隐马尔科夫模型

覃一发

2021 年 7 月 11 日

摘要

隐马尔科夫模型描述由隐藏的马尔科夫链随机生成观测序列的过程，属于生成模型，是可用于标注问题的统计学习模型。本章依次介绍其基本概念和要素，常见的三大任务-概率计算、学习、预测以及对应算法。隐马尔科夫模型广泛引用与语音识别、模式识别、自然语言处理、生物信息学等领域。

1 从红蓝球盒子开始

假设由3个盒子，每个盒子里有红、蓝两种颜色的球，数量由下表给出。

表 1: 各盒子的红、蓝球数量

	Box 1	**Box 2**	**Box 3**
Red	2	3	5
Blue	4	3	1

按照以下的方法抽球。

1. 从3个盒子里以等概率随机选取1个盒子，从该盒子随机抽取1个球，记录颜色后放回；
2. 从当前盒子随机转移到下一个盒子，转移规则是从盒子1到盒子2概率为1，盒子2转移到盒子1概率为0.4，到盒子3为0.6，盒子3停留自身或转移到盒子2概率都是0.5；

3. 确定转移的盒子后，记录颜色并放回；

4. 如此重复5次得到颜色的观测序列。

$$O = (Red, Red, Blue, Red, Blue) \tag{1}$$

对应的盒子序列是$S = (box1, box2, box3, box3, box2)$。

上述的抽球玩法正是隐马尔科夫模型的一个简单例子，其中包含如下3个重要元素。

初始3个盒子被抽中的概率π。

$$\pi = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}) \tag{2}$$

盒子转移的概率分布A。A_{ij} 表示从盒子i转移到盒子j的概率大小。

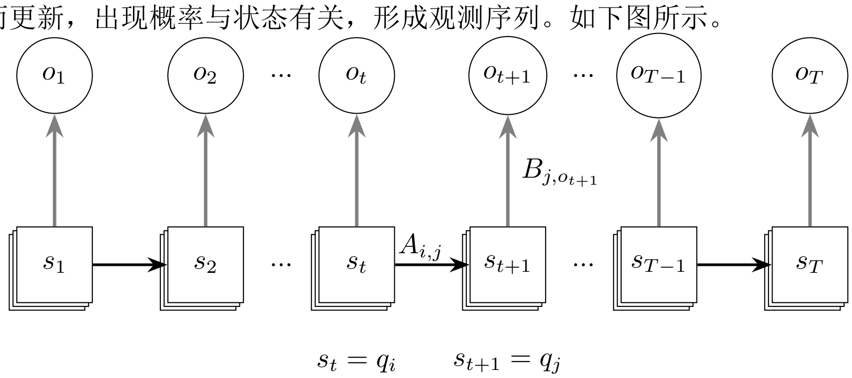
$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0.4 & 0 & 0.6 \\ 0 & 0.5 & 0.5 \end{bmatrix} \tag{3}$$

盒子序号和观测到的颜色的对应概率B。B_{ij} 表示从盒子i中抽到红色(j=0)、蓝色(j=1)的概率大小。

$$B = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{1}{2} & \frac{1}{2} \\ \frac{1}{6} & \frac{5}{6} \end{bmatrix} \tag{4}$$

2 隐马尔科夫基本概念

隐马尔可夫(Hidden Markov Model, 以下简称HMM)中有状态和观测。状态根据马尔科夫链在各状态间切换，形成隐藏状态序列。观测随状态更新而更新，出现概率与状态有关，形成观测序列。如下图所示。



