云南大学数学与与统计学院

上机实践报告

课程名称：信息论基础实验	**年级：**2013	**上机实践成绩：**
指导教师：陆正福	**姓名：**金洋	
上机实践名称：信道容量迭代计算实验	**学号：**20131910023	**上机实践日期：** **2016/4/30**
上机实践编号：No. 6	**组号：**	**上机实践时间：** **8:05**

一、实验目的

（1） 熟悉信道容量的迭代算法

（2） 学习如何将复杂的公式转化为程序

（3） 掌握 Java 语言数值计算程序的设计和调试技术

二、实验内容

编程实现信道容量的迭代算法

(1) 已知：信源符号个数 n 、信宿符号个数 m ，信道转移概率矩阵 $p(y|x)$

(2) 输入：任意一个信道的转移概率矩阵，信源符号个数，信宿符号个数和每个具体的转移概率在运行时输入。

(3) 输出：最佳信源分布 $s(x)$ ，信道容量 $Capacity$

实验所用的迭代算法如下：

$$\varphi_{ji} = \frac{s_i^k p_{ij}}{\sum_{i=1}^{n} s_i^k p_{ij}}$$

$$s_i^{k+1} = \frac{\exp(\sum_{j=1}^{m} p_{ij} \log \varphi_{ji}^k)}{\sum_{i=1}^{n} \exp(\sum_{j=1}^{m} p_{ij} \log \varphi_{ji}^k)}$$

$$c^{k+1} = \log(\sum_{i=1}^{n} \exp(\sum_{j=1}^{m} p_{ij} \log \varphi_{ji}^k))$$

三、实验环境

1. 个人计算机，任意可以完成实验的平台，如 Java 平台、Python 语言、R 语言、Matlab 平台、Magma 平台等。
2. 对于信息与计算科学专业的学生，建议选择 Java、Python、R 等平台。
3. 对于非信息与计算科学专业的学生，建议选择 Matlab、Magma 等平台。

四、实验记录与实验结果分析

（注意记录实验中遇到的问题。实验报告的评分依据之一是实验记录的细致程度、实验过程的真实性、实验结果的解释和分析。**如果涉及实验结果截屏，应选择白底黑字。**）

1. 将迭代部分做成一个类：

```
package IT6;

public class ChannelCapacity {
	protected int N,M;
	protected int time;
	protected double C;
	protected double[][] p;
	protected double[] S;
	protected double[] SS;


	public ChannelCapacity() {

	}


	public ChannelCapacity(int N,int M,double [][] p) {
		this.N=N;
		this.M=M;
		this.p=p;
		S=new double[N];
		SS=new double[N];
	}

	public void compute() {


		/** 初始化数据 */
		for (int i = 0; i < N; i++)
			S[i] = (double) 1 / N;// 赋值

		/** 迭代 */
		boolean flag = true;
		double[][] phi = new double[M][N];

		while (flag) {
			/** 首先计算 phi ji */
			for (int j = 0; j < M; j++) {
				double sum = 0;
				for (int i = 0; i < N; i++) {
					sum = sum + S[i] * p[i][j];
				}
				for (int i = 0; i < N; i++)
					phi[j][i] = (S[i] * p[i][j]) / sum;
```

```
                }
                /** 迭代计算 S */

                // 计算分母
                double sum1 = 0;
                for (int i = 0; i < N; i++) {
                      boolean flag3 = true;
                      double sum2 = 0;
                      for (int j = 0; j < M; j++) {
                            if (phi[j][i] != 0)
                                  sum2 = sum2
                                              + (p[i][j] *
(Math.log(phi[j][i]) / Math
                                                                  .log(Math.E)));
                            else if (phi[j][i] == 0 && p[i][j] != 0) {
                                  flag3 = false;
                            }// exp(log0)=0,下同
                            else if (phi[j][i] == 0 && p[i][j] == 0)
                                  sum2 = sum2 + 0;// 0log0=0
                      }

                      if (flag3)
                            sum1 = sum1 + Math.exp(sum2);
                      else
                            sum1 = sum1 + 0;
                }

                /** 计算 SS[i] */

                for (int i = 0; i < N; i++) {
                      boolean flag1 = true;// 若有无穷比无穷
                      double sum6 = 0;
                      for (int j = 0; j < M; j++) {
                            if (phi[j][i] != 0)
                                  sum6 = sum6 + p[i][j]
                                              * (Math.log(phi[j][i]) /
Math.log(Math.E));
                            else if (phi[j][i] == 0 && p[i][j] != 0) {
                                  flag1 = false;
                            } else if (phi[j][i] == 0 && p[i][j] == 0)
                                  sum6 = sum6 + 0;
                      }
                      if (flag1)
                            SS[i] = Math.exp(sum6) / sum1;
                      else
                            SS[i] = 0;
                }
                double distance = 0;
                for (int i = 0; i < N; i++) {
```

```
                    distance = distance + Math.pow(SS[i] - S[i], 2);//
计算范数
                }
                if (distance < 0.00001)
                    flag = false;
                else {
                    for (int i = 0; i < N; i++)
                        S[i] = SS[i];
                }
                C = Math.log(sum1) / Math.log(2);
                time++;
            }

        }

        public void display() {
            System.out.println("迭代次数: "+time);
            System.out.print("最佳信源分布: (");
            for (int i = 0; i < N - 1; i++) System.out.printf("%5.3f,
",S[i]);
            System.out.printf("%5.3f",S[N - 1]);
            System.out.println(")");
            System.out.printf("信道容量: %8.3f\n" ,C);

        }



}
```

