深蓝学院
shenlanxueyuan.com

第四章 HMM
作业代码讲解

作业内容：

实践

1.已知

考虑盒子和球模型$\lambda = (A, B, \pi)$，状态集合$Q = \{1,2,3\}$，观测集合$V = \{红，白\}$，

$$A = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}, \mathrm{B} = \begin{bmatrix} 0.5 & 0.5 \\ 0.4 & 0.6 \\ 0.7 & 0.3 \end{bmatrix}, \pi = (0.2, 0.4, 0.4)^T$$

设$T = 3$，$O = (红，白，红)$，

2.前后向算法-概率计算问题

- 请用Python编程实现前向算法和后向算法，分别计算 $P(O|\lambda)$；

3.Viterbi算法-解码问题

- 请用Python编程实现Viterbi算法，求最优状态序列，即最优路径 $I^* = (i_1^*, i_2^*, i_3^*)$.
- 程序输入和函数接口已写好，请独立完成算法核心部分！Have Fun ☺
- https://github.com/nwpuaslp/ASR_Course/tree/master/04-HMM

文件结构：

我们这次的文件结构很简单，只有一个hmm.py文件，我们需要完成算法的文件。

hmm.py：

1.前向算法-需要完成作业的位置

```python
def forward_algorithm(O, HMM_model):…
```

2.后向算法-需要完成作业的位置

```python
def backward_algorithm(O, HMM_model):…
```

3.Viterbi算法-需要完成作业的位置

```python
def Viterbi_algorithm(O, HMM_model):…


if __name__ == "__main__":
    color2id = { "RED": 0, "WHITE": 1 }
    # model parameters
    pi = [0.2, 0.4, 0.4]
    A = [[0.5, 0.2, 0.3],
         [0.3, 0.5, 0.2],
         [0.2, 0.3, 0.5]]
    B = [[0.5, 0.5],
         [0.4, 0.6],
         [0.7, 0.3]]
    # input
    observations = (0, 1, 0)
    HMM_model = (pi, A, B)
    # process
    observ_prob_forward = forward_algorithm(observations, HMM_model)
    print(observ_prob_forward)

    observ_prob_backward = backward_algorithm(observations, HMM_model)
    print(observ_prob_backward)

    best_prob, best_path = Viterbi_algorithm(observations, HMM_model)
    print(best_prob, best_path)
```

初始状态分布π

状态转移矩阵A

观测概率分布B

注意行和列的概念

观测序列O

前向算法：

- **前向算法**
  - **前向概率定义**：给定隐马尔可夫模型 λ，定义到时刻 t 部分观测序列为 $o_1, o_2, \ldots, o_t$ 且状态为 q_i 的概率为前向概率，记作（可省略 λ）

$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, i_t = q_i | \lambda)$$

  - **算法 10.2（观测序列概率的前向算法）**

输入：隐马尔可夫模型 λ，观测序列 O；

输出：观测序列概率 $P(O|\lambda)$.

（1）初值

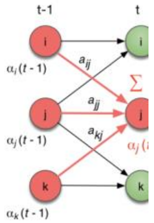
$$\alpha_1(i) = \pi_i b_i(o_1), \quad i = 1, 2, \cdots, N \tag{10.15}$$

（2）递推　对 $t = 1, 2, \cdots, T-1$，

$$\alpha_{t+1}(i) = \left[\sum_{j=1}^{N} \alpha_t(j) a_{ji}\right] b_i(o_{t+1}), \quad i = 1, 2, \cdots, N \tag{10.16}$$

（3）终止

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{10.17}$$ ■




```python
def forward_algorithm(O, HMM_model):
    """HMM Forward Algorithm.
    Args:
        O: (o1, o2, ..., oT), observations
        HMM_model: (pi, A, B), (init state prob, transition prob, emitting prob)
    Return:
        prob: the probability of HMM_model generating O.
    """
    pi, A, B = HMM_model
    T = len(O)
    N = len(pi)
    prob = 0.0
    # Begin Assignment

    # Put Your Code Here

    # End Assignment
    return prob
```

