Lecture 22: Hidden Markov Models

Chapter 11

Dinucleotide Frequency

- Consider all 2-mers in a sequence
  {AA,AC,AG,AT,CA,CC,CG,CT,GA,GC,GG,GT,TA,TC,TG,TT}
- Given 4 nucleotides:
  each with probability of occurrence is ~ ¼.
  Thus, one would expect that the probability of occurrence of any given dinucleotide is ~ 1/16.
- However, the frequencies of dinucleotides in DNA sequences vary widely.
- In particular, CG is typically underrepresented (frequency of CG is typically < 1/16)

Example

- From a 291829 base sequence

AA	0.120214646984	GA	0.056108392614
AC	0.055409350713	GC	0.037792809463
AG	0.068848773935	GG	0.043357731266
AT	0.083425853585	GT	0.046828954041
CA	0.074369148950	TA	0.077206436668
CC	0.044927148868	TC	0.056207766218
CG	0.008179475581	TG	0.063698479926
CT	0.066857875186	TT	0.096567155996

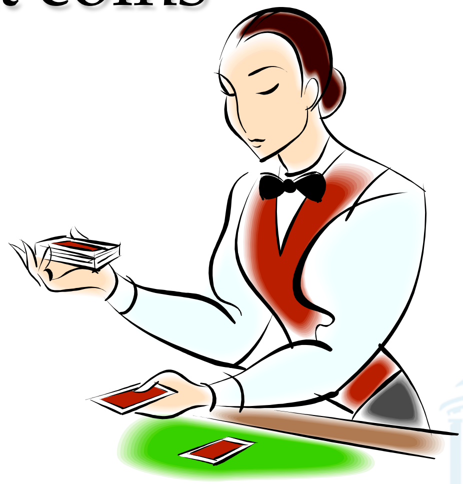
- Expected value 0.0625
- CG is 7 times smaller than expected

Why so few CGs?

- CG is the least frequent dinucleotide because C in CG is easily *methylated*. And, methylated Cs are easily mutated into Ts.
- However, methylation is suppressed around genes and transcription factor regions
- So, CG appears at *relatively* higher frequency in these important areas
- These localized areas of higher CG frequency are called *CG-islands*
- Finding the CG islands within a genome is among the most reliable gene finding approaches

CG Island Analogy

- The CG islands problem can be modeled by a toy problem named *"The Fair Bet Casino"*
- The outcome of the game is determined by coin flips with two possible outcomes: **H**eads or **T**ails
- However, there are two different coins
  - A **F**air coin: **H**eads and **T**ails with same probability ½.
  - The **B**iased coin: **H**eads with prob. ¾, **T**ails with prob. ¼.



The "Fair Bet Casino" (cont'd)

- Thus, we define the probabilities:
  - $P(H \mid Fair) = P(T \mid Fair) = ½$
  - $P(H \mid Bias) = ¾$, $P(T \mid Bias) = ¼$
  - The crooked dealer doesn't want to get caught switching between coins, so he does so infrequently
  - Changes between Fair and Biased coins with probability 10%

The Fair Bet Casino Problem

- **Input:** A sequence $x = x_1x_2x_3\ldots x_n$ of coin tosses made by some combination of the two possible coins (***F*** or ***B***).

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

Problem…

Fair Bet Casino Problem

Any observed outcome of coin tosses *could* have been generated by *either* coin, or any combination.

But, all outcomes are not equally likely. What coin combination has the highest probability of generating the observed series of tosses?

↓

Decoding Problem

P(x | fair coin) vs. P(x | biased coin)

- Suppose first, that the dealer never exchanges coins.
- Some definitions:
  - P(x | Fair): prob. of the dealer generating the outcome x using the *Fair* coin.
  - P(x | Biased): prob. of the dealer generating outcome x using the *Biased* coin .

P(x | fair coin) vs. P(x | biased coin)

- $P(x \mid \text{Fair}) = P(x_1 \ldots x_n \mid \text{Fair}) =$

  $$\prod_{i=1,n} p\,(x_i \mid \text{Fair}) = (1/2)^n$$

- $P(x \mid \text{Biased}) = P(x_1 \ldots x_n \mid \text{Biased coin}) =$

  $$\prod_{i=1,n} p\,(x_i \mid \text{Biased}) = (3/4)^k (1/4)^{n-k} = 3^k / 4^n$$

  – Where k is the number of ***H***eads in x.

P(x | fair coin) vs. P(x | biased coin)

- When is a sequence equally likely to have come from the Fair or Biased coin?

$$P(x \mid \text{Fair}) = P(x \mid \text{Biased})$$

$$1/2^n = 3^k/4^n$$

$$2^n = 3^k$$

$$n = k \log_2 3$$

- when $k = n / \log_2 3 \quad (k \sim 0.63\, n)$
- So when the number of heads is greater than 63% the dealer most likely used the biased coin

Log-odds Ratio

- We can define the ***log-odds ratio*** as follows:

$$\log_2(P(x \mid \text{Fair}) \;/\; P(x \mid \text{Biased})) =$$
$$= \Sigma^k_{i=1} \log_2(p(x_i \mid Fair) \;/\; p(x_i \mid Biased))$$
$$= n - k \log_2 3$$

- The log-odds ratio is a means for deciding which of two alternative hypotheses is most likely
- "Zero-crossing" measure; if the log-odds ratio > 0 then the numerator is more likely, if it is < 0 then the denominator is more likely, they are equally likely if the log-odds ratio = 0

Computing Log-odds Ratio in Sliding Windows

$$x_1\, x_2 \boxed{x_3\, x_4\, x_5\, x_6\, x_7}\, x_8 \ldots x_n$$

Consider a *sliding window* of the outcome sequence. Find the log-odds for this short window.

