Genome 559
Intro to Statistical and Computational Genomics

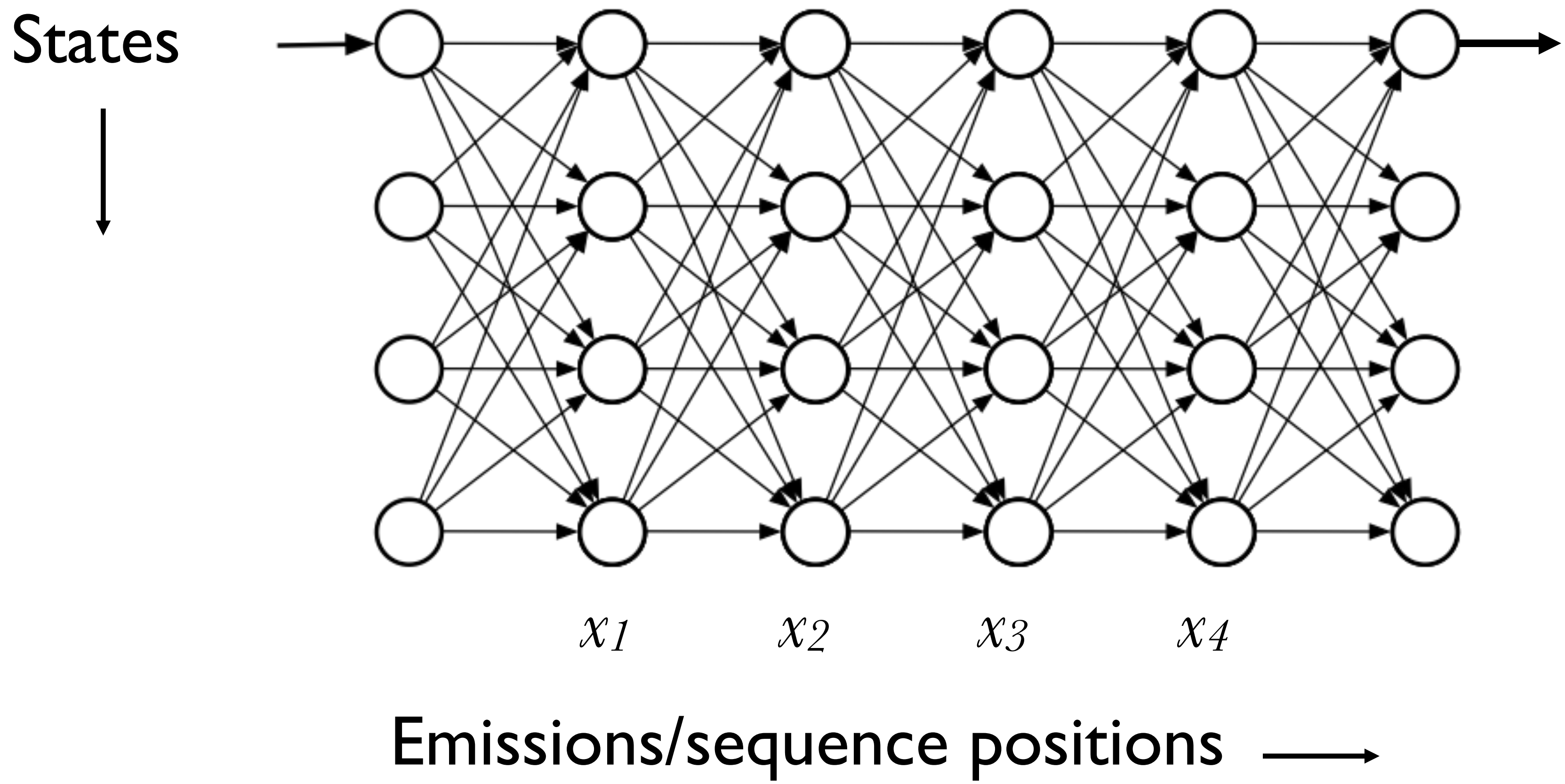
18a: Hidden Markov Models

Hidden Markov Models
(HMMs; Claude Shannon, 1948)

