一站式解决：隐马尔可夫模型（HMM）全过程推导及实现

AI科技大本营 2019-10-19

作者 | 永远在你身后

转载自知乎用户永远在你身后

> 【导读】隐马尔可夫模型（Hidden Markov Model，HMM）是关于时许的概率模型，是一个生成模型，描述由一个隐藏的马尔科夫链随机生成不可观测的状态序列，每个状态生成一个观测，而由此产生一个观测序列。

定义抄完了，下面我们从一个简单的生成过程入手，顺便引出HMM的参数。

假设有4个盒子，每个盒子里面有不同数量的红、白两种颜色的球，具体如下表：

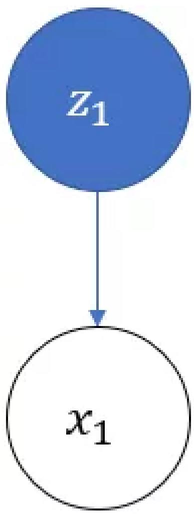
盒子编号	1	2	3	4
红球数	5	3	6	8
白球数	5	7	4	2

本栗子引用自《统计学习方法》

现在从这些盒子中抽取若干（ T ）个球，每次抽取后记录颜色，再放回原盒子，采样的规则如下：

开始时，按照一个**初始概率分布**随机选择第一个盒子，这里将第一个盒子用 z_1 表示：

将z_1的值用变量q_i 表示。因为有4个盒子可共选择，所以 $i \in \{1,2,3,4\}$。然后随机从该盒子中抽取一个球，使用x_1表示：



将x_1 的值用变量 v_j 表示。因为只有两种球可供选择，所以$j \in \{1,2\}$ 。一共有4个箱子，2种球，结合前面的箱子的详细数据，可以得到从每一个箱子取到各种颜色球的可能性，用一个表格表示：

	v_1	v_2
q_1	0.5	0.5
q_2	0.3	0.7
q_3	0.6	0.4
q_4	0.8	0.2

进一步，可以用一个矩阵（称为观测概率矩阵，也有资料叫做发射矩阵）来表示该表

$$B = [b_{ij}]_{4\times 2}$$

其中 b_{ij} 表示在当前时刻给定 的条件下，给定 $z_t = q_i$ 的条件下， $x_t = v_j$ 的概率：

t 表示当前的时刻，例如现在是第1时刻；然后是前面标注的**初始概率分布**，这个概率分布可以用一个向量（称作初始状态概率向量）来表示：

$$\pi = [\pi_i]_4^T$$

其中的 π_i 表示 z_1 取各个值的概率：

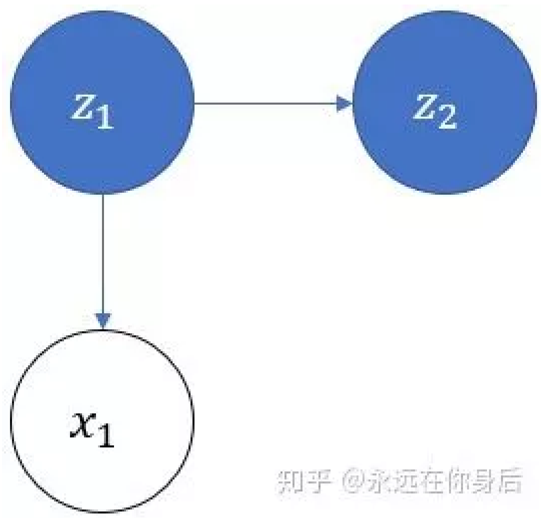
$$\pi_i = P(z_1 = q_i), \qquad i = 1, 2, 3, 4$$

例如该分布是均匀分布的话，对应的向量就是

$$\pi = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}^T$$

记录抽取的球的颜色后将其放回，然后在按照如下规则选择下一个盒子（ z_2 ）：

> 如果当前是盒子1，则选择盒子2
> 如果当前是盒子2或3，则分布以概率0.4和0.6选择前一个或后一个盒子
> 如果当前是盒子4，则各以0.5的概率停留在盒子4或者选择盒子3



同样，也可以根据以上规则做出一个表格，其中首列表示当前盒子，首行表示下一个盒子

	q_1	q_2	q_3	q_4
q_1	0	1	0	0
q_2	0.4	0	0.6	0
q_3	0	0.4	0	0.6
q_4	0	0	0.5	0.5

同样使用一个矩阵（称为状态转移矩阵）来表示上表

$$A=[a_{ij}]_{4\times 4}$$

其中 a_{ij} 表示在当前时刻处于状态 q_i 的条件下，到下一时刻转移到状态 q_j 的概率

$$a_{ij}=P(z_{t+1}=q_j|z_t=q_i),\qquad i=1,2,3,4;\quad j=1,2,3,4$$

以上，生成过程的主要流程就介绍完了，简单概括就是：盒子，取球，盒子，取球......直到生成指定数量（T）的数据后停止。如果对这个过程还有不太理解的话，可以看看文章开头给出的关于马尔科夫链的链接。

现在，整理一下参数：有两个矩阵，一个向量：

$$\begin{aligned}A&=[a_{ij}]_{N\times N}\\B&=[b_{ij}]_{N\times M}\\\pi&=[\pi_i]_N^T\end{aligned}$$

其中N表示隐变量z的状态数量，M表示观测变量x可能的取值数量，在后面的讨论中，用 λ 表示所有的参数。下面，根据这个栗子，写一个数据生成代码

```
import numpy as np

class HMM(object):
    def __init__(self, N, M, pi=None, A=None, B=None):
        self.N = N
```

```python
        self.M = M
        self.pi = pi
        self.A = A
        self.B = B

    def get_data_with_distribute(self, dist): # 根据给定的概率分布随机返回数据（索
        r = np.random.rand()
        for i, p in enumerate(dist):
            if r < p: return i
            r -= p

    def generate(self, T: int):
        '''
        根据给定的参数生成观测序列
        T: 指定要生成数据的数量
        '''
        z = self.get_data_with_distribute(self.pi)    # 根据初始概率分布生成第一个
        x = self.get_data_with_distribute(self.B[z])  # 生成第一个观测数据
        result = [x]
        for _ in range(T-1):        # 依次生成余下的状态和观测数据
            z = self.get_data_with_distribute(self.A[z])
            x = self.get_data_with_distribute(self.B[z])
            result.append(x)
        return result

if __name__ == "__main__":
    pi = np.array([.25, .25, .25, .25])
    A = np.array([
        [0,  1,  0, 0],
        [.4, 0, .6, 0],
        [0, .4, 0, .6],
        [0, 0, .5, .5]])
    B = np.array([
        [.5, .5],
        [.3, .7],
        [.6, .4],
        [.8, .2]])
    hmm = HMM(4, 2, pi, A, B)
    print(hmm.generate(10))  # 生成10个数据

```

```
# 生成结果如下
[0, 0, 1, 1, 1, 1, 0, 1, 0, 0]    # 0代表红球，1代表白球
```

现在，参数介绍完了，数据生成过程也了解了，接下来就是解决HMM的基本问题了，一共有三个

- **概率计算：给定参数** $\lambda=(\pi,A,B)$ **和观测序列** $X=(x_1,x_2,\dots,x_T)$ **，计算观测序列** X **的条件概率** $P(X|\lambda)$
- **参数学习：给定观测序列** X **，反推参数** π,A,B
- **解码问题：给定参数** λ **和观测序列** $X=(x_1,\dots,x_T)$ **，求可能性最大的** $Z=(z_1,\dots,z_T)$