Disadvantages:

- the length of CG-island (appropriate window size) is not known in advance
- different window sizes may classify the same position differently

Key Elements of this Problem

- There is an unknown, *hidden*, state for each observation (Was the coin the Fair or Biased?)
- Outcomes are modeled probabilistically:
  - P(H|Fair) = P(T|Fair) = ½
  - P(H|Bias) = ¾, P(T|Bias) = ¼
- Transitions between states are modeled probabilistically:
  - $P(\pi_i = \text{Biased} \mid \pi_{i-1} = \text{Biased}) = a_{BB} = 0.9$
  - $P(\pi_i = \text{Biased} \mid \pi_{i-1} = \text{Fair}) = a_{FB} = 0.1$
  - $P(\pi_i = \text{Fair} \mid \pi_{i-1} = \text{Biased}) = a_{BF} = 0.1$
  - $P(\pi_i = \text{Fair} \mid \pi_{i-1} = \text{Fair}) = a_{FF} = 0.9$

Hidden Markov Model (HMM)

- A generalization of this class of problem
- Can be viewed as an abstract machine with *k hidden* states that emits symbols from an alphabet Σ.
- Each state has its own probability distribution, and the machine switches between states according to this probability distribution.
- While in a certain state, the machine makes 2 decisions:
  - What state should I move to next?
  - What symbol - from the alphabet Σ - should I emit?

Why “Hidden”?

- Observers can see the emitted symbols of an HMM but have *no ability to know which state the HMM is currently in*.
- Thus, the goal is to infer the *most likely hidden states of an HMM* based on the given sequence of emitted symbols.

HHHTHTHHTTTTHTHTHTHHHTHTHTHT
BBBFFFFFFFFFFFFFFFFFBBBFFFFFF?

HMM Parameters

Σ: set of emission characters.

Ex.: $\Sigma = \{0, 1\}$ for coin tossing
(0 for ***T***ails and 1 ***H***eads)

$\Sigma = \{1, 2, 3, 4, 5, 6\}$ for dice tossing

Q: set of hidden states, emitting symbols from Σ.

Q = {F,B} for coin tossing

HMM Parameters (cont'd)

$A = (a_{kl})$: a |Q| x |Q| matrix of probability of changing from state k to state l. *Transition matrix*

$a_{FF} = 0.9 \quad a_{FB} = 0.1$

$a_{BF} = 0.1 \quad a_{BB} = 0.9$

$E = (e_k(b))$: a |Q| x |Σ| matrix of probability of emitting symbol b while being in state k. *Emission matrix*

$e_F(0) = \frac{1}{2} \quad e_F(1) = \frac{1}{2}$

$e_B(0) = \frac{1}{4} \quad e_B(1) = \frac{3}{4}$

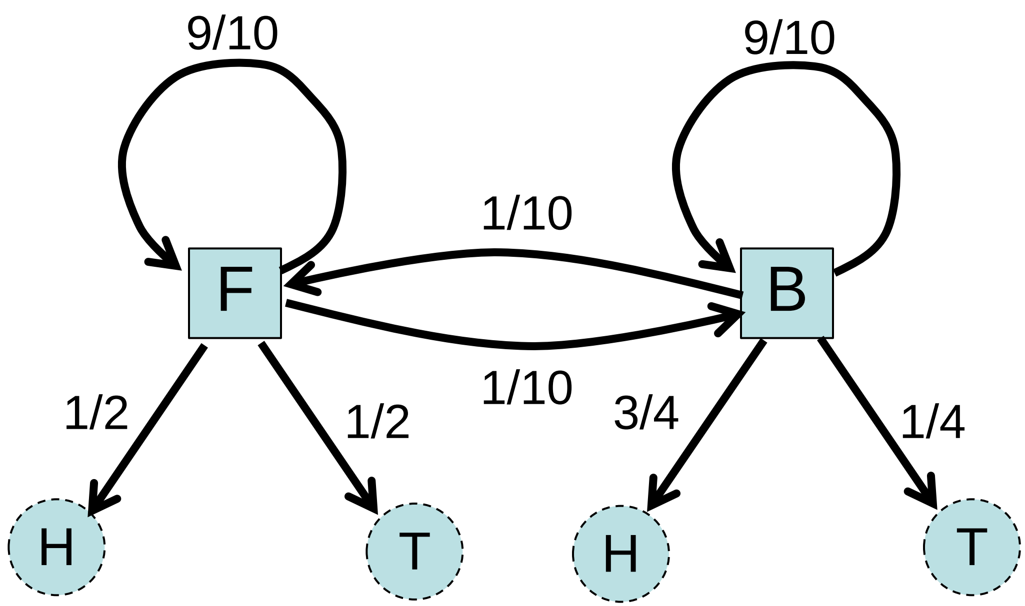
HMM for Fair Bet Casino

- The *Fair Bet Casino* in *HMM* terms:
  $\Sigma = \{0, 1\}$ (0 for ***T***ails and 1 ***H***eads)
  $Q = \{F,B\}$ – *F* for Fair & *B* for Biased coin.
- Transition Probabilities *A*, Emission Probabilities *E*

