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

States: Three states - sunny, cloudy, rainy.

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

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

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

ALLYVDWQILPVIL
Weight matrix construction

SLLPAIVEL YLLPAIVHI TLWVDPYEV GLVPFLVSV KLLEPVLLL LLDVPTAAV LLDVPTAAV LLDVPTAAV
LLDVPTAAV VFRLGGPRG MVDTLLLL YMNGTMSQV MLSVPLLL SLLGLLVEV ALLPPNIL TLIKIQHTL
HLIDYLVTS ILAPPVVKL ILFQQLVLIL GILGFVFTL STNRQSGRO GLDVLTAKV RILGAVAKV QVCRIFTI
IIIFGHENRV ILMEHIKLL ILDQKINEV SLAGGIIGV LLEIENVSE FLLWATAEA SLDFBGSY KKRERRAPSIL
LERPQGNEI ALSNEVKLIL ANELQLHV DLERVESL FLQGENISNF LADSHHTYL GLESEFTEYLS TSTAPAHGV
PLDGEYFTL GVLVGGALI RTLDKVLEV HLSTAFARV RLDSYVRSL YMNGTMSQV GILGFVFTL ILKEFHVGV
ILGFVFTLT LLFGYPVYV GLSPTVWS WLLSLPQFV FLPSDFPS CLGGLLTMV FIAGNSAYE KLGFYQNM
KLVALGINA DLMGYICLV MLLAVYCL ALLPPNIL TLIKIQHTL HLIDYLVTS ILAPPVVKL
GIGLAGIVL IAIAGILAI LIVGILAA VDGIGILT VAGIGVLT AAGIIGIQI QAGIGILLA
KARDPHSG KACDPSGHSV SPLLYTAVL RGPQGAVT NLPVMATV GLHCQELVL FPQHFQTV
AVFDRKSDA LLDFVRFMGL VLVKSPNH LLVFQCHI LLGRNSFEV PLTFGVYKVL VLEWRFSDR TINAWKVV
GLCTLVAML FIDSYIQV IIASVVGIL VMAGVGSPY LLWTLVVL SVPDRLALL LMDGCSGSI CTTSTQQLV
VLHDDLLEA LMWITQCFEL SSLMWITQC LSNTYTLAL RGPQGAVT NLPVMATV GLHCQELVL FPQHFQTV
ISNDVCAQV VKTDGNPPE SVYDFWVLFL FLYGALLA VLLFSSDFR LMWAKIGPV SLLLELEEE VLSSRFSGWA
YTAFTPISI RLMSQDSFPL RLPPIFSCS FLWGFRAAY VLLQETELV SLGFIDFV SLDQSVVEL RLMMTIFYI
NMFTPNYGV LMIIPILNIV TLFQGHSV SLTVTTTVF VLLWATL LIKLFLHLW VLLTPIVVL
VVVGVFMPF IFHNYGAEISSL MIMVKCEMI MLGTQMEV MLGTQMEV SLADTSNASL LWAARPLVL GVALQTMQK
GLYDGMEHL KVMEVHLFL YLQVVFAL LAMQAEALA LMQAEALAF VYDGREHTV YLSSGALNRL MFNPVLYL
EEAGIGILTV LTLDSQVSM LSTPPPTVRK KVAELVHFL IMGVGVLGV ALCRWGLLL LLLFAGVQCV VLLCESTAV
YLSFARV YLLEMLWRL SLDDYNHLV RTLDKVLivol GLVEYVQV KLIAMNTRV FIIYAGSLSA KLVANNTRL
FLDEFMFGV ALQPGTALL VLGLDGLVLL SLYSFPEPE ALYVDSLFF SLLQHLLG ELTLLGELFLK MIAVYDKDL
AAAGILTV FLPSDFPS SVRDRLARL LSRLLVRRI LISSWIRLLA AAGIGILTV AVPDEIPPL FAYDGKDYI
AAAGILTV FLPSDFPS AAIGILTV FLPSDFPS AAIGILTV FLWGRPLV ETBSEQSNV 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

\[ S = \sum_i S(a_i) \]

Probability of observation given Model

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

Probability of observation given Prior

\[ S = \log \left( \frac{P(a | M)}{P(a | B)} \right) \]
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