Notation:

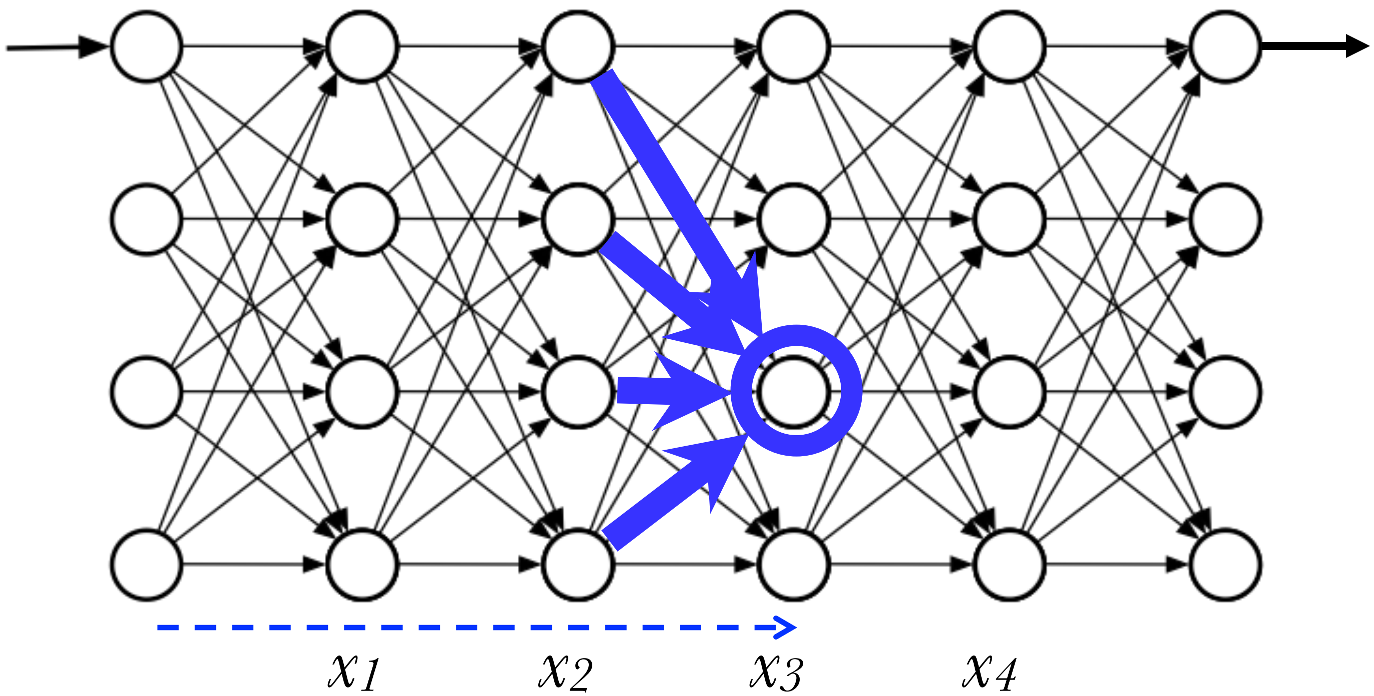
States:	$1, 2, 3, \ldots$
Paths:	sequences of states $\pi = (\pi_1, \pi_2, \ldots)$
Transitions:	$a_{k,l} = P(\pi_i = l \mid \pi_{i-1} = k)$
Emissions:	$e_k(b) = P(x_i = b \mid \pi_i = k)$
Observed data:	emission sequence
Hidden data:	state/transition sequence

An HMM (unrolled)



