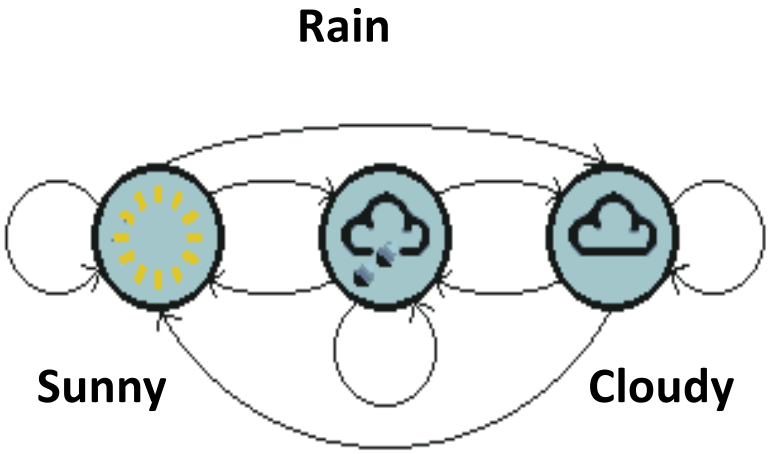
Hidden Markov Models, HMM's

Morten Nielsen
Department of Health
Technology, DTU

Objectives

- Introduce Hidden Markov models and understand that they are just weight matrices with gaps
- How to construct an HMM
- How to "align/score" sequences to HMM's
  - Viterbi decoding
  - Forward decoding
  - Backward decoding
  - Posterior Decoding
- Use and construct a Profile HMM
  - HMMer

Markov Chains



weather yesterday \ weather today	Sun	Cloud	Rain
Sun	0.5	0.25	0.25
Cloud	0.375	0.125	0.375
Rain	0.125	0.625	0.375

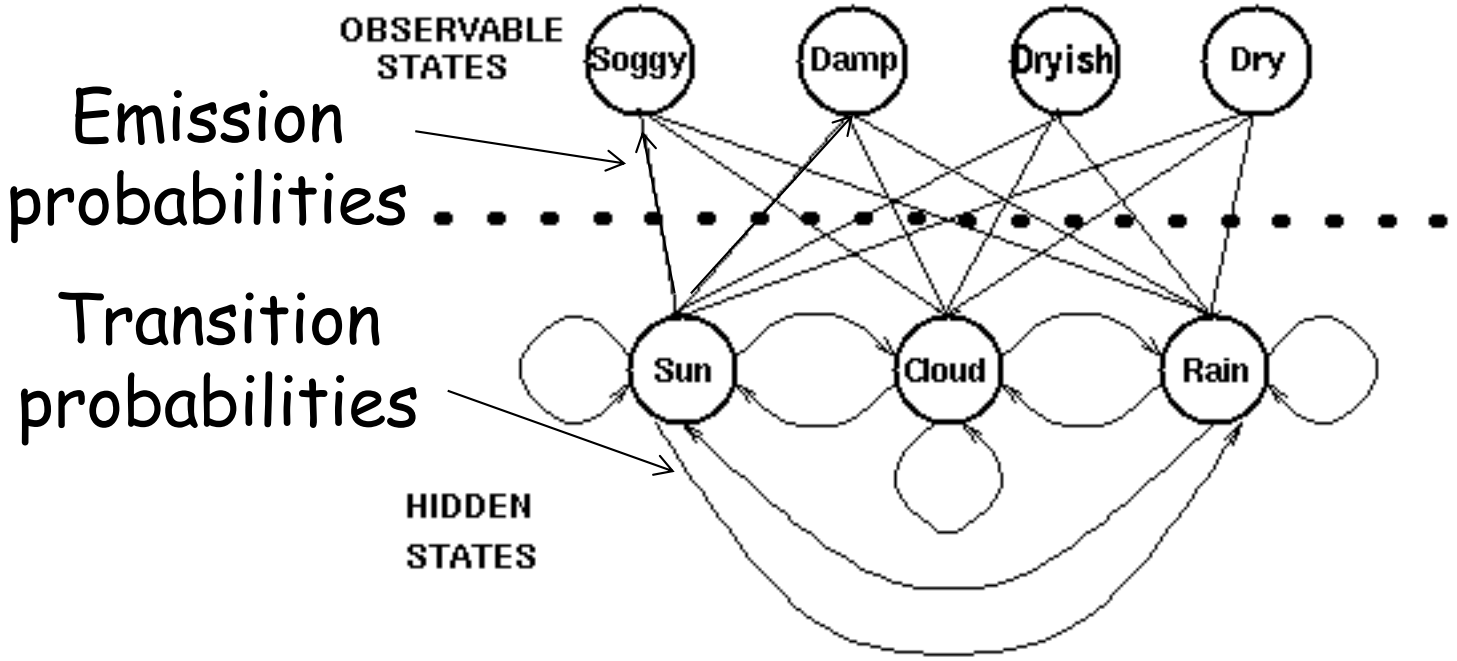
States : Three states - sunny, cloudy, rainy.

State transition matrix : The probability of the weather given the previous day's weather.

Sun	Cloud	Rain
1.0	0.0	0.0

Initial Distribution : Defining the probability of the system being in each of the states at time 0.

Hidden Markov Models



Hidden states : the (TRUE) states of a system that may be described by a Markov process (e.g., the weather).

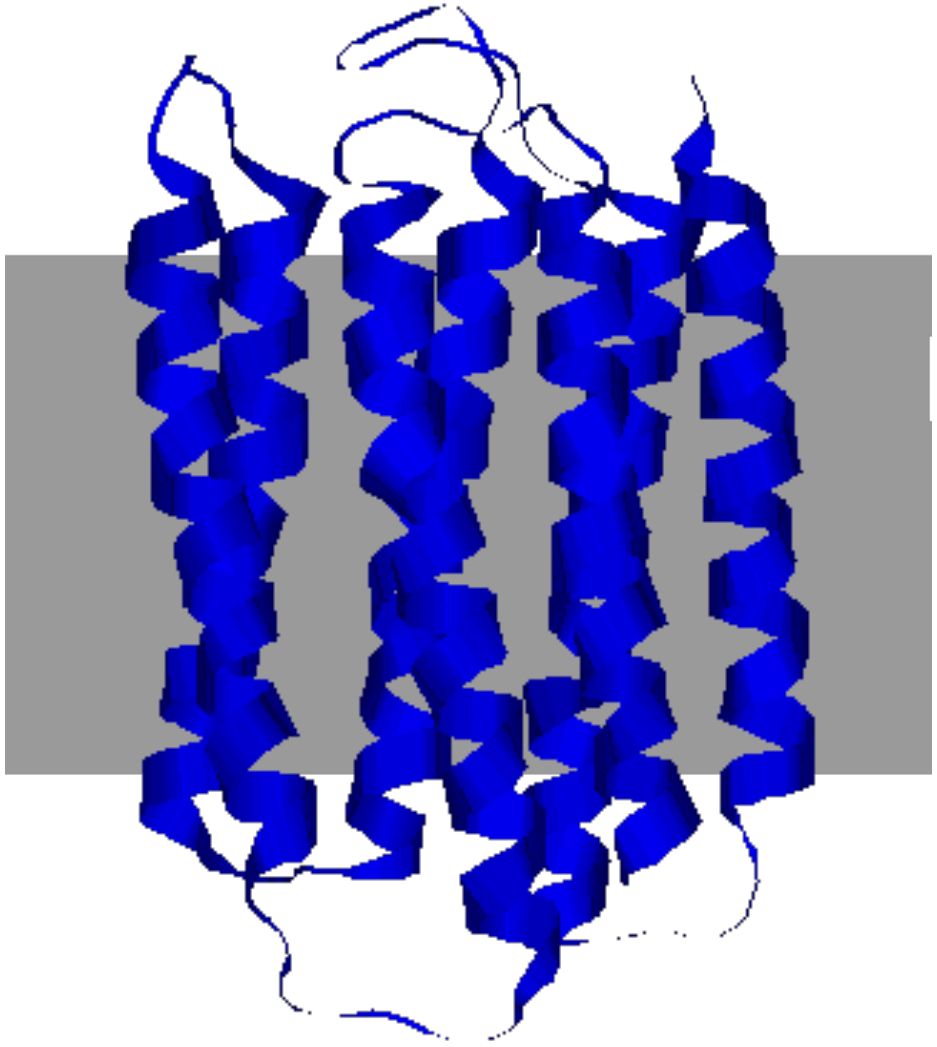
Observable states : the states of the process that are `visible' (e.g., seaweed dampness).

TMHMM (trans-membrane HMM)
(Sonnhammer, von Heijne, and Krogh)

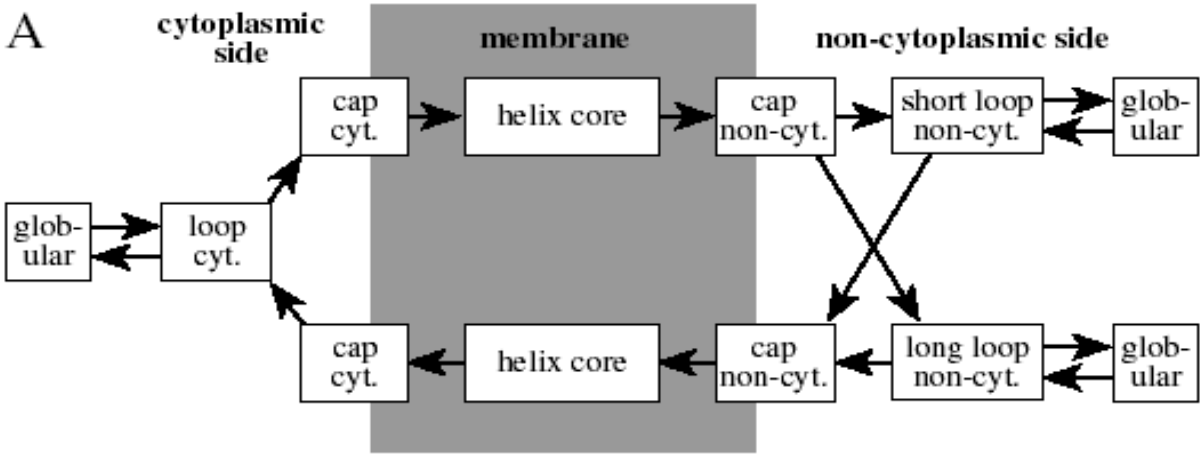
Extra cellular

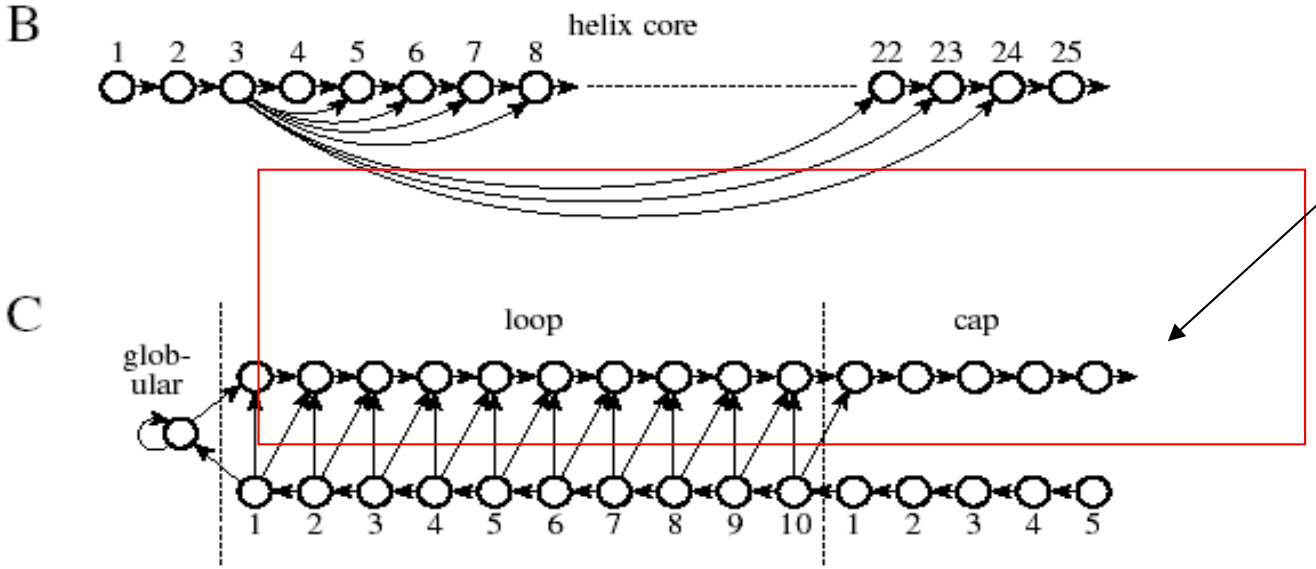
Trans membrane

Intra cellular



TMHMM (trans-membrane HMM)
(Sonnhammer, von Heijne, and Krogh)





Model TM length distribution.
Power of HMM.
Difficult in alignment.

ALLYVDWQILPVIL

Weight matrix construction

CENTERFO
RBIOLOGI
CALSEQU
ENCEANA
LYSIS CBS

SLLPAIVEL YLLPAIVHI TLWVDPYEV GLVPFLVSV KLLEPVLLL LLDVPTAAV LLDVPTAAV LLDVPTAAV
LLDVPTAAV VLFRGGPRG MVDGTLLLL YMNGTMSQV MLLSVPLLL SLLGLLVEV ALLPPINIL TLIKIQHTL
HLIDYLVTS ILAPPVVKL ALFPQLVIL GILGFVFTL STNRQSGRQ GLDVLTAKV RILGAVAKV QVCERIPTI
ILFGHENRV ILMEHIHKL ILDQKINEV SLAGGIIGV LLIENVASL FLLWATAEA SLPDFGISY KKREEAPSL
LERPGGNEI ALSNLEVKL ALNELLQHV DLERKVESL FLGENISNF ALSDHHIYL GLSEFTEYL STAPPAHGV
PLDGEYFTL GVLVGVALI RTLDKVLEV HLSTAFARV RLDSYVRSL YMNGTMSQV GILGFVFTL ILKEPVHGV
ILGFVFTLT LLFGYPVYV GLSPTVWLS WLSLLVPFV FLPSDFFPS CLGGLLTMV FIAGNSAYE KLGEFYNQM
KLVALGINA DLMGYIPLV RLVTLKDIV MLLAVLYCL AAGIGILTV YLEPGPVTA LLDGTATLR ITDQVPFSV
KTWGQYWQV TITDQVPFS AFHHVAREL YLNKIQNSL MMRKLAILS AIMDKNIIL IMDKNIILK SMVGNWAKV
SLLAPGAKQ KIFGSLAFL ELVSEFSRM KLTPLCVTL VLYRYGSFS YIGEVLVSV CINGVCWTV VMNILLQYV
ILTVILGVL KVLEYVIKV FLWGPRALV GLSRYVARL FLLTRILTI HLGNVKYLV GIAGGLALL GLQDCTMLV
TGAPVTYST VIYQYMDDL VLPDVFIRC VLPDVFIRC AVGIGIAVV LVVLGLLAV ALGLGLLPV GIGIGVLAA
GAGIGVAVL IAGIGILAI LIVIGILIL LAGIGLIAA VDGIGILTI GAGIGVLTA AAGIGIIQI QAGIGILLA
KARDPHSGH KACDPHSGH ACDPHSGHF SLYNTVATL RGPGRAFVT NLVPMVATV GLHCYEQLV PLKQHFQIV
AVFDRKSDA LLDFVRFMG VLVKSPNHV GLAPPQHLI LLGRNSFEV PLTFGWCYK VLEWRFDSR TLNAWVKVV
GLCTLVAML FIDSYICQV IISAVVGIL VMAGVGSPY LLWTLVVLL SVRDRLARL LLMDCSGSI CLTSTVQLV
VLHDDLLEA LMWITQCFL SLLMWITQC QLSLLMWIT LLGATCMFV RLTRFLSRV YMDGTMSQV FLTPKKLQC
ISNDVCAQV VKTDGNPPE SVYDFFVWL FLYGALLLA VLFSSDFRI LMWAKIGPV SLLLELEEV SLSRFSWGA
YTAFTIPSI RLMKQDFSV RLPRIFCSC FLWGPRAYA RLLQETELV SLFEGIDFY SLDQSVVEL RLNMFTPYI
NMFTPYIGV LMIIPLINV TLFIGSHVV SLVIVTTFV VLQWASLAV ILAKFLHWL STAPPHVNV LLLLTVLTV
VVLGVVFGI ILHNGAYSL MIMVKCWMI MLGTHTMEV MLGTHTMEV SLADTNSLA LLWAARPRL GVALQTMKQ
GLYDGMEHL KMVELVHFL YLQLVFGIE MLMAQEALA LMAQEALAF VYDGREHTV YLSGANLNL RMFPNAPYL
EAAGIGILT TLDSQVMSL STPPPGTRV KVAELVHFL IMIGVLVGV ALCRWGLLL LLFAGVQCQ VLLCESTAV
YLSTAFARV YLLEMLWRL SLDDYNHLV RTLDKVLEV GLPVEYLQV KLIANNTRV FIYAGSLSA KLVANNTRL
FLDEFMEGV ALQPGTALL VLDGLDVLL SLYSFPEPE ALYVDSLFF SLLQHLIGL ELTLGEFLK MINAYLDKL
AAGIGILTV FLPSDFFPS SVRDRLARL SLREWLLRI LLSAWILTA AAGIGILTV AVPDEIPPL FAYDGKDYI
AAGIGILTV FLPSDFFPS AAGIGILTV FLPSDFFPS AAGIGILTV FLWGPRALV ETVSEQSNV ITLWQRPLV

