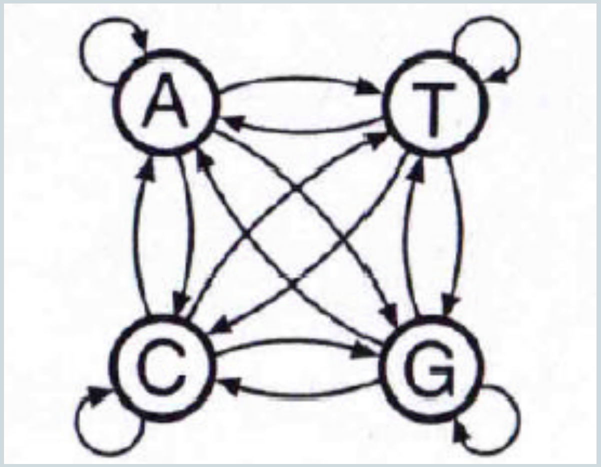
Lecture Thu 3.11.

MARKOV CHAINS AND HIDDEN MARKOV MODELS

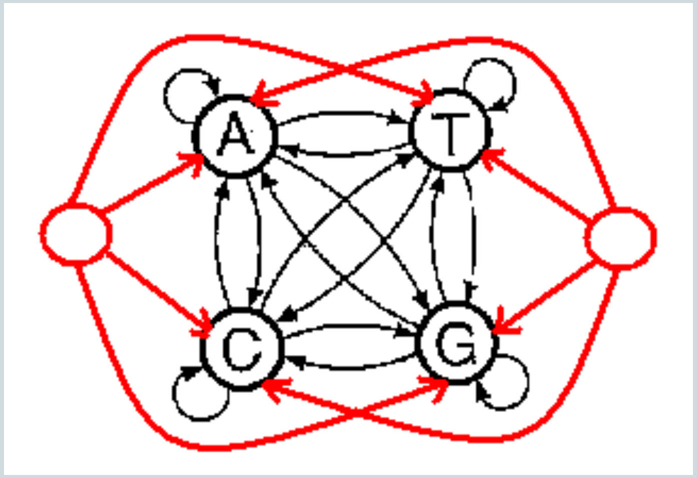
Markov chain as a probabilistic finite state machine

- Markov chains can be represented as probabilistic finite state machines (or automatons)
  - There is a state corresponding to each symbol
  - When the state is entered the corresponding symbolis printed out
- Transitions between states are taken according the transition probabilities


$$\begin{tikzcd}
A \arrow[loop above] \arrow[r, bend left] \arrow[d, bend left] \arrow[dr] & T \arrow[loop above] \arrow[l, bend left] \arrow[d, bend left] \arrow[dl] \\
C \arrow[loop below] \arrow[u, bend left] \arrow[r, bend left] \arrow[ur] & G \arrow[loop below] \arrow[l, bend left] \arrow[u, bend left] \arrow[ul]
\end{tikzcd}$$

Markov chain as a probabilistic finite state machine

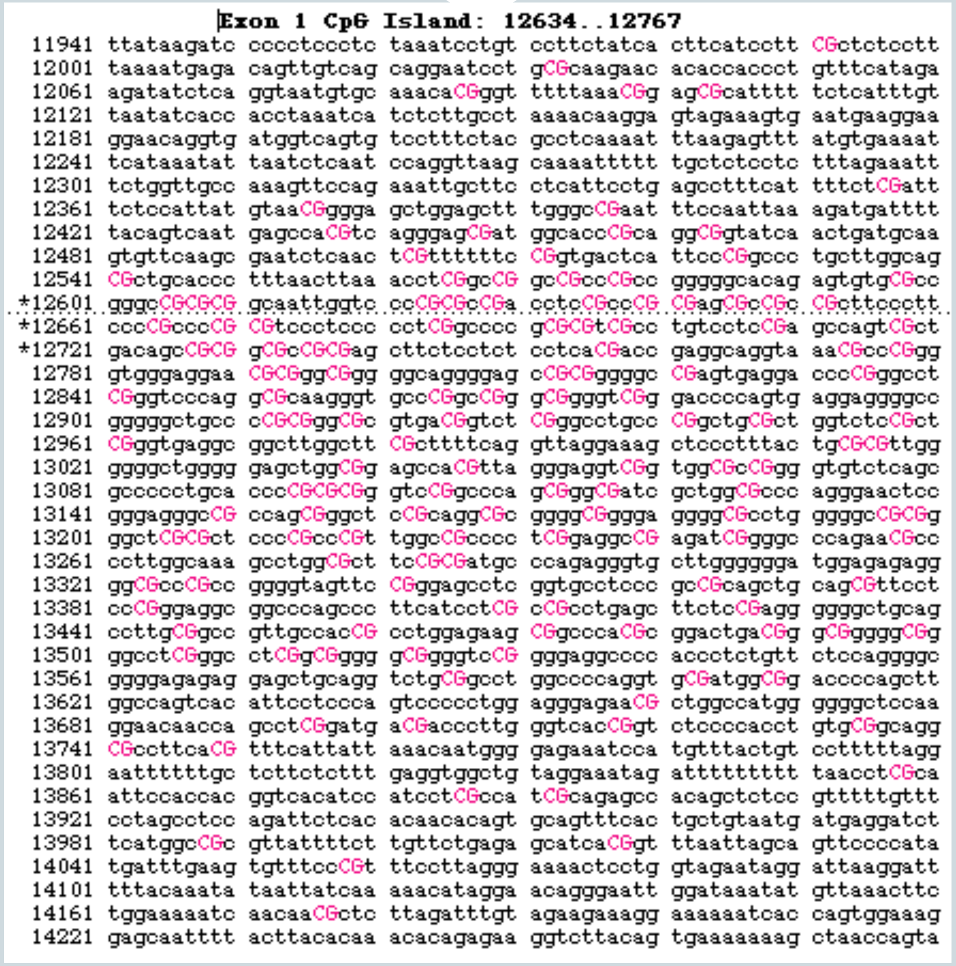
- It is sometimes convenient to add special start and end states
- Chain always begins from the start state
- The transition probabilities from start to normal states can be set as uniform (here 0.25) or some prior probabilities (e.g. base frequencies)



Example: CpG Islands

- **CpG** dinucleotides are rarer than would be expected from the independent probabilities of **C** and **G.**
  - Note: the notation CpG denotes a dinucleotide along a single strand of DNA, do not confuse with C-G base pairing which goes across two strands
- Biological explanation: When **CpG** occurs, C is typically chemically modified by methylation and there is a relatively high chance of **methyl-C** mutating into **T**
- High CpG frequency may be biologically significant; e.g., may signal promoter region ("start" of a gene).
- A **CpG island** is a region where **CpG** dinucleotides are much more abundant than elsewhere.