不过，在讨论这三个问题的相关算法之前，首先要给出两个假设，在后面的推导过程中会不断的用到：

齐次马尔可夫假设：即任意时刻的状态**只依赖**于前一时刻的状态，与其他时刻的状态无关（当然，初始时刻的状态由参数π决定）：

$$P(z_t|z_{t-1},z_{t-2},\dots,z_1,x_t,\dots,x_1)=P(z_t|z_{t-1}),\qquad t=2,3,\dots,T$$

观测独立假设：即任意时刻的观测只依赖于该时刻的状态，与其他无关：概率计算算法

$$P(x_t|x_{t-1},x_{t-2},\dots,x_1,z_t,z_{t-1},\dots,z_1)=P(x_t|z_t),\qquad t=1,2,\dots,T$$

概率计算算法

现在，来看第一个问题，关于概率的计算，由于存在隐变量，所以X的边际概率需要将所有的联合概率 $P(X,Z)$ 加和得到：

$$P(X|\lambda)=\sum_Z P(X,Z|\lambda)\qquad(1)$$

由于给出了T个观测数据，所以相应的状态也有T 个：

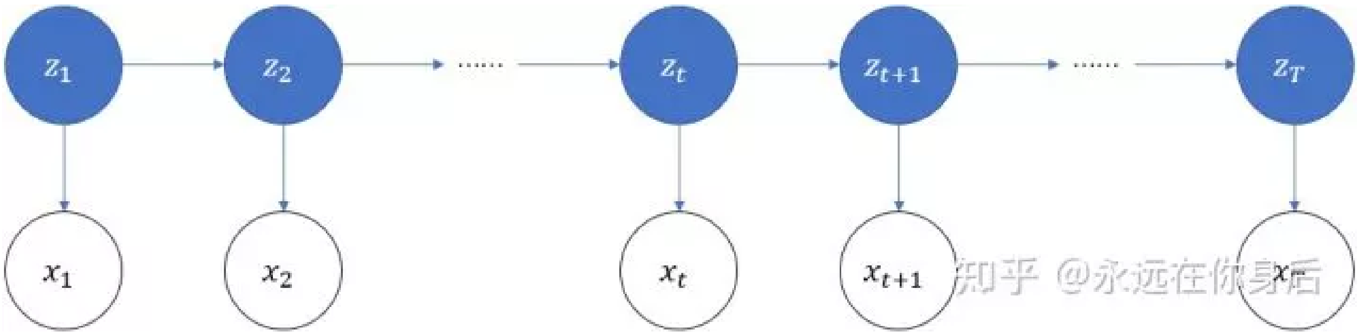
$$Z=(z_1,z_2,\dots,z_T)$$

将（1）式中的 $\sum_z$ 展开得到：

$$\sum_{Z} = \sum_{z_1}\sum_{z_2}\cdots\sum_{z_T}$$

即使不考虑内部的计算，这起码也是 $O(N^T)$ 阶的计算量，所以需要更有效的算法，下面介绍两种：**前向算法和后向算法。**

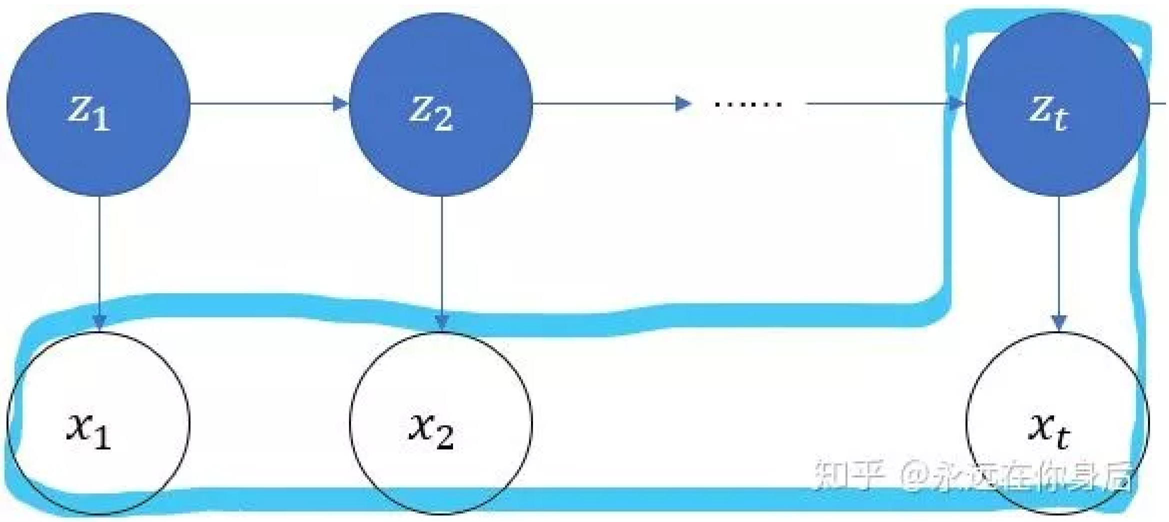
设有T 个序列，如下图所示:



现在定义一个前向概率 $\alpha_t(i)$ ，它t时刻的状态以及1,2,...t时刻的观测在给定参数下的联合概率：

$$\alpha_t\left(i\right) = P\left(x_i, x_2, \ldots, x_t, z_t = q_i | \lambda\right) \qquad (2)$$

也就是下图中标记的那一部分



根据定义，可以得到它的初值：

$$\begin{aligned}\alpha_1(i) &= P(x_1, z_1 = q_i|\lambda) \\ &= P(z_1 = q_i|\lambda) P(x_1|z_1 = q_i, \lambda) \\ &= \pi_i b_i(x_1)\end{aligned}$$

其中 $b_i(x)$ 表示由状态 q_i 生成给定观测数据的概率，例如设t时刻观测数据 $x_t = v_j$ ，有

$$b_i(x_t) = b_i(x_t = v_j) = P(x_t = v_j|z_t = q_i) = b_{ij}$$

接着，根据（2）式，还可以得到：

$$\begin{aligned}\alpha_T(i) &= P(x_i, x_2, \ldots, x_T, z_T = q_i|\lambda) \\ &= P(X, z_T = q_i|\lambda)\end{aligned}$$

由此公式，遍历 z_T 的取值求和，可以得到 X 的边际概率

$$\sum_i^N \alpha_T(i) = \sum_i^N P(X, z_T = q_i|\lambda) = P(X|\lambda)$$

假设已知 $\alpha_t(1), \alpha_t(2), \ldots, \alpha_t(N)$ ，推导 $\alpha_{t+1}(*)$ ，在（2）式的基础上

$$\alpha_{t+1}(j) = P(x_1, \ldots, x_{t+1}, z_{t+1} = q_j|\lambda) \qquad (3)$$

引入变量 $z_t = q_i$ ，有：

$$\begin{aligned}&\alpha_{t+1}(j) \\ &= \sum_{i=1}^N P(x_1, \ldots, x_{t+1}, z_t = q_i, z_{t+1} = q_j|\lambda) \\ &= \sum_{i=1}^N {\color{red}P(x_{t+1}|x_1, \ldots, x_t, z_t = q_i, z_{t+1} = q_j, \lambda)} {\color{blue}P(x_1, \ldots, x_t, z_t = q_i, z_{t+1} = q_j|\lambda)}\end{aligned}$$

首先，来看上式中红色部分，根据观测独立假设（下文再引用该假设时称作假设2）：

$$P(x_{t+1}|x_1, \ldots, x_t, z_t = q_i, z_{t+1} = q_j, \lambda) = P(x_{t+1}|z_{t+1} = q_j) = b_j(x_{t+1})$$

