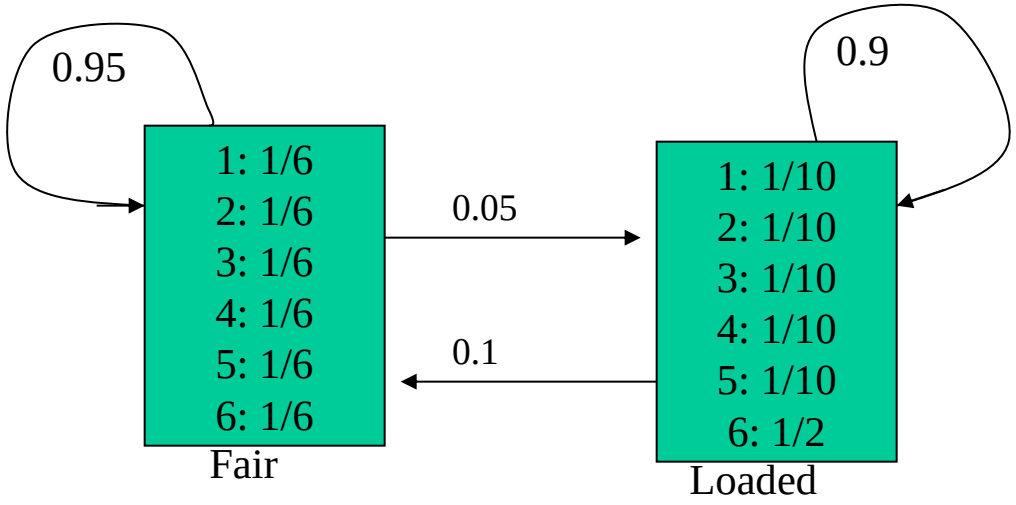
CISC 436/636 Computational Biology & Bioinformatics (Fall 2016)

Hidden Markov Models (I)

a. The model
b. The decoding: Viterbi algorithm

Hidden Markov models

- A Markov chain of states
- At each state, there are a set of possible observables (symbols), and
- The states are not directly observable, namely, they are hidden.
- E.g., Casino fraud



- Three major problems
  - Most probable state path
  - The likelihood
  - Parameter estimation for HMMs

A biological example: CpG islands

- Higher rate of Methyl-C mutating to T in CpG dinucleotides → generally lower CpG presence in genome, except at some biologically important ranges, e.g., in promoters, -- called CpG islands.
- The conditional probabilities $P_{\pm}(N|N')$ are collected from ~ 60,000 bps human genome sequences, + stands for CpG islands and – for non CpG islands.

P_+	A	C	G	T
A	.180	.274	.426	.120
C	.171	.368	**.274**	.188
G	.161	.339	.375	.125
T	.079	.355	.384	.182

P_-	A	C	G	T
A	.300	.205	.285	.210
C	.322	.298	**.078**	.302
G	.248	.246	.298	.208
T	.177	.239	.292	.292

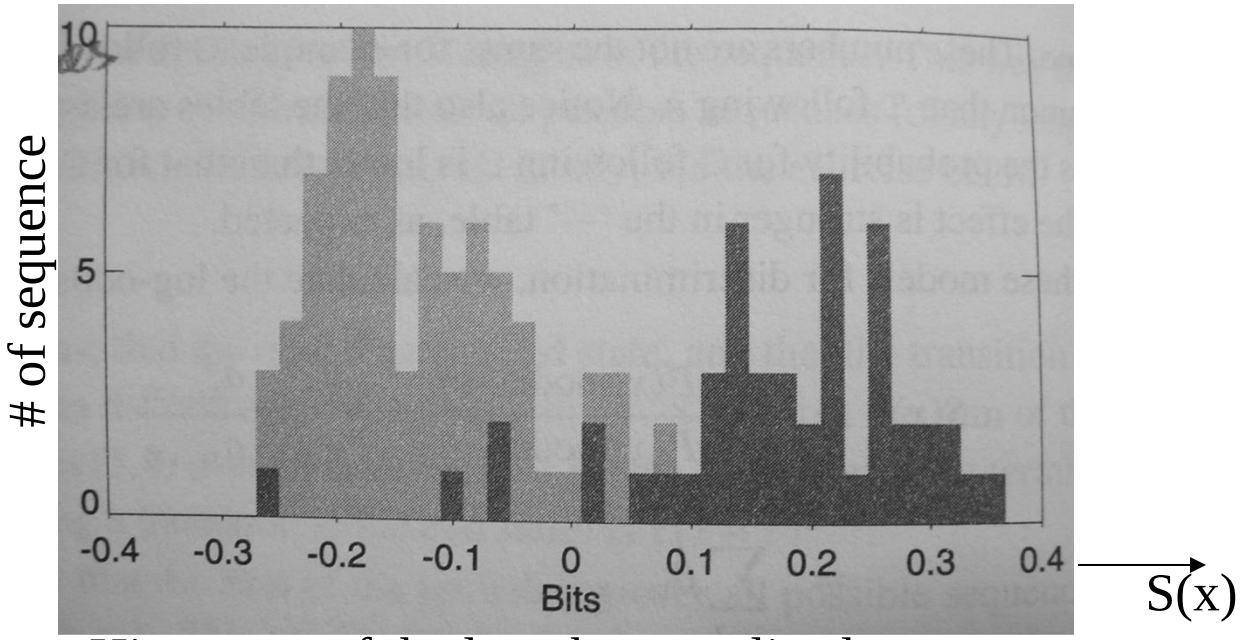
Task 1: given a sequence x, determine if it is a CpG island.