Example: CpG island



```
                    Exon 1 CpG Island: 12634..12767
 11941 ttataagatc cccctccctc taaatcctgt ccttctatca cttcatcctt CGctctcctt
 12001 taaaatgaga cagttgtcag caggaatcct gCGcaagaac acaccaccct gtttcataga
 12061 agatatctca ggtaatgtgc aaacaCGggt ttttaaaCGg agCGcatttt tctcatttgt
 12121 taatatcacc acctaaatca tctcttgcct aaaacaagga gtagaaagtg aatgaaggaa
 12181 ggaacaggtg atggtcagtg tcctttctac gcctcaaaat ttaagagttt atgtgaaaat
 12241 tcataaatat taatctcaat ccaggttaag caaaattttt tgctctcctc tttagaaatt
 12301 tctggttgcc aaagttccag aaattgcttc ctcattcctg agcctttcat tttctCGatt
 12361 tctccattat gtaaCGggga gctggagctt tgggcCGaat ttccaattaa agatgatttt
 12421 tacagtcaat gagccaCGtc agggagCGat ggcaccCGca ggCGgtatca actgatgcaa
 12481 gtgttcaagc gaatctcaac tCGttttttc CGgtgactca ttccCGgccc tgcttggcag
 12541 CGctgcaccc tttaacttaa acctCGgcCG gcCGccCGcc gggggcacag agtgtgCGcc
*12601 gggcCGCGCG gcaattggtc ccCGCGcCGa cctcCGccCG CGagCGcCGc CGcttccctt
*12661 cccCGcccCG CGtccctccc cctCGgcccc gCGCGtCGcc tgtcctcCGa gccagtCGct
*12721 gacagcCGCG gCGcCGCGag cttctcctct cctcaCGacc gaggcaggta aaCGccCGgg
 12781 gtgggaggaa CGCGggCGgg ggcaggggag cCGCGggggc CGagtgagga cccCGggcct
 12841 CGggtcccag gCGcaagggt gccCGgcCGg gCGgggtCGg gaccccagtg aggaggggcc
 12901 gggggctgcc cCGCGggCGc gtgaCGgtct CGggcctgcc CGgctgCGct ggtctcCGct
 12961 CGggtgaggc ggcttggctt CGcttttcag gttaggaaag ctccctttac tgCGCGttgg
 13021 ggggctgggg gagctggCGg agccaCGtta gggaggtCGg tggCGcCGgg gtgtctcagc
 13081 gccccctgca cccCGCGCGg gtcCGgccca gCGggCGatc gctggCGccc agggaactcc
 13141 gggagggcCG ccagCGggct cCGcaggCGc ggggCGggga ggggCGcctg ggggcCGCGg
 13201 ggctCGCGct cccCGccCGt tggcCGcccc tCGgaggcCG agatCGgggc ccagaaCGcc
 13261 ccttggcaaa gcctggCGct tcCGCGatgc ccagagggtg cttgggggga tggagagagg
 13321 ggCGccCGcc ggggtagttc CGggagcctc ggtgcctccc gcCGcagctg cagCGttcct
 13381 ccCGggaggc ggcccagccc ttcatcctCG cCGcctgagc ttctcCGagg ggggctgcag
 13441 ccttgCGgcc gttgccacCG cctggagaag CGgcccaCGc ggactgaCGg gCGggggCGg
 13501 ggcctCGggc ctCGgCGggg gCGgggtcCG gggaggcccc accctctgtt ctccaggggc
 13561 ggggagagag gagctgcagg tctgCGgcct ggccccaggt gCGatggCGg accccagctt
 13621 ggccagtcac attcctccca gtccccctgg agggagaaCG ctggccatgg ggggctccaa
 13681 ggaacaacca gcctCGgatg aCGacccttg ggtcacCGgt ctccccacct gtgCGgcagg
 13741 CGccttcaCG tttcattatt aaacaatggg gagaaatcca tgtttactgt ccttttttagg
 13801 aatttttttgc tcttctcttt gaggtggctg taggaaatag attttttttt taacctCGca
 13861 attccaccac ggtcacatcc atcctCGcca tCGcagagcc acagctctcc gttttgttt
 13921 cctagcctcc agattctcac acaacacagt gcagtttcac tgctgtaatg atgaggatct
 13981 tcatggcCGc gttattttct tgttctgaga gcatcaCGgt ttaattagca gttccccata
 14041 tgatttgaag tgtttccCGt ttccttaggg aaaactcctg gtagaatagg attaaggatt
 14101 tttacaaata taattatcaa aaacatagga acagggaatt ggataaatat gttaaacttc
 14161 tggaaaaatc aacaaCGctc ttagatttgt agaagaaagg aaaaaatcac cagtggaaag
 14221 gagcaatttt acttacacaa acacagagaa ggtcttacag tgaaaaaaag ctaaccagta
```

Two problems

1. Given a short genome sequence, decide if it comes from a CpG islands or not.
2. Given a long DNA sequence, locate all the CpG islands in it.

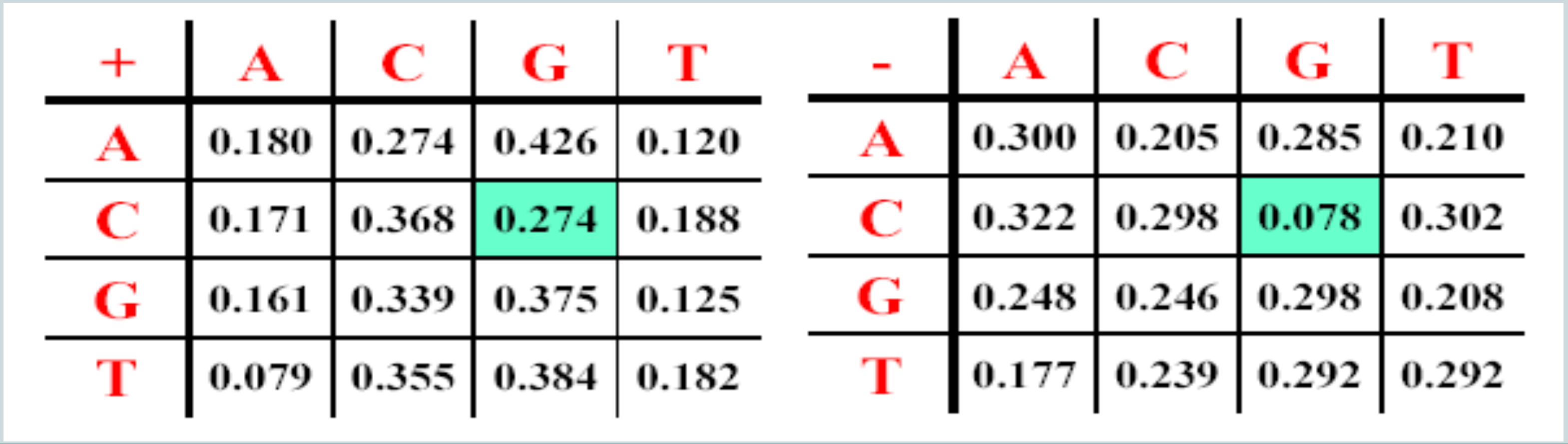
Modelling CpG islands with Markov chains