PSSM construction

- Calculate amino acid frequencies at each position using
  - Sequence weighting
  - Pseudo counts
- Define background model
  - Use background amino acids frequencies
- PSSM is

$$S(a_i) = \log \frac{p(a_i)}{q(a)}$$

More on scoring

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
1	0.6	0.4	-3.5	-2.4	-0.4	-1.9	-2.7	0.3	-1.1	1.0	0.3	0.0	1.4	1.2	-2.7	1.4	-1.2	-2.0	1.1	0.7
2	-1.6	-6.6	-6.5	-5.4	-2.5	-4.0	-4.7	-3.7	-6.3	1.0	5.1	-3.7	3.1	-4.2	-4.3	-4.2	-0.2	-5.9	-3.8	0.4
3	0.2	-1.3	0.1	1.5	0.0	-1.8	-3.3	0.4	0.5	-1.0	0.3	-2.5	1.2	1.0	-0.1	-0.3	-0.5	3.4	1.6	0.0
4	-0.1	-0.1	-2.0	2.0	-1.6	0.5	0.8	2.0	-3.3	0.1	-1.7	-1.0	-2.2	-1.6	1.7	-0.6	-0.2	1.3	-6.8	-0.7
5	-1.6	-0.1	0.1	-2.2	-1.2	0.4	-0.5	1.9	1.2	-2.2	-0.5	-1.3	-2.2	1.7	1.2	-2.5	-0.1	1.7	1.5	1.0
6	-0.7	-1.4	-1.0	-2.3	1.1	-1.3	-1.4	-0.2	-1.0	1.8	0.8	-1.9	0.2	1.0	-0.4	-0.6	0.4	-0.5	-0.0	2.1
7	1.1	-3.8	-0.2	-1.3	1.3	-0.3	-1.3	-1.4	2.1	0.6	0.7	-5.0	1.1	0.9	1.3	-0.5	-0.9	2.9	-0.4	0.5
8	-2.2	1.0	-0.8	-2.9	-1.4	0.4	0.1	-0.4	0.2	-0.0	1.1	-0.5	-0.5	0.7	-0.3	0.8	0.8	-0.7	1.3	-1.1
9	-0.2	-3.5	-6.1	-4.5	0.7	-0.8	-2.5	-4.0	-2.6	0.9	2.8	-3.0	-1.8	-1.4	-6.2	-1.9	-1.6	-4.9	-1.6	4.5

$$S = \sum_i S(a_i)$$

$$S = \sum_i \log \frac{p(a_i)}{q(a_i)}$$

$$S = \log \left(\frac{\prod_i p(a_i)}{\prod_i q(a_i)} \right)$$

Probability of observation given Model

Probability of observation given Prior (background)

$$S = \log \left(\frac{P(a \mid M)}{P(a \mid B)} \right)$$

Hidden Markov Models

- Weight matrices do not deal with insertions and deletions
- In alignments, this is done in an ad-hoc manner by optimization of the two gap penalties for first gap and gap extension
- HMM is a natural framework where insertions/deletions are dealt with explicitly

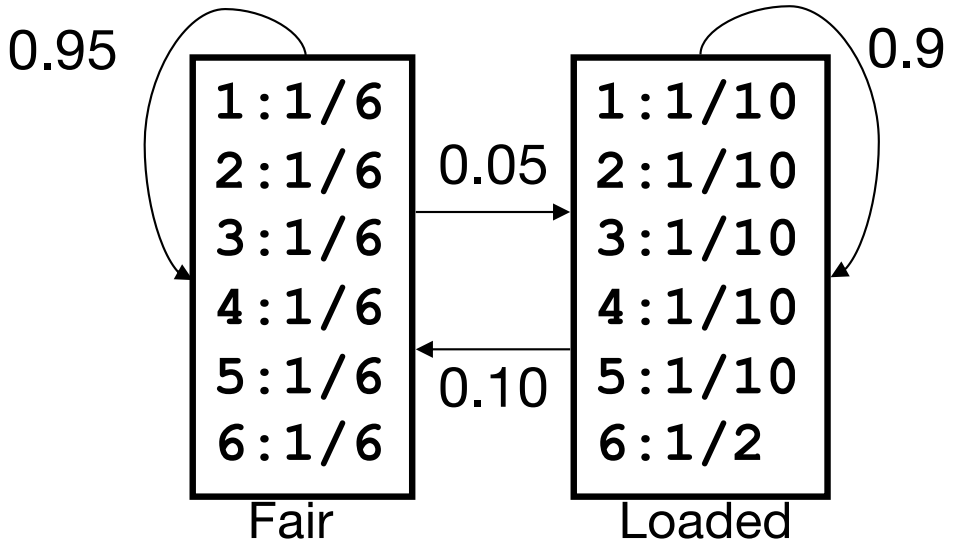
Multiple sequence alignment
Learning from evolution

CENTERFO
RBIOLOGI
CALSEQU
ENCEANA
LYSIS CBS

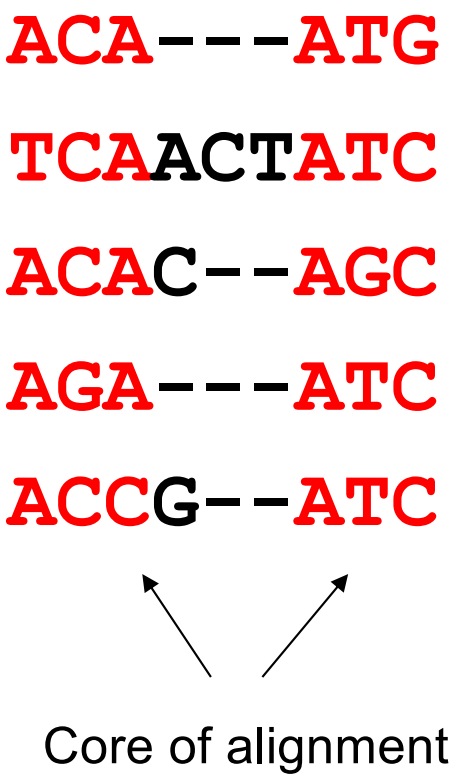
What is hidden?

The unfair casino: Loaded die p(6) = 0.5; switch fair to load:0.05; switch load to fair: 0.1

- Model generates numbers
  - 312453666641
- Does not tell which die was used
- Alignment (decoding) can give the most probable solution/path (Viterbi)
  - FFFFFFLLLLLL
- Or most probable set of states
  - FFFFFFLLLLLL



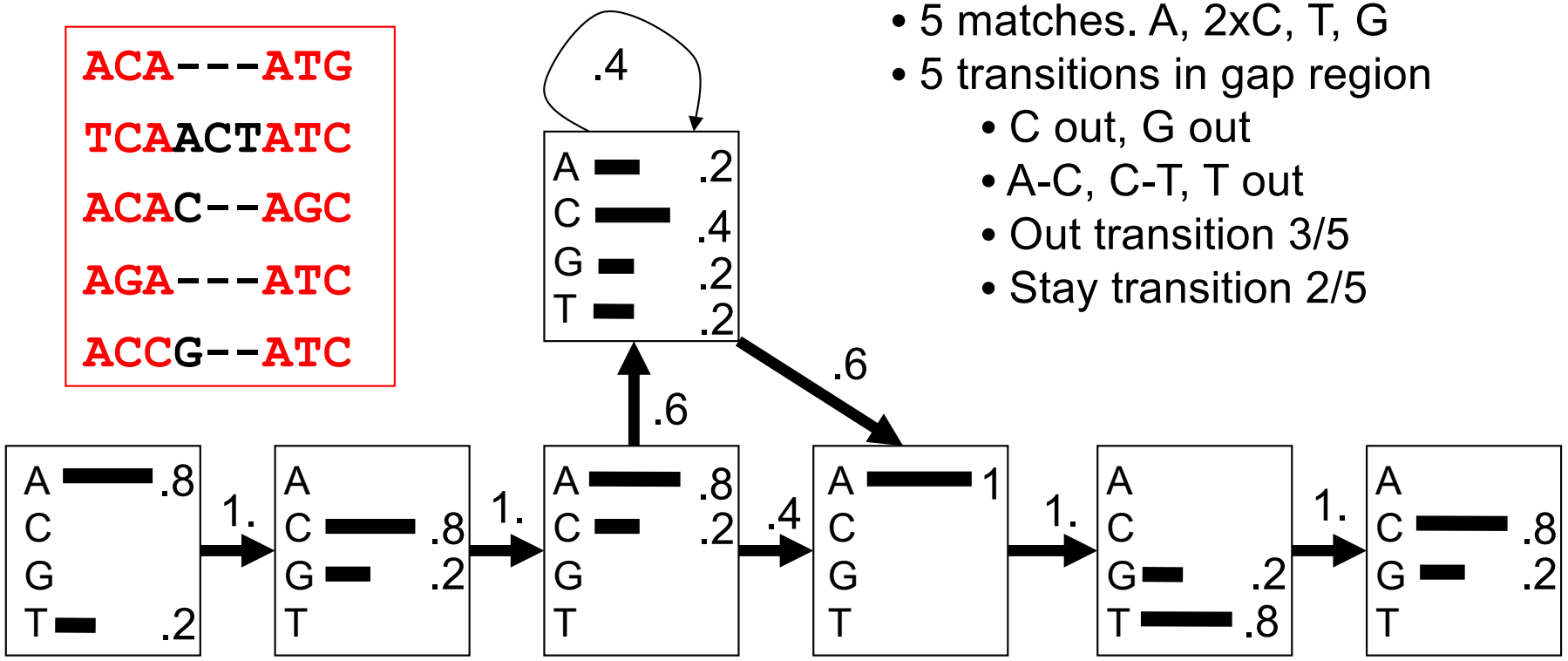
HMM (a simple example)



- Example from A. Krogh
- Core region defines the number of states in the HMM (red)
- Insertion and deletion statistics are derived from the non-core part of the alignment (black)

HMM construction (supervised learning)


```
ACA---ATG
TCAACTATC
ACAC--AGC
AGA---ATC
ACCG--ATC
```



- 5 matches. A, 2xC, T, G
- 5 transitions in gap region
  - C out, G out
  - A-C, C-T, T out
  - Out transition 3/5
  - Stay transition 2/5

ACA---ATG $0.8 \times 1 \times 0.8 \times 1 \times 0.8 \times 0.4 \times 1 \times 1 \times 0.8 \times 1 \times 0.2 = 3.3 \times 10^{-2}$

Scoring a sequence to an HMM

ACA---ATG 0.8x1x0.8x1x0.8x0.4x1x0.8x1x0.2 = 3.3×10^{-2}

TCAACTATC 0.2x1x0.8x1x0.8x0.6x0.2x0.4x0.4x0.4x0.2x0.6x1x1x0.8x1x0.8 = 0.0075×10^{-2}

ACAC--AGC = 1.2×10^{-2}

Consensus:

ACAC--ATC = 4.7×10^{-2}, **ACA---ATC** = 13.1×10^{-2}

Exceptional:

TGCT--AGG = 0.0023×10^{-2}

Align sequence to HMM - Null model

- Score depends strongly on length
- Null model is a random model. For length L the score is 0.25^L
- Log-odds score for sequence S
- Log(P(S)/0.25^L)
- Positive score means more likely than Null model
- **This is just like we did for PSSM log(p/q)!**

```
ACA---ATG = 4.9
TCAACTATC = 3.0
ACAC--AGC = 5.3
AGA---ATC = 4.9
ACCG--ATC = 4.6
Consensus:
ACAC--ATC = 6.7
ACA---ATC = 6.3
Exceptional:
TGCT--AGG = -0.97
```

↙ Note!