<table>
<thead>
<tr>
<th>Query</th>
<th>Accession</th>
<th>Protein</th>
<th>GenBank</th>
<th>Ensembl</th>
<th>SwissProt</th>
<th>GenPept</th>
<th>TrEMBL</th>
<th>PDB</th>
<th>EMBL</th>
<th>PRF</th>
<th>RefSeq</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAMDIRCYHSGG</td>
<td>Q8K2P6</td>
<td>HAMDIRCYHSGG</td>
<td>-PLHL-GEI-EDFNGQSCIVCPWHKYKITLATGE-GLYQ</td>
<td>QSINPKDPS</td>
<td>QSINPKDPS</td>
<td>QSINPKDPS</td>
<td>QSINPKDPS</td>
<td>QSINPKDPS</td>
<td>QSINPKDPS</td>
<td>QSINPKDPS</td>
<td>QSINPKDPS</td>
</tr>
<tr>
<td>HAMDIRCYHSGG</td>
<td>Q8TAC1</td>
<td>HAMDIRCYHSGG</td>
<td>-PLHL-GEI-EDFNGQSCIVCPWHKYKITLATGE-GLYQ</td>
<td>QSINPKDPS</td>
<td>QSINPKDPS</td>
<td>QSINPKDPS</td>
<td>QSINPKDPS</td>
<td>QSINPKDPS</td>
<td>QSINPKDPS</td>
<td>QSINPKDPS</td>
<td>QSINPKDPS</td>
</tr>
<tr>
<td>FAVQDTCTHDGW</td>
<td>Q07947</td>
<td>FAVQDTCTHDGW</td>
<td>ALSE-GYI-DGD-VECTLHFHKFCVRTGK-VKL-PA</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>YATDNLCHTGS</td>
<td>P0A185</td>
<td>YATDNLCHTGS</td>
<td>RMSD-GYI-EGRE-IETPLHQGRFDVCTGK-ALC-APV</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>YATDNLCHTGS</td>
<td>P0A186</td>
<td>YATDNLCHTGS</td>
<td>RMSD-GYI-EGRE-IETPLHQGRFDVCTGK-ALC-APV</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>YATDNLCHTGS</td>
<td>Q51493</td>
<td>YATDNLCHTGS</td>
<td>RMSD-GFI-EGRE-IETPLHQGRFDVCTGK-ALC-APV</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FAVQDTCTHDGW</td>
<td>A5W4F0</td>
<td>FAVQDTCTHDGW</td>
<td>ALSD-GYI-DGD-VECTLHFHKFCVRTGK-VKL-PA</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FAVQDTCTHDGW</td>
<td>P0C620</td>
<td>FAVQDTCTHDGW</td>
<td>ALSD-GYI-DGD-VECTLHFHKFCVRTGK-VKL-PA</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FAVQDTCTHDGW</td>
<td>P08086</td>
<td>FAVQDTCTHDGW</td>
<td>ALSD-GYI-DGD-VECTLHFHKFCVRTGK-VKL-PA</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FATQDCTHGEW</td>
<td>Q52440</td>
<td>FATQDCTHGEW</td>
<td>SLD-GYI-LGD-VECSLMGKFCVRTGK-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FAVDDRCSHGA</td>
<td>Q7N4V8</td>
<td>FAVDDRCSHGA</td>
<td>SISE-GYI-ED-NATVECPHTASFCLRTGK-ALCL-PA</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FATQDCTHGDW</td>
<td>P37332</td>
<td>FATQDCTHGDW</td>
<td>SLD-GYI-EGD-VECSLMGKFCVRTGK-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>YAINDRCSHGN</td>
<td>A7ZPY3</td>
<td>YAINDRCSHGN</td>
<td>SMSE-GYI-EDD-ATVECPHLAASFCLKTGK-ALCL-PA</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>YAINDRCSHGN</td>
<td>P0ABW1</td>
<td>YAINDRCSHGN</td>
<td>SMSE-GYI-EDD-ATVECPHLAASFCLKTGK-ALCL-PA</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>YAINDRCSHGN</td>
<td>A8A346</td>
<td>YAINDRCSHGN</td>
<td>SMSE-GYI-EDD-ATVECPHLAASFCLKTGK-ALCL-PA</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>YAINDRCSHGN</td>
<td>P0ABW0</td>
<td>YAINDRCSHGN</td>
<td>SMSE-GYI-EDD-ATVECPHLAASFCLKTGK-ALCL-PA</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>YAINDRCSHGN</td>
<td>P0ABW2</td>
<td>YAINDRCSHGN</td>
<td>SMSE-GYI-EDD-ATVECPHLAASFCLKTGK-ALCL-PA</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>YAINDRCSHGN</td>
<td>Q3YZ13</td>
<td>YAINDRCSHGN</td>
<td>SMSE-GYI-EDD-ATVECPHLAASFCLKTGK-ALCL-PA</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>YALDNLEPGSEANVLSR</td>
<td>Q06458</td>
<td>YALDNLEPGSEANVLSR</td>
<td>GLL-GDA-GEPIVISPLKQRI-LRDG-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Legend:**
- **Core**
- **Insertion**
- **Deletion**
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)

