CSE 527
Lectures ~12-13

Markov Models and Hidden Markov Models
Markov & Hidden Markov Models

Independence

- A key issue: All models we’ve talked about so far assume *independence* of nucleotides in different positions - definitely unrealistic.
Example: “CpG Islands”

- CpG - 2 adjacent nucs, same strand (not Watson-Crick pair)
- C of CpG is often methylated (in Eukaryotes)
- 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 promoter (& other) regions, CpG remain unmethylated, so CpG → TpG less likely there: makes “CpG Islands”; often mark gene-rich regions
CpG Islands

- CpG Islands
  - More CpG than elsewhere
  - More C & G than elsewhere, too
  - Typical length: few 100 to few 1000 bp

- Questions
  - Given short sequence (say 200 bp), is it a CpG island or not?
  - Given long sequence (say, 10-100kb), find CpG islands in it?
Markov Chains

A sequence $x_1, x_2, \ldots$ of random variables is a \textit{k-th order Markov chain} if, for all $i$:

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

i.e., $i^{th}$ value is independent of all but the previous $k$ values

- Example 1: Uniform random ACGT
- Example 2: Weight matrix model
- Example 3: ACGT, but $\downarrow \text{Pr(G following C)}$

\{0th order, 1st order\}
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 $x$

$x = x_1 x_2 \ldots x_n$

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

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

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

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

$= \prod_{i=0}^{n-1} a_{x_i, x_{i+1}}$ (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:
Discrimination/Classification

Log likelihood ratio of CpG model vs background model

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

<table>
<thead>
<tr>
<th>( \beta )</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>
Figure 3.2 The histogram of the length-normalised scores for all the sequences. CpG islands are shown with dark grey and non-CpG with light grey.
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(s)?” not “Which model?”
Hidden Markov Models (HMMs)

States: \(1, 2, 3, \ldots\)
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></th>
<th>1: 1/6</th>
<th>2: 1/6</th>
<th>3: 1/6</th>
<th>4: 1/6</th>
<th>5: 1/6</th>
<th>6: 1/6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1: 1/10</th>
<th>2: 1/10</th>
<th>3: 1/10</th>
<th>4: 1/10</th>
<th>5: 1/10</th>
<th>6: 1/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

.95 → .05 → .10 → .90
<table>
<thead>
<tr>
<th>Rolls</th>
<th>3151162464464424531132163116415213362514454363165662656666</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die</td>
<td>FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLL</td>
</tr>
<tr>
<td>Viterbi</td>
<td>FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rolls</th>
<th>6511664531326512456366646316366316232645523626666625151631</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die</td>
<td>LLLLLLLLLLLLLLLLLLLLLLFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
<tr>
<td>Viterbi</td>
<td>LLLLLLLLLLLLLLLLLLLLLLFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rolls</th>
<th>22255441666565636564324364131513465146353411126414626253356</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die</td>
<td>FFFFFFFFFLLLMLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
<tr>
<td>Viterbi</td>
<td>FFFFFFFFFLLLMLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rolls</th>
<th>366163666466232534413661661163252562462255265252266643535336</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die</td>
<td>LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
<tr>
<td>Viterbi</td>
<td>LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rolls</th>
<th>23312162536441443233516324363365562466662632666612355245242</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die</td>
<td>FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLL</td>
</tr>
<tr>
<td>Viterbi</td>
<td>FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLL</td>
</tr>
</tbody>
</table>

Figure 3.5 The numbers show 300 rolls of a die as described in the example. Below is shown which die was actually used for that roll (F for fair and L for loaded). Under that the prediction by the Viterbi algorithm is shown.
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:

1. Most probable single path
   $$\pi^* = \arg \max \ P(x, \pi)$$

2. Sequence of most probable states
   $$\hat{\pi}_i = \arg \max_k \ P(\pi_i = k \mid x)$$

3. ...
The Viterbi Algorithm: The most probable path

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

- Possibly there are \(10^{99}\) paths of prob \(10^{-99}\)
- More commonly, one path dominates others.
  (If not, other approaches may be preferable.)
- Key problem: exponentially many paths \(\pi\)
• Conceptually, sometimes convenient
• Note exponentially many paths
$v_l(i) =$ probability of the most probable path emitting $x_1, x_2, \ldots, x_i$ and ending in state $l$

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

Initialize:

$v_l(0) = \begin{cases} 
1 & \text{if } l = \text{Begin state} \\
0 & \text{otherwise}
\end{cases}$
Viterbi Traceback

