CSE 527
Lectures ~12-13

Markov Models and Hidden Markov Models

Markov & Hidden Markov Models

- Reference: Durbin, Eddy, Krogh and Mitchison, "Biological Sequence Analysis" Cambridge, 1998

Independence

- A key issue: All models we've talked about so far assume *independence* of nucleotides in different positions - definitely unrealistic.

Example: "CpG Islands"

- CpG - 2 adjacent nucs, same strand (not Watson-Crick)
- C of CpG is often *methylated* (in Eukaryotes)
- Methyl-C mutates to T relatively easily
- Net: CpG is less common than expected genome-wide: f(CpG) < f(C)*f(G)
- BUT in promoter (& other) regions, CpG remain unmethylated, so CpG ->TpG less likely there: makes "CpG Islands"

CpG Islands

- CpG Islands
  - More CpG than elsewhere
  - More C & G than elsewhere, too
  - Typical length: few 100 to few 1000 bp
- Questions
  - Given short sequence (say 200 bp), is it a CpG island or not?
  - Given long sequence (say, 10-100kb), fing CpG islands in it?

Markov Chains

A sequence of random variables $X_1 X_2 \ldots$ is a K-th order Markov chain if $\forall i$:

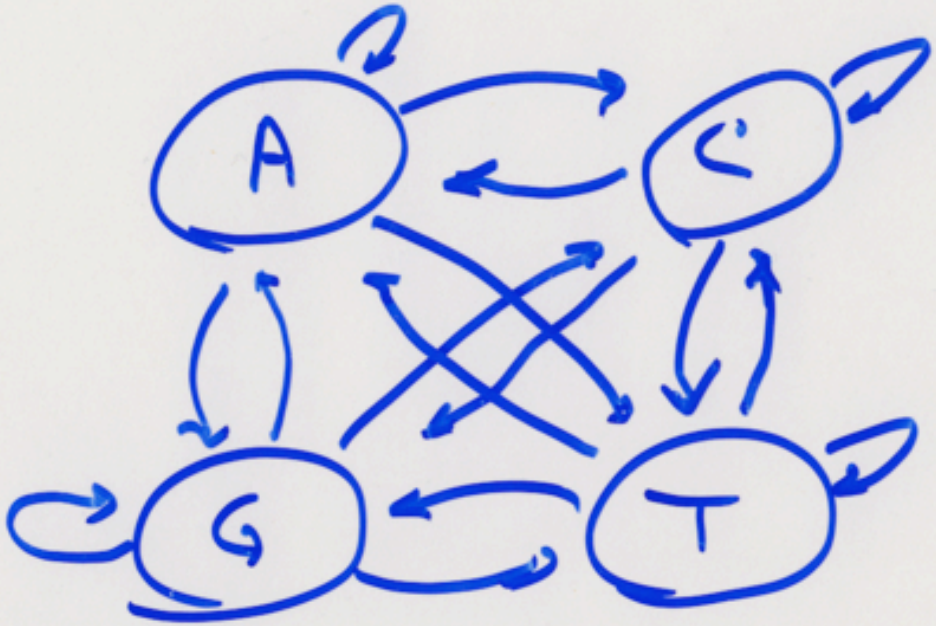
$$Pr(X_i \mid X_1 X_2 \cdots X_{i-1}) = Pr(X_i \mid X_{i-K} X_{i-K+1} \cdots X_{i-1})$$

i.e. i^{th} value is independent of all but previous K values

Example 1: uniform random A A C T A G ... } 0th order M.M.

Example 2: Weight Matrix Model } 0th order M.M.

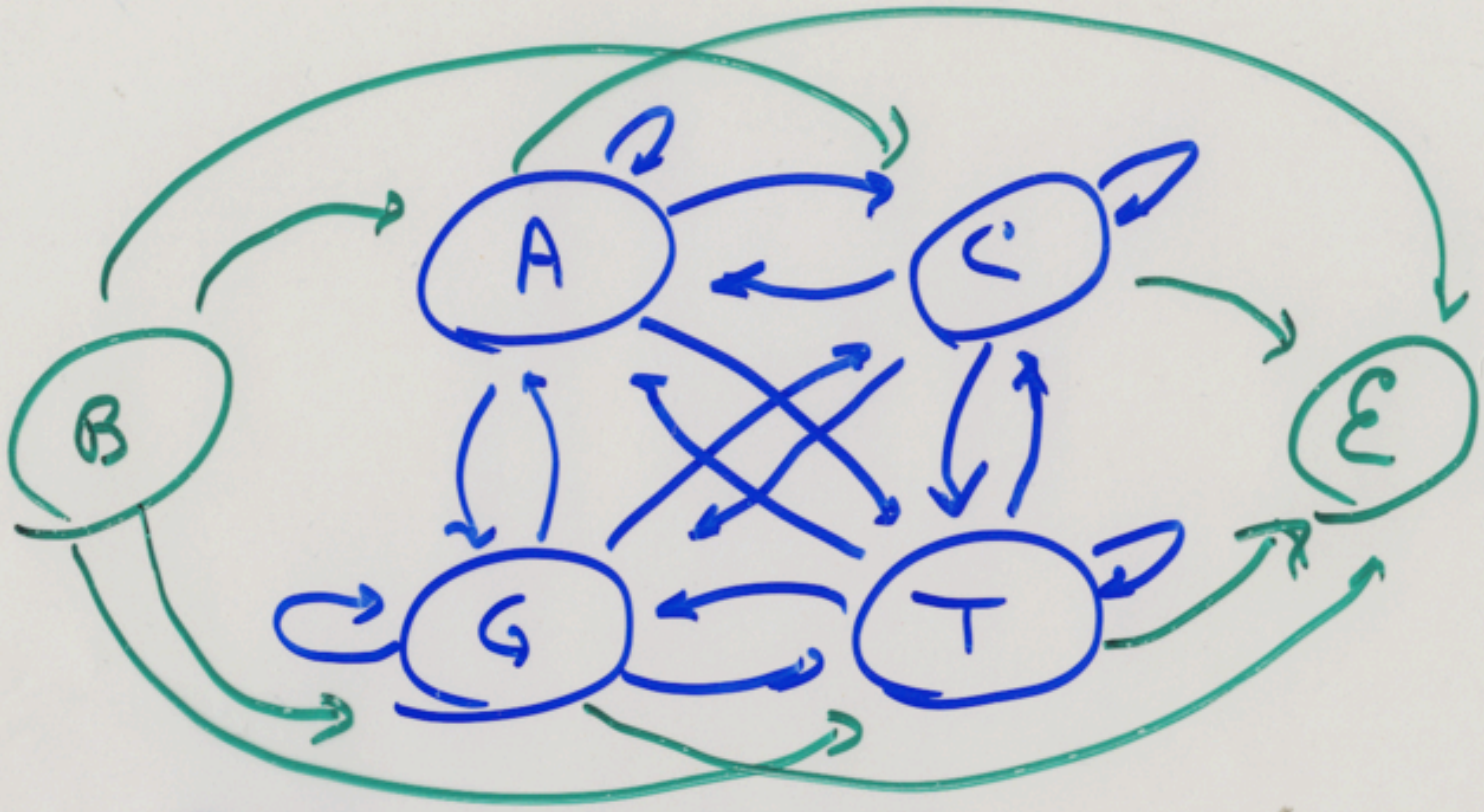
Example 3: A, C, G, T, but Pr(G following C) lower: 1st order MM



States : A, C, G, T

Emission : corresponding letter

Transition : $a_{st} = p(x_i = t \mid x_{i-1} = s)$



States : A, C, G, T

Emission : corresponding letter

Transition : $a_{st} = P(x_i = t \mid x_{i-1} = s)$

Begin / End States

Probability of Emitting Sequence x

$$x = x_1 \; x_2 \; \cdots \; x_n$$

$$P(x) = P(x_1, x_2, \ldots, x_n)$$

$$= P(x_n \mid x_{n-1} \; x_{n-2} \cdots x_1) \cdot P(x_{n-1} \mid x_{n-2} \cdots x_1) \cdots P(x_1)$$

$$= P(x_n \mid x_{n-1}) \cdot P(x_{n-1} \mid x_{n-2}) \cdots P(x_2 \mid x_1) \cdot P(x_1)$$

$$= P(x_1) \prod_{i=1}^{n-1} a_{x_i, x_{i+1}}$$

Training

- Max likelihood estimates for transition probabilities are just the frequencies of transitions when emitting the training sequences

and derived two Markov chain models, one for the regions labelled as CpG islands (the '+' model) and the other from the remainder of the sequence (the '−' model). The transition probabilities for each model were set using the equation

$$a_{st}^{+} = \frac{c_{st}^{+}}{\sum_{t'} c_{st'}^{+}}, \tag{3.3}$$

and its analogue for a_{st}^{-}, where c_{st}^{+} is the number of times letter t followed letter s in the labelled regions. These are the maximum likelihood (ML) estimators for the transition probabilities, as described in Chapter 1.

