Recitation 3: Hidden Markov Models (HMMs)

6.864/6.806: Advanced Natural Language Processing

Dylan D. Doblar

March 5, 2021

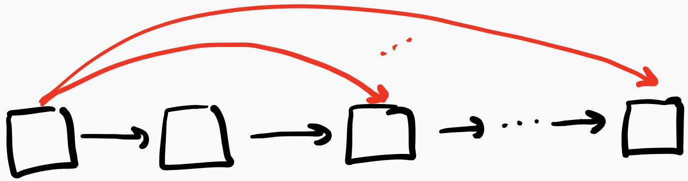
Agenda

1. Sequential modeling
2. Markov chains
3. Hidden Markov Models (HMMs)
4. Baum-Welch
5. Practical advice for Homework 1

Sequential Modeling

Many NLP tasks of interest have sequential structure.

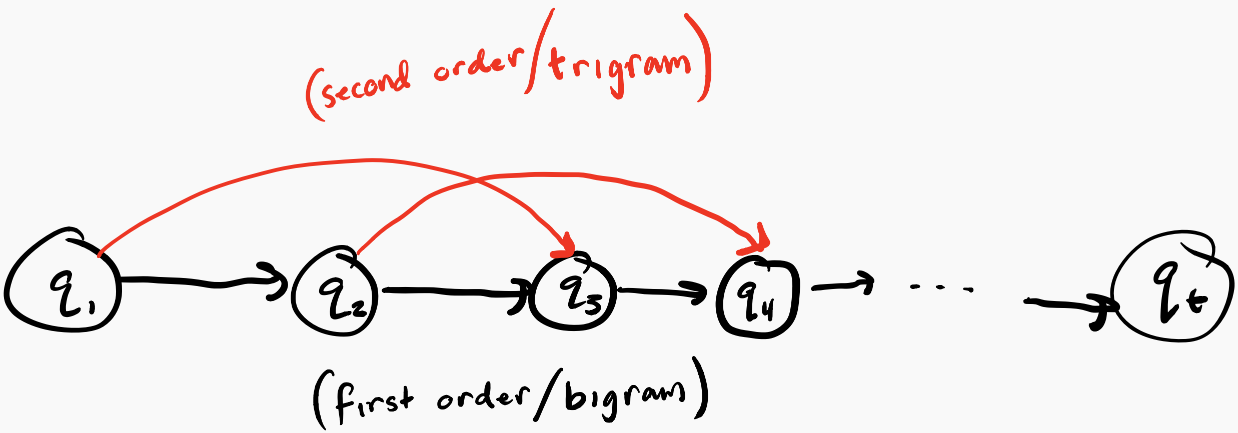
- Part-of-speech (POS) tagging
- Named entity recognition (NER)
- Machine translation
- Speech recognition



Markov Property

The future is independent of the past given the present:
Given a sequence of state variables $q_1, q_2, \ldots, q_t$,

$$\mathbb{P}(q_t | q_1, \ldots, q_{t-1}) = \mathbb{P}(q_t | q_{t-1}).$$



Markov Chains

Markov Property

The future is independent of the past given the present:

Given a sequence of state variables $q_1, q_2, \ldots, q_t$,

$$\mathbb{P}(q_t | q_1, \ldots, q_{t-1}) = \mathbb{P}(q_t | q_{t-1}).$$

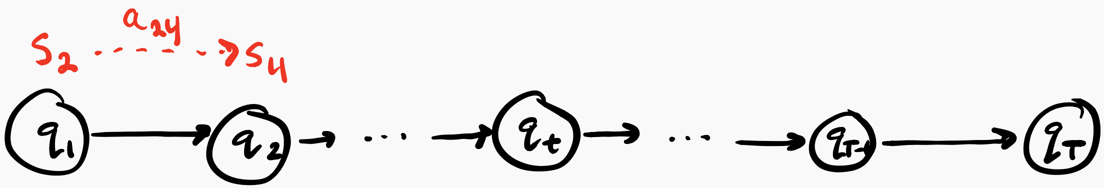
Markov Chain

$$
\begin{aligned}
S &= \{s_1, s_2, \ldots, s_N\} && \text{possible states} \\
A &= \left[a_{ij}\right] \in \mathbb{R}^{N \times N} && \text{state transition probability matrix} \\
\pi &= \left[\pi_i\right] \in \mathbb{R}^{N} && \text{initial state distribution}
\end{aligned}
$$

