Hidden Markov Models

Ben Langmead

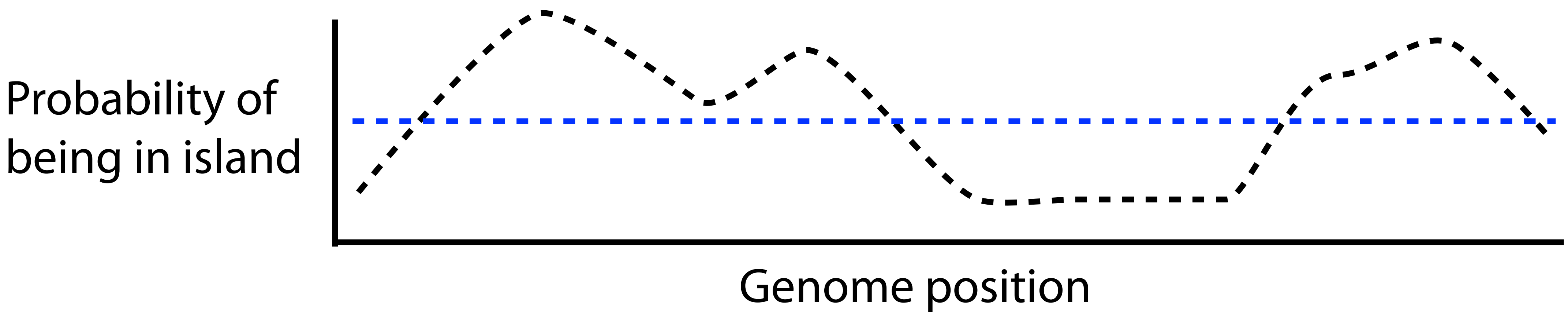
JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Department of Computer Science

Sequence models

Can we use Markov chains to pick out CpG islands from the rest of the genome?

Markov chain assigns a score to a string; doesn't naturally give a "running" score across a long sequence

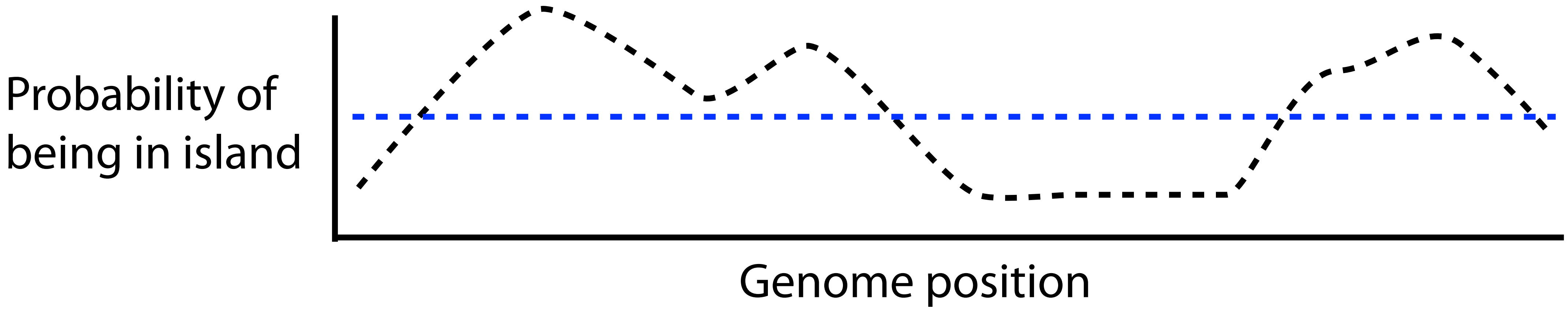


We could use a *sliding window*

(a) Pick window size *w*, (b) score every *w*-mer using Markov chains, (c) use a cutoff to find islands

Smoothing before (c) might also be a good idea

Sequence models



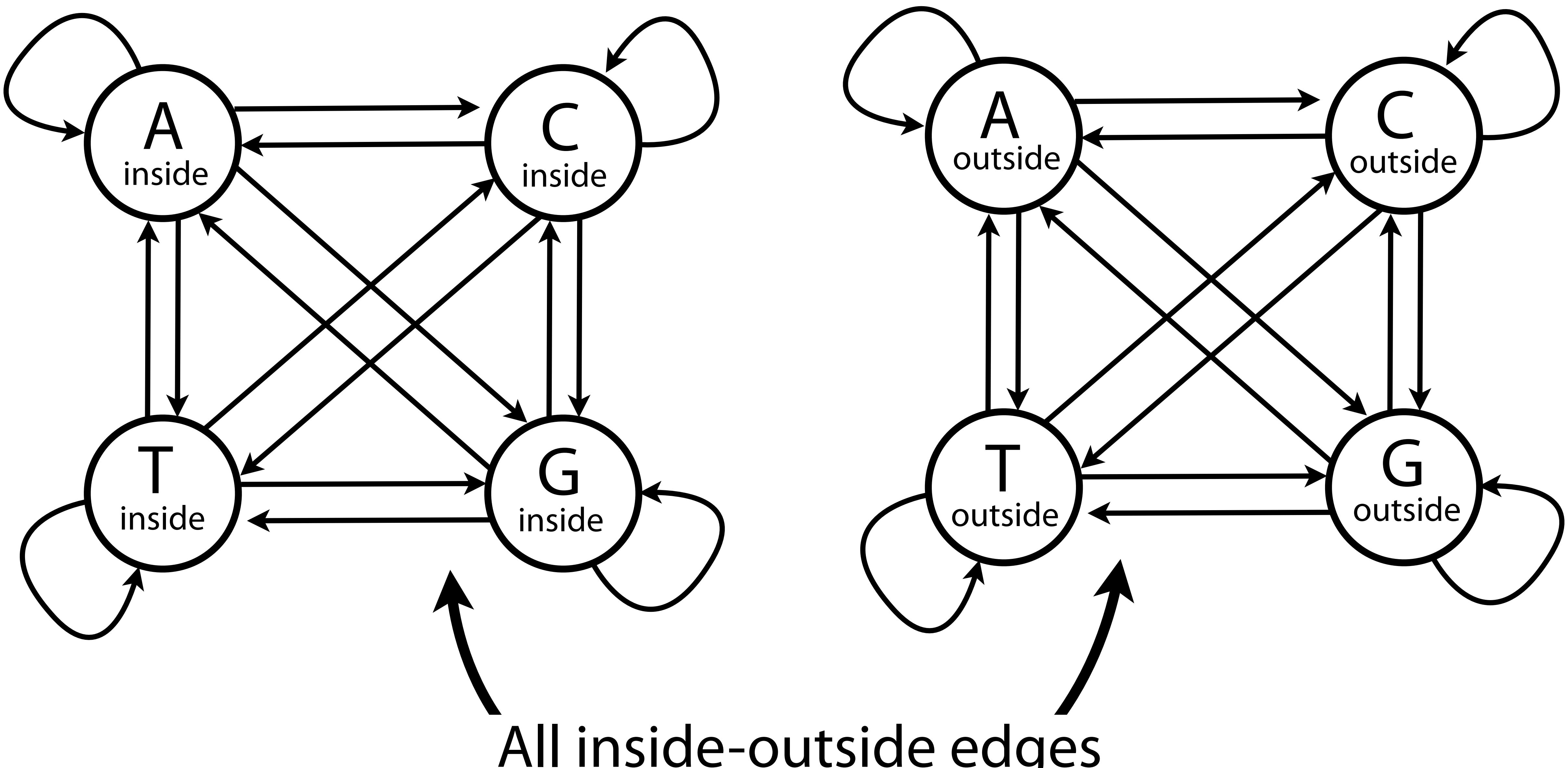
Choosing *w* involves an assumption about how long the islands are

- If *w* is too large, we'll miss small islands
- If *w* is too small, we'll get many small islands where perhaps we should see fewer larger ones

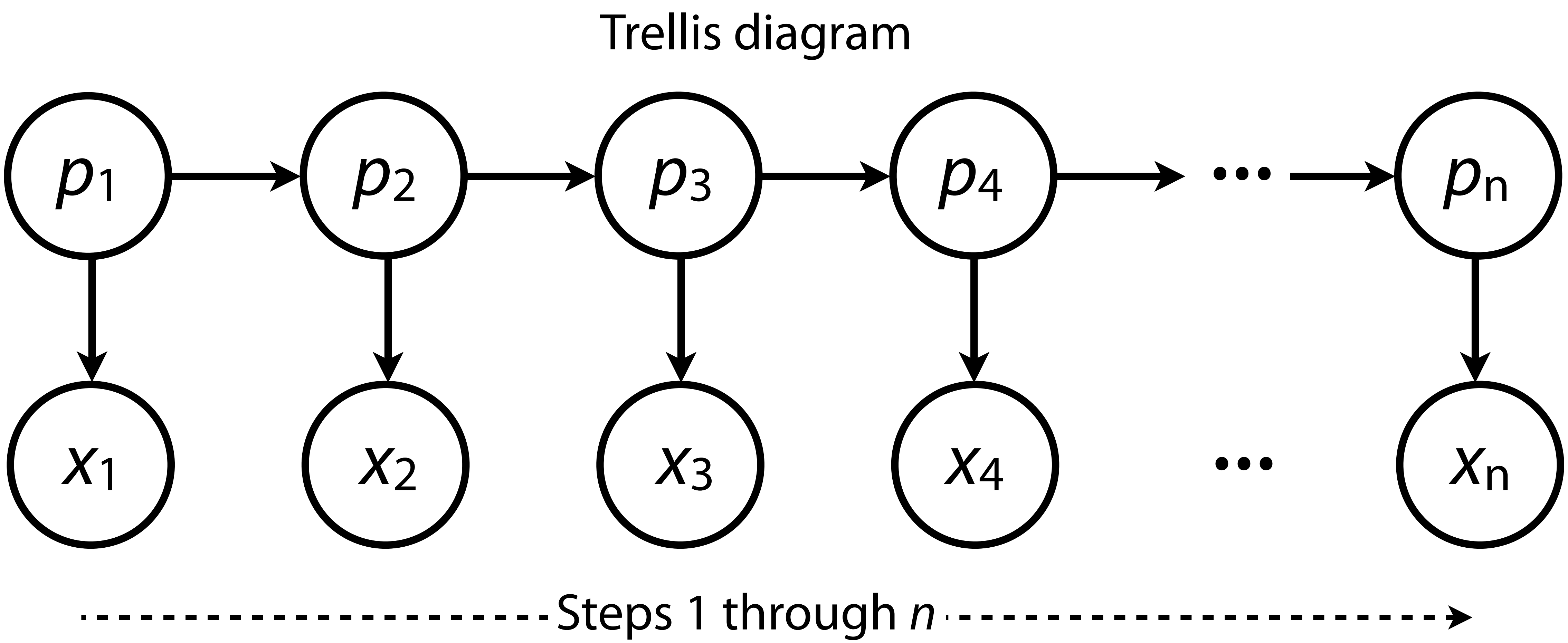
In a sense, we want to switch *between Markov chains* when entering or exiting a CpG island

Sequence models

Something like this:



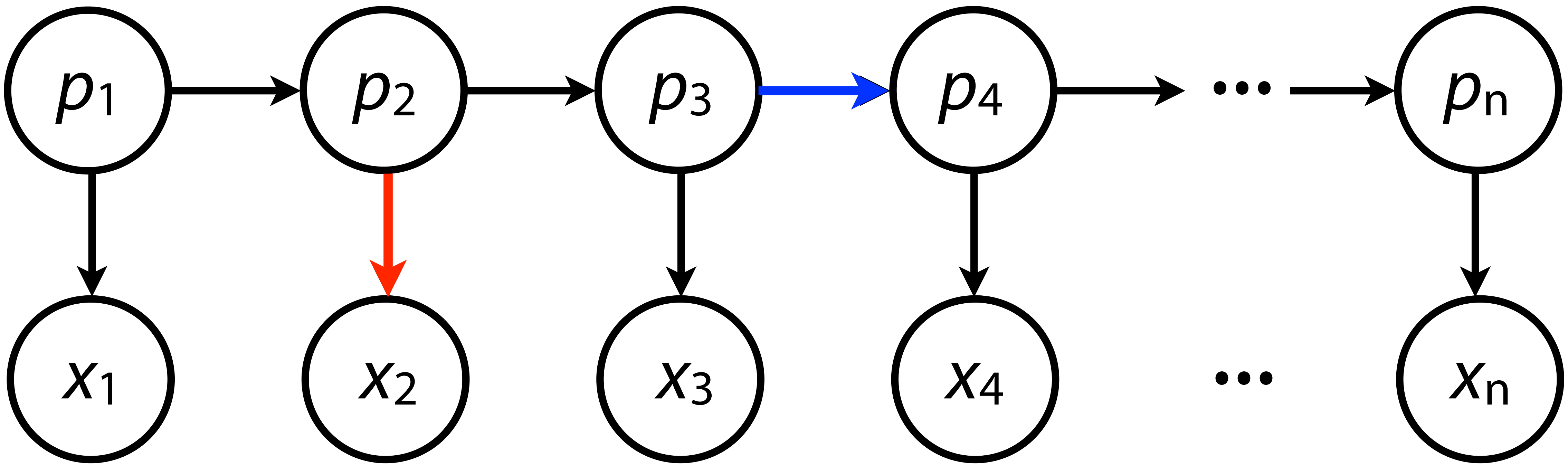
Hidden Markov Model



$p = \{ p_1, p_2, ..., p_n \}$ is a sequence of *states* (AKA a *path*). Each p_i takes a value from set Q. We ***do not*** observe p.

$x = \{ x_1, x_2, ..., x_n \}$ is a sequence of *emissions*. Each x_i takes a value from set Σ. We ***do*** observe x.

Hidden Markov Model



Like for Markov chains, edges capture conditional independence:

x_2 is conditionally independent of everything else given p_2

p_4 is conditionally independent of everything else given p_3

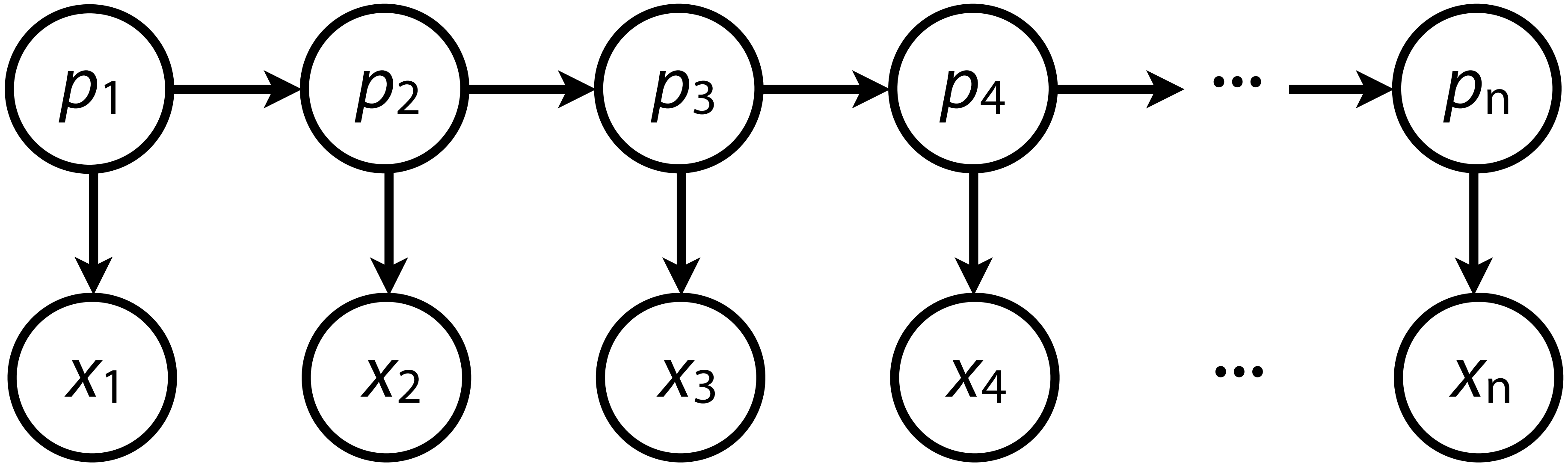
Probability of being in a particular state at step *i* is known once we know what state we were in at step *i*-1. Probability of seeing a particular emission at step *i* is known once we know what state we were in at step *i*.

Hidden Markov Model

Example: occasionally dishonest casino

Dealer repeatedly flips a coin. Sometimes the coin is *fair*, with P(heads) = 0.5, sometimes it's *loaded*, with P(heads) = 0.8. Dealer occasionally switches coins, invisibly to you.

How does this map to an HMM?

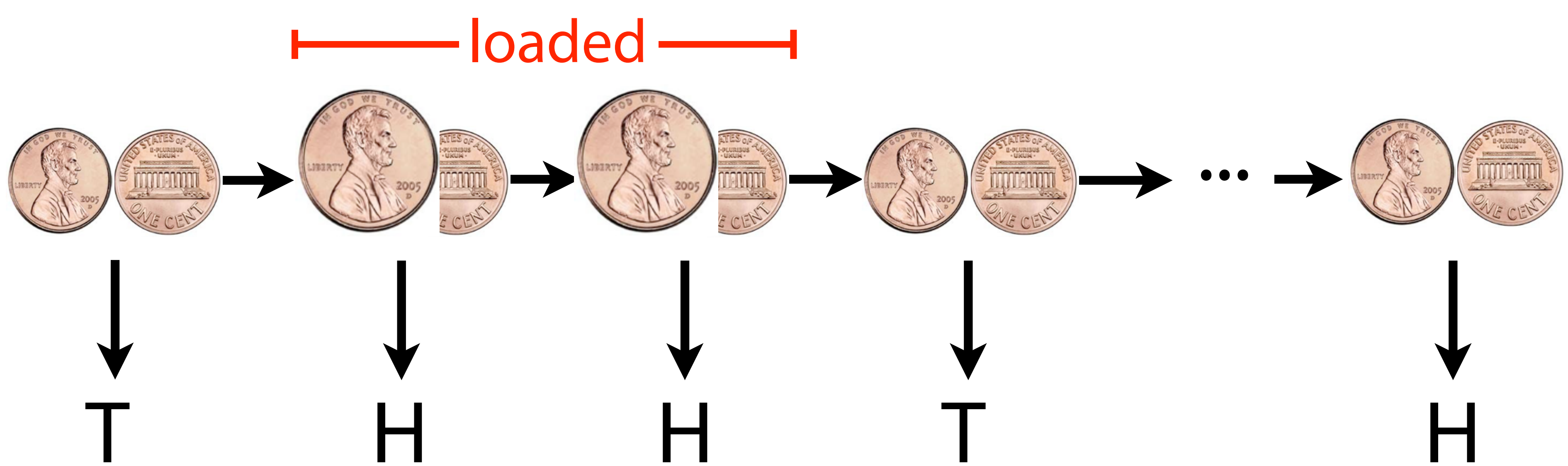


Hidden Markov Model

Example: occasionally dishonest casino

Dealer repeatedly flips a coin. Sometimes the coin is *fair*, with P(heads) = 0.5, sometimes it's *loaded*, with P(heads) = 0.8. Dealer occasionally switches coins, invisibly to you.

How does this map to an HMM?



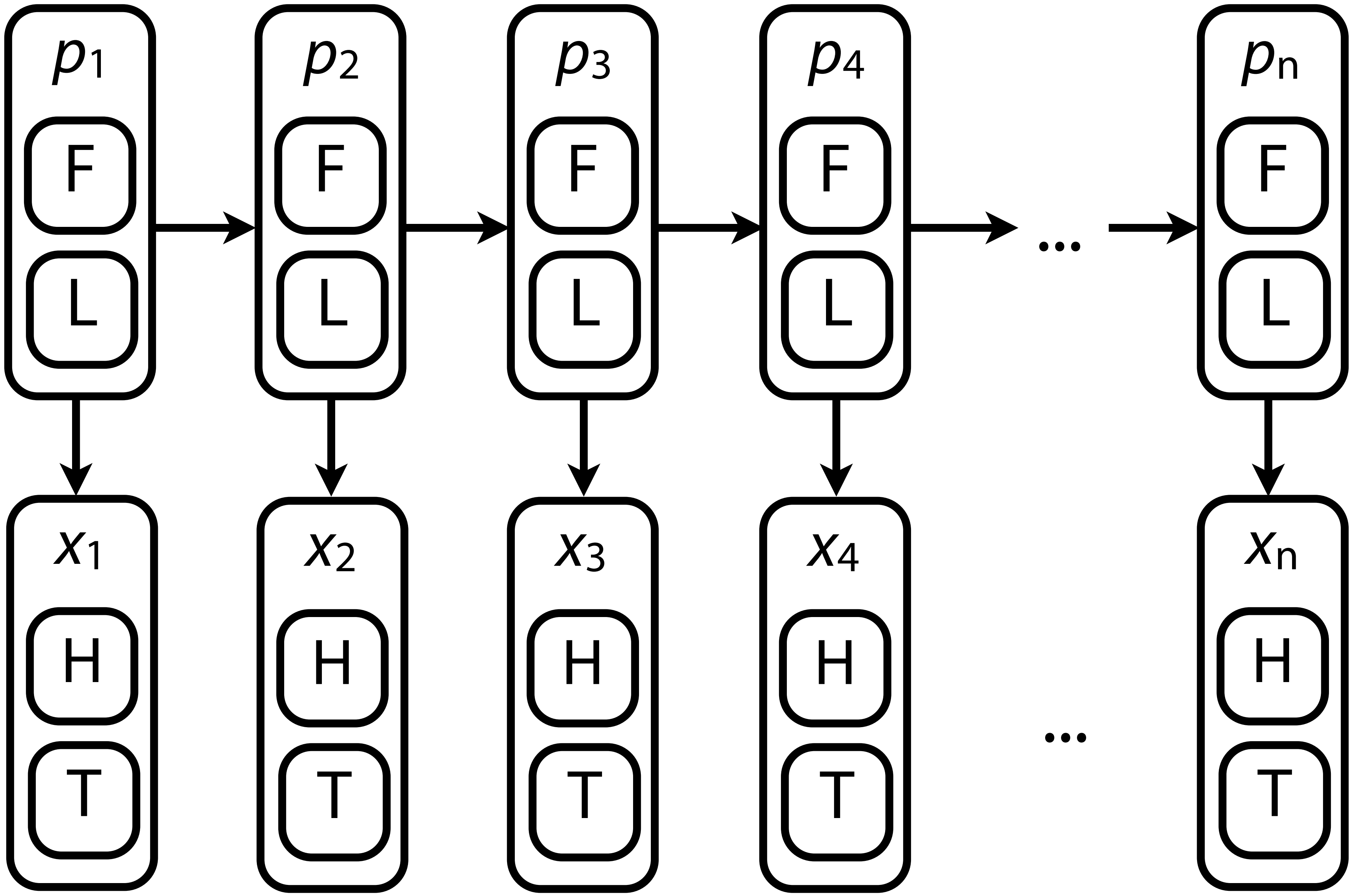
Emissions encode flip outcomes (observed), states encode loadedness (hidden)

