Hidden Markov Models (III): Viterbi Decoding and the Forward-Backward Algorithm

CMSC 473/673

UMBC

October 16th, 2017
Course Announcement: Assignment 2

Due Saturday, 10/21 at 11:59 PM (~5.5 days)

Any questions?
Recap from last time...
Hidden Markov Models: Part of Speech

\[ p(British \ Left \ Waffles \ on \ Falkland \ Islands) \]

(i): Adjective \rightarrow Noun \rightarrow Verb \rightarrow Prep \rightarrow Noun \rightarrow Noun

(ii): Noun \rightarrow Verb \rightarrow Noun \rightarrow Prep \rightarrow Noun \rightarrow Noun

Class-based model

\[ p(w_i|z_i) \]

Bigram model of the classes

\[ p(z_i|z_{i-1}) \]

Model all class sequences

\[ \sum_{z_1,..,z_N} p(z_1, w_1, z_2, w_2, ..., z_N, w_N) \]
Hidden Markov Model Terminology

\[ p(z_1, w_1, z_2, w_2, \ldots, z_N, w_N) = p(z_1 | z_0)p(w_1 | z_1) \cdots p(z_N | z_{N-1})p(w_N | z_N) \]
\[ = \prod_{i=1}^{N} p(w_i | z_i) p(z_i | z_{i-1}) \]

Each \( z_i \) can take the value of one of \( K \) latent states

Transition and emission distributions do not change

**Q:** How many different probability values are there with \( K \) states and \( V \) vocab items?

**A:** \( VK \) emission values and \( K^2 \) transition values

**Q:** How many different paths are there with \( K \) states and \( N \) observed items?

**A:** \( K^N \)

**Goal:**
Find a way to compute with an exponential number of items efficiently
Q: What’s the probability of 
(N, w₁), (V, w₂), (V, w₃), (N, w₄)?

A: 
(0.7 * 0.7) * (0.8 * 0.6) * (0.35 * 0.1) * (0.6 * 0.05) = 0.0002822
Hidden Markov Model Tasks

\[ p(z_1, w_1, z_2, w_2, \ldots, z_N, w_N) = p(z_1 | z_0)p(w_1 | z_1) \cdots p(z_N | z_{N-1})p(w_N | z_N) \]

Calculate the (log) likelihood of an observed sequence \( w_1, \ldots, w_N \)

Calculate the most likely sequence of states (for an observed sequence)

Learn the emission and transition parameters
2 (3)-State HMM Likelihood

Up until here, all the computation was the same

Let’s reuse what computations we can
Forward Probability

\[ \alpha(i, s) = \sum_{s'} \alpha(i - 1, s') \cdot p(s | s') \cdot p(\text{obs at } i | s) \]

\( \alpha(i, s) \) is the total probability of all paths:

1. that start from the beginning
2. that end (currently) in \( s \) at step \( i \)
3. that emit the observation \( \text{obs at } i \)
2 (3) -State HMM Likelihood with Forward Probabilities

\[
\begin{align*}
\alpha[1, V] &= (0.2 \times 0.2) = 0.04 \\
\alpha[1, N] &= (0.7 \times 0.7) = 0.49
\end{align*}
\]

\[
\begin{align*}
\alpha[2, V] &= \alpha[1, V] \times (0.8 \times 0.6) + \\
\alpha[1, N] \times (0.35 \times 0.6) = 0.2436
\end{align*}
\]

\[
\begin{align*}
\alpha[2, N] &= \alpha[1, N] \times (0.15 \times 0.2) + \\
\alpha[1, V] \times (0.6 \times 0.2) = 0.0195
\end{align*}
\]

\[
\begin{align*}
\alpha[3, V] &= \alpha[2, V] \times (0.35 \times 0.1) + \\
\alpha[2, N] \times (0.8 \times 0.1)
\end{align*}
\]

\[
\begin{align*}
\alpha[3, N] &= \alpha[2, N] \times (0.6 \times 0.05) + \\
\alpha[2, V] \times (0.2 \times 0.05)
\end{align*}
\]

<table>
<thead>
<tr>
<th>N</th>
<th>V</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>0.7</td>
<td>0.2</td>
</tr>
<tr>
<td>N</td>
<td>0.15</td>
<td>0.8</td>
</tr>
<tr>
<td>V</td>
<td>0.6</td>
<td>0.35</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>w1</th>
<th>w2</th>
<th>w3</th>
<th>w4</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>0.7</td>
<td>0.2</td>
<td>0.05</td>
</tr>
<tr>
<td>N</td>
<td>0.6</td>
<td>0.2</td>
<td>0.05</td>
</tr>
<tr>
<td>V</td>
<td>0.2</td>
<td>0.6</td>
<td>0.1</td>
</tr>
</tbody>
</table>
2 (3) - State HMM Likelihood with Forward Probabilities

\[
\begin{align*}
\alpha[2, V] &= \alpha[1, N] \times (0.8 \times 0.6) + \\
&\quad \alpha[1, V] \times (0.35 \times 0.6) = 0.2436
\end{align*}
\]

\[
\begin{align*}
\alpha[3, V] &= \alpha[2, V] \times (0.35 \times 0.1) + \\
&\quad \alpha[2, N] \times (0.8 \times 0.1)
\end{align*}
\]

\[
\begin{align*}
\alpha[2, N] &= \alpha[1, N] \times (0.15 \times 0.2) + \\
&\quad \alpha[1, V] \times (0.6 \times 0.2) = 0.0195
\end{align*}
\]

