sequence.

3. What is the probability of being in state E_i when O_t is emitted?
   Example: Is a given residue localized to the membrane?

We then discussed several approaches to answering these questions:

- Calculating $P(O|\lambda)$ using the Forward or Backward algorithms
- Inferring the state path that emitted O using Viterbi or Posterior decoding
- Inferring the state that emitted O_t using the Forward and Backward algorithms

These tools can be used to answer biological questions in a variety of ways. For example, one approach to predicting whether O is a transmembrane protein is to calculate $P(O|\lambda_{TM})$, the probability that O was emitted by the transmembrane model. However, the resulting probability can be difficult to interpret. How big must the probability be to convince us that O is in fact a transmembrane sequence? To answer the question, it is useful to construct a model representing a null hypothesis and to calculate $P(O|\lambda_0)$, the probability that O was emitted by this null model. If the resulting likelihood ratio

$$\frac{P(O|\lambda_{TM})}{P(O|\lambda_0)}$$

is much greater than one, then we can infer that O is a transmembrane sequence.

An alternate approach would be to infer the state path that emitted O using the Viterbi or posterior decoding. If the resulting path includes membrane states, then we can conclude that O is a transmembrane sequence. If the entire sequence is labeled with C states or with E states, then we conclude that it is not.

5.7 Designing HMMs: Motif discovery and modeling

There are three major computational tasks associated with conserved motifs found in multiple sequences: Discovery, Modeling, and Recognition. In previous sections, we discussed the recognition problem: Given an HMM, how do we use it to ask questions about patterns in a new, unlabeled sequence? Here, we consider modeling and discovery. For HMMs, modeling and discovery are closely coupled. There are two major issues to consider: designing the HMM topology and estimating the parameters of the model. A fundamental tradeoff drives HMM design: On the one hand, more complex models, with more parameters, can yield more accurate and biologically realistic models. On the other hand, as the number of parameters increases, so does the amount of data needed to estimate parameters without overfitting.

5.7.1 HMM topology

The topology of an HMM is determined by the set of states, $E_1, \ldots, E_N$, and how they are connected; in other words, we must specify which pairs of states will be connected by edges with non-zero transition probabilities. We could just choose a fully connected graph, but typically this has too many parameters to estimate. Instead, we can exploit biological knowledge. The goal is to choose a topology that limits the number of states and edges required, while still being expressive enough to represent the structure of the biological pattern of interest.

The choice of model topology can have a strong impact on the properties of the patterns we will discover. HMMs map symbols to states. Since changes in state define motif boundaries, how states are defined will influence the results of boundary detection. Inter-site dependencies and flexibility in pattern length can also be encoded in the topology of an HMM.

The transmembrane models we have discussed illustrate some of the issues to consider. For example, the three state model in Fig. 5.5 is not sufficiently restrictive to emit only transmembrane protein sequences. 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.

We can impose additional order dependencies on sequences generated by the model by modifying the topology. Suppose the goal is to generate sequences that always start and end in the cytosol, with one or more passes through the cell membrane into the extracellular matrix, and back through membrane to the cytosol. By adding additional states, we can obtain a model that only emits sequences that satisfy these conditions (Fig. 5.8). Note that this model has silent *Start* and *End* states, which we have not encountered before. These states do not emit symbols. They serve to ensure that the entry and exit from the model occur in specific states.

The topology of the model also influences the distribution of the lengths of sequences that the model can emit. For example, a simple self loop with probability p (Fig. 5.9) results in sequences with lengths that follow an exponentially decaying (geometric) distribution. The probability that this model will emit a sequence of length l is $(1-p)p^{l-1}$. This does not correspond to the length distribution of real amino acid sequences. More realistic length distributions can be obtained with more complex topologies. Some of these are described in Durbin, section 3.4.

