1、马尔科夫链

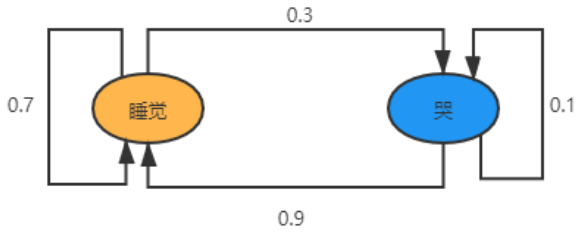
有向图模型（贝叶斯网络）：用有向图表示变量间的依赖关系；

无向图模型（马尔可夫网）：用无向图表示变量间的相关关系。

HMM 就是贝叶斯网络的一种--虽然它的名字里有和"马尔可夫网"一样的马尔可夫。

对变量序列建模的贝叶斯网络又叫动态贝叶斯网络。HMM 就是最简单的动态贝叶斯网络。

注意，马尔可夫过程其原始模型是马尔可夫链。该过程具有如下特性：在已知系统当前状态的条件下，它未来的演变不依赖于过去的演变。也就是说，一个马尔可夫过程可以表示为系统在状态转移过程中，第 T+1 次结果只受第 T 次结果的影响，即只与当前状态有关，而与过去状态，即与系统的初始状态和此次转移前的所有状态无关。



上图就是一个非常简单的**马尔可夫链**。两个节点分别表示睡和哭。几条边表示节点之间的转移概率。

睡之后，0.7 的可能又接着睡，只有 0.3 的可能变成哭。而哭之后，0.9 的可能是睡，也就有 0.1 的可能接着哭。

假设这是某个孩子的预报模型（这个孩子就只有哭和睡），则下一个状态只和上一个状态有关。和之前是在哭还是睡的状态没有关系。那么我们只要知道现在的状态，就可以推测接下来是睡还是哭的可能性了。

由马尔科夫链演化而成了隐含马尔可夫模型（Hidden Markov Model, HMM）！

2、HMM概述

2.1、HMM模型介绍

隐马尔科夫模型（Hidden Markov Model，以下简称HMM）是比较经典的机器学习模型了，它在语言识别，自然语言处理，模式识别等领域得到广泛的应用。

当然，随着目前深度学习的崛起，尤其是RNN，LSTM等神经网络序列模型的火热，HMM的地位有所下降。

但是作为一个经典的模型，学习HMM的模型和对应算法，对我们解决问题建模的能力提高以及算法思路的拓展还是很好的。

2.2、HMM模型应用

首先我们来看看什么样的问题解决可以用HMM模型。

使用HMM模型时我们的问题一般有这两个特征：

- 我们的问题是基于序列的，比如时间序列，或者状态序列。
- 我们的问题中有两类数据，一类序列数据是可以观测到的，即观测序列；而另一类数据是不能观察到的，即隐藏状态序列，简称状态序列。

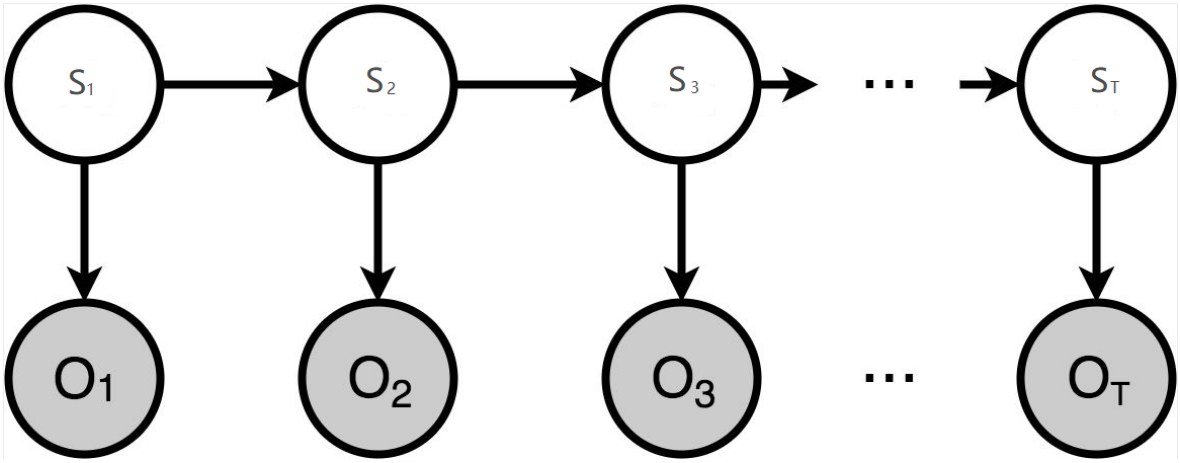
有了这两个特征，那么这个问题一般可以用HMM模型来尝试解决。这样的问题在实际生活中是很多的。比如：我现在在打字写博客，我在键盘上敲出来的一系列字符就是观测序列，而我实际想写的一段话就是隐藏序列，输入法的任务就是从敲入的一系列字符尽可能的猜测我要写的一段话，并把最可能的词语放在最前面让我选择，这就可以看做一个HMM模型了。再举一个，我在和你说话，我发出的一串连续的声音就是观测序列，而我实际要表达的一段话就是状态序列，你大脑的任务，就是从这一串连续的声音中判断出我最可能要表达的话的内容。