然后是蓝色部分，根据齐次马尔可夫假设（下文再引用该假设时称作假设1）

$$
\begin{aligned}
& P\left(x_1, \ldots, x_t, z_t = q_i, z_{t+1} = q_j \mid \lambda\right) \\
& = P\left(z_{t+1} = q_j \mid x_1, \ldots, x_t, z_t = q_i, \lambda\right) P\left(x_1, \ldots, x_t, z_t = q_i \mid \lambda\right) \\
& = P\left(z_{t+1} = q_j \mid z_t = q_i\right) P\left(x_1, \ldots, x_t, z_t = q_i \mid \lambda\right) \\
& = a_{ij} \alpha_t(i)
\end{aligned}
$$

将上述结果代入（3）式，得到

$$
\alpha_{t+1}(j) = \sum_{i=1}^{N} a_{ij} b_j\left(x_{t+1}\right) \alpha_t(i)
$$

以上就是前向算法的推导，下面根据一个栗子来写代码，假设前面抽了五个球，分别是：红、红、白、白、红，求概率

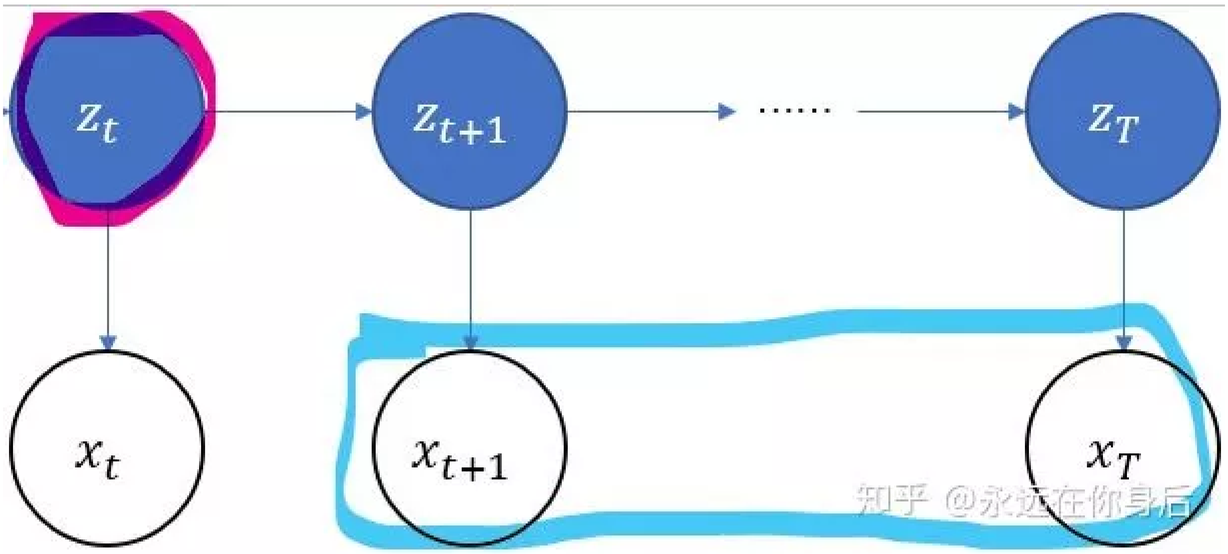
```
class HMM(object):
    def evaluate(self, X):
        '''
        根据给定的参数计算条件概率
        X: 观测数据
        '''
        alpha = self.pi * self.B[:,X[0]]
        for x in X[1:]:
            # alpha_next = np.empty(self.N)
            # for j in range(self.N):
            #     alpha_next[j] = np.sum(self.A[:,j] * alpha * self.B[j,x])
            # alpha = alpha_next
            alpha = np.sum(self.A * alpha.reshape(-1,1) * self.B[:,x].reshape(
        return alpha.sum()

print(hmm.evaluate([0,0,1,1,0]))    # 0.026862016
```

上面的注释中的代码是按照公式来写的，可以看出，时间复杂度降为了 $O(TN^2)$ ，比之前至少 $O(N^T)$ 的起步价已经好太多了.

接着，再讨论后向算法，首先定义后向概率：

$$\beta_t(i) = P(x_T, x_{T-1}, \ldots, x_{t+1} | z_t = q_i, \lambda) \qquad (4)$$

也就是下图中的部分



并且规定初始值

$$\beta_T(1) = \beta_T(2) = \ldots = \beta_T(N) = 1$$

根据（4）式，还可以得到

$$\beta_1(i) = P(x_T, x_{T-1}, \ldots, x_2 | z_1 = q_i, \lambda)$$

然后来看上式和要计算的概率 $P(X|\lambda)$ 之间的关系

$$
\begin{aligned}
P(X|\lambda) &= P(x_1, x_2, \ldots, x_T|\lambda) \\
&= \sum_{i=1}^{N} P(x_1, x_2, \ldots, x_T, z_1 = q_i|\lambda) \\
&= \sum_{i=1}^{N} P(x_1|x_2, \ldots, x_T, z_1 = q_i, \lambda) P(x_2, \ldots, x_T, z_1 = q_i|\lambda) \\
&= \sum_{i=1}^{N} P(x_1|z_1 = q_i) P(z_1 = q_i|\lambda) P(x_T, x_{T-1}, \ldots, x_2|z_1 = q_i, \lambda) \\
&= \sum_{i=1}^{N} b_i(x_1) \pi_i \beta_1(i)
\end{aligned}
$$

另外说一点，如果对于前向算法还有印象的话，你会发现：上面 $\pi_i b_i(x_1)$ 也就是 $\alpha_1(i)$ 的定义。实际上，对于任意时刻t，存在以下等式

$$
P(X|\lambda) = \sum_{i}^{N} \alpha_t(i) \beta_t(i)
$$

接着，假设已知所有的 β_{t+1} ，来推导 β_t

$$
\begin{aligned}
\beta_t(i) &= P(x_T, \ldots, x_{t+1}|z_t = q_i, \lambda) \\
&= \sum_{j=1}^{N} P(x_T, \ldots, x_{t+1}, z_{t+1} = q_j|z_t = q_i, \lambda) \\
&= \sum_{j=1}^{N} {\color{red} P(x_T, \ldots, x_{t+1}|z_{t+1} = q_j, z_t = q_i, \lambda)} {\color{blue} P(z_{t+1} = q_j|z_t = q_i, \lambda)}
\end{aligned}
$$

观察上式，蓝色部分自然就是 a_{ij} 。而红色部分，根据假设2， z_t 与 $x_t, x_{t+1}, \ldots, x_T$ 都是无关（即互相独立），可以省去，所以这部分最终变为：

$$
\begin{aligned}
& P(x_T, \ldots, x_{t+1}|z_{t+1} = q_j, z_t = q_i, \lambda) \\
&= P(x_T, \ldots, x_{t+1}|z_{t+1} = q_j, \lambda) \\
&= P(x_{t+1}|x_T, \ldots, x_{t+2}, z_{t+1} = q_j, \lambda) P(x_T, \ldots, x_{t+2}|z_{t+1} = q_j, \lambda) \\
&= P(x_{t+1}|z_{t+1} = q_j) P(x_T, \ldots, x_{t+2}|z_{t+1} = q_j, \lambda) \\
&= b_j(x_{t+1}) \beta_{t+1}(j)
\end{aligned}
$$

综合上述结果，最终代入到 $\beta_t(i)$ ，得到

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)$$

推导完毕，上代码

```
def evaluate_backward(self, X):
        beta = np.ones(self.N)
        for x in X[:0:-1]:
            beta_next = np.empty(self.N)
            for i in range(self.N):
                beta_next[i] = np.sum(self.A[i,:] * self.B[:,x] * beta)
            beta = beta_next
        return np.sum(beta * self.pi * self.B[:,X[0]])
```