In addition to specifying the model topology, we must choose the alphabet and decide which states will emit which symbols. The larger the alphabet, the greater the number of emission probabilities that must be estimated from training data. The transmembrane models we discussed in class used 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 simpler representation, it also requires fewer training sequences to

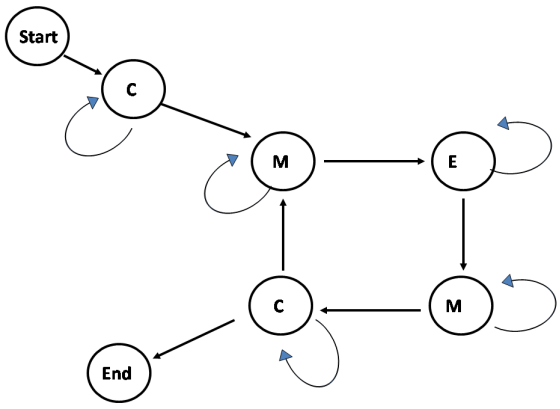

Figure 5.8: An HMM that emits transmembrane sequences that start and end in the cytosol and make at least one pass through the membrane to the extracellular matrix and back through the membrane to the cytosol. Note the use of silent Start and End states ensures that sequences start and end in the cytosol.

learn the parameters since all hydrophobic (resp., hydrophilic) residues contribute to the estimation of a single parameter.

5.7.2 Parameter estimation

Once the states and connectivity have been chosen, the parameters of an HMM are estimated from training data. We are given observed sequences, $O^1, O^2, ..., O^k$, and wish to construct an HMM with parameters, λ, to model these sequences. If the sequences are labeled, the transition and emission probabilities can be estimated easily from the observed transition and emission frequencies. If the sequences are unlabeled, we must first discover the conserved pattern using a machine learning algorithm.

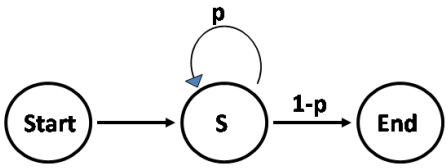

Figure 5.9: An HMM with one non-silent state that emits a single symbol, σ, with unit emission probability ($e_s(\sigma) = 1$). The probability that a sequence is emitted by this HMM decreases exponentially with the length of the sequence.

Labeled sequences:

When we are given labeled sequences in which every symbol O_t^d is associated with a state, $q_t^d = E_i$, the parameters can be estimated by tabulating the emission and transition frequencies in the data. Note that inferring parameters from counts in labeled data is a form of maximum likelihood estimation; we are assuming that the emission and transition probabilities that best model the motif of interest are those that maximize the probability of the observed symbols and states in the training data.

The transition probabilities are calculated by tabulating the number of observed state changes in the data:

$$a_{ij} = \frac{\sum_{d=1}^{k} A_{ij}^d}{\sum_{d=1}^{k} \sum_{j\prime} A_{ij\prime}^d},$$

where A_{ij}^d is the number of pairs of adjacent symbols, $O_t^d\ O_{t+1}^d$, that are labeled E_iE_j. The emission probabilities are given by

$$e_i(\sigma) = \frac{\sum_{d=1}^{k} \mathcal{E}_i^d(\sigma)}{\sum_{d=1}^{k} \sum_{\alpha\in\Sigma} \mathcal{E}_i^d(\alpha)}$$

where $\mathcal{E}_i^d(\sigma)$ is the number of instances in O^d where the symbol σ is labeled with state E_i. Finally, the initial probability π_i is given by

$$\pi_i = \frac{1}{k} \sum_{d=1}^{k} I^d(i), \tag{5.4}$$

where

$$I^d(i) = \begin{cases} 1, & \text{if } q_1^d = E_i \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

is an indicator variable that is equal to one when the first symbol in O^d is labeled with state E_i and zero otherwise.