\[
\begin{align*}
\alpha[3, N] &= \alpha[2, V] \times (0.6 \times 0.05) + \\
&\quad \alpha[2, N] \times (0.2 \times 0.05)
\end{align*}
\]
2 (3) - State HMM Likelihood with Forward Probabilities

\[
\begin{align*}
\alpha[2, V] &= \alpha[1, N] \cdot (0.8 \cdot 0.6) + \\
&= \alpha[1, V] \cdot (0.35 \cdot 0.6) = 0.2436 \\
\alpha[3, V] &= \alpha[2, V] \cdot (0.35 \cdot 0.1) + \\
&= \alpha[2, N] \cdot (0.8 \cdot 0.1)
\end{align*}
\]

\[
\begin{align*}
\alpha[2, N] &= \alpha[1, N] \cdot (0.15 \cdot 0.2) + \\
&= \alpha[1, V] \cdot (0.6 \cdot 0.2) = 0.0195 \\
\alpha[3, N] &= \alpha[2, V] \cdot (0.6 \cdot 0.05) + \\
&= \alpha[2, N] \cdot (0.2 \cdot 0.05)
\end{align*}
\]
2 (3) - State HMM Likelihood with Forward Probabilities

\[ \alpha[2, V] = \alpha[1, N] \times (.8 \times .6) + \alpha[1, V] \times (.35 \times .6) = 0.2436 \]
\[ \alpha[3, V] = \alpha[2, V] \times (.35 \times .1) + \alpha[2, N] \times (.8 \times .1) \]
\[ \alpha[3, N] = \alpha[2, V] \times (.6 \times .05) + \alpha[2, N] \times (.2 \times .05) \]

\[ p(V|\text{start}) = \alpha[1, V] \times (1 \times .2 \times .2) = .04 \]
\[ p(w_1|N) = \alpha[1, N] \times (1 \times .7 \times .7) = .49 \]
\[ p(w_2|V) = \alpha[2, V] \times (1 \times .2 \times .2) = .04 \]
\[ p(w_3|N) = \alpha[2, N] \times (1 \times .2 \times .2) = .0195 \]
\[ p(w_4|N) = \alpha[2, N] \times (1 \times .2 \times .2) = .0195 \]

\[
\begin{array}{c|c|c|c|c}
\text{N} & \text{V} & \text{end} \\
\hline
\text{start} & .7 & .2 & .1 \\
\text{N} & .15 & .8 & .05 \\
\text{V} & .6 & .35 & .05 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c}
\text{ } & w_1 & w_2 & w_3 & w_4 \\
\hline
\text{N}\text{start} & .7 & .2 & .05 & .05 \\
\text{N} & .7 & .2 & .05 & .05 \\
\text{V} & .2 & .6 & .1 & .1 \\
\end{array}
\]
Forward Algorithm

\( \alpha \): a 2D table, \((N+2) \times K^*\)

- \(N+2\): number of observations (+2 for the BOS & EOS symbols)
- \(K^*\): number of states

Use dynamic programming to build the \( \alpha \) left-to-right
Forward Algorithm

\[ \alpha = \text{double}[N+2][K^*] \]
\[ \alpha[0][*] = 1.0 \]

for \( i = 1; i \leq N+1; ++i \) {
    for (state = 0; state < K^*; ++state) {
        \[ p_{\text{obs}} = p_{\text{emission}}(\text{obs}_i \mid \text{state}) \]
        for (old = 0; old < K^*; ++old) {
            \[ p_{\text{move}} = p_{\text{transition}}(\text{state} \mid \text{old}) \]
            \[ \alpha[i][\text{state}] += \alpha[i-1][\text{old}] \times p_{\text{obs}} \times p_{\text{move}} \]
        }
    }
}
Forward Algorithm

\[ \alpha = \text{double}[N+2][K^*] \]
\[ \alpha[0][*] = 1.0 \]

for \( i = 1; \ i \leq N+1; \ ++i \) {
    for (state = 0; \ state < K^*; \ ++state) {
        \[ p_{\text{obs}} = p_{\text{emission}}(\text{obs}_i \mid \text{state}) \]
        for (old = 0; \ old < K^*; \ ++old) {
            \[ p_{\text{move}} = p_{\text{transition}}(\text{state} \mid \text{old}) \]
            \[ \alpha[i][\text{state}] += \alpha[i-1][\text{old}] \times p_{\text{obs}} \times p_{\text{move}} \]
        }
    }
}

Q: How do we return the likelihood of the sequence?

A: \[ \alpha[N+1][\text{end}] \]
Forward Algorithm in Log-Space

\( \alpha = \text{double}[N+2][K^*] \)
\( \alpha[0][*] = 0.0 \)

for(\( i = 1; i \leq N+1; ++i \)) {
    for(\( \text{state} = 0; \text{state} < K^*; ++\text{state} \)) {
        \( p_{\text{obs}} = \log p_{\text{emission}}(\text{obs}_i | \text{state}) \)
        for(\( \text{old} = 0; \text{old} < K^*; ++\text{old} \)) {
            \( p_{\text{move}} = \log p_{\text{transition}}(\text{state} | \text{old}) \)
            \( \alpha[i][\text{state}] = \log\text{add}(\alpha[i][\text{state}], \alpha[i-1][\text{old}] + p_{\text{obs}} + p_{\text{move}}) \)
        }
    }
}

\( \text{logadd}(lp, lq) = \begin{cases} 
    lp + \log(1 + \exp(lq - lp)), & \text{if } lp \geq lq \\
    lq + \log(1 + \exp(lp - lq)), & \text{if } lq > lp
\end{cases} \)
Hidden Markov Model Tasks

\[ p(z_1, w_1, z_2, w_2, ..., z_N, w_N) = p(z_1 | z_0)p(w_1 | z_1) \cdots p(z_N | z_{N-1})p(w_N | z_N) = \prod_i p(w_i | z_i) p(z_i | z_{i-1}) \]

