How to Calculate $P(O)$

Calculating $P(O)$

Forward Algorithm

Backward Algorithm

Forward-Backward Algorithm

Calculating $P(O)$

Let $O = (\sigma_{k_1}, \ldots, \sigma_{k_T})$ and let $S = (s_1, \ldots, s_n)$. Then

$$P(O | S) = \prod_{t=1}^{T} P(\eta_{t} | \xi_{t} = s_{t}) = \prod_{t=1}^{T} a_{k_{t}k_{t}}$$

$$P(S) = v_{i_{1}} \cdot \prod_{t=2}^{T} p_{k_{t-1}k_{t}}$$

$$P(O \cap S) = P(O | S) \cdot P(S) = \left( \prod_{t=1}^{T} a_{k_{t}k_{t}} \right) \cdot \left( v_{i_{1}} \cdot \prod_{t=2}^{T} p_{k_{t-1}k_{t}} \right) = (a_{i_{1}k_{1}} \cdot v_{i_{1}}) \cdot \left( \prod_{t=2}^{T} p_{k_{t-1}k_{t}} \right)$$

Trellis

We can visualize the forward variable by a trellis, i.e., a graph with a node for each state-time pair, connected only to nodes at $t - 1$ and $t + 1$.

The forward algorithm (1)

We reshuffle the expression and introduce a set of accumulators, so-called forward variables, one for each time $t$ and state $i$:

$$\alpha_t(i) = P(O_{<t} \cap \xi_t = s_i) = P(\eta_1 = \sigma_{k_1}, \ldots, \eta_t = \sigma_{k_t}; \xi_t = s_i)$$

This is the joint probability of being in state $s_i$ at time $t$ and the observed signal sequence $O_{<t} = \sigma_{k_1}, \ldots, \sigma_{k_t}$ from time 1 to time $t$.

The forward algorithm (2)

Note that:

$$P(O) = P(\eta_1 = \sigma_{k_1}, \ldots, \eta_T = \sigma_{k_T}) =$$

$$= \sum_{i_1=1}^{n} P(\eta_1 = \sigma_{k_1}, \ldots, \eta_T = \sigma_{k_T}; \xi_T = s_i) =$$

$$= \sum_{i_1=1}^{n} \alpha_T(i)$$

and that:

$$\alpha_t(i) = P(\eta_1 = \sigma_{k_1}; \xi_t = s_i) = a_{k_0i} \cdot v_i \quad \text{for} \quad i = 1, \ldots, n$$

and that:

$$\alpha_{t+1}(j) = \left[ \sum_{i=1}^{n} \alpha_t(i) \cdot p_{j} \right] a_{k_{t+1}j} \quad \text{for} \quad t = 1, \ldots, T - 1; j = 1, \ldots, n$$
The forward algorithm (3)

since

\[
P(O_{t+1}; s_t = s_j) = \sum_{i=1}^{n} P(O_{t}; s_i) \cdot P(\eta_{t+1} = \sigma_{k_{t+1}}; \xi_{t+1} = s_j | O_{t}; \xi_t = s_i)
\]

\[
= \sum_{i=1}^{n} P(O_{t}; s_i) \cdot P(\eta_{t+1} = \sigma_{k_{t+1}}; \xi_{t+1} = s_j) \cdot P(\xi_{t+1} = s_j | \xi_t = s_i)
\]

The backward algorithm (1)

Alternatively, we can define the set of backward variables:

\[
\beta_t(i) = P(O; \xi_t = s_i) = P(\eta_t = \sigma_{k_t}; \ldots; \eta_T = \sigma_{k_T} | \xi_t = s_i)
\]

Note that

\[
P(O) = \sum_{i=1}^{n} P(\eta_1 = \sigma_{k_1}, \xi_1 = s_i) \cdot P(\eta_2 = \sigma_{k_2}, \ldots; \eta_T = \sigma_{k_T} | \xi_1 = s_i)
\]

\[
= \sum_{i=1}^{n} \alpha_t(i) \cdot \beta_t(i)
\]

The backward algorithm (2)

Let us define

\[
\beta_T(i) = 1 \quad \text{for } i = 1, \ldots, n
\]

and note that

\[
\beta_t(i) = \sum_{j=1}^{n} P_{ij} \cdot \beta_{t+1}(j) \quad \text{for } t = 1, \ldots, T-1; i = 1, \ldots, n
\]

The backward algorithm (3)

since

\[
P(O; \xi_t = s_i) = \sum_{j=1}^{n} P(O_{t}; \xi_{t+1} = s_j | \xi_t = s_i)
\]

\[
= \sum_{j=1}^{n} P(O_{t}; \xi_t = s_i, \xi_{t+1} = s_j) \cdot P(\xi_{t+1} = s_j | \xi_t = s_i)
\]

\[
= \sum_{j=1}^{n} P(\eta_{t+1} = \sigma_{k_{t+1}}; \xi_{t+1} = s_j) \cdot P(O_{t+1}; \xi_{t+1} = s_j)
\]

\[
\cdot P(\xi_{t+1} = s_j | \xi_t = s_i)
\]
The forward-backward algorithm (1)

\[ P(O) = \sum_{i=1}^{n} P(O; \xi_t = s_i) \]
\[ = \sum_{i=1}^{n} P(O_{\leq t}; \xi_t = s_i) \cdot P(O_{> t}; \xi_t = s_i) \]
\[ = \sum_{i=1}^{n} P(O_{\leq t}; \xi_t = s_i) \cdot P(O_{> t} | \xi_t = s_i) \]
\[ = \sum_{i=1}^{n} \alpha_t(i) \cdot \beta_t(i) \]

Calculating the Most Likely State Sequence

- **Task:**
  Find the most likely state sequence \( S \) for an observation sequence \( O \).

