**18.417 Introduction to Computational Molecular Biology**

Lecture 17: November 4, 2004 — Scribe: Han-Pang Chiu

Lecturer: Ross Lippert — Editor: Russ Cox

# Hidden Markov Models

## The CG island phenomenon

The nucleotide frequencies in the human genome are

| $A$ | $C$ | $T$ | $G$ |
|---|---|---|---|
| 29.5 | 20.4 | 20.5 | 29.6 |

Frequencies of particular nucleotide pairs tend to be steady across the human genome. One exception to this is $CG$, which appears with expected frequency in and around genes but hardly appears in junk DNA. This occurs because $CG$ in junk DNA is typically methylated, and methylated $CG$ tends to mutated into $TG$. *$CG$ islands* are runs of DNA where $CG$ has much higher-than-normal frequency, indicating the likely presence of genetic data.

Heuristics like sliding window searches can be used to locate CG islands, but instead, we'd like to analyze the sequence to find what the most likely CG islands are.

We can model the CG island problem as the "Casino with Two Coins" problem.

## Casino With Two Coins

In the Casino with Two Coins problem (the book refers to it as the Fair Bet Casino problem), the casino has two coins, one fair and one biased. The input to the problem is a binary sequence, the results of the coin tosses. The output is the most likely "tagging sequence" indicating which coin was used for each toss.

Given a particular tagging sequence $S_1, S_2, \ldots, S_k$ and a corresponding data sequence $X_1, X_2, \ldots, X_k$, we define the probability of the data given the tagging model as

$$\Pr\{\text{data} \mid \text{tagging}\} = \prod_{i=1}^{n} \Pr\{X_i \mid S_i\}.$$

We will write $\Pr\{X_i \mid S_i\}$ as $\Pr\{S_i \rightsquigarrow X_i\}$, read $S_i$ emits $X_i$.

Not all tagging sequences are created equal: some might be more likely than others. For example, perhaps we know that the dealer in the casino changes coins infrequently. To model this, we also take as input a matrix specifying the probability of moving from state $S$ to state $S'$, written $\Pr\{S \to S'\}$.

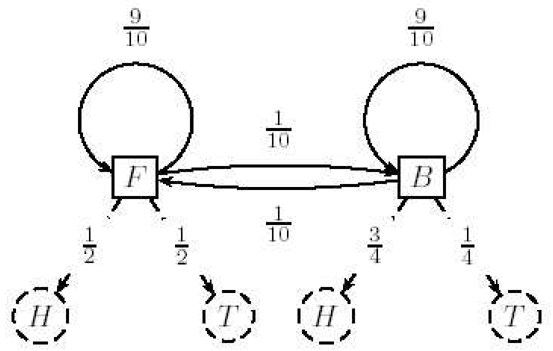
The probability of a tagging sequence is given by

$$\Pr\{\text{tagging} \mid \text{transition probabilities}\} = \Pr\{\text{start in } S_1\} \prod_{i=1}^{n-1} \Pr\{S_i \to S_{i+1}\}.$$

The individual probabilities used so far make up the *hidden Markov model* or HMM, so named because it is a Markov model (a state machine) where the states (the tagging) cannot be directly observed. For example, if the dealer has a $\frac{1}{10}$ chance of changing coins at each step, and if the biased coin is $\frac{3}{4}$ heads, then the Markov model is given by:

$$\Pr\{F \to F\} = \Pr\{B \to B\} = \frac{9}{10} \qquad \Pr\{F \to B\} = \Pr\{B \to F\} = \frac{1}{10}$$
$$\Pr\{F \rightsquigarrow H\} = \frac{1}{2} \qquad \Pr\{F \rightsquigarrow T\} = \frac{1}{2}$$
$$\Pr\{B \rightsquigarrow H\} = \frac{3}{4} \qquad \Pr\{B \rightsquigarrow T\} = \frac{1}{4}$$

We might depict such a model graphically as:



## Notation of Hidden Markov Model

Typically we introduce some shorthands for the probabilities that make up the HMM. Since they are actually complete probability distributions, the probabilities for each set of outcomes must sum to one.

$$
\begin{aligned}
T &= [t_{ij}] = [\Pr\{i \to j\}]; \sum_j t_{ij} = 1 \\
E &= [e_{ix}] = [\Pr\{i \rightsquigarrow x\}]; \sum_x e_{ix} = 1 \\
\pi &= [P_i] = [\Pr\{\text{model starts in state } i\}]; \sum P_i = 1
\end{aligned}
$$

Sometimes $\pi$ is omitted and it is assumed to be the left eigenvector of $T$ with eigenvalue 1. We know $\pi$ has such an eigenvector because it has a trivial right eigenvector (1) with eigenvalue 1.

