Hidden Markov Models Implementation

Chaoyu Li, chaoyuli@usc.edu, 6641732094

DSCI-552: Machine Learning for Data Science

Instructor: Satish Kumar Thittamaranahalli

Apr. 18, 2022

Table of Contents

1 Basic Algorithm and Code Functions Description

1.1 Implementation of HMM

1.1.1 Basic Algorithm of HMM (Viterbi algorithm)

Table 5.2 from [Müller, FMP, Springer 2015]

Algorithm: VITERBI

Input: HMM specified by $\Theta = (\mathcal{A}, A, C, \mathcal{B}, B)$
Observation sequence $O = (o_1 = \beta_{k_1}, o_2 = \beta_{k_2}, \ldots, o_N = \beta_{k_N})$

Output: Optimal state sequence $S^* = (s_1^*, s_2^*, \ldots, s_N^*)$

Procedure: Initialize the $(I \times N)$ matrix $\mathbf{D}$ by $\mathbf{D}(i,1) = c_i b_{ik_1}$ for $i \in [1:I]$. Then compute in a nested loop for $n = 2, \ldots, N$ and $i = 1, \ldots, I$:

$$\begin{aligned} \mathbf{D}(i,n) &= \max_{j \in [1:I]} \left(a_{ji} \cdot \mathbf{D}(j, n-1) \right) \cdot b_{ik_n} \\ \mathbf{E}(i,n-1) &= \operatorname{argmax}_{j \in [1:I]} \left(a_{ji} \cdot \mathbf{D}(j, n-1) \right) \end{aligned}$$

Set $i_N = \operatorname{argmax}_{j \in [1:I]} \mathbf{D}(j,N)$ and compute for decreasing $n = N-1, \ldots, 1$ the maximizing indices

$$i_n = \operatorname{argmax}_{j \in [1:I]} \left(a_{j i_{n+1}} \cdot \mathbf{D}(j,n) \right) = \mathbf{E}(i_{n+1}, n).$$

The optimal state sequence $S^* = (s_1^*, \ldots, s_N^*)$ is defined by $s_n^* = \alpha_{i_n}$ for $n \in [1:N]$.

1.1.2 Code Functions of HMM

- **viterbi(obs, states, prior_prob, trans_prob, emission_prob)**

It is the main function of the Viterbi algorithm. It accepts five arguments: observation sequence, states sequence, the prior probability matrix, the transmission probability matrix and the emission probability matrix. In this assignment, the arguments are shown below:

  - States: (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
  - Observation: (8, 6, 4, 6, 5, 4, 5, 5, 7, 9)
  - Prior probability = {1: 0.1, 2: 0.1, 3: 0.1, 4: 0.1, 5: 0.1, 6: 0.1, 7: 0.1, 8: 0.1, 9: 0.1, 10: 0.1}
  - Transmission probability = {
    1: {2: 1.0, else: 0},
    2: {1: 0.5, 3: 0.5, else: 0},

3: {2: 0.5, 4: 0.5, else: 0},
4: {3: 0.5, 5: 0.5, else: 0},
5: {4: 0.5, 6: 0.5, else: 0},
6: {5: 0.5, 7: 0.5, else: 0},
7: {6: 0.5, 8: 0.5, else: 0},
8: {7: 0.5, 9: 0.5, else: 0},
9: {8: 0.5, 10: 0.5, else: 0},
10: {9: 1.0, else: 0}}

- Emission probability = {
  1: {1: 0.5, 2: 0.5, else: 0},
  2: {1: 0.33, 2: 0.33, 3: 0.33, else: 0},
  3: {2: 0.33, 3: 0.33, 4: 0.33, else: 0},
  4: {3: 0.33, 4: 0.33, 5: 0.33, else: 0},
  5: {4: 0.33, 5: 0.33, 6: 0.33, else: 0},
  6: {5: 0.33, 6: 0.33, 7: 0.33, else: 0},
  7: {6: 0.33, 7: 0.33, 8: 0.33, else: 0},
  8: {7: 0.33, 8: 0.33, 9: 0.33, else: 0},
  9: {8: 0.33, 9: 0.33, 10: 0.33, else: 0},
  10: {9: 0.5, 10: 0.5, else: 0}}

2 Data Structure Used

2.1 HMM data structure

A. The possible states are stored in the data structure of Tuple. The format is:

(1, 2, 3, 4, 5, 6, 7, 8, 9, 10).