One solution: compute the log-odds ratio scored by the two Markov chains:

$$S(x) = \log [\ P(x \mid \text{model } +) / P(x \mid \text{model } -)]$$

where $P(x \mid \text{model } +) = P_+(x_2|x_1)\ P_+(x_3|x_2)\ldots\ P_+(x_L|x_{L-1})$ and

$P(x \mid \text{model } -) = P_-(x_2|x_1)\ P_-(x_3|x_2)\ldots\ P_-(x_L|x_{L-1})$



Histogram of the length-normalized scores
(CpG sequences are shown as dark shaded)

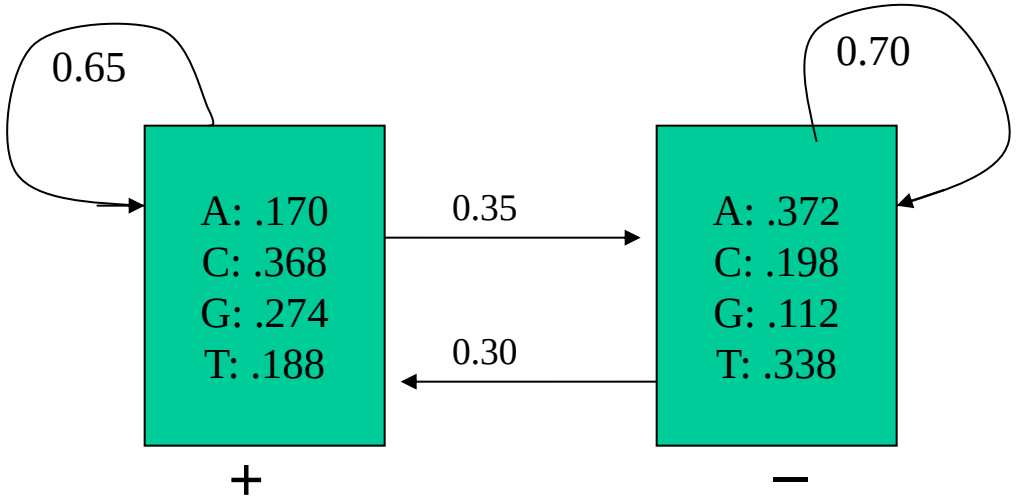
Task 2: For a *long* genomic sequence x, label these CpG islands, if there are any.

Approach 1: Adopt the method for Task 1 by calculating the log-odds score for a window of, say, 100 bps around every nucleotide and plotting it.

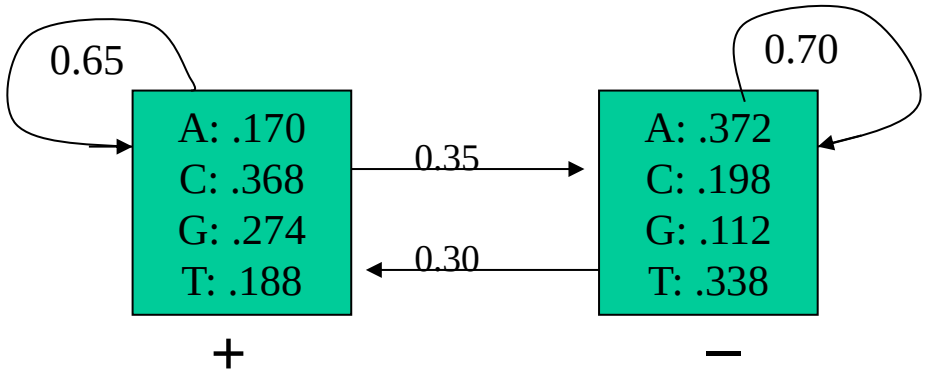
Problems with this approach:

- Won't do well if CpG islands have sharp boundary and variable length
- No effective way to choose a good Window size.

