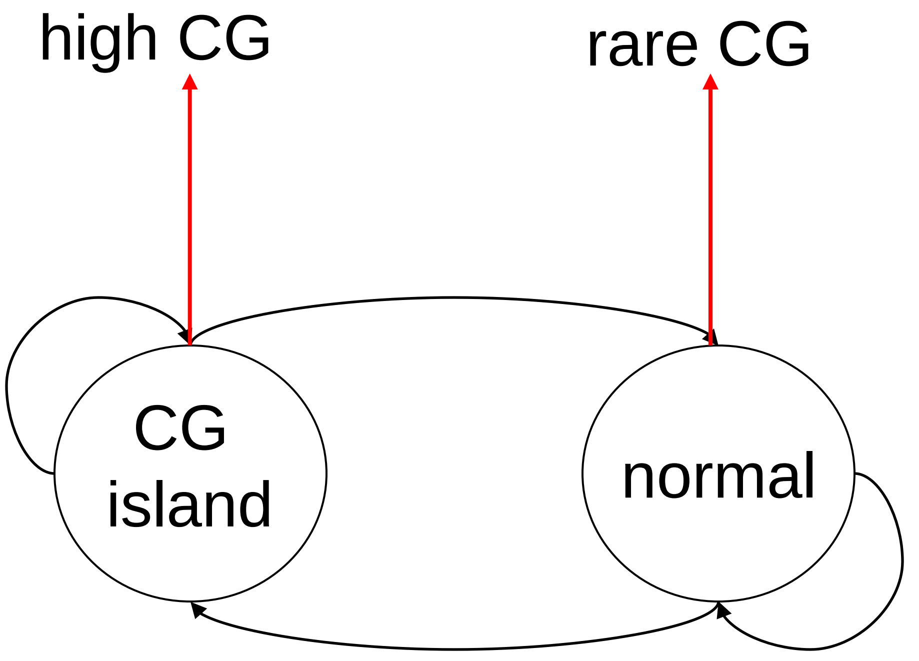
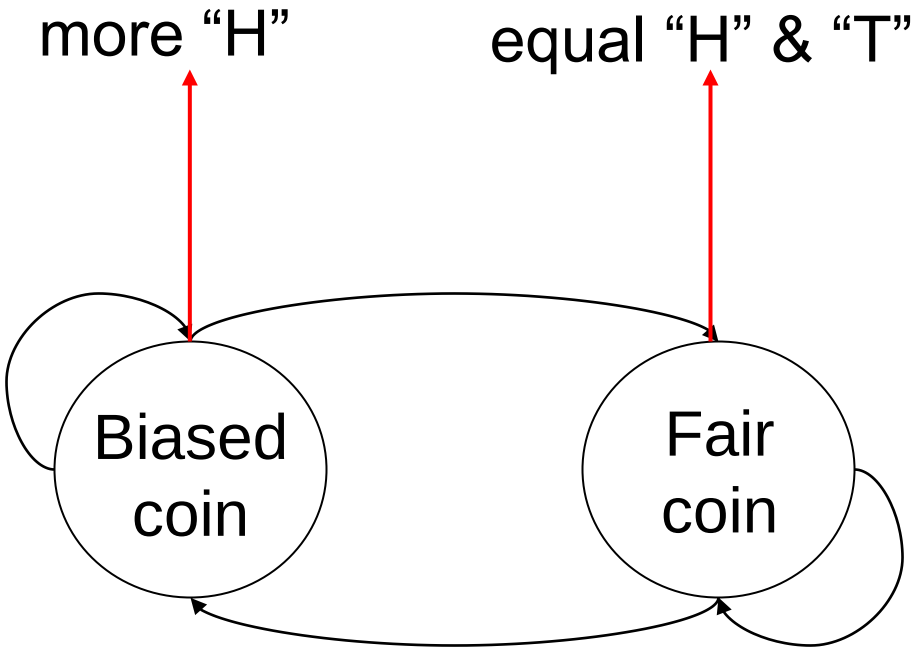
Hidden Markov Models
Chapter 11

CG “islands”

- The dinucleotide “CG” is rare
  - C in a “CG” often gets “methylated” and the resulting C then mutates to T
- Methylation is suppressed in some areas of genome, called “CG islands”
- Such CG islands often found around genes
- Problem: find CG islands in whole genome



- Two states.
- Each state emits sequence.
- Sequence emitted by “CG island” state is high in CG frequency
- Concatenation of sequence emissions = genome



- Two states.
- Each state emits a “coin toss” result.
- Sequence emitted by “Biased coin” state is high in “Heads” frequency

CG Islands and the "Fair Bet Casino"

- The *CG* islands problem can be modeled after a problem named ***"The Fair Bet Casino"***
- The game is to flip coins, which results in only two possible outcomes: **H**ead or **T**ail.
- The **F**air coin will give **H**eads and **T**ails with same probability ½.
- The **B**iased coin will give **H**eads with prob. ¾.