HMM参数(π, A, B)

step

1.创建alpha列表，根据pi和观测概率B计算初值，保存在alpha列表中(对应步骤(1))；

2.根据alpha列表，状态转移矩阵A，观测概率分布B，计算出所有的alpha(对应步骤(2))；

3.求和获得观测序列的概率(对应步骤(3))。

返回前向概率值

后向算法：

- **后向算法**
  - **后向概率定义**：给定隐马尔可夫模型 λ，定义在时刻 t状态为 q_i 的条件下，从 $t+1$ 到 T 的部分观测序列为 $o_{t+1}, o_{t+2}, \ldots, o_T$ 的概率为后向概率，记作（可省略 λ）

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T | i_t = q_i, \lambda)$$

  - **算法 10.3 （观测序列概率的后向算法）**

    输入：隐马尔可夫模型 λ，观测序列 O；

    输出：观测序列概率 $P(O|\lambda)$.

    （1）

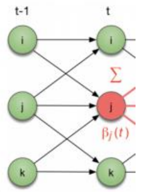
$$\beta_T(i) = 1, \quad i = 1, 2, \cdots, N \tag{10.19}$$

    （2）对 $t = T-1, T-2, \cdots, 1$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad i = 1, 2, \cdots, N \tag{10.20}$$

    （3）

$$P(O|\lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i) \tag{10.21}$$ ■




```python
def backward_algorithm(O, HMM_model):
    """HMM Backward Algorithm.
    Args:
        O: (o1, o2, ..., oT), observations
        HMM_model: (pi, A, B), (init state prob, transition prob, emitting prob)
    Return:
        prob: the probability of HMM_model generating O.
    """
    pi, A, B = HMM_model
    T = len(O)
    N = len(pi)
    prob = 0.0
    # Begin Assignment

    # Put Your Code Here

    # End Assignment
    return prob
```

step

1.创建beta列表，保存beta T时刻值(对应步骤(1))；

2.根据beta列表，状态转移矩阵A，观测概率分布B，计算出所有的beta(对应步骤(2));

3.求和获得观测序列的概率(对应步骤(3))。

*注意*后向算法和前向算法的不同，通俗讲后向是从后往前算。

Viterbi算法：

算法 10.5（维特比算法）

输入：模型 $\lambda=(A,B,\pi)$ 和观测 $O=(o_1,o_2,\cdots,o_T)$；

输出：最优路径 $I^*=(i_1^*,i_2^*,\cdots,i_T^*)$.

（1）初始化

$$\delta_1(i)=\pi_i b_i(o_1),\quad i=1,2,\cdots,N$$
$$\psi_1(i)=0,\quad i=1,2,\cdots,N$$

（2）递推．对 $t=2,3,\cdots,T$

$$\delta_t(i)=\max_{1\leqslant j\leqslant N}[\delta_{t-1}(j)a_{ji}]b_i(o_t),\quad i=1,2,\cdots,N$$
$$\psi_t(i)=\arg\max_{1\leqslant j\leqslant N}[\delta_{t-1}(j)a_{ji}],\quad i=1,2,\cdots,N$$

（3）终止

$$P^*=\max_{1\leqslant i\leqslant N}\delta_T(i)$$
$$i_T^*=\arg\max_{1\leqslant i\leqslant N}[\delta_T(i)]$$

（4）最优路径回溯．对 $t=T-1,T-2,\cdots,1$

$$i_t^*=\psi_{t+1}(i_{t+1}^*)$$

求得最优路径 $I^*=(i_1^*,i_2^*,\cdots,i_T^*)$.

```python
def Viterbi_algorithm(O, HMM_model):
    """Viterbi decoding.
    Args:
        O: (o1, o2, ..., oT), observations
        HMM_model: (pi, A, B), (init state prob, transition prob, emitting prob)
    Returns:
        best_prob: the probability of the best state sequence
        best_path: the best state sequence
    """
    pi, A, B = HMM_model
    T = len(O)
    N = len(pi)
    best_prob, best_path = 0.0, []
    # Begin Assignment

    # Put Your Code Here

    # End Assignment
    return best_prob, best_path
```

Viterbi算法是一种解码，获取最佳路径的方法，best_path用于返回我们获取的最佳路径。编程中我们需要创建delta、phi来保存路径中的概率值和状态值。到终止时刻时，我们获得了最后时刻的最佳状态，然后从phi中回溯，找到最佳路径best_path。