Markov Models and Hidden Markov Models
Dosage Compensation and X-Inactivation

2 copies (mom/dad) of each chromosome 1-23
Mostly, both copies of each gene are expressed
  E.g., A B O blood group defined by 2 alleles of 1 gene
Women (XX) get double dose of X genes (vs XY)?
So, early in embryogenesis:
  • One X randomly inactivated in each cell
    • Choice maintained in daughter cells
Calico: a major coat color gene is on X

How?
Reminder: Proteins “Read” DNA

E.g.:
MyoD

http://www.rcsb.org/pdb/explore/jmol.do?structureId=1MDY&bionumber=1
Down in the Groove

Different patterns of hydrophobic methyls, potential H bonds, etc. at edges of different base pairs. They’re accessible, esp. in major groove.

Figure 7-7 Molecular Biology of the Cell 5/e (© Garland Science 2008)
DNA Methylation

CpG - 2 adjacent nts, same strand
(not Watson-Crick pair; “p” mnemonic for the phosphodiester bond of the DNA backbone)

C of CpG is often (70-80%) methylated in mammals i.e., CH$_3$ group added (both strands)
Same Pairing

Methyl-C alters major groove profile (TF binding), but not base-pairing, transcription or replication

Figure 7-7 Molecular Biology of the Cell 5/e (© Garland Science 2008)
DNA Methylation—Why

In vertebrates, it generally silences transcription
  (Epigenetics) X-inactivation, imprinting, repression of mobile elements, cancers, aging, and developmental differentiation

E.g., if a stem cell divides, one daughter fated to be liver, other kidney, need to
  (a) Turn off liver genes in kidney & vice versa,
  (b) Remember that through subsequent divisions

How? One way:
  (a) Methylate genes, esp. promoters, to silence them
  (b) After ÷, DNA methyltransferases convert hemi- to fully-methylated
      (not trivial: deleting methyltransferase is embryonic-lethal in mice)

Major exception: promoters of “housekeeping” genes
“CpG Islands”

Methyl-C mutates to T relatively easily

Net: CpG is less common than expected genome-wide:
\[ f(\text{CpG}) < f(\text{C}) \times f(\text{G}) \]

BUT in some regions (e.g. active promoters), CpGs remain unmethylated, so CpG → TpG less likely there: makes “CpG Islands”; often mark gene-rich regions
CpG Islands

More CpG than elsewhere (say, CpG/GpC>50%)
More C & G than elsewhere, too (say, C+G>50%)
Typical length: few 100 to few 1000 bp

Questions
Is a short sequence (say, 200 bp) a CpG island or not?
Given long sequence (say, 10-100kb), find CpG islands?
Markov & Hidden Markov Models

References (see also online reading page):
Independence

A key issue: Previous models we’ve talked about assume independence of nucleotides in different positions - definitely unrealistic.

Markov models allow us to relax that assumption.
Markov Chains

A sequence $x_1, x_2, \ldots$ of random variables is a $k$-th order Markov chain if, for all $i$, $i^{th}$ value is independent of all but the previous $k$ values:

$$P(x_i \mid x_1, x_2, \ldots, x_{i-1}) = P(x_i \mid x_{i-k}, x_{i-k+1}, \ldots, x_{i-1})$$

Example 1: Uniform random ACGT
Example 2: Weight matrix model
Example 3: ACGT, but ↓ Pr(G following C)
A Markov Model (1st order)

States: A,C,G,T
Emissions: corresponding letter
Transitions: $a_{st} = P(x_i = t \mid x_{i-1} = s)$
A Markov Model (1st order)

States: A, C, G, T
Emissions: corresponding letter
Transitions: \( a_{st} = P(x_i = t \mid x_{i-1} = s) \)
Begin/End states
Pr of emitting sequence $x$

$$x = x_1 \ x_2 \ldots \ x_n$$

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

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

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

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

$$= \prod_{i=0}^{n-1} a_{x_i, x_{i+1}} \quad \text{(with Begin state)}$$
Training