2.3、HMM模型定义

HMM 是一个关于时序的概率模型，它的变量分为两组：

- 状态变量$\{S_1, S_2, \ldots, S_T\}$，其中 S_T，表示 T 时刻的系统状态；
- 观测变量$\{O_1, O_2, \ldots, O_T\}$，其中 O_T ，表示 T时刻的观测值。

状态变量和观测变量各自都是一个时间序列，每个状态/观测值都和一个时刻相对应，如下图所示：



一般假定状态序列H_T是隐藏的，不能被观测到的，因此状态变量是隐变量，这就是 HMM中的 hidden 的来源。

这个隐藏的，不可观测的状态序列是由一个马尔可夫链随机生成的，这是 HMM 中的第一个 M 的含义。

一条隐藏的马尔可夫链随机生成了一个不可观测的状态序列 state sequence，然后每个状态又对应生成了一个观测结果。这些观测值按照时序排列后就成了观测序列 observation sequence。这两个序列是一一对应的，每个对应的位置又对应着一个时刻。

一般而言，HMM 的状态变量取值是离散的，而观测变量的取值，则可以是离散的，也可以是连续的。不过为了方便讨论，也因为在大多数应用中观测变量也是离散的，因此，我们下面仅讨论状态变量和观测变量都是离散的情况。

3、HMM模型基本假设

HMM 的定义建立在两个假设之上：

3.1、假设1：

假设隐藏的马尔可夫链在任意时刻 t 的状态只依赖于前一个时刻(t-1)的状态，与其它时刻的状态及观测无关，也与时刻 t 无关。用公式表达就是：

$$\mathrm{P(S_t|S_{t-1},O_{t-1},\ldots,S_1,O_1)=P(S_t|S_{t-1})}\quad \mathrm{t=1,2,\ldots,t}$$

这一假设又叫做**齐次马尔可夫假设**。

3.2、假设2：

假设任意时刻的观测只依赖于该时刻的马尔可夫链状态，与其它观测及状态无关。用公式表达就是：

$$\mathrm{P(O_t|S_r,O_r,\ldots,S_{t+1},O_{t+1},S_t,O_t,S_{t-1},O_{t-1},\ldots,S_1,O_1)=P(O_t|S_t)}$$

这叫观测**独立性假设**。

4、HMM模型解决三个问题

4.1、概率问题

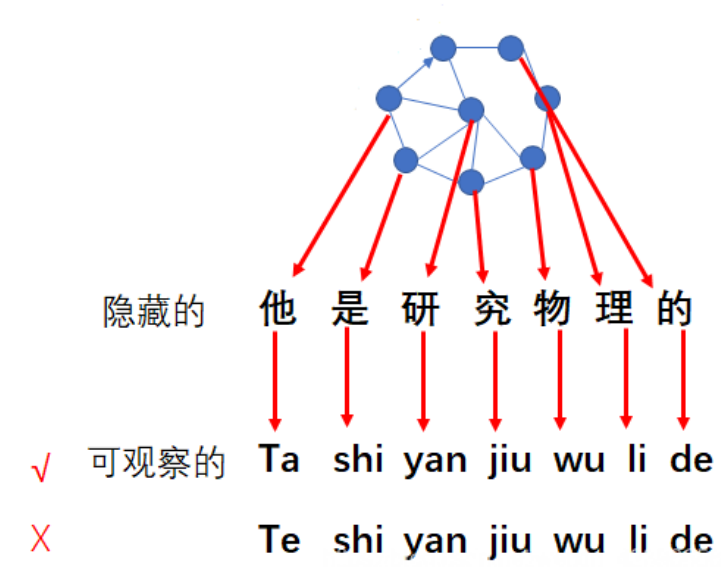
概率计算问题，又称为评估问题。

已知信息：

- 模型 $\lambda=[A,B,\Pi]$
- 观测序列 $\{O_1,O_2,\ldots,O_r\}$

求解目标：在给定模型 λ 下，计算观测序列 O 出现的概率：$P(O|\lambda)$？

下面举个语音识别的例子拿过来进行解释：



这里的任务呢就是我们语音输入得到可观察的序列，但是存在这样一个问题就是如果说话者普通话不标准或者环境干扰大，此时假如把ta发音te了，那么我们需要计算得到的可观察序列的概率有多大，而且我们把最大的那个当做本次的正确输入，例如如果本次输入的是下面一行的语音，但是通过解码后发现上面一行的概率最大， 因此就把上面的一行当做正确的语音输入。因此这就是第一个基本问题了，用处在语音识别中很明显。

