EECS 496: Sequential Decision Making

Soumya Ray

sray@case.edu

Office: Olin 516

Office hours : None today

(possibly interesting talk: https://thedaily.case.edu/howard-t-mcmyler-lecture-automation-and-procreation/)

Announcements

- First programming assignment available
- Repositories at https://csevcs.case.edu/git/2019_fall_496_groupN
  - Replace N with group number
  - Make sure you can access it WELL BEFORE the deadline
  - Do NOT alter the URL
  - If trouble, report error returned by git clone/send screenshot

Recap

- What do sequential probabilistic models represent?
- We see _____ generated by a ____ ____ which has a ____ that ____ over time.
- The ____ can be _____ or _____.
- We can represent a sequential process in a generative way through a ___ ___ ___ ___ ___.
- These model the ____ distribution, ___.
- Each state in such a model is parametrized by an ___ distribution and a _____ distribution.
- Using these, they factorize the joint distribution as ?
- What is Markov about these models? What is hidden?
- To calculate the likelihood of a sequence, we can use ___ ___. This uses the intuition that many ___ ___ share ____, which only need to be computed ___. This algorithm is called the _____.
- We can also calculate the most likely path using a similar idea. This is called the ____ algorithm.

Today

- Sequential Probabilistic Models (Ch 15)

Forward Algorithm

- Initialize: $\alpha_{START}(0)=1, \alpha_k(0)=0, k \neq START$

- Recursion:

Emitting observation i

Transition to state k

$$\alpha_k(i)=\Pr(o_i \mid s_i = k)\sum_p \alpha_p(i-1)\Pr(s_i = k \mid s_{i-1} = p)$$

$$\alpha_k(i)=\Pr(o_1,\ldots,o_i, s_i = k)$$

$$\alpha_p(i-1)=\Pr(o_1,\ldots,o_{i-1}, s_{i-1} = p)$$

Viterbi Algorithm

- Initialize: $\gamma_{START}(0) = 1, \gamma_k(0) = 0, k \neq START$

- Recursion: Emitting observation i Transition to state k

$$\gamma_k(i) = \Pr(o_i \mid s_i^* = k) \max_p \gamma_p(i-1) \Pr(s_i^* = k \mid s_{i-1}^* = p)$$

$$\gamma_k(i) = \Pr(o_1, \ldots, o_i, s_i^* = k)$$

$$\gamma_p(i-1) = \Pr(o_1, \ldots, o_{i-1}, s_{i-1}^* = p)$$

To get the path, store the arg max's of the recursive computation.

Learning the Model Parameters

- Given data (observations), how do we set values for the probabilities in a graphical model?

- General strategy: use "Maximum Likelihood Estimation"
  - Find values of the probabilities that maximize the likelihood of the observations

Parameter Estimation

- We are given a set of observation sequences and the model structure
- Estimate parameters to maximize likelihood of the observed sequences
  - Parameters of the emission distributions for each state $Pr(b|S=k)$ (b is a vocabulary element)
  - Parameters of the transition distribution for each pair of states with an edge between them $Pr(S_{i+1}=k|S_i=j)$

Case 1: Annotated Observations

- Suppose each training observation is annotated with state information

- How can we set parameters in this case?

o_1	o_2	o_3	….	o_n
s_1	s_2	s_3	….	s_n

Case 1:Annotated Observations

- Emission distribution

$$\Pr(b \mid S = k) = \frac{\#(k \text{ emits b})}{\#(k \text{ emits anything})}$$

- Transition Distribution

$$\Pr(S = k \mid S = j) = \frac{\#(j \text{ transitions to } k)}{\#(j \text{ transitions to anything})}$$

- Can show that these are *maximum likelihood estimates* (MLEs)

Case 2: No annotations

- The state sequence generating the training sequences is *hidden*

- How to deal with this?
  - The “Baum-Welch” algorithm or “Forward-Backward” algorithm
  - An application of (soft) Expectation Maximization, used to handle *missing information*

The EM procedure

- Initialize: Guess the values of the model parameters
- E (Expectation) step: Find the expected missing values / distribution over missing values given current parameters
- M (Maximization) step: Find the maximum likelihood parameter estimates given the outcome of the E step
- Iterate until convergence

Baum-Welch Algorithm

- Start by randomly initializing all the emission and transition probabilities
- Proceed in two steps until (local) convergence
  - E step: Since we don't know the state sequences, we calculate the *expected* number of times each transition or emission is used (as if annotating observations with *possible* state sequences and averaging over them)
  - M step: Update the emission and transition probabilities using the expected counts (same as in the fully observed case above)

Baum-Welch Algorithm (E step)

Expected number of times b is emitted by state k according to our training sample: $E_{k;b}$
For each training sequence, for each time step where b is present, find the probability k was used at that point; add up all the probabilities

Pr(S=k emitted this symbol)?

$O^1 =$	o_1	o_2	o_3	$\mathbf{b}$	....		o_n
$O^2 =$	o_1	o_2	$\mathbf{b}$		....		o_m
$O^3 =$	o_1	o_2	o_3		....		o_k
$O^4 =$	$\mathbf{b}$	o_2	o_3		....	$\mathbf{b}$	o_p
$O^5 =$	o_1	o_2	o_3		....		o_n
$O^6 =$	$\mathbf{b}$	$\mathbf{b}$	$\mathbf{b}$		....		o_q

Baum-Welch Algorithm (E step)

Expected number of times b is emitted by state k according to our training sample: $E_{k;b}$
For each training sequence, for each time step where b is present, find the probability k was used at that point; add up all the probabilities

$$E_{k;b} = \sum_{j \in sequences} \sum_{i | o_i^j = b} \Pr(s_i = k \mid \mathbf{o}^j)$$

$$= \sum_{j \in sequences} \frac{1}{\Pr(\mathbf{o}^j)} \sum_{i | o_i^j = b} \Pr(s_i = k, \mathbf{o}^j)$$