Agenda

- HW2 – due today
- Collect printouts
- Questions about the homework to be posted to the class list, or to compling723.fall2011@gmail.com
- Language Modeling with <s>
- Logarithmic Math
- HW3 – online today, due in two weeks
- Forward Algorithm
- Viterbi Algorithm

Language Modeling with <s>

- Example corpus

<table>
<thead>
<tr>
<th>corpus.txt</th>
<th>w1</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;s&gt; hello &lt;/s&gt;</td>
<td>ε 0</td>
</tr>
<tr>
<td>&lt;s&gt; bye &lt;/s&gt;</td>
<td>hello 1</td>
</tr>
<tr>
<td>&lt;s&gt; hello &lt;/s&gt;</td>
<td>bye 2</td>
</tr>
<tr>
<td>&lt;s&gt; bye bye &lt;/s&gt;</td>
<td>&lt;s&gt; 3</td>
</tr>
<tr>
<td>&lt;/s&gt; 4</td>
<td></td>
</tr>
</tbody>
</table>

Language Modeling with <s>

- Compile into a (weighted) FSM LM

![FSM diagram]

Language Modeling with <s>

- Determinize and minimize FSM LM

![Determinized FSM diagram]

Log-Domain Mathematics

When multiplying many numbers together, we run the risk of underflow errors, one solution is to transform everything into the log domain:

<table>
<thead>
<tr>
<th>Linear domain</th>
<th>log domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>x^y</td>
<td>e^y \cdot x</td>
</tr>
<tr>
<td>x+y</td>
<td>e^{x+y}</td>
</tr>
</tbody>
</table>

logAdd(x,y) computes the log-domain sum of x and y when both x and y are already in log domain. In the linear domain:

\[ \log(x + y) = \log(x + e^y) = \log(x) + \log(1 + e^y) = \log(x) + \log(1 + \frac{e^y}{x}) \]

Major point: \( \logAdd(x,y) \) is NOT same as \( \log(x+y) = \log(x) + \log(y) \)
A little trick with logs...

Recall: $e^{x+y} = e^x e^y$

$$\log(e^A + e^B) = \log(e^{A-B} + e^B) = \log(e^B e^{A-B} + e^B) = \log(e^B(1 + e^{A-B})) = \log e^B + \log(e^{A-B} + 1) = B + \log(e^{A-B} + 1) = A + \log(e^{B-A} + 1)$$

Don’t want $e^{A-B}$ to be large. Hence, if $A > B$, calculate $A + \log(e^{B-A} + 1)$.

**Agenda**

- HW2 – due today
  - Collect printouts
  - Questions about the homework to be posted to the class list, or to compling723.fall2011@gmail.com
- Logarithmic Math
- HW3 – online today, due in two weeks
- Forward Algorithm
- Viterbi Algorithm

**HMM Independence**

- The probability of an output symbol depends only on the state generating it

$$P(o_1 \mid q_1, \ldots, o_N; \alpha_1, \alpha_2, \ldots, \alpha_T) = P(o_i | q_i)$$

- Where $x$ are the (hidden) states and $y$ are our (observed) events:

**HMMs: Three Problems**

- **Likelihood:** Given an HMM $\lambda = (A, B, \Pi)$, and a sequence of observed events $O$, find $P(O | \lambda)$
- **Decoding:** Given an HMM $\lambda = (A, B, \Pi)$, and an observation sequence $O$, find the most likely (hidden) state sequence
- **Learning:** Given a set of observation sequences and the set of states $Q$ in $\lambda$, compute the parameters $A$ and $B$
Assuming $\lambda_{stock}$ models the stock market, how likely are we to observe the sequence of outputs?

- Computes $P(O|\lambda_{stock})$ using dynamic programming.

### Forward Algorithm: Formal Definition

- **Initialization**
  \[ a_0(j) = \pi_j b_j(\omega_1), 1 \leq j \leq N \]

- **Recursion**
  \[ a_t(j) = \sum_{i=1}^N a_{t-1}(i) a_{ij} b_j(\omega_t), 1 \leq j \leq N, 2 \leq t \leq T \]

- **Termination**
  \[ P(O|\lambda) = \sum_{i=1}^N a_T(i) \]

- **Find** $P(O|\lambda_{stock})$
Forward Algorithm

\[ \alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t); 1 \leq j \leq N, 2 \leq t \leq T \]

Forward Algorithm: Initialization

\[ \alpha_1(j) = \pi_j b_j(o_1), 1 \leq j \leq N \]

HMMs: Three Problems

- Likelihood: Given an HMM \( \lambda = (A, B, \pi) \), and a sequence of observed events \( O \), find \( P(O|\lambda) \)
- Decoding: Given an HMM \( \lambda = (A, B, \pi) \), and an observation sequence \( O \), find the most likely (hidden) state sequence
- Learning: Given a set of observation sequences and the set of states \( Q \) in \( \lambda \), compute the parameters \( A \) and \( B \)

Decoding

Given \( \lambda_{stock} \) as our model and \( O \) as our observations, what are the most likely states the market went through to produce \( O \)?
Decoding

- "Decoding" because states are hidden
- First try:
  - Compute $P(O)$ for all possible state sequences, then choose sequence with highest probability
  - What's the problem here?
- Second try:
  - For each possible hidden state sequence, compute $P(O)$ using the forward algorithm
  - What's the problem here?

