

# HMM Lecture Notes

Tuesday, November 3rd

Rose Hoberman and Dannie Durand

## 1 Overview

In the past two weeks, we have discussed algorithms for recognizing new patterns, given an HMM for a pattern of interest. We now return to the questions of *modeling* and *discovery*.

PSSM's can capture the statistical properties of the family, but they have the following limitations:

1. They cannot capture positional dependencies.
2. They do not handle variable length patterns well.
3. They cannot recognize indels in the query sequence.

In contrast, HMMs handle gaps and pairwise dependencies well.

## 2 Notation

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

## 3 Topology

- Characteristics: number of nodes, alphabet, which edges to consider. 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 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

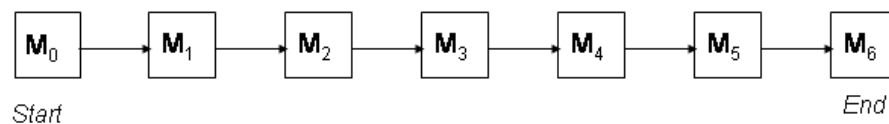
an exponentially decaying (geometric) distribution  $P(l \text{ residues}) = (1 - p)p^{l-1}$ . There are topologies that assume other length distributions (see Durbin, 3.4 for more on this subject).

### A basic topology:

Suppose we wish to construct an HMM for the WEIRD motif, based on the following alignment which has no gaps and no positional dependencies:

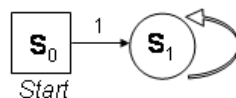
```
WEIRD
WEIRD
WEIRE
WEIQH
```

We can recognize the WEIRD motif using an HMM with this topology:



where the transition probabilities are  $a_{i,j} = 1$  if  $j = i + 1$  and zero, otherwise. The emission probabilities are  $e_j(\alpha) = F[\alpha, j]$ , where  $F[\alpha, j]$  is the same frequency matrix that we derived for the PSSM example, using pseudocounts. The Start and End states ( $M_0$  and  $M_6$ ) are silent. The above model is our alternate hypothesis,  $H_A$ .

To score a new sequence, we also need a background model (the null hypothesis,  $H_0$ ):



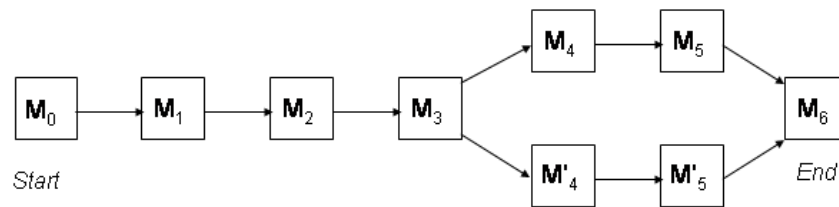
In this model, all transition probabilities are equal to one. The emission probabilities are  $e_j(\alpha) = p(\alpha)$ , where  $p(\alpha)$  is the background frequency of residue  $\alpha$ . We can then score a new sequence,  $O$ , by calculating  $\log \frac{P(O|H_A)}{P(O|H_0)}$ . In class, we saw that we obtain a score equivalent to  $\sum_{i=1}^5 S[o_i, i]$ , the score we would have obtained with the PSSM for the WEIRD motif.

**Positional dependencies:**

Now suppose 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

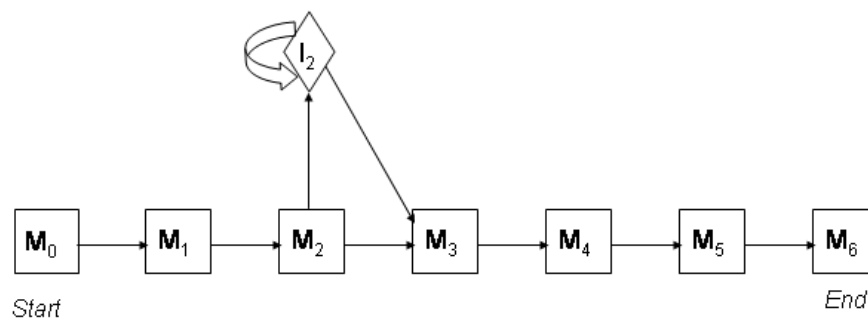
A PSSM for this motif, however, would give the sequences WEIRD and WEIRH equally good scores. So would the basic HMM above. We can construct an HMM to model this pairwise dependency like this:



where the emission probabilities are  $e_{M_4}(R) = 1$ ,  $e_{M_5}(D) = 1$ ,  $e_{M'_4}(Q) = 1$  and  $e_{M'_5}(H) = 1$ .