Core of alignment
HMM construction (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
TCAACTATC
ACAC--AGC
AGA---ATC
ACCG--ATC

ACA---ATG 0.8x1x0.8x1x0.8x0.4x1x1x0.8x1x0.2 = 3.3x10^{-2}
Scoring a sequence to an HMM

\[
\begin{align*}
\text{ACA---ATG} & \quad 0.8x1x0.8x1x0.8x0.4x1x0.8x1x0.2 = 3.3 \times 10^{-2} \\
\text{TCAACTATC} & \quad 0.2x1x0.8x1x0.8x0.6x0.2x0.4x0.4x0.4x0.2x0.6x1x1x0.8x1x0.8 = 0.0075 \times 10^{-2} \\
\text{ACAC--AGC} & = 1.2 \times 10^{-2}
\end{align*}
\]

Consensus:
\[
\begin{align*}
\text{ACAC--ATC} & = 4.7 \times 10^{-2}, \text{ ACA---ATC} = 13.1 \times 10^{-2}
\end{align*}
\]

Exceptional:
\[
\begin{align*}
\text{TGCT--AGG} & = 0.0023 \times 10^{-2}
\end{align*}
\]
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 \( S \)
  \[ \log( \frac{P(S)}{0.25^L} ) \]
- Positive score means more likely than Null model

  - **This is just like we did for PSSM log(p/q)!**

  \[
  \begin{align*}
  \text{ACA}--\text{ATG} &= 4.9 \\
  \text{TCAACTATC} &= 3.0 \\
  \text{ACAC}--\text{AGC} &= 5.3 \\
  \text{AGA}--\text{ATC} &= 4.9 \\
  \text{ACCG}--\text{ATC} &= 4.6 \\
  \text{Consensus:} \\
  \text{ACAC}--\text{ATC} &= 6.7 \\
  \text{ACA}--\text{ATC} &= 6.3 \\
  \text{Exceptional:} \\
  \text{TGCT}--\text{AGG} &= -0.97
  \end{align*}
  \]

  \( \checkmark \) 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

The unfair casino:

- Loaded dice: $p(6) = 0.5$;
  - Switch fair to load: $0.05$
  - Switch load to fair: $0.1$

Fair

<table>
<thead>
<tr>
<th>Number</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1/6$</td>
</tr>
<tr>
<td>2</td>
<td>$1/6$</td>
</tr>
<tr>
<td>3</td>
<td>$1/6$</td>
</tr>
<tr>
<td>4</td>
<td>$1/6$</td>
</tr>
<tr>
<td>5</td>
<td>$1/6$</td>
</tr>
<tr>
<td>6</td>
<td>$1/6$</td>
</tr>
</tbody>
</table>

Loaded

<table>
<thead>
<tr>
<th>Number</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1/10$</td>
</tr>
<tr>
<td>2</td>
<td>$1/10$</td>
</tr>
<tr>
<td>3</td>
<td>$1/10$</td>
</tr>
<tr>
<td>4</td>
<td>$1/10$</td>
</tr>
<tr>
<td>5</td>
<td>$1/10$</td>
</tr>
<tr>
<td>6</td>
<td>$1/2$</td>
</tr>
</tbody>
</table>
Model decoding (Viterbi)

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

\[
\begin{align*}
\text{FFF} &= 0.5 \times 0.167 \times 0.95 \times 0.167 \times 0.95 \times 0.167 = 0.0021 \\
\text{FFL} &= 0.5 \times 0.167 \times 0.95 \times 0.167 \times 0.05 \times 0.5 = 0.00333 \\
\text{FLF} &= 0.5 \times 0.167 \times 0.05 \times 0.5 \times 0.1 \times 0.167 = 0.000035 \\
\text{FLL} &= 0.5 \times 0.167 \times 0.05 \times 0.5 \times 0.9 \times 0.5 = 0.00094 \\
\text{LFF} &= 0.5 \times 0.1 \times 0.1 \times 0.167 \times 0.95 \times 0.167 = 0.00013 \\
\text{LFL} &= 0.5 \times 0.1 \times 0.1 \times 0.167 \times 0.05 \times 0.5 = 0.000021 \\
\text{LLF} &= 0.5 \times 0.1 \times 0.9 \times 0.5 \times 0.1 \times 0.167 = 0.00038 \\
\text{LLL} &= 0.5 \times 0.1 \times 0.9 \times 0.5 \times 0.9 \times 0.5 = 0.0101
\end{align*}
\]
Or in log space

