Sequence Models 3

CS 287
INFERENCE IN AN AUTHORSHIP PROBLEM

A comparative study of discrimination methods applied to the authorship of the disputed Federalist papers

FREDERICK MOSTELLER
Harvard University
and
Center for Advanced Study in the Behavioral Sciences
AND
DAVID L. WALLACE
University of Chicago

This study has four purposes: to provide a comparison of discrimination methods; to explore the problems presented by techniques based strongly on Bayes’ theorem when they are used in a data analysis of large scale; to solve the authorship question of The Federalist papers; and to propose routine methods for solving other authorship problems.

Word counts are the variables used for discrimination. Since the topic written about heavily influences the rate with which a word is used, care in selection of words is necessary. The filler words of the language such as an, of, and upon, and, more generally, articles, prepositions, and conjunctions provide fairly stable rates, whereas more meaningful words like war, executive, and legislature do not.
Review: Markov Models

- In general, intractable to solve sequence prediction,

\[
\arg\max_{c_{1:n}} f(x, c_{1:n})
\]

- Today, focus on (first-order) Markov models,

\[
f(x, c_{1:n}) = \sum_{i=1}^{n} \log \hat{y}(c_{i-1}) c_i
\]

- Can extend these ideas to higher-order models.

\[
f(x, c_{1:n}) = \sum_{i=1}^{n} \log \hat{y}(c_{i-2}, c_{i-1}) c_i
\]
Review: Greedy Search

\[ f(x, c_{1:n}) = \sum_{i=1}^{n} \log \hat{y}(c_{i-1})c_i \]
Review: Greedy Search

\[ f(x, c_{1:n}) = \sum_{i=1}^{n} \log \hat{y}(c_{i-1}) c_i \]
Review: Greedy Search

\[ f(x, c_{1:n}) = \sum_{i=1}^{n} \log \hat{y}(c_{i-1})c_i \]
Review: Greedy Search

\[ f(x, c_{1:n}) = \sum_{i=1}^{n} \log \hat{y}(c_{i-1}) c_i \]
Review: Greedy Search

\[ f(x, c_{1:n}) = \sum_{i=1}^{n} \log \hat{y}(c_{i-1}) c_i \]

Mayor DeBlasio from New York
Review: Dynamic Programming over a Lattice

- Several different varieties: Viterbi, forward, backward
- Recursive Definition:
  - Base Case: Start with the score for sequence of length 1.
  - Inductive Case: Compute all sequences of length $i$ from $i - 1$

Mayor DeBlasio from New York
Review: Viterbi Algorithm

**procedure** VITERBI

$$\pi \in \mathbb{R}^{(n+1) \times \mathcal{C}}$$ initialized to $$-\infty$$

$$\pi[0, \langle s \rangle] = 0$$

**for** $$i = 1$$ **to** $$n$$ **do**

**for** $$c_i \in \mathcal{C}$$ **do**

$$\pi[i, c_i] = \max_{c_{i-1}} \pi[i-1, c_{i-1}] + \log \hat{y}(c_{i-1}) c_i$$

**return** $$\max_{c_n \in \mathcal{C}} \pi[n, c_n]$$

- Time Complexity?
- Space Complexity?
Quiz: Reverse it

Each of the algorithms goes left-to-right (forward) when producing a sequence. Sometimes you can derive right-to-left versions of these algorithms. Consider right-to-left cases of the following. Which are possible to run, how does the algorithm change, and do you get the same solutions?

1. Right-to-left greedy search on a Markov model.
2. Right-to-left Viterbi search on a Markov model.
3. Right-to-left greedy search on an RNN model.
Answer: R-to-L Greedy Search

- In general, may not work.
- May require computing and renormalizing one step in the past \((O(|C|^2))\)
- Likely gives a different solution than forward greedy.

```plaintext
procedure GreedySearch
    s = 0  \(\triangleright\) Running score
    \(c_{n+1} = \langle/\rangle\)  \(\triangleright\) Final Symbol
    for \(i = n\) to 1 do
        \(c_i \leftarrow \arg \max_{c'_i} \hat{y}(c'_i) c_{i+1}\)
        \(s \leftarrow s + \log \hat{y}(c_i) c_{i+1}\)
    return \(c, s\)
```
Answer: R-to-L Greedy Search (worst-case)

\[ f(x, c_{1:n}) = \sum_{i=1}^{n} \log \hat{y}(c_{i-1})c_i \]

