隐马尔可夫模型

课程作业要求

本次作业是课程大作业。

请阅读以下整个文档，并补充完成整个文档。注意每一章节的问题部分，有对应分值，请完成。本次作业占课程分数 70%。

注意，本作业涉及到一定的概率论知识，要求对概率、条件概率、联合概率、贝叶斯公式、全概率公式较为熟悉。如果这方面数学知识不是很了解，虽不影响本作业的完成，但难度会较大，可考虑选择另一份作业“图的应用”完成。本作业提到了隐马尔可夫过程，但对马尔可夫过程的严格数学定义并不要求。

需要说明的是，本次是课程大作业，因此每一个问题请不要只写一个答案，**请尽量给出详细的解答过程**，避免因答案错误而一分不得。

完成文档后请输出为 pdf 文档，并按照如下方式重命名：**姓名-学号-隐马尔可夫模型.pdf**

概述

隐马尔科夫模型（Hidden Markov Model，以下简称 HMM）是比较经典的机器学习模型了，它在语言识别，自然语言处理，模式识别等领域得到广泛的应用。本文档是对课本 15 章动态规划思考题 15-7 的扩展。参考阅读：1. 课程网站给出的英文文献；2. 李航《统计学系方法》。

隐马尔可夫模型主要是研究现实生活中这样的一类问题：一个事件，是由一系列状态决定，状态在每一个时刻t 会以某种概率进行转换（这是马尔可夫模型）。然而，我们无法直接测量状态，只能观测到某状态下所生成的事件。这里状态到事件是第二个概率模型。这种状态无法观测，需要从状态产生的可观测事件来反推状态的，则是隐马尔可夫模型，即“状态转变的马尔可夫模型被隐藏在可观测事件”背后。举一个例子：假设笔记本的电池状态有“良好”、“老化”、“损毁”三种状态，则在人一个时刻t，电池必然处于一种状态，如“良好”，而在$t+1$ 时刻，电池会从“良好”状态以某种概率模型跳转到“良好”、“老化”、“损毁”中的一种状态。这是马尔可夫过程。然而我们很难直接检测电池的状态，我们直观上能看到的是笔记本的续航时间。比如，当电池处于“良好”状态的时候，有 0.8 的可能续航时间为 10 小时，0.2 的可能续航时间为 5 小时，而在电池处于“老化”状态下，只有 0.4 的可能续航时间是 10 小时，0.6 的可能续航时间为 5 小时。此时，我们根据观测到的续航时间的序列，以及两个概率（电池状态跳转概率与电池状态与续航时间之间的概率），则可以通过隐马尔可夫模型建模，从而求出电池的状态序列。

总结一下，我们来看看什么样的问题解决可以用 HMM 模型。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。比如：我现在在打字写博客，我在键盘上敲出来的一系列字符就是观测序列，而我实际想写的一段话就是隐藏序列，输入法的任务就是从敲入的一系列字符尽可能的猜测我要写的一段话，并把最可能的词语放在最前面让我选择，这就可以看做一个 HMM 模型了。再举一个，我在和你说话，我发出的一串连续的声音就是观测序列，而我实际要表达的一段话就是状态序列，你大脑的任务，就是从这一串连续的声音中判断出我最可能要表达的话的内容。以下是 HMM 的一些应用领域：

- 语音识别、中文分词或光学字符识别
- 机器翻译
- 生物信息学和基因组学
  - 基因组序列中蛋白质编码区域的预测
  - 对于相互关联的 DNA 或蛋白质族的建模
  - 从基本结构中预测第二结构元素
  - 通信中的译码过程
  - 地图匹配算法
- *还有更多…*