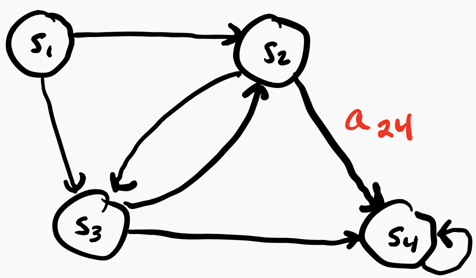
Markov Chains

Two graphical representations of a Markov chain:

Probabilistic graphical model



State transition diagram



Hidden Markov Models

Hidden Markov Models (HMMs)

$$
\begin{aligned}
S &= \{s_1, s_2, \dots, s_N\} && \text{possible states} \\
W &= \{w_1, w_2, \dots, w_V\} && \text{possible observations} \\
A &= \left[a_{ij}\right] \in \mathbb{R}^{N \times N} && \text{state transition probability matrix} \\
B &= \left[b_j(k)\right] \in \mathbb{R}^{N \times V} && \text{observation (emission) probability matrix} \\
\pi &= \left[\pi_i\right] \in \mathbb{R}^{N} && \text{initial state distribution}
\end{aligned}
$$

$a_{ij} = \mathbb{P}(q_{t+1} = s_j | q_t = s_j)$; $b_j(k) = \mathbb{P}(o_t = w_k | q_t = s_j)$; $\pi_i = \mathbb{P}(q_1 = s_i)$

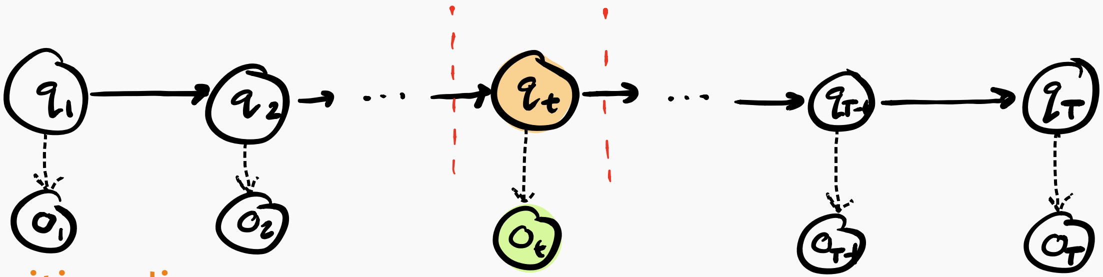
An HMM is fully parameterized by $\lambda = \{A, B, \pi\}$.

Output independence

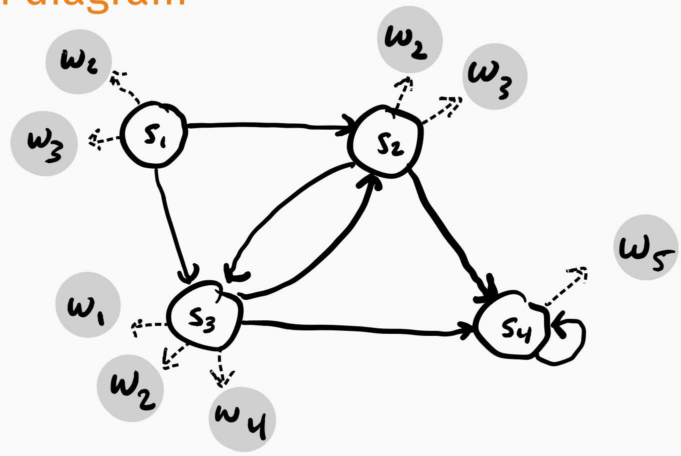
$$\mathbb{P}(o_t | q_1, \dots, q_T, o_1, \dots, o_T) = \mathbb{P}(o_t | q_t)$$

We can incorporate observations into our graphical representations:

Probabilistic graphical model



State transition diagram



What can we do with an HMM?