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

E.g., from 48 CpG islands in 60k bp:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0.180</td>
<td>0.274</td>
<td>0.426</td>
<td>0.120</td>
<td>A</td>
<td>0.300</td>
<td>0.205</td>
<td>0.285</td>
<td>0.210</td>
</tr>
<tr>
<td>C</td>
<td>0.171</td>
<td>0.368</td>
<td><strong>0.274</strong></td>
<td>0.188</td>
<td>C</td>
<td>0.322</td>
<td>0.298</td>
<td><strong>0.078</strong></td>
<td>0.302</td>
</tr>
<tr>
<td>G</td>
<td>0.161</td>
<td>0.339</td>
<td>0.375</td>
<td>0.125</td>
<td>G</td>
<td>0.248</td>
<td>0.246</td>
<td>0.298</td>
<td>0.208</td>
</tr>
<tr>
<td>T</td>
<td>0.079</td>
<td>0.355</td>
<td>0.384</td>
<td>0.182</td>
<td>T</td>
<td>0.177</td>
<td>0.239</td>
<td>0.292</td>
<td><strong>0.292</strong></td>
</tr>
</tbody>
</table>

From DEKM 18
Discrimination/Classification

Log likelihood ratio of CpG model vs background model

\[ S(x) = \log \frac{P(x|+\text{model})}{P(x|-\text{model})} = \sum_{i=1}^{L} \log \frac{a_{x_{i-1},x_i}^+}{a_{x_{i-1},x_i}^-} = \sum_{i=1}^{L} \log \beta_{x_{i-1},x_i} \]

<table>
<thead>
<tr>
<th>β</th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-0.740</td>
<td>0.419</td>
<td>0.580</td>
<td>-0.803</td>
</tr>
<tr>
<td>C</td>
<td>-0.913</td>
<td>0.302</td>
<td>1.812</td>
<td>-0.685</td>
</tr>
<tr>
<td>G</td>
<td>-0.624</td>
<td>0.461</td>
<td>0.331</td>
<td>-0.730</td>
</tr>
<tr>
<td>T</td>
<td>-1.169</td>
<td>0.573</td>
<td>0.393</td>
<td>-0.679</td>
</tr>
</tbody>
</table>

From DEKM 19
CpG Island Scores

Figure 3.2  Histogram of length-normalized scores.
Questions

Q1: Given a short sequence, is it more likely from feature model or background model? Above

Q2: Given a long sequence, where are the features in it (if any)

   ---

   **Approach 1:** score 100 bp (e.g.) windows
   Pro: simple
   Con: arbitrary, fixed length, inflexible

   **Approach 2:** combine +/- models.
Combined Model

Emphasis is “Which (hidden) state?” not “Which model?”
Hidden Markov Models
(HMMs; Claude Shannon, 1948)

States: 1, 2, 3, ...
Paths: sequences of states $\pi = (\pi_1, \pi_2, \ldots)$
Transitions: $a_{k,l} = P(\pi_i = l \mid \pi_{i-1} = k)$
Emissions: $e_k(b) = P(x_i = b \mid \pi_{i} = k)$

Observed data: emission sequence
Hidden data: state/transition sequence
The Occasionally Dishonest Casino

1 fair die, 1 “loaded” die, occasionally swapped
<table>
<thead>
<tr>
<th>Rolls</th>
<th>31511624644644245311321631164152133625144543631656626566666</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die</td>
<td>FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLL</td>
</tr>
<tr>
<td>Viterbi</td>
<td>FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLL</td>
</tr>
<tr>
<td>Rolls</td>
<td>65116645313265124563666463163666316232645236266666625151631</td>
</tr>
<tr>
<td>Die</td>
<td>LLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFF</td>
</tr>
<tr>
<td>Viterbi</td>
<td>LLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFFFFFF</td>
</tr>
<tr>
<td>Rolls</td>
<td>222555441666566563564324364131513465146353411126414626253356</td>
</tr>
<tr>
<td>Die</td>
<td>FFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFFFL</td>
</tr>
<tr>
<td>Viterbi</td>
<td>FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF</td>
</tr>
<tr>
<td>Rolls</td>
<td>36616366646623253441366166116325256246225265252266435353336</td>
</tr>
<tr>
<td>Die</td>
<td>LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFF</td>
</tr>
<tr>
<td>Viterbi</td>
<td>LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFF</td>
</tr>
<tr>
<td>Rolls</td>
<td>2331216253644144323351632436336656246666626326666612355245242</td>
</tr>
<tr>
<td>Die</td>
<td>FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF</td>
</tr>
<tr>
<td>Viterbi</td>
<td>FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF</td>
</tr>
</tbody>
</table>

**Figure 3.5**

Rolls: Visible data—300 rolls of a die as described above.