从这些例子中，我们可以发现，HMM 模型可以无处不在。但是上面的描述还不精确，下面我们用精确的数学符号来表述我们的 HMM 模型。

HMM 模型的定义

对于 HMM 模型，首先我们假设Q是所有可能的隐藏状态的集合，VV 是所有可能的观测状态的集合，即：

$$Q = q_1, q_2, ..., q_N, \quad V = v_1, v_2, ... v_M$$

其中，N是可能的隐藏状态数，M是所有的可能的观察状态数。

对于一个长度为T的序列，I对应的状态序列,O是对应的观察序列，即：

$$I = i_1, i_2, ..., i_T, \quad O = o_1, o_2, ... o_T$$

其中，任意一个隐藏状态$i_t \in Q$,任意一个观察状态$o_t \in V$。

HMM 模型做了两个很重要的假设如下：

1. 齐次马尔科夫链假设。即任意时刻的隐藏状态只依赖于它前一个隐藏状态。当然这样假设有点极端，因为很多时候我们的某一个隐藏状态不仅仅只依赖于前一个隐藏状态，可能是前两个或者是前三个。但是这样假设的好处就是模型简单，便于求解。如果在时刻t的隐藏状态是$i_t = q_i$，在时刻$t+1$ 的隐藏状态是$i_{t+1} = q_j$， 则从时刻t到时刻$t+1$ 的 HMM 状态转移概率a_{ij}可以表示为：

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

这样a_{ij}可以组成马尔科夫链的状态转移矩阵A:

$$A = [aij]_{N \times N}$$

1. 观测独立性假设。即任意时刻的观察状态只仅仅依赖于当前时刻的隐藏状态，这也是一个为了简化模型的假设。如果在时刻tt 的隐藏状态是$i_t = q_j$，而对应的观察状态为$o_t = v_k$，则该时刻观察状态v_k在隐藏状态q_j下生成的概率为$b_j(k)$，满足：

$$b_j(k) = P(o_t = v_k | i_t = q_j)$$

这样$b_j(k)$可以组成观测状态生成的概率矩阵B:

$$B = [b_j(k)]_{N \times M}$$

除此之外，我们需要一组在时刻$t = 1$ 的隐藏状态概率分布Π:

$$\Pi = [\pi(i)]_N$$

其中$\pi(i) = P(i_1 = q_i)$

一个 HMM 模型，可以由隐藏状态初始概率分布Π，状态转移概率矩阵A和观测状态概率矩阵B决定。Π, A决定状态序列，B决定观测序列。因此，HMM 模型可以由一个三元组λ表示如下：

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

一个 HMM 模型实例

下面我们用一个简单的实例来描述上面抽象出的 HMM 模型。

想象一个乡村诊所。村民有着非常理想化的特性，要么健康要么发烧。他们只有问诊所的医生的才能知道是否发烧。 聪明的医生通过询问病人的感觉诊断他们是否发烧。村民只回答他们感觉正常、头晕或冷。

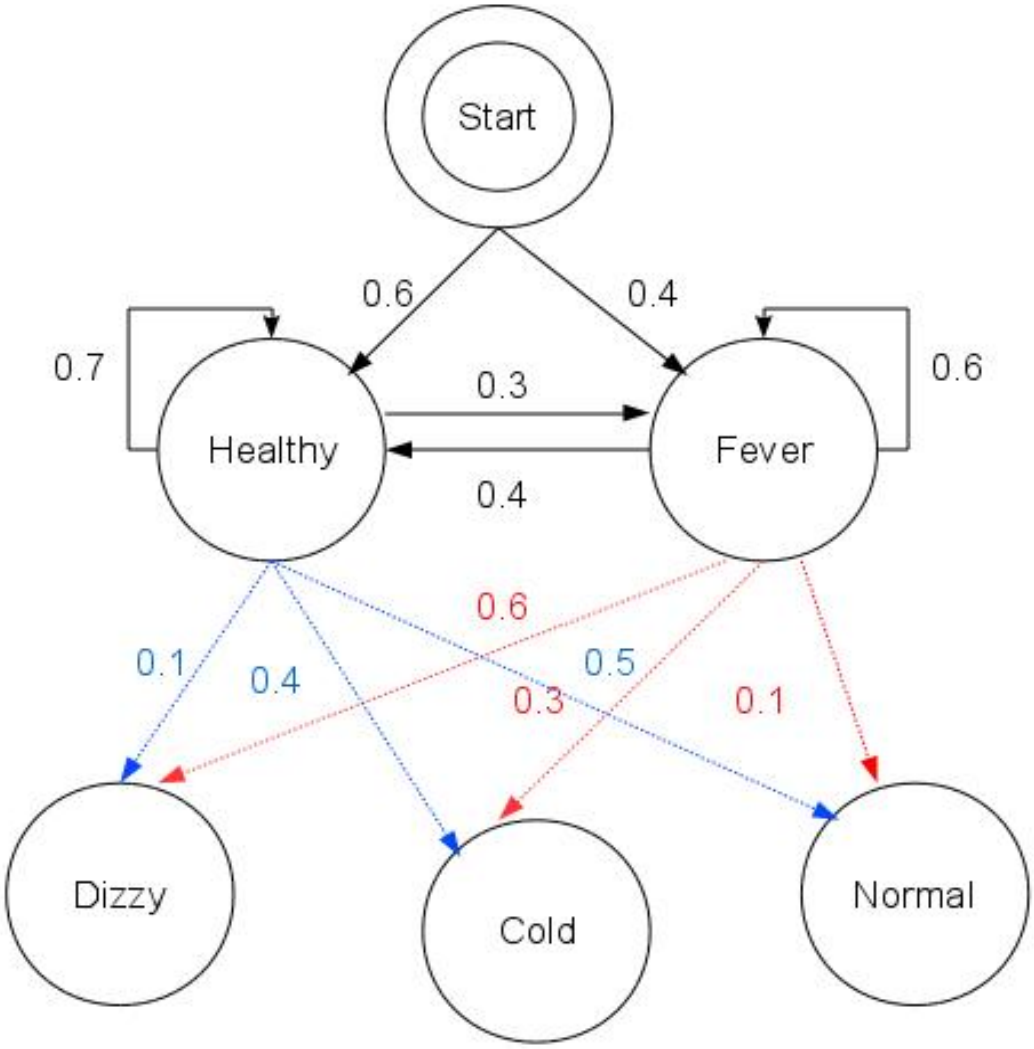
假设一个病人每天来到诊所并告诉医生他的感觉。医生相信病人的健康状况如同一个离散马尔可夫链。病人的状态有两种“健康”和“发烧”，但医生不能直接观察到，这意味着这是隐藏状态。每天病人会告诉医生自己有以下几种由他的健康状态决定的感觉的一种：正常、冷或头晕。这些是观察结果。 整个系统为一个隐马尔可夫模型(HMM)。

