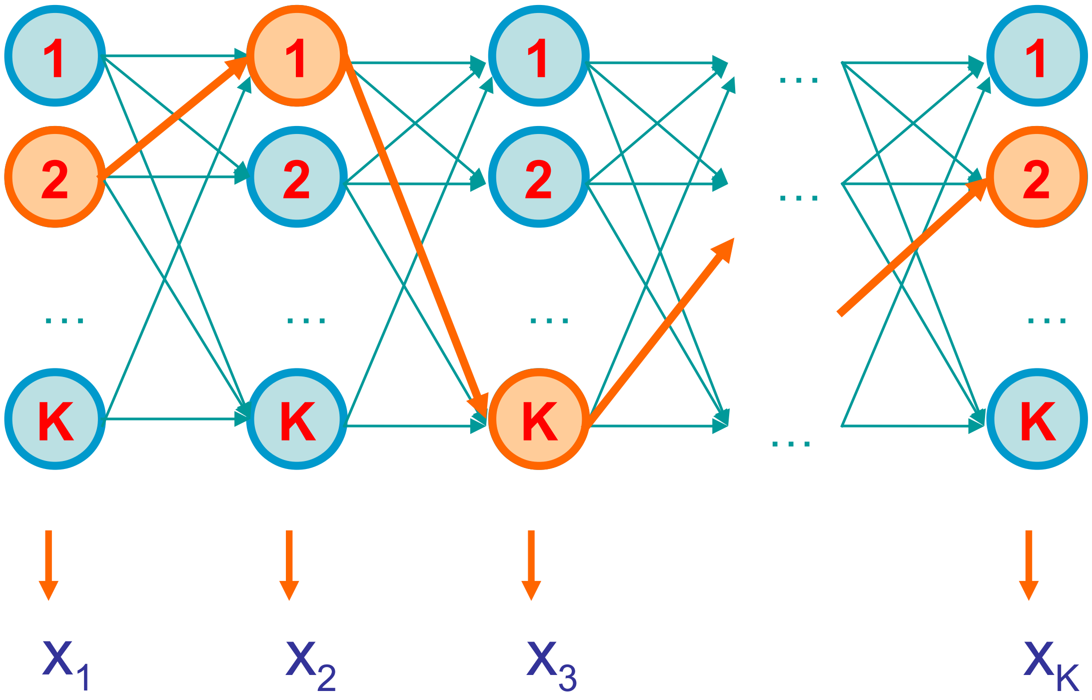
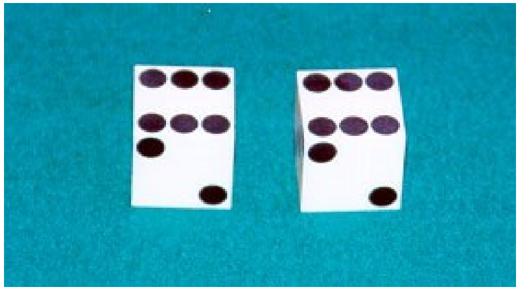
Hidden Markov Models



Example: The dishonest casino



A casino has two dice:

- Fair die
  P(1) = P(2) = P(3) = P(4) = P(5) = P(6) = 1/6
- Loaded die
  P(1) = P(2) = P(3) = P(4) = P(5) = 1/10
  P(6) = 1/2

Casino player switches between fair and loaded die with probability 1/20 at each turn

<u>Game:</u>

1. You bet $1
2. You roll (always with a fair die)
3. Casino player rolls (maybe with fair die, maybe with loaded die)
4. Highest number wins $2

Question # 1 – Decoding

GIVEN

A sequence of rolls by the casino player

124552646214614613613	666166466163661636616361651	5615115146123562344
FAIR	LOADED	FAIR

QUESTION

What portion of the sequence was generated with the fair die, and what portion with the loaded die?

This is the **DECODING** question in HMMs

Question # 2 – Evaluation

GIVEN

A sequence of rolls by the casino player

$$\underbrace{\mathbf{1245526462146146136136661664661636616366163616515615115146123562344}}_{\text{Prob} = 1.3 \times 10^{-35}}$$

QUESTION

How likely is this sequence, given our model of how the casino works?