The “Fair Bet Casino” (cont’d)

- Thus, we define the probabilities:
  - $P(H|F) = P(T|F) = ½$
  - $P(H|B) = ¾$, $P(T|B) = ¼$
  - The crooked dealer chages 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 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 any sequence of states!

Need to incorporate a way to grade different sequences differently.

Decoding Problem

Hidden Markov Model (HMM)

- 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.

HMM Parameters

Σ: set of emission characters.

Ex.: $\Sigma = \{H, T\}$ for coin tossing

Q: set of hidden states, each 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.*

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

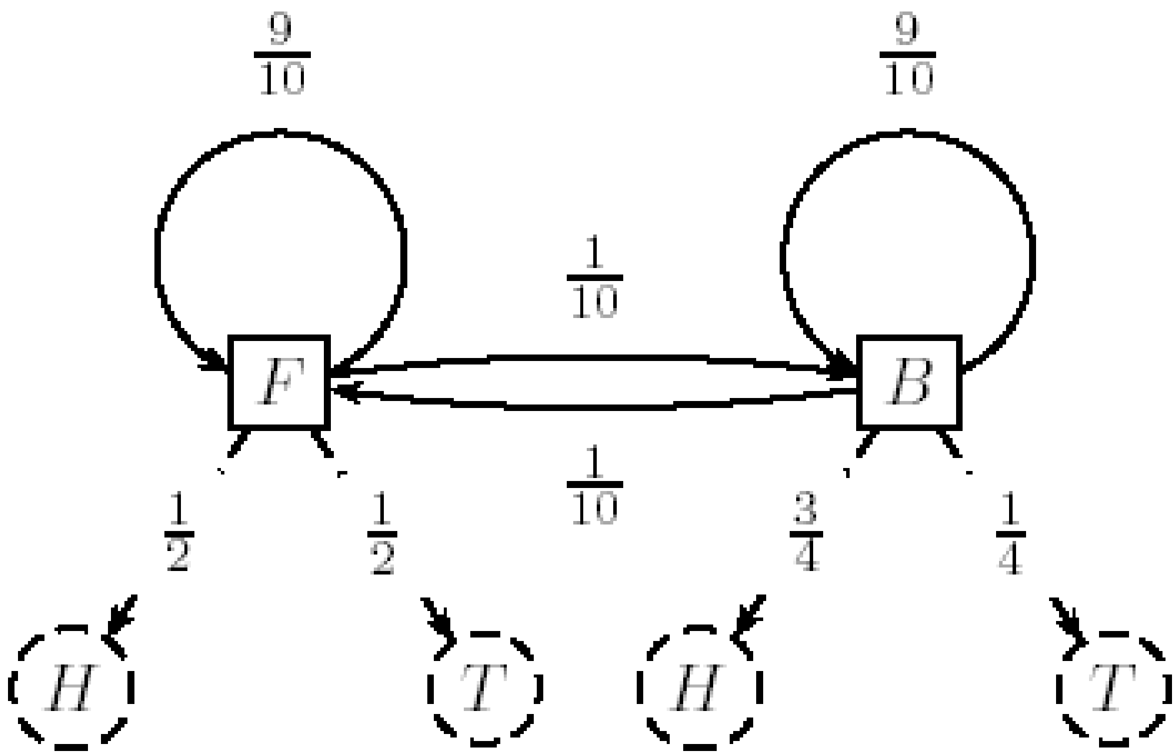
$$a_{BF} = 0.1 \qquad a_{BB} = 0.9$$

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

$$e_F(0) = \tfrac{1}{2} \qquad e_F(1) = \tfrac{1}{2}$$

$$e_B(0) = \tfrac{1}{4} \qquad e_B(1) = \tfrac{3}{4}$$

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 states.
- Consider path π = FFFBBBBBFFF and sequence x = 01011101001

Probability that x_i was emitted from state π_i

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	$^9/_{10}$	$^9/_{10}$	$^1/_{10}$	$^9/_{10}$	$^9/_{10}$	$^9/_{10}$	$^9/_{10}$	$^1/_{10}$	$^9/_{10}$	$^9/_{10}$