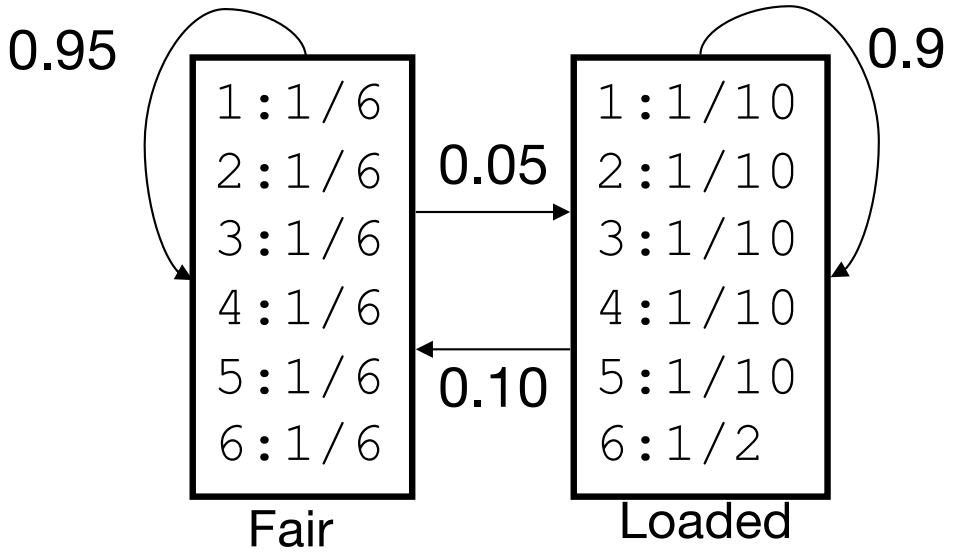
Aligning a sequence to an HMM

- Find the path through the HMM states that has the highest probability
  - For alignments, we found the path through the scoring matrix that had the highest sum of scores
- The number of possible paths rapidly gets very large making brute force search infeasible
  - Just like checking all path for alignment did not work
- Use dynamic programming
  - The Viterbi algorithm does the job

The Viterbi algorithm

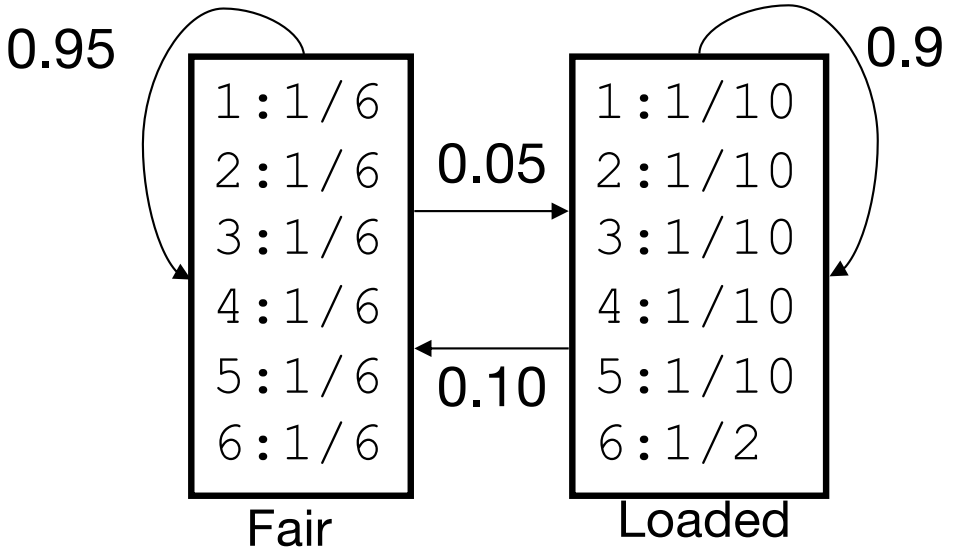
The unfair casino: Loaded dice p(6) = 0.5; switch fair to load:0.05; switch load to fair: 0.1

- Model generates numbers
  - 312453666641



Model decoding (Viterbi)

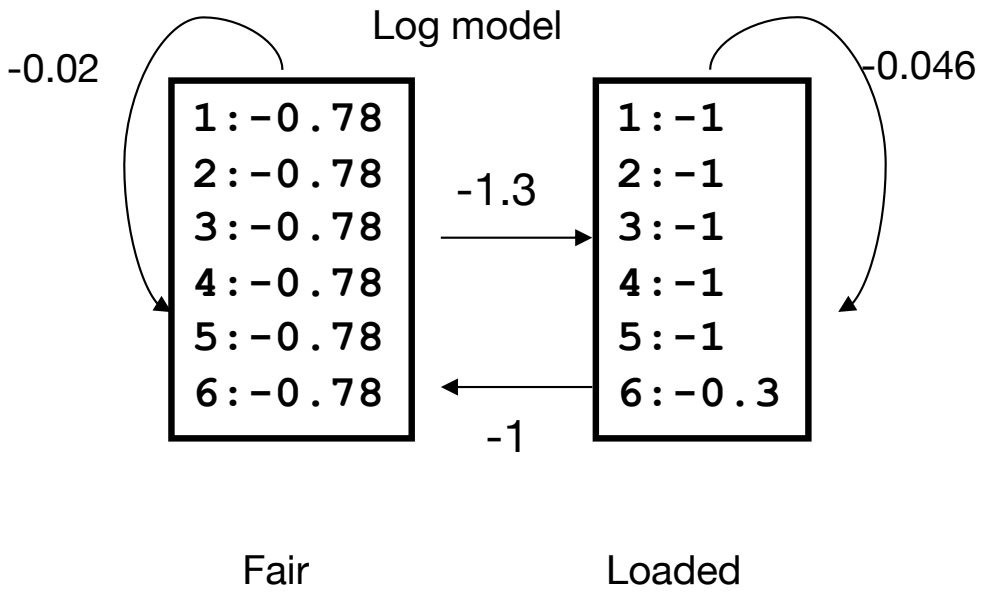
- Example: 566. What was the most likely series of dice used to generate this output?
- Use Brute force




```
FFF = 0.5*0.167*0.95*0.167*0.95*0.167 = 0.0021
FFL = 0.5*0.167*0.95*0.167*0.05*0.5 = 0.00333
FLF = 0.5*0.167*0.05*0.5*0.1*0.167 = 0.000035
FLL = 0.5*0.167*0.05*0.5*0.9*0.5 = 0.00094
LFF = 0.5*0.1*0.1*0.167*0.95*0.167 = 0.00013
LFL = 0.5*0.1*0.1*0.167*0.05*0.5 = 0.000021
LLF = 0.5*0.1*0.9*0.5*0.1*0.167 = 0.00038
LLL = 0.5*0.1*0.9*0.5*0.9*0.5 = 0.0101
```

Or in log space

- Example: 566. What was the most likely series of dice used to generate this output?




```
Log(P(LLL|M)) = log(0.5*0.1*0.9*0.5*0.9*0.5) = log(0.0101)
                                    or
Log(P(LLL|M)) = log(0.5)+log(0.1)+log(0.9)+log(0.5)+log(0.9)+log(0.5)
              = -0.3 -1 -0.046 -0.3 -0.046 -0.3 = -1.99
```

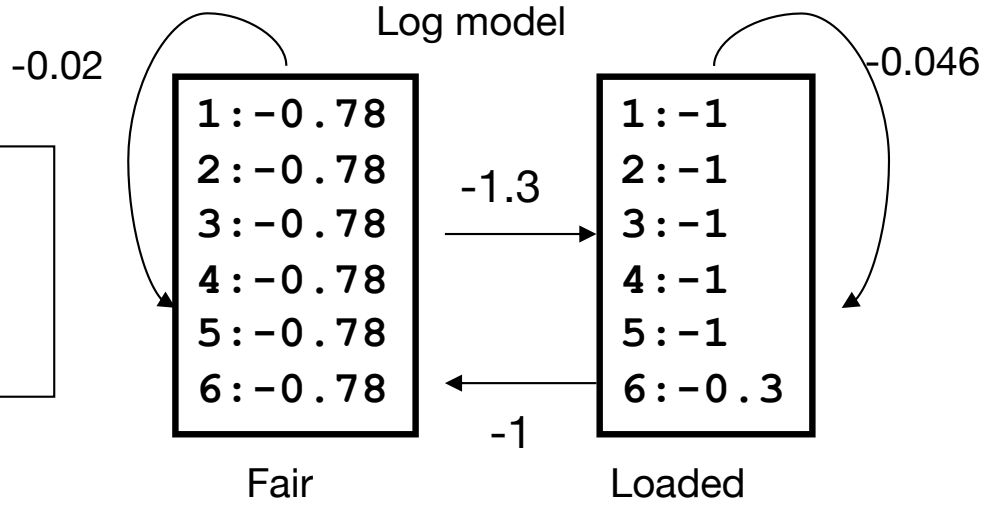
Model decoding (Viterbi)

- Example: 566611234. What was the most likely series of dice used to generate this output?

Pick the fair die at random

Make a 5 with the fair die

```
F = 0.5*0.167
log(F) = log(0.5) + log(0.167) = -1.08
L = 0.5*0.1
log(L) = log(0.5) + log(0.1) = -1.30
```



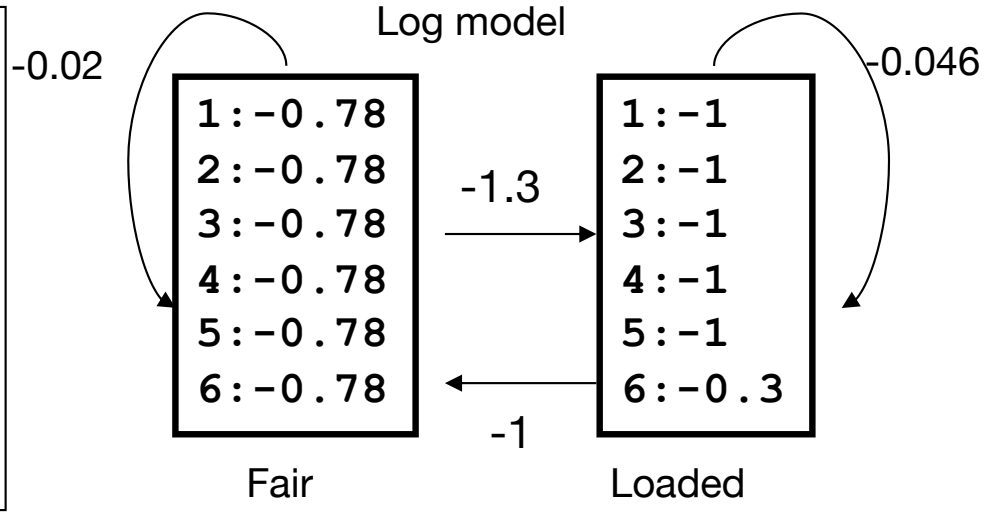
	5	6	6	6	1	1	2	3	4
F	-1.08								
L	-1.30								

Model decoding (Viterbi)

CENTERFO
RBIOLOGI
CALSEQU
ENCEANA
LYSIS CBS

- Example: 566611234. What was the most likely series of dice used to generate this output?

```
FF = 0.5*0.167*0.95*0.167
Log(FF) = -0.30 -0.78 - 0.02 -0.78 = -1.88
LF = 0.5*0.1*0.1*0.167
Log(LF) = -0.30 -1 -1 -0.78 = -3.08
FL = 0.5*0.167*0.05*0.5
Log(FL) = -0.30 -0.78 - 1.30 -0.30 = -2.68
LL = 0.5*0.1*0.9*0.5
Log(LL) = -0.30 -1 -0.046 -0.3 = -1.65
```



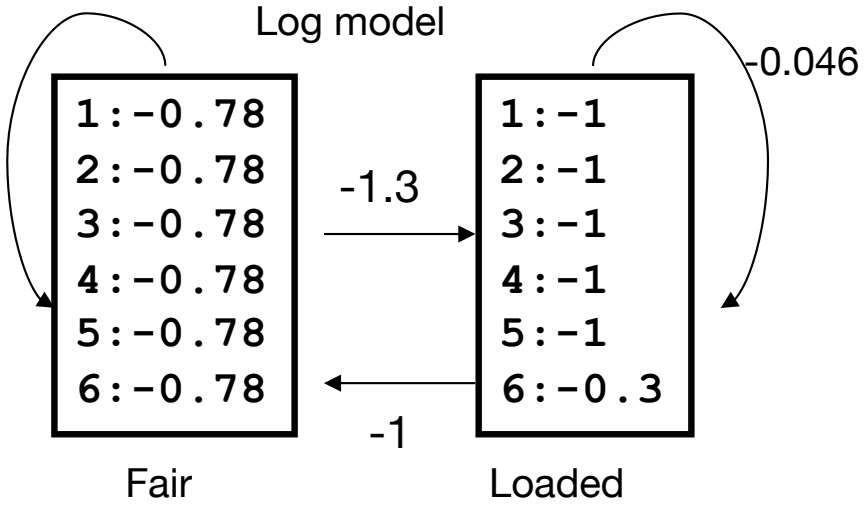
	5	6	6	6	1	1	2	3	4
F	-1.08								
L	-1.30								

Model decoding (Viterbi)

- Example: 566611234. What was the most likely series of dice used to generate this output?

```
FF = 0.5*0.167*0.95*0.167
Log(FF) = -0.30 -0.78 - 0.02 -0.78 = -1.88
LF = 0.5*0.1*0.1*0.167
Log(LF) = -0.30 -1 -1 -0.78 = -3.08
FL = 0.5*0.167*0.05*0.5
Log(FL) = -0.30 -0.78 - 1.30 -0.30 = -2.68
LL = 0.5*0.1*0.9*0.5
Log(LL) = -0.30 -1 -0.046 -0.3 = -1.65
```

-0.02

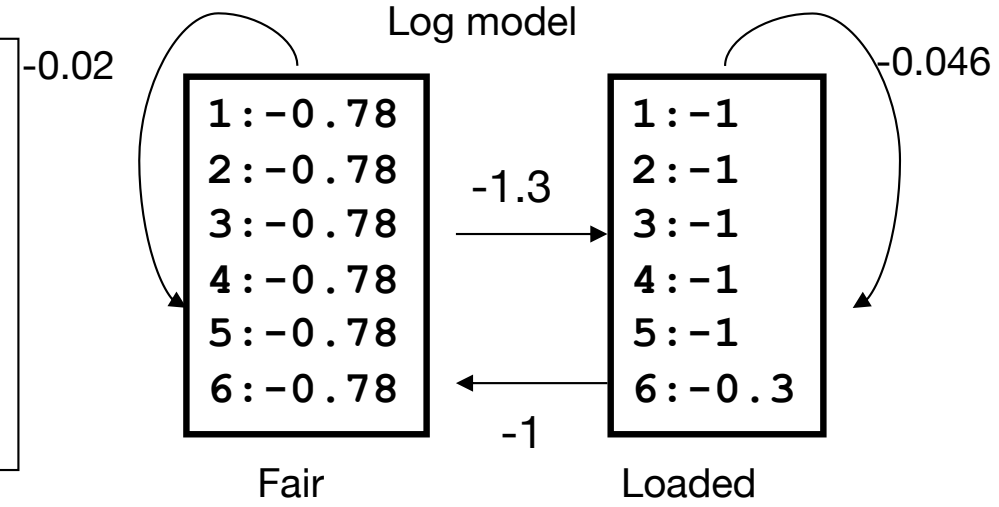


	5	6	6	6	1	1	2	3	4
F	-1.08	-1.88							
L	-1.30	-1.65							

Model decoding (Viterbi)

- Example: 566611234. What was the most likely series of dice used to generate this output?

```
FFF = 0.5*0.167*0.95*0.167*0.95*0.167 = 0.0021
FLF = 0.5*0.167*0.05*0.5*0.1*0.167 = 0.000035
LFF = 0.5*0.1*0.1*0.167*0.95*0.167 = 0.00013
LLF = 0.5*0.1*0.9*0.5*0.1*0.167 = 0.00038
FLL = 0.5*0.167*0.05*0.5*0.9*0.5 = 0.00094
FFL = 0.5*0.167*0.95*0.167*0.05*0.5 = 0.00333
LFL = 0.5*0.1*0.1*0.167*0.05*0.5 = 0.000021
LLL = 0.5*0.1*0.9*0.5*0.9*0.5 = 0.0101
```

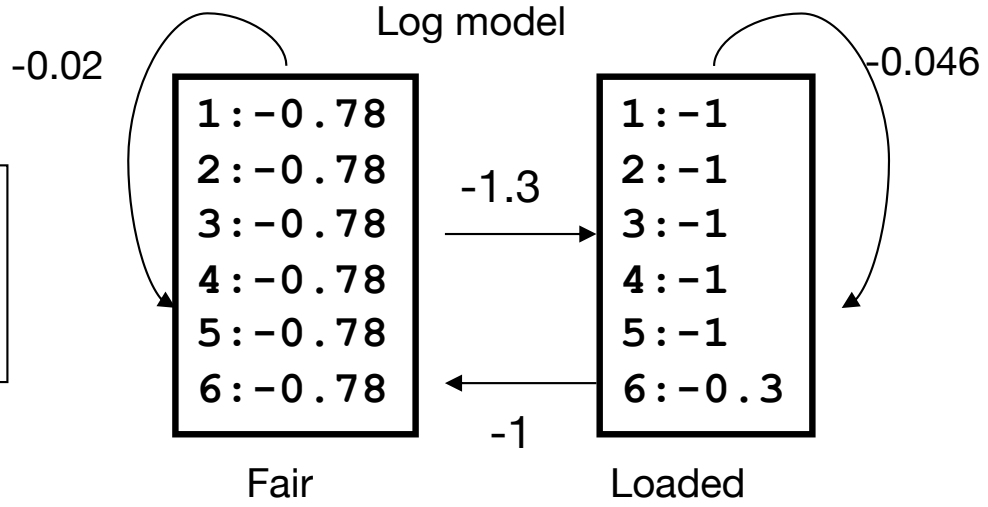


	5	6	6	6	1	1	2	3	4
F	-1.08	-1.88							
L	-1.30	-1.65							

Model decoding (Viterbi)

- Example: 566611234. What was the most likely series of dice used to generate this output?

