HMM和Viterbi算法

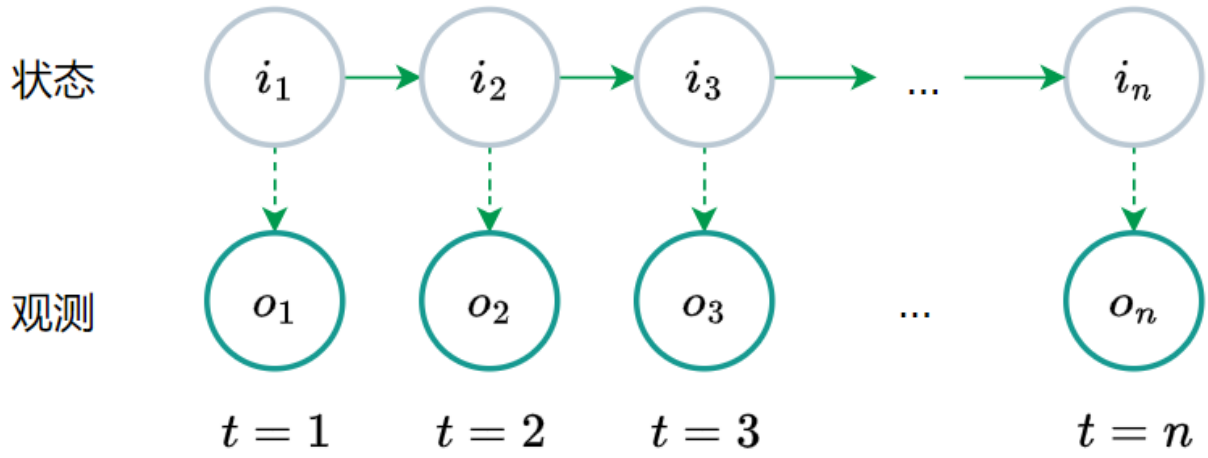
隐马尔可夫模型(hidden Markov model,HMM)是可用于标注问题的统计学习模型，描述由隐藏的马尔可夫链随机生成观测序列的过程，属于生成模型，描述了两个相关序列的依赖关系。隐马尔可夫模型在语音识别、自然语言处理、生物信息、模式识别等领域有着广泛的应用。其主要思想来源于**马尔可夫过程**[1] ，它假设当前时刻状态只与前一时刻的状态有关，而与更早的状态无关。

1.隐马尔可夫模型

1.1 隐马尔可夫模型的定义

隐马尔可夫模型是关于时序的概率模型，描述由一个隐藏的马尔可夫链随机生成不可观测的状态随机序列，再由各个状态生成一个观测从而产生观测随机序列的过程。

隐藏的马尔可夫链随机生成的状态的序列，称为**状态序列**(state sequence); 每个状态生成一个观测，而由此产生的观测的随机序列，称为**观测序列**(observation sequence)。序列的每一个位置又可以看作是一个时刻。



长度为n观测序列生成

1.2 隐马尔可夫模型两个基本假设

1. 齐次马尔可夫假设

   在任意时刻，系统的状态只与其前一时刻的状态有关，而与更早之前的状态和操作无关。也就是说，当前状态的概率只与上一个状态的概率有关，与其他状态的概率无关。对于天气预报，假定今天的气温只与昨天有关而和前天无关。当然这种假设未必适合所有的应用，但是至少对以前很多不好解决的问题给出了近似解。

2. 观测独立性假设

   假设在任意时刻，系统的观察值只与当前状态有关，而与其他状态和观察值无关。也就是说，在给定当前状态的情况下，当前时刻的观察值与其他时刻的观察值无关。

这两个假设使得隐马尔可夫模型具有了很好的数学性质，使得可以通过有限数量的状态和参数来准确地表示一个复杂的系统，而且在实际应用中得到了广泛应用。

1.3 隐马尔可夫模型三个基本要素

隐马尔可夫模型由**初始概率分布**、**状态转移概率分布**以及**观测概率分布**确定。

隐马尔可夫模型由**初始状态概率向量**π、**状态转移概率矩阵**A和**观测概率矩阵**B决定。初始化概率矩阵和状态转移概率矩阵决定状态序列，观测概率矩阵决定观测序列。因此，A, B, π称为隐马尔可夫模型的三个基本元素。

初始状态概率向量π、**状态转移概率矩阵**A确定了隐藏的马尔可夫链，生成不可观测的状态序列。**观测概率矩阵**B确定了如何从状态生成观测，与状态序列综合确定了如何产生观测序列。马尔可夫模型λ用三元符号表示，即：

$$\lambda = (\pi, A, B)$$

1.4 隐马尔可夫模型数学表达

- 状态集

  Q是所有可能的状态的集合，V 是所有可能的观测的集合:

$$Q = \{q_1, q_2, \ldots, q_N\}, \quad V = \{v_1, v_2, \ldots v_M\}$$

  其中，N 是可能的状态数，M 是可能的观测数。

- 状态序列

  I是长度为T的状态序列，O是对应的观测序列:

$$I = (i_1, i_2, \cdots, i_T), \quad O = (o_1, o_2, \cdots, o_T)$$

- 状态转移概率矩阵 A

$$A = [a_{ij}]_{N \times N}$$

  其中：

$$a_{ij} = P\left(i_{t+1} = q_j \mid i_t = q_i\right), \quad i = 1, 2, \cdots, N; \quad j = 1, 2, \cdots, N$$