- Problem 1:Given a short genome sequence, decide if it comes from a CpG islands or not.
- Markov chain modelling approach:
  - Pick a set of known CpG islands and build a first order Markov chain (transition table) from the sequences: "+ model"
  - Pick a set of non- CpG island sequences and build a first order Markov chain (transition table) from them: "- model"
- Transition probabilities are obtained by counting dinucleotide frequencies
  - c^{+}_{st} is the frequency of 'st' in the sequence
  - a^{+}_{st} denotes the transition probability $s \rightarrow t$

$$\boxed{a^{+}_{st} = \frac{c^{+}_{st}}{\sum_{t'} c^{+}_{st'}}}$$

Modelling CpG islands with Markov chains

- Problem 1:Given a short genome sequence, decide if it comes from a CpG islands or not.
- Markov chain modelling approach:
  - Pick a set of known CpG islands and build a first order Markov chain (transition table) from the sequences: "+ model"
  - Pick a set of non- CpG island sequences and build a first order Markov chain (transition table) from them: "- model"



+	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

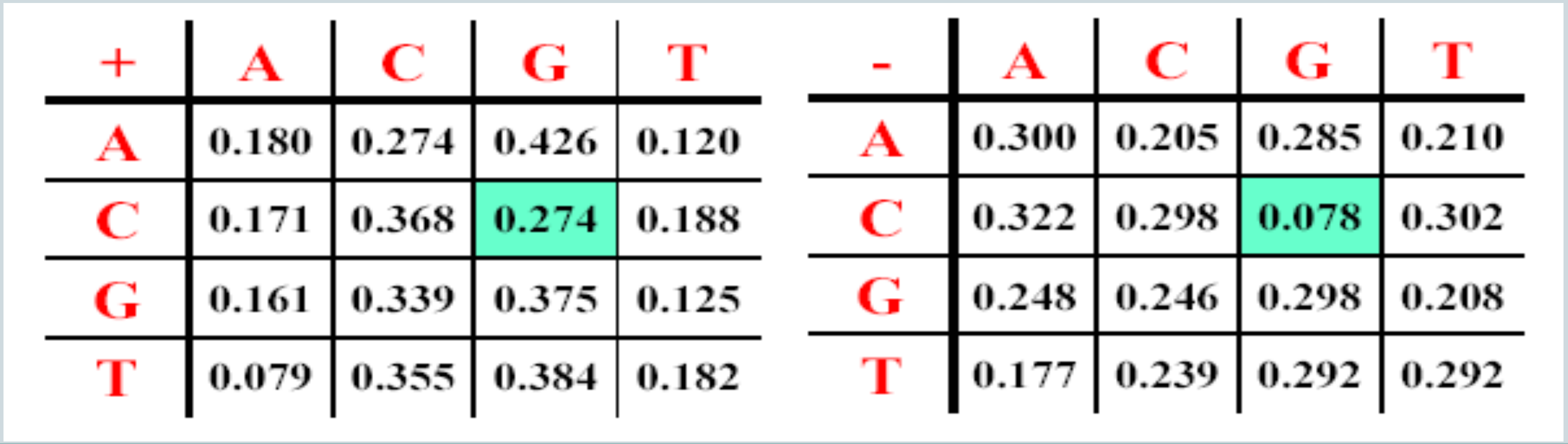
-	A	C	G	T
A	0.300	0.205	0.285	0.210
C	0.322	0.298	0.078	0.302
G	0.248	0.246	0.298	0.208
T	0.177	0.239	0.292	0.292

Modelling CpG islands with Markov chains

- Compute the probability of the new sequence $x_1 \ldots x_L$ using both models

$$P(x) = P(x_L \mid x_{L-1})\, P(x_{L-1} \mid x_{L-2}) \ldots (x_2 \mid x_1) P(x_1)$$

$$= P(x_1) \prod_{i=2}^{L} a_{x_{i-1} x_i}$$



+	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

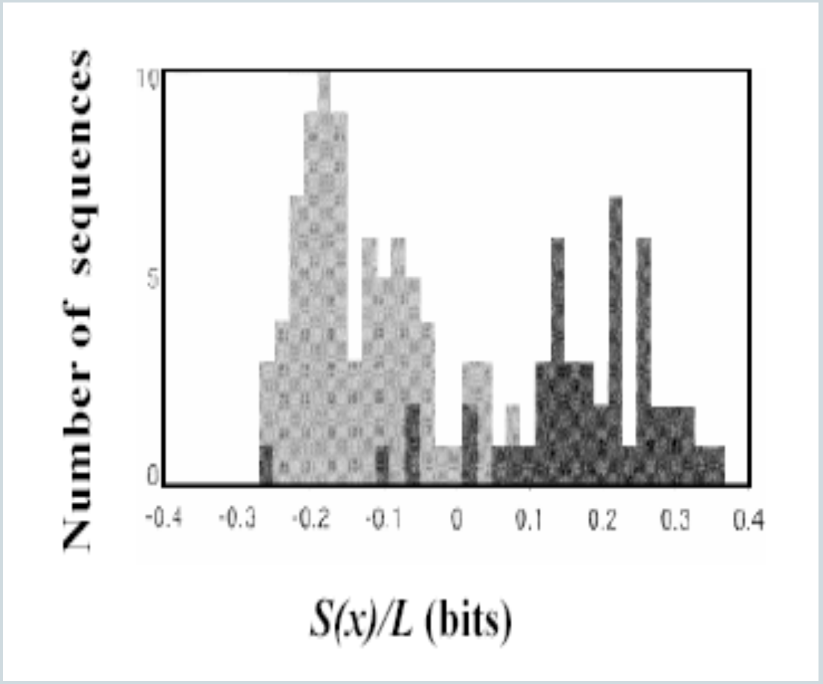
Modelling CpG islands with Markov chains

- Given the two probabilites P(x|model +) and P(x| model -), we compute a log-odds score S(x) to reflect the relative goodness of the models
- If S(x) > 0 it is the more likely the sequence comes from a CpG island than not

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

Modelling CpG islands with Markov chains

- The S(x) scores for a set of CpG-island and non CpG-island sequences are shown
  - Normalized by sequence length to get an average score per nucleotide
- CpG islands sequences shown in dark grey and non-CpG sequences in light grey
- Assigning sequences with S (x)/L > 0 as CpG islands would give a good but not perfect classification



Two problems

1. Given a short genome sequence, decide if it comes from a CpG islands or not.✔
2. Given a long DNA sequence, locate all the CpG islands in it.

Locating CpG islands with Markov chains

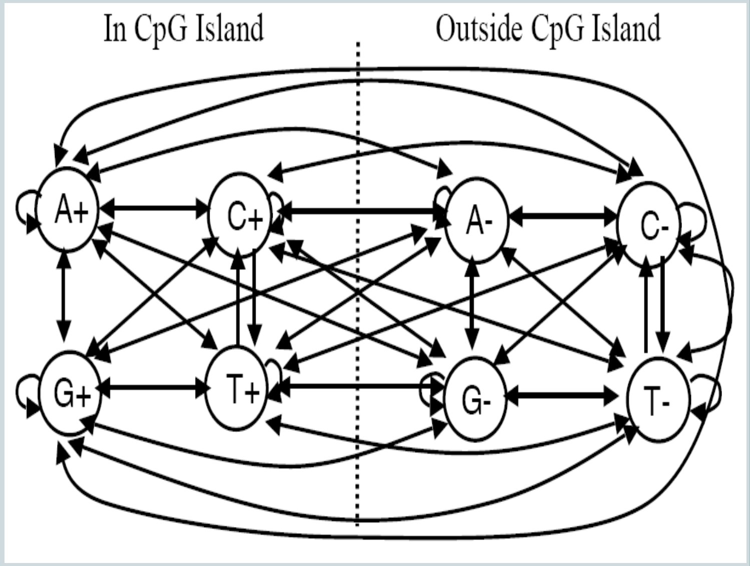
- Problem 2: Given a long DNA sequence, locate all the CpG islands in it
- The Markov chain scheme does not give any indication of
  - Where the CpG island starts
  - The length of the island
- As sliding window approach is possible:
  - Slide a window (x_k,…,x_{k+l}) over the long sequence, k=1…L
  - Compute the S(x) score from each window
  - GpG islands would then possibly stand out as regions with positive S(x) scores computed from the windows