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

\[
\begin{align*}
\text{Log}(P(LLL|M)) &= \log(0.5 \times 0.1 \times 0.9 \times 0.5 \times 0.9 \times 0.5) = \log(0.0101) \\
\text{or}
\text{Log}(P(LLL|M)) &= \log(0.5) + \log(0.1) + \log(0.9) + \log(0.5) + \log(0.9) + \log(0.5) \\
&= -0.3 -1 -0.046 -0.3 -0.046 -0.3 = -1.99
\end{align*}
\]
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

Log model

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>6</td>
<td>-0.3</td>
<td>-0.3</td>
</tr>
</tbody>
</table>

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
Model decoding (Viterbi)

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

<table>
<thead>
<tr>
<th>F</th>
<th>5</th>
<th></th>
<th>6</th>
<th></th>
<th>6</th>
<th>6</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>-1.08</td>
<td></td>
<td></td>
<td>-1.30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Log model

- $\log(FF) = -0.30 -0.78 - 0.02 -0.78 = -1.88$
- $\log(LF) = -0.30 -1 -1 -0.78 = -3.08$
- $\log(FL) = -0.30 -0.78 - 1.30 -0.30 = -2.68$
- $\log(LL) = -0.30 -1 -0.046 -0.3 = -1.65$
Model decoding (Viterbi)

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

<table>
<thead>
<tr>
<th>Dice</th>
<th>Fair</th>
<th>Loaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>-0.78</td>
<td>-1</td>
</tr>
<tr>
<td>2:</td>
<td>-0.78</td>
<td>-1</td>
</tr>
<tr>
<td>3:</td>
<td>-0.78</td>
<td>-1</td>
</tr>
<tr>
<td>4:</td>
<td>-0.78</td>
<td>-1</td>
</tr>
<tr>
<td>5:</td>
<td>-0.78</td>
<td>-1</td>
</tr>
<tr>
<td>6:</td>
<td>-0.78</td>
<td>-0.3</td>
</tr>
</tbody>
</table>

Log model:
- \( \text{Log}(FF) = -0.30 - 0.78 - 0.02 - 0.78 = -1.88 \)
- \( \text{Log}(LF) = -0.30 - 1 - 1 - 0.78 = -3.08 \)
- \( \text{Log}(FL) = -0.30 - 0.78 - 1.30 - 0.30 = -2.68 \)
- \( \text{Log}(LL) = -0.30 - 1 - 0.046 - 0.3 = -1.65 \)

\( \text{FF} = 0.5*0.167*0.95*0.167 \)
\( \text{LF} = 0.5*0.1*0.1*0.167 \)
\( \text{FL} = 0.5*0.167*0.05*0.5 \)
\( \text{LL} = 0.5*0.1*0.9*0.5 \)
Model decoding (Viterbi)

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

$$\begin{align*}
FFF &= 0.5 \times 0.167 \times 0.95 \times 0.167 \times 0.95 \times 0.167 = 0.0021 \\
FLF &= 0.5 \times 0.167 \times 0.05 \times 0.5 \times 0.1 \times 0.167 = 0.000035 \\
LFF &= 0.5 \times 0.1 \times 0.1 \times 0.167 \times 0.95 \times 0.167 = 0.00013 \\
LFL &= 0.5 \times 0.1 \times 0.9 \times 0.5 \times 0.1 \times 0.167 = 0.00038 \\
FLL &= 0.5 \times 0.167 \times 0.05 \times 0.5 \times 0.9 \times 0.5 = 0.00094 \\
FFL &= 0.5 \times 0.167 \times 0.95 \times 0.167 \times 0.05 \times 0.5 = 0.00333 \\
LFL &= 0.5 \times 0.1 \times 0.1 \times 0.167 \times 0.05 \times 0.5 = 0.000021 \\
LLL &= 0.5 \times 0.1 \times 0.9 \times 0.5 \times 0.9 \times 0.5 = 0.0101
\end{align*}$$

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>-1.08</td>
<td>-1.88</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1.30</td>
<td>-1.65</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Model decoding (Viterbi)

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

\[
\begin{align*}
\text{FFF} &= 0.5 \times 0.167 \times 0.95 \times 0.167 \times 0.95 \times 0.167 = 0.0021 \\
\log(P(\text{FFF})) &= -2.68 \\
\text{LLL} &= 0.5 \times 0.1 \times 0.9 \times 0.5 \times 0.9 \times 0.5 = 0.0101 \\
\log(P(\text{LLL})) &= -1.99
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>-1.08</td>
<td>-1.88</td>
<td>-2.68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>-1.30</td>
<td>-1.65</td>
<td>-1.99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Model decoding (Viterbi)