(In this case there were almost 60 000 nucleotides, and ML estimators are adequate. If the number of counts of each type had been small, then a Bayesian estimation process would have been more appropriate, as discussed in Chapter 11 and below for HMMs.) The resulting tables are

+	A	C	G	T
A	0.180	0.274	0.426	0.120
C	0.171	0.368	0.274	0.188
G	0.161	0.339	0.375	0.125
T	0.079	0.355	0.384	0.182

−	A	C	G	T
A	0.300	0.205	0.285	0.210
C	0.322	0.298	0.078	0.302
G	0.248	0.246	0.298	0.208
T	0.177	0.239	0.292	0.292

where the first row in each case contains the frequencies with which an A is followed by each of the four bases, and so on for the other rows, so each row

Discrimination/Classification

Calculate log likelihood ratio for CpG model vs background model

$$S(x) = \log \frac{P(x \mid +\text{model})}{P(x \mid -\text{model})} = \sum_{i=1}^{n} \log \frac{a^{+}_{x_{i-1},x_i}}{a^{-}_{x_{i-1},x_i}}$$

les the probability for G following C is lower than that for C following G,
ı the effect is stronger in the '−' table, as expected.
e these models for discrimination, we calculate the log-odds ratio

$$S(x) = \log \frac{P(x|\text{model}+)}{P(x|\text{model}-)} = \sum_{i=1}^{L} \log \frac{a^{+}_{x_{i-1}x_i}}{a^{-}_{x_{i-1}x_i}}$$

$$= \sum_{i=1}^{L} \beta_{x_{i-1}x_i}$$

is the sequence and $\beta_{x_{i-1}x_i}$ are the log likelihood ratios of corresponding
n probabilities. A table for β is given below in bits:[1]

β	A	C	G	T
A	−0.740	0.419	0.580	−0.803
C	−0.913	0.302	1.812	−0.685
G	−0.624	0.461	0.331	−0.730
T	−1.169	0.573	0.393	−0.679

ɜ 3.2 shows the distribution of scores, $S(x)$, normalised by dividing by
gth, i.e. as an average number of bits per molecule. If we had not nor-
by length, the distribution would have been much more spread out.

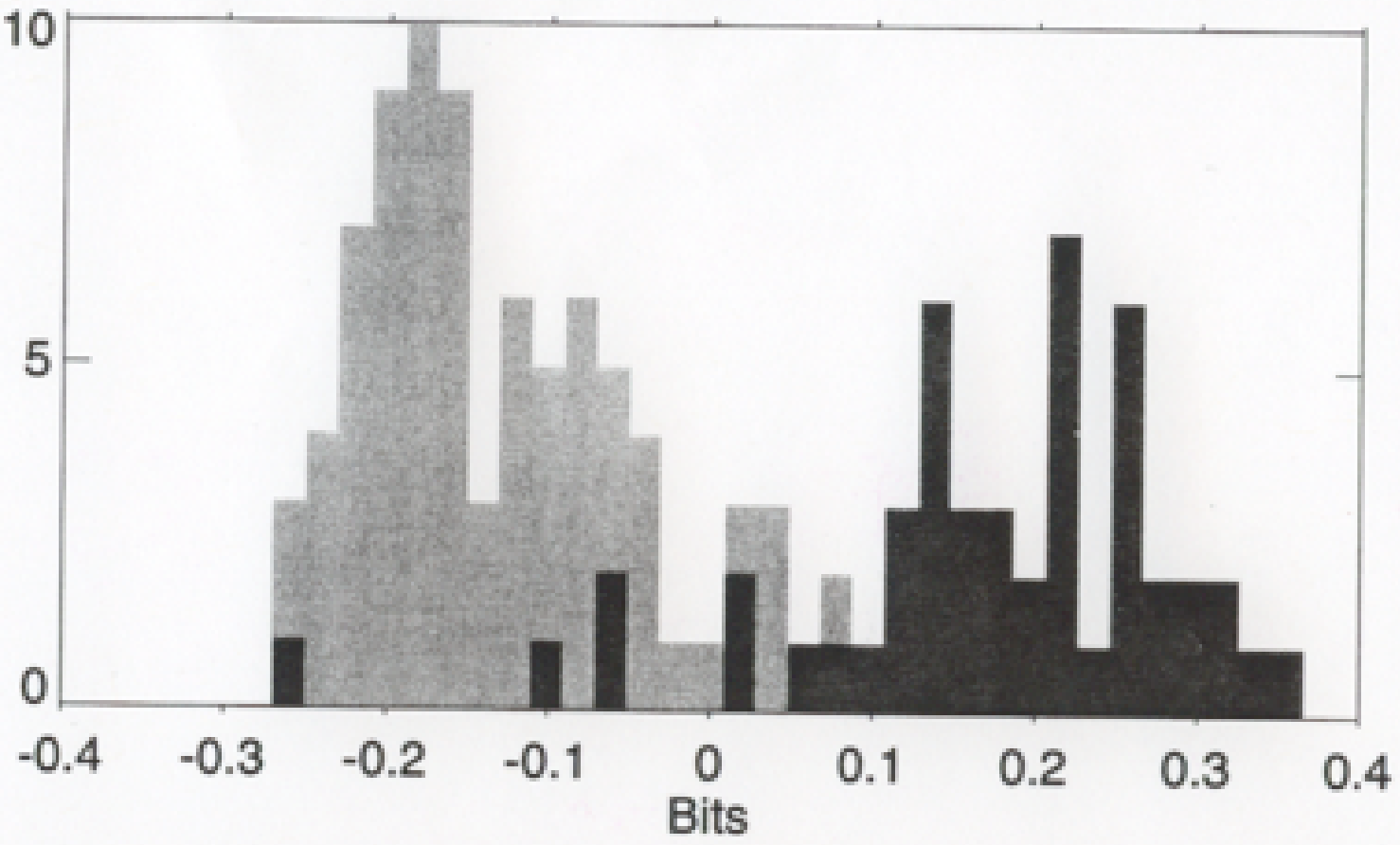


Figure 3.2 *The histogram of the length-normalised scores for all the sequences. CpG islands are shown with dark grey and non-CpG with light grey.*

Above answers Q1: "given short sequence, is it more likely to be from feature model or background model?"

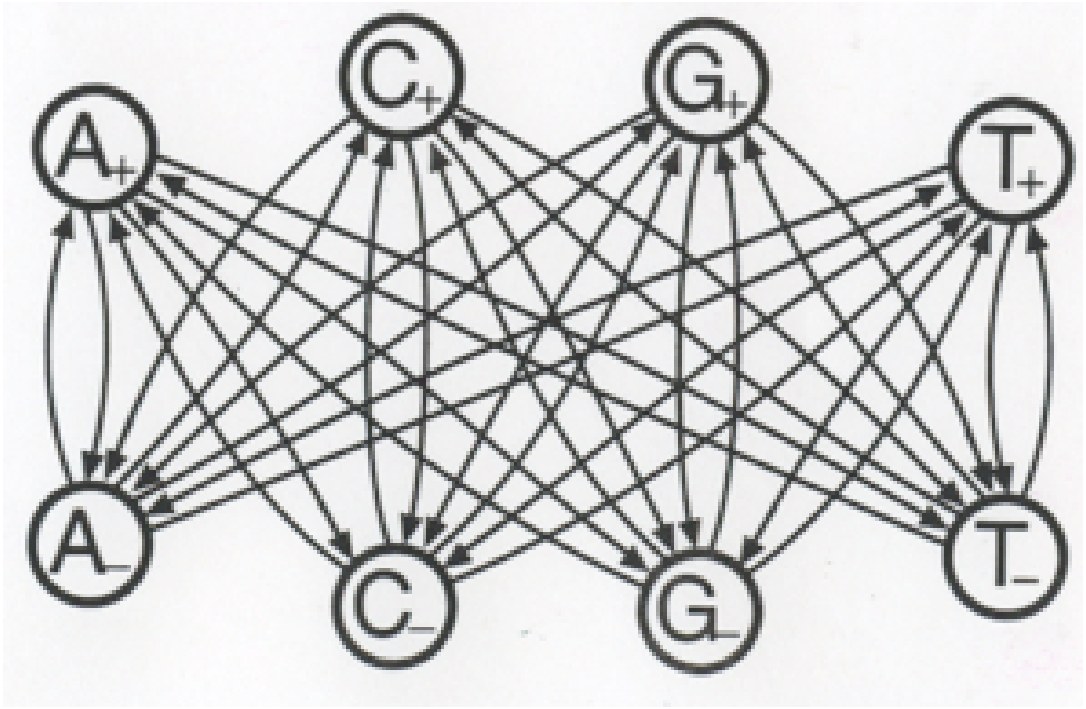
What about Q2: "Given long sequence where are features in it?"

