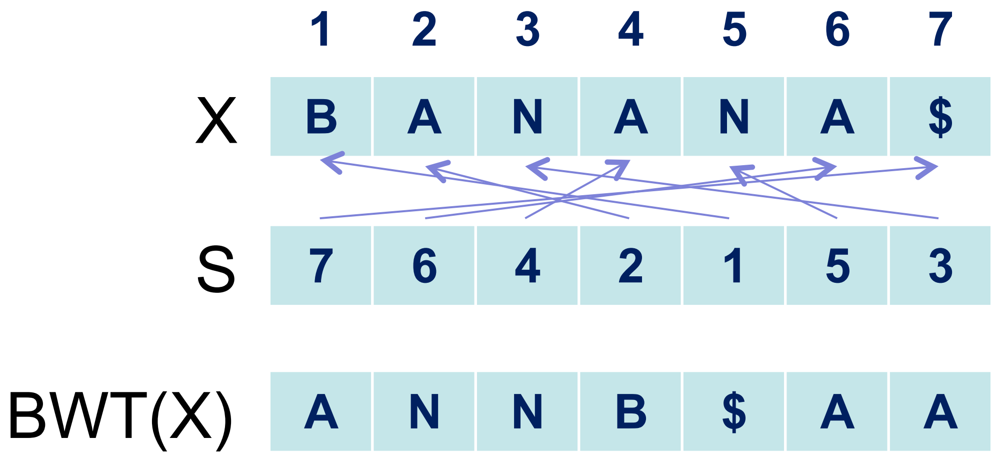
Review: Suffix Arrays and BWT

$BANANA	1	$BANANA
A$BANAN	2	A$BANAN
ANA$BAN	3	ANA$BAN
ANANA$B	4	ANANA$B
BANANA$	5	BANANA$
NA$BANA	6	NA$BANA
NANA$BA	7	NANA$BA

Suffixes are sorted in the BWT matrix

Define suffix array S:

S(i) = j, where $X_j \ldots X_n$ is the i-th suffix lexicographically



	1	2	3	4	5	6	7
X	B	A	N	A	N	A	$
S	7	6	4	2	1	5	3
BWT(X)	A	N	N	B	$	A	A

BWT(X) constructed from S:
At each position, take the letter to the left of the one pointed by S

Review: Reconstructing BANANA

```
$BANANA
A$BANAN
ANA$BAN
ANANA$B
BANANA$
NA$BANA
NANA$BA
```

BWT matrix of string 'BANANA'

	A	N	N	B	$	A	A
C()	1	5	5	4	0	1	1
index i	1	1	2	1	1	2	3
LF()	2	6	7	5	1	3	4

C(a) character array: # letter occs before a

i: indicating i-th occurr. of 'a' in BWT

LF() = C() + i

Reconstruct BANANA:

```
S := ""; r := 1; c := BWT[r];
UNTIL c = '$' {
        S := cS;
        r := LF(r);
        c := BWT(r); }
```

Credit: Ben Langmead thesis

Searching for query "ANA"

```
$BANANA
A$BANAN
ANA$BAN
ANANA$B
BANANA$
NA$BANA
NANA$BA
```

BWT matrix of
string 'BANANA'

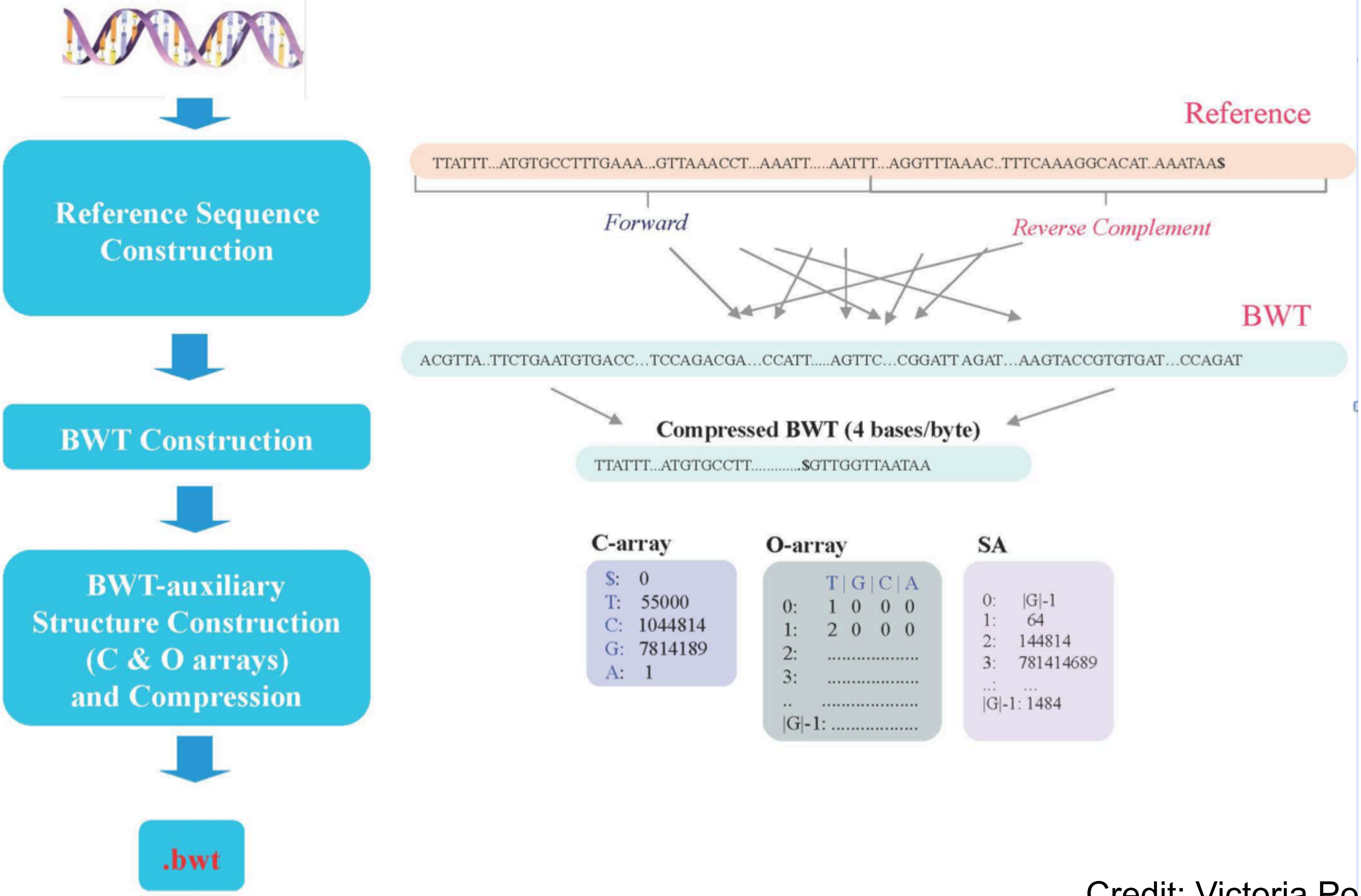
```
Let
LFC(r, a) = C(a) + i, where i = #'a's up to r in BWT

ExactMatch(W[1…k]) {

    a := W[k];
    low := C(a) +1;
    high := C(a+1);    // a+1: lexicographically next char
    i := k – 1;
    while (low <= high && i >= 1) {
            a = W[i];
            low = LFC(low – 1, a) + 1;
            high = LFC(high, a);
            i := i – 1; }
    return (low, high);
}
```