Die: Hidden data—which die was actually used for that roll (F = fair, L = loaded).

Viterbi: the prediction by the Viterbi algorithm is shown.
Inferring hidden stuff

Joint probability of a given path $\pi$ & emission sequence $x$:

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

But $\pi$ is hidden; what to do? Some alternatives:

Most probable single path

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

Sequence of most probable states

$$\hat{\pi}_i = \arg\max_{k} P(\pi_i = k \mid x)$$

Etc.
The Viterbi Algorithm: The most probable path

Viterbi finds: $$\pi^* = \arg \max_{\pi} P(x, \pi)$$

Possibly there are $$10^{99}$$ paths of prob $$10^{-99}$$
  (If so, non-Viterbi approaches may be preferable.)

More commonly, one path (+ slight variants) dominate others; Viterbi finds that

Key problem: exponentially many paths $$\pi$$
Unrolling an HMM

Conceptually, sometimes convenient
Note exponentially many paths
Viterbi

\[ \nu_l(i) = \text{probability of the most probable path emitting } x_1, x_2, \ldots, x_i \text{ and ending in state } l \]

Initialize:

\[
\nu_l(0) = \begin{cases} 
1 & \text{if } l = \text{Begin state} \\
0 & \text{otherwise}
\end{cases}
\]

General case:

\[
\nu_l(i + 1) = e_l(x_{i+1}) \cdot \max_k (\nu_k(i) a_{k,l})
\]
HMM Casino Example

(Excel spreadsheet on web; download & play...)
HMM Casino Example

(Excel spreadsheet on web; download & play…)
Viterbi Traceback

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

\[ v_l(i + 1) = e_l(x_{i+1}) \cdot \max_k (v_k(i) a_{k,l}) \]
Figure 3.5

Rolls: Visible data—300 rolls of a die as described above.
Die: Hidden data—which die was actually used for that roll (F = fair, L = loaded).
Viterbi: the prediction by the Viterbi algorithm is shown.

From DEKM 33
Most probable path != sequence of most probable states

Another example, based on casino dice again:
Suppose $p(\text{fair} \leftrightarrow \text{loaded})$ transitions are $10^{-99}$ and roll sequence is $1111166\ldots666$; then fair state is more likely all through 1’s & well into the run of 6’s, but eventually loaded wins, and the improbable $F \rightarrow L$ transitions make Viterbi = all $L$.

* = max prob
= Viterbi
Is Viterbi “best”? 

Viterbi finds \( \pi^* = \arg \max_\pi P(x, \pi) \)

Most probable (Viterbi) path goes through 5, but most probable state at 2nd step is 6 (i.e., Viterbi is not the only interesting answer.)
An HMM (unrolled)

States

Emissions/sequence positions

$X_1$  $X_2$  $X_3$  $X_4$
Viterbi: best path to each state

Viterbi score:

\[ v_l(i + 1) = e_l(x_{i+1}) \cdot \max_k (v_k(i) a_{k,l}) \]

Viterbi path\(^R\):

\[ \text{back}_l(i + 1) = \arg\max_k (v_k(i) a_{k,l}) \]
For each state/time, want \textit{total} probability of all paths leading to it, with given emissions.

\[
\begin{align*}
    f_k(i) & \triangleq P(x_1 \ldots x_i, \pi_i = k) \\
    f_l(i + 1) & = e_l(x_{i+1}) \sum_k f_k(i) a_{k,l} \\
    P(x) & = \sum_{\pi} P(x, \pi) = \sum_k f_k(n) a_{k,\text{end}}
\end{align*}
\]
The Backward Algorithm

Similar: for each state/time, want total probability of all paths from it, with given emissions, conditional on that state.