We may wish to include pseudocounts to account for cases not observed in the training data. Pseudocounts are incorporated into the emission probabilities in the same way that we used pseudocounts in the definition of the frequency matrix for PSSMs. The probability of emitting σ from state E_i is

$$\begin{aligned} e_i(\sigma) &= \frac{\sum_{d=1}^{k} \mathcal{E}_i^d(\sigma) + b}{\sum_{\alpha\in\Sigma} \left(\sum_{d=1}^{k} e_i^d(\alpha) + b \right)} \\ &= \frac{\sum_{d=1}^{k} \mathcal{E}_i^d(\sigma) + b}{\left(\sum_{\alpha\in\Sigma} \sum_{d=1}^{k} e_i^d(\alpha) \right) + |\sum| b} \end{aligned} \tag{5.6}$$

where b is a pseudocount. In class, we have used $b = 1$ as a pseudocount. There are more sophisticated approaches to selecting a pseudocount. We will not cover them in this course; for those interested in learning more about this on their own, Durbin's discussion of Dirichlet mixtures in Section 11.5 of his book provides a starting point.

We can also use pseudocounts to account for state transitions that are allowed, but not observed in the training data. Let $\mathcal{N}(i)$, the neighborhood of state E_i, be the set of states that can be reached from E_i in a single transition. In other words, $\mathcal{N}(i)$ is the set of states, E_j such that a_{ij} has not been explicitly defined to be zero in the design of the topology. Then,

$$a_{ij} = \frac{\left(\sum_{d=1}^{k} A_{ij}^{d}\right) + b}{\sum_{j' \in \mathcal{N}(i)} \left(\left(\sum_{d=1}^{k} A_{ij'}^{d}\right) + b\right)}$$

$$= \frac{\left(\sum_{d=1}^{k} A_{ij}^{d}\right) + b}{\left(\sum_{j' \in \mathcal{N}(i)} \sum_{d=1}^{k} A_{ij'}^{d}\right) + |\mathcal{N}(i)|b} \tag{5.7}$$

As an example, consider the three-state transmembrane model in Fig. **??**. For this model, seven transition probabilities must be inferred: a_{CC}, a_{CM}, a_{MC}, a_{MM}, a_{ME}, a_{EM}, and a_{EE}. To estimate A_{CC}, for example, given the following labeled sequence

H	H	H	L	L	H	L	H	L	L	H	H	H	H	H
C	C	C	C	C	C	C	C	C	C	M	M	M	M	M

we count the number of CC pairs and normalize by the number of pairs of the form $C*$ where $*$ can be any state. Since there are nine pairs of adjacent symbols labeled CC and one pair labeled CM, for this sequence $a_{CC} = 0.9$ without pseudocounts. With pseudocounts,

$$a_{CC} = \frac{A_{CC} + b}{\sum_{j' \in \mathcal{N}(C)} [A_{Cj'} + b]} ,$$

where $\mathcal{N}(C) = \{C, M\}$. With a pseudocount of $b = 1$, we obtain

$$\begin{aligned} a_{CC} &= \frac{9+1}{(9+1)+(1+1)} , \\ &= \frac{10}{12}. \end{aligned}$$

To obtain the emission probabilities from state C, note that C is associated with five hydrophobic and five hydrophilic residues. Thus,

$$\begin{aligned} e_C(\texttt{ H}) &= \frac{\mathcal{E}_C(\texttt{ H}) + b}{\sum_{\alpha \in \{\texttt{ H, L}\}} \mathcal{E}_C(\alpha) + 2b} , \\ &= \frac{5+1}{10+2} \end{aligned}$$

or $e_C(\mathtt{H}) = 0.5$, again assuming a pseudocount of $b = 1$.

The other transition and emission probabilities are estimated similarly. We estimate the initial probability π_C by counting the number of sequences that begin in the cytosol, and normalizing by the total number of sequences.

Unlabeled sequences:

If the sequences are *unlabeled*, then it is necessary to both discover the motif and learn the model parameters. The motif is discovered automatically, but implicitly, through the process of parameter inference. Once the parameters have inferred, the parameters are used to obtain an explicit model of the motif via Viterbi or posterior decoding.