Credit: Ben Langmead thesis

BWT Index Construction



Reference Sequence Construction

BWT Construction

BWT-auxiliary Structure Construction (C & O arrays) and Compression

.bwt

Reference

TTATTT...ATGTGCCTTTGAAA...GTTAAACCT...AAATT.....AATTT...AGGTTTAAAC..TTTCAAAGGCACAT..AAATAA$

Forward

Reverse Complement

BWT

ACGTTA..TTCTGAATGTGACC…TCCAGACGA…CCATT.....AGTTC…CGGATT AGAT…AAGTACCGTGTGAT…CCAGAT

Compressed BWT (4 bases/byte)

TTATTT...ATGTGCCTT............$GTTGGTTAATAA

C-array

$:	0
T:	55000
C:	1044814
G:	7814189
A:	1

O-array

	T	G	C	A
0:	1	0	0	0
1:	2	0	0	0
2:	....			
3:	....			
..	....			
|G|-1:	....			

SA

0:	|G|-1
1:	64
2:	144814
3:	781414689
..:	...
|G|-1:	1484

Credit: Victoria Popic

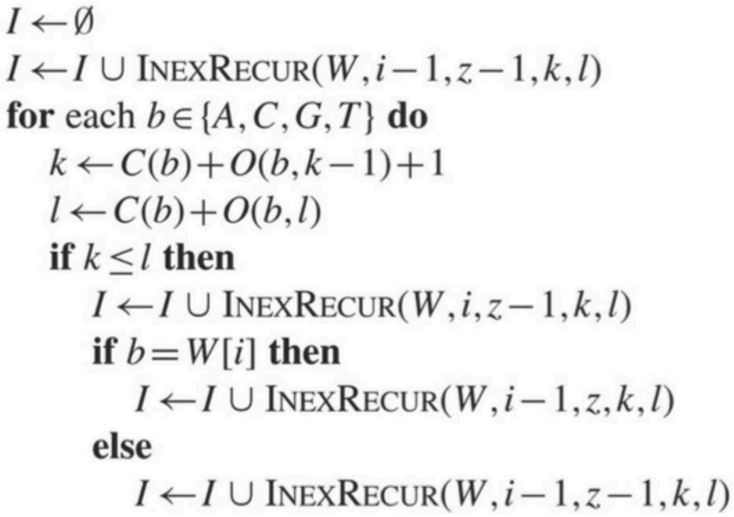
BWA Inexact Match

Allow up to **n** mismatches/gaps

Backward search:
Given read W, keep track of multiple partial alignments

Partial alignment: (i, z, L, U)

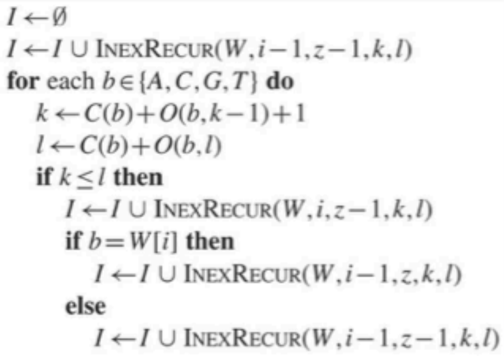
i: current position
z: remaining non-matches allowed
L: current low
U: current high



```
$I \leftarrow \emptyset$
$I \leftarrow I \cup \text{InexRecur}(W, i-1, z-1, k, l)$
for each $b \in \{A, C, G, T\}$ do
    $k \leftarrow C(b) + O(b, k-1) + 1$
    $l \leftarrow C(b) + O(b, l)$
    if $k \leq l$ then
        $I \leftarrow I \cup \text{InexRecur}(W, i, z-1, k, l)$
        if $b = W[i]$ then
            $I \leftarrow I \cup \text{InexRecur}(W, i-1, z, k, l)$
        else
            $I \leftarrow I \cup \text{InexRecur}(W, i-1, z-1, k, l)$
```

Credit: Victoria Popic

BWA Inexact Match



```
$I \leftarrow \emptyset$
$I \leftarrow I \cup \text{INEXRECUR}(W, i-1, z-1, k, l)$
for each $b \in \{A, C, G, T\}$ do
    $k \leftarrow C(b) + O(b, k-1) + 1$
    $l \leftarrow C(b) + O(b, l)$
    if $k \leq l$ then
        $I \leftarrow I \cup \text{INEXRECUR}(W, i, z-1, k, l)$
        if $b = W[i]$ then
            $I \leftarrow I \cup \text{INEXRECUR}(W, i-1, z, k, l)$
        else
            $I \leftarrow I \cup \text{INEXRECUR}(W, i-1, z-1, k, l)$
```