Calculate the (log) likelihood of an observed sequence \( w_1, ..., w_N \)

Calculate the most likely sequence of states (for an observed sequence)

Learn the emission and transition parameters
HMM Most-Likely Sequence Task

Maximize over all latent sequence joint likelihoods

$$\max_{z_1, \ldots, z_N} p(z_1, w_1, z_2, w_2, \ldots, z_N, w_N)$$

Q: In a K-state HMM for a length N observation sequence, how many comparisons (different latent sequences) do we make?
HMM Most-Likely Sequence Task

Maximize over all latent sequence joint likelihoods

\[
\max_{z_1, \ldots, z_N} p(z_1, w_1, z_2, w_2, \ldots, z_N, w_N)
\]

Q: In a K-state HMM for a length N observation sequence, how many comparisons (different latent sequences) do we make?

A: \(K^N\)
HMM Most-Likely Sequence Task

Maximize over all latent sequence joint likelihoods

\[
\max_{z_1, \ldots, z_N} p(z_1, w_1, z_2, w_2, \ldots, z_N, w_N)
\]

Q: In a K-state HMM for a length N observation sequence, how many comparisons (different latent sequences) do we make?

A: \(K^N\)

Goal:
Find a way to compute this exponential comparison efficiently (in polynomial time)
HMM Most-Likely Sequence Task

Maximize over all latent sequence joint likelihoods

\[
\max_{z_1, \ldots, z_N} p(z_1, w_1, z_2, w_2, \ldots, z_N, w_N)
\]

Q: In a K-state HMM for a length N observation sequence, how many comparisons (different latent sequences) do we make?

A: \(K^N\)

Goal:
Find a way to compute this exponential comparison efficiently (in polynomial time)
What’s the Maximum?

9  7  6  3  32  4  1
What’s the Maximum?

```
max_val = -\infty
max_index = -1

for(i = 0; i < N; ++i) {
    if(obs[i] > max_val) {
        max_val = obs[i]
        max_index = i
    }
}
return (max_val, max_index)
```
What’s the Maximum?

```
max_val = -\infty
max_index = -1

for(i = 0; i < N; ++i) {
    if(obs[i] > max_val) {
        max_val = obs[i]
        max_index = i
    }
}
return (max_val, max_index)
```
What’s the Maximum?

3

7

9  6

4

32 1
What’s the Maximum?
What’s the Maximum?
What’s the Maximum?
What’s the Maximum?

Q: What “index” do we return?
What’s the Maximum?

Q: What “index” do we return?

A₁: Pointer to node
What’s the Maximum?

Q: What “index” do we return?

A₁: Pointer to node

A₂: Path to node from root (right, left)
What’s the Maximum *Weighted Path*?
What’s the Maximum *Weighted Path*?
What’s the Maximum Weighted Path?
What’s the Maximum Weighted Path?
What’s the Maximum Weighted Path?
What’s the Maximum Weighted Path?
What’s the Maximum \textit{Weighted Path}? 

Q: What “index” do we return?
What’s the Maximum *Weighted* Path?

**Q:** What “index” do we return?

**A₁:** Pointer to list of nodes
What’s the Maximum *Weighted Path*?

**Q:** What “index” do we return?

**A1:** Pointer to list of nodes

**A2:** Path (1, 1, 3)
What’s the Maximum Value?

Consider “any shared path ending with B (AB, BB, or CB) → B”
maximize across the previous hidden state values

\[ v(i, B) = \max_{s'} v(i - 1, s') \times p(B | s') \times p(\text{obs at } i | B) \]

\( v(i, B) \) is the maximum probability of any paths to that state B from the beginning (and emitting the observation)
What’s the Maximum Value?

consider “any shared path ending with B (AB, BB, or CB) → B”
maximize across the previous hidden state values

\[ v(i, B) = \max_{s'} v(i-1, s') \times p(B | s') \times p(\text{obs at } i | B) \]

\( v(i, B) \) is the maximum probability of any paths to that state B from the beginning (and emitting the observation)
What’s the Maximum Value?

consider “any shared path ending with B (AB, BB, or CB) → B”

maximize across the previous hidden state values

computing v at time $i-1$ will correctly incorporate (maximize over) paths through time $i-2$:
we correctly obey the Markov property

$v(i, B) = \max_{s'} v(i - 1, s') \ast p(B \mid s') \ast p(\text{obs at } i \mid B)$

$v(i, B)$ is the maximum probability of any paths to that state B from the beginning (and emitting the observation)
2 (3)-State Viterbi

Up until here, all the computation was the same
Let’s reuse what computations we can
2 (3) -State Viterbi

### Initial States
- \( v[1, V] = (0.2\times0.2) = 0.04 \)
- \( p(V|\text{start}) \)
- \( v[1, N] = (0.7\times0.7) = 0.49 \)
- \( p(N|\text{start}) \)

### Transition Probabilities
- \( p(w_1|N) \)
- \( p(w_2|V) \)
- \( p(w_3|V) \)
- \( p(w_4|N) \)

### Emission Probabilities
- \( p(V) \)
- \( p(N) \)

### Viterbi Algorithm Steps
1. \( v[1, V] = \max\{v[1, N] \times (0.8\times0.6), v[1, V] \times (0.35\times0.6)\} = 0.2352 \)
2. \( v[2, V] = \max\{v[2, V] \times (0.35\times0.1), v[2, N] \times (0.8\times0.1)\} \)
3. \( v[3, V] = \max\{v[2, V] \times (0.6\times0.05), v[2, N] \times (0.2\times0.05)\} \)