A	Fair	Biased
Fair	0.9	0.1
Biased	0.1	0.9

E	Tails(0)	Heads(1)
Fair	½	½
Biased	¼	¾

HMM for Fair Bet Casino (cont'd)



HMM model for the *Fair Bet Casino* Problem

Hidden Paths

- A *path* $\pi = \pi_1 \ldots \pi_n$ in the HMM is defined as a sequence of hidden states.
- Consider path π = FFFBBBBBFFF and sequence x = 01011101001

x	=	0	1	0	1	1	1	0	1	0	0	1
π	=	F	F	F	B	B	B	B	B	F	F	F
$P(x_i \mid \pi_i)$		½	½	½	¾	¾	¾	¼	¾	½	½	½
$P(\pi_{i-1} \rightarrow \pi_i)$		½	$^9/_{10}$	$^9/_{10}$	$^1/_{10}$	$^9/_{10}$	$^9/_{10}$	$^9/_{10}$	$^9/_{10}$	$^1/_{10}$	$^9/_{10}$	$^9/_{10}$

Probability that x_i was emitted from state π_i

Transition probability from state π_{i-1} to state π_i

P(x | π) Calculation

- P(x | π): Probability that sequence x was generated by the path π:

$$P(x \mid \pi) = P(\pi_0 \rightarrow \pi_1) \cdot \prod_{i=1}^{n} P(x_i \mid \pi_i) \cdot P(\pi_i \rightarrow \pi_{i+1})$$