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Hidden Markov Model

States encode which coin is used

F = fair
L = loaded



Emissions encode flip outcomes

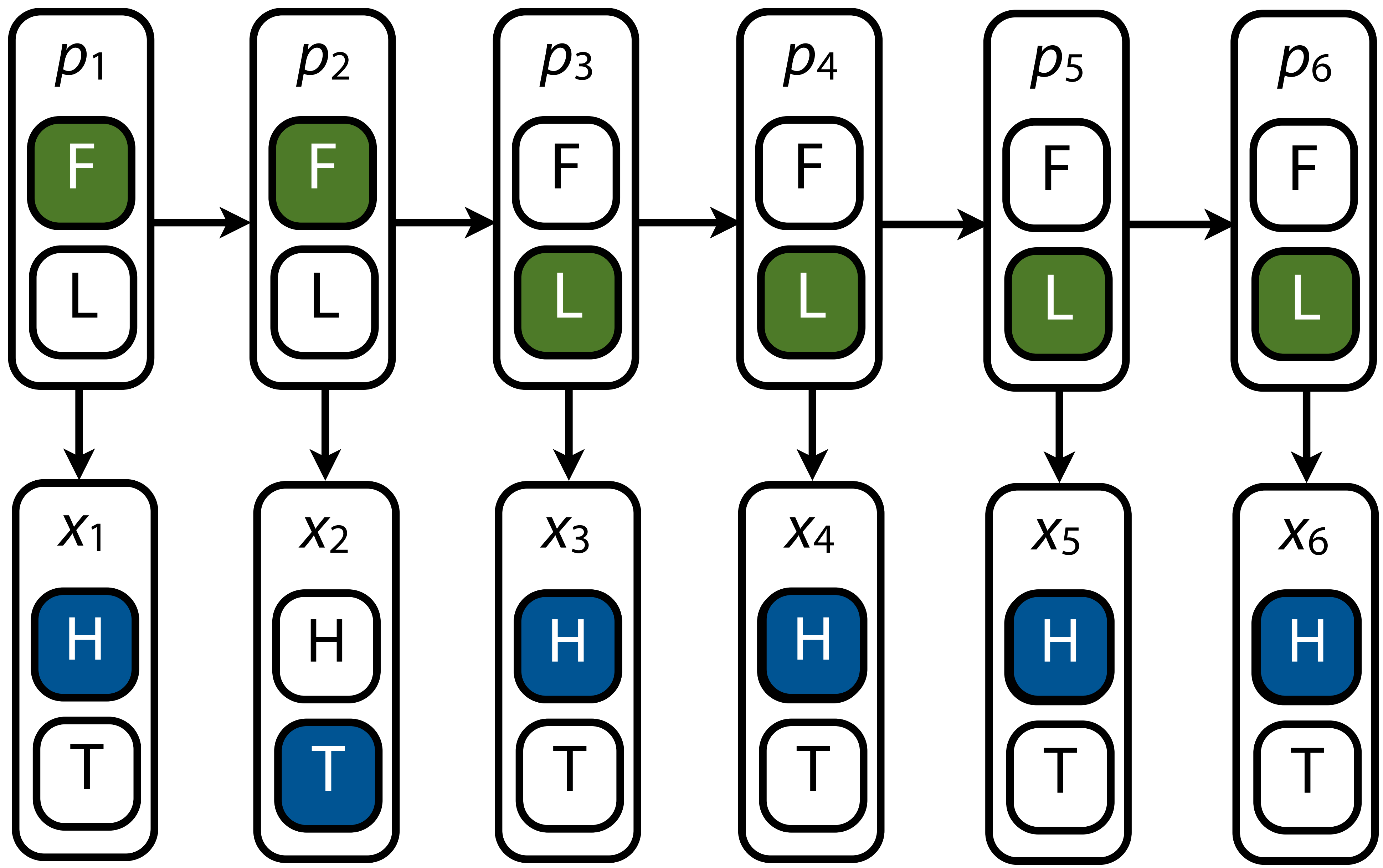
H = heads
T = tails

Hidden Markov Model

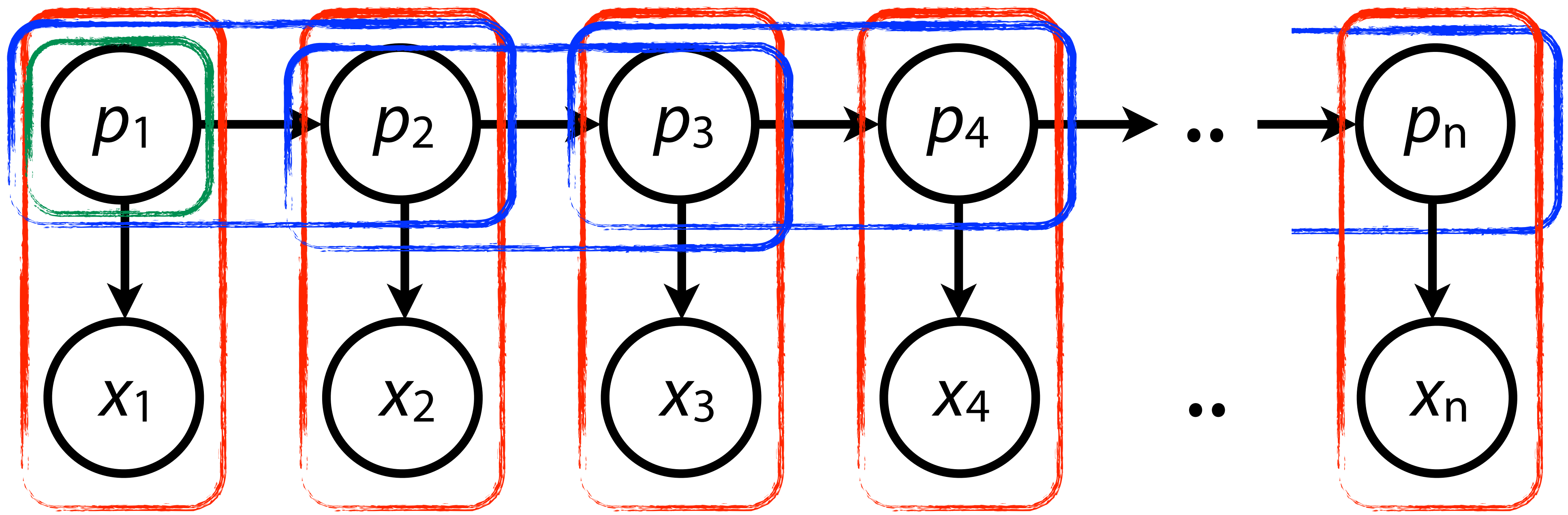
Example with six coin flips:

$p = \text{FFLLLL}$

$x = \text{HTHHHH}$



Hidden Markov Model



$$P(p_1, p_2, \ldots, p_n, x_1, x_2, \ldots, x_n) = \prod_{k=1}^{n} P(x_k \mid p_k) \prod_{k=2}^{n} P(p_k \mid p_{k-1})\, P(p_1)$$

$|Q| \times |\Sigma|$ emission matrix E encodes $P(x_i \mid p_i)$s $E[p_i, x_i] = P(x_i \mid p_i)$

$|Q| \times |Q|$ transition matrix A encodes $P(p_i \mid p_{i-1})$s $A[p_{i-1}, p_i] = P(p_i \mid p_{i-1})$

$|Q|$ array I encodes initial probabilities of each state $I[p_i] = P(p_1)$

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Hidden Markov Model

Dealer repeatedly flips a coin. Coin is sometimes *fair*, with P(heads) = 0.5, sometimes *loaded*, with P(heads) = 0.8. Dealer occasionally switches coins, invisibly to you.

After each flip, dealer switches coins with probability 0.4

A:

	F	L
F	0.6	0.4
L	0.4	0.6

E:

	H	T
F	0.5	0.5
L	0.8	0.2

$|Q| \times |\Sigma|$ emission matrix E encodes $P(x_i \mid p_i)$s $\quad E[p_i, x_i] = P(x_i \mid p_i)$

$|Q| \times |Q|$ transition matrix A encodes $P(p_i \mid p_{i-1})$s $\quad A[p_{i-1}, p_i] = P(p_i \mid p_{i-1})$

Hidden Markov Model

A	F	L
F	0.6	0.4
L	0.4	0.6

E	H	T
F	0.5	0.5
L	0.8	0.2

Given A & E (right), what is the joint probability of $\boldsymbol{p}$ & $\boldsymbol{x}$?

$\boldsymbol{p}$	F	F	F	L	L	L	F	F	F	F	F
$\boldsymbol{x}$	T	H	T	H	H	H	T	H	T	T	H
P(x_i | p_i)	0.5	0.5	0.5	0.8	0.8	0.8	0.5	0.5	0.5	0.5	0.5
P(p_i | p_{i-1})	-	0.6	0.6	0.4	0.6	0.6	0.4	0.6	0.6	0.6	0.6

If P(p_1 = F) = 0.5, then joint probability = $0.5^9\ 0.8^3\ 0.6^8\ 0.4^2$ = 0.0000026874

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Hidden Markov Model

Given flips, can we say when the dealer was using the loaded coin?

We want to find p^*, the most likely path given the emissions.

$$p^* = \underset{p}{\operatorname{argmax}}\, \mathrm{P}(\, p \mid x\,) \;\; = \underset{p}{\operatorname{argmax}}\, \mathrm{P}(\, p, x\,)$$