4.2、预测问题

预测问题，又称为解码问题。

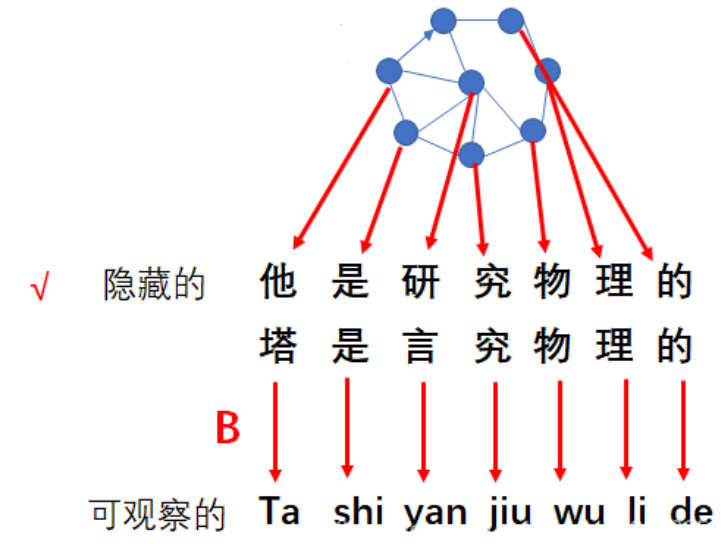
已知信息：

- 模型 $\lambda = [A, B, \Pi]$
- 观测序列 $\{O_1, O_2, \ldots, O_r\}$

给出观测序列O和模型$\lambda = [A, B, \Pi]$，选择一个状态序列S，能最好的解释观测序列O。

求解目标：计算在给定模型 λ 下，使已知观测序列 O 的条件概率 P(O|S)最大的状态序列$\{S_1, S_2, \ldots, S_T\}$，即给定观测序列，求最有可能与之对应的状态序列，使得该状态序列"最好地解释"观测序列。

下面举个语音识别的例子拿过来进行解释：



这里先默认语音的输入是正确的，如可观察序列是标准正确的，那么我们现在想通过可观察的序列即拼音，去寻找一个对应的文字（这就是语音识别的任务了），那么对应的句子那么多，哪个才是正确的呢？其实就是转态转移概率最大的那个句子就是最优的，如上图，隐藏层的第一行的汉子就是很好的"解释"了拼音且概率是最大的，而第二行就不是很好的解释，概率当然不是最大的，因此第二个基本问题就是这个意思，至于如何求这个概率，一般通过训练学习建模来构建算法模型。

4.3、学习问题

学习问题，又称为训练问题。

已知信息：

- 观测序列 $\{O_1, O_2, \ldots, O_r\}$

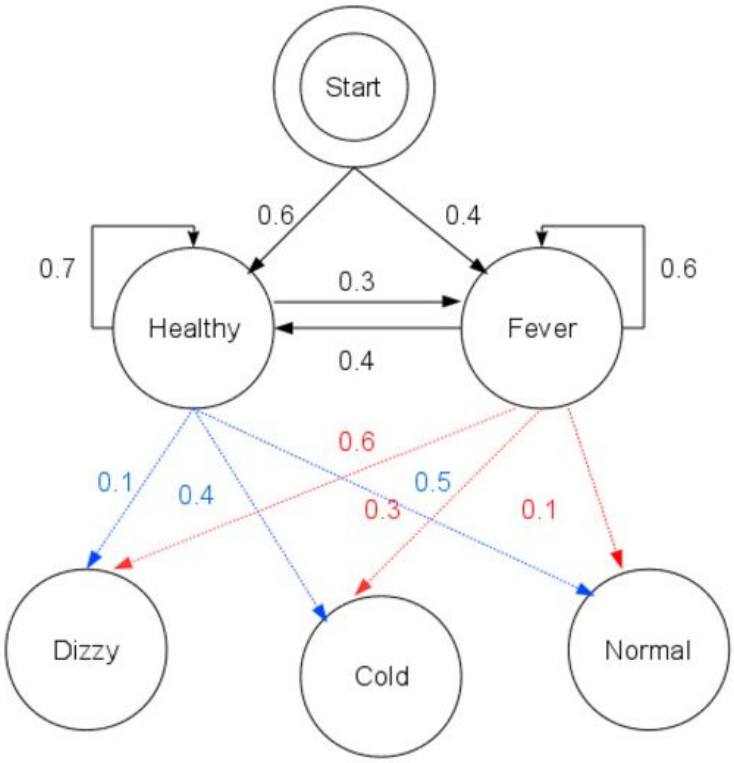
求解目标：估计模型$\lambda = [A, B, \Pi]$参数，使得该模型下观测序列概率$P(O|\lambda)$最大。也就是训练模型，使其最好地描述观测数据。

即根据最大似然估计调整模型$\lambda = [A, B, \Pi]$ 的参数，使得$P(O|\lambda)$最大!