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

\[ b_k(i) = \sum_l a_{k,l} e_l(x_{i+1}) b_l(i + 1) \]

\[ b_k(n) = a_{k,\text{end}} \]
In state $k$ at step $i$?

\[
P(x, \pi_i = k) = P(x_1, \ldots, x_i, \pi_i = k) \cdot P(x_{i+1}, \ldots, x_n \mid x_1, \ldots, x_i, \pi_i = k)
\]
\[
= P(x_1, \ldots, x_i, \pi_i = k) \cdot P(x_{i+1}, \ldots, x_n \mid \pi_i = k)
\]
\[
= f_k(i) \cdot b_k(i)
\]

\[
P(\pi_i = k \mid x) = \frac{P(x, \pi_i = k)}{P(x)} = \frac{f_k(i) \cdot b_k(i)}{P(x)}
\]
Posterior Decoding, I

Alternative 1: what’s the most likely state at step $i$?

$$\hat{\pi}_i = \arg \max_k P(\pi_i = k \mid x)$$

Note: the sequence of most likely states $\neq$ the most likely sequence of states. May not even be legal!
1 fair die, 1 “loaded” die, occasionally swapped
Figure 3.5

Rolls: Visible data—300 rolls of a die as described above.
Die: Hidden data—which die was actually used for that roll (F = fair, L = loaded).
Viterbi: the prediction by the Viterbi algorithm is shown.
Figure 3.6 The posterior probability of being in the state corresponding to the fair die in the casino example. The x axis shows the number of the roll. The shaded areas show when the roll was generated by the loaded die.
Posterior Decoding, II

Alternative 1: what’s most likely state at step $i$?

$$\hat{\pi}_i = \arg \max_k P(\pi_i = k \mid x)$$

Alternative 2: given some function $g(k)$ on states, what’s its expectation. E.g., what’s probability of “+” model in CpG HMM ($g(k)=1$ iff $k$ is “+” state)?

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

Data: 41 human sequences, totaling 60kbp, including 48 CpG islands of about 1kbp each

Viterbi:
Found 46 of 48
plus 121 “false positives”

Posterior Decoding:
same 2 false negatives
plus 236 false positives

Post-process:
46/48
67 false pos
46/48
83 false pos

Post-process: merge within 500; discard < 500
Training

Given model topology & training sequences, learn transition and emission probabilities

If $\pi$ known, then MLE is just frequency observed in training data

$$a_{k,l} = \frac{\text{count of } k \rightarrow l \text{ transitions}}{\text{count of } k \rightarrow \text{anywhere transitions}}$$

$$e_k(b) = \ldots$$

If $\pi$ hidden, then use EM:

- given $\theta$, estimate $\pi$;
- given $\pi$, estimate $\theta$; repeat

2 ways
Viterbi Training

given \( \theta \), estimate \( \pi \); given \( \pi \) estimate \( \theta \); repeat

Make initial estimates of parameters \( \theta \)
Find Viterbi path \( \pi \) for each training sequence
Count transitions/emissions on those paths, getting new \( \theta \)
Repeat

Not rigorously optimizing desired likelihood, but still useful & commonly used.
(Arguably good if you’re doing Viterbi decoding.)
Baum-Welch Training

EM: given $\theta$, estimate $\pi$ ensemble; then re-estimate $\theta$

$$
P(\pi_i = k, \pi_{i+1} = l \mid x, \theta) = \frac{f_k(i \mid \theta) a_{k,l} e_l(x_{i+1}) b_l(i + 1 \mid \theta)}{P(x \mid \theta)}
$$

Estimated $\hat{\#}$ of $k \rightarrow l$ transitions $\hat{A}_{k,l}$ on set of seqs $x^i$

$$
= \sum_{\text{training seqs } x^j} \sum_i P(\pi_i = k, \pi_{i+1} = l \mid x^j, \theta)
$$

New estimate $\hat{a}_{k,l} = \frac{\hat{A}_{k,l}}{\sum_l \hat{A}_{k,l}}$

Emissions: similar

AKA “the forward-backward alg”
Log-odds (vs all F) per roll

True model: 0.101 bits
300-roll est.: 0.097 bits
30k-roll est.: 0.100 bits
(NB: overestimated)
HMMs in Action: Pfam

http://pfam.xfam.org

Proteins fall into families, both across & within species

  Ex: Globins, Zinc fingers, Leucine zippers, GPCRs, ...

