

HMM 笔记

郑华滨

2015.5.28

目 录

1	引言	2
2	模型定义	2
2.1	符号	3
2.2	模型假设	3
2.3	模型参数	3
3	三个基本问题	4
4	第一个问题	4
4.1	暴力算法	4
4.2	前向算法	5
4.3	后向算法	8
4.4	小结	9
5	第三个问题	9
5.1	EM 算法框架	9

1 引言

本文介绍了最基本的隐马尔可夫模型 (Hidden Markov Model, HMM) 的模型定义、模型使用算法、模型训练算法。其中，模型使用所用到的前向—后向算法 (forward-backward algorithm)、维特比算法 (Viterbi algorithm)、模型训练所用到的 Expectation Maximization 算法 (EM algorithm) 都会有详细的数学推导。

本文的第二节对 HMM 模型进行了形象化的、形式化的描述，第三节先界定了关于 HMM 模型使用、模型训练的三个问题，第四、五、六节分别解决第一、第三、第二个问题，其中第四节涉及前向算法和后向算法，第五节涉及 Expectation Maximization 算法，第六节涉及前向—后向算法和维特比算法。第七节是全篇总结。

2 模型描述

HMM 模型是用来处理序列化数据 (sequential data) 的一种模型，对于一个观察到的长度为 T 序列： $o = o_1 o_2 o_3 \dots o_T$ ，HMM 假设有一个与之对应的内在的、本质的隐藏序列 $s = s_1 s_2 s_3 \dots s_T$ ，两者之间一一对应，我们把前者称为观察变量构成的观察序列 (observation)，后者称为状态变量构成的状态序列 (state)。状态序列所代表的是我们所没有观察到的本质变化，而观察序列是作为状态序列的结果“发射” (emit) 出来的，当然这只是直观的解释，在实际的数学推导上观察序列只是作为一组隐含变量 (latent variable) 引入原来的图模型而已。

一个经常举的例子是，我们可以把天气变化看作一个观察序列，例如“晴阴晴雨雨阴雷……”，同时假设另外有一个隐藏序列“abbedf……”，与之一一对应。不过，我们只关注这些“abcd”的数学意义，并不一定要有实际的物理意义，当然你高兴的话可以为它们引入宗教意义，例如 a 表示这天是北欧雷神托尔值班，b 表示这天是中国雷神雷震子值班，c 表示今天是中国雨神萧敬腾值班，然后从 abc 到阴晴雨的过程，则是一个根据值班人员心情概率随机决定今天天气的过程……

除了状态序列这个最基本的假设之外，HMM 模型还假设了以下三点：

- 每一个观察变量仅仅依赖于与之对应的那个状态变量，这叫“一一对应性”，用一组条件概率 $P(o_t | s_t)$ 来表示这种依赖关系；
- 每一个状态变量仅仅依赖于它之前的状态变量（一般只依赖于前一个），这叫“马尔可夫性”，其中只依赖于前一个则叫“一阶马尔可夫性”，具体到一阶 (first order) 的情况，用一组条件概率 $P(s_{t+1} | s_t)$ 来表示这种依赖关系；
- 上述的条件概率分布中含有变量 t ，然而它们并不随时间变化，例如不管哪一天轮到托尔值班，打雷的概率都是 99%，而不会今天是 99% 明天是 88%，这叫“时序平稳性”。

下面给出具体的数学描述。

2.1 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: $S = \{S_1, S_2, \dots, S_N\}$, $s_t \in S$, 当 $1 \leq t \leq T$

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

转移概率 (transition probability): $P(s_{t+1}|s_t)$, 当 $1 \leq t \leq T-1$

发射概率 (emission probability): $P(o_t|s_t)$, 当 $1 \leq t \leq T$

初始概率 (initial probability): $P(s_1)$

2.2 模型假设

观察序列和状态序列之间的一一对应性, 即每个观察变量仅仅依赖于与之对应的状态变量:

$$P(o_t|o_1, o_2, \dots, o_{t-1}, o_{t+1}, \dots, o_T, s_1, s_2, \dots, s_T) = P(o_t|s_t) \quad (2.1)$$