医生知道村民的总体健康状况，还知道发烧和没发烧的病人通常会抱怨什么症状。换句话说，医生知道隐马尔可夫模型的参数。你可以用程序语言（Python）写下来：

```
states = ('Healthy', 'Fever')

observations = ('normal', 'cold', 'dizzy')

start_probability = {'Healthy': 0.6, 'Fever': 0.4}

transition_probability = {
   'Healthy' : {'Healthy': 0.7, 'Fever': 0.3},
   'Fever' : {'Healthy': 0.4, 'Fever': 0.6},
   }

emission_probability = {
   'Healthy' : {'normal': 0.5, 'cold': 0.4, 'dizzy': 0.1},
   'Fever' : {'normal': 0.1, 'cold': 0.3, 'dizzy': 0.6},
   }
```

在这段代码中, 起始概率 `start_probability` Π 表示病人第一次到访时医生认为其所处的 HMM 状态，他唯一知道的是病人倾向于是健康的。 转移概率 `transition_probability` A表示潜在的马尔可夫链中健康状态的变化。在这个例子中，当天健康的病人仅有 30%的机会第二天会发烧。放射概率 `emission_probability` B表示每天病人感觉的可能性。假如他是健康的，50%会感觉正常。如果他发烧了，有 60%的可能感觉到头晕。

描述这个过程的 HMM 可用下图表示：



HMM 观测序列的生成（问题 1）

根据上述的描述，请给出 HMM 观测序列生成过程的伪代码。

Input: HMM 的模型$\lambda = (A, B, \Pi)$, 观测序列的长度T

Output: 观测序列$O = o_1, o_2, \ldots o_T$

Step 1: 根据初始状态概率分布Π生成隐藏状态i_1.

Step 2: for t from 1 to T

(补充完成 Step 2 需要的内容，注意，包括隐藏状态的生成，以及根据隐藏状态生成观测状态)

Final: 所有的o_t一起形成观测序列$O = o_1, o_2, \ldots o_T$

Mark: 10 points

Step 2: for t from 1 to T

a. 生成隐藏状态 it。

根据前一个隐藏状态 it-1 和状态转移矩阵 A，根据转移概率选择下一个隐藏状态 it。

b. 生成观测状态 ot。

根据当前隐藏状态 it 和观测状态概率矩阵 B，根据发射概率选择观测状态 ot。

HMM 模型三个基本问题

1. 计算观察序列概率，称之为似然（Likelihood）问题。即给定模型$\lambda = (A, B, \Pi)$和观测序列$O = o_1, o_2, \ldots o_T$，计算在模型λ下观测序列O出现的概率$P(O|\lambda)$。这个问题的求解需要用到**前向算法**(Forward Algorithm)，是 HMM 模型三个问题中最简单的。

