装备升级问题的 Markov 解法

理论预备

一、一般随机过程

1、 定义：

依赖于一个变动参数t的一族随机变量$\{X(t), t \in T\}$。其中，变动参数t的所有可取值的集合T为参数空间。X(t)的值所构成的集合S成为随机过程的状态空间。

例如，从时间$t = 0$开始记录某电话总机的呼叫次数，设$t = 0$时没有呼叫，至时刻t的呼叫次数记作N_t，则随机变量族$\{N_t, t \ge 0\}$是随机过程。

2、 马氏过程：

如果已知在时间t系统处于状态X的条件下，在时刻$\tau(\tau > t)$系统所处状态与时刻t以前系统所处的状态无关，此过程便为马尔可夫过程（随机过程的一个子类）。

例如，在布朗运动中，已知时刻t下的运动状态条件下，微粒在t后的运动情况和微粒在t以前的情况无关。若X(t)表示微粒在时刻t的位置，则X(t)是马尔可夫过程。

二、马尔可夫链

1、 定义：

设$\{X_n, n = 1,2,\ldots\}$是一个随机变量序列，用“$X_n = i$”表示时刻n系统处于状态i这一事件，称$p_{ij}(n) = p(X_{n+1} = j \mid X_n = i)$为事件“$X_n = i$”出现的条件下，事件“$X_{n+1} = j$”出现的概率，又称它为系统的一步转移概率。若对任意的非负整数i_1、i_2、$i_3 \ldots i_{n-1}$、i、j以及一切$n \ge 0$，有

$$p(X_{n+1} = j \mid X_n = i, X_k = i_k, k = 1,2,\ldots n-1) = p(X_{n+1} = j \mid X_n = i) = p_{ij}(n)$$

则称$\{X_n\}$是一个马尔可夫链。一步转移概率有以下的性质：

$$p_{ij} \ge 0，(i,j = 1,2,\ldots,n)$$

$$\sum_{j=1}^{n} p_{ij} = 1，\ (i,j = 1,2,\ldots,n)$$

把各个状态之间的一步转移概率排成矩阵，成为状态矩阵

$$P = \begin{pmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{pmatrix}$$

每个状态i对应状态矩阵P的第i行。

三、k步转移概率与k步转移矩阵

1、 k步转移概率

系统从状态i恰好经过k步转移到状态j的概率。记作$p_{ij}^{(k)} = p(X_{k+1} = j \mid X_1 = i)$。

2、 k步转移矩阵

$$P^{(k)} = \left(p_{ij}^{(k)}\right)_{n\times n}$$

显然，$P^{(k)}$为概率矩阵，即有：

$$p_{ij}^{(k)} \ge 0，(i,j = 1,2,\ldots,n)$$

$$\sum_{j=1}^{n} p_{ij}^{(k)} = 1，\ (i,j = 1,2,\ldots,n)$$

3、切普曼·柯尔莫哥洛夫方程

$$p_{ij}^{(n)} = \sum_k p_{ik}^{(m)} p_{kj}^{(n-m)}$$

$$P^{(n)} = P^{(m)} P^{(n-m)}$$

应用以上方程可以推出：

$$P^{(n)} = P^{(n-1)} P = P^n$$

问题及解决

开门见山，直接提出问题：

已知：某装备基础等级为 1，最高等级为N。当装备在等级i的状态时，进行一次升级行为之后，到达等级j的状态的概率为p_{ij}，即一步转移概率矩阵是

$$P = \begin{pmatrix} p_{11} & \cdots & p_{1N} \\ \vdots & \ddots & \vdots \\ p_{N1} & \cdots & p_{NN} \end{pmatrix}$$

问题：此装备从等级x到等级$y(1 \le x < y \le N)$，平均需要进行多少次升级（升级次数的数学期望）？

分析思路：

首先需要计算进行k次升级到达等级y的概率。注意这个概率并不是k步转移矩阵中的那个$p_{xy}^{(k)}$。

理由：我们要求的这个事件的停止条件是“只要升到等级y就停止”；而k步转移中的停止条件是“只要升到次数到k就停止”，而不管k步之前是否有无到达过等级y。但是，这个概率却可以藉由k步转移方法计算得出。

“只要升到等级y就停止，共升级了k次”的言外之意是“前$k-1$次都没有到达过等级y”，也就是说，前$k-1$次的各个一步转移概率矩阵中，第y列元素必须都等于 0。

