Markov Assumption

- The Markov assumption states that the probability of the occurrence of an observed token $o_t$ at time $t$ depends only on the occurrence of $o_{t-1}$ at time $t-1$

- Chain rule:
  \[ P(o) = P(o_0, \ldots, o_T) = P(o_0) \prod_{t=1}^{T-1} P(o_t | o_0, \ldots, o_{t-1}) \]

- Markov assumption:
  \[ P(o) = P(o_0, \ldots, o_T) = P(o_0) \prod_{t=1}^{T-1} P(o_t | o_{t-1}) \]

The Trellis

Parameters of an HMM

- States: a set of state nodes $n = n_0, \ldots, n_{N-1}$
- Transition probabilities: $a = a_{0,0}, a_{0,1}, \ldots, a_{N-1,N-1}$ where each $a_{ij}$ represents the probability of transitioning to $n_j$ given that we're coming from state $n_i$
- 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 $n_i$ at time $t$
- Initial state distribution: $\pi_i$ is the probability that $n_i$ is a start state
The Three Basic HMM Problems

- Problem 1: Decoding. Given the observation sequence $o= o_0, o_1, \ldots, o_T$, and an HMM model $\lambda = (a, b, \pi)$, how do we find the state sequence that best explains the observations?

- Problem 2: Evaluation. Given the observation sequence $o= o_0, o_1, \ldots, o_T$ and an HMM model $\lambda = (a, b, \pi)$, how do we compute the probability of $o$ given the model?

- Problem 3: Learning. How do we adjust the model parameters $\lambda = (a, b, \pi)$, so as to maximize $P(o \mid \lambda)$?

Problem 1: Decoding

- For Problem 1, we want to find the path with the highest probability.
- We want to find the state sequence $q = q_0, q_1, \ldots, q_T$, such that $q = \arg \max P(q \mid o, \lambda)$.
- Naïve computation is very expensive. Given $T$ observations and $N$ states, there are $N^T$ possible state sequences.
- Even small HMMs, e.g. $T=10$ and $N=10$, contain 10 billion different paths.
- Solution: use dynamic programming.

Viterbi Algorithm

- Similar to computing the forward probabilities, but instead of summing over transitions from incoming states, compute the maximum.
- Forward:
  $$\alpha_t(j) = \sum_{i=0}^{N-1} \alpha_{t-1}(i)a_{ij}b_j(o_t)$$
- Viterbi Recursion:
  $$\delta_t(j) = \max_{0 \leq \ell < N} \delta_{t-1}(i)a_{ij}b_j(o_t)$$
Viterbi Algorithm

- Initialization: \( \delta_0(j) = \pi_j b_j(o_0) \quad 0 \leq j < N \)
- Induction:
  \[
  \delta_t(j) = \max_{0 \leq i < N} \delta_{t-1}(i) a_{ij} b_j(o_t) 
  \]
  \[
  \psi_t(j) = \arg \max_{0 \leq i < N} \delta_{t-1}(i) a_{ij} \quad 0 < t < T, 0 \leq j < N 
  \]
- Termination: \( p^* = \max_{0 \leq i < N} \delta_{T-1}(i) \quad q^*_{T-1} = \arg \max_{0 \leq i < N} \delta_{T-1}(i) \)
- Reconstruction: \( q_t^* = \psi_{t+1}(q_{t+1}) \quad t = T - 2, \ldots, 0 \)

Problem 2: Probability of an Observation Sequence

- What is \( P(o \mid \lambda) \)?
- The probability of a observation sequence is the sum of the probabilities of all possible state sequences in the HMM, i.e., the sum over all paths that generate \( o \) through an HMM efficiently.
- The solution to Problem 1 (decoding) gives us the max over all paths that generate \( o \) through an HMM efficiently.