W = ACTGTGT

Partial alignment 4-tuple: (i = 4, z = 3, L , U)

Recursive step:

A	C	T	G	gap-ref	gap-read
AGT	CGT	TGT	GGT	~~T~~GT	*GT
z-1	z-1	z	z-1	z-1	z-1
i-1	i-1	i-1	i-1	i-1	i
L^AU^A	L^CU^C	L^TU^T	L^GU^G	LU	L^AU^A L^CU^C L^TU^T L^GU^G
...GAGT ...GTGT	...GCGT ...GTGT	...GTGT ...GTGT	...GGGT ...GTGT	...G-GT ...GTGT	...GT[A/C/T/G]GT ...GT - GT

L^A=C(A) + O(A, L-1) + 1

U^A=C(A) + O(A, L)

Credit: Victoria Popic

BWA Heuristics

- Lower bound array D, where $D(i)$:= LB on number of differences of exactly matching R[0,i] with the reference (can be computed in O(|R|) time → check $n < D(i)$ instead of $n < 0$)

- Process best partial alignments first: use a *min*-priority heap to store alignment entries (instead of recursion)

- Prune out alignments considered sub-optimal (although they might have fewer than n differences):

  dynamically adjust search parameters (e.g. n):

  (1) stop if # top hits exceeds a threshold (=30),

  (2) set $n = nbest + 1$, where $nbest$ is the # of differences in top hit

- Seeding: limit the number of differences in the *seed* sequence (first k bp)

- Disallow indels at the ends of the read

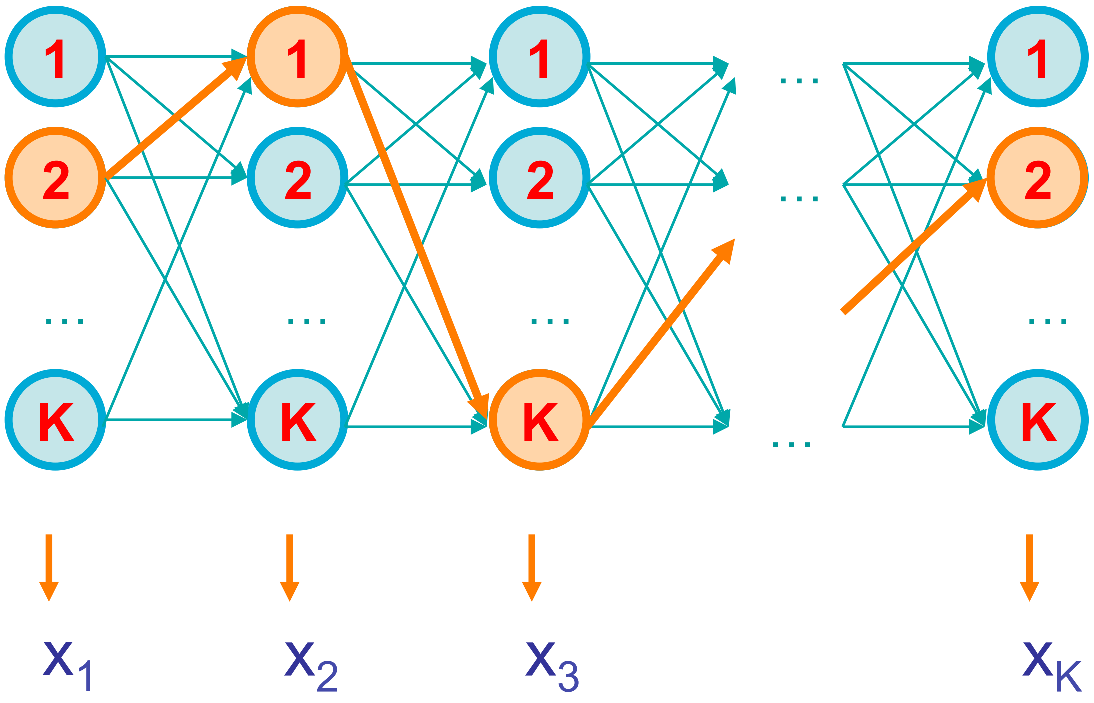
Li H, Durbin R.
Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 2009. 7154 cites

Langmead B, Salzberg SL.
Fast gapped-read alignment with Bowtie2. Nature Methods, 2012. 3017 cites

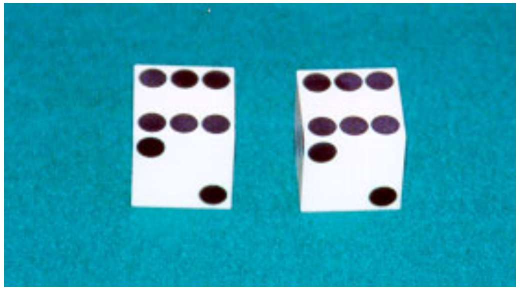
Li H
Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM

Credit: Victoria Popic

Hidden Markov Models



Example: The Dishonest Casino



A casino has two dice:

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

Casino player switches back-&-forth between fair and loaded die once every 20 turns

Game:

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 – Evaluation

GIVEN

A sequence of rolls by the casino player

$$\underbrace{12455264621461461361366616646616366163661636165156151151461235 62344}_{\boxed{\text{Prob} = 1.3 \times 10^{-35}}}$$

QUESTION

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

This is the **EVALUATION** problem in HMMs

Question # 2 – Decoding

GIVEN

A sequence of rolls by the casino player

124552646214614613613	6661664661636616366163616	515615115146123562344
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 # 3 – Learning