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

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>-1.08</td>
<td>-1.88</td>
<td>-2.68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>-1.30</td>
<td>-1.65</td>
<td>-1.99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Log model:
- Fair: 1:-1, 2:-1, 3:-1, 4:-1, 5:-1, 6:-0.3
- Loaded: 1:-1

-1.08 -1.88 -2.68 -1.3 -1.65 -1.99
Model decoding (Viterbi)

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

\[ -0.78 - 0.02 - 2.68 = -3.48 \]

\[ -0.78 - 1 - 1.99 = -3.77 \]

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>-1.08</td>
<td>-1.88</td>
<td>-2.68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>-1.30</td>
<td>-1.65</td>
<td>-1.99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fair

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Loaded

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ F -0.02 \]

\[ L -0.046 \]
Model decoding (Viterbi)

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

\[-0.78 - 0.02 - 2.68 = -3.48\]
\[-0.78 - 1 - 1.99 = -3.77\]
Model decoding (Viterbi)

- Now we can formalize the algorithm!

\[ P_i(i+1) = p_i(i+1) \cdot \max_k (P_k(i) \cdot a_{kl}) \quad \text{or} \quad \log(P_i(i+1)) = \log(p_i(i+1)) + \max_k (\log(P_k(i)) + \log(a_{kl})) \]
Model decoding (Viterbi)

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

**Initialization**

- **Log model**
  - $F = 0.5 \times 0.167$
  - $\log(F) = \log(0.5) + \log(0.167) = -1.08$
  - $L = 0.5 \times 0.1$
  - $\log(L) = \log(0.5) + \log(0.1) = -1.30$

### Table

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>-1.08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>-1.30</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Fair**
  - 1:
  - 2:
  - 3:
  - 4:
  - 5:
  - 6:

- **Loaded**
  - 1:
  - 2:
  - 3:
  - 4:
  - 5:
  - 6: -0.3

- **Values**
  - 1: -0.78
  - 2: -0.78
  - 3: -0.78
  - 4: -0.78
  - 5: -0.78
  - 6: -0.78

- **Log model**
  - 1: -1.3
  - 2: -1
  - 3: -1
  - 4: -1
  - 5: -1
  - 6: -0.3

- **Initialization**
  - **F**
  - **L**
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

\[
P_i(i+1) = p_i(i+1) \cdot \max_k(P_k(i) \cdot a_{kl}) \quad \text{or} \quad \log(P_i(i+1) = \log(p_i(i+1)) + \max_k(\log(P_k(i)) + \log(a_{kl}))
\]

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-1.08</td>
<td>-1.30</td>
</tr>
<tr>
<td>6</td>
<td>-1.88</td>
<td>-1.65</td>
</tr>
<tr>
<td>6</td>
<td>-2.68</td>
<td>-1.99</td>
</tr>
<tr>
<td>6</td>
<td>-3.48</td>
<td></td>
</tr>
</tbody>
</table>
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

\[ P_i(i+1) = p_i(i+1) \cdot \max_k (P_k(i) \cdot a_{kl}) \text{ or} \]
\[ \log(P_i(i+1)) = \log(p_i(i+1)) + \max_k (\log(P_k(i) + \log(a_{kl}))) \]

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>-1.08</td>
<td>-1.88</td>
<td>-2.68</td>
<td>-3.48</td>
<td>-4.92</td>
<td>-6.52</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>-1.30</td>
<td>-1.65</td>
<td>-1.99</td>
<td>-3.39</td>
<td>-6.53</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Log model

-0.02
-0.046

Fair

Loaded

1: -0.78
2: -0.78
3: -0.78
4: -0.78
5: -0.78
6: -0.78

1: -1
2: -1
3: -1
4: -1
5: -1
6: -0.3

-1
1.3
Model decoding (Viterbi). Can you do it?
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

\[ P_l(i+1) = p_l(i+1) \cdot \max_k P_k(i) \cdot a_{kl} \quad \text{or} \]
\[ \log(P_l(i+1)) = \log(p_l(i+1)) + \max_k (\log(P_k(i)) + \log(a_{kl})) \]

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>-1.08</td>
<td>-1.88</td>
<td>-2.68</td>
<td>-3.48</td>
<td>-4.12</td>
<td>-4.92</td>
<td>-5.72</td>
<td>-6.52</td>
<td>-7.33</td>
</tr>
<tr>
<td>L</td>
<td>-1.30</td>
<td>-1.65</td>
<td>-1.99</td>
<td>-2.34</td>
<td>-3.39</td>
<td>-4.44</td>
<td>-5.49</td>
<td>-6.53</td>
<td>-7.57</td>
</tr>
</tbody>
</table>
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
  - LLLLFFFFFF

