

Hidden markov model

Victor Kitov

v.v.kitov@yandex.ru

Markov model

- z_1, z_2, \dots, z_N - some random sequence

$$p(z_1, z_2, \dots, z_N) = p(z_1)p(z_2|z_1)p(z_3|z_1, z_2)\dots p(z_N|z_1\dots z_{N-1})$$

- Markov model of order k :

$$p(z_n|z_1, \dots, z_{n-1}) = p(z_n|z_{n-k}\dots z_{n-1})$$

- it is simpler
 - but easier to estimate
- Markov model of order k corresponds to Markov model of order 1, if we consider sequences of length k :

$$z_{n-1} \rightarrow \tilde{z}_{n-1} = (z_{n-1}, \dots, z_{n-k})$$

So its enough to consider only Markov sequences of order 1 (with larger set of states).

Hidden Markov model

At $t = 1$ HMM is in some random state with probability

$$p(y_1 = i) = \pi_i$$

For each time $t = 1, 2, \dots$ HMM:

- is in some hidden state $y_t \in \{1, 2, \dots, S\}$
- generates some observable output x_t with probability $p(x_t | y_t) = b_{y_t}(x_t)$
- From t to $t + 1$ HMM changes state with probability transition matrix $A = \{a_{ij}\}_{i,j=1}^S$:

$$a_{ij} = p(y_{t+1} = j | y_t = i)$$

Definitions

- We will consider $x_t \in \{1, 2, \dots, R\}$, then $b_y(x)$ corresponds to matrix $B = \{b_{ir}\}_{i=1, \dots, S}^{r=1, \dots, R}$
- Parameters of HMM $\theta = \{\pi, A, B\}$.
- Suppose our HMM process lasted for T periods.
- Define:
 - $X := x_1 x_2 \dots x_T$, $Y := y_1 y_2 \dots y_T$
 - $X_{[i,j]} := x_i x_{i+1} \dots x_j$, $Y_{[i,j]} := y_i y_{i+1} \dots y_j$

Probability calculation

Then

$$p(X|Y) = \prod_{t=1}^T b_{y_t}(x_t)$$

$$p(Y) = \pi_{y_1} \prod_{t=1}^{T-1} a_{y_t y_{t+1}}$$

Together these two formulas give

$$p(Y, X) = p(Y)p(X|Y) = \pi_{y_1} \prod_{t=1}^{T-1} a_{y_t y_{t+1}} \prod_{t=1}^T b_{y_t}(x_t)$$

Problems occur when we need to calculate $P(X) = \sum_Y p(X, Y)$, because this contains exponentially rising with T number of terms.

Forward algorithm

- Define $\alpha_t(i, X) := p(y_t = i, x_1 \dots x_t)$
- We can calculate α_t recursively:

$$\alpha_1(j, X) = p(y_1 = j, x_1) = p(y_1 = j)p(x_1 | y_1 = j) = \pi_j b_j(x_1)$$

$$\begin{aligned} \alpha_{t+1}(j, X) &= p(y_{t+1} = j, x_1 \dots x_{t+1}) = \sum_{i=1}^S p(y_t = i, y_{t+1} = j, x_1 \dots x_t x_{t+1}) \\ &= \sum_{i=1}^S p(y_t = i, x_1 \dots x_t) p(y_{t+1} = j | y_t = i) p(x_{t+1} | y_{t+1} = j) \\ &= \sum_{i=1}^S \alpha_t(i, X) a_{ij} b_j(x_{t+1}) \end{aligned}$$

Forward algorithm

- Define $\alpha_t(i, X) := p(y_t = i, x_1 \dots x_t)$
- We can calculate α_t recursively:

$$\alpha_1(j, X) = p(y_1 = j, x_1) = p(y_1 = j)p(x_1 | y_1 = j) = \pi_j b_j(x_1)$$

$$\begin{aligned} \alpha_{t+1}(j, X) &= p(y_{t+1} = j, x_1 \dots x_{t+1}) = \sum_{i=1}^S p(y_t = i, y_{t+1} = j, x_1 \dots x_{t+1}) \\ &= \sum_{i=1}^S p(y_t = i, x_1 \dots x_t) p(y_{t+1} = j | y_t = i) p(x_{t+1} | y_{t+1} = j) \\ &= \sum_{i=1}^S \alpha_t(i, X) a_{ij} b_j(x_{t+1}) \end{aligned}$$

- Now its trivial to calculate $P(X) = \sum_{i=1}^S \alpha_T(i, X)$.
- Computational complexity of full forward pass $O(TS^2)$.
 - for $t = 1, 2, \dots, T$ summation over S terms for each of S states.
 - It can be reduced to TM where M is the number of non-zero entries in A if we set apriori some transitions as impossible.

