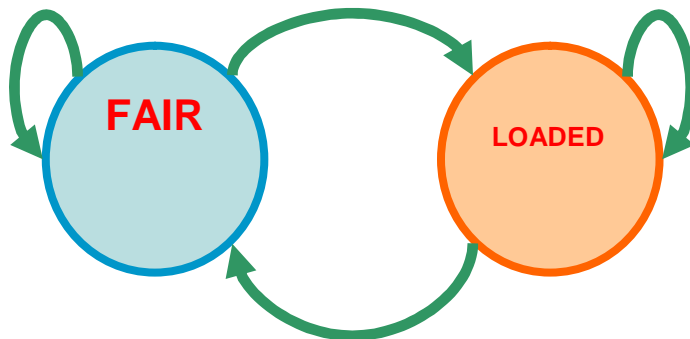


CS 262 Computational Genomics  
Lecture 5 Notes  
Professor: Serafim Batzoglou

## 1 Dishonest Casino Review

The casino has a two pairs of die, one pair is loaded, and the other pair is fair. The casino can switch out the pair of die without the player knowing. Want to determine if for a given round of play if that round is more likely to be using the fair or loaded die. For which part of sequences of die roll is it rolled by the fair die and which part by the loaded die.

Finally, if we have a lot of examples of where the casino players roll a specific die, we can learn the parameters of the models and what are the underlying probabilities.



## 2 Markov Review

A Hidden Markov Model consists of a constant state at each time step as defined by

An alphabet is a set of output by the model  $\Sigma = \{ b_1, b_2, \dots, b_M \}$

A Set of States  $Q = \{ 1, \dots, K \}$

Transition Probabilities: The probability of transitioning between states

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

Start Probabilities:  $a_{0i}$

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

Emission Probabilities

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

### 2.1 Markov Property

The Markov Property says that anything that has happened so far can be summarized by the current state we are in. Thus the probability that the next state  $\pi_{t+1} = k$  based on what we have seen can be defined as simply the conditionally probability based on the current state:

$$P(\pi_{t+1} = k \mid \pi_t).$$

$$\begin{aligned}
 P(\pi_{t+1} = k \mid \text{“whatever happened so far”}) &= \\
 P(\pi_{t+1} = k \mid \pi_1, \pi_2, \dots, \pi_t, x_1, x_2, \dots, x_t) &= \\
 P(\pi_{t+1} = k \mid \pi_t)
 \end{aligned}$$

## 2.2 Decoding Markov Chains

Given a particular sequence, we want to figure out what happened in that sequence.

Given a sequence

$$\mathbf{x} = x_1 \dots x_N,$$

A **parse** of  $x$  is a sequence of states

$$\boldsymbol{\pi} = \pi_1, \dots, \pi_N$$

Parsing, also called decoding, is defining a particular sequence of underlying states.

An example of a parse is for a given sequence: 34261455666464126643146141 is to represent it as

34261455	6664641266	4314614
----------	------------	---------

FAIR            LOADED            FAIR

However, there are many parses. Because there are two possible states for each number, there are  $2^n$  possible parses or in general with  $k$  states there are  $k^n$  possible parses.

## 2.3 The likelihood of a Parse

The probability of a Parse is the probability of a given sequence and a given parse of that sequence or  $P(x, \pi)$ . It does not matter how unlikely sequence  $x$  is, but only to figure out what is the parse now that we have sequence  $x$ .

To determine a likelihood of a parse is to simply multiply the arrows of the state transitions that led to that part for the given sequence.