It is also convenient to introduce the diagonal matrix $D_x$ whose diagonal has $E_{ix}$ (for some fixed $x$) as its entries.

Under the new notation, our first problem is to compute the likelihood of a particular observation $X$ given HMM parameters $T$, $E$, and $\pi$. Combining the equations we gave earlier, we have that

$$\Pr\{X \mid T, E, \pi\} = \pi D_{x_1} T \ldots T D_{x_n} 1.$$

It is conventional to split this computation into two halves $f(k,i)$ and $b(k,i)$. The forward half $f(k,i)$ is the probability of ending up in state $i$ after $k$ moves, and the backward half $b(k,i)$ is the probability of ending up in state $i$ when $k$ moves away the end of the sequence.

That is,

$$
\begin{aligned}
[f(k,i)] &= [\Pr\{x_1, \ldots, x_k, s_k = i \mid T, E, \pi\}] = \pi D_{x_1} T \ldots T D_{x_k} \\
[b(k,i)] &= [\Pr\{s_k = i, x_{k+1}, \ldots, x_n \mid T, E, \pi\}] = T D_{x_{k+1}} \ldots T D_{x_n} 1
\end{aligned}
$$

Combining these we get the original probability:

$$\Pr\{X \mid T, E, \pi\} = [f(k,i)][b(k,i)] = \sum_i \Pr\{\text{paths going through } i\}.$$

This tells us that

$$\Pr\{s_k = i \mid T, E, \pi, X\} = \frac{f(k,i)b(k,i)}{\sum_i f(k,i)b(k,i)},$$

which suggests an obvious method for computing the most likely tagging: simply take the most likely state for each position.

Unfortunately, this obvious method does not work. Really the most likely sequence is $\arg\max_S \Pr\{S \mid T, E, \pi, X\}$. To find the maximum sequence, we can use a dynamic programming algorithm due to Viterbi. The necessary state is an analog to $f(k,i)$, except that instead of summing over all incoming arrows we can only select the best one.

The Viterbi dynamic programming algorithm can be viewed as a tropicalization of the original $f$ and $b$ functions. This is examined in detail in the next lecture.

## Training an HMM

Much of the time we don't know the parameters of the HMM, so instead we *train* it on inputs, trying to derive parameters that maximum the likelihood of seeing the observed outputs. That is, given one or more sequences $X$, find $T, E, \pi$ that maximize $\Pr\{X \mid T, E, \pi\}$.

Unfortunately, this problem is NP-hard, so we'll have to content ourselves with heuristic approaches.

### Expectation Maximization (EM)

One very effective heuristic approach for training an HMM is called *expectation maximization*, due to Baum and Welch. It is a local gradient search heuristic that follows from techniques for gradient search on constrained polynomials. It may not find a global maximum, so typically it is run a handful of times, starting from different initial conditions each time.

We will assume that $\pi$ is the eigenvector of $T$ as discussed above, so we need only worry about $E$ and $T$. We start with some initial $E$ and $T$, usually chosen randomly. Then we compute a new $\hat{E}$ and $\hat{T}$ using the following rules:

$$\begin{aligned}
\Pr\{i \rightsquigarrow x\} &= \frac{1}{C} \sum_{k:x_k=x} \Pr\{\text{path has } s_k = i\} \\
&= \frac{1}{C} \sum_{k:x_k=x} f(k,i)b(k,i) \\
\Pr\{i \to j\} &= \frac{1}{C} \sum_{k} \Pr\{\text{path has } s_k = i,\ s_{k+1} = j, \text{ and } j \text{ emits } x_k\} \\
&= \frac{1}{C} \sum_{k} \left(f(k,i) \Pr\{i \to j\} \Pr\{j \rightsquigarrow x_k\}\, b(k+1,j)\right)
\end{aligned}$$

This recomputing is iterated until the $\hat{E}$ and $\hat{T}$ converge.

In general, expectation maximization sets $\Pr\{\hat{a}\}$ in proportion to the number of times event $a$ needs to happen compared to other events happening. The $\frac{1}{C}$ is a normalization term to make the probabilities sum to 1 when necessary.