隐马尔可夫模型

前向概率

（1）初值：

$$a_1(i) = \pi_i b_i(o_1), \quad i = 1, 2, \cdots, N$$

（2）递推：

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

后向概率

（1）初值：

$$\beta_T(i) = 1, \quad i = 1, 2, \cdots, N$$

（2）递推：

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

回到本题，待求的概率为

$$P(i_4 = q_3 | O, \lambda) = \gamma_4(3) = \frac{\alpha_4(3)\beta_4(3)}{\sum\limits_{j=1}^{N} a_4(j)\beta_4(j)}$$

编程计算上述结果

```
import numpy as np

A = np.array([[0.5, 0.1, 0.4],
              [0.3, 0.5, 0.2],
              [0.2, 0.2, 0.6]])
B = np.array([[0.5, 0.5],
              [0.4, 0.6],
              [0.7, 0.3]])
pi = np.array([0.2, 0.3, 0.5])
O = np.array([0, 1, 0, 0, 1, 0, 1, 1])
T = 8

# 前向-初值
alpha = np.zeros((4,3))
for i in range(3):
    alpha[0][i] =  pi[i]*B[i][O[0]]
# 前向-递推
for t in range(1,4):
    for i in range(3):
        alpha[t][i] = B[i][O[t]] * np.sum(alpha[t-1]*A[:,i])
# 后向-初值
beta = np.zeros((T,3))
for i in range(3):
    beta[T-1][i] = 1
# 后向-递推
for t in range(T-2,2,-1):
    for i in range(3):
        beta[t][i] = np.sum(A[i]*B[:,O[t+1]]*beta[t+1])
```

```
# 打印中间结果
print('前向概率矩阵 alpha[t,i](t≤4):\n',alpha,'\n')
print('后向概率矩阵 beta[t,i](仅计算t≥4):\n',beta,'\n')
# 计算结果
res = alpha[3][2]*beta[3][2]/np.dot(alpha[3],beta[3]).sum()
print('本题的计算结果为:',res)
```

输出结果如下，由此可得 $P(i_4 = q_3|O,\lambda) \approx 0.537$

```
前向概率矩阵 alpha[t,i](t≤4):
 [[0.1         0.12        0.35      ]
 [0.078       0.084       0.0822    ]
 [0.04032     0.026496    0.068124  ]
 [0.0208668   0.01236192  0.04361112]]

后向概率矩阵 beta[t,i](仅计算t≥4):
 [[0.          0.          0.        ]
 [0.          0.          0.        ]
 [0.04586531  0.05280909  0.04280618]
 [0.105521    0.100883    0.111934  ]
 [0.1861      0.2415      0.1762    ]
 [0.43        0.51        0.4       ]
 [1.          1.          1.        ]]

本题的计算结果为: 0.5369518160647323
```