This is the **EVALUATION** problem in HMMs

Question # 3 – Learning

GIVEN

A sequence of rolls by the casino player

124552646214614613613666166466163661636616361651561511514612356234

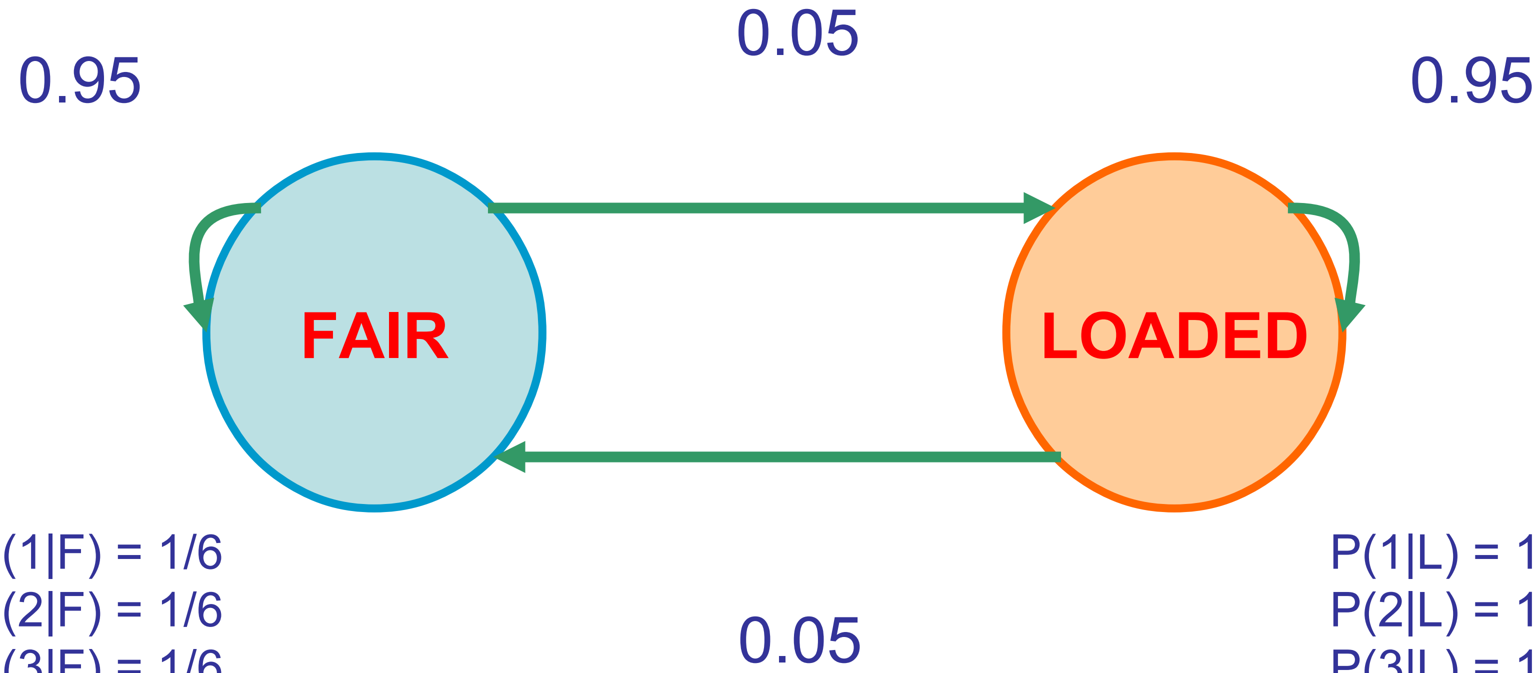
Prob(6) = 64%

QUESTION

How “loaded” is the loaded die? How “fair” is the fair die? How often does the casino player change from fair to loaded, and back?

This is the **LEARNING** question in HMMs

The dishonest casino model



P(1|F) = 1/6
P(2|F) = 1/6
P(3|F) = 1/6
P(4|F) = 1/6
P(5|F) = 1/6
P(6|F) = 1/6

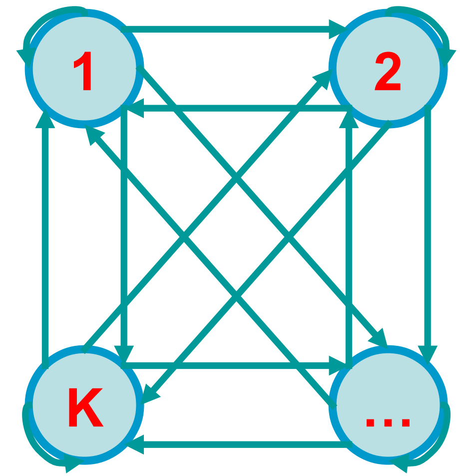
P(1|L) = 1/10
P(2|L) = 1/10
P(3|L) = 1/10
P(4|L) = 1/10
P(5|L) = 1/10
P(6|L) = 1/2

An HMM is memoryless

At each time step t,

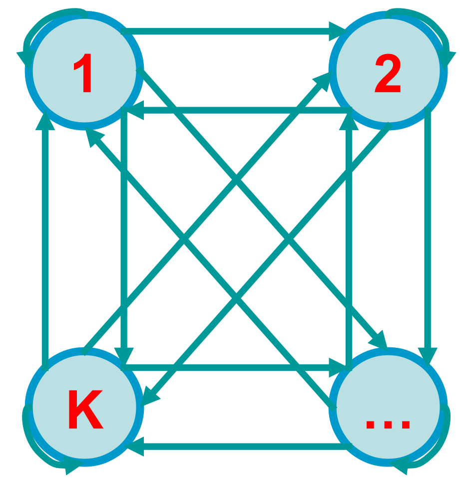
the only thing that affects future states

is the current state π_t



An HMM is memoryless

At each time step t,
the only thing that affects future states
is the current state π_t



