Hidden Markov Models
The three basic HMM problems
(note: change in notation)
Mitch Marcus
CSE 391

Parameters of an HMM
- **States**: A set of states \( S = s_1, \ldots, s_n \)
- **Transition probabilities**: \( A = a_{1,1}, a_{1,2}, \ldots, a_{n,n} \)
  Each \( a_{i,j} \) represents the probability of transitioning from state \( s_i \) to \( s_j \).
- **Emission probabilities**: a set \( B \) of functions of the form \( b_i(o_t) \) which is the probability of observation \( o_t \) being emitted by \( s_i \)
- **Initial state distribution**: \( \pi_i \) is the probability that \( s_i \) is a start state

The Three Basic HMM Problems
- **Problem 1 (Evaluation)**: Given the observation sequence \( O = o_1, \ldots, o_T \) and an HMM model \( \lambda = (A, B, \pi) \), how do we compute the probability of \( O \) given the model?
- **Problem 2 (Decoding)**: Given the observation sequence \( O = o_1, \ldots, o_T \) and an HMM model \( \lambda = (A, B, \pi) \), how do we find the state sequence that best explains the observations?
- **Problem 3 (Learning)**: How do we adjust the model parameters \( \lambda = (A, B, \pi) \), to maximize \( P(O | \lambda) \)?

Problem 1: Probability of an Observation Sequence
- Q: What is \( P(O | \lambda) \)?
- A: the sum of the probabilities of all possible state sequences in the HMM.
  - The probability of each state sequence is itself the product of the state transitions and emit probabilities
  - Naive computation is very expensive. Given \( T \) observations and \( N \) states, there are \( N^T \) possible state sequences.
    - (for \( T=10 \) and \( N=10 \), 10 billion different paths!!)
- Solution: linear time dynamic programming!

Forward Probabilities: \( \alpha \)
- For a given HMM \( \lambda \), given that the state is \( i \) at time \( t \) (with change of notation: some arbitrary time), what is the probability that the partial observation \( o_1 \ldots o_t \) has been generated?
  \[ \alpha_t(i) = P(o_1 \ldots o_t, q_t = s_i | \lambda) \]
- Forward algorithm computes \( \alpha_t(i) \) \( 1 \leq i \leq N \), \( 1 \leq t \leq T \) in time \( O(N^2T) \) using the trellis
Forward Algorithm: Induction step

\[ \alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t) \]

Forward Algorithm

- Initialization:
  \[ \alpha_t(i) = \pi b_i(o_1), \quad 1 \leq i \leq N \]

- Induction:
  \[ \alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t), \quad 2 \leq t \leq T, \quad 1 \leq j \leq N \]

- Termination:
  \[ P(O | \lambda) = \sum_{t=1}^{T} \alpha_T(i) \]

Forward Algorithm Complexity

- Naïve approach requires exponential time to evaluate all \( N^T \) state sequences
- Forward algorithm using dynamic programming takes \( O(N^2 T) \) computations

Backward Probabilities: \( \beta \)

- For a given HMM \( \lambda \), given that the state is \( i \) at time \( t \), what is the probability that the partial observation \( o_{t+1} \ldots o_T \) will be generated?

\[ \beta_t(i) = P(o_{t+1} \ldots o_T | q_t = s_i, \lambda) \]

- Analogous to forward probability, just in the other direction: 
  Backward algorithm computes \( \beta_t(i) \) \( 1 \leq i \leq N, 1 \leq t \leq T \) in time \( O(N^2 T) \) using the trellis

Backward Algorithm

- Initialization:
  \[ \beta_t(i) = 1, \quad 1 \leq i \leq N \]

- Induction (Backward):
  \[ \beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad T-1 \geq t \geq 1, 1 \leq i \leq N \]

- Termination:
  \[ P(O | \lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i) \]
Problem 2: Decoding

- The Forward algorithm gives the sum of all paths through an HMM efficiently.
- Here, we want to find the highest probability path.
- We want to find the state sequence $Q = q_1 \ldots q_T$ such that

$$Q = \arg \max_{Q'} P(Q' | O, \lambda)$$

Viterbi Algorithm

- Just like the forward algorithm, but instead of summing over transitions from incoming states, compute the maximum.