  表示时刻t处于状态q_i 的条件下在时刻t+1转移到状态q_j 的概率。

- 观测概率矩阵B

$$B = [b_j(k)]_{N \times M}$$

  其中：

$$b_j(k) = P\left(o_t = v_k \mid i_t = q_j\right), \quad k = 1, 2, \cdots, M; \quad j = 1, 2, \cdots, N$$

  是在时刻t处于状态q_j的条件下生成观测v_k的概率。

- 初始状态概率向量π

$$\pi = (\pi_i)$$

  其中：

$$\pi_i = P(i_1 = q_i), \quad i = 1, 2, \cdots, N$$

是$t = 1$时刻处于状态q_i的概率。

- 例子1

假设我们有一个人在走路，他走路时会出现三种状态：慢走、快走和停止；同时他的朋友跟在他后面观察他的行动，根据他的脚步声分为三种观测值：脚步声轻、脚步声重和没有脚步声。我们可以用HMM来描述这个问题：

1. 状态集合：{慢走、快走、停止}，表示人在走路时的三种状态；
2. 观测集合：{脚步声轻、脚步声重、没有脚步声}，表示朋友观察到的三种不同的观测值类型；
3. 初始状态概率：表示开始时人在慢走、快走或停止状态的概率分别为0.4、0.3和0.3；
4. 状态转移概率矩阵：表示在不同状态之间转移的概率，例如从慢走状态转换为停止状态的概率为0.3，从停止状体转换为慢走状态的概率为0.4，其它概率参考下表；

状态转移矩阵	慢走	快走	停止
慢走	0.2	0.5	0.3
快走	0.25	0.3	0.45
停止	0.4	0.4	0.2

1. 观测概率矩阵：表示在每种状态下观测到不同观测值的概率分布，例如在慢走状态下观测到脚步声轻的概率为0.7，观测到没有脚步声的概率为0.2，观测到脚步声重的概率为0.1，其它概率参考下表；

观测概率矩阵	脚步声轻	脚步声重	没有脚步声
慢走	0.7	0.1	0.2
快走	0.2	0.7	0.1
停止	0.1	0.5	0.4

通过这个模型，我们可以预测人的走路状态和朋友观察到的脚步声类型。对于观测序列的推断和未来状态的预测，我们可以使用HMM标准算法来实现。

1.5 隐马尔可夫三个基本问题

1. 概率计算问题：

   已知模型参数和观测序列, 计算观测序列出现的概率。 评估问题一般使用前向算法或者后向算法进行解决，其中前向算法相对简单，容易理解一些。后向算法较难理解。设想有两个描述两人语音的HMM模型，那么给一个新的语音序列，利用前向算法或者后向算法就可以计算这个语音序列更可能是哪个人的。

   给定模型 $\lambda = (A, B, \pi)$ 和一个观测序列 $O = (o_1, o_2, \ldots, o_T)$，计算观测序列出现的概率 $P(O|\lambda)$ 。

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

   其中，$\alpha_t(i)$ 是在时刻 t，处于状态 i 并观测到 O 的概率，可以用前向概率进行递推计算。

2. 学习问题：

   模型参数未知，推断模型参数。有两种可能的场景，一种是监督学习的场景，已知诸多观测序列和对应的状态序列，推断模型参数，第二种是非监督学习的场景，只知道诸多观测序列，推断模型参数。监督学习的场景，学习方法相对简单。非监督学习的场景，一般使用EM期望最大化方法进行迭代求解。

   给定观测序列 $O = (o_1, o_2, \ldots, o_T)$，如何估计参数 $\lambda = (A, B, \pi)$，使得在该参数下观测序列的概率最大。

   $$\boldsymbol{\lambda}_{new} = \arg\max_{\lambda} P(O|\lambda)$$

3. 预测问题：

   也叫做解码问题。已知模型参数和观测序列 , 计算该观测序列对应的最可能的状态序列。

2.预测问题

要解决的问题：给定一个模型和某个特定的输出序列，如何找到最可能产生这个输出的状态序列？预测问题一般使用贪心近似算法或者维特比算法解决。其中贪心近似算法相对简单一些，但不一定能找到全局最优解。维特比算法可以找到全局最优，是一种动态规划算法。

给定观测序列 $O = (o_1, o_2, \ldots, o_T)$ 和模型 $\lambda = (A, B, \pi)$，如何求出在该模型下，最有可能的隐含状态序列。

示例2：假设有如下 HMM 模型，

- 状态变量 S: sunny (晴天) 和 rainy (雨天)。

- 观测变量 O: umbrella (带伞) 和 no-umbrella(没带伞)。
- 隐含状态转移概率矩阵 $A = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}$。
- 观测概率矩阵 $B = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}$。
- 初始状态概率 $\pi = [0.6 \quad 0.4]$。

已知观测序列为 $O = \{umbrella, umbrella, no-umbrella, umbrella\}$，求在该模型下，**最有可能的隐含状态序列**。

2.1 近似算法

近似算法的基本思路：在每个时刻t，选择该时刻最有可能出现的状态i_t^*，从而得到一个状态序列 $I = \{i_1^*, i_2^*, \ldots, i_n^*\}$，将其作为预测的结果；

