**Modeling Sequence Data:**

**Markov Models**

CS4780/5780 – Machine Learning
Fall 2013
Thorsten Joachims
Cornell University

Reading:
Manning/Schütze, Sections 9.1-9.3 (except 9.3.1)
Leeds Online HMM Tutorial (except Forward and Forward/Backward Algorithm)

---

**“Less Naïve” Bayes Classifier**

- Example: Classify sentences as insulting / not insulting

<table>
<thead>
<tr>
<th>Test</th>
<th>Insult: 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>I:</td>
<td>(Peter is nice and not stupid)</td>
</tr>
<tr>
<td>II:</td>
<td>(Peter is not nice and stupid)</td>
</tr>
</tbody>
</table>

- Assumption (l words in document)

\[ P(X = x | Y = +1) = P(W_i = w_i | W_{i-1} = w_{i-1}, Y = +1) \]

\[ P(X = x | Y = -1) = P(W_i = w_i | W_{i-1} = w_{i-1}, Y = -1) \]

- Decision Rule

\[ h_{\text{less}}(x) = \arg \max_{y \in \{+1, -1\}} P(Y = y) P(W_i = w_i | W_{i-1} = w_{i-1}, Y = y) \]

---

**Markov Model**

- **Definition**
  - Set of States: \( s_1, \ldots, s_k \)
  - Start probabilities: \( P(S_1 = s) \)
  - Transition probabilities: \( P(S_i = s | S_{i-1} = s') \)

- **Random walk on graph**
  - Start in state \( s \) with probability \( P(S_1 = s) \)
  - Move to next state with probability \( P(S_i = s | S_{i-1} = s') \)

- **Assumptions**
  - Limited dependence: Next state depends only on previous state, but no other state (i.e. first order Markov model)
  - Stationary: \( P(S_i = s | S_{i-1} = s') \) is the same for all \( i \)

---

**Part-of-Speech Tagging Task**

- Assign the correct part of speech (word class) to each word in a document

  "The OT planet/NN Jupiter/NNP and/CC its/PRP moons/NNS are/VBP in/N effect/NN a/DT mini-solar/JJ system/NN ,/, and/CC Jupiter/NNP itself/PRP is/VBZ often/RB called/VBN a/DT star/NN ,/ and/CC Jupiter/NNP itself/PRP is/VBZ often/RB called/VBN a/DT star/NN that/IN never/RB caught/VBN fire/NN ./"

- Needed as an initial processing step for a number of language technology applications
  - Information extraction
  - Answer extraction in QA
  - Base step in identifying syntactic phrases for IR systems
  - Critical for word-sense disambiguation (WordNet apps)
  - …

---

**Why is POS Tagging Hard?**

- **Ambiguity**
  - He will race/VB the car.
  - When will the race/NN end?
  - I bank/VB at CFCU.
  - Go to the bank/NN!

- Average of \( \sim 2 \) parts of speech for each word
  - The number of tags used by different systems varies a lot. Some systems use < 20 tags, while others use > 400.

---

**The POS Learning Problem**

- **Example**

<table>
<thead>
<tr>
<th>Sentence</th>
<th>POS</th>
</tr>
</thead>
<tbody>
<tr>
<td>I:</td>
<td>(I, bank, of, CFCU)</td>
</tr>
<tr>
<td>II:</td>
<td>(PHP, V, PREP, S)</td>
</tr>
<tr>
<td>III:</td>
<td>(Go, to, the, bank)</td>
</tr>
<tr>
<td>IV:</td>
<td>(V, PREP, DET, X)</td>
</tr>
</tbody>
</table>
Hidden Markov Model for POS Tagging

• States
  – Think about as nodes of a graph
  – One for each POS tag
  – special start state (and maybe end state)
• Transitions
  – Think about as directed edges in a graph
  – Edges have transition probabilities
• Output
  – Each state also produces a word of the sequence
  – Sentence is generated by a walk through the graph

Hidden Markov Model

• States: \( y \in \{s_1, \ldots, s_k\} \)
• Outputs symbols: \( x \in \{o_1, \ldots, o_m\} \)
• Starting probability \( P(Y_1 = y_1) \)
  – Specifies where the sequence starts
• Transition probability \( P(Y_i = y_i \mid Y_{i-1} = y_{i-1}) \)
  – Probability that one states succeeds another
• Output/Emission probability \( P(X_i = x_i \mid Y_i = y_i) \)
  – Probability that word is generated in this state

=> Every output-state sequence has a probability

\[
P(x, y) = \prod_{i=2}^{l} P(x_i \mid y_i) P(y_i \mid y_{i-1})
\]

Estimating the Probabilities

• Given: Fully observed data
  – Pairs of output sequence with their state sequence
• Estimating transition probabilities \( P(Y_i \mid Y_{i-1}) \)

\[
\hat{p}(y_i \mid y_{i-1}) = \frac{\text{# times state } a \text{ follows state } b}{\text{# of times state } b \text{ occurs}}
\]

• Estimating emission probabilities \( P(X_i \mid Y_i) \)

\[
\hat{p}(x_i \mid y_i) = \frac{\text{# times output } a \text{ is observed in state } b}{\text{# of times state } b \text{ occurs}}
\]

• Smoothing the estimates
  – Laplace smoothing -> uniform prior
  – See naive Bayes for text classification
• Partially observed data
  – Expectation Maximization (EM)