Approach 1: score, say, 100 base windows.

Pro: simple

Con: arbitrary window; fixed len

Approach 2: combine +2 - models


A+
C+
G+
T+
A−
C−
G−
T−

Hidden Markov Models (HMMs)

States : $1, 2, \cdots$

Paths : Sequences of states $\pi = (\pi_1, \pi_2, \cdots, \pi_n)$

Transitions : $a_{kl} = Prob(\pi_i = l \mid \pi_{i-1} = k)$

Emissions : $e_k(b) = Prob(x_i = b \mid \pi_i = k)$

Observed Data : only emission seq.

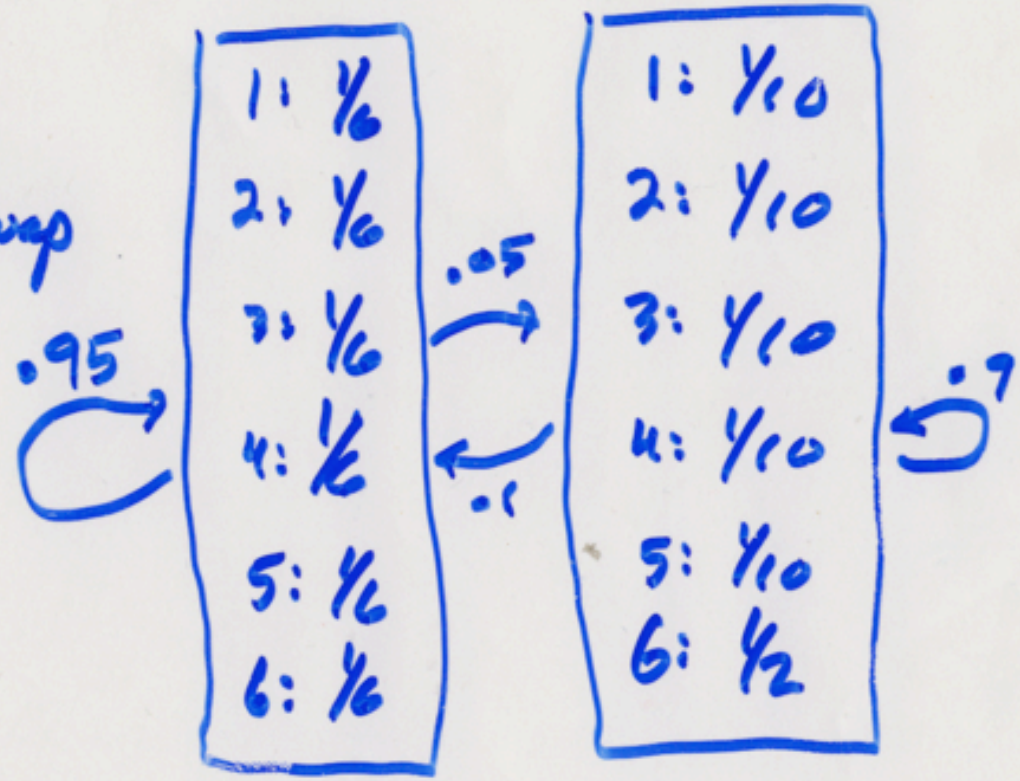
Hidden Data : The state/transition seq.

Example: "The Occasionally Dishonest Casino"

1 fair die

1 loaded die

occasionally swap them



```
Rolls   315116246446644245311321631164152133625144543631656626566666
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLL

Rolls   651166453132651245636664631636663162326455236266666625151631
Die     LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLLFFFFFFFFF
Viterbi LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF

Rolls   222555441666566563564324364131513465146353411126414626253356
Die     FFFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls   366163666466232534413661661163252562462255265252266435353336
Die     LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Rolls   233121625364414432335163243633665562466662632666612355245242
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLFFFFFFFFFF
```

Figure 3.5 *The numbers show 300 rolls of a die as described in the example. Below is shown which die was actually used for that roll (F for fair and L for loaded). Under that the prediction by the Viterbi algorithm is shown.*

the model as described earlier. Each roll was generated either with the fair die (F) or the loaded one (L), as shown below the outcome of the roll in Figure 3.5. The Viterbi algorithm was used to predict the state sequence, i.e. which die was used for each of the rolls. Generally, as you can see, the Viterbi algorithm has recovered the state sequence fairly well. □

Exercise

Joint probability of Given path π, Seq x

$$P(x,\pi) = a_{0\pi_1} \prod_{i=1}^{n} e_{\pi_i}(x_i)\, a_{\pi_i \pi_{i+1}}$$

But π hidden

CGCG from
$C_+ G_+ C_+ G_+$
$C_- G_- C_- G_-$
$C_+ G_-$ $C_+ G_-$ etc ?

Alternatives

1. Most probable (single) path $\pi^* = \underset{\pi}{\operatorname{argmax}}\, P(x,\pi)$
2. Sequence of most probable states
$$\hat{\pi}_i = \underset{k}{\operatorname{argmax}}\, (P(\pi_i = k \mid x))$$
3. ...

The Viterbi Algorithm: Most Probable Path

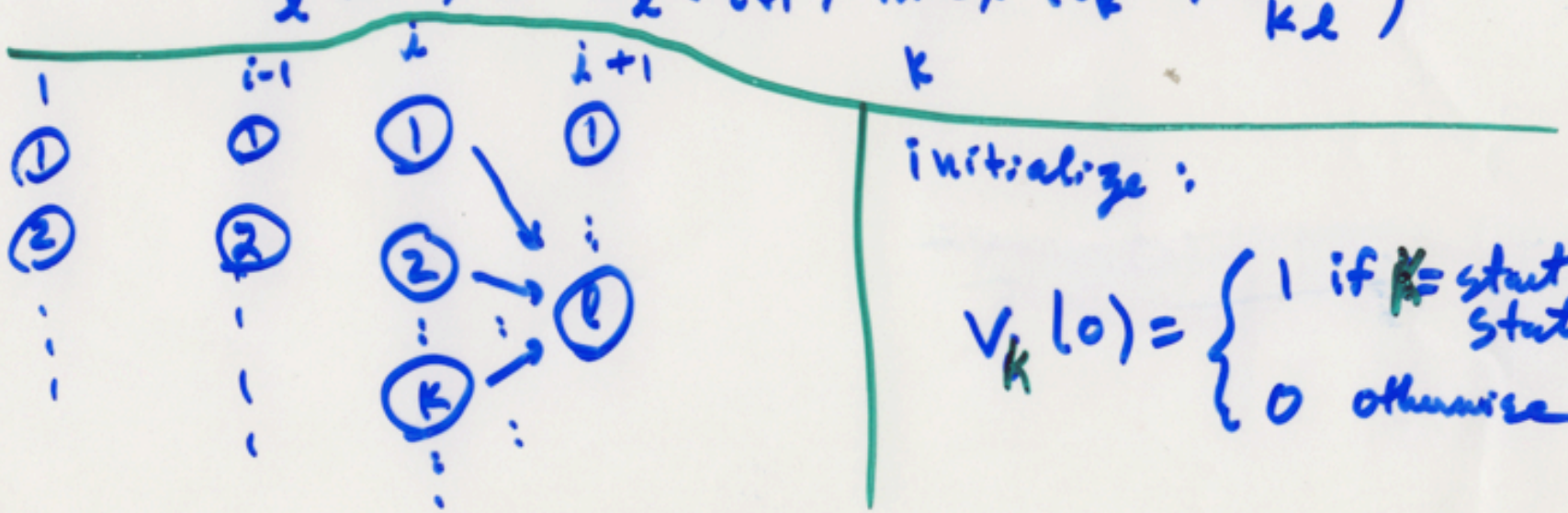
Want $\pi^* = \underset{\pi}{\operatorname{argmax}}\ P(x, \pi)$

- Often true that 1 path dominates all others
  (if not, other approaches may be preferable)
- Key Problem: exponentially many π

Viterbi

$V_k(i)$ = Probability of most probable path ending in state k after emitting $x_1 \dots x_i$

$$V_\ell(i+1) = e_\ell(x_{i+1}) \max_k (V_k(i)\, a_{k\ell})$$



initialize:

$$V_k(0) = \begin{cases} 1 & \text{if } k = \text{start state} \\ 0 & \text{otherwise} \end{cases}$$

