Hidden Markov Models

Stochastic Regular Grammars

I. Holmes

Department of Bioengineering
University of California, Berkeley

Spring semester

Outline

- Early motivation: isochores
  - Long regions of uniform GC content (which is correlated with gene density, recombination frequency...)
    - e.g. Major Histocompatibility Complex (MHC) class II and class III sequences on human chromosome 6
    - Lengths 900.9 kb, 642.1 kb; GC-content 41%, 52%
  - Gary Churchill: first Hidden Markov Model for isochore detection (1989)
  - Earliest non-thermodynamic hit to “isochore” on PubMed is 1986, Alonso *et al*
  - HMM analogy: occasionally dishonest casino (Durbin *et al*)

Notation

- Hidden Markov model: notation
  - Let x denote hidden state, y observed symbol. State space includes START and END
  - Let $e(x, y)$ be probability of emitting character y in state x
  - Let $t(i, j)$ be probability of transition to state j if currently in state i

- Idea of a particular partitioning as a “path” through the HMM
  - Let y_n be observed nucleotide at position n and let x_n be hidden state of position n
    - Let L be length of sequence
    - For convenience, set x_0 =START and x_{L+1} =END
    - Let $Y = \{y_1 \dots y_L\}$ represent entire observed sequence, $X = \{x_0 \dots x_{L+1}\}$ hidden state sequence
    - Each $X = \{x_k\}$ represents a “path” (sketch); there are $\sim K^L$ possible paths for K states
  - Joint likelihood is $P(X, Y) = t(x_0, x_1) \prod_{k=1}^{L} e(x_k, y_k) t(x_k, x_{k+1})$
    - Note that $P(X, Y) \equiv P(X, Y|x_0)$. Conditioning on x_0 is assumed throughout
- Two questions, two algorithms. (i) Viterbi: what X maximises $P(X, Y)$? (ii) Forward: what is $P(Y)$?

Viterbi algorithm for finding ML hidden state path

$$\begin{aligned}
\max_X P(X,Y) &= \max_{x_L}\max_{x_{L-1}}\dots\max_{x_1} t(x_0,x_1)\prod_{k=1}^{L} e(x_k,y_k)t(x_k,x_{k+1}) \\
&= \max_{x_L} t(x_L,x_{L+1})e(x_L,y_L)\max_{x_{L-1}} t(x_{L-1},x_L)e(x_{L-1},y_{L-1})\dots\max_{x_1} t(x_1,x_2)e(x_1,y_1)t(x_0,x_1) \\
&= \max_{x_L} t(x_L,\text{END})V_L(x_L)
\end{aligned}$$

where

$$V_n(x_n) = \begin{cases} e(x_n,y_n)\max_{x_{n-1}} t(x_{n-1},x_n)V_{n-1}(x_{n-1}) & \text{if } n>0 \\ 1 & \text{if } n=0 \end{cases}$$

Note that

$$V_n(x_n) = \max_{x_1\dots x_{n-1}} P(x_1\dots x_n, y_1\dots y_n | x_0)$$

or, in words, $V_n(x)$ is the maximum likelihood of any path ending in state x and emitting symbols $y_1 \dots y_n$.

The ML state sequence, $\text{argmax}_X P(X,Y)$, is recovered by **traceback** from V_L to V_1.

Forward algorithm for summing over all possible hidden state paths

$$\begin{aligned}
P(Y) &= \sum_X P(X,Y) \\
&= \sum_{x_L} \sum_{x_{L-1}} \dots \sum_{x_1} t(x_0,x_1) \prod_{k=1}^{L} e(x_k,y_k) t(x_k,x_{k+1}) \\
&= \sum_{x_L} t(x_L,x_{L+1}) e(x_L,y_L) \sum_{x_{L-1}} t(x_{L-1},x_L) e(x_{L-1},y_{L-1}) \dots \sum_{x_1} t(x_1,x_2) e(x_1,y_1) t(x_0,x_1) \\
&= \sum_{x_L} t(x_L,\mathrm{END}) F_L(x_L)
\end{aligned}$$

where

$$F_n(x_n) = \begin{cases} e(x_n,y_n) \sum_{x_{n-1}} t(x_{n-1},x_n) F_{n-1}(x_{n-1}) & \text{if } n>0 \\ 1 & \text{if } n=0 \end{cases}$$

Note that

$$F_n(x_n) = P(x_n, y_1 \dots y_n | x_0) = \sum_{x_1 \dots x_{n-1}} P(x_1 \dots x_n, y_1 \dots y_n | x_0)$$

or, in words, $F_n(x)$ is the sum of likelihoods of all paths ending in state x and emitting symbols $y_1 \dots y_n$.