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:

$$P(s_t|s_1, s_2, \dots, s_{t-1}) = P(s_t|s_{t-1}), \quad \text{当 } 2 \leq t \leq T \quad (2.2)$$

时序平稳性, 即转移概率分布和发射概率分布不随时间变化:

$$P(s_{t_1}|s_{t_1-1}) = P(s_{t_2}|s_{t_2-1}), \quad \text{当 } s_{t_1} = s_{t_2}, s_{t_1-1} = s_{t_2-1}, 2 \leq t_1, t_2 \leq T \quad (2.3)$$

2.3 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 模型参数主要是转移概率、发射概率以及初始概率:

$$a_{ij} = P(s_t = S_j | s_{t-1} = S_i) \quad (2.4a)$$

$$b_{jk} = P(o_t = O_k | s_t = S_j) \quad (2.4b)$$

$$\pi_i = P(s_1 = S_i) \quad (2.4c)$$

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:

$$\lambda = (A, B, \pi) \quad (2.5)$$

这就表示 HMM 模型的所有参数。

3 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 $\lambda_0 = \arg \max_{\lambda} P(o|\lambda)$ ？

第二个问题属于模型使用问题，例如在词性标注 (part-of-speech tagging) 问题中，给定一个句子，也就是一个词序列，我们关心应该给每个词标上什么样的词性，如动词、名词、形容词，如果我们把词序列作为观察序列，词性序列作为状态序列，那就刚好对应到第二个问题。第三个问题属于模型训练问题，或者说参数优化问题 (parameter optimization)。第一个问题在我所知的范围内，并没有作为一个模型使用问题而出现，而是作为在解决第三个问题的过程中需要解决的子问题，可能在其他一些任务中（例如语音识别、笔迹识别就用到了 HMM）就是作为模型使用问题存在的。

参考文献??? 是按照上面列出的顺序提出和解决这三个问题的，但是由于第一和第三个问题联系紧密，而第二个问题和它们关系不大，所以本文将适当调整讲述顺序。

4 第一个问题

4.1 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 $P(o, s|\lambda)$ ：

$$P(o|s, \lambda) = \prod_{t=1}^T P(o_t|s, \lambda) = \prod_{t=1}^T P(o_t|s_t, \lambda) = \prod_{t=1}^T B(s_t, o_t) \quad (4.1)$$

$$P(s|\lambda) = P(s_1|\lambda) \prod_{t=2}^T P(s_t|s_{t-1}, \lambda) = \pi(s_1) \prod_{t=2}^T A(s_{t-1}, s_t) \quad (4.2)$$

$$P(o, s|\lambda) = P(o|s, \lambda)P(s|\lambda) = \pi(s_1)B(s_1, o_1) \prod_{t=2}^T A(s_{t-1}, s_t)B(s_t, o_t) \quad (4.3)$$

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

$$A(S_i, S_j) = a_{ij} \quad (4.4a)$$

$$B(S_j, O_k) = b_{jk} \quad (4.4b)$$

$$\pi(S_i) = a_i \quad (4.4c)$$

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 $P(o, s|\lambda)$ 中的 s "margin out" 掉，即 $P(o|\lambda) = \sum_s P(o, s|\lambda)$ ，展开后有：

$$P(o|\lambda) = \sum_{s_1, s_2, \dots, s_T} \pi(s_1) B(s_1, o_1) \prod_{t=2}^T A(s_{t-1}, s_t) B(s_t, o_t) \quad (4.5)$$

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 $O(T)$ 规模的乘积，总的复杂度是 $O(TN^T)$ ，这显然是不行的。

