上海理工大学光电信息与计算机工程学院

**《信息论与编码》实验报告**

****

**专　　业 智能科学与技术**

**姓 名　　 高浩琦**

**学　 号 13**

**年　　级 2020级**

**指导教师 胡春燕**

**成 绩：**

**教师签字：**

目录

[**实验一 信源与信息熵** 5](#_Toc123761114)

[**实验二 信源编码** 8](#_Toc123761115)

[**实验三 信道与信息率失真函