CSCE 970 Lecture 5: Hidden Markov Models

Stephen D. Scott

February 14, 2001 ♡

Introduction

- When classifying sequence data, cannot assume that the label of one part is independent of the other parts
  - E.g. natural language understanding, speech recognition, genomic sequences
- So we'll model a sequence of classes, conditioned on sequence of observed feature vectors
- For other applications, we may want to assign a single label to an entire (arbitrary length) sequence of feature vectors
  - Build multiple models, one per type
- In both cases, the Markov model is a sequence generator
  - We classify a new feature vector by measuring how likely it is generated by the model

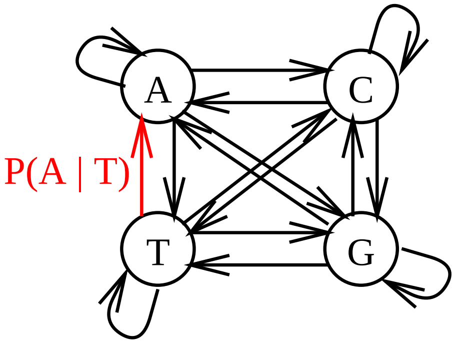
An Example from Computational Biology

CpG Islands

- Genomic sequences are one-dimensional series of letters from {A,C,G,T}, frequently many thousands of letters (bases, nucleotides, residues) long
- The sequence "CG" (written "CpG") tends to appear more frequently in some places than in others
- Such CpG islands are usually 10^2–10^3 bases long
- Questions:
  1. Given a short segment, is it from a CpG island?
  2. Given a long segment, where are its islands?

Modeling CpG Islands

- Model will be a CpG generator
- Since focusing on sequence data, want probability of next symbol to depend on current symbol
- Will use a standard (non-hidden) Markov model
  - Probabilistic state machine
  - Each state emits a symbol



The Markov Property

- A first-order Markov model (what we study) has the property that observing feature vector (symbol) $\mathbf{x}_i$ while in state π_i depends only on the previous state π_{i-1} (which generated $\mathbf{x}_{i-1}$)

- Standard model has 1-1 correspondence between symbols and states, thus

  $$P(\mathbf{x}_i \mid \mathbf{x}_{i-1}, \ldots, \mathbf{x}_1) = P(\mathbf{x}_i \mid \mathbf{x}_{i-1})$$

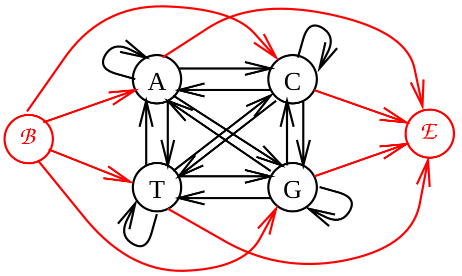
  and

  $$P(\mathbf{x}_1, \ldots, \mathbf{x}_L) = P(\mathbf{x}_1) \prod_{i=2}^{L} P(\mathbf{x}_i \mid \mathbf{x}_{i-1})$$

Begin and End States

- For convenience, can add special "begin" ($\mathcal{B}$) and "end" ($\mathcal{E}$) states to clarify equations and define a distribution over sequence lengths

- These states emit empty (null) symbols $\mathbf{x}_0$ and $\mathbf{x}_{L+1}$ to mark ends of sequence



$$P(\mathbf{x}_1, \ldots, \mathbf{x}_L) = \prod_{i=1}^{L+1} P(\mathbf{x}_i \mid \mathbf{x}_{i-1})$$

- Will represent both with single state named 0

Markov Chains for Discrimination

- How do we use this to differentiate islands from non-islands?

