Hidden Markov Models

Saeedeh Akbari

Department of Computer Engineering

Sharif University of Technology

Fall 2023

Adapted with modifications from lecture notes prepared by Phillip Compeau

Bioinformatics Algorithms: An Active Learning Approach

Outline

- Introduction on Probability
- CG-islands
- The "Fair Bet Casino"
- Hidden Markov Model
- Decoding Algorithm
- Viterbi Algorithm
- Forward-Backward Algorithm

Simple probabilistic models for DNA sequences

- Assume nature generates a type of DNA sequence as follows:
  1. A box of dice, each with four faces: {A,C,G,T}
  2. Select a die suitable for the type of DNA
  3. Roll it, append the symbol to a string.
  4. Repeat 3, until all symbols have been generated.
- Given a string say X="GATTCCAA…" and two dice
  - M1 has the distribution of pA=pC=pG=pT=0.25.
  - M2 has the distribution: pA=pT=0.20, pC=pG=0.30
- What is the probability of the sequence being generated by M1 or M2?

Markov models for DNA sequences

We have assumed *independence* of nucleotides in different positions - unrealistic in biology

Outline

- Introduction on Probability
- CG-islands
- The "Fair Bet Casino"
- Hidden Markov Model
- Decoding Algorithm
- Viterbi Algorithm
- Forward-Backward Algorithm

CG-Islands

- Given 4 nucleotides: probability of occurrence is ~ 1/4. Thus, probability of occurrence of a dinucleotide is ~ 1/16.
- However, the frequencies of dinucleotides in DNA sequences vary widely.
- In particular, *CG* is typically underepresented (frequency of *CG* is typically < 1/16)

Why *CG*-Islands?

- *CG* is the least frequent dinucleotide because *C* in *CG* is easily *methylated and* has the tendency to mutate into T afterwards
- However, the methylation is suppressed around genes in a genome. So, *CG* appears at relatively high frequency within these *CG* islands
- So, finding the *CG* islands in a genome is an important problem

Outline

- Introduction on Probability
- CG-islands
- The "Fair Bet Casino"
- Hidden Markov Model
- Decoding Algorithm
- Viterbi Algorithm
- Forward-Backward Algorithm

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:
  - $P(H \mid F) = P(T \mid F) = ½$
  - $P(H \mid B) = ¾$, $P(T \mid B) = ¼$
  - The crooked dealer 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 two possible coins (***F*** or ***B***).

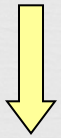
Output: A sequence $\pi = \pi_1\, \pi_2\, \pi_3\ldots\, \pi_n$, with each π_i being either F or B indicating that x_i is the result of tossing the Fair or Biased coin respectively.

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

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

- Suppose first that dealer never changes coins. Some definitions:
  - P(x | fair coin): prob. of the dealer using the F coin and generating the outcome x.
  - P(x | biased coin): prob. of the dealer using the B coin and generating outcome x.

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

- P(x | fair coin)=P($x_1 \ldots x_n$ | fair coin)=

  $\Pi_{i=1,n}\, p\ (x_i \mid \text{fair coin}) = (1/2)^n$

- P(x | biased coin)= P($x_1 \ldots x_n$ | biased coin)=

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

  - k - the number of ***H***eads in x.

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

- P(*x* | fair coin) = P(*x* | biased coin)

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

$$2^n = 3^k$$

$$n = k \log_2 3$$

when $k = n / \log_2 3$ $(k \sim 0.67n)$

Log-odds Ratio

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

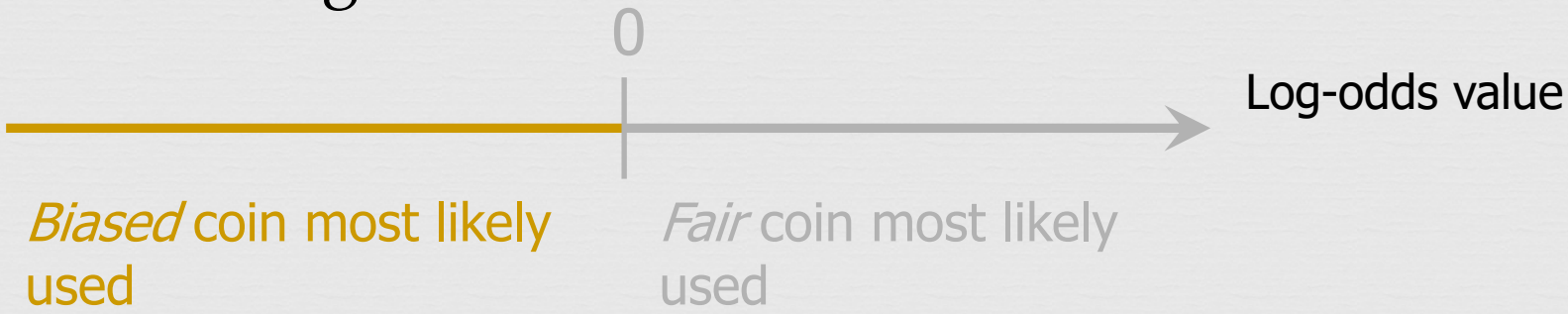
$$\log_2(\mathrm{P}(x \mid \text{fair coin}) \,/\, \mathrm{P}(x \mid \text{biased coin}))$$

$$= n - k \log_2 3$$

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 is not known in advance (window size)

Outline

- Introduction on Probability
- CG-islands
- The "Fair Bet Casino"
- Hidden Markov Model
- Decoding Algorithm
- Viterbi Algorithm
- Forward-Backward Algorithm

Markov models

- A sequence of random variables is a ***k-th order Markov chain*** if, for all *i*, i^{th} value is independent of all but the previous *k* values:

$$P(x_i \mid x_1, x_2, \ldots, x_{i-1}) = P(x_i \mid x_{i-k}, x_{i-k+1}, \ldots, x_{i-1})$$

- First order (k=1): $P(x_i|x_1, x_2, ..., x_{i-1}) = P(x_i|x_{i-1})$
- Second order: $P(x_i|x_1, x_2, ..., x_{i-1}) = P(x_i|x_{i-1}, x_{i-2})$
- 0^{th} order: (independence) $P(x_i|x_1, x_2, ..., x_{i-1}) = P(x_i)$

First order Markov model

$$P(x_i|x_1, x_2, ..., x_{i-1}) = P(x_i|x_{i-1})$$

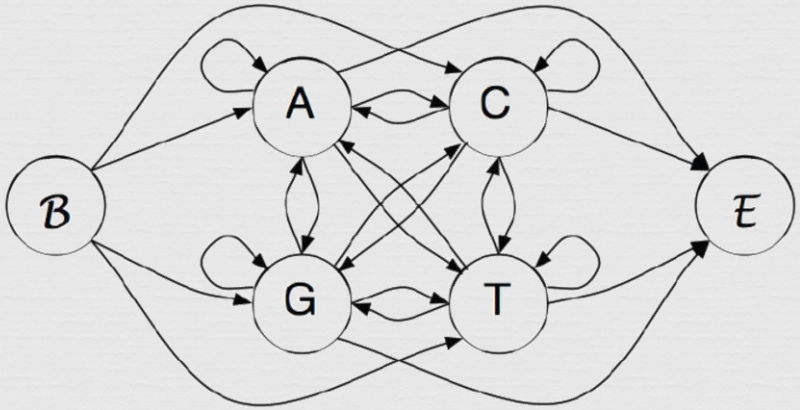
$$\begin{aligned}
& P(x_1 x_2 x_3 \cdots x_n) \\
= \quad & P(x_n|x_1 x_2 \cdots x_{n-1}) P(x_1 x_2 ... x_{n-1}) \\
= \quad & P(x_n|x_{n-1}) P(x_1 x_2 \cdots x_{n-1}) \\
= \quad & P(x_n|x_{n-1}) P(x_{n-1}|x_{n-2}) \cdots P(x_2|x_1) P(x_1) \\
= \quad & P(x_1) \prod_{i=2}^{n} P(x_i|x_{i-1})
\end{aligned}$$

