Hidden Markov Model - Part 2

Viterbi algorithm and Baum–Welch algorithm

Po-Chuan, Chien

May 9, 2017

Viterbi algorithm

Viterbi algorithm

- Purpose: given a sequence of observed events, find out the most likely sequence of hidden states, ***Viterbi path***.
- Review:
  State space $S = s_1, s_2, s_3, \ldots, s_K$
  Observation space $Y = y_1, y_2, y_3, \ldots, y_T$
  Initial state $\Pi = [\pi_1, \pi_2, \ldots, \pi_K]$
  Transition probability $A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,K} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ a_{K,1} & a_{K,2} & \cdots & a_{K,K} \end{bmatrix}$

Algorithm details

- Approach: dynamic programming.
- Given observed sequence $y_1, y_2, \ldots, y_T$, find out the most likely sequence of $x_1, x_2, \ldots, x_K$.
- Let $V_{t,k}$ be the probability when the hidden state of the t^{th} observation is k

  $V_{1,2} = P(y_1|2) \times \pi_2$

  $V_{1,k} = P(y_1|k) \times \pi_k$

  $V_{t,k} = P(y_t|k) \times \max_{x \in S} (a_{x,k} \times V_{t-1,x})$

Algorithms details

- To restore the path, start from the last observed element, y_T, the last hidden state is

$$\arg\max_{x \in S} \left(V_{y_T, x}\right)$$

  and the remaining path is restored in the same manner.
- Time complexity = $O\left(T \times |S|^2\right)$

Baum–Welch algorithm

Baum–Welch algorithm

- Background:
  A: state transition probability
  B: an $N \times K$ matrix, where $B_{n,k} =$ the probability that result n is chosen at state k
  Π: initial state probability
  $\theta = (A, B, \Pi)$: hidden Markov chain
- If A, B, and Π are determined, the hidden Markov chain is also determined.

Baum–Welch algorithm

- Purpose: given observed sequence Y, find out the most likely θ, or

$$\theta^* = \arg\max_{\theta} P(Y|\theta)$$

- Approach: maximum likelihood estimation.
- To begin with, we set A, B, and Π randomly or by previous information.

Algorithm details - forward procedure

- Let
$$\alpha_i(t) = P(Y_1 = y_1, \ldots, Y_t = y_t, X_t = i|\theta)$$
the probability of the sequence $Y_1 \ldots Y_t$ being seen in state i at time t.
- $\alpha_i(t)$ can be obtained from the following recursion
    1. $\alpha_i(x) = \pi_i b_{i,y_x}$
    2. $\alpha_j(t+1) = b_{j,y_{t+1}} \sum_{i=1}^{K} \alpha_i(t) a_{i,j}$

Algorithm details - backward procedure

- Let

$$\beta_i(t) = P(Y_{t+1} = y_{t+1}, \ldots, Y_T = y_T | X_t = i, \theta)$$

  the probability of the ending partial sequence if we start from state i at time t.
- $\beta_i(t)$ is also found by a recursion:
  1. $\beta_i(T) = 1$
  2. $\beta_i(t) = \sum_{j=1}^{K} \beta_j(t+1) a_{i,j} b_{j,y_{t+1}}$

Algorithm details - update

- Let

$$\gamma_i(t) = P(X_t = i | Y, \theta) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{K} \alpha_j(t)\beta_j(t)} = \frac{\text{passing } i}{\text{all paths}}$$

  the probability of passing state i at time t.

- Let

$$\xi_{ij}(t) = P(X_t = i, X_{t+1} = j | Y, \theta) =$$

$$\frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_{j,y_{t+1}}}{\sum_{i=1}^{K} \sum_{j=1}^{K} \alpha_i(t) a_{ij} \beta_j(t+1) b_{j,y_{t+1}}} = \frac{\text{from } i \text{ to } j}{\text{from any to any}}$$

  the probability of passing from state i to state j from time t to $t+1$

Algorithm details - update

- Expected initial probability:

$$\pi_i^* = \gamma_i(1)$$

- The probability from state *i* to state *j*

$$a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} = \frac{\text{at here and move}}{\text{at here}}$$

- Expected probability of getting *val* at state *i*

$$b_{i,val}^* = \frac{\sum_{t=1}^{T} 1_{y_t=val} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)} = \frac{\text{get } val \text{ here}}{\text{at here}}$$

Algorithms - epilogue

1. Forward procedure
2. Backward procedure
3. Update

- The above algorithm can be repeated for several times until the convergence level is desired.

Thanks for listening
Are you still alive?