Language and Statistics II

Lecture 3: Sequences (cont’d.)
Noah Smith
Quick Review

• Markov/n-gram models
• Can be a source model (e.g., ASR) or a channel model (e.g., textcat)
• (Weighted) lattices and $n$-gram models
  – Finding the best path
• Adding classes deterministically (Brown et al., 1990) and stochastically (HMMs)
HMMs

Joint probability of **classes** and **words** is easy.

$$p(c_1^n, s_1^n) = \left( \prod_{i=1}^{n} \eta(s_i | c_i) \cdot \gamma(c_i | c_{i-m}^{i-1}) \right) \cdot \gamma(\text{stop} | c_{n-m+1}^n)$$

$$p(s_1^n) = \sum_{c_1^n \in \Lambda^n} \left( \prod_{i=1}^{n} \eta(s_i | c_i) \cdot \gamma(c_i | c_{i-m}^{i-1}) \right) \cdot \gamma(\text{stop} | c_{n-m+1}^n)$$

Marginal probability of words?

Naïve algorithm: $O(2^n)$
Inference with HMMs

• Many inference problems can be solved in polynomial time!
  – Unlike general graphical models (why?)
  – Dynamic programming (a.k.a. sum-product or max-product algorithms)

• Probability of a sequence:
  – **forward** algorithm
  – **backward** algorithm
Deriving the Backward Algorithm

\[ p(s_1^n) = p(s_1^n \mid C_0 = \text{start}) \rightarrow \alpha(0, \text{start}) \]

\[
= \sum_{c_1^n \in \Lambda^n} \left( \prod_{i=1}^{n} \eta(s_i \mid c_i) \cdot \gamma(c_i \mid c_{i-1}) \right) \gamma(\text{stop} \mid c_n) \\
= \sum_{c_1 \in \Lambda} \sum_{c_2^n \in \Lambda^{n-1}} \left( \prod_{i=1}^{n} \eta(s_i \mid c_i) \cdot \gamma(c_i \mid c_{i-1}) \right) \gamma(\text{stop} \mid c_n) \\
= \sum_{c_1 \in \Lambda} \eta(s_1 \mid c_1) \cdot \gamma(c_1 \mid C_0 = \text{start}) \cdot p(s_2^n \mid c_1) \\
= \sum_{c \in \Lambda} \eta(s_1 \mid c) \cdot \gamma(c \mid C_1 = c) \cdot p(s_2^n \mid C_1 = c) \rightarrow \alpha(1, c) \]
Backward Algorithm
(Bigram HMM Equations)

\[ \alpha(i, c') = \sum_{c \in \Lambda} \eta(s_{i+1} | c) \cdot \gamma(c | c') \cdot \alpha(i + 1, c) \]

\[ p(s_1^n) = \alpha(0, \text{start}) \]

\[
\begin{bmatrix}
\alpha(0, \text{start}) \\
\alpha(0, c_1) \\
\vdots \\
\alpha(0, c_{|\Lambda|}) \\
\alpha(0, \text{stop})
\end{bmatrix}
\]

\[
\begin{bmatrix}
\alpha(i, \text{start}) \\
\alpha(i, c_1) \\
\vdots \\
\alpha(i, c_{|\Lambda|}) \\
\alpha(i, \text{stop})
\end{bmatrix}
\]

\[
\begin{bmatrix}
\alpha(i + 1, \text{start}) \\
\alpha(i + 1, c_1) \\
\vdots \\
\alpha(i + 1, c_{|\Lambda|}) \\
\alpha(i + 1, \text{stop})
\end{bmatrix}
\]

\[ \alpha(n + 1, \text{stop}) = 1 \]
Forward Algorithm
(Bigram HMM Equations)

\[
\beta(i,c') = \sum_{c \in \Lambda} \eta(s_i | c') \cdot \gamma(c' | c) \cdot \beta(i-1,c)
\]

\[
\beta(0, \text{start}) = 1
\]

\[
\begin{bmatrix}
\beta(0, \text{start}) = 1 \\
0 \\
\vdots \\
0 \\
0
\end{bmatrix}
\]

\[
\begin{bmatrix}
\beta(i, \text{start}) \\
\beta(i,c_1) \\
\vdots \\
\beta(i,|\Lambda|) \\
\beta(i, \text{stop})
\end{bmatrix}
\]

\[
\begin{bmatrix}
\beta(i + 1, \text{start}) \\
\beta(i + 1,c_1) \\
\vdots \\
\beta(i + 1,|\Lambda|) \\
\beta(i + 1, \text{stop})
\end{bmatrix}
\]