和前向算法差不多，而且是照着公式写的，就不写注释了，还是使用前面的栗子，跑了一下发现结果是一样的。我想，同时写两个BUG得出同一个结果的概率应该很小很小吧

学习算法

现在，概率计算的问题就解决了，接着来看第二个问题，参数学习，这里需要用到EM算法，不熟悉的可以参考一下：https://zhuanlan.zhihu.com/p/85236423

设已知 $\lambda^{(t)}$ ，根据EM算法中的E步，先化简 Q 函数。把 Q 函数的通用公式搬过来，将 θ 改成 λ

$$\begin{aligned} Q\left(\lambda, \lambda^{(t)}\right) &= \mathbb{E}_{Z|X,\lambda^{(t)}}[\log P(X,Z|\lambda)] \\ &= \sum_{Z} \log P(X,Z|\lambda) P\left(Z|X,\lambda^{(t)}\right) \\ &= \sum_{Z} \log\left(P(Z|\lambda) P(X|Z,\lambda)\right) P\left(Z|X,\lambda^{(t)}\right) \end{aligned}$$

然后，对Q函数中的每一项进行化简，首先是第一项，用到了齐次马尔可夫假设：

$$
\begin{aligned}
P(Z|\lambda) &= P(z_1, \ldots, z_T|\lambda) \\
&= P(z_T|z_1, \ldots, z_{T-1}, \lambda) P(z_1, \ldots, z_{T-1}|\lambda) \\
&= P(z_T|z_{T-1}) P(z_1, \ldots, z_{T-1}|\lambda) \\
&= P(z_T|z_{T-1}) P(z_{T-1}|z_{T-2}) P(z_1, \ldots, z_{T-2}|\lambda) \\
&= \ldots \\
&= \pi(z_1) \prod_{i=2}^{T} P(z_i|z_{i-1})
\end{aligned}
$$

其中 $\pi(z_1)$ 表示对应 z_1 取值的概率，例如 $z_1 = v_i$ ，则有：

$$
\pi(z_1) = \pi(z_1 = v_i) = \pi_i
$$

接着是第二项，用到了观测独立假设

$$
\begin{aligned}
P(X|Z,\lambda) &= P(x_1, x_2, \ldots, x_T|z_1, z_2, \ldots, z_T, \lambda) \\
&= P(x_T|x_1, \ldots, x_{T-1}, z_1, \ldots, z_T,\ \lambda) P(x_1, \ldots, x_{T-1}|z_1, \ldots, z_T, \lambda) \\
&= P(x_T|z_T) P(x_1, \ldots, x_{T-1}|z_1, \ldots, z_T, \lambda) \\
&= P(x_T|z_T) P(x_{T-1}|z_{T-1}) P(x_1, \ldots, x_{T-2}|z_1, \ldots, z_T, \lambda) \\
&= \ldots \\
&= \prod_{i=1}^{T} P(x_i|z_i)
\end{aligned}
$$

又因我们要求使Q函数最大化的参数，即：

$$
\begin{aligned}
\lambda^{(t+1)} &= \arg\max_{\lambda} Q\left(\lambda, \lambda^{(t)}\right) \\
&= \arg\max_{\lambda} \ldots P\left(Z|X, \lambda^{(t)}\right)
\end{aligned}
$$

展开 $P\left(Z|X, \lambda^{(t)}\right)$ ，有

$$
P\left(Z|X, \lambda^{(t)}\right) = \frac{P\left(Z, X|\lambda^{(t)}\right)}{P\left(X|\lambda^{(t)}\right)}
$$

可以看到 $P\left(X|\lambda^{(t)}\right)$ 在当前参数下是常量，故省去。将这全部代入到 Q 函数，得到：

$\lambda^{(t+1)}$

$= \arg\max Q\left(\lambda, \lambda^{(t)}\right)$

$$
\begin{aligned}
&= \arg\max_{\lambda} Q\left(\lambda, \lambda^{(t)}\right) \\
&= \arg\max_{\lambda} \sum_{Z} \log\left(P(Z|\lambda) P(X|Z,\lambda)\right) P\left(Z|X, \lambda^{(t)}\right) \\
&= \arg\max_{\lambda} \sum_{Z} \left[\log\left(\pi(z_1)\left(\prod_{i=2}^{T} P(z_i|z_{i-1})\right)\left(\prod_{i=1}^{T} P(x_i|z_i)\right)\right) P\left(Z, X|\lambda^{(t)}\right)\right] \\
&= \arg\max_{\lambda} \sum_{Z} \left[\left(\log \pi(z_1) + \log\left(\prod_{i=2}^{T} P(z_i|z_{i-1})\right) + \log\left(\prod_{i=1}^{T} P(x_i|z_i)\right)\right) P\left(Z, X|\lambda\right.\right. \\
&= \arg\max_{\lambda} \sum_{Z}\left(\log \pi(z_1) P\left(Z, X|\lambda^{(t)}\right)\right) + \\
&\quad \sum_{Z}\left(\log\left(\prod_{i=2}^{T} P(z_i|\, z_{i-1})\right) P\left(Z, X|\lambda^{(t)}\right)\right) + \\
&\quad \sum_{Z}\left(\log\left(\prod_{i=1}^{T} P(x_i|\, z_i)\right) P\left(Z, X|\lambda^{(t)}\right)\right)
\end{aligned}
$$

因为 $\lambda^{(t+1)} = \left(\pi^{(t+1)}, A^{(t+1)}, B^{(t+1)}\right)$ ，观测上式会发现，其中红绿蓝三个部分正好分别
这三个参数。所以当求解其中一个参数时，另外两项就可以直接去掉了，首先是 π

$$
\pi^{(t+1)} = \arg\max_{\pi} \sum_{Z} \log \pi(z_1) P\left(Z, X|\lambda^{(t)}\right)
$$

这样还不能直接求导，需要进行化简：

$$
\begin{aligned}
\sum_{Z} \log \pi(z_1) P\left(Z, X|\lambda^{(t)}\right) &= \sum_{z_1}\sum_{z_2}\cdots\sum_{z_T} \log \pi(z_1) P\left(Z, X|\lambda^{(t)}\right) \\
&= \sum_{z_1} \log \pi(z_1) \sum_{z_2}\cdots\sum_{z_T} P\left(z_T, \ldots, z_2, z_1, X|\lambda^{(t)}\right) \\
&= \sum_{z_1} \log \pi(z_1) \sum_{z_2}\cdots\sum_{z_{T-1}} P\left(z_{T-1}, \ldots, z_2, z_1, X|\lambda^{(t)}\right) \\
&= \sum_{z_1} \log \pi(z_1) P\left(z_1, X|\lambda^{(t)}\right)
\end{aligned}
$$

注意上面公式中红色部分，这是一个对联合概率求和的计算，得到以下结果：

$$
\sum_{z_T} P\left(z_T, \ldots, z_2, z_1, X|\lambda^{(t)}\right) = P\left(z_{T-1}, \ldots, z_2, z_1, X|\lambda^{(t)}\right)
$$

然后在后面 $\sum_{z_{T-1}}$ 的求和中重复此操作，直到剩下 $\sum_{z_1}$ ，也就是上面的最终结果。这种化简
面计算 $A^{(t+1)}, B^{(t+1)}$ 中还会用到，这里特别提一下，后文就直接写结果了

除此之外，因为 π 是一个概率分布，有一个约束条件： $\sum_{i=1}^{N} \pi_i = 1$ ，所以需要构造一个拉日函数，再对这个函数进行求导

$$L(\pi, \eta) = \sum_{i=1}^{N} \log \pi_i P\left(z_1 = q_i, X|\lambda^{(t)}\right) + \eta\left(\sum_{i=1}^{N} \pi_i - 1\right)$$

对 π_i 求导

$$\frac{\partial L(\pi, \eta)}{\pi_i} = \frac{1}{\pi_i} P\left(z_1 = q_i, X|\lambda^{(t)}\right) + \eta$$