### Transition Matrix

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>V</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>0.7</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>N</td>
<td>0.15</td>
<td>0.8</td>
<td>0.05</td>
</tr>
<tr>
<td>V</td>
<td>0.6</td>
<td>0.35</td>
<td>0.05</td>
</tr>
</tbody>
</table>

### Observation Sequence

| w | \( p(w_1|N) \) | \( p(w_2|V) \) | \( p(w_3|V) \) | \( p(w_4|N) \) |
|---|---|---|---|---|
| N | 0.7 | 0.2 | 0.05 | 0.05 |
| V | 0.2 | 0.6 | 0.1 | 0.1 |
2 (3) -State Viterbi

Use dynamic programming to build the v left-to-right

\[
v[1, V] = \max\{v[1, N] \times (0.7 \times 0.7), v[1, V] \times (0.35 \times 0.6)\} = 0.49
\]

\[
v[2, V] = \max\{v[2, V] \times (0.35 \times 0.1), v[2, N] \times (0.8 \times 0.1)\}
\]

\[
v[3, N] = \max\{v[2, V] \times (0.6 \times 0.05), v[2, N] \times (0.2 \times 0.05)\}
\]

\[
v[3, V] = \max\{v[2, V] \times (0.35 \times 0.1), v[2, N] \times (0.8 \times 0.1)\}
\]

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>V</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>.7</td>
<td>.2</td>
<td>.1</td>
</tr>
<tr>
<td>N</td>
<td>.15</td>
<td>.8</td>
<td>.05</td>
</tr>
<tr>
<td>V</td>
<td>.6</td>
<td>.35</td>
<td>.05</td>
</tr>
</tbody>
</table>
2 (3) -State Viterbi

$$v[1, V] = \max\{v[1, N] \cdot (.7 \cdot .6), v[1, V] \cdot (.35 \cdot .6)\} = .49$$
$$v[1, N] = (.7 \cdot .7) = .49$$
$$p(V|\text{start})$$
$$p(N|\text{start})$$

$$v[2, V] = \max\{v[2, V] \cdot (.35 \cdot .1), v[2, N] \cdot (.8 \cdot .1)\}$$
$$v[3, V] = \max\{v[2, V] \cdot (.35 \cdot .1), v[2, N] \cdot (.8 \cdot .1)\}$$
$$v[3, N] = \max\{v[2, V] \cdot (.6 \cdot .05), v[2, N] \cdot (.2 \cdot .05)\}$$

Keep backpointers: record the state that produced the maximum.
Viterbi Algorithm

\[ \mathbf{v} = \text{double}[N+2][K^*] \]
\[ \mathbf{b} = \text{int}[N+2][K^*] \]

\[ \mathbf{v}[0][*] = 0 \]
\textbf{Viterbi Algorithm}

\vspace{1em}

\begin{verbatim}
\textbf{v} = \text{double}[\text{N+2}][\text{K}^*]
\textbf{b} = \text{int}[\text{N+2}][\text{K}^*]

\textbf{v}[0][*] = 0

\textbf{for}(i = 1; i \leq \text{N+1}; ++i) \{ \\
    \textbf{for}(\text{state} = 0; \text{state} < \text{K}^*; ++\text{state}) \{ \\
    \textbf{\}}
\textbf{\}}
\end{verbatim}
Viterbi Algorithm

\[ v = \text{double}[N+2][K^*] \]
\[ b = \text{int}[N+2][K^*] \]

\[ v[0][*] = 0 \]

for \( i = 1; i \leq N+1; ++i \) {
    for (state = 0; state < K^*; ++state) {
        \[ p_{\text{obs}} = p_{\text{emission}}(\text{obs}_i \mid \text{state}) \]
        for (old = 0; old < K^*; ++old) {
            \[ p_{\text{move}} = p_{\text{transition}}(\text{state} \mid \text{old}) \]
        }
    }
}

\[ b = \text{int}[N+2][K^*] \]

\[ b[0][*] = 0 \]

for \( i = 1; i \leq N+1; ++i \) {
    for (state = 0; state < K^*; ++state) {
        \[ b[i][state] = \max \{ v[i-1][old] \cdot p_{\text{move}}(\text{state} \mid \text{old}) \mid 0 \leq old < K^* \} \]
    }
}

\[ \text{path}(N) = \arg \max_{0 \leq state < K^*} v[N][state] \]

\[ \text{path}(i) = \text{argmax}_{0 \leq state < K^*} \{ \text{prev}(i) \mid b[i][state] \}, 1 \leq i \leq N \]
v = double[N+2][K*]
b = int[N+2][K*]

v[0][*] = 0

for(i = 1; i ≤ N+1; ++i) {
    for(state = 0; state < K*; ++state) {
        pobs = p_emission(obs_i | state)
        for(old = 0; old < K*; ++old) {
            pmove = p_transition(state | old)
            if(v[i-1][old] * pobs * pmove > v[i][state]) {
                v[i][state] = v[i-1][old] * pobs * pmove
                b[i][state] = old
            }
        }
    }
}
Viterbi Algorithm

\[ v = \text{double}[N+2][K^*] \]
\[ b = \text{int}[N+2][K^*] \]

\[ v[0][*] = 0 \]

for \( i = 1; i \leq N+1; ++i \) {
    for(state = 0; state < K*; ++state) {
        pobs = p\text{emission}(\text{obs}_i \mid \text{state})
        for(old = 0; old < K*; ++old) {
            pmove = p\text{transition}(\text{state} \mid \text{old})
            if(\[v[i-1][old] \times pobs \times pmove > v[i][state]\]) {
                v[i][state] = v[i-1][old] \times pobs \times pmove
                b[i][state] = old
            }
        }
    }
}