4.2 前向算法

首先注意到，按照乘法分配律有：

$$\sum_{x=1}^N f(x, y) g(y) = g(y) \sum_{x=1}^N f(x, y) \quad (4.6)$$

观察式子(4.5)发现有类似的结构，同时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：

$$P(o|\lambda) = \sum_{s_T} B(s_T, o_T) \prod_{t=T-1}^1 \sum_{s_t} A(s_t, s_{t+1}) B(s_t, o_t) \pi(s_1) \quad (4.7)$$

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组新的函数 α_t 来写出一个递归形式的表示：

$$\alpha_t(s_t) = \begin{cases} B(s_1, o_1) \pi(s_1) & \text{当 } t = 1 \\ B(s_t, o_t) \prod_{u=t-1}^1 \sum_{s_u} A(s_u, s_{u+1}) B(s_u, o_u) \pi(s_1) & \text{当 } 2 \leq t \leq T \end{cases} \quad (4.8)$$

其实就是把式子(4.7)从后往前“截断”到 $B(s_t, o_t)$ 左边的部分。按照定义， $\alpha_t(s_t)$ 具有如下递推性质：

$$\alpha_t(s_t) = B(s_t, o_t) \sum_{s_{t-1}} A(s_{t-1}, s_t) \alpha_{t-1}(s_{t-1}) \quad \text{当 } 2 \leq t \leq T \quad (4.9)$$

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 $p(o|\lambda)$ ：

$$p(o|\lambda) = \sum_{s_T} \alpha_T(s_T) \quad (4.10)$$

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(4.9)其实是这样的矩阵运算：

$$\alpha_t = A^T \cdot \alpha_{t-1} \circ b(o_t) \quad (4.11)$$

其中， \circ 表示阿玛达乘积 (Hadamard product)，即逐位相乘，列向量 α_t 、 $b(o_t)$ 分别为：

$$\alpha_t = [\alpha_t(s_1), \alpha_t(s_2), \dots, \alpha_t(s_N)]^T \quad (4.12a)$$

$$b(o_t) = [B(s_1, o_t), B(s_2, o_t), \dots, B(s_N, o_t)]^T \quad (4.12b)$$

故而每次迭代的复杂度是 $O(N^2)$ ，迭代 $T - 1$ 次，最后一次算 $p(o|\lambda)$ 复杂度是 $O(N)$ ，总的复杂度是 $O(TN^2)$ ，比暴力算法的 $O(TN^T)$ 高不知道哪去了

刚刚是把式(4.7)从后往前“截断”到 $B(s_t, o_t)$ 左边，但其实也可以“截断”到 $B(s_t, o_t)$ 右边。定义一个新的函数 α'_t ：

$$\alpha'_t(s_t) = \begin{cases} \pi(s_1) & \text{当 } t = 1 \\ \prod_{u=t-1}^1 \sum_{s_u} A(s_u, s_{u+1}) B(s_u, o_u) \pi(s_1) & \text{当 } 2 \leq t \leq T \end{cases} \quad (4.13)$$

其递推性质为：

$$\alpha'_t(s_t) = \sum_{s_{t-1}} A(s_{t-1}, s_t) B(s_{t-1}, o_{t-1}) \alpha'_{t-1}(s_{t-1}) \quad \text{当 } 2 \leq t \leq T \quad (4.14)$$

从 $\alpha'_1(s_1)$ 一直迭代算到 $\alpha'_T(s_T)$ ，然后再最后跨一步算 $p(o|\lambda)$ ：

$$p(o|\lambda) = \sum_{s_T} B(s_T, o_T) \alpha'_T(s_T) \quad (4.15)$$

式(4.13)、(4.14)、(4.15)与式(4.8)、(4.9)、(4.10)表示的计算过程是等价的，复杂度同样也是 $O(TN^2)$ 。

推导到这里，可以问一个问题： α_t 和 α'_t 是否具有某种概率意义？后面我们会发现，这个问题非常重要。

先从 α_t 开始, 观察其定义(4.8), 发现:

$$\begin{aligned}\alpha_1(s_1) &= B(s_1, o_1)\pi(s_1) \\ &= P(o_1|s_1, \lambda)P(s_1|\lambda) \\ &= P(o_1, s_1|\lambda)\end{aligned}\tag{4.16}$$

