HGEN 48600/STAT 35450: Fundamentals of Computational Biology

Lecture 10 — February 4

Lecturer: John Novembre *Scribe: Chris Stamper*

Note: These lecture notes are still rough, and have only have been mildly proofread.

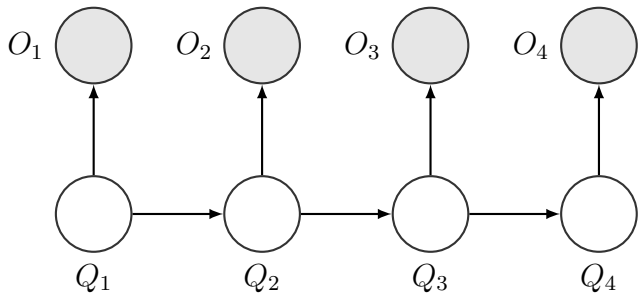
10.1 Hidden Markov Models

10.1.1 Hidden Markov Models Introduction

Hidden Markov Models have a variety of applications and are widley used in many disparate fields. One common application is their use in speech recognition. (Please see Lawrence Rabiner's 'A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition')

10.1.2 Uses of Hidden Markov Models

Hidden Markov Models are perfect for an underlying Markov Model with noisy data.



Meaning that if one conditioned on Q_3 the observation would only depend on it and nothing else.

10.1.3 Basic Structure of Hidden Markov Models

The basic structure of a Hidden Markov Model is:

- A Transition Probability Matrix: $(A = \{a_{ij}, i = 1, ...N, j = 1...N\})$ with

$$a_i j = P(q_t + 1 = S_j | q_t = S_i), \quad 1 \leq i, j \leq N, 1 \leq t \leq T - 1. \tag{10.1}$$

- An Emission Probability Matrix: B_{NXM} is a matrix with all $b_j(i)$

$$P(O_t = i | Q_t = j) = b_j(i), \quad 1 \leq j \leq N, 1 \leq i \leq M, 1 \leq t \leq T. \tag{10.2}$$

- An Initial State Distribution: $\pi = \{\pi_i, i = 1..., N\}$ or with

$$\pi_i = P(q_1 = S_i), \quad 1 \leq i \leq N. \tag{10.3}$$

Which means there are three main problems for Hidden Markov Models.

1. $P(O|\lambda)$: How do we compute efficiently?
2. Given O and λ, what is the most probable sequence, Q.
3. How can we estimate λ? AKA How can we find a $rp_{max}\lambda P(O|\lambda)$?

10.2 Algorithms for solving the problmes of Hidden Markov Models

10.2.1 Forward Algorithm

For problem 1 of HMMs we need a way to solve

$$P(O|\lambda) = \sum_Q P(O, Q|\lambda)$$

Which we can accomplish by using the forward algorithm.

- Initialize

$$\alpha_1(i) = \pi_i b_i(O_1) = P(O, Q_1) \tag{10.4}$$

- Induction

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i)a_{ij}]b_j(O_{t+1}), \quad 1 \leq t \leq T-1 \ \textit{and} \ 1 \leq j \leq N \tag{10.5}$$

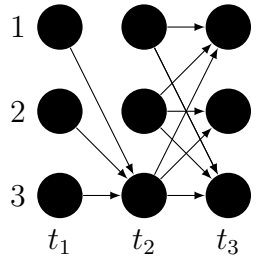
- Termination

$$P(O|\lambda) = \sum_j \alpha_T(j) \tag{10.6}$$

10.2.2 Viterbi Algorithm

For Problem 2 we need a way for the argmax of $P(O, Q|\lambda)$. In other words we need a way to consider all possible Q and find the Q that is maximal.

To visualize this we can imagine a lattice graph:



In which we need to take the maximum of the three paths coming from some i at time 2 and going to some j at time 3.

To do this we can make use of the Viterbi Algortithm.
In which we have:

- An array defined as: $\psi_{t+1}j = argmax_i[\delta_t(i)a_{ij}], \quad 1 \leq j \leq N, 1 \leq t \leq T-1$
- An auxillary variable, δ: $\delta_t(i) = max_{q_1 q_2 ... q_{t-1}} P\{q_1, q_2, ...q_{t-1} = i, O_1, O_2, ..., O_{t-1}|\lambda\}$

Which we can use in the algortihm.

- Initialize

$$\begin{aligned} \delta_1(i) &\leftarrow \pi_i b_i(O_1), \quad 1 \leq i \leq N \\ \psi_1(i) &\leftarrow 0, \quad 1 \leq i \leq N \\ t &\leftarrow 1 \end{aligned} \tag{10.7}$$

- Repeat

$$\begin{aligned} \psi_{t+1}(j) &\leftarrow argmax_i[\delta_t(i)a_(ij)], \quad 1 \leq j \leq N \\ \delta_{t+1}(j) &\leftarrow \delta_t(\psi_{t+1}(j))a_{\psi_{t+1}(j),j}b_j(O_{t+1}), \quad 1 \leq j \leq N \\ t &\leftarrow t+1 \end{aligned} \tag{10.8}$$

- Until

$$\begin{aligned} t &= T \\ P^* &\leftarrow max_{1 \leq i \leq N}[\delta_T(i)] \\ i_T^* &\leftarrow argmax_{1 \leq i \leq N}[\delta_T(i)] \\ &\text{State Sequence Backtracking:} \\ q_T^* &\leftarrow S_{i_T^*} \\ t &\leftarrow T \end{aligned} \tag{10.9}$$

- Repeat

$$\begin{aligned} i_{t-1}^* &\leftarrow \psi_t(i_t^*) \\ q_{t-1}^* &\leftarrow S_{i_{t-1}^*} \\ t &\leftarrow t-1 \\ &\text{until} \\ t &= 1 \\ Q^* &\leftarrow q_1^*, ...q_T^* \end{aligned} \tag{10.10}$$

10.2.3 Forward-Backward Algorithm

To solve the final problem, efficiently learning the parameters of a HMM, we can make use of a backward recursive procedure.
Our goal is that we want the backward variable, $B_t(i)$, or:
$B_t(i) = P(O_{t+1}, ..., O_T|Q_t = j, \lambda)$

such that

$\alpha_t(j)B_t(j) = P(O_1, ..., O_T|Q_t = j, \lambda) \propto P(Q_t = j|O, \lambda)$
and

$P(Q_t = j|O, \lambda) = \frac{\alpha_t(j)B_t(j)}{\sum_i \alpha_t(i)B_t(i)} = \lambda_t(j)$

To compute this we can use the Backwards Algorithm, AKA the Forward-Backward Algorithm:

- Initialize

$$B_T(i) = 1 \tag{10.11}$$

- Induction

$$B_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_t) B_t(j), \quad t = T-1, T-2, ..., 1 \text{ and } 1 \leq i \leq N \tag{10.12}$$

- Continue unitl

$$P(Q_t = j|O, \lambda) = \frac{\alpha_t(j) B_t(j)}{\sum_i \alpha_t(i) B_t(i)} = \lambda_t(j) \tag{10.13}$$