上式中， $P\left(z_1 = q_i, X|\lambda^{(t)}\right)$ 与我们要优化的参数没有关系，所以 L 函数求导还是很简单的，包括后面对 A, B 两个矩阵的求导，也是一样的。接着令导数为0，并两边同时乘上 π_i 到

$$P\left(z_1 = q_i, X|\lambda^{(t)}\right) + \eta\pi_i = 0 \qquad (5)$$

因为 π_i 是 π 的分量的通向，所以对所有的分量求导都是一样的结果。将这些结果相加

$$\sum_{i}^{N} P\left(z_1 = q_i, X|\lambda^{(t)}\right) + \eta\pi_i = 0$$

再利用 π 的约束，就得到了：

$$\eta \sum_{i}^{N} \pi_i = -\sum_{i}^{N} P\left(z_1 = q_i, X|\lambda^{(t)}\right)$$
$$\eta = -P\left(X|\lambda^{(t)}\right)$$

将结果代入（5）式，得到

$$\begin{aligned}\pi_i^{(t+1)} &= -\frac{P\left(z_1=q_i, X|\lambda^{(t)}\right)}{\eta} \\ &= \frac{P\left(z_1=q_i, X|\lambda^{(t)}\right)}{P\left(X|\lambda^{(t)}\right)}\end{aligned}$$

其中，$P\left(X|\lambda^{(t)}\right)$ 就是当前参数下观测数据的概率，就是第一个问题所求解的。另外，利用第一个问题中定义的前向概率和后向概率，有：

$$\begin{aligned}&\alpha_t(i)\beta_t(i) \\ &= P(x_i, x_2, \ldots, x_t, z_t=q_i|\lambda) P(x_T, x_{T-1}, \ldots, x_{t+1}|z_t=q_i, \lambda) \\ &= P(x_i, x_2, \ldots, x_t|z_t=q_i, \lambda) P(x_T, x_{T-1}, \ldots, x_{t+1}|z_t=q_i, \lambda) P(z_t=q_i|\lambda) \\ &= P(x_i, x_2, \ldots, x_T|z_t=q_i, \lambda) P(z_t=q_i|\lambda) \\ &= P(X, z_t=q_i|\lambda)\end{aligned}$$

最终得到：

$$\pi_i^{(t1)} = \frac{\alpha_1(i)\beta_1(i)}{P\left(X|\lambda^{(t)}\right)}$$

接着来看矩阵A的迭代公式

$$\begin{aligned}A^{(t+1)} &= \arg\max_A \sum_Z \left[\log\left(\prod_{i=1}^{T-1} P(z_{i+1}|z_i)\right) P\left(Z, X|\lambda^{(t)}\right)\right] \\ &= \arg\max_A \sum_Z \left[\left(\sum_{i=1}^{T-1} \log P(z_{i+1}|z_i)\right) P\left(Z, X|\lambda^{(t)}\right)\right]\end{aligned}$$

同样，将上式化简，另外为了在后面方便引用，将该式设为一个函数f

$$\begin{aligned}f &= \sum_Z \left[\left(\sum_{i=1}^{T-1} \log P(z_{i+1}|z_i)\right) P\left(Z, X|\lambda^{(t)}\right)\right] \\ &= \sum_Z \left[(\log P(z_2|z_1) + \ldots + \log P(z_T|z_{T-1})) P\left(Z, X|\lambda^{(t)}\right)\right] \\ &= \sum_Z \log P(z_2|z_1) P\left(Z, X|\lambda^{(t)}\right) + \ldots + \sum_Z \log P(z_T|z_{T-1}) P\left(Z, X|\lambda^{(t)}\right)\end{aligned}$$

可以看到，一共是 $T-1$ 个相似的项，我们提一个（红色部分）出来化简，看看能不能找到通项公式

$$
\begin{aligned}
&\sum_{Z} \log P\left(z_{2} \mid z_{1}\right) P\left(Z, X \mid \lambda^{(t)}\right) \\
&=\sum_{z_{1}} \sum_{z_{2}} \ldots \sum_{z_{T}} \log P\left(z_{2} \mid z_{1}\right) P\left(z_{T}, \ldots, z_{1}, X \mid \lambda^{(t)}\right) \\
&=\sum_{z_{1}} \sum_{z_{2}} \log P\left(z_{2} \mid z_{1}\right) \ldots \sum_{z_{T}} P\left(z_{T}, \ldots, z_{1}, X \mid \lambda^{(t)}\right) \\
&=\sum_{z_{1}} \sum_{z_{2}} \log P\left(z_{2} \mid z_{1}\right) P\left(z_{1}, z_{2}, X \mid \lambda^{(t)}\right) \\
&=\sum_{i=1}^{N} \sum_{j=1}^{N} \log P\left(z_{2}=q_{j} \mid z_{1}=q_{i}\right) P\left(z_{1}=q_{i}, z_{2}=q_{j}, X \mid \lambda^{(t)}\right) \\
&=\sum_{i=1}^{N} \sum_{j=1}^{N} \log a_{i j} P\left(z_{1}=q_{i}, z_{2}=q_{j}, X \mid \lambda^{(t)}\right)
\end{aligned}
$$

这样，就化简出了通向公式，将它代入f中，得到

$$
\begin{aligned}
f=&\sum_{z_{2}} \log P\left(z_{2} \mid z_{1}\right) P\left(z_{2}, X \mid \lambda^{(t)}\right)+\sum_{z_{T}} \log P\left(z_{T} \mid z_{T-1}\right) P\left(z_{T}, X \mid \lambda^{(t)}\right) \\
&=\sum_{i=1}^{N} \sum_{j=1}^{N} \log a_{i j} P\left(z_{1}=q_{i}, z_{2}=q_{j}, X \mid \lambda^{(t)}\right)+\ldots \\
&+\sum_{i=1}^{N} \sum_{j=1}^{N} \log a_{i j} P\left(z_{T-1}=q_{i}, z_{T}=q_{j}, X \mid \lambda^{(t)}\right) \\
&=\sum_{t=1}^{T-1} \sum_{i=1}^{N} \sum_{j=1}^{N} \log a_{i j} P\left(z_{t}=q_{i}, z_{t+1}=q_{j}, X \mid \lambda^{(t)}\right)
\end{aligned}
$$

因为 A 是一个概率分布的矩阵，例如前面的栗子，每一行的和等于1

	q_1	q_2	q_3	q_4
q_1	0	1	0	0
q_2	0.4	0	0.6	0
q_3	0	0.4	0	0.6
q_4	0	0	0.5	0.5

所以A是有约束的：

$$\sum_{j=1}^{N} a_{*j} = 1$$

同样，使用拉格朗日乘数法，构造目标函数

$$L(A,\eta) = \sum_{t=1}^{T-1}\sum_{i=1}^{N}\sum_{j=1}^{N} \log a_{ij} P\left(z_t = q_i, z_{t1} = q_j, X|\lambda^{(t)}\right) \sum_{k=1}^{N} \eta_k \left(\sum_{j=1}^{N} a_{kj} - 1\right)$$

将该函数对矩阵A的每一个元素求（偏）导并令导数为0：

$$\frac{\partial L(A,\eta)}{\partial a_{ij}} = 0 \Rightarrow \frac{1}{a_{ij}} \sum_{t=1}^{T-1} P\left(z_t = q_i, z_{t+1} = q_j, X|\lambda^{(t)}\right) + \sum_{k=1}^{N} \eta_k = 0 \qquad (6)$$

