**7.隐马尔科夫模型**

**7.1定义：**

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可观测的观测随机序列的过程。

一个HMM模型由五部分组成μ = (Q, V, A, B, π)：

Q：可能状态集合, Q = {q1, q2, ……, qN}, N是可能的状态数

V：可能观测集合, V = {v1, v2, ……, vM}, M是可能的观测数

A：状态转移概率矩阵, A = [aij]N\*N, aij = P(it+1=qj|it=qi)

B：符号发射概率矩阵, B = [bj(k)]N\*M, bj(k) = P(ot=vk|it=qj)

π：初始状态概率分布, π = (πi), πi = P(i1=qi)

假设I表示长度为T的状态序列I = (i1, i2, ……, iT)，O是长度为T的观测序列O = (o1, o2, ……, oT)，**观测序列生成过程**描述如下：

输入：隐马尔科夫模型μ = (A, B, π)和观测序列长度T；

输出：观测序列O = (o1, o2, ……, oT)。

（1）按照初始状态分布π产生状态i1；

（2）令t = 1；

（3）按照状态it的观测概率分布生成ot；

（4）按照状态it的状态转移概率分布{}产生状态it+1，it+1 = 1，2，……，N；

（5）令t=t+1，如果t < T，转到（3），否则终止。

**从定义可知HMM模型作了两个基本假设：**

1）齐次马尔可夫性假设：假设隐藏的马尔可夫链在任意时刻t的状态只依赖于其前一时刻的状态，与其他时刻的状态及观测无关，也与时刻t无关。

P (it | it-1, ot-1, ……, i1, o1) = P (it | it-1), t = 1, 2, 3, ……, T

2）观测独立性假设：假设任意时刻的观测只依赖于该时刻的马尔可夫链的状态，与其他观测与状态无关。

P (ot | iT, oT, iT-1, oT-1, ......, i1, o1) = P (oi | it)

**7.2 HMM模型的三个基本问题：**

（1）评估问题：给定模型μ=(A, B, π)和观测序列O=(o1, o2, ……, oT)，计算在模型μ下观测序列O出现的概率P(O | μ)。

解决算法：前向算法、后向算法。

（2）学习问题：已知观测序列O=(o1, o2, ……, oT)，学习模型μ=(A, B, π)的参数，使得该模型下观测序列概率P(O | μ)最大。

解决算法：极大似然估计、Baum-Welch算法。

（3）解码问题：已知模型μ=(A, B, π)和观测序列O=(o1, o2, ……, oT)，求对给定观测序列条件概率P(I | O)最大的状态序列I=(i1, i2, ……, iT)，即给定观测序列，求最有可能的对应的状态序列。

解决算法：近似算法、维特比算法。

评估问题是为了解决给定模型μ和观测序列O时条件概率P(O | μ)的估计，是下面两个问题的基础；学习问题主要是为了学习模型的A, B, π三个参数，这就是HMM模型的模型训练，是一个典型的EM算法问题；解码问题是为了解决给定模型μ和观测序列O时最优状态序列的求解，是使用HMM模型的预测。

**7.2.1评估问题**

评估问题理论上可以使用直接计算的方法估计P(O | μ)：通过列举所有可能长度为T的状态序列I = (i1, i2, ……, it)，求各个状态序列I与观测序列O = (o1, o2, ……, ot)的联合概率P(O, I | μ)，然后对所有可能的状态序列求和得到P(O | μ)。

状态序列I = (i1, i2, ……, it)的概率：

对固定状态序列I = (i1, i2, …, it)，观测序列O = (o1, o2, …, ot)的概率P(O |I, μ)：

O和I同时出现的联合概率：

观测序列O的概率P(O | μ)是所有可能状态序列I和O的联合概率之和：

直接计算的方法，因为每一个状态序列长度为T，可能的状态序列I有NT种，所以直接计算法的时间复杂度是O(TNT)阶的，计算量非常大，尤其长度T很长的时候计算量是爆炸的。所以我们采用动态规划的前向算法和后向算法：

**1）前向算法**

前向概率：给定隐马尔科夫模型μ，定义到时刻t部分观测序列为o1, o2, ……,ot且状态为qi的概率为前向概率，记作

αt(i) = P (o1, o2, ……, ot, it=qi | μ)

算法流程：

输入：隐马尔科夫模型μ，观测序列O；

输出：观测序列概率P(O | μ)。

(1)初值：α1(i) = πibi(o1)，i=1, 2, 3, ……, N

(2)递推。对t = 1，2，3，……，T-1，

(3)终止：P(O | μ) = ∑ αT(i)