Approach 2: using hidden Markov model



- The model has two states, "+" for CpG island and "-" for non CpG island. Those numbers are made up here, and shall be fixed by learning from training examples.
- The notations: a_{kl} is the transition probability from state k to state l; $e_k(b)$ is the emission frequency – probability that symbol b is seen when in state k.



The probability that sequence x is emitted by a state path π is:

$$P(x, \pi) = \prod_{i=1 \text{ to } L} e_{\pi i}(x_i)\, a_{\pi i\, \pi i+1}$$

```
i:123456789
x:TGCGCGTAC
π:--++++---
```

$P(x, \pi) = 0.338 \times 0.70 \times 0.112 \times 0.30 \times 0.368 \times 0.65 \times 0.274 \times 0.65 \times 0.368 \times 0.65 \times 0.274 \times 0.35 \times 0.338 \times 0.70 \times 0.372 \times 0.70 \times 0.198.$

Then, the probability to observe sequence x in the model is

$$P(x) = \Sigma_{\pi} P(x, \pi),$$

which is also called the likelihood of the model.

Decoding: Given an observed sequence x, what is the most probable state path, i.e.,

$$\pi^* = \text{argmax}_{\pi} P(x, \pi)$$

Q: Given a sequence x of length L, how many state paths do we have?
A: N^L, where N stands for the number of states in the model.
As an exponential function of the input size, it precludes enumerating all possible state paths for computing P(x).

Let $v_k(i)$ be the probability for the most probable path ending at position i with a state k.

Viterbi Algorithm

Initialization: $v_0(0) = 1$, $v_k(0) = 0$ for $k > 0$.

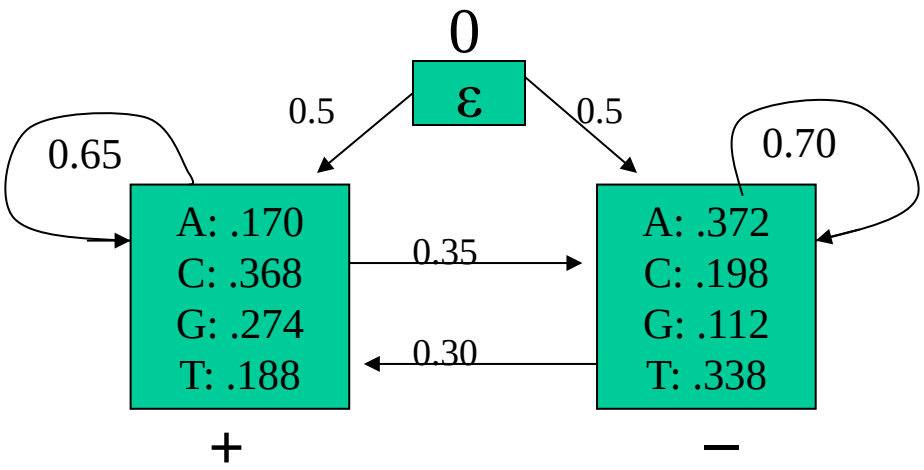
Recursion: $v_k(i) = e_k(x_i) \max_j (v_j(i-1)\, a_{jk})$;

$ptr_i(k) = \text{argmax}_j (v_j(i-1)\, a_{jk})$;

Termination: $P(x, \pi^*) = \max_k (v_k(L)\, a_{k0})$;

$\pi^*_L = \text{argmax}_j (v_j(L)\, a_{j0})$;

Traceback: $\pi^*_{i-1} = ptr_i(\pi^*_i)$.