描述HMM（一阶）需要三大要素，分别是初始状态的概率分布向量，状态转换的概率方阵，从状态对应各观测的概率矩阵，分别记为π， A，B，如Eq.5。

$$h = (\pi, A, B) \tag{5}$$

π_i是初始时刻状态为q_i的概率；A_{ij}是从状态q_i转移到q_j的概率；B_{ij}是状态q_i呈现为第j种观测相v_j的概率。状态的集合记为Q，观测的集合记为V，M,N分别是所有可能的观测、状态数量。

$$Q = \{q_1, q_2, ..., q_N\}, V = \{v_1, v_2, ..., v_M\} \tag{6}$$

长度为T的状态序列记为S，观测序列记为O。

$$S = (s_1, s_2, ..., s_T), O = (o_1, o_2, ..., o_T) \tag{7}$$

很自然地$s_i \in Q, i = 1, 2, ..., T; o_i \in V, i = 1, 2, ..., T$。

3 隐马尔科夫模型的三个基本任务

HMM有3个基本问题，费别是观测序列的概率计算、HMM参数估计、状态序列的预测解码问题。

- 概率计算。
  已知$h = (\pi, A, B)$,求特定观测序列O出现的概率$P(O|h)$。
- 参数估计。
  已知观测序列O，求$h = (\pi, A, B)$使得概率$P(O|h)$最大化。
- 预测解码。
  已知观测序列O以及$h = (\pi, A, B)$，求状态序列使得$P(S|h, O)$最大化。

3.1 观测序列的概率计算

3.1.1 前向算法

我们计算的目标是观测序列$O = (o_1, o_2, ..., o_T)$出现的概率$P(O|h)$。由于在概率计算任务里模型参数h已固定，为了方便后文的$P(O|h)$都改写为$P(O)$。

每个时刻都对应多种可能的隐藏状态，即序列包含了隐变量$s_t, t = 1, ..., T$，见图3.1.1。为了简单起见，我们只选取1个隐变量，利用边际概率公式把$P(O)$对隐变量展开,见公式8。前向算法选取的隐变量是最后的节点状态s_T。每个时刻的状态s_t都属于集合$Q = \{q_1, q_2, ..., q_M\}$。

$$P(O) = \sum_{i=1}^{N} P(o_1, o_2, ..., o_T, s_T = q_i) \tag{8}$$

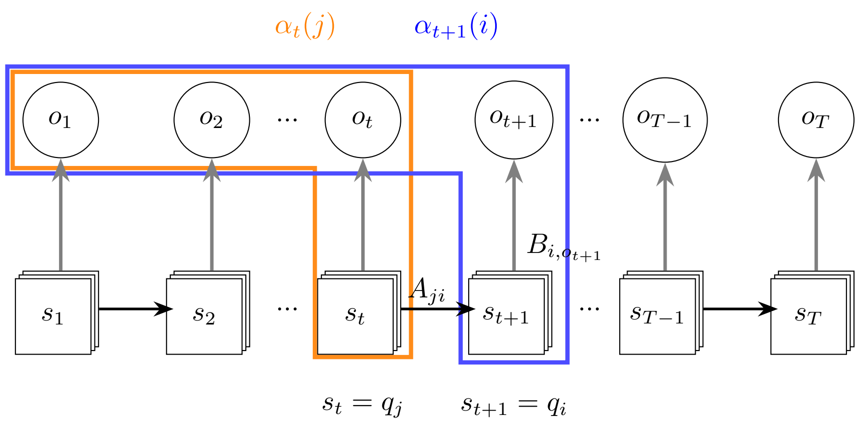
为了描述方便，我们定义$\alpha_t(i)$，见公式9。

$$\alpha_t(i) = P(o_1, o_2, ..., o_t, s_t = q_i) \tag{9}$$

那么观测序列的概率即可表示为$\alpha_T(i)$之和。

$$P(O) = \sum_{i=1}^{N} \alpha_T(i) \tag{10}$$

如图3.1.1所示。



从图中可以观察到相邻时刻的α存在依赖关系。$\alpha_t(i)$描述的是观测序列从o_1到o_t，且t时刻状态$s_t = q_i$的概率。那么$\alpha_{t+1}(i)$，需要计算t时刻所有可能的状态j对应的概率$\alpha_t(j)$，乘以转移到t+1轮的状态i的概率A_{ji}，并对j求和之后，再乘以在t+1时刻被观测到o_{t+1}的概率，如公式11所示。

$$\alpha_{t+1}(i) = \sum_{j=1}^{N} \alpha_t(j) A_{ji} B_{i,o_{t+1}} \tag{11}$$

α_1稍微特殊一点。

$$\alpha_1(i) = P(o_1, s_1 = q_i) = \pi_i B_{i,o_1} \tag{12}$$

显然，从α_1开始递推，直到α_T，对隐状态求和即求得$P(O)$。

算法前向算法(矩阵形式)