Given a sequence  $\mathbf{x} = x_1 \dots x_N$  and a parse  $\boldsymbol{\pi} = \pi_1, \dots, \pi_N$ ,

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

$$P(x, \pi) = P(x_1, \dots, x_N, \pi_1, \dots, \pi_N) = a_{0\pi_1} a_{\pi_1\pi_2} \dots a_{\pi_{N-1}\pi_N} e_{\pi_1}(x_1) \dots e_{\pi_N}(x_N)$$

A compact way to write  $a_{0\pi_1} a_{\pi_1\pi_2} \dots a_{\pi_{N-1}\pi_N} e_{\pi_1}(x_1) \dots e_{\pi_N}(x_N)$

Number all parameters  $a_{ij}$  and  $e_i(b)$ ;  $n$  params

**Example:**

$$a_{0\text{Fair}} : \theta_1; a_{0\text{Loaded}} : \theta_2; \dots e_{\text{Loaded}}(6) = \theta_{18}$$

Here  $\theta_1 = 1 - \theta_2$

In this particular model we have 18 parameters. However, not all of them are free: 1 start parameter is not free, 2 transition parameters, and 2 emission parameters are not free. Thus there are 13 free parameters.

Then, count in for a particular  $x$  and  $\pi$  the # of times each parameter  $j = 1, \dots, n$  occurs.

Each time there is an arrow from state  $i$  to state  $j$  then  $a_{ij}$  is multiplied

$F(j, x, \pi) = \# \text{ parameter } \theta_j \text{ occurs in } (x, \pi)$

(call  $F(.,.,.)$  the *feature counts*)

Then,  $P(x, \pi) = \prod_{j=1..n} \theta_j^{F(j, x, \pi)} = \exp[\sum_{j=1..n} \log(\theta_j) \times F(j, x, \pi)]$

## 2.4 Example: the Dishonest Casino

### 2.4.1 The Probability that they are all Fair die rolls

Let the sequence of rolls be:

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

Then, what is the likelihood of

$\pi = \mathbf{F, F, \dots, F}$ ?

(say initial probs  $a_{0\text{Fair}} = 1/2$ ,  $a_{0\text{Loaded}} = 1/2$ )

$1/2 \times P(1 | \text{Fair}) P(\text{Fair} | \text{Fair}) P(2 | \text{Fair}) P(\text{Fair} | \text{Fair}) \dots P(4 | \text{Fair}) =$

$1/2 \times (1/6)^{10} \times (0.95)^9 = .00000000521158647211 \approx 0.5 \times 10^{-9}$

The  $1/2$  comes from the initial probability. Each value has an emission probability of  $(1/6)$ . There are 9 transitions from Fair to Fair each with a probability of .95.

### 2.4.2 The Probability that they are all Loaded die rolls

Let the sequence of rolls be:

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

Then, what is the likelihood of

$\pi = \mathbf{L, L, \dots, L}$ ?

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

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

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

The  $\frac{1}{2}$  comes from the initial probability. Non-6 values have an emission probability of  $(1/10)$  and there are 9/10 non-6 values. There is one value of 6 with a probability of  $\frac{1}{2}^1$ . There are 9 transitions from loaded to loaded with a probability of .95 per transition.

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

### 2.4.3 Sequence of rolls have many 6's.

Let the sequence of rolls be:

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

Now, what is the likelihood  $\pi = F, F, \dots, F$ ?

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

What is the likelihood

$$\pi = L, L, \dots, L?$$

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

There are 4 non-6 values with a emission probability of  $1/10$  and 6 values of 6 each with an emission probability of  $(1/2)$  thus  $(1/2)^6$ . There are 9 state transitions of loaded die with a probability of .95 per transition.

So, it is 100 times more likely the die is loaded. However, this is not the only possible parse because some could be fair and some could be loaded.

## 3 Main questions of HMM's

1. Evaluation  
GIVEN a HMM  $M$ , and a sequence  $x$ ,  
FIND  $\text{Prob}[x | M]$

Or, how likely is the sequence to appear given the model.

2. Decoding  
GIVEN a HMM  $M$ , and a sequence  $x$ ,

FIND the sequence  $\pi$  of states that maximizes  $P[ x, \pi | M ]$

Classically, find the maximum likelihood decoding or find the underlying sequence of states that maximizes the sequence given the sequence of states  $\pi$ .

### 3. Learning

GIVEN a HMM  $M$ , with unspecified transition/emission probs.,  
and a sequence  $x$ ,

FIND parameters  $\theta = (e_i(\cdot), a_{ij})$  that maximize  $P[ x | \theta ]$

If we do not know the underlying states, we can use tricks called Expectation Maximization that can give a reasonable answer, although not an optimal answer.