Q: How do we return the most likely tag sequence?
Viterbi Algorithm

\[ v = \text{double}[N+2][K^*] \]
\[ b = \text{int}[N+2][K^*] \]

\[ v[0][*] = 0 \]

for \( i = 1; i \leq N+1; ++i \) {
  for (state = 0; state < K^*; ++state) {
    \( p_{obs} = p_{\text{emission}}(\text{obs}_i | \text{state}) \)
    for (old = 0; old < K^*; ++old) {
      \( p_{move} = p_{\text{transition}}(\text{state} | \text{old}) \)
      if \( v[i-1][old] \times p_{obs} \times p_{move} > v[i][state] \) {
        \( v[i][state] = v[i-1][old] \times p_{obs} \times p_{move} \)
        \( b[i][state] = old \)
      }
    }
  }
}

Q: How do we return the most likely tag sequence?

A: \((t_i)_i\), where
\[ t_{i-1} = b[i][t_i] \]
Viterbi Algorithm in Log-Space

\[ v = \text{double}[N+2][K^*] \]
\[ b = \text{int}[N+2][K^*] \]

\[ v[0][*] = -\infty \]

for \( i = 1; i \leq N+1; ++i \) {
    for \( \text{state} = 0; \text{state} < K^*; ++\text{state} \) {
        \[ p_{\text{obs}} = \log p_{\text{emission}}(\text{obs}_i \mid \text{state}) \]
        for \( \text{old} = 0; \text{old} < K^*; ++\text{old} \) {
            \[ p_{\text{move}} = \log p_{\text{transition}}(\text{state} \mid \text{old}) \]
            if \( v[i-1][\text{old}] + p_{\text{obs}} + p_{\text{move}} > v[i][\text{state}] \) {
                \[ v[i][\text{state}] = v[i-1][\text{old}] + p_{\text{obs}} + p_{\text{move}} \]
                \[ b[i][\text{state}] = \text{old} \]
            }
        }
    }
}
Forward vs. Viterbi

\[ \alpha = \text{double}[N+2][K^*] \]

\[ \alpha[0][*] = 1.0 \]

for \((i = 1; i \leq N+1; ++i)\) {
    for (state = 0; state < K*; ++state) {
        \[ p_{obs} = p_{emission}(\text{obs}_i \mid \text{state}) \]
        for (old = 0; old < K*; ++old) {
            \[ p_{move} = p_{transition}(\text{state} \mid \text{old}) \]
            \[ \alpha[i][state] += \alpha[i-1][old] \ast p_{obs} \ast p_{move} \]
        }
    }
}

\[ v = \text{double}[N+2][K^*] \]

\[ b = \text{int}[N+2][K^*] \]

\[ v[0][*] = 0 \]

for \((i = 1; i \leq N+1; ++i)\) {
    for (state = 0; state < K*; ++state) {
        \[ p_{obs} = p_{emission}(\text{obs}_i \mid \text{state}) \]
        for (old = 0; old < K*; ++old) {
            \[ p_{move} = p_{transition}(\text{state} \mid \text{old}) \]
            if (\[ v[i-1][old] \ast p_{obs} \ast p_{move} > v[i][state] \]) {
                \[ v[i][state] = v[i-1][old] \ast p_{obs} \ast p_{move} \]
                \[ b[i][state] = old \]
            }
        }
    }
}
Forward vs. Viterbi

\[ \alpha = \text{double}[N+2][K]\]

\[ \mathbf{v} = \text{double}[N+2][K]\]

\[ b = \text{int}[N+2][K]\]

\[ \alpha[0][*] = 1.0 \]

\[ \mathbf{v}[0][*] = 0 \]

for (i = 1; i \leq N+1; ++i) {
    for (state = 0; state < K; ++state) {
        p_{obs} = p_{\text{emission}}(\text{obs}_i | \text{state})
        for (old = 0; old < K; ++old) {
            p_{move} = p_{\text{transition}}(\text{state} | \text{old})
            \alpha[i][\text{state}] += \alpha[i-1][\text{old}] * p_{obs} * p_{move}
        }
    }
}

for (i = 1; i \leq N+1; ++i) {
    for (state = 0; state < K; ++state) {
        p_{obs} = p_{\text{emission}}(\text{obs}_i | \text{state})
        for (old = 0; old < K; ++old) {
            p_{move} = p_{\text{transition}}(\text{state} | \text{old})
            if (\mathbf{v}[i-1][\text{old}] * p_{obs} * p_{move} > \mathbf{v}[i][\text{state}]) {
                \mathbf{v}[i][\text{state}] = \mathbf{v}[i-1][\text{old}] * p_{obs} * p_{move}
                b[i][\text{state}] = old
            }
        }
    }
}
Hidden Markov Model Tasks

\[ p(z_1, w_1, z_2, w_2, \ldots, z_N, w_N) = p(z_1 | z_0)p(w_1 | z_1) \cdots p(z_N | z_{N-1})p(w_N | z_N) \]

Calculate the (log) likelihood of an observed sequence \( w_1, \ldots, w_N \)

Calculate the most likely sequence of states (for an observed sequence)

Learn the emission and transition parameters
“The farther backward you can look, the farther forward you can see.”

commonly attributed to Winston Churchill
HMM Probabilities

Forward Values

$$\alpha(i, s) = \sum_{s'} \alpha(i - 1, s') \cdot p(s \mid s') \cdot p(\text{obs at } i \mid s)$$

$$\alpha(i, s)$$ is the total probability of all paths:
1. that start from the beginning
2. that end (currently) in $$s$$ at step $$i$$
3. that emit the observation $$\text{obs at } i$$
HMM Probabilities