输入：$\pi \in \mathbf{R^n}, A \in \mathbf{R^{n\times n}}, B \in \mathbf{R^{n\times m}}, O = (o_1, o_2, ..., o_T)$

输出：观测序列O的概率P

$\boldsymbol{\alpha} = \mathbf{0}^{n\times T}$

$\boldsymbol{\alpha}_{:,1} = \pi \odot \boldsymbol{B}_{:,o_1}$

while $t < T$ do

    $\boldsymbol{\alpha}_{:,t+1} = (\boldsymbol{A^T}\boldsymbol{\alpha}_{:,t}) \odot \boldsymbol{B}_{:,o_{t+1}}$

    $t \leftarrow t + 1$

end while

$P = \mathbf{1} \bullet \boldsymbol{\alpha}_{:,T}$

return P

显而易见，对于时刻t，$A^T\alpha_{\mathbf{t}}$共N^2次乘法，哈达玛积$\odot$ 共N次乘法，过程重复T次，共计$O((N^2+N)T)$即$O(N^2T)$的算法复杂度。

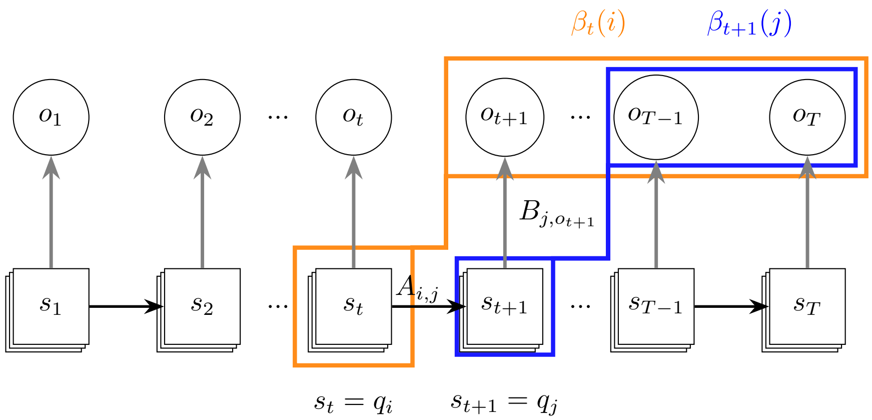
3.1.2 后向算法

前向算法选取的是最后时刻状态作为隐变量展开$P(O)$，那么不妨试一试使用最初时刻状态展开。

$$\begin{aligned} P(O) &= \sum_{i=1}^{N} P(o_1, o_2, ..., o_T, s_1 = q_i) \\ &= \sum_{i=1}^{N} P(s_1 = q_i) P(o_1, o_2, ..., o_T | s_1 = q_i) \\ &= \sum_{i=1}^{N} P(s_1 = q_i) P(o_1 | s_1 = q_i) P(o_2, ..., o_T | s_1 = q_i) \end{aligned} \tag{13}$$

有趣的是，后向算法并没有选择$\sum_{i=1}^{N} P(o_1, o_2, ..., o_T, s_1 = q_i)$作为递推终点，而是进一步分解。公式13 第一、二步是利用边际概率公式、联合概

率公式。第三步是因为$P(o_1)$和$P(o_2,..,o_T)$并不独立，受制于时刻1状态s_1，但当s_1给定后则实现了条件独立。



后向算法选择把$P(o_2,...,o_T|s_1 = q_i)$作为递推终点，记为$\beta_1(i)$。那么根据矩阵A和B的定义，$P(O)$如下。

$$\begin{aligned} P(O) &= \sum_{i=1}^{N} \pi_i B_{i,o_1} P(o_2,...,o_T|s_1 = q_i) \\ &= \sum_{i=1}^{N} \pi_i B_{i,o_1} \beta_1(i) \end{aligned} \tag{14}$$