## 4 Notation

The model  $M$  is: architecture (#states, etc), parameters  $\theta = a_{ij}, e_i(\cdot)$

$P[x | M]$  is the same with  $P[ x | \theta ]$ , and  $P[ x ]$ ,  
when the architecture, and the parameters, respectively, are implied

$P[ x, \pi | M ]$ ,  $P[ x, \pi | \theta ]$  and  $P[ x, \pi ]$  are the same  
when the architecture, and the parameters, are implied

**LEARNING:** we write  $P[ x | \theta ]$   
to emphasize that we are seeking the  $\theta^*$  that maximizes  $P[ x | \theta ]$

## 5 Problem 1: Decoding

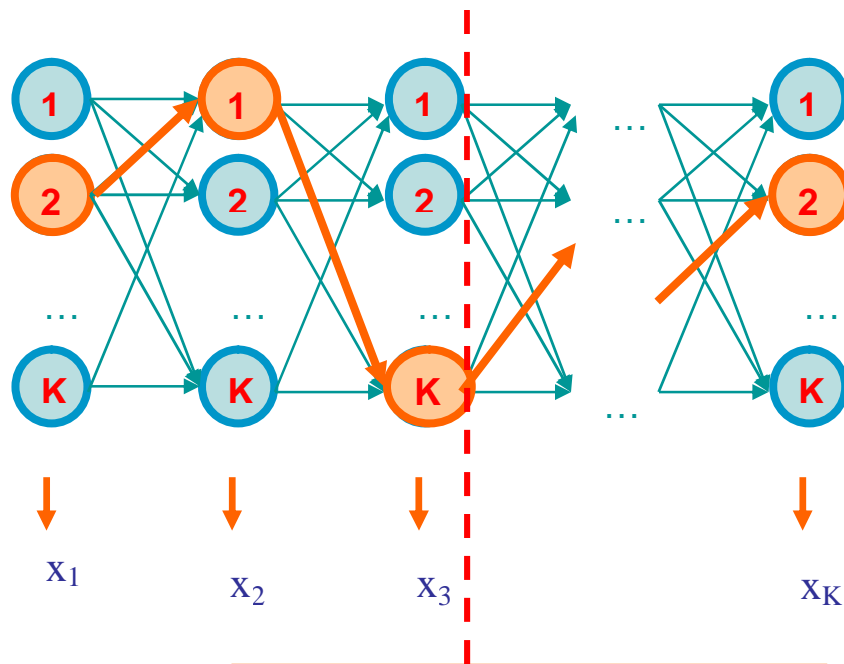
Given a sequence, find the most likely parse of a sequence.

In genomic models, we want to find the most likely underlying sequence of states that describe a gene structure such as exons, introns, and inter-genetic sequence.

GIVEN  $x = x_1 x_2 \dots x_N$

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

$\pi^* = \operatorname{argmax}_{\pi} P[ x, \pi ]$   
Maximizes  $a_{0\pi_1} e_{\pi_1}(x_1) a_{\pi_1\pi_2} \dots a_{\pi_{N-1}\pi_N} e_{\pi_N}(x_N)$



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

If we are given that at time point i, and in some state k the maximum parse sequence will pass through that point. Then we must only maximize the product of everything leading into state k and leaving state k. We can do this using a dynamic programming matrix.

The Maximum of all products of arrows to the left of the line at state k is defined as \$V\_k(i)\$.

$$V_k(i) = \max_{\{\pi_1 \dots \pi_{i-1}\}} P[X_1 \dots X_{i-1}, \pi_1, \dots, \pi_{i-1}, X_i, \pi_i = k]$$

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

**Inductive assumption:** Given that for all states k, and for a fixed position i,

$$V_k(i) = \max_{\{\pi_1 \dots \pi_{i-1}\}} P[X_1 \dots X_{i-1}, \pi_1, \dots, \pi_{i-1}, X_i, \pi_i = k]$$

What is \$V\_i(i+1)\$?