$$= a_{\pi_0, \pi_1} \cdot \prod e_{\pi_i}(x_i) \cdot a_{\pi_i, \pi_{i+1}}$$

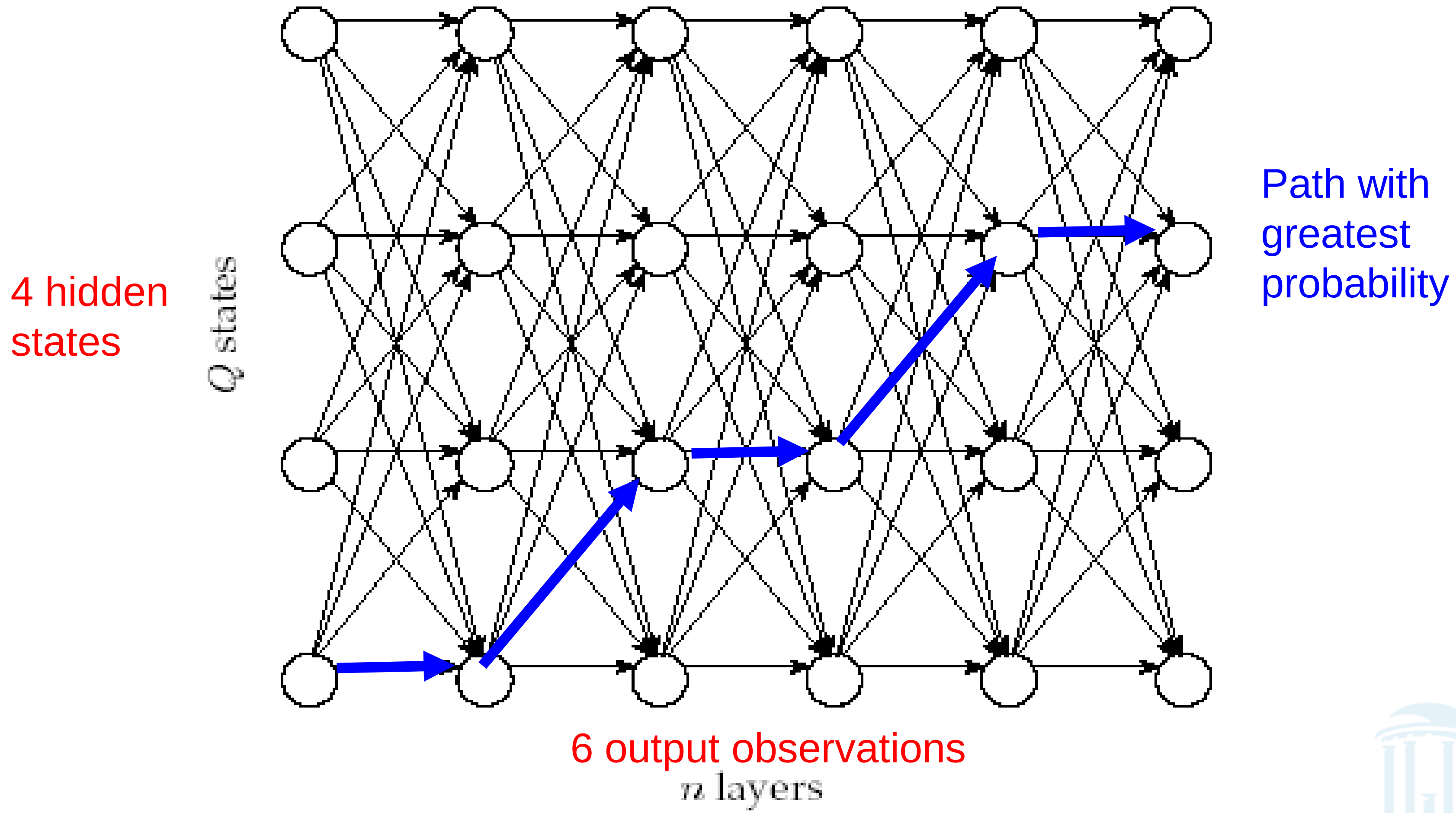
Decoding Problem

- **Goal:** Find an optimal hidden path of state transitions given a set of observations.
- **Input:** Sequence of observations $x = x_1 \ldots x_n$ generated by an HMM $M(\Sigma, Q, A, E)$
- **Output:** A path that maximizes $P(x \mid \pi)$ over all possible paths π.

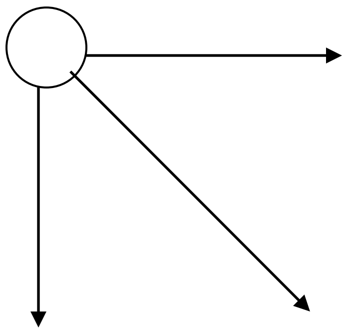
Building Manhattan for Decoding Problem

- In 1967 Andrew Viterbi developed a dynamic programming algorithm to solve the *Decoding Problem*.
  - Organized as a graph of possible hidden state transitions
  - Viewed as a "manhattan grid"
- Every choice of $\pi = \pi_1 \dots \pi_n$ corresponds to a path in the graph.
- The only valid direction in the graph is *eastward*.
- This graph has $|Q|^2(n-1)$ edges.

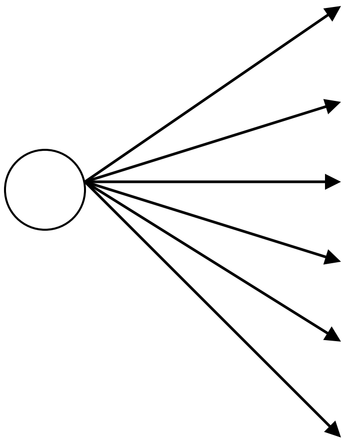
Directed Acyclic Graph for Decoding Problem



Decoding Problem vs. Alignment Problem



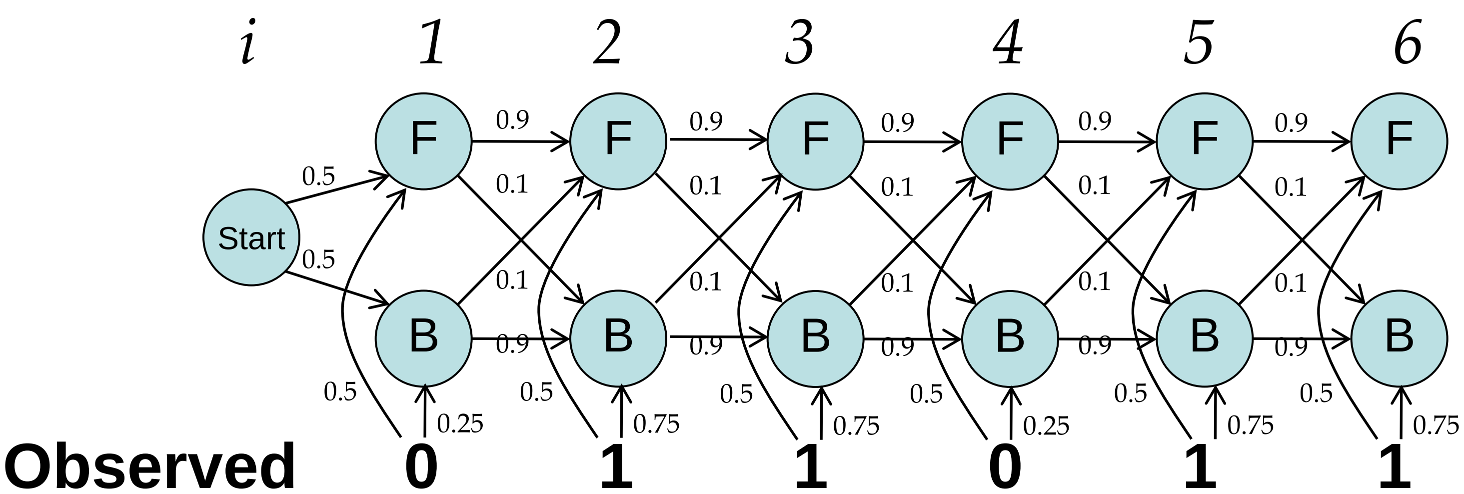
Valid directions in the *alignment problem*.