\[
p(s_1^n) = \beta(n + 1, \text{stop})
\]

\[
\begin{bmatrix}
\beta(n + 1, \text{start}) \\
\beta(n + 1,c_1) \\
\vdots \\
\beta(n + 1,|\Lambda|) \\
\beta(n + 1, \text{stop})
\end{bmatrix}
\]
Forward and Backward Probabilities

\[ \alpha(i, c) = p\left(s_{i+1}^n \mid C_i = c \right) \]

\[ \beta(i, c) = p\left(s_i^i, C_i = c \right) \]

\[ \alpha(i, c) \cdot \beta(i, c) = p\left(s_{i+1}^n, C_i = c \right) \]

\[ \frac{\alpha(i, c) \cdot \beta(i, c)}{\beta(n + 1, \text{stop})} = p\left(C_i = c \mid s_{1}^n \right) \]

\[ \sum_{i=1}^{n} \frac{\alpha(i, c) \cdot \beta(i, c)}{\beta(n + 1, \text{stop})} = E\left[ \left\{ i : C_i = c \right\} \mid s_{1}^n \right] \]

“backward” probability

“forward” probability

posterior probability that \( s_i \) is labeled with class \( c \)

expected count of class \( c \)
Why it Works

Nothing to the right of $c_i$ can influence the distribution over $c_{i-1}$. 

Diagram with circles and arrows indicating the flow of influence.
Why it Works

Nothing to the left of $c_i$ can influence the distribution over $c_{i+1}$. 
What about a trigram HMM?

\[ \forall c, \quad \alpha(n + 1, c, \text{stop}) = 1 \]

\[ \alpha(i, c, c') = \sum_{c'' \in \Lambda} \eta(s_{i+1} \mid c'') \cdot \gamma(c'' \mid c, c') \cdot \alpha(i + 1, c', c'') \]

\[ p(s_i^n) = \alpha(0, \text{start}, \text{start}) \]
HMM Problem 2: Most Probable Path

\[ \beta^* (0, \text{start}) = 1 \]

\[ \beta^* (i, c') = \max_{c \in \Lambda} \eta(s_i \mid c') \cdot \gamma(c' \mid c) \cdot \beta^* (i - 1, c) \]

\[ \max_{c_i^n \in \Lambda^n} p(s_1^n, c_1^n) = \beta^* (n + 1, \text{stop}) \]

How to recover the path itself?

Is it necessary to go left to right?
HMM Problem 3: Minimum Expected Label Loss Path

\[ \hat{c}_i = \arg\max_{c \in \Lambda} p \left( C_i = c \mid s_1^n \right) = \arg\max_{c \in \Lambda} \alpha(i, c) \beta(i, c) \]

\[ \hat{c}_1^n = \arg\min_{c_{1}^{n} \in \Lambda^{n}} E \left[ \left| \left\{ i : C_i \neq c_i \right\} \right| \right] \]
HMM Problem 4: Most Probable Path Through a Lattice

- Lattice unweighted?
  - No problem! Slight generalization of Viterbi: index states, not word positions.

- Lattice weighted?
  - NP-hard!
  - Casacuberta & de la Higuera (2000); Lyngsoe & Pedersen (2002)
## Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>$n$-gram</th>
<th>HMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best sequence in an unweighted lattice</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Best sequence in a weighted lattice (product score)</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Total probability of unweighted lattice</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Total probability of weighted lattice (product score)</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
Extensions to HMMs

• Higher $n$ (how are algorithms affected?)
• Factored states, multiple levels
  – Lots of neat work by Bilmes (UW).
  – Also a toolkit.
• Mixture with Markov model
• Alternative estimation criteria (coming soon)
Hidden Markov Model

\[ \gamma(c_i | c_{i-2}, c_{i-1}) \]

\[ \eta(s_i | c_i) \]
Hidden Markov Model  
(Variant with more conditioning)

\[ \gamma(c_i | c_{i-2}, c_{i-1}) \]

\[ \eta(s_i | c_{i-1}, c_i) \]
Hidden Markov Model
(Factored-state variant)

\[ \eta(s_i | c_i) \]

\[ \phi(q_i | q_{i-1}) \]

\[ \gamma(c_i | c_{i-1}, q_i) \]
Summary So Far

- Tradeoffs in modeling
- Model ≠ application ≠ inference algorithm
- Review of HMMs (model, well-known applications, common algorithms)
- Lots of dynamic programming tricks

Next:
- Sequence labeling alternatives (features, estimation) … log-linear models
- Weighted finite-state NLP
- Beyond sequence labeling: parsing