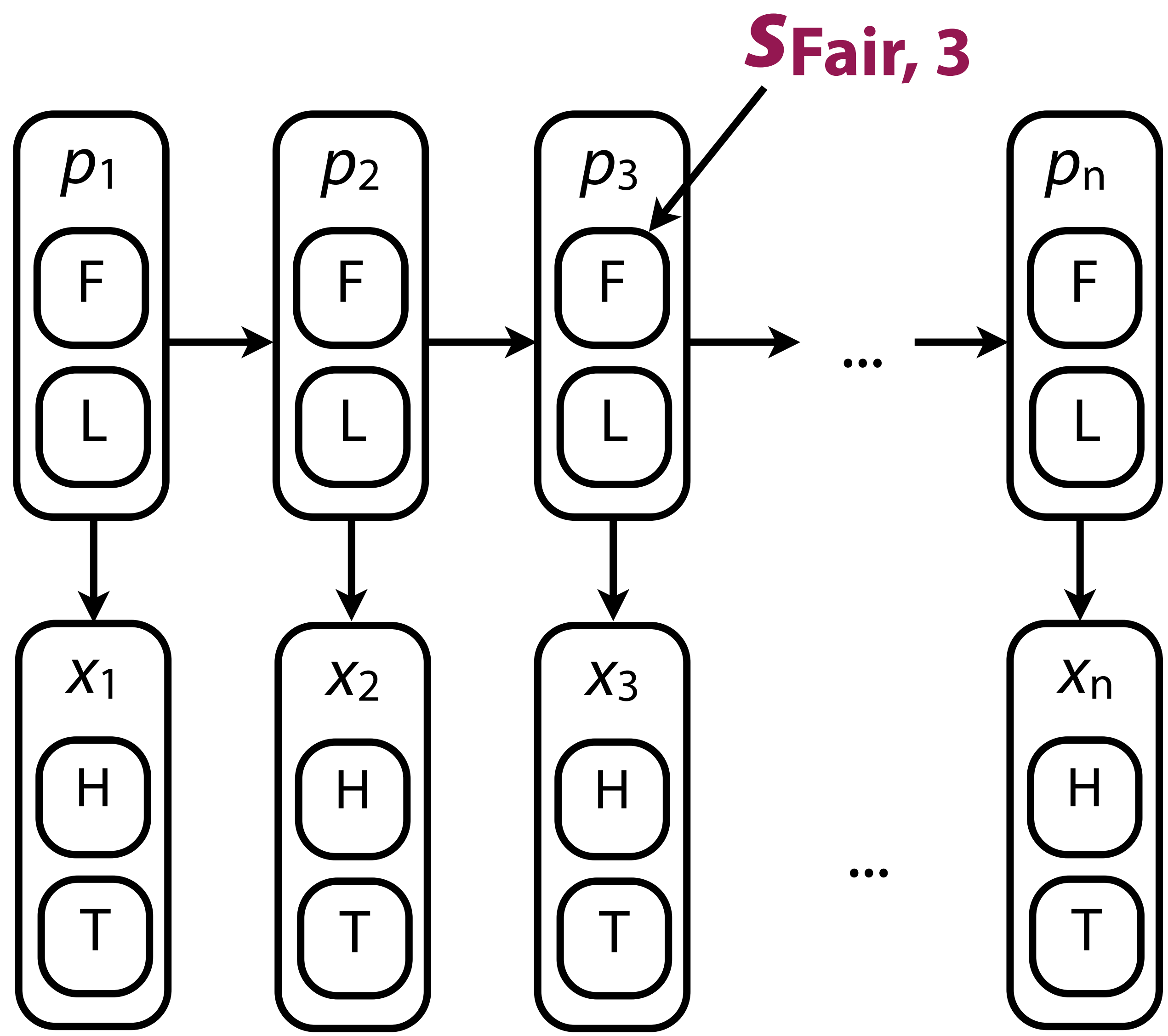
This is *decoding*. *Viterbi* is a common decoding algorithm.

Hidden Markov Model: Viterbi algorithm

Bottom-up dynamic programming

$s_{k,\,i}$ = score of the most likely path up to step *i* with p_i = k

Start at step 1, calculate successively longer $s_{k,\,i}$'s



JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Hidden Markov Model: Viterbi algorithm

Given transition matrix *A* and emission matrix *E* (right), what is the most probable path *p* for the following *x*?

Initial probabilities of F/L are 0.5

A	F	L
F	0.6	0.4
L	0.4	0.6

E	H	T
F	0.5	0.5
L	0.8	0.2

p	?	?	?	?	?	?	?	?	?	?	?
x	T	H	T	H	H	H	T	H	T	T	H
$s_{\text{Fair}, i}$	0.25	?	?	?	?	?	?	?	?	?	?
$s_{\text{Loaded}, i}$	0.1	?	?	?	?	?	?	?	?	?	?

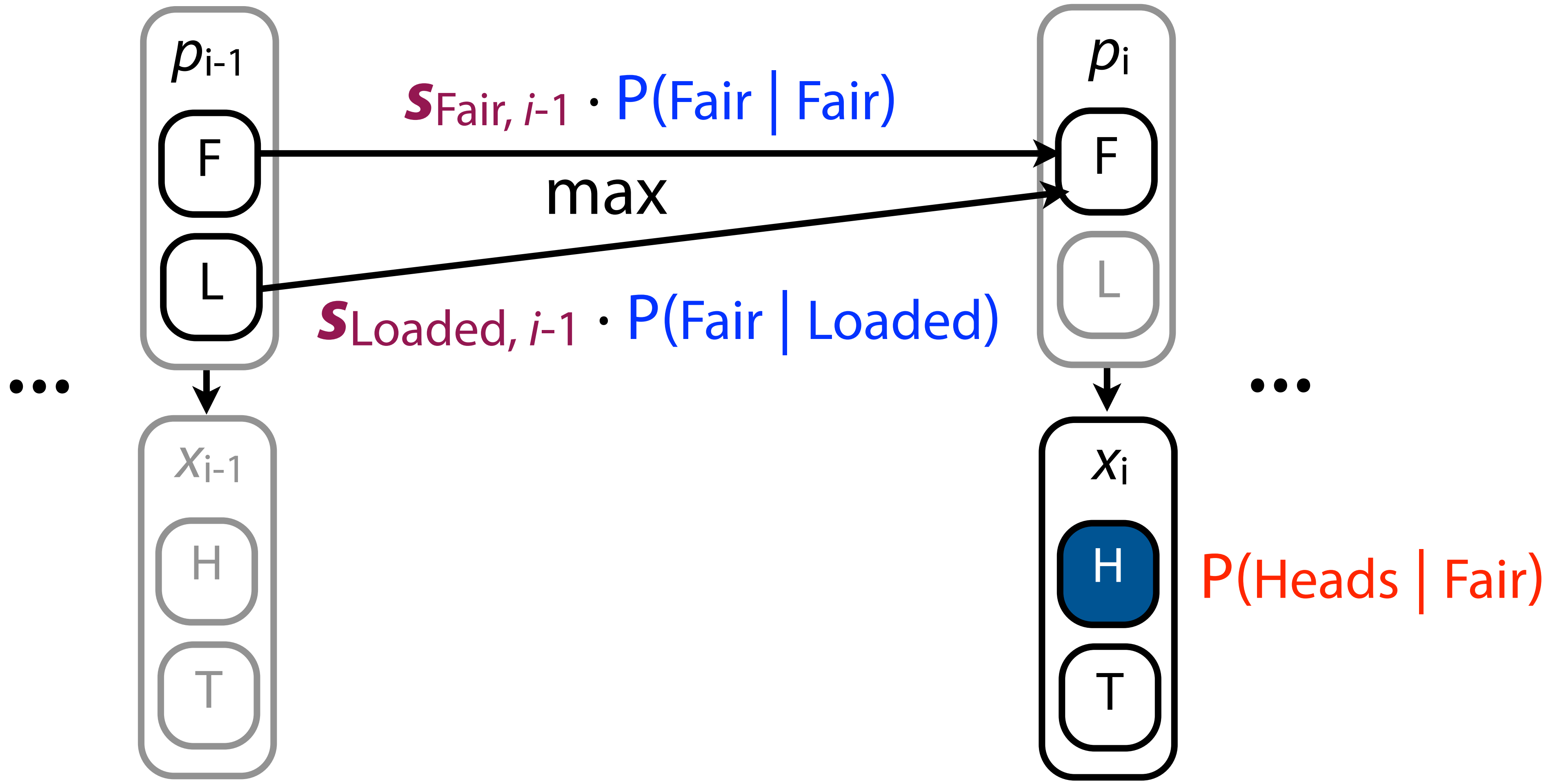
Viterbi fills in all the question marks

Hidden Markov Model: Viterbi algorithm

$$s_{Fair,i} = P(\text{Heads} \mid \text{Fair}) \cdot \max_{k \in \{\text{Fair, Loaded}\}} \{ s_{k,i-1} \cdot P(\text{Fair} \mid k) \}$$

Emission prob

Transition prob



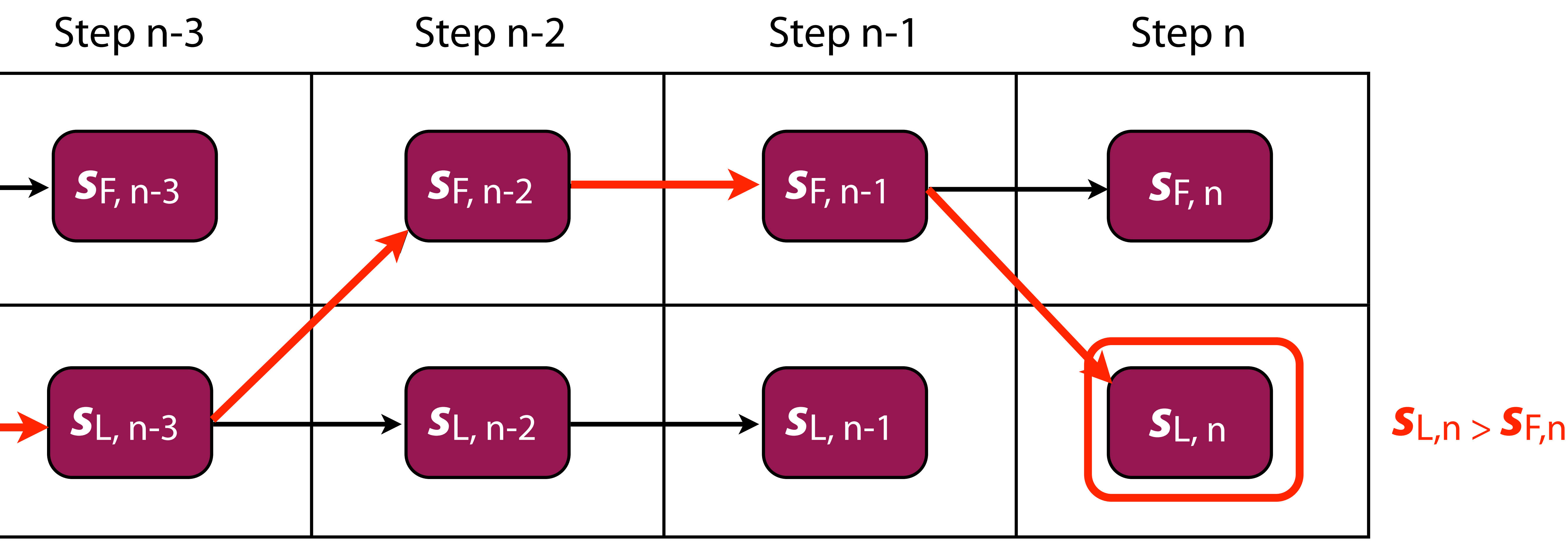
Hidden Markov Model: Viterbi algorithm

	Step 1	Step 2	Step 3	Step 4	...
x	**T**	**H**	**T**	**H**	
$S_{F,i}$	$p(T\|F) = 0.5$ $p(F) = 0.5$ $S_{F,1} = 0.5^2$	$p(H\|F) = 0.5$ max: $p(F\|F) = 0.6$ $0.5^3 \cdot 0.6$ $p(F\|L) = 0.4$ $0.2 \cdot 0.4 \cdot 0.5^2$ F wins max $S_{F,2} = 0.5^3 \cdot 0.6$	$p(T\|F) = 0.5$ max: $p(F\|F) = 0.6$ $0.5^4 \cdot 0.6^2$ $p(F\|L) = 0.4$ $0.2 \cdot 0.4 \cdot 0.5^2 \cdot 0.6 \cdot 0.8$ F wins max $S_{F,3} = 0.5^4 \cdot 0.6^2$	etc	
$S_{L,i}$	$p(T\|L) = 0.2$ $p(L) = 0.5$ $S_{L,1} = 0.2 \cdot 0.5$	$p(H\|L) = 0.8$ max: $p(L\|F) = 0.4$ $0.4 \cdot 0.5^2 \cdot 0.8$ $p(L\|L) = 0.6$ $0.2 \cdot 0.5 \cdot 0.6 \cdot 0.8$ F wins max $S_{L,2} = 0.4 \cdot 0.5^2 \cdot 0.8$	etc	etc	