Locating CpG islands with Markov chains

- Window approach not completely satisfactory:
  - with fixed window length, we could not properly model the variable length CpG islands
  - e.g. islands much shorter than the window length could be missed
  - No direct predictions of where the island starts and ends

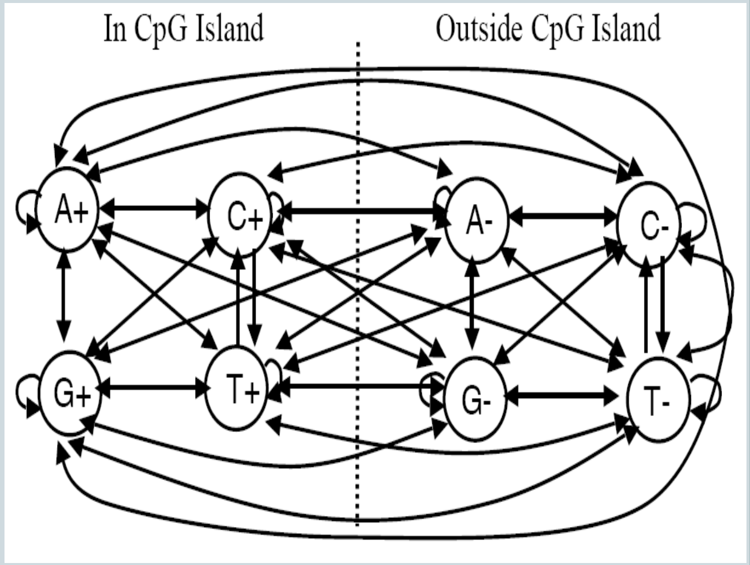
Locating CpG islands with Markov chains

- A better approach would be to build a single model that incorporates both the CpG island and the non-CpG island models
- We have 8 states (A_+,C_+,…), 4 for both models, with all pairwise transitions possible
- In addition, transitions between the two parts are possible with small probability (edges across the vertical line)



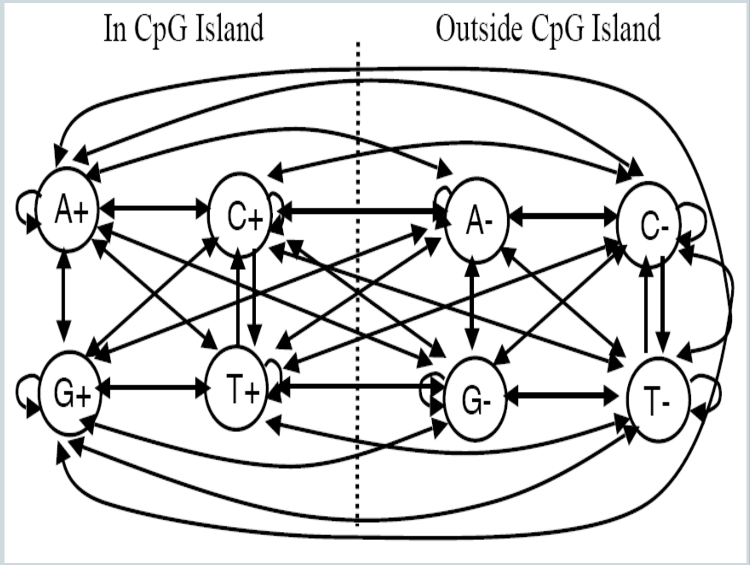
Locating CpG islands with Markov chains

- Transition probabilities within the '+' part of the model are set close to the original CpG island model, '-' part set close to the '-' model
- The probabilities of any transition from '+' to '-' state are set higher on average than vice versa
  - Model is more likely to spend time on the '-' part than '+' part



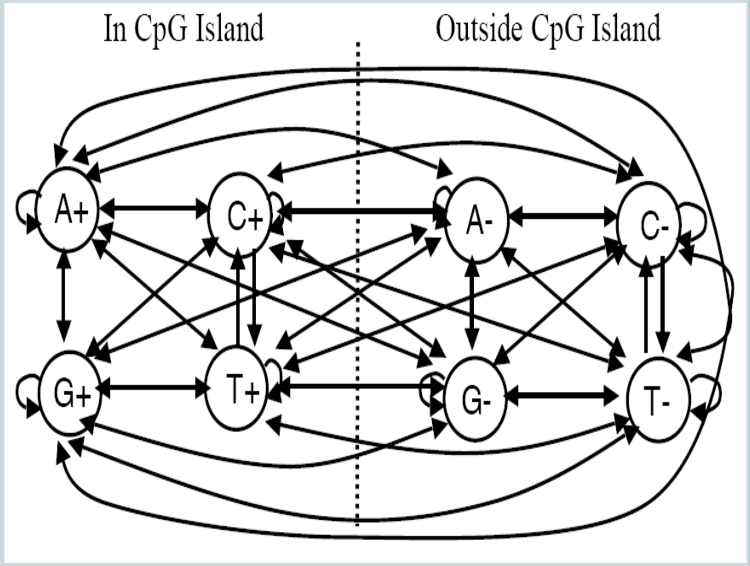
Locating CpG islands with Markov chains

- Transition probabilities within the '+' part of the model are set close to the original CpG island model, '-' part set close to the '-' model
- The probabilities of any transition from '+' to '-' state are set higher on average than vice versa
  - Model is more likely to spend time on the '-' part than '+' part



Towards Hidden Markov Models

- The model outputs nucleotide A both when in A_- and A_+ states
- Thus by looking at the generated symbol sequence alone, we cannot directly tell if '+' model or the '-' minus model was used to generate or *emit* any given symbol
  - The state is said to be hidden



State:	A_+ , C_+ , G_+ , T_+ , A- , C-, G- , T-
Emitted character:	A C G T A C G T

Hidden Markov Model

- A Hidden Markov Model is composed of the following components
  - Set of (hidden) states, capable of emitting symbols according to a probability distribution
  - Set of transitions between the states, with transition probabilities
- Two kinds of sequences:
  - State sequence (hidden) $\Pi = (\pi_1 ,\ldots, \pi_L)$ called the *path*
  - Symbol sequence (observed): $(x_1 ,\ldots.,x_L)$

Hidden Markov Models

- The probability of a state only depends on the previous state (Markov assumption)
  - $a_{kl} = P(\pi_i = l \mid \pi_{i-1} = k)$
- The probability of emitting a symbol only depends on the current state *k*
  - $e_k(b) = P(x_i = b \mid \pi_i = k)$
  - In particular, emitting a symbol does not depend on the previously emitted symbol x_{i-1}