By analogy to Viterbi traceback, a ML state sequence can be sampled by stochastic traceback from F_L to F_1.

Implementation issues

- probability underflow for long sequences
- slow floating-point multiply
- motivation for (possibly discretized) log-space scores

Definition of the posterior probability that position n is in state k: sum over paths

$$P(x_n = k|Y) = \frac{\sum_X P(X,Y)\delta(x_n = k)}{P(Y)}$$

Splitting the path into three parts: $< n$, $= n$ and $> n$

$$\begin{aligned}
P(x_n|Y) &= \sum_{x_1 \ldots x_{n-1}} \sum_{x_{n+1} \ldots x_L} \\
&\qquad \frac{P(x_1 \ldots x_n, y_1 \ldots y_n | x_0)\; P(x_{n+1} \ldots x_{L+1}, y_{n+1} \ldots y_L | x_n)}{P(Y)} \\
&= \frac{F_n(x_n)\; B_n(x_n)}{P(Y)} \\
B_n(x_n) &= P(x_{L+1}, y_{n+1} \ldots y_L | x_n) \\
&= \sum_{x_{n+1} \ldots x_L} P(x_{n+1} \ldots x_{L+1}, y_{n+1} \ldots y_L | x_n)
\end{aligned}$$

- Likewise,

$$P(x_n, x_{n+1} | Y) = \frac{F_n(x_n) t(x_n, x_{n+1}) e(x_{n+1}, y_{n+1}) B_{n+1}(x_{n+1})}{P(Y)}$$

  and (useful for compression)

$$P(y_{n+1} = k | y_1 \ldots y_n) = \frac{\sum_i \sum_j F_n(i) t(i,j) e(j,k)}{\sum_i F_n(i)}$$

- The Backward algorithm

$$B_n(x_n) = \begin{cases} \sum_{x_{n+1}} t(x_n, x_{n+1}) e(x_{n+1}, y_{n+1}) B_{n+1}(x_{n+1}) & \text{if } n < L \\ t(x_L, \text{END}) & \text{if } n = L \end{cases}$$

The Baum-Welch training algorithm:

$$P(X, Y) = \left(\prod_{i,j} t(i,j)^{u(i,j)} \right) \left(\prod_{i,k} e(i,k)^{f(i,k)} \right)$$

where $u(i,j)$ is the number of transitions $i \rightarrow j$ and $f(i,k)$ is the number of emissions of character k from state i. Sufficient statistics for EM algorithm are therefore

$$\begin{aligned} \hat{t}(i,j) &= \sum_{n=0}^{L} P(x_n = i, x_{n+1} = j | Y) \\ \hat{e}(i,k) &= \sum_{n=1}^{L} P(x_n = i | Y) \delta(y_n = k) \end{aligned}$$

where $\hat{t}$ and $\hat{e}$ are the posterior expectations of u and f. Dirichlet (mixture) priors can be used for t and e.

Note that $\hat{t}(i,j) = \frac{\partial \log P(X)}{\partial \log t(i,j)}$ and $\hat{e}(i,k) = \frac{\partial \log P(X)}{\partial \log e(i,k)}$

Implementation issues: log-space probability addition

$$\begin{aligned} \log(e^a + e^b) &= \max(a,b) + \oplus(|a-b|) \\ \oplus(|a-b|) &= \log(1 + e^{-|a-b|}) \end{aligned}$$

If scores are discretized, can implement $\oplus$ as a lookup table for speed.