$$\begin{aligned}\alpha_2(s_2) &= B(s_2, o_2) \sum_{s_1} A(s_1, s_2)\alpha_1(s_1) \\ &= P(o_2|s_2, \lambda) \sum_{s_1} P(s_2|s_1, \lambda)P(o_1, s_1|\lambda) \\ &= P(o_2|s_2, \lambda) \sum_{s_1} P(o_1, s_1, s_2|\lambda) \\ &= P(o_2|s_2, \lambda)P(o_1, s_2|\lambda) \\ &= P(o_1, o_2, s_2|\lambda)\end{aligned}\tag{4.17}$$

$$\begin{aligned}\alpha_3(s_3) &= \dots \\ &= P(o_1, o_2, o_3, s_3|\lambda)\end{aligned}\tag{4.18}$$

于是猜想:

$$\alpha_t(s_t) = P(o_1, o_2, \dots, o_t, s_t|\lambda) \quad \text{当 } 1 \leq t \leq T \tag{4.19}$$

可用数学归纳法证明。假设 $\alpha_t(s_{t-1}) = P(o_1, o_2, \dots, o_{t-1}, s_{t-1}|\lambda)$ 成立, 则有:

$$\begin{aligned}\alpha_t(s_t) &= B(s_t, o_t) \sum_{s_{t-1}} A(s_{t-1}, s_t)\alpha_{t-1}(s_{t-1}) \\ &= P(o_t|s_t, \lambda) \sum_{s_{t-1}} P(s_t|s_{t-1}, \lambda)P(o_1, o_2, \dots, o_{t-1}, s_{t-1}|\lambda) \\ &= P(o_t|s_t, \lambda) \sum_{s_{t-1}} P(o_1, o_2, \dots, o_{t-1}, s_{t-1}, s_t|\lambda) \\ &= P(o_t|s_t, \lambda)P(o_1, o_2, \dots, o_{t-1}, s_t|\lambda) \\ &= P(o_1, o_2, \dots, o_t, s_t|\lambda)\end{aligned}\tag{4.20}$$

再结合初始条件(4.16), 可证明(4.19)。类似可证:

$$\alpha'_t(s_t) = P(o_1, o_2, \dots, o_{t-1}, s_t|\lambda) \quad \text{当 } 1 \leq t \leq T \tag{4.21}$$

当然也可以直接利用结论(4.19)证明:

$$\begin{aligned}\alpha'_t(s_t) &= \frac{\alpha_t(s_t)}{B(s_t, o_t)} \\ &= \frac{P(o_1, o_2, \dots, o_t, s_t|\lambda)}{P(o_t|s_t, \lambda)} \\ &= \frac{P(o_1, o_2, \dots, o_{t-1}, s_t, \lambda)P(o_t|s_t, \lambda)P(s_t|\lambda)}{P(o_t|s_t, \lambda)} \\ &= P(o_1, o_2, \dots, o_{t-1}, s_t|\lambda)\end{aligned}\tag{4.22}$$

式子(4.19)和(4.21)就是 α_t 和 α'_t 的概率意义。参考文献??? 中的推导则是先定义了 α_t 的概率意义，然后再推出这组函数之间的递推关系的。本文采取了不同的推导路径，为的是让 α_t 的引入显得不那么 magic。

4.3 后向算法

上一小节是从 s_1 开始重排，称之为前向算法 (forward algorithm)，但也可以从 s_T 开始，从后到前，称之为后向算法 (backward algorithm)：

$$P(o|\lambda) \sum_{s_1} \pi(s_1) B(s_1, o_1) \prod_{t=2}^T \sum_{s_t} A(s_{t-1}, s_t) B(s_t, o_t) \quad (4.23)$$