Valid directions in the *decoding problem.*

Viterbi Decoding of Fair-Bet Casino

- Each vertex represents a possible state at a given position in the output sequence
- The observed sequence conditions the likelihood of each state
- Dynamic programming reduces search space to: $|Q|$+transition_edges × (n-1) = 2+4 × 5 from naïve 2^6



Decoding Problem

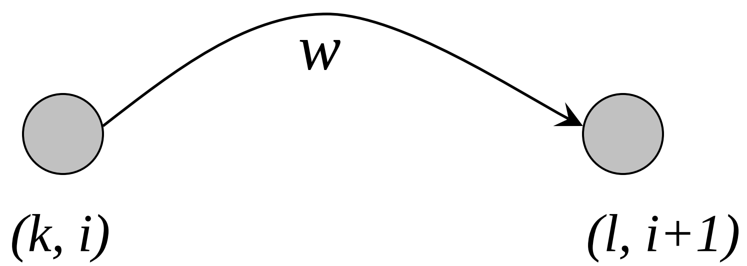
- The *Decoding Problem* is reduced to finding a longest path in the *directed acyclic graph (DAG)*

- **Notes:** the length of the path in this problem is defined as the *product* of its edges' weights, not their *sum*. (But, using the log of the weights makes it a sum again!)

Decoding Problem (cont'd)

- Every path in the graph has the probability $P(x \mid \pi)$.
- The Viterbi algorithm finds the path that maximizes $P(x \mid \pi)$ among all possible paths.
- The Viterbi algorithm runs in $\boldsymbol{O(n|Q|^2)}$ time.

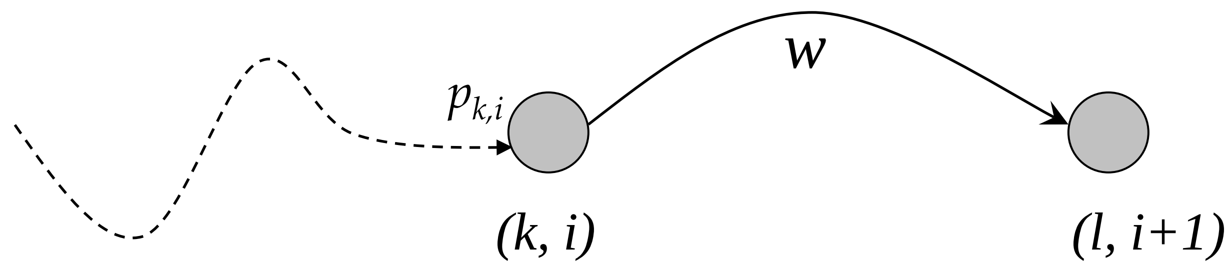
Decoding Problem: weights of edges



???

Decoding Problem: weights of edges

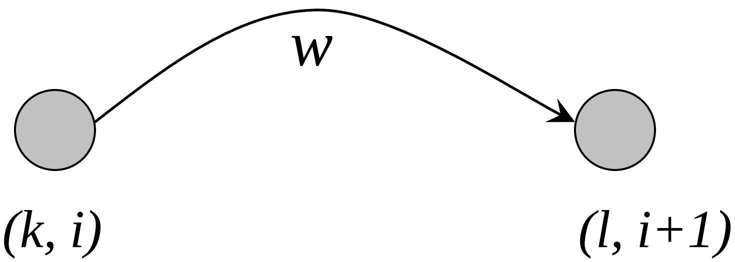
Each edge is a factor in the product



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

Decoding Problem: weights of edges

i-th term $= e_{\pi_i}(x_i) \,.\, a_{\pi_i, \pi_{i+1}} = \boldsymbol{e_l(x_{i+1}).\ a_{kl}}$ *for* $\pi_i = k, \pi_{i+1} = l$



The weight $\boldsymbol{w = e_l(x_{i+1}).\ a_k}$

Solve for the path of highest probability

Observation: a prefix is also an optimal path

Where have we seen this before?

Dynamic Program's Recursion

$$s_{l,i+1} = \max_{k \in Q} \{s_{k,i} \cdot \text{weight of edge between } (k,i) \text{ and } (l,i+1)\}$$

$$= \max_{k \in Q} \{ s_{k,i} \cdot a_{kl} \cdot e_l (x_{i+1}) \}$$

$$= e_l (x_{i+1}) \cdot \max_{k \in Q} \{s_{k,i} \cdot a_{kl}\}$$

Decoding Problem (cont'd)