5、HMM模型算法示例

5.1、问题概述

比如，给定的HMM模型参数已知，求出三天观察是(Dizzy,Cold,Normal)的概率是多少？对应的HMM模型参数已知的意思，就是说的A转移概率分布矩阵(trainsition_probability)，B观测状态矩阵(emission_probability)，Pi初始状态分布矩阵是已经知道的。



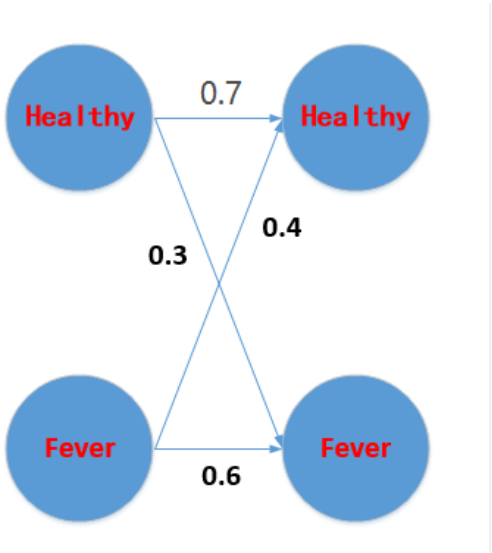
由上图所示，也就是说，可以写成如下代码：

```
trainsition_probability = [[0.7,0.3],[0.4,0.6]]
emission_probability = [[0.5,0.4,0.1],[0.1,0.3,0.6]]
pi = [0.6,0.4]
```

在第一个问题中，我们需要求解出三天观察是(Dizzy,Cold,Normal)的概率是多少?
这里为了演示简单，我只求解出俩天观察为(Dizzy,Cold)的概率是多少!

5.2、概率计算

这个问题太好求解了，最暴力的方法就是将路径全部遍历一遍。下面尽可能通俗易懂的说明一下：
首先画出时间序列状态图如下：



下面，详细走一遍一条路径的暴力算法，这样既可以避开公式的晦涩，也不失正确性。其它路径完全类似
第一天为Healthy的概率为：

$$P(Healthy) = 0.6$$

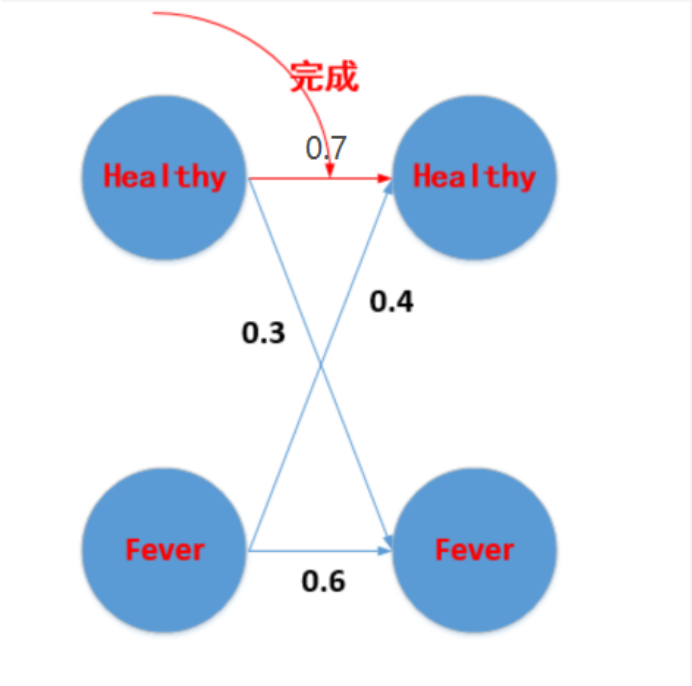
在第一天为Healthy的基础上，观察为Dizzy的概率为：

$$P(Dizzy|Healthy) = 0.6 \times P(Health >> Dizzy) = 0.6 \times 0.1 = 0.06$$

然后求出在第一天为Healthy的基础上，并且第一天表现为Dizzy的前提下，第二天也为Healthy的概率为：

$$P(Healthy|Healthy, Dizzy) = 0.06 \times 0.7 = 0.042$$

上面求完的时候，代表下图中的红线已经转移完了。



继续，在前面的基础上，第二天观察为Cold的概率为：

$$P(Cold|(Healthy, Dizzy), (Healthy)) = 0.06 \times 0.7 \times 0.4 = 0.0168$$

上面所求即是在第一天隐含状态为Healthy和第二天隐含状态为Healthy的基础上，观察序列为Dizzy，Cold的概率。现在我们已经完成一条路径的完整结果了。