由特殊到普遍，$\beta_t(i)$为

$$\beta_t(i) = \sum_{i=1}^{N} P(o_{t+1},...,o_T|s_t = q_i) \tag{15}$$

实际上$\beta_t(i)$表示的是t时刻从状态q_i出发，观测序列为$(o_{t+1},...,o_T)$的概率。观察图3.1.1,可以发现相邻时刻的依赖关系。

$$\beta_t(i) = \sum_{j=1}^{N} A_{i,j} B_{j,o_t} \beta_{t+1}(j) \tag{16}$$

T时刻本身已经是终点，所以概率$\beta_T(i)$为1。

$$\beta_T(i) = 1 \tag{17}$$

从β_T开始往前推，得到β_1，再利用公式14求得目标概率。

矩阵形式向我们展示了后向算法为何是后向。

$$
\begin{aligned}
\boldsymbol{\beta} &\in \boldsymbol{R}^{n\times T} \\
\boldsymbol{\beta}_{:,T} &= \mathbf{1} \\
\boldsymbol{\beta}_{:,t} &= \boldsymbol{A}(\boldsymbol{\beta}_{:,t+1} \odot \boldsymbol{B}_{:,o_{t+1}}) \\
P(O) &= \pi \bullet (\mathbf{B}_{:,\mathbf{o_1}} \odot \boldsymbol{\beta}_{:,1})
\end{aligned}
\tag{18}
$$

算法: 后向算法(矩阵形式)

输入：$\pi \in \mathbf{R^n}, A \in \mathbf{R^{n\times n}}, B \in \mathbf{R^{n\times m}}, O = (o_1, o_2, ..., o_T)$

输出：观测序列O的概率P

---

1 $\boldsymbol{\beta} = \mathbf{0}^{n\times T}$

2 $\boldsymbol{\beta}_{:,t} = \mathbf{1}$

3 while $t > 1$ do

3 $\quad\boldsymbol{\beta}_{:,t} = \boldsymbol{A}(\boldsymbol{\beta}_{:,t+1} \odot \boldsymbol{B}_{:,o_{t+1}})$

4 $\quad t \leftarrow t-1$

5 end while

6 $P = \pi \bullet (\mathbf{B}_{:,\mathbf{o_1}} \odot \boldsymbol{\beta}_{:,1})$

7 return P

类似的分析可得算法复杂度依然为$O(N^2T)$。

3.1.3 双向算法

利用前向概率和后向概率的定义，可以把观测序列的概率写成如下形式。

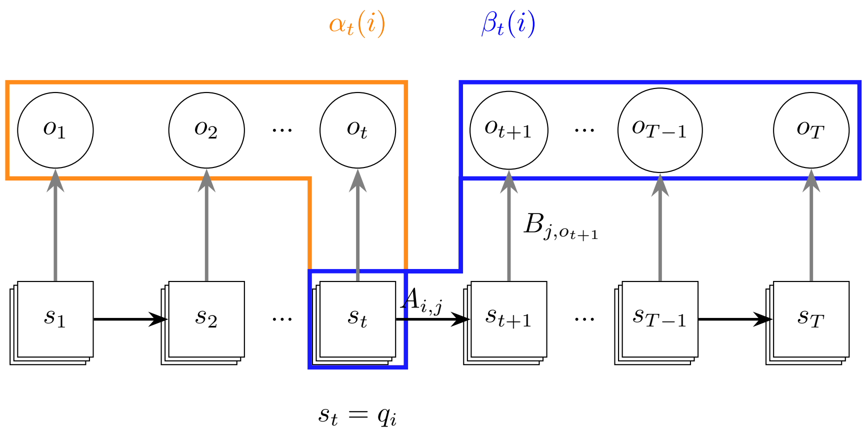
$$P(O) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) \tag{19}$$

李航《统计学习方法》把概率以下方式展开。

$$P(O) = \sum_{i,j=1}^{M} \alpha_t(i)A_{i,j}B_{j,o_{t+1}}\beta_{t+1}(j) \tag{20}$$

但这实际上是把$\beta_t(i)$按照公式16展开而已，显得稍微繁冗。因此我们采用公式19即可。由于本文讨论的是一阶马尔科夫链，因此序列$o_1, ..., o_t$与$o_{t+1}, ..., o_T$仅

