String Algorithms and Data Structures
Hidden Markov Models

CS 199-225

December 5, 2022

Brad Solomon

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Learning Objectives

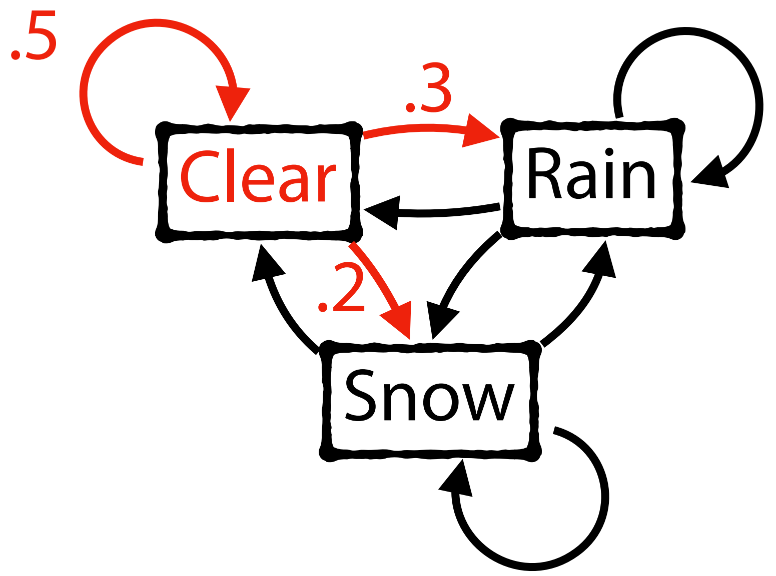
Review Markov Chains

Introduce Hidden Markov Models

Introduce the Viterbi algorithm for finite discrete HMMs

Markov Chain

A **finite Markov Chain** has a set of states S and a finite matrix M



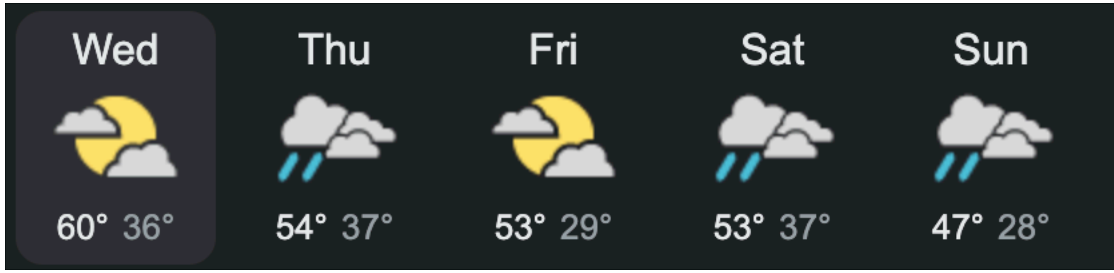
$$S = \{Clear, Rain, Snow\}$$

$$M = \begin{pmatrix} .5 & .3 & .2 \\ .5 & .4 & .1 \\ .2 & .1 & .7 \end{pmatrix}$$

Markov Assumption

Probability of state x_k depends only on previous state x_{k-1}

Ex: Let $x = \{C, R, C, R, R\}$



$$P(x) = P(x_k, x_{k-1}, \ldots x_1)$$

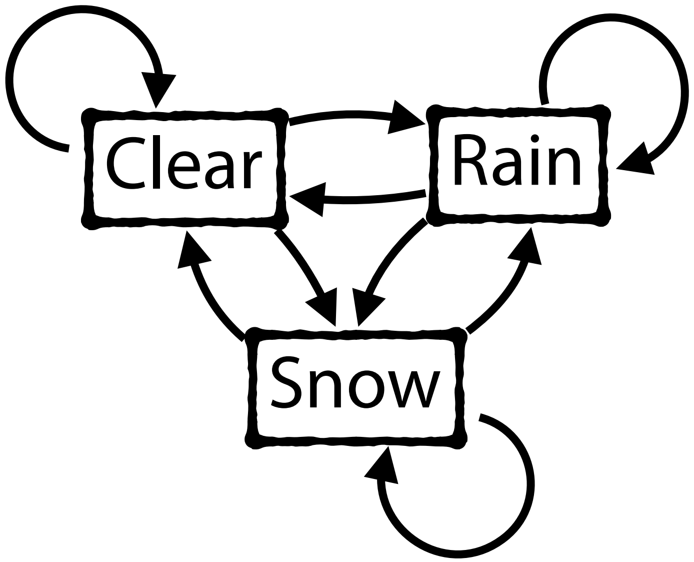
$$= P(x_k \mid x_{k-1}, \ldots x_1) P(x_{k-1}, \ldots x_1)$$

$$= P(x_k \mid x_{k-1}, \ldots x_1) P(x_{k-1} \mid x_{k-2}, \ldots x_1) \ldots P(x_2 \mid x_1) P(x_1)$$

$$P(x) \approx P(x_k \mid x_{k-1}) P(x_{k-1} \mid x_{k-2}) \ldots P(x_2 \mid x_1) P(x_1)$$

Markov Chain

Given a Markov Chain and an initial state, all subsequent states can be represented either as **a series of random states** or a transition probability.



$$M = \begin{pmatrix} .5 & .3 & .2 \\ .5 & .4 & .1 \\ .2 & .1 & .7 \end{pmatrix}$$