Probability of emitting sequence x

$$
\begin{aligned}
x &= x_1\, x_2\, \ldots\, x_n \\
P(x) &= P(x_1, x_2, \ldots, x_n) \\
&= P(x_1) \cdot P(x_2 \mid x_1) \cdots P(x_n \mid x_{n-1}, \ldots, x_1) \\
&= P(x_1) \cdot P(x_2 \mid x_1) \cdots P(x_n \mid x_{n-1}) \\
&= P(x_1) \prod_{i=1}^{n-1} a_{x_i, x_{i+1}} \\
&= \prod_{i=0}^{n-1} a_{x_i, x_{i+1}} \qquad \text{(with Begin state)}
\end{aligned}
$$

A 1st order Markov model for CpG islands

- Essentially a finite state automaton (FSA)
- Transitions are probabilistic (instead of deterministic)



- 4 states: A, C, G, T
- 16 transitions: $a_{st} = P(x_i = t \mid x_{i-1} = s)$
- Begin/End states

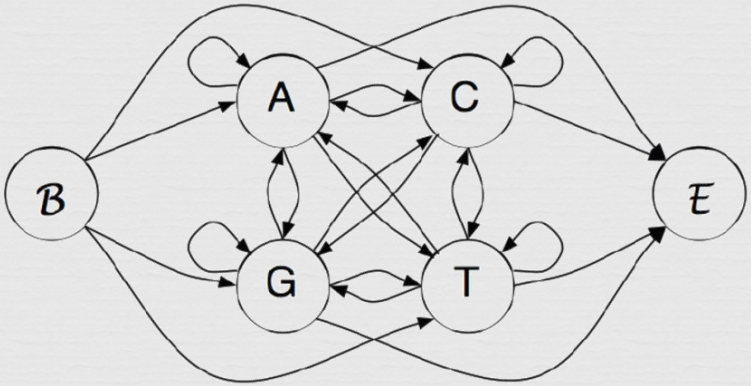
Probability of a sequence

- What's the probability of ACGGCTA in this model?

  $P(A) * P(C|A) * P(G|C) \ldots P(A|T)$

  $= a_{BA}\, a_{AC}\, a_{CG} \ldots a_{TA}$

- Equivalent: follow the path in the automaton, and multiply the transition probabilities on the path



Hidden Markov Model (HMM)

- Introduced in the 70's for speech recognition
- Have been shown to be good models for bio sequences
  - Alignment
  - Gene prediction
  - Protein domain analysis
  - …
- An observed sequence data that can be modeled by a Markov chain
  - State path unknown
  - Model parameter known or unknown
- Observed data: emission sequences $X = (x_1 x_2 \ldots x_n)$
- Hidden data: state sequences $\Pi = (\pi_1 \pi_2 \ldots \pi_n)$

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?

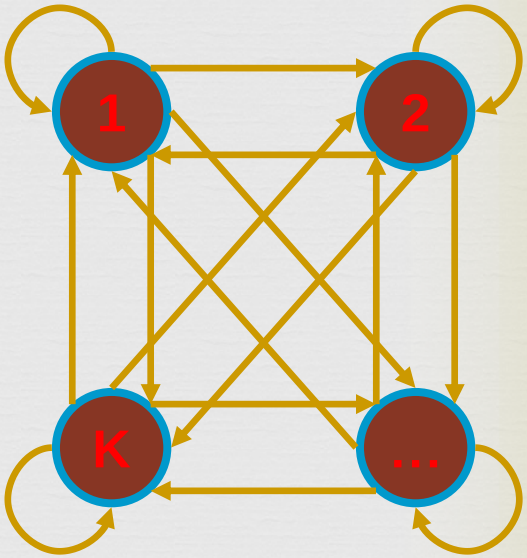
Why "Hidden"?

- 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.

Hidden Markov model (HMM)

Definition: A hidden Markov model (HMM) is a five-tuple

- Alphabet $\Sigma = \{ b_1, b_2, \ldots, b_M \}$
- Set of states $Q = \{ 1, \ldots, K \}$



- Transition probabilities between any two states

  a_{ij} = transition prob from state i to state j

  $a_{i1} + \ldots + a_{iK} = 1$, for all states $i = 1 \ldots K$

- Start probabilities a_{0i}

  $a_{01} + \ldots + a_{0K} = 1$

- Emission probabilities within each state

  $e_k(b) = P(x_i = b \mid \pi_i = k)$

  $e_k(b_1) + \ldots + e_k(b_M) = 1$, for all states $k = 1 \ldots K$

HMM Parameters

Σ: set of emission characters.

Ex.: Σ = {H, T} for coin tossing

Σ = {1, 2, 3, 4, 5, 6} for dice 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.

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

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

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

$e_F(0) = ½$ $e_F(1) = ½$

$e_B(0) = ¼$ $e_B(1) = ¾$

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.

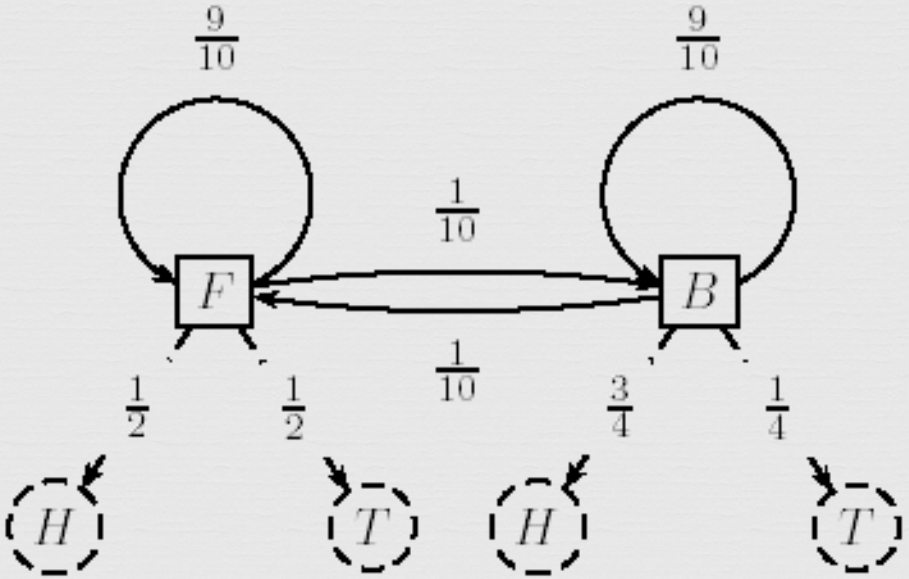
Transition Probabilities A

	Fair	Biased
Fair	$a_{FF} = 0.9$	$a_{FB} = 0.1$
Biased	$a_{BF} = 0.1$	$a_{BB} = 0.9$

Emission Probabilities E

	Tails(0)	Heads(1)
Fair	$e_F(0) = ½$	$e_F(1) = ½$
Biased	$e_B(0) = ¼$	$e_B(1) = ¾$

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 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

$P(x \mid \pi)$ Calculation

- $P(x \mid \pi)$: Probability that sequence x was generated by the path π:

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

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

$P(x \mid \pi)$ Calculation

- $P(x \mid \pi)$: Probability that sequence x was generated by the path π:

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

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

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

if we count from $i=0$ instead of $i=1$

Outline

- Introduction on Probability
- CG-islands
- The "Fair Bet Casino"
- Hidden Markov Model
- Decoding Algorithm
- Viterbi Algorithm
- Forward-Backward Algorithm

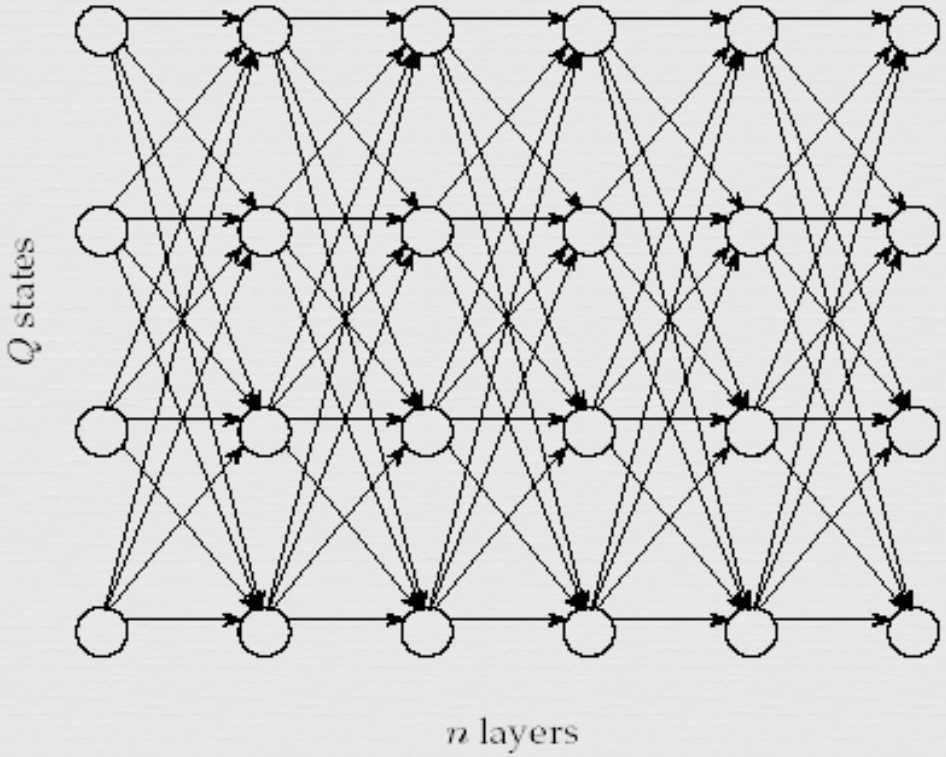
Decoding Problem

- **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 $P(x \mid \pi)$ over all possible paths π.

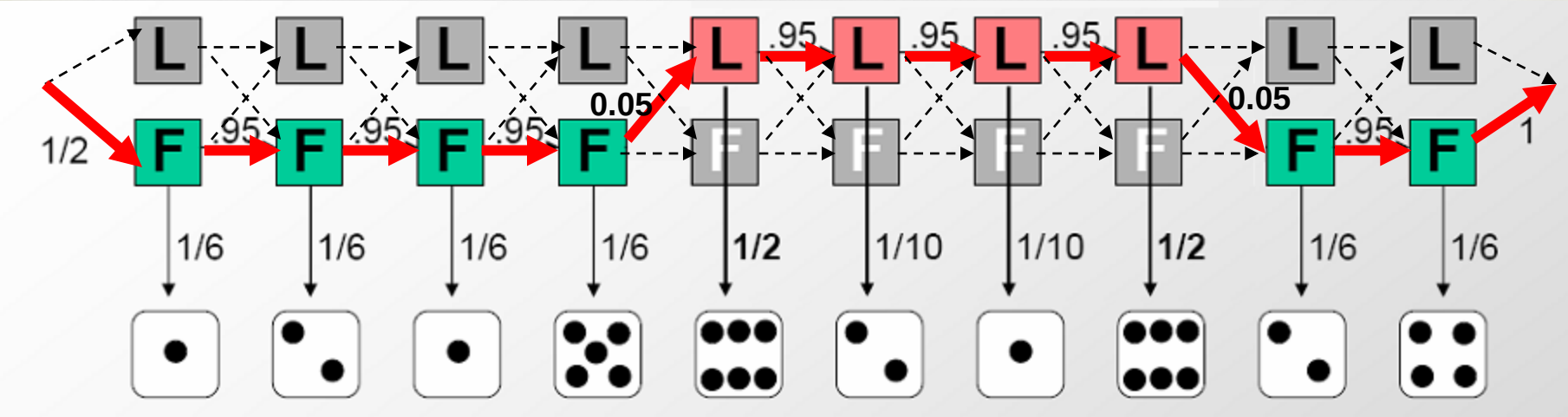
Building Manhattan for Decoding Problem

- Andrew Viterbi used the Manhattan grid model to solve the *Decoding Problem*.
- Every choice of $\pi = \pi_1 \ldots \pi_n$ corresponds to a path in the graph.
- The only valid direction in the graph is *eastward*.
- This graph has $|Q|^2(n-1)$ edges.

Edit Graph for Decoding Problem



Example



- What's the probability of
  - ∏ = Fair, Fair, Fair, Fair, Load, Load, Load, Load, Fair, Fair
  - X = 1, 2, 1, 5, 6, 2, 1, 6, 2, 4?

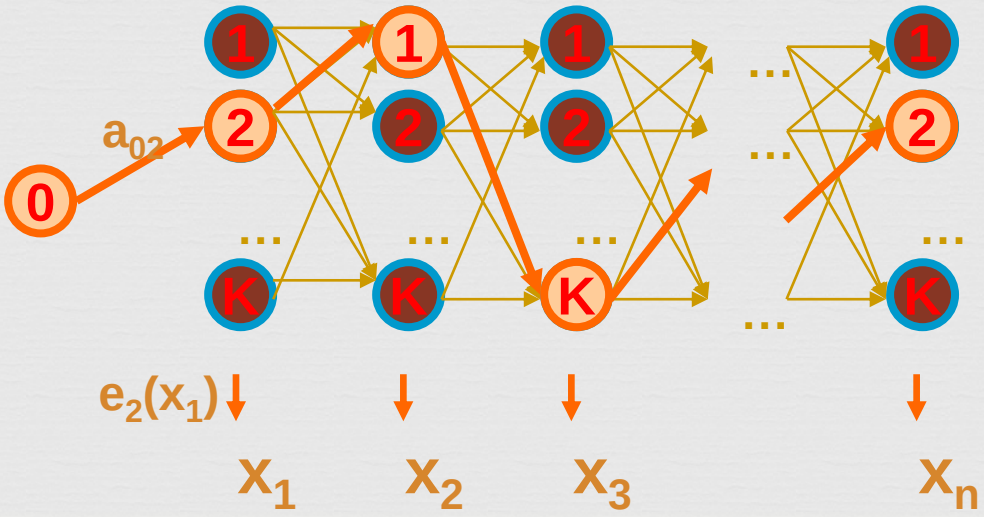
$P = ½ * P(1 \mid F)\, P(F_{i+1} \mid F_i) \ldots P(5 \mid F)\, P(L_{i+1} \mid F_i)\, P(6 \mid L)\, P(L_{i+1} \mid L_i) \ldots P(4 \mid F)$