仅通过s_t这一时刻状态关联起来。如果s_t给定，那么 $P(o_1,...,o_t,s_t=q_i)$与$P(o_{t+1},...,o_T,s_t=q_i)$是独立的。参照下图。



算法: 双向算法(矩阵形式)

输入：$\pi \in \mathbf{R^n}, A \in \mathbf{R^{n\times n}}, B \in \mathbf{R^{n\times m}}, O=(o_1,o_2,...,o_T)$

输出：观测序列O的概率P，各个时刻的前向概率矩阵$\boldsymbol{\alpha}$和后向矩阵$\boldsymbol{\beta}$

$\boldsymbol{\alpha} = \mathbf{0}^{n\times T}, \boldsymbol{\beta} = \mathbf{0}^{n\times T}$
$\boldsymbol{\beta}_{:,T} = \mathbf{1}$
while $t > 1$ do
    $\boldsymbol{\beta}_{:,t} = \boldsymbol{A}(\boldsymbol{\beta}_{:,t+1} \odot \boldsymbol{B}_{:,o_{t+1}})$
    $t \leftarrow t-1$
end while
$\boldsymbol{\alpha}_{:,1} = \pi \odot \boldsymbol{B}_{:,o_1}$
while $t < T$ do
    $\boldsymbol{\alpha}_{:,t+1} = (\boldsymbol{A^T}\boldsymbol{\alpha}_{:,t}) \odot \boldsymbol{B}_{:,o_{t+1}}$
    $t \leftarrow t+1$
end while
$d = (\boldsymbol{\alpha} \odot \boldsymbol{\beta})_{:,T/2}$
$P = d/||d||$
return $P, \boldsymbol{\alpha}, \boldsymbol{\beta}$

从现在起，前向概率和后向概率分别用$\boldsymbol{\alpha}$和$\boldsymbol{\beta}$的矩阵元来表示。

$$
\begin{aligned}
\alpha_t(i) = \boldsymbol{\alpha}_{i,t} = P(o_1, o_2, ..., o_t, s_t = q_i) \\
\beta_t(i) = \boldsymbol{\beta}_{i,t} = P(o_{t+1}, ..., o_T, s_t = q_i)
\end{aligned}
\tag{21}
$$

3.1.4 若干概率和期望值

利用序列O概率的计算产物，可以得到一些有用的概率和期望值。

1. 给定模型h和观测序列O，时刻t状态为q_i的概率。

类似地，我们也使用$\boldsymbol{\gamma}$的矩阵元$\boldsymbol{\gamma}_{i,t}$来表示给定时刻$t$状态为$q_i$的概率。

$$
\boldsymbol{\gamma}_{i,t} = P(s_t = q_i | o_1, ..., o_T) \tag{22}
$$

参照前文的前向后向算法以及示意图，轻易得到$\gamma_t(i)$。

$$
\begin{aligned}
\boldsymbol{\gamma}_{i,t} &= \frac{\boldsymbol{\alpha}_{i,t}\boldsymbol{\beta}_{i,t}}{P(O)} \\
&= \frac{\boldsymbol{\alpha}_{i,t}\boldsymbol{\beta}_{i,t}}{\sum_{i=1}^{N} \boldsymbol{\alpha}_{i,t}\boldsymbol{\beta}_{i,t}}
\end{aligned}
\tag{23}
$$

写成矩阵形式。

$$
\begin{aligned}
\boldsymbol{\gamma} &= \frac{\boldsymbol{\alpha} \odot \boldsymbol{\beta}}{P(O)} \\
P(O) &= \mathbf{1} \bullet (\boldsymbol{\alpha} \odot \boldsymbol{\beta})_{:,T/2}
\end{aligned}
\tag{24}
$$

2. 给定模型h和观测序列O，时刻t状态为q_i且时刻$t+1$状态为q_j的概率。

类似地，我们用三阶矩阵$\boldsymbol{\gamma}$的矩阵元$\boldsymbol{\gamma}_{t,i,j}$表示这一概率。

$$
\boldsymbol{\xi}_{t,i,j} = P(s_t = q_i, s_{t+1} = q_j | o_1, ..., o_T) \tag{25}
$$