With parameters $\lambda = \{A, B, \pi\}$ and observations $O = \{o_1, \ldots, o_T\}$:

- Scoring: compute $\mathbb{P}(O|\lambda)$ given observations
  - $\rightarrow$ Forward-backward algorithm
- Matching: optimal state sequence $\{q_1, \ldots, q_T\}$
  - $\rightarrow$ Viterbi algorithm

If we only have observations $O = \{o_1, \ldots, o_T\}$:

- **Training: parameter estimation for** $\lambda = \{A, B, \pi\}$
  - $\rightarrow$ **Baum-Welch estimation**

Interlude: Expectation-Maximization (EM) Algorithm

The EM algorithm is a technique to find maximum-likelihood estimators in latent variable models. Two iterative steps (after random initialization of parameters):

E Estimate latent variables, assuming fixed parameters

M Maximize likelihood of parameters w.r.t. estimated variables

Baum-Welch estimation is a form of EM for HMMs.

Baum-Welch estimation

Intuition

- Start with estimates for A, B, and π
- Compute the probabilities of (1) being in each state s_i and (2) transitioning from each state s_i to every other state s_j, for all time, assuming the estimated parameters are fixed
- Assuming the probabilities of being in each state are fixed, update A, B, and π to maximize likelihood
- Repeat
- Profit

Forward Algorithm

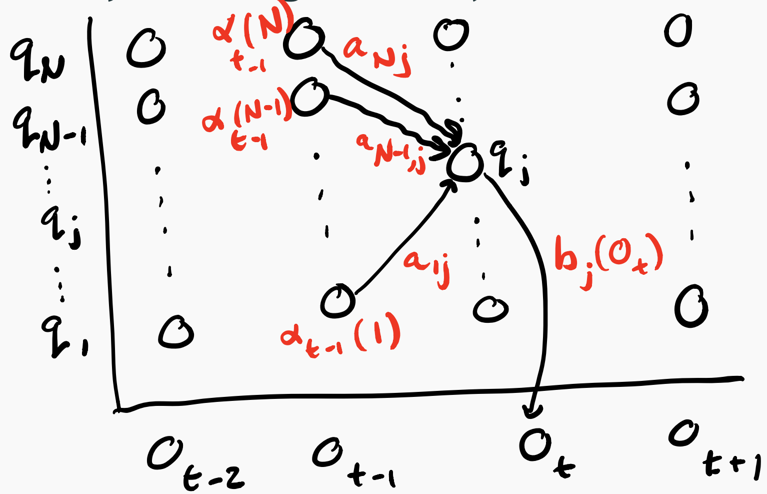
We are interested in $\mathbb{P}(O|\lambda) \triangleq \mathbb{P}(O) = \sum_Q \mathbb{P}(O, Q) = \sum_Q \mathbb{P}(O|Q)\mathbb{P}(Q)$.

Initialize forward variable $\alpha_t(j) = \mathbb{P}(o_1, \ldots, o_t, q_t = j|\lambda)$ with $\alpha_1(j) = \pi_j b_j(o_1)$, then compute

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t) \qquad \forall t \in \{2, \ldots, T\},\ j \in \{1, \ldots, N\}$$

and finally $\mathbb{P}(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$.

$\rightarrow$ $\alpha_t(j)$ is the probability of being in state j after the first t observations.



Backward Algorithm

Similar to the forward algorithm, we want β, the backward probability of observations $t+1$ to T, given that we are in state i at time t:

$$\beta_t(i) = \mathbb{P}(o_{t+1}, \ldots, o_T | q_t = i, \lambda)$$

Initialize $\beta_T(i) = 1 \; \forall i \in \{1, \ldots, N\}$, then recursively compute

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \; \forall i \in \{1, \ldots, N\}, \; t \in \{1, \ldots T-1\}$$

On termination, compute $\mathbb{P}(O|\lambda) = \sum_{j=1}^{N} \pi_j b_j(o_1) \beta_1(j)$.

Forward-Backward Algorithm

How do α and β help compute a_{ij} from just O?

We want to know the probability of transitioning from state i to j.