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:*

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

$$= \prod_{i=1}^{n} P(\pi_{i-1} \to \pi_i)P(x_i \mid \pi_i)$$

$$= \prod_{i=1} a_{\pi_{i-1},\pi_i} e_{\pi_i}(x_i)$$

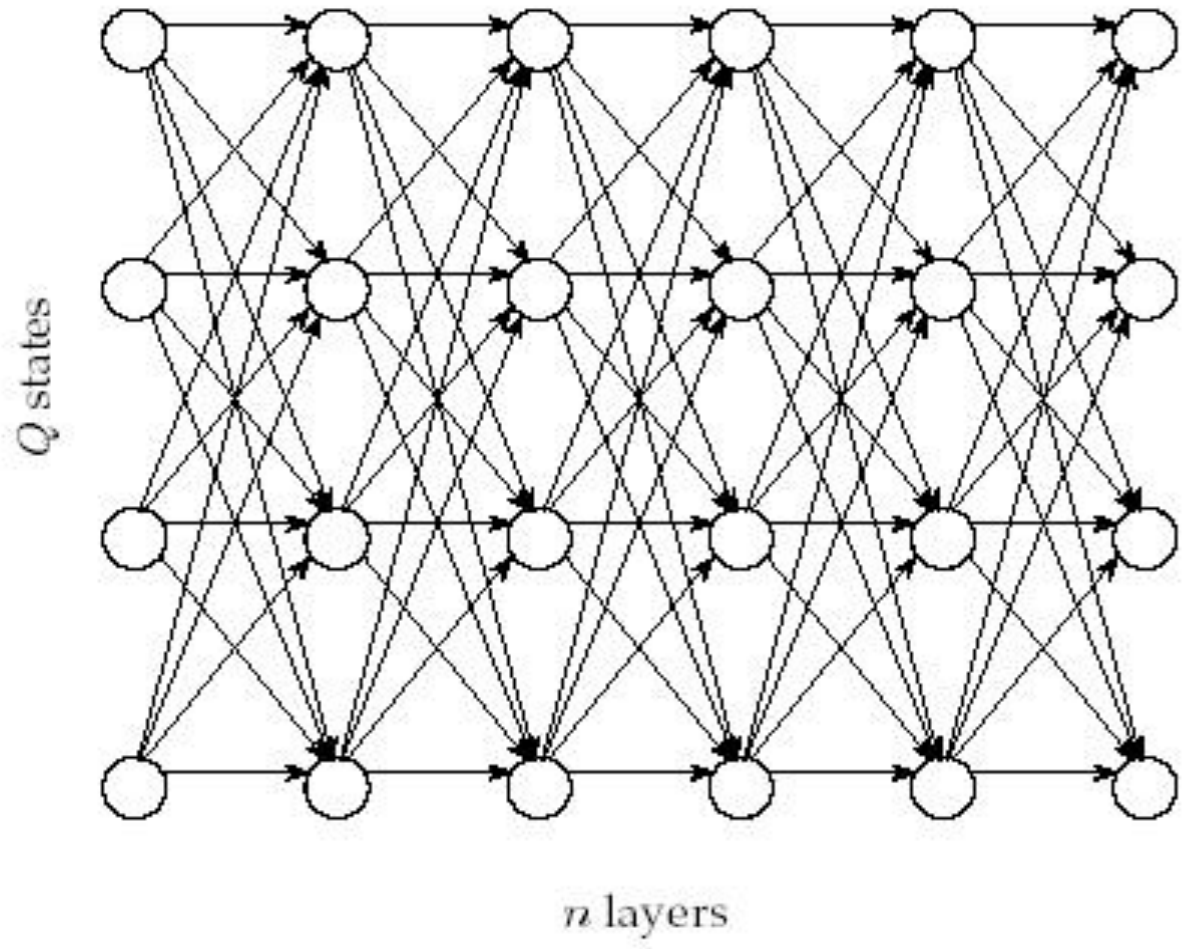
Decoding Problem

- **Goal:** Find an optimal hidden path of states given 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 $P(x,\pi)$ over all possible paths π.