Null states

- Sparse transition matrices are good (Forward/Backward and Viterbi time complexities are $\propto$ no. of transitions)
- Convenience: reduce number of transitions (e.g. delete states of profile HMM)
- Need to do a *topological sort* of null states so that they're filled in the correct order
- Awkwardness: null cycles. These break the toposort

Can always eliminate null states as follows

$$\mathbf{t} = \mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d}$$

$$\mathbf{a} = \left(\begin{array}{cc} * & 0 \\ 0 & 0 \end{array} \right), \mathbf{b} = \left(\begin{array}{cc} 0 & * \\ 0 & 0 \end{array} \right), \mathbf{c} = \left(\begin{array}{cc} 0 & 0 \\ * & 0 \end{array} \right), \mathbf{d} = \left(\begin{array}{cc} 0 & 0 \\ 0 & * \end{array} \right)$$

$$\mathbf{t}' = \left(\begin{array}{cc} * & 0 \\ 0 & 0 \end{array} \right) = \mathbf{a} + \sum_{n=0}^{\infty} \mathbf{b}\mathbf{d}^n\mathbf{c} = \mathbf{a} + \mathbf{b}(\mathbf{I} - \mathbf{d})^{-1}\mathbf{c}$$

where **a** (and $\mathbf{t}'$) contain emit→emit transitions, **b** emit→null, **c** null→emit and **d** null→null.

- Factor graph representation of HMM; similarity to pruning/peeling and parsimony (Forward-Backward $\subset$ Sum-Product)
- General-purpose HMM implementations: DART library (C++)

- Motivation: pairwise sequence alignment, pairwise genefinding, etc.
- Let x denote hidden state, y character in sequence Y, z character in sequence Z
- Let $\Delta y(x)$ be 1 if state x emits a character to Y, and 0 otherwise; likewise $\Delta z(x) = 1$ iff x emits to Z

- Emission probability $e(x, y, z)$ is defined as follows:
  - If $\Delta y(x) = 1$ and $\Delta z(x) = 0$, then x is called a **delete** state and $e(x, y, z) \equiv e_d(x, y)$ is a function of x and y only
  - If $\Delta y(x) = 0$ and $\Delta z(x) = 1$, then x is called an **insert** state and $e(x, y, z) \equiv e_i(x, z)$ is a function of x and z only
  - If $\Delta y(x) = 1$ and $\Delta z(x) = 1$, then x is called a **match** state and $e(x, y, z) \equiv e_m(x, y, z)$ is a function of x, y and z
  - If $\Delta y(x) = 0$ and $\Delta z(x) = 0$, then x is called a **null** state and $e(x, y, z)$ is a function of x only (typically just 1)
  - We will assume for now that there are no null states (apart from START and END).

- As before, $t(i,j)$ is the probability of transition to state j if currently in state i
- Suppose sequence lengths are K, L so observed data are $Y = \{y_1 \dots y_K\}$ and $Z = \{z_1 \dots z_L\}$
- Again we have a state path $x_1, x_2 \dots x_N$ and for convenience we set $x_0 =$START and $x_{N+1} =$END.
    - Denote by Λ_{kl} the event that there exists a *break* at (k,l):

$$\Lambda_{kl} \ \Rightarrow \ \exists n : \sum_{i=1}^{n} \Delta y(x_i) = k, \ \sum_{i=1}^{n} \Delta z(x_i) = l$$

    So Λ_{kl} means that, at some point n on the state path, the model has emitted k symbols to Y and l symbols to Z.

Viterbi

$$V_{kl}(x_n) = \max_{x_1 \ldots x_{n-1}} P\left(\Lambda_{kl}, x_1 \ldots x_n, y_1 \ldots y_k, z_1 \ldots z_l | x_0\right)$$

Recursion (assuming no null states)

$$V_{kl}(x_n) = \begin{cases} e(x_n, y_k, z_l) \max\limits_{x_{n-1}} t(x_{n-1}, x_n) & \\ \quad \times V_{k-\Delta y(x_n), l-\Delta z(x_n)}(x_{n-1}) & \text{if } k > 0 \text{ or } l > 0 \\ 1 & \text{if } k = l = 0 \text{ and } x_n = \text{START} \\ 0 & \text{if } k = l = 0 \text{ and } x_n \neq \text{START} \\ 0 & \text{if } k < 0 \text{ or } l < 0 \end{cases}$$