If we could observe the hidden states q_t, we'd just use frequencies:

$$\hat{a}_{ij} = \frac{\text{num. } s_i \to s_j \text{ transitions}}{\text{num. } s_i \to s_* \text{ transitions}}$$

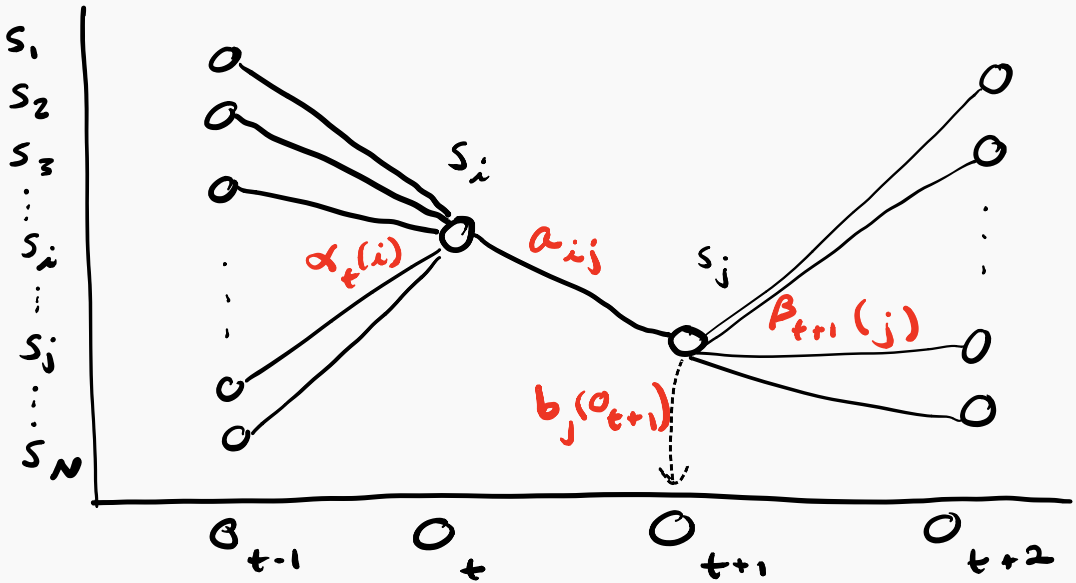
We don't know the numerator and denominator, but can we estimate them? (What if we had estimates for the probability of $s_i \to s_j \; \forall t$ and the probability of $s_i \to s_* \forall t$?)

$$\begin{aligned} \xi_t(i,j) &= \mathbb{P}(q_t = s_i, q_{t+1} = s_j | O, \lambda) \\ &= \frac{\mathbb{P}(q_t = s_i, q_{t+1} = s_j, O | \lambda)}{\mathbb{P}(O|\lambda)} \end{aligned}$$

Forward-Backward Algorithm

$$\xi_t(i,j) = \frac{\mathbb{P}(q_t = s_i, q_{t+1} = s_j, O|\lambda)}{\mathbb{P}(O|\lambda)}$$

How to compute numerator?



Forward-Backward Algorithm

$$\xi_t(i,j) = \frac{\mathbb{P}(q_t = s_i, q_{t+1} = s_j, O|\lambda)}{\mathbb{P}(O|\lambda)}$$

How to compute numerator?

$$\mathbb{P}(q_t = s_i, q_{t+1} = s_j, O|\lambda) = \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

What about the denominator?

$$\mathbb{P}(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) = \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i) = \sum_{i=1}^{N} \alpha_t(i) \beta_t(i)$$

Forward-Backward Algorithm

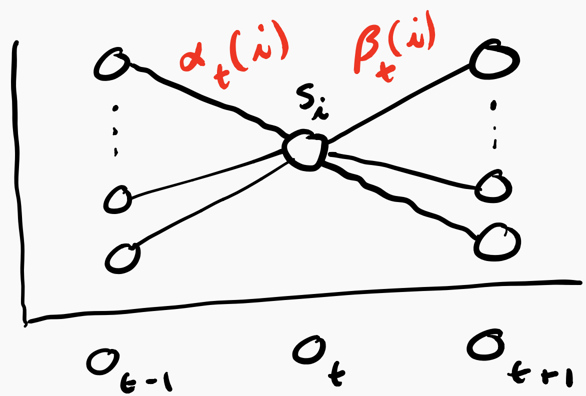
We have the first quantity we wanted:

$$\xi_t(i,j) = \mathbb{P}(q_t = s_i, q_{t+1} = s_j | O, \lambda) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}$$