Dynamic programming for the Decoding Problem

- Andrew Viterbi used Dynamic programming to solve the *Decoding Problem*.
- Build a graph with one node for every (i,j), representing the possibility that the i^{th} position of π was emitted from state j.
- Every choice of $\pi = \pi_1 \ldots \pi_n$ corresponds to a path in the graph.
- Edges are weighted. Sum of edge weights on a path π corresponds to $P(x,\pi)$

Graph for Decoding Problem



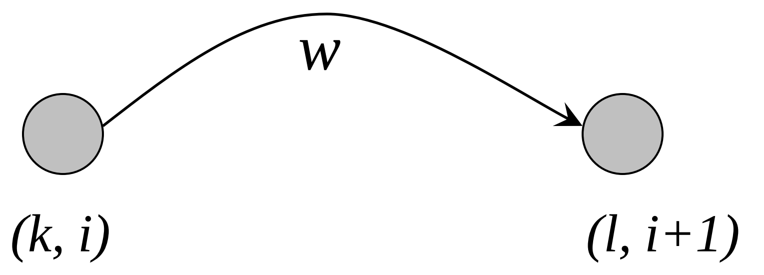
Decoding Problem

- Every path in the graph has the probability $P(x,\pi)$.

- The Viterbi algorithm finds the path that maximizes $P(x,\pi)$ among all possible paths.

- The Viterbi algorithm runs in $\boldsymbol{O(n|Q|^2)}$ time.

Decoding Problem: weights of edges

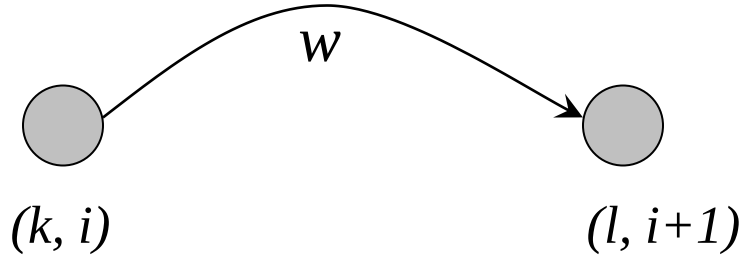


The weight ***w*** is given by:

???

Decoding Problem: weights of edges

$$P(x,\pi) = \prod_{i=1}^{n} e_{\pi_i}(x_i) \,.\, a_{\pi_{i-1}, \pi_i}$$



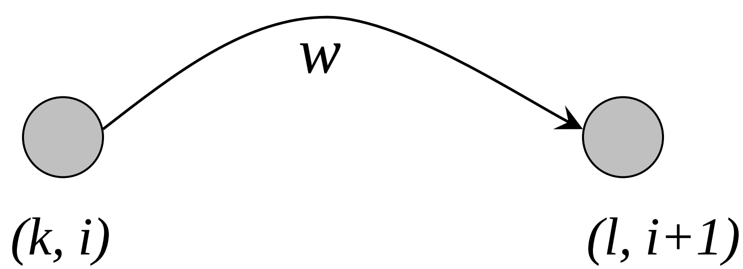
The weight ***w*** is given by:

??

Decoding Problem: weights of edges

$$i+1\text{-th term} = e_{\pi_{i+1}}(x_{i+1}) \,.\, a_{\pi_i, \pi_{i+1}}$$

$$= \boldsymbol{e_l(x_{i+1}) .\ a_{kl}}$$



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

Dynamic programming recurrence

- $s_{k,i}$ is the probability of the most likely path for the prefix $x_1 \ldots x_i$ that ends in k
- $s_{l,i+1} = \max_k \{s_{k,i}$ x weight of edge from (k,i) to (l,i+1)}

  $= \max_k \{s_{k,i}\, a_{kl}\, e_l(x_{i+1})\}$
- Let π^* be the optimal path. Then,

  $P(x,\pi^*) = \max_k \{s_{k,n} \cdot a_{k,end}\}$
- Time complexity: $O(n\,|Q|^2)$

Viterbi Algorithm

- The value of the product can become extremely small, which leads to overflowing.
- To avoid overflowing, use log value instead: $S_{k,i} = \log s_{k,i}$
- New recurrence:

$$S_{l,i+1} = \log e_l(x_{i+1}) + \max_k \{S_{k,i} + \log(a_{kl})\}$$

Forward-Backward Problem

Given: a sequence $x = x_1\ x_2\ \ldots x_n$ generated by an HMM.

Goal: find the probability that x_i was generated by state k.

That is, given x, what is $P(\pi_i = k \mid x)$?

Forward Algorithm

- Define $f_{k,i}$ (*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:

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

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+1} and π_n also affect $P(\pi_i = k|x)$.

forward x_i backward

Backward Algorithm (cont'd)

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

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

Backward-Forward Algorithm

- The probability that the dealer used a biased coin at any moment i:

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

$P(x)$ is the sum of $P(x, \pi_i = k)$ over all k