Statistical NLP: Lecture 11

Hidden Markov Models

(Ch 9)

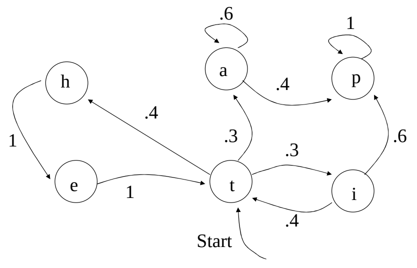
Markov Models

- Markov models are statistical tools that are useful for NLP because they can be used for part-of-speech-tagging and speech recognition.
- Their first use was in modeling the letter sequences in works of Russian literature
- They were later developed as a general statistical tool.
- More specifically, they model a sequence (perhaps through time) of random variables that are not necessarily independent.
- They rely on two assumptions: ***Limited Horizon*** and ***Time Invariant***.

Markov Assumptions

- Let $X=(X_1, .., X_t)$ be a sequence of random variables taking values in some finite set $S=\{s_1, …, s_n\}$, the state space, the Markov properties are:
- ***Limited Horizon***: $P(X_{t+1}=s_k|X_1, .., X_t)=P(X_{t+1} = s_k |X_t)$ i.e., a word's tag only depends on the previous tag.
- ***Time Invariant***: $P(X_{t+1}=s_k|X_1, .., X_t)=P(X_2 =s_k|X_1)$ i.e., the dependency does not change over time.
- If X possesses these properties, then X is said to be a Markov Chain

Example of a Markov Chain



Hidden Markov Models (HMM)

- In an HMM, you don't know the state sequence that the model passes through, but only some probabilistic function of it.
- Example: The crazy soft drink machine: it can be in two states, cola preferring and iced tea preferring, but it switches between them randomly after each purchase according to some probabilities.
- The question is: What is the probability of seeing the output sequence {lemonade, iced-tea} if the machine always starts off in the cola preferring mode.

Why Use Hidden Markov Models?

- HMMs are useful when one can think of underlying events probabilistically generating surface events. Example: Part-of-Speech-Tagging, Speech Recognition.
- HMMs can efficiently be trained using the EM Algorithm.
- Another example where HMMs are useful is in generating parameters for generalized linear interpolation of n-gram models.

General Form of an HMM

- An HMM is specified by a five-tuple (S, K, Π, A, B) where S and K are the set of states and the output alphabet, and Π, A, B are the probabilities for the initial state, state transitions, and symbol emissions, respectively.
- Given a specification of an HMM, we can simulate the running of a Markov process and produce an output sequence using the algorithm shown on the next page.
- More interesting than a simulation, however, is assuming that some set of data was generated by a HMM, and then being able to calculate probabilities and probable underlying state sequences.

A Program for a Markov Process

```
t:= 1;
Start in state $s_i$ with probability $\pi_i$ (i.e., $X_1$=i)
Forever do
    Move from state $s_i$ to state $s_j$ with probability $a_{ij}$ (i.e., $X_{t+1}$ = j)
    Emit observation symbol $o_t$ = k with probability $b_{ijk}$
    t:= t+1
End
```

The Three Fundamental Questions for HMMs

- Given a model μ=(A, B, Π), how do we efficiently compute how likely a certain observation is, that is, P(O| μ)
- Given the observation sequence O and a model μ, how do we choose a state sequence (X_1, …, X_{T+1}) that best explains the observations?
- Given an observation sequence O, and a space of possible models found by varying the model parameters μ = (A, B, π), how do we find the model that best explains the observed data?

Finding the probability of an observation (I)

- Given the observation sequence O=(o_1, …, o_T) and a model μ= (A, B, ∏), we wish to know how to efficiently compute P(O| μ). This process is called ***decoding***.
- For any state sequence X=(X_1, …, X_{T+1}), we find: $P(O|\mu)=\Sigma_{X1\ldots XT+1}\, \pi_{X1} \Pi_{t=1}^{T} a_{XtXt+1}\, b_{XtXt+1o\_t}$
- This is simply the sum of the probability of the observation occurring according to each possible state sequence.
- Direct evaluation of this expression, however, is extremely inefficient.

Finding the probability of an observation (II)

- In order to avoid this complexity, we can use ***dynamic programming*** or ***memoization*** techniques.
- In particular, we use ***treillis*** algorithms.
- We make a square array of states versus time and compute the probabilities of being in each state at each time in terms of the probabilities of being in each state at the preceding time.
- A treillis can record the probability of all initial subpaths of the HMM that end in a certain state at a certain time. The probability of longer subpaths can then be worked out in terms of the shorter subpaths.

Finding the probability of an observation (III): The ***forward procedure***

- A ***forward variable***, $\alpha_i(t)= P(o_1o_2\ldots o_{t-1}, X_t=i|\mu)$ is stored at (s_i, t) in the trellis and expresses the total probability of ending up in state s_i at time t.
- Forward variables are calculated as follows:
- ***Initialization***: $\alpha_i(1)= \pi_i,\ 1\leq i \leq N$
- ***Induction:*** $\alpha_j(t+1)=\Sigma_{i=1}^{N}\alpha_i(t)a_{ij}b_{ijo\_t}\quad 1\leq t\leq T,\ 1\leq j\leq N$
- ***Total:*** $P(O|\mu)= \Sigma_{i=1}^{N}\alpha_i(T+1)$
- This algorithm requires $2N^2T$ multiplications (much less than the direct method which takes $(2T+1)N^{T+1}$

Finding the probability of an observation (IV): The ***backward procedure***

- The backward procedure computes ***backward variables*** which are the total probability of seeing the rest of the observation sequence given that we were in state s_i at time t.
- Backward variables are useful for the problem of parameter estimation.