The Viterbi Algorithm

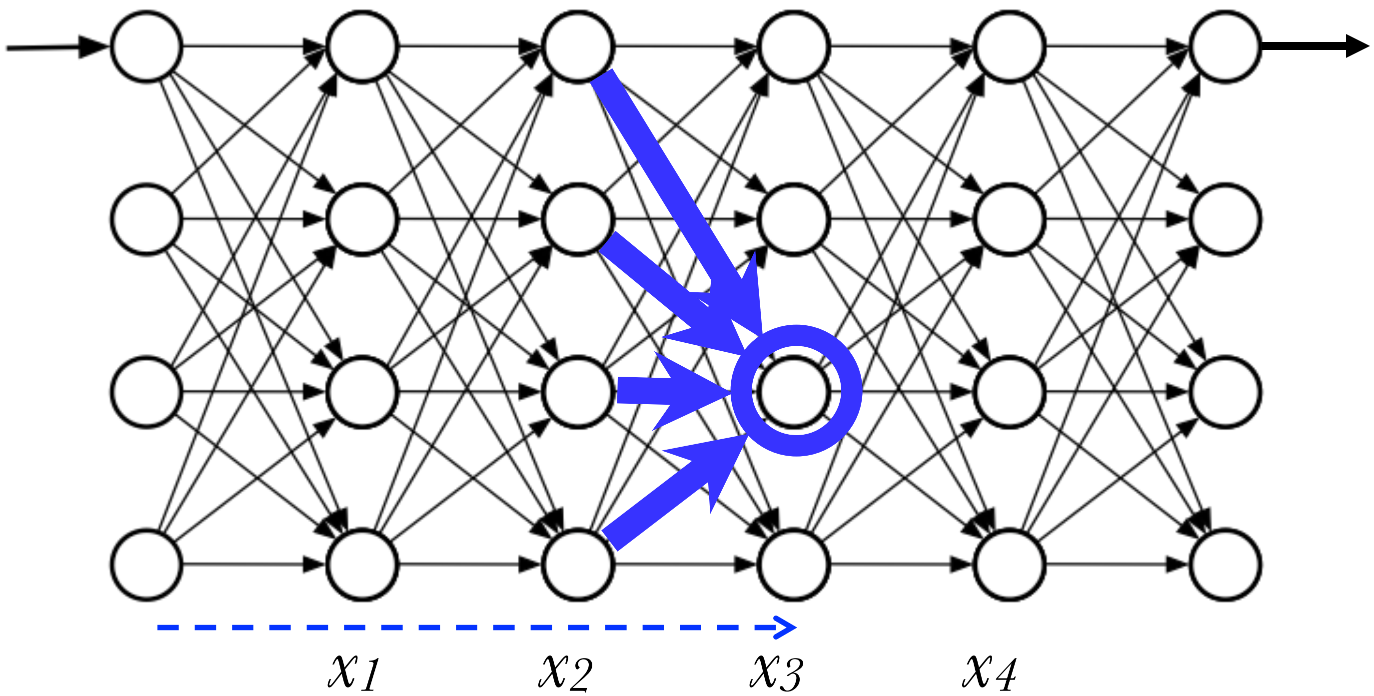
For each state/time, want *max* probability of any path leading *to* it, with given emissions



$$v_l(i+1) = \underbrace{e_l(x_{i+1})}_{\text{emission}} \cdot \max_k (v_k(i)\, \underbrace{a_{k,l}}_{\text{transition}})$$

The Forward Algorithm

For each state/time, want *total* probability of all paths leading *to* it, with given emissions