Hidden Markov Model: Viterbi algorithm

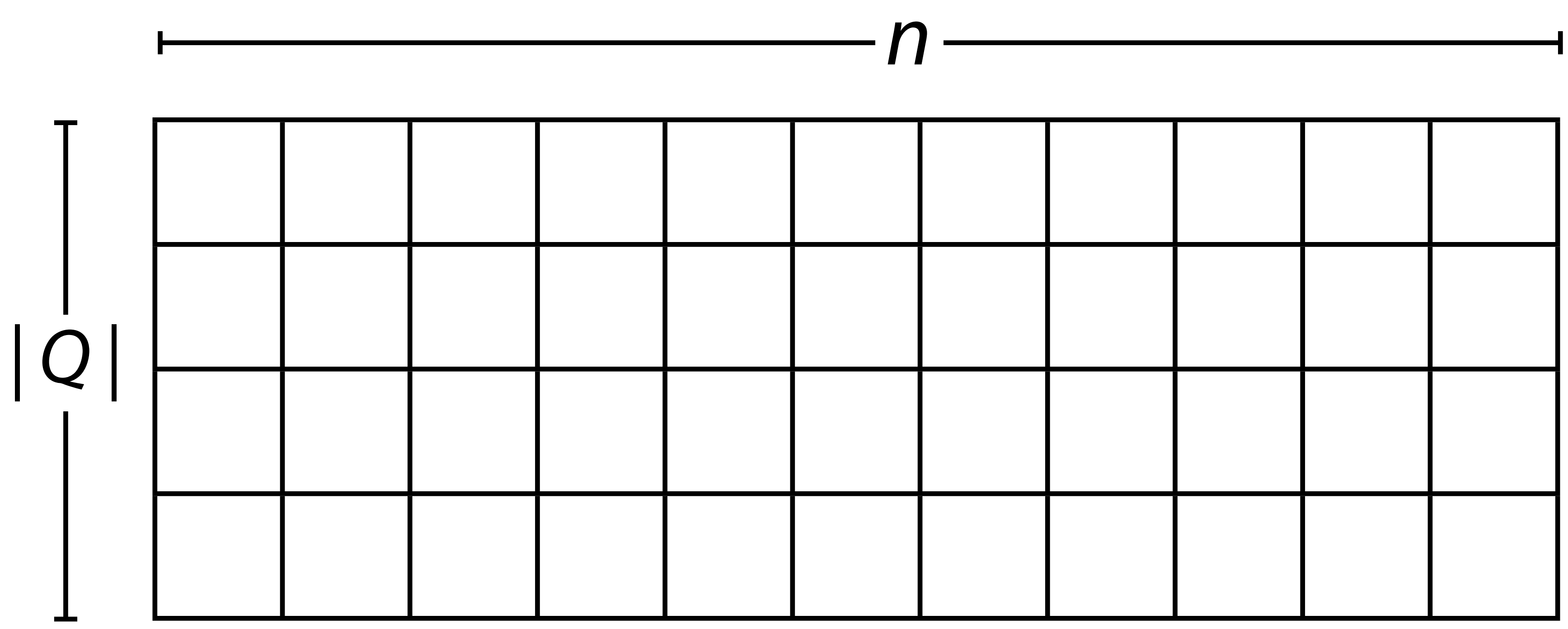
Pick state in step *n* with highest score; *backtrace* for most likely path

Backtrace according to which state *k* "won" the max in:



Hidden Markov Model: Viterbi algorithm

How much work did we do, given Q is the set of states and n is the length of the sequence?



\# $s_{k,i}$ values to calculate $= n \cdot |Q|$, each involves max over $|Q|$ products

$O(n \cdot |Q|^2)$

Matrix A has $|Q|^2$ elements, E has $|Q||\Sigma|$ elements, I has $|Q|$ elements

Hidden Markov Model: Implementation

```python
def viterbi(self, x):
    ''' Given sequence of emissions, return the most probable path
        along with its joint probability. '''
    x = map(self.smap.get, x) # turn emission characters into ids
    nrow, ncol = len(self.Q), len(x)
    mat   = numpy.zeros(shape=(nrow, ncol), dtype=float) # prob
    matTb = numpy.zeros(shape=(nrow, ncol), dtype=int)   # backtrace
    # Fill in first column
    for i in xrange(0, nrow):
        mat[i, 0] = self.E[i, x[0]] * self.I[i]
    # Fill in rest of prob and Tb tables
    for j in xrange(1, ncol):
        for i in xrange(0, nrow):
            ep = self.E[i, x[j]]
            mx, mxi = mat[0, j-1] * self.A[0, i] * ep, 0
            for i2 in xrange(1, nrow):
                pr = mat[i2, j-1] * self.A[i2, i] * ep
                if pr > mx:
                    mx, mxi = pr, i2
            mat[i, j], matTb[i, j] = mx, mxi
    # Find final state with maximal probability
    omx, omxi = mat[0, ncol-1], 0
    for i in xrange(1, nrow):
        if mat[i, ncol-1] > omx:
            omx, omxi = mat[i, ncol-1], i
    # Backtrace
    i, p = omxi, [omxi]
    for j in xrange(ncol-1, 0, -1):
        i = matTb[i, j]
        p.append(i)
    p = ''.join(map(lambda x: self.Q[x], p[::-1]))
    return omx, p # Return probability and path
```

`mat` holds the $s_{k,i}$'s

`matTb` holds traceback info

`self.E` holds emission probs

`self.A` holds transition probs

`self.I` holds initial probs

Calculate $s_{k,i}$'s

Find maximal $s_{k,n}$

Backtrace

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Hidden Markov Model: Viterbi algorithm

```
>>> hmm = HMM({"FF":0.6, "FL":0.4, "LF":0.4, "LL":0.6},
...           {"FH":0.5, "FT":0.5, "LH":0.8, "LT":0.2},
...           {"F":0.5, "L":0.5})
>>> prob, _ = hmm.viterbi("THTHHHTHTTH")
>>> print prob
2.86654464e-06
>>> prob, _ = hmm.viterbi("THTHHHTHTTH" * 100)
>>> print prob
0.0
```

Repeat string 100 times

Occasionally dishonest casino setup

What happened? Underflow!

Python example: http://nbviewer.ipython.org/7460513

Hidden Markov Model: Viterbi algorithm

```
>>> hmm = HMM({"FF":0.6, "FL":0.4, "LF":0.4, "LL":0.6},
...           {"FH":0.5, "FT":0.5, "LH":0.8, "LT":0.2},
...           {"F":0.5, "L":0.5})
>>> prob, _ = hmm.viterbi("THTHHHTHTTH")
>>> print prob
2.86654464e-06
>>> prob, _ = hmm.viterbi("THTHHHTHTTH" * 100)
>>> print prob
0.0
>>> logprob, _ = hmm.viterbiL("THTHHHTHTTH" * 100)
>>> print logprob
-1824.4030071946879
```

Repeat string 100 times

log-space Viterbi

When multiplying many numbers in (0, 1], we quickly approach the smallest number representable in a machine word. Past that we have *underflow* and processor rounds down to 0.

Switch to log space. Multiplies become adds.

Python example: http://nbviewer.ipython.org/7460513

Hidden Markov Model: Implementation

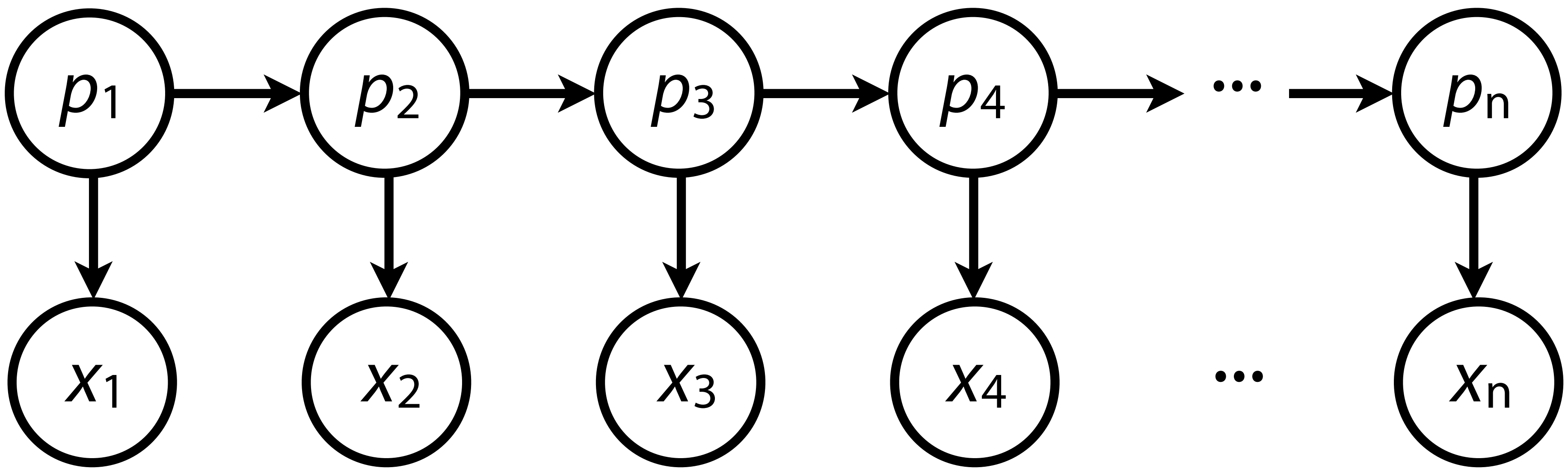
log-space version

```
def viterbiL(self, x):
    ''' Given sequence of emissions, return the most probable path
        along with log2 of its joint probability. '''
    x = map(self.smap.get, x) # turn emission characters into ids
    nrow, ncol = len(self.Q), len(x)
    mat   = numpy.zeros(shape=(nrow, ncol), dtype=float) # prob
    matTb = numpy.zeros(shape=(nrow, ncol), dtype=int)   # backtrace
    # Fill in first column
    for i in xrange(0, nrow):
        mat[i, 0] = self.Elog[i, x[0]] + self.Ilog[i]   *
    # Fill in rest of log prob and Tb tables
    for j in xrange(1, ncol):
        for i in xrange(0, nrow):
            ep = self.Elog[i, x[j]]                          *
            mx, mxi = mat[0, j-1] + self.Alog[0, i] + ep, 0    *
            for i2 in xrange(1, nrow):
                pr = mat[i2, j-1] + self.Alog[i2, i] + ep    *
                if pr > mx:
                    mx, mxi = pr, i2
            mat[i, j], matTb[i, j] = mx, mxi
    # Find final state with maximal log probability
    omx, omxi = mat[0, ncol-1], 0
    for i in xrange(1, nrow):
        if mat[i, ncol-1] > omx:
            omx, omxi = mat[i, ncol-1], i
    # Backtrace
    i, p = omxi, [omxi]
    for j in xrange(ncol-1, 0, -1):
        i = matTb[i, j]
        p.append(i)
    p = ''.join(map(lambda x: self.Q[x], p[::-1]))
    return omx, p # Return log probability and path
```