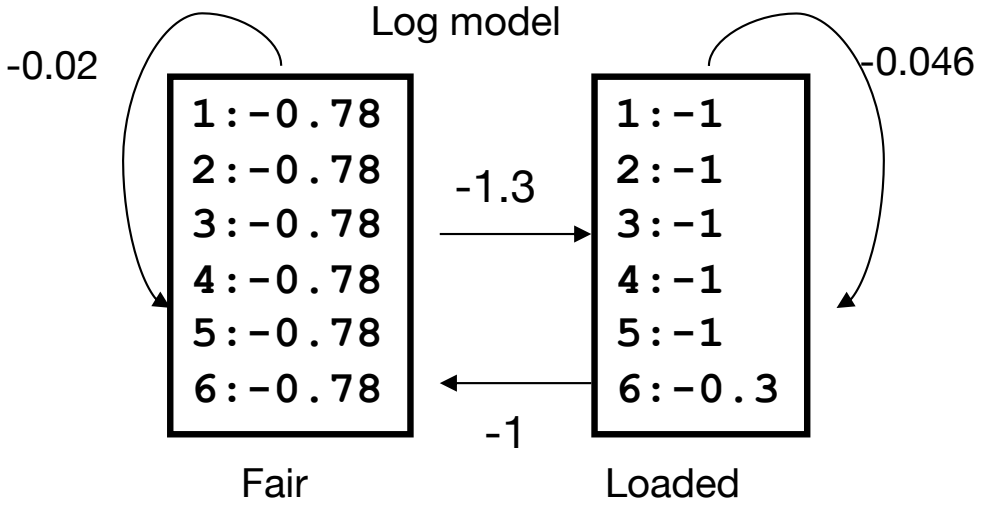
```
FFF = 0.5*0.167*0.95*0.167*0.95*0.167 = 0.0021
Log(P(FFF))=-2.68
LLL = 0.5*0.1*0.9*0.5*0.9*0.5 = 0.0101
Log(P(LLL))=-1.99
```



	5	6	6	6	1	1	2	3	4
F	-1.08	-1.88	-2.68						
L	-1.30	-1.65	-1.99						

Model decoding (Viterbi)

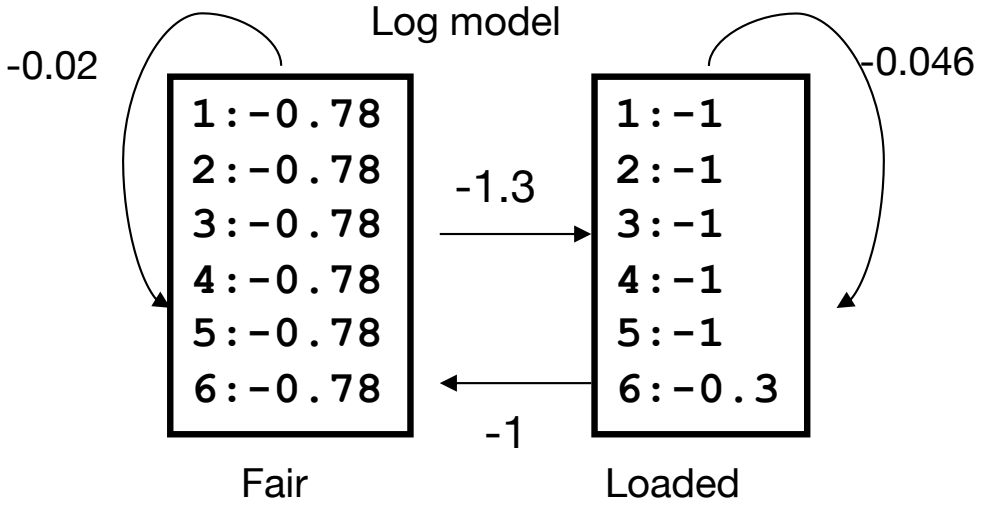
- Example: 566611234. What was the most likely series of dice used to generate this output?



	5	6	6	6	1	1	2	3	4
F	-1.08	-1.88	-2.68						
L	-1.30	-1.65	-1.99						

Model decoding (Viterbi)

- Example: 566611234. What was the most likely series of dice used to generate this output?



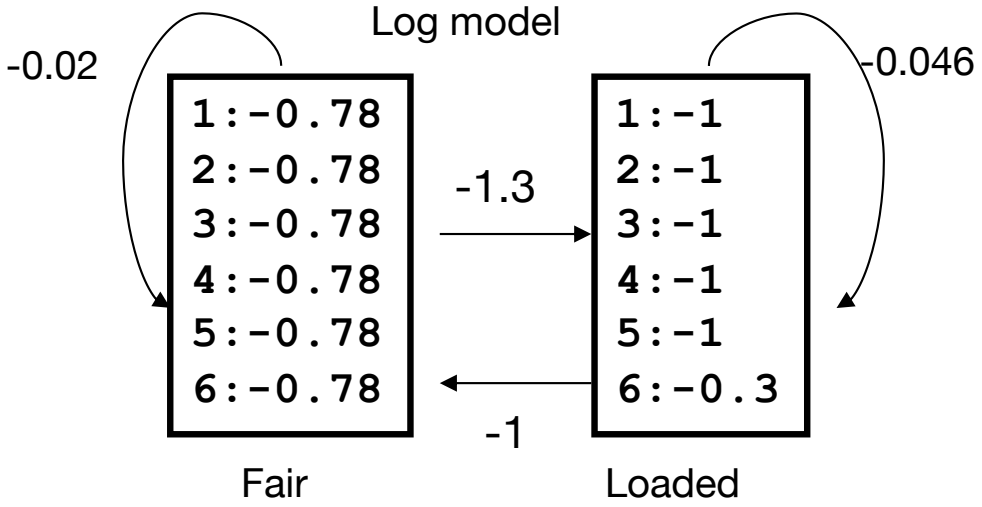
$$-0.78 - 0.02 - 2.68 = -3.48$$

$$-0.78 - 1 - 1.99 = -3.77$$

	5	6	6	6	1	1	2	3	4
F	-1.08	-1.88	-2.68						
L	-1.30	-1.65	-1.99						

Model decoding (Viterbi)

- Example: 566611234. What was the most likely series of dice used to generate this output?



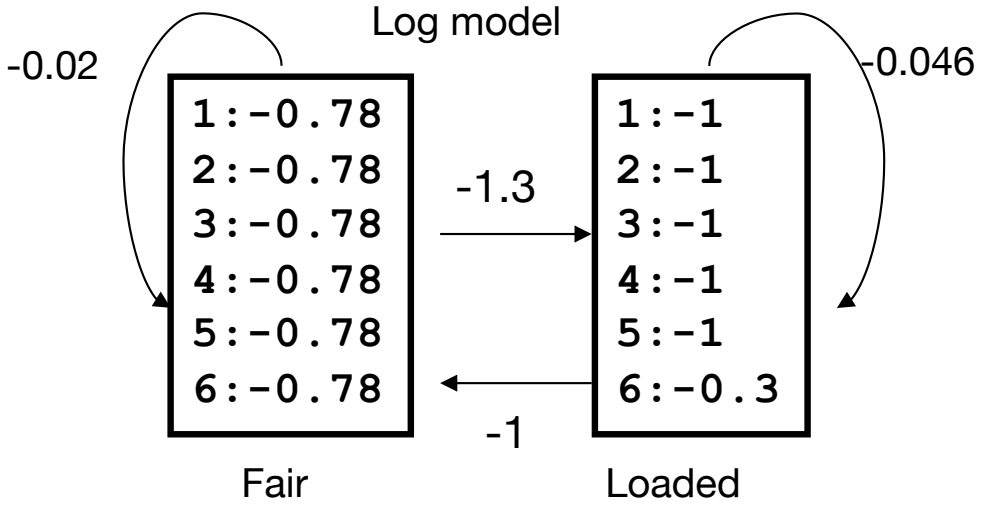
$$-0.78 - 0.02 - 2.68 = -3.48$$

$$-0.78 - 1 - 1.99 = -3.77$$

	5	6	6	6	1	1	2	3	4
F	-1.08	-1.88	-2.68	-3.48					
L	-1.30	-1.65	-1.99						

Model decoding (Viterbi)

- Now we can formalize the algorithm!



$$P_l(i+1) = p_l(i+1) \bullet \max_k(P_k(i) \bullet a_{kl}) \quad or$$

$$\log(P_l(i+1) = \log(p_l(i+1)) + \max_k(\log(P_k(i) + \log(a_{kl}))$$

New match

Old max score for being in state k after having made i

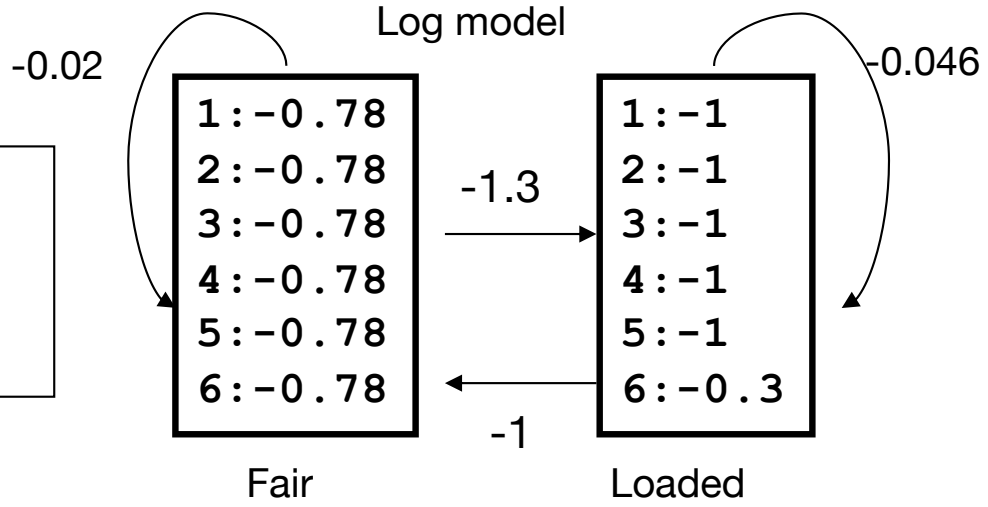
Transition from k to l

Model decoding (Viterbi)

- Example: 566611234. What was the most likely series of dice used to generate this output?

Initialization

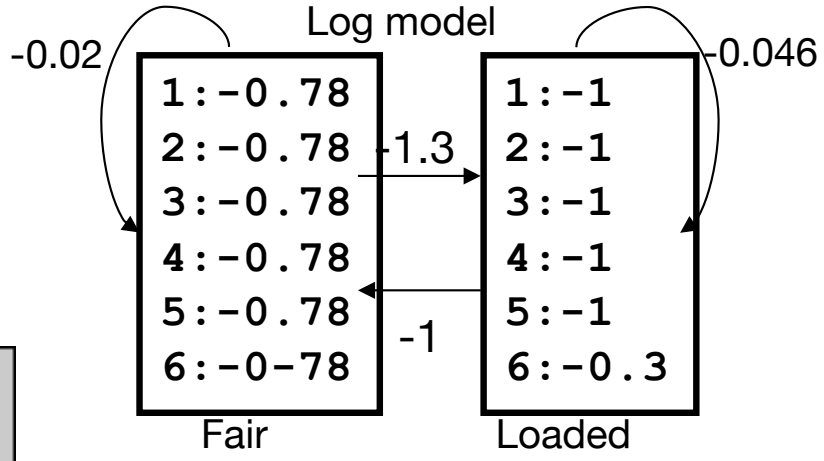
```
F = 0.5*0.167
log(F) = log(0.5) + log(0.167) = -1.08
L = 0.5*0.1
log(L) = log(0.5) + log(0.1) = -1.30
```



	5	6	6	6	1	1	2	3	4
F	-1.08								
L	-1.30								

Model decoding (Viterbi). Can you do it?

- Example: 566611234. What was the mots likely series of dice used to generate this output?
- Fill out the table using the Viterbi recursive algorithm
  - Add the arrows for backtracking
- Find the optimal path



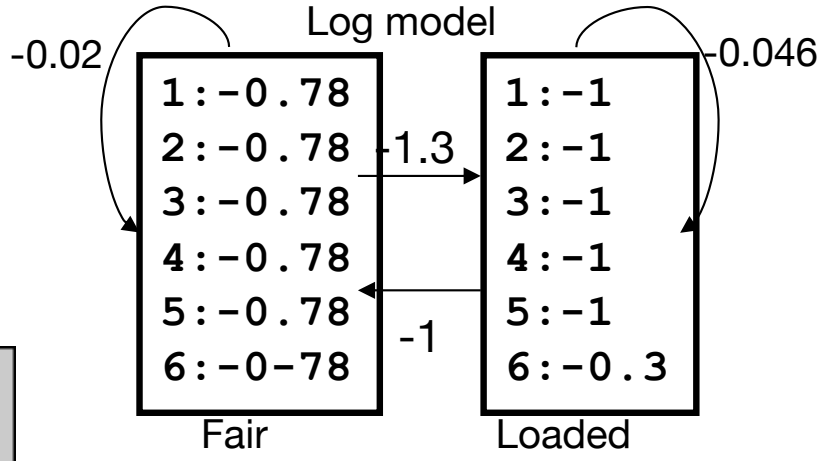
$$P_l(i+1) = p_l(i+1) \bullet \max_k(P_k(i) \bullet a_{kl}) \quad or$$

$$\log(P_l(i+1) = \log(p_l(i+1)) + \max_k(\log(P_k(i) + \log(a_{kl}))$$

	5	6	6	6	1	1	2	3	4
F	-1.08	-1.88	-2.68	-3.48					
L	-1.30	-1.65	-1.99						

Model decoding (Viterbi). Can you do it?