$= ½ \times 0.95^7\, 0.05^2 \times (1/6)^6 \times (1/10)^2 \times (1/2)^2 = 5 \times 10^{-11}$

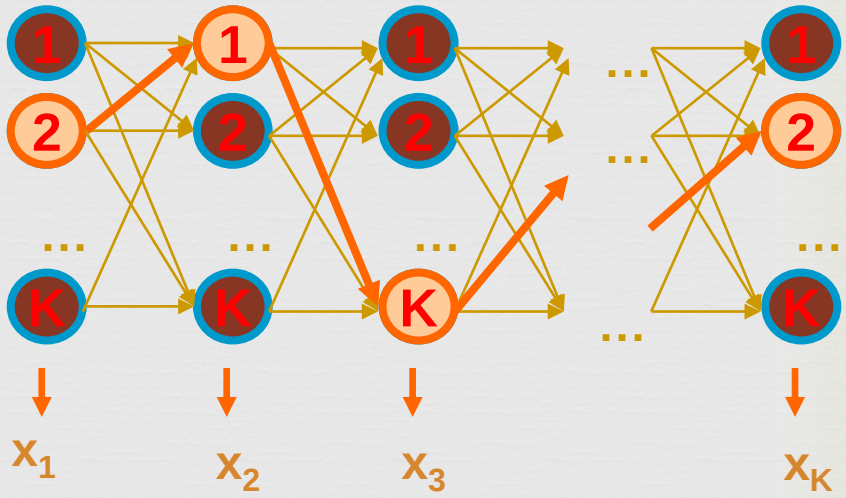
Generating a sequence by the model

Given a HMM, we can generate a sequence of length n as follows:

1. Start at state π_1 according to prob $a_{0\pi 1}$
2. Emit letter x_1 according to prob $e_{\pi 1}(x_1)$
3. Go to state π_2 according to prob $a_{\pi 1 \pi 2}$
4. … until emitting x_n



Probability of a sequence



Given a sequence $x = x_1 \ldots\ldots x_N$
and a parse $\pi = \pi_1, \ldots\ldots, \pi_N$

To find how likely is the parse:
(given our HMM)

$P(x, \pi) = P(x_1, \ldots, x_N, \pi_1, \ldots\ldots, \pi_N)$

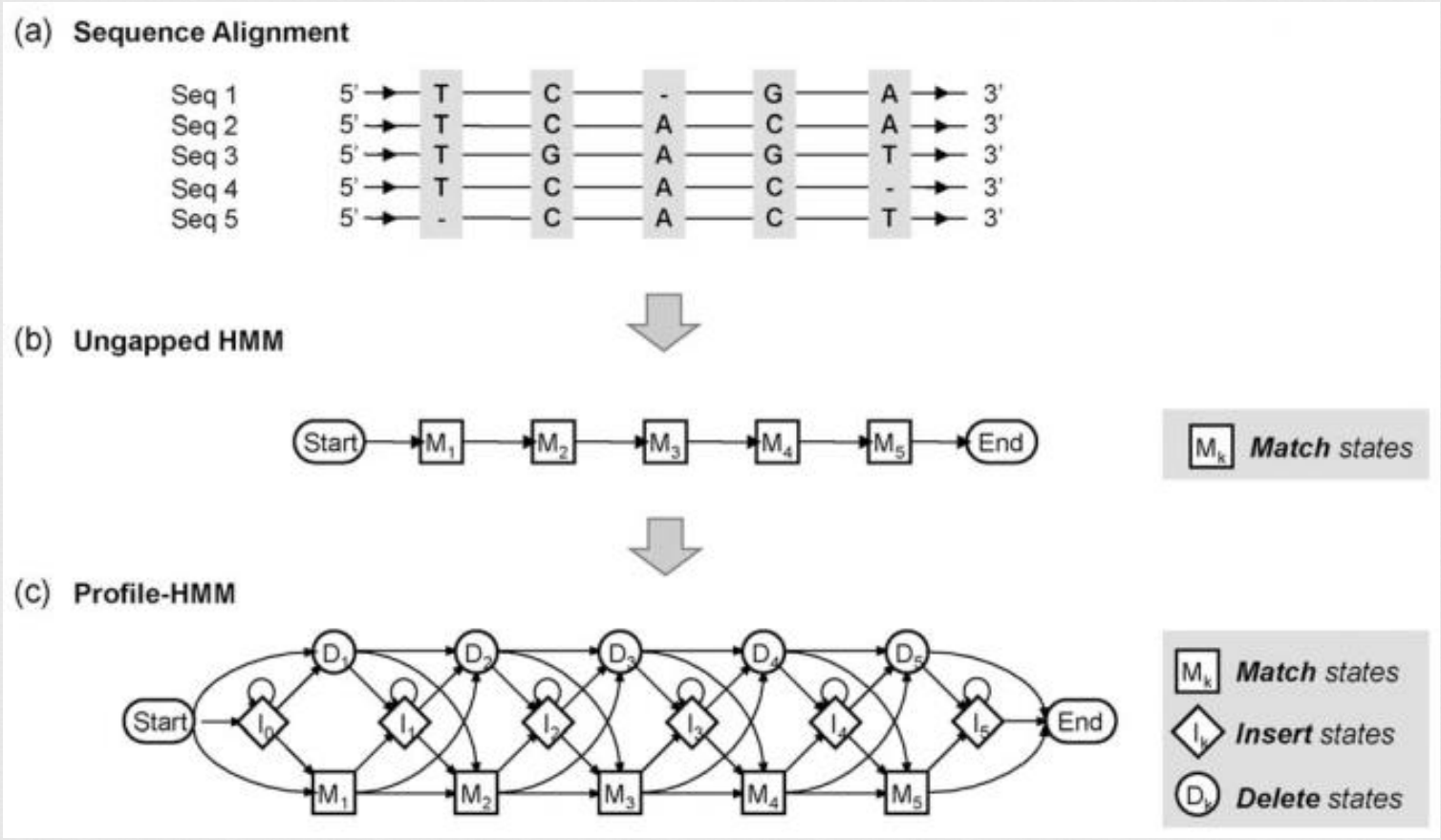
$= P(x_N, \pi_N \mid \pi_{N-1})\, P(x_{N-1}, \pi_{N-1} \mid \pi_{N-2}) \ldots\ldots P(x_2, \pi_2 \mid \pi_1)\, P(x_1, \pi_1)$

$= P(x_N \mid \pi_N)\, P(\pi_N \mid \pi_{N-1}) \ldots\ldots P(x_2 \mid \pi_2)\, P(\pi_2 \mid \pi_1)\, P(x_1 \mid \pi_1)\, P(\pi_1)$

$= a_{0\pi1}\, a_{\pi1\pi2} \ldots\ldots a_{\pi N-1 \pi N}\, e_{\pi1}(x_1) \ldots\ldots e_{\pi N}(x_N)$

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

Profile HMMs: Example1

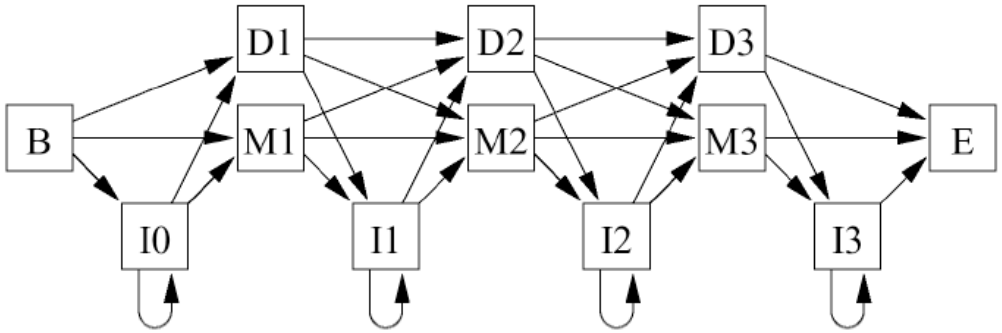


(**a**) Multiple sequence alignment for constructing the profile-HMM.
(**b**) The ungapped HMM that represents the consensus sequence of the alignment.
(**c**) The final profile-HMM that allows insertions and deletions.

