HMM的Baum-Welch算法和Viterbi算法公式推导细节

标签： 统计学习方法 HMM 维特比算法 Baum-welch

2016-03-23 21:34 5428人阅读 评论(2) 收藏 举报

分类： 机器学习（29）

目录(?) [+]

前言

在上一篇博文中，我简单地介绍了隐马尔科夫模型HMM，并且重点介绍了HMM的三个问题中的第一个，即概率计算问题。首先回顾一下这三个问题都是什么以及解决每个问题的主流**算法**：

1. 概率计算问题即模型评价问题——前向算法和后向算法
2. 学习问题即参数估计问题——Baum-Welch算法
3. 预测问题即解码问题——Viterbi算法

在上一篇概率计算问题的最后，我列出了几个用前向概率和后向概率表示的一些有意义的概率值和期望的计算，它们的直接意义就是用于表示学习问题和预测问题公式推导中复杂的中间结果的表示。所以，要想彻底搞懂Baum-Welch算法和Viterbi算法算法，就必须清楚地明白这些概率和期望到底是怎么计算出来的。

然而，本博文并不打算将这两个算法全部的公式推导写下来，那太繁杂了。如果想窥探这两个算法的细节，直接看李航博士的《统计学习方法》对应的内容就好了。本文只是将这两个算法推导中的一些隐晦的地方做一个通俗的解释，希望能给像我一样数学功底一般的朋友带来帮助。

Baum-Welch算法

Baum-Welch算法是为了解决HMM的参数估计问题而提出的，而且是没有标注也就是HMM的状态序列未知的参数估计问题。具体来说，就是已知观测序列$O=(o_1,o_2,\dots,o_T)$，估计模型参数$\lambda=(A,B,\pi)$，使得在该模型下观测序列概率$P(O|\lambda)$最大。由于状态序列未知，因此这可以看做是一个含有隐变量的参数估计问题，解决这一问题的经典算法就是EM算法。Baum-Welch算法就是EM算法在隐马尔科夫模型学习中的具体体现。下面简单叙述一下该算法。

首先按照EM算法，我们需要先写出Q函数。Q函数是完全数据的对数似然函数关于给定模型参数和观测变量的前提下对隐变量的条件概率分布的期望。如下：

$$Q(\lambda,\bar{\lambda})=\sum_I \log P(O,I|\lambda)P(I|O,\bar{\lambda})$$

我们写出Q函数之后后面就要对它进行极大化，也就是说EM算法的M步骤。既然是最大化，那么只要保证不影响最终的结果，对Q函数进行对于最大化来说没有影响的常数因子乘除是可以的。我们注意到Q函数的后部分

$$P(I|O,\bar{\lambda}) = \frac{P(O,I|\bar{\lambda})}{P(O|\bar{\lambda})}$$

而$P(O|\lambda)$便是概率计算问题中我们解决的问题，对于固定的模型参数来说它是一个常量，因此我们为了后边计算方便可以在上面原先的Q函数的基础上乘以它，使得Q函数成为：

$$Q(\lambda,\bar{\lambda}) = \sum_{I} \log P(O,I|\lambda)P(O,I|,\bar{\lambda})$$

为什么要这么做呢？这是为了后面将概率计算问题中有意义的一些概率计算公式直接套进去。

又因为完全数据可以写成这样：

$$\log P(O,I|\lambda) = \pi_{i_1} b_1(o_1) a_{i_1 i_2} b_2(o_2) \dots a_{i_{T-1} i_T} b_T(o_T)$$

于是Q函数可以写成：

$$Q(\lambda,\bar{\lambda}) = \sum_{I} \log \pi_{i_1} P(O,I|,\bar{\lambda}) + + \sum_{I} \left(\sum_{t=1}^{T-1} \log a_{i_t i_{t+1}}\right) P(O,I|,\bar{\lambda})$$

$$+ \sum_{I} \left(\sum_{t=1}^{T} \log b_{i_t}(o_t)\right) P(O,I|,\bar{\lambda})$$