Forward Probabilities

- What is the probability that, given an HMM \( \lambda \), at time \( t \) the state is \( i \) and the partial observation \( o_0 \ldots o_t \) has been generated?

  \[
  \alpha_t(i) = P(o_0 \ldots o_t, q_t = n_i \mid \lambda) = P(o_0 \ldots o_t, q_t = n_i \mid \lambda) 
  \]
Forward Algorithm

- Initialization: \( \alpha_0(j) = \pi_j b_j(o_0) \quad 0 \leq j < N \)

- Induction:
  \[
  \alpha_t(j) = \sum_{i=0}^{N-1} \alpha_{t-1}(i) a_{ij} b_j(o_t) \quad 0 < t < T, 0 \leq j < N
  \]

- Termination: \( P(o \mid \lambda) = \sum_{i=0}^{N-1} \alpha_{T-1}(i) \)

Forward Algorithm Complexity

- In the naïve approach to solving problem 1 it takes on the order of \( 2T \times N^T \) computations
- The forward algorithm takes on the order of \( N^2 T \) computations

Backward Probabilities

- Or, we can instead compute right-to-left, analogous to the forward probability, but in the opposite direction
- What is the probability that, given an HMM \( \lambda \) and given the state at time \( t \) is \( i \), the partial observation \( o_{t+1..T} \) is generated?
  \[
  \beta_t(i) = P(o_{t+1..T} \mid q_t = n_i, \lambda) = P(o_{t+1}...o_{T-1} \mid q_t = n_i, \lambda)
  \]

Backward Probabilities
Backward Algorithm

- Initialization: \( \beta_{T-1}(i) = 1, \ 0 \leq i < N \)

- Induction:
  \[
  \beta(i) = \sum_{j=0}^{N-1} a_i b_j(o_i) \beta(i) \quad i = T-2...0, \ 0 \leq i < N
  \]

- Termination:
  \[
  P(o \mid \lambda) = \sum_{i=0}^{N-1} \pi_i \beta_0(i)
  \]

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, which has two drawbacks:
  - Annotation is 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 maximum likelihood model \( \lambda^* \), such that \( \lambda^* = \arg\max_{\lambda} P(o \mid \lambda) \)

Parameter Re-estimation

- Use the forward-backward algorithm (a.k.a. the Baum-Welch algorithm), which is a hill-climbing algorithm
- Using an initial parameter instantiation, the forward-backward algorithm iteratively re-estimates the parameters, and improves the probability that the given observations are generated by the new parameters
Parameter Re-estimation

- Three parameters need to be re-estimated:
  - Initial state distribution: \( \pi_i \)
  - Transition probabilities: \( a_{ij} \)
  - Emission probabilities: \( b_i(o) \)

Re-estimating Transition Probabilities

- Expected number of transitions:
  
  what’s the probability of being in state \( n_i \) at time \( t \) and going to state \( n_j \) given the current model and parameters?

  \[
  \xi_t(i, j) = P(q_t = n_i, q_{t+1} = n_j | o, \lambda)
  \]
Re-estimating Transition Probabilities

- Remember, the intuition behind the re-estimation equation for transition probabilities is
  \[ \hat{a}_{i,j} = \frac{\text{expected number of transitions from state } n_i \text{ to state } n_j}{\text{expected number of transitions from state } n_i} \]

- Formally:
  \[ \hat{a}_{i,j} = \frac{\sum_{t=0}^{T-2} \xi_t(i,j)}{\sum_{j'=0}^{N-1} \sum_{j''=0}^{N-1} \xi_t(i,j'')} \]

Re-estimating Transition Probabilities

- We can rewrite this more neatly as
  \[ \hat{a}_{i,j} = \frac{\sum_{t=0}^{T-2} \xi_t(i,j)}{\sum_{i'=0}^{N-1} \gamma_t(i')} \]

  where we define
  \[ \gamma_t(i) = \sum_{j'=0}^{N-1} \xi_t(i,j') \]

  to be the probability of being in state \( n_i \) at time \( t \), given the complete observation \( o \)

Review of Probabilities

- Forward probability: \( \alpha_t(i) \)
  - The probability of being in state \( n_i \) given the partial observation \( o_0, \ldots, o_t \)

- Backward probability: \( \beta_t(i) \)
  - The probability of being in state \( n_i \) given the partial observation \( o_{t+1}, \ldots, o_T \)

- Transition probability: \( \xi_t(i,j) \)
  - The probability of going from state \( n_i \) to state \( n_j \) given the complete observation \( o_0, \ldots, o_{T-1} \)

- State probability: \( \gamma_t(i) \)
  - The probability of being in state \( n_i \) given the complete observation \( o_0, \ldots, o_{T-1} \)