Viterbi Traceback

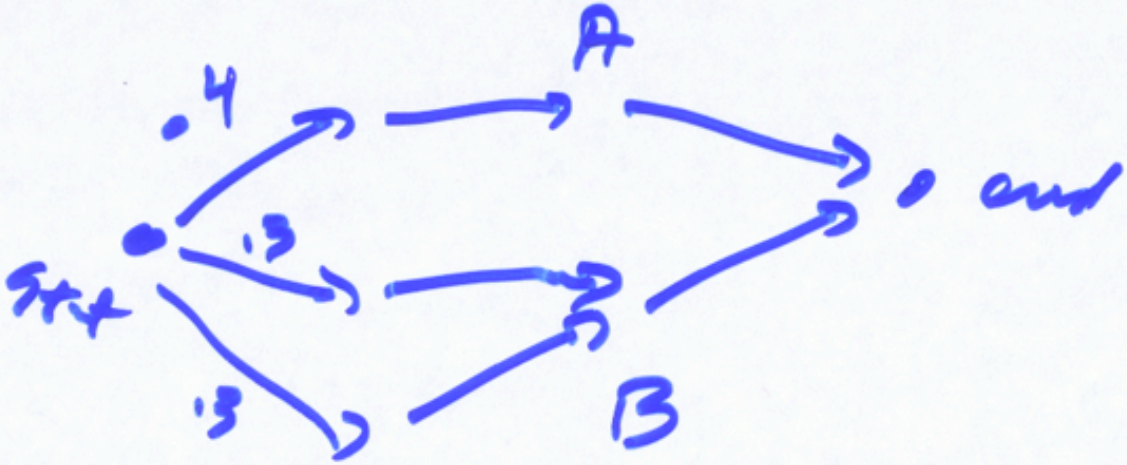
- Above finds *probability* of best path
- To find the path itself, trace backward to state *k* attaining the max at each stage

Yet to come

- More on HMMs:
  - Viterbi, forward, backward
  - Posterior decoding
  - Training: Viterbi & Baum-Welch
  - Model structure

HMM

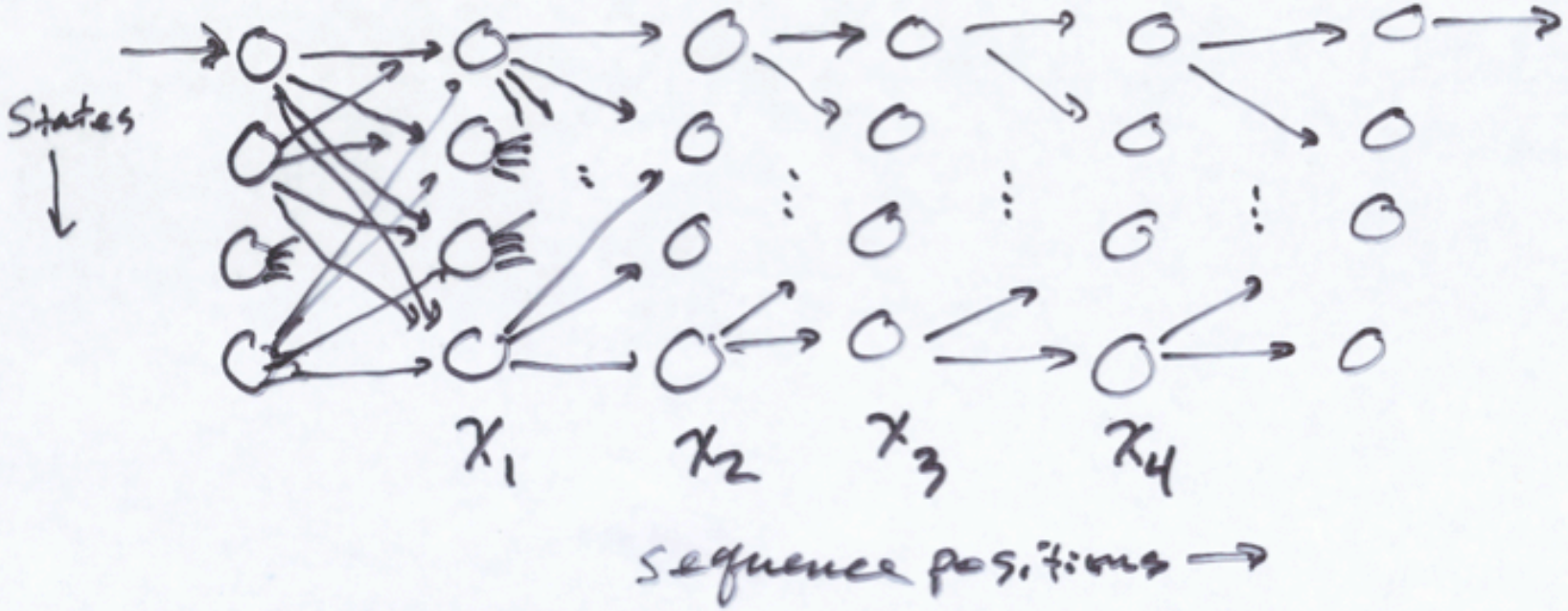
Viterbi $\max_{\pi} P(X, \pi)$



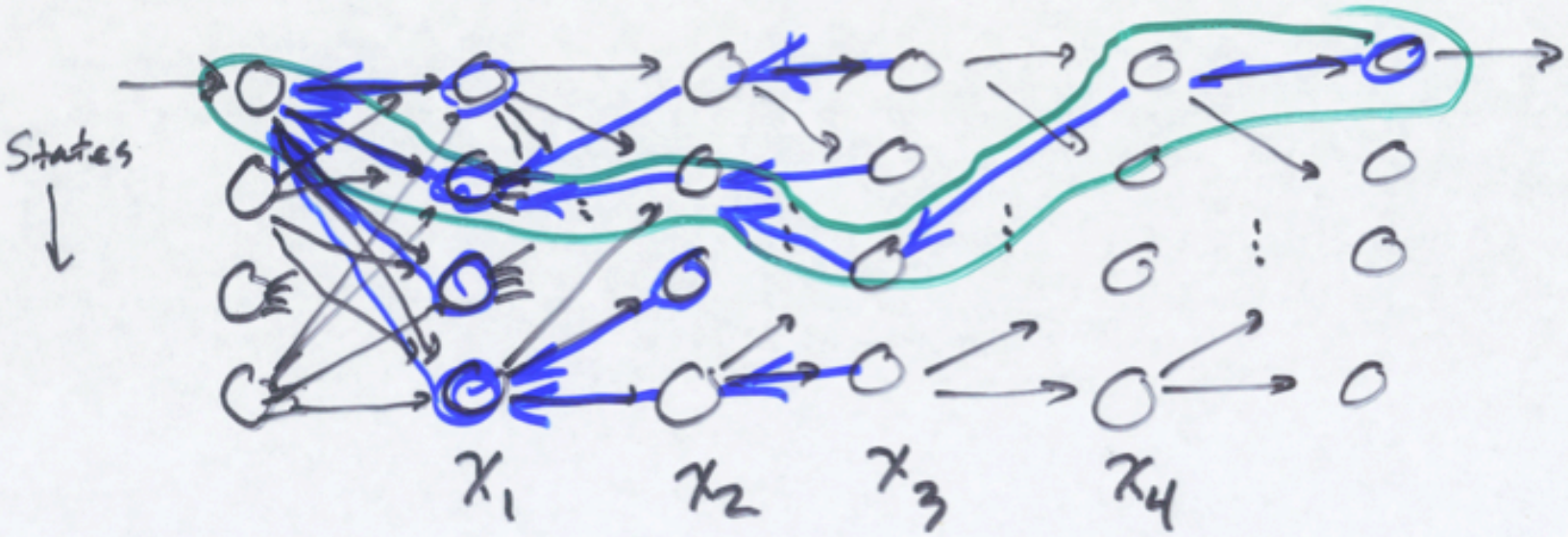
Most probable path thru A

but B is most probable state

at step 2.