此时我们看到待估计的参数刚好分别出现在三个项中，所以只需对各个项分别极大化。然后直接极大化我们无法对公式进行细致描述，因此需要将以上Q函数形式修改一下，变成下面这样：

$$Q(\lambda,\bar{\lambda}) = \sum_{i=1}^{N} \log \pi_i P(O,i_1 = i|,\bar{\lambda}) + \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t=1}^{T-1} \log a_{ij} P(O,i_t = i, i_{t+1} = j|,\bar{\lambda})$$

$$+ \sum_{j=1}^{N}\sum_{t=1}^{T} \log b_j(o_t) P(O,i_t = j|,\bar{\lambda})$$

可以看到，我们将三项中分别的对I的求和进行了划分。由于隐变量$I = (i_1, i_2, \dots, i_T)$。原来的求和需要遍历所有$I$的取值，然后进行求和，然而这基本是不可能完成的任务。改写后，我们将遍历的空间进行了划分，同时很好地将$P(O,I|,\bar{\lambda})$部分改写后也融入到求和其中。比如第一项，对I的遍历等价于先固定状态i_1，使其分别取值所有可能的状态（共有N个可取的离散状态），而$i_2, \dots, i_T$仍然像原来一样随便取值。这样，就把I空间划分成了N个更小的空间。然后再把这N个空间的结果相加，等价于原来对空间I进行遍历。

而且，改写之后，$P(O,I|\bar{\lambda})$部分变的可以表示了。如果对Q函数的三项分别求极大，在计算后会发现，最后的结果可以用前一篇博文末尾的一些有意义的概率来表示。这也就是之前对Q函数进行修改的原因。

OK，Baum-welch算法就介绍到这里。

Viterbi算法

Viterbi算法应用于HMM的识别问题，也称解码问题。它通过应用了动态规划的思想避免了复杂度很高的运算，为识别时效性提供了强有力的支持。这个算法并不难理解，这里只是对其一个细节进行阐述。

Viterbi算法实际上解决$P(I|O,\lambda)$最大化的问题，给定观测序列求其最可能对应的状态序列。算法首先需要导入两个变量δ和ψ。δ是在时刻t状态为i的所有单个路径$(i_1,i_2,\ldots,i_t)$中概率的最大值：

$$\delta_t(i)=\max_{i_1,i_2\ldots,i_{t-1}} P(i_t=i,i_{t-1},\ldots,i_1,o_t,\ldots,o_1|\lambda)$$

由定义可得变量δ的递推公式：

$$\delta_{t+1}(i)=\max_{i_1,i_2\ldots,i_t} P(i_{t+1}=i,i_t,\ldots,i_1,o_{t+1},\ldots,o_1|\lambda)$$

$$=\max_{1\le j\le N}[\delta_t(j)a_{ji}]b_i(o_{t+1})$$

算法在设定初始值$\delta_1(i)=\pi_i b_i(o_1)$之后就不断迭代，终止情况是：

$$P^*=\max_{1\le j\le N}\delta_T(i)$$

算法的主体就是这样，那么现在问题来了。识别问题我们要解决$P(I|O,\lambda)$最大的问题，到这里Viterbi算法怎么成了$P(I,O|\lambda)$？

个人觉得，还是应为$P(I,O|\lambda)=P(I|O,\lambda)P(O|\lambda)$，而$P(O|\lambda)$相对最大化问题而言是个常数，可以省去。但是为什么要省去呢？这是因为$\delta_{t+1}(i)$的迭代过程中$\delta_t(i)$要和a_{ji}相乘，而a_{ji}是转移概率，表示为：

$$a_{ji}=P(i_{t+1}=q_i|i_t=q_j)$$

再经过齐次马尔科夫假设，可以扩展为：

$$a_{ji}=P(i_{t+1}=q_i|i_t=q_j,o_1,o_2,\ldots,o_t)$$

这样其才能和$\delta_t(j)$相乘。仅此而已。

上一篇 隐马尔科夫模型HMM的前向算法和后向算法