B. The observation sequence is stored in the data structure of Tuple. The format is:

(8, 6, 4, 6, 5, 4, 5, 5, 7, 9).

C. The prior probability is stored in the data structure of Dictionary. The format is:

{1: 0.1, 2: 0.1, 3: 0.1, 4: 0.1, 5: 0.1, 6: 0.1, 7: 0.1, 8: 0.1, 9: 0.1, 10: 0.1}

D. The transmission probability is stored in the data structure of Dictionary. The format is:

{1: {2: 1.0, 1: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0, 10: 0},

…

10: {9: 1.0, 1: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 2: 0, 10: 0}}

E. The emission probability is stored in the data structure of Dictionary. The format is:

{1: {1: 0.5, 2: 0.5, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0, 10: 0},

…

10: {9: 0.5, 10: 0.5, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 1: 0, 2: 0}}

3 Challenges Faced and Optimizations

3.1 Challenges

A. It is kind of hard to implement the dynamic programming algorithm in this ML problem.

3.2 Optimizations

A. I used the dictionary to update the transitional probabilities because the key properties are so convenient here.

B. I chose the Viterbi algorithm here because it can keep the problem tractable.

4 Execution of my algorithm and results

- Print of result:

  #Step1:

  {1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0, 5: 0.0, 6: 0.0, 7: 0.03333333333333333, 8: 0.03333333333333333, 9: 0.03333333333333333, 10: 0.0}

  #Step2: {1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0, 5: 0.0, 6: 0.005555555555555555, 7: 0.005555555555555555, 8: 0.0, 9: 0.0, 10: 0.0}

  #Step3: {1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0, 5: 0.0009259259259259257, 6: 0.0, 7: 0.0, 8: 0.0, 9: 0.0, 10: 0.0}

  #Step4: {1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0, 5: 0.0, 6: 0.00015432098765432096, 7: 0.0, 8: 0.0, 9: 0.0, 10: 0.0}

  #Step5: {1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0, 5: 2.572016460905349e-05, 6: 0.0, 7: 0.0, 8: 0.0, 9: 0.0, 10: 0.0}

  #Step6: {1: 0.0, 2: 0.0, 3: 0.0, 4: 4.286694101508915e-06, 5: 0.0, 6: 0.0, 7: 0.0, 8: 0.0, 9: 0.0, 10: 0.0}

  #Step7: {1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0, 5: 7.144490169181524e-07, 6: 0.0, 7: 0.0, 8: 0.0, 9: 0.0, 10: 0.0}

  #Step8: {1: 0.0, 2: 0.0, 3: 0.0, 4: 1.190748361530254e-07, 5: 0.0, 6: 1.190748361530254e-07, 7: 0.0, 8: 0.0, 9: 0.0, 10: 0.0}

  #Step 9: {1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0, 5: 0.0, 6: 0.0, 7: 1.984580602550423e-08, 8: 0.0, 9: 0.0, 10: 0.0}

  #Step 10:

  {1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0, 5: 0.0, 6: 0.0, 7: 0.0, 8: 3.3076343375840383e-09, 9: 0.0, 10: 0.0}

  The most likely sequence is: [7, 6, 5, 6, 5, 4, 5, 6, 7, 8]

  The possible of the sequence above is: 3.3076343375840383e-09

5 Contributions

All works were completed by Chaoyu Li.