参照前文的双算法20，轻易得到$\boldsymbol{\xi}_{t,i,j}$。

$$
\boldsymbol{\xi}_{t,i,j} = \frac{\boldsymbol{\alpha}_{i,t}\boldsymbol{A}_{i,j}\boldsymbol{B}_{j,o_{t+1}}\boldsymbol{\beta}_{j,t+1}}{P(O)} \tag{26}
$$

写成矩阵形式。

$$
\boldsymbol{\xi}_{t,:,:} = \frac{\boldsymbol{A} \odot (\boldsymbol{\alpha}_{:,t} \otimes (\boldsymbol{B}_{:,o_{t+1}} \odot \boldsymbol{\beta}_{:,t+1}))}{P(O)}, t = 1, 2, ..., T-1 \tag{27}
$$

3. 给定观测O状态q_i出现次数的期望值为$\sum_{t=1}^{T} \boldsymbol{\gamma}_{i,t}$。

4. 给定观测O状态q_i转移到q_j次数的期望值为$\sum_{t=1}^{T-1} \boldsymbol{\xi}_{t,i,j}$。

3.2 HMM的参数估计

隐马尔科夫模型的学习，如果训练数据同时包含观测序列和对应的状态序列，那么可以由监督学习实现；如果仅包含观测序列，则依靠无监督学习。后者可以依靠Baum-Welch算法，实际上该算法是EM算法在HMM的特化。

3.2.1 HMM的有监督学习：给定状态序列、观测序列估计参数

如果训练数据包含n个长度相同的观测序列和对应的状态序列$\{(O_1, S_1), (O_2, S_2), ..., (O_n, S_n)\}$，那么利用极大似然来估计参数即可。

1. 估计初始状态概率π 把样本中初始时刻为状态q_i的频数除以所有频数之和即可。

$$\pi_i = \frac{n_{s_1=q_i}}{\sum_i n_{s_1=q_i}} \tag{28}$$

2. 估计转移概率$A_{i,j}$ 把样本中t时刻从状态q_i转移到t+1时刻q_j的频数记为$\widetilde{A}_{i,j}$，则转移矩阵A的矩阵元$A_{i,j}$如下。

$$A_{i,j} = \frac{\widetilde{A}_{i,j}}{\sum_{i,j} \widetilde{A}_{i,j}} \tag{29}$$

3. 估计观测概率$B_{j,k}$ 把样本中t时刻状态为q_j，观测$o_t = v_k$的次数记为$\widetilde{B}_{j,k}$，则观测矩阵B的矩阵元$B_{j,k}$如下。

$$B_{j,k} = \frac{\widetilde{B}_{j,k}}{\sum_k \widetilde{B}_{j,k}} \tag{30}$$

3.2.2 HMM的无监督学习：仅从观测序列估计参数

训练数据包含n个长度相同的观测序列$\{O_1, O_2, ..., O_n\}$，以及隐藏状态集合Q的元素个数，目标是学习HMM的参数 $h = (A, B, \pi)$。显然隐藏状态序列是隐变量，HMM是含有隐变量的概率模型，其参数学习可以用EM来实现。

$$P(O|h) = \sum_S P(O|S,h)P(S|h) \tag{31}$$

我们先来简短回顾一下EM算法。EM算法核心是对Q函数的极大化。Q函数的意义是先用上轮参数条件、已有数据算出隐变量的分布，再求当前

轮次的参数条件下的完全数据的概率对数在隐变量分布下的期望。说白了其实就是完全数据在隐变量分布下的“似然”期望值，Q函数越大，说明新参数下的模型与数据“相似”程度进步越大。Y, Z, θ分别为观测变量，隐变量，模型参数，$\theta^{(i)}$是上轮模型参数。

$$Q(\theta, \theta^{(i)}) = \sum_Z P(Z|Y, \theta^{(i)}) \log P(Y, Z|\theta) \tag{32}$$

