Algorithms in Bioinformatics

Hidden Markov Models

Recap

- HW4
- HW2 is graded
- HW5 & essay are up

Chapter 11: Hidden Markov Models

Least frequent dinucleotide: CG

- C is easily methylated, & then a tendency to mutate to a T.

However, this is suppressed around genes in CG-Islands.

An important problem is to identify these islands.

Model this probabilistically:

Consider sliding windows of some length, & calculate some probability of it being an island

First, we consider a related "toy" problem

"Fair Bet Casino":

A crooked dealer is flipping a coin,
- either fair, or biased one that outputs H with probability 3/4

The dealer needs to be sneaky, so only changes coins w/ probability 1/10.

Goal: Try to figure out if he's using a fair or biased coin:

Fair Bet Casino Problem:
Given a sequence of coin tosses, determine when the dealer used a fair coin and when he used a biased coin.

Input: A sequence $x = x_1\, x_2\, x_3 \ldots x_n$ of coin tosses (either H or T) made by two possible coins (F or B).

Output: A sequence $\pi = \pi_1\, \pi_2\, \pi_3 \cdots \pi_n$, with each π_i being either F or B indicating that x_i is the result of tossing the fair or biased coin, respectively.

Unfortunately, doesn't quite make sense...

Consider if the dealer never changes coins first.

Fair coin: $P(H) = \frac{1}{2}$, $P(T) = \frac{1}{2}$ ($P(1)$, $P(0)$)

Biased coin: $P(H) = \frac{3}{4}$, $P(T) = \frac{1}{4}$

Let $x = x_1 x_2 x_3 \cdots x_n$

be the sequence of flips

$P(x \mid \text{fair coin}) =$

$$\left(\frac{1}{2}\right)\left(\frac{1}{2}\right)\cdots\left(\frac{1}{2}\right) = \left(\frac{1}{2}\right)^n = \frac{1}{2^n}$$

If k heads are in x,

$P(x \mid \text{biased coin}) =$

$$\left(\frac{1}{4}\right)^{n-k}\left(\frac{3}{4}\right)^{k} = \frac{3^k}{4^k}\cdot\frac{1}{4^{n-k}} = \frac{3^k}{4^n}$$

(T: $n-k$, H: k)

Idea: if $P(x \mid \text{fair}) > P(x \mid \text{biased})$, then it probably was a fair coin

If we solve for k:

$$4^n \cdot \frac{1}{2^n} \geq \frac{3^k}{4^n} \cdot 4^n$$

$$\frac{2^{2n}}{2^n} \geq 3^k \Rightarrow 2^n \geq 3^k$$

$$\log_2 2^n \geq \log_2 3^k$$

$$n \geq k \log_2 3$$

So: if $k < \frac{n}{\log_2 3}$, then most likely a fair coin.

However, the dealer changes coins!

Log-odds ratio:

$$\log_2 \frac{P(x \mid \text{fair coin})}{P(x \mid \text{biased coin})}$$

$$= \sum_{i=1}^{k} \log_2 \frac{p^+(x_i)}{p^-(x_i)}$$

$$= n - k \log_2 3$$

(where $p^+(x_i)$ = prob of unbiased
$p^-(x_i)$ = prob of biased)

So: - slide a window of some length along H/T sequence
- if it falls below 0, probably biased
- if above 0, probably fair

Hidden Markov Models:

An abstract machine w/ability to produce output using coin tossing.

Our machine is in one of k "hidden" states.

Each step, it decides

- What state will I move to next?
- What symbol from Σ (an alphabet) will I output?

Essentially, will decide randomly, but using a biased / distribution

(Observer can see output, not states.)

Goal: Infer states

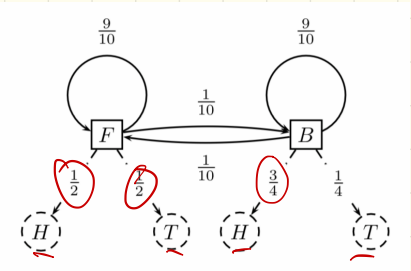
Formal dfn: M (a HMM) is described by

- Σ, an alphabet (earlier H or T)
- Q, a set of states, each of which outputs symbols (w/ some probability)
- a $|Q| \times |Q|$ matrix A
  $A[k][l]$ = probability of going to state l from state k
- E: a $|Q| \times |\Sigma|$ matrix
  $E[k][b]$ = probability of outputting b from state k