定义

- 前向概率：给定马尔可夫模型，到t时刻部分观测序列为$o_1, o_2, \ldots, o_t$ 且状态为q_i的概率为前向概率（t及t时刻之前特定观测序列下，t时刻出现某种状态的概率），

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

- 后向概率：给定马尔可夫模型，在t时刻状态为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（某种状态下，从当前时刻开始，往后出现指定观测序列的概率）。

$$\beta_t(i) = P\left(o_{t+1}, o_{t+2}, \cdots, o_T \mid i_t = q_i, \lambda\right)$$

给定隐马尔可夫模型λ 和观测序列O，在时刻t 处于状态q_i 的概率$\gamma_t(i)$是

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O \mid \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

每一个时刻t 最有可能的状态i_t^*是

$$i_t^* = \arg \max_{1 \leqslant i \leqslant N} \left[\gamma_t(i)\right], \quad t = 1, 2, \cdots, T$$

从而可得到状态序列$I = i_1^*, i_2^*, \ldots, i_n^*$。

特点：

- 计算简单；
- 缺点是不能保证预测的状态序列整体是最有可能的状态序列；

2.1.1 前向算法

前面已经对前向概率进行了定义，前向算法可以递推地计算前向概率及观测序列概率。

具体来说，前向算法的计算过程分为两个步骤：

第一步：计算初值，即计算出第一个观测值的所有可能状态的概率。

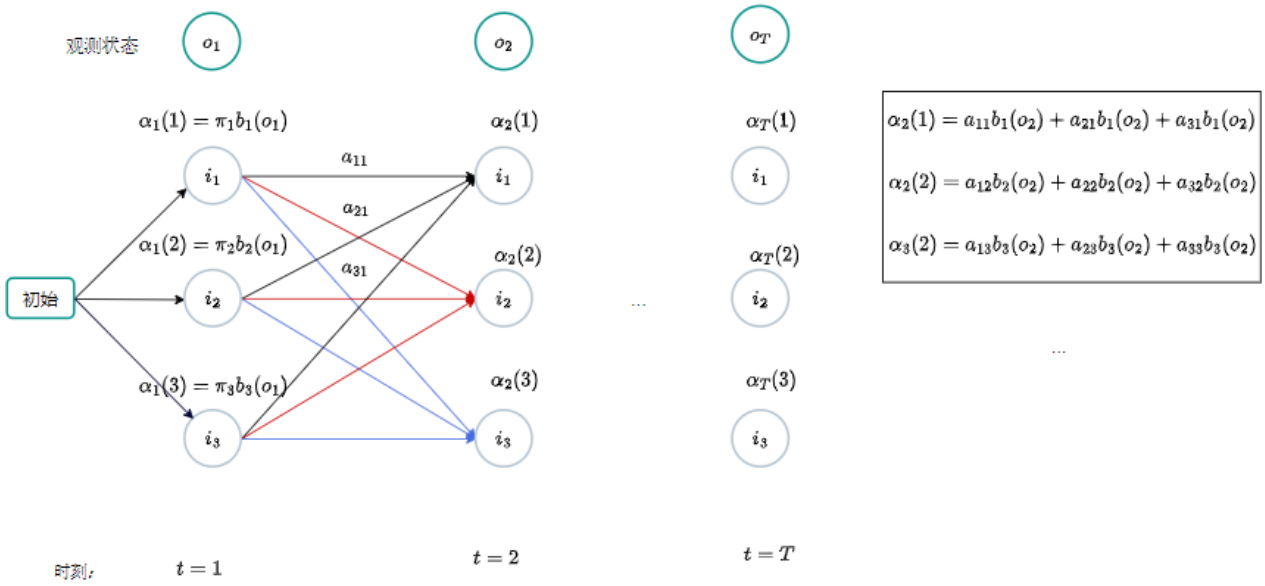
$$\alpha_1(i) = \pi_i b_i\left(o_1\right), \quad i = 1, 2, \cdots, N$$