Hidden Markov Models

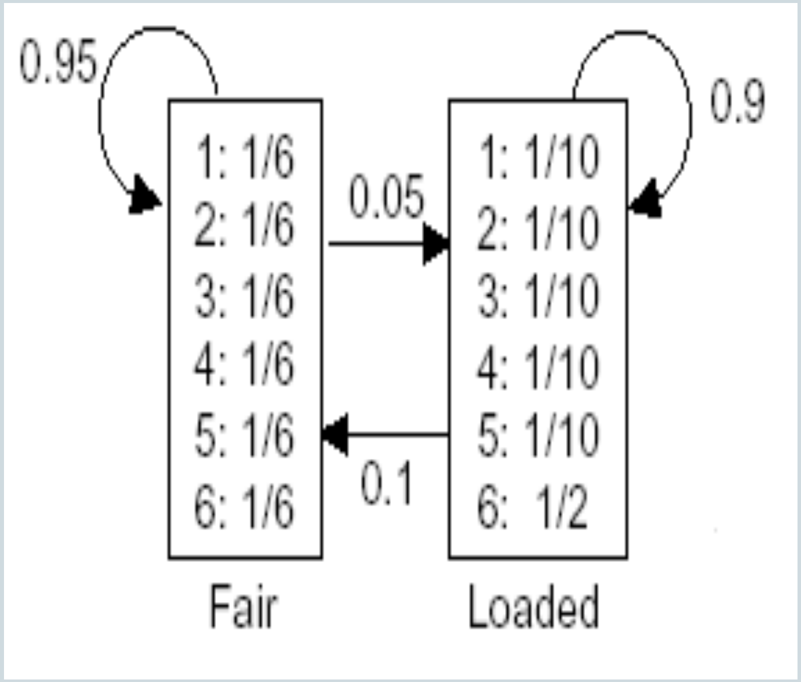
- The probability that the sequence x is generated given the path Π is

$$P(x,\pi) = a_{\pi_0,\pi_1} \prod_{i=1}^{L} e_{\pi_i}(x_i)\, a_{\pi_i,\pi_{i+1}}$$

- Above we denote : π_0 = begin and π_{L+1} = end

Example: occasionally dishonest casino

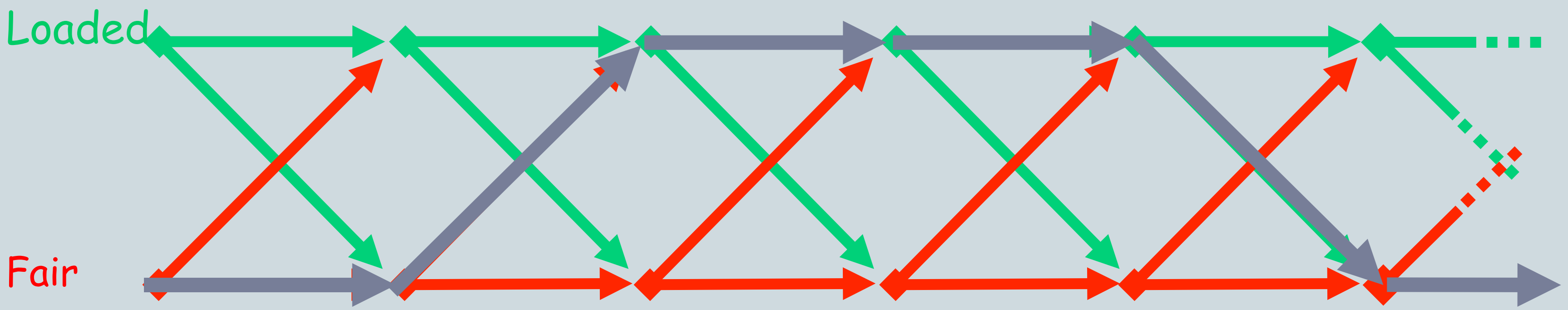
- Casino uses a fair die most of the time, but switches to the loaded die once in a while
- Can we detect which of the dice is in use at any given time, just by observing the sequence of rolls?



Sequential view

Observed sequence of die rolls:

$$\begin{tikzcd} 1 \arrow[r] & 4 \arrow[r] & 3 \arrow[r] & 6 \arrow[r] & 6 \arrow[r] & 4 \arrow[r] & {} \end{tikzcd}$$

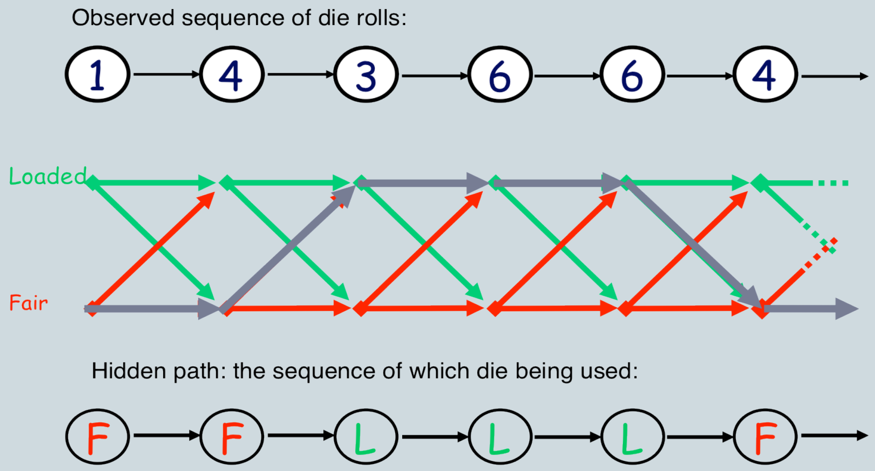


Hidden path: the sequence of which die being used:

$$\begin{tikzcd} F \arrow[r] & F \arrow[r] & L \arrow[r] & L \arrow[r] & L \arrow[r] & F \arrow[r] & {} \end{tikzcd}$$

Decoding: finding the most probable path

- How can we make good guesses when the casino has switched to the loaded die?
- Decoding: Finding the most probable state sequence (path π) to have generated the observed rolls
- The set of possible paths (Π) is exponential sized, so need efficient algorithms

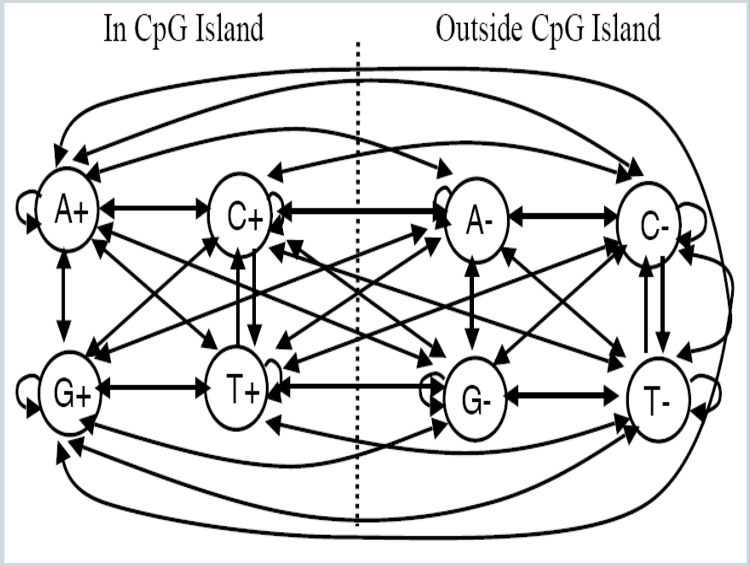

Observed sequence of die rolls:
1
4
3
6
6
4
Loaded
Fair
Hidden path: the sequence of which die being used:
F
F
L
L
L
F

$$\pi^* = \arg\max_{\pi \in \Pi} P(x,\pi)$$

$$P(x,\pi) = a_{\pi_0,\pi_1} \prod_{i=1}^{L} e_{\pi_i}(x_i) a_{\pi_i,\pi_{i+1}}$$