Ex: Fair bet casino HMM

$\Sigma = \{H, T\}$

$Q = \{F, B\}$



$A[F][F] = A[B][B] = 9/10$

$A[F][B] = A[B][F] = 1/10$

$E[F][H] = 1/2$

$E[B][H] = 3/4$

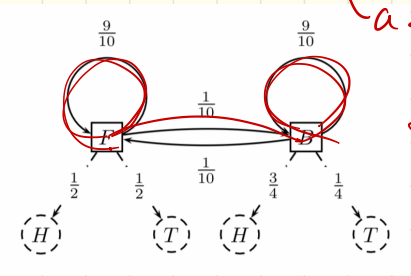
$E[F][T] = 1/2$

$E[B][T] = 1/4$

A path in a HMM is just a sequence of states,

$\pi = \pi_1 \pi_2 \pi_3 \cdots \pi_n$

↑ a single state from Q



$\pi' = FBF$ ↑ $n=3$

ie $\pi = FFFBBBBFFF$ ↑ $n=10$ here

Suppose output was:

$x = THTHHHTHTTH$

Can line up probabilities:

$$\begin{array}{cc} & \begin{matrix} T & H & T & H & H & H & T & H & T & T & H \end{matrix} \\ x & \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \\ \pi = & \begin{pmatrix} F & F & F & B & B & B & B & B & F & F & F \end{pmatrix} \\ P(x_i|\pi_i) & \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{3}{4} & \frac{3}{4} & \frac{3}{4} & \frac{1}{4} & \frac{3}{4} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{pmatrix} \end{array}$$

↑ probability x_i was the output of state π_i

(These were in E.)

Can also look at $p(\pi_i \to \pi_{i+1})$:

the probability of going from state π_i to π_{i+1}

(These were in matrix A).

Get larger view:

$$\begin{array}{c} x \\ \pi \\ P(x_i|\pi_i) \\ P(\pi_{i-1} \to \pi_i) \end{array} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ F & F & F & B & B & B & B & B & F & F & F \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{3}{4} & \frac{3}{4} & \frac{3}{4} & \frac{1}{4} & \frac{3}{4} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{2}{2} & \frac{9}{10} & \frac{9}{10} & \frac{1}{10} & \frac{9}{10} & \frac{9}{10} & \frac{9}{10} & \frac{9}{10} & \frac{1}{10} & \frac{9}{10} & \frac{9}{10} \end{pmatrix}$$

↖ Pick 1st state randomly

Then the probability of generating x through path π is:

$$\left(\frac{1}{2} \cdot \frac{1}{2}\right)\left(\frac{1}{2}\right)\left(\frac{9}{10}\right)\cdots$$

$$\approx 2.66 \times 10^{-6}$$

(Note: this assumed we knew π and observed x.)

If we only know:

$X = \text{THTHHHTHTTH}$ (not π)

- how good is a particular π?
- can we construct the most likely π?

For above x, FFFBBBFFFFF is actually better:

x		0	1	0	1	1	1	0	1	0	0	1
π		F	F	F	B	B	B	F	F	F	F	F
$P(x_i \mid \pi_i)$	$=$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
$P(\pi_{i-1} \to \pi_i)$		$\frac{1}{2}$	$\frac{9}{10}$	$\frac{9}{10}$	$\frac{1}{10}$	$\frac{9}{10}$	$\frac{9}{10}$	$\frac{1}{10}$	$\frac{9}{10}$	$\frac{9}{10}$	$\frac{9}{10}$	$\frac{9}{10}$

E's

A's

$$\text{prob} = \left(\frac{1}{2} \times \frac{1}{2}\right)\left(\frac{1}{2} \cdot \frac{9}{10}\right) \cdots$$

$$\approx 3.54 \times 10^{-6}$$

(was 2. ...)

π is called "hidden".

Goal:

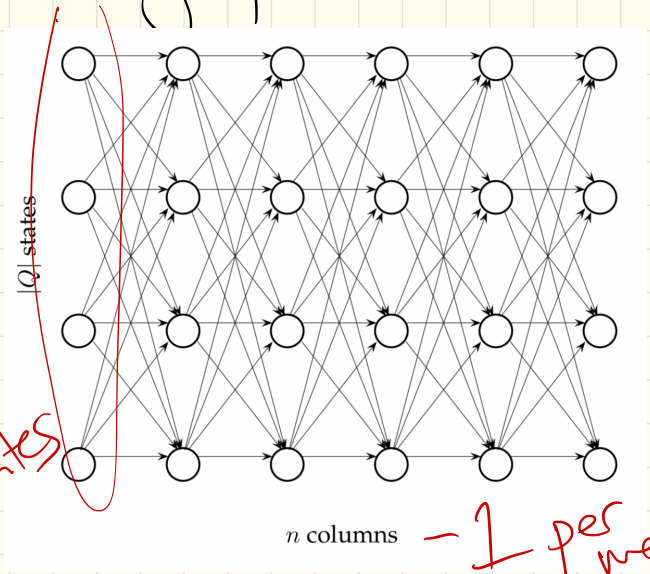
Decoding Problem:
Find an optimal hidden path of states given observations.

Input: Sequence of observations $x = x_1 \dots x_n$ generated by an HMM $\mathcal{M}(\sum, Q, A, E)$.

Output: A path that maximizes $P(x|\pi)$ over all possible paths π.