State l is the state following \$\pi\_i\$, or \$\pi\_{i+1}\$.

From definition,

$$\begin{aligned} V_l(i+1) &= \max_{\{\pi_1 \dots \pi_i\}} P[X_1 \dots X_i, \pi_1, \dots, \pi_i, X_{i+1}, \pi_{i+1} = l] \\ &= \max_{\{\pi_1 \dots \pi_i\}} P(X_{i+1}, \pi_{i+1} = l \mid X_1 \dots X_i, \pi_1, \dots, \pi_i) P[X_1 \dots X_i, \pi_1, \dots, \pi_i] \\ &= \max_{\{\pi_1 \dots \pi_i\}} P(X_{i+1}, \pi_{i+1} = l \mid \pi_i) P[X_1 \dots X_i, \pi_1, \dots, \pi_i] \\ &= \max_k [P(X_{i+1}, \pi_{i+1} = l \mid \pi_i = k) \max_{\{\pi_1 \dots \pi_{i-1}\}} P[X_1 \dots X_{i-1}, \pi_1, \dots, \pi_{i-1}, X_i, \pi_i = k]] \\ &= \max_k [P(X_{i+1} \mid \pi_{i+1} = l) P(\pi_{i+1} = l \mid \pi_i = k) V_k(i)] \end{aligned}$$

$$= e_j(x_{i+1}) \max_k a_{kl} V_k(i)$$

(This last step is defined from the Markov Assumption)

## 5.1 Viterbi Algorithm

Input:  $x = x_1, \dots, x_N$

### Initialization:

$$V_0(0) = 1 \quad (\text{Have not visited a state, 0 is the imaginary first position})$$

$$V_k(0) = 0, \text{ for all } k > 0$$

### Iteration:

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

$$\text{Ptr}_j(i) = \text{argmax}_k a_{kj} V_k(i-1) \quad (\text{Back Pointer to reconstruct optimal path})$$

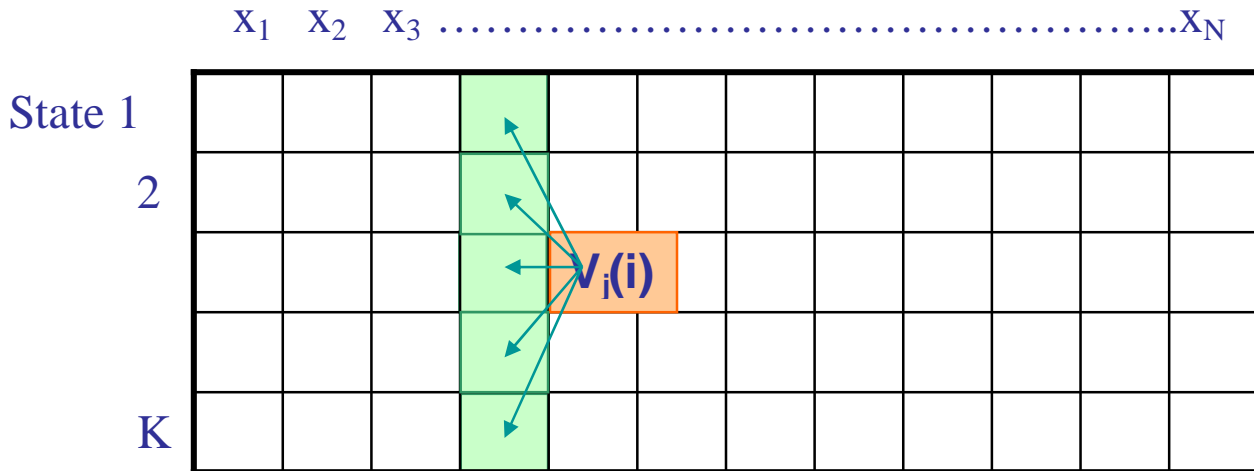
### Termination:

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

### Traceback:

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

$$\pi_{i-1}^* = \text{Ptr}_{\pi_i}(i) \quad (\text{Back Pointers})$$



At every cell at this matrix, perform computation of  $V_j(i)$  is the maximum of the value in the green cell \* the transition arrow value from the green cell to the pink cell times the initial probability of  $X_i$  from state  $j$ .