- Example: 566611234. What was the most likely series of dice used to generate this output?
- Fill out the table using the Viterbi recursive algorithm
  - Add the arrows for backtracking
- Find the optimal path

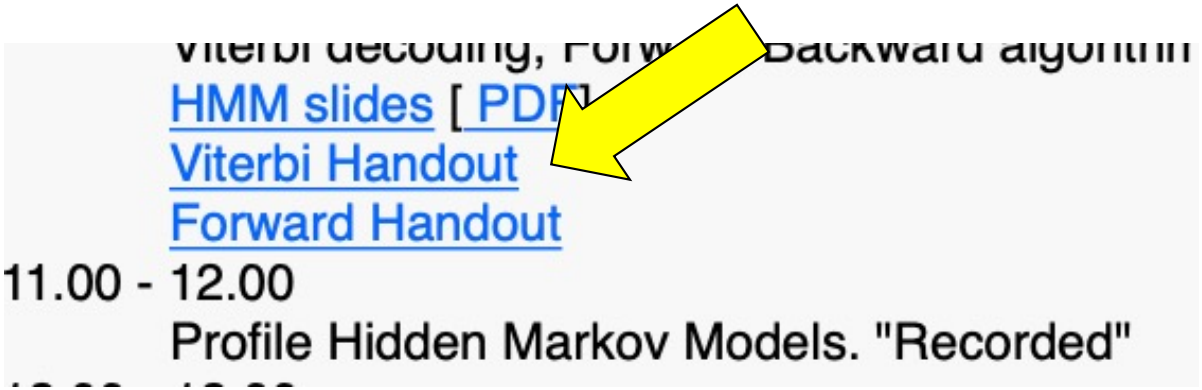


$$P_l(i+1) = p_l(i+1) \bullet \max_k(P_k(i) \bullet a_{kl}) \quad or$$

$$\log(P_l(i+1) = \log(p_l(i+1)) + \max_k(\log(P_k(i) + \log(a_{kl}))$$

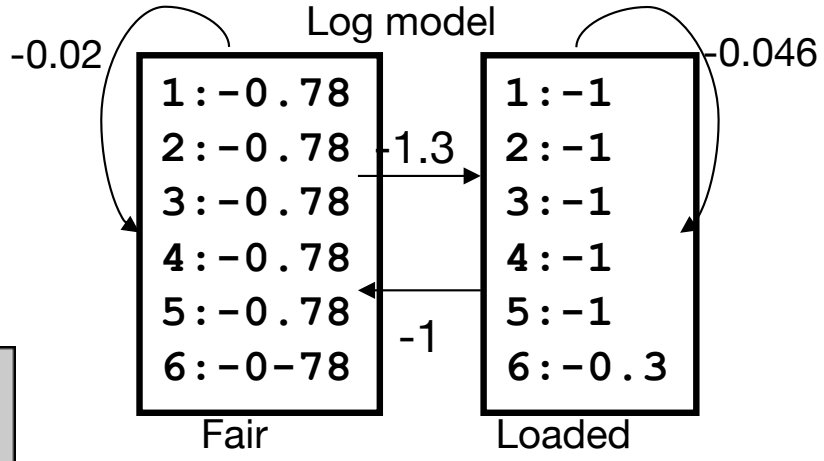
	5	6	6	6	1	1	2	3	4
F	-1.08	-1.88	-2.68	-3.48		-4.92		-6.52	
L	-1.30	-1.65	-1.99		-3.39			-6.53	

Model decoding (Viterbi). Can you do it?



Model decoding (Viterbi). Can you do it?

- Example: 566611234. What was the most lilely series of dice used to generate this output?
- Fill out the table using the Viterbi recursive algorithm
  - Add the arrows for backtracking
- Find the optimal path



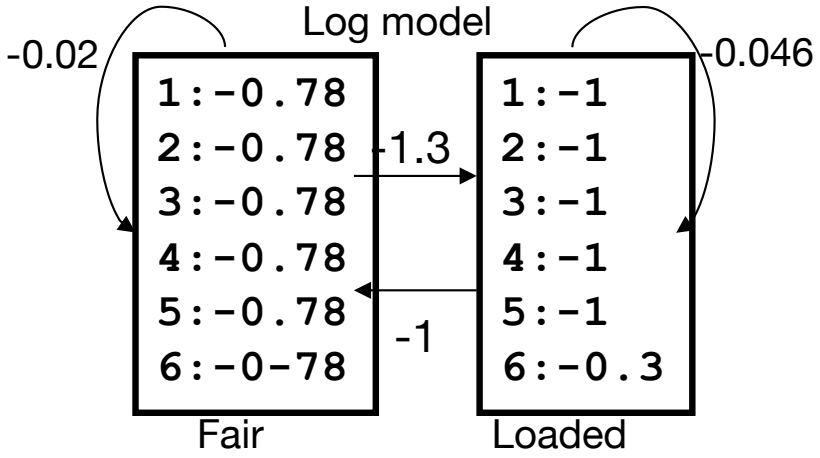
$$P_l(i+1) = p_l(i+1) \bullet \max_k(P_k(i) \bullet a_{kl}) \quad or$$

$$\log(P_l(i+1) = \log(p_l(i+1)) + \max_k(\log(P_k(i) + \log(a_{kl}))$$

	5	6	6	6	1	1	2	3	4
F	-1.08	-1.88	-2.68	-3.48	-4.12	-4.92	-5.72	-6.52	-7.33
L	-1.30	-1.65	-1.99	-2.34	-3.39	-4.44	-5.49	-6.53	-7.57

Model decoding (Viterbi). Can you do it?

- Example: 566611234. What was the most likely series of dice used to generate this output?
- The most likely path is
  - LLLLFFFFF



	5	6	6	6	1	1	2	3	4
F	-1.08	-1.88	-2.68	-3.48	-4.12	-4.92	-5.72	-6.52	-7.33
L	-1.30	-1.65	-1.99	-2.34	-3.39	-4.44	-5.49	-6.53	-7.57

Model decoding (Viterbi).

- What happens if you have three dice?

	5	6	6	6	1	1	2	3	4
F	-1.0								
L1	-1.2								
L2	-1.3								

$$P_l(i+1) = p_l(i+1) \bullet \max_k(P_k(i) \bullet a_{kl}) \quad or$$

$$\log(P_l(i+1) = \log(p_l(i+1)) + \max_k(\log(P_k(i) + \log(a_{kl}))$$

And if you have a trans-membrane model

- What is the most likely path (alignment) of a protein sequence to the model

	D	G	V	L	I	M	A	D	Q
iC	-1.0								
M	-1.2								
xC	-1.3								

$$P_l(i+1) = p_l(i+1) \bullet \max_k(P_k(i) \bullet a_{kl}) \quad or$$

$$\log(P_l(i+1) = \log(p_l(i+1)) + \max_k(\log(P_k(i) + \log(a_{kl}))$$

The Forward algorithm

- The Viterbi algorithm finds the most probable path giving rise to a given sequence
- One other interesting question would be
  - What is the probability that a given sequence can be generated by the hidden Markov model
    - Calculated by summing over all path giving rise to a given sequence

The Forward algorithm

- Calculate summed probability over all path giving rise to a given sequence

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

- The number of possible paths is very large making (once more) brute force calculations infeasible
  - Use dynamic (recursive) programming

The Forward algorithm

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

- Say we know the probability of generating the sequence up to and including x_i ending in state k

$$f_k(i) = P(x_1, x_2, \ldots, x_i, \pi_i = k)$$

- Then the probability of observing the element i+1 of x ending in state l is

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

- where $p_l(x_{i+1})$ is the probability of observing x_{i+1} is state l, and a_{kl} is the transition probability from state k to state l
- Then

$$\boxed{P(x) = \sum_k f_k(L)}$$

Forward algorithm

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

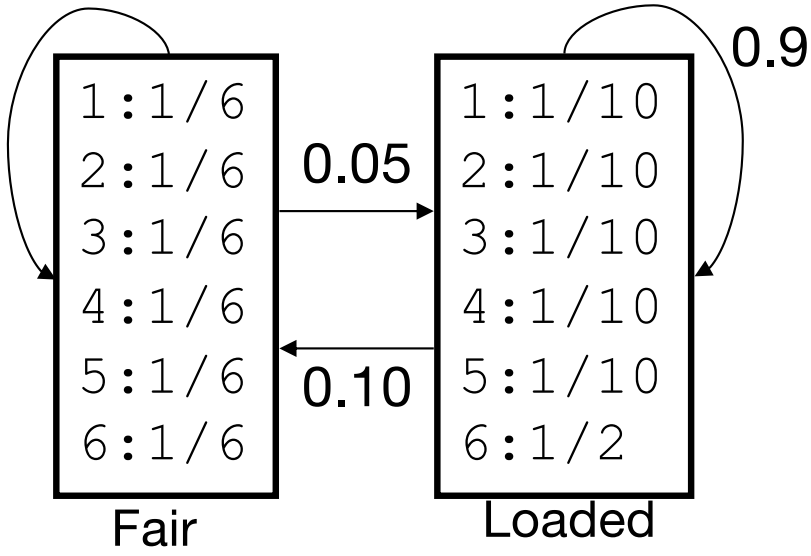
$$f_k(0) = 1$$

$$a_{0l} = \pi_l$$

Initialization

$$f_F(5) = 0.167 \cdot 0.5 = 0.083$$

$$f_L(5) = 0.1 \cdot 0.5 = 0.05$$



	5	6	6	6	1	1	2	3	4
F									
L									

Forward algorithm

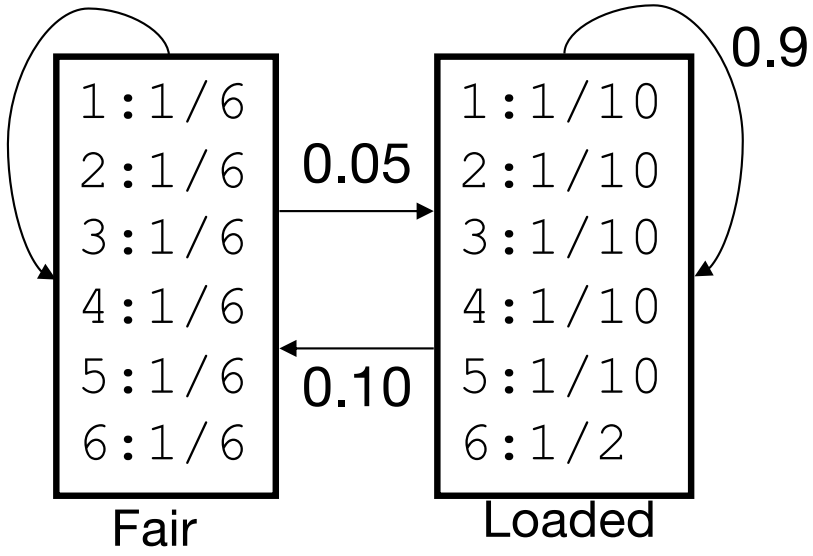
$$f_l(i+1) = p_l(x_{i+1}) \cdot \sum_k f_k(i) \cdot a_{kl}$$

$$f_k(0) = 1$$

$$a_{0l} = \pi_l$$

$$f_F(5) = 0.167 \cdot 0.5 = 0.083$$

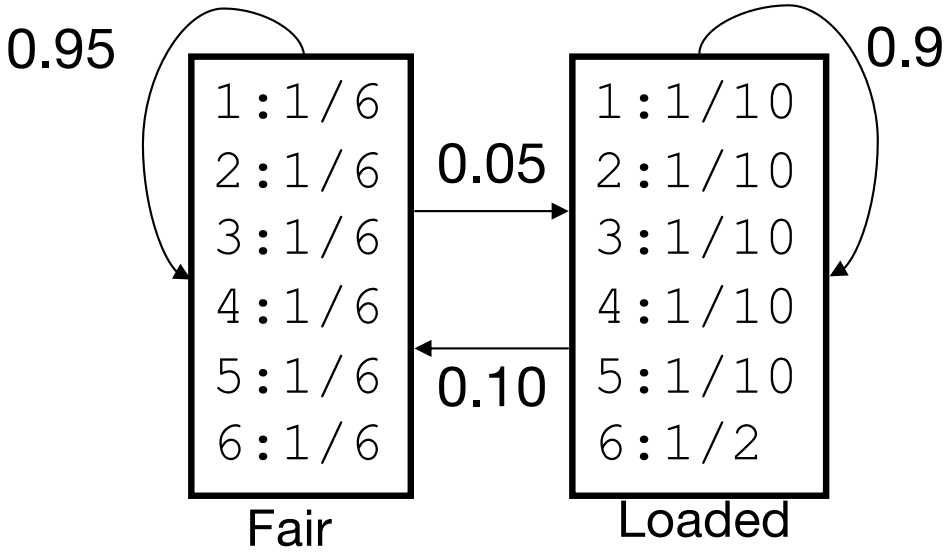
$$f_L(5) = 0.1 \cdot 0.5 = 0.05$$



	5	6	6	6	1	1	2	3	4
F	8.3e-2								
L	5e-2								

Forward algorithm

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

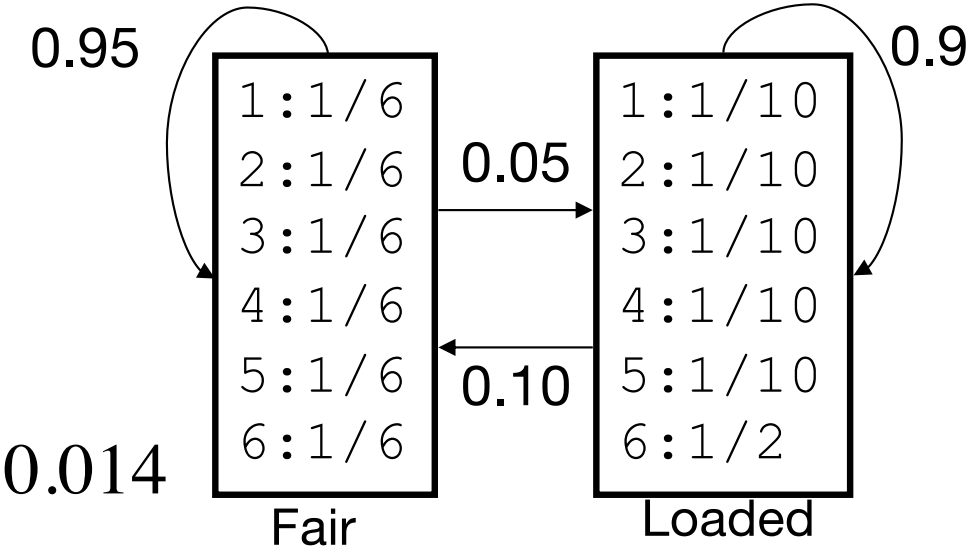


	5	6	6	6	1	1	2	3	4
F	8.3e-2								
L	5e-2								

Forward algorithm

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

$$0.167 \cdot (0.083 \cdot 0.95 + 0.05 \cdot 0.1) = 0.014$$

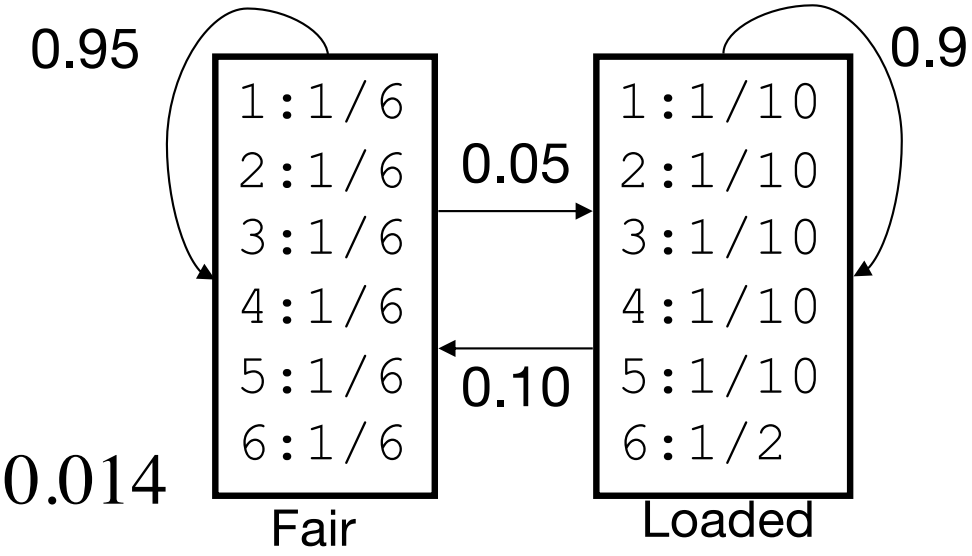


	5	6	6	6	1	1	2	3	4
F	0.083								
L	0.05								

Forward algorithm

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

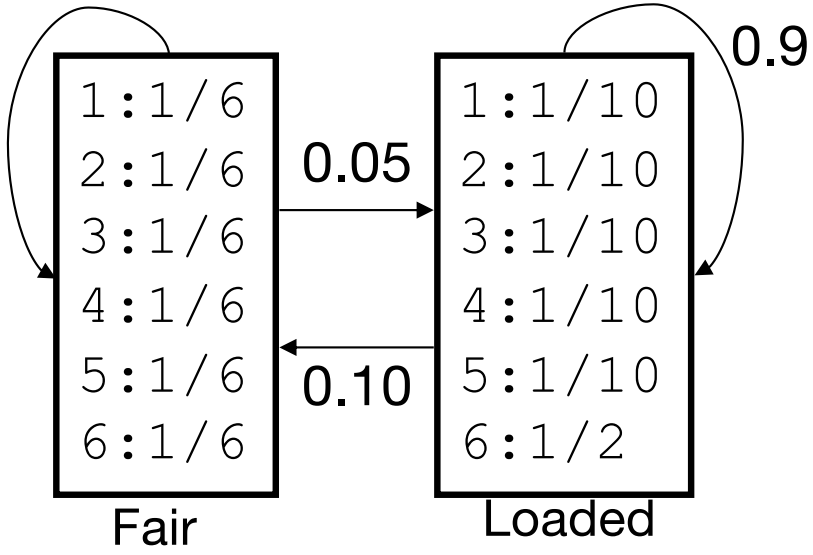
$$0.167 \cdot (0.083 \cdot 0.95 + 0.05 \cdot 0.1) = 0.014$$



	5	6	6	6	1	1	2	3	4
F	0.083	0.014							
L	0.05								

Forward algorithm.
Can you do it yourself?