Forward

$$\begin{aligned} F_{kl}(x_n) &= P\left(\Lambda_{kl}, x_n, y_1 \ldots y_k, z_1 \ldots z_l | x_0\right) \\ &= \sum_{x_1 \ldots x_{n-1}} P\left(\Lambda_{kl}, x_1 \ldots x_n, x_{N+1}, y_1 \ldots y_k, z_1 \ldots z_l | x_0\right) \end{aligned}$$

Recursion (assuming no null states)

$$F_{kl}(x_n) = \begin{cases} e(x_n, y_k, z_l) \sum_{x_{n-1}} t(x_{n-1}, x_n) & \\ \quad \times F_{k-\Delta y(x_n), l-\Delta z(x_n)}(x_{n-1}) & \text{if } k > 0 \text{ or } l > 0 \\ 1 & \text{if } k = l = 0 \text{ and } x_n = \text{START} \\ 0 & \text{if } k = l = 0 \text{ and } x_n \neq \text{START} \\ 0 & \text{if } k < 0 \text{ or } l < 0 \end{cases}$$

Backward

$$\begin{aligned} B_{kl}(x_n) &= P\left(\Lambda_{kl}, x_{N+1}, y_{k+1} \dots y_K, z_{l+1} \dots z_L \middle| x_n\right) \\ &= \sum_{x_{n+1} \dots x_N} P\left(\Lambda_{kl}, x_{n+1} \dots x_{N+1}, y_{k+1} \dots y_K, z_{l+1} \dots z_L \middle| x_n\right) \end{aligned}$$

Recursion (assuming no null states)

$$B_{kl}(x_n) = \begin{cases} \displaystyle\sum_{x_{n+1}} t(x_n, x_{n+1}) e(x_{n+1}, y_{k+1}, z_{l+1}) & \\ \quad \times B_{k+\Delta y(x_n+1), l+\Delta z(x_n+1)}(x_{n+1}) & \text{if } k < K \text{ or } l < L \\ t(x_n, \text{END}) & \text{if } k = K \text{ and } l = L \\ 0 & \text{if } k > K \text{ or } l > L \end{cases}$$

Evidence, posterior probabilities & EM counts

$$
\begin{aligned}
P(Y,Z) &= \sum_x F_{KL}(x) t(x, \text{END}) \\
P(\Lambda_{kl}, x_n | Y, Z) &= \frac{F_{kl}(x_n) B_{kl}(x_n)}{P(Y)} \\
P(\Lambda_{kl}, x_n, x_{n+1} | Y) &= \frac{F_{kl}(x_n) t(x_n, x_{n+1}) e(x_{n+1}, y_{k+1}, z_{l+1}) B_{k+\Delta y(x_n+1), l+\Delta z(x_n+1)}(x_{n+1})}{P(Y)} \\
\hat{t}(i,j) &= \sum_{k=0}^{K} \sum_{l=0}^{L} P(\Lambda_{kl}, x_n = i, x_{n+1} = j | Y, Z) \\
\hat{e}_m(x,y,z) &= \sum_{k:y_k=y} \sum_{l:z_l=z} P(\Lambda_{kl}, x_n = x | Y, Z) \\
\hat{e}_d(x,y) &= \sum_{k:y_k=y} \sum_{l=0}^{L} P(\Lambda_{kl}, x_n = x | Y, Z) \\
\hat{e}_i(x,z) &= \sum_{k=0}^{K} \sum_{l:z_l=z} P(\Lambda_{kl}, x_n = x | Y, Z)
\end{aligned}
$$

Decision theory (“optimal accuracy”).