Parameters are inferred using maximum likelihood estimation. Given sequences $O^1, O^2, \ldots O^k$, we seek $\lambda_l = \{a_{ij}, e_i(\cdot), \pi_i\}$ such that the probability of observing the input sequences is maximized. Stated formally,

$$\begin{aligned} \lambda &= \operatorname*{argmax}_{\lambda_l} \mathcal{L}(\lambda_l) \\ &= \operatorname*{argmax}_{\lambda_l} \sum_d P(O^d|\lambda_l) \\ &= \operatorname*{argmax}_{\lambda_l} \sum_d \sum_Q P(O^d|\lambda_l, Q) \ . \end{aligned}$$

Except for very small problem instances, finding a global maximum is intractable. We would have to calculate $\mathcal{L}(\lambda_l)$ for all possible combinations of parameters, λ_l, to find the parameters that maximize $P(O^1 \ldots O^k|\lambda_l)$. Instead, heuristics are used. These are typically 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 *Baum-Welch* algorithm (Algorithm 3) is used to estimate the parameters of a Hidden Markov model from unlabeled training data. Baum-Welch belongs to a family of algorithms, called Expectation Maximization (EM) algorithms, that work by alternating between estimating the likelihood of the data, given the current estimate of the parameters and re-estimating the parameters from the current likelihoods. Baum-Welch is based on algorithms that we have already encountered: Given labeled data, we can estimate the model parameters using Equations 5.4-5.7, as described in the previous section. Given a model with parameters, we can label unlabeled sequences using Viterbi or posterior decoding.

Informally, Baum-Welch is an iterative algorithm that alternately applies these two procedures. First, an initial estimate of the parameter values is required, for example, based on prior knowledge of the biology underlying the model or on a uniform prior. With this initial estimate, the model is used to label the training data, typically using posterior

Algorithm 1: Baum-Welch

Input:

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

Initialization:

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

Iteration:

Repeat

{

For each sequence, O^d,

{

Calculate $\alpha(t,i)$, $\beta(t,i)$ and $P(O_d)$ using Forward and Backward algorithms.

$A^d_{ij} = \frac{1}{P(O^d)} \cdot \sum_t \alpha(t,i) a_{ij} e_j(O^d_{t+1}) \beta(t+2, j)$

$\mathcal{E}^d_i(\sigma) = \frac{1}{P(O^d)} \sum_{\{t|O^d_t = \sigma\}} \alpha(t,i)\, \beta(t+1, i).$

}

$a_{ij} = \frac{\sum_d A^d_{ij}}{\sum_d \sum_l A^d_{il}}$

$e_i(\sigma) = \frac{\sum_d \mathcal{E}^d_i(\sigma)}{\sum_d \sum_\alpha \mathcal{E}^d_i(\alpha)}$

$\pi_i = \sum_{d=1}^k \frac{I^d(i)}{P(O^d)}$

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

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

}

Until ($\mathcal{L}(\lambda)$ stops changing.)

Algorithm 3: **Baum Welch**

decoding with the Forward and Backward algorithm. Once the sequences have been labeled, the parameters are re-estimated from the labeled data. The training sequences are then re-labeled using this new estimate of the parameters. The algorithm iterates, alternately labeling the data with the current estimate of the parameter values and then re-estimating the parameters from the labeled data. At each iteration, the likelihood is guaranteed to remain unchanged or increase. This iterative process terminates when the likelihood ceases to improve.

It is instructive to note the similarities and differences between the Baum-Welch algorithm and the Gibbs sampler. Like Baum-Welch, the Gibbs sampler alternates between re-estimating parameters (i.e., a PSSM) from the current estimate of the motif and inferring a new instance of the motif from the updated parameters. However, unlike Baum-Welch, where every training sequence is relabeled at each iteration, in the Gibbs sampler, only one sequence is relabeled at each iteration. A second major difference between the two methods is that the Gibbs sampler is guaranteed to converge to a global optimum given enough time. In contrast, the Baum-Welch algorithm is only guaranteed to find a local optimum.

