

1 最基本的 HMM 模型

1.1 模型定义

1.1.1 符号

序列长度: T

一个状态序列: $s = s_1 s_2 \dots s_T$

一个观察序列: $o = o_1 o_2 \dots o_T$

状态符号数: N

观察符号数: M

状态符号集: $S = \{S_1, S_2, \dots, S_N\}$, 有 $s_t \in S, 1 \leq t \leq T$

观察符号集: $O = \{O_1, O_2, \dots, O_M\}$, 有 $o_t \in O, 1 \leq t \leq T$

1.1.2 模型假设

状态序列的一阶马尔可夫性, 即每个状态变量仅仅依赖于上一个状态变量:

$$P(s_t | s_1, s_2, \dots, s_{t-1}) = P(s_t | s_{t-1}), \text{ 其中 } 2 \leq t \leq T \quad (1)$$

时序平稳性, 即转移概率分布不随时间变化:

$$P(s_{t_1} | s_{t_1-1}) = P(s_{t_2} | s_{t_2-1}), \quad 2 \leq t_1, t_2 \leq T \quad (2)$$

每个观察变量仅仅依赖于与之对应的状态变量:

$$P(o_t | o_1, o_2, \dots, o_{t-1}, o_{t+1}, \dots, o_T, s_1, s_2, \dots, s_T) = P(o_t | s_t) \quad (3)$$

1.1.3 模型参数

一般来说, 状态符号和观察符号的个数 N 和 M 是由具体的问题场景预先确定下来了, 所以我们感兴趣的 HMM 模型参数如下:

$$a_{ij} = P(s_t = S_j | s_{t-1} = S_i)$$

$$b_{jk} = P(o_t = O_k | s_t = S_j)$$

$$\pi_i = P(s_1 = S_i)$$

其中 $1 \leq i, j \leq N, 1 \leq k \leq M, 2 \leq t \leq T$, 故所有的 a_{ij} 构成矩阵 $A_{N \times N}$, 所有的 b_{jk} 构成矩阵 $B_{N \times M}$, 所有的 π_i 构成向量 $\pi_{N \times 1}$, 记三元组:

$$\lambda = (A, B, \pi)$$

就表示 HMM 模型的所有参数。

1.2 三个基本问题

当我们定义了如上所述的 HMM 模型之后，要用它来干一些有意义的事情之前，要先解决三个基本问题：

- 已知一组模型参数 $\lambda = (A, B, \pi)$ ，给定一个观察序列一个观察序列 $o = o_1 o_2 \dots o_T$ ，如何**高效地**计算出现的概率 $P(o|\lambda)$ ？
- 已知一组模型参数 $\lambda = (A, B, \pi)$ ，给定一个观察序列一个观察序列 $o = o_1 o_2 \dots o_T$ ，如何按照某种有意义的准则来找出一个状态序列 $s = s_1 s_2 \dots s_T$ ，使之能够最好地“解释该观察序列的出现”？
- 给定一个观察序列一个观察序列 $o = o_1 o_2 \dots o_T$ ，如何求使这个观察序列出现概率最大的一组模型参数 $\lambda_0 = \arg \max_{\lambda} P(o|\lambda)$ ？

第一个问题属于用模型进行评估（evaluation）的问题，第二个问题属于用模型和数据进行推断（inference）的问题（我瞎猜的），第三个问题属于对模型进行参数优化（parameter optimization）的问题。。

1.2.1 第一个问题：暴力算法

现在，有了模型参数 λ ，有了观察序列 $o = o_1 o_2 \dots o_T$ ，我们可以先假设知道状态序列 $s = s_1 s_2 \dots s_T$ ，可以容易写出条件概率 $P(o|s, \lambda)$ 和 $P(s|\lambda)$ ，由这两个条件概率可以写出 $P(o, s|\lambda)$ ：

$$\begin{aligned} P(o|s, \lambda) &= P(o_1, o_2, \dots o_T | s, \lambda) \\ &= \prod_{t=1}^T P(o_t | s, \lambda) = \prod_{t=1}^T P(o_t | s_t, \lambda) \\ &= \prod_{t=1}^T B(s_t, o_t) \end{aligned} \tag{4}$$

$$\begin{aligned} P(s|\lambda) &= P(s_1, s_2, \dots s_T | \lambda) \\ &= P(s_1 | \lambda) \prod_{t=2}^T P(s_t | s_{t-1}, \lambda) \\ &= \pi(s_1) \prod_{t=2}^T A(s_{t-1}, s_t) \end{aligned} \tag{5}$$

$$\begin{aligned} P(o, s|\lambda) &= P(o|s, \lambda) P(s|\lambda) \\ &= \pi(s_1) \prod_{t=2}^T A(s_{t-1}, s_t) \prod_{t=1}^T B(s_t, o_t) \\ &= \pi(s_1) B(s_1, o_1) A(s_1, s_2) B(s_2, o_2) \dots A(s_{T-1}, s_T) B(s_T, o_T) \end{aligned} \tag{6}$$