13

Finding the probability of an observation (V): The ***backward procedure***

- Let $\beta_i(t) = P(o_t \dots o_T \mid X_t = i, \mu)$ be the backward variables.
- Backward variables can be calculated working backward through the treillis as follows:
- ***Initialization***: $\beta_i(T+1) = 1, 1 \le i \le N$
- ***Induction:*** $\beta_i(t) = \Sigma_{j=1}^{N} a_{ij} b_{ijo_t} \beta_j(t+1), 1 \le t \le T, 1 \le i \le N$
- ***Total:*** $P(O|\mu) = \Sigma_{i=1}^{N} \pi_i \beta_i(1)$
- Backward variables can also be combined with forward variables:
  $P(O|\mu) = \Sigma_{i=1}^{N} \alpha_i(t) \beta_i(t), 1 \le t \le T+1$

14

Finding the Best State Sequence (I)

- One method consists of finding the states individually:
- For each t, $1 \le t \le T+1$, we would like to find X_t that maximizes $P(X_t|O, \mu)$.
- Let $\gamma_i(t) = P(X_t = i | O, \mu) = P(X_t = i, O|\mu)/P(O|\mu) = \alpha_i(t)\beta_i(t)/\Sigma_{j=1}^{N} \alpha_j(t)\beta_j(t)$
- The individually most likely state is
  $$\hat{X}_t = \text{argmax}_{1 \le i \le N}\, \gamma_i(t), 1 \le t \le T+1$$
- This quantity maximizes the expected number of states that will be guessed correctly. However, it may yield a quite unlikely state sequence.

15

Finding the Best State Sequence (II): The Viterbi Algorithm

- The ***Viterbi algorithm*** efficiently computes the most likely state sequence.
- Commonly, we want to find the most likely complete path, that is: $\text{argmax}_X\, P(X|O,\mu)$
- To do this, it is sufficient to maximize for a fixed O: $\text{argmax}_X\, P(X,O|\mu)$
- We define
  $\delta_j(t) = \max_{X1..Xt-1} P(X_1 \dots X_{t-1}, o_1..o_{t-1}, X_t = j|\mu)$
  $\psi_j(t)$ records the node of the incoming arc that led to this most probable path.

16

Finding the Best State Sequence (II): The Viterbi Algorithm

The Viterbi Algorithm works as follows:

- ***Initialization***: $\delta_j(1) = \pi_j, 1 \le j \le N$
- ***Induction:*** $\delta_j(t+1) = \max_{1 \le i \le N} \delta_i(t) a_{ij} b_{ijo\_t}\ 1 \le j \le N$

  Store backtrace:

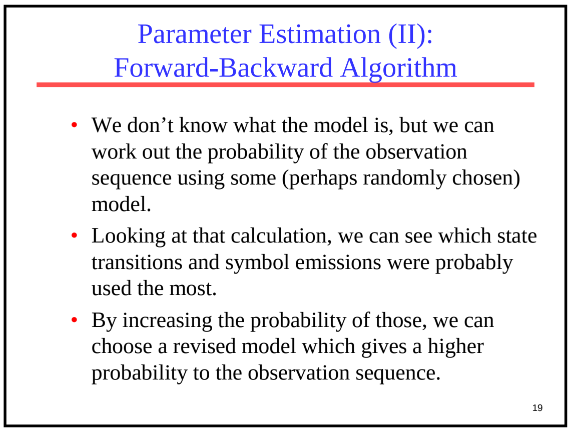
  $\psi_j(t+1) = \text{argmax}_{1 \le i \le N} \delta_j(t) a_{ij} b_{ijo\_t}\ 1 \le j \le N$
- ***Termination and path readout:***
  $\hat{X}_{T+1} = \text{argmax}_{1 \le i \le N} \delta_j(T+1)$
  $\hat{X}_t = \psi_{\hat{X}_{t+1}}(t+1)$
  $\hat{P}(X) = \max_{1 \le i \le N} \delta_j(T+1)$

17

Parameter Estimation (I)

- Given a certain observation sequence, we want to find the values of the model parameters $\mu = (A, B, \pi)$ which best explain what we observed.
- Using Maximum Likelihood Estimation, we can want find the values that maximize $P(O|\mu)$, i.e. $\text{argmax}_\mu\, P(O_{training}|\mu)$
- There is no known analytic method to choose μ to maximize $P(O|\mu)$. However, we can locally maximize it by an iterative hill-climbing algorithm known as ***Baum-Welch*** or ***Forward-Backward algorithm***. (special case of the EM Algorithm)

18



Parameter Estimation (II): Forward-Backward Algorithm

- We don’t know what the model is, but we can work out the probability of the observation sequence using some (perhaps randomly chosen) model.
- Looking at that calculation, we can see which state transitions and symbol emissions were probably used the most.
- By increasing the probability of those, we can choose a revised model which gives a higher probability to the observation sequence.