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>5.108</td>
<td>-1.88</td>
<td>-2.68</td>
<td>-3.48</td>
<td>-4.12</td>
<td>-4.92</td>
<td>-5.72</td>
<td>-6.52</td>
<td>-7.33</td>
</tr>
<tr>
<td>L</td>
<td>-1.30</td>
<td>-1.65</td>
<td>-1.99</td>
<td>-2.34</td>
<td>-3.39</td>
<td>-4.44</td>
<td>-5.49</td>
<td>-6.53</td>
<td>-7.57</td>
</tr>
</tbody>
</table>
Model decoding (Viterbi).

- What happens if you have three dice?

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>-1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L1</td>
<td>-1.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>-1.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
P_i(i+1) = p_i(i+1) \cdot \max_k (P_k(i) \cdot a_{kl}) \quad \text{or} \quad \log(P_i(i+1)) = \log(p_i(i+1)) + \max_k (\log(P_k(i)) + \log(a_{kl}))
\]
And if you have a trans-membrane model

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

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>G</th>
<th>V</th>
<th>L</th>
<th>I</th>
<th>M</th>
<th>A</th>
<th>D</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>iC</td>
<td>-1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>-1.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xC</td>
<td>-1.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
P_i(i+1) = p_i(i+1) \cdot \max_k (P_k(i) \cdot a_{kl}) \]  
or
\[
\log(P_i(i+1) = \log(p_i(i+1)) + \max_k (\log(P_k(i) + \log(a_{kl})))
\]
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

\[ P(x) = \sum P(x, \pi) \]

• 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

\[ P(x) = \sum_k f_k(L) \]
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**

\[ f_F(5) = 0.167 \cdot 0.5 = 0.083 \]

\[ f_L(5) = 0.1 \cdot 0.5 = 0.05 \]
Forward algorithm

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

\[ f_k(0) = 1 \]

\[ a_{0l} = \pi_l \]

\[ f_F(5) = 0.167 \cdot 0.5 = 0.083 \]

\[ f_L(5) = 0.1 \cdot 0.5 = 0.05 \]
Forward algorithm

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

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>8.3e-2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>L</td>
<td>5e-2</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
</tbody>
</table>
Forward algorithm

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

\[ 0.167 \cdot (0.083 \cdot 0.95 + 0.05 \cdot 0.1) = 0.014 \]
### Forward algorithm

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

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

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>0.083</td>
<td>0.014</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>0.05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 \( P(x) \)?

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>8.30e-2</td>
<td>2.63e-3</td>
<td>6.08e-4</td>
<td>1.82e-4</td>
<td>3.66e-5</td>
<td>1.09e-6</td>
<td>1.79e-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>5.00e-2</td>
<td>2.46e-2</td>
<td>1.14e-2</td>
<td>4.71e-4</td>
<td>4.33e-5</td>
<td>4.00e-7</td>
<td>4.14e-8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Forward algorithm.
Can you do it yourself?
Forward algorithm

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

\[ P(x) = \sum_k f_k(L) \]

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

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

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!

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Backward algorithm

\[ P(x, \pi_i = k) = P(x_1, x_2, ..., x_i, \pi_i = k) \cdot P(x_{i+1}, ..., x_L | \pi_i = k) \]
\[ P(x, \pi_i = k) = f_k(i) \cdot b_k(i) \]

\[
\begin{align*}
  f_k(i) &= P(x_1, x_2, ..., x_i, \pi_i = k) \\
  f_k(i) &= p_k(x_i) \cdot \sum_l f_l(i-1) \cdot a_{lk}
\end{align*}
\]

\[
\begin{align*}
  b_k(i) &= P(x_{i+1}, x_{i+2}, ..., x_L | \pi_i = k) \\
  b_k(i) &= \sum_l a_{kl} \cdot p_l(x_{i+1}) \cdot b_l(i+1)
\end{align*}
\]

\[
\begin{align*}
  P(\pi_i = k | x) &= \frac{P(x, \pi_i = k)}{P(x)} = \frac{f_k(i) \cdot b_k(i)}{P(x)}
\end{align*}
\]
Backward algorithm

\[ b_k(i) = \sum_{l} a_{kl} \cdot p_l(x_{i+1}) \cdot b_l(i+1) \]

\[ 0.167 \cdot 0.95 \cdot 1 + 0.1 \cdot 0.05 \cdot 1 = 0.163 \]

\[ 0.167 \cdot 0.1 \cdot 1 + 0.1 \cdot 0.9 \cdot 1 = 0.107 \]
Backward algorithm

\[ b_k(i) = \sum_{l} a_{kl} \cdot p_l(x_{i+1}) \cdot b_l(i+1) \]
Backward algorithm

• Note that the sum of first column of the backward matrix 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 $X$?
What is the posterior probability that observation $x_i$ came from state $k$ given the observed sequence $X$.

\[
P(\pi_i = k \mid x) = \frac{P(x, \pi_i = k)}{P(x)} = \frac{f_k(i) \cdot b_k(i)}{P(x)}
\]
The probability is context dependent.