第二步：递推，即计算出从第二个观测值开始到第 T 个观测值的所有可能状态序列的概率。

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

第三步：终止。

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



2.1.2 后向算法

后向算法的具体过程为：

第一步：初始化后向概率，对最终时刻的所有状态q_i规定$\beta_T(i) = 1$ 。

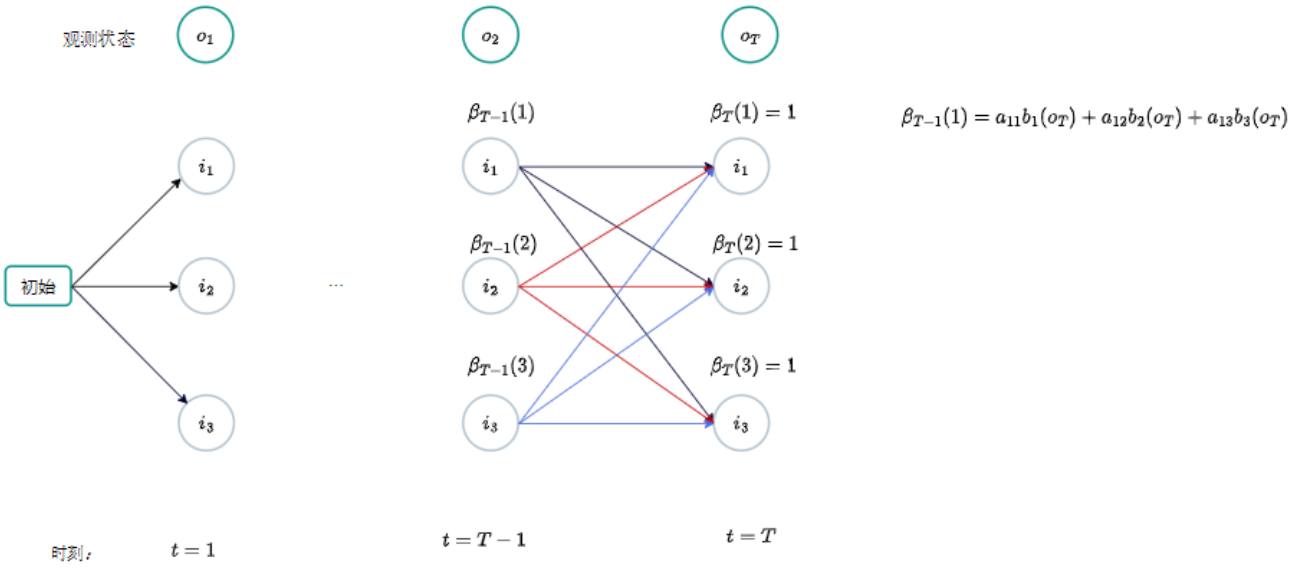
$$\beta_T(i) = 1, \quad i = 1, 2, \cdots, N$$

第二步：后向概率的递推公式。为了计算在时刻t状态为q_i条件下时刻$t+1$之后的观测序列为$o_{t+1}, o_{t+2} \ldots o_T$的后向概率 $\beta_t(i)$ ，只需考虑在时刻$t+1$所有可能的N个状态 q_j的转移概率(即a_{ij}项)，以及在此状态下的观测o_{t+1}的观测概率(即$b_j(o_{t+1})$项)，然后考虑状态q_j之后的观测序列的后向概率(即$\beta_{t+1}(j)$项。

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

第三步：求 $P(O\mid\lambda)$的思路与第二步一致，只是初始概率π_i，代替转移概率。

$$P(O\mid\lambda)=\sum_{i=1}^{N}\pi_i b_i\left(o_1\right)\beta_1(i)$$



2.2 Viterbi算法

用动态规划(dynamic programming)解隐马尔科夫模型预测问题，即用动态规划求概率最大路径（最优路径），该路径对应着一个状态序列。

最优路径具有特性：全局最优必是局部最优；依据这一原理，只需从时刻$t=1$开始，递推地计算在时刻t 状态为i 的各条部分路径的最大概率，直到得到时刻$t=T$ 状态为i 的各条路径的最大概率。时刻$t=T$ 的最大概率即为最优路径的概率P^*，最优路径的终结点i_T^*也同时得到。之后，为了找出最优路径的各个节点，从终结点i_T^* 开始，由后向前逐步求得节点$i_{T-1}^*,\ldots,i_1^*$ 得到最优路径$I=i_1^*,i_2^*,\ldots,i_n^*$ 。

定义变量：

- δ ，时刻t 状态为i 的所有单个路径$(i_1,i_2,\ldots i_t)$ 中概率最大值为

$$\delta_t(i)\quad=\max_{i_1,i_2,\cdots,i_{t-1}}P\left(i_t=i,i_{t-1},\cdots,i_1,o_t,\cdots,o_1\mid\lambda\right),\quad i=1,2,\cdots,N$$

变量δ 的递推公式为：

$$\begin{aligned}\delta_{t+1}(i)&=\max_{i_1,i_2,\cdots,i_t}P\left(i_{t+1}=i,i_t,\cdots,i_1,o_{t+1},\cdots,o_1\mid\lambda\right)\\&=\max_{1\leqslant j\leqslant N}\left[\delta_t(j)a_{ji}\right]b_i\left(o_{t+1}\right),\quad i=1,2,\cdots,N;t=1,2,\cdots,T-1\end{aligned}$$

- Ψ ，时刻t 状态为i 的所有单个路径$(i_1, i_2, \ldots i_t)$ 中概率最大的路径的第$t-1$ 个结点为

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

维特比算法具体过程：

输入：模型$\lambda = (A, B, \pi)$和观测序列$O = (o_1, o_2, \cdots, o_T)$;
输出：最优路径$I^* = (i_1^*, i_2^*, \cdots, i_T^*)$。

（1）初始化

$$\delta_1(i) = \pi_i b_i\left(o_1\right), \quad i = 1, 2, \cdots, N$$
$$\Psi_1(i) = 0, \quad i = 1, 2, \cdots, N$$

（2）递推。对 $t = 2, 3, \cdots, T$

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

（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 $t = T-1, T-2, \cdots, 1$

$$i_t^* = \Psi_{t+1}\left(i_{t+1}^*\right)$$

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

示例说明：

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 $\}$,

$$A = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5 & 0.5 \\ 0.4 & 0.6 \\ 0.7 & 0.3 \end{bmatrix}, \quad \pi = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \end{bmatrix}$$