5.3、所有路径概率计算

那么同理，我们就可以求出其它三条路径。

（1）在第一天隐含状态为Healthy和第二天隐含状态为Fever的基础上，观察序列为Dizzy，Cold的概率

$$P(Cold|(Healthy, Dizzy), (Fever)) = 0.06 \times 0.3 \times 0.3 = 0.0054$$

（2）在第一天隐含状态为Fever和第二天隐含状态为Healthy的基础上，观察序列为Dizzy，Cold的概率

$$P(Cold|(Fever, Dizzy), (Healthy)) = 0.24 \times 0.4 \times 0.4 = 0.00384$$

（3）在第一天隐含状态为Fever和第二天隐含状态为Fever的基础上，观察序列为Dizzy，Cold的概0.24率

$$P(Cold|(Fever, Dizzy), (Fever)) = 0.24 \times 0.6 \times 0.3 = 0.0432$$

5.4、最终概率计算

前面提出的问题（俩天观察为(Dizzy,Cold)的概率是多少？）的结果就是将这四个结果相加起来就可以了。是不是很简单！

$$P(Dizzy, Cold) = 0.0168 + 0.0054 + 0.003845 + 0.0432 = 0.069245$$

其实这个算法在现实中是不可行的。我给的例子由于是为了讲解容易，状态值和观察值都很小，但是实际中的问题，**隐状态的个数是非常大的。**

那么我们的计算量是不可以忍受的。

6、HMM模型前向算法示例

6.1、问题概述

这里我们用盒子与球的例子来显示前向概率的计算。

我们的观察集是：

$$\mathrm{V}=\{红，白\}，\mathrm{M}=2$$

我们的状态集合是：

$$\mathrm{Q}=\{盒子一，盒子二，盒子三\}，\mathrm{N}=3$$

球的颜色观测序列

$$\mathrm{O}=\{红，白，红\}$$

观察序列和状态序列的长度都是3。

初始状态分布为：

$$\Pi=[0.2,0.4,0.4]^{\mathrm{T}}$$

状态转移概率分布矩阵为（三个盒子之间的状态转移、转变）：

$$\mathrm{A}=\begin{bmatrix}0.5 & 0.2 & 0.3\\ 0.3 & 0.5 & 0.2\\ 0.2 & 0.3 & 0.5\end{bmatrix}$$

观测状态矩阵

$$\mathrm{B}=\begin{bmatrix}0.5 & 0.5\\ 0.4 & 0.6\\ 0.7 & 0.3\end{bmatrix}$$

6.2、时刻1（红球）

按照前向算法，首先计算时刻1三个状态的前向概率：

隐藏状态是盒子1的概率为：

$$\alpha_1(1)=\pi_1 b_1(o_1)=0.2\times 0.5=0.1$$

隐藏状态是盒子2的概率为：

$$\alpha_1(2)=\pi_2 b_2(o_1)=0.4\times 0.4=0.16$$

隐藏状态是盒子3的概率为：

$$\alpha_1(3)=\pi_3 b_3(o_1)=0.4\times 0.7=0.28$$

6.3、时刻2（白球）

现在，我们开始递推，首先递推时刻2三个状态的前向概率：

隐藏状态是盒子1的概率为：

$$\alpha_2(1)=[\sum_{i=1}^{3}\alpha_1(i)\times a_{i1}]\times b_1(o_2)$$

$$=[0.1*0.5+0.16*0.3+0.28*0.2]*0.5=0.077$$

隐藏状态是盒子2的概率为：

$$\alpha_2(2) = [\sum_{i=1}^{3} \alpha_1(i) \times a_{i2}] \times b_2(o_2)$$

$$= [0.1 * 0.2 + 0.16 * 0.5 + 0.28 * 0.3] * 0.6 = 0.1104$$

隐藏状态是盒子3的概率为：

$$\alpha_2(3) = [\sum_{i=1}^{3} \alpha_1(i) \times a_{i3}] \times b_3(o_2)$$

$$= [0.1 * 0.3 + 0.16 * 0.2 + 0.28 * 0.5] * 0.3 = 0.0606$$

6.4、时刻3（红球）

继续递推，现在我们递推时刻3三个状态的前向概率：

隐藏状态是盒子1的概率为：

$$\alpha_3(1) = [\sum_{i=1}^{3} \alpha_2(i) \times a_{i1}] \times b_1(o_3)$$

$$= [0.077 * 0.5 + 0.1104 * 0.3 + 0.0606 * 0.2] * 0.5 = 0.04187$$

隐藏状态是盒子2的概率为：

$$\alpha_3(2) = [\sum_{i=1}^{3} \alpha_2(i) \times a_{i2}] \times b_2(o_3)$$

$$= [0.077 * 0.2 + 0.1104 * 0.5 + 0.0606 * 0.3] * 0.4 = 0.035512$$

隐藏状态是盒子3的概率为：

$$\alpha_3(3) = [\sum_{i=1}^{3} \alpha_2(i) \times a_{i3}] \times b_3(o_3)$$

$$= [0.077 * 0.3 + 0.1104 * 0.2 + 0.0606 * 0.5] * 0.7 = 0.05284$$

6.5、最终概率计算

最终我们求出观测序列:O={红，白，红}的概率为：

$$P(O|\lambda)=\sum_{i=1}^{3}\alpha_3(i)$$

$$=0.04187+0.035512+0.05284$$

$$=0.130222$$

7、维特比算法

7.1、维特比算法概述

维特比算法是一个通用的解码算法，是基于动态规划的求序列最短路径的方法。

既然是动态规划算法，那么就需要找到合适的局部状态，以及局部状态的递推公式。在HMM中，维特比算法定义了两个局部状态用于递推。

第一个局部状态是在时刻 t 隐藏状态为 i 所有可能的状态转移路径 $(i_1,i_2,\ldots,i_t)$ 中的概率最大值。记为 $\delta_t(i)$:

$$\delta_t(i)=\max_{i_1,i_2,\ldots i_t}P(i_t=i,i_1,i_2,\ldots,i_{t-1},o_t,o_{t-1},\ldots,o_1|\lambda),\quad i=1,2,\ldots,$$