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

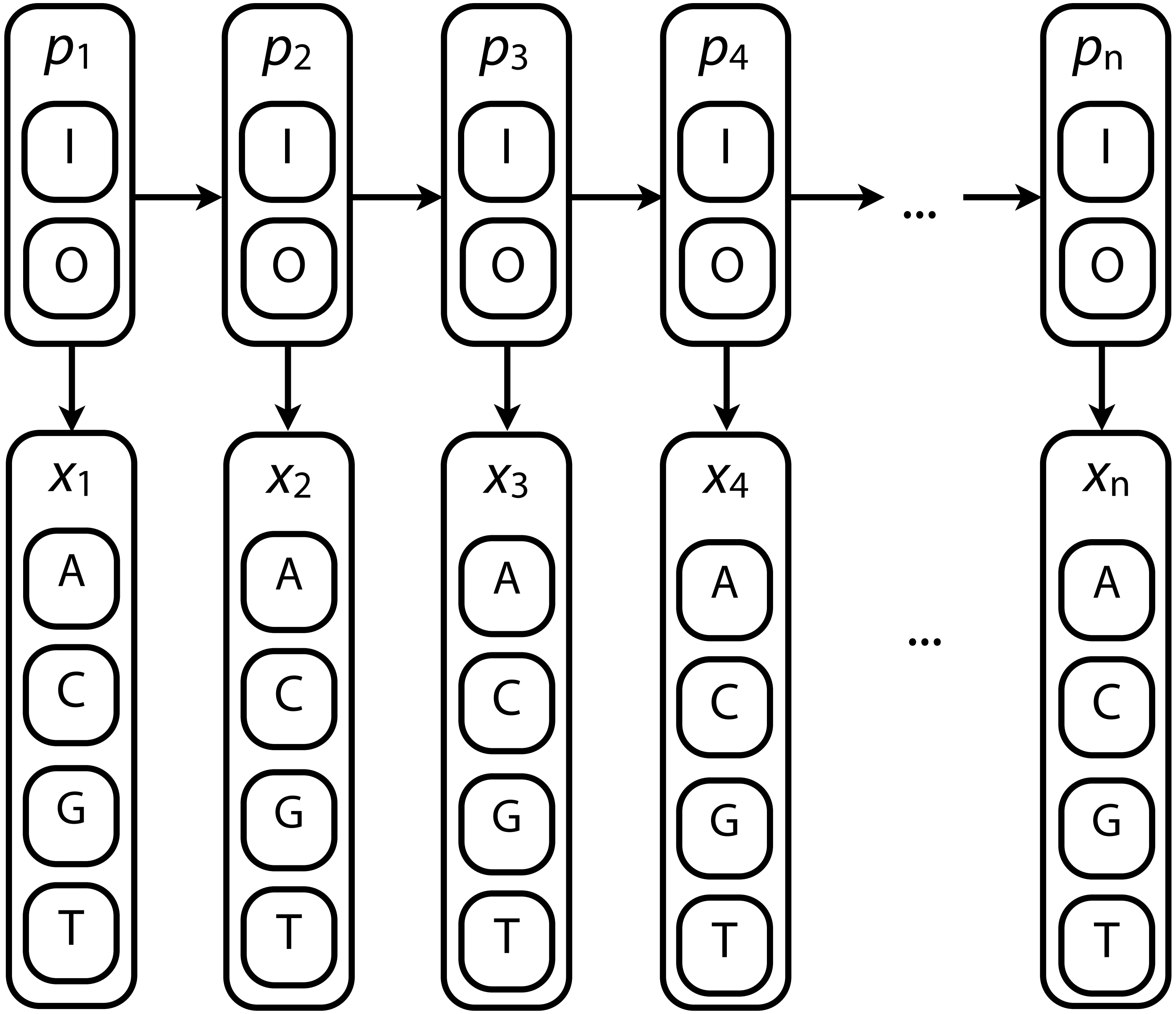
Hidden Markov Model

Task: design an HMM for finding CpG islands?



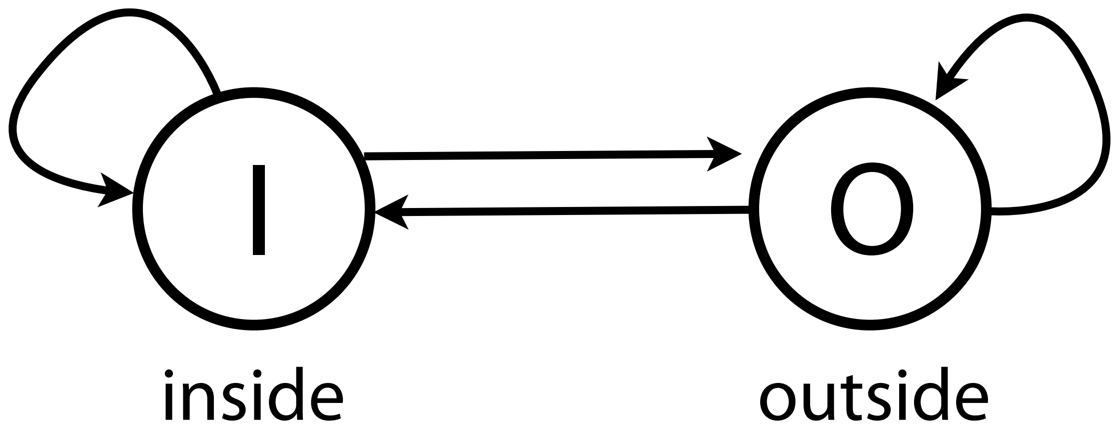
Hidden Markov Model

Idea 1: Q = { inside, outside }, Σ = { A, C, G, T }



Hidden Markov Model

Idea 1: $Q = \{$ inside, outside $\}$, $\Sigma = \{$ A, C, G, T $\}$



A	I	O
I		
O		

Transition matrix

Estimate as fraction of positions where we transition from inside to outside

E	A	C	G	T
I				
O				

Emission matrix

Estimate as fraction of nucleotides inside islands that are C

Hidden Markov Model

Example 1 using HMM idea 1:

A	I	O
I	0.8	0.2
O	0.2	0.8

E	A	C	G	T
I	0.1	0.4	0.4	0.1
O	0.25	0.25	0.25	0.25

x: ATATATACGCGCGCGCGCGCGATATATATATATA

p: OOOOOOOIIIIIIIIIIIIIIOOOOOOOOOOOOO

(from Viterbi)

Python example: http://nbviewer.ipython.org/7460513

Hidden Markov Model

Example 2 using HMM idea 1:

A	I	O
I	0.8	0.2
O	0.2	0.8

E	A	C	G	T
I	0.1	0.4	0.4	0.1
O	0.25	0.25	0.25	0.25

x: ATATCGCGCGCGATATATCGCGCGCGATATATAT

p: 0000IIIIIIII000000IIIIIIII00000000

(from Viterbi)

Python example: http://nbviewer.ipython.org/7460513

Hidden Markov Model

Example 3 using HMM idea 1:

A	I	O
I	0.8	0.2
O	0.2	0.8

E	A	C	G	T
I	0.1	0.4	0.4	0.1
O	0.25	0.25	0.25	0.25

x: ATATATACCCCCCCCCCCCCCATATATATATATA

p: 0000000IIIIIIIIIIIIII0000000000000

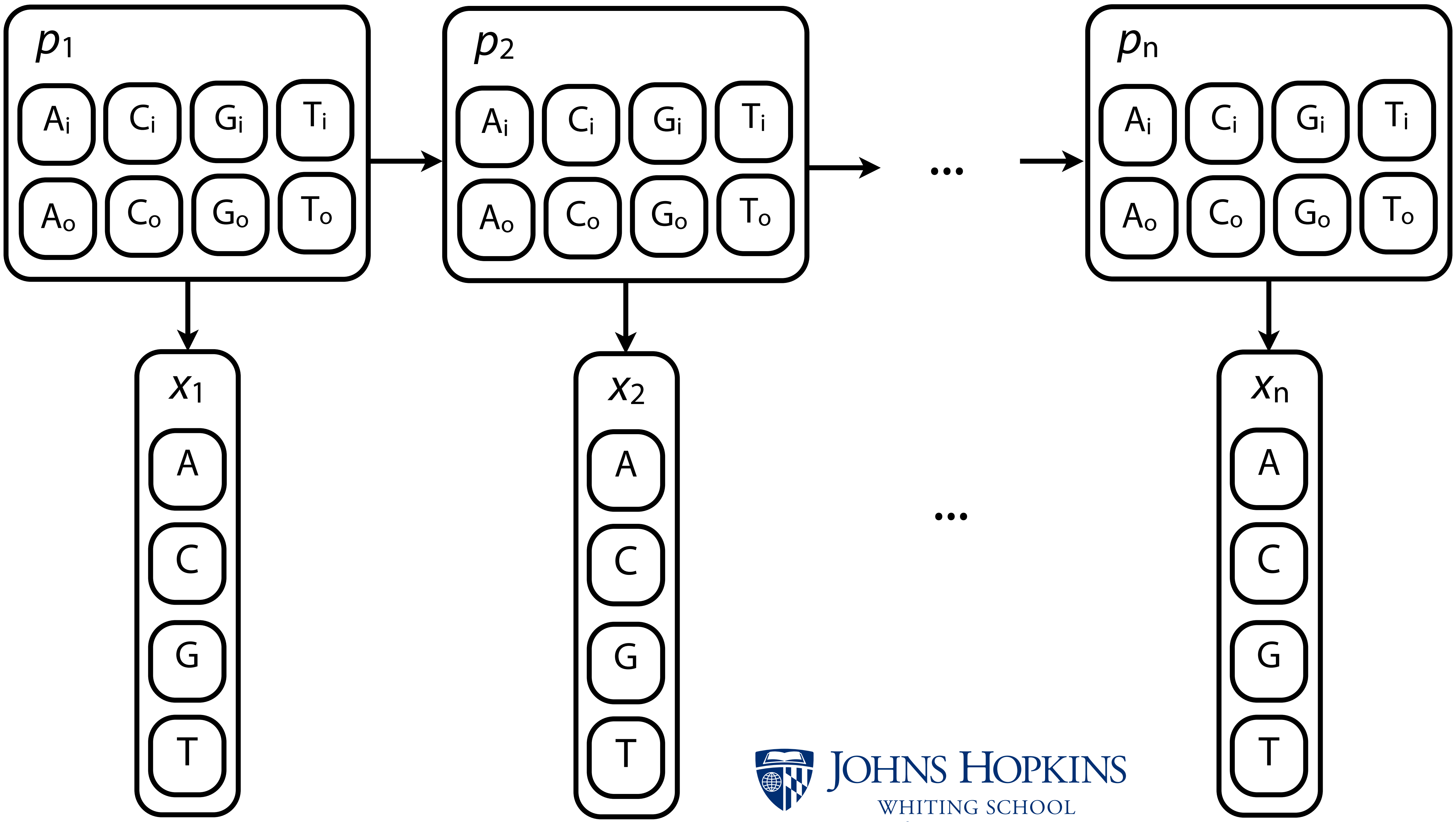
(from Viterbi)

Oops - not a CpG island!