$X_0 = \text{Clear}$

$X_1 = \text{Clear}$

$X_2 = \text{Snow}$

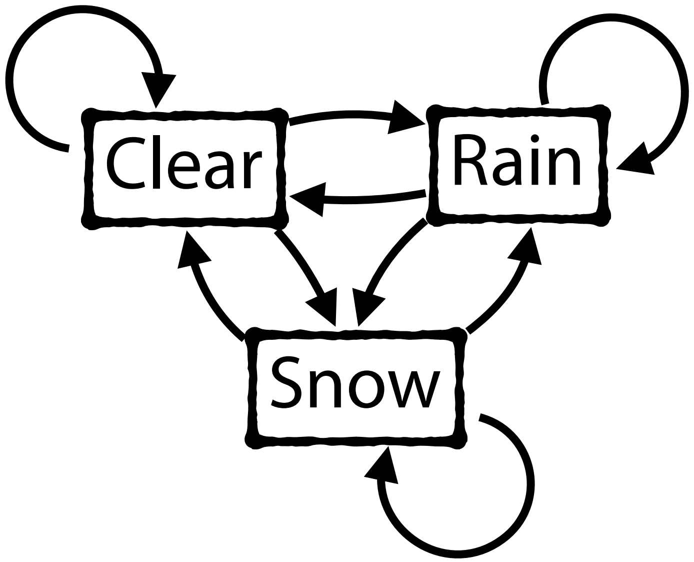
$X_3 = \text{Snow}$

$X_4 = \text{Snow}$

$X_5 = \text{Rain}$

Markov Chain

Given a Markov Chain and an initial state, all subsequent states can be represented either as a series of random states or a **transition probability.**



$$M = \begin{pmatrix} .5 & .3 & .2 \\ .5 & .4 & .1 \\ .2 & .1 & .7 \end{pmatrix}$$

$$P_0 = (.4 \quad .3 \quad .3)$$

$$P_1 = (.41 \quad .27 \quad .32)$$

$$P_2 = (.404 \quad .263 \quad .333)$$

$$P_3 = (.401 \quad .259 \quad .340)$$

```
>>> cpg_conds, _ = markov_chain_from_dinucs(samp_cpg)
>>> print(cpg_conds)
[[ 0.19152248,  0.27252589,  0.39998803,  0.1359636 ],
 [ 0.18921984,  0.35832388,  0.25467081,  0.19778547],
 [ 0.17322219,  0.33142737,  0.35571338,  0.13963706],
 [ 0.09509721,  0.33836493,  0.37567927,  0.19085859]]
>>> default_conds, _ = markov_chain_from_dinucs(samp_def)
>>> print(default_conds)
[[ 0.33804066,  0.17971034,  0.23104207,  0.25120694],
 [ 0.37777025,  0.25612117,  0.03987225,  0.32623633],
 [ 0.30257815,  0.20326794,  0.24910719,  0.24504672],
 [ 0.21790184,  0.20942905,  0.2642385 ,  0.3084306 ]]
>>> print(np.log2(cpg_conds) - np.log2(def_conds))
[[-0.87536356,  0.59419041,  0.81181564, -0.85527103],
 [-0.98532149,  0.49570561,  2.64256972, -0.7126391 ],
 [-0.79486196,  0.68874785,  0.51821792, -0.79549511],
 [-1.22085697,  0.73036913,  0.48119354, -0.69736839]]
```