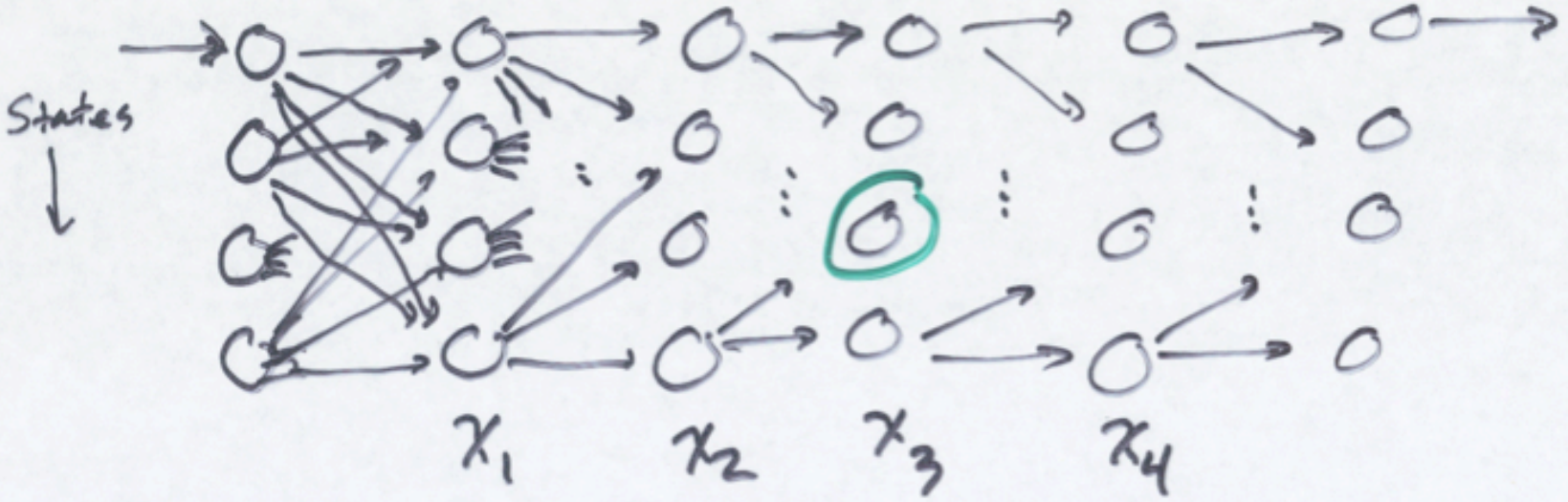

States
x_1
x_2
x_3
x_4
sequence positions →

Viterbi: Best path to each node



$$V_\ell(i+1) = e_\ell(x_{i+1}) \cdot \max_k \left(V_k(i) \cdot a_{k\ell} \right)$$

Forward Algorithm



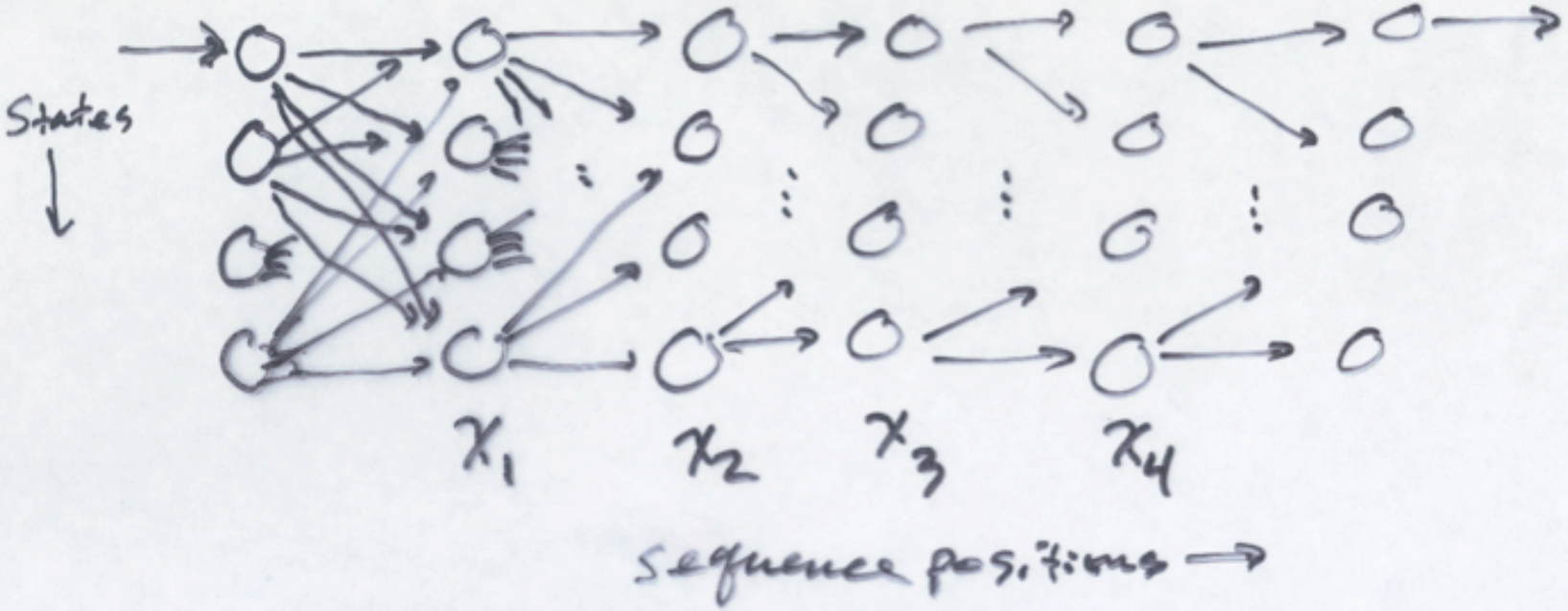
For each state/time want total prob. of all paths leading to it

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

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

$$P(x) = \sum_k f_k(n)\, a_{k0}$$

Backward Algorithm



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

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

$$b_k(n) = a_{k0}$$

$$P(x, \pi_i = k) = \underbrace{P(x_1 \cdots x_i, \pi_i = k)}_{f_k(i)} \underbrace{P(x_{i+1} \cdots x_n \mid x_1 \cdots x_i, \pi_i = k)}_{= P(x_{i+1} \cdots x_n \mid \pi_i = k) \; b_k(i)}$$

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

Posterior Decoding, I

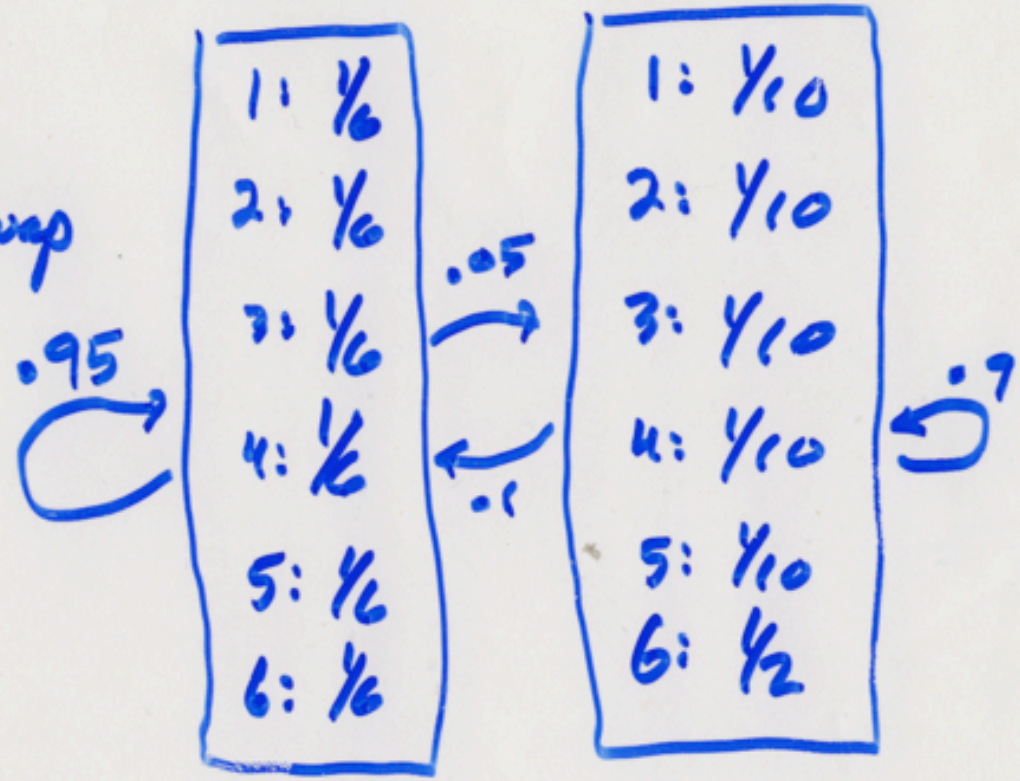
Alt! $\hat{\pi}_i = \underset{k}{\operatorname{argmax}} \left(p(\pi_i = k \mid x) \right)$

Example: "The Occasionally Dishonest Casino"

1 fair die

1 loaded die

occasionally swap them



```
Rolls   315116246446644245311321631164152133625144543631656626566666
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLL

Rolls   651166453132651245636664631636663162326455236266666625151631
Die     LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLLFFFFFFFFF
Viterbi LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF

Rolls   222555441666566563564324364131513465146353411126414626253356
Die     FFFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls   366163666466232534413661661163252562462255265252266435353336
Die     LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Rolls   233121625364414432335163243633665562466662632666612355245242
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLFFFFFFFFFF
```

Figure 3.5 *The numbers show 300 rolls of a die as described in the example. Below is shown which die was actually used for that roll (F for fair and L for loaded). Under that the prediction by the Viterbi algorithm is shown.*

the model as described earlier. Each roll was generated either with the fair die (F) or the loaded one (L), as shown below the outcome of the roll in Figure 3.5. The Viterbi algorithm was used to predict the state sequence, i.e. which die was used for each of the rolls. Generally, as you can see, the Viterbi algorithm has recovered the state sequence fairly well. □