Python example: http://nbviewer.ipython.org/7460513

Hidden Markov Model

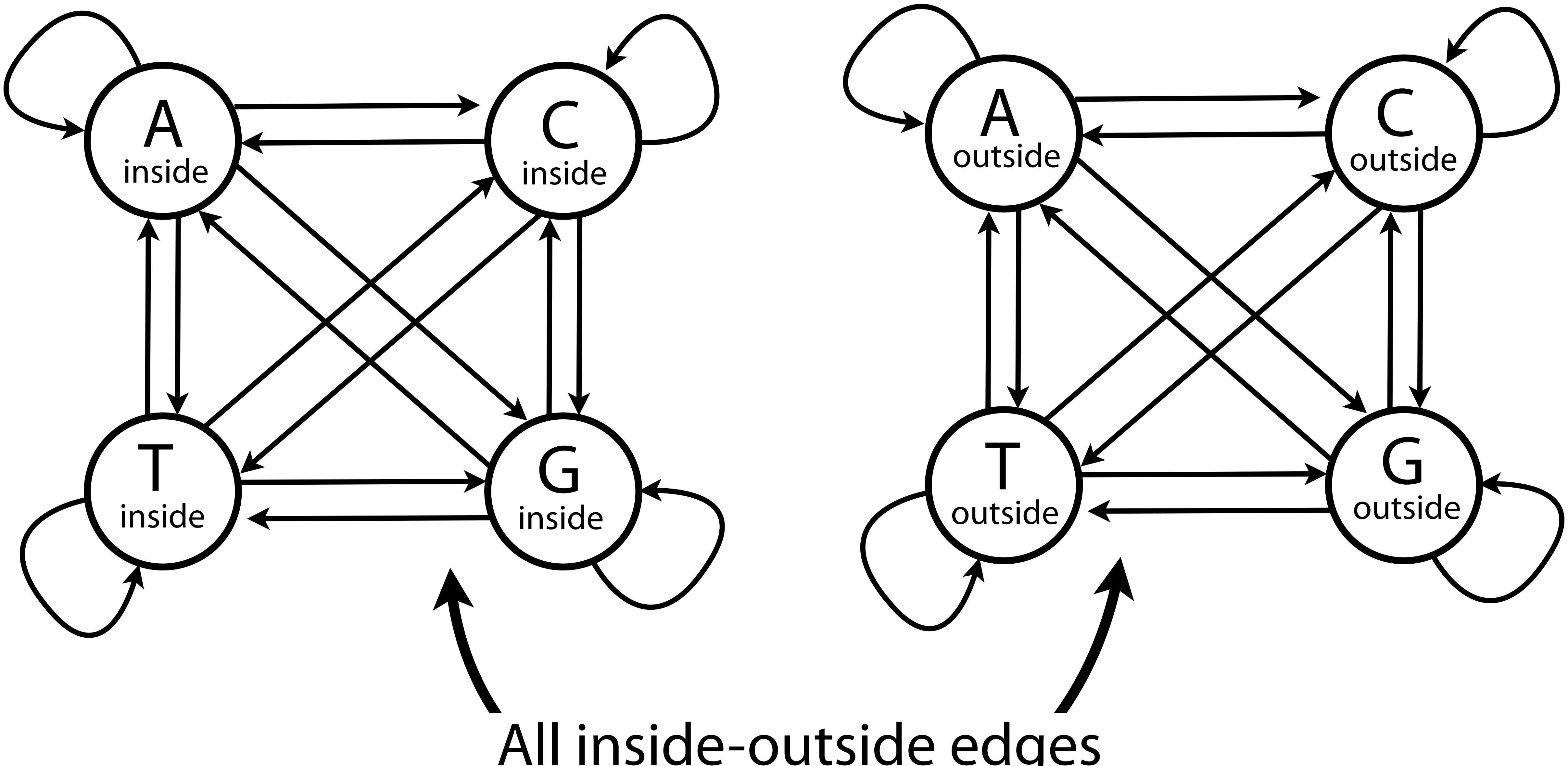
Idea 2: $Q = \{ A_i, C_i, G_i, T_i, A_o, C_o, G_o, T_o \}$, $\Sigma = \{ A, C, G, T \}$



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Hidden Markov Model

Idea 2: $Q = \{ A_i, C_i, G_i, T_i, A_o, C_o, G_o, T_o \}$, $\Sigma = \{ A, C, G, T \}$



Hidden Markov Model

Idea 2: $Q = \{ A_i, C_i, G_i, T_i, A_o, C_o, G_o, T_o \}$, $\Sigma = \{ A, C, G, T \}$

A	A_i	C_i	G_i	T_i	A_o	C_o	G_o	T_o
A_i								
C_i								
G_i								
T_i								
A_o								
C_o								
G_o								
T_o								

Estimate $P(C_i \mid T_i)$ as fraction of all dinucleotides where first is an inside T, second is an inside C

E	A	C	G	T
A_i	1	0	0	0
C_i	0	1	0	0
G_i	0	0	1	0
T_i	0	0	0	1
A_o	1	0	0	0
C_o	0	1	0	0
G_o	0	0	1	0
T_o	0	0	0	1

Hidden Markov Model

Actual trained transition matrix A:

```
A:
[[  1.85152516e-01   2.75974026e-01   4.00289017e-01   1.37026750e-01
    3.19045117e-04   3.19045117e-04   6.38090233e-04   2.81510397e-04]
 [  1.89303979e-01   3.58523577e-01   2.52868527e-01   1.97836007e-01
    4.28792308e-04   5.72766368e-04   3.75584503e-05   4.28792308e-04]
 [  1.72369088e-01   3.29501650e-01   3.55446538e-01   1.40829292e-01
    3.39848138e-04   4.94038497e-04   7.64658311e-04   2.54886104e-04]
 [  9.38783432e-02   3.40823149e-01   3.75970400e-01   1.86949063e-01
    2.56686367e-04   5.57197235e-04   1.05804868e-03   5.07112091e-04]
 [  0.00000000e+00   3.78291020e-05   0.00000000e+00   0.00000000e+00
    2.94813496e-01   1.94641138e-01   2.86962055e-01   2.23545482e-01]
 [  0.00000000e+00   7.57154865e-05   0.00000000e+00   0.00000000e+00
    3.26811872e-01   2.94079570e-01   6.17258712e-02   3.17306971e-01]
 [  0.00000000e+00   5.73810399e-05   0.00000000e+00   0.00000000e+00
    2.57133507e-01   2.33483327e-01   2.94234944e-01   2.15090841e-01]
 [  0.00000000e+00   3.11417347e-05   0.00000000e+00   0.00000000e+00
    1.79565378e-01   2.32469115e-01   2.94623408e-01   2.93310958e-01]]
```

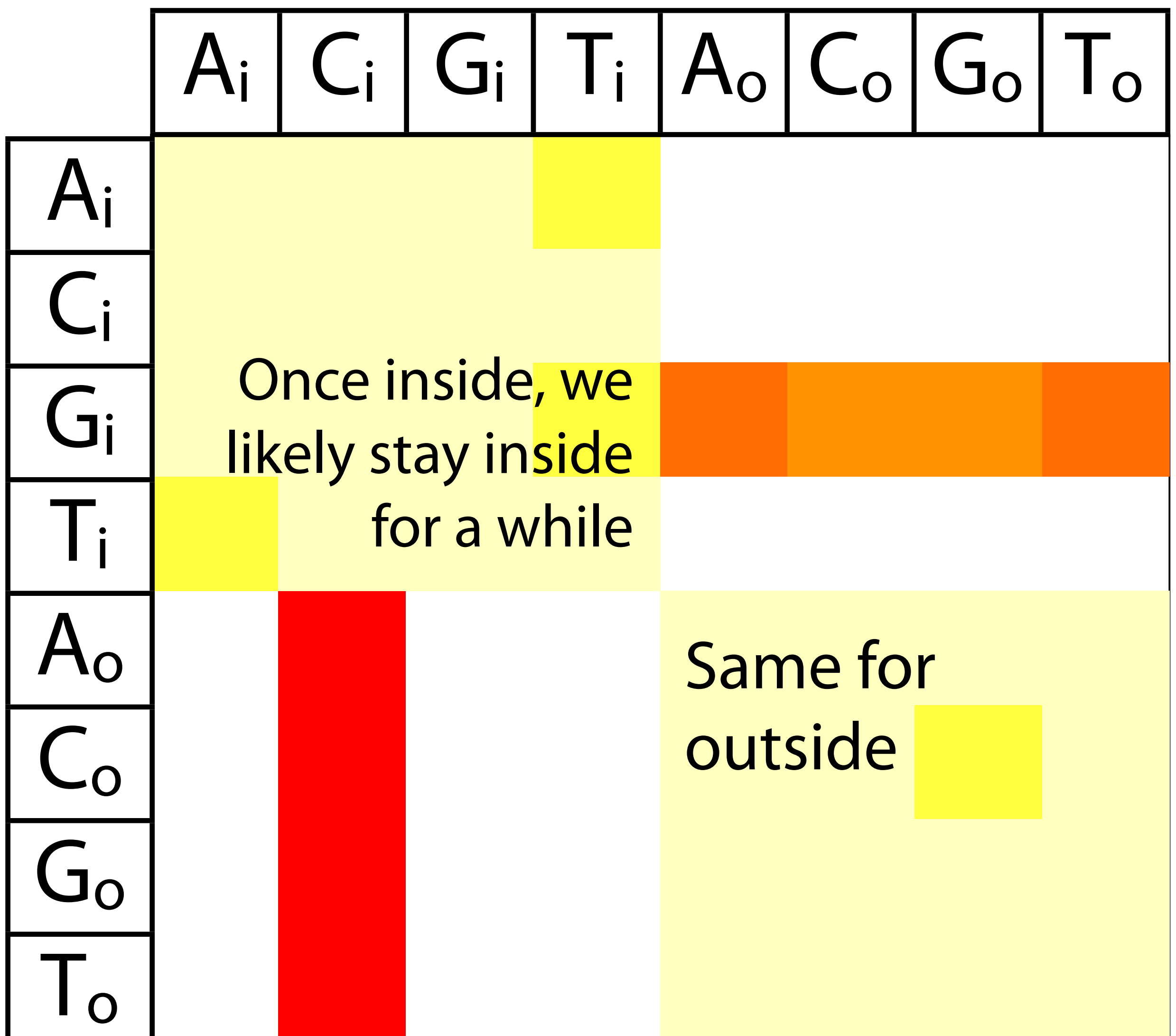
Hidden Markov Model

Actual trained transition matrix A:

Red & orange: low probability

Yellow: high probability

White: probability = 0



← CpG islands end with G (in fact, they end with CG)

↖ CpG islands start with C (in fact, they start with CG)

Hidden Markov Model

Viterbi result; lowercase = *outside*, uppercase = *inside*:

```
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCG
CGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCG
CGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
```

Hidden Markov Model

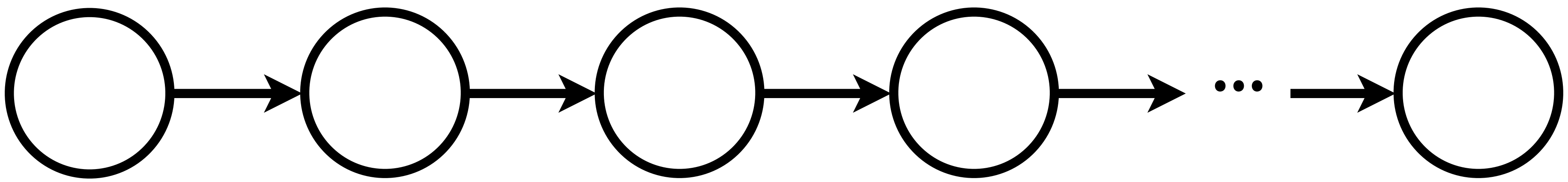
Viterbi result; lowercase = *outside*, uppercase = *inside*:

```
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatgggggggggggggggggggggggggggggggggggggggggggggggggggggggg
gggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggg
ggggggggggggggggggggggggggggggggggggggggggggatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
atatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatatat
```