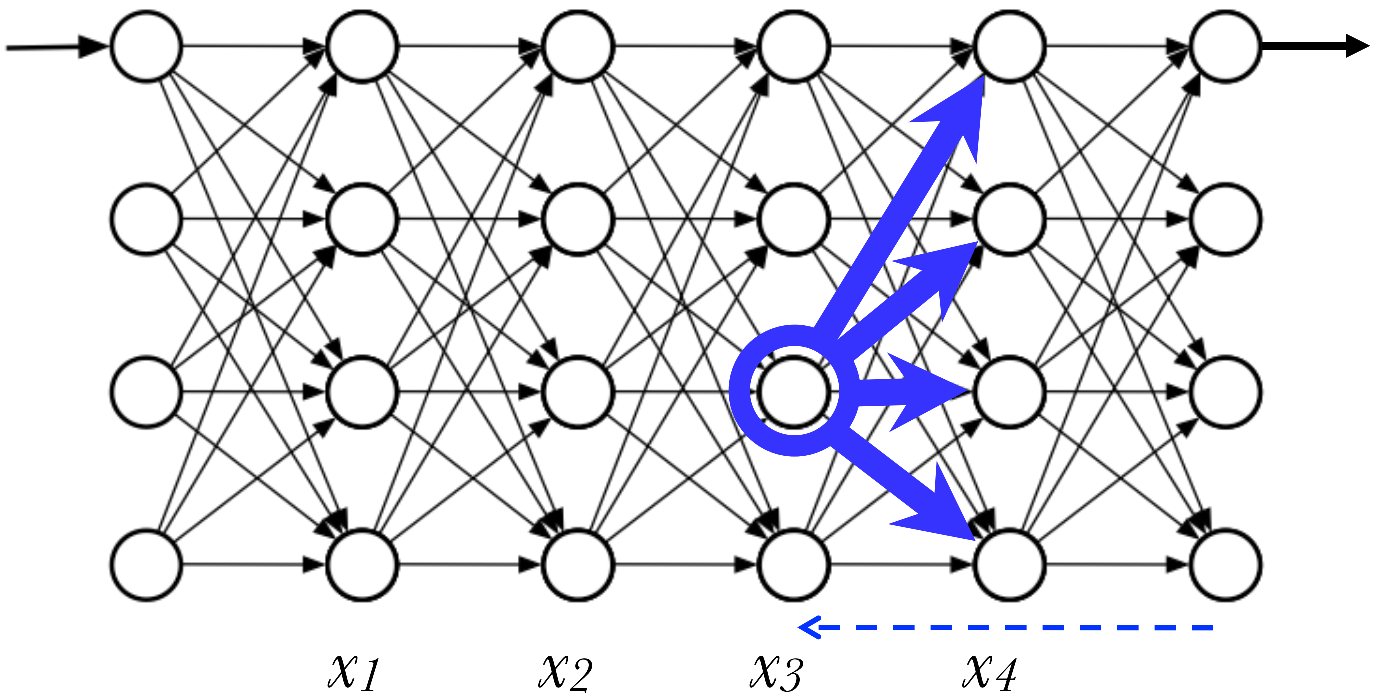
$$f_k(i) \triangleq P(x_1 \ldots x_i,\ \pi_i = k)$$

$$f_l(i+1) = e_l(x_{i+1}) \sum_k f_k(i) a_{k,l}$$

$$P(x) = \sum_\pi P(x,\pi) = \sum_k f_k(n) a_{k,0}$$

The Backward Algorithm

Similar: for each state/time, want total probability of all paths *from* it, with given emissions, *conditional* on that state.



$$b_k(i) \triangleq P(x_{i+1} \cdots x_n \mid \pi_i = k)$$

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

$$b_k(n) = a_{k,0}$$

In state *k* at step *i* ?

$$
\begin{aligned}
&P(x,\ \pi_i = k) \\
&\quad = \quad P(x_1, \ldots, x_i,\ \pi_i = k) \cdot P(x_{i+1}, \ldots, x_n \mid x_1, \ldots, x_i,\ \pi_i = k) \\
&\quad = \quad P(x_1, \ldots, x_i,\ \pi_i = k) \cdot P(x_{i+1}, \ldots, x_n \mid \pi_i = k) \\
&\quad = \quad f_k(i) \cdot b_k(i)
\end{aligned}
$$