其中对于函数 $A(\cdot, \cdot)$ 、 $B(\cdot, \cdot)$ 、 $\pi(\cdot)$ ，有：

$$A(S_i, S_j) = a_{ij}$$

$$B(S_j, O_k) = b_{jk}$$

$$\pi(S_i) = a_i$$

事实上我们并不知道状态序列是什么，而每一种状态序列都有可能，因此需要对整个状态序列空间求和，把 s "margin out" 掉：

$$\begin{aligned} P(o|\lambda) &= \sum_s P(o, s|\lambda) \\ &= \sum_{s_1, s_2, \dots, s_T} \pi(s_1) B(s_1, o_1) A(s_1, s_2) B(s_2, o_2) \dots A(s_{T-1}, s_T) B(s_T, o_T) \end{aligned} \quad (7)$$

分析这个运算的复杂度：总共需要进行 $O(N^T)$ 规模的求和，每个求和需要做 $O(2T)$ 规模的乘积，总的复杂度是 $O(2TN^T)$ ，这显然是不行的。

1.2.2 第一个问题：sum-product 算法

观察式子 (???)，注意到它做了很多重复计算，例如 $\pi(s_1)$ 跟 s_2 无关，却要在 s_2 上求和时重复地计算，更不要说后面的 $s_3, s_4 \dots$ 了，所以先想办法用乘法分配律先提出共同的因子。先尝试从 s_1 开始，从前到后：

$$\begin{aligned} P(o|\lambda) &= \sum_{s_2, \dots, s_T} \dots \sum_{s_1} \pi(s_1) B(s_1, o_1) A(s_1, s_2) \\ &= \sum_{s_3, \dots, s_T} \dots \sum_{s_2} A(s_2, s_3) B(s_2, o_2) \sum_{s_1} A(s_1, s_2) B(s_1, o_1) \pi(s_1) \\ &= \dots \\ &= \sum_{s_T} B(s_T, o_T) \sum_{s_{T-1}} A(s_{T-1}, s_T) B(s_{T-1}, o_{T-1}) \dots \sum_{s_1} A(s_1, s_2) B(s_1, o_1) \pi(s_1) \end{aligned} \quad (8)$$

似乎复杂度降低了不少，但是这个式子在算法实现的角度上不好算，进一步观察发现它具有一定的周期递推特性，所以接下来通过引入一个新的函数 α 来写出一个递归形式的表示：

$$\alpha_t(s_t) = \begin{cases} B(s_1, o_1) \pi(s_1) & \text{当 } t = 1 \\ B(s_t, o_t) \sum_{s_{t-1}} \dots \sum_{s_{t-2}} \dots \pi(s_1) & \text{当 } 2 \leq t \leq T \end{cases} \quad (9)$$

其实就是把式子 (???) 从后往前“截断”到 $B(s_t, o_t)$ 为止的部分。按照定义， $\alpha_t(s_t)$ 具有如下递推性质：

$$\alpha_t(s_t) = B(s_t, o_t) \sum_{s_{t-1}} A(s_{t-1}, s_t) \alpha_{t-1}(s_{t-1}), \text{ 当 } 2 \leq t \leq T$$

这样从算法实现的角度就可以不断迭代从 $\alpha_1(s_1)$ 一直迭代算到 $\alpha_T(s_T)$ ，然后再最后跨一步算 $p(o|\lambda)$ ：

$$p(o|\lambda) = \sum_{s_T} \alpha_T(s_T)$$

考察上述计算的复杂度， $t = 1$ 时不用计算，从 $t = 2 \rightarrow T$ 共 $T - 1$ 次迭代，每次迭代按照递归公式 (???) 其实是这样的—个矩阵运算（ \circ 表示阿玛达乘积 Hadamard product，即逐位相乘）：

$$\alpha_t = A^T \cdot \alpha_{t-1} \circ b(o_t)$$

其中，列向量 α_t 、 $b(o_t)$ 分别为：

$$\alpha_t = [\alpha_t(s_1), \alpha_t(s_2), \dots, \alpha_t(s_N)]^T$$

$$b(o_t) = [B(s_1, o_t), B(s_2, o_t), \dots, B(s_N, o_t)]^T$$

故而每次迭代的复杂度是 $O(N^2 + N)$ ，最后一次算 $p(o|\lambda)$ 复杂度是 $O(N)$ ，总的复杂度是 $(T - 1) * O(N^2 + N) + O(N) = O(N^2 T)$ ，比暴力算法高不知道哪去了！也可以从 s_T 开始，从后到前：

$$\begin{aligned} P(o|\lambda) &= \sum_{s_1, \dots, s_{T-1}} \dots \sum_{s_T} A(s_{T-1}, s_T) B(s_T, o_T) \\ &= \sum_{s_1, \dots, s_{T-2}} \dots \sum_{s_{T-1}} A(s_{T-2}, s_{T-1}) B(s_{T-1}, o_{T-1}) \sum_{s_T} A(s_{T-1}, s_T) B(s_T, o_T) \\ &= \dots \\ &= \sum_{s_1} \pi(s_1) B(s_1, o_1) \sum_{s_2} A(s_1, s_2) B(s_2, o_2) \dots \sum_{s_T} A(s_{T-1}, s_T) B(s_T, o_T) \end{aligned} \quad (10)$$