GIVEN

A sequence of rolls by the casino player

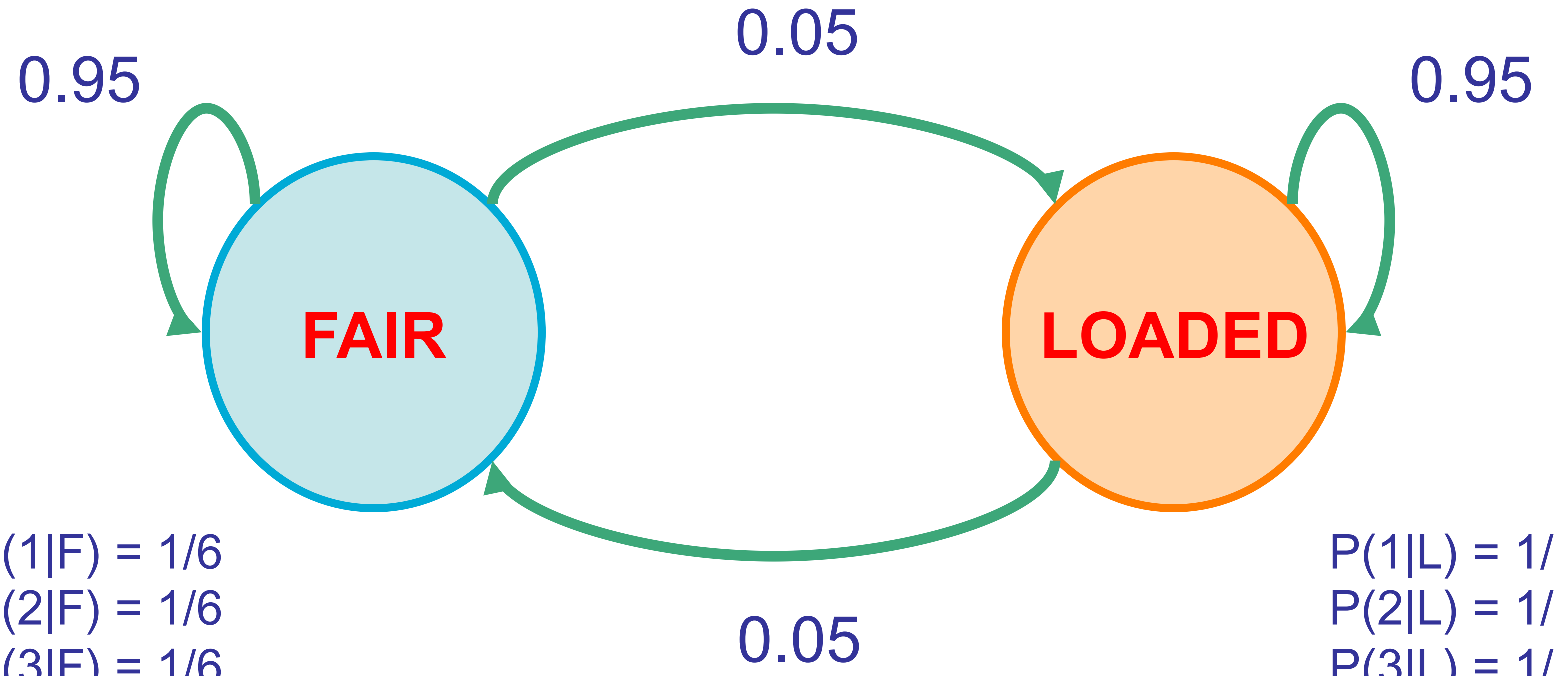
$$124552646214614613613\underbrace{6661664661636616366163616}_{\text{Prob(6) = 64\%}}515615115146123562344$$

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

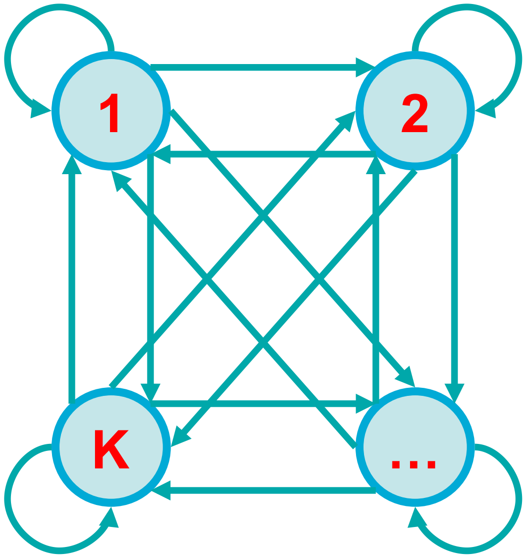


P(1|F) = 1/6
P(2|F) = 1/6
P(3|F) = 1/6
P(4|F) = 1/6
P(5|F) = 1/6
P(6|F) = 1/6

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

A HMM is memory-less

At each time step t,
the only thing that affects future states
is the current state π_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 $Q = \{ 1, ..., K \}$
- 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…K