记：

$$I_x = (0 \ldots 1 \ldots 0)，第\ x\ 个元素为\ 1$$

$$I_y = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}，第\ y\ 个元素为\ 1$$

$$Ey = \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & 0 & \vdots \\ 0 & \cdots & 1 \end{pmatrix}，单位矩阵对角线上第\ y\ 个元素为\ 0$$

那么，事件“从等级x到等级y，只要升到等级y就停止，共升级了k次”的概率记作$p_{x\to y}^{(k)}$，就有：

$$p_{x\to y}^{(k)} = I_x (PEy)^{k-1} P I_y$$

于是，升级次数$t_{x\to y}$的数学期望（即平均次数）：

$$R_{x\to y} = E(t_{x\to y}) = \sum_{k=1}^{\infty} k \cdot p_{x\to y}^{(k)} = I_x \left[\sum_{k=1}^{\infty} k \cdot (PEy)^{k-1} \right] P I_y$$

具体计算：

由于PEy是一般矩阵，所以可以使用一般的方法。将PEy进行 Jordan 分解如下

$$PEy = AJA^{-1}$$

那么

$$R_{x\to y} = \sum_{k=1}^{\infty} k \cdot I_x (AJA^{-1})^{k-1} PI_y = I_x A\left(\sum_{k=1}^{\infty} k \cdot J^{k-1}\right) A^{-1} PI_y$$

由于J的特殊结构，大多数情况下

$$\sum_{k=1}^{\infty} k \cdot J^{k-1}$$

是可以手算出结果的，即便不能也可以由电脑计算。

如果$I - PEy$可逆且PEy的所有特征根的模都< 1，那么有以下计算$R_{x\to y}$的更简单方法些：记

$$S(n) = \left[\sum_{k=1}^{n} k \cdot (PEy)^{k-1}\right]$$

两边都乘PEy

$$S(n)PEy = \left[\sum_{k=1}^{n} k \cdot (PEy)^{k}\right]$$

上面两式相减，得

$$S(n)(I - PEy) = \left[\sum_{k=1}^{n} (PEy)^{k-1}\right] - n(PEy)^n$$

两边都乘$I - PEy$

$$S(n)(I - PEy)(I - PEy) = \left[\sum_{k=1}^{n} (PEy)^{k-1}\right](I - PEy) - n(PEy)^n (I - PEy)$$

即

$$S(n)(I - PEy)(I - PEy) = I + n(PEy)^{n+1} - (n + 1)(PEy)^n$$

由于前面假设$I - PEy$可逆，所以

$$S(n) = [I + n(PEy)^{n+1} - (n + 1)(PEy)^n][(I - PEy)^{-1}]^2$$

由PEy的所有特征根的模都< 1，得$(PEy)^n$收敛，所以

$$n(PEy)^{n+1} - (n + 1)(PEy)^n \to 0$$

所以

$$S(\infty) = [(I - PEy)^{-1}]^2$$

从而

$$R_{x\to y} = I_x[(I - PEy)^{-1}]^2 PI_y$$

举例模拟

例子：承接上面的问题，给出一步转移矩阵如下

$$P=\begin{pmatrix}0.1 & 0.9 & 0 & 0 & 0\\ 0.1 & 0.1 & 0.8 & 0 & 0\\ 0.1 & 0.1 & 0.1 & 0.7 & 0\\ 0.1 & 0.1 & 0.1 & 0.1 & 0.6\\ 0 & 0 & 0 & 0 & 1\end{pmatrix}$$

计算由 1 级到 5 级，平均需要的升级次数。

$$\mathrm{PEy}=\begin{pmatrix}0.1 & 0.9 & 0 & 0 & 0\\ 0.1 & 0.1 & 0.8 & 0 & 0\\ 0.1 & 0.1 & 0.1 & 0.7 & 0\\ 0.1 & 0.1 & 0.1 & 0.1 & 0\\ 0 & 0 & 0 & 0 & 0\end{pmatrix}$$

计算可得PEy最大特征根的模$=0.8119<1$，且

$$\mathrm{I}-\mathrm{PEy}=\begin{pmatrix}0.9 & -0.9 & 0 & 0 & 0\\ -0.1 & 0.9 & -0.8 & 0 & 0\\ -0.1 & -0.1 & 0.9 & -0.7 & 0\\ -0.1 & -0.1 & -0.1 & 0.9 & -0.6\\ 0 & 0 & 0 & 0 & 1\end{pmatrix}$$