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

\[ \text{ACA}---\text{ATG} \quad 0.8 \times 1 \times 0.8 \times 1 \times 0.8 \times 0.4 \times 1 \times 1 \times 0.8 \times 1 \times 0.2 = 3.3 \times 10^{-2} \]
Un-supervised learning

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

\[
\gamma_k^i = \sum_l \varepsilon_{kl}^i
\]
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

\[ \sum_{i=1}^{T-1} \epsilon_{kl}^i \]

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

Now

\[ a_{kl} = \frac{\sum_{i=1}^{T-1} \varepsilon_{kl}^i}{\sum_{i=1}^{T-1} \gamma_k^i} \]

 Estimate probability of transition between state k and l

and

\[ e_{k}^{a} = \frac{\sum_{i=1, O_i = v_a}^{T} \gamma_k^i}{\sum_{i=1}^{T} \gamma_k^i} \]

 Estimate probability emitting symbol a in state k
Use these relations

\[
\begin{align*}
a_{kl} &= \frac{\sum_{i=1}^{T-1} \varepsilon_{ki}^i}{\sum_{i=1}^{T-1} \gamma_k^i} \\
e^a_k &= \frac{\sum_{i=1, O_i = v_a}^{N-1} \gamma_k^i}{\sum_{i=1}^{N-1} \gamma_k^i}
\end{align*}
\]

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

<table>
<thead>
<tr>
<th>Conserved</th>
<th>deletion</th>
<th>Non-conserved</th>
<th>Insertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDGSLAFVPSEF--SISPGEKIVFKNNAGFPHNIVFDEDSIPSGVDASKISMSSEEEDLLN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TVNAGAI--PGPLIAERLKGEQONVRVTNTLDEDTSIHWHGLLVPGMDGVPGVSFPG---I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-TSMAPAFGVQEFYRTVKQGDEVTVTIT----NIDQIED-VSHGFVVN HGVSME---I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IE--KMKYLTPVFYTITKAGETVYWNVEVMPHNVAFKKGIV--GEDAFRGEVMTKD---</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-TSVAPSFSQPSF-LTVKGEDEVTVTVTNLDE----IDDLTHGFTMNGHVAM---V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASAETMVFEPDFLVLEIGFGDRVFVPTHK-SHNAATIDGMVPEGVEGFKSRINDE----</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TVNGQ--FPGPLLAGVAREGDQVLKVVNHVAENITIHWHVQLTGWADGPAYVTQCPIT</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

$P_1, P_2, P_3, P_4, P_5, P_6, P_7$

$D_2, D_3, D_4, D_5, D_6$

$L_1 - Y_2A_3V_4R_5 - I_6$

$P_1D_2P_3P_4I_4P_5D_6P_7$
Profile HMM

- Un-gapped profile HMM is just a sequence profile
Profile HMM

• Un-gapped profile HMM is just a sequence profile

\[
\begin{align*}
P_1 & \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_5 \rightarrow P_6 \rightarrow P_7 \\
\alpha_{lk} &= 1.0
\end{align*}
\]

- A: 0.05
- C: 0.01
- D: 0.08
- E: 0.08
- F: 0.03
- G: 0.02
- ..
- V: 0.08
- Y: 0.01
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)