Similar to “aligning” a set of states to a sequence

### Time:

$O(K^2 * N)$  Have  $K * N$  cells because for each cell have to do  $K$  calculations.

### Space:

$O(KN)$  May be able to cut the space even further.

Can we sweep the matrix horizontally or only do it vertically??

As we sweep the matrix, every cell depends on a number of other cells and so can only sweep in a way that follows the dependencies. The previous column is necessary to calculate a cell in the current column.

### 5.1.1 Underflows and Logs

Underflows are a significant problem

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

These numbers become extremely small probabilities and underflows the computer stored value.

**Solution:** Take the logs of all values

$$V_i(i) = \log e_k(x_i) + \max_k [ V_k(i-1) + \log a_{ki} ]$$

Should initial  $V_0(0)$  to 0 instead of 1.

### 5.1.2 Viterbi Example

Let x be a long sequence with a portion of ~ 1/6 6's, followed by a portion of ~ 1/2 6's...

x = 123456123456...12345 6626364656...1626364656

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

FFF.....FLLL.....L

The optimal parse is a set of Fairs in the beginning and a set of Loads at the end and the optimal cutting position will be at a non-6 followed by a 6. You would prefer to go one step forward and take a non-6 with a Fair state than a 6. And similarly, you would prefer to go one step back and take a 6 with a loaded die rather than a Fair state die because of the transition probabilities.

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  $.95^6 \times (1/2)^3 \times (1/10)^3 = 9.0 \times 10^{-5}$

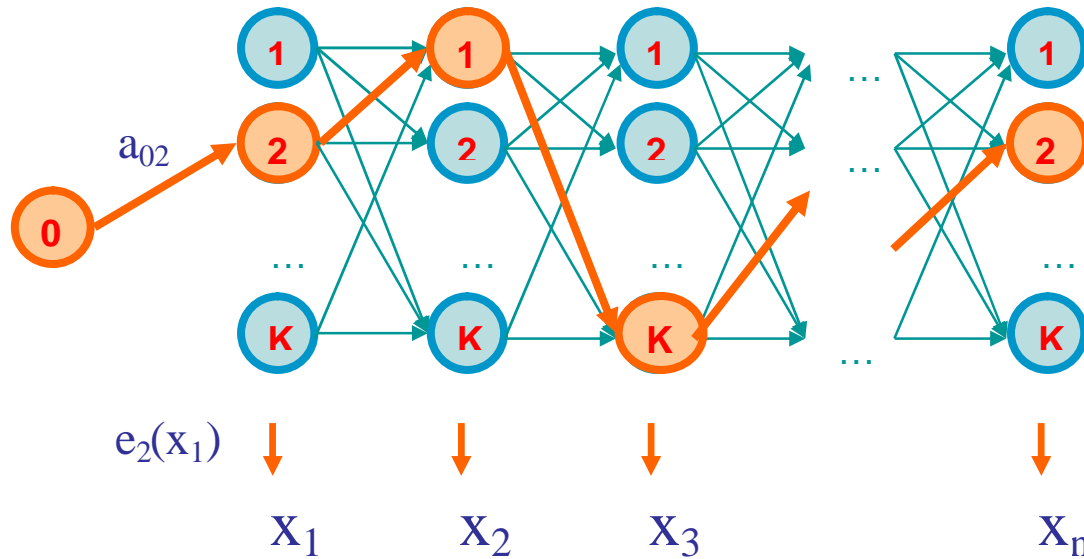
## 6 Problem 2: Evaluation

Find the likelihood a sequence is generated by the model

HMM's are generative by starting a state and using the transitions to move to any other state and stop at any random state that you end up in. You can calculate how likely you were to generate this sequence by multiplying all of the arrows from the start state.