Exercise

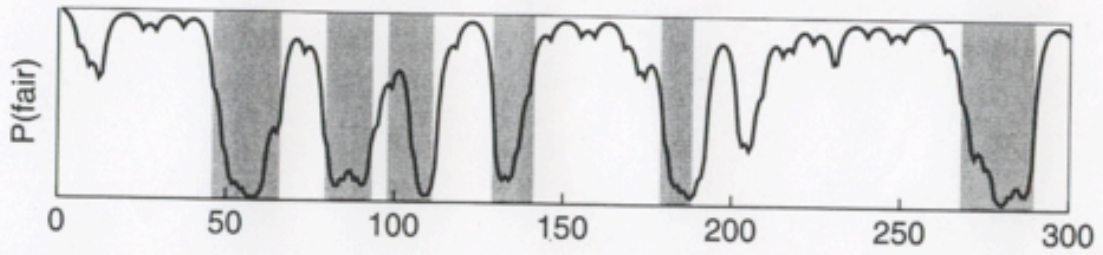


Figure 3.6 *The posterior probability of being in the state corresponding to the fair die in the casino example. The x axis shows the number of the roll. The shaded areas show when the roll was generated by the loaded die.*

The first approach is to define a state sequence $\hat{\pi}_i$ that can be used in place of π_i^*,

$$\hat{\pi}_i = \underset{k}{\text{argmax}}\, P(\pi_i = k|x). \tag{3.15}$$

As suggested by its definition, this state sequence may be more appropriate when we are interested in the state assignment at a particular point i, rather than the complete path. In fact, the state sequence defined by $\hat{\pi}_i$ may not be particularly likely as a path through the entire model; it may even not be a legitimate path at all if some transitions are not permitted, which is normally the case.

The second, and perhaps more important, new decoding approach arises when it is not the state sequence itself which is of interest, but some other property derived from it. Assume we have a function $g(k)$ defined on the states. The natural value to look at then is

$$G(i|x) = \sum_k P(\pi_i = k|x)g(k) \tag{3.16}$$

Posterior Decoding, II

Alt1

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

Alt2

$g(k)$ function on states

$$G(i \mid x) = \sum_k P(\pi_i = k \mid x) \cdot g(k)$$

CpG Islands Again

Data: 41 human seqs, totaling 60Kbp, w/ 48 CpG islands
avg length ~ 1Kbp each

Viterbi

Found 46 of 48
Plus 121 "false pos"

Post process:
merge within 500
discard < 500

46/48
Plus 67 false pos

Posterior decoding

Same 2 false neg
236 false pos

again 46/48
Plus 83 false neg

Training

Given model topology

Given i.i.d. training sequences

Want to learn transition & Emission probabilities

If π known, then

MLE $a_{kl} = \dfrac{\text{Count } k \to l}{\text{Count } k \to \text{anywhere}}$

$e_k(b) =$ similar

π hidden

Use EM: given π can estimate θ

given θ ... π

Viterbi training

make initial parameter estimates

Calc Viterbi path for each training sequence

Count transitions & emissions

→ new θ

iterate

---

not rigorously optimizing desired likelihood.

(But still useful)

Baum-Welch Training

$$P(\pi_i = k, \pi_{i+1} = l \mid x, \theta)$$

$$= \frac{f_k(i) \cdot e_l(x_{i+1}) \cdot b_l(i+1)}{P(x \mid \theta)}$$

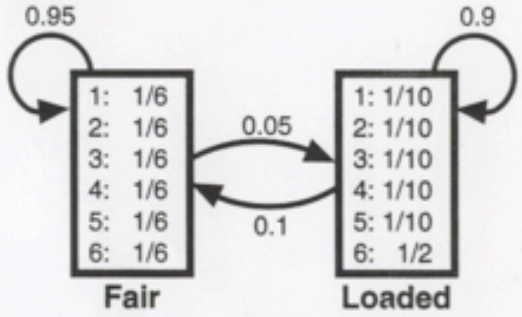
$$E(\#\text{ of } k \to l) = \sum_{\text{training seqs}} \frac{1}{P(x_j)} \sum_i$$

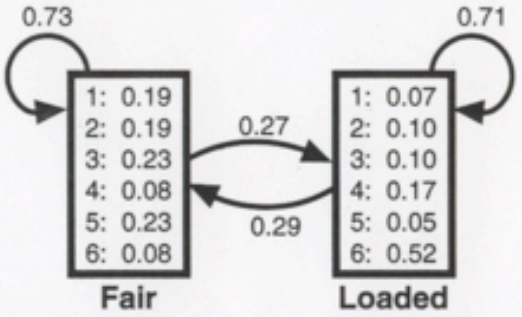
Emissions: similar

Example: The occasionally dishonest casino, part 5

We are suspicious that a casino is operated as described in the example on p. 54, but we do not know for certain. Night after night we collect data by simply observing rolls. When we have enough, we want to estimate a model. Assume the data we collected were the 300 rolls shown in Figure 3.5. From this sequence of observations a model was estimated by the Baum–Welch algorithm. Initially all the probabilities were set to random numbers. Here are diagrams of the model that generated the data (identical to the one in the example on p. 54) and the estimated model.

True Model



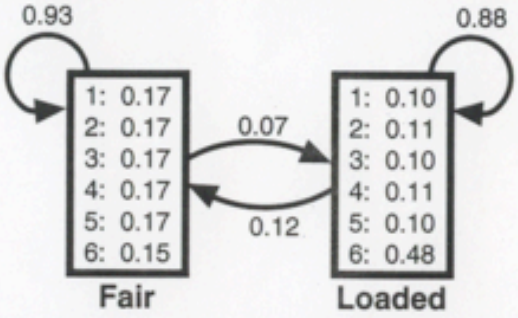


Learned Model (300 rolls)

You can see they are fairly similar, although the estimated transition probabilities are quite different from the real ones. This is partly a problem of local minima, and by trying more times it is actually possible to obtain a model closer to the correct one. However, from a limited amount of data it is never possible to estimate the parameters exactly.

To illustrate the last point, 30 000 random rolls were generated (data are not

shown!), and a model was estimated. This came very close to the correct one:



Learned Model (30,000 rolls)

To see how good these models are compared to just assuming a fair die all the time, the log-odds per roll was calculated using the 300 observations for the three models:

The correct model	0.101 bits
Model estimated from 300 rolls	0.097 bits
Model estimated from 30 000 rolls	0.100 bits

The worst model estimated from 300 rolls has almost the same log-odds as the two other models. That is because it is being tested on the same data as it was estimated from. Testing it on an independent set of rolls yields significantly lower log-odds than the other two models. □

Exercises

3.5 Derive the result (3.19). Use the fact that

$$P(\pi_i = k, \pi_{i+1} = l | x, \theta) = \frac{1}{P(x|\theta)} P(x, \pi_i = k, \pi_{i+1} = l | \theta),$$

