

**随机过程实验**

**实验题目 隐马尔可夫模型**

**学 号 1180300922**

**姓 名 王志泓**

**指导教师 范晓鹏**

**日 期 2020.11.7**

1. 实验目的

了解马尔可夫链的原理以及实际应用，了解隐马尔可夫模型（HMM）的特性，学习解决隐马尔可夫模型的三个问题

概率计算：计算特定观测序列的概率forward/backward算法；

预测问题：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状态序列viterbi解码算法(基于动态规划的思想)；

学习问题：给定观测序列，估计模型的参数，使得在该模型下观测序列的条件概率最大:baum-welch算法。

1. 实验内容

**问题一：**

假设晴天和雨天的初始概率分别为0.6和0.4，如果前一天是晴天，则第二天晴天和雨天概率仍然是0.6和0.4，如果前一天是雨天，则第二天晴天和雨天概率分别为0.3和0.7。

1.试写出天气（晴天、雨天）的状态转移矩阵。

2.根据初始概率和状态转移矩阵，随机生成20天的天气序列。（用1表示晴天，2表示雨天）

**问题二：**

一朋友每天根据天气{天晴，下雨}按以下概率决定当天的活动{公园散步,购物,清理房间}中的一种

emission\_probability = {

'Sunny' : {'walk': 0.6, 'shop': 0.3, 'clean': 0.1},

'Rainy' : {'walk': 0.1, 'shop': 0.4, 'clean': 0.5},

}

1. 请按照问题一生成的天气序列，以及以上概率，来生成这位朋友这20天的活动序列（用1表示散步，2表示购物，3表示清理）

**问题三：**

问题二中的朋友每天在朋友圈发布当天的活动

1.假设他连续三天发布的活动状态分别是1 2 3，请计算这三天天气序列为1 2 2的概率。

2.假设他连续二十天发布的状态是2 1 3 2 3 2 2 3 3 1 2 1 1 1 2 3 3 3 3 2 ，请推测这20天的天气。

1. 实验过程

**对于问题一**，根据题设，可以得到状态转移矩阵为

其中表示由转化为的概率。初始概率向量乘以状态转移矩阵的次方得到的向量即表示在第天天气变化为晴天或雨天的概率，利用MATLAB中的函数生成一个均匀分布在0-1内的随机数，如果随机数数值小于天气变化为晴天的概率，即天气变化为晴天，反之则变化为阴天。

**对于问题二**，根据问题一产生的天气序列，进行类似于问题一的判断，根据随机数的数值决定在不同天气进行的不同活动，例如，若天气为晴天，随机数数值为，则活动为散步。重复20次即可得到活动序列。

**对于问题三**，对于第一问，可以使用穷举法，即分别计算不同天气序列下活动状态为1 2 3的概率,则已知活动序列为1 2 3的前提下，天气序列为1 2 2的概率为

对于第二问，假如采用穷举法，穷举出所有可能的状态序列再比较他们的概率值，则时间复杂度是, 显然这样的时间复杂度是无法接受的，而通过维特比算法能把时间复杂度降到 。用动态规划的思路去思考这个问题，记 为上一个观测现象对应的各个隐含状态的概率， 为现在的观测现象对应的各个隐含状态的概率。则求解实际上只依赖于。为每个隐含状态维护一个路径 ， 表示到达状态 前的最优状态序列。通过前面的计算选出那个最有可能产生当前状态 的上一状态 后，往 中插入 。则依照这种方法遍历完所有的观测序列后，只需要选择 中概率值最大的那个 作为最终的隐含状态，同时从 中取出 作为该最终隐含状态前面的状态序列。

1. 实验结果

**问题一及问题二代码如下：**

**﻿**clear;

init = [0.6,0.4];

matrix = [0.6,0.4;0.3,0.7];

weather = [];

for i = 1:20

p = mtimes(init,matrix^i);

if rand() < p(1)

weather(i) = 1;

else

weather(i) = 2;

end

end

weather

state = [];

for j = 1:20

r = rand();

if (weather(j) == 1) && (r < 0.6)

state(j) = 1;

elseif (weather(j) == 1) && (r > 0.6) && (r < 0.9)

state(j) = 2;

elseif (weather(j) == 1) && (r > 0.9)

state(j) = 3;

elseif (weather(j) == 2) && (r < 0.1)

state(j) = 1;

elseif (weather(j) == 2) && (r > 0.1) && (r < 0.5)

state(j) = 2;

else

state(j) = 3;

