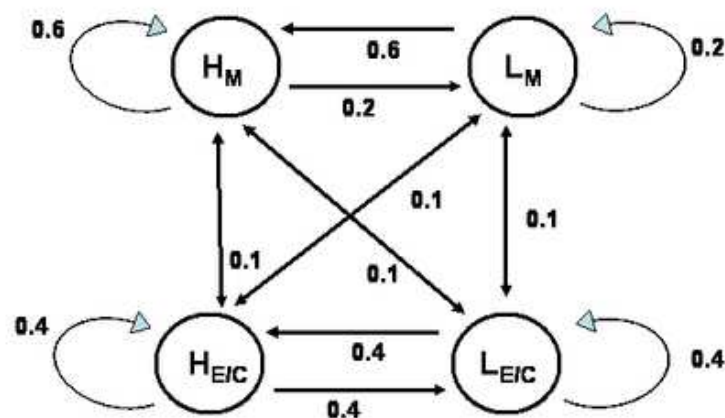
The above models are useful for classifying a sequence fragment where all residues are of the same class (i.e., all TM or all E/C) but less useful for finding boundaries in a sequence with transitions from regions of one class to regions of another class.

2 Hidden Markov Models

We now consider the following, more difficult, problem: Given a sequence 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 constructed the following Hidden Markov Model by adding transitions connecting the TM and E/C models. These new transitions indicate the boundaries between one class of region and another.

Four-state transmembrane HMM:



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 $\pi = (\pi_i)$

Components 1 - 3 specify the structure of the model, 3-5 specify parameters. We refer to the emission probabilities, the transition probabilities and the initial distribution, collectively as the parameters, designated $\lambda = (a_{ij}, e_i(a), \pi)$.

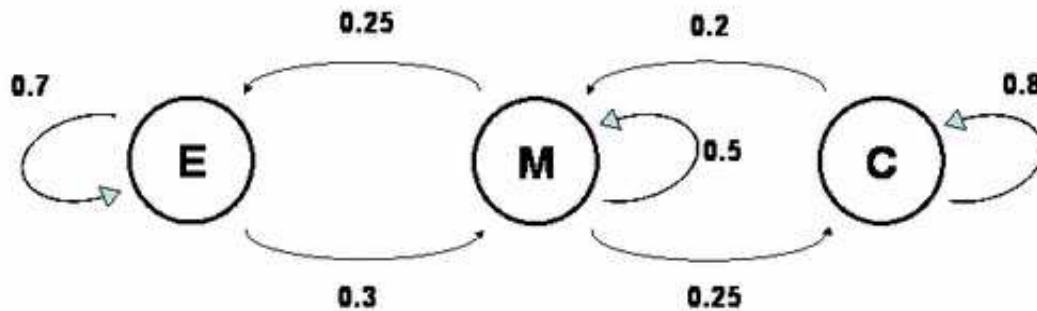
Notation The sequence of visited states: $Q = q_1, q_2, q_3, \dots$ (the state sequence) The sequence of emitted symbols: $O = O_1, O_2, O_3, \dots$ (the observed sequence).

Note that this is a *generative* model. It gives the probability of generating a particular sequence (hence, the emission probabilities.) This allows us to ask, what is the probability of a sequence of interest, if it were generated by a particular HMM?

HMMs differ from Markov chains in a number of ways:

- In HMM, the sequence of states visited is hidden. Unlike Markov Chains, there is no longer a one-to-one correspondence between states and output symbols. 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.
- 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 states H_M and $H_{E/C}$. Similarly, a state can emit more than one symbol. An example of this can be seen in the three state HMM below.

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 to 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:

Given labeled examples, we can still learn parameters for each separate model, then choose reasonable transitions between them (or learn them given long labeled data).

$$a_{ij} = \frac{A_i}{\sum_{j'} A_{ij'}} \quad e_i(x) = \frac{E_i(x)}{\sum_x E_i(x')}$$

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

We assume that all transmembrane sequences start in the cytosol.

3 The Viterbi algorithm

In order to find the boundaries between the transmembrane, extracellular and cytosolic regions, we seek $\operatorname{argmax}_Q P(Q|O)$, the most probable path through the model given the observed sequence, O . 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 $\operatorname{argmax}_Q P(Q, O)$ 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 $P(Q|O)$:

$$\operatorname{argmax}_Q P(Q|O) = \operatorname{argmax}_Q \frac{P(Q, O)}{P(O)} = \operatorname{argmax}_Q P(Q, O)$$

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:

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

The probability of the most probable path is the max of $\delta_T(i)$ over all states, S_i , and 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.

Running time: $O(TN^2)$

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

In class, we used the Viterbi algorithm to determine the most likely path through the three-state HMM above for the sequence *LHHL*.

4 The Forward Algorithm

We may also wish to determine the probability of a particular sequence, O , being generated to compare likelihood of different sequences or compare the likelihood of one sequence under different models. For example, suppose we want to compute the probability of observing a particular transmembrane sequence, given our HMM. We can compute this by summing over all states in each path: $P(O) = \sum_Q P(O, Q)$. As before, this becomes **intractable** as soon as number of states gets large, since the number of state sequences grows exponentially (N^T).

The Forward algorithm is a dynamic programming algorithm allows us to calculate, $P(O|\lambda)$, the probability that the model generated the observed sequence *efficiently*. Note, we do not know the state sequence so we must consider all state sequences.

We do this by iteratively calculating the probability of being in state i after generating the sequence up to observation O_t . We designate this quantity:

$$\alpha_t(i) = P(O_1, O_2, O_3, \dots, O_t, q_t = S_i)$$

Initialization:

$$\alpha_1(i) = \pi_i e_i(O_1)$$

Iteration:

$$\alpha_{t+1}(i) = \sum_{j=1}^N \alpha_t(j) * a_{ji} * e_i(O_{t+1})$$

The probability of observing the entire sequence is given by the sum over all possible final states:

$$P(O) = \sum_{i=1}^N \alpha_T(i)$$

Running time: $O(TN^2)$

Note that this algorithm is very similar to the Viterbi algorithm except that it uses a sum rather than taking the maximum.