将两边同时乘上 a_{ij} ，得到 $a_{ij}^{(t+1)}$

$$\begin{aligned} a_{ij} \sum_{k=1}^{N} \eta_k &= -\sum_{t=1}^{T-1} P\left(z_t = q_i, z_{t+1} = q_j, X|\lambda^{(t)}\right) \\ a_{ij}^{(t+1)} &= -\frac{\sum_{t=1}^{T-1} P\left(z_t = q_i, z_{t+1} = q_j, X|\lambda^{(t)}\right)}{\sum_{k=1}^{N} \eta_k} \end{aligned} \qquad (7)$$

注意一下上面的下标t与上标中（t+1）它们是不同的，由于变量比较多，各种ijk比较多，所以这里需要注意一下。然后利用 a_{ij} 的约束，代入（6）式，得到：

$$\sum_{j=1}^{N}\sum_{t=1}^{T-1} P\left(z_t=q_i, z_{t+1}=q_j, X|\lambda^{(t)}\right)+\sum_{j=1}^{N} a_{ij}\sum_{k=1}^{N}\eta_k=0$$

然后化简：

$$\begin{aligned}\sum_{j=1}^{N} a_{ij}\sum_{k=1}^{N}\eta_k&=\sum_{k=1}^{N}\eta_k\\&=\sum_{j=1}^{N}\sum_{t=1}^{T-1} P\left(z_t=q_i, z_{t+1}=q_j, X|\lambda^{(t)}\right)\\&=\sum_{t=1}^{T-1} P\left(z_t=q_i, X|\lambda^{(t)}\right)\end{aligned}$$

代入（7）式，得到

$$a_{ij}^{(t+1)}=\frac{\sum_{t=1}^{T-1} P\left(z_t=q_i, z_{t+1}=q_j, X|\lambda^{(t)}\right)}{\sum_{t=1}^{T-1} P\left(z_t=q_i, X|\lambda^{(t)}\right)} \quad (8)$$

（8）式中，分母部分前面已经解决了，下面来看分子部分，进行化简

$$\begin{aligned}&P\left(z_t=q_i, z_{t+1}=q_j, X|\lambda^{(t)}\right)\\&=P\left(z_t=q_i, X_{1:t}|\lambda^{(t)}\right) P\left(z_{t+1}=q_j, X_{t+1:T}|z_t=q_i, X_{1:t}, \lambda^{(t)}\right)\\&=\alpha_t(i) P\left(z_{t+1}=q_j, X_{t+1:T}|z_t=q_i, \lambda^{(t)}\right)\\&=\alpha_t(i) P\left(X_{t+2:T}|x_{t+1}, z_{t+1}=q_j, z_t=q_i, \lambda^{(t)}\right) P\left(x_{t+1}, z_{t+1}=q_j|z_t=q_i, \lambda^{(t)}\right)\\&=\alpha_t(i) P\left(X_{t+2:T}|z_{t+1}=q_j, \lambda^{(t)}\right) P\left(x_{t+1}, z_{t+1}=q_j|z_t=q_i, \lambda^{(t)}\right)\\&=\alpha_t(i)\beta_{t+1}(j) P\left(z_{t+1}=q_j|z_t=q_i, \lambda^{(t)}\right) P\left(x_{t+1}|z_{t+1}=q_j, z_t=q_i, \lambda^{(t)}\right)\\&=\alpha_t(i)\beta_{t+1}(j) P\left(z_{t+1}=q_j|z_t=q_i\right) P\left(x_{t+1}|z_{t+1}=q_j\right)\\&=\alpha_t(i)\beta_{t+1}(j) a_{ij} b_j\left(x_{t+1}\right)\end{aligned}$$

注意，上面的化简中， $X_{1:t}=\left(x_1, x_2, \ldots, x_t\right)$ 。然后红色和蓝色部分的化简用到了前面前面提过的两个假设，将条件中不被依赖的变量去掉了。最后代入（8）式得到：

$$
\begin{aligned}
a_{ij}^{(t+1)} &= \frac{\sum_{t=1}^{T-1} \alpha_t(i)\beta_{t+1}(j)a_{ij}b_j(x_{t+1})}{\sum_{t=1}^{T-1} P\left(z_t = q_i, X|\lambda^{(t)}\right)} \\
&= \frac{\sum_{t=1}^{T-1} \alpha_t(i)\beta_{t+1}(j)a_{ij}b_j(x_{t+1})}{\sum_{t=1}^{T-1} \alpha_t(i)\beta_t(i)}
\end{aligned}
$$

最后，就剩观测概率矩阵（B）的迭代公式

$$
\begin{aligned}
B^{(t+1)} &= \arg\max_B \sum_Z \left[\log\left(\prod_{i=1}^T P(x_i|z_i)\right) P\left(Z, X|\lambda^{(t)}\right)\right] \\
&= \arg\max_B \sum_Z \left[\left(\sum_{i=1}^T \log P(x_i|z_i)\right) P\left(Z, X|\lambda^{(t)}\right)\right]
\end{aligned}
$$

同样，拆开化简

$$
\begin{aligned}
f &= \sum_Z \left[\left(\sum_{i=1}^T \log P(x_i|z_i)\right) P\left(Z, X|\lambda^{(t)}\right)\right] \\
&= \sum_Z \left[(\log P(x_1|z_1) + \ldots + \log P(x_T|z_T))\, P\left(Z, X|\lambda^{(t)}\right)\right] \\
&= \sum_Z \log P(x_1|z_1) P\left(Z, X|\lambda^{(t)}\right) + \ldots + \sum_Z \log P(x_T|z_T) P\left(Z, X|\lambda^{(t)}\right)
\end{aligned}
$$

分析第一项：

$$
\begin{aligned}
&\sum_Z \log P(x_1|z_1) P\left(Z, X|\lambda^{(t)}\right) \\
&= \sum_{z_1}\sum_{z_2}\cdots\sum_{z_T} \log P(x_1|z_1) P\left(Z, X|\lambda^{(t)}\right) \\
&= \sum_{z_1} \log P(x_1|z_1) \sum_{z_2}\cdots\sum_{z_T} P\left(z_1, \ldots, z_T, X|\lambda^{(t)}\right) \\
&= \sum_{z_1} \log P(x_1|z_1) P\left(z_1, X|\lambda^{(t)}\right)
\end{aligned}
$$

代入f，得到

$$
\begin{aligned}
f &= \sum_{z_1} \log P(x_1|z_1) P\left(z_1, X|\lambda^{(t)}\right) + \ldots + \sum_{z_T} \log P(x_T|z_T) P\left(z_T, X|\lambda^{(t)}\right) \\
&= \sum_{t=1}^{T} \sum_{z_t} \log P(x_t|z_t) P\left(z_t, X|\lambda^{(t)}\right) \\
&= \sum_{t=1}^{T} \sum_{i=1}^{N} \log P(x_t|z_t = q_i) P\left(z_t = q_i, X|\lambda^{(t)}\right) \\
&= \sum_{t=1}^{T} \sum_{i=1}^{N} \log b_i(x_t) P\left(z_t = q_i, X|\lambda^{(t)}\right)
\end{aligned}
$$

以前面的栗子为例，矩阵B同样有约束

	v_1	v_2
q_1	0.5	0.5
q_2	0.3	0.7
q_3	0.6	0.4
q_4	0.8	0.2

也是要求每一行的和等于1

$$
\sum_{x} b_j(x) = \sum_{k=1}^{M} b_{jk} = 1
$$

M是矩阵B的列数，前面已经定义过的，构造拉格朗日函数：

$$
L(B, \eta) = \sum_{t=1}^{T} \sum_{i=1}^{N} \log b_i(x_t) P\left(z_t = q_i, X|\lambda^{(t)}\right) + \sum_{i=1}^{N} \eta_i \left(\sum_{k=1}^{M} b_{jk} - 1\right)
$$