2. 预测问题，也称为解码问题。即给定模型$\lambda = (A, B, \Pi)$和观测序列$O = o_1, o_2, \ldots o_T$，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**(Viterbi algorithm)。这个问题是 HMM 模型三个问题中复杂度居中的算法。

3. 模型参数学习问题。即给定观测序列$O = o_1, o_2, \ldots o_T$，估计模型$\lambda = (A, B, \Pi)$的参数，使该模型下观测序列的条件概率$P(O|\lambda)$最大。这个问题的求解需要用到基于最大期望算法（Expectation Maximization，EM 算法）的**鲍姆-韦尔奇算法**（Baum-Welch 算法），也被称之为前向后向算法。这个问题是 HMM 模型三个问题中最复杂的，在本文不做介绍。

前向算法（问题 2、问题 3，共 40 points）

我们首先求解第一个问题，我们已知 HMM 模型的参数$\lambda = (A, B, \Pi)$。其中A是隐藏状态转移概率的矩阵，B是观测状态生成概率的矩阵，Π是隐藏状态的初始概率分布。同时我们也已经得到了观测序列$O = o_1, o_2, \ldots o_T$，现在我们要求观测序列O在模型λ下出现的条件概率$P(O|\lambda)$。

这个问题看上去非常简单，因为我们知道所有的隐藏状态之间的转移概率和所有从隐藏状态到观测状态生成概率，那么我们穷举即可。以下是该方法的描述：

任意一个隐藏序列$I = i_1, i_2, \ldots, i_T$出现的概率是：

$$P(I|\lambda) = \pi_{i_1} a_{i_1 i_2} a_{i_2 i_3} \ldots a_{i_{T-1} i_T}$$

对于固定的状态序列$I = i_1, i_2, \ldots, i_T$，我们要求的观察序列$O = o_1, o_2, \ldots o_T$出现的概率是：

$P(O|I,\lambda) = b_{i_1}(o_1)b_{i_2}(o_2)\dots b_{i_T}(o_T)$

则O和I联合出现的概率是：

$P(O,I|\lambda) = P(I|\lambda)P(O|I,\lambda) = \pi_{i_1}a_{i_1i_2}a_{i_2i_3}\dots a_{i_{T-1}i_T}b_{i_1}(o_1)b_{i_2}(o_2)\dots b_{i_T}(o_T)$

然后求边缘概率分布，即可得到观测序列O在模型λ下出现的条件概率$P(O|\lambda)$：

$$P(O|\lambda) = \sum_I P(O,I|\lambda) = \sum_I \pi_{i_1}a_{i_1i_2}a_{i_2i_3}\dots a_{i_{T-1}i_T}b_{i_1}(o_1)b_{i_2}(o_2)\dots b_{i_T}(o_T)$$

在这里，I代表所有隐藏序列的可能性，即所有隐藏序列的排列组合。

问题 2

上述算法的计算复杂度是非常高的。若假设每一次四则运算需要消耗单位时间 1，N代表隐藏状态的个数，求上述算法的计算复杂度。

Mark: 10 points

假设隐藏状态的个数为 N，观测序列的长度为 T。
1. 初始化步骤：
初始化前向概率矩阵 alpha[N][T]：这需要分配空间来存储 alpha 矩阵，其大小为 N * T。因此，初始化步骤的时间复杂度为 O(N * T)。
2. 递推步骤：
对于每个时间步 t = 2 to T：
对于每个隐藏状态 i = 1 to N：
计算当前时间步的前向概率：需要进行 N 次乘法和 N-1 次加法，因此计算当前时间步的前向概率的复杂度为 O(N)。
更新前向概率矩阵 alpha[N][T]：需要更新 alpha 矩阵中的每个元素，因此更新前向概率矩阵的复杂度为 O(N)。
总计算量为 O(T * N * (N + N)) = O(N^2 * T)
3. 终止步骤：
计算观测序列的概率：需要对隐藏状态的概率进行加法运算，因此计算观测序列的概率的复杂度为 O(N)。
故前向算法的总计算复杂度为 O(N * T) + O(N^2 * T) + O(N) = O(N^2 * T)。

问题 3（Mark 30 points)