Backward algorithm

Define

$$\beta_t(i, \mathbf{X}) := p(X_{t+1}X_{t+2}\dots X_T | y_t = i)$$

As probability of empty event:

$$\beta_T(i, \mathbf{X}) = p(\emptyset | y_T = i) = 1 \quad i = 1, 2, \dots, S$$

We can calculate β_t recursively:

$$\begin{aligned} \beta_t(i, \mathbf{X}) &= p(x_{t+1}\dots x_T | y_t = i) \\ &= \sum_{j=1}^S p(y_{t+1} = j | y_t = i) p(x_{t+1} | y_{t+1} = j) \times \\ &\quad \times p(x_{t+2}\dots x_T | y_{t+1} = j) \\ &= \sum_{j=1}^S a_{ij} b_j(x_{t+1}) \beta_{t+1}(j, \mathbf{X}) \end{aligned}$$

Properties of forward-backward calculation

$$\sum_{i=1}^S \alpha_t(i, X) \beta_t(i, X) = p(X) \quad \forall t = 1, 2, \dots, T$$

$$p(y_t = i | X) = \frac{\alpha_t(i, X) \beta_t(i, X)}{p(X)}$$

$$p(y_t = i, y_{t+1} = j | X) = \frac{\alpha_t(i, X) a_{ij} b_j(x_{t+1}) \beta_{t+1}(j, X)}{p(X)}$$

- This calculation leads to numerical underflow as $\alpha_t(j, X) \rightarrow 0$ and $\beta_t(j, X) \rightarrow 0$ when $T \rightarrow \infty$.
 - Use feasible calculation with $\alpha'_t(j, X)$ and $\beta'_t(j, X)$ that don't $\rightarrow 0$ as T gets large.

Feasible calculation

Define

$$\alpha'_t(i, \mathbf{X}) := p(y_t = i | \mathbf{X}_{[1,t]})$$

$$\eta_t(i, \mathbf{X}) := p(y_t = i, x_t | \mathbf{X}_{[1,t-1]})$$

$$\eta_t(\mathbf{X}) := p(x_t | \mathbf{X}_{[1,t-1]})$$

Then

$$\alpha'_t(i, \mathbf{X}) = \frac{\eta_t(i, \mathbf{X})}{\eta_t(\mathbf{X})}$$

Feasible recurrent calculation

$$\eta_1(i, X) = p(y_1 = i, x_1) = \pi_i b_i(x_1) \quad \eta(X) = p(x_1) = \sum_{s=1}^S \eta_1(s, X)$$

For $t = 1, 2, \dots, T - 1$:

$$\begin{aligned} \eta_{t+1}(j, X) &= p(y_{t+1} = j, x_{t+1} | X_{[1,t]}) \\ &= \sum_{i=1}^S p(y_t = i | X_{[1,t]}) p(y_{t+1} = j | y_t = i) p(x_{t+1} | y_{t+1} = j) \\ &= \sum_{i=1}^S \alpha'_t(i, X) a_{ij} b_j(x_{t+1}) \\ \eta_{t+1}(X) &= \sum_{j=1}^S \eta_{t+1}(j, X) \end{aligned}$$

Feasible calculation

Define

$$\beta'_t(i, \mathbf{X}) := \frac{p(X_{[t+1, T]} | y_t = i)}{p(X_{[t+1, T]} | X_{[1, t]})}$$

These values can be calculated recursively

$$\beta'_T(i, \mathbf{X}) = 1$$

$$\begin{aligned} \beta'_t(i, \mathbf{X}) &= \frac{\sum_{j=1}^S p(y_{t+1} = j | y_t = i) p(x_{t+1} | y_{t+1} = j) p(X_{[t+2, T]} | y_{t+1} = j)}{p(x_{t+1} | X_{[1, t]}) p(X_{[t+2, T]} | X_{[1, t+1]})} \\ &= \frac{\sum_{j=1}^S a_{ij} b_j(x_{t+1}) \beta'_{t+1}(j, \mathbf{X})}{\eta_{t+1}(\mathbf{X})}, \quad t = T - 1, \dots, 1. \end{aligned}$$

Feasible calculation

$$p(y_t = i | X) = \alpha'_t(i, X) \beta'_t(i, X)$$
$$p(y_t = i, y_{t+1} = j | X) = \frac{\alpha'_t(i, X) a_{ij} b_j(x_{t+1}) \beta'_{t+1}(j, X)}{\eta_{t+1}(X)}$$

α' , β' do not lead to problem of numerical underflow (do not $\rightarrow 0$ as T gets large) and provide a practical way to calculate probabilities of hidden states.