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

Posterior Decoding

Question 1: Probability of state k at step i

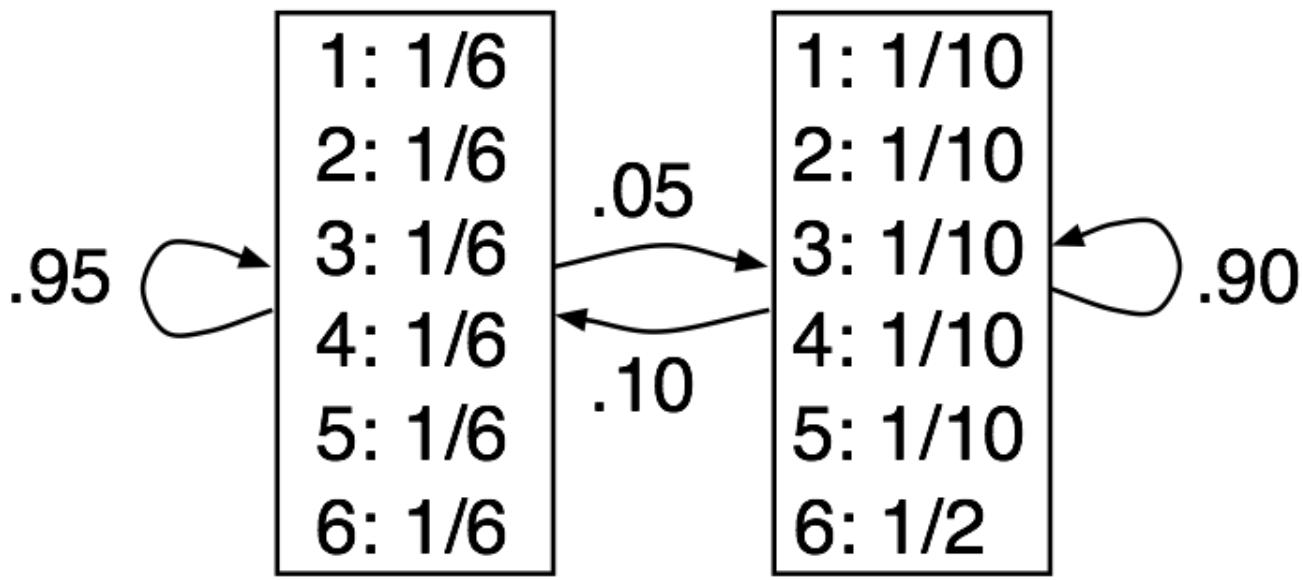
$$P(\pi_i = k \mid x)$$

Question 2: What's the most likely state at step *i*?

$$\hat{\pi}_i = \arg\max_k P(\pi_i = k \mid x)$$

The Occasionally Dishonest Casino

1 fair die, 1 “loaded” die, occasionally swapped



```
Rolls    315116246446644245311321631164152133625144543631656626566666
Die      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLL
Viterbi  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLL

Rolls    651166453132651245636664631636663162326455236266666625151631
Die      LLLLLLFFFFFFFFFFFLLLLLLLLLLLLLLLLLFFLLLLLLLLLLLLLLLFFFFFFFFF
Viterbi  LLLLLLFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFF

Rolls    222555441666566563564324364131513465146353411126414626253356
Die      FFFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLL
Viterbi  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls    366163666466232534413661661163252562462255265252266435353336
Die      LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi  LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Rolls    233121625364414432335163243633665562466662632666612355245242
Die      FFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
Viterbi  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
```

Figure 3.5

Rolls: Visible data–300 rolls of a die as described above.
Die: Hidden data–which die was actually used for that roll (F = fair, L = loaded).
Viterbi: the prediction by the Viterbi algorithm is shown.