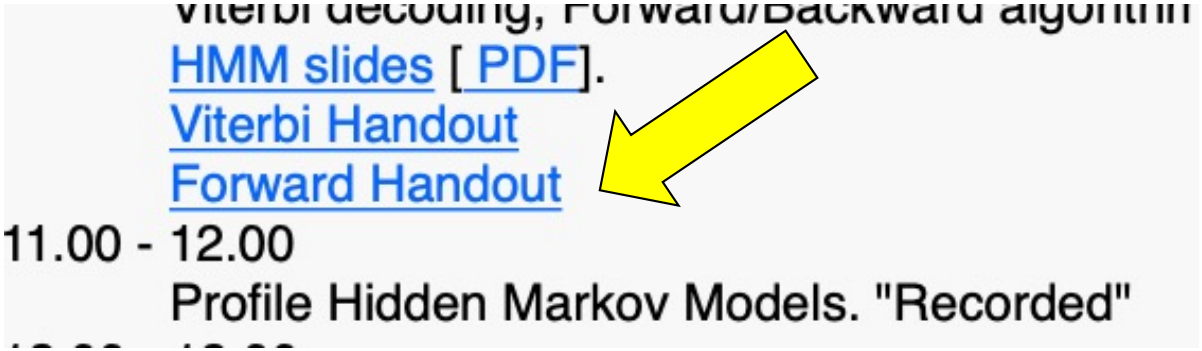
$$f_l(i+1) = p_l(x_{i+1}) \cdot \sum_k f_k(i) \cdot a_{kl}$$



Fill out the empty cells in the table!
What is P(x)?

	5	6	6	6	1	1	2	3	4
F	8.30e-2		2.63e-3	6.08e-4	1.82e-4	3.66e-5		1.09e-6	1.79e-7
L	5.00e-2	2.46e-2	1.14e-2		4.71e-4	4.33e-5		4.00e-7	4.14e-8

Forward algorithm.
Can you do it yourself?

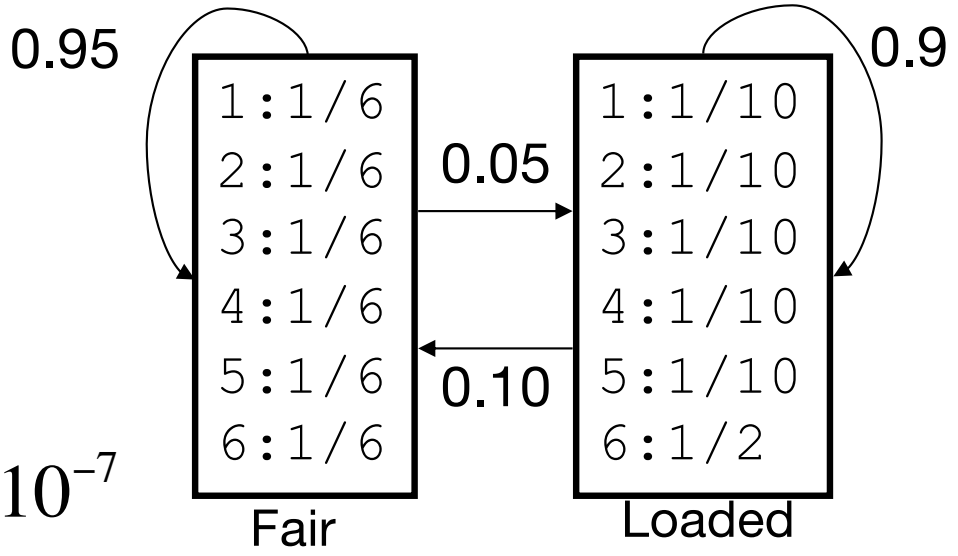


Forward algorithm

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

$$P(x) = \sum_k f_k(L)$$

$$P(x) = (1.79 + 0.414) \cdot 10^{-7} = 2.2 \cdot 10^{-7}$$



	5	6	6	6	1	1	2	3	4
F	8.30e-2	1.40e-2	2.63e-3	6.08e-4	1.82e-4	3.66e-5	6.48e-6	1.09e-6	1.79e-7
L	5.00e-2	2.46e-2	1.14e-2	5.20e-3	4.71e-4	4.33e-5	4.08e-6	4.00e-7	4.14e-8

The Posterior decoding (Backward algorithm)

- One other interesting question would be
  - What is the probability that an observation x_i came from a state k given the observed sequence x

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

The Backward algorithm

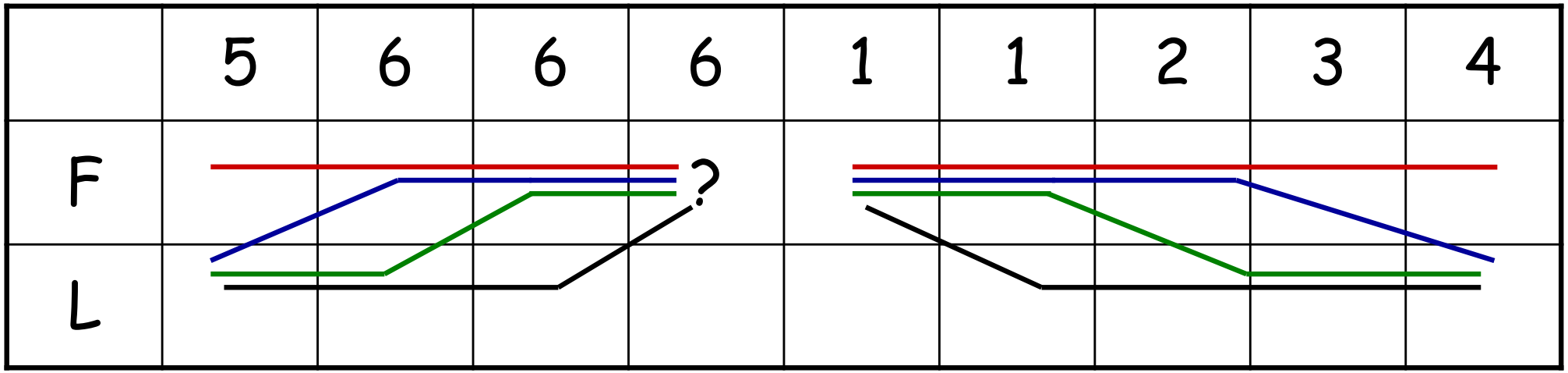
$$P(x,\pi_i = k) = P(x_1,x_2,\ldots,x_i,\pi_i = k)\cdot P(x_{i+1},\ldots,x_L \mid \pi_i = k)$$

The probability of generation the sequence up to and including x_i ending in state k

Forward algorithm!

The probability of generation the rest of the sequence starting from state k

Backward algorithm!



The Backward algorithm

$$P(x,\pi_i = k) = P(x_1,x_2,...,x_i,\pi_i = k)\cdot P(x_{i+1},...,x_L \mid \pi_i = k)$$

$$P(x,\pi_i = k) = f_k(i)\cdot b_k(i)$$

$$f_k(i) = P(x_1,x_2,..,x_i,\pi_i = k)$$

$$f_k(i) = p_k(x_i)\cdot \sum_l f_l(i-1)\cdot a_{lk}$$

← Forward algorithm

$$b_k(i) = P(x_{i+1},x_{i+2},...,x_L \mid \pi_i = k)$$

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

← Backward algorithm

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

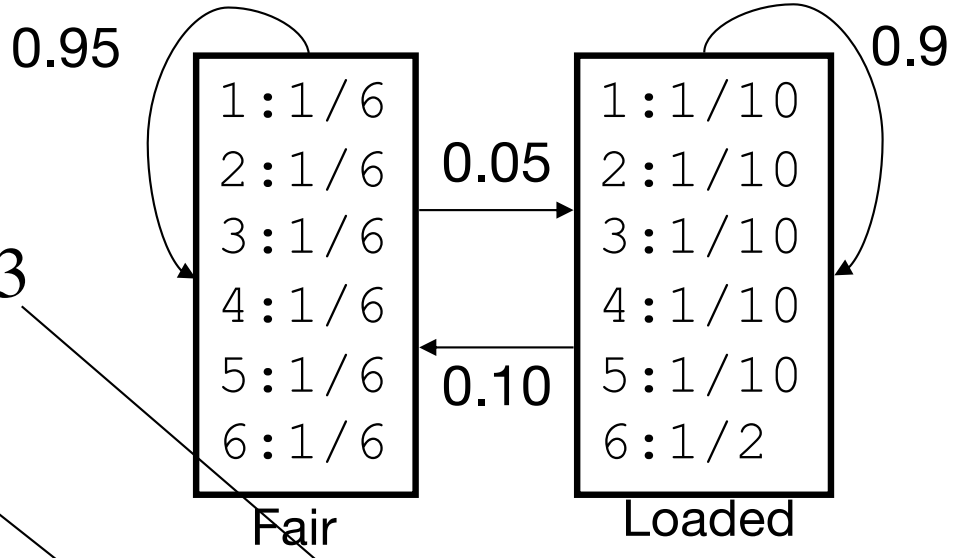
← Forward-Backward algorithm

Backward algorithm

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

$$0.167 \cdot 0.95 \cdot 1 + 0.1 \cdot 0.05 \cdot 1 = 0.163$$

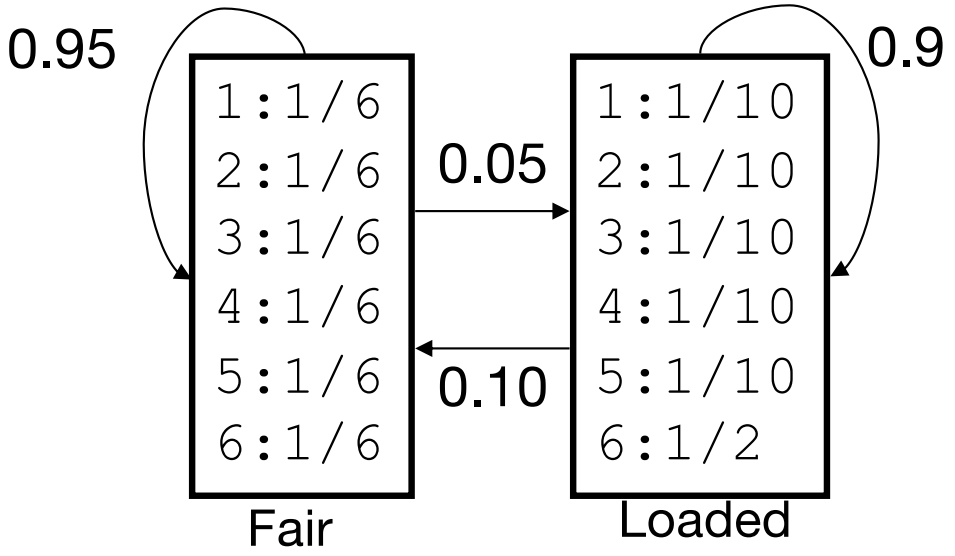
$$0.167 \cdot 0.1 \cdot 1 + 0.1 \cdot 0.9 \cdot 1 = 0.107$$



	5	6	6	6	1	1	2	3	4
F								0.163	1
L								0.107	1

Backward algorithm

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



	5	6	6	6	1	1	2	3	4
F	6.83e-7	3.18e-6	1.76e-5	1.07e-5	6.69e-4	4.19e-3	2.61e-2	0.163	1
L	3.27e-6	7.14e-6	1.52e-5	2.98e-5	2.08e-4	1.54e-3	1.23e-2	0.107	1

Backward algorithm

- Note that the sum of first column of the backward matric is NOT equal to the sum of the last column of the forward matrix
- This is because the first column of the backward matrix gives the probability values of generating the sequence AFTER having generated the first observation
- You hence cannot get the P(x) value directly from the backward matrix

Posterior decoding

- What is the posterior probability that observation x_i came from state k given the observed sequence X?

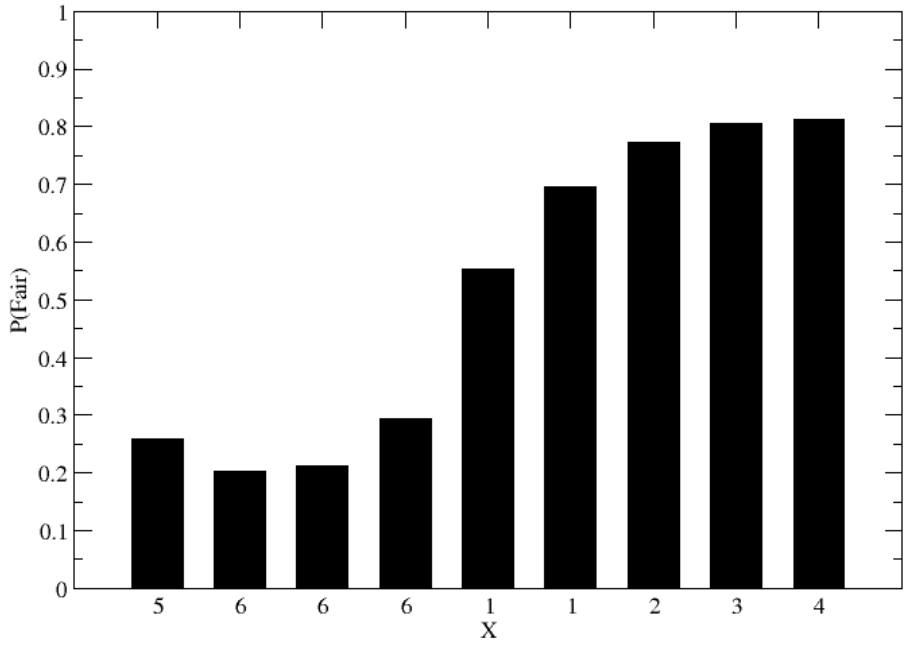
or

- What is the probability that a given amino acid is part of the trans-membrane helix given the protein sequence is X?

Posterior decoding

- What is the posterior probability that observation x_i came from state k given the observed sequence X.