实际上在上述求解的过程，我们看到了一个重要事实，即*隐马尔可夫模型是一个典型的状态推移过程，整个问题的解由子问题构成，具备最优子结构，且后续子问题的求解依赖于前序子问题*。这正是典型的**动态规划**问题。请复习动态规划一章，并注意对比隐马尔可夫模型和斐波那契数列推导的相似性。接下来将使用动态规划算法重构上述过程，并求解其计算复杂度。

和动态规划问题类似，我们按照动态规划的五步法来来构建：

1. 定义子问题。我们定义时刻t时隐藏状态为q_i，观测状态的序列为$o_1, o_2, \ldots o_t$的概率为子问题，又称之为前向概率，$f_t(i) = P(o_1, o_2, \ldots o_t, i_t = q_i|\lambda)$。

2. 猜测。假设q_i已经得到，则 t->t+1 时刻，实际上$f_{t+1}(i)$的状态，是由N个隐藏状态$q_1, \ldots q_N$，跳转到q_i，并且此时从隐藏状态q_i -> 观测状态o_{t+1}。

3. （**Mark: 10 points**) 子问题求解。请根据 2，写出$f_{t+1}(i)$与$f_t(i)$之间的递推关系式，并依据此证明该问题具有最优子结构，可适用于动态规划算法。

在隐马尔可夫模型中，ft(i) 表示在时刻 t 时处于隐藏状态 i 的概率，ft+1(i) 表示在时刻 t+1 时处于隐藏状态 i 的概率。根据隐马尔可夫模型的定义，隐藏状态的转移概率为 a(i, j)，观测状态的发射概率为 b(j, o_t)。

ft+1(i) = sum(ft(j) * a(j, i) * b(i, o_t+1))，其中 j 取遍所有可能的隐藏状态。

这个关系式表示，在时刻 t+1 时处于隐藏状态 i 的概率 ft+1(i) 可以通过时刻 t 时所有隐藏状态 j 的概率 ft(j) 乘以隐藏状态的转移概率 a(j, i) 和观测状态的发射概率 b(i, o_t+1) 来计算得到。

证明该问题具有最优子结构:

假设我们已经得到了时刻 t 时隐藏状态为 j 的前向概率 ft(j)，我们想要求解时刻 t+1 时隐藏状态为 i 的前向概率 ft+1(i)。假设在时刻 t 时的最优子结构已经得到，即对于所有隐藏状态 j，ft(j) 的值都是最优的。我们需要证明，在时刻 t+1 时，以隐藏状态 i 结尾的最优子结构也能得到最优解。

假设存在一个最优解，即存在一个隐藏状态序列 s'，使得 ft+1(s') 的概率最大。

现在考虑将 s' 中倒数第二个状态 j 替换为另一个状态 k，得到新的隐藏状态序列 s。根据递推关系式，对于时刻 t+1，ft+1(i) 可以通过 ft(j) 和隐藏状态的转移概率 a(j, i) 以及观测状态的发射概率 b(i, o_t+1) 计算得到。由于 s' 是最优解，ft+1(s') 的概率最大，因此根据递推关系式，ft+1(i) 的值已经是最优的，无法通过替换隐藏状态 j 为 k 来得到更大的概率。

这证明了在时刻 t+1 时，以隐藏状态 i 结尾的最优子结构也能得到最优解。

4. 递归+求解。根据 2、3 获得了递归过程，这里需要构建 TxN 矩阵所有$f_t(i)$。这里的初始状态是$f_1(i) = \pi_i b_i(o_1), i = 1,2, \ldots N$

5. (**Mark: 10 points** ）根据 1-4 完成原问题的解法，通过结合所有子问题的解$f_T(i)$，求出原问题。补充完成前向算法的伪代码。

Input：HMM 模型$\lambda = (A, B, \Pi)$，观测序列$O = (o_1, o_2, \ldots o_T)$

Output：观测序列概率$P(O|\lambda)$

for state i=1 to N

Initialize: 计算 1 时刻隐藏状态i的前向概率

(补充初始状态计算前向概率的操作)

Recursion: 递推时刻 *2,3,…T*时刻的前向概率

for t = **_2__** *to* **__T__**:

for i = **___1__** to __N___:

$f_{t+1}(i)$ = **sum(ft(j) * a(j, i) * b(i, o_t+1))，其中 j 取遍所有可能的隐藏状态。**

(补充 t 状态计算前向概率的操作以及 for 循环的次数)

Final: $P(O|\lambda) = \boldsymbol{sum}(\boldsymbol{F}(\boldsymbol{T}, \boldsymbol{i})), \boldsymbol{i} = \mathbf{1}\ \boldsymbol{to}\ \boldsymbol{N}$.