如果我们给该函数乘以一个常数项$P(Y|\theta^{(i)})$，那么Q函数的“似然”的本质就更容易理解了。显然该常数不影响新参数θ 的估计。

$$\begin{aligned} Q(\theta, \theta^{(i)}) P(Y|\theta^{(i)}) &= P(Y|\theta^{(i)}) \sum_Z \frac{P(Y, Z|\theta^{(i)})}{P(Y|\theta^{(i)})} \log P(Y, Z|\theta) \\ &= \sum_Z P(Y, Z|\theta^{(i)}) \log P(Y, Z|\theta) \end{aligned} \tag{33}$$

回到HMM,我们先确定完全数据的对数似然，即所谓的E步。$P(O, S|\widetilde{h})$是根据老参数估计的完全数据的概率。

$$Q(h, \widetilde{h}) = \sum_S P(O, S|\widetilde{h}) \log P(O, S|h) \tag{34}$$

结合HMM，完全数据概率如下(分配-呈现-转换-呈现-...转换-呈现)。

$$P(O, S|h) = \pi_{s_1} B_{s_1, o_1} A_{s_1, s_2} B_{s_2, o_2} ... A_{s_{T-1} s_T} B_{s_T, o_T} \tag{35}$$

于是，完全数据的对数似然可以写成

$$\begin{aligned} \log Q(h, \widetilde{h}) = &\sum_S P(O, S|\widetilde{h}) \log \pi_{s_1} + \\ &\sum_S P(O, S|\widetilde{h}) \sum_{t=1}^{t=T-1} \log A_{s_t, s_{t+1}} + \\ &\sum_S P(O, S|\widetilde{h}) \sum_{t=1}^{t=T} \log B_{s_t, o_t} \end{aligned} \tag{36}$$

我们注意到，上式右边三项分别只和与π, A, B有关，那么在对完全数据极大化估计$h = (\pi, A, B)$时，可以分项独立进行。但是，上式是对所有可能存在的长度为T的状态序列进行求和，序列的个数等于状态数的序列长度次方N^T，计算复杂度过高，我们需要分别转换一下。

例如对于式36右边第一项，可以作如下处理。

$$\sum_S P(O,S|\widetilde{h})\log\pi_{s_1} = \sum_i P(O,s_1=q_i|\widetilde{h})\log\pi_i \tag{37}$$

这是因为$P(O,S|\widetilde{h})$是对序列S求和，但S中与π_{s_1}有关的仅仅是s_1而已，那么改成针对s_1的所有可能取值q_i即可达到同样的目的。

同理，式36右边第二项针对S和t求和，与S有关的仅有s_t与s_{t+1}两项，因此改成如下形式。

$$\sum_S P(O,S|\widetilde{h})\sum_{t=1}^{t=T-1}\log A_{s_t,s_{t+1}} = \sum_{i=1}^{i=N}\sum_{j=1}^{j=N}\sum_{t=1}^{t=T-1} P(O,s_t=q_i,s_{t+1}=q_j|\widetilde{h})\log A_{i,j} \tag{38}$$

式36右边第三项针对S和t求和，与S有关的仅有s_t一项，改成针对s_t的所有可能取值求和。但给定观测序列O后o_t已全部确定，那么改成如下形式。

$$\sum_S P(O,S|\widetilde{h})\sum_{t=1}^{t=T}\log B_{s_t,o_t} = \sum_{i=1}^{i=N}\sum_{j=1}^{j=M}\sum_{t=1}^{t=T-1} P(O,s_t=q_i|\widetilde{h})I(o_t=v_j)\log B_{i,j} \tag{39}$$

以下我们可以开展EM算法的M步了，即完全数据的对数似然针对模型参数进行极大化。

针对第一项，注意到π_i满足约束条件$\sum_i \pi_i$，利用拉格朗日乘子法，写出拉格朗日函数。

$$\sum_i^N P(O,s_1=q_i|\widetilde{h})\log\pi_i + \lambda(\sum_i \pi_i - 1) \tag{40}$$