From DEKM

Posterior Decoding

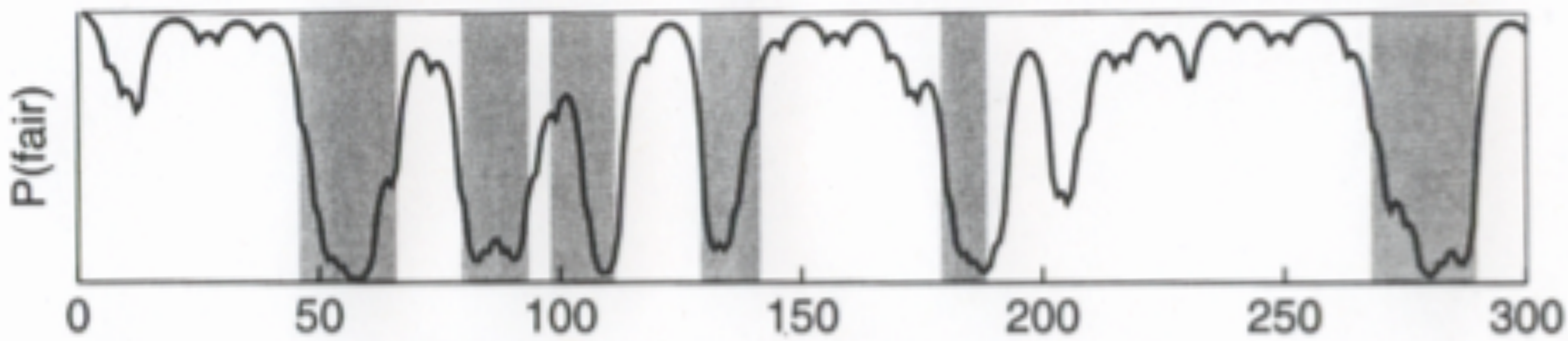


Figure 3.6 *The posterior probability of being in the state corresponding to the fair die in the casino example. The x axis shows the number of the roll. The shaded areas show when the roll was generated by the loaded die.*

From DEKM

Training

Given model topology & training sequences,
learn transition and emission probabilities

If π known, then MLE is just frequency observed in training data

$$a_{k,l} = \frac{\text{count of } k \rightarrow l \text{ transitions}}{\text{count of } k \rightarrow \text{anywhere transitions}}$$

$$e_k(b) = \ldots$$

$\leftarrow$ + pseudocounts?

If π hidden, then use EM:
given π, estimate θ; given θ estimate π. } 2 ways

Viterbi Training

given π, estimate θ; given θ estimate π

Make initial estimates of parameters θ
Find Viterbi path π for each training sequence
Count transitions/emissions on those paths,
  getting new θ
Repeat

Not rigorously optimizing desired likelihood, but
  still useful & commonly used.
  (Arguably good if you're doing Viterbi decoding.)

Baum-Welch Training

EM: given θ, estimate π ensemble; then re-estimate θ

$$P(\pi_i = k,\ \pi_{i+1} = l \mid x, \theta)$$
$$= \quad \frac{f_k(i \mid \theta)\ a_{k,l}\ e_l(x_{i+1})\ b_l(i+1 \mid \theta)}{P(x \mid \theta)}$$

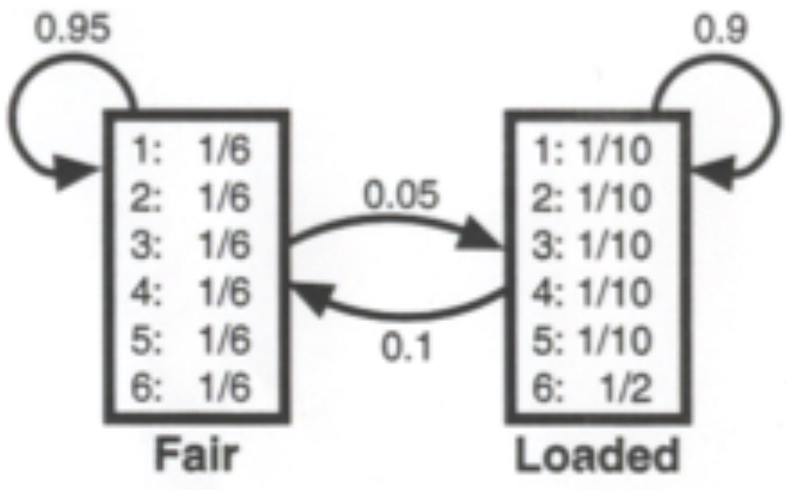
Estimated # of $k \rightarrow l$ transitions $\hat{A}_{k,l}$