**Insertions:**

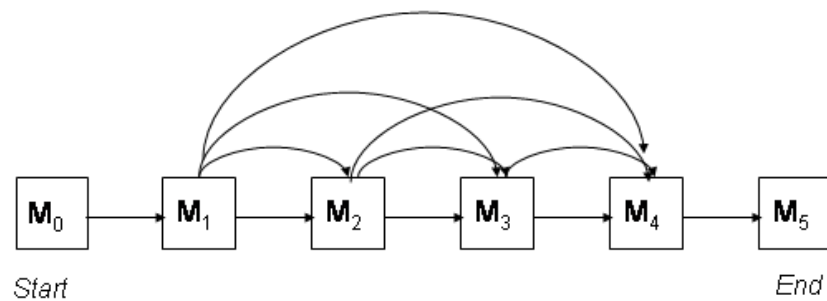
We can modify the basic HMM to recognize query sequences with insertions such as  $O = \text{WECIRD}$ :



where the emission probabilities for the insertion states are the background frequencies.

### Deletions:

Suppose our query sequences has a deletion, e.g.,  $O = \text{WERD}$ . One approach to capturing such deletions would be to add edges allowing us to jump over any set of match states:

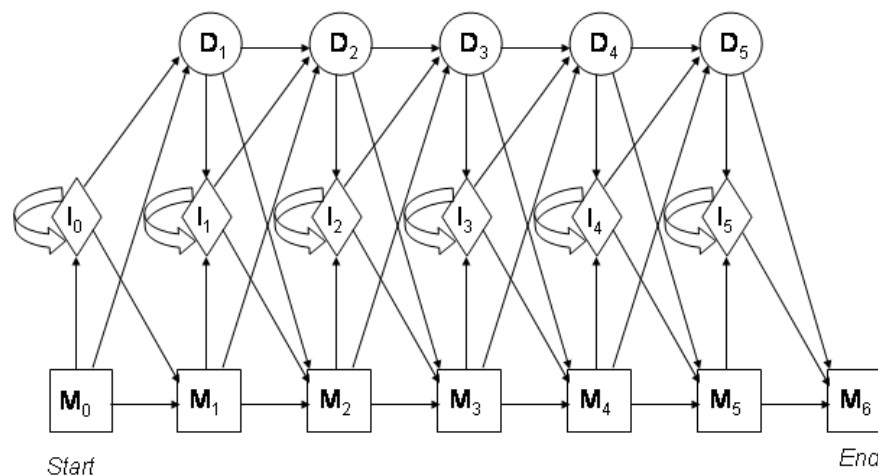


The disadvantage to this approach is that we would need a very large set of training data to infer the transitions, one in which all deletions of all possible sizes were represented. Instead, we can model long deletions as sequences of short ones, as seen below

## 4 Profile HMMs

A *Profile HMM* is a standard topology for modeling sequence motifs. It was proposed by Krogh and Haussler in 1994. This is a profile HMM for a pattern of length five.

**A profile HMM of length 5**



Each insert and match state emits the 20 AA and delete emits “-”. The emission and transition probabilities must be estimated from data.

### Parameter estimation:

Given labeled training data (i.e., we are given the state path), we use maximum likelihood to estimate the parameters. In general,

$$e_k(\sigma) = \frac{E_i(\sigma) + b}{\sum_j E_j(\sigma) + 20b}$$

$$a(i, j) = \frac{A(i, j)}{\sum_l A(i, l)}$$

where  $E_i(\sigma)$  is the number of instances in the training data where symbol  $\sigma$  is emitted in state  $i$  and  $A(i, j)$  is the number of transitions from  $i$  to  $j$  in the training data plus a pseudocount to take transitions that are not observed into account. Zeros are bad for estimation because we want a distribution over all possible sequences with a peak for family members. Unless there are many family members, use smoothing (simplest is to add pseudocounts).