Profile HMMs: Example2

An alignment of proteins from the HMM:

-	E	G	-	K	-
-	E	A	-	K	-
P	D	-	-	K	L
-	E	G	I	W	-



The states giving this alignment:

B → M1 → M2 → M3 → E

B → M1 → M2 → M3 → E

B → I0 → M1 → D2 → M3 → I3 → E

B → M1 → M2 → I2 → M3 → E

Outline

- Introduction on Probability
- CG-islands
- The "Fair Bet Casino"
- Hidden Markov Model
- Decoding Algorithm
- Viterbi Algorithm
- Forward-Backward Algorithm

Decoding Problem as Finding a Longest Path in a DAG

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

- **Notes:** the length of the path is defined as the *product* of its edges' weights, not the *sum*.

Decoding

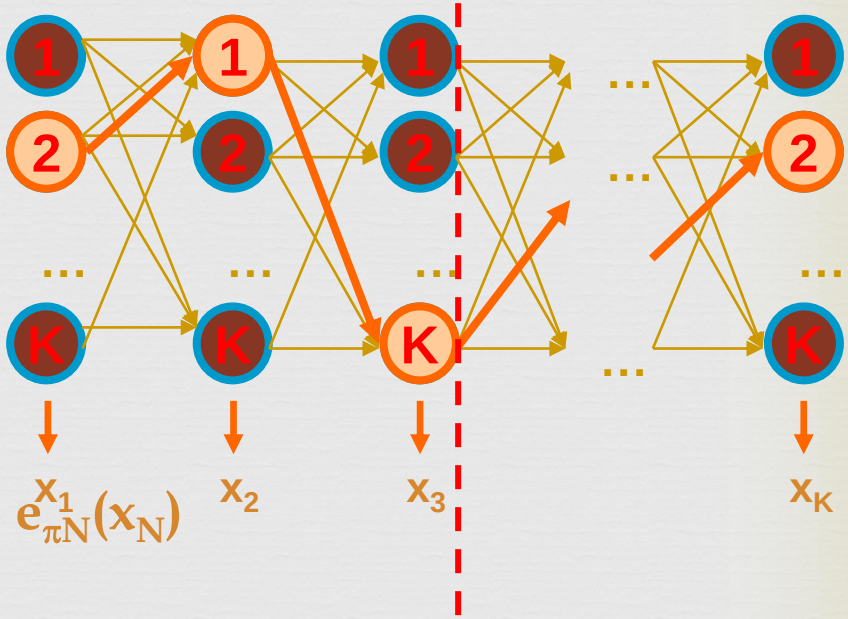
GIVEN $x = x_1 x_2 \ldots\ldots x_N$

Find $\pi = \pi_1, \ldots\ldots, \pi_N$,
to maximize $P[x, \pi]$

$\pi^* = \text{argmax}_\pi P[x, \pi]$

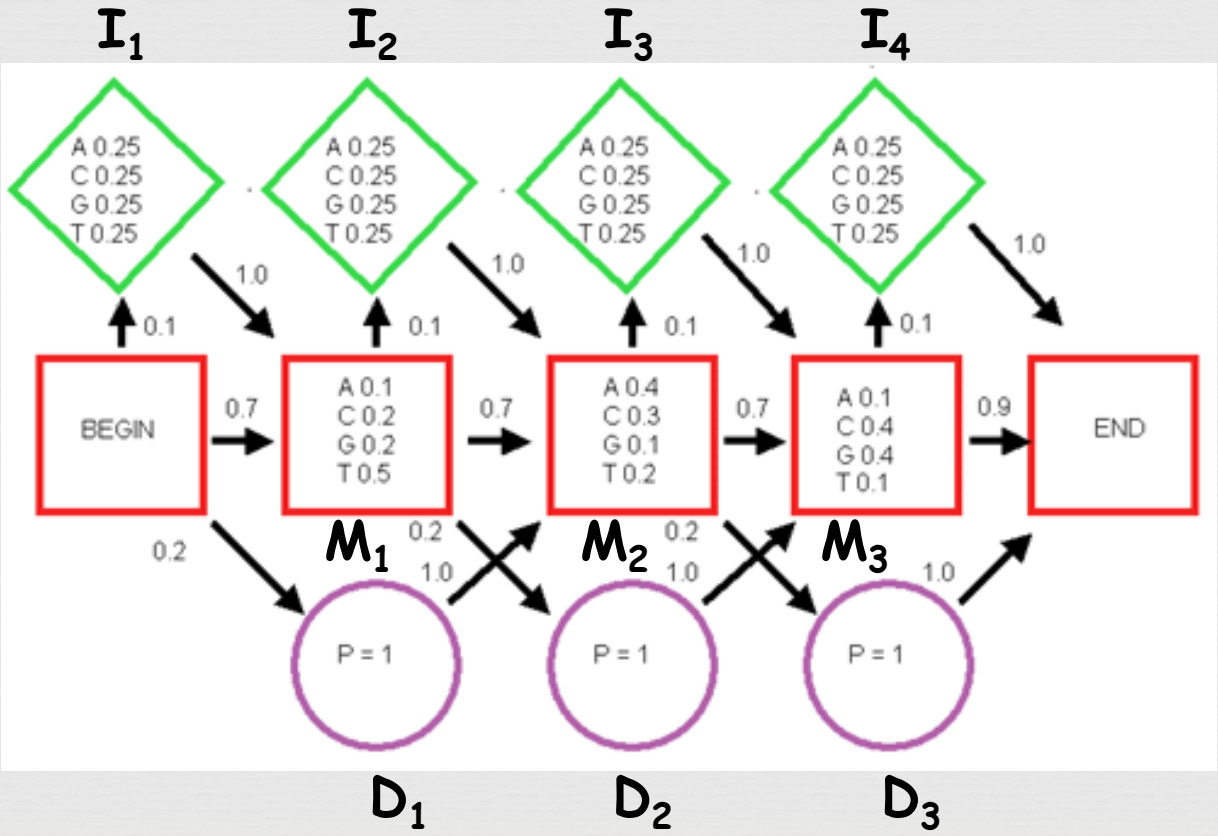
Maximizes $a_{0\pi 1}\, e_{\pi 1}(x_1)\, a_{\pi 1 \pi 2} \ldots\ldots a_{\pi N-1 \pi N}\, e_{\pi N}(x_N)$

Dynamic Programming!