（**Mark: 10 points**）根据上述过程，给出前向算法的算法复杂度。

初始化步骤的复杂度为 O(N)，其中 N 是状态数。
递推步骤的复杂度为 O(T * N)，其中 T 是观测序列的长度，N 是状态数。
对最后一个时刻 T 的前向概率进行求和。对隐藏状态 i 进行一次求和操作。因此，最终步骤的复杂度为 O(N)，其中 N 是状态数。
前向算法的总体复杂度为 O(T * N)，其中 T 是观测序列的长度，N 是状态数。

Viterbi 算法（问题 4，共 30 points）

接下来，我们研究 HMM 的一个重要问题，即给定模型和观测序列，求给定观测序列条件下，最可能出现的对应的隐藏状态序列。这是 HMM 的解码问题，也是该模型最为广泛应用的一种场景。HMM 模型的解码问题最常用的算法是维特比算法，当然也有其他的算法可以求解这个问题。同时维特比算法是一个通用的求序列最短路径的动态规划算法，也可以用于很多其他问题。我们采用通用的动态规划算法来构造这个问题的解，并回答问题四。

在 HMM 模型的解码问题中，给定模型$\lambda = (A, B, \Pi)$和观测序列$O = o_1, o_2, \dots o_T$，求给定观测序列 O 条件下，最可能出现的对应的状态序列$I^* = i_1^*, i_2^*, \dots i_T^*$，即$P(I^*|O)$要最大化。实际上，即是在给定$O = o_1, o_2, \dots o_T$，我们穷举所有可能的$I = i_1, i_2, \dots i_T$序列，并计算$P(I|O)$，哪一个值最大则选取哪一组$I$为$I^*$。接下来我们遵循动态规划五步法来构建算法：

1. 子问题的定义。我们定义，时刻t隐藏状态为i 所有可能的状态转移路径 $i_1, i_2, \dots i_t$中的概率最大值为$v_t(i)$：

$v_t(i) = \max P(i_t = i, i_1, i_2, \dots i_{t-1}, o_t, o_{t-1}, \dots o_1|\lambda), i = 1,2, \dots N$

这个概率是在λ模型下，观察为$o_1, \dots, o_t$，状态为$q_0, \dots, q_{t-1}, i$的**联合概率**。$v_t(i)$它要求在所有长度为t的路径中寻找最大的联合概率(当然也需要找到这条路径)，要求是t时刻的状态是一定为i。

1. 猜测。假设$v_t(i)$已经得到，则 $t \to t+1$ 时刻，$v_{t+1}(i)$ 代表$t+1$ 时刻，观测序列为$o_1, \ldots, o_{t+1}$，且此时状态为i的最大联合概率。很显然，我们需要先求N个隐藏状态$q_1, \ldots q_N$，跳转到q_i，并且此时从隐藏状态q_i 到观测状态o_{t+1}的概率，而$v_{t+1}(i)$是这一系列概率中的最大值。注意这里和前向算法的不同，前向算法是求和，这里是求最大值。

2. （**Mark: 10 points**) 子问题求解。请根据 2，写出$v_{t+1}(i)$与$v_t(i)$之间的递推关系式，并依据此证明该问题具有最优子结构，可适用于动态规划算法。

为了计算 vt+1(i)，我们需要考虑所有可能的前一个状态 j，然后选择能够使得 vt(j) * a(j, i) * b(i, ot+1) 最大的 j 值。这个计算可以表示为：

vt+1(i) = max(vt(j) * a(j, i) * b(i, ot+1))，其中 j 取遍所有可能的隐藏状态。

要证明问题具有最优子结构:

在 Viterbi 算法中，我们通过递推关系计算每个时刻的最大联合概率 vt(i)。在计算 vt+1(i) 时，我们需要考虑前一个时刻的最大联合概率 vt(j)，然后选择能够使得 vt(j) * a(j, i) * b(i, ot+1) 最大的 j 值。这样的选择保证了当前时刻的最优解（最大联合概率）包含了前一个时刻的最优解。因此，Viterbi 算法满足最优子结构性质。