- **Method:**
  Use dynamic programming (Viterbi algorithm), using a trellis

\[ \delta_t(i) = \max_{s_{t-1}} P(S_{t-1}; \xi_t = s_i; O_{\leq t}) = \max_{s_{t-1}, \ldots, s_{t-1}} P(\xi_1 = s_{t-1}, \ldots, \xi_t = s_i; O_{\leq t}) \]

Joint probability of most likely state sequence from time 1 to time \( T \) ending in state \( s_i \) and of the observed signal sequence \( O_{\leq T} \):

\[ P^* = \max_i \delta_T(i) \]

The forward-backward algorithm (2)

We can define a set of forward-backward variables:

\[ \gamma_t(i) = P(\xi_t = s_i; O) = \frac{P(O; \xi_t = s_i)}{P(O)} = \frac{\alpha_t(i) \cdot \beta_t(i)}{\sum_{i=1}^{n} \alpha_t(i) \cdot \beta_t(i)} \]

This is the probability of being in state \( s_i \) at time \( t \) conditional on the entire observed signal sequence \( O \) from time 1 to time \( T \).

What is in the Trellis?

- **Trellis stores at time** \( t \) **for each state** \( i \) the maximum value of all possible transitions from any other state, based on:
  - the path probability ending at \( t - 1 \),
  - the transition probability between source and target state
  - the emission probability of the \( t \)-th segment of the observation sequence for state \( i \)
- **Store a backtrace** that records the node of the incoming arc that leads to most probable state transition at \( t \).

The Viterbi algorithm (1)

\[ \delta_t(i) = \max_{s_{t-1}} P(S_{t-1}; \xi_t = s_i; O_{\leq t}) = \max_{s_{t-1}, \ldots, s_{t-1}} P(\xi_1 = s_{t-1}, \ldots, \xi_t = s_i; O_{\leq t}) \]

Initializatin:

\[ \delta_1(i) = \nu_i \cdot a_{i, i} \text{ for } i = 1, \ldots, n \]

Induction:

\[ \delta_t(j) = \max_i \delta_{t-1}(i) \cdot p_{ij} \cdot \delta_{t-1} \text{ for } t = 2, \ldots, T; j = 1, \ldots, n \]

Store a Backtrace:

\[ \psi_t(j) = \arg \max_i (\delta_{t-1}(i) \cdot p_{ij}) \text{ for } t = 2, \ldots, T; j = 1, \ldots, n \]
### The Viterbi algorithm (3)

**Task:** Extracting the most likely state sequence out of the trellis

**Method:** Working backwards through the backpointers

Ending state $s_t$ of most likely state sequence from time 1 to time $T$ for observed signal sequence $O_{1:T}$:

$$s^*_t = \arg \max_{1 \leq s \leq N} \delta_T(i)$$

Working backwards: Ending state $s_t$ of most likely state sequence from time 1 to time $t$ for observed signal sequence $O_{1:t}$:

$$s^*_t = \psi_{t+1}(s^*_{t+1}) \text{ for } t = 1, \ldots, T - 1$$

### Viterbi notes

Main difference between the Viterbi algorithm and calculating $P(O)$ is:

- **Viterbi:** accumulators ($\delta$) represent maximums

$$\delta(j) = \max \{ \delta(t-1)(i) \cdot p_{jk} \} \cdot a_{jk} \text{ for } t = 2, \ldots, T; j = 1, \ldots, n$$

- **$P(O)$:** accumulators ($\alpha$) represent sums

$$\alpha(t+1)(j) = \sum_{i=1}^{n} \alpha(t)(i) \cdot p_{ij} \cdot a_{jk} \text{ for } t = 1, \ldots, T-1; j = 1, \ldots, n$$

Same number of required calculations: $O(n^2 T)$

### Viterbi Example

Initial vector: [0.45 0.35 0.15 0.05]

<table>
<thead>
<tr>
<th>t / t+1</th>
<th>DT</th>
<th>JJ</th>
<th>NN</th>
<th>VB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>0.03</td>
<td>0.42</td>
<td>0.50</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Transition matrix:

- JJ: 0.01 0.25 0.65 0.09
- NN: 0.07 0.03 0.15 0.75
- VB: 0.30 0.25 0.15 0.30

Emission matrix:

- bit: 0.85 0.05 0.03 0.05
- dogs: 0.01 0.10 0.45 0.10
- myth: 0.02 0.02 0.02 0.60
- female: 0.01 0.60 0.25 0.05
- moth: 0.12 0.13 0.25 0.20

Sentence: *a myth is a female moth*