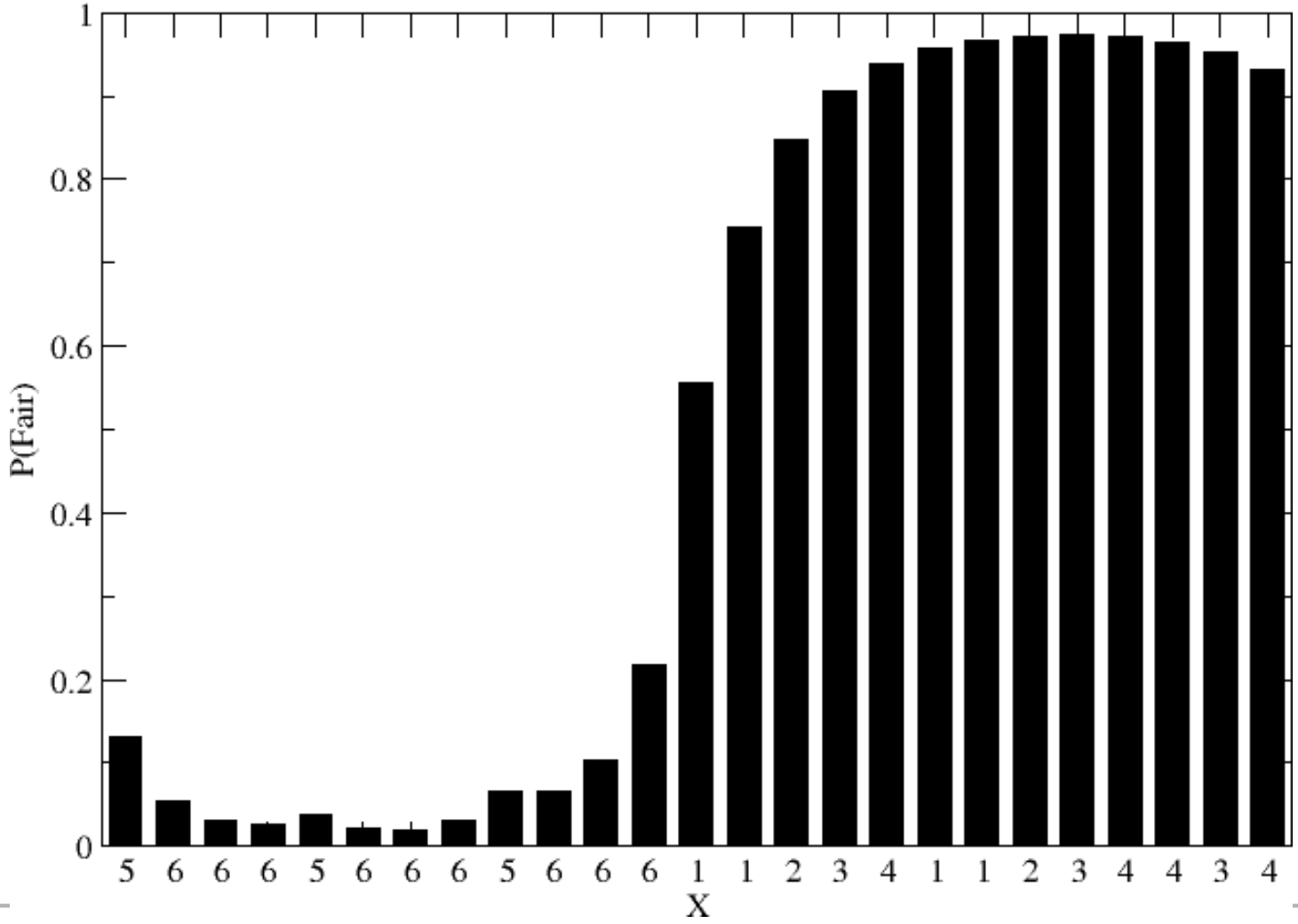
$$P(\pi_i = k \mid x) = \frac{P(x, \pi_i = k)}{P(x)} = \frac{f_k(i) \cdot b_k(i)}{P(x)}$$



Posterior decoding

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

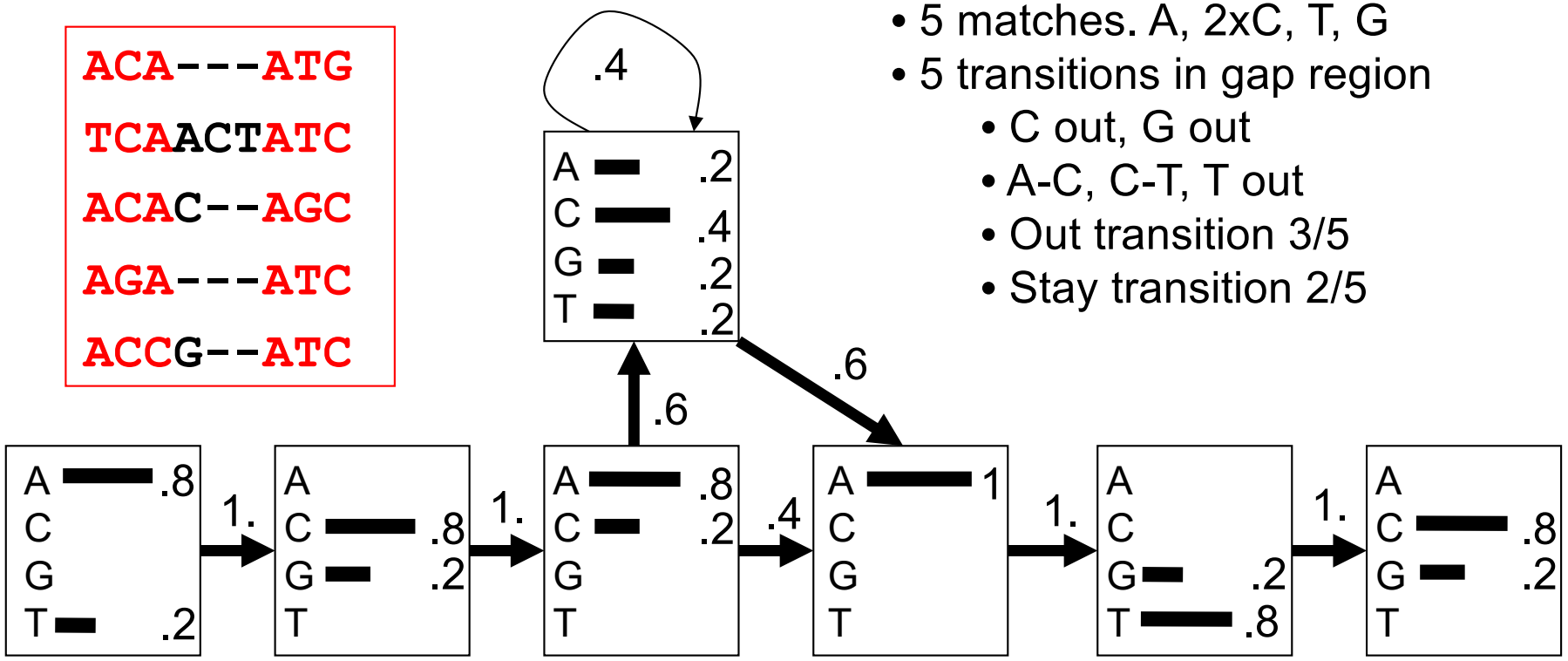
The probability is context dependent



Training of HMM

- Supervised training
  - If each symbol is assigned to one state, all parameters can be found by simply counting number of emissions and transitions as we did for the DNA model
- Un-supervised training
  - We do not know to which state each symbol is assigned so another approach is needed
  - Find emission and transition parameters that most likely produces the observed data
  - Baum-Welsh does this

Supervised learning



- 5 matches. A, 2xC, T, G
- 5 transitions in gap region
  - C out, G out
  - A-C, C-T, T out
  - Out transition 3/5
  - Stay transition 2/5

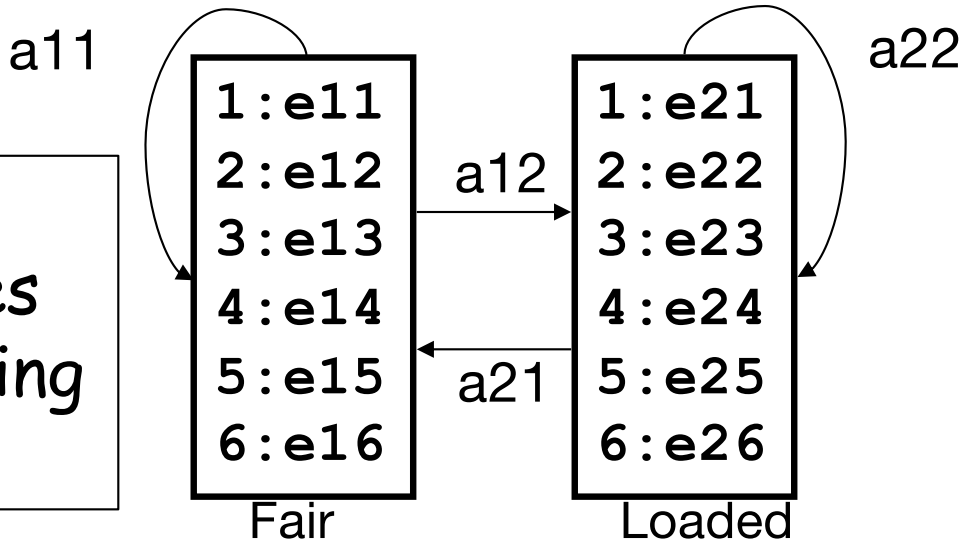
ACA---ATG 0.8x1x0.8x1x0.8x0.4x1x1x0.8x1x0.2 = 3.3×10^{-2}

Un-supervised learning

CENTERFO
RBIOLOGI
CALSEQU
ENCEANA
LYSIS CBS

3124536666414566675435666666331234

Can we find the model parameters that optimizes the probability of observing these sequences?



Baum-Welsh

The probability of being in state k at time i, and state l at time i + 1, given the model and the observation sequence is

$$\varepsilon_{kl}^{i} = \frac{1}{P(x)} \cdot f_k(i) \cdot a_{kl} \cdot e_l(x_{i+1}) \cdot b_l(i+1)$$

The probability of being in state k at time i, given the observation sequence O is

$$\gamma_k^i = \frac{1}{P(x)} \cdot f_k(i) \cdot b_k(i)$$

Note

$$\gamma_k^i = \sum_l \varepsilon_{kl}^i$$

Baum-Welsh

Now

$$p(k) = \sum_{i=1}^{T-1} \gamma_k^i$$

is the expected number of times that state k is visited, or the expected number of transitions made from state k (given the observed sequence), and

$$\sum_{i=1}^{T-1} \varepsilon_{kl}^i$$

is the expected number of transitions from state k to state l (given the observed sequence)

Baum-Welsh

Now

$$\overline{a_{kl}} = \frac{\sum_{i=1}^{T-1} \varepsilon_{kl}^{i}}{\sum_{i=1}^{T-1} \gamma_{k}^{i}}$$

Estimate probability of transition between state k and l

and

$$\overline{e_{k}^{a}} = \frac{\sum_{i=1, O_i = v_a}^{T} \gamma_{k}^{i}}{\sum_{i=1}^{T} \gamma_{k}^{i}}$$

Estimate probability emitting symbol a in state k

Baum-Welsh

Use these relations

$$\overline{a_{kl}} = \frac{\sum_{i=1}^{T-1} \varepsilon_{kl}^{i}}{\sum_{i=1}^{T-1} \gamma_{k}^{i}} \qquad \overline{e_{k}^{a}} = \frac{\sum_{i=1, O_i = v_a}^{N-1} \gamma_{k}^{i}}{\sum_{i=1}^{N-1} \gamma_{k}^{i}}$$

To update a, and e, and iterate until convergence

HMM's and weight matrices

- In the case of un-gapped alignments HMM's become simple weight-matrices
- To achieve high performance, the emission frequencies are estimated using the techniques of
  - Sequence weighting
  - Pseudo counts

Profile HMM's

- Alignments based on conventional scoring matrices (BLOSUM62) scores all positions in a sequence in an equal manner
- Some positions are highly conserved, some are highly variable (more than what is described in the BLOSUM matrix)
- Profile HMM's are ideal suited to describe such position specific variations

Sequence profiles

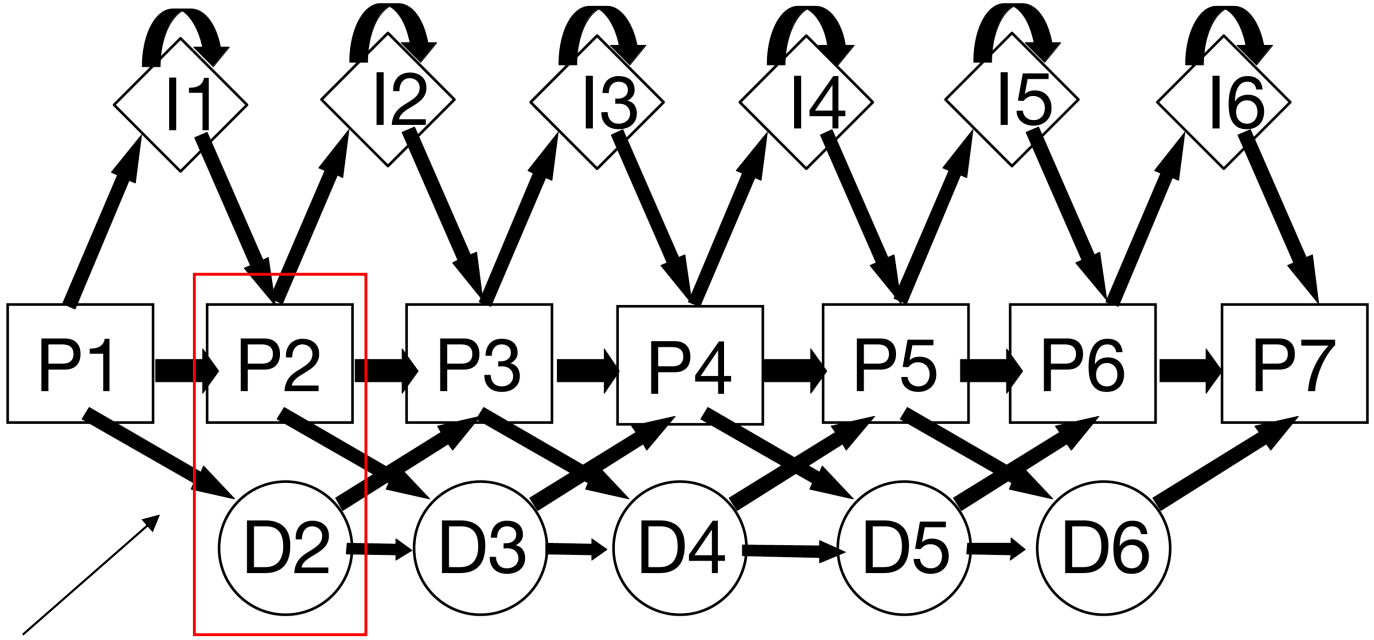
Matching any thing
but G => large
negative score

Any thing can match

HMM vs. alignment

- Detailed description of core
  - Conserved/variable positions
- Price for insertions/deletions varies at different locations in sequence
- These features cannot be captured in conventional alignments