在 Viterbi 算法中，我们计算每个时刻的最大联合概率 vt(i)。在计算 vt+1(i) 时，我们需要考虑前一个时刻的最大联合概率 vt(j)。由于隐藏状态的数量是有限的，因此同一个子问题（例如计算 vt+1(i)）可能会在不同的时刻被多次求解。这意味着我们可以通过存储和重用之前计算的结果来避免重复计算，从而实现子问题的重叠性质。

所以该问题具有最优子结构。

3. 递归+求解。根据 2、3 获得了递归过程，这里需要构建$T * N$矩阵或是相应的数据结构存储所有$v_t(i)$。这里的初始状态是$v_1(i) = \pi_i b_i(o_1), i = 1,2, \ldots N$。注意到一个问题，从 3 的递归式子，我们只记录了t时刻的隐藏状态i出现的联合概率$v_t(i)$，并没有记录具体的状态。根据定义，时刻t隐藏状态为i，$\Phi_t(i)$代表 $t-1$ 时刻所处的状态，则$\Phi_t(i)$是使得转移概率最大的隐藏状态，即：

$$\Phi_t(i) = \text{argmax}_{1 \le j \le N} v_{t-1} a_{ji}$$

4. (**Mark: 20 points** ）根据 1-4 的讨论，我们知道 T 时刻，最终的$P(I^*|O)$为：

$$P(I^*|O) = \max_{1 \le i \le N} v_T(i)$$

此时，有

$$i_T^* = \text{argmax}_{1 \le i \le N} v_T(i),$$

即使得上述$P(I^*|O)$成立的i。此时我们有$\Phi_T(i) = i_T^*$。因此，为了求得隐藏状态序列，我们从$t = 1$ 开始构建完$v_t(i)$矩阵，然后从$t = T-1$ 时刻开始，逆向求每一个隐藏状态：

$i_t^* = \Phi_{t+1}(i_{t+1}^*)$

根据上述讨论，请总结为伪代码（或是某种编程语言完成）（**10 points)**，并简要分析一下算法复杂度（**10 points**）。

Input:
- 观测序列 O = (o1, o2, ..., oT)
- 隐藏状态集合 S = {s1, s2, ..., sN}
- 初始状态概率向量 π = (π1, π2, ..., πN)
- 状态转移概率矩阵 A = [[a11, a12, ..., a1N], [a21, a22, ..., a2N], ..., [aN1, aN2, ..., aNN]]
- 观测状态发射概率矩阵 B = [[b1(o1), b1(o2), ..., b1(oT)], [b2(o1), b2(o2), ..., b2(oT)], ..., [bN(o1), bN(o2), ..., bN(oT)]]

Output:
- 最优隐藏状态序列 I = (i1, i2, ..., iT)

```
Procedure Viterbi(O, S, π, A, B):
    T = length(O)  // 观测序列的长度
    N = length(S)  // 隐藏状态的数量

    // Step 1: 初始化
    for i = 1 to N:
        v[1][i] = π[i] * B[i][o1]
        backpointer[1][i] = 0

    // Step 2: 递推
    for t = 2 to T:
        for i = 1 to N:
            max_prob = 0
            max_state = 0
            for j = 1 to N:
                prob = v[t-1][j] * A[j][i] * B[i][ot]
                if prob > max_prob:
                    max_prob = prob
                    max_state = j
            v[t][i] = max_prob
            backpointer[t][i] = max_state

    // Step 3: 终止
    max_prob = 0
    max_state = 0
    for i = 1 to N:
        if v[T][i] > max_prob:
            max_prob = v[T][i]
```

```
        max_state = i
    IT = max_state

    // Step 4: 回溯
    I[T] = IT
    for t = T-1 to 1:
        I[t] = backpointer[t+1][I[t+1]]

    return I
```

Viterbi 算法的复杂度主要取决于观测序列的长度 T 和隐藏状态的数量 N。

时间复杂度分析：

初始化阶段需要计算初始时刻的最大联合概率 vt(i)，时间复杂度为 O(N)。

递推阶段需要计算每个时刻 t 的最大联合概率 vt(i)，对于每个 t 和每个隐藏状态 i，需要遍历前一个时刻的 N 个状态 j，时间复杂度为 O(N^2)。这个过程需要执行 T-1 次，因此总的时间复杂度为 O(TN^2)。