- Start probabilities a_{0i}

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

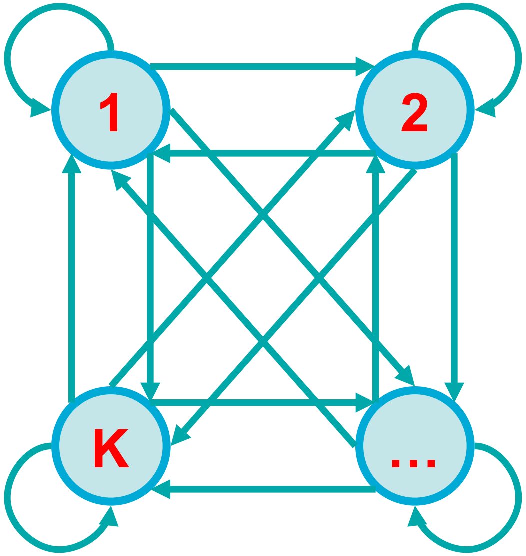
~~End Probabilities a_{i0}~~

in Durbin; not needed

- Emission probabilities within each state

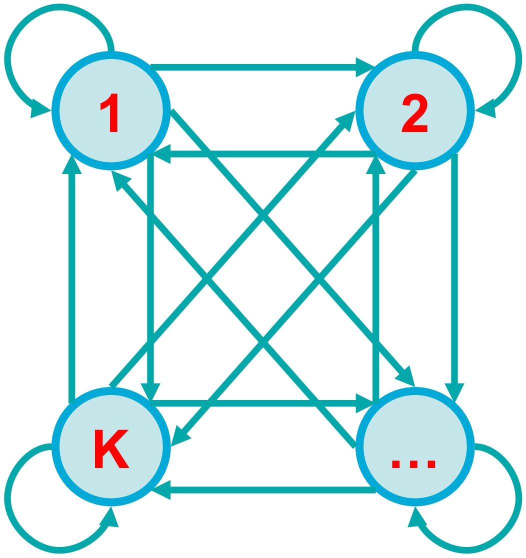
$e_i(b) = P(x_i = b \mid \pi_i = k)$

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



A HMM is memory-less

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



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

A HMM is memory-less

At each time step t,

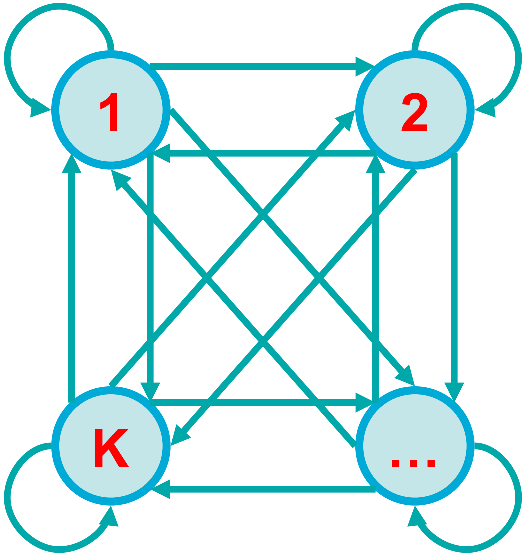
the only thing that affects x_t

is the current state π_t

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

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

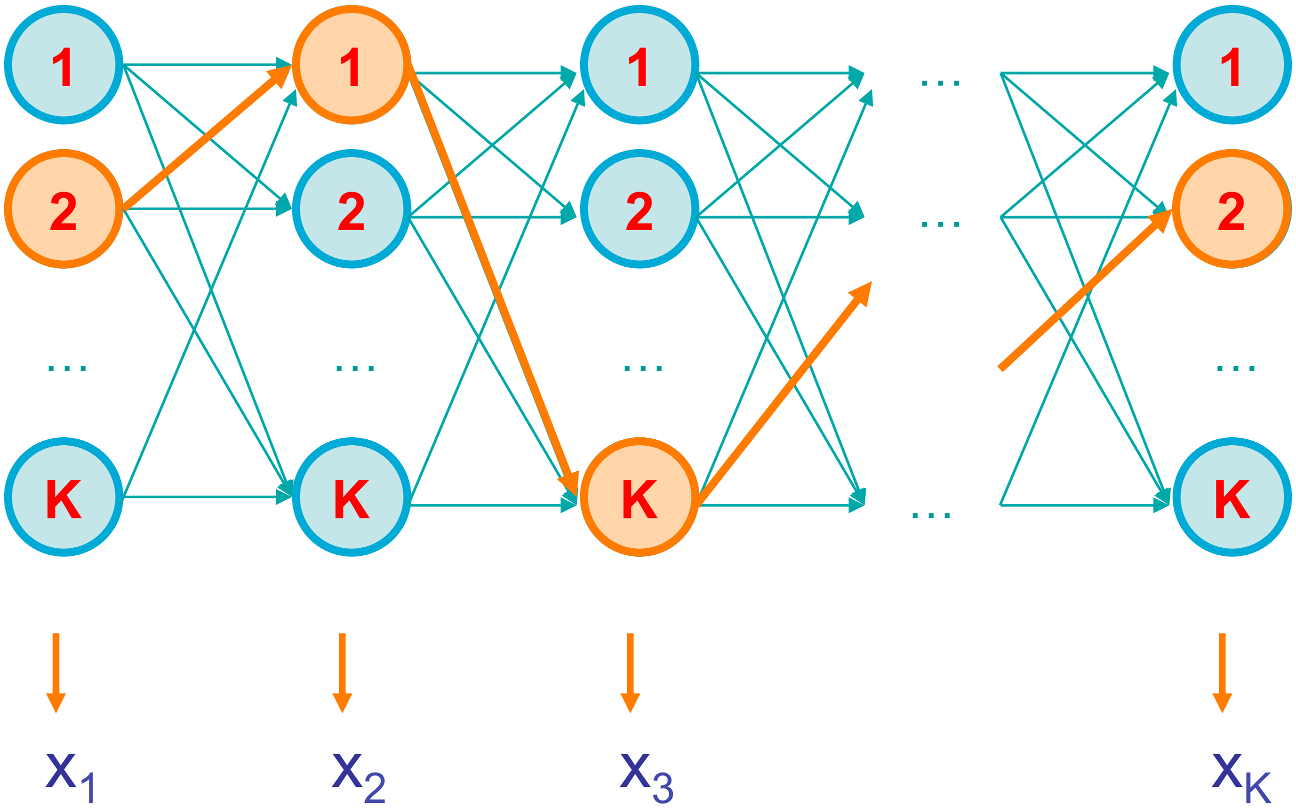
$P(x_t = b \mid \pi_t)$



A parse of a sequence

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

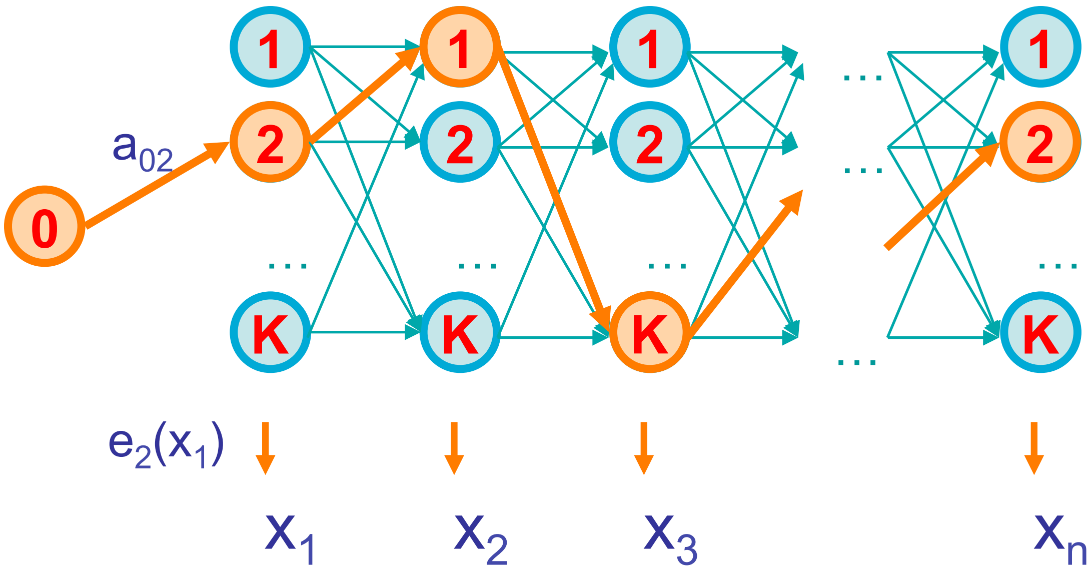
A parse of x is a sequence of states $\pi = \pi_1, \ldots\ldots, \pi_N$



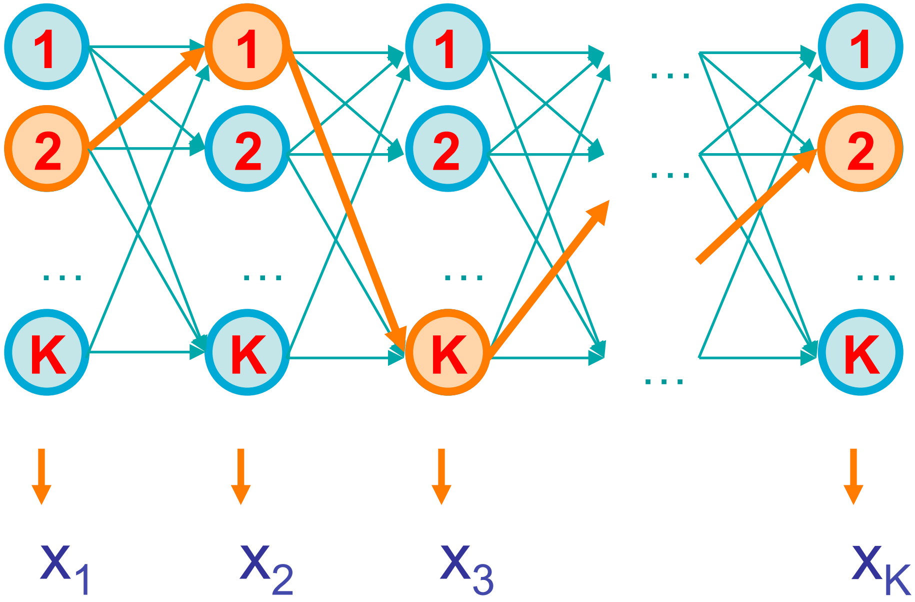
Generating a sequence by the model

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)

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

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

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