Summary

Viterbi

best single path MAX of product

Forward

Summary over all paths
Sum of product

backward

Similar

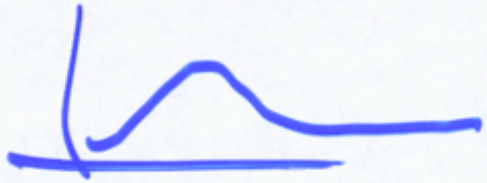
Baum Welch

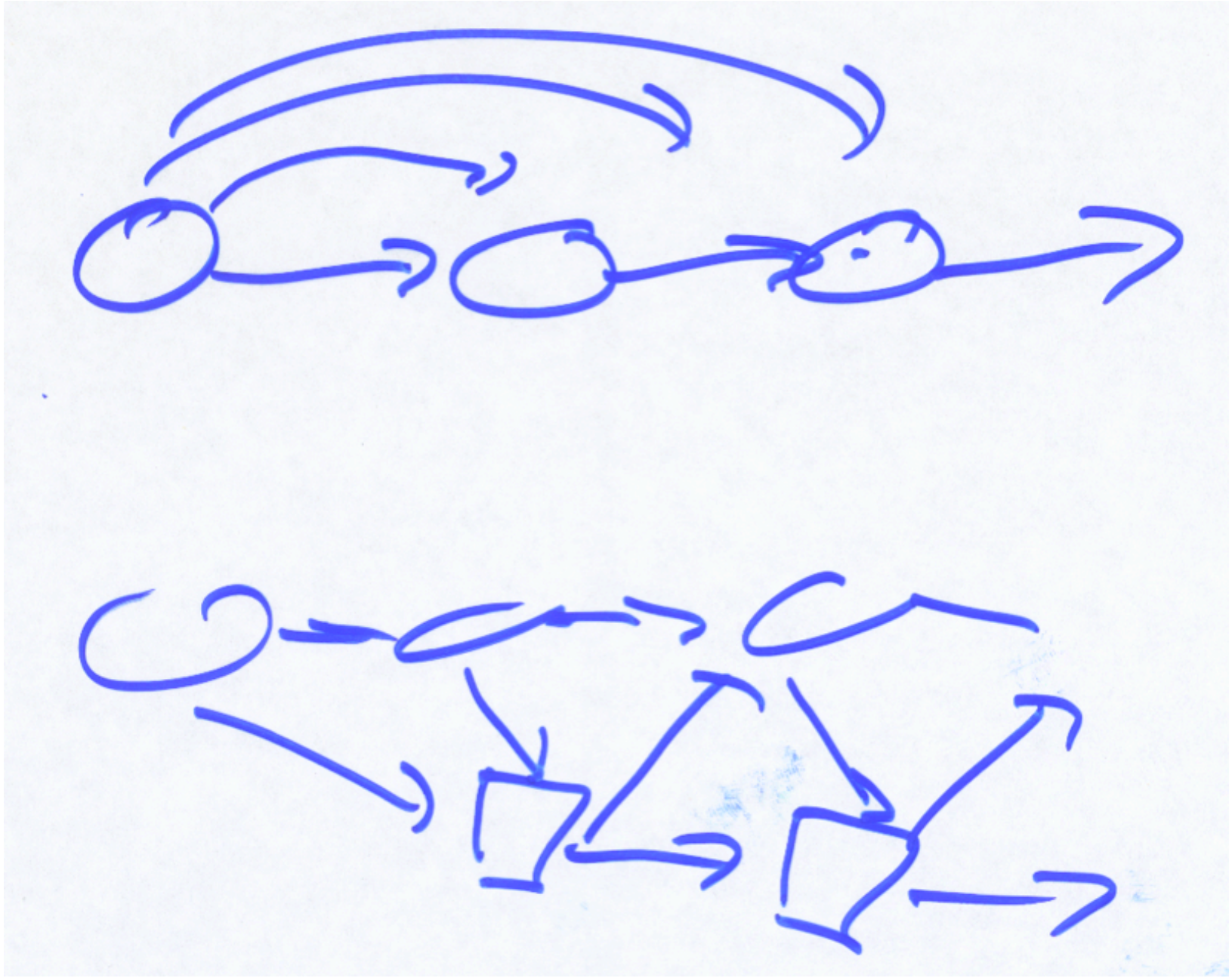
Training based on EM & F/B

Model Structure

Define structure as well as you can

$p^r(1-p)$





HMM's in Action: pfam

- Proteins fall into families, both across & within species
  Ex: Globins, GPCRs, ...

- Identifying family is very useful - suggests function, etc.

- So search & alignment are important

- One successful approach profile HMM's

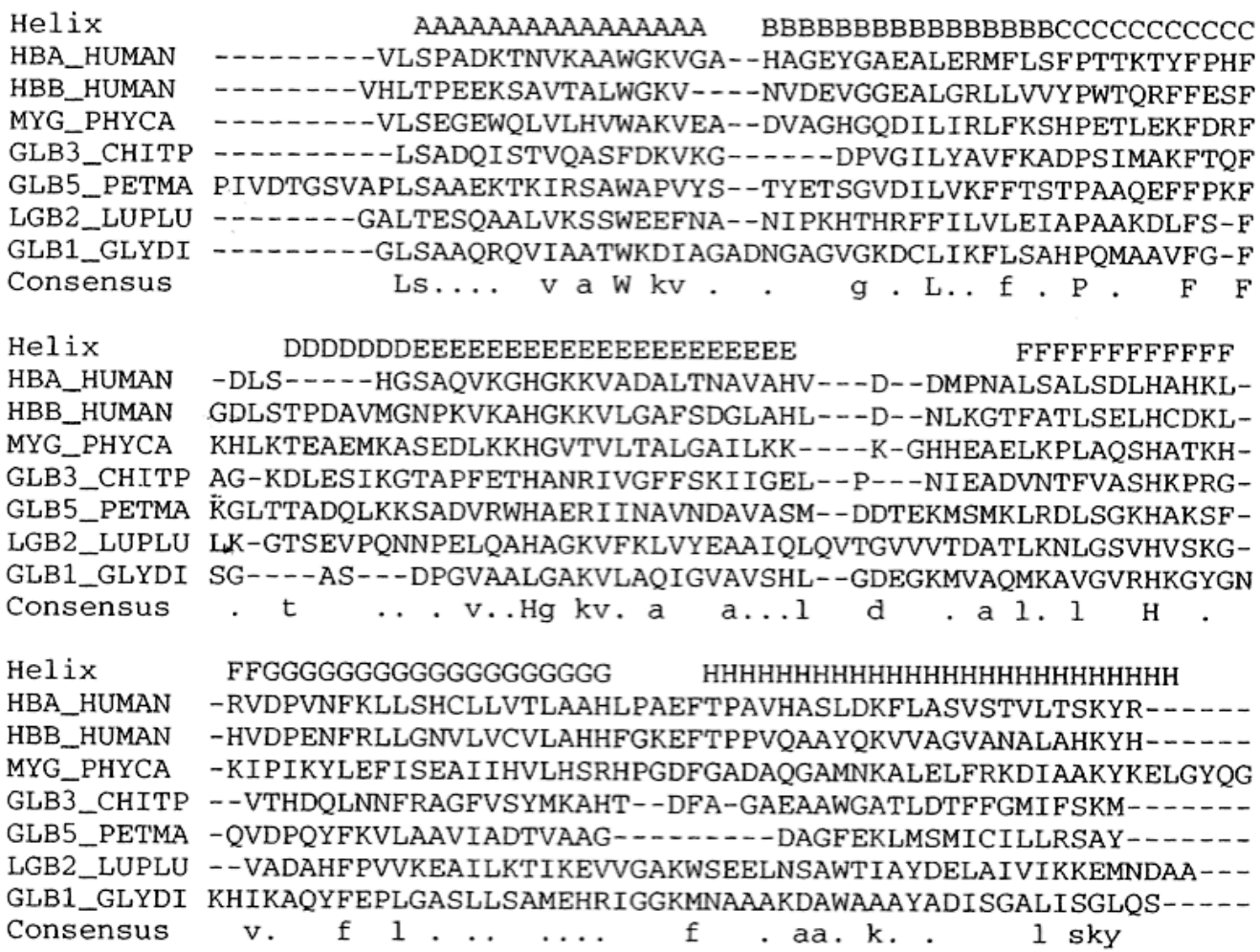


Figure 5.1 *An alignment of seven globins from Bashford, Chothia & Lesk [1987]. To the left is the protein identifier in the* SWISS-PROT *database [Bairoch & Apweiler 1997]. The eight alpha helices are shown as* A–H *above the alignment. A consensus line below the alignment indicates residues that are identical among at least six of the seven sequences in upper case, ones identical in four or five sequences in lower case, and positions where there is a residue identical in three sequences with a dot.*

Profile Hmm Structure

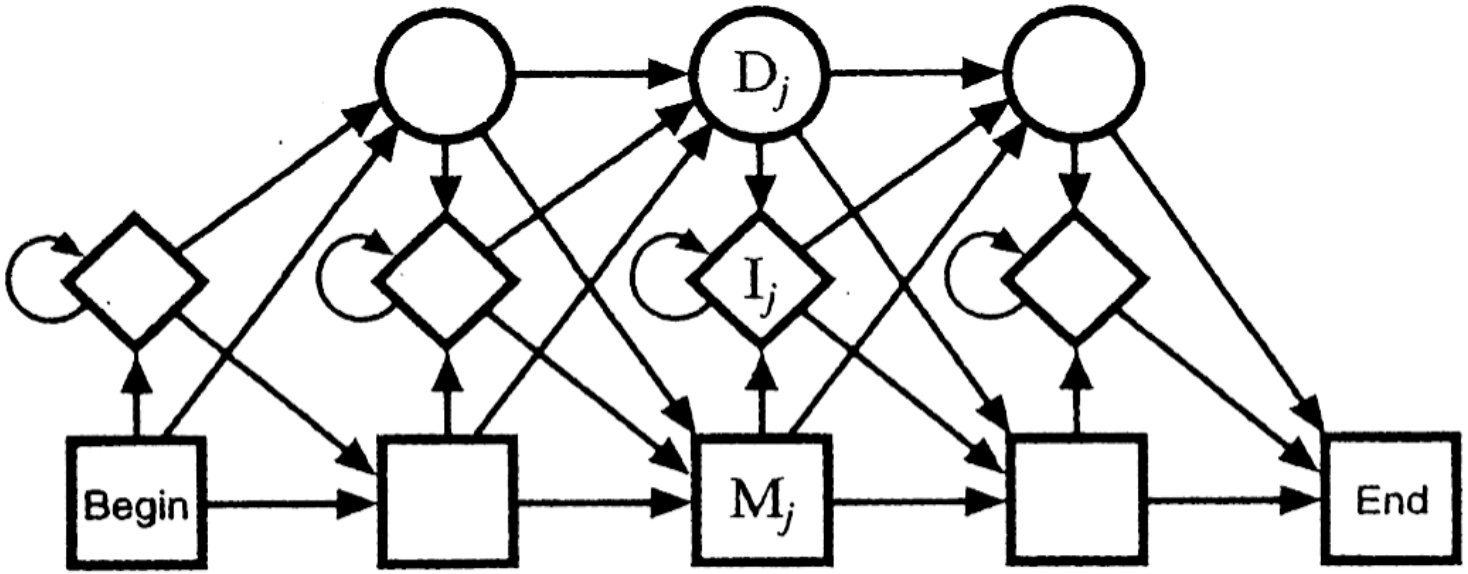


Figure 5.2 *The transition structure of a profile HMM. We use diamonds to indicate the insert states and and circles for the delete states.*