由$\delta_t(i)$ 的定义可以得到 δ 的递推公式:

$$\delta_{t+1}(i)=\max_{i_1,i_2,\ldots i_t}P(i_{t+1}=i,i_1,i_2,\ldots,i_t,o_{t+1},o_t,,\ldots,o_1|\lambda)$$

$$=\max_{1\le j\le N}[\delta_t(j)\times a_{ji}]\times b_i(o_{t+1})$$

第二个局部状态由第一个局部状态递推得到。

我们定义在时刻 t 隐藏状态为 i 的所有单个状态转移路径 $(i_1,i_2,\ldots,i_N)$ 中概率最大的转移路径中第 t-1 个节点的隐藏状态为 $\psi_t(i)$,其递推表达式可以表示为:

$$\psi_t(i)=arg\max_{1\le j\le N}[\delta_{t-1(j)\times a_{ji}}]$$

有了这两个局部状态，我们就可以从时刻0一直递推到时刻 T，然后利用 $\psi_t(i)$ 记录的前一个最可能的状态节点回溯，直到找到最优的隐藏状态序列。

7.2、维特比算法流程

现在我们来总结下维特比算法的流程:

输入：HMM模型 $\lambda=(A,B,\Pi)$，观测序列 $O=(o_1,o_2,\ldots,o_T)$

输出：最有可能的隐藏状态序列 $I^*=i_1^*,i_2^*,\ldots,i_T^*$

1、初始化局部状态

$$\delta_1(i)=\pi_i\times b_i(o_1),i=1,2,\ldots,N$$

$$\psi_1(i)=0,i=1,2,\ldots,N$$

2、进行动态规划递推时刻 t=2,3,...T 时刻的局部状态:

$$\delta_t(i)=\max_{1\le j\le N}[\delta_{t-1}(j)\times a_{ji}]\times b_i(o_t),i=1,2,\ldots,N$$

$$\psi_t(i)=arg\max_{1\le j\le N}[\delta_{t-1}\times a_{ji}],i=1,2,\ldots,N$$

3、计算时刻 T 最大的$\delta_T(i)$，即为最可能隐藏状态序列出现的概率。计算时刻 T 最大的$\psi_T(i)$，即为时刻 T 最可能的隐藏状态。

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

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

4、利用局部状态 $\psi(i)$ 开始回溯。对于 t = T−1，T−2，...，1：

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

最终得到最有可能的隐藏状态序列：

$$I^* = i_1^*, i_2^*, \ldots, i_T^*$$

7.3、维特比算法示例

7.3.1、问题概述

这里我们用盒子与球的例子来显示前向概率的计算。

我们的观察集是：

$$\mathrm{V} = \{红，白\}，\mathrm{M} = 2$$

我们的状态集合是：

$$\mathrm{Q} = \{盒子一，盒子二，盒子三\}，\mathrm{N} = 3$$

球的颜色观测序列

$$\mathrm{O} = \{红，白，红\}$$

观察序列和状态序列的长度都是3。

初始状态分布为：

$$\Pi = [0.2, 0.4, 0.4]^{\mathrm{T}}$$

状态转移概率分布矩阵为（三个盒子之间的状态转移、转变）：

$$\mathrm{A} = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}$$

观测状态矩阵

$$\mathrm{B} = \begin{bmatrix} 0.5 & 0.5 \\ 0.4 & 0.6 \\ 0.7 & 0.3 \end{bmatrix}$$

7.3.2、时刻1（红球）

按照维特比算法，首先需要得到三个隐藏状态在时刻1时对应的各自两个局部状态，此时观测状态为1：

隐藏状态是盒子1的概率为：

$$\delta_1(1) = \pi_1 b_1(o_1) = 0.2 \times 0.5 = 0.1$$

隐藏状态是盒子2的概率为：

$$\delta_1(2) = \pi_2 b_2(o_1) = 0.4 \times 0.4 = 0.16$$

隐藏状态是盒子3的概率为：

$$\delta_1(3) = \pi_3 b_3(o_1) = 0.4 \times 0.7 = 0.28$$

$$\psi_1(1) = \psi_1(2) = \psi_1(3) = 0$$

7.3.3、时刻2（白球）

现在开始递推三个隐藏状态在时刻2时对应的各自两个局部状态，此时观测状态为2：

$$\delta_2(1) = \max_{1 \le j \le 3}[\delta_1(j) \times a_{j1}] \times b_1(o_2)$$

$$= \max_{1 \le j \le 3}[0.1 \times 0.5, 0.16 \times 0.3, 0.28 \times 0.2] \times 0.5$$

$$= 0.028$$

$$\psi_2(1) = 3$$

$$\delta_2(2) = \max_{1 \le j \le 3}[\delta_1(j) \times a_{j2}] \times b_2(o_2)$$

$$= \max_{1 \le j \le 3}[0.1 \times 0.2, 0.16 \times 0.5, 0.28 \times 0.5] \times 0.6$$