CpG: rows A, C, G, T

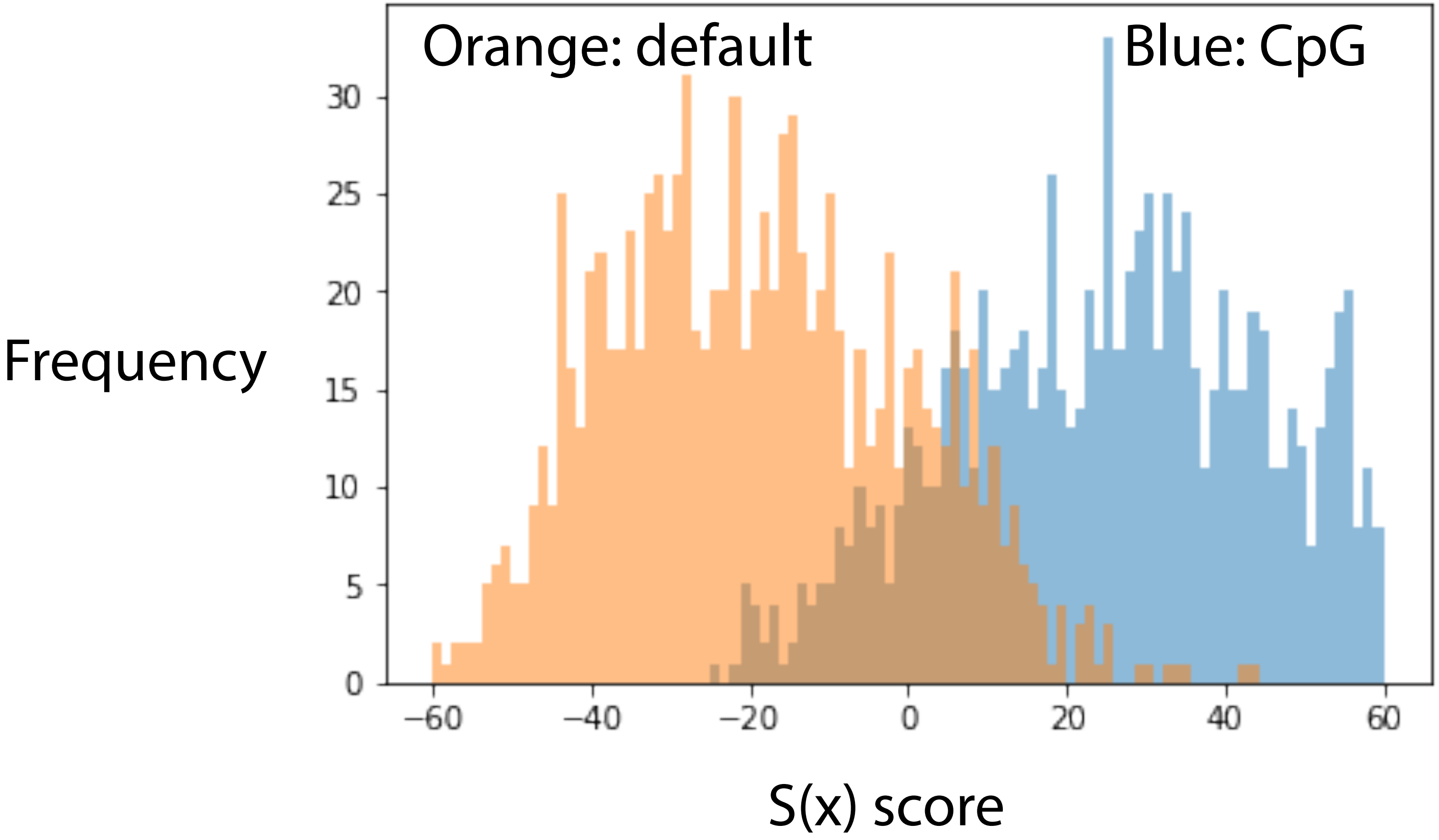
Default: rows A, C, G, T

Log ratio: rows A, C, G, T

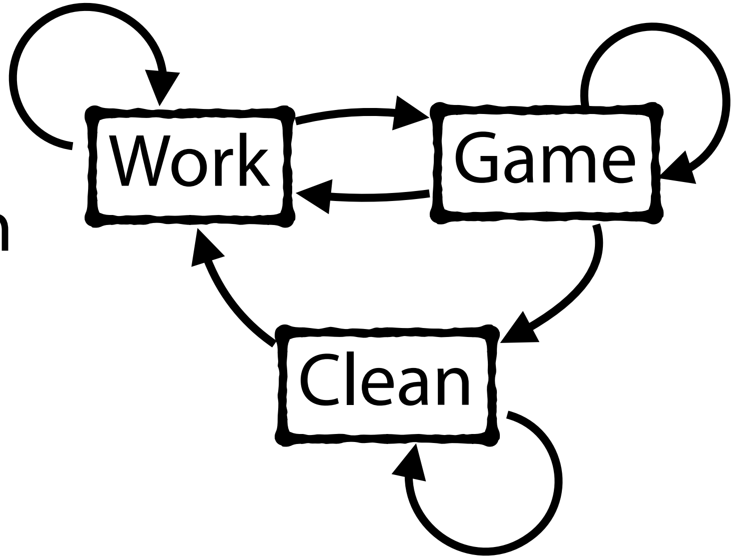
Columns: A C G T

Markov Chain in Sequencing

Drew 1,000 100-mers from inside CpG islands and another 1,000 from outside, and calculated S(x) for all



Markov Chain Matrix



If I'm working at time 0, what is probability that I'm working at time t?

Claim: $Pr(X_t = v \mid X_0 = u) = M^t[u,\ v]$

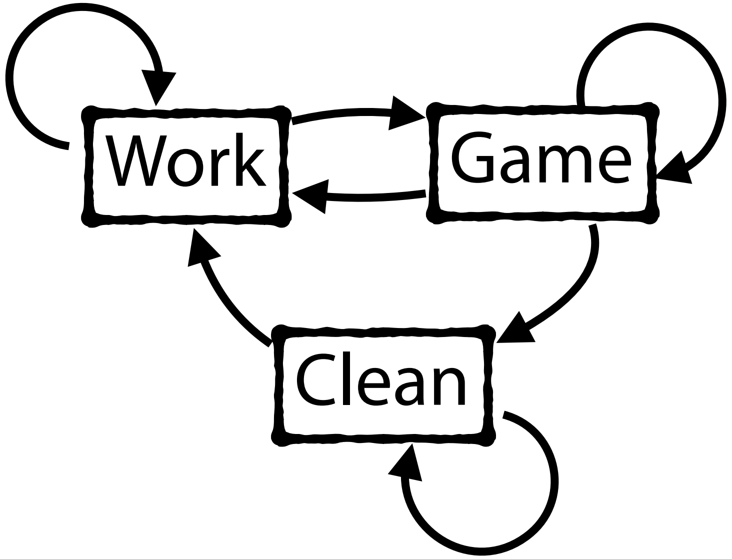
Base Case:

$$M = \begin{pmatrix} .4 & .6 & 0 \\ .1 & .6 & .3 \\ .5 & 0 & .5 \end{pmatrix}$$

T=1:

T=2:

Markov Chain Matrix



Claim: $Pr(X_t = v \mid X_0 = u) = M^t[u, v]$

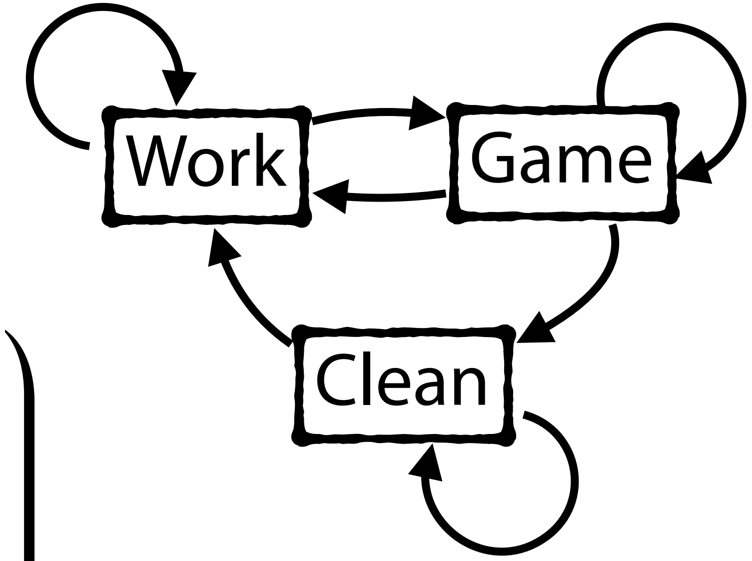
Induction:

Assume $Pr(X_{t-1} = v \mid X_0 = u) = M^{t-1}[u, v]$.