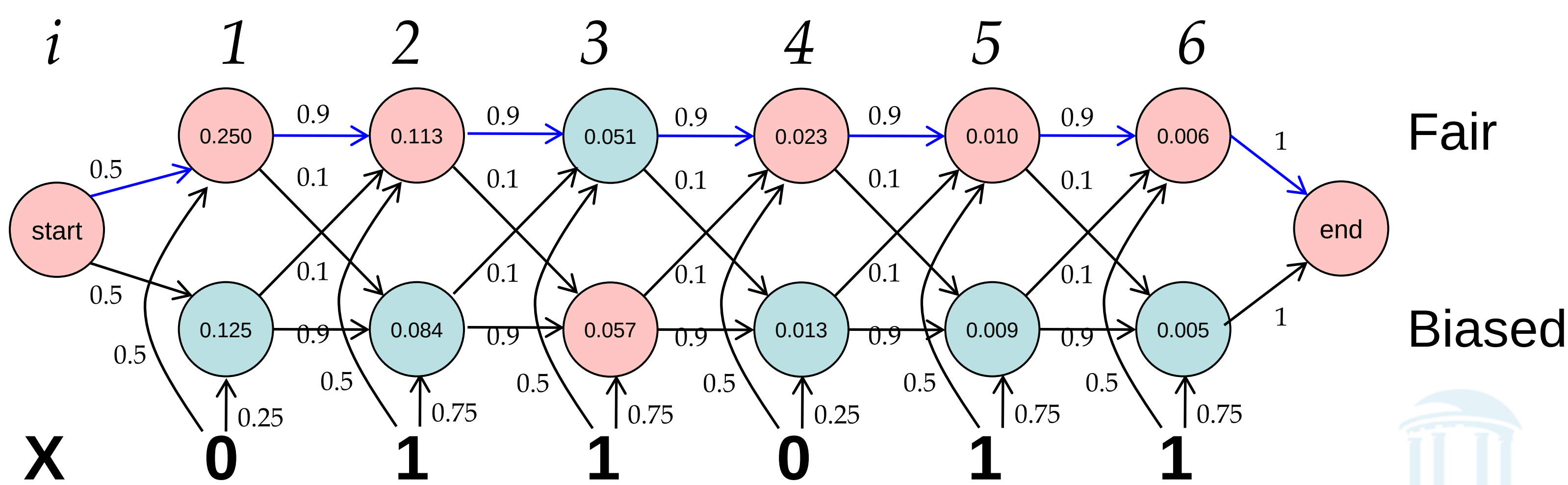
- Initialization:
  - $a_{start,k} = 1/|Q|$
  - $s_{k,0} = 0$ for $k \neq begin$

- Let π^* be the optimal path. Then,

$$P(x|\pi^*) = \max_{k \in Q} \{s_{k,n} \,.\, a_{k,end}\}$$

Viterbi for Fair Bet Casino

- Solves all subproblems implied by emitted subsequence
- How likely is the best path? 0.006
- What is it? FFBFFF



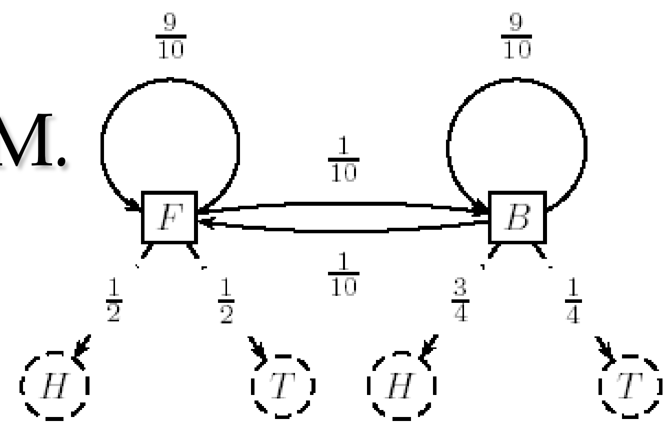
Viterbi Algorithm

- Rather than addition Viterbi uses multiplication
- Converts edge weights to logs, and then it is back to addition, which has another advantage
- The value of the product can become extremely small, which leads to underflow.
- Logs avoid underflow.

$$s_{k,i+1} = \log e_l(x_{i+1}) + \max_{k \in Q} \{s_{k,i} + \log(a_{kl})\}$$

Forward-Backward Problem

Given: a sequence of coin tosses generated by an HMM.



Goal: find the most probable coin that the dealer was using at a particular time.

$$P(\pi_i = k \mid x) = \frac{P(x, \pi_i = k)}{P(x)}$$

Sum of probabilities of all paths in state k at i

Sum of probability of all paths

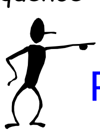
Illustrating the difference

Not a lot worse than the best solution

x = THHH p

FFFF (0.0228)
BFFF (0.0013)
FBFF (0.0004)
BBFF (0.0019)
FFBF (0.0004)
BFBF (0.0000)
FBBF (0.0006)
BBBF (0.0028)
FFFB (0.0038)
BFFB (0.0002)
FBFB (0.0001)
BBFB (0.0003)
FFBB (0.0057)
BFBB (0.0003)
FBBB (0.0085)
BBBB (0.0384)

Viterbi solution, the most likely sequence states.



P(x) = 0.0877

High probability output x =THHH (0.0877 > $(1/2)^4$)

x = THHH p

FFFF (0.0228)
FFBF (0.0004)
FFFB (0.0038)
FFBB (0.0057)
BFFF (0.0013)
BFBF (0.0000)
BFFB (0.0002)
BFBB (0.0003)

$P(\pi_2=F|x) = 0.0345/0.0877 = 0.3936$

FBFF (0.0004)
FBBF (0.0006)
FBFB (0.0001)
FBBB (0.0085)
BBFF (0.0019)
BBBF (0.0028)
BBFB (0.0003)
BBBB (0.0384)

$P(\pi_2=B|x) = 0.0532/0.0877 = 0.6064$

The forward-backward algorithm tells us how likely we were using the biased coin at the second flip.

Forward Algorithm

- Defined $f_{k,i}$ (*forward probability*) as the probability of emitting the prefix $x_1 \ldots x_i$ and reaching the state $\pi = k$.
- The recurrence for the forward algorithm is:

$$f_{k,i} = e_k(x_i) \cdot \sum_{l \in Q} f_{l,i-1} \cdot a_{l,k} \qquad f_{start,0} = 1$$

Probability of emitting x_i at step i

Probability of transitioning from state l at time i-1 to state k at time i

- Same as Viterbi recurrence except using summation instead of maximum

Backward Algorithm

- However, *forward probability* is not the only factor affecting $P(\pi_i = k|x)$.

