Hidden Markov Models

COSC 6336 Natural Language Processing

Spring 2018

With adapted material from Yang Liu, who borrowed material from Tanja Schultz and Dan Jurafsky

The Three Basic Problems for HMMs

- Problem 1 (**Evaluation**): Given the observation sequence $O=(o_1 o_2 \ldots o_T)$, and an HMM model $\Phi = (A,B)$, how do we efficiently compute $P(O|\Phi)$, the probability of the observation sequence, given the model
- Problem 2 (**Decoding**): Given the observation sequence $O=(o_1 o_2 \ldots o_T)$, and an HMM model $\Phi = (A,B)$, how do we choose a corresponding state sequence $Q=(q_1 q_2 \ldots q_T)$ that is optimal in some sense (i.e., best explains the observations)
- Problem 3 (**Learning**): How do we adjust the model parameters $\Phi = (A,B)$ to maximize $P(O|\Phi)$?

The Learning Problem

Learning: Given an observation sequence O and the set of possible states in the HMM, learn the HMM parameters A and B.

- **Baum-Welch = Forward-Backward Algorithm** (Baum 1972)
- Is a special case of the EM or Expectation-Maximization algorithm (Dempster, Laird, Rubin)
- The algorithm will let us train the transition probabilities $A=\{a_{ij}\}$ and the emission probabilities $B=\{b_i(o_t)\}$ of the HMM

Starting out with Observable Markov Models

- How to train?
- Run the model on the observation sequence O.
- Since it's not hidden, we know which states we went through, hence which transitions and observations were used.
- Given that information, training:
  - B = {$b_k(o_t)$}: Since every state can only generate one observation symbol, observation likelihoods B are all 1.0
  - A = {a_{ij}}:

$$a_{ij} = \frac{C(i \to j)}{\sum_{q \in Q} C(i \to q)}$$

Extending Intuition to HMMs

- For HMMs, cannot compute these counts directly from observed sequences
- Baum-Welch (forward-backward) intuitions:
  - Iteratively estimate the counts
    - Start with an estimate for a_{ij} and b_k, iteratively improve the estimates
  - Get estimated probabilities by:
    - computing the forward probability for an observation
    - dividing that probability mass among all the different paths that contributed to this forward probability
  - Two related probabilities: the forward probability and the backward probability

Recall: The Forward Algorithm

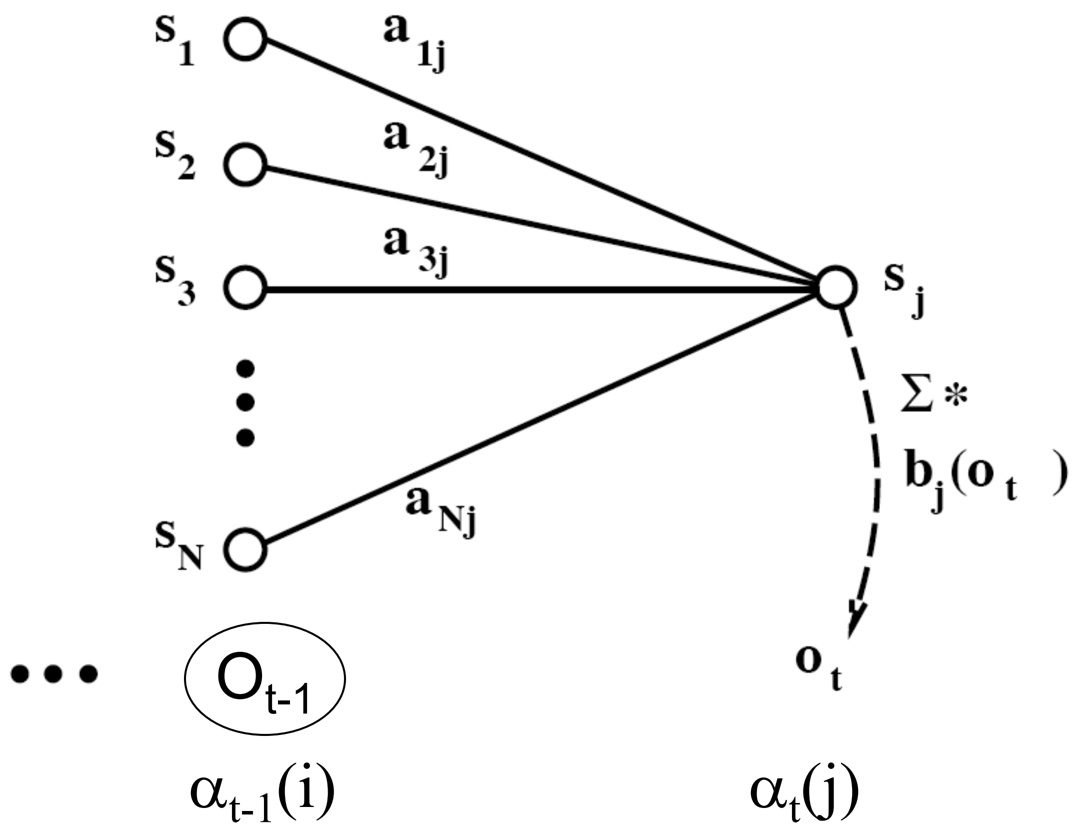
- The Idea: Fold these exponential paths into a simple trellis, so that all possible paths will remerge into N states at every time slice.
- We define the *forward probability* as follows: $\alpha_t(i) = P(o_0 o_1 \cdots o_t, q_t = i|\Phi)$
- this is the probability that the HMM Φ is in state i at time t having generated partial observation O_1^t.
- We compute it by induction:
  - Initialization: $\alpha_1(i) = \pi_i P(o_1|q_i), 1 \leq i \leq N$
  - (equivalently: $\alpha_1(i) = \pi_i b_i(o_1), 1 \leq i \leq N$
  - Induction:

$$\alpha_t(j) = [\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}] b_j(o_t),$$
$$2 \leq t \leq T, 1 \leq j \leq N \quad (3)$$

  - Termination: $P(O|\Phi) = \sum_{i=1}^{N} \alpha_T(i)$

The inductive step, from Rabiner and Juang

- Computation of $\alpha_t(j)$ by summing all previous values $\alpha_{t-1}(i)$ for all *i*



The Backward algorithm

- We compute backward prob by induction:

1. **Initialization:**

$$\beta_T(i) = a_{i,F}, \quad 1 \leq i \leq N$$

2. **Recursion** (again since states 0 and q_F are non-emitting):

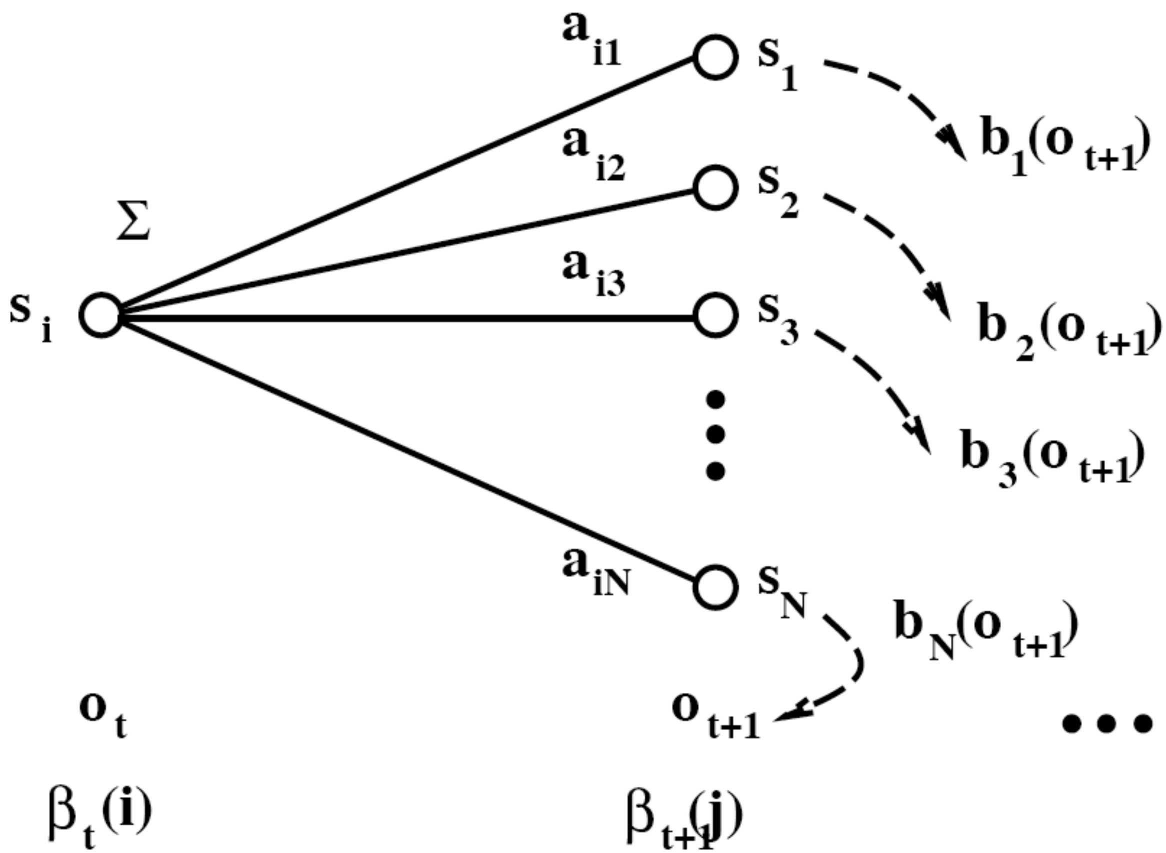
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j), \quad 1 \leq i \leq N, 1 \leq t < T$$