Show holds for $Pr(X_t = w \mid X_0 = u) = M^t[u, w]$

$$M = \begin{pmatrix} .4 & .6 & 0 \\ .1 & .6 & .3 \\ .5 & 0 & .5 \end{pmatrix}$$

Markov Chain Matrix



What happens as $t \to \infty$?

$$M = \begin{pmatrix} .4 & .6 & 0 \\ .1 & .6 & .3 \\ .5 & 0 & .5 \end{pmatrix} \quad M^3 = \begin{pmatrix} .238 & .492 & .270 \\ .307 & .402 & .291 \\ .335 & .450 & .215 \end{pmatrix}$$

$$M^{10} = \begin{pmatrix} .2940 & .4413 & .2648 \\ .2942 & .4411 & .2648 \\ .2942 & .4413 & .2648 \end{pmatrix}$$

$$M^{60} = \begin{pmatrix} .2941 & .4412 & .2647 \\ .2941 & .4412 & .2647 \\ .2941 & .4412 & .2647 \end{pmatrix}$$

Markov Chain Stationary Distribution

A probability vector π is called a **stationary distribution** for a Markov Chain if it satisfies the stationary equation: $\pi = \pi M$

$$M = \begin{pmatrix} .4 & .6 & 0 \\ .1 & .6 & .3 \\ .5 & 0 & .5 \end{pmatrix}$$

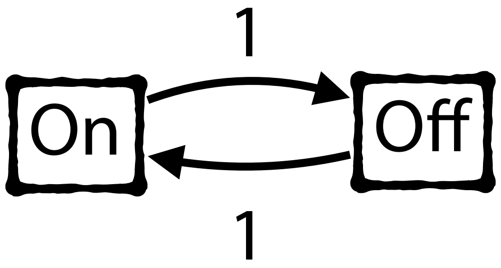
$$\pi[W] = .4\pi[W] + .1\pi[G] + .5\pi[C]$$

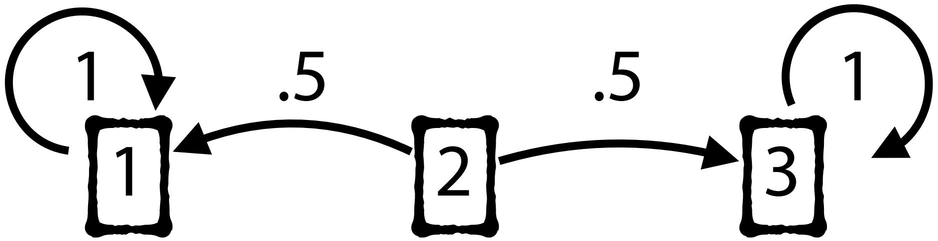
$$\pi[S] = .6\pi[W] + .6\pi[G] + 0\pi[C]$$

$$\pi[E] = 0\pi[W] + .3\pi[G] + .5\pi[C]$$

Markov Chain Stationary Distribution

Stationary distributions can be calculated using the system of equation (and that all probabilities sum to 1). **But not every Markov Chain has a steady state (and some have infinitely many)!**





Markov Chain Monte Carlo

There are ways to prove whether a Markov Chain has a stationary distribution, but several algorithms exist that approximate!

Gibbs Sampling:

Randomly assign values to a probability vector $\pi = (\theta_1, \theta_2, \ldots, \theta_d)$.

For each i, $1 \leq i \leq d$:

Update value θ_i based on

$(\theta_1, \ldots, \theta_{i-1})^{t+1}, (\theta_{i+1}, \ldots, \theta_d)^t$

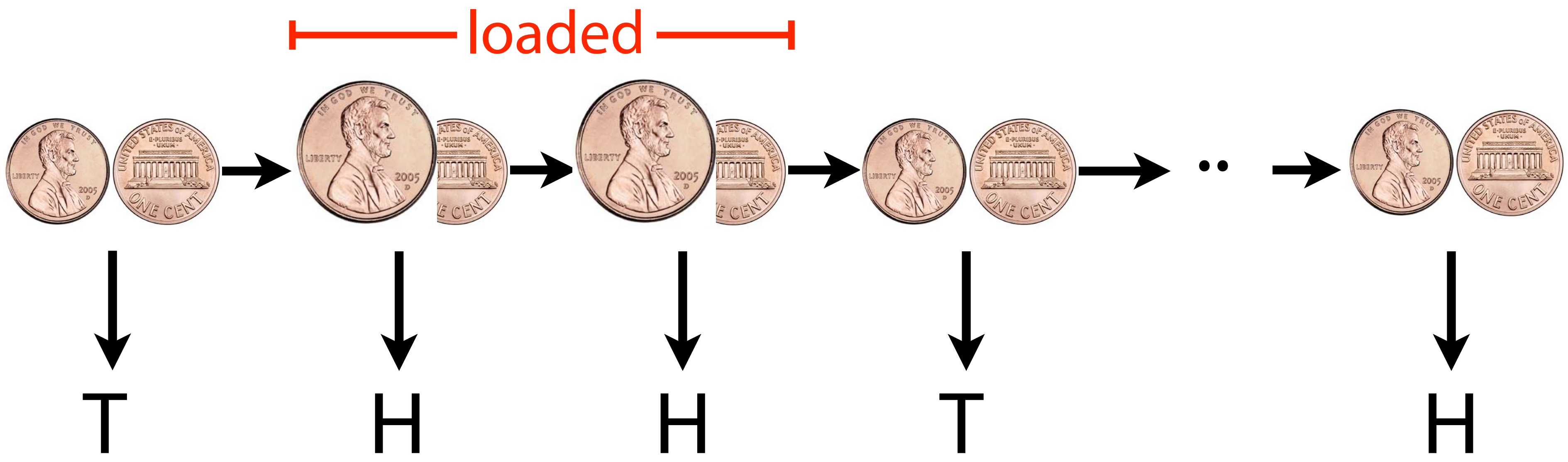
Repeat for different starting i

Hidden Markov Models

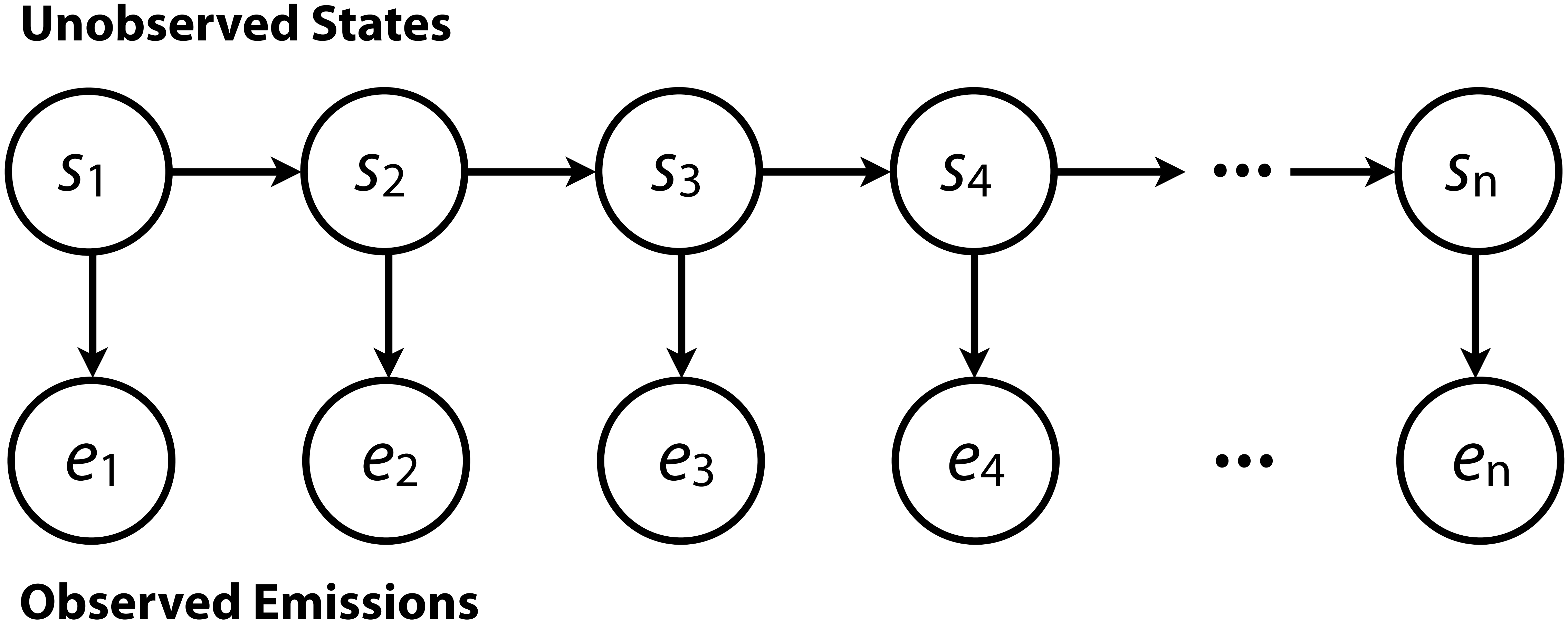
In the real world, we often don't know the underlying markov chain!

Instead, we have observations that can be used to predict our current state.

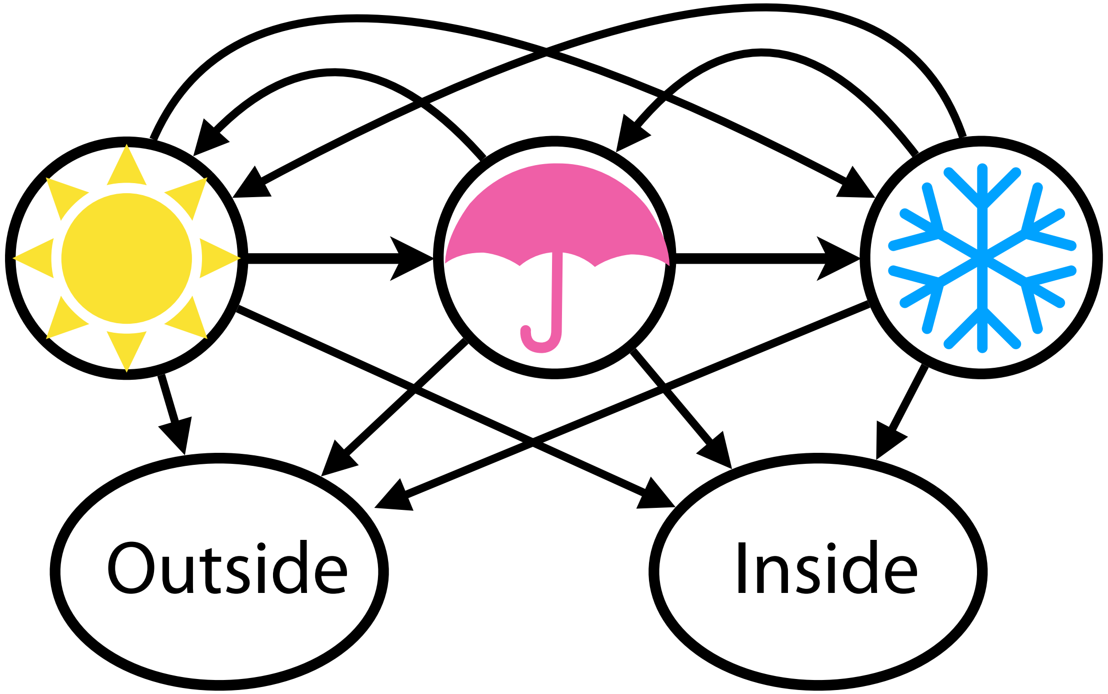
Ex: Repeated coin flips but *sometimes* I cheat and use a fixed coin.