- Above finds \textit{probability} of best path
- To find the path itself, trace \textit{backward} to the state \(k\) attaining the max at each stage
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 (Viterbi is not the only interesting answer.)
An HMM (unrolled)
Viterbi: best path to each state

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

For each state/time, want total probability of all paths leading to it, with given emissions

\[ f_k(i) = P(x_1 \ldots x_i, \pi_i = k) \]
\[ f_l(i + 1) = e_l(x_{i+1}) \sum_k f_k(i) a_{k,l} \]
\[ P(x) = \sum_\pi P(x, \pi) = \sum_k f_k(n) a_{k,0} \]
The Backward Algorithm

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

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

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

\[ b_k(n) = a_{k,0} \]
In state $k$ at step $i$?

\[ P(x, \pi_i = k) \]

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

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

\[ = f_k(i) \cdot b_k(i) \]

\[ P(\pi_i = k | x) = \frac{P(x, \pi_i = k)}{P(x)} = \frac{f_k(i) \cdot b_k(i)}{P(x)} \]
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!
The Occasionally Dishonest Casino

1 fair die, 1 “loaded” die, occasionally swapped
<table>
<thead>
<tr>
<th>Rolls</th>
<th>315116246446644245311321631164152133625144543631656626566666</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die</td>
<td>FFFFLLLLLLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
<tr>
<td>Viterbi</td>
<td>FFFFLLLLLLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
<tr>
<td>Rolls</td>
<td>65116453132651245636664631636631623264553626666625151631</td>
</tr>
<tr>
<td>Die</td>
<td>LLLLLFFFFLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
<tr>
<td>Viterbi</td>
<td>LLLLLFFFFLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
<tr>
<td>Rolls</td>
<td>2225554166656656356434613153146514635411126414626253356</td>
</tr>
<tr>
<td>Die</td>
<td>FFFFFFLLLLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
<tr>
<td>Viterbi</td>
<td>FFFFFFLLLLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
<tr>
<td>Rolls</td>
<td>366163666466232534136616611632525624622552652522664353336</td>
</tr>
<tr>
<td>Die</td>
<td>LLLLLLLLLLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
<tr>
<td>Viterbi</td>
<td>LLLLLLLLLLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
<tr>
<td>Rolls</td>
<td>233121662536441443233516324363365562466662632666612355245242</td>
</tr>
<tr>
<td>Die</td>
<td>FFFFLLLLLLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
<tr>
<td>Viterbi</td>
<td>FFFFLLLLLLLLLLLLLLLLLLLLLLLLLL</td>
</tr>
</tbody>
</table>

**Figure 3.5** The numbers show 300 rolls of a die as described in the example. Below is shown which die was actually used for that roll (F for fair and L for loaded). Under that the prediction by the Viterbi algorithm is shown.
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

(merge within 500; discard < 500)
COMBI Seminar

Dr. William Noble
“Identifying remote protein homologs by network propagation”

Wednesday, November 16, 2005
1:30-2:30pm
HSB K-069
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 $\pi$, estimate $\theta$; given $\theta$ estimate $\pi$.  

\{ 2 ways \}
Viterbi Training

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

- 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

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 $\#$ of $k \rightarrow l$ transitions $\hat{A}_{k,l}$

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

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

Emissions: similar
True Model

Learned Model (300 rolls)

Log-odds per roll

True model 0.101 bits
300-roll est. 0.097 bits
30k-roll est. 0.100 Bits

(NB: overfitting)
HMM Summary

- Viterbi – best single path \((\text{max of products})\)
- Forward – Sum over all paths \((\text{sum of products})\)
- Backward – similar
- Baum-Welch – Training via EM and forward/backward (aka the forward/backward algorithm)
- Viterbi training – also EM, but Viterbi-based
HMMs in Action: Pfam

- Proteins fall into families, both across & within species
  - Ex: Globins, GPCRs, Zinc Fingers, Leucine zippers,...
- Identifying family is very useful, suggests function, etc.
- So, search & alignment are both important
- 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.

Mj: Match states (20 emission probabilities)
Ij: Insert states (Background emission probabilities)
Dj: Delete states (silent - no emission)
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.
How Profile HMM’s used

- Search
  - Forward or Viterbi
- Scoring
  - Log likelihood (length adjusted)
  - Log odds vs background
  - Z scores from either
- Alignment
  - Viterbi

next slides
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.
Z-Scores

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
- 7973 families in Rfam 18.0, 8/2005 (covers ~75% of proteins)
Model building refinements

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

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

($\sim$50 training sequences)

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

($\sim$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.
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)
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.