# 前向概率

def forward\_prob(t, i, o1\_t, s\_matrix, e\_matrix, pai):  
 *'''* ***:param*** *t: 时刻t* ***:param*** *i: 状态i* ***:param*** *o1\_t: 1到t时刻的观测序列* ***:param*** *s\_matrix: 状态转移矩阵* ***:param*** *e\_matrix: 概率发射矩阵* ***:param*** *pai: 初始状态向量* ***:return****: 时刻t状态i的前向概率  
 '''* if not isinstance(t, int) or t <= 0:  
 raise ValueError  
 if not isinstance(i, int) or i <= 0:  
 raise ValueError  
 if t != len(o1\_t):  
 raise ValueError  
 N = len(s\_matrix)  
 if i > N:  
 raise ValueError  
  
 if t == 1:  
 alpha = pai[i - 1] \* e\_matrix[i - 1][o1\_t[0]-1]  
 else:  
 alpha = sum((forward\_prob(t-1, j, o1\_t[:-1], s\_matrix, e\_matrix, pai)\*s\_matrix[j-1][i-1] for j in range(1, N+1))) \* e\_matrix[i - 1][o1\_t[-1]-1]  
  
 return alpha

# 前向算法  
def calc\_forward(seq, s\_matrix, e\_matrix, pai):

*'''****:param*** *seq: 长度为T的观测序列  
'''*  
  
 T = len(seq)  
 N = len(s\_matrix)  
 prob = sum((forward\_prob(T, i, seq, s\_matrix, e\_matrix, pai) for i in range(1, N+1)))  
  
 return prob

**2）后向算法**

后向概率：给定隐马尔科夫模型μ，定义在时刻t状态为qi的条件下，从t+1到T的部分观测序列为ot+1, ot+2, ……,oT的概率为后向概率，记作

βt (i) = P (ot+1, ot+2, ……, oT | it=qi, μ)

算法流程：

输入：隐马尔科夫模型μ，观测序列O；

输出：观测序列概率P(O | μ)。

(1)初始化后向概率，对最终时刻的所有状态qi规定：βT(i)=1, i=1, 2, 3, …, N

(2)对t=T-1, T-2, ……, 1，

，i=1，2，……，N

(3)终止

利用前向概率和后向概率的定义可以将观测序列概率P(O | μ)统一写成：

，t = 1，2，……，T-1

# 后向算法  
def backward\_prob(t, i, ot1\_T, s\_matrix, e\_matrix, T):  
 *'''* ***:param*** *t: 时刻t* ***:param*** *i: 状态i* ***:param*** *ot1\_T: t+1到T时刻的观测序列* ***:param*** *s\_matrix: 状态转移矩阵* ***:param*** *e\_matrix: 概率发射矩阵* ***:param*** *T: 观测序列的最大长度T* ***:return****: 时刻t状态i的后向概率  
 '''* if not isinstance(t, int) or t <= 0:  
 raise ValueError  
 if not isinstance(i, int) or i <= 0:  
 raise ValueError  
 if not isinstance(T, int) or T <= 0:  
 raise ValueError  
 if T - t != len(ot1\_T):  
 raise ValueError  
 N = len(s\_matrix)  
 if i > N:  
 raise ValueError

if t == T:  
 return 1  
 else:  
 beta = sum((s\_matrix[i-1][j-1] \* e\_matrix[j-1][ot1\_T[0]-1]\*backward\_prob(t+1, j, ot1\_T[1:], s\_matrix, e\_matrix, T) for j in range(1, N+1)))  
  
 return beta  
# 后向算法  
def calc\_backward(seq, s\_matrix, e\_matrix, pai):  
  
 T = len(seq)  
 N = len(s\_matrix)  
 prob = sum((backward\_prob(1, i, seq[1:], s\_matrix, e\_matrix, T)\*pai[i-1] \* e\_matrix[i-1][seq[0]-1] for i in range(1, N+1)))  
  
 return prob

动态规划的方法采用前一时刻的计算结果计算当前时刻的结果，避免了历史时刻状态的重复计算，从而降低了计算量，前向算法和后向算法的时间复杂度为O(TN2)阶。

**7.2.2学习问题**

**1）监督学习：**训练数据包括观测序列和对应的状态序列——**极大似然估计**。给定训练数据包含S个长度相同的观测序列和对应的状态序列{(O1,I1), (O2,I2), ……, (Os, Is)}，利用极大似然估计来估计隐马尔科夫模型的参数：

（1）转移概率aij的估计

设样本中时刻t处于状态i时刻t+1转移到状态j的频数为Aij，那么状态转移概率aij的估计是：

， i = 1，2，……，N；j = 1，2，……，N

（2）观测概率bj(k)的估计

设样本中状态为j并观测为k的频数Bjk，那么状态为j观测为k的概率bj(k)的估计是：

， j = 1，2，……，N；k = 1，2，……，M

（3）初始状态概率πi的估计为S个样本中初始状态为qi的频率。

监督学习需要使用由人工标注的训练数据，往往代价很高。

**2）非监督学习：**训练数据只有观测序列——**Baum-Welch算法**。给定训练数据只包含S个长度为T的观测序列{O1, O2, ……, Os}而没有对应的状态序列，那么隐马尔科夫模型是一个以观测序列数据为观测数据O，以状态序列数据为不可观测隐数据I的概率模型：