Forward:

$$a_t(j) = \sum_{i=1}^{N} a_{t-1}(i) a_y(i) b_j(o_i)$$

Viterbi Recursion:

$$\delta_t(j) = \max_{i \in \mathcal{N}} \delta_{t-1}(i) a_y(i) b_j(o_i)$$

Not quite what we want....

- Viterbi recursion computes the maximum probability path to state $j$ at time $t$ given that the partial observation $o_1 \ldots o_t$ has been generated.
- But we want the path itself that gives the maximum probability.
- Solution:
  1. Keep backpointers
  2. Find $\arg \max \delta_t(j)$
  3. Chase backpointers from state $j$ at time $T$ to find state sequence (backwards)

Core Idea of Viterbi Algorithm

Viterbi Algorithm

- Initialization: $\delta(0) = \pi b_j(o_1) \quad 1 \leq i \leq N$
- Induction:

$$\delta_t(j) = \max_{i \in \mathcal{N}} \delta_{t-1}(i) a_y(i) b_j(o_i) \quad 2 \leq t \leq T, 1 \leq j \leq N$$

$$\psi_t(j) = \max_{i \in \mathcal{N}} \delta_{t-1}(i) a_y(i) \quad (\text{Backpointers})$$

- Termination: $q_T^* = \arg \max \delta_T(j) \quad (\text{Final state!})$
- Backpointer path: $q_t^* = \psi_T(q_{t+1}^*) \quad t = T-1, \ldots, 1$

Problem 3: Learning

- Up to now we’ve assumed that we know the underlying model $\lambda = (A, B, \pi)$.
- Often these parameters are estimated on annotated training data, but:
  - Annotation is often difficult and/or expensive
  - Training data is different from the current data
- We want to maximize the parameters with respect to the current data, i.e., we’re looking for a model $\lambda'$, such that $\lambda' = \arg \max_{\lambda} P(O | \lambda)$
Problem 3: Learning (If Time Allows…)

- Unfortunately, there is no known way to analytically find a **global** maximum, i.e., a model \( \hat{\lambda} = \arg \max_{\lambda} P(O | \lambda) \)
- But it is possible to find a **local** maximum
- Given an initial model \( \lambda \), we can always find a model \( \hat{\lambda} \), such that \( P(O | \hat{\lambda}) \geq P(O | \lambda) \)

Re-estimating Transition Probabilities: Step 1

- The intuition behind the re-estimation equation for transition probabilities is
  \[
  \hat{a}_{ij} = \frac{\text{expected number of transitions from state } s_i \text{ to state } s_j}{\text{expected number of transitions from state } s_i}
  \]
- Formally:
  \[
  \hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{j'} \xi_t(i, j')}
  \]
Re-estimating Transition Probabilities

- Defining $\gamma_i(t) = \sum_j \xi(i, j)$
  As the probability of being in state $s_i$ given the complete observation $O$

- We can say:
  $$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi(i, j)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

Re-estimating Initial State Probabilities

- Initial state distribution: $\pi_i$ is the probability that $s_i$ is a start state

- Re-estimation is easy:
  $$\hat{\pi}_i = \gamma_i(1)$$

Re-estimation of Emission Probabilities

- Emission probabilities are re-estimated as
  $$\hat{b}_i(k) = \frac{\sum \xi(o, v_k) \gamma_i(t)}{\sum \gamma_i(t)}$$

- Formally:
  $$\hat{b}_i(k) = \frac{\sum \delta(o, v_k) \gamma_i(t)}{\sum \gamma_i(t)}$$

  where $\delta(o, v_k) = 1$, if $o = v_k$, and 0 otherwise

- Note that $\delta$ here is the Kronecker delta function and is not related to the $\delta$ in the discussion of the Viterbi algorithm!!

The Updated Model

- Coming from $\lambda = (A, B, \pi)$ we get to
  $\lambda' = (\hat{A}, \hat{B}, \hat{\pi})$ by the following update rules:

  $$\hat{a}_{i,j} = \frac{\sum \xi(i, j)}{\sum \gamma_i(t)}$$

  $$\hat{b}_i(k) = \frac{\sum \delta(o, v_k) \gamma_i(t)}{\sum \gamma_i(t)}$$

  $$\hat{\pi}_i = \gamma_i(1)$$

Expectation Maximization

- The forward-backward algorithm is an instance of the more general EM algorithm
  - The E Step: Compute the forward and backward probabilities for a given model
  - The M Step: Re-estimate the model parameters