对π_i求导使其导数为0。

$$P(O,s_1=q_i|\widetilde{h}) + \lambda\pi_i = 0 \tag{41}$$

对i求和之后得到λ如下。

$$\lambda = -P(O|\widetilde{h}) \tag{42}$$

代入式41，得到π_i。

$$\begin{aligned}\pi_i &= \frac{P(O,s_1=q_i|\widetilde{h})}{P(O|\widetilde{h})} \\ &= \gamma_{i,t}\end{aligned} \tag{43}$$

同样方式处理第二项，约束条件为$\sum_{j=1}^{N} A_{i,j} = 1$，利用拉格朗日乘子法写出函数并对$A_{i,j}$求导，使其为0。类似可得$A_{i,j}$。其实这里根据i=1,2,...,N进行了N次操作。

$$\begin{aligned} A_{i,j} =& \frac{\sum_{t=1}^{t=T-1} P(O, s_t = q_i, s_{t+1} = q_j|\widetilde{h})}{\sum_{t=1}^{t=T} P(O, s_t = q_i|\widetilde{h})} \\ =& \frac{\sum_{t=1}^{t=T-1} \boldsymbol{\xi}_{t,i,j}}{\sum_{t=1}^{t=T} \boldsymbol{\gamma}_{i,t}} \end{aligned} \tag{44}$$

处理第三项时，约束条件为$\sum_{j=1}^{M} B_{i,j} = 1$，类似操作可得$B_{i,j}$。由于约束条件有M个，同样进行了M次操作。

$$\begin{aligned} B_{i,j} &= \frac{\sum_{t=1}^{t=T-1} P(O, s_t = q_i|\widetilde{h}) I(o_t = v_j)}{\sum_{t=1}^{t=T-1} P(O, s_t = q_i|\widetilde{h})} \\ &= \frac{\sum_{t=1}^{t=T} \boldsymbol{\gamma}_{i,t} I(o_t = v_j)}{\sum_{t=1}^{t=T} \boldsymbol{\gamma}_{i,t}} \end{aligned} \tag{45}$$

我们注意到，以上结果都可以利用双向算法的副产物的副产物$\boldsymbol{\gamma}$和$\boldsymbol{\xi}$来表示。

算法 Baun-Welch算法

输入：观测序列 $O=(o_1,o_2,...,o_T)$，初始HMM模型$h^{(0)}=(\pi^{(0)},\boldsymbol{A^{(0)}},\boldsymbol{B^{(0)}})$
输出：HMM参数 $h=(\pi,\boldsymbol{A},\boldsymbol{B})$

(1) 对于$i=0$，把O与$h^{(0)}$输入双向模型，得到$P,\boldsymbol{\alpha^{(0)}},\boldsymbol{\beta^{(0)}}$
(2) 递推。对于$i=1,2,...$，

$$\begin{aligned}&\boldsymbol{\gamma}=\boldsymbol{\alpha^{(i)}}\odot\boldsymbol{\beta^{(i)}}\\&\boldsymbol{\gamma}_{:,t}=\boldsymbol{\gamma}_{:,t}/||\boldsymbol{\gamma}_{:,t}||,t=1,2,...,T\end{aligned}\tag{46}$$

3.3 预测状态序列

3.3.1 近似算法

近似算法的思路是在每个时刻t选择在该时刻出现概率最高的状态s_t^*，得到状态序列$S^*=(s_1^*,s_2^*,...,s_T^*)$，并作为预测的结果。

给定HMM模型$h=(\pi,\boldsymbol{A},\boldsymbol{B})$和观测序列O，在时刻t处于状态$q_i$的概率是$\boldsymbol{\gamma}_{i,t}$。

$$\gamma_{i,t}=\frac{\boldsymbol{\alpha}_{i,t}\boldsymbol{\beta}_{i,t}}{P(O)}\tag{47}$$

在每一时刻t，最有可能的状态$i^*=argmax_{1\le i\le N}\boldsymbol{\gamma}_{i,t}$，$\quad t=1,2,...,T$。从而得到序列$S^*=(s_1^*,s_2^*,...,s_T^*)$。

近似算法虽然计算简单，但是每个节点局部最优不保证整体最优，因为这样预测出的状态序列中可能存在转移概率为0的相邻状态。

3.3.2 Viterbi算法

Viterbi算法 Φ