Viterbi Algorithm

- "Decoding" = computing most likely state sequence
- Another dynamic programming algorithm
- Efficient: polynomial vs. exponential (brute force)
- Same idea as the forward algorithm
  - Store intermediate computation results in a trellis
  - Build new cells from existing cells

Viterbi Algorithm: Formal Definition

- Initialization
  - $v_i(j) = \pi a_i(n_i); 1 \leq i \leq N$
  - $\delta T_i(t) = 0$
- Recursion
  - $n_j(t) = \max_{i=1}^{N} v_{i-1}(i) a_j b_j(o_t); 1 \leq i \leq N, 2 \leq t \leq T$
  - $\delta T_i(t) = \arg \max_{i=1}^{N} v_{i-1}(i) o_t$
- Termination
  - $P^* = \max_{j=1}^{N} v_T(j)$
  - $q^*_T = \arg \max_{j=1}^{N} v_T(j)$

Viterbi vs. Forward

- Maximization instead of summation over previous paths
- This algorithm is still missing something!
  - In forward algorithm, we only care about the probabilities
  - What's different here?
- We need to store the most likely path (transition):
  - Use "backpointers" to keep track of most likely transition
  - At the end, follow the chain of backpointers to recover the most likely state sequence

Viterbi Algorithm

- $O = \uparrow \downarrow \uparrow$
- find most likely state sequence given $\lambda_{stock}$
Viterbi Algorithm

Viterbi Algorithm: Initialization
\[ \alpha_1(\text{Bear}) = \alpha_1(\text{Bull}) = \alpha_1(\text{Static}) = 0 \]

Viterbi Algorithm: Recursion
\[ \alpha_t(j) = \max_{i=1}^{N} \left( \alpha_{t-1}(i) \cdot b_{ij} \right) \]

Viterbi Algorithm: Recursion
\[ B_t(i) = \max_{j=1}^{N} \left( \alpha_{t-1}(i) \cdot b_{ij} \cdot a_{ji} \right) \]

Work through the rest of the algorithm...

POS Tagging with HMMs
Modeling the problem

• What’s the problem?
  • The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

• What should the HMM look like?
  • States: part-of-speech tags (t1, t2, ..., tN)
  • Output symbols: words (w1, w2, ..., wN)

  Given HMM \( \lambda \) (A, B, \( \pi \)), POS tagging = reconstructing the best state sequence given input
  • Use Viterbi decoding (best = most likely)

But wait…

HMM Training

• What are appropriate values for A, B, \( \pi \)?
  • Before HMMs can decode, they must be trained…
    • A: transition probabilities
    • B: emission probabilities
    • \( \pi \): prior

  Two training methods:
    • Supervised training: start with tagged corpus, count stuff to estimate parameters
    • Unsupervised training: start with untagged corpus, bootstrap parameter estimates and improve estimates iteratively

HMMs: Three Problems

• Likelihood: Given an HMM \( \lambda = (A, B, \pi) \), and a sequence of observed events O, find \( P(O|\lambda) \)

• Decoding: Given an HMM \( \lambda = (A, B, \pi) \), and an observation sequence O, find the most likely (hidden) state sequence

• Learning: Given a set of observation sequences and the set of states Q in \( \lambda \), compute the parameters A and B

Supervised Training

• A tagged corpus tells us the hidden states!

• We can compute Maximum Likelihood Estimates (MLEs) for the various parameters
  • MLE = fancy way of saying ‘count and divide’

• These parameter estimates maximize the likelihood of the data being generated by the model

Supervised Training

• Transition Probabilities
  • Any \( P(t_i | t_{i-1}) = C(t_i, t_{i-1}) / C(t_{i-1}) \), from the tagged data

  Example: for \( P(\text{NN}|\text{VB}) \), count how many times a noun follows a verb and divide by the total number of times you see a verb

• Emission Probabilities
  • Any \( P(w_i | t_i) = C(w_i, t_i) / C(t_i) \), from the tagged data

  For \( P(\text{bank}|\text{NN}) \), count how many times bank is tagged as a noun and divide by how many times anything is tagged as a noun

• Priors
  • Any \( P(t_i = q_i) = \pi_i = C(t_i) / N \), from the tagged data

  For \( \pi_{\text{NN}} \), count the number of times NN occurs and divide by the total number of tags (states)

  A better way?

Agenda

• HW2 – due today
  • Collect printouts
  • Questions about the homework to be posted to the class list

• HW3 – online today, due in two weeks
  • Language Modeling with <s>
  • Logarithmic Math
  • Forward Algorithm, Viterbi Algorithm

  Next time:
    • Unsupervised training ‘teaser’
    • Other HMM/tagging tasks