- The sequence of transitions and emissions that the HMM undergoes between π_i and π_{i+1} also affect $P(\pi_i = k|x)$.

Backward Algorithm (cont'd)

- *Backward probability* $b_{k,i}$ ≡ the probability of being in state $\pi_i = k$ and emitting the *suffix* $x_{i+1} \ldots x_n$.
- The *backward algorithm*'s recurrence:

$$b_{k,i} = \sum_{l \in Q} e_l(x_{i+1}) \cdot b_{l,i+1} \cdot a_{k,l} \qquad b_{end,n+1} = 1$$

This sums the probabilities of paths that start from state k at time i and reach the end state. The recurrence extends the paths backwards, starting from the end state.

Forward-Backward Algorithm

- The probability that the dealer used a biased coin at any moment *i* is as follows:

$$P(\pi_i = k|x) = \frac{P(x, \pi_i = k)}{P(x)} = \frac{f_k(i) \, . \, b_k(i)}{P(x)}$$

What can we do so far

Given HMM H=$M(\Sigma, Q, A, E)$ and observed sequence $x=x_1 \ldots x_n$ we can now

- Use the Viterbi decoding algorithm to find the hidden path $\pi = \pi_1 \ldots \pi_n$ that maximizes $P(x \mid \pi)$
  - Chooses the single best path

- Use the forward-backward algorithm to find the probability M was in state k at time i
  - Finds individual likelihoods for each state

HMM Parameter Estimation

- So far, we have assumed that the transition and emission probabilities are known.

- However, in most HMM applications, these probabilities are not known. Can we discover them?
  - this is parameter estimation
  - much harder than state estimation

HMM Parameter Estimation (cont'd)

- Let Θ be a vector containing all of the unknown transition and emission probabilities (A and E).
- Given training sequences $x = x^1,\ldots,x^m$, let $P(x \mid \Theta)$ be the maximum probability of x given the assignment of parameters Θ.
- Our goal is to find

$$max_{\Theta} \prod_{j=1}^{m} P(x_i|\Theta)$$

i.e. find A and E that maximize the probability of the training sequences

A Parameter Estimation Approach

- If hidden states were known, we could use our training data to estimate parameters. For the observed sequences x, let A_{kl} be the number of transitions from state k to l, and let $E_k(b)$ be the number of times b is emitted in state k, then we can estimate

$$a_{kl} = \frac{A_{kl}}{\sum_{q \in Q} A_{kq}} \qquad e_k(b) = \frac{E_k(b)}{\sum_{\sigma \in \Sigma} E_k(\sigma)}$$

- But we don't know the hidden state sequences -- we only know the observed output streams x
- Approach: *make an intelligent guess of* π, use the equations above to estimate parameters, then run Viterbi to estimate the hidden state. Re-estimate the parameters and repeat until the state assignments or parameter values converge (Baum-Welch)
- Such iterative approaches are called Expectation Maximization (EM) methods of parameter estimation

Profile Alignment using HMMs

- Distant species of functionally related sequences may have weak pairwise similarities with individual known species, and thus fail individual pairwise significance tests.
- However, they may have weak similarities with ***many*** known species.
- The goal is to consider all similarities once using multiple alignment
- Related sequences are often better represented by a *consensus profile* than any multiple alignment.

Consensus Profile Representation

Aligned DNA sequences can be represented by a $4 \cdot n$ profile matrix reflecting the frequencies of nucleotides in every aligned position.

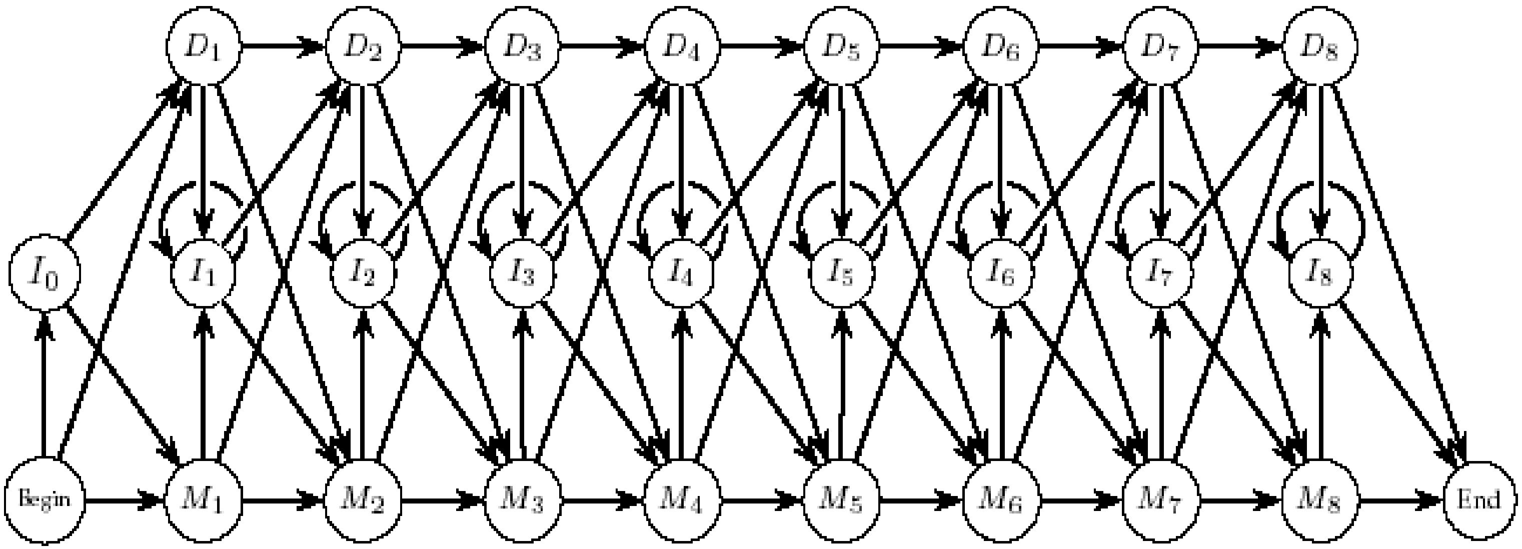
A	.72	.14	0	0	.72	.72	0	0
T	.14	.72	0	0	0	.14	.14	.86
G	.14	.14	.86	.44	0	.14	0	0
C	0	0	.14	.56	.28	0	.86	.14