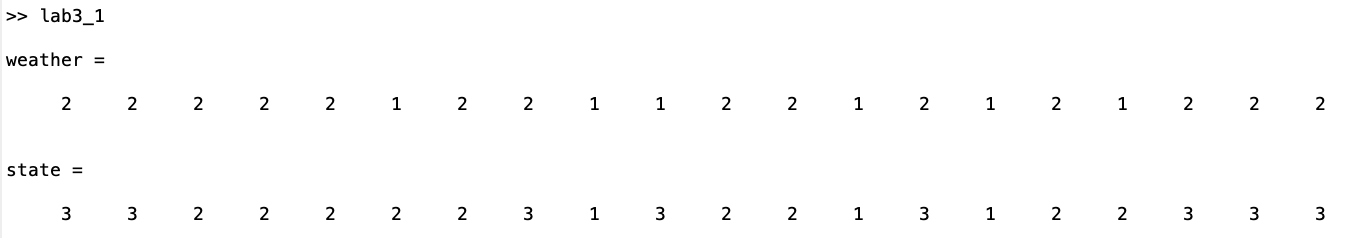
end

end

state

**问题一及问题二结果如下：**

**1.1 状态转移矩阵：**

****

**图1-程序运行结果**

**1.2 随机生成20天的天气序列：**

2 2 2 2 2 1 2 2 1 1 2 2 1 2 1 2 1 2 2 2

**2.1 对应的20天的活动序列：**

3 3 2 2 2 2 2 3 1 3 2 2 1 3 1 2 2 3 3 3

**问题三代码如下：**

﻿states = ('1', '2')

observations = ('1', '2', '3')

start\_probability = {'1': 0.6, '2': 0.4}

transition\_probability = {

'1' : {'1': 0.6, '2': 0.4},

'2' : {'1': 0.3, '2': 0.7},

}

emission\_probability = {

'1' : {'1': 0.6, '2': 0.3, '3': 0.1},

'2' : {'1': 0.1, '2': 0.4, '3': 0.5},

}

def Viterbit(obs, states, s\_pro, t\_pro, e\_pro):

path = { s:[] for s in states}

curr\_pro = {}

for s in states:

curr\_pro[s] = s\_pro[s]\*e\_pro[s][obs[0]]

for i in xrange(1, len(obs)):

last\_pro = curr\_pro

curr\_pro = {}

for curr\_state in states:

max\_pro, last\_sta = max(((last\_pro[last\_state]\*t\_pro[last\_state][curr\_state]\*e\_pro[curr\_state][obs[i]], last\_state)

for last\_state in states))

curr\_pro[curr\_state] = max\_pro

path[curr\_state].append(last\_sta)

max\_pro = -1

max\_path = None

for s in states:

path[s].append(s)

if curr\_pro[s] > max\_pro:

max\_path = path[s]

max\_pro = curr\_pro[s]

for i in range(0,len(obs)):

print(str(max\_path[i])+' '),

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

obs = ['2', '1', '3', '2', '3', '2', '2', '3', '3', '1', '2', '1', '1', '1', '2', '3', '3', '3', '3', '2' ]

Viterbit(obs, states, start\_probability, transition\_probability, emission\_probability)

**问题三结果如下：**

**3.1 天气序列为1 2 2的概率：**

**3.2 推测的天气序列：**

2 1 2 2 2 2 2 2 2 1 2 1 1 1 2 2 2 2 2 2

1. 心得体会

在正常的马尔可夫模型中，状态对于观察者来说是直接可见的。这样状态的转换概率便是全部的参数。而在隐马尔可夫模型中，状态并不是直接可见的，但受状态影响的某些变量则是可见的。每一个状态在可能输出的符号上都有一概率分布。因此输出符号的序列能够透露出状态序列的一些信息。

因此HMM广泛应用在语音识别、中文断词/分词或光学字符识别、机器翻译、生物信息学和基因组学中。

维特比算法运用了动态规划的思路，将计算状态序列的复杂度大大降低。