M_j: Match states (20 emission probabilities)

I_j: Insert states (Background emission probabilities)

D_j: Delete states (silent - no emission)

How Profile HMM used

- Search

  Forward or Viterbi algorithm

  Scoring -

  log likelihood (length adjusted) } See next slide

  log odds vs background

- Alignment

  Viterbi

Likelihood vs Odds Scores

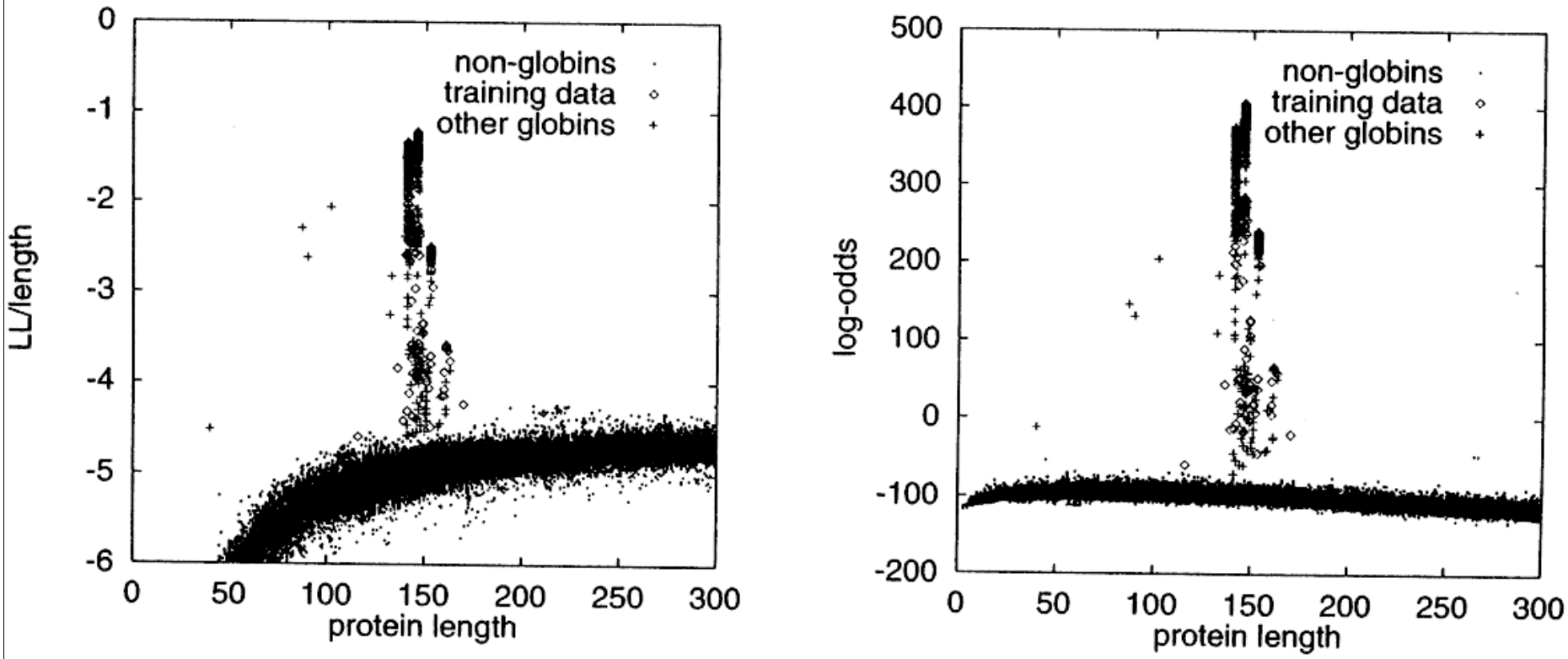


Figure 5.5 *To the left the length-normalized LL score is shown as a function of sequence length. The right plot shows the same for the log-odds score.*

Z-Scores

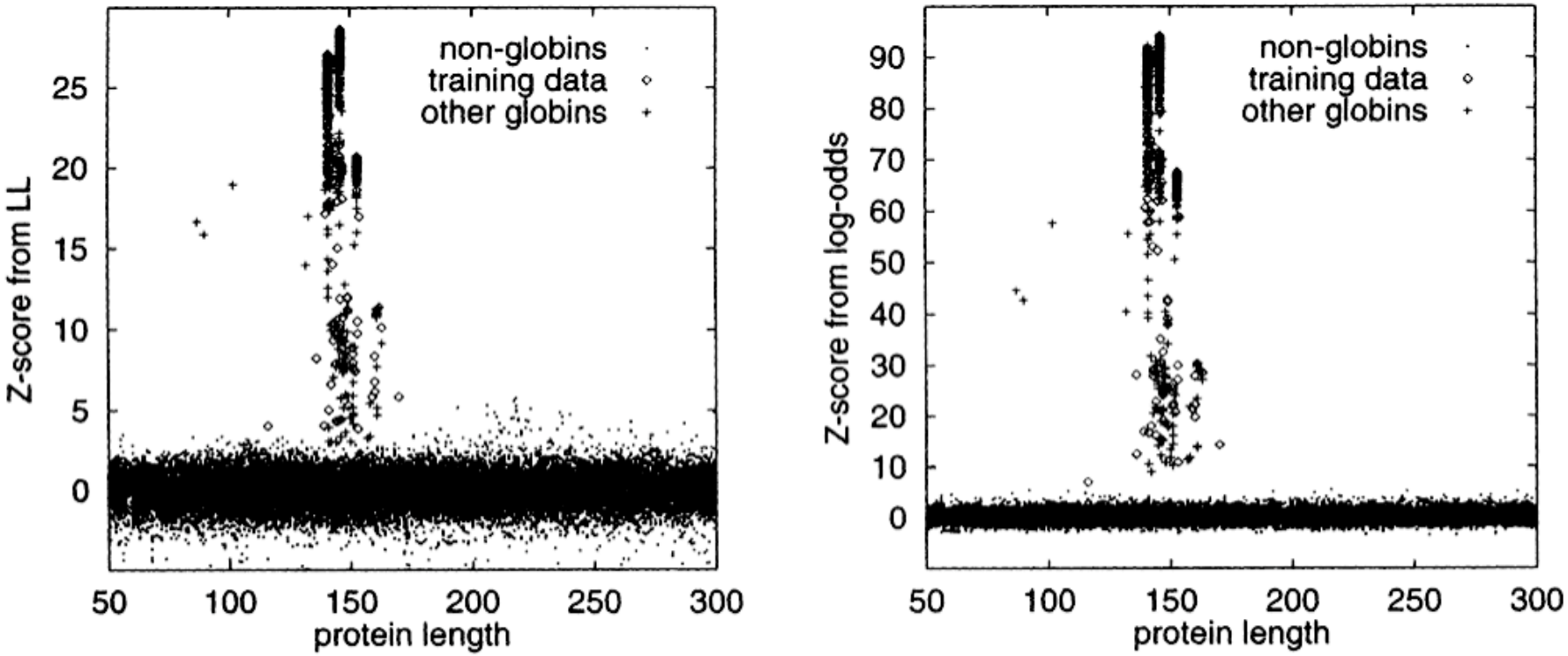


Figure 5.6 *The Z-score calculated from the LL scores (left) and the log-odds (right).*

Model Building Refinements

- Pseudocounts (count=0 common in training w/ 20 AAs)

  - eg $e_i(a) = \dfrac{c_{ia} + A \cdot q_a}{\sum_a c_{ia} + A}$

    q_a = background

    $A \sim 20$

    (~50 training)

- Pseudocount "mixtures"

  eg separate pseudocount vectors for various contexts (hydrophobic regions, buried regions ...)

  (~10-20 training)

Refinements (cont.)

- Weighting: May need to down weight highly similar sequences to reflect sampling bias, phylogenetic info, etc.

- Match-Insert Assignment

  Simple threshold, eg "$>50\%$ gap $\Rightarrow$ insert"

  May not be optimal

  Can use Forward Alg-like dyn. prog. method to compute Max a posteriori assignment