<table>
<thead>
<tr>
<th>O</th>
<th>O</th>
<th>O</th>
<th>O</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-PER</td>
<td>I-PER</td>
<td>I-PER</td>
<td>I-PER</td>
<td>I-PER</td>
</tr>
<tr>
<td>I-ORG</td>
<td>I-ORG</td>
<td>I-ORG</td>
<td>I-ORG</td>
<td>I-ORG</td>
</tr>
<tr>
<td>I-LOC</td>
<td>I-LOC</td>
<td>I-LOC</td>
<td>I-LOC</td>
<td>I-LOC</td>
</tr>
</tbody>
</table>

Mayor DeBlasio from New York
Answer: R-to-L Greedy Search (worst-case)

$$f(x, c_{1:n}) = \sum_{i=1}^{n} \log \hat{y}(c_{i-1}) c_i$$
Answer: R-to-L Greedy Search (worst-case)

\[ f(x, c_{1:n}) = \sum_{i=1}^{n} \log \hat{y}(c_{i-1})c_i \]
Answer: R-to-L Greedy Search (worst-case)

\[ f(x, c_{1:n}) = \sum_{i=1}^{n} \log \hat{y}(c_{i-1})c_i \]
Answer: R-to-L Greedy Search (worst-case)

$$f(x, c_{1:n}) = \sum_{i=1}^{n} \log \hat{y}(c_{i-1})c_i$$

Mayor DeBlasio from New York
Answer: Backward Viterbi

- Same speed, same results.
- Similar inductive rule applies.
- Construct sequences starting at the end.

**Procedure** \textsc{BackwardViterbi}

\[ \pi \in \mathbb{R}^{(n+1) \times C} \text{ initialized to } -\infty \]

\[ \pi[n + 1, \langle /s \rangle] = 0 \]

for \( i = n \) to 1 do

for \( c_i \in C \) do

\[ \pi[i, c_i] = \max_{c_i' + 1} \pi[i + 1, c_i' + 1] + \log \hat{y}(c_i)c_i' + 1 \]

return \( \max_{c_1 \in C} \pi[1, c_1] \)
Backward Viterbi

Mayor DeBlasio from New York
Backward Viterbi

Mayor
DeBlasio
from
New
York
Backward Viterbi

Mayor DeBlasio from New York
Backward Viterbi

Mayor DeBlasio from New York
Backward Viterbi

Mayor DeBlasio from New York
Answer: Right-to-Left Greedy RNN

- Does not work.

- Have suffix $c_n \ldots c_i$ but RNN is prefix-function $\hat{y}(c_1, \ldots c_i)$.

- Backward greedy would require enumerating all prefixes.
Alternative Solution: Backward RNN

- Can train RNN in the reverse direction.
- RNN suffix transducer trained on $\hat{y}(c_n, \ldots c_i)$.
- However note this is a different model.
- Even if you had exact search, this may yield a different output.
Today’s Lecture

- More Dynamic Programming
  - Max-Marginals
  - Forward-Backward
  - Probabilistic Marginals
  - Pruning

- Next Lecture: Conditional Random Fields
Question 1: Tag Fill-In

Assume we are given $c_{1:i-1}$ and $c_{i+1:n}$,

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>I-PER</td>
<td>I-PER</td>
<td>I-PER</td>
<td>I-PER</td>
<td>I-PER</td>
<td>I-PER</td>
</tr>
<tr>
<td>I-ORG</td>
<td>I-ORG</td>
<td>I-ORG</td>
<td>I-ORG</td>
<td>I-ORG</td>
<td>I-ORG</td>
</tr>
<tr>
<td>I-LOC</td>
<td>I-LOC</td>
<td>I-LOC</td>
<td>I-LOC</td>
<td>I-LOC</td>
<td>I-LOC</td>
</tr>
</tbody>
</table>

Mayor    DeBlasio from New York

What is the best completion, i.e.

$$\arg \max_{c_i} f(x, c_{1:i-1}: c'_i: c_{i+1:n})$$
Question 1: Tag Fill-In

Assume we are given $c_{1:i-1}$ and $c_{i+1:n}$,

$$\text{Mayor} \quad \text{DeBlasio} \quad \text{from} \quad \text{New} \quad \text{York}$$

What is the best completion, i.e.

$$\arg \max_{c_i} f(x, c_{1:i-1} : c_i' : c_{i+1:n})$$
Answer

- Markov model. Score involving $c_i$ are local ($c_{i-1}$ and $c_{i+1}$).