3. **Termination:**

$$P(O|\lambda) = \alpha_T(q_F) = \beta_1(0) = \sum_{j=1}^{N} a_{0j}\, b_j(o_1)\, \beta_1(j)$$

Inductive Step of the Backward Algorithm (Figure after Rabiner and Juang)

- Computation of $\beta_t(i)$ by weighted sum of all successive values β_{t+1}



Extending Intuition to HMMs

- For HMMs, cannot compute these counts directly from observed sequences
- Baum-Welch (forward-backward) intuitions:
  - Iteratively estimate the counts
    - Start with an estimate for a_{ij} and b_k, iteratively improve the estimates
  - Get estimated probabilities by:
    - computing the forward probability for an observation
    - dividing that probability mass among all the different paths that contributed to this forward probability
  - Two related probabilities: the forward probability and the backward probability

Intuition for Re-estimation of a_{ij}

- We will estimate $\hat{a}_{ij}$ via this intuition:

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

- Numerator intuition:
  - Assume we had some estimate of probability that a given transition *i*→*j* was taken at time *t* in observation sequence.
  - If we knew this probability for each time *t*, we could sum over all *t* to get expected value (count) for *i*→*j*.

Re-estimation of a_{ij}

- Let γ_t be the probability of being in state *i* at time *t* and state *j* at time *t+1*, given $O_{1..T}$ and model Φ:

$$\gamma_t(i,j) = P(q_t = i, q_{t+1} = j \mid O, \Phi)$$

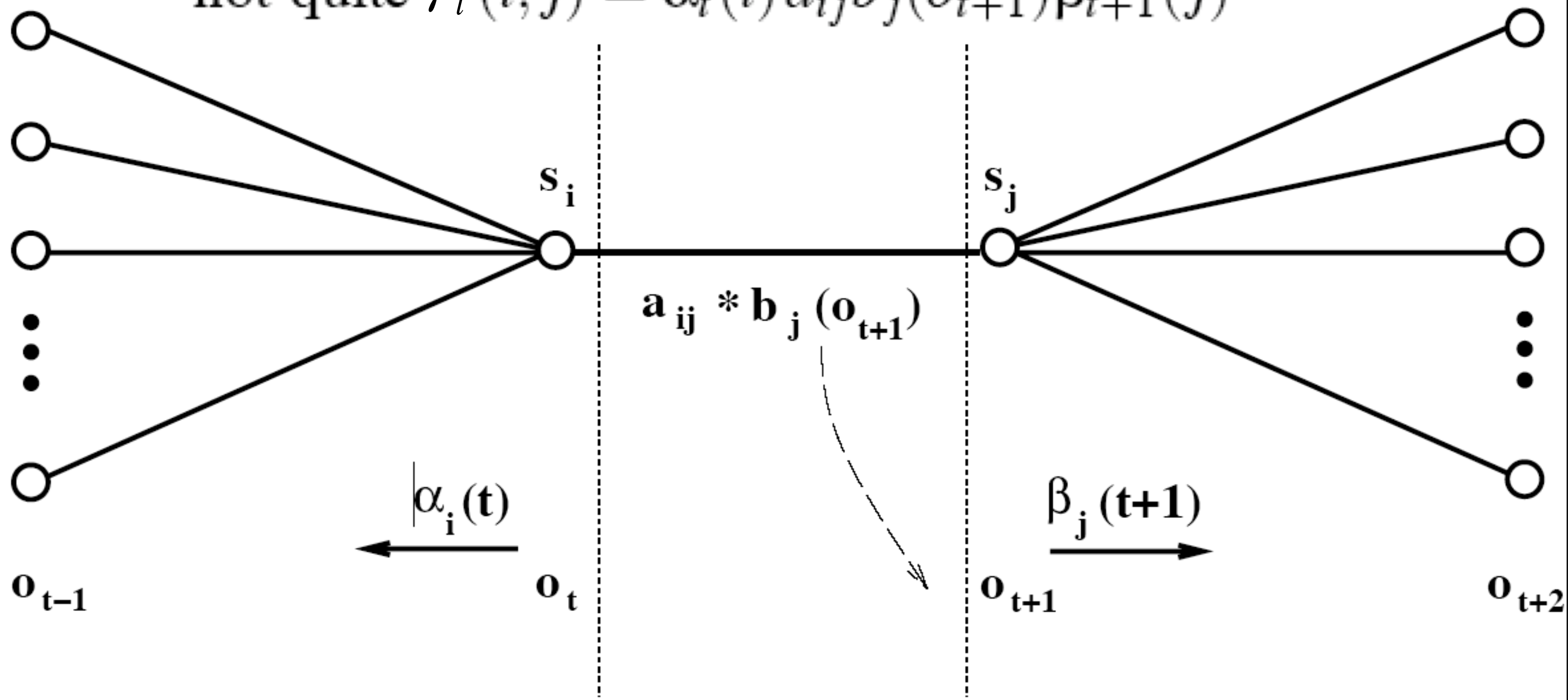
- We can compute γ from not-quite-γ, which is:

$$not\_quite\_\gamma_t(i,j) = P(q_t = i, q_{t+1} = j, O \mid \Phi)$$

Computing not-quite-γ

The four components of $P(q_t = i, q_{t+1} = j, O \mid \Phi) : \alpha, \beta, a_{ij}$ and $b_j(o_t)$

$$\text{not-quite-}\gamma_t(i,j) = \alpha_t(i)\, a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$



From not-quite-γ to γ

$$\text{not-quite-}\gamma_t(i,j) = \alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j) \tag{8}$$

$$\gamma_t(i,j) = P(q_t = i, q_{t+1} = j | O, \Phi) \tag{9}$$

$$\textit{not-quite-}\gamma_t(i,j) = P(q_t = i, q_{t+1} = j, O | \Phi) \tag{10}$$

$$P(X|O,\Phi) = \frac{P(X,O|\Phi)}{P(O|\Phi)} \tag{11}$$

$$P(O|\Phi) = \alpha_T(N) = \beta_T(1) = \sum_{j=1}^{N} \alpha_t(j)\beta_t(j) \tag{12}$$

$$\gamma_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(N)} \tag{13}$$

From γ to a_{ij}

- $\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$
- The expected number of transitions from state i to state j is the sum over all t of γ.
- The total expected number of transitions out of state i is the sum over all transitions out of state i.
- Final formula for reestimated a_{ij}:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \gamma_t(i,j)} \quad (14)$$

Re-estimating the Observation Likelihood b

- This is the probability of a given symbol v_k from the observation vocabulary V, given a state j: $\hat{b}_j(v_k)$.

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state j and observing symbol } v_k}{\text{expected number of times in state } j}$$

- For this we will need to know the probability of being in state j at time t, which we will call $\xi_t(j)$ (ξ for **s**tate):