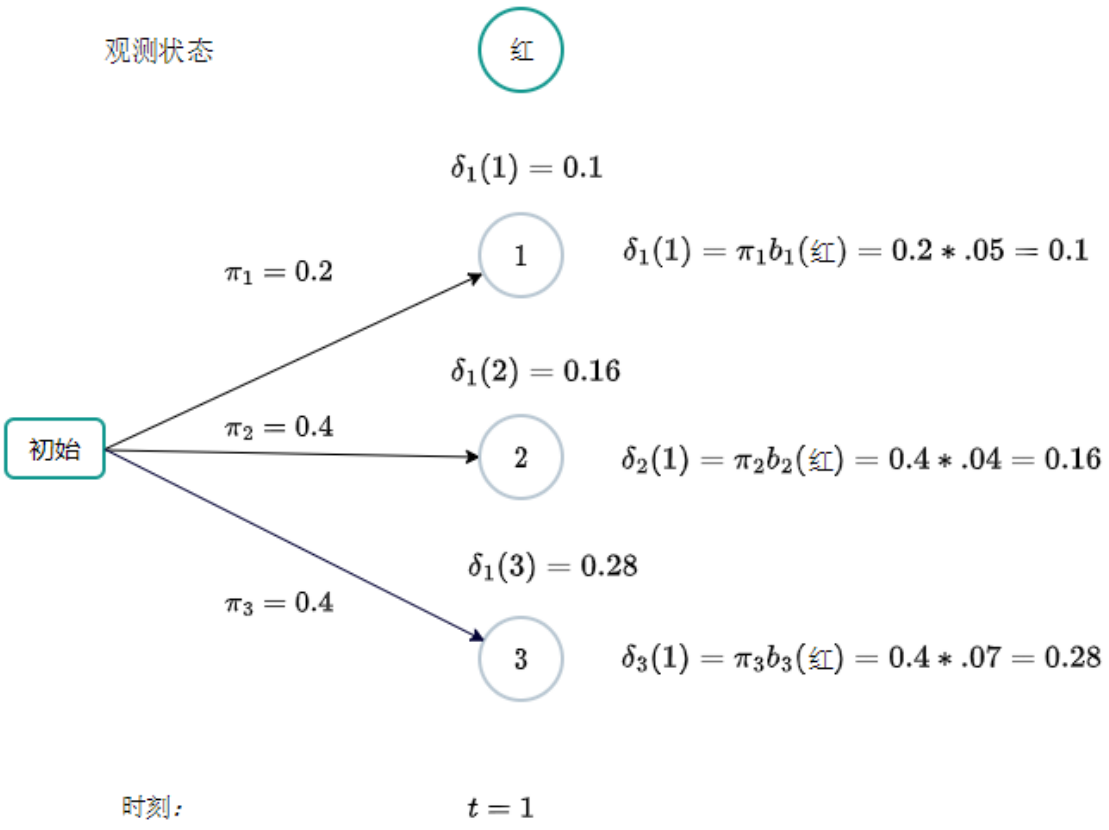
设 $T = 3, O = (\text{ 红, 白, 红})$, 试用前向算法计算 $P(O \mid \lambda)$ 。

(1) 初始化。在 $t = 1$ 时, 对每一个状态 $i, i = 1, 2, 3$, 求状态为 i 观测 o_1 为红的 概率, 记此概率为 $\delta_1(i)$, 则

$$\delta_1(i) = \pi_i b_i\left(o_1\right) = \pi_i b_i(\text{ 红}), \quad i = 1, 2, 3$$

代入实际数据

$$\delta_1(1) = 0.10, \quad \delta_1(2) = 0.16, \quad \delta_1(3) = 0.28$$



记 $\Psi_1(i) = 0, i = 1, 2, 3$ 。

(2) 在 $t = 2$ 时, 对每个状态 $i, i = 1, 2, 3$, 求在 $t = 1$ 时状态为 j 观测为红并在$t = 2$ 时状态为 i 观测 o_2 为白的路径的最大概率, 记此最大概率为 $\delta_2(i)$, 则