**Forward Values**

\[
\alpha(i, s) = \sum_{s'} \alpha(i - 1, s') \times p(s' | s') \times p(\text{obs at } i | s)
\]

\(\alpha(i, s)\) is the total probability of all paths:

1. that start from the beginning
2. that end (currently) in \(s\) at step \(i\)
3. that emit the observation \(\text{obs at } i\)

**Backward Values**

\[
\beta(i, s) = \sum_{s'} \beta(i + 1, s') \times p(s' | s) \times p(\text{obs at } i + 1 | s')
\]

\(\beta(i, s)\) is the total probability of all paths:

1. that start at step \(i\) at state \(s\)
2. that terminate at the end
3. \((that \ emit \ the \ observation \ \text{obs at } i+1)\)
Backward Algorithm

\( \beta \): a 2D table, \((N+2) \times K^*\)

- \(N+2\): number of observations (+2 for the BOS & EOS symbols)
- \(K^*\): number of states

Use dynamic programming to build the \( \beta \) right-to-left
Backward Algorithm

\( \beta = \text{double}[N+2][K^*] \)
\( \beta[n+1][*] = 1.0 \)

\[
\text{for}(i = N; \ i \geq 0; \ --i) \ { \\
\quad \text{for}(\text{next} = 0; \ \text{next} < K^*; \ ++\text{next}) \ { \\
\quad \quad p_{\text{obs}} = p_{\text{emission}}(\text{obs}_{i+1} | \ \text{next}) \ \\
\quad \quad \text{for}(\text{state} = 0; \ \text{state} < K^*; \ ++\text{state}) \ { \\
\quad \quad \quad p_{\text{move}} = p_{\text{transition}}(\text{next} | \ \text{state}) \ \\
\quad \quad \quad \beta[i][\text{state}] += \beta[i+1][\text{next}] \times p_{\text{obs}} \times p_{\text{move}} \ \\
\quad \quad \} \ \\
\quad \} \ \\
\} \]
**Backward Algorithm**

\[ \beta = \text{double}[N+2][K^*] \]

\[ \beta[n+1][*] = 1.0 \]

for \( i = N; i \geq 0; --i \) {
    for \( \text{next} = 0; \text{next} < K^*; ++\text{next} \) {
        \[ p_{\text{obs}} = p_{\text{emission}}(\text{obs}_{i+1} | \text{next}) \]
        for \( \text{state} = 0; \text{state} < K^*; ++\text{state} \) {
            \[ p_{\text{move}} = p_{\text{transition}}(\text{next} | \text{state}) \]
            \[ \beta[i][\text{state}] += \beta[i+1][\text{next}] \times p_{\text{obs}} \times p_{\text{move}} \]
        }
    }
}

**Q: What does \( \beta[0][\text{START}] \) represent?**
Backward Algorithm

\[ \beta = \text{double}[N+2][K^*] \]
\[ \beta[n+1][*] = 1.0 \]

for \( i = N; i \geq 0; --i \) {
  for (next = 0; next < K^*; ++next) {
    for (state = 0; state < K^*; ++state) {
      p_{\text{obs}} = p_{\text{emission}}(\text{obs}_{i+1} \mid \text{next})
      p_{\text{move}} = p_{\text{transition}}(\text{next} \mid \text{state})
      \beta[i][state] += \beta[i+1][next] * p_{\text{obs}} * p_{\text{move}}
    }
  }
}

**Q:** What does \( \beta[0][\text{START}] \) represent?

**A:** Total probability of all paths from stop to start, for the observed sequence.
Backward Algorithm

\[ \beta = \text{double}[N+2][K^*] \]
\[ \beta[n+1][*] = 1.0 \]

for \( i = N; i \geq 0; --i \) {
    for (next = 0; next < K*; ++next) {
        \[ p_{\text{obs}} = p_{\text{emission}}(\text{obs}_{i+1} \mid \text{next}) \]
        for (state = 0; state < K*; ++state) {
            \[ p_{\text{move}} = p_{\text{transition}}(\text{next} \mid \text{state}) \]
            \[ \beta[i][\text{state}] += \beta[i+1][\text{next}] \times p_{\text{obs}} \times p_{\text{move}} \]
        }
    }
}

Q: What does \( \beta[0][\text{START}] \) represent?

A: The marginal likelihood of the observed sequence
Backward Algorithm

\[ \beta = \text{double}[N+2][K^*] \]
\[ \beta[n+1][*] = 1.0 \]

for (i = N; i \geq 0; --i) {
  for (next = 0; next < K^*; ++next) {
    p_{obs} = p_{\text{emission}}(\text{obs}_{i+1} \mid \text{next})
    for (state = 0; state < K^*; ++state) {
      p_{move} = p_{\text{transition}}(\text{next} \mid \text{state})
      \beta[i][state] += \beta[i+1][next] \times p_{obs} \times p_{move}
    }
  }
}

Q: What does \[ \beta[0][\text{START}] \] represent?

