

HMM Lecture Notes

Dannie Durand and Rose Hoberman

Thursday, November 5th

1 Notation

1. N states ($S_1..S_N$)
2. M symbols in alphabet, Σ
3. parameters, λ :
 1. initial distribution of states $\pi(i)$
 2. transition probabilities $a_{ij} = P(q_t = S_i | q_{t-1} = S_j)$. Note that $\sum_{i=1}^N a_{ij} = 1, \forall j$
 3. emission probabilities $e_i(a)$ probability state i emits a
4. Sequence of symbols: $O = O_1, O_2, \dots, O_T$
5. Sequence of states: $Q = q_1, q_2, \dots, q_T$

2 Using HMM's for pattern discovery

So far, we have focused on how to use an HMM to ask questions. Now we will consider how to construct a new HMM. There are two main issues: choosing the topology and inferring the parameters for that topology. We will postpone the question of topology until the next lecture and first consider parameter estimation.

2.1 Parameter estimation

Once the topology of the HMM has been determined, the parameters of the model, $\lambda = \{a_{ij}, e_i(\cdot), \pi_i\}$, are determined from observed sequences, O^1, O^2, \dots, O^k . There are two cases to consider: *labeled* and *unlabeled* data. In both cases, we obtain parameter values using maximum likelihood estimation.

Labeled data If the sequences are *labeled*, the main problem is to design the topology. Once the states and connectivity have been chosen, the transition and emission probabilities can be estimated easily using MLE, as follows:

$$\pi_i = \frac{P_i}{k}, \quad a_{ij} = \frac{A_{ij}}{\sum_{j'} A_{ij'}}, \quad e_i(a) = \frac{E_i(a)}{\sum_b E_i(b)},$$

where P_i is the number of sequences in which the first symbol is labeled with state i , A_{ij} is the number of adjacent symbols labeled with states i and j , respectively, and $E_i(a)$ is the number of

instances of symbol a labeled with state i . Note that for some models, we may want to use modified versions of the above equations to incorporate pseudocounts in the calculation of the parameters.

Unlabeled data If the sequences are *unlabeled*, then it is necessary both to design the topology and to learn the motif and the model parameters. Given sequences O^1, O^2, \dots , we wish to determine λ^* , the parameter values that maximize the likelihood of the sequences:

$$\lambda^* = \operatorname{argmax}_{\lambda} \sum_d P(O^d | \lambda) = \operatorname{argmax}_{\lambda} \sum_d \sum_Q P(O^d | \lambda, Q)$$

However, finding a global maximum is intractable. We would have to enumerate over all parameter sets, λ , as well as over all state paths, Q , for each sequence, O^d . Instead, people settle for heuristics which are 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 algorithm most commonly used for estimating the parameter values, the *Baum Welch* algorithm, belongs to a family of algorithms called Expectation Maximization (EM) algorithms. They all work by guessing initial parameter values, then estimating the likelihood of the data under the current parameters. These likelihoods can then be used to re-estimate the parameters, iteratively until a local maximum is reached. The Baum Welch algorithm learns the parameters from the data and implicitly, also discovers the motif. To determine the motif explicitly, we use the Viterbi algorithm on the new HMM to label the states of each input sequence.

The intuition behind the algorithm is as follows. We are given the observed, unlabeled sequences (denoted $O^d = O_1^d, O_2^d, \dots$). For a given path, Q , let $\mathcal{L}(\lambda) = \sum_d P(O^d | \lambda, Q)$ be the score of the parameters λ .

Choose initial values, $\lambda = (\pi, a_{ij}, e_i(\cdot))$.

Repeat:

For each sequence, O^d

Label O^d using posterior decoding, $P(q_t = S_i | O_t) = \alpha_t(i) \cdot \beta_{t+1}(i)$

Estimate new parameter values, λ using the method for labeled data described above until $(\mathcal{L}(\lambda))$ is stable).