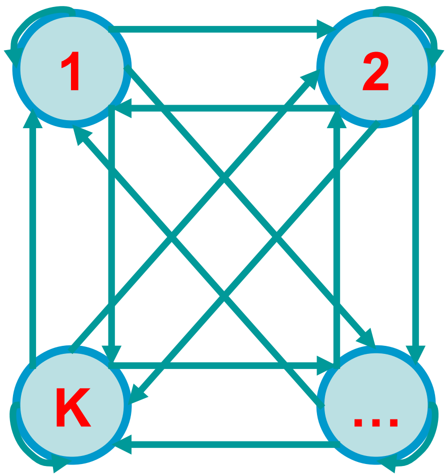
$$P(\pi_{t+1} = k \mid \text{"whatever happened so far"}) =$$
$$P(\pi_{t+1} = k \mid \pi_1, \pi_2, \ldots, \pi_t, x_1, x_2, \ldots, x_t) =$$
$$P(\pi_{t+1} = k \mid \pi_t)$$

An HMM is memoryless

At each time step t,

the only thing that affects x_t

is the current state π_t



$P(x_t = b \mid \text{“whatever happened so far”}) =$

$P(x_t = b \mid \pi_1, \pi_2, \ldots, \pi_t, x_1, x_2, \ldots, x_{t-1}) =$

$P(x_t = b \mid \pi_t)$

Definition of a hidden Markov model

Definition: A hidden Markov model (HMM)

- 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 $i = 1 \ldots K$

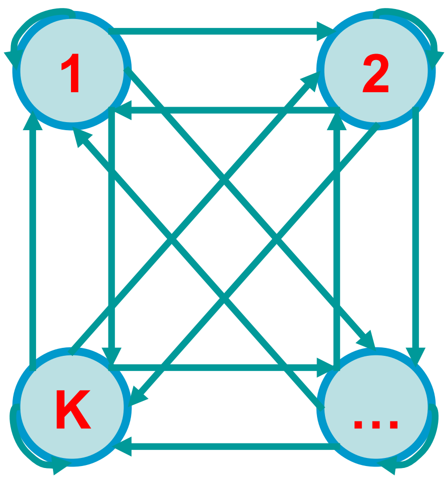
- Start probabilities a_{0i}

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

- Emission probabilities within each state

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

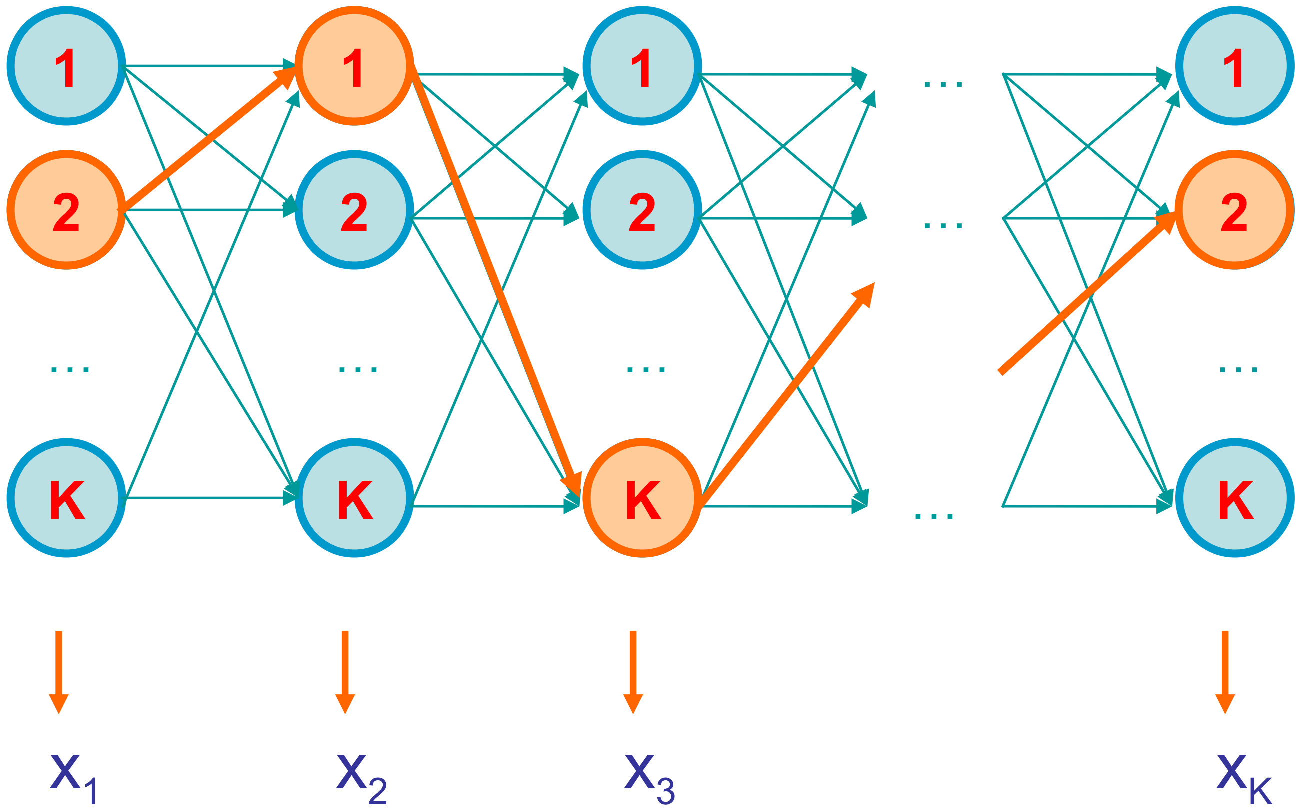
  $e_i(b_1) + \ldots + e_i(b_M) = 1$, for all states $i = 1 \ldots K$