其行列式$|\mathrm{I}-\mathrm{PEy}|=0.3024\neq 0$，即$\mathrm{I}-\mathrm{PEy}$可逆。

所以针对这个问题，有四种解决方法（因计算量庞大，故借助软件，这里使用的是 Matlab）：

1、基础的循环方法；

2、非 Jordan 分解的 Markov 方法；

3、用 Jordan 分解的 Markov 方法；

4、逆矩阵方法。

Matlab 程序代码：

```
clc;clear all;
format long g;

P=[0.1    0.9     0       0       0
    0.1    0.1    0.8     0       0
    0.1    0.1    0.1     0.7     0
    0.1    0.1    0.1     0.1    0.6
    0       0       0       0       1];

disp('=====================================================');
x=input('输入基础等级: ');
y=input('输入目标等级: ');
disp('=====================================================');

Ey=eye(5,5);Ey(y,y)=0;
Ix=zeros(1,5);Ix(x)=1;
Iy=zeros(5,1);Iy(y)=1;

%一、循环方法====================
tic;
N=10000;
```

```
total_sum=0;
%===概率判定矩阵，用以判断升级结果指向
T=zeros(5,6);
for j=2:6
    i=1;
    while i<=j-1
        T(:,j)=T(:,j)+P(:,i);
        i=i+1;
    end
end

for ii=1:N
    n=x;
    m=y;
    sum=0;
    while n<m
        a=rand(1);
        for i=1:5
            if  a>=T(n,i)&&a<T(n,i+1)
                n=i;
                sum=sum+1;
                break;
            end
        end
    end
    total_sum=total_sum+sum;
end
fprintf('一、基础的循环方法结果：%g次；用时：%g秒。\n',total_sum/N,toc);
disp('======================================================');

%  二、非Jordan分解的Markov方法====
tic;
R=0;
for n=1:1000
    R=R+n*Ix*(P*Ey)^(n-1)*P*Iy;
end
fprintf('二、非Jordan分解的Markov方法结果：%g次；用时：%g秒。\n',R,toc);
disp('======================================================');

%三、进行Jordan分解的Markov方法====
tic;
R=0;
[A,J]=jordan(P*Ey);
for n=1:100
```

```
    R=R+n*Ix*A*J^(n-1)*inv(A)*P*Iy;
end
fprintf('三、用Jordan分解的Markov方法结果：%g次；用时：%g秒。\n',R,toc);
disp('=========================================================');

%四、逆矩阵解法====================
tic;
R=0;
R=Ix*(inv(eye(5)-P*Ey))^2*P*Iy;
fprintf('四、逆矩阵方法结果：%g次；用时：%g秒。\n',R,toc);
disp('=========================================================');
```

运行结果如下：

```
===============================================================
输入基础等级: 1
输入目标等级: 5
===============================================================
一、基础的循环方法结果：7.7907 次；用时：0.567763 秒。
===============================================================
二、非 Jordan 分解的 Markov 方法结果：7.79101 次；用时：0.0140116 秒。
===============================================================
三、用 Jordan 分解的 Markov 方法结果：7.79101 次；用时：1.23351 秒。
===============================================================
四、逆矩阵方法结果：7.79101 次；用时：0.000714057 秒。
===============================================================
```

需要说明的是：前三种方法是普适的，当然运算量也很大；第四种方法却有其局限性，即要求矩阵$\mathrm{I-PEy}$可逆，否则将导致错误，例如：

计算从 1 级到 3 级的平均升级次数的时候，由于

$$\mathrm{I-PEy}=\begin{pmatrix} 0.9 & -0.9 & 0 & 0 & 0 \\ -0.1 & 0.9 & 0 & 0 & 0 \\ -0.1 & -0.1 & 1 & -0.7 & 0 \\ -0.1 & -0.1 & 0 & 0.9 & -0.6 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

其行列式$|\mathrm{I-PEy}|=0$，即$\mathrm{I-PEy}$不可逆，故而不能使用第四种方法。如果使用，结果如下：

```
===============================================================
输入基础等级: 1
输入目标等级: 3
===============================================================
一、基础的循环方法结果：2.5061 次；用时：0.184854 秒。
===============================================================
二、非 Jordan 分解的 Markov 方法结果：2.5 次；用时：0.0141719 秒。
===============================================================
三、用 Jordan 分解的 Markov 方法结果：2.5 次；用时：0.61763 秒。
```

```
==================================================================
Warning: Matrix is singular to working precision.
> In Up_grade at 76
四、逆矩阵方法结果：NaN 次；用时：0.000833905 秒。
==================================================================
```

从上面的结果看出，前三种方法都得出了正确结果，而在调用第四种方法的时候给出错误提示"Warning: Matrix is singular to working precision."意为"矩阵奇异，无法正确计算"。

