Hidden Markov Model 03 Learning

Chen Gong

09 January 2020

首先我们回顾一下，上一节讲的有关 Evaluation 的问题。Evaluation 可以被我们描述为在已知模型 λ 的情况下，求观察序列的概率。也就是：

$$P(O|\lambda)=\sum_{I}P(O,I|\lambda)=\sum_{i_1}\cdots\sum_{i_T}\pi_{i_1}\prod_{t=2}^{T}a_{i_{t-1},i_t}\prod_{t=1}^{T}b_{i_1}(o_t) \tag{1}$$

此时的算法复杂度为 $\mathcal{O}(N^T)$。算法的复杂度太高了，所以，就有了后来的 forward 和 backward 算法。那么就有如下定义：

$$\begin{aligned}
&\alpha_t(i)=P(o_1,\cdots,o_t,i_t=q_i|\lambda)\\
&\beta_t(i)=P(o_{t+1},\cdots,o_T|i_t=q_i,\lambda)\\
&\alpha_T(i)=P(O,i_T=q_i)\rightarrow P(O|\lambda)=\sum_{i=1}^{N}\alpha_T(i)\\
&\beta_1(i)=P(o_2,\cdots,o_T|i_1=q_i,\lambda)\rightarrow P(O|\lambda)=\sum_{i=1}^{N}\pi_i b_i(o_1)\beta_1(i)
\end{aligned} \tag{2}$$

而使用 forward 和 backward 算法的复杂度为 $\mathcal{O}(TN^2)$。这一节，我们就要分析 Learning 的部分，Learning 就是要在已知观测数据的情况下求参数 λ，也就是：

$$\lambda_{MLE}=\arg\max_{\lambda}P(O|\lambda) \tag{3}$$

1 Learning

我们需要计算的目标是：

$$\lambda_{MLE}=\arg\max_{\lambda}P(O|\lambda) \tag{4}$$

又因为：

$$P(O|\lambda)=\sum_{i_1}\cdots\sum_{i_T}\pi_{i_1}\prod_{t=2}^{T}a_{i_{t-1},i_t}\prod_{t=1}^{T}b_{i_1}(o_t) \tag{5}$$

对这个方程的 λ 求偏导，实在是太难算了。所以，我们考虑使用 EM 算法。我们先来回顾一下 EM 算法：

$$\theta^{(t+1)}=\arg\max_{\theta}\int_{z}\log P(X,Z|\theta)\cdot P(Z|X,\theta^{(t)})dZ \tag{6}$$

而 $X\rightarrow O$ 为观测变量；$Z\rightarrow I$ 为隐变量，其中 I 为离散变量；$\theta\rightarrow\lambda$ 为参数。那么，我们可以将公式改写为：

$$\lambda^{(t+1)} = \arg\max_{\lambda} \sum_{I} \log P(O, I|\lambda) \cdot P(I|O, \lambda^{(t)}) \tag{7}$$

这里的 $\lambda^{(t)}$ 是一个常数，而：

$$P(I|O, \lambda^{(t)}) = \frac{P(I, O|\lambda^{(t)})}{P(O|\lambda^{(t)})} \tag{8}$$

并且 $P(O|\lambda^{(t)})$ 中 $\lambda^{(t)}$ 是常数，所以这项是个定量，与 λ 无关，所以 $\frac{P(I,O|\lambda^{(t)})}{P(O|\lambda^{(t)})} \propto P(I, O|\lambda^{(t)})$。所以，我们可以将等式 (7) 改写为：

$$\lambda^{(t+1)} = \arg\max_{\lambda} \sum_{I} \log P(O, I|\lambda) \cdot P(I, O|\lambda^{(t)}) \tag{9}$$

这样做有什么目的呢？很显然这样可以把 $\log P(O, I|\lambda)$ 和 $P(I, O|\lambda^{(t)})$ 变成一种形式。其中，$\lambda^{(t)} = (\pi^{(t)}, \mathcal{A}^{(t)}, \mathcal{B}^{(t)})$，而 $\lambda^{(t+1)} = (\pi^{(t+1)}, \mathcal{A}^{(t+1)}, \mathcal{B}^{(t+1)})$。