For our Profile HMM, the estimation of the emission probabilities might look like this:

$$\begin{aligned}
 e_{M_0}(i) &= e_{M_0}(i) = 0 \forall i \\
 e_{I_k}(i) &= q_i \forall i, k \\
 e_{D_k}(i) &= 0 \quad e_{D_k}("-") = 1 \\
 e_{M_k}(i) &= \frac{E_k(i) + b}{\sum_j E_k(j) + 20b}
 \end{aligned}$$

### An example: A profile HMM for a variable length motif

Profile HMM's like the one above can be used to model variable length motifs, such as this one:

```

VG--H
V---N
VE--D
IAADN

```

The length of the HMM should be the average of the length of the sequences. The above sequences are of lengths 3, 2, 3 and 5, respectively, yielding an average of 3.25. Our HMM will have a silent start state  $M_0$ , match states  $M_1, M_2, M_3$ , insertion states  $I_0, I_1, I_2, I_3$ , deletion states  $D_1, D_2, D_3$  and a silent end state  $M_4$ .

In order to estimate the parameters, we need to assign labels to the data using the multiple alignment. Positions in the alignment that have gaps in less than 50% of the rows correspond to match states. Those with more than 50% gaps correspond to insertion states:

```

V   G   -   -   H
V   -   -   -   N
V   E   -   -   D
I   A   A   D   N
M1 M2 I2 I2 M3

```

This yields the following labeled sequences:

V	G	H
$M_1$	$M_2$	$M_3$

V	-	H
$M_1$	$D_2$	$M_3$

V	E	D
$M_1$	$M_2$	$M_3$

I	A	A	D	N
$M_1$	$M_2$	$I_2$	$I_2$	$M_3$

From these labeled sequences, we can estimate the parameters. For example,

$$e_{M_1}(V) = \frac{3 + 1}{4 + 20}$$

and

$$a_{M_2 I_2} = \frac{1 + 1}{(2 + 1) + (1 + 1) + (0 + 1)}$$

The three sums in the denominator correspond to all possible transitions out of state  $M_2$ , plus pseudocounts. Specifically, in the training sequences there are two transitions from  $M_2$  to  $M_3$ , one one transition from  $M_2$  to  $I_2$  and no transitions from  $M_2$  to  $D_3$ .

## Pattern recognition with profile HMM's

Given a new, unlabeled sequence,  $O$ , does it contain the motif? Two approaches

- Calculate  $\log \frac{P(O|H_A)}{P(O|H_0)}$  using the Forward algorithm. This gives a score but doesn't tell us the location.
- Find the most likely path using the Viterbi algorithm. The location of the motif corresponds to the symbols emitted by the match states. If no symbols were emitted by match states, then the motif is not present in  $O$ .

There are specialized versions of the Forward and Viterbi algorithms for profile HMM's (see Durbin, pp 109-110.)

## 5 Baum Welch

In general, if we have labeled data (that is, we know the state sequence), we can obtain the parameters using Maximum Likelihood Estimation. The Baum-Welch algorithm is used to estimate the model parameters when the state path is unknown. Given sequences  $O^1, O^2, \dots$ , we wish to determine  $\lambda = (a_{ij}, e_i(\cdot))$ . We generally want to choose parameters that will maximize the likelihood of our data.

However, finding a global maximum is intractable. We would have to enumerate over all parameter sets,  $\lambda_k$ , and then calculate

$$\text{Score}(\lambda_k) = \sum_d P(O^d | \lambda_k) = \sum_d \sum_Q P(O^d | \lambda_k, Q)$$

for each  $\lambda_k$ . 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 won't really discuss this.

The algorithm (BW) used for selecting the parameter values 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.

**Setup** The alphabet and the number of states,  $N$ , are fixed. We are given the observed sequences (denoted  $O^d = O_1^d, O_2^d, \dots$ ).