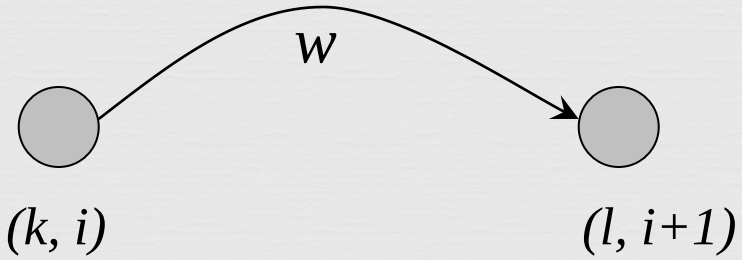
Given that we end up in state k at step i, maximize product to the left and right

Decoding Problem: weights of edges



Decoding Problem: weights of edges

$$P(x|\pi) = \prod_{i=0}^{n} e_{\pi_{i+1}}(x_{i+1}) \cdot a_{\pi_i, \pi_{i+1}}$$

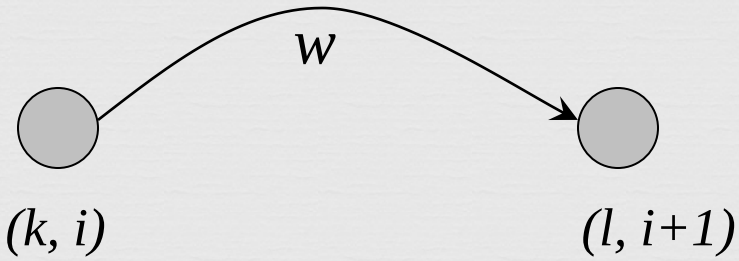


The weight ***w*** is given by:

??

Decoding Problem: weights of edges

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



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

Decoding Problem

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

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

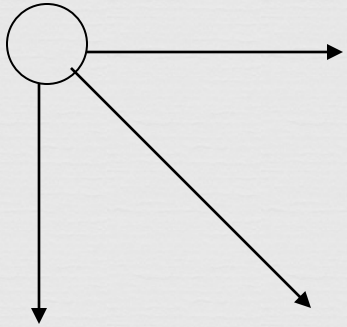
$e_l(x_{i+1}) \cdot \max_{k \in Q} \{s_{k,i} \cdot a_{kl}\}$

Decoding Problem (cont'd)

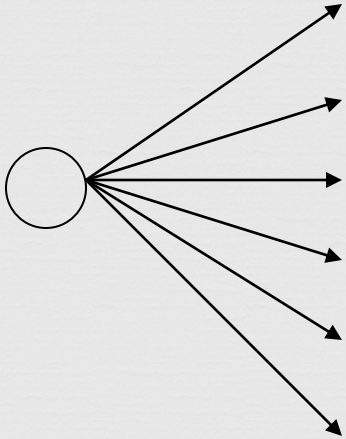
- Initialization:
  - $s_{begin,0} = 1$
  - $s_{k,0} = 0$ for $k \neq begin$.
- Let π^* be the optimal path. Then,

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

Decoding Problem vs. Alignment Problem



Valid directions in the *alignment problem.*



Valid directions in the *decoding problem.*

Viterbi Algorithm

- 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 $O(n \mid Q \mid^2)$ time.

Viterbi Algorithm

- The value of the product can become extremely small, which leads to overflowing.
- To avoid overflowing, use log value instead.

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

where, $s_{l,i+1}$ and $s_{k,i}$ are logarithmic values

Viterbi Alg.– main idea

Induction: Given that for all states k, and for a fixed position i,

$V_k(i) = \max_{\{\pi 1 \ldots \pi i-1\}} P[x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-1}, x_i, \pi_i = k]$

What is $V_l(i+1)$?

From definition,

$$
\begin{aligned}
V_l(i+1) &= \max_{\{\pi 1 \ldots \pi i\}} P[\, x_1 \ldots x_i, \pi_1, \ldots, \pi_i, x_{i+1}, \pi_{i+1} = l \,] \\
&= \max_{\{\pi 1 \ldots \pi i\}} P(x_{i+1}, \pi_{i+1} = l \mid x_1 \ldots x_i, \pi_1, \ldots, \pi_i)\, P[x_1 \ldots x_i, \pi_1, \ldots, \pi_i] \\
&= \max_{\{\pi 1 \ldots \pi i\}} P(x_{i+1}, \pi_{i+1} = l \mid \pi_i)\, P[x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-1}, x_i, \pi_i] \\
&= \max_k \left[P(x_{i+1}, \pi_{i+1} = l \mid \pi_i = k) \max_{\{\pi 1 \ldots \pi i-1\}} P[x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-1}, x_i, \pi_i = k] \right] \\
&= \max_k \left[P(x_{i+1} \mid \pi_{i+1} = l)\, P(\pi_{i+1} = l \mid \pi_i = k)\, V_k(i) \right] \\
&= e_l(x_{i+1}) \max_k a_{kl}\, V_k(i)
\end{aligned}
$$

The Viterbi Algorithm

Input: $x = x_1 \ldots\ldots x_N$

Initialization:

$V_0(0) = 1$ (0 is the imaginary first position)

$V_k(0) = 0$, for all $k > 0$

Iteration:

$V_j(i) = e_j(x_i) \times \max_k a_{kj} V_k(i - 1)$

$Ptr_j(i) = \text{argmax}_k a_{kj} V_k(i - 1)$

Termination:

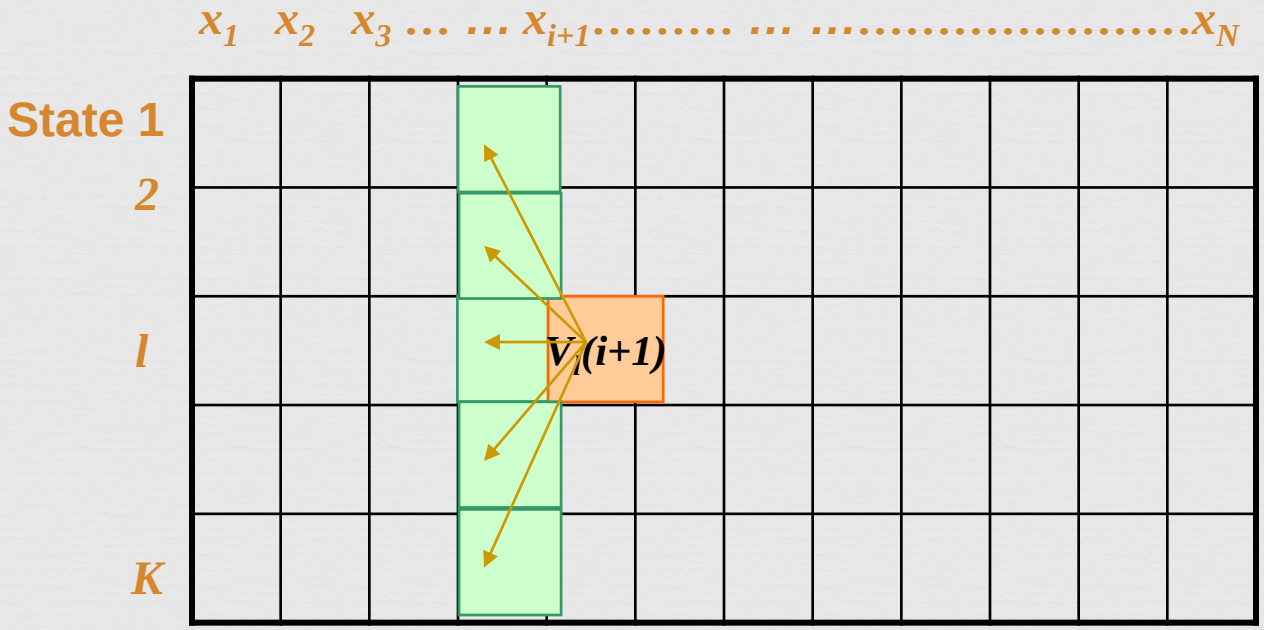
$P(x, \pi^*) = \max_k V_k(N)$

Traceback:

$\pi_N^* = \text{argmax}_k V_k(N)$

$\pi_{i-1}^* = Ptr_{\pi i}(i)$

The Viterbi Algorithm



Similar to "aligning" a set of states to a sequence

Time: O(K^2N)

Space: O(KN)

$$V_l(i+1) = r_l(x_{i+1}) + \max_{k=1..K}(V_k(i) + w_{kl})$$

Outline

- Introduction on Probability
- CG-islands
- The "Fair Bet Casino"
- Hidden Markov Model
- Decoding Algorithm
- Viterbi Algorithm
- Forward-Backward Algorithm

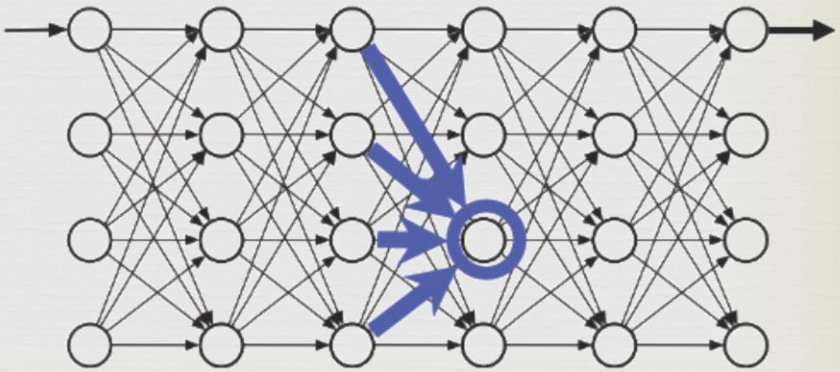
Forward Algorithm

- 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:

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

Forward Algorithm

$$
\begin{aligned}
f_k(i) &= P(x_1 \ldots x_i, \pi_i = k) \\
&= P(x_1 \ldots x_{i-1}, \pi_i = k) P(x_i \mid \pi_i = k) \\
&= P(x_i \mid \pi_i = k) \sum_j P(x_1 \ldots x_{i-1}, \pi_{i-1} = j, \pi_i = k) \\
&= e_k(x_i) \sum_j P(x_1 \ldots x_{i-1}, \pi_{i-1} = j) P(\pi_i = k \mid \pi_{i-1} = j) \\
&= e_k(x_i) \sum_j P(x_1 \ldots x_{i-1}, \pi_{i-1} = j) a_{jk} \\
&= e_k(x_i) \sum_j f_j(i-1) a_{jk}
\end{aligned}
$$



The Forward Algorithm

We can compute $f_k(i)$ for all k, i, using dynamic programming!

Initialization:

$f_0(0) = 1$

$f_k(0) = 0$, for all $k > 0$

Iteration:

$$f_k(i) = e_k(x_i) \sum_l f_l(i-1)\, a_{lk}$$

Termination:

$$P(x) = \sum_k f_k(N)$$

Relation between Forward and Viterbi

VITERBI	FORWARD
Initialization:	**Initialization:**
$V_0(0) = 1$	$\mathbf{f_0(0) = 1}$
$V_k(0) = 0$, for all $k > 0$	$\mathbf{f_k(0) = 0}$, **for all** $\mathbf{k > 0}$
Iteration:	**Iteration:**
$V_j(i) = e_j(x_i)\ \max_k V_k(i-1)\, a_{kj}$	$\mathbf{f_l(i) = e_l(x_i) \sum_k f_k(i-1)\, a_{kl}}$
Termination:	**Termination:**
$P(x, \pi^*) = \max_k V_k(N)$	$\mathbf{P(x) = \sum_k f_k(N)}$

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 $P(\pi_i = k|x)$.

Backward Algorithm

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

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

The Backward Algorithm

Define the backward probability:

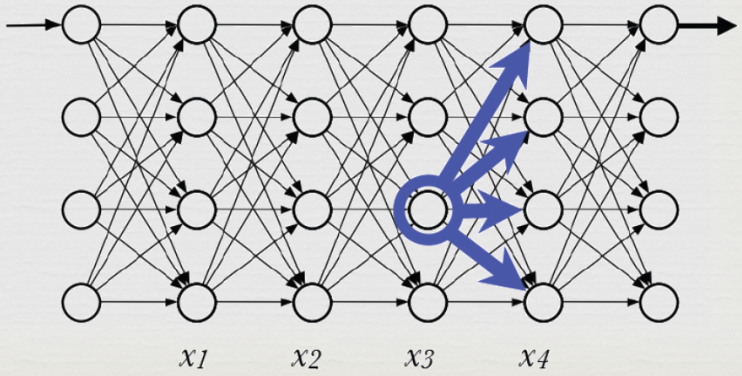
$b_k(i) = P(x_{i+1} \ldots x_N \mid \pi_i = k)$ *"starting from i^{th} state = k, generate rest of x"*

$= \sum_{\pi i+1 \ldots \pi N} P(x_{i+1}, x_{i+2}, \ldots, x_N, \pi_{i+1}, \ldots, \pi_N \mid \pi_i = k)$

$= \sum_l \sum_{\pi i+1 \ldots \pi N} P(x_{i+1}, x_{i+2}, \ldots, x_N, \pi_{i+1} = l, \pi_{i+2}, \ldots, \pi_N \mid \pi_i = k)$

$= \sum_l e_l(x_{i+1})\, a_{kl} \sum_{\pi i+1 \ldots \pi N} P(x_{i+2}, \ldots, x_N, \pi_{i+2}, \ldots, \pi_N \mid \pi_{i+1} = l)$

$= \sum_l e_l(x_{i+1})\, a_{kl}\, b_l(i+1)$



The Backward Algorithm

We can compute $b_k(i)$ for all k, i, using dynamic programming

Initialization:

$b_k(N) = 1$, for all k

Iteration:

$b_k(i) = \sum_l e_l(x_{i+1})\, a_{kl}\, b_l(i+1)$

Termination:

$P(x) = \sum_l a_{0l}\, e_l(x_1)\, b_l(1)$

Viterbi, Forward, Backward

VITERBI

Initialization:

$V_0(0) = 1$

$V_k(0) = 0$, for all $k > 0$

Iteration:

$V_l(i) = e_l(x_i)\ \max_k V_k(i-1)\ a_{kl}$

Termination:

$P(x, \pi^*) = \max_k V_k(N)$

FORWARD

Initialization:

$f_0(0) = 1$

$f_k(0) = 0$, for all $k > 0$

Iteration:

$f_l(i) = e_l(x_i) \sum_k f_k(i-1)\ a_{kl}$

Termination:

$P(x) = \sum_k f_k(N)$

BACKWARD

Initialization:

$b_k(N) = 1$, for all k

Iteration:

$b_l(i) = \sum_k e_l(x_i+1)\ a_{kl}\ b_k(i+1)$

Termination:

$P(x) = \sum_k a_{0k}\ e_k(x_1)\ b_k(1)$

Forward-Backward Problem

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

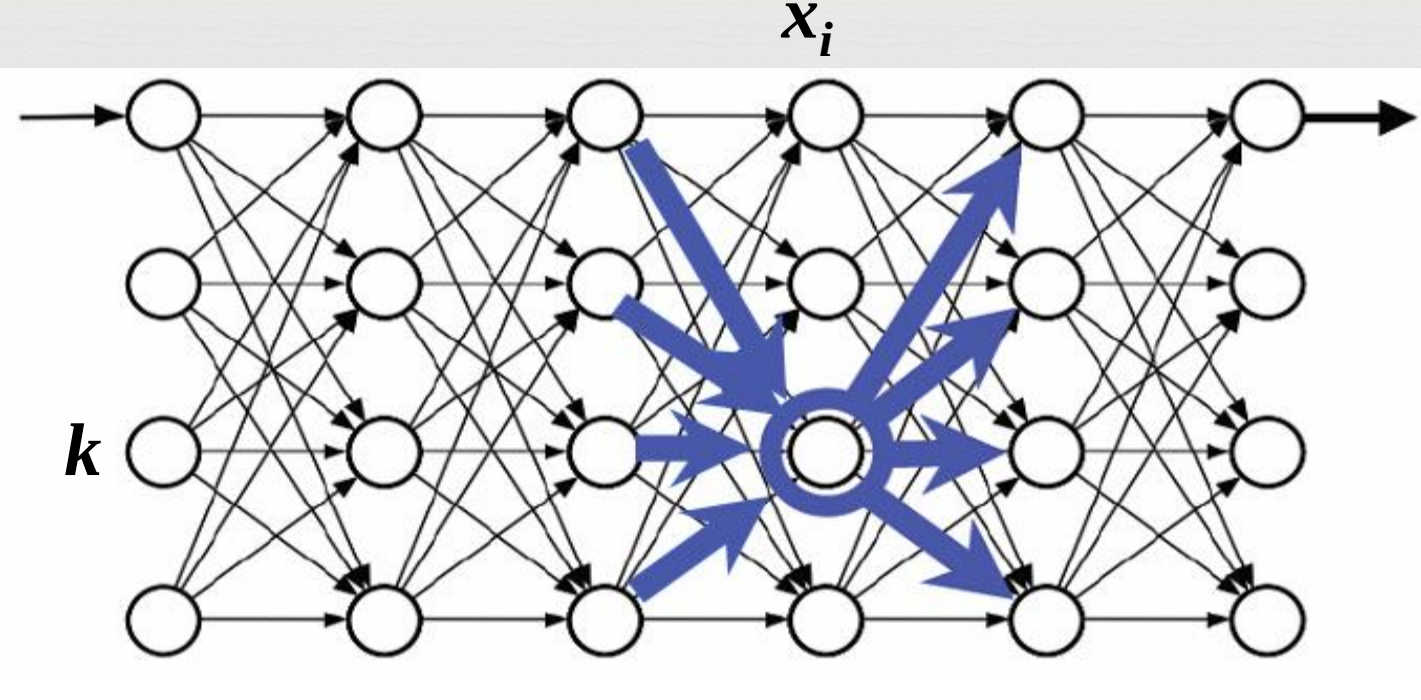
Goal: find the probability that the dealer was using a biased coin at a particular time.

Backward-Forward Algorithm

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

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

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



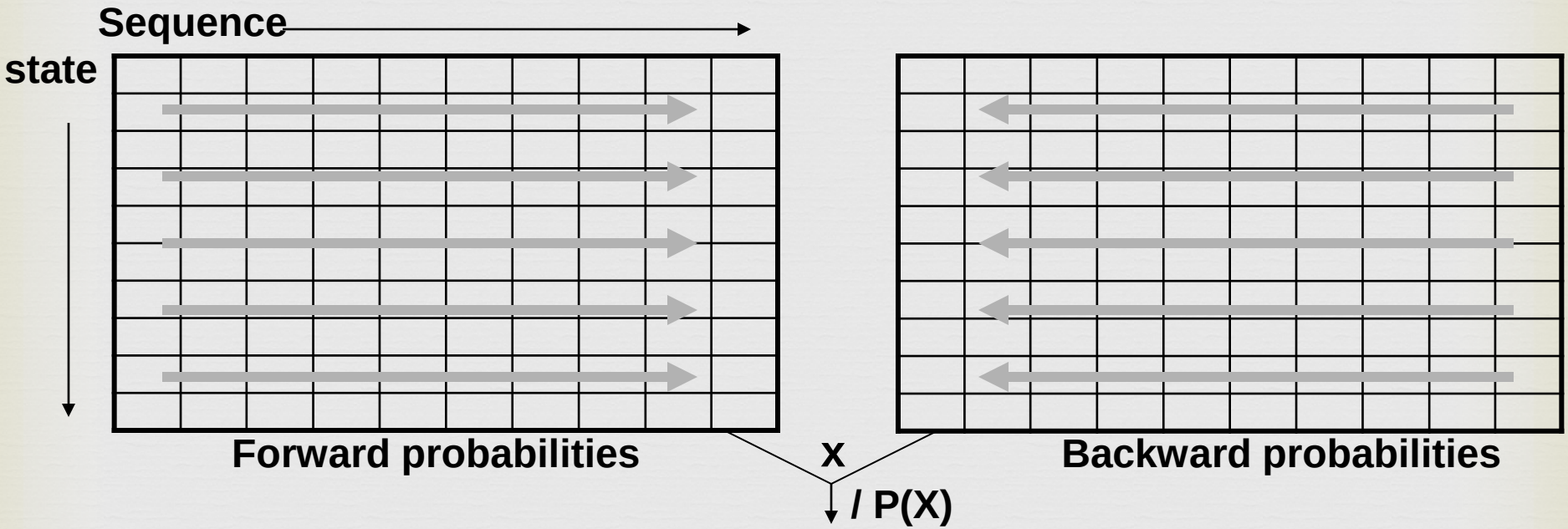
$$
\begin{aligned}
& P(x,\, \pi_i = k) \\
&= \quad P(x_1, \ldots, x_i,\, \pi_i = k) \cdot P(x_{i+1}, \ldots, x_n \mid x_1, \ldots, x_i,\, \pi_i = k) \\
&= \quad P(x_1, \ldots, x_i,\, \pi_i = k) \cdot P(x_{i+1}, \ldots, x_n \mid \pi_i = k) \\
&= \quad f_k(i) \cdot b_k(i)
\end{aligned}
$$

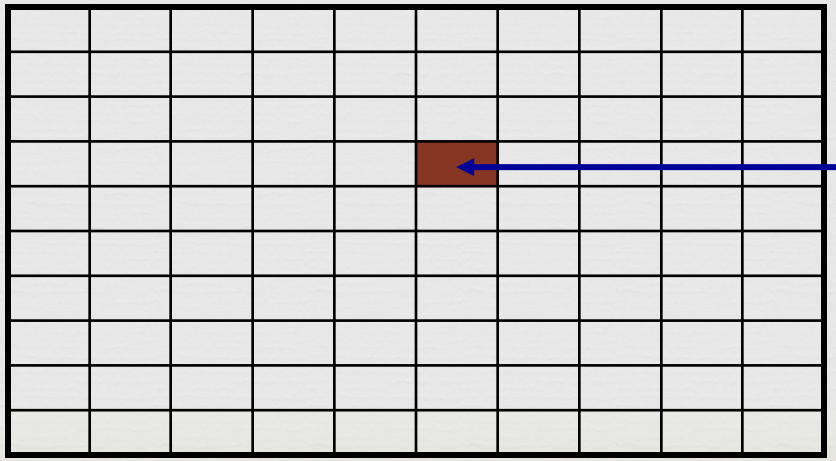
Backward-Forward Algorithm

- Compute $f_k(i)$, for each state k and pos i
- Compute $b_k(i)$, for each state k and pos I
- Compute $P(x) = \Sigma_k f_k(N)$
- Compute $P(\pi_i=k \mid x) = f_k(i) * b_k(i) / P(x)$

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

The prob of x, with the constraint that x_i was generated by state k





Space: O(KN)

Time: O(K^2N)