Given the observed, unlabeled sequences, the parameters are re-estimated in the inner loop of the algorithm. A_{ij} is the expected number of transitions from E_i to E_j. For a given sequence, O^d, probability of transiting from state E_i to E_j at time t is $P(q_t^d = i, q_{t+1}^d = j|O^d, \lambda)$. The number of total transitions from E_i to E_j can be obtained by summing over all time steps, $t = 1$ to T and all input sequences:

$$A_{ij} = \sum_d \sum_t P(q_t^d = i, q_{t+1}^d = j|O^d, \lambda) \tag{5.8}$$

To facilitate the calculation, we again use the trick of converting the conditional probability into a joint probability:

$$\begin{aligned} P(q_t^d = i, q_{t+1}^d = j|O^d, \lambda) &= \frac{P(q_t^d = i, q_{t+1}^d = j, O^d)}{P(O^d)} \\ &= \frac{\alpha(t,i)a_{ij}e_j(O_{t+1}^d)\beta(t+2,j)}{P(O^d)} . \end{aligned}$$

The term $\alpha(t,i)$ is the probability that the model has emitted symbols $O_1^d \ldots O_t^d$ and is in state E_i at time t. This probability can be obtained using the Forward algorithm. The term in the denominator, $P(O^d)$, is also calculated with the Forward Algorithm. The terms a_{ij} and $e_j(O_{t+1}^d)$ give the probability of making the transition from E_i to E_j and emitting O_{t+1}. The Backward algorithm yields $\beta_{t+2}(j)$, the probability of emitting the rest of the sequence if the model was in state E_j at time $t+1$. From this we can estimate

$$A_{ij} = \sum_d \frac{\sum_t \alpha(t,i)\, a_{ij}\, e_j(O_{t+1}^d)\, \beta(t+2,j)}{P(O^d)} \tag{5.9}$$

Note that instead of explicitly labeling the data and then counting state transitions as we do with labeled data, the association of symbols and states is implicit in the re-estimation process in the inner loop of the algorithm.

$\mathcal{E}_i(\sigma)$ is the expected number of times that σ is emitted from state E_i:

$$\mathcal{E}_i(\sigma) = \sum_d P(q_t^d = E_i, O_t^d = \sigma | O^d, \lambda) \tag{5.10}$$

$$= \sum_d \frac{\sum_{\{t|O_t^d=\sigma\}} \alpha(t,i)\,\beta(t+1,i)}{P(O^d)}\,. \tag{5.11}$$

Again, the quantities on the right hand side can be calculated using the Forward and Backward algorithms. Finally, the initial probability π_i is given by

$$\pi_i = \sum_{d=1}^{k} p(q_1^d = S_i | O_d, \lambda) \tag{5.12}$$

$$= \sum_{d=1}^{k} \frac{I^d(i)}{P(O^d)} \tag{5.13}$$

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 maximum/critical point). See Durbin, Section 11.6 for discussion of avoiding local maxima and other typical pitfalls with this algorithm.

The *Baum-Welch* algorithm estimates the values of the parameters from training data and, thus, implicitly discovers the motif. *Baum-Welch* does output an explicit description of the motif. To determine the motif explicitly, the Viterbi or posterior decoding are used to label each of the input sequences.

5.8 Profile HMMs

In 1994, Krogh, Haussler[2] and colleagues introduced a generic HMM topology specifically designed to model conserved sequence motifs. It captures the propensity to observe specific amino acids or nucleotides at each position in a pattern and allows for insertions and deletions. This topology, called a *Profile HMM*, can be customized for a broad range of conserved motifs by selecting the appropriate length for a given motif and initializing the parameters to capture the specific properties of the motif.

Here, we introduce the features of the Profile HMM model by showing how it could be used to model the `WEIRD` motif based on the following alignment, which has no gaps and no positional dependencies:

[2]Krogh, A., Brown, M., Mian, I. S., Sjolander, K., and Haussler, D. (1994). Hidden Markov models in computational biology: Applications to protein modeling. J. Mol. Biol., 235:1501-1531.