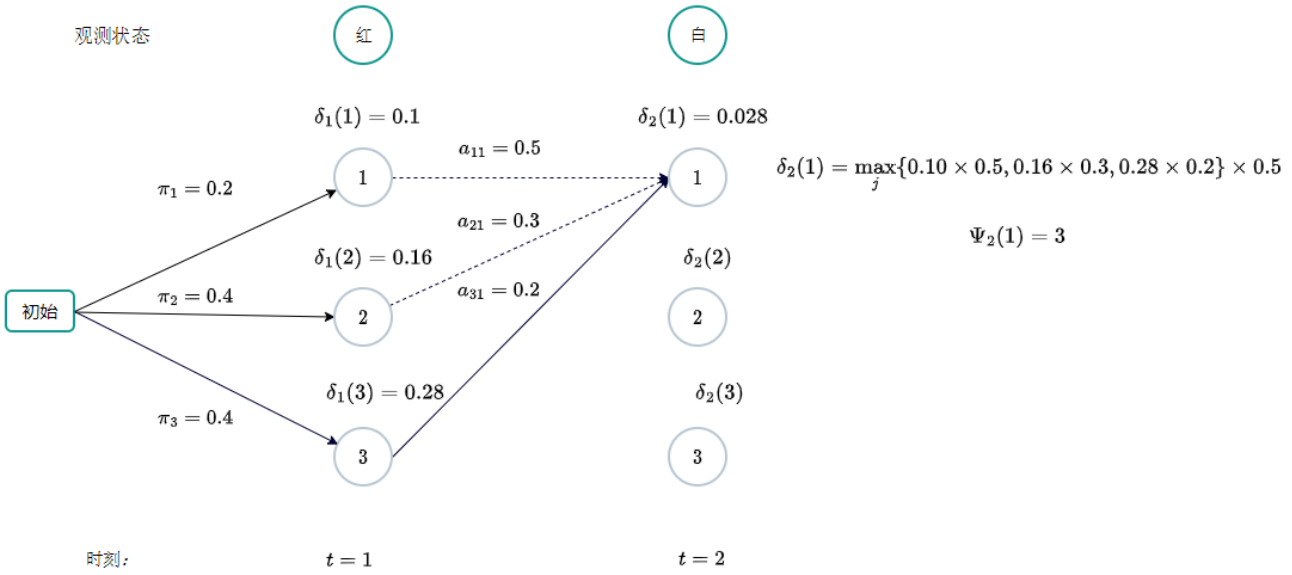
$$\delta_2(i) = \max_{1 \leqslant j \leqslant 3} [\delta_1(j) a_{ji}] b_i\left(o_2\right)$$

同时, 对每个状态 $i, i = 1, 2, 3$, 记录概率最大路径的前一个状态 j :

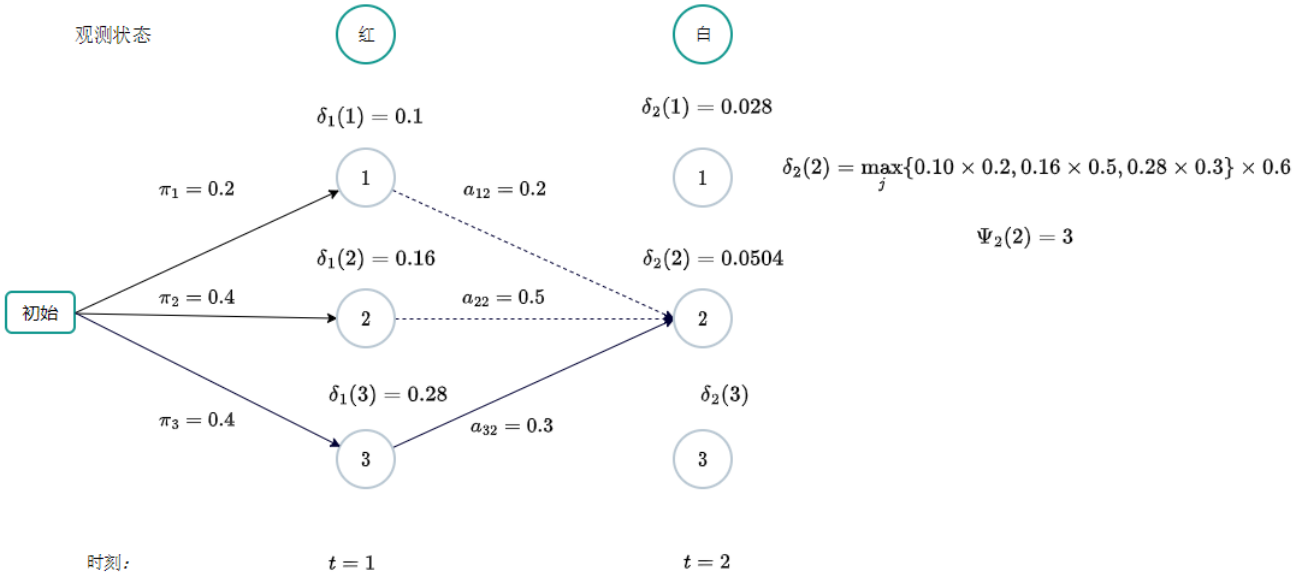
$$\Psi_2(i) = \arg \max_{1 \leqslant j \leqslant 3} [\delta_1(j) a_{ji}], \quad i = 1, 2, 3$$

计算:

$$\begin{aligned} \delta_2(1) &= \max_{1 \leqslant j \leqslant 3} [\delta_1(j) a_{j1}] b_1\left(o_2\right) \\ &= \max_j \{0.10 \times 0.5, 0.16 \times 0.3, 0.28 \times 0.2\} \times 0.5 \\ &= 0.028 \end{aligned}$$



$$
\begin{aligned}
\delta_2(2) &= \max_{1\leqslant j\leqslant 3}\left[\delta_1(j)a_{j1}\right]b_2\left(o_2\right) \\
&= \max_j\{0.10\times0.5,0.16\times0.3,0.28\times0.2\}\times0.5 \\
&= 0.0504 \\
\Psi_2(2) &= 3
\end{aligned}
$$