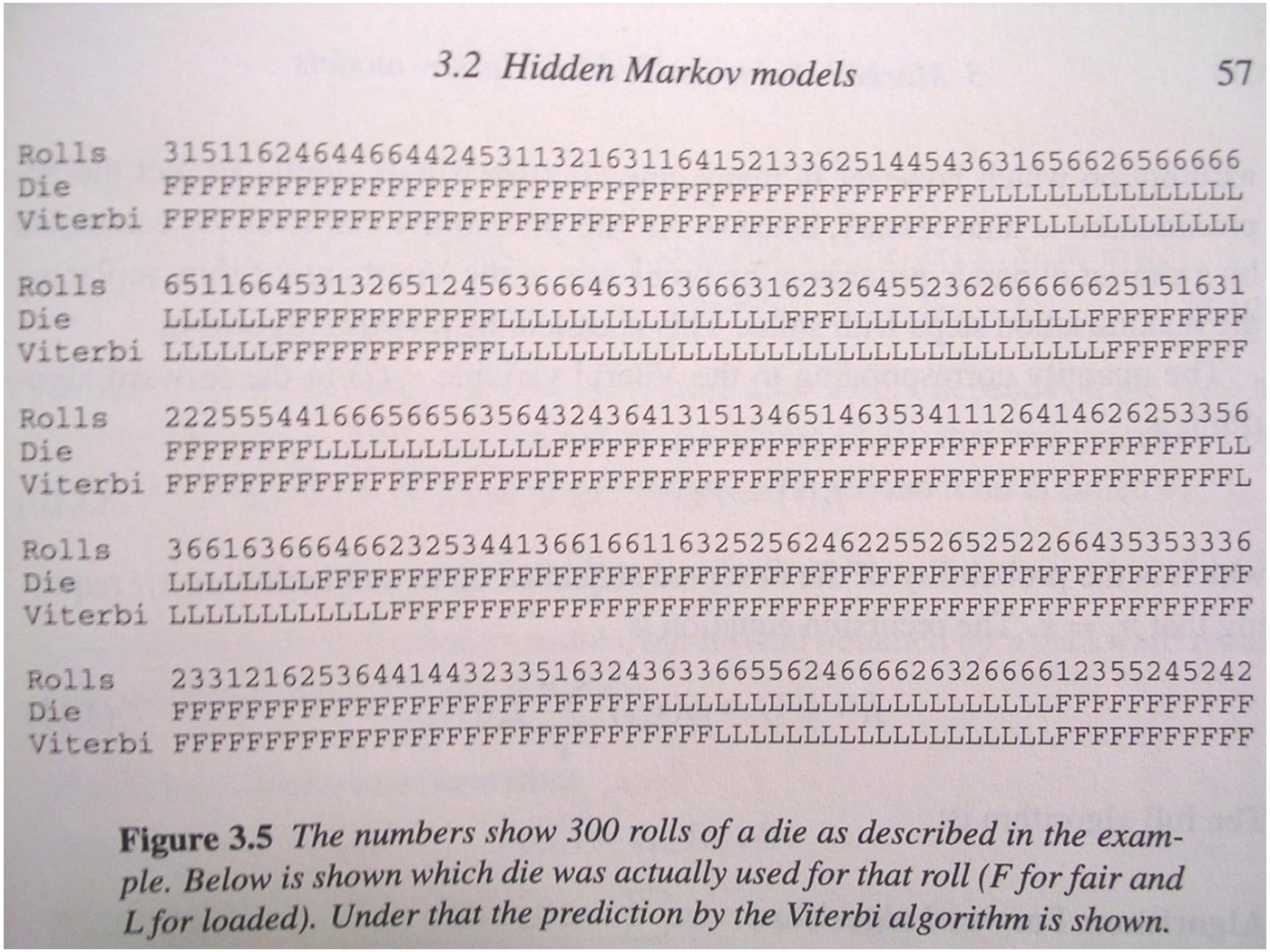
$v_k(i) = e_k(x_i) \max_j (v_j(i-1)\, a_{jk})$;

$V_k(i)$	ε	A	A	G	C	G
0	1	0	0	0	0	0
–	0	.186	.048	.0038	.00053	.000042
+	0	.085	.0095	.0039	.00093	.00038
k		–	–	+	+	+

i

hidden state path

Casino Fraud: investigation results by Viterbi decoding




```
Rolls    315116246446644245311321631164152133625144543631656626566666
Die      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLL

Rolls    651166453132651245636664631636663162326455236266666625151631
Die      LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLFFFFFFFFF
Viterbi  LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF

Rolls    222555441666566563564324364131513465146353411126414626253356
Die      FFFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLL
Viterbi  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls    366163666466232534413661661163252562462255265252266435353336
Die      LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi  LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Rolls    233121625364414432335163243633665562466662632666612355245242
Die      FFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFF
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 log transformation for Viterbi algorithm

$v_k(i) = e_k(x_i) \max_j (v_j(i-1)\, a_{jk})$;

$\underline{a}_{jk} = \log a_{jk}$;
$\underline{e}_k(x_i) = \log e_k(x_i)$;
$\underline{v}_k(i) = \log v_k(i)$;

$\underline{v}_k(i) = \underline{e}_k(x_i) + \max_j (\underline{v}_j(i-1) + \underline{a}_{jk})$;