

Lecture 13: March 4

*Lecturer: Gene Myers**Scribe: Petter Risholm*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

13.1 Project

13.1.1 Some journals to get inspiration/ideas from

Several of the recent proceedings in Bioinformatics can now be found at the front desk in Soda Hall. These papers can give inspiration and ideas for term papers. Next week a term paper proposal is due, so it would be advisable to flip thru these proceedings this week. Some of the journals one can look at are:

- RECOMB conference
- International Conference on System and Molecular Biology (ISMB)
- Genome Informatics Workshop.
- There is also a book, Comparative Genomics by Sankoff and Nadeau, on reserve at the front desk.

13.1.2 Possible Project Topics

- Some of the most interesting current research is focused on Singular Nucleotide Polymorphisms (SNPs) and Haplotypes. SNPs are sites in the DNA sequence where individuals differ at a single DNA base. Haplotypes is the pattern of SNPs on a chromosome in a block. Some of the tools used in this research are clustering methods and statistical analysis.
- Expression analysis
- Structure analysis
- Evolutionary trees
- Text extraction
- Motif findings
- Comparative Genomics (how do you line up genomes). Decode genome by looking at all genomes simultaneously.
- Data Management Systems

13.1.3 Some topics which interests Professor Myers

- Comparative genomics: Aligning genomes and gene finding.
- Extreme assembly.
- Open source assembler.
- Image processing of in situ hybridization data. Take several pictures of all genes and the staining pattern they produce by injecting RNA into the gene. Collect these pictures and make a big database on which we can do datamining to get information on how the genes co-express themselves.

13.2 BLAST (Continuing from last lecture)

Look at the slides which were used in class to get more indepth information. Blast is a heuristic sequence searching package for finding high scoring local alignments between a query sequence and a database. BLAST relaxes the concept of a local alignment by not allowing indels. Indels in protein sequences changes the functionality of proteins so it is a reasonable heuristic. The scoring scheme can be based on both BLOSSOM and PAM scoring matrices. BLAST starts by finding short seed sequences which score high. This is done by generating the neighborhood $N_k(T)$ and finding its score. Using a Dayhoff (PAM) scoring scheme, every neighborhood match larger than a fixed length with a higher score then some threshold will be marked and used as seeds in the next phase. BLAST then goes through the database to find all these matches and extends them as an ungapped alignment in both directions stopping at the maximum scoring extension. If the score drops below a set value x stop. If it has score higher than a preset threshold s , report it. We are going to miss about 1 per thousand hits in theory. In practice you rarely miss anything you are after.

13.3 Hidden Markov Models (HMM)

13.3.1 Markov Models

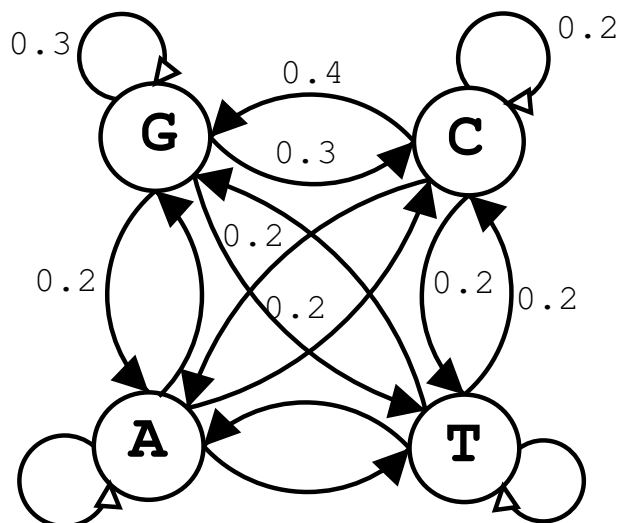


Figure 13.1: Markov model

Figure 13.1 shows a Markov model for a DNA chain. We can easily find the probability of seeing a DNA sequence X by tracing the sequence through the diagram and multiplying together the transition probabilities. The Markov model also indicates that the sum of the transition probabilities out of a node should sum to 1.

13.3.2 HMM's

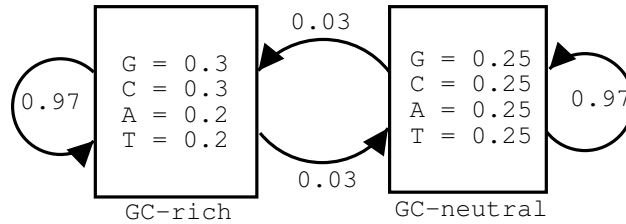


Figure 13.2: Hidden Markov Model

The difference between a Markov model and a Hidden Markov Model is that in an HMM we know the emissions, but we don't know which state the character was emitted from (we don't know the state sequence of the pattern). In the Markov model we know both the emissions and which state emitted the emission. Figure 13.2 shows the Markov model in 13.2 (the probabilities in the two figures are not correlated) as an HMM.