Likelihood of a parse

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

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

To find how likely this scenari

(given our HMM)

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

$P(x_N \mid \pi_N)\ P(\pi_N \mid \pi_{N-1})$

$a_{0\pi1}\ a_{\pi1\pi2} \ldots\ldots a_{\pi N\text{-}1\pi N}\ e_{\pi1}(x_1$

A compact way to write

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

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

Example:

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

Then, count in x and π the # of times each parameter j = 1, …, n occurs

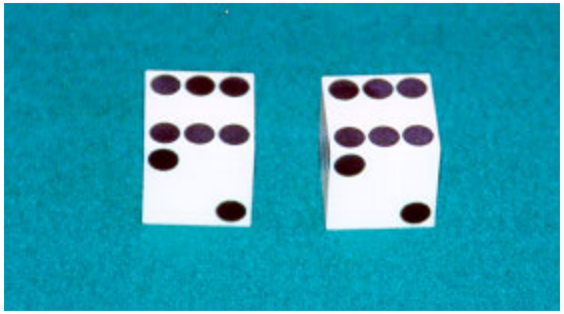
$F(j, x, \pi)$ = # parameter θ_j occurs in (x, π)

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

$$\mathbf{P(x, \pi) = \prod_{j=1\ldots n} \theta_j^{F(j,\, x,\, \pi)} =}$$

$$\mathbf{= \exp\left[\sum_{j=1\ldots n} \log(\theta_j) \times F(j, x, \pi)\right]}$$

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

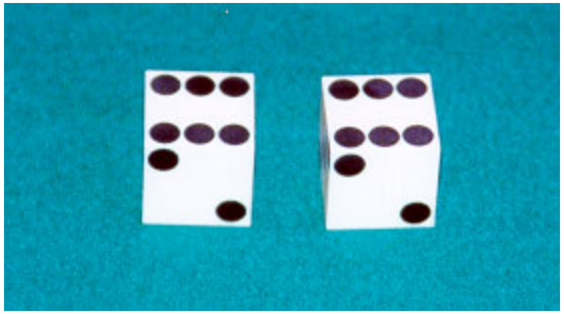
π = Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair?

(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

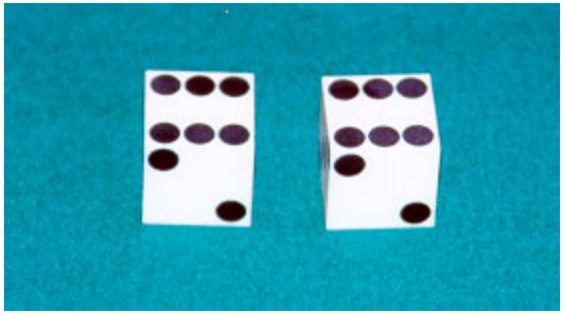
π = Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded?