- Define two Markov models: one for islands ("+" or ω_1), one for non-islands ("–", ω_2)

  - Each model gets 4 states (A, C, G, T)

  - Take training set of known islands and non-islands

  - Let c_{st}^{+} = number of times symbol t followed symbol s in an island, then set:

    $$P^{+}(t \mid s) = \frac{c_{st}^{+}}{\sum_{t'} c_{st'}^{+}}$$

- Example probabilities in [Durbin et al., p. 50]

- Now score a sequence $X = \langle \mathbf{x}_1, \ldots, \mathbf{x}_L \rangle$ by summing the log-odds ratios:

$$\log\left(\frac{P(X \mid +)}{P(X \mid -)}\right) = \sum_{i=1}^{L+1} \log\left(\frac{P^{+}(\mathbf{x}_i \mid \mathbf{x}_{i-1})}{P^{-}(\mathbf{x}_i \mid \mathbf{x}_{i-1})}\right)$$

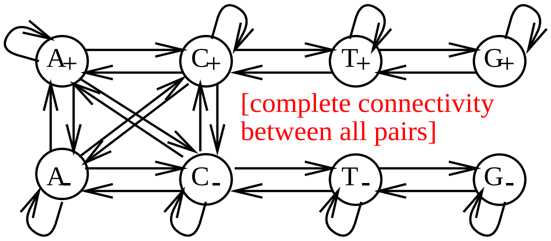
Hidden Markov Models

- Recall the second CpG question: Given a long sequence, where are its islands?

  - Could use tools just presented by passing a fixed-width window over the sequence and computing scores

  - Trouble if islands' lengths vary

  - Prefer single, unified model for islands vs. non-islands

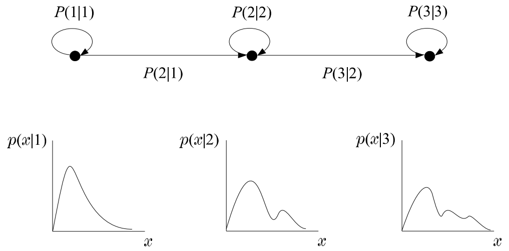


[complete connectivity between all pairs]

  - Within the + group, transition probabilities similar to those for the separate + model, but there is a small chance of switching to a state in the – group

What's Hidden in an HMM?

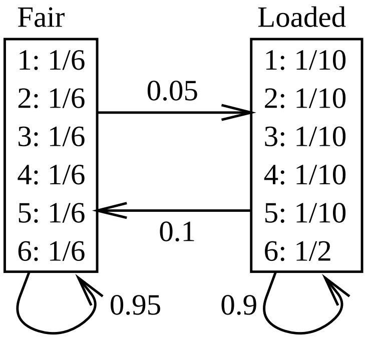
- No longer have one-to-one correspondence between states and emitted characters
  - E.g. was C emitted by C_+ or C_-?
- Must differentiate the symbol sequence X from the state sequence $\pi = \langle \pi_1, \ldots, \pi_L \rangle$
  - State transition probabilities same as before: $P(\pi_i = \ell \mid \pi_{i-1} = j)$ (i.e. $P(\ell \mid j)$)
  - Now each state has a prob. of emitting any value: $p(\mathbf{x}_i = \mathbf{x} \mid \pi_i = j)$ (i.e. $p(\mathbf{x} \mid j)$)



[In CpG HMM, emission probs discrete and $= 0$ or 1]

Example: The Occasionally Dishonest Casino

- Assume that a casino is typically fair, but with probability 0.05 it switches to a loaded die, and switches back with probability 0.1



- Given a sequence of rolls, what's hidden?

The Viterbi Algorithm

- Probability of seeing symbol sequence X and state sequence π is

$$P(X, \pi) = P(\pi_1 \mid 0) \prod_{i=1}^{L} p(\mathbf{x}_i \mid \pi_i)\, P(\pi_{i+1} \mid \pi_i)$$

- Can use this to mind most likely path:

$$\pi^* = \operatorname*{argmax}_{\pi} P(X, \pi)$$

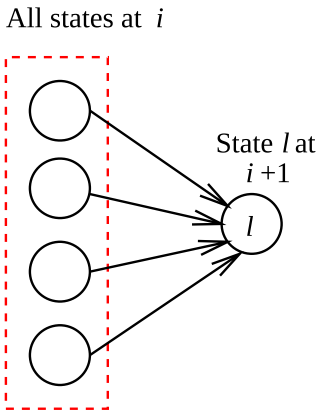
and trace it to identify islands (paths through $+$ states)

- There are an exponential number of paths through chain, so how do we find the most likely one?