Re-estimating Initial State Probabilities

- Initial state distribution: \( \pi_i \) is the probability that \( n_i \) is a start state

- Re-estimation is easy:
  \[ \hat{\pi}_i = \text{expected number of times in state } n_i \text{ at time } 0 \]

- Formally:
  \[ \hat{\pi}_i = \gamma_0(i) \]
Re-estimation of Emission Probabilities

- Emission probabilities are re-estimated as
  \[
  \hat{b}_i(k) = \frac{\text{expected number of times in state } n_i \text{ and observe symbol } w_k}{\text{expected number of times in state } n_i}
  \]

- Formally:
  \[
  \hat{b}_i(k) = \frac{T-1}{\sum_{t=0}^{T-1} \gamma_t(i)} \sum_{t=0}^{T-1} \delta(o_t, w_k) \gamma_t(i)
  \]

Where \( \delta(o_t, w_k) = 1 \), if \( o_t = w_k \), 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.

To iteratively update the model

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

  \[
  \hat{a}_{i,j} = \frac{\sum_{t=2}^{T-2} \xi_t(i,j)}{\sum_{t=2}^{T-2} \gamma_t(i)}, \quad \hat{b}_i(k) = \frac{\sum_{t=0}^{T-1} \delta(o_t, w_k) \gamma_t(i)}{\sum_{t=0}^{T-1} \gamma_t(i)}, \quad \hat{\pi}_i = \gamma_0(i)
  \]

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
Hidden Markov Models

Hidden Markov Model (HMM)

- HMMs allow you to estimate probabilities of unobserved events
- Given plain text, which underlying parameters generated the surface
- E.g., in speech recognition, the observed data is the acoustic signal and the words are the hidden parameters

HMMs and their Usage

- HMMs are very common in Computational Linguistics:
  - Speech recognition (observed: acoustic signal, hidden: words)
  - Handwriting recognition (observed: image, hidden: words)
  - Part-of-speech tagging (observed: words, hidden: part-of-speech tags)
  - Machine translation (observed: foreign words, hidden: words in target language)

Noisy Channel Model

- In speech recognition you observe an acoustic signal \( A = a_1, \ldots, a_n \) and you want to determine the most likely sequence of words \( W = w_1, \ldots, w_n \): \( P(W \mid A) \)
- Problem: \( A \) and \( W \) are too specific for reliable counts on observed data, and are very unlikely to occur in unseen data
Noisy Channel Model

- Assume that the acoustic signal (A) is already segmented wrt word boundaries
- \( P(W \mid A) \) could be computed as
  \[
  P(W \mid A) = \prod_{w_i} \max_{a_i} P(w_i \mid a_i)
  \]
- Problem: Finding the most likely word corresponding to an acoustic representation depends on the context
- E.g., /pre-z\textsuperscript{s}ns/ could mean “presents” or “presence” depending on the context

Noisy Channel Model

- Given a candidate sequence W we need to compute \( P(W) \) and combine it with \( P(W \mid A) \)
- Applying Bayes’ rule:
  \[
  \arg \max_W P(W \mid A) = \arg \max_W \frac{P(A \mid W)P(W)}{P(A)}
  \]
- The denominator \( P(A) \) can be dropped, because it is constant for all \( W \)

Decoding

The decoder combines evidence from

- The likelihood: \( P(A \mid W) \)
  This can be approximated as:
  \[
  P(A \mid W) = \prod_{i=1}^n P(a_i \mid w_i)
  \]
- The prior: \( P(W) \)
  This can be approximated as:
  \[
  P(W) = P(w_1) \prod_{i=2}^n P(w_i \mid w_{i-1})
  \]
Search Space

- Given a word-segmented acoustic sequence list all candidates

<table>
<thead>
<tr>
<th>'bot</th>
<th>ik-'spen-siv</th>
<th>'pre-z ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>boat</td>
<td>excessive</td>
<td>presidents</td>
</tr>
<tr>
<td>bald</td>
<td>expensive</td>
<td>presence</td>
</tr>
<tr>
<td>bold</td>
<td>expressive</td>
<td>presents</td>
</tr>
<tr>
<td>bought</td>
<td>inactive</td>
<td>press</td>
</tr>
</tbody>
</table>

- Compute the most likely path