Bioinformatics – Lecture Notes

Announcements

Remember: Project Proposals are due April 11.

Class 22 – April 4, 2002

A. Hidden Markov Models

1. Definitions

Example - Consider the example we talked about in class last time with the coins. However, this time instead of a 3 coins with a heads or tail outcome, assume that you have a 4 sided die with the letters A, C, T, G on the faces. You roll the die n-times for a sequence of length n.

Example – This example can be extended to protein sequences if we consider a 20-sided die with a different amino acids on each face. Alternatively, we can consider the 64 codons.

To see how Hidden Markov Models can be used, consider the following example:

The occasionally dishonest casino example - In a casino, the use a fair die most of the time, but occasionally they switch to a loaded die. The loaded die has probability 0.5 of a six and 0.1 for the numbers 1 to 5. Assume that the casino switches from a fair to a loaded die with probability 0.05 before each roll and the probability of switching back is 0.1. Then the switch between the dice is a Markov process. In each state the outcome of a roll has different probabilities. This is an example of a hidden Markov Model.

In this example, if we just see the outcome (emissions) of the die rolls, the state sequence is hidden. The *emission probabilities* ($e_k(b)$) are the probabilities of each outcome in a particular outcome b when in state k. The transition probabilities (a_{kl}) is the probability that the state will change from state k to state l. These can be written as follows:

$$a_{kl} = \Pr[\pi_i = l \mid \pi_{i-1} = k]$$
$$e_k(b) = \Pr[x_i = b \mid \pi_i = k]$$

We can write down the joint probability of an observed sequence x and a state sequence π:

$$\Pr[x,\boldsymbol{p}] = a_{0p_1}\prod_{i=1}^{L} e_{p_i}(x_i)\, a_{p_i p_{i+1}} \quad (*)$$

2. The Viterbi Algorithm: Computing the most probable state path

   Looking at the emitted sequence, we need to have a way to determine the most likely sequence of states that yields the data we are considering, ie.

$$\boldsymbol{p}^* = \arg\max_{p} \Pr[x,\boldsymbol{p}]$$

   The most probable path π^* can be found recursively. Let the probability of the most probably path ending in state k with observation i, $v_k(i)$, be know for all the states. These probabilities can be calculated for observation x_{i+1} by

$$v_l(i+1) = e_l(x_{i+1})\max_k (v_k(i)a_{kl})$$

   with the initial condition v_0(beginning state) = 1. After running through iteration for all observations in the sequence, the actual state sequence can be found by using the traceback. (see the algorithm in Durbin et al or Clote and Backhofen)

   On a more practical note, often when we are taking the product of many probabilities the numbers get really small. This can cause underflow errors on a computer. To remedy this, use the $\log(v_l(i))$ instead.

   Figure 3.5 (in Durbin et al) shows data generated in a simulation, the actual die used and the prediction by the Viterbi algorithm. Notice that while it is not exactly correct, it is close.

3. Probability of a sequence in a Hidden Markov Model

   Earlier in the semester, we showed how to calculate the probability of a sequence derived from a Markov Chain

$$\Pr[x] = \Pr[x_1]\prod_{i=2}^{L} a_{x_{i-1}x_i}$$

   where $a_{x_{i-1}x_i} = \Pr[x_i \mid x_{i-1}]$. We want to be able to do the same for the Hidden Markov Model. This is slightly more complicated because many different state paths can give rise to the same sequence x. This requires that we sum up the probabilities to obtain the full probability of sequence x.

$$\Pr[x] = \sum_{p} \Pr[x,\boldsymbol{p}]$$

   Because the number of paths increases exponentially with the length of the sequence, enumerating all paths in too computationally expensive. Instead, we can use equation (*) above for the most probable path π^* as a approximation for Pr[x]. Alternatively we can use a dynamic programming method similar to the Viterbi algorithm called the *forward algorithm*. The term analogous to *Viterbi variable* $v_k(i)$ in the forward algoritm is

$$f_k(i) = \Pr[x_1 ... x_i, \boldsymbol{p}_i = k]$$

is the probability of the observed sequence up until and including x_i, with the requirment that $\pi_i = k$. The recursion equation is

$$f_l(i+1) = e_l(x_{i+1}) \sum_k f_k(i) a_{kl}$$

(see the algorithm in Durbin et al or Clote and Backhofen)

Similar to the Viterbi algorthm, one might work in log space to avoid underflow errors.