The Viterbi Algorithm
(cont'd)

- Assume that we know (for all k) $v_k(i)$ = probability of most likely path ending in state k with observation $\mathbf{x}_i$
- Then

$$v_\ell(i+1) = p(\mathbf{x}_{i+1} \mid \ell) \max_k \{v_k(i)\, P(\ell \mid k)\}$$



The Viterbi Algorithm
(cont'd)

- Given the formula, can fill in table with dynamic programming:
  - $v_0(0) = 1$, $v_k(0) = 0$ for $k > 0$
  - For $i = 1$ to L
    - $v_\ell(i) = p(\mathbf{x}_i \mid \ell) \max_k\{v_k(i-1)P(\ell \mid k)\}$
    - $\mathrm{ptr}_i(\ell) = \mathrm{argmax}_k\{v_k(i-1)P(\ell \mid k)\}$
  - $P(X, \pi^*) = \max_k\{v_k(L)P(0 \mid k)\}$
  - $\pi^*_L = \mathrm{argmax}_k\{v_k(L)P(0 \mid k)\}$
  - For $i = L$ to 1
    - $\pi^*_{i-1} = \mathrm{ptr}_i(\pi^*_i)$
- To avoid underflow, use $\log(v_\ell(i))$ and add

The Forward Algorithm

- Given a sequence X, find $P(X) = \sum_\pi P(X, \pi)$
- Use dynamic programming like Viterbi, replacing max with sum, and $v_k(i)$ with $f_k(i) = P(\mathbf{x}_1, \ldots, \mathbf{x}_i, \pi_i = k)$ (= prob. of observed sequence through $\mathbf{x}_i$, stopping in state k)
  - $f_0(0) = 1$, $f_k(0) = 0$ for $k > 0$
  - For $i = 1$ to L
    - $f_\ell(i) = p(\mathbf{x}_i \mid \ell) \sum_k f_k(i-1)P(\ell \mid k)$
  - $P(X) = \sum_k f_k(L)P(0 \mid k)\}$
- To avoid underflow, can again use logs, though exactness of results compromised

The Backward Algorithm

- Given a sequence X, find the probability that $\mathbf{x}_i$ was emitted by state k, i.e.

$$P(\pi_i = k \mid X) = \frac{P(\pi_i = k, X)}{P(X)}$$

$$= \frac{\overbrace{P(\mathbf{x}_1, \ldots, \mathbf{x}_i, \pi_i = k)}^{f_k(i)} \; \overbrace{P(\mathbf{x}_{i+1}, \ldots, \mathbf{x}_L \mid \pi_i = k)}^{b_k(i)}}{\underbrace{P(X)}_{\text{computed by forward alg}}}$$

- Algorithm:
  - $b_k(L) = P(0 \mid k)$ for all k
  - For $i = L - 1$ to 1
    - $b_k(i) = \sum_\ell P(\ell \mid k)\, p(\mathbf{x}_{i+1} \mid \ell)\, b_\ell(i+1)$

Example Use of Forward/Backward Algorithm

- Define $g(k) = 1$ if $k \in \{\mathrm{A}_+, \mathrm{C}_+, \mathrm{G}_+, \mathrm{T}_+\}$ and 0 otherwise
- Then $G(i \mid X) = \sum_k P(\pi_i = k \mid X)\, g(k)$ = probability that $\mathbf{x}_i$ is in an island
- For each state k, compute $P(\pi_i = k \mid X)$ with forward/backward algorithm
- Technique applicable to any HMM where set of states is partitioned into classes
  - Use to label individual parts of a sequence