$$= 0.0504$$

$$\psi_2(2) = 3$$

$$\delta_2(3) = \max_{1 \le j \le 3}[\delta_1(j) \times a_{j3}] \times b_3(o_2)$$

$$= \max_{1 \le j \le 3}[0.1 \times 0.3, 0.16 \times 0.2, 0.28 \times 0.5] \times 0.6$$

$$= 0.042$$

$$\psi_2(3) = 3$$

7.3.4、时刻3（白球）

继续递推三个隐藏状态在时刻3时对应的各自两个局部状态，此时观测状态为1：

$$\delta_3(1) = \max_{1 \le j \le 3}[\delta_2(j) \times a_{j1}] \times b_1(o_3)$$

$$= \max_{1 \le j \le 3}[0.028 \times 0.5, 0.0504 \times 0.3, 0.042 \times 0.2] \times 0.5$$

$$= 0.00756$$

$$\psi_3(1) = 2$$

$$\delta_3(2) = \max_{1 \le j \le 3}[\delta_2(j) \times a_{j2}] \times b_2(o_3)$$

$$= \max_{1 \le j \le 3}[0.028 \times 0.2, 0.0504 \times 0.5, 0.042 \times 0.3] \times 0.4$$

$$= 0.01008$$

$$\psi_3(2) = 2$$

$$\delta_3(3) = \max_{1 \le j \le 3}[\delta_2(j) \times a_{j3}] \times b_3(o_3)$$

$$= \max_{1 \le j \le 3}[0.028 \times 0.3, 0.0504 \times 0.2, 0.042 \times 0.5] \times 0.7$$

$$= 0.0147$$

$$\psi_3(3) = 3$$

7.3.5、状态回溯

此时已经到最后的时刻，我们开始准备回溯。

- 此时最大概率为$\delta_3(3) = 0.0147$，从而得到 $i_3^* = 3$。

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

- 由于$\psi_3(3) = 3$，所以 $i_2^* = 3$
- 由于$\psi_2(3) = 3$，所以 $i_1^* = 3$

从而最终的最可能隐藏状态序列为：[3,3,3]，表示球的观测序列 $O = \{红，白，红\}$ 最有可能出现的的盒子是：[盒子三，盒子三，盒子三]。

8、代码演练

8.1、参数估计问题

参数估计问题，也叫学习问题。已知观察序列，来对HMM模型的参数进行估计。

```
'''
下面抽取5次，得到已知的观察序列，
来对HMM的参数进行估计，即使用MultinomialHMM进行参数的训练
'''
import numpy as np
import hmmlearn.hmm as hmm

states = ['盒子1', '盒子2', '盒子3']
obs = ['红球', '白球'] # 0表示红球，1表示白球
n_states = len(states)
m_obs = len(obs)

model = hmm.MultinomialHMM(n_components=n_states, n_iter=10, tol=0.001,algorithm='map')
X2 = np.array([
    [0, 1, 0, 0, 1],
```

```
    [0, 0, 0, 1, 1],
    [1, 1, 0, 1, 0],
    [0, 1, 0, 1, 1],
    [0, 0, 0, 1, 0]
])
model.fit(X2)
print("输出根据数据训练出来的π")
print(model.startprob_.round(3))
print("输出根据数据训练出来的A")
print(model.transmat_.round(3))
print("输出根据数据训练出来的B")
print(model.emissionprob_.round(3))
# 每次运行结果可能会不同
'''
输出根据数据训练出来的π
[0.296 0.702 0.002]
输出根据数据训练出来的A
[[0.313 0.302 0.384]
 [0.293 0.272 0.435]
 [0.358 0.382 0.26 ]]
输出根据数据训练出来的B
[[0.648 0.352]
 [0.744 0.256]
 [0.298 0.702]]
'''
```