Python 代码：

```
# -*- coding: utf-8 -*-
"""
装备升级问题的处理方法：
1、基本的循环方法；
2、Markov 链方法（没有使用 Jordan 分解，原因是 Python 里没有找到对应的函数，如果现编写 Jordan 分解函数，耗时耗力）；
3、Markov 链特殊情况：逆矩阵方法。
"""
import time
import random
import numpy as np
from scipy import linalg as la

#确定并输出一步转移概率矩阵 P 如下：
P=np.matrix([[0.1,0.9,0.0,0.0,0.0],
             [0.1,0.1,0.8,0.0,0.0],
             [0.1,0.1,0.1,0.7,0.0],
             [0.1,0.1,0.1,0.1,0.6],
             [0.0,0.0,0.0,0.0,1.0]])
print "=================================================="
print P
print "=================================================="

#从键盘获取基础等级 x 和目标等级 y：
x=int(raw_input('Input basic grade: '))
y=int(raw_input('Input target grade: '))
print "=================================================="

Ey=np.eye(5)
Ey[y-1,y-1]=0
Ix=np.zeros(5)
Ix[x-1]=1
Iy=np.zeros(5).reshape(5,1)
Iy[y-1]=1
```

```
begin=time.time()

#一、循环方法=====================
#计算并输出累计矩阵 T：
T=np.matrix([[0.0]*6]*5)
for i in range(5):
        T[:,i+1]=T[:,i]+P[:,i]
print T
print "=================================================="

#模拟过程，总模拟次数为 N:
N=100000.0
total_sum=0
for i in range(int(N)):
        n=x-1
        m=y-1
        s=0
        while n<m:
                a=random.random()
                for i in range(5):
                        if (a>=T[n,i])and(a<T[n,i+1]):
                                n=i
                                s=s+1
                                break
        total_sum=total_sum+s
print "The result of method 1:",total_sum/N,"times."
end1=time.time()
print 'Total time of method 1:',end1-begin,'seconds.'
print "=================================================="

#二、非 Jordan 分解的 Markov 方法====
R=0
for i in range(1,1000):
        R=R+i*Ix*(P*Ey)**(i-1)*P*Iy
print "The result of method 2:",R[0,0],"times."
end2=time.time()
print 'Total time of method 2:',end2-end1,'seconds.'
print "=================================================="

#三、逆矩阵解法====================
I=np.identity(5)
if la.det(I-P*Ey)!=0:
        R=Ix*((I-P*Ey).I)**2*P*Iy
```

```
        print "The result of method 3:",R[0,0],"times."
        end3=time.time()
        print 'Total time of method 3:',end3-end2,'seconds.'
else:
        print 'Matrix I-Q is singular, progress ends.'
print "=================================================="
```

结果：

初始等级 1，目标等级 5：

```
==================================================
[[ 0.1   0.9   0.    0.    0. ]
 [ 0.1   0.1   0.8   0.    0. ]
 [ 0.1   0.1   0.1   0.7   0. ]
 [ 0.1   0.1   0.1   0.1   0.6]
 [ 0.    0.    0.    0.    1. ]]
==================================================
Input basic grade: 1
Input target grade: 5
==================================================
[[ 0.    0.1   1.    1.    1.    1. ]
 [ 0.    0.1   0.2   1.    1.    1. ]
 [ 0.    0.1   0.2   0.3   1.    1. ]
 [ 0.    0.1   0.2   0.3   0.4   1. ]
 [ 0.    0.    0.    0.    0.    1. ]]
==================================================
The result of method 1: 7.78967 times.
Total time of method 1: 13.4219999313 seconds.
==================================================
The result of method 2: 7.79100529101 times.
Total time of method 2: 0.0780000686646 seconds.
==================================================
The result of method 3: 7.79100529101 times.
Total time of method 3: 0.0 seconds.
==================================================
```

初始等级 1，目标等级 3：

```
==================================================
[[ 0.1   0.9   0.    0.    0. ]
 [ 0.1   0.1   0.8   0.    0. ]
 [ 0.1   0.1   0.1   0.7   0. ]
 [ 0.1   0.1   0.1   0.1   0.6]
 [ 0.    0.    0.    0.    1. ]]
==================================================
Input basic grade: 1
Input target grade: 3
==================================================
```

```
[[ 0.    0.1   1.    1.    1.    1. ]
 [ 0.    0.1   0.2   1.    1.    1. ]
 [ 0.    0.1   0.2   0.3   1.    1. ]
 [ 0.    0.1   0.2   0.3   0.4   1. ]
 [ 0.    0.    0.    0.    0.    1. ]]
==================================================
The result of method 1: 2.50414 times.
Total time of method 1: 3.46799993515 seconds.
==================================================
The result of method 2: 2.5 times.
Total time of method 2: 0.0940001010895 seconds.
==================================================
Matrix I-Q is singular, progress ends.
==================================================
```