Specifying an HMM

- Two problems: defining structure (set of states) and parameters (transition and emission probabilities)
- Start with latter problem, i.e. given a training set $X_1, \ldots, X_N$ of independently generated sequences, learn a good set of parameters θ
- Goal is to maximize the (log) likelihood of seeing the training set given that θ is the set of parameters for the HMM generating them:

$$\sum_{j=1}^{N} \log(P(X_j;\theta))$$

When State Sequence Known

- Estimating parameters when e.g. islands already identified in training set
- Let $A_{k\ell}$ = number of $k \to \ell$ transitions and $E_k(b)$ = number of emissions of b (assume scalar) in state k

$$P(\ell \mid k) = A_{k\ell} \Big/ \left(\sum_{\ell'} A_{k\ell'} \right)$$

$$P(b \mid k) = E_k(b) \Big/ \left(\sum_{b'} E_k(b') \right)$$

- Be careful if little training data available
  - E.g. an unused state k will have undefined parameters
  - Workaround: Add pseudocounts $r_{k\ell}$ to $A_{k\ell}$ and $r_k(b)$ to $E_k(b)$ that reflect prior biases about parobabilities
  - Increased train data decr. prior's influence
  - More general priors [Sjölander et al. 96]

The Baum-Welch Algorithm

- Used for estimating parameters when state sequence unknown in training set
- Special case of the expectation maximization (EM) algorithm [Theod, pp. 36–39, 443–445]
- Start with arbitrary $P(\ell \mid k)$ and $P(b \mid k)$, and use them to estimate $A_{k\ell}$ and $E_k(b)$ as the expected number of occurrences given the training set*:

$$A_{k\ell} = \sum_{j=1}^{N} \frac{1}{P(X_j)} \sum_{i=1}^{L} f_k^j(i) P(\ell \mid k) P(\mathbf{x}_{i+1}^j \mid \ell) b_\ell^j(i+1)$$

$$E_k(b) = \sum_{j=1}^{N} \sum_{i:\mathbf{x}_i^j=b} P(\pi_i = k \mid X_j) = \sum_{j=1}^{N} \frac{1}{P(X_j)} \sum_{i:\mathbf{x}_i^j=b} f_k^j(i) b_k^j(i)$$

- Then use these numbers (& pseudocounts) to recompute $P(\ell \mid k)$ and $P(b \mid k)$
- After each iteration, compute log likelihood and halt if little improvement

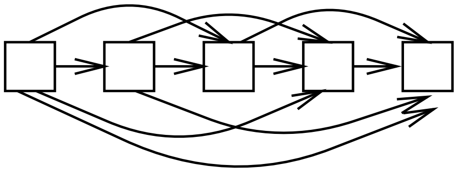
*Superscript j corresponds to jth train example

HMM Structure

- How to specify HMM states and connections?
- States come from background knowledge on problem, e.g. size-4 alphabet, $+/-$, $\Rightarrow$ 8 states
- Connections:
  - Tempting to specify complete connectivity and let Baum-Welch sort it out
  - Problem: Huge number of parameters could lead to local max
  - Better to use background knowledge to invalidate some connections by initializing $P(\ell \mid k) = 0$
    - Baum-Welch will respect this

Silent States

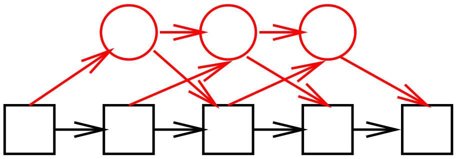
- May want to allow model to generate sequences with certain parts deleted
  - E.g. when aligning DNA or protein sequences against a fixed model or matching a sequence of spoken words against a fixed model, some parts of the input might be omitted



- Problem: Huge number of connections, slow training, local maxima

Silent States
(cont'd)

- Silent states (like begin and end states) don't emit symbols, so they can "bypass" a regular state



- If there are no purely silent loops, can update Viterbi, forward, and backward algorithms to work with silent states [Durbin et al., p. 71]

- Used extensively in profile HMMs for modeling sequences of protein families (aka multiple alignments)