A parse of a sequence

Given a sequence $x = x_1 \ldots\ldots x_N$,

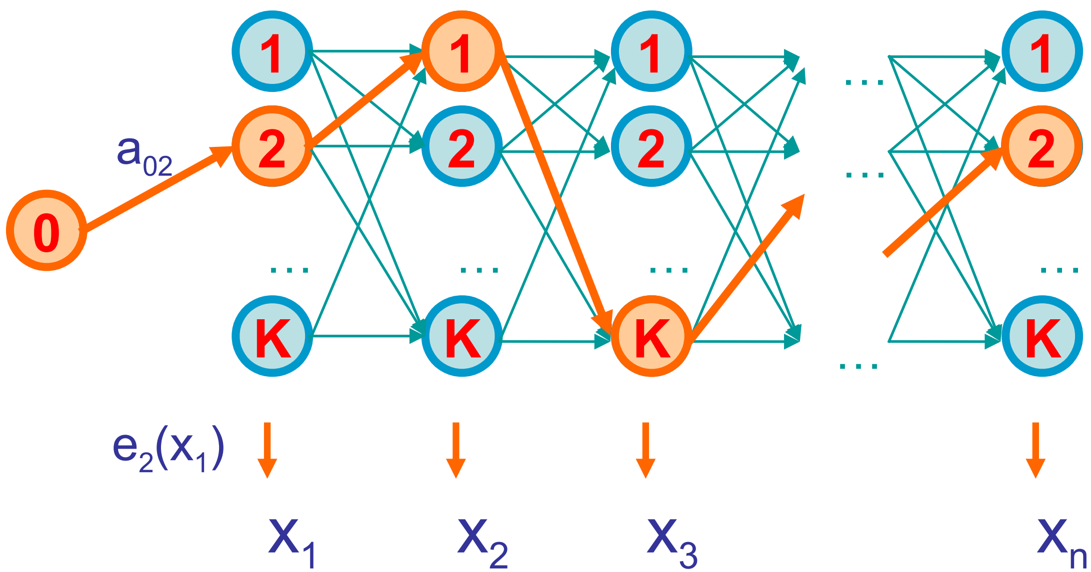
A <u>parse</u> of x is a sequence of states $\pi = \pi_1, \ldots\ldots, \pi_N$



Generating a sequence by the model

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

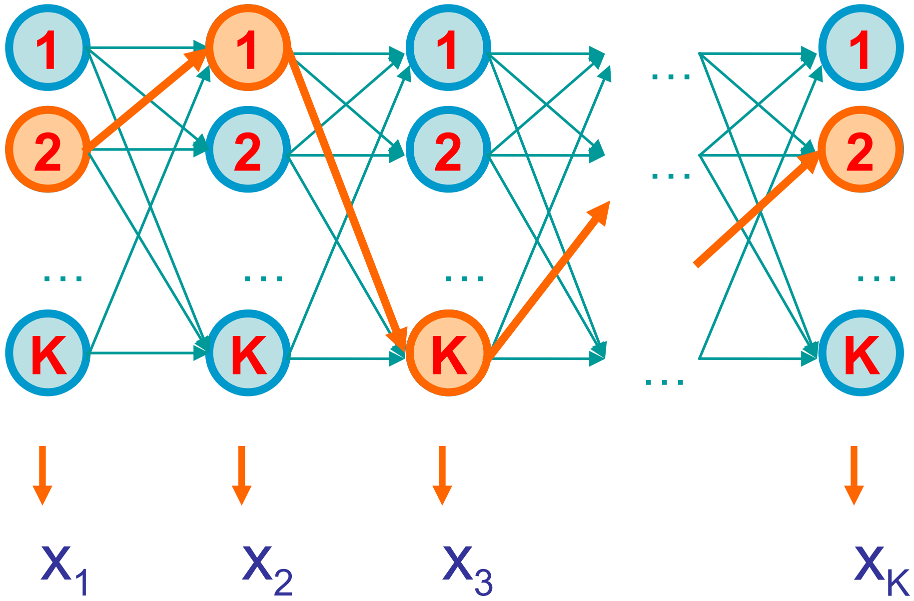
1. Start at state π_1 according to prob $a_{0\pi 1}$
2. Emit letter x_1 according to prob $e_{\pi 1}(x_1)$
3. Go to state π_2 according to prob $a_{\pi 1 \pi 2}$
4. … until emitting x_n



Likelihood of a parse

Given a sequence $x = x_1 \ldots\ldots x_N$

and a parse $\pi = \pi_1, \ldots\ldots, \pi_N$,



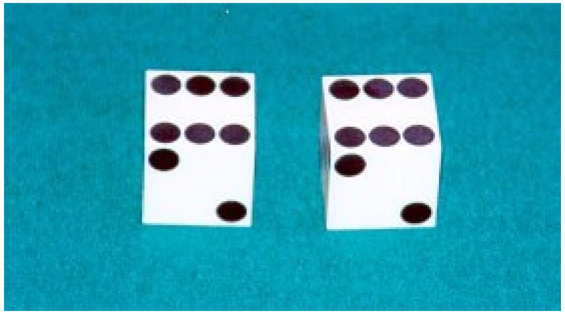
To find how likely this scenario is:
(given our HMM)