价值衡量

仅仅计算升级次数的数学期望是不够的，理由是，在每个等级进行一次升级操作（目标等级自然比当前等级高）的成本不是一成不变的。如果在等级i进行一次升级操作的成本记作c_i，那么成本向量就是

$$c = (c_1\ c_2 \dots c_N)$$

在从等级x到等级y的过程中，如果计算出在各个等级上进行升级次数的数学期望

$$r = (r_1\ r_2 \dots r_N)$$

那么，从等级x到等级y的总成本期望就是

$$C_{x\to y} = \sum_{i=1}^{N} c_i r_i = c \cdot r$$

很显然

$$R_{x\to y} = \sum_{i=1}^{N} c_i$$

剩下的问题就是计算$r = (r_1\ r_2 \dots r_N)$。

之前计算的事件"从等级x到等级y，只要升到等级y就停止，共升级了k次"的概率为$p_{x\to y}^{(k)}$，而且：

$$p_{x\to y}^{(k)} = I_x (PEy)^{k-1} P I_y$$

那么由归一化特点，必然有

$$\sum_{k=1}^{\infty} p_{x\to y}^{(k)} = I_x \left[\sum_{k=1}^{\infty} (PEy)^{k-1} P\right] I_y = 1$$

即在上述事件序列中，装备处于等级y的积累概率为 1，考虑到概率的意义，也即此过程中到达等级y的次数期望。那么对于

$$\sum_{k=1}^{\infty} (PEy)^{k-1} P$$

其第y列自然全是 1。第x行中第y个元素之前的各个数值就是装备从等级x到等级y过程中对应等级出现的次数期望。

另外，由于计数发生在升级操作之后，也即对升级结果状态计数，为了得到升级前状态计数，只要将初始状态计数结果加 1，结束状态计数结果减 1 即可。

所以

$$r = I_x\left\{\left[\sum_{k=1}^{\infty}(PEy)^{k-1}P\right] + I\right\}Ey$$

具体模拟计算向量r（借用之前的程序，只选 Matlab 中的基本循环和非 Jordan 方法）：

```
clc;clear all;
format short;

P=[0.1    0.9    0        0        0
     0.1    0.1    0.8    0        0
     0.1    0.1    0.1    0.7      0
     0.1    0.1    0.1    0.1      0.6
     0        0      0        0        1];

disp('=======================================================');
x=input('输入基础等级: ');
y=input('输入目标等级: ');
disp('=======================================================');

Ix=zeros(1,5);Ix(x)=1;
Ey=eye(5,5);Ey(y,y)=0;
r=[0 0 0 0 0];

tic;
N=100000;
total_sum=0;
T=zeros(5,6);
for j=2:6
        i=1;
        while i<=j-1
                T(:,j)=T(:,j)+P(:,i);
                i=i+1;
        end
end

for ii=1:N
        n=x;
        m=y;
        su=0;
        while n<m
```

```
            a=rand(1);
            r(n)=r(n)+1;
            for i=1:5
                if   a>=T(n,i)&&a<T(n,i+1)
                    n=i;
                    su=su+1;
                    break;
                end
            end
        end
        total_sum=total_sum+su;
end
disp('循环方法结果：')
r=r/N
disp('======================================================');

%  二、非 Jordan 分解的 Markov 方法====
R=0;
for n=1:1000
        R=R+(P*Ey)^(n-1)*P;
end
disp('理论方法结果：')
r=Ix*(R+eye(5))*Ey
disp('======================================================');
```

模拟结果：

```
======================================================
输入基础等级: 1
输入目标等级: 5
======================================================
循环方法结果：
r =
        1.7791        2.2030        2.1444        1.6675                0
======================================================
理论方法结果：
r=
        1.7791        2.2024        2.1429        1.6667                0
======================================================
```

当然如果$(PEy)^{k-1}$可逆，也可以使用逆矩阵的方法，此处不赘述。

得到r之后，用

$$C_{x\to y}=\sum_{i=1}^{N}c_i r_i = c\cdot r$$

计算最终成本即可。

QQ：707509279