Working with HMMs

Given a sequence $x$ and an HMM, there tasks we would like to perform:

- Decoding: Find a sequence of states that explains $x$  
  - i.e. Viterbi
- Evaluation: Given an HMM, and $x$, Compute $p(x)$
- Learning: Update parameters $\theta$ of the model to maximize $p(x|\theta)$

Posterior Decoding

Sometimes, we wish to use the entire sequence of data to evaluate the probability $p(\pi_i = k)$. This requires making two passes on the data – one with the forward algorithm, and one with the backwards algorithm.

\[
p(\pi_i = k \mid x = x_1, \ldots, x_i, \ldots, x_m)
\]

\[
p(\pi_i = k \mid x) = p(\pi = k, x) / p(x)
\]  
//where $p(x)$ is the probability you get from the forward algorithm

\[
p(\pi_i = k, x) = p(\pi_i = k, x_1, \ldots, x_i) / p(x_{i+1}, \ldots, x_m \mid x_1, \ldots, x_i, \pi_i = k)
\]  
//where $p(x)$ is the probability you get from the forward algorithm

We can decompose the right side into the following:

\[
p(\pi_i = k, x_1, \ldots, x_i) = f_k(i)
\]  
, the value from the forward algorithm

\[
p(x_{i+1}, \ldots, x_m \mid x_1, \ldots, x_i, \pi_i = k) = b_k(i), \text{ which is the “backward” algorithm}
\]

The Backwards Algorithm

\[
b_k(i) = p(x_{i+1}, \ldots, x_m \mid x_1, \ldots, x_i, \pi_i = k)
\]

Initialization: $b_k(m) = 1$

Recursion: $b_k(i) = \Sigma_l (a_{kl}) (e_l) (x_{i+1}) (b_{l(i+1)})$

Decoding Options

- Viterbi: $\pi^*$: max $P(x, \pi)$ (most likely global explanation)
- Posterior decoding: $\hat{\pi}_i = \arg\max_k p(\pi_i = k \mid x)$
Learning
If we want to train a model from data, we need to make sure our model is small enough to train from this data. Otherwise, you get OVERFITTING of your model to the data.

Step 1: Fix states, semantics
Parameters: $\theta$
We need a parameter $\theta$ for each of the emission probabilities $e_k(b)$ and transition probabilities $\alpha_{kl}$

Now, we wish to maximize $\log P(x_1, x_2, \ldots, x_n | \theta)$
Since each sequence is assumed to be iid, this corresponds to maximizing the sum of log scores for each sequence:

$$= \sum_{j=1}^{n} \log (P(x_i | \theta))$$

MLE

Estimation when sequence of states is known:

For example a set of experimentally verified “true” CpG island and surrounding DNA. Loaded die, fair die, and casino guy telling you when he is switching a set of genes (“true”)

$A_{kl}$ = number of times we transition from state $k$ to state $l$ in the data
$E_k(b)$ = number of times state $k$ emits $b$ in the data

MLE parameters:

$$\alpha_{kl} = \frac{A_{kl}}{\sum_i A_{ki}}$$

$$e_k(b) = \frac{E_k(b)}{\sum_x E_k(x)}$$

This is the intuitive thing to do.

However, there is a problem. What is there is an event that occurs with small probability relative to the size of our data (ie, a transition that we believe occurs around 1 in every 10000 observations, but our data set is only 5000 long)? It is very possible that we observer a probability of 0 for this event. However, if probability is 0, then the model will never allow this event to occur, even if the event appears in subsequent training data.

Solution: Add a pseudocount, or a prior count. This assures that no probability will be 0, and the more we want our prior beliefs to influence the outcome, the larger we make these numbers.

$$\alpha_{kl} := \alpha_{kl} + \text{pseudocount}$$

$$e_k(b) := e_k(b) + \text{pseudocount}$$
Estimation when sequence of states is unknown:

**Baum-Welch and Viterbi Training**

**Baum-Welch algorithm:**
1. Start with some initial $\theta$: $a_{kl}, e_k(b)$
2. Estimate $A_{kl}$, $E_k(b)$
   - Expected $A_{kl}$, $E_k(b)$, given the current parameters $a_{kl}$, $e_k(b)$ and $x^1, \ldots, x^j$
3. Update $a_{kl}$, $e_k(b)$
4. Go to step 2.

\[ A_{kl} = P(\pi_i = k, \pi_{i+1} = l|x, \theta) \]
\[ = \frac{f_k(i) a_{kl} e_k(x_{i+1}) b_l(i+1)}{P(x)} \]
\[ A_{kl} = \sum_{x^i \in \text{seq}.} \frac{1}{P(x^i)} \sum_{i=1}^{\mid x^i \mid} f_k(i) a_{kl} e_k(x + 1) b_l(i + 1) \]

\[ E_k(b) = \sum_{x^j} \frac{1}{P(x^j)} \sum_{i=1, x^j=b}^{\mid x^j \mid} f_k(i) b_k(i) \]

**Viterbi Training:**
1. Start with $a_{kl}$, $e_k(b)$
2. For each $x^j$, run Viterbi to get $\pi^*_j$
3. Estimate $A_{kl}$, $E_k(b)$ using the path $\pi^*_j$
4. Calculate the new parameters
5. Go to step 2.

(Comment: In Baum-Welch, there is not update to get zeroes; however, in Viterbi Learning, you can have zeroes.)