Protein families can be represented by a $20 \cdot n$ profile representing frequencies of amino acids.

HMM Alignment

- One method of performing sequence comparisons to a consensus profile is to use a HMM
- Emission probabilities, $e_i(a)$, from the profile
- Transition probabilities from our score matrix δ_{ij}.
- Explicitly model insertions and deletions as separate states at each position

Profile HMM



A profile HMM

States of Profile HMM

- Match states $M_1 \ldots M_n$ (plus *begin/end* states)
- Insertion states $I_0 I_1 \ldots I_n$
- Deletion states $D_1 \ldots D_n$

- Assumption:

$$e_{I_j}(a) = p(a)$$

where $p(a)$ is the frequency of the occurrence of the symbol a in all the sequences.

Transition Probabilities in Profile HMM

- $log(a_{MI})+log(a_{IM})$ = gap initiation penalty

- $log(a_{II})$ = gap extension penalty

Profile HMM Alignment

- Define $v^M_j(i)$ as the logarithmic likelihood score of the best path for matching $x_1..x_i$ to profile HMM ending with x_i emitted by the state M_j.

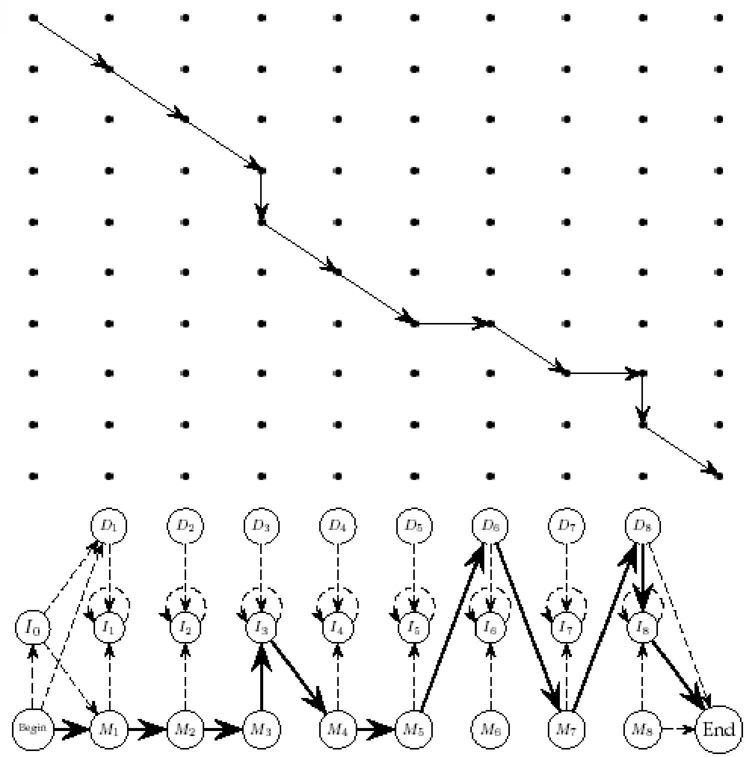
- $v^I_j(i)$ and $v^D_j(i)$ are defined similarly.

Profile HMM Alignment: Dynamic Programming

$$v^M_j(i) = \log\left(e_{M_j}(x_i)/p(x_i)\right) + \max \begin{cases} v^M_{j-1}(i-1) + \log(a_{M_{j-1},M_j}) \\ v^I_{j-1}(i-1) + \log(a_{I_{j-1},M_j}) \\ v^D_{j-1}(i-1) + \log(a_{D_{j-1},M_j}) \end{cases}$$

$$v^I_j(i) = \log\left(e_{I_j}(x_i)/p(x_i)\right) + \max \begin{cases} v^M_j(i-1) + \log(a_{M_j}, I_j) \\ v^I_j(i-1) + \log(a_{I_j}, I_j) \\ v^D_j(i-1) + \log(a_{D_j}, I_j) \end{cases}$$

Paths in Edit Graph and Profile HMM



A path through an edit graph and the corresponding path through a profile HMM