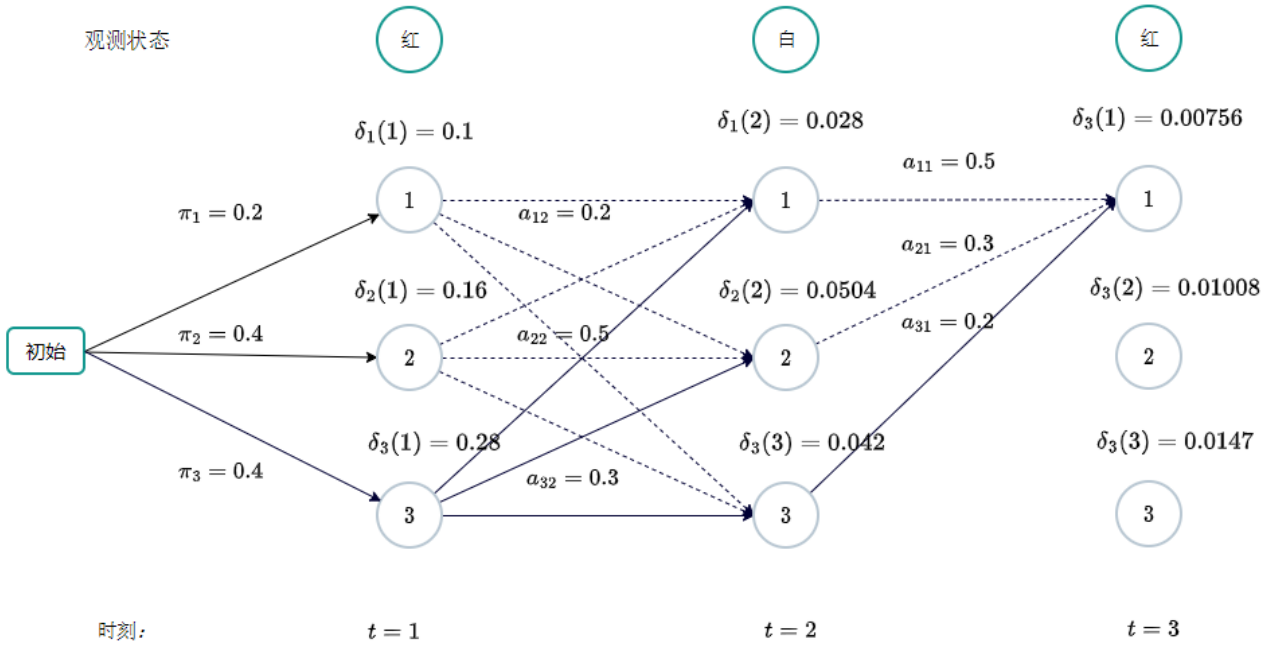


同理可得

$$
\begin{aligned}
\delta_2(3) &= 0.042 \\
\Psi_2(3) &= 3
\end{aligned}
$$

同样, 在 $t=3$ 时,

$$
\begin{aligned}
\Psi_3(i) &= \arg\max_{1\leqslant j\leqslant 3}\left[\delta_2(j)a_{ji}\right] \\
\delta_3(1) &= 0.00756 \\
\Psi_3(1) &= 2
\end{aligned}
$$



$$\delta_3(1)=\max_j\{0.028\times0.2,0.0504\times0.3,0.042\times0.2\}\times0.5$$

$$\Psi_3(1)=3$$

$$\delta_3(2)=0.01008$$
$$\Psi_3(2)=2$$
$$\delta_3(3)=0.0147$$
$$\Psi_3(3)=3$$

(3) 以 P^* 表示最优路径的概率, 则

$$P^*=\max_{1\leqslant i\leqslant 3}\delta_3(i)=0.0147$$

最优路径的终点是 :

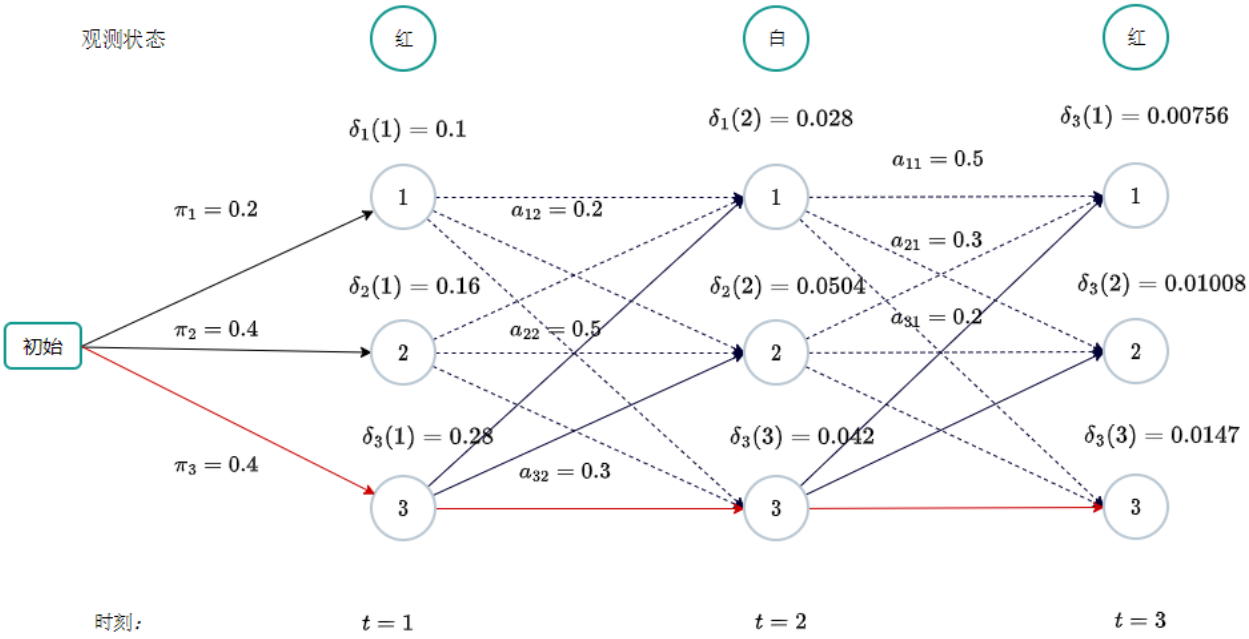
$$i_3^*=\arg\max_i\left[\delta_3(i)\right]=3$$