- Can solve by looking one-step forward and backward

$$\begin{align*}
\arg \max_{c_i'} f(\mathbf{x}, c_{1:n}) &= \arg \max_{c_i'} \log \hat{y}(c_{i-1})_{c'_i} + \log \hat{y}(c'_i)_{c_{i+1}} \\
&= \arg \max_{c_i'} \log \hat{y}(c_{i-1})_{c'_i} + \log \hat{y}(c'_i)_{c_{i+1}}
\end{align*}$$

- Can be solved in $O(|C|)$ or $O(|C|^2)$ depending on model.
Exercise 2: Sequence fill-in

Assume we are not given $c_{1:i-1}$ and $c_{i+1:n}$,

$$M(i, c'_i) = \max_{c_{1:i-1}, c_{i+1:n}} f(x, c_{1:i-1} : c'_i : c_{i+1:n})$$

<table>
<thead>
<tr>
<th>O</th>
<th>O</th>
<th>O</th>
<th>O</th>
<th>O</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-PER</td>
<td>I-PER</td>
<td>I-PER</td>
<td>I-PER</td>
<td>I-PER</td>
<td>I-PER</td>
</tr>
<tr>
<td>I-ORG</td>
<td>I-ORG</td>
<td>I-ORG</td>
<td>I-ORG</td>
<td>I-ORG</td>
<td>I-ORG</td>
</tr>
<tr>
<td>I-LOC</td>
<td>I-LOC</td>
<td>I-LOC</td>
<td>I-LOC</td>
<td>I-LOC</td>
<td>I-LOC</td>
</tr>
</tbody>
</table>

Mayor  DeBlasio  from  New  York
Markov property. Two prefix and suffix score can are independent.

\[ M(i, c'_i) = \max_{c_{1:i-1} : c'_i : c_{i+1:n}} f(x, c_{1:i-1} : c'_i : c_{i+1:n}) \]

\[ = \max_{c_{1:i-1}} \log \hat{y}(c'_{i-1})_{c_i} + \sum_{j=1}^{i} \log \hat{y}(c_{j-1})_{c_j} \]

\[ + \max_{c_{i+1:n}} \log \hat{y}(c'_{i})_{c_{i+1}} + \sum_{j=i+1}^{n} \log \hat{y}(c_j)_{c_{j+1}} \]
Viterbi Forward-Backward

\[
\max_{c_{1:i-1}} \log \hat{y}(c'_{i-1})_{c_i} + \sum_{j=1}^{i} \log \hat{y}(c_{j-1})_{c_j} \\
+ \max_{c_{i+1:n}} \log \hat{y}(c'_{i})_{c_{i+1}} + \sum_{j=i+1}^{n} \log \hat{y}(c_{j})_{c_{j+1}}
\]

- Forward Viterbi scores (max prefix)

\[
\pi^\alpha[i, c'_i] = \max_{c_{1:i-1}} \log \hat{y}(c'_{i-1})_{c_i} + \sum_{j=1}^{i} \log \hat{y}(c_{j-1})_{c_j}
\]

- Backward Viterbi scores (max suffix)

\[
\pi^\beta[i, c'_i] = \max_{c_{i+1:n}} \log \hat{y}(c'_{i})_{c_{i+1}} + \sum_{j=i+1}^{n} \log \hat{y}(c_{j})_{c_{j+1}}
\]
Mayor De Blasio from New York
Computing All Max-Marginals

\[
\max_{c_i} \max_{c_{1:i-1:i+1:n}} f(x, c_{1:n})
\]

▶ Compute \( \pi^\alpha \) using Viterbi forward
▶ Compute \( \pi^\beta \) using Viterbi backward
▶ Compute the argmax

\[
M(i, c'_i) = \pi^\alpha[i, c'_i] + \pi^\beta[i, c'_i]
\]