The process of labeling the sequences using posterior decoding and re-estimating the parameters is achieved by repeated applications of the Forward and Backward algorithms to obtain $\alpha_t(i)$ and $\beta_{t+1}(i)$. The contribution of each sequence to the new parameter values should be weighted by $P(O^d | \lambda)$, the probability of the sequence under the current model parameters. This probability can also be estimated using either the Forward or the Backward algorithm.

More formally, for a given set of parameter values λ and a given sequence, O^d , the probability of transiting from state i to j at time t using posterior decoding is

$$\begin{aligned} P(q_t^d = i, q_{t+1}^d = j | O^d, \lambda) &= P(O^d) \cdot P(q_t^d = i, q_{t+1}^d = j, O^d) \\ &= P(O^d) \cdot \alpha_t(i) a_{ij} e_j(O_{t+1}^d) \beta_{t+1}(j) \end{aligned}$$

The term $\alpha_t(i)$ is the probability that the model has emitted symbols $O_1^d \dots O_t^d$ and is in state S_i at time t . This probability can be obtained using the Forward algorithm. Similarly, the Backward algorithm yields $\beta_{t+1}(j)$, the probability of emitting the rest of the sequence if we are in state j at time $t+1$. The remaining two terms, a_{ij} and $e_j(O_{t+1}^d)$ give the probability of making the transition from i to j and emitting the $t+1$ st character. From this we can estimate

$$A_{ij} = \sum_d P(O^d) \sum_t \alpha_t(i) a_{ij} e_i(O_{t+1}^d) \beta_{t+2}(i) \quad (1)$$

The probability $P(O^d)$ can be estimated using current parameter values using the Forward algorithm. Similarly,

$$E_i(\sigma) = \sum_d P(O^d) \sum_{\{t | O_t^d = \sigma\}} \alpha_t(i) \beta_{t+1}(i). \quad (2)$$

P_i , the number of sequences starting in state i , is given by $k \cdot \sum_l \pi_l e_i(O_l^d)$, where k is the total number of sequences. From P_i , A_{ij} , and $E_i(\sigma)$ we re-estimate the parameters. A formal statement of the Baum Welch algorithm is given at the end of these notes.

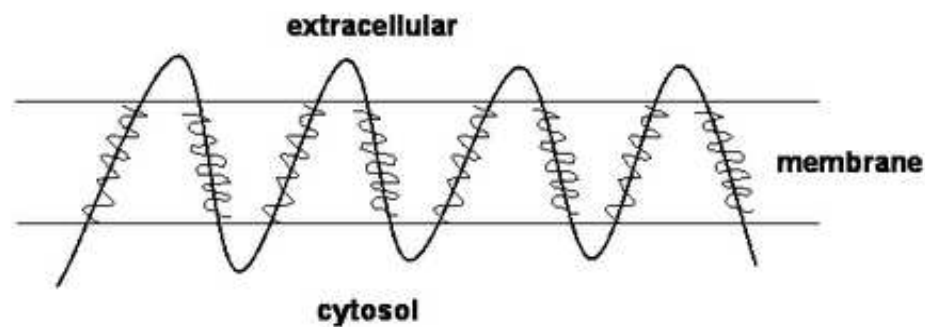
Convergence: It can be proven that if current estimate is replaced by these new estimates then the likelihood of the data will not decrease (i.e. will increase unless already at a local maxima/critical point). See Durbin, Section 11.6 for discussion of avoiding local maxima and other typical pitfalls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

2.2 Topology

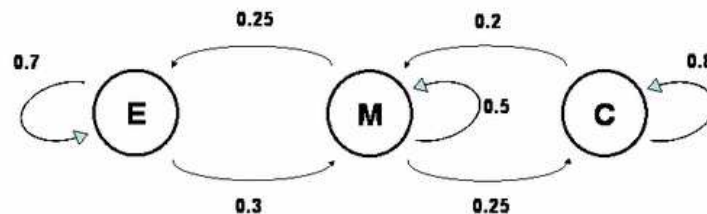
Finally, we must consider how to design the topology of the HMM. This includes the set of states, S_1, \dots, S_N , and how they are connected; in other words, we must specify which states will be connected by edges with non-zero values of a_{ij} . We must also choose the alphabet and decide which states will emit symbols.