The second, the probability of being in state s_i at time t is

$$\gamma_t(i) = \mathbb{P}(q_t = s_i | O, \lambda) = \sum_{j=1}^{N} \xi_t(i,j) = \frac{\alpha_t(i)\beta_t(i)}{\mathbb{P}(O|\lambda)}$$

what about $\gamma_T(i)$? (don't have $\xi_T(i,j)$)



Forward-Backward Algorithm

We have the first quantity we wanted:

$$\xi_t(i,j) = \mathbb{P}(q_t = s_i, q_{t+1} = s_j | O, \lambda) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_t(i) \beta_t(i)}$$

The second, the probability of being in state s_i at time t is

$$\gamma_t(i) = \mathbb{P}(q_t = s_i | O, \lambda) = \sum_{j=1}^{N} \xi_t(i,j) = \frac{\alpha_t(i) \beta_t(i)}{\mathbb{P}(O|\lambda)}$$

Expected number of transitions from s_i to s_j is

$$\mathbb{E}[s_i \to s_j] = \sum_{t=1}^{T-1} \xi_t(i,j)$$

Expected number of transitions out of s_i is

$$\mathbb{E}[s_i \to s_*] = \sum_{t=1}^{T-1} \gamma_t(i)$$

Baum-Welch estimation

$$\begin{aligned}\hat{a}_{ij} &= \frac{\text{num. } s_i \to s_j \text{ transitions}}{\text{num. transitions from } s_i} \\ &\approx \frac{\mathbb{E}[s_i \to s_j]}{\mathbb{E}[s_i \to s_*]} \\ &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}\end{aligned}$$

(recall $\xi_t(i,j) = \mathbb{P}(q_t = s_i, q_{t+1} = s_j | O, \lambda)$ and $\gamma_t(j) = \mathbb{P}(q_t = s_j | O, \lambda)$)

Baum-Welch estimation

What about observations? Similar to transitions:

$$\begin{aligned}\hat{b}_j(k) &= \frac{\text{num. } w_k \text{ observations in } s_j}{\text{num. times in } s_j} \\ &\approx \frac{\mathbb{E}[s_j, w_k]}{\mathbb{E}[s_j]} \\ &= \frac{\sum_{t=1; o_t = w_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}\end{aligned}$$

(recall $\gamma_t(j) = \mathbb{P}(q_t = s_j | O, \lambda)$)

To estimate initial state distribution, just take

$$\hat{\pi}_i = \mathbb{E}[q_1 = s_i] = \mathbb{P}(q_1 = s_j | O, \lambda) = \gamma_1(i).$$

Baum-Welch estimation

After random initialization of A, B, and π, iterate E and M steps:

E step:

$$\xi_t(i,j) := \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \qquad \forall t, i, j$$

$$\gamma_t(i) := \sum_{j=1}^{N} \xi_t(i,j) \qquad \forall t, j$$

M step:

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

$$\hat{b}_j(k) := \frac{\sum_{t=1; o_t = w_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \qquad \forall j, k$$

$$\hat{\pi}_i := \gamma_1(i) \qquad \forall i$$

Homework 1 tips

- Dynamic programming
  - Store any values you compute for a_{ij}, $b_j(o_t)$, $\alpha_t(i)$, $\beta_t(i)$, $\xi_t(i,j)$, and $\gamma_t(i)$ in an array for easy access in future iterations (memoization)
- Use vectorized `numpy` operations
  - For `forward`, `backward`, and `forward_backward`, you should only need one for loop (to loop over t)
  - Pay attention to shapes of arrays to make sure they are what you think they are
- Do computation in log-space to avoid numerical issues
  - $\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$
    $\rightarrow \log \alpha_t(j) = \log \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right] + \log b_j(o_t)$
    $= \log \left[\sum_{i=1}^{N} \exp(\log \alpha_{t-1}(i) + \log a_{ij})\right] + \log b_j(o_t)$
    $= \mathit{logsumexp}(\log \alpha_{t-1}(i) + \log a_{ij}) + \log b_j(o_t)$
- Refer to lecture slides and Section A.5 of [JM20] (link on Canvas)

References

[JM20] Daniel Jurafsky and James H Martin.
Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition, 2020.