[Viterbi 1967] gave an approach to solve.

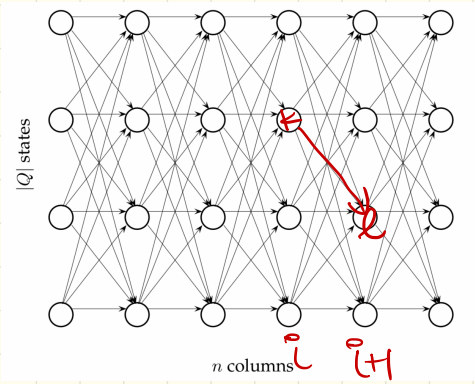
Build a graph:



all states

1 per measurement

Idea (in more detail):



Goal: Have weight on edges, so product of the weights for $\pi = \pi_1 \ldots \pi_n$ $= P(x|\pi)$

How?

- Set edge from (k,i) to $(l,i+1)$ to be $= \rightarrow E[l][x_{i+1}] \times A[k][l]$

output prob. transition prob.

Why? Compare $p_{k,i}$ to $p_{l,i+1}$:

$$
\begin{aligned}
p_{l,i+1} &= \prod_{j=1}^{i+1} e_{\pi_j}(x_j) \cdot a_{\pi_{j-1},\pi_j} \\
&= \left(\prod_{j=1}^{i} e_{\pi_j}(x_j) \cdot a_{\pi_{j-1},\pi_j} \right) \cdot \left(e_{\pi_{i+1}}(x_{i+1}) \cdot a_{\pi_i,\pi_{i+1}} \right) \\
&= p_{k,i} \cdot e_l(x_{i+1}) \cdot a_{kl} \\
&= p_{k,i} \cdot \text{weight of edge from } (k,i) \text{ to } (l,i+1)
\end{aligned}
$$

This changes the decoding problem into a longest path problem (in a DAG).

(ie Manhatten distance)

Then - use dynamic programming longest path solution from chapter 6.

Note: Now we need to multiply weights (& not add) to get length.

Trick: take log!

$$\log(ab) = \log a + \log b$$

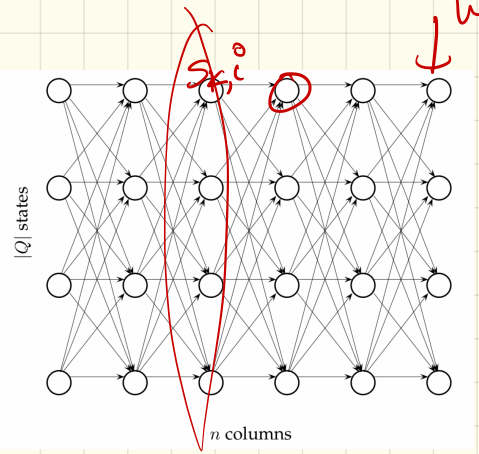
Dynamic program:

For any state l:

$$\begin{aligned} s_{l,i+1} &= \max_{k \in Q}\{s_{k,i} \cdot \text{ weight of edge between } (k,i) \text{ and } (l,i+1)\} \\ &= \max_{k \in Q}\{s_{k,i} \cdot a_{kl} \cdot e_l(x_{i+1})\} \\ &= e_l(x_{i+1}) \cdot \max_{k \in Q}\{s_{k,i} \cdot a_{kl}\} \end{aligned}$$

Pick best weight here

Data Structure:



Runtime: $O(|Q| \times n)$ table spots

Each takes $O(|Q|)$ time to fill in

$\Rightarrow O(|Q|^2 n)$

HMM Parameter estimation

Another problem:
probabilities are really often not known!

Need to be estimated from data.

So: we know Q & E (the states), but not E or A.

Let θ be a vector combining unknown transition & estimation (E & A) probabilities.

Given an observed x, let $P(x|\theta)$ = max probability of x given θ

Goal: $\max_{\theta} P(x|\theta)$

Instead of a single x, we often get a set of m training sequences $x^1, \ldots, x^m$

Goal is then:

$$\max_{\theta} \prod_{i=1}^{m} P(x^i | \theta)$$

Hard! (optimization in a multi-dimensional space)

Heuristics & estimates:

If we knew states, & a_{kl} = # of transitions from k to l

e_{kb} = # times b outputted from state k

then set

$$A[k][l] = \frac{a_{kl}}{\sum_{q \in Q} a_{kq}}$$

$$E[k][b] = \frac{e_{kb}}{\sum_{\sigma \in \Sigma} e_{k\sigma}}$$

Note: We don't know States!

- Usually start with a guess
- Compute empirical estimates for transmission & emission probabilities
- Solve problem to get a less wild guess

Ex: Baum-Welch algorithm

- GLIMMER
- GENSCAN

many others based on similar iterative, local improvement strategy

Next time — some applications of HMM

or on to Randomization