CpG island example

- Consider an observed sequence CGCG
- Many different state sequences can generate it, e.g.
  - (C+,G+,C+,G+)
  - (C-,G-,C-,G-)
  - (C+,G-,C+,G-)
- However, they do so with very different probabilities.
- Which is the most probable path?



$$\pi^* = \arg\max_{\pi \in \Pi} P(x,\pi)$$

Viterbi algorithm

- Assume we know the probability $v_k(i)$ of the most probable path $(\pi_0\pi_1\ldots\pi_i)$ ending at state k for the prefix $x_1,\ldots,x_i$
- Then the most probable path ending in state l for the extended prefix $x_1,\ldots,x_i,x_{i+1}$ is found by finding a state k that maximizes the combined probability of
  - Taking the best path to k $(\pi_0\pi_1\ldots k)$, probability $v_k(i)$
  - Making a transition from k to l, probability a_{kl}
- Combine with the probability of Emitting x_{i+1} in state l to get the probability of the path

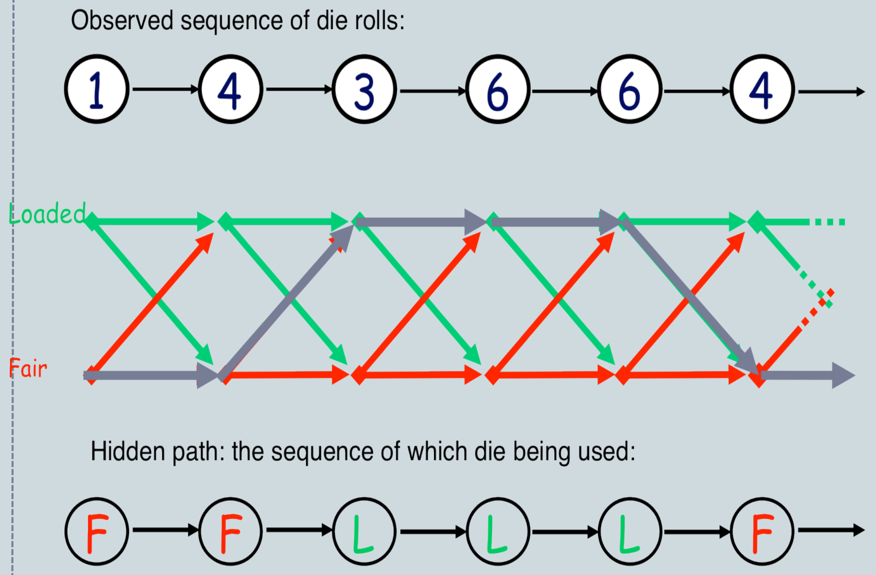
$$v_l(i+1) = e_l(x_{i+1}) \max_k v_k(i) a_{kl}$$

$$\pi^* = \arg\max_{\pi \in \Pi} \mathrm{P}(x,\pi)$$

Viterbi at the casino

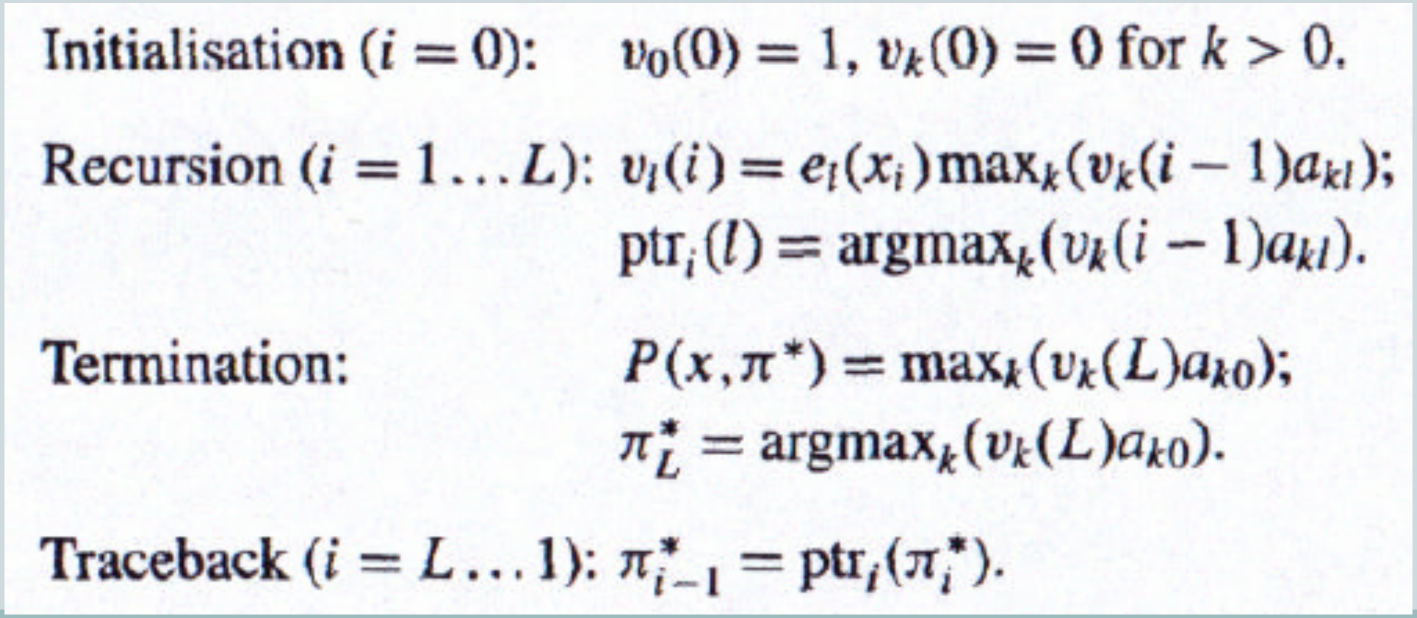
- $V_{loaded}(5)$ is the maximum of two probabilities: the most probable sequences such that either
  - 4'th throw used a loaded die and it is continued to be used for 5th throw, or
  - The die was switched from fair to loaded after 4th throw
- Simple recurrence gives the result:

$$v_{loaded}(5) = e_{loaded}(6) \max(v_{loaded}(4) a_{loaded,loaded}, v_{fair}(4) a_{fair,loaded})$$


Observed sequence of die rolls:
1
4
3
6
6
4
Loaded
Fair
Hidden path: the sequence of which die being used:
F
F
L
L
L
F