▶ Time complexity?
▶ Space complexity?
Mayor DeBlasio from New York
Mayor De Blasio from New York
Mayor DeBlasio from New York
Mayor De Blasio from New York
Mayor DeBlasio from New York
Mayor De Blasio from New York
Mayor De Blasio from New York
Mayor DeBlasio from New York
Max-Marginals with Backpointers

- Can also include forward and backward backpointers
- Get best sequence through any tag in $O(n|\mathcal{C}|^2)$. 
Max-Marginal Property

If $c_i^*$ is part of highest-scoring sequence then max-marginal at $M(i, c_i^*)$ is $\max_{c_1:n} f(x, c_1:n)$ and by definition is at least as large as any other max-marginal $M(j, c_j)$ for all $j, c_j$. 
Pruning by Bounding (Sketch)

First run greedy over full problem.

Score $s^{(greedy)}$ is less than optimal.
Pruning by Bounding (Sketch)

Each edge is the \texttt{max} of the edges in the full lattice.

Can compute max-marginals \( M \) in much less time.

Mayor DeBlasio from New York
Pruning by Bounding (Sketch)

- For all $j, c_j$, if $M(j, c_j) < s^{greedy}$ can prune.

- Running Viterbi over pruned lattice is provably optimal.

Mayor DeBlasio from New York
Contents

Probabilistic Models
Let us return to the case of discriminative probabilistic models.

- Model of

\[ p(y = \delta(c_{1:n})|\mathbf{x}) \]
Exercise 3: Smoothing

Assume we are given $c_{1:i-1}$ and $c_{i+1:n}$,

<table>
<thead>
<tr>
<th>O</th>
<th>O</th>
<th>O</th>
<th>O</th>
<th>O</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-PER</td>
<td>I-PER</td>
<td>I-PER</td>
<td>I-PER</td>
<td>I-PER</td>
<td>I-PER</td>
</tr>
<tr>
<td>I-ORG</td>
<td>I-ORG</td>
<td>I-ORG</td>
<td>I-ORG</td>
<td>I-ORG</td>
<td>I-ORG</td>
</tr>
<tr>
<td>I-LOC</td>
<td>I-LOC</td>
<td>I-LOC</td>
<td>I-LOC</td>
<td>I-LOC</td>
<td>I-LOC</td>
</tr>
</tbody>
</table>

Mayor  DeBlasio  from  New  York

What is the probability of $y_i$, i.e.

$$p(y_i = \delta(c_i) | y_{1:i-1} = \delta(c_{1:i-1}), y_{i+1:n} = \delta(c_{i+1:n}), x)$$
Answer

- Same idea. Score involving $c_i$ are local ($i - 1$ and $i + 1$).

- Can compute “smoothing” distribution from local information

$$ p(y_i = \delta(c_i) | y_{1:i-1}, y_{i+1:n}, x) \propto p(y_i | y_i-1) p(y_{i+1} | y_i) $$

$$ = \hat{y}(c_{i-1})_c \hat{y}(c_i')_{c_{i+1}} $$
Exercise 4

Assume we are not given \( c_{1:i-1} \) and \( c_{i+1:n} \).

What is the best completed sequence, i.e.

\[
p(y_i = \delta(c_i)|x)
\]
Answer: Marginalization

- Similar idea. Score involving $c_i$ are local ($i − 1$ and $i + 1$).

\[ p(y_i = \delta(c'_i)|x) = \sum_{c_{1:i-1}:c_{i+1:n}} p(y_i = \delta(c'_i), y_{1:i-1,i+1:n}|x) \]
\[ = \sum_{c_{1:i-1}} p(y_{1:i-1}|x)p(y_i = \delta(c'_i)|y_{i-1}, x) \times \sum_{c_{i+1:n}} p(y_{i+1}|y_i =, x)p(y_{i+1:n}|x) \]
\[ = \sum_{c_{1:i-1}} \hat{y}(c_{i-1})c'_i \prod_{j=1}^{i-1} \hat{y}(c_{j-1})c_j \times \sum_{c_{i+1:n}} \hat{y}(c'_i)c_{i+1} \prod_{j=i+1}^{n} \hat{y}(c_j)c_{j+1} \]
Similar idea. Score involving $c_i$ are local ($i-1$ and $i+1$).