<table>
<thead>
<tr>
<th>QUERY</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HAMDIRCYHSGG</td>
<td>PLHL</td>
<td>GEI</td>
<td>EDFNGQSCIVCPWHKYKITLATGE</td>
<td>GLYQSINPKDPS</td>
<td></td>
</tr>
<tr>
<td>HAMDIRCYHSGG</td>
<td>PLHL</td>
<td>GEI</td>
<td>EDFNGQSCIVCPWHKYKITLATGE</td>
<td>GLYQSINPKDPS</td>
<td></td>
</tr>
<tr>
<td>HAMDIRCYHSGG</td>
<td>PLHL</td>
<td>GI</td>
<td>EDFGRPCIVCPWHKYKITLATGE</td>
<td>GLYQSINPKDPS</td>
<td></td>
</tr>
<tr>
<td>HAMDIRCYHSGG</td>
<td>PLHL</td>
<td>GEI</td>
<td>EDFNGQSCIVCPWHKYKITLATGE</td>
<td>GLYQSINPKDPS</td>
<td></td>
</tr>
<tr>
<td>Q8K2P6</td>
<td>ALSE</td>
<td>GYI</td>
<td>DGD</td>
<td>VECTLHFGKFCVRTGK</td>
<td>VKAL</td>
</tr>
<tr>
<td>Q8K2P6</td>
<td>ALSE</td>
<td>GYI</td>
<td>DGD</td>
<td>VECTLHFGKFCVRTGK</td>
<td>VKAL</td>
</tr>
<tr>
<td>Q8K2P6</td>
<td>ALSE</td>
<td>GYI</td>
<td>DGD</td>
<td>VECTLHFGKFCVRTGK</td>
<td>VKAL</td>
</tr>
<tr>
<td>Q8K2P6</td>
<td>ALSE</td>
<td>GYI</td>
<td>DGD</td>
<td>VECTLHFGKFCVRTGK</td>
<td>VKAL</td>
</tr>
</tbody>
</table>

**Core** | **Insertion** | **Deletion**
The HMMer program

• HMMer is an 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
SDPEISEQDEEKKKYTSVCVGREEDIRKSERMTAVVHDREVVFYHKGEYHAMDIRCYS
GGPLHLGEIEDFNGQSCIVCPWHKYKITLATGEGLYQSINPKDPSAKPKWCSKGVQRIH
TVKVDNGNIYVTLSEPFKCDSDYYATGEFKVIQSSS
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
  blastpbgp -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 DPEISEQDEEKKKYTECVCGREEDIRKS-ERMTAVHD-RE--V-V-IF--Y-H-KGE-Y
  Q8K2P6  DPEISEQDEEKKKYTECVCGREEDIRKS-ERMTAVHD-RE--V-V-IF--Y-H-KGE-Y
  Q8TAC1  ----SAQDPEKREYSSVCGREDDIKKS-ERMTAVHD-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

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Description</th>
<th>Score</th>
<th>E-value</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>2E4Q.A</td>
<td>mol:aa ELECTRON TRANSPORT</td>
<td>163.7</td>
<td>6.7e-45</td>
<td>1</td>
</tr>
<tr>
<td>2E4P.B</td>
<td>mol:aa ELECTRON TRANSPORT</td>
<td>163.7</td>
<td>6.7e-45</td>
<td>1</td>
</tr>
<tr>
<td>2E4P.A</td>
<td>mol:aa ELECTRON TRANSPORT</td>
<td>163.7</td>
<td>6.7e-45</td>
<td>1</td>
</tr>
<tr>
<td>2E4Q.C</td>
<td>mol:aa ELECTRON TRANSPORT</td>
<td>163.7</td>
<td>6.7e-45</td>
<td>1</td>
</tr>
<tr>
<td>2YVJ.B</td>
<td>mol:aa OXIDOREDUCTASE/ELECTRON TRANSPORT</td>
<td>163.7</td>
<td>6.7e-45</td>
<td>1</td>
</tr>
<tr>
<td>1FQT.A</td>
<td>mol:aa OXIDOREDUCTASE</td>
<td>160.9</td>
<td>4.5e-44</td>
<td>1</td>
</tr>
<tr>
<td>1FQT.B</td>
<td>mol:aa OXIDOREDUCTASE</td>
<td>160.9</td>
<td>4.5e-44</td>
<td>1</td>
</tr>
<tr>
<td>2QPZ.A</td>
<td>mol:aa METAL BINDING PROTEIN</td>
<td>137.3</td>
<td>5.6e-37</td>
<td>1</td>
</tr>
<tr>
<td>2Q3W.A</td>
<td>mol:aa ELECTRON TRANSPORT</td>
<td>116.2</td>
<td>1.3e-30</td>
<td>1</td>
</tr>
<tr>
<td>1VM9.A</td>
<td>mol:aa ELECTRON TRANSPORT</td>
<td>116.2</td>
<td>1.3e-30</td>
<td>1</td>
</tr>
</tbody>
</table>
Validation. CE structural alignment

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


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 = 2.03Å  Z-Score = 5.5  Gaps = 20 (19.8%)
CPU = 1s  Sequence identities = 18.1%

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

Chain 1: 68 GKFCVRTGKVKS------PPPC--------EPLKVYPIRIEGRDVLVDFSRAALH
Chain 2: 85 YKITLATGEGLYQSINPKDPSAKPKWCSGVKQRIHTVKVDNGNIYVTL-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/)

- **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