Proof

$$\begin{aligned} p(y_t = i | X) &= \frac{p(y_t = i; X_{[1,t]}; X_{[t+1,T]})}{p(X_{[1,t]}; X_{[t+1,T]})} \\ &= \frac{p(X_{[1,t]})p(y_t = i | X_{[1,t]})p(X_{[t+1,T]} | y_t = i)}{p(X_{[1,t]})p(X_{[t+1,T]} | X_{[1,t]})} \\ &= \alpha'_t(i, X) b'_t(i, X) \end{aligned}$$

Proof

$$\begin{aligned}
 p(y_t = i, y_{t+1} = j | X) &= \frac{p(y_t = i, y_{t+1} = j, X_{[1,t]}, x_{t+1}, X_{[t+2,T]})}{p(X_{[1,t]}, x_{t+1}, X_{[t+2,T]})} \\
 &= \frac{p(X_{[1,t]})p(y_t = i | X_{[1,t]})p(y_{t+1} = j | y_t = i)}{p(X_{[1,t]})p(x_{t+1} | X_{[1,t]})} \times \\
 &\quad \times \frac{p(x_{t+1} | y_{t+1} = j)p(X_{[t+2,T]} | y_{t+1} = j)}{p(X_{[t+2,T]} | X_{[1,t+1]})} \\
 &= \frac{\alpha'_t(i, X) a_{ij} b_j(x_{t+1}) \beta'_{t+1}(j, X)}{\eta_{t+1}(X)}
 \end{aligned}$$

Viterbi algorithm

- Problem: for given $X_{[1,T]}$ find maximum probable $Y_{[1,T]}$.
 - full search considers S^T variants, impractical!
- Define

$$y_1^*, \dots, y_T^* := \arg \max_{y_1, \dots, y_T} p(y_1, \dots, y_T, x_1, \dots, x_T)$$

$$\varepsilon_t(i, X) := \max_{y_1, \dots, y_{t-1}} p(y_1 \dots y_{t-1} y_t = i, x_1 \dots x_t)$$

$$v_t(i, X) := \arg \max_j p(y_1 \dots y_{t-2}, y_{t-1} = j, y_t = i, x_1 \dots x_t)$$

- Viterbi algorithm:
 - based on dynamic programming approach
 - forward pass: calculation of $\varepsilon_t(i, X)$ for all $t = 1, 2, \dots, T$ and $i = 1, 2, \dots, S$.
 - backward pass: calculation of y_T^* and recursively y_t^* for $t = T - 1, T - 2, \dots, 1$.

Viterbi algorithm: forward pass

Definitions:

$$\varepsilon_t(i, X) := \max_{y_1, \dots, y_{t-1}} p(y_1 \dots y_{t-1} y_t = i, x_1 \dots x_t)$$

$$v_t(i, X) := \arg \max_j p(y_1 \dots y_{t-2}, y_{t-1} = j, y_t = i, x_1 \dots x_t)$$

Init:

$$\varepsilon_1(i, X) = p(x_1, y_1 = i) = \pi_i b_i(x_1)$$

For $t = 1, \dots, T - 1$:

$$\begin{aligned} \varepsilon_{t+1}(i, X) &= \max_{y_1 \dots y_{t-1} j} p(x_1 \dots x_t x_{t+1}, y_1 \dots y_{t-1} y_t = j, y_{t+1} = i) \\ &= \max_j \max_{y_1 \dots y_{t-1}} p(y_1 \dots y_{t-1} y_t = j, x_1 \dots x_t) p(x_{t+1} y_{t+1} = i | y_1 \dots y_{t-1} y_t = j, x_1 \dots x_t) \\ &= \max_j \max_{y_1 \dots y_{t-1}} p(y_1 \dots y_{t-1} y_t = j, x_1 \dots x_t) p(x_{t+1} y_{t+1} = i | y_t = j) \\ &= \max_j \max_{y_1 \dots y_{t-1}} p(y_1 \dots y_{t-1} y_t = j, x_1 \dots x_t) p(y_{t+1} = i | y_t = j) p(x_{t+1} | y_{t+1} = i) \\ &= \max_j \varepsilon_t(j, X) a_{ji} b_i(x_{t+1}) \end{aligned}$$

$$v_{t+1}(i, X) = \arg \max_j \varepsilon_t(j, X) a_{ji}$$

Viterbi algorithm: backward pass

Definitions

$$y_1^*, \dots, y_T^* := \arg \max_{y_1, \dots, y_T} p(y_1, \dots, y_T, x_1, \dots, x_T)$$

$$\varepsilon_t(i, X) := \max_{y_1, \dots, y_{t-1}} p(y_1 \dots y_{t-1} y_t = i, x_1 \dots x_t)$$

$$v_t(i, X) := \arg \max_j p(y_1 \dots y_{t-2}, y_{t-1} = j, y_t = i, x_1 \dots x_t)$$

Init:

$$p^*(X) = \max_j \varepsilon(j, X)$$

$$y_T^*(X) = \arg \max_j \varepsilon(j, X)$$

For $t = T - 1, T - 2, \dots, 1$:

$$y_t^*(X) = v_{t+1}(y_{t+1}^*(X))$$