Identifying family very useful: suggests function, etc.

So, search & alignment are both important

Q. Why not just use Blast/Smith-Waterman?

A. There is more info in *multiple* examples (e.g., psiBLAST)

One very successful approach: profile HMMs
Alignment of 7 globins. A-H mark 8 alpha helices.
Consensus line: upper case = 6/7, lower = 4/7, dot=3/7.
Could we have a profile (aka weight matrix) w/ indels?
Profile Hmm Structure

Figure 5.2 *The transition structure of a profile HMM.*

M_j: Match states (20 emission probabilities)
I_j: Insert states (Background emission probabilities)
D_j: Delete states (silent - no emission)

From DEKM 53
Silent States

Example: chain of states, can skip some

Problem: many parameters.

A solution: chain of “silent” states; fewer parameters (but less detailed control)

Algorithms: basically the same.
Using Profile HMM’s

Search
Forward or Viterbi

Scoring
- Log likelihood (length adjusted)
- Log odds vs background
- Z scores from either

Alignment
Viterbi

} next slides
Likelihood vs Odds Scores

Figure 5.5 To the left the length-normalized LL score is shown as a function of sequence length. The right plot shows the same for the log-odds score.
Figure 5.6 The Z-score calculated from the LL scores (left) and the log-odds (right).
Pfam Model Building

Hand-curated “seed” multiple alignments
Train profile HMM from seed alignment
Hand-chosen score threshold(s)
Automatic classification/alignment of all other protein sequences

Pfam 25.0 (March 2011, 12273 families; covers ≈ 75% of human proteins)
Pfam 27.0 (March 2013, 14831 families; ≈ 90%)
Pfam 31.0 (March 2017, 16712 families)
Model-building refinements

Pseudocounts (count = 0 common when training with 20 aa’s)

\[ e_i(a) = \frac{C_{i,a} + A \cdot q_a}{\sum_a C_{i,a} + A}, \quad A \sim 20, \quad q_a = \text{background} \]

(≈50 training sequences)

Pseudocount “mixtures”, e.g. separate pseudocount vectors for various contexts (hydrophobic regions, buried regions,...)

(≈10-20 training sequences)
More refinements

Weighting: may need to down weight highly similar sequences to reflect phylogenetic or sampling biases, etc.

Match/insert assignment: Simple threshold, e.g. “> 50% gap ⇒ insert”, may be suboptimal. Can use forward-algorithm-like dynamic programming to compute max a posteriori assignment.
Numerical Issues

Products of many probabilities $\rightarrow 0$
For Viterbi: just add logs
For forward/backward: also work with logs, but you need sums of products, so need “log-of-sum-of-product-of-exp-of-logs”, e.g., by table/interpolation
Keep high precision and perhaps scale factor
Working with log-odds also helps.
Model structure

Define it as well as you can.

In principle, you can allow all transitions and hope to learn their probabilities from data, but it usually works poorly – too many local optima.
Duration Modeling

Self-loop duration:
- geometric $p^n(1-p)$
- min, then geometric
- “negative binomial”
- More general: possible (but slower)
HMM Summary

Inference
- Viterbi – best single path (max of products)
- Forward – sum over all paths (sum of products)
- Backward – similar
- Posterior decoding

Model building
- Semi-supervised – typically fix architecture (e.g. profile HMM), then learn parameters
- Baum-Welch – training via EM and forward/backward (aka the forward/backward algorithm)
- Viterbi training – also “EM-like”, but Viterbi-based
HMM Summary (cont.)

Search:
  Viterbi or forward

Scoring:
  Odds ratio to background
  Z-score
  E-values, etc., too

Excellent tools available (SAM, HMMer, Pfam, …)

A very widely used tool for biosequence analysis