终止阶段需要找到最终时刻 T 的最大联合概率和对应的隐藏状态 iT，时间复杂度为 O(N)。

回溯阶段需要回溯计算每个时刻的隐藏状态，需要执行 T-1 次，每次回溯的时间复杂度为 O(1)。因此，总的时间复杂度为 O(T)。

总的时间复杂度为 O(TN^2)。

空间复杂度分析：

算法使用了两个矩阵 v[t][i] 和 backpointer[t][i]，大小为 T×N，因此空间复杂度为 O(TN)。

此外，需要存储初始状态概率向量 π、状态转移概率矩阵 A 和观测状态发射概率矩阵 B，它们的大小分别为 N、N×N 和 N×T，因此额外需要的空间复杂度为 O(N^2 + NT)。

总的空间复杂度为 O(TN)。

综上所述，Viterbi 算法的时间复杂度为 O(TN^2)，空间复杂度为 O(TN)。

HMM 的应用 -- 中文分词（Mark: 20 points）

HMM 模型在语音识别与自然语言处理中有广泛应用。接下来，我们以中文分词为例子，讲述一下 HMM 的应用。

中文分词是自然语言处理中一个最基础的问题，就是一段话怎么切分。比如：

今天的天气怎么样

可以按如下方式分词

今天\天气\怎么样

这样一个任务交给机器自动切分，很容易想到基于词典的匹配，但是匹配会面临如下问题：

1. 词典需要人工不停录入，否则没法识别新词。这也就是机器学习中一直强调的泛化能力；

2. 不同场景下分词方式不一样，比如：

   南京\市长\是\个\好\同志

而下面这句话就不能把“市长”分到一起了：

南京市\长江\大桥

因此，我们需要构建更先进的分词方法，其中一种则是基于统计的方法。首先我们转换下思维，把分词问题做个转换：分词问题就是对句子中的每个字打标注，标注要么是一个词的开始（B），要么是一个词的中间位置（M），要么是一个词的结束位置（E），还有单个字的词，用 S 表示。比如下面的句子可以这样标注：

我想去乌鲁木齐

SSSBMME

结合上述讨论：

1. （**10 points**) 如果采用 HMM 模型来做，请问这里的状态、观测序列、状态转移矩阵、观测状态概率矩阵分别对应什么呢？

在中文分词中，HMM 模型的状态对应分词标记（B、M、E、S），观测序列对应待分词的句子中的每个字，状态转移矩阵描述了分词标记之间的转移关系，观测状态概率矩阵描述了在每个分词标记下，观测到某个字的概率。通过学习这些概率参数，HMM 模型可以根据观测序列（待分词的句子）推断出最可能的分词结果（状态序列）。

2. (**10 points**) 请查阅相关资料，假设给定了分好词的语料库，请问：a) 初始状态的概率分布是什么，如何求？ b) 如何统计并得到状态转移矩阵和观测状态概率矩阵？ c) 训练好后，给定一个句子，应该使用什么算法以及如何来做分词？

a） 初始状态的概率分布表示在模型开始时，处于各个状态的概率。在中文分词中，可以根据分好词的语料库统计每个状态（B、M、E、S）作为句子中第一个字的概率分布。具体计算方法是统计语料库中以某个状态开头的字的数量，然后将其除以总字数得到概率。

b）状态转移矩阵：可以根据分好词的语料库统计每个状态之间的转移频次，即从一个状态转移到另一个状态的次数。然后将这些频次除以对应状态的总频次，得到状态转移的概率。

观测状态概率矩阵：可以根据分好词的语料库统计每个状态下每个字出现的频次，然后将这些频次除以对应状态的总频次，得到观测到某个字的概率。

c）给定一个句子，应该使用 Viterbi 算法来进行分词：

首先，将待分词的句子作为观测序列输入到已经训练好的 HMM 模型中。 利用 Viterbi 算法，根据模型的状态转移矩阵和观测状态概率矩阵，计算出最可能的状态序列，即最可能的分词结果。根据最可能的状态序列，将句子进行分词，得到最终的分词结果。

Viterbi 算法是一种动态规划算法，用于在 HMM 模型中找到给定观测序列的最可能的状态序列。它通过在每个时刻计算局部最优解，并利用状态转移概率和观测状态概率进行递推，最终得到全局最优解。在中文分词中，Viterbi 算法可以用于根据观测序列（待分词的句子）计算最可能的分词结果（状态序列）。

致谢

本作业到此就结束了。感谢大家这个学期的选课和支持！