Profile HMM's



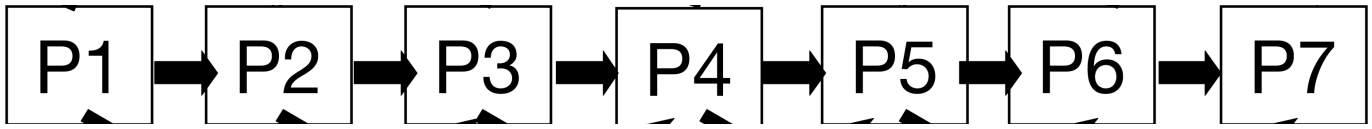
All P/D pairs must be visited once

$L_1 - Y_2A_3V_4R_5 - I_6$

$P_1D_2P_3P_4I_4P_5D_6P_7$

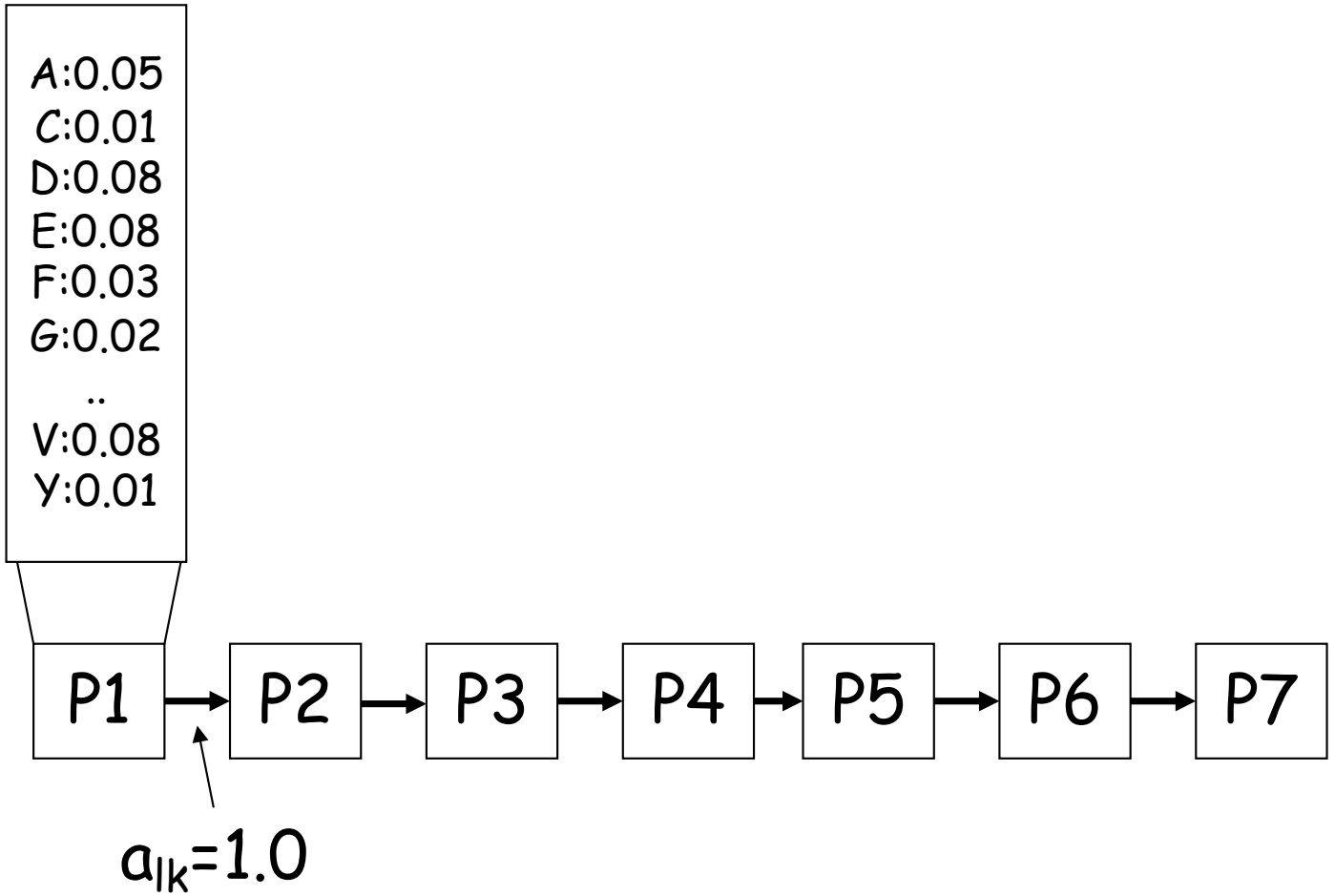
Profile HMM

- Un-gapped profile HMM is just a sequence profile



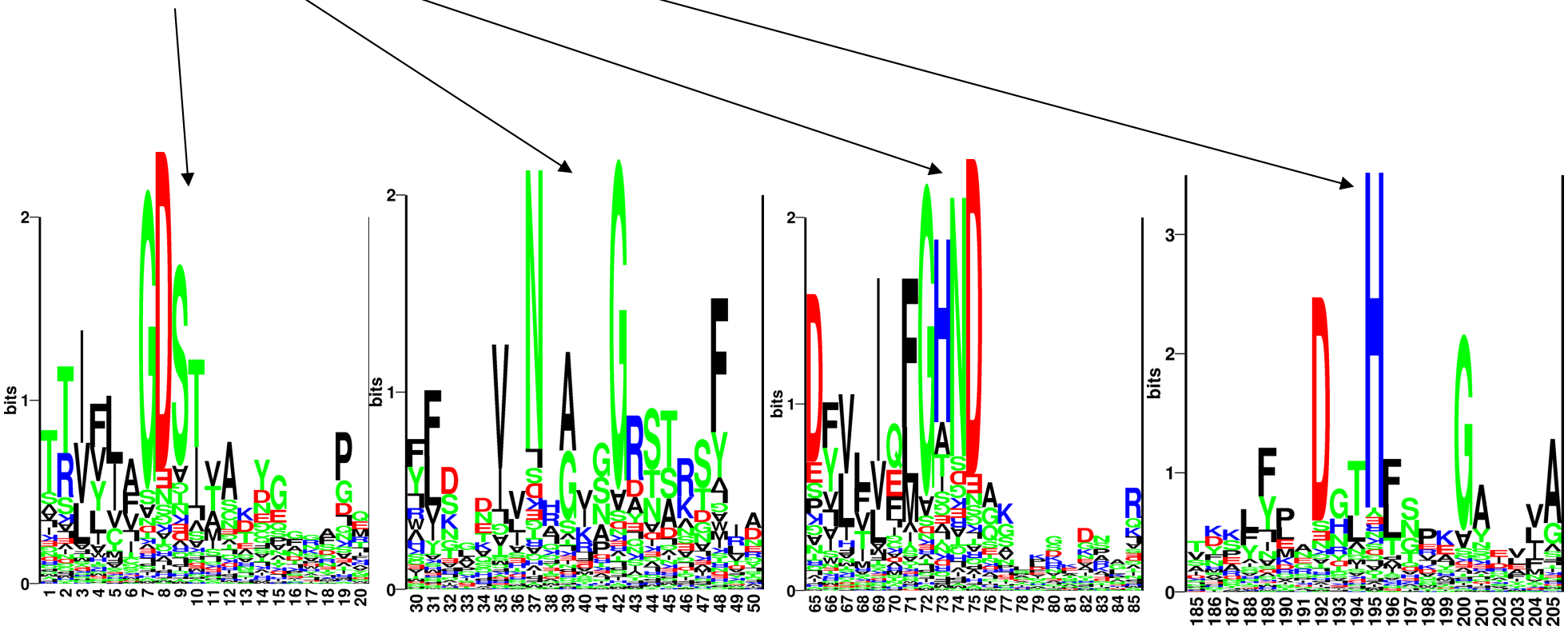
Profile HMM

- Un-gapped profile HMM is just a sequence profile



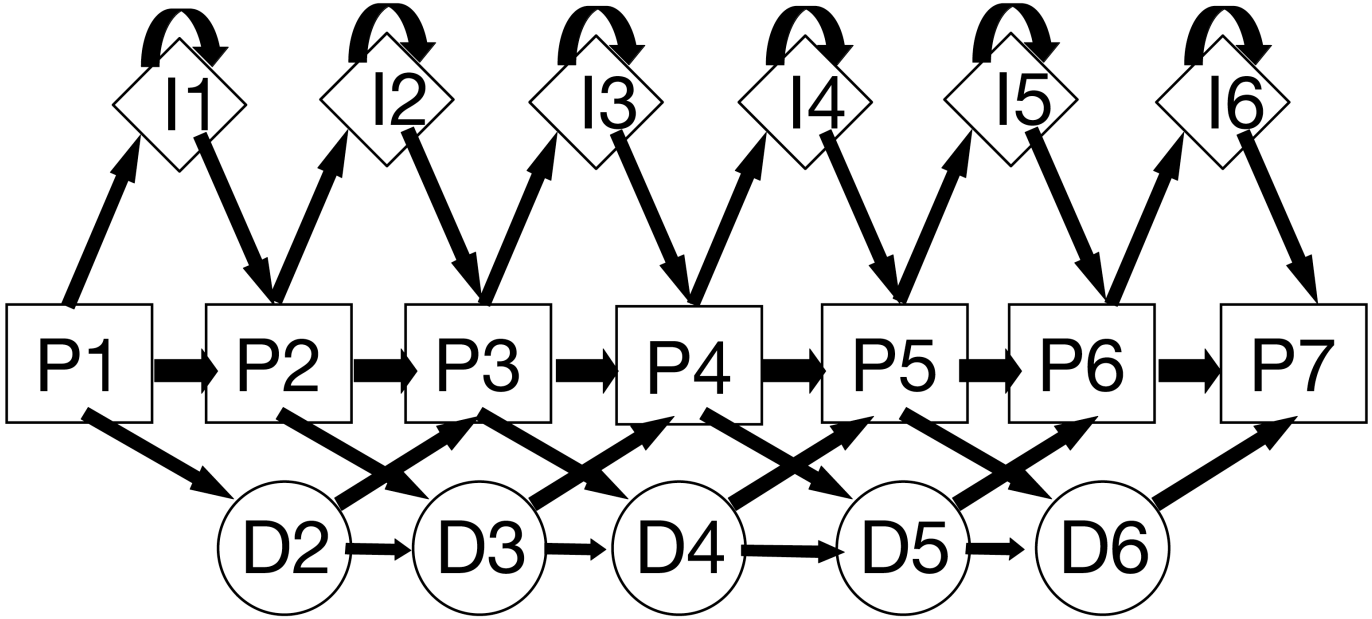
Example. Where is the active site?

- Sequence profiles might show you where to look!
- The active site could be around
  - S9, G42, N74, and H195



Profile HMM

- Profile HMM (deletions and insertions)



Profile HMM (deletions and insertions)

The HMMer program

- HMMer is a open source program suite for profile HMM for biological sequence analysis
- Used to make the Pfam database of protein families
  - http://pfam.sanger.ac.uk/

A HMMer example

- Example from the CASP8 competition
- What is the best PDB template for building a model for the sequence T0391

```
>T0391 rieske ferredoxin, mouse, 157 residues
SDPEISEQDEEKKKYTSVCVGREEDIRKSERMTAVVHDREVVIFYHKGEYHAMDIRCYHS
GGPLHLGEIEDFNGQSCIVCPWHKYKITLATGEGLYQSINPKDPSAKPKWCSKGVKQRIH
TVKVDNGNIYVTLSKEPFKCDSDYYATGEFKVIQSSS
```

A HMMer example

- What is the best PDB template for building a model for the sequence T0391
- Use Blast
  - No hits
- Use Psi-Blast
  - No hits
- Use Hmmer

A HMMer example

- Use Hmmer
  - Make multiple alignment using Blast
  - Make model using
    - hmmbuild
  - Find PDB template using
    - hmmsearch

A HMMer example

- Make multiple alignment using Blast

  blastpgp -j 3 -e 0.001 -m 6 -i T0391.fsa -d sp -b 10000000 -v 10000000 > T0391.fsa.blastout

- Make Stockholm format

```
# STOCKHOLM 1.0
QUERY DPEISEQDEEKKKYTSVCVGREEDIRKS-ERMTAVVHD-RE--V-V-IF--Y-H-KGE-Y
Q8K2P6 DPEISEQDEEKKKYTSVCVGREEDIRKS-ERMTAVVHD-RE--V-V-IF--Y-H-KGE-Y
Q8TAC1 ----SAQDPEKREYSSVCVGREDDIKKS-ERMTAVVHD-RE--V-V-IF--Y-H-KGE-Y
```

- Build HMM

  hmmbuild T0391.hmm T0391.fsa.blastout.sto

- Search for templates in PDB

  hmmsearch T0391.hmm pdb > T0391.out

A HMMer example

CENTERFO
RBIOLOGI
CALSEQU
ENCEANA
LYSIS CBS

Sequence	Description	Score	E-value	N
2E4Q.A	mol:aa ELECTRON TRANSPORT	163.7	6.7e-45	1
2E4P.B	mol:aa ELECTRON TRANSPORT	163.7	6.7e-45	1
2E4P.A	mol:aa ELECTRON TRANSPORT	163.7	6.7e-45	1
2E4Q.C	mol:aa ELECTRON TRANSPORT	163.7	6.7e-45	1
2YVJ.B	mol:aa OXIDOREDUCTASE/ELECTRON TRANSPORT	163.7	6.7e-45	1
1FQT.A	mol:aa OXIDOREDUCTASE	160.9	4.5e-44	1
1FQT.B	mol:aa OXIDOREDUCTASE	160.9	4.5e-44	1
2QPZ.A	mol:aa METAL BINDING PROTEIN	137.3	5.6e-37	1
2Q3W.A	mol:aa ELECTRON TRANSPORT	116.2	1.3e-30	1
1VM9.A	mol:aa ELECTRON TRANSPORT	116.2	1.3e-30	1

Validation. CE structural alignment


```
CE 2E4Q A 3D89 A (run on IRIX machines at CBS)

Structure Alignment Calculator, version 1.01, last modified: May 25,
   2000.

CE Algorithm, version 1.00, 1998.

Chain 1: /usr/cbs/bio/src/ce_distr/data.cbs/pdb2e4q.ent:A (Size=109)
Chain 2: /usr/cbs/bio/src/ce_distr/data.cbs/pdb3d89.ent:A (Size=157)

Alignment length = 101 Rmsd = 2.03A Z-Score = 5.5 Gaps = 20(19.8%)
   CPU = 1s Sequence identities = 18.1%

Chain 1:    2 TFTKACSVDEVPPGEALQVSHDAQKVAIFNVDGEFFATQDQCTHGEWSLSEGGYLDG----DVVECSLHM
Chain 2:   16 TSVCVGREEDIRKSERMTAVVHDREVVIFYHKGEYHAMDIRCYHSGGPLH-LGEIEDFNGQSCIVCPWHK

Chain 1:   68 GKFCVRTGKVKS-----PPPC---------EPLKVYPIRIEGRDVLVDFSRAALH
Chain 2:   85 YKITLATGEGLYQSINPKDPSAKPKWCSKGVKQRIHTVKVDNGNIYVTL-SKEPF
```

HMM packages

- HMMER (http://hmmer.wustl.edu/)
  - S.R. Eddy, WashU St. Louis. Freely available.
- SAM (http://www.cse.ucsc.edu/research/compbio/sam.html)
  - R. Hughey, K. Karplus, A. Krogh, D. Haussler and others, UC Santa Cruz. Freely available to academia, nominal license fee for commercial users.
- META-MEME (http://metameme.sdsc.edu/)
  - William Noble Grundy, UC San Diego. Freely available. Combines features of PSSM search and profile HMM search.
- NET-ID, HMMpro (http://www.netid.com/html/hmmpro.html)
  - Freely available to academia, nominal license fee for commercial users.
  - Allows HMM architecture construction.

- EasyGibbs (http://www.cbs.dtu.dk/biotools/EasyGibbs/)
  - Webserver for Gibbs sampling of proteins sequences