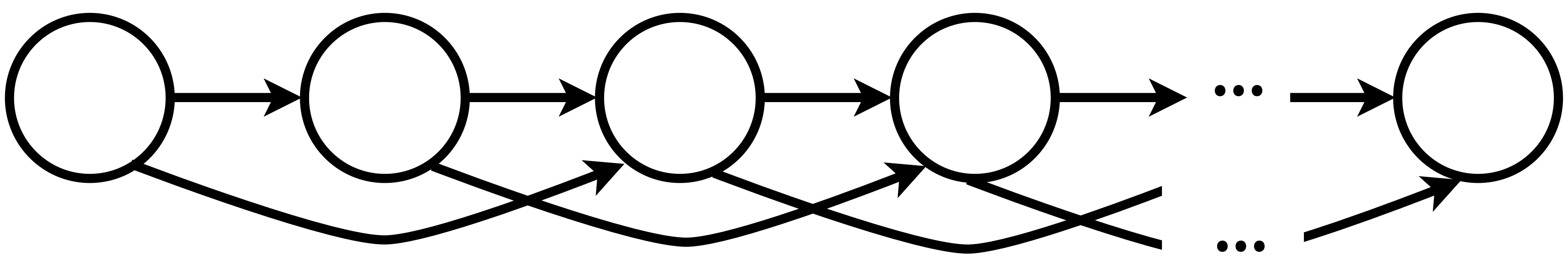
Hidden Markov Model

Many of the Markov chains and HMMs we've discussed are *first order,* but we can also design models of higher orders

First-order Markov chain:

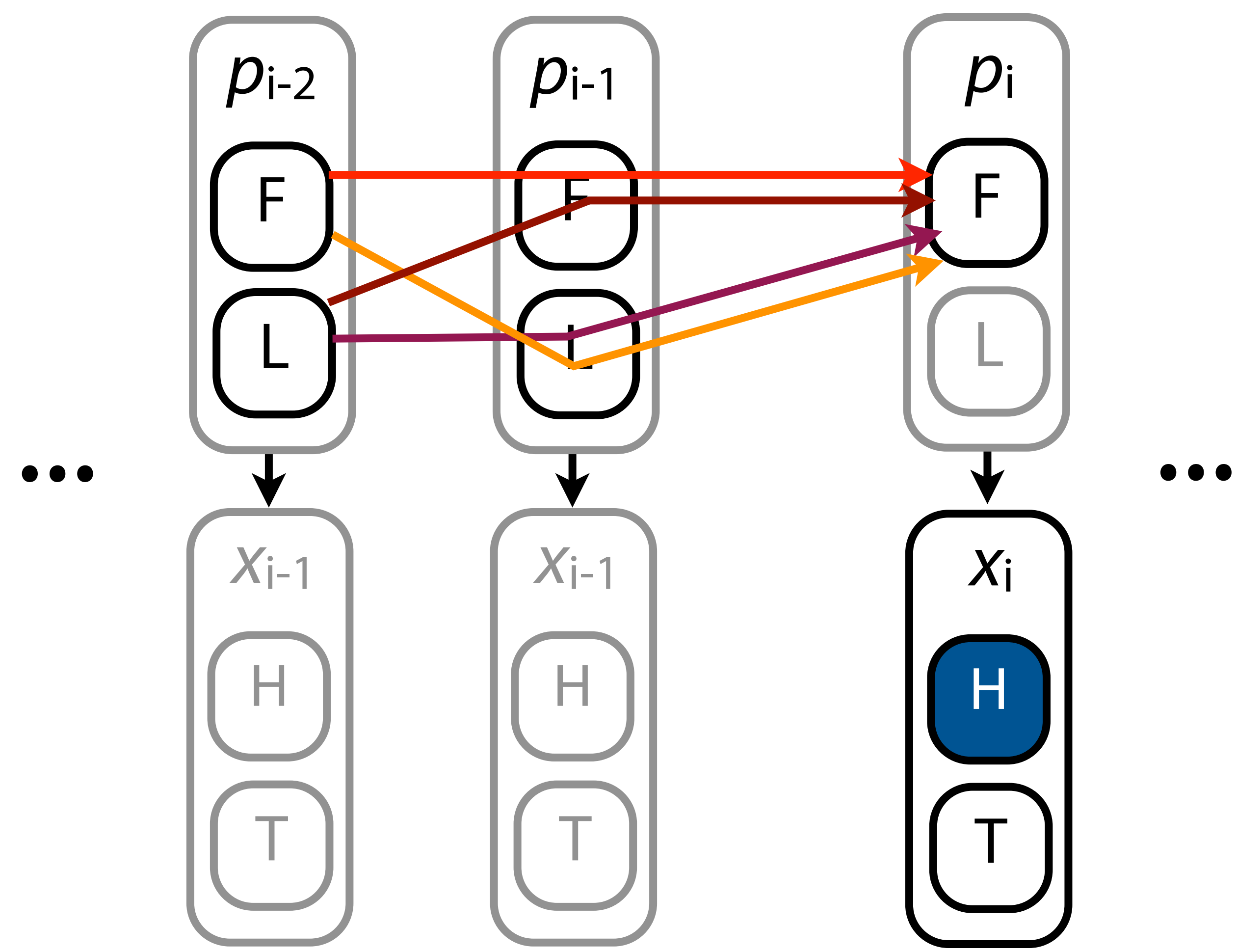


Second-order Markov chain:



Hidden Markov Model

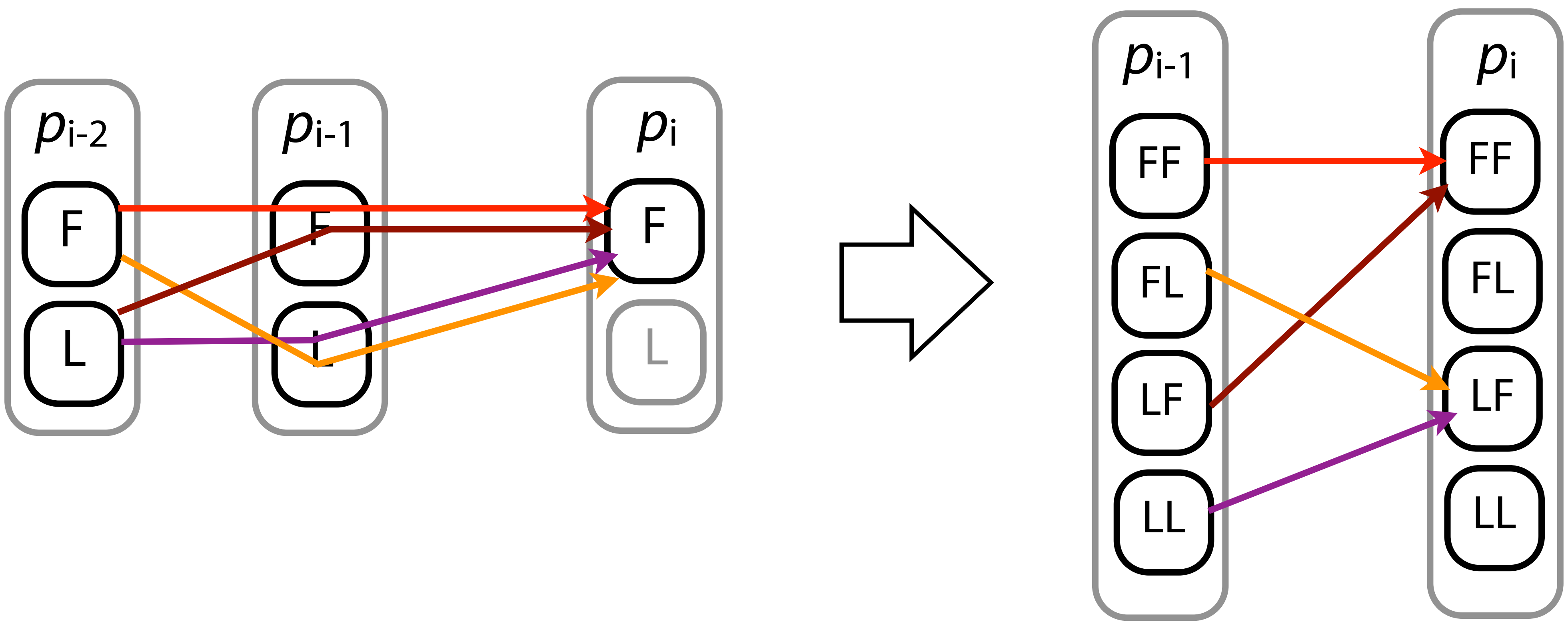
For higher-order HMMs, Viterbi $s_{k,i}$ no longer depends on just the previous state assignment



Can sidestep the issue by expanding the state space...

Hidden Markov Model

Now *one* state encodes the last *two* "loadedness"es of the coin



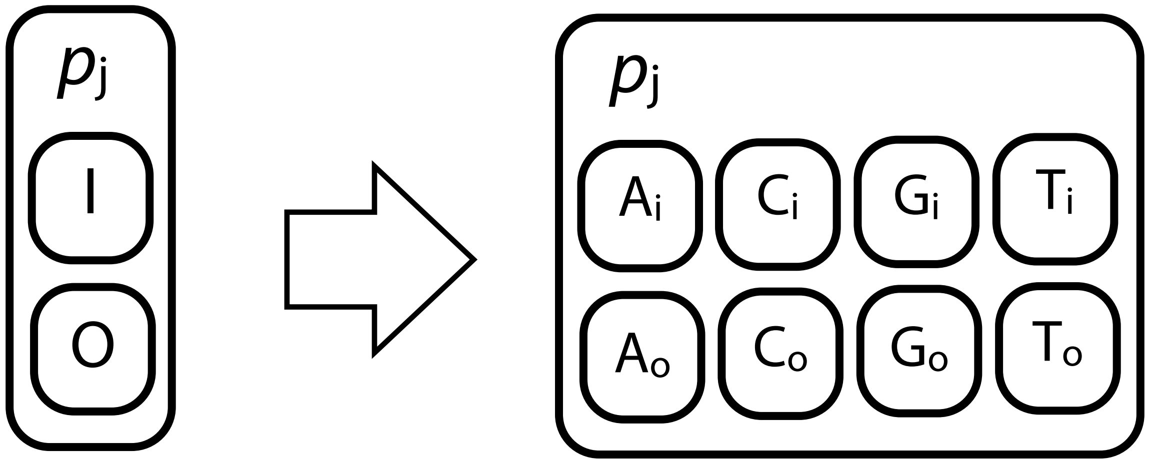
$Q = \{ \mathrm{F}, \mathrm{L} \}$

$Q = \{ \mathrm{F}, \mathrm{L} \} \times \{ \mathrm{F}, \mathrm{L} \}$

After expanding, usual Viterbi works fine.

Hidden Markov Model

We also expanded the state space here:



$Q = \{ \mathrm{I}, \mathrm{O} \}$ $\qquad$ $Q = \{ \mathrm{I}, \mathrm{O} \} \times \{ \mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T} \}$