Hidden Markov Models



Hidden Markov Models

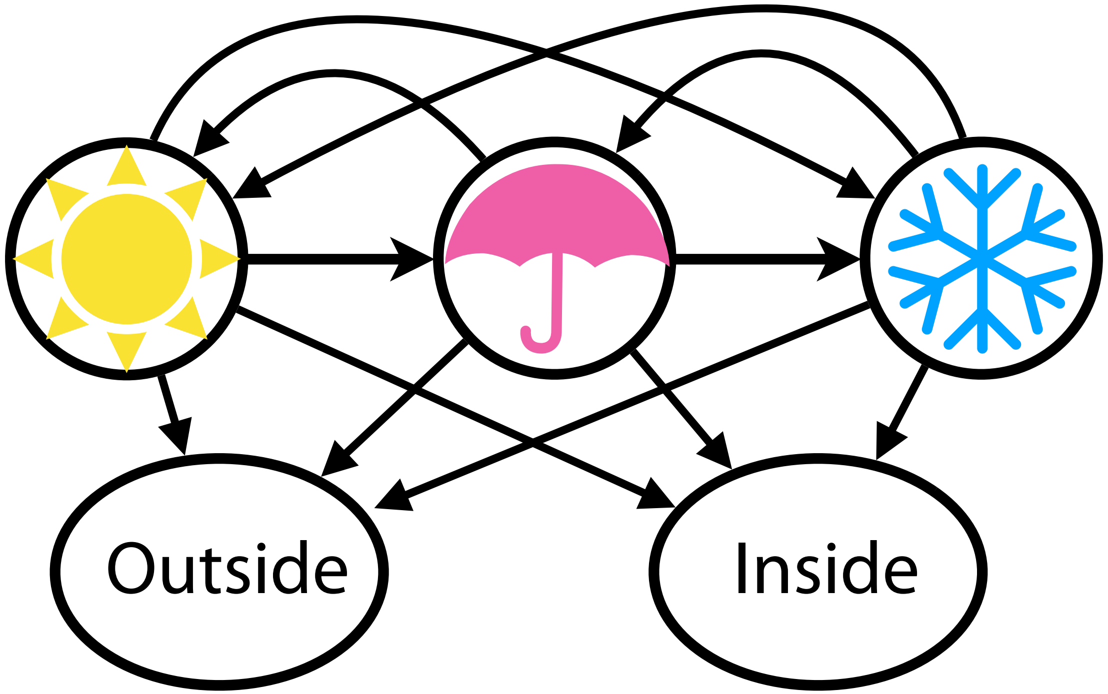


$$M = \begin{pmatrix} .5 & .3 & .2 \\ .5 & .4 & .1 \\ .2 & .1 & .7 \end{pmatrix} \quad E = \begin{pmatrix} .8 & .2 \\ .3 & .7 \\ .5 & .5 \end{pmatrix}$$

Pr({O, I, O} | {C, R, S})?

Pr({O, I, O}, {C, R, S} | P(T_0 = C) = 0.4)?

Hidden Markov Models



$$M = \begin{pmatrix} .5 & .3 & .2 \\ .5 & .4 & .1 \\ .2 & .1 & .7 \end{pmatrix} \quad E = \begin{pmatrix} .8 & .2 \\ .3 & .7 \\ .5 & .5 \end{pmatrix}$$

Pr({O, I, O})?

If I go outside for three days, what was the most likely weather?

Hidden Markov Models

If I go outside for three days, what was the most likely weather?

$$M = \begin{pmatrix} .5 & .3 & .2 \\ .5 & .4 & .1 \\ .2 & .1 & .7 \end{pmatrix} \qquad E = \begin{pmatrix} .8 & .2 \\ .3 & .7 \\ .5 & .5 \end{pmatrix}$$

Viterbi Algorithm

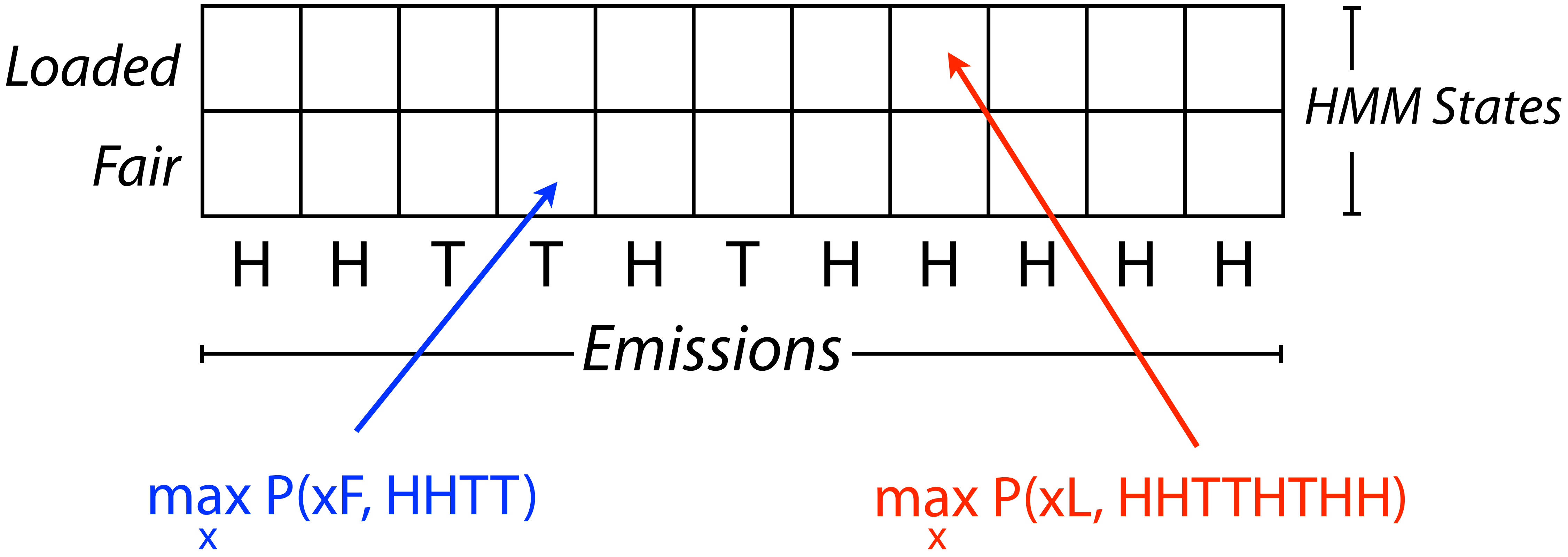
We can brute force all possible combinations…

… or we can use the Markov Assumption with Dynamic Programming

$$M = \begin{pmatrix} .6 & .4 \\ .4 & .6 \end{pmatrix} \qquad E = \begin{pmatrix} .5 & .5 \\ .8 & .2 \end{pmatrix}$$