\[
p(y_i = \delta(c'_i)|x) = \sum_{c_{1:i-1}:c_{i+1:n}} p(y_i = \delta(c'_i), y_{1:i-1,i+1:n}|x) \\
= \sum_{c_{1:i-1}} p(y_{1:i-1}|x) p(y_i = \delta(c'_i)|y_{i-1}, x) \\
\times \sum_{c_{i+1:n}} p(y_{i+1}|y_i = , x) p(y_{i+1:n}|x) \\
= \sum_{c_{1:i-1}} \hat{y}(c_{i-1}) c'_i \prod_{j=1}^{i-1} \hat{y}(c_{j-1}) c_j \\
\times \sum_{c_{i+1:n}} \hat{y}(c'_i) c_{i+1} \prod_{j=i+1}^{n} \hat{y}(c_j) c_{j+1}
\]
Forward and Backward

- **Forward scores (sum over prefixes)**

  \[ \alpha[i, c'_i] = \sum_{c_1:i-1} \hat{y}(c_{i-1}) c'_i \prod_{j=1}^{i-1} \hat{y}(c_{j-1}) c_j \]

- **Backward scores (sum over suffixes)**

  \[ \beta[i, c'_i] = \sum_{c_{i+1:n}} \hat{y}(c'_i) c_{i+1} \prod_{j=i+1}^{n} \hat{y}(c_j) c_{j+1} \]
Forward Algorithm

procedure **Forward**

\( \alpha \in \mathbb{R}^{0,...,n} \times \mathcal{C} \)

\( \alpha[0, \langle s \rangle] = 1 \)

**for** \( i = 1 \) **to** \( n \) **do**

**for** \( c_i \in \mathcal{C} \) **do**

\[ \alpha[i, c_i] = \sum_{c_{i-1}} \alpha[i - 1, c_{i-1}] \times \hat{y}(c_{i-1})c_i \]

**return** \( \alpha \)
procedure Backward

\[ \beta \in \mathbb{R}^{\{1, \ldots, n+1\} \times C} \]

\[ \beta[n + 1, \langle s \rangle] = 1 \]

for \( i = n \) to 1 do

for \( c_i \in C \) do

\[ \beta[i, c_i] = \sum_{c_{i+1}} \beta[i + 1, c_{i+1}] \times \hat{y}(c_i)_{c_{i+1}} \]

return \( \beta \)
Exercise 5: Edge Marginals

What is the probability of using an edge, i.e.

\[ p(y_i = \delta(c'_i), y_{i+1} = \delta(c'_{i+1}), |x) \]
Edge Marginals

\[ \hat{y}(c_i') c_{i+1} \times \sum_{c_{1:i-1}} \hat{y}(c_{i-1}) c'_i \prod_{j=1}^{i-1} \hat{y}(c_{j-1}) c_j \]

\[ \times \sum_{c_{i+2:n}} \hat{y}(c'_{i+1}) c_{i+1} \prod_{j=i+1}^{n} \hat{y}(c_j) c_{j+1} \]

- Compute \( \alpha \) using Forward
- Compute \( \beta \) using Backward
- Multiply in the edge

\[ \hat{y}(c_i') c_{i+1} \alpha[i, c'_i] \times \beta[i + 1, c'_{i+1}] \]
Mayor De Blasio from New York
Marginals versus Max-Marginals

- Max-Marginals: Most-likely sequence through decision
- Marginals: Sum of sequences through decision.
- Possibly very different values.
- Edge with highest marginal may not be in best sequence.
Edge Marginal Decoding

- For all $i$
  $$p(y_i = \delta(c_i), y_{i+1} = \delta(c_{i+1}) | x)$$

- But this is a Markov model!

- Replace lattice with edge marginals
  $$f(x, c_{1:n}) = \sum_i \log p(y_i = \delta(c_i), y_{i+1} = \delta(c_{i+1}) | x)$$

- Posterior decoding.
  $$\arg \max_{c_{1:n}} f(x, c_{1:n})$$