将该函数对矩阵B的每一项元素求导，得到：

$$
\frac{\partial L(B, \eta)}{\partial b_{jk}} = \sum_{t=1}^{T} \frac{1}{b_{jk}} P\left(z_t = q_j, X|\lambda^{(t)}\right) I(x_t = v_k) + \sum_{i=1}^{N} \eta_i
$$

这里需要注意， $L(B,\eta)$ 中只有 $\log b_i(x_t)$ 满足 $x_t=v_k$ 是导数才不为0，这样是上式中红色部分的足有，该函数的作用是满足条件值为1，否则为0。接着，令导数为0

$$
\begin{aligned}
b_{jk}\sum_{i=1}^{N}\eta_i &= -\sum_{t=1}^{T}P\left(z_t=q_j,X|\lambda^{(t)}\right)I\left(x_t=v_k\right)\\
b_{jk}^{(t+1)} &= -\frac{\sum_{t=1}^{T}P\left(z_t=q_j,X|\lambda^{(t)}\right)I\left(x_t=v_k\right)}{\sum_{i=1}^{N}\eta_i} \qquad (9)
\end{aligned}
$$

同样利用B的约束条件，得到

$$
\sum_{k=1}^{M}\sum_{t=1}^{T}P\left(z_t=q_j,X|\lambda^{(t)}\right)I\left(x_t=v_k\right)+\sum_{k}^{M}b_{jk}\sum_{i=1}^{N}\eta_i
$$

化 简 得 到 (9) 式 的 分 母

$$
\begin{aligned}
\sum_{k}^{M} b_{jk} \sum_{i=1}^{N} \eta_i = \sum_{i=1}^{N} \eta_i \\
&= -\sum_{k=1}^{M} \sum_{t=1}^{T} P\left(z_t = q_j, X | \lambda^{(t)}\right) I\left(x_t = v_k\right) \\
&= -\sum_{t=1}^{T} P\left(z_t = q_j, X | \lambda^{(t)}\right) \sum_{k=1}^{M} I\left(x_t = v_k\right) \\
&= -\sum_{t=1}^{T} P\left(z_t = q_j, X | \lambda^{(t)}\right)
\end{aligned}
$$

上面的化简中，$\sum_{k=1}^{M} I\left(x_t = v_k\right) = 1$ 。代入（9）式，最终得到

$$
\begin{aligned}
b_{jk}^{(t+1)} &= \frac{\sum_{t=1}^{T} P\left(z_t = q_j, X | \lambda^{(t)}\right) I\left(x_t = v_k\right)}{\sum_{t=1}^{T} P\left(z_t = q_j, X | \lambda^{(t)}\right)} \\
&= \frac{\sum_{t=1}^{T} \alpha_t(j) \beta_t(j) I\left(x_t = v_k\right)}{\sum_{t=1}^{T} \alpha_t(j) \beta_t(j)}
\end{aligned}
$$

最后，整理一下公式，

$$
\begin{aligned}
\pi_i^{(t+1)} &= \frac{P\left(z_1 = q_i, X | \lambda^{(t)}\right)}{P\left(X | \lambda^{(t)}\right)} = \frac{\alpha_1(i) \beta_1(i)}{P\left(X | \lambda^{(t)}\right)} \\
a_{ij}^{(t+1)} &= \frac{\sum_{t=1}^{T-1} \alpha_t(i) \beta_{t+1}(j) a_{ij} b_j\left(x_{t+1}\right)}{\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i)} \\
b_{jk}^{(t+1)} &= \frac{\sum_{t=1}^{T} \alpha_t(j) \beta_t(j) I\left(x_t = v_k\right)}{\sum_{t=1}^{T} \alpha_t(j) \beta_t(j)}
\end{aligned}
$$

根据上面的公式，直接敲代码了

```
class HMM(object):
    def fit(self, X):
        '''
        根据给定观测序列反推参数
        '''
```

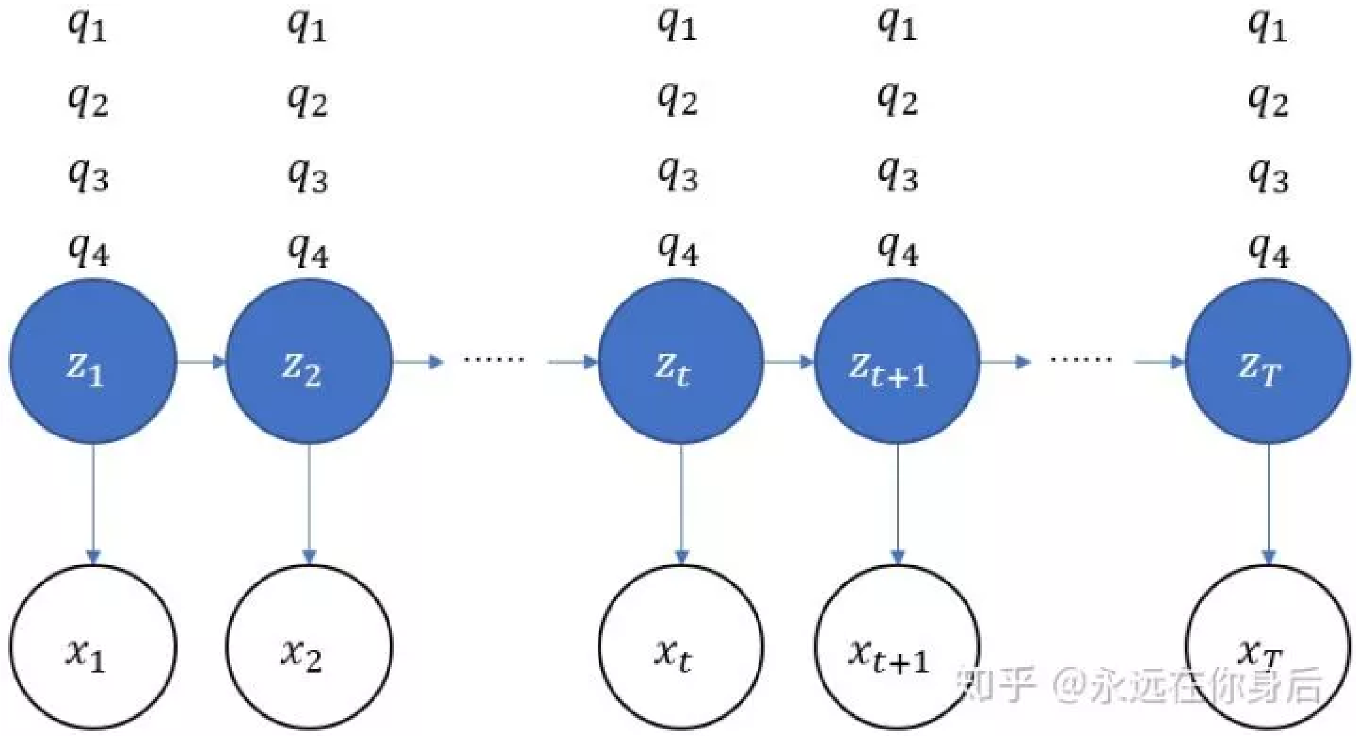
```
        # 初始化参数 pi, A, B
        self.pi = np.random.sample(self.N)
        self.A = np.ones((self.N,self.N)) / self.N
        self.B = np.ones((self.N,self.M)) / self.M
        self.pi = self.pi / self.pi.sum()
        T = len(X)
        for _ in range(50):
            # 按公式计算下一时刻的参数
            alpha, beta = self.get_something(X)
            gamma = alpha * beta

            for i in range(self.N):
                for j in range(self.N):
                    self.A[i,j] = np.sum(alpha[:-1,i]*beta[1:,j]*self.A[i,j]*s

            for j in range(self.N):
                for k in range(self.M):
                    self.B[j,k] = np.sum(gamma[:,j]*(X == k)) / gamma[:,j].sum

            self.pi = gamma[0] / gamma[-1].sum()


    def get_something(self, X):
        '''
        根据给定数据与参数，计算所有时刻的前向概率和后向概率
        '''
        T = len(X)
        alpha = np.zeros((T,self.N))
        alpha[0,:] = self.pi * self.B[:,X[0]]
        for i in range(T-1):
            x = X[i+1]
            alpha[i+1,:] = np.sum(self.A * alpha[i].reshape(-1,1) * self.B[:,x

        beta = np.ones((T,self.N))
        for j in range(T-1,0,-1):
            for i in range(self.N):
                beta[j-1,i] = np.sum(self.A[i,:] * self.B[:,X[j]] * beta[j])

        return alpha, beta

