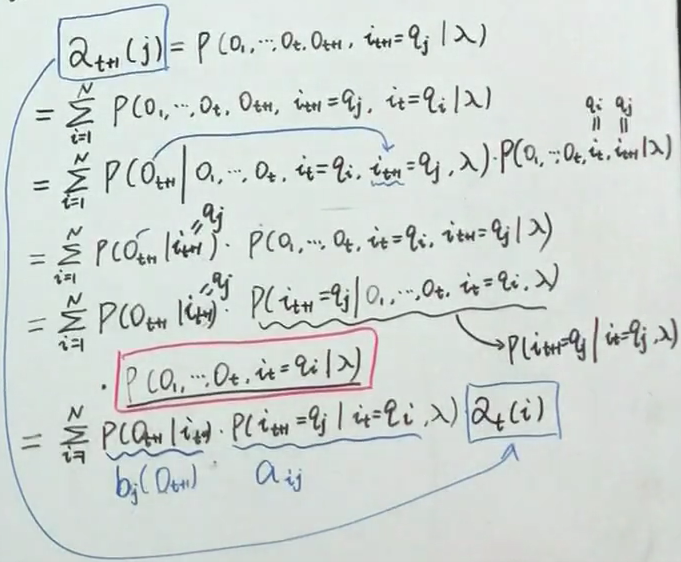
1. （必填）自己提出的问题的理解（罗列全部）：
2. **提出的问题1：**

P198，10.16递推式是如何推出的？

讨论后的理解：



1. **提出的问题2：**

P201 公式10.19对初始状态的初始化，为什么的初始值为 1？

讨论后的理解：

假设掷骰子作为实验对象，可以这么理解，已知第t次掷骰子所用的骰子是，那么它表示的就是从t+1次到第T次的看到的点数为：的概率。已知最后一次所用的骰子为，那么第T次之后，为任意值的概率，故而为1。

1. （必填）别人提出的问题的理解（选择几个问题罗列，并给出理解）：

1. **问题3**：

什么样的问题需要HMM模型？

自己的理解：

使用HMM模型时我们的问题一般有这两个特征：

１）我们的问题是基于序列的，比如时间序列，或者状态序列。

２）我们的问题中有两类数据，一类序列数据是可以观测到的，即观测序列；而另一类数据是不能观察到的，即隐藏状态序列，简称状态序列。

有了这两个特征，那么这个问题一般可以用HMM模型来尝试解决。这样的问题在实际生活中是很多的。

2. **问题4：**

p202页中10.22式是如何推导得到的

自己的理解：

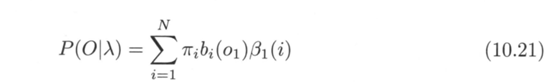
统一的概率模型为



这是通过前向概率和后向概率的定义得到的，并不是推导，特别的当t=1和t=T-1时分别为式(10.17)和式(10.21)。

即为





1. **问题5：**

什么是马尔克夫过程？

自己的理解：

马尔可夫过程，由状态集合，初始状态和状态转移矩阵组成，以天气为例子，状态集合为，晴天、阴天和下雨。初始状态为，晴天，状态转移矩阵，表示每个状态之间的迁移概率，用马尔可夫过程模型，已经可以观察和model各种状态序列，但如果想要切开表面，发现事物的本质，就需要隐马尔可夫模型(Hidden Markov Model)

三、（必填）读书计划

1、本周完成的内容章节：统计学习方法（第十章：10.1-10.2）

2、下周计划：统计学习方法（第十章剩余部分）

四、（选做）读书摘要及理解或伪代码的具体实现（读书摘要、伪代码的具体实现代码等可以写到这个部分）

1、读书摘要及理解（选做）

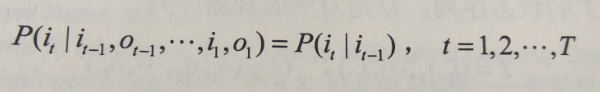
**HMM模型的五个基本元素**

HMM是个五元组 λ ＝（ S, O , π ，A，B）

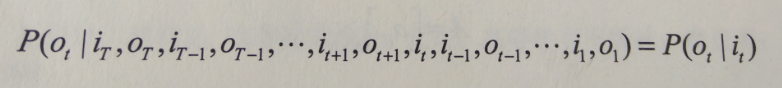
S:状态值集合，O：观察值集合，π：初始化概率，A：状态转移概率矩阵，B：给定状态下，观察值概率矩阵

**HMM模型的两个假设**

1）齐次马尔科夫假设：齐次马尔科夫假设，通俗地说就是 HMM 的任一时刻 t 的某一状态只依赖于其前一时刻的状态，与其它时刻的状态及观测无关，也与时刻 t 无关。