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

- Start at state  $\pi_1$  according to prob  $a_{0\pi_1}$
- Emit letter  $x_1$  according to prob  $e_{\pi_1}(x_1)$
- Go to state  $\pi_2$  according to prob  $a_{\pi_1 \pi_2}$
- ... until emitting  $x_n$



THUS,

Given a sequence  $x$ ,

- What is the probability that  $x$  was generated by the model?  
How likely is 3366?  
Sum over all possibilities of sequences. All Fair, All Loaded, Fair Loaded Combo.
- Given a position  $i$ , what is the most likely state that emitted  $x_i$ ?  
What is the more likely at this position, is it loaded or fair?

Example: the dishonest casino

Say  $x = 12341\dots23162616364616234112\dots21341$   
Sequence of winning 6's

What is the optimal parse of this sequence?

Possible: Fair until first 6, then loaded until last 6, then fair again.

BOX = 62616364616

$P(\text{box: FFFFFFFFFF}) =$

$$(1/6)^{11} * 0.95^{12} =$$

$$2.76^{-9} * 0.54 =$$

$$1.49^{-9}$$

$P(\text{box: LLLLLLLLLL}) =$

$$\begin{aligned} & [ (1/2)^6 * (1/10)^5 ] * 0.95^{10} * 0.05^2 = \\ & 1.56 * 10^{-7} * 1.5^{-3} = \\ & 0.23^{-9} \end{aligned}$$

For this example, the optimal parse is fair all the way through because the probability is higher than if they are all loaded.

Most likely path:  $\pi = FF\dots F$

However: marked letters more likely to be L than unmarked letters

We want to be able to quantify all possibilities of the sequences. To accomplish this, we will develop algorithms that allow us to compute:

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

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

$P(\pi_i = k \mid x)$  Probability that the  $i^{\text{th}}$  state is  $k$ , given  $x$   
 A more refined measure of which states  $x$  may be in

## 6.1 The Forward Algorithm

We want to calculate

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

Sum over all possible ways of generating  $x$ :

$$P(x) = \sum_{\pi} P(x, \pi) = \sum_{\pi} P(x \mid \pi) P(\pi)$$

To avoid summing over an exponential number of paths  $\pi$ , define

$$f_k(i) = P(x_1 \dots x_i, \pi_i = k) \quad (\text{the forward probability})$$

Define the forward probability:

$$\begin{aligned} f_k(i) &= P(x_1 \dots x_i, \pi_i = k) \\ &= \sum_{\pi_1 \dots \pi_{i-1}} P(x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-1}, \pi_i = k) e_k(x_i) \\ &= \sum_1 \sum_{\pi_1 \dots \pi_{i-2}} P(x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-2}, \pi_{i-1} = 1) a_{1k} e_k(x_i) \\ &= \sum_1 P(x_1 \dots x_{i-1}, \pi_{i-1} = 1) a_{1k} e_k(x_i) \\ &= e_k(x_i) \sum_1 f_1(i-1) a_{1k} \end{aligned}$$

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

### Initialization:

$$\begin{aligned} f_0(0) &= 1 \\ f_k(0) &= 0, \text{ for all } k > 0 \end{aligned}$$

### Iteration:

$$f_k(i) = e_k(x_i) \sum_1 f_1(i-1) a_{1k}$$

**Termination:**

$$P(x) = \sum_k f_k(N) a_{k0} \quad (\text{MAIN POINT})$$

Where,  $a_{k0}$  is the probability that the terminating state is  $k$  (usually =  $a_{0k}$ )

$a_{k0}$  is termination probability and being difficult to use the Professor suggests defining:

**Termination:**

$$P(x) = \sum_k f_k(N) \quad (\text{MAIN POINT})$$

## 6.2 Viterbi versus Forward

<b>VITERBI</b>	<b>FORWARD</b>
<p><b>Initialization:</b></p> $V_0(0) = 1$ $V_k(0) = 0, \text{ for all } k > 0$	<p><b>Initialization:</b></p> $f_0(0) = 1$ $f_k(0) = 0, \text{ for all } k > 0$
<p><b>Iteration:</b></p> $V_i(i) = e_i(x_i) \max_k V_k(i-1) a_{ki}$	<p><b>Iteration:</b></p> $f_i(i) = e_i(x_i) \sum_k f_k(i-1) a_{ki}$
<p><b>Termination:</b></p> $P(x, \pi^*) = \max_k V_k(N)$	<p><b>Termination:</b></p> $P(x) = \sum_k f_k(N) a_{k0}$