测试类:

```
package IT6;
import java.util.Scanner;

public class TestChannelCapacity {
    public static int N;
    public static int M;
    public static double C;

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("输入信源符号个数 n: ");
        int n= input.nextInt();
```

```
            System.out.println("输入信宿符号个数 m: ");
            int m= input.nextInt();

            System.out.println("请输入信道转移概率矩阵: ");
            double[][] p = new double[n][m];
            for (int i = 0; i < n; i++)
                  for (int j = 0; j < m; j++) {
                        p[i][j] = input.nextFloat();
                  }
            ChannelCapacity CC=new ChannelCapacity(n,m,p);
            CC.compute();
            CC.display();



      }
}
```

测试结果:

①

②

2. 对于数值计算，matlab 更合适，我们尝试使用其来完成该算法：

以下是自己所编的但是并未运行成功：

```
function [s,Capacity]=ChannelCapacity(n,m,py_x)
%Input   -n 信源符号个数
%          -m 信宿符号个数
%          -py_x n*m 矩阵，信道转移概率矩阵
%Output -s 1*n 矩阵 最佳信源分布
s=ones(1,n)/n;
C0=0;
C1=0;

while (1)
    for j=1:m
        for i=1:n
            a(j,i)=s(i)*py_x(i,j);
        end
    end

    temp=sum(a,2);
```

```
    for j=1:m
        for i=1:n
            phi(j,i)=a(j,i)/temp(i);
        end
    end




    for i=1:n
        s(i) =exp(py_x(i,:)*log2(phi(:,i)));
    end
    temp=sum(s);
    s=s./temp;

    temp1=0;
    for i=1:n
        temp2=0;
        for j=1:m
            temp2=temp2+py_x(i,j)*log2(phi(j,i));
        end
        temp1=temp1+exp(temp2);
    end
    C0=C1;C1=log2(temp1);

    if (C1-C0<1e-6)
        break;
    end

end
Capacity=C1;
```