8.2、解码问题

已知观察序列，求什么样的隐藏状态序列最可能生成一个给定的观察序列

```
import numpy as np
import hmmlearn.hmm as hmm

# 首先定义变量
status = ['盒子1', '盒子2', '盒子3']   # 隐藏的状态集合
obs = ['红球', '白球']     # 观察值集合
n_status = len(status)    # 隐藏状态的长度
m_obs = len(obs)          # 观察值的长度

# 初始概率分布： π 表示初次抽时，抽到1盒子的概率是0.2，抽到2盒子的概率是0.4，抽到3盒子的概率是0.4
start_probability = np.array([0.2, 0.4, 0.4])

# 状态转移概率矩阵 A[0][0]=0.5 表示当前我抽到1盒子，下次还抽到1盒子的概率是0.5
transition_probability = np.array([[0.5, 0.2, 0.3],
                                   [0.3, 0.5, 0.2],
                                   [0.2, 0.3, 0.5]])

# 观测概率矩阵 B：B[2][0]=0.7,表示第三个盒子抽到红球概率0.7，B[2][1]=0.3，表示第三个盒子抽到白球概率0.3
emission_probalitity = np.array([[0.5, 0.5],
                                 [0.4, 0.6],
                                 [0.7, 0.3]])

# 下面开始定义模型
'''
hmmlearn中主要有两种模型，分布为：GaussianHMM和MultinomialHMM；
如果观测值是连续的，那么建议使用GaussianHMM，否则使用MultinomialHMM

参数：
初始的隐藏状态概率π参数为: startprob；
状态转移矩阵A参数为: transmat;
状态和观测值之间的转移矩阵B参数为:
emissionprob_(MultinomialHMM模型中)或者在GaussianHMM模型中直接给定均值(means)和方差/协方差矩阵(covars)

'''
# 观测值，球是红或者黑，是离散的，n_status隐藏状态的长度
model = hmm.MultinomialHMM(n_components=n_status)
model.startprob_ = start_probability
model.transmat_ = transition_probability
model.emissionprob_ = emission_probalitity

'''
下面运行viterbi预测问题。已知观察序列（红球 白球 红球），
求什么样的隐藏状态序列（盒子2 盒子3 盒子2）最可能生成一个给定的观察序列。

status = ['盒子1', '盒子2', '盒子3']
obs = ['红球', '白球']
```

```
'''
se = np.array([[0, 1, 0]]).T   # （红球 白球 红球）
logprob, box_index = model.decode(se, algorithm='viterbi')
print("颜色:", end="")
print(" ".join(map(lambda t: obs[t], [0, 1, 0])))
print("盒子:", end="")
print(" ".join(map(lambda t: status[t], box_index)))
print("概率值:", end="")
print(np.exp(logprob)) # 这个是因为在hmmlearn底层将概率进行了对数化，防止出现乘积为0的情况
'''
颜色:红球 白球 红球
盒子:盒子3 盒子3 盒子3
概率值:0.014699999999999996
'''
```

8.3、股票走势预测

8.3.1、加载数据

```
import pandas as pd
import numpy as np
from hmmlearn.hmm import GaussianHMM
import warnings
warnings.filterwarnings('ignore')

data = pd.read_csv('apple_stock.csv')
print(data.shape)
data.head()
```

8.3.2、日期转换

```
# 日期格式转换
data['Date'] = pd.to_datetime(data['Date'], format = '%Y/%m/%d', errors = 'ignore')
data.info()
data.head()
```

8.3.3、数据提取

```
# 去除第一天的数据,因为第一天会被用来计算第二天的涨跌值特征，所以马尔可夫链实际是从第二天开始训练的。
close_v = data['Close'].values
volume = data['Volume'].values[1:]

# 计算数组中前后两个值差额
diff  = np.diff(close_v)
close_v = close_v[1:]

X = np.column_stack([diff,volume])
X.shape
```

8.3.4、算法建模

n_components 参数指定了使用3个隐藏层状态，表示股票：涨、平、跌三种状态

covariance_type定义了协方差矩阵类型为对角线类型，即每个特征的高斯分布有自己的方差参数，相互之间没有影响

n_iter参数定义了最大迭代次数

```
model =  GaussianHMM(n_components=3, covariance_type='diag',n_iter=100000)
model.fit(X)
print('每个隐含层均值和方差数据：')
for i in range(model.n_components):
    print('第{0}号隐藏状态'.format(i))
    print('mean = ', model.means_[i])
    print('var = ', np.diag(model.covars_[i]))
    print('---------------')

'''
每个隐含层均值和方差数据：
第0号隐藏状态
mean =  [-1.14524842e-01  1.23845771e+08]
var =  [3.84678157e+00 2.24454700e+15]
---------------
第1号隐藏状态
mean =  [1.68138743e-01 2.79598061e+07]
var =  [1.00638562e+00 5.12991123e+13]
---------------
第2号隐藏状态
```

```
mean =  [5.04087051e-02 5.66296542e+07]
var =  [2.81382699e+00 2.27530289e+14]
---------------
'''
```

因为可见层输入采用了两个输入特征（涨跌幅、成交量），所以每个结点有两个元素，分别代表该状态的涨跌幅均值、成交量均值。

由于系统的目标预测涨跌幅，所以这里只关心第一个特征的均值，有如下结论：

- 状态0的均值是-1.14524842e-01，约为-0.1145，认为该状态是**跌**。
- 状态1的均值是1.68138743e-01，约为0.1681，认为该状态是**涨**。
- 状态2的均值是5.04087051e-02，约为0.0504，认为该状态是**平**。

由涨跌幅特征的方差，可以得到的信息为：

- 状态**跌**的方差为3.845，是三个状态中方差最大的，也就是说该状态的预测不是特别可信。
- 状态**涨**的方差为1.006，是三个状态中方差最小的，也就是说该状态的预测非常可信。
- 状态**平**的方差为2.814，可信居中。

8.3.5、状态转移

```
print('Transition matrix')
print(model.transmat_.round(3))
'''
Transition matrix
[[0.898 0.    0.102]
 [0.    0.915 0.085]
 [0.083 0.043 0.874]]
'''
```

第一行最大的数值是0.915，因此**跌**倾向于保持自己的状态，即第二天仍旧为**跌**。

第二行最大的数值是0.898，得知**涨**后第二天倾向于变为**涨**。

根据第三行转变状态最大数值0.874，平后第二天仍旧倾向于**平**。