½ × P(1 | Loaded) P(Loaded, Loaded) … P(4 | Loaded) =

$½ \times (1/10)^9 \times (1/2)^1 (0.95)^9 = .00000000015756235243 \sim= 0.16 \times 10^{-9}$

Therefore, it 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

Question # 1 – Evaluation

GIVEN

A sequence of rolls by the casino player

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

QUESTION

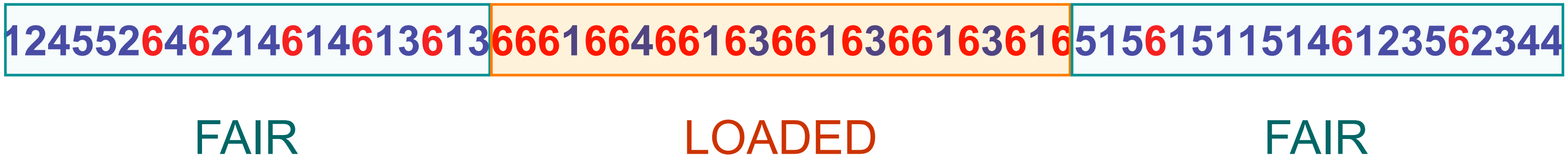
How likely is this sequence, given our model of how the casino works?

This is the **EVALUATION** problem in HMMs

Question # 2 – Decoding

GIVEN

A sequence of rolls by the casino player



124552646214614613613 6661664661636616366163616 515615115146123562344

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 # 3 – Learning

GIVEN

A sequence of rolls by the casino player

$$124552646214614613613\underbrace{6661664661636616366163616}_{\text{Prob(6) = 64\%}}515615115146123562344$$

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 three main questions on HMMs

1. **Evaluation**

   GIVEN a HMM M, and a sequence x,
   FIND Prob[x | M]

2. **Decoding**

   GIVEN a HMM M, and a sequence x,
   FIND the sequence π of states that maximizes P[x, π | 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 | θ]

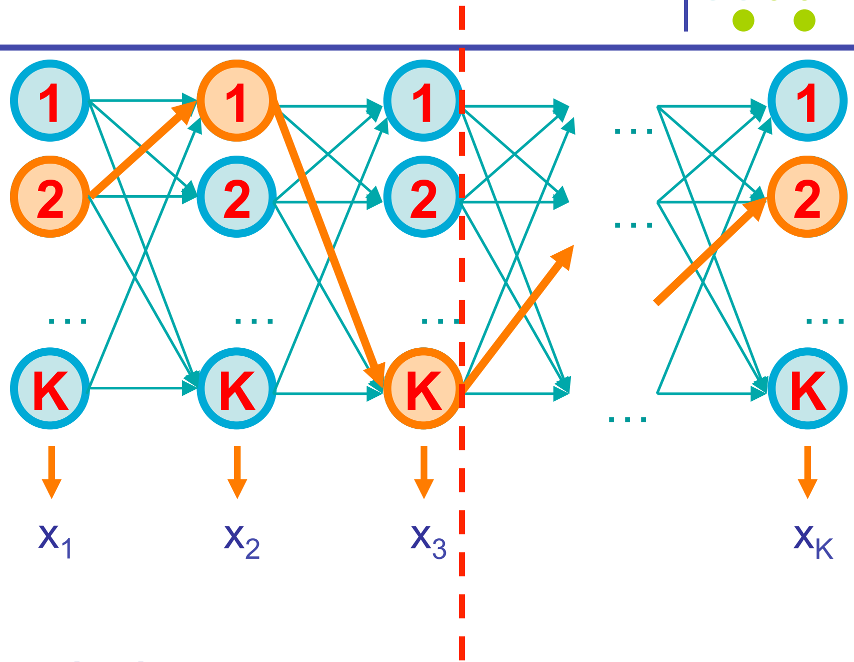
Problem 1: Decoding

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 $P[\, x, \pi \,]$



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

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

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

Dynamic Programming!

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

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

Decoding – main idea

Inductive assumption: 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:

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

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

Termination:

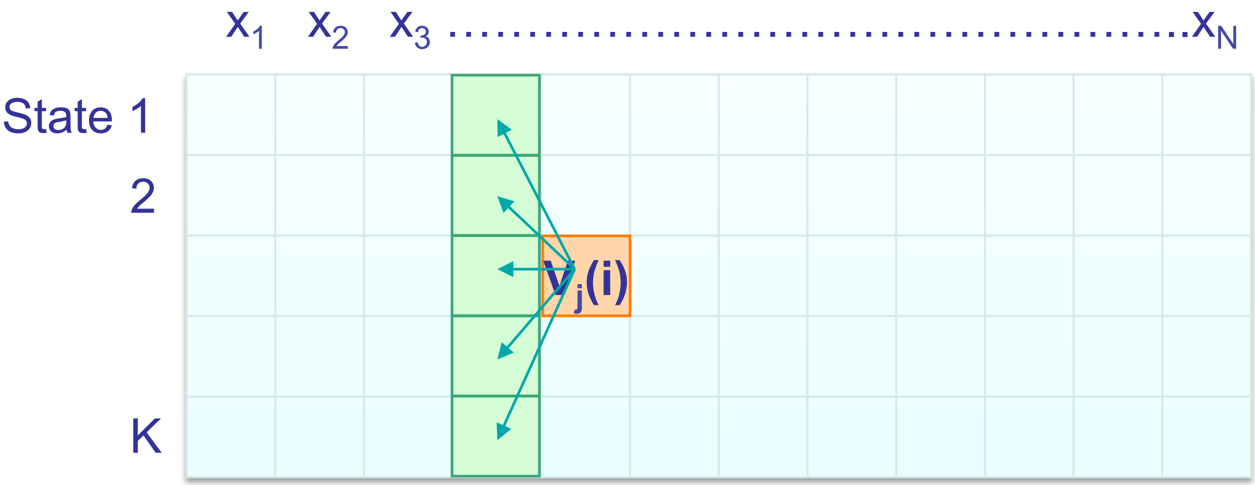
$P(x, \pi^*) = \max_k V_k(N)$

Traceback:

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

$\pi_{i-1}^* = \text{Ptr}_{\pi i} (i)$

The Viterbi Algorithm


x_1 x_2 x_3 .... x_N
State 1
2
K
$V_j(i)$

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

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

<u>Solution:</u> 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 $.95^6\times(1/6)^6 = 1.6\times10^{-5}$
parsed as L, contribute $.95^6\times(1/2)^1\times(1/10)^5 = 0.4\times10^{-5}$

“162636” parsed as F, contribute $.95^6\times(1/6)^6 = 1.6\times10^{-5}$
parsed as L, contribute $.95^6\times(1/2)^3\times(1/10)^3 = 9.0\times10^{-5}$

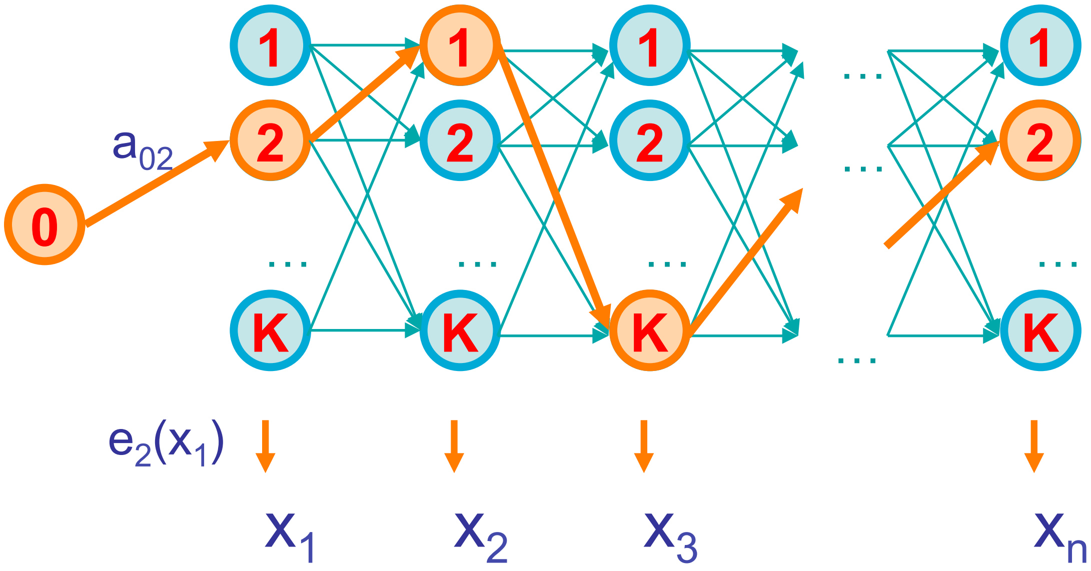
Problem 2: Evaluation

Find the likelihood 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 ca

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

P(box: LLLLLLLLLLL) =
$[\ (1/2)^{6} * (1/10)^{5}\ ] * 0.95^{10} * 0.05^{2}$ =
$1.56*10^{-7}\quad *\quad 1.5^{-3}$ =
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 → L → 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 ith state is k, given x

A more refined measure of which states 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 characters of x 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:

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

Termination:

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

Relation between Forward and Viterbi

VITERBI

Initialization:

$V_0(0) = 1$

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

Iteration:

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

Termination:

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

FORWARD

Initialization:

$f_0(0) = 1$

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

Iteration:

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

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 ith 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:

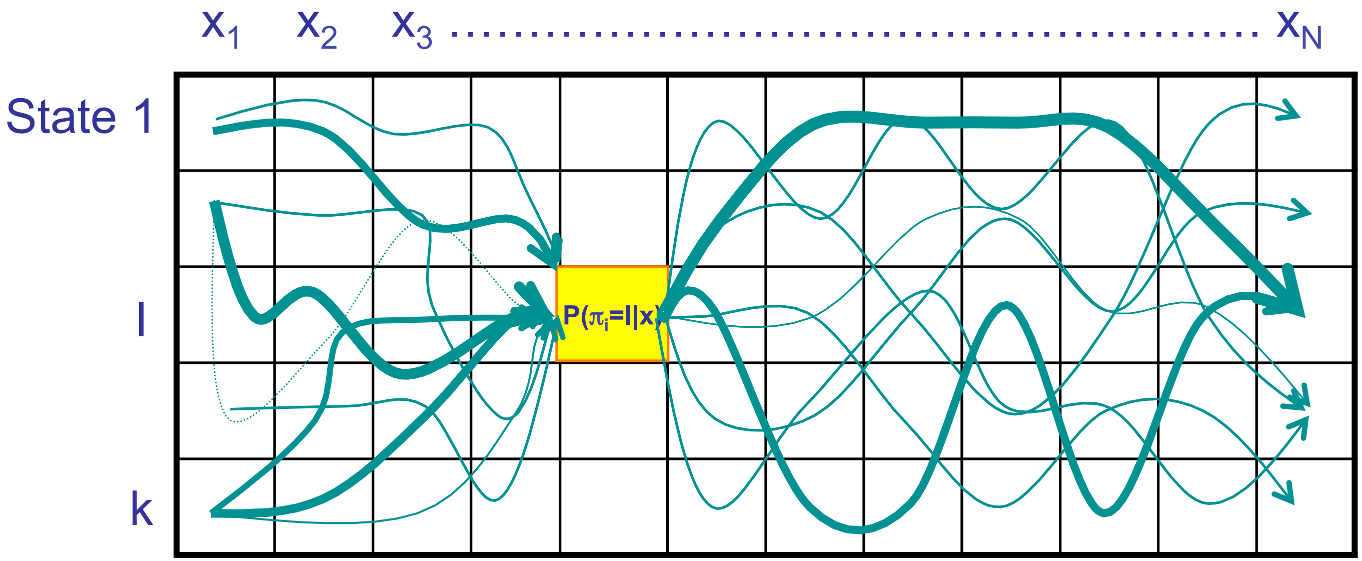
Define $\pi^{\wedge}$ by Posterior Decoding:

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

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 prob 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:

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

Termination:

$$P(x) = \mathbf{\Sigma_k}\ f_k(N)$$

BACKWARD

Initialization:

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

Iteration:

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

Termination:

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