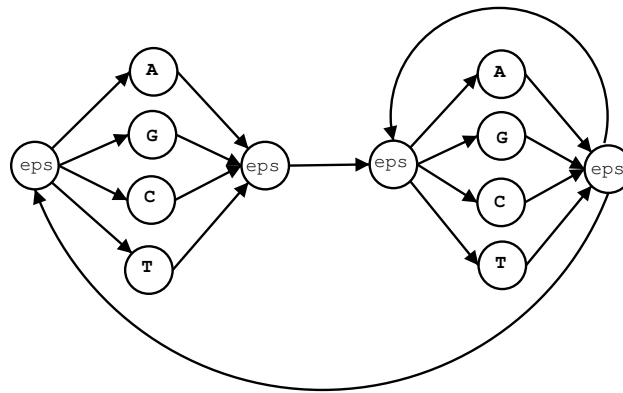


Figure 13.3: HMM modeled by a Markov chain

The HMM can easily be written as a Markov Model. Figure 13.3 shows how the HMM in figure 13.2 can be written as a HMM where we have introduced some (epsilon) silent states. Let us say we have a long genome, and we know that there are some regions along the genome which are GC-rich and others which are GC-neutral. See figure 13.4. This case is modeled as a HMM in figure 13.2 where we have two states. One state denotes being in the GC-rich region while the other tells us that we are in the GC-neutral region.

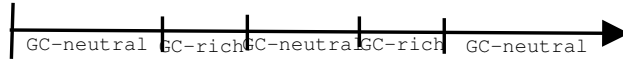


Figure 13.4: Genome with GC rich and GC neutral regions

13.3.3 HMM as a FSA

An HMM can also be looked at as a Finite State Automaton (FSA):

$$\langle V, E, \rho, \lambda, \theta, \Phi \rangle .$$

The transition probabilities are defined as $\rho : E \rightarrow \mathfrak{R}$ such that $\sum_u (v \rightarrow u) = 1$. λ , which are the emission probabilities are defined as: $\lambda : (V, \Sigma) \rightarrow \mathfrak{R}$ such that $\sum_a (v, a) = 1$ where in the case of a DNA sequence a is a letter. We do not use any epsilon states except that θ is silent. Let us look at an example where we would like to know the probability of emitting the string X :

$$Prob(X, \pi : \theta \rightarrow \pi_1 \rightarrow \pi_2 \rightarrow \dots \rightarrow \pi_{|X|}) = \prod_{i=1}^{|X|} \rho(\pi_{i-1} \rightarrow \pi_i) \lambda(\pi_i, a_i),$$

where $|X|$ denotes the length of the string X . We now know that if we are given a state sequence we will know the character sequence generated by the HMM.

13.4 Viterbi Algorithm

Usually we do not have the state sequence which generated a particular string, but often this is just what we are interested in. The Viterbi algorithm is a way of finding the most probable path through the HMM that would emit the sequence of characters we have observed. Given X and an HMM, we would like to know the most probable path through the HMM that would emit X . In mathematical terms:

$$\pi^* = \operatorname{argmax}_{\pi} P(x, \pi).$$

The most probable path π^* can be found recursively with simple dynamic programming. Let $P(i, s) = \max$ probable path generating x_1, x_2, \dots, x_i and ending in state s . This can be written as:

$$[\max_{t \rightarrow s} P(i-1, t) \rho(t \rightarrow s)] \lambda(i, a_i)$$

The best path is found by traceback or any of the other techniques we have discussed in class. Let θ be the start state and Φ be the end state, then the score of the best path is $P(X, \pi^*) = P(n, \Phi)$.

13.5 Forward and Backwards Algorithms

To get the forward path we do the following:

$$P(X|HMM) = \sum_{\pi \in \theta \Rightarrow \Phi} P(X|\pi) = F(n, \Phi)$$

The score for a particular node in the forward recurrence is:

$$F(i, s) = \sum_{\pi \in \theta \rightarrow^* s} P(x_1, x_2, \dots, x_i | \pi) = \lambda(s, a_i) \sum_{t \rightarrow s} P(i-1, t) \rho(t \rightarrow s)$$

The reverse recurrence is:

$$R(i, s) = \sum_{\pi \in s \rightarrow^* \Phi} P(x_i, x_{i+1}, \dots, x_n | \pi)$$

. In the reverse recurrence, s is silent. We can now get the posterior probabilities of the different states:

$$\frac{P(\pi_i = s | x) = F(i, s) R(i, s)}{F(n, \Phi)}$$

13.6 Parameter estimation

Let us say we have developed an HMM, but we do not know the values of the parameters. To get an estimate of the parameters we need a training set, i.e. a whole set of DNA sequences which we can train the HMM with. Given a topology: $V, \epsilon, \theta, \phi$ and a training set, we would like to know ρ and λ :

$$\langle V, E, -, -, \theta, \phi \rangle .$$

We start off by setting all the parameters to an arbitrary value.

While we have more training data **do**:

Fit all x_i to the HMM.

Set parameters based on the fit.

We can also look at it another way. As we train the HMM we record every edge taken thru the HMM. When all the training sets have run through the HMM, we look at every state and all of the edges that leave the states. Since we have recorded all fits we can calculate the probability of taking the different edges out of a state.

$$\rho(s \rightarrow t) = \frac{\#(s \rightarrow t)}{\sum_l \#(s \rightarrow l)}$$

The emission probabilities can be estimated this way:

$$\lambda(s, a) = \frac{\#(s, a)}{\sum_b \#(s, b)}$$

where a denotes a particular character and b is any character. Parameters can also be calculated, in a bit more sophisticated way, by using the forward and backward recurrences. Doing it this way gives us the expected number of times each transition and emission is used given the training sequences. This is a gradient descent algorithm.

References

[DEKM98] R. DURBIN, S. EDDY, G. MITCHINSON and A. KROGH, Biological sequence analysis, *Cambridge university press* (1998).