The intuition behind the algorithm is as follows

1. Choose some initial values for  $\lambda(\pi, a_{ij}, e_i(\cdot))$ .
2. Determine "probable paths"  $Q^d = q_1^d, q_2^d, \dots$
3. Count the expected number of transitions,  $A_{ij}$ , from state  $i$  to state  $j$ , given the current estimate of  $\lambda$ .
4. Count,  $E_i(\sigma)$ , the expected number of times character  $\sigma$  is emitted from state  $i$ .
5. Re-estimate  $\lambda(\pi, a_{ij}, e_i(\cdot))$  from  $A_{ij}$  and  $E_i(\sigma)$ ,
6. if not converged, go to step 2

For a given sequence,  $O^d$ , probability of transiting from state  $i$  to  $j$  at time  $t$  is

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

The term  $\alpha(t, i)$  is the probability that we are in state  $i$  at time  $t$  and 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

$$A_{ij} = \sum_d \frac{1}{P(O^d)} \sum_t \alpha(t, i) a_{ij} e_i(O_{t+1}^d) \beta(t+1, i) \quad (1)$$

The probability of  $O^d$  can be estimated using current parameter values using the forward algorithm.

Similarly,

$$E_i(\sigma) = \sum_d \frac{1}{P(O^d)} \sum_{\{t | O_t^d = \sigma\}} \alpha(t, i) \beta(t, i). \quad (2)$$

From  $A_{ij}$  and  $E_i(\sigma)$  we re-estimate the parameters.

Stated formally:

### Algorithm: Baum Welch

#### Input:

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

#### Initialization:

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

score =  $\sum_d P(O^d | \lambda')$ .

Repeat

{

$\lambda = \lambda', S = S'$

For each sequence,  $O^d$ ,

{

/\* Calculate ‘‘probable paths’’  $Q^d = q_1^d, q_2^d, \dots$  \*/

Calculate  $\alpha(t, i)$  for  $O^d$  using the Forward algorithm.

Calculate  $\beta(t, i)$  for  $O^d$  using the Backward algorithm.

Calculate the contribution of  $O^d$  to  $A$  using (1).

Calculate the contribution of  $O^d$  to  $E$  using (2).

}

}

$$\begin{aligned}
 & \} \\
 a_{ij} &= \frac{A_{ij}}{\sum_l A_{il}} \\
 e_i(\sigma) &= \frac{E_i(\sigma)}{\sum_{\tau} E_i(\tau)} \\
 \text{score} &= \sum_d P(O^d | a_{ij}, e_i()). \\
 & \} \\
 & \text{Until (the change in score is less than some predefined threshold.)}
 \end{aligned}$$

The estimation of “probable paths”  $Q^d = q_1^d, q_2^d, \dots$  in the inner loop is done efficiently using dynamic programming. This was not covered in class. This is done using the Forward and Backward algorithms. You are not responsible for the details of how the Forward and Backward algorithms are used in Baum Welch, but you should understand the basic idea of the algorithm and are responsible for knowing when Baum Welch should be applied.

**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.

**Optimizing Simultaneously** Before we knew parameters or state sequence, so we held one fixed and optimized the other. Now we’re optimizing both simultaneously.

## 6 Multiple Sequence Alignment

- **Setting length & topology:** Given a set of unaligned sequences, let the length of HMM (i.e., the number of match states) be the average length of sequences.
- **Learn parameters** Guess “good” initial parameters ( $a_i(M_j) \gg a_i(I_j)$  or  $a_i(D_j)$ ). Train model using Baum Welch (to avoid local maxima and other typical problems, see Durbin for discussion of some approaches).
- **Most probable path gives alignment** Use the Viterbi algorithm ( $\pi^* = \operatorname{argmax}_j P(\pi, s_j) \forall s^j$ ) to find path most likely to produce each sequence. Use these paths to determine the alignment. If  $O_t^d$  and  $O_u^c$  were emitted by same match state, then align positions  $t$  and  $u$ . See Ewens and Grant, p 338 for an example.
- **Model surgery:** The topology of the model can be iteratively refined. If more than half of the sequences enter the delete state at a particular position remove that match state from the topology. If more than half of the sequences enter the insert state at a given position, add match states (number equal to average length of the insertion).
- **No alignment for insertions.** This method doesn’t say how to align indel sequences of different length. Correspond to unconserved portions, not meaningfully alignable. Often just left-justified and shaded.

- **Quick and elegant:** Compared with the exact dynamic programming algorithm for multiple sequence alignment, which runs in exponential time, this approach can align many sequences quickly. Doesn't use precomputed substitution matrix and gap penalties. (No parameter selection problem and more specific to data).

An example of this is given in Ewens and Grant, pp. 337 - 339.