(4) 由最优路径的终点 i_3^*, 逆向找到 i_2^*, i_1^* :

在 $t=2$ 时, $\quad i_2^*=\Psi_3\left(i_3^*\right)=\Psi_3(3)=3$

在 $t=1$ 时, $\quad i_1^*=\Psi_2\left(i_2^*\right)=\Psi_2(3)=3$

于是求得最优路径, 即最优状态序列 $I^*=(i_1^*,i_2^*,i_3^*)=(3,3,3)$ 。



3.附件

matlab 代码实现：

```
% 参考
% 下面给出一个具体的例子来测试该函数的表现：
clc;
% pi = [0.6 0.4];
% A = [0.7 0.3;0.4 0.6];
% B = [0.1 0.4 0.5; 0.7 0.2 0.1];
% observations = [1 2 3];
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];
observations = [1, 2, 1, 1];

[path, prob] = viterbi(pi, A, B, observations);
disp(path); % [3 3 3]
disp(prob); % 0.0147

function [path, prob] = viterbi(pi, A, B, observations)
  % 输入：
  % pi: 初始状态分布（一个行向量）
  % A: 状态转移矩阵
```

```
    % B: 观测概率矩阵（行表示状态，列表示观测值）
    % observations: 观测序列（一个行向量）
    % 输出：
    % path: 最可能的状态序列（一个行向量）
    % prob: 对应的概率

    % 初始化
    T = length(observations);
    N = length(pi);
    delta = zeros(N, T);
    psi = zeros(N, T);
    path = zeros(1, T);

    % 第一步：计算初始概率
    delta(:, 1) = pi(:) .* B(:, observations(1));
    psi(:, 1) = 0;

    % 第二步：递推计算
    for i = 2:T

        for j = 1:N
            [delta(j, i), psi(j, i)] = max(delta(:, i - 1) .* A(:, j));
            delta(j, i) = delta(j, i) * B(j, observations(i));
        end

    end

    % 第三步：回溯得到最优路径
    [prob, path(T)] = max(delta(:, T));

    for i = T - 1:-1:1
        path(i) = psi(path(i + 1), i + 1);
    end

end
```

Python代码实现

```
import numpy as np

def viterbi(obs, states, start_p, trans_p, emit_p):
    """
    :param obs: list of observed values
    :param states: list of states
    :param start_p: dict of starting probabilities for each state
    :param trans_p: dict of transition probabilities between states
    :param emit_p: dict of observation probabilities for each state
    :return: tuple containing the most likely sequence of hidden states and the probability of that sequence
```

```
    """
    V = [{}]
    path = {}

    # initialize base cases at t=0
    for state in states:
        V[0][state] = start_p[state] * emit_p[state][obs[0]]
        path[state] = [state]

    # iterate over time steps t > 0
    for t in range(1, len(obs)):
        V.append({})
        new_path = {}

        # calculate the probability of arriving at each state at time t and store the most likely path to that state
        for current_state in states:
            (prob, state) = max((V[t-1][prev_state] * trans_p[prev_state][current_state] * emit_p[current_state]
[obs[t]], prev_state) for prev_state in states)
            V[t][current_state] = prob
            new_path[current_state] = path[state] + [current_state]

        # update the current most likely path to each state
        path = new_path

    # determine the final state with the highest probability
    (prob, state) = max((V[len(obs) - 1][current_state], current_state) for current_state in states)

    return (path[state], prob)

# define the possible states and observations
states = ['H', 'C']
obs = ['normal', 'cold', 'dizzy']

# define the initial probabilities, transition probabilities, and observation probabilities
start_p = {'H': 0.6, 'C': 0.4}
trans_p = {
    'H': {'H': 0.7, 'C': 0.3},
    'C': {'H': 0.4, 'C': 0.6}
}
emit_p = {
    'H': {'normal': 0.5, 'cold': 0.4, 'dizzy': 0.1},
    'C': {'normal': 0.1, 'cold': 0.3, 'dizzy': 0.6}
}

# decode the sequence of observations and determine the most likely sequence of hidden states
(result, probability) = viterbi(obs, states, start_p, trans_p, emit_p)

# print the result
print("The most likely sequence of hidden states is:", result)
```

```
print("The probability of the sequence is:", probability)
```

[1] 60分钟看懂HMM的基本原理

[2] Hidden Markov Models (HMM) - MATLAB & Simulink - MathWorks 中国