2）观测独立假设：观测独立性假设，是任一时刻的观测只依赖于该时刻的马尔科夫链的状态，与其他观测及状态无关。



**HMM模型的三个基本问题**

1） 评估观察序列概率。即给定模型λ=(A,B,Π)和观测序列O={o1,o2,...oT}，计算在模型λ下观测序列O出现的概率P(O|λ)，这个问题的求解需要用到前向后向算法，这个问题是HMM模型三个问题中最简单的。

2）模型参数学习问题。即给定观测序列O={o1,o2,...oT}，估计模型λ=(A,B,Π)的参数，使该模型下观测序列的条件概率P(O|λ)最大。这个问题的求解需要用到基于EM算法的鲍姆-韦尔奇算法，这个问题是HMM模型三个问题中最复杂的。

3）预测问题，也称为解码问题。即给定模型λ=(A,B,Π)和观测序列O={o1,o2,...oT}，求给定观测序列条件下，最可能出现的对应的状态序列，这个问题的求解需要用到基于动态规划的维特比算法，这个问题是HMM模型三个问题中复杂度居中的算法。

2、代码实现（选做）

import numpy as np

def Forward(trainsition\_probability,emission\_probability,pi,obs\_seq):

"""

:param trainsition\_probability:trainsition\_probability是状态转移矩阵

:param emission\_probability: emission\_probability是发射矩阵

:param pi: pi是初始状态概率

:param obs\_seq: obs\_seq是观察状态序列

:return: 返回结果

"""

trainsition\_probability = np.array(trainsition\_probability)

emission\_probability = np.array(emission\_probability)

print emission\_probability[:,0]

pi = np.array(pi)

Row = np.array(trainsition\_probability).shape[0]

F = np.zeros((Row,Col)) #最后要返回的就是F，就是我们公式中的alpha

F[:,0] = pi \* np.transpose(emission\_probability[:,obs\_seq[0]]) #这是初始化求第一列,就是初始的概率\*各自的发射概率

print F[:,0]

for t in range(1,len(obs\_seq)): #这里相当于填矩阵的元素值

for n in range(Row): #n是代表隐藏状态的

F[n,t] = np.dot(F[:,t-1],trainsition\_probability[:,n])\*emission\_probability[n,obs\_seq[t]] #对应于公式,前面是对应相乘

return F

def Backward(trainsition\_probability,emission\_probability,pi,obs\_seq):

"""

:param trainsition\_probability:trainsition\_probability是状态转移矩阵

:param emission\_probability: emission\_probability是发射矩阵

:param pi: pi是初始状态概率

:param obs\_seq: obs\_seq是观察状态序列

:return: 返回结果

"""

trainsition\_probability = np.array(trainsition\_probability)

emission\_probability = np.array(emission\_probability)

pi = np.array(pi) #要进行矩阵运算，先变为array类型

Row = trainsition\_probability.shape[0]

Col = len(obs\_seq)

F = np.zeros((Row,Col))

F[:,(Col-1):] = 1 #最后的每一个元素赋值为1

for t in reversed(range(Col-1)):

for n in range(Row):

F[n,t] = np.sum(F[:,t+1]\*trainsition\_probability[n,:]\*emission\_probability[:,obs\_seq[t+1]])

return F

if \_\_name\_\_ == '\_\_main\_\_':

trainsition\_probability = [[0.7,0.3],[0.4,0.6]]

emission\_probability = [[0.5,0.4,0.1],[0.1,0.3,0.6]]

pi = [0.6,0.4]

#然后下面先得到前向算法,在A,B,pi参数已知的前提下，求出特定观察序列的概率是多少?

obs\_seq = [0,1]

Row = np.array(trainsition\_probability).shape[0]

Col = len(obs\_seq)

F = Forward(trainsition\_probability,emission\_probability,pi,obs\_seq) #得到前向算法的结果

F\_backward = Backward(trainsition\_probability,emission\_probability,pi,obs\_seq) #得到后向算法的结果

res\_forward = 0

for i in range(Row): #将最后一列相加就得到了我们最终的结果

res\_forward+=F[i][Col-1] #求和于最后一列

emission\_probability = np.array(emission\_probability)

#下面是得到后向算法的结果

res\_backword = 0

res\_backward = np.sum(pi\*F\_backward[:,0]\*emission\_probability[:,obs\_seq[0]]) #一定要乘以发射的那一列

print "res\_backward = {}".format(res\_backward)

print "res\_forward = {}".format(res\_forward)