A: \[ \alpha[N+1][\text{END}] \]
2 (3) -State HMM Likelihood with Backward Probabilities

\[ p(V|N) \]
\[ p(N|V) \]

\[ \beta[1, V] = \beta[2, N] \times (.6 \times .2) + \beta[2, V] \times (.35 \times .6) \]
\[ \beta[1, N] = \beta[2, N] \times (.15 \times .2) + \beta[2, V] \times (.8 \times .6) \]

\[ \beta[2, V] = \beta[3, N] \times (.6 \times .05) + \beta[3, V] \times (.35 \times .1) \]
\[ \beta[2, N] = \beta[3, N] \times (.15 \times .05) + \beta[3, V] \times (.8 \times .1) \]

\[ \beta[3, V] = \beta[4, V] \times (.35 \times .1) + \beta[4, N] \times (.6 \times .05) \]
\[ \beta[3, N] = \beta[4, V] \times (.8 \times .1) + \beta[4, N] \times (.15 \times .05) \]

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>V</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>.7</td>
<td>.2</td>
<td>.1</td>
</tr>
<tr>
<td>N</td>
<td>.15</td>
<td>.8</td>
<td>.05</td>
</tr>
<tr>
<td>V</td>
<td>.6</td>
<td>.35</td>
<td>.05</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>w1</th>
<th>w2</th>
<th>w3</th>
<th>w4</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>.7</td>
<td>.2</td>
<td>.05</td>
</tr>
<tr>
<td>V</td>
<td>.6</td>
<td>.6</td>
<td>.1</td>
</tr>
</tbody>
</table>
Why Do We Need Backward Values?

\[ \alpha(i, s) \] is the total probability of all paths:
1. that start from the beginning
2. that end (currently) in \( s \) at step \( i \)
3. that emit the observation \( \text{obs at } i \)

\[ \beta(i, s) \] is the total probability of all paths:
1. that start at step \( i \) at state \( s \)
2. that terminate at the end
3. (that emit the observation \( \text{obs at } i+1 \))
Why Do We Need Backward Values?

\[ \alpha(i, s) \] is the total probability of all paths:
1. that start from the beginning
2. that end (currently) in \(s\) at step \(i\)
3. that emit the observation \(\text{obs at } i\)

\[ \beta(i, s) \] is the total probability of all paths:
1. that start at step \(i\) at state \(s\)
2. that terminate at the end
3. (that emit the observation \(\text{obs at } i+1\))
Why Do We Need Backward Values?

\[ \alpha(i, B) \] and \[ \beta(i, B) \] are defined as:

\[ \alpha(i, s) \] is the total probability of all paths:
1. that start from the beginning
2. that end (currently) in \( s \) at step \( i \)
3. that emit the observation \( \text{obs} \) at \( i \)

\[ \beta(i, s) \] is the total probability of all paths:
1. that start at step \( i \) at state \( s \)
2. that terminate at the end
3. (that emit the observation \( \text{obs} \) at \( i+1 \))

\[ \alpha(i, B) * \beta(i, B) = \text{total probability of paths through state } B \text{ at step } i \]
Why Do We Need Backward Values?

\[ \alpha(i, B) \] 
\[ \beta(i, B) \]

\[ \alpha(i, s) * \beta(i, s) = \text{total probability of paths through state } s \text{ at step } i \]

\( \alpha(i, s) \) is the total probability of all paths:
1. that start from the beginning
2. that end (currently) in \( s \) at step \( i \)
3. that emit the observation \( \text{obs at } i \)

\( \beta(i, s) \) is the total probability of all paths:
1. that start at step \( i \) at state \( s \)
2. that terminate at the end
3. (that emit the observation \( \text{obs at } i+1 \) )

we can compute posterior state probabilities (normalize by marginal likelihood)
Why Do We Need Backward Values?

\[ \alpha(i, B) \]

\[ \beta(i+1, s) \]

\( \alpha(i, s) \) is the total probability of all paths:
1. that start from the beginning
2. that end (currently) in \( s \) at step \( i \)
3. that emit the observation \( \text{obs at } i \)

\( \beta(i, s) \) is the total probability of all paths:
1. that start at step \( i \) at state \( s \)
2. that terminate at the end
3. (that emit the observation \( \text{obs at } i+1 \))
Why Do We Need Backward Values?

\[ \alpha(i, B) \] is the total probability of all paths:
1. that start from the beginning
2. that end (currently) in \( s \) at step \( i \)
3. that emit the observation \( \text{obs at } i \)

\[ \beta(i+1, s') \] is the total probability of all paths:
1. that start at step \( i \) at state \( s \)
2. that terminate at the end
3. \( (that \ emit \ the \ observation \ \text{obs at } i+1) \)
Why Do We Need Backward Values?

\[ \alpha(i, B) \] is the total probability of all paths:
1. that start from the beginning
2. that end (currently) in \( s \) at step \( i \)
3. that emit the observation \( \text{obs at } i \)

\[ \beta(i, s') \] is the total probability of all paths:
1. that start at step \( i \) at state \( s \)
2. that terminate at the end
3. (that emit the observation \( \text{obs at } i+1 \))

\[ \alpha(i, B) \times p(s' | B) \times p(\text{obs at } i+1 | s') \times \beta(i+1, s') = \text{total probability of paths through the } B \rightarrow s' \text{ arc (at time } i) \]
With Both Forward and Backward Values

$\alpha(i, s) * \beta(i, s) = \text{total probability of paths through state } s \text{ at step } i$