```

```
if __name__ == "__main__":
    import matplotlib.pyplot as plt
    def triangle_data(T):   # 生成三角波形状的序列
        data = []
        for x in range(T):
            x = x % 6
            data.append(x if x <= 3 else 6-x)
        return data

    data = np.array(triangle_data(30))
    hmm = HMM(10, 4)
    hmm.fit(data)               # 先根据给定数据反推参数
    gen_obs = hmm.generate(30)  # 再根据学习的参数生成数据
    x = np.arange(30)
    plt.scatter(x, gen_obs, marker='*', color='r')
    plt.plot(x, data, color='g')
    plt.show()
```

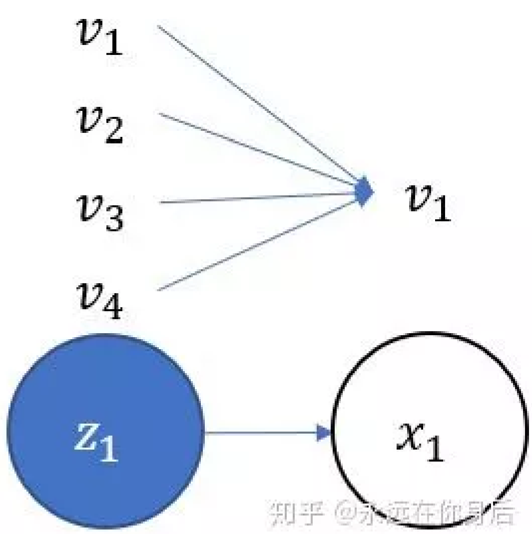
上面的代码，使用最开始的栗子无法收敛，或者收敛到坑里（公式和书上《统计学习方法》是一样的），但是使用别人的例子又能很好的工作。调了一晚上后我觉得还是把它贴上来算了，希望大神发现了问题所在能告知一下。

预测算法

最后一个问题了，解决这个问题的算法叫做维特比（Viterbi）算法。实际上它是一个动态规划求解最优路径的算法，这里的最优路径不过就是对应成最大概率而已，比前面两个问题容易解决得多。直接上例子，如下图所示：



$x_1, ..., x_T$ 和 λ 是已知的， $z_1, ..., z_T$ 是不能确定的，而且它们的取值都可能是 q_1 到 q_4 中的任意一个。先来看初始条件和终止条件，假设 $x_1 = v_1$ ，如下图



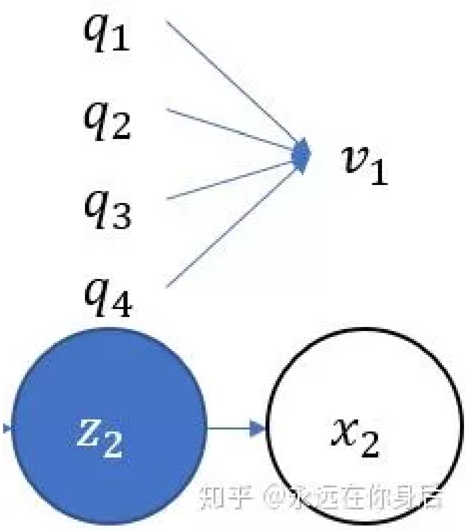
定义一个变量 $\delta_1(i)$ ，表示从 z_1 的取值从 q_1 到 q_4 ，然后再生成 $x_1 = v_1$ 的概率，因为 z_1 之前没有 z_0 ，所有 z_1 的取值由 π 决定

$$\delta_1(i) = \pi_i b_i(x_1)$$

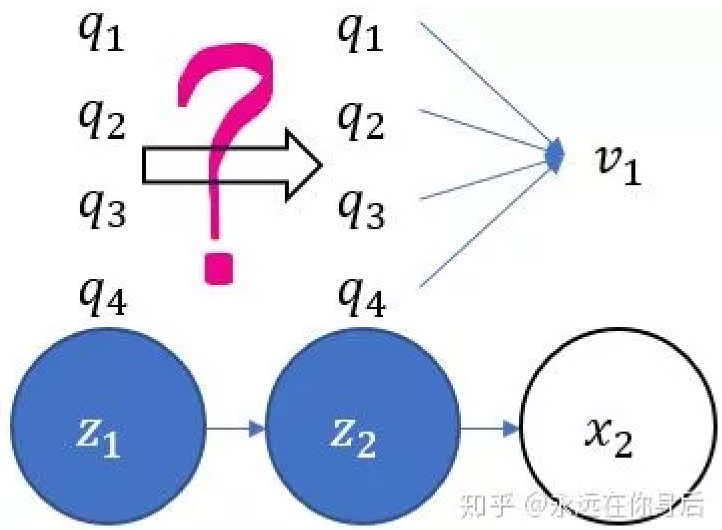
如果 $T = 1$ ，那么最优路径（索引）自然就是

$$i_T^* = \arg\max_i \delta_T(i)$$

接着，假设还有 $x_2=v_1$ ，末端的计算自然还是一样



问题在于从 z_1 到 z_2 如何计算最大概率



其实这个可以推广到 z_t 到 z_{t+1} ，其中 z_t 已经是最优路径。所以按照上图可以很自然的得到最大概率

$$\delta_T(j)=\left[\max_i \delta_{T-1}(i)\, a_{ij}\right] b_j\left(x_{T-1}\right)$$

然后，又因为我们所求的是路径，所以还要记录最大概率所对应的索引值

$$\varphi_T(j)=\arg\max_i \delta_{T-1}(i)\, a_{ij}$$

举个具体的例子：

$$\varphi_2(1) = \arg\max_i \delta_1(i)\, a_{i1}$$

上式的含义表示，在 z_1 部分已经是最优的情况下，考察 z_1 所有的取值转移到 $z_2 = q_1$ 时的最优路径。然后是该变量的初始值 $\varphi_1(i) = 0$

最后，就是回溯最优路径，因为已知

$$i_T^* = \arg\max_i \delta_T(i)$$

然后从 φ_T 从查找获得 i_{T-1}^*

$$i_{T-1}^* = \varphi_T(i_T^*)$$

不多说了，上代码：

```
class HMM(object):
    def decode(self, X):
        T = len(X)
        x = X[0]
        delta = self.pi * self.B[:,x]
        varphi = np.zeros((T, self.N), dtype=int)
        path = [0] * T
        for i in range(1, T):
            delta = delta.reshape(-1,1)     # 转成一列方便广播
            tmp = delta * self.A
            varphi[i,:] = np.argmax(tmp, axis=0)
            delta = np.max(tmp, axis=0) * self.B[:,X[i]]
        path[-1] = np.argmax(delta)
        # 回溯最优路径
        for i in range(T-1,0,-1):
            path[i-1] = varphi[i,path[i]]
        return path
```

*(*本文为 AI科技大本营转载文章，转载请联系作者)*