We could just choose a fully connected graph, but this has too many parameters to estimate. Instead we can exploit domain knowledge. Choose a topology that limits the number of states and edges while still being expressive enough to represent the relationships they believe to exist.

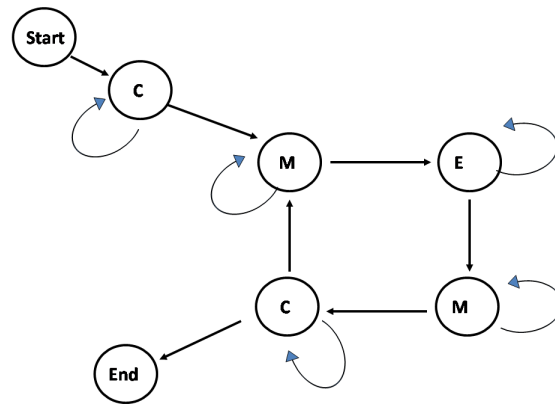
The transmembrane models we discussed at the end of October, illustrate some of the issues to consider. Recall that the goal was to model a sequence that started and ended in the cytosol, with multiple passes through the cell membrane, into the extracellular region, and back through membrane to the cytosol:



First, we chose to use 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 simple representation, it also requires fewer training sequences to learn the parameters since all hydrophobic (resp. hydrophilic) residues contribute to a single parameter rather than one parameter for each amino acid. We considered two models, a four state model in which each state emitted exactly one symbol, and a three state model, where all states emit both symbols, but the emission probabilities vary depending on the state.

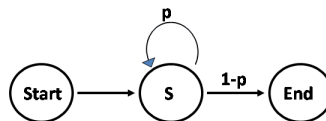


Second, note that the three state model above is not sufficiently restrictive to emit transmembrane proteins. For example, it can emit sequences that are entirely cytosolic or sequences that pass from the cytosol into the membrane and back to the cytosol, without ever passing through the extracellular region. By adding additional states, we can obtain a model that only emits sequences that start and end in the cytosol and pass through the membrane, the extra cellular region and the membrane before returning to the cytosol:



Thus, we can use topology to impose order dependencies on the model. Note, however, that while this model emits sequences of symbols with the correct sequence of states, it does not constrain these segments to be lengths typical for the associated cellular compartment.

Third, the choice of topology can impose a probability distribution on the length of the sequences that the HMM recognizes. For example, a simple self loop with probability p results in an exponentially decaying (geometric) distribution $P(l\text{residues}) = (1 - p)p^{l-1}$. We discussed topologies that assume other length distributions. These are described in Durbin, 3.4.



Algorithm: Baum Welch**Input:**

A set of observed sequences, O^1, O^2, \dots

Initialization:

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

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

Repeat

{

$$A = E = \pi = 0.$$

For each sequence, O^d ,

Calculate $\alpha_t(i)$, $\beta_t(i)$ and $P(O^d)$ using Forward and Backward algorithms.

$$\pi_i = \pi_i + P(O^d) \alpha_1(i)$$

For $t = 1$ to T

$$A_{ij} = A_{ij} + P(O^d) \cdot \alpha_t(i) a_{ij} e_i(O_{t+1}^d) \beta_{t+2}(i)$$

$$E_i(O_t^d) = E_i(O_t^d) + P(O^d) \alpha_t(i) \beta_{t+1}(i).$$

$$a_{ij} = \frac{A_{ij}}{\sum_l A_{il}}$$

$$e_i(\sigma) = \frac{E_i(\sigma)}{\sum_\tau E_i(\tau)}$$

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

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

}

Until (the change in $\mathcal{L}(\lambda)$ is less than some predefined threshold.)