参数学习可以用EM算法实现。

（1）确定完全数据的对数似然函数

观测数据O = (o1, o2, ……, oT)

隐数据I = (i1, i2, ……, iT)

完全数据(O, I) = (o1, o2, ……, oT, i1, i2, ……, iT)

完全数据的对数似然函数log P(O, I | μ)

（2）EM算法的E步：求Q函数Q(μ, )：

其中，μ是要极大化的HMM参数，是HMM的当前估计

因为

所以函数Q(μ, )可以写成

（3）EM算法的M步：极大化Q函数Q(μ, μ)求模型参数A, B, π。

由于要极大化的参数在E步的式子中单独地出现在3个项中，所以只需对各项分别极大化。利用拉格朗日乘子法：

其中I(ot=vk)为指示函数，当且仅当ot=vk时I(ot=vk)为1，其余为0。

**Baum-Welch算法：**

输入：观测数据 O = (o1, o2, ……, oT)；

输出：隐马尔科夫模型参数 μ = (A, B, π)。

(1)初始化

对 n=0，选取aij(0), bj(k)(0), πi(0)，得到模型 μ(0) = (A(0), B(0), π(0))。

(2)递推。对 n=1, 2, ……,

其中γt(i)表示时刻t处于状态qi的概率，ξt(i, j)表示时刻t处于状态qi且时刻t+1处于状态qj的概率。

(3)终止。得到模型参数μ(n+1) = (A(n+1), B(n+1), π(n+1))。

**7.2.3解码问题**

**1）近似算法：**

在每个时刻t选择在该时刻最有可能出现的状态it\*，从而得到一个状态序列 I\* = (i1\*, i2\*, ……, iT\*)，将它作为预测的结果。

给定隐马尔科夫模型μ和观测序列O，在时刻t处于状态qi的概率γt(i)是：

在每一时刻t最有可能的状态it\*是：

， t = 1，2，……，T

从而得到状态序列I\* = (i1\*, i2\*, ……, iT\*)。

优点：计算简单。

缺点：不能保证预测的状态序列整体是最有可能的状态序列。

**2）维特比算法：**

根据动态规划原理，最优路径具有这样的特性：

如果最优路径在时刻t通过节点it\*，那么这一路径从节点it\*到终点iT\*的部分路径，对于从it\*到iT\*的所有可能的部分路径来说，必须是最优的。

定义两个变量δ和ψ：

在时刻t状态为i的所有单个路径(i1, i2, ……, it)中概率最大值为

， i = 1，2，……，N

由定义可得变量δ的递推公式：

i = 1，2，……，N；t = 1，2，……，T-1

在时刻t状态为i的所有单个路径(i1, i2, ……, it)中概率最大的路径的第t-1个节点为

， i = 1，2，……，N

算法流程：

输入：模型 μ=(A, B, π) 和观测 O={o1, o2, ……, oT}；

输出：最优路径 I\* = (i1\*, i2\*, ……, iT\*)。

(1)初始化

， i = 1，2，……，N

， i = 1，2，……，N

(2)递推。对 t = 2, 3, ……, T

， i = 1，2，……，N

， i = 1，2，……，N

(3)终止

(4)最优回溯路径。对 t = T-1, T-2, ……, 1

求得最优路径I\* = (i1\*, i2\*, ……, iT\*)。

# 维特比变量  
def get\_delta\_psi(t, i, o1\_t, s\_matrix, e\_matrix, pai):  
  
 N = len(s\_matrix)  
 if t == 1:  
 delta = pai[i-1] \* e\_matrix[i-1][o1\_t[0]-1]  
 psi = 0  
 else:  
 dt = np.array([get\_delta\_psi(t-1, j, o1\_t[:-1], s\_matrix, e\_matrix, pai)[0]\*s\_matrix[j-1][i-1] for j in range(1, N+1)])  
 idx = np.argmax(dt)  
 delta = dt[idx] \* e\_matrix[i-1][o1\_t[-1]-1]  
 psi = idx + 1  
  
 return delta, psi  
# 维特比算法  
def viterbi(seq, s\_matrix, e\_matrix, pai):  
  
 T = len(seq)  
 N = len(s\_matrix)  
 history = [[get\_delta\_psi(t, i, seq[:t], s\_matrix, e\_matrix, pai) for i in range(1, N + 1)] for t in range(1, T + 1)]  
 idx = np.argmax(list(zip(\*history[-1]))[0])  
 prob = history[-1][idx][0]  
 i\_T = idx + 1  
 i\_path = [i\_T]  
 for t in range(T, 1, -1):  
 i\_t = history[T-1][idx][1]  
 i\_path.insert(0, i\_t)  
 idx = i\_t - 1  
  
 return prob, tuple(i\_path)