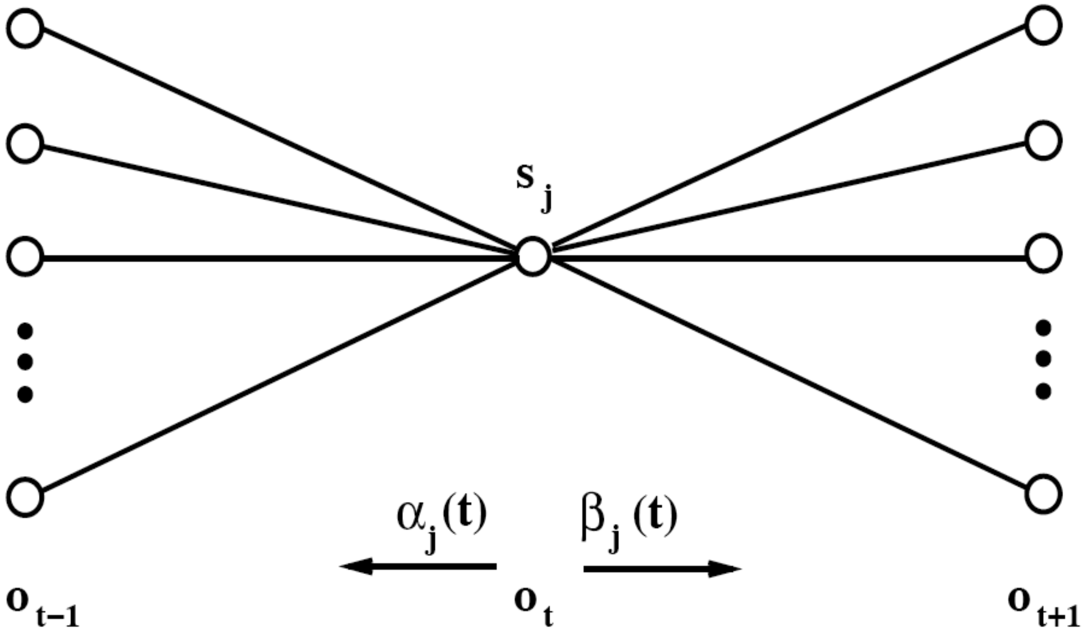
- $\xi_t(j) = P(q_t = j | O, \Phi)$

- We compute this by including the observation sequence in the probability and then normalizing:

- $\xi_t(j) = \frac{P(q_t = j, O | \Phi)}{P(O | \Phi)}$

Computing ξ

Computation of $\xi_j(t)$, the probability of being in state *j* at time *t.*



- $\xi_t(j) = \dfrac{P(q_t = j, O|\Phi)}{P(O|\Phi)}$
- $\xi_t(j) = \dfrac{\alpha_t(j)\beta_t(j)}{P(O|\Phi)}$

Reestimating the observation Likelihood *b*

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state j and observing symbol } v_k}{\text{expected number of times in state } j}$$

- For numerator, sum $\xi_j(t)$ for all *t* in which o_t is symbol v_k:

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t. O_t = v_k}^{T} \xi_j(t)}{\sum_{t=1}^{T} \xi_j(t)}$$

Summary of *A* and *B*

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \gamma_t(i,j)}$$

The ratio between the expected number of transitions from state i to j and the expected number of all transitions from state i

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t. O_t = v_k}^{T} \xi_j(t)}{\sum_{t=1}^{T} \xi_j(t)}$$

The ratio between the expected number of times the observation data emitted from state j is v_k, and the expected number of times any observation is emitted from state j

The Forward-Backward Algorithm

function FORWARD-BACKWARD(*observations* of len T, *output vocabulary* V, *hidden state set* Q) **returns** $HMM=(A,B)$

initialize A and B
iterate until convergence

E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall\, t \text{ and } j$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall\, t,\ i, \text{ and } j$$

M-step

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)} \qquad \hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

return A, B

Summary: Forward-Backward Algorithm

1) Initialize Φ=(A,B,π)
2) Compute α, β, ξ
3) Estimate new Φ' =(A,B,π)
4) Replace Φ with Φ'
5) If not converged go to 2

Embedded Training of HMMs

- The entire procedure:

1. Choose an estimate for *a* and *b*
2. Re-estimate *a* and *b*
3. Repeat until convergence

- How do we get initial estimates for *a* and *b*?
- For *a* we assume that from any state all the possible following states are equiprobable
- For *b* we can use a small hand-labelled training corpus

Summary

- We learned the Baum-Welch algorithm for learning the A and B matrices of an individual HMM
- It doesn't require training data to be labeled at the state level; all you have to know is that an HMM covers a given sequence of observations, and you can learn the optimal A and B parameters for this data by an iterative process.

The Learning Problem: Caveats

- Network structure of HMM is always created by hand
  - no algorithm for double-induction of optimal structure and probabilities has been able to beat simple hand-built structures.
- Baum-Welch only guaranteed to find a local max, rather than global optimum