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

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

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

Example: the dishonest casino



Let the sequence of rolls be:

x = 1, 2, 1, 5, 6, 2, 1, 5, 2, 4

Then, what is the likelihood of

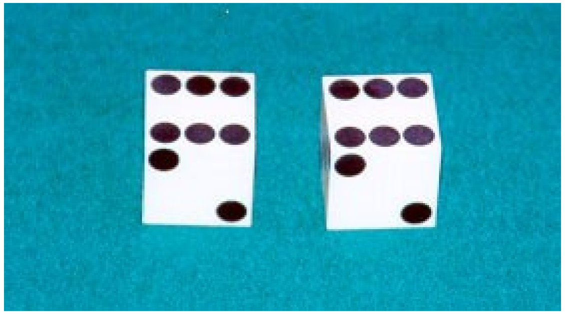
π = Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair?

(say initial probs $a_{0Fair} = ½$, $a_{oLoaded} = ½$)

$½ \times P(1 \mid Fair)\, P(Fair \mid Fair)\, P(2 \mid Fair)\, P(Fair \mid Fair) \ldots P(4 \mid Fair) =$

$½ \times (1/6)^{10} \times (0.95)^{9} = .00000000521158647211 \sim= 0.5 \times 10^{-9}$

Example: the dishonest casino



So, the likelihood the die is fair in this run is just 0.521×10^{-9}

What is the likelihood of

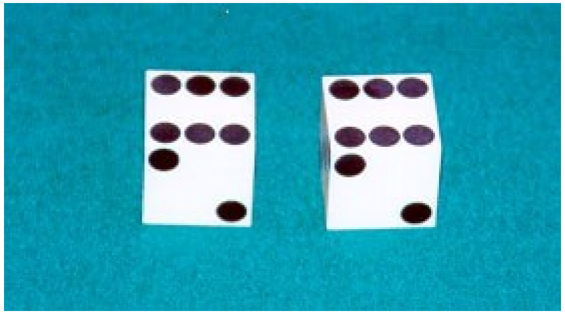
π = Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded?

$\frac{1}{2} \times P(1 \mid \text{Loaded})\, P(\text{Loaded}, \text{Loaded}) \ldots P(4 \mid \text{Loaded}) =$

$\frac{1}{2} \times (1/10)^9 \times (1/2)^1 (0.95)^9 = .00000000015756235243 \sim= 0.16 \times 10^{-9}$

Therefore, it's somewhat more likely that all the rolls are done with the fair die, than that they are all done with the loaded die

Example: the dishonest casino



Let the sequence of rolls be:

x = 1, 6, 6, 5, 6, 2, 6, 6, 3, 6

Now, what is the likelihood π = F, F, …, F?

$\frac{1}{2} \times (1/6)^{10} \times (0.95)^{9}$ ~= 0.5×10^{-9}, same as before

What is the likelihood

π = L, L, …, L?

$\frac{1}{2} \times (1/10)^{4} \times (1/2)^{6} (0.95)^{9} = .00000049238235134735$ ~= 0.5×10^{-7}

So, it is 100 times more likely the die is loaded

The three main questions on HMMs

1. **Decoding**

GIVEN a HMM M, and a sequence x,
FIND the sequence π of states that maximizes $P[\ x, \pi \mid M\ ]$

2. **Evaluation**

GIVEN a HMM M, and a sequence x,
FIND $Prob[\ x \mid M\ ]$

3. **Learning**

GIVEN a HMM M, with unspecified transition/emission probs., and a sequence x,
FIND parameters $\theta = (e_i(.), a_{ij})$ that maximize $P[\ x \mid \theta\ ]$

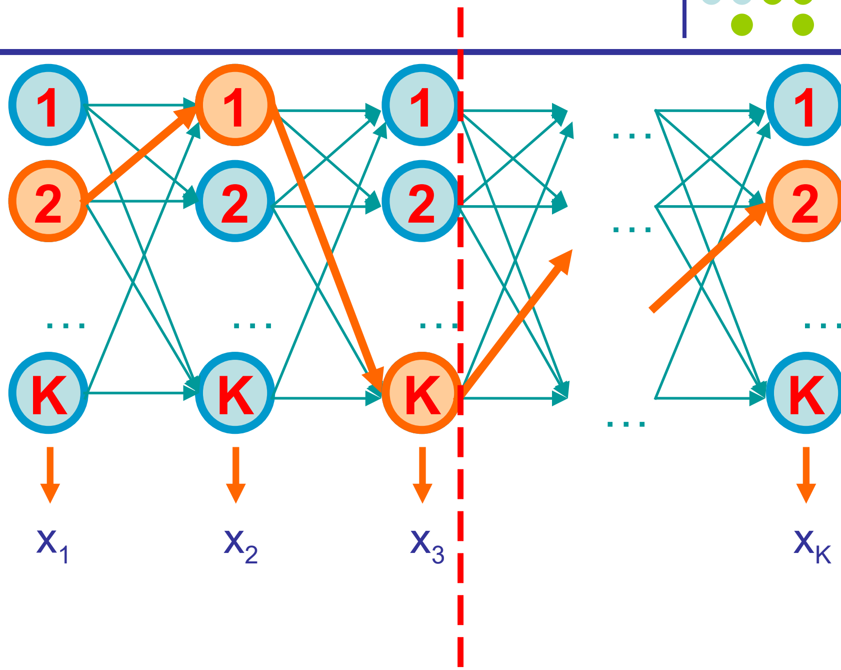
Problem 1: Decoding

Find the most likely parse of a sequence

Decoding

GIVEN $x = x_1x_2\ldots\ldots x_N$

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



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

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

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

Dynamic Programming!