Viterbi algorithm

- Dynamic programming sweep over the sequences
- To recover the best state sequence fast a traceback pointer ptr$_k$(i) is stored for each (i,k)



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

Recursion ($i = 1 \ldots L$): $v_l(i) = e_l(x_i)\max_k(v_k(i-1)a_{kl})$;

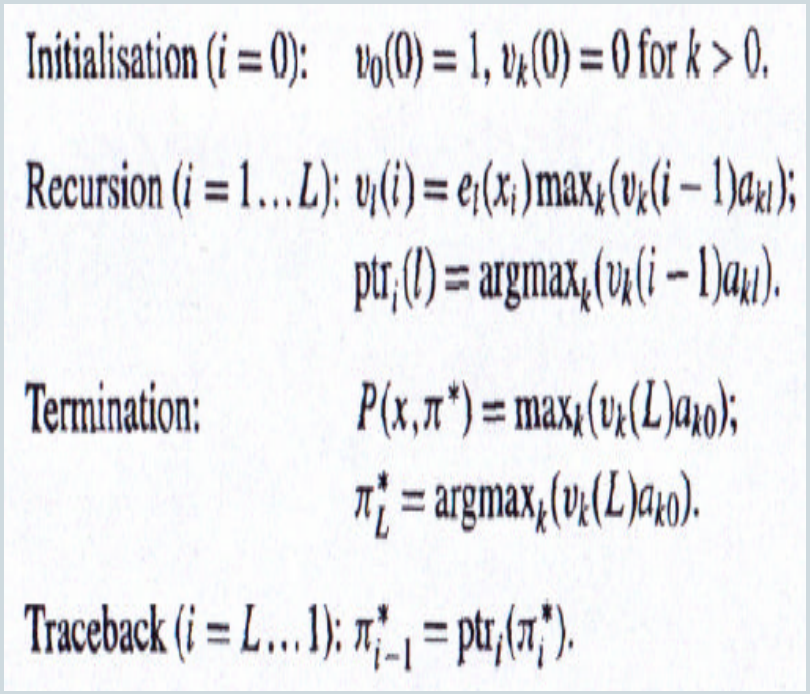
$\mathrm{ptr}_i(l) = \mathrm{argmax}_k(v_k(i-1)a_{kl})$.

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

$\pi_L^* = \mathrm{argmax}_k(v_k(L)a_{k0})$.

Traceback ($i = L \ldots 1$): $\pi_{i-1}^* = \mathrm{ptr}_i(\pi_i^*)$.

Viterbi at the casino



Initialisation ($i = 0$): $v_0(0) = 1,\ v_k(0) = 0$ for $k > 0$.

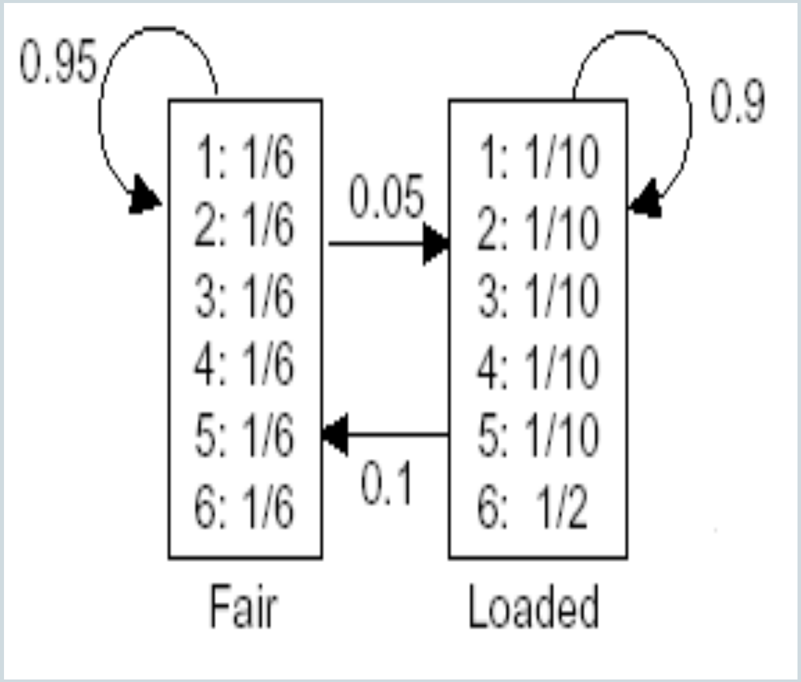
Recursion ($i = 1 \dots L$): $v_l(i) = e_l(x_i)\max_k(v_k(i-1)a_{kl})$;

$\mathrm{ptr}_i(l) = \mathrm{argmax}_k(v_k(i-1)a_{kl})$.

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

$\pi_L^* = \mathrm{argmax}_k(v_k(L)a_{k0})$.

Traceback ($i = L \dots 1$): $\pi_{i-1}^* = \mathrm{ptr}_i(\pi_i^*)$.