与前向算法相同，可以“截断”到 $B(s_t, o_t)$ 左边或者右边，分别得到一组函数 β_t ：

$$\beta_t(s_t) = \begin{cases} B(s_T, o_T) & \text{当 } t = T \\ B(s_t, o_t) \prod_{u=t+1}^T \sum_{s_u} A(s_{u-1}, s_u) B(s_u, o_u) & \text{当 } 1 \leq t \leq T-1 \end{cases} \quad (4.24)$$

$$\beta_t(s_t) = B(s_t, o_t) \sum_{s_{t+1}} A(s_t, s_{t+1}) \beta_t(s_{t+1}) \quad \text{当 } 1 \leq t \leq T-1 \quad (4.25)$$

$$P(o|\lambda) = \sum_{s_1} \pi(s_1) \beta_t(s_1) \quad (4.26)$$

$$\beta_t(s_t) = P(o_t, o_{t+1}, \dots, o_T | s_t, \lambda) \quad \text{当 } 1 \leq t \leq T \quad (4.27)$$

和一组函数 β'_t ：

$$\beta'_t(s_t) = \begin{cases} 1 & \text{当 } t = T \\ \prod_{u=t+1}^T \sum_{s_u} A(s_{u-1}, s_u) B(s_u, o_u) & \text{当 } 1 \leq t \leq T-1 \end{cases} \quad (4.28)$$

$$\beta'_t(s_t) = \sum_{s_{t+1}} A(s_t, s_{t+1}) B(s_{t+1}, o_{t+1}) \beta'_t(s_{t+1}) \quad \text{当 } 1 \leq t \leq T-1 \quad (4.29)$$

$$P(o|\lambda) = \sum_{s_1} \pi(s_1) B(s_1, o_1) \beta'_t(s_1) \quad (4.30)$$

$$\beta'_t(s_t) = P(o_{t+1}, o_{t+2}, \dots, o_T | s_t, \lambda) \quad \text{当 } 1 \leq t \leq T \quad (4.31)$$

这些式子的推导过程与上一小节的推导过程很相似，推导出来的形式也是很相似的，故略去不写。

4.4 小结

给定一组参数模型 $\lambda = (A, B, \pi)$ 和一个观察序列 $o = o_1 o_2 \dots o_T$ ，要计算 $P(o|\lambda)$ ，第一个直接的思路是把对 $P(o, s|\lambda)$ 进行 “margin out”，但是发现直接按公式计算复杂度太高。第二个尝试是利用乘法分配律减少重复计算，由此导出了前向和后向算法，而这两者中又都有“截断”到 $B(s_t, o_t)$ 左边和右边两种做法。希望这样的表述能够让读者更容易把握推导的整体脉络。

单就解决第一个问题而言，前向和后向算法用一个就可以，但是在接下来解决第三个问题即模型训练问题的过程中，需要同时用到 α_t 和 β_t ，综合起来正是所谓的前向-后向算法 (forward-backward algorithm)。

同样地，单就第一个问题，无论是前向还是后向，“截断”到 $B(s_t, o_t)$ 左边或者右边都可以，但是我们还是把两种可能性都列了出来，目的是为了第三个问题中更为清晰的表述。

5 第三个问题

要求最优化参数 $\lambda_0 = \arg \max_{\lambda} P(o|\lambda)$ ，这个问题本身其实是属于用最大似然法优化参数，至于用贝叶斯方法怎么优化参数，并不在本文的讨论范围内。最大似然法一般不直接优化似然函数，而是优化对数似然函数 $\log P(o|\lambda)$ ，然后对参数 λ 求导，令等于零，求解……但这在 HMM 上做不了，因为式子(4.5)很难应用这种方法。事实上，目前并没有 HMM 参数优化问题的解析解。

HMM 的参数训练一般是用 Baum-Welch 算法，只能保证得到局部最优的参数。它其实就是通用的 EM 算法框架在 HMM 参数训练问题上的具体使用。接下来先介绍 EM 算法，再具体介绍 Baum-Welch 算法。

5.1 EM 算法框架

Expectation Maximization(EM) 算法是一个通用的算法框架，只有应用在像 HMM 参数训练这样的具体问题上时