偶尔作弊的赌场——动态规划在HMM中的体现

原创 曲宗福 LOA算法学习笔记 2021-02-16 20:30

01 HMM的三个问题

1. **概率计算问题。**给定模型 $\lambda=(A,B,\pi)$ 和观测序列 $O=(o_1,o_2,\cdots,o_T)$，计算在模型 λ 下观测序列 O 出现的概率 $P(O|\lambda)$。
2. **学习问题。**已知观测序列 $O=(o_1,o_2,\cdots,o_T)$，估计模型 $\lambda=(A,B,\pi)$ 参数。
3. **预测问题，也称为解码问题。**已知 $\lambda=(A,B,\pi)$ 和观测序列 $O=(o_1,o_2,\cdots,o_T)$，求最有可能的对应的中的状态序列。

其中第1和第3种问题都用到了动态规划。

02 HMM的概率计算问题

首先定义前向概率：

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

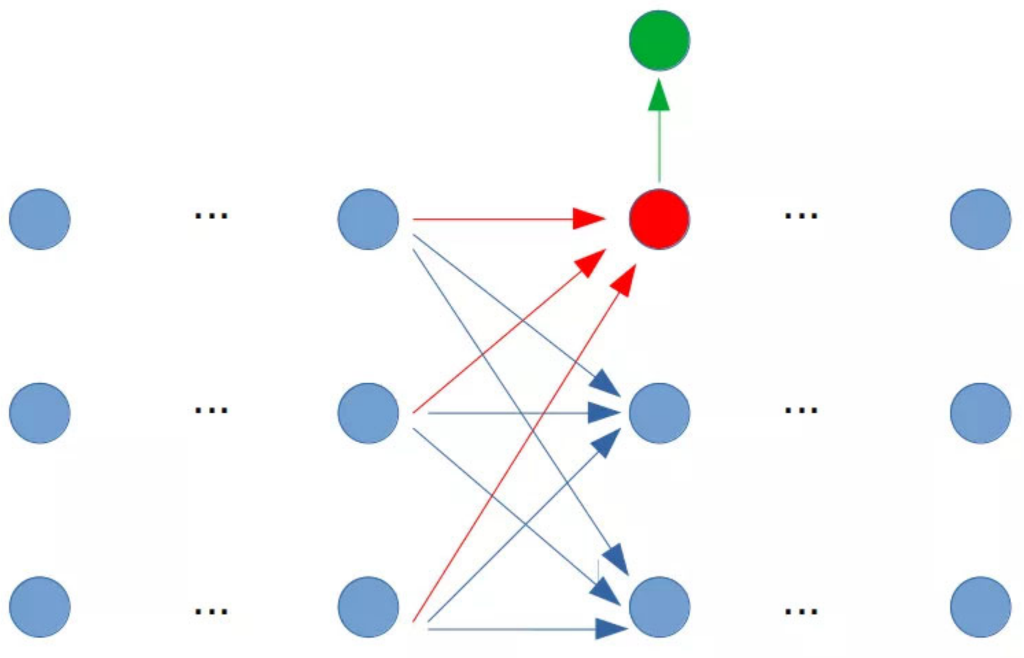
即在时刻t，部分观察序列为 $o_1,o_2,\cdots,o_t$ 且状态为 q_i 的概率。
下面介绍前向算法的过程：
1. 初始化：

$$\alpha_1(i)=\pi_i b_i(o_1)$$

2. 我们可以得到状态转移方程为：

$$\alpha_{t+1}(i)=[\sum_{j=1}^{N}\alpha_t(j)a_{ji}]b_t(o_{t+1}),\qquad i=1,2,3,\cdots,N$$

其中 a_{ij} 表示从第i个状态转变成第j的状态的概率



上图中横向表示第几次观测，纵向表示状态，红色点表示当前正在计算的前向概率,绿色点表示观测状态。
3. 终止:

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

在偶尔作弊的赌场问题中，给出了开始时使用哪一种骰子的概率，以及过程中的骰子转移概率，以及生成概率，这样就可以计算出给定的序列出现的概率。

03 预测问题(viterbi算法)