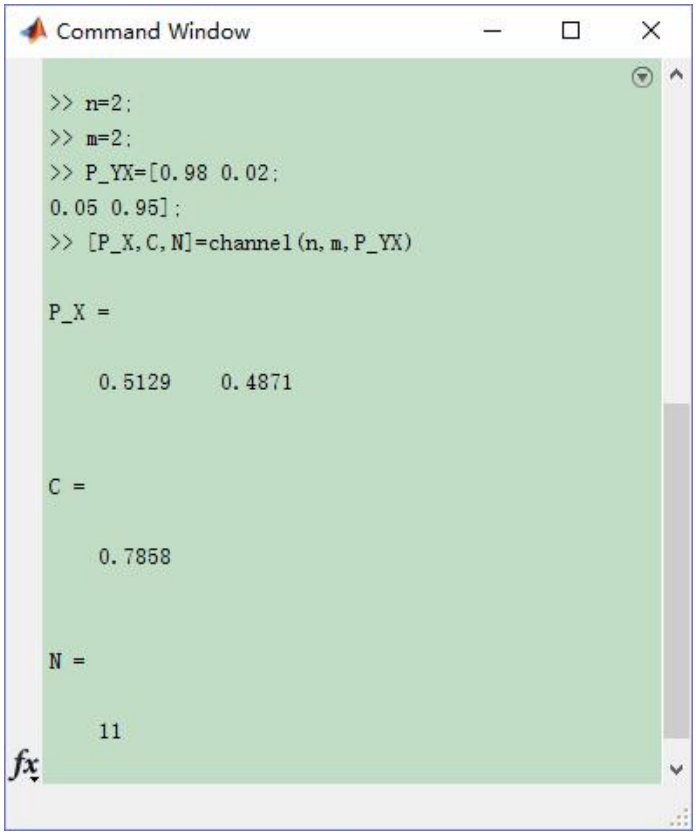
参考网上的程序，编写如下，比前两者更加精简：

Channel.java

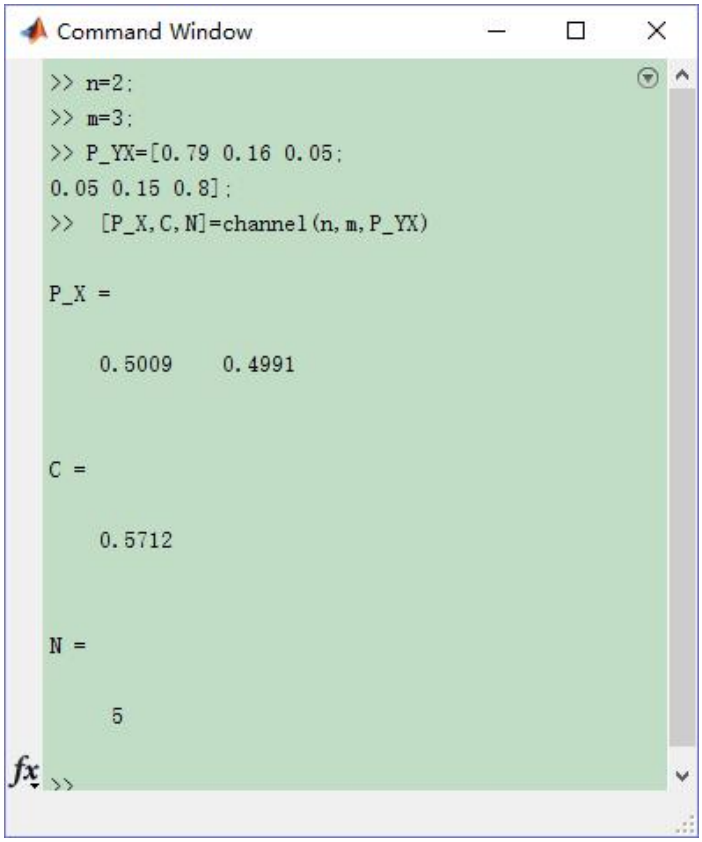
```
function [P_X,C,N]=channel(n,m,P_YX)
%Input   -n  信源符号个数
%          -m  信宿符号个数
%          -p n*m 矩阵，信道转移概率矩阵
%Output -S 1*n 矩阵  最佳信源分布
%          -Ck  信道容量
%          -N  迭代次数



e=1e-7;%停止迭代的容差限
C1=1;
C=0;
N=0;

P_X=ones(1,n)/n;

%迭代求解
while (abs(C1-C))>e
    P_Y=P_X*P_YX;

    I1=sum((P_YX.*log2(P_YX))');
    I2=log2(P_Y)*(P_YX');
    BETA=exp(I1-I2);
    B=P_X*(BETA');
    C1=log(B);C=log(max(BETA));
    P_X=P_X.*BETA/B;
    N=N+1;
end
```

测试结果：

①


Command Window
>> n=2;
>> m=2;
>> P_YX=[0.98 0.02;
0.05 0.95];
>> [P_X,C,N]=channel(n,m,P_YX)
P_X =
0.5129 0.4871
C =
0.7858
N =
11

②


Command Window
>> n=2;
>> m=3;
>> P_YX=[0.79 0.16 0.05;
0.05 0.15 0.8];
>> [P_X,C,N]=channel(n,m,P_YX)
P_X =
0.5009 0.4991
C =
0.5712
N =
5
>>

与 java 的一致；

五、实验体会

（请认真填写自己的真实体会）

1. 因为有式子 $S = R\Phi = SP\Phi$，这为求 S 提供了迭代计算的可能性；

2. 通过迭代算法可以找到一组最佳信源分布，可按照如下方式迭代：

$$\varphi_{ji} = \frac{s_i^k p_{ij}}{\sum_{i=1}^{n} s_i^k p_{ij}}$$

$$s_i^{k+1} = \frac{\exp(\sum_{j=1}^{m} p_{ij} \log \varphi_{ji}^k)}{\sum_{i=1}^{n} \exp(\sum_{j=1}^{m} p_{ij} \log \varphi_{ji}^k)}$$

$$c^{k+1} = \log(\sum_{i=1}^{n} \exp(\sum_{j=1}^{m} p_{ij} \log \varphi_{ji}^k))$$

当 $c^{(k+1)}$ 和 $c^{(k)}$ 足够接近时，则可结束迭代。

3. Java 做数值计算的时候比较不方便，不如 MATLAB 那般可以直接使用矩阵计算。但是一些式子并未提供矩阵形式，这使得有时使用 MATLAB 计算也不一定能够简化。将分式改写为矩阵形式也有一定困难，但是若能改写，则将大大简化运算。

六、参考文献

1. Thomas M. Cover, Joy A. Thomas. Elements of Information Theory (2nd Edition) [M]. John Wiley & Sons, Inc. Chapter 7

2. 郑海波. 信息论-matlab 求信道容量（迭代法） [EB/OL]. http://blog.csdn.net/nupt123456789/article/details/8243137, 2012-11-30.