$V_k(i) = \max_{\{\pi1\ldots\,\pi i\text{-}1\}} P[x_1\ldots x_{i\text{-}1}, \pi_1, \ldots, \pi_{i\text{-}1}, x_i, \pi_i = k]$

$=$ Prob. of most likely sequence of states ending at state $\pi_i = k$

Decoding – main idea

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

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

What is $V_l(i+1)$?

From definition,

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

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:

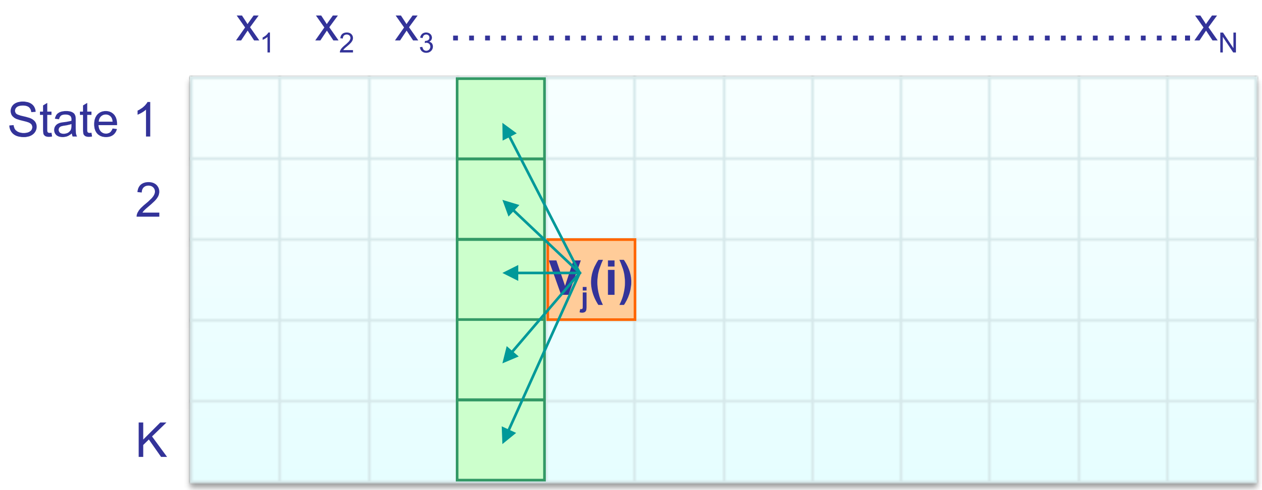
$P(x, \pi^*) = \max_k V_k(N)$

Traceback:

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

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

The Viterbi Algorithm



Time:

$O(K^2N)$

Space:

$O(KN)$

Viterbi Algorithm – a practical detail

Underflows are a significant problem

$$P[\, x_1, \ldots, x_i, \pi_1, \ldots, \pi_i \,] = a_{0\pi_1} a_{\pi_1\pi_2} \ldots\ldots a_{\pi_i} e_{\pi_1}(x_1) \ldots\ldots e_{\pi_i}(x_i)$$

These numbers become extremely small – underflow

Solution: Take the logs of all values

$$V_l(i) = \log e_k(x_i) + \max_k [\, V_k(i-1) + \log a_{kl} \,]$$

Example

Let x be a long sequence with a portion of ~ 1/6 6's,
followed by a portion of ~ ½ 6's…

x = 123456123456…12345 6626364656…1626364656

Then, it is not hard to show that optimal parse is (exercise):

FFF…………………...F LLL………………………...L

6 characters "123456" parsed as F, contribute $.95^6 \times (1/6)^6 = 1.6 \times 10^{-5}$
parsed as L, contribute $.95^6 \times (1/2)^1 \times (1/10)^5 = 0.4 \times 10^{-5}$

"162636" parsed as F, contribute $.95^6 \times (1/6)^6 = 1.6 \times 10^{-5}$
parsed as L, contribute $.95^6 \times (1/2)^3 \times (1/10)^3 = 9.0 \times 10^{-5}$