$\alpha(i, s) * p(s' | B) * p(\text{obs at } i+1 | s') * \beta(i+1, s') = \text{total probability of paths through the } s \rightarrow s' \text{ arc (at time } i)
With Both Forward and Backward Values

\[ \alpha(i, s) \times \beta(i, s) = \text{total probability of paths through state } s \text{ at step } i \]

\[ p(z_i = s \mid w_1, \ldots, w_N) = \frac{\alpha(i, s) \times \beta(i, s)}{\alpha(N + 1, \text{END})} \]

\[ \alpha(i, s) \times p(s' \mid B) \times p(\text{obs at } i+1 \mid s') \times \beta(i+1, s') = \text{total probability of paths through the } s \rightarrow s' \text{ arc (at time } i) \]
With Both Forward and Backward Values

\[ \alpha(i, s) \times \beta(i, s) = \text{total probability of paths through state } s \text{ at step } i \]

\[ p(z_i = s \mid w_1, \ldots, w_N) = \frac{\alpha(i, s) \times \beta(i, s)}{\alpha(N + 1, \text{END})} \]

\[ \alpha(i, s) \times p(s' \mid B) \times p(\text{obs at } i+1 \mid s') \times \beta(i+1, s') = \text{total probability of paths through the } s \rightarrow s' \text{ arc (at time } i) \]

\[ p(z_i = s, z_{i+1} = s' \mid w_1, \ldots, w_N) = \frac{\alpha(i, s) \times p(s' \mid s) \times p(\text{obs}_{i+1} \mid s') \times \beta(i + 1, s')}{\alpha(N + 1, \text{END})} \]
Expectation Maximization (EM)

0. Assume *some* value for your parameters

Two step, iterative algorithm

1. E-step: count under uncertainty, assuming these parameters

\[ p(z_i) \rightarrow \text{count}(z_i, w_i) \]

2. M-step: maximize log-likelihood, assuming these uncertain counts

\[ p^{(t)}(z) \leftrightarrow \text{estimated counts} \rightarrow p^{(t+1)}(z) \]
Expectation Maximization (EM)

0. Assume *some* value for your parameters

Two step, iterative algorithm

1. E-step: count under uncertainty, assuming these parameters

   \[ p(z_i) \xrightarrow{\text{count}} \text{count}(z_i, w_i) \]

2. M-step: maximize log-likelihood, assuming these uncertain counts

   \[ p^{(t)}(z) \xrightarrow{\text{estimated counts}} p^{(t+1)}(z) \]
Expectation Maximization (EM)

0. Assume *some* value for your parameters

Two step, iterative algorithm

1. E-step: count under uncertainty, assuming these parameters

\[
p^*(z_i = s | w_1, \ldots, w_N) = \frac{\alpha(i, s) \cdot \beta(i, s)}{\alpha(N + 1, \text{END})}
\]

\[
p^*(z_i = s, z_{i+1} = s' | w_1, \ldots, w_N) = \frac{\alpha(i, s) \cdot p(s'|s) \cdot p(\text{obs}_{i+1} | s') \cdot \beta(i + 1, s')}{\alpha(N + 1, \text{END})}
\]

\[p(z_i) \quad \rightarrow \quad \text{count}(z_i, w_i)\]

2. M-step: maximize log-likelihood, assuming these uncertain counts

\[p^{(t)}(z) \quad \leftrightarrow \quad \text{estimated counts} \quad \rightarrow \quad p^{(t+1)}(z)\]
M-Step

“maximize log-likelihood, assuming these uncertain counts”

\[ p^{\text{new}}(s'|s) = \frac{c(s \rightarrow s')}{\sum_x c(s \rightarrow x)} \]

if we observed the hidden transitions...
M-Step

“maximize log-likelihood, assuming these uncertain counts”

\[
p^{\text{new}}(s' | s) = \frac{\mathbb{E}_{s \to s'}[c(s \to s')]}{\sum_x \mathbb{E}_{s \to x}[c(s \to x)]}
\]

we don’t the hidden transitions, but we can approximately count
M-Step

“maximize log-likelihood, assuming these uncertain counts”

\[
p_{\text{new}}(s' | s) = \frac{\mathbb{E}_{s \rightarrow s'}[c(s \rightarrow s')]}{\sum_x \mathbb{E}_{s \rightarrow x}[c(s \rightarrow x)]}
\]

we don’t the hidden transitions, but we can approximately count

we compute these in the E-step, with our \( \alpha \) and \( \beta \) values
\[
\alpha = \text{computeForwards}() \\
\beta = \text{computeBackwards}() \\
L = \alpha[N+1][\text{END}] \\
\]

\[
\text{for}(i = N; i \geq 0; --i) \{ \\
\quad \text{for}(\text{next} = 0; \text{next} < K^*; ++\text{next}) \{ \\
\quad \quad c_{\text{obs}}(\text{obs}_{i+1} | \text{next}) += \alpha[i+1][\text{next}] \times \beta[i+1][\text{next}] / L \\
\quad \quad \text{for}(\text{state} = 0; \text{state} < K^*; ++\text{state}) \{ \\
\quad \quad \quad u = p_{\text{obs}}(\text{obs}_{i+1} | \text{next}) \times p_{\text{trans}}(\text{next} | \text{state}) \\
\quad \quad \quad c_{\text{trans}}(\text{next} | \text{state}) += \alpha[i][\text{state}] \times u \times \beta[i+1][\text{next}] / L \\
\quad \quad \} \\
\quad \} \\
\} \\
\]
0. Assume *some* value for your parameters

Two step, iterative algorithm

1. **E-step**: count under uncertainty, assuming these parameters

   \[ p^*(z_i = s | w_1, ..., w_N) = \frac{\alpha(i, s) \cdot \beta(i, s)}{\alpha(N + 1, \text{END})} \]

   \[ p(z_i) \rightarrow \text{count}(z_i, w_i) \]

2. **M-step**: maximize log-likelihood, assuming these uncertain counts

   \[ p^*(z_i = s, z_{i+1} = s' | w_1, ..., w_N) = \frac{\alpha(i, s) \cdot p(s'|s) \cdot p(\text{obs}_{i+1} | s') \cdot \beta(i + 1, s')}{\alpha(N + 1, \text{END})} \]

Baum-Welch