4. The backward algorithm and posterior state probabilities

Sometimes we might want to know the most probable state for an observation x_i. This can be re-phrased as the probability that observation x_i came from state k given the observed sequence ($\Pr[\pi_i = k \mid x]$). This is called the *posterior probability.*

To do this we use the *backward algorithm* which is a bit indirect. First, the probability of producing the entire observed sequence with the ith symbol being produced by state k, ie.

$$\begin{aligned} \Pr[x, \boldsymbol{p}_i = k] &= \Pr[x_1 ... x_i, \boldsymbol{p}_i = k] \Pr[x_{i+1} ... x_L \mid x_1 ... x_i, \boldsymbol{p}_i = k] \\ &= \Pr[x_1 ... x_i, \boldsymbol{p}_i = k] \Pr[x_{i+1} ... x_L \mid \boldsymbol{p}_i = k] \end{aligned} \qquad (**)$$

because everything after k only depends on the state at k. The first term is the $f_k(i)$ from the forward algorithm. The second term is

$$b_k(i) = \Pr[x_{i+1} ... x_l \mid \boldsymbol{p}_i = k]$$

which is obtained by backward recursion starting at the end of the sequence. (see backward algorithm Durbin et al or Clote and Backofen)

Once we have computed $f_k(i)$ from the forward algorithm, $b_k(i)$ from the backward algorithm and Pr[x] from either, we re-write equation (**) as

$$\Pr[x, \boldsymbol{p}_i = k] = f_k(i)\, b_k(i)$$

then we can calculate

$$\Pr[\boldsymbol{p}_i = k \mid x] = \frac{\Pr[x, \boldsymbol{p}_i = k]}{\Pr[x]}$$

$$= \frac{f_k(i)\, b_k(i)}{\Pr[x]}$$

by the definition of conditional probability. (see Fig 3.6)

5. Posterior decoding

   When many different paths have close to the same probability as the most probable one, it is not reasonable to only choose the most probable one. One approach is to define a second state sequence

$$\hat{\boldsymbol{p}} = \arg\max_{p} \Pr[\boldsymbol{p}_i = k \mid x]$$

   This state sequence might be more appropriate when we are interested in the state assignment at a particular point i, rather than the complete path. In Figure 3.7 (Durbin et al), the dishonest casino example is used with the probability of switching from a fair to loaded die is 0.01. In this case the Viterbi alorgithm never visits the loaded die state. However, if we look at the posterior probabilities, it is clear where the loaded die is used.

B. Parameter Estimation for Hidden Markov Models

When we are determining a Hidden Markov Model, the first step is to define the model in the first place. In our example, we know the form of the model. Sometimes we do not know this. There are two parts to designing a model: 1) defining the model structure or topology and 2) assigning the parameter values for the emission and transition probabilities.

To do this we need to have a set of example sequences called the *training set* which we will call $x^1,\ldots,x^n$. These are assumed to independent so that the joint probability of all sequences is simply the product or the probabilities of the individual sequences. Since this is a product, it makes sense to work in log space.

1. Estimating when the state sequence is known

   When the paths are all know, we can count the number of times that each particular transition or emission is used in the training set. We can call these A_{kl} and $E_k(b)$, respectively. We can compute the maximum likelihood estimators for a_{kl} and $e_k(b)$ by

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \qquad \text{and} \qquad e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')} \qquad (***)$$

In order for this to work, we have to have enough data in the training set so that our problem is not underdetermined.

2. Baum Welch and Viterbi training: Estimation when the paths are unknown

Definition – The *Baum-Welch Score* is the likelihood $L_O(M)$ that a model M generates O is defined by

$$\begin{aligned} L_O(M) &= \Pr[O \mid M] \\ &= \sum_{p} \Pr[O, \boldsymbol{p} \mid M] \\ &= \sum_{p} \Pr[O \mid \boldsymbol{p}, M] \Pr[\boldsymbol{p} \mid M] \end{aligned}$$

Definition – The *Viterbi Score* is the same as the Baum Welch Score except with the maximum in place of sum over all paths of states

a) Baum-Welch Method

This method is a very common method. It estimates A_{kl} and $E_k(b)$ by considering the probable paths for the training sequences using the current values of a_{kl} and $e_k(b)$. Then Eq. (***) is used to derive new values of the a's and e's. Clote and Backofen show the proof that the overall log likelihood increases with each iteration and that the process converges to a local maximum. One must realize that since it is a local maximum, the local maximum you end up in is strongly dependent on the initial values of the parameters.

b) Expectation Maximization
c) Baldi-Chauvin Gradient Descents

C. Applications

a) Multiple sequence alignment
b) Protein motifs
c) Eukaryotic DNA promoter regions