## 6.3 The Backward Algorithm

Want to know the probability that a particular roll was given by a loaded die. Knowing this is more descriptive than simply the optimal parse. The optimal parse has the maximum likelihood and is consistent, but different parts of the sequence affect one another that is reflected in the optimal parse and in individual positions and states.

We want to compute

$$P(\pi_i = k | x), \text{ the probability distribution on the } i^{\text{th}} \text{ position, given } x$$

We start by computing

$$\begin{aligned}
 P(\pi_i = k, x) &= P(x_1 \dots x_i, \pi_i = k, x_{i+1} \dots x_N) \\
 &= P(x_1 \dots x_i, \pi_i = k) P(x_{i+1} \dots x_N | x_1 \dots x_i, \pi_i = k) \\
 &= P(x_1 \dots x_i, \pi_i = k) P(x_{i+1} \dots x_N | \pi_i = k) \quad (\text{B/c of the Markov Property}) \\
 &\quad \text{FORWARD} \qquad \qquad \text{BACKWARD}
 \end{aligned}$$

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

We would like to calculate that the probability of a given state is  $k$  given the entire sequence. We can calculate  $P(x)$ , so if we can calculate  $P(\pi_i = k, x)$  then we can get  $(\pi_i = k | x)$ . We defined  $P(\pi_i = k, x)$  above. We already have FORWARD, only need BACKWARD.

Define the backward probability:

$$\begin{aligned}
 b_k(i) &= P(x_{i+1} \dots x_N \mid \pi_i = k) && \text{(Start from state } k \text{ and move forward to } x_N) \\
 &= \sum_{\pi_{i+1} \dots \pi_N} P(x_{i+1}, x_{i+2}, \dots, x_N, \pi_{i+1}, \dots, \pi_N \mid \pi_i = k) \\
 &= \sum_l \sum_{\pi_{i+1} \dots \pi_N} P(x_{i+1}, x_{i+2}, \dots, x_N, \pi_{i+1} = l, \pi_{i+2}, \dots, \pi_N \mid \pi_i = k) \\
 &= \sum_l e_l(x_{i+1}) a_{kl} \sum_{\pi_{i+1} \dots \pi_N} P(x_{i+2}, \dots, x_N, \pi_{i+2}, \dots, \pi_N \mid \pi_{i+1} = l) \quad \text{(Markov Property)} \\
 &= \sum_l e_l(x_{i+1}) a_{kl} \mathbf{b}_l(i+1)
 \end{aligned}$$

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

**Initialization:**

$$b_k(N) = a_{k0}, \text{ for all } k$$

**Iteration:**

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

**Termination:**

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

## 6.4 Computational Complexity of Forward and Backwards

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

$$\text{Time: } O(K^2N)$$

$$\text{Space: } O(KN)$$

About twice as slow as Viterbi because one must run two algorithms in one.

Useful implementation technique to avoid underflows

**Viterbi:** sum of logs

What is the larger quantity? Forward  $k$ 's of  $x_i$  or Viterbi  $k$ 's of  $x_i$ ?

Forward is much larger.

Thus,

**Forward/Backward:** rescaling at each position by multiplying by a constant

Cannot use logs b/c it is difficult to do logs over a summation.

## 6.5 Posterior Decoding

We can now calculate

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

Then, we can ask

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

Define  $\hat{\pi}_i$  by Posterior Decoding:

$$\hat{\pi}_i = \operatorname{argmax}_k P(\pi_i = k \mid x)$$

Why is  $\hat{\pi}_i$  not the same as  $\pi_i^*$ ?