- Decision theory: maximise expected “reward”, making use of the posterior distribution
- Overlap score: an objective function (i.e. reward) that compares predicted alignment α with true alignment α'
  - Overlap score is $|\alpha \cap \alpha'|$, where an alignment is viewed as a set of match co-ords $\alpha = \{(k_1, l_1), (k_2, l_2) \ldots\}$
  - Several other good objective functions (e.g. “Cline shift score”); overlap is simpler, albeit less realistic
    - NB also $\delta(\alpha = \alpha')$ which only rewards perfect alignments, yielding a multiplicative, Viterbi-like recursion
  - Example criteria: how good is alignment for structure prediction? homology detection? benchmark of choice?
    - e.g. PROBCONS (Batzoglou *et al*) uses the sum-of-pairs score, same as the BAliBASE benchmark

- Posterior expectation of overlap score for an alignment (NB only match states have $\Delta y(x) \Delta z(x) \neq 0$)

$$\begin{aligned} A[\alpha] &= \sum_{(k,l)\in\alpha} P(\text{match}, k, l) \\ P(\text{match}, k, l) &= \sum_{x} \Delta y(x) \Delta z(x) P(\Lambda_{kl}, x_n = x) \end{aligned}$$

- Optimal accuracy recursion: let α_{kl} be any alignment up to (k, l), so $k' \leq k$ and $l' \leq l$ for all $(k', l') \in \alpha$. Then

$$\begin{aligned} O_{kl} &= \max_{\alpha_{kl}} A\left[\alpha_{kl}\right] \\ &= \begin{cases} \max \begin{pmatrix} O_{k-1,l-1} + P(\text{match}, k, l), \\ O_{k,l-1}, \\ O_{k-1,l} \end{pmatrix} & \text{if } k, l \geq 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

- Optimal alignment α recovered by traceback from O_{KL}.

- Dynamic programming algorithms whose finite state automata are almost or exactly Pair HMMs
    - Needleman-Wunsch; Smith-Waterman; Gotoh; Altschul, Proteins 1998
    - General implementations: DART library (C++), Exonerate (C), HMMoC (Java/C++), ...

- Can readily extend the Pair HMM to a multi-sequence HMM for multiple sequence alignment
  - Arbitrary number N of output sequences $Y^{(1)}, Y^{(2)}, Y^{(3)} \ldots Y^{(N)}$ of lengths $L_1 \ldots L_N$ (see e.g. Holmes 2003)
  - Dynamic programming time/memory complexity is $O(\prod_n L_n)$—not cheap
  - Ultimately, would like to structure $\Delta Y^{(n)}(x)$, $t(x, x')$ and $e(x, y^{(1)} \ldots y^{(N)})$ according to some underlying phylogenetic tree
    - The DP algorithms can also be tree structured, c.f. “progressive alignment”
  - For now, we ignore phylogenetic structure of indels (Δ,t) and concentrate on substitution model (e)

- Initial, simplistic, restrictive concept of Evolutionary HMM, lacking a good gap model:
  - **A single-sequence HMM that emits phylogenetically-correlated multiple alignment columns, instead of single characters.**
  - Follows e.g. Goldman, Thorne & Jones, 1996 (*"Using evolutionary trees in protein secondary structure prediction..."*)
  - For now, gaps will be glossed over heuristically (disallowed/treated as wildcards/excessively gappy columns discarded/etc.)
- For a tree with N leaves, the emitted symbol alphabet is Ω^N where Ω is the single-character alphabet

- Now the emission function $e(x, \mathbf{y})$ for $\mathbf{y} \in \Omega^N$ is expensive (and unnecessary) to tabulate
    - We can implement it as an instance of pruning instead: DP within DP
    - Assume an underlying continuous-time discrete-state Markov chain with rate matrix $\mathbf{R}^{(x)}$ and initial distribution $\pi^{(x)}$
    - Let $e(x, \mathbf{y})$ be probability of observing characters $\mathbf{y}$ at (leaf) nodes of a phylogenetic tree, with this process
        - Tree, like alignment, will be specified as an input to our DP recursions
- The counts $\hat{t}(x, x')$ are still relevant, but $\hat{e}(x, \mathbf{y})$ are used to accumulate EM update counts for $\mathbf{R}^{(x)}$
    - Can update t's and $\mathbf{R}^{(x)}$'s simultaneously or asynchronously; c.f. Neal and Hinton

- Many applications in genomic biology
    - Annotation of multiple alignments of DNA, RNA or protein sequences
    - Isochores; gene prediction; DNA-protein binding site modeling and analysis; protein transmembrane structure, signal peptide, domain profiles... etc.