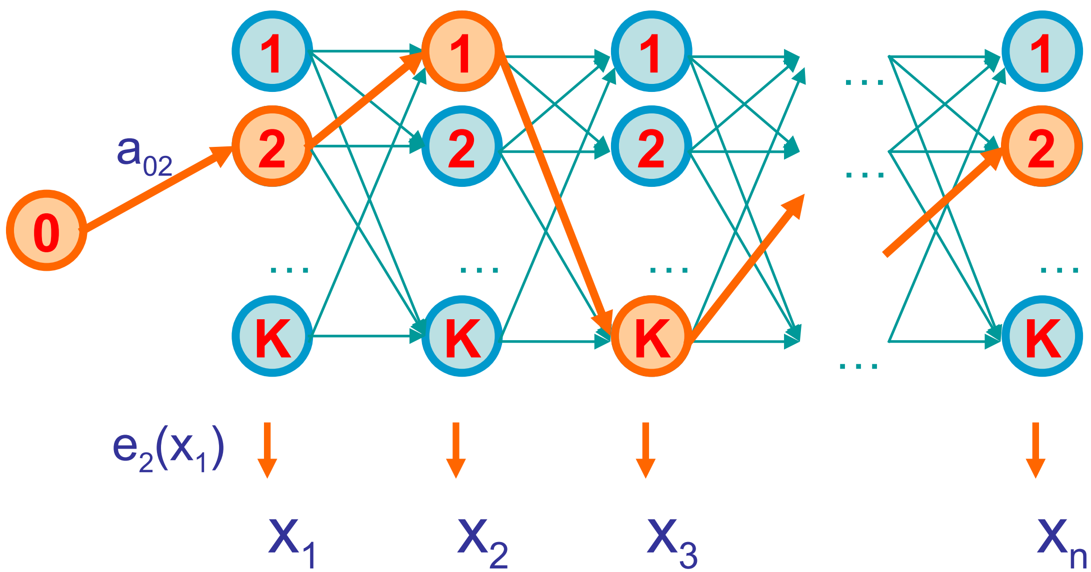
Problem 2: Evaluation

Compute the likelihood that a sequence is generated by the model

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



A couple of questions

Given a sequence x,

- What is the probability that x
- Given a position i, what is th

Example: the dishonest cas

P(box: FFFFFFFFFFF) =
$(1/6)^{11} * 0.95^{12}$ =
$2.76^{-9} * 0.54$ =
1.49^{-9}

P(box: LLLLLLLLLLL) =
$[(1/2)^6 * (1/10)^5] * 0.95^{10} * 0.05^2$ =
$1.56*10^{-7} * 1.5^{-3}$ =
0.23^{-9}

Say x = $\underbrace{12341\ldots231}_{F}\boxed{62616364616}\underbrace{234112\ldots21341}_{F}$

Most likely path: π = FF……F
(too "unlikely" to transition F $\rightarrow$ L $\rightarrow$ F)
However: marked letters more likely to be L than unmarked letters

Evaluation

We will develop algorithms that allow us to compute:

$P(x)$ Probability of x given the model

$P(x_i \ldots x_j)$ Probability of a substring of x given the model

$P(\pi_i = k \mid x)$ "Posterior" probability that the i^{th} state is k, given x

A more refined measure of <u>which states</u> x may be in

The Forward Algorithm

We want to calculate

P(x) = probability of x, given the HMM

Sum over all possible ways of generating x:

$$P(x) = \Sigma_\pi P(x, \pi) = \Sigma_\pi P(x \mid \pi) P(\pi)$$

To avoid summing over an exponential number of paths π, define

$f_k(i) = P(x_1 \ldots x_i, \pi_i = k)$ (the forward probability)

"generate i first observations and end up in state k"

The Forward Algorithm – derivation

Define the forward probability:

$$f_k(i) = P(x_1 \ldots x_i,\ \pi_i = k)$$

$$= \sum_{\pi 1 \ldots \pi i\text{-}1} P(x_1 \ldots x_{i-1},\ \pi_1, \ldots,\ \pi_{i-1},\ \pi_i = k)\ e_k(x_i)$$

$$= \sum_l \boldsymbol{\sum_{\pi 1 \ldots \pi i\text{-}2} P(x_1 \ldots x_{i-1},\ \pi_1, \ldots,\ \pi_{i-2},\ \pi_{i-1} = l)}\ a_{lk}\ e_k(x_i)$$

$$= \sum_l \boldsymbol{P(x_1 \ldots x_{i-1},\ \pi_{i-1} = l)}\ a_{lk}\ e_k(x_i)$$

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

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

Initialization:

$V_0(0) = 1$

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

Iteration:

$V_j(i) = e_j(x_i)\ \mathbf{max}_k\ V_k(i-1)\, a_{kj}$

Termination:

$P(x, \pi^*) = \mathbf{max}_k\ V_k(N)$

FORWARD

Initialization:

$f_0(0) = 1$

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

Iteration:

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

Termination:

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

Motivation for the Backward Algorithm

We want to compute

$$P(\pi_i = k \mid x),$$

the probability distribution on the i^{th} position, given x

We start by computing

$$
\begin{aligned}
P(\pi_i = k, x) &= P(x_1 \ldots x_i, \pi_i = k, x_{i+1} \ldots x_N) \\
&= P(x_1 \ldots x_i, \pi_i = k)\, P(x_{i+1} \ldots x_N \mid x_1 \ldots x_i, \pi_i = k) \\
&= \boxed{P(x_1 \ldots x_i, \pi_i = k)}\, \boxed{P(x_{i+1} \ldots x_N \mid \pi_i = k)}
\end{aligned}
$$

Forward, $f_k(i)$ **Backward, $b_k(i)$**

Then, $P(\pi_i = k \mid x) = P(\pi_i = k, x) / P(x)$

The Backward Algorithm – derivation

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"*

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

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

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

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

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:

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

Termination:

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

Computational Complexity

What is the running time, and space required, for Forward and Backward?

