NLP
Introduction to NLP

*Hidden Markov Models (1/2)*
Markov Models

• Sequence of random variables that aren’t independent

• Examples
  – Weather reports
  – Text
  – Stock market numbers
Properties

- **Limited horizon:**
  \[ P(X_{t+1} = s_k | X_1, \ldots, X_t) = P(X_{t+1} = s_k | X_t) \]

- **Time invariant (stationary)**
  \[ = P(X_2 = s_k | X_1) \]

- **Definition:** in terms of a transition matrix \( A \) and initial state probabilities \( \Pi \).
Example
Visible MM

\[ P(X_1, \ldots X_T) = P(X_1) \ P(X_2|X_1) \ P(X_3|X_1, X_2) \ldots \ P(X_T|X_1, \ldots, X_{T-1}) \]

\[ = P(X_1) \ P(X_2|X_1) \ P(X_3|X_2) \ldots \ P(X_T|X_{T-1}) \]

\[ = \pi_{X_1} \prod_{t=1}^{T-1} a_{X_t, X_{t+1}} \]

\[ P(d, a, b) = P(X_1=d) \ P(X_2=a|X_1=d) \ P(X_3=b|X_2=a) \]

\[ = 1.0 \times 0.7 \times 0.8 \]

\[ = 0.56 \]
Hidden MM

• **Motivation**
  – Observing a sequence of symbols
  – The sequence of states that led to the generation of the symbols is hidden
  – The states correspond to hidden (latent) variables

• **Definition**
  – Q = states
  – O = observations, drawn from a vocabulary
  – $q_0, q_f$ = special (start, final) states
  – A = state transition probabilities
  – B = symbol emission probabilities
  – $\Pi$ = initial state probabilities
  – $\mu = (A, B, \Pi)$ = complete probabilistic model
Hidden MM

• Uses
  – Part of speech tagging
  – Speech recognition
  – Gene sequencing
Hidden Markov Model (HMM)

- Can be used to model state sequences and observation sequences
- Example:
  \[ P(s, w) = \prod_i P(s_i | s_{i-1}) P(w_i | s_i) \]
Generative Algorithm

• Pick start state from $\Pi$
• For $t = 1..T$
  – Move to another state based on $A$
  – Emit an observation based on $B$
State Transition Probabilities

Start

\( \text{G} \) 0.8

\( \text{H} \) 0.4

0.2

0.6

0.8
Emission Probabilities

- $P(O_t=k|X_t=s_i, X_{t+1}=s_j) = b_{ijk}$

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>0.7</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>H</td>
<td>0.3</td>
<td>0.5</td>
<td>0.2</td>
</tr>
</tbody>
</table>
All Parameters of the Model

• Initial
  – \( P(G|\text{start}) = 1.0, P(H|\text{start}) = 0.0 \)

• Transition
  – \( P(G|G) = 0.8, P(G|H) = 0.6, P(H|G) = 0.2, P(H|H) = 0.4 \)

• Emission
  – \( P(x|G) = 0.7, P(y|G) = 0.2, P(z|G) = 0.1 \)
  – \( P(x|H) = 0.3, P(y|H) = 0.5, P(z|H) = 0.2 \)
Observation sequence “yz”

- Starting in state G (or H), \( P(yz) = ? \)
- Possible sequences of states:
  - GG
  - GH
  - HG
  - HH
- \( P(yz) = P(yz|GG) + P(yz|GH) + P(yz|HG) + P(yz|HH) = \)
  \[ = .8 \times .2 \times .8 \times .1 \]
  \[ + .8 \times .2 \times .2 \times .2 \]
  \[ + .2 \times .5 \times .4 \times .2 \]
  \[ + .2 \times .5 \times .6 \times .1 \]
  \[ = .0128 + .0064 + .0080 + .0060 = .0332 \]
States and Transitions

• An HMM is essentially a weighted finite-state transducer
  – The states encode the most recent history
  – The transitions encode likely sequences of states
    • e.g., Adj–Noun or Noun–Verb
    • or perhaps Art–Adj–Noun
  – Use MLE to estimate the probabilities