首先定义来来两个变量， δ, ϕ ，定义在时刻t状态为i的所有单个路径中概率最大值为:

$$\delta_t(i) = \max_{i_1, i_2, \cdots, i_{t-1}} P(i_t = i, i_{t-1}, \cdots, i_1, o_t, cdots, o_1|\lambda)$$

$$\delta_t(i) = \max_{1 \leq j \leq N} [\delta_t(j) a_{ij}] b_t(o_{t+1}), i = 1, 2, \cdots, N; t = 1, 2, \cdots, T-1$$

定义在时刻t状态为i的所有单个路径($i_1, i_2, \cdots, i_{t-1}, i$)中概率最大的路径的第t-1个节点为:

$$\phi_t(i) = \arg \max_{1 \leq j \leq N} [\delta_{t-1}(j) a_{ij}], i = 1, 2, \cdots, N$$

下面介绍维特比算法的过程:

1. 初始化:

$$\delta_1(i) = \pi_i b_i(o_1), \qquad i = 1, 2, 3, \cdots, N$$

$$\phi_1(i) = 0, \qquad i = 1, 2, 3, \cdots, N$$

2. 状态转移方程:

$$\delta_t(i) = \max_{1 \le j \le N} [\delta_t(j) a_{ij}] b_t(o_{t+1}), i = 1, 2, \cdots, N; t = 1, 2, \cdots, T-1$$

$$\phi_t(i) = \arg \max_{1 \le j \le N} [\delta_{t-1}(j) a_{ij}], i = 1, 2, \cdots, N$$

3. 终止:

$$P^* = \max_{1 \le j \le N} \delta_T(i)$$

$$i_T^* = \arg \max_{1 \le j \le N} \delta_T(i)$$

4. 最优回溯路径

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

最优回溯路径: $I^* = (i_1^*, i_2^*, \cdots, i_T^*)$

使用代码实现viterbi:

```
import numpy as np


def viterbi(A, B, pi, O, N, M, T):
    '''使用维特比算法进行预测
    参数分别表示状态转移矩阵，状态生成矩阵，初始概率矩阵
    观测值，隐状态数目，观察状态树名，观察序列长度'''
    DP = np.zeros(N * T).reshape(N, T)                  # 动态规划所维护的数组
    back_find = np.zeros(N * T).reshape(N, T)           # 回溯表格
    back_route = np.zeros(T).reshape(1, T)              # 回溯路径
    for i in range(0, N):
```

```
        DP[i][0] = pi[i][0] * B[i][O[0][0]]        #初始化
    for i in range(1, T):
        for j in range(0, N):
            for k in range(0, N):
                if DP[j][i] < DP[k][i-1] * A[k][j] * B[j][O[i]]:
                   DP[j][i] = DP[k][i-1] * A[k][j] * B[j][O[i]]
                   back_find[j][i] = k
    temp_1 = 0
    temp = 0
    for i in range(0, N):
        if temp < DP[i][T-1]:
            temp = DP[i][T-1]
            temp_1 = i
    #print(temp_1)
    back_route[0][T-1] = temp_1
    for i in range(T-2, -1, -1):
        back_route[0][i] = back_find[int (back_route[0][i+1])][i+1]
    print(back_route)


#F为状态0，L为状态1，在偶尔作弊的赌场问题中因状态数目N共有2种，可观察状态共有6种，序列长度为6
T = 6
M = 6
N = 2
A = np.array([0.8, 0.2, 0.1, 0.9]).reshape(N, N)
B = np.array([1/6, 1/6, 1/6, 1/6, 1/6, 1/6, 1/10, 1/10, 1/10, 1/10, 3/10, 3/10]).reshape(N, M)
O = np.array([0, 2, 3, 4, 4, 5]).reshape(T, 1)
pi = np.array([0.6, 0.4]).reshape(N, 1)


viterbi(A, B, pi, O, N, M, T)
```