Time: $O(K^2N)$
Space: $O(KN)$

Useful implementation technique to avoid underflows

Viterbi: sum of logs

Forward/Backward: rescaling at each few positions by multiplying by a constant

Posterior Decoding

We can now calculate

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

$P(\pi_i = k \mid x) =$

$P(\pi_i = k , x)/P(x) =$

$P(x_1, \ldots, x_i, \pi_i = k, x_{i+1}, \ldots x_n) / P(x) =$

$P(x_1, \ldots, x_i, \pi_i = k)\, P(x_{i+1}, \ldots x_n \mid \pi_i = k) / P(x) =$

$f_k(i)\, b_k(i) / P(x)$

Then, we can ask

What is the most likely state at position i of sequence x:

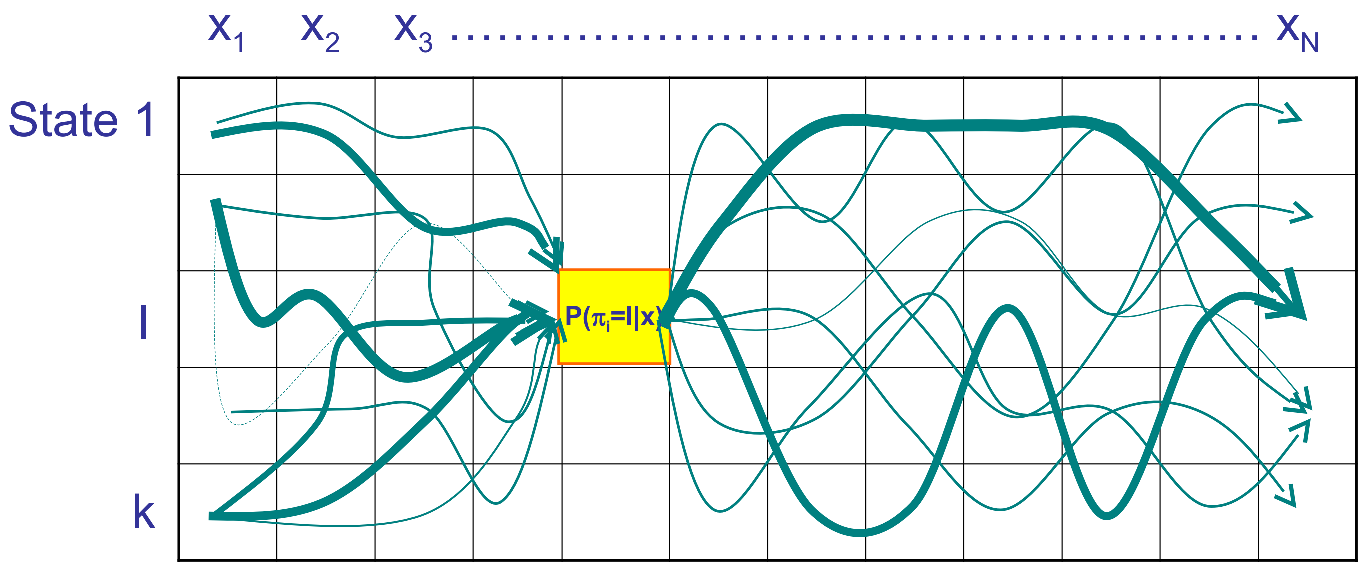
Define $\pi^\wedge$ by Posterior Decoding:

$$\pi^\wedge_i = \text{argmax}_k\, P(\pi_i = k \mid x)$$

Posterior Decoding

- For each state,
  - Posterior Decoding gives us a curve of likelihood of state for each position
  - That is sometimes more informative than Viterbi path π^*
- Posterior Decoding may give an invalid sequence of states (of probability 0)
  - Why?

Posterior Decoding



- $P(\pi_i = k \mid x) \;= \sum_\pi P(\pi \mid x)\ \mathbf{1}(\pi_i = k)$

  $= \sum_{\{\pi:\pi[i] = k\}} P(\pi \mid x)$

$\mathbf{1}(\psi) = 1$, if ψ is true
0, otherwise

Viterbi, Forward, Backward

VITERBI

Initialization:

$V_0(0) = 1$

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

Iteration:

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

Termination:

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

FORWARD

Initialization:

$f_0(0) = 1$

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

Iteration:

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

Termination:

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

BACKWARD

Initialization:

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

Iteration:

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

Termination:

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

Problem 3: Learning

Find the parameters that maximize the likelihood of the observed sequence

Estimating HMM parameters

- Easy if we know the sequence of hidden states
  - Count # times each transition occurs
  - Count #times each observation occurs in each state
- Given an HMM and observed sequence,
  we can compute the distribution over paths,
  and therefore the expected counts
- “Chicken and egg” problem

Solution: Use the EM algorithm

- Guess initial HMM parameters
- **E step:** Compute distribution over paths
- **M step:** Compute max likelihood parameters
- But how do we do this efficiently?

The forward-backward algorithm

- Also known as the Baum-Welch algorithm
- Compute probability of each state at each position using forward and backward probabilities
$\rightarrow$ (Expected) observation counts
- Compute probability of each pair of states at each pair of consecutive positions *i* and *i+1* using *forward(i)* and *backward(i+1)*
$\rightarrow$ (Expected) transition counts

$$\text{Count}(k \rightarrow l) = \Sigma_i \, f_k(i) \, a_{kl} \, b_l(i+1) \, / \, P(x)$$