• Another way to think of an HMM
  – It’s a natural extension of Naïve Bayes to sequences
Emissions

• Estimating the emission probabilities
  – Harder than transition probabilities
  – There may be novel uses of word/POS combinations

• Suggestions
  – It is possible to use standard smoothing
  – As well as heuristics (e.g., based on the spelling of the words)
Sequence of Observations

• The observer can only see the emitted symbols
• Observation likelihood
  – Given the observation sequence $S$ and the model $\mu = (A, B, \Pi)$, what is the probability $P(S | \mu)$ that the sequence was generated by that model.
• Being able to compute the probability of the observations sequence turns the HMM into a language model
Tasks with HMM

- Given $\mu = (A, B, \Pi)$, find $P(O | \mu)$
  - Uses the Forward Algorithm
- Given $O$, $\mu$, find $(X_1, ... X_{T+1})$
  - Uses the Viterbi Algorithm
- Given $O$ and a space of all possible $\mu_{1..m}$, find model $\mu_i$ that best describes the observations
  - Uses Expectation–Maximization
Inference

• Find the most likely sequence of tags, given the sequence of words
  \[ t^* = \arg\max \, P(t|w) \]
• Given the model \( \mu \), it is possible to compute \( P(t|w) \) for all values of \( t \)
  – In practice, there are way too many combinations
• Greedy Search
• Beam Search
  – One possible solution
  – Uses partial hypotheses
  – At each state, only keep the k best hypotheses so far
  – May not work
Viterbi Algorithm

• Find the best path up to observation i and state s
• Characteristics
  – Uses dynamic programming
  – Memoization
  – Backpointers
HMM Trellis

- \( P(H|G) \)
- \( P(B|B) \)
- \( P(y|G) \)
- \( P(H|G) \)
- \( P(B|B) \)
HMM Trellis

\[ P(G, t=1) = P(\text{start}) \times P(G | \text{start}) \times P(y | G) \]
HMM Trellis

\[ P(H, t=1) = P(\text{start}) \times P(H|\text{start}) \times P(y|H) \]
HMM Trellis

\[
P(H,t=2) = \max (P(G,t=1) \times P(H|G) \times P(z|H), P(H,t=1) \times P(H|H) \times P(z|H))
\]
HMM Trellis

\[ P(H, t=2) \]
HMM Trellis
HMM Trellis

P(end,t=3)
HMM Trellis

\[ P(\text{end}, t=3) = \max (P(\text{G}, t=2) \times P(\text{end} | \text{G}), \quad P(\text{H}, t=2) \times P(\text{end} | \text{H})) \]
HMM Trellis

\[ P(\text{end}, t=3) = \max (P(G, t=2) \times P(\text{end} | G), P(H, t=2) \times P(\text{end} | H)) \]

\[ P(\text{end}, t=3) = \text{best score for the sequence} \]

Use the backpointers to find the sequence of states.
Some Observations

• Advantages of HMMs
  – Relatively high accuracy
  – Easy to train

• Higher–Order HMM
  – The previous example was about bigram HMMs
  – How can you modify it to work with trigrams?
How to compute $P(O)$

- Viterbi was used to find the most likely sequence of states that matches the observation.
- What if we want to find all sequences that match the observation?
- We can add their probabilities (because they are mutually exclusive) to form the probability of the observation.
- This is done using the Forward Algorithm.
The Forward Algorithm

• Very similar to Viterbi
• Instead of $\max$ we use $\sum$

```plaintext
init t = 0, transition matrix $x_{ij}$, emission probabilities, $p(y_j|x_i)$, observed sequence, $y(1 : t)$

for $t = t + 1$

$\alpha_t(x_t) = p(y_t|x_t) \sum_{x_{t-1}} p(x_t|x_{t-1}) \alpha_{t-1}(x_{t-1})$.

until $t = T$

return $p(y(1 : t)) = \alpha_T$
```

NLP