In general, at a given particular point, you may arrive with many underlying states to a given state, none of which is

For each state,

Posterior Decoding gives us a curve of likelihood of state for each position

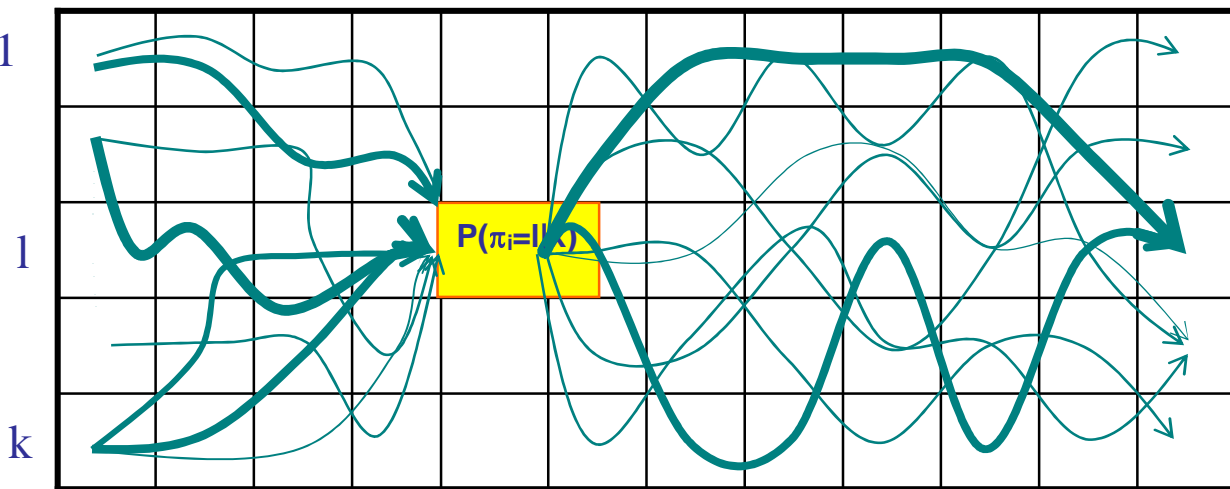
That is sometimes more informative than Viterbi path  $\pi^*$

Posterior Decoding may give an invalid sequence of states

If a transition probability between two states is 0 then

$x_1 \quad x_2 \quad x_3 \dots \dots \dots x_N$

State 1



Each cell gives a probability of reaching this state given all possible paths.

$$P(\pi_i = k | x) = \sum_{\pi} P(\pi | x) I(\pi_i = k)$$

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

**Indicator Variable:**  
 $1(\psi) = 1$ , if  $\psi$  is true  
 $0$ , otherwise

How do we compute  $P(\pi_i = 1, \pi_j = 1' | x)$ ?

$$P(\pi_i = 1, \pi_j = 1' | x) = \frac{f_i(i) b_{1'}(j)}{P(x)}$$

## 7 Summary Comparison

VITERBI	FORWARD	BACKWARD
<p><b>Initialization:</b>  <math>V_0(0) = 1</math>  <math>V_k(0) = 0</math>, for all <math>k &gt; 0</math></p>	<p><b>Initialization:</b>  <math>f_0(0) = 1</math>  <math>f_k(0) = 0</math>, for all <math>k &gt; 0</math></p>	<p><b>Initialization:</b>  <math>b_k(N) = a_{k0}</math>, for all <math>k</math></p>
<p><b>Iteration:</b></p>	<p><b>Iteration:</b></p>	<p><b>Iteration:</b></p>

$V_1(i) = e_i(x_i) \max_k V_k(i-1) a_{ki}$ <b>Termination:</b> $P(x, \pi^*) = \max_k V_k(N)$	$f_1(i) = e_i(x_i) \sum_k f_k(i-1) a_{ki}$ <b>Termination:</b> $P(x) = \sum_k f_k(N) a_{k0}$	$b_1(i) = \sum_k e_i(x_{i+1}) a_{ki} b_k(i+1)$ <b>Termination:</b> $P(x) = \sum_k a_{0k} e_k(x_1) b_k(1)$
--	--	---