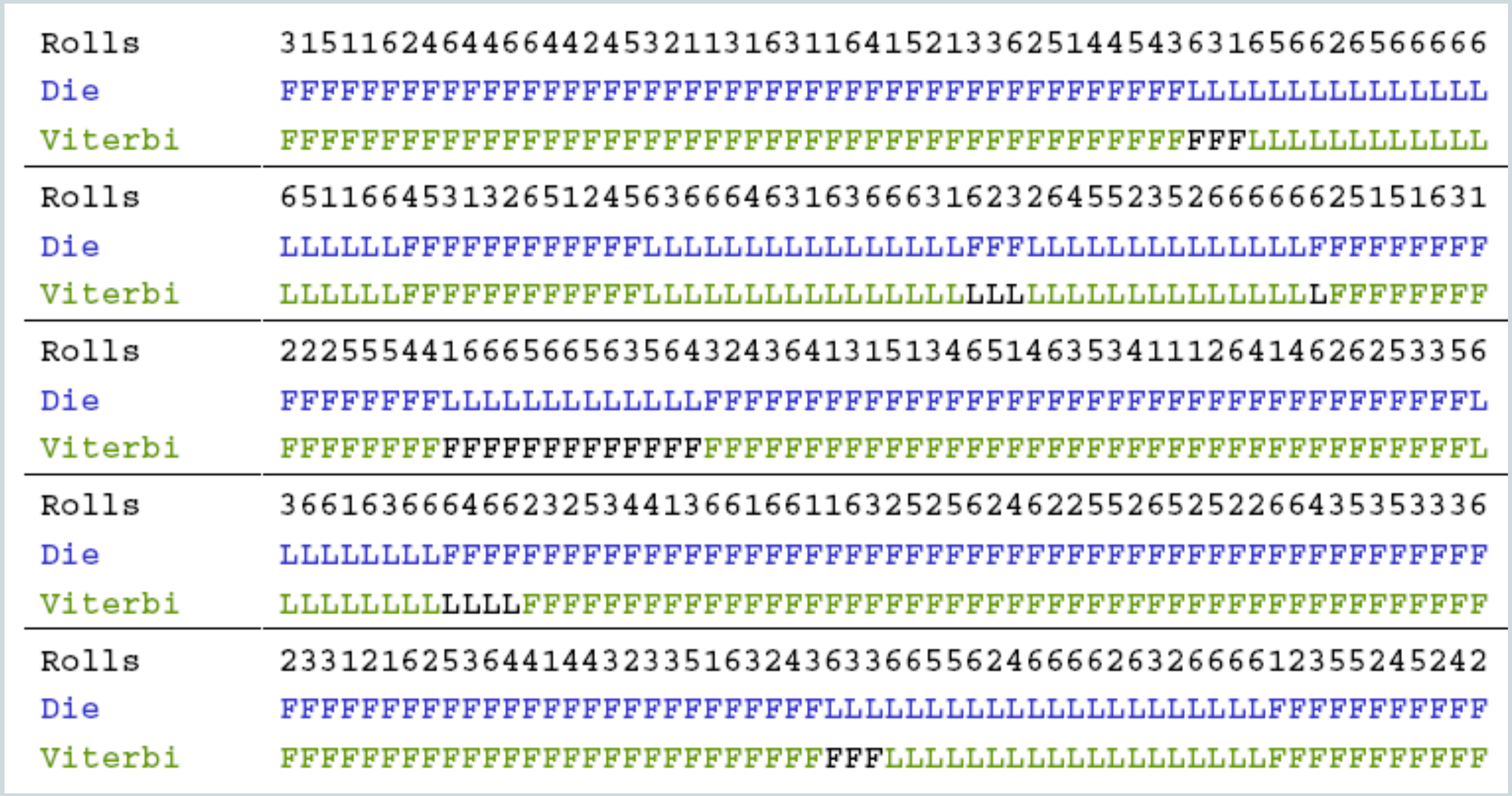


Viterbi at the casino

v	0	1	2	3	4	5	6
Start	1	0	0	0	0	0	0
Fair	0	8.3333E-02	1.3194E-02	2.0891E-03	3.3078E-04	5.2373E-05	8.2924E-06
Loaded	0	5.0000E-02	4.5000E-03	2.0250E-03	9.1125E-04	8.2013E-05	3.6906E-05
Die		4	2	6	6	3	6
Log -odds FAIR vs LOADED		0.73696559	1.5519337	0.04497371	-1.4619863	-0.6470182	-2.1539782

Viterbi at the casino

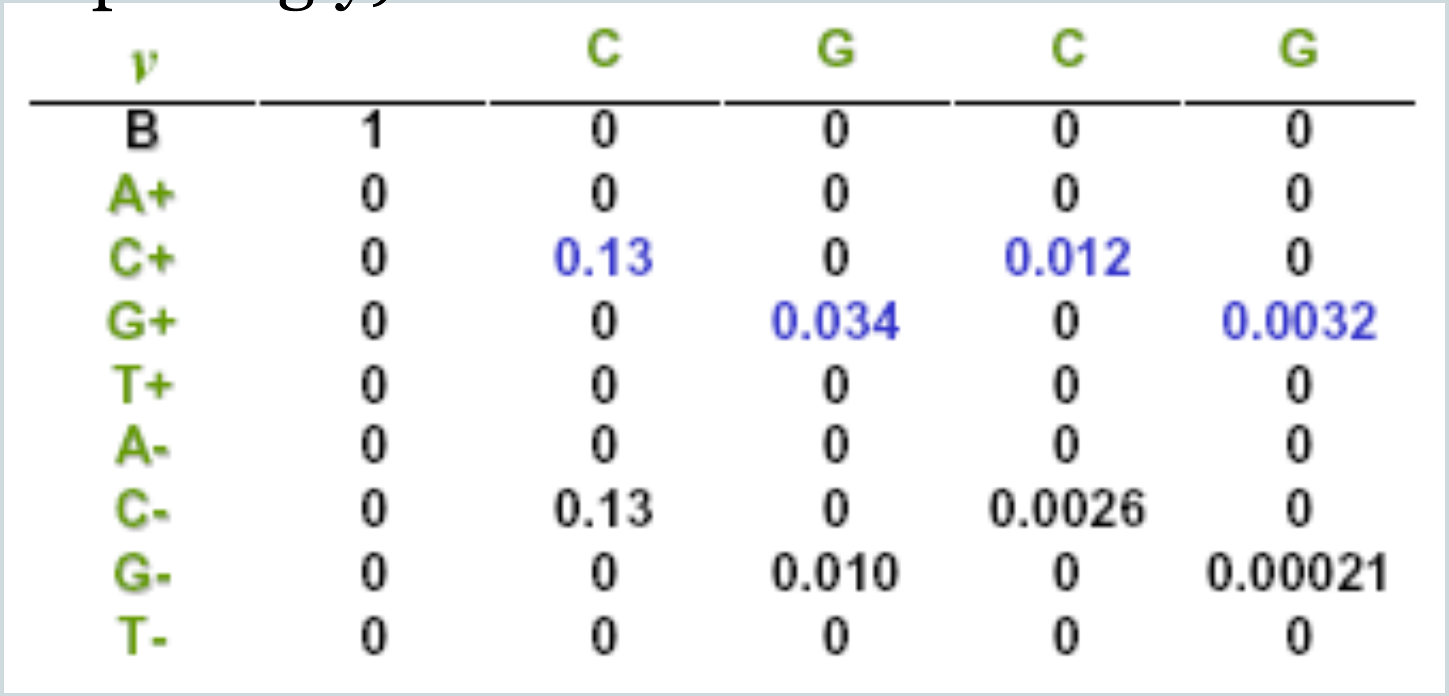
- Viterbi estimates remarkably well the correct die



```
Rolls      315116246446644245321131631164152133625144543631656626566666
Die        FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi    FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLL
-----------------------------------------------------------------------
Rolls      651166453132651245636664631636663162326455235266666625151631
Die        LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLFFFFFFFFF
Viterbi    LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF
-----------------------------------------------------------------------
Rolls      222555441666566563564324364131513465146353411126414626253356
Die        FFFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL
Viterbi    FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL
-----------------------------------------------------------------------
Rolls      366163666466232534413661661163252562462255265252266435353336
Die        LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi    LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
-----------------------------------------------------------------------
Rolls      233121625364414432335163243633665562466662632666612355245242
Die        FFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
Viterbi    FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
```

Viterbi on the CpG island

- Table v for the sequence CGCG
- The most probable path stays on the '+' side (unsurprisingly)



v		C	G	C	G
B	1	0	0	0	0
A+	0	0	0	0	0
C+	0	0.13	0	0.012	0
G+	0	0	0.034	0	0.0032
T+	0	0	0	0	0
A-	0	0	0	0	0
C-	0	0.13	0	0.0026	0
G-	0	0	0.010	0	0.00021
T-	0	0	0	0	0

Complexity and Implementation

- The time-complexity of the Viterbi algorithm is O(L | Q|², where L is the length of the sequence, and Q is the set of states
- Space complexity is O(LQ), i.e. the size of the tables to be filled
- Important implementation issue: to avoid numerical underflow when multiplying small probabilities, it is better to use log-probabilities instead:

$$V_l(i+1) = E_l(x_{i+1}) + \max_k (V_k(i) + A_{kl})$$

- Capital V,E,A denote the logarithms of the original quantities