我们定义：

$$Q(\lambda, \lambda^{(t)}) = \sum_{I} \log P(O, I|\lambda) \cdot P(O, I|\lambda^{(t)}) \tag{10}$$

而其中，

$$P(O|\lambda) = \sum_{i_1} \cdots \sum_{i_T} \pi_{i_1} \prod_{t=2}^{T} a_{i_{t-1}, i_t} \prod_{t=1}^{T} b_{i_1}(o_t) \tag{11}$$

所以，

$$Q(\lambda, \lambda^{(t)}) = \sum_{I} \left[\left(\log \pi_{i_1} + \sum_{t=2}^{T} \log a_{i_{t-1}, i_t} + \sum_{t=1}^{T} \log b_{i_t}(o_t) \right) \cdot P(O, I|\lambda^{(t)}) \right] \tag{12}$$

2 以 $\pi^{(t+1)}$ 为例

这小节中我们以 $\pi^{(t+1)}$ 为例，在公式 $Q(\lambda, \lambda^{(t)})$ 中，$\sum_{t=2}^{T} \log a_{i_{t-1}, i_t}$ 与 $\sum_{t=1}^{T} \log b_{i_t}(o_t)$ 与 π 无关，所以，

$$\begin{aligned} \pi^{(t+1)} &= \arg\max_{\pi} Q(\lambda, \lambda^{(t)}) \\ &= \arg\max_{\pi} \sum_{I} [\log \pi_{i_1} \cdot P(O, I|\lambda^{(t)})] \\ &= \arg\max_{\pi} \sum_{i_1} \cdots \sum_{i_T} [\log \pi_{i_1} \cdot P(O, i_1, \cdots, i_T|\lambda^{(t)})] \end{aligned} \tag{13}$$

我们观察 $\{i_2, \cdots, i_T\}$ 就可以知道，联合概率分布求和可以得到边缘概率。所以：

$$\begin{aligned} \pi^{(t+1)} &= \arg\max_{\pi} \sum_{i_1} [\log \pi_{i_1} \cdot P(O, i_1|\lambda^{(t)})] \\ &= \arg\max_{\pi} \sum_{i=1}^{N} [\log \pi_i \cdot P(O, i_1 = q_i|\lambda^{(t)})] \qquad (s.t. \sum_{i=1}^{N} \pi_i = 1) \end{aligned} \tag{14}$$

2.1 拉格朗日乘子法求解

根据拉格朗日乘子法，我们可以将损失函数写完：

$$\mathcal{L}(\pi,\eta)=\sum_{i=1}^{N}\log\pi_i\cdot P(O,i_1=q_i|\lambda^{(t)})+\eta(\sum_{i=1}^{N}\pi_i-1) \tag{15}$$

使似然函数最大化，则是对损失函数 $\mathcal{L}(\pi,\eta)$ 求偏导，则为：

$$\frac{\mathcal{L}}{\pi_i}=\frac{1}{\pi_i}P(O,i_1=q_i|\lambda^{(t)})+\eta=0 \tag{16}$$

$$P(O,i_1=q_i|\lambda^{(t)})+\pi_i\eta=0 \tag{17}$$

又因为 $\sum_{i=1}^{N}\pi_i=1$，所以，我们将公式 (17) 进行求和，可以得到：

$$\sum_{i=1}^{N}P(O,i_1=q_i|\lambda^{(t)})+\pi_i\eta=0\Rightarrow P(O|\lambda^{(t)})+\eta=0 \tag{18}$$

所以，我们解得 $\eta=-P(O|\lambda^{(t)})$，从而推出：

$$\pi_i^{(t+1)}=\frac{P(O,i_1=q_i|\lambda^{(t)})}{P(O|\lambda^{(t)})} \tag{19}$$

进而，我们就可以推导出 $\pi^{(t+1)}=(\pi_1^{(t+1)},\pi_2^{(t+1)},\cdots,\pi_N^{(t+1)}$。而 $\mathcal{A}^{(t+1)}$ 和 $\mathcal{B}^{(t+1)}$ 也都是同样的求法。这就是大名鼎鼎的 Baum Welch 算法，实际上思路和 EM 算法一致。不过在 Baum Welch 算法诞生之前，还没有系统的出现 EM 算法的归纳。所以，这个作者还是很厉害的。