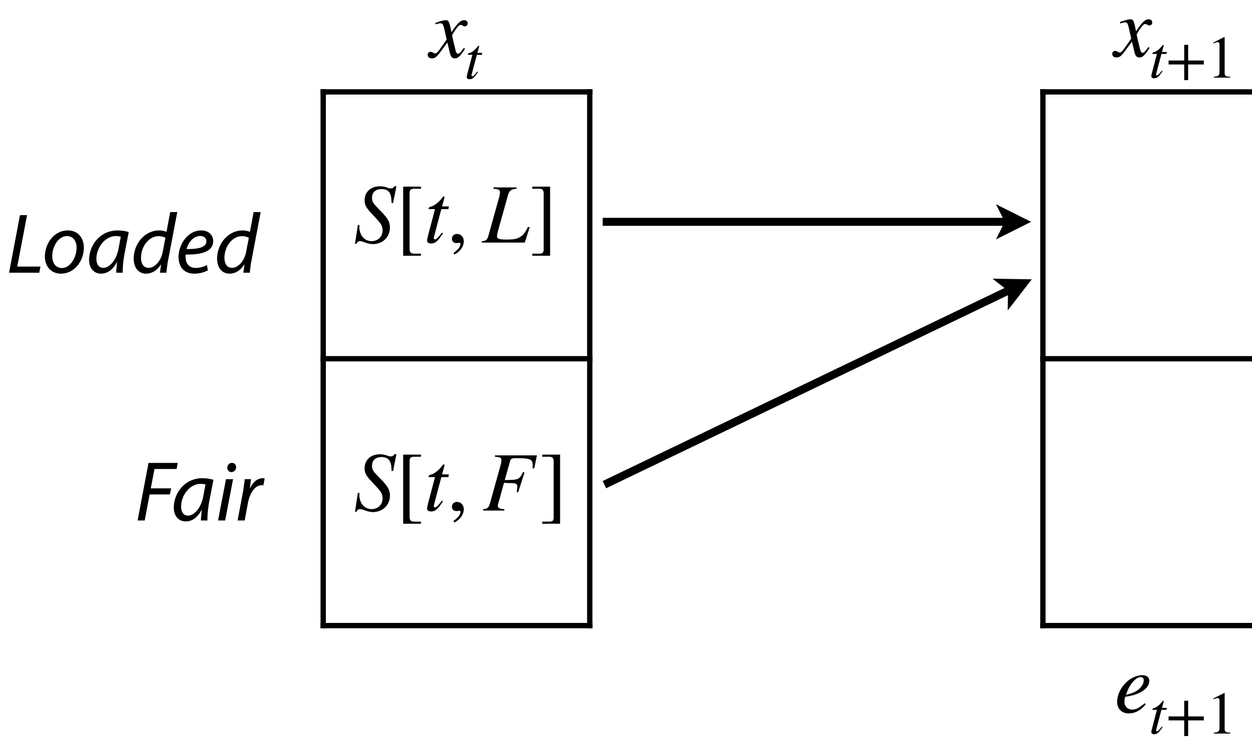
Example by Ben Langmead

Viterbi Algorithm



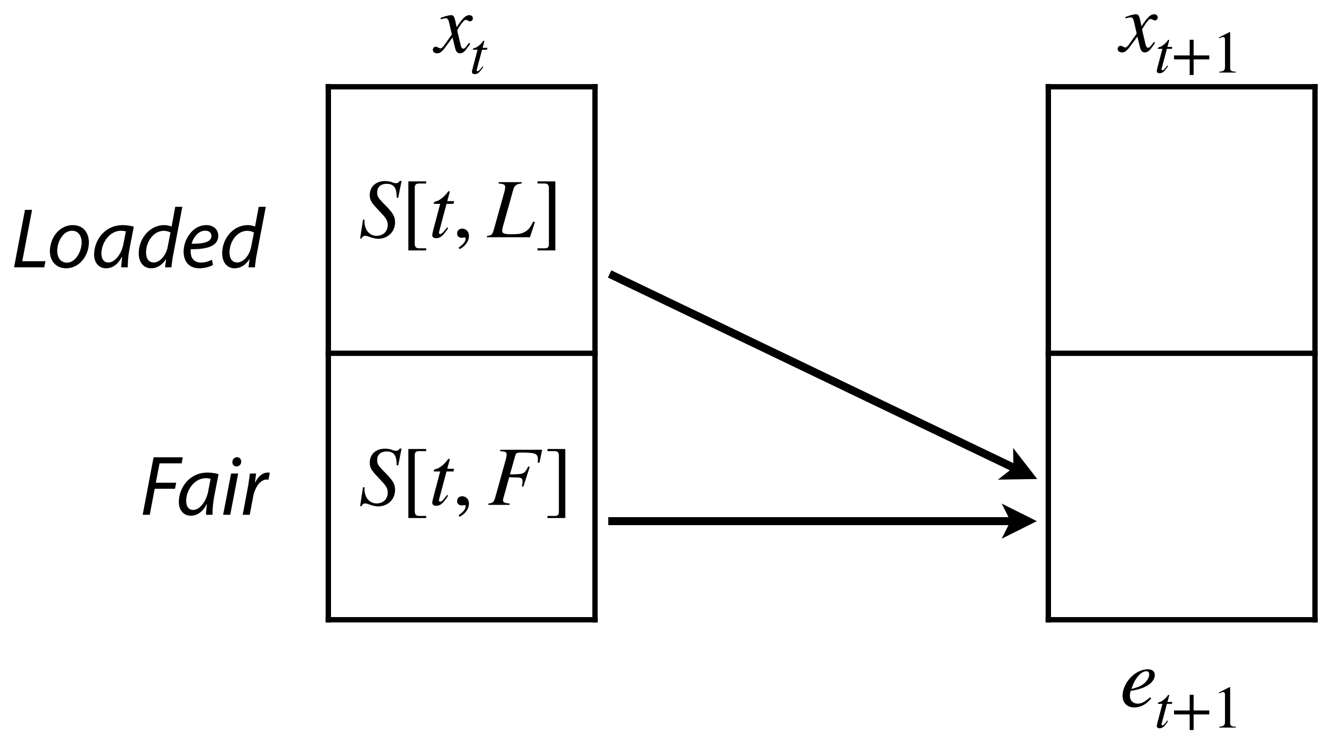
$S_{k,i}$ = *greatest joint probability* of observing the length-*i* prefix of *e* and any sequence of states ending in state *k*