- Implementation: `xrate` (distributed with the DART library)
  - S-expressions for the underlying alignment grammar
    - Here "alignment grammar" means the alphabet Ω, the HMM transition matrix t, the Δ's, the $\mathbf{R}$'s and the π's
    - More generally, can use stochastic context-free grammars as well as HMMs, & states can emit several co-evolving columns
  - Stockholm format for alignment, tree, Viterbi annotation, likelihoods and posterior probabilities
    - Allows internal nodes as well as leaves to be specified
  - By default, gaps in the multiple alignment are treated as wildcards (unobserved character is summed over)
    - A smarter handling of gaps soon leads us to indel rate models and so-called **"Statistical Alignment"**
  - Log messages (type `xrate -help` and `xrate -loghelp`)
  - Command-line options (type `xrate -help`)

- HMMs model $P(Y) = \sum_X P(X, Y)$ (*generative* modeling). ML training maximizes this probability.
- Intuitively, since we are interested in predicting X correctly, it may make more sense to model conditional probability $P(X|Y)$ (*discriminative* modeling)
- Consider the conditional likelihood for an HMM, expressed in terms of the *feature vector* $\{u, f\}$ implied by X:

$$\log P(X|Y) = \frac{1}{P(Y)} \exp \left(\sum_{i,j} u(i,j) \log t(i,j) + \sum_{i,k} f(i,k) \log e(i,k) \right)$$

where $P(Y)$ is computed by the Forward algorithm.

- We can write down a likelihood $P(X|Y)$ for a similarly trellis-structured graphical model as follows

$$P(X|Y) = \frac{1}{Z} \exp\left(\sum_{i,j} u(i,j)a(i,j) + \sum_{i,k} f(i,k)b(i,k) \right)$$

where Z is a partition function (computed by a Forward-like sum-product algorithm)

$$Z = \sum_{X'} \exp\left(\sum_{i,j} u_{X'}(i,j)a(i,j) + \sum_{i,k} f_{X'}(i,k)b(i,k) \right)$$

For equivalence with the HMM, set $a(i,j) = \log t(i,j)$ and $b(i,k) = \log e(i,k)$.

- A *linear-chain conditional random field* is essentially such a trellis-structured model, lacking the normalization constraints on the parameters $\theta = \{a, b\}$ that are implicit in the generative HMM
    - Sutton & McCallum, "An Introduction to Conditional Random Fields for Relational Learning"
    - Lafferty, McCallum & Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data"
- To avoid overfitting, and in place of the normalization constraints, it is useful to add a *regularization* term, e.g. a Gaussian prior on $a()$ and $b()$ with variance σ^2, penalizing large weights. Then the function to be optimized is

$$\ell(\theta) = \log P(X|Y) - \sum_{i,j} \frac{a(i,j)^2}{2\sigma^2} + \sum_{i,k} \frac{b(i,k)^2}{2\sigma^2}$$

- Parameter estimation proceeds by **numerical optimization** of $\ell(\theta)$, typically using quasi-Newton algorithms
    - e.g. the BFGS algorithm: Dimitri P. Bertsekas. Nonlinear Programming. Athena Scientific, 2nd edition, 1999
- The partial derivatives $\frac{\partial \ell}{\partial a(i,j)}$ and $\frac{\partial \ell}{\partial b(i,k)}$ are computed by direct analogy to $\hat{t}(i,j)$ and $\hat{u}(i,k)$ in Baum-Welch.

- In the absence of normalization constraints, we are free to add more "features" without having to directly account for them by subdividing the state space of the HMM. For example: a run of T's, a palindromic motif, etc. In the generative (HMM) view, all such features must be explicitly identified with a path through the model, so that nothing is counted more than once. In a discriminative framework, it doesn't matter if a residue is counted twice.
- In the trellis factor graph view, functions relating x_i and y_i are $P(x_i|Y)$ rather than $P(y_i|x_i)$
- HMMs and linear CRFs form what Ng and Jordan (2002) call a "generative-discriminative pair".
  - Other such pairs include naive Bayes *vs* logistic regression.

Summary

- HMMs