$$= \quad \sum_{\text{training seqs } x^j} \sum_i P(\pi_i = k,\ \pi_{i+1} = l \mid x^j, \theta)$$

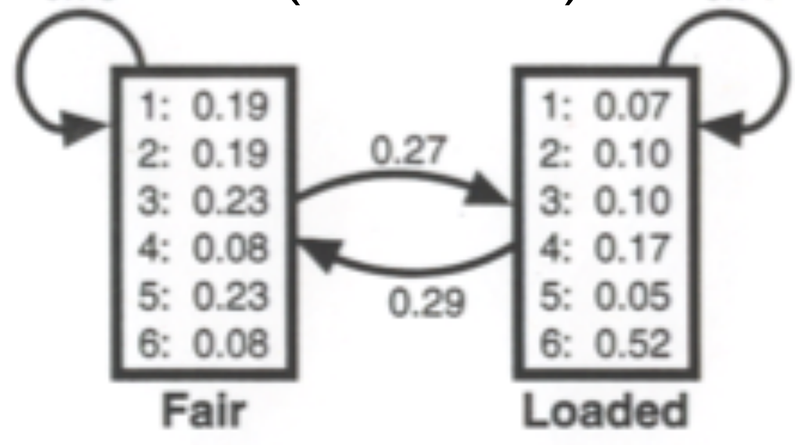
New estimate $\hat{a}_{k,l} \;=\; \dfrac{\hat{A}_{k,l}}{\sum_l \hat{A}_{k,l}}$

Emissions: similar

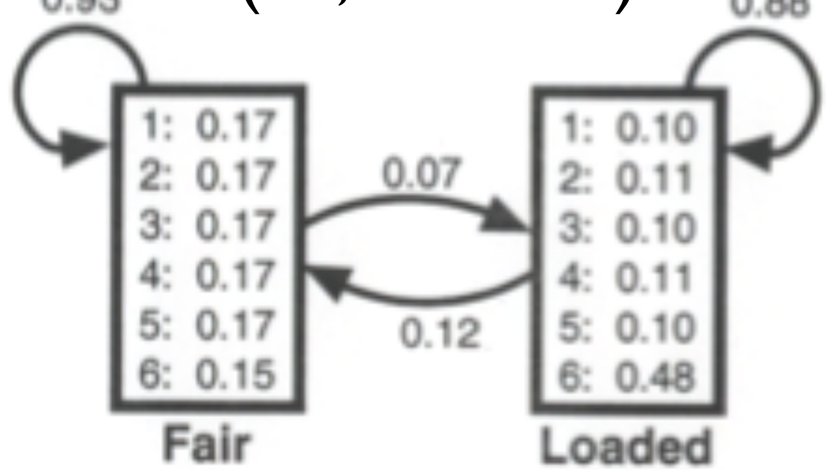
True Model



B-W Learned Model (300 rolls)



B-W Learned Model (30,000 rolls)



Log-odds (vs all F) per roll

True model	0.101 bits
300-roll est.	0.097 bits
30k-roll est.	0.100 bits

(NB: overestimated)

From DEKM

Summary

Forward/backward all estimation of quantities like prob of being in a given state at a given time

Training from labeled data (non-hidden path) is usual MLE estimation: count frequency of specific transitions/emissions

From *unlabeled* data, can use *estimated* counts and iterate (the "EM" algorithm).