Viterbi Algorithm



$$S[t+1,\ L] =$$

Viterbi Algorithm



$$S[t+1,\ F] =$$

Viterbi Algorithm

Loaded											
Fair											
	H	H	T	T	H	T	H	H	H	H	H

$$M = \begin{pmatrix} .6 & .4 \\ .4 & .6 \end{pmatrix}$$

$$E = \begin{pmatrix} .5 & .5 \\ .8 & .2 \end{pmatrix}$$

Assume we start with Fair/Loaded with equal probability

$$S[0, L] = 0.5 \cdot \boldsymbol{P(H \mid L)} \qquad S[0, F] = 0.5 \cdot \boldsymbol{P(H \mid F)}$$

$$= 0.5 \cdot \boldsymbol{0.8} \qquad\qquad = 0.5 \cdot \boldsymbol{0.5}$$

Viterbi Algorithm

$$M = \begin{pmatrix} .6 & .4 \\ .4 & .6 \end{pmatrix}$$

Loaded	0.4										
Fair	0.25										
	H	H	T	T	H	T	H	H	H	H	H

$$E = \begin{pmatrix} .5 & .5 \\ .8 & .2 \end{pmatrix}$$

$S[1, L] =$

Viterbi Algorithm

Loaded	0.4	0.19									
Fair	0.25										
	H	H	T	T	H	T	H	H	H	H	H

$$M = \begin{pmatrix} .6 & .4 \\ .4 & .6 \end{pmatrix}$$

$$E = \begin{pmatrix} .5 & .5 \\ .8 & .2 \end{pmatrix}$$

$S[1, F] =$

Viterbi Algorithm

Loaded	0.4	0.19									
Fair	0.25	0.08									
	H	H	T	T	H	T	H	H	H	H	H

$$M = \begin{pmatrix} .6 & .4 \\ .4 & .6 \end{pmatrix}$$

$$E = \begin{pmatrix} .5 & .5 \\ .8 & .2 \end{pmatrix}$$

Viterbi Algorithm

These get small — now log_2 scaled

-1.32	-2.38	-5.44	-8.35	-8.08	-11.1	-11.6	-12.6	-13.7	-14.7	-15.8
-2	-3.64	-4.7	-6.4	-8.2	-9.9	-11.7	-13.4	-14.9	-16	-17
H	H	T	T	H	T	H	H	H	H	H

Traceback: Same as edit distance!

Start from largest value and remember 'where I came from'

Viterbi Algorithm

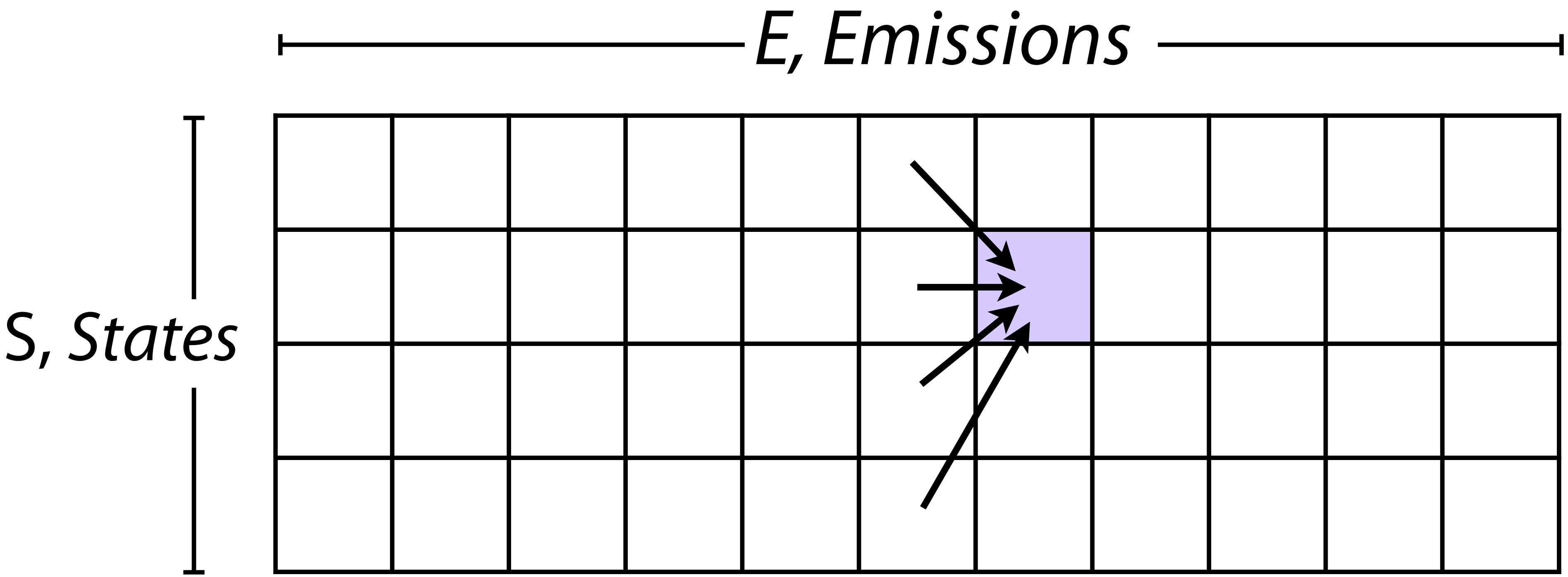
These get small — now log_2 scaled

-1.32	-2.38	-5.44	-8.35	-8.08	-11.1	-11.6	-12.6	-13.7	-14.7	-15.8
-2	-3.64	-4.7	-6.4	-8.2	-9.9	-11.7	-13.4	-14.9	-16	-17
H	H	T	T	H	T	H	H	H	H	H

Traceback: Same as edit distance!

Start from largest value and remember 'where I came from'

Viterbi Algorithm



What is running time?

What will you get out of this class?

Understand fundamental string algorithms

Experience applying data structures, algorithms, and algorithm design principles to real world problems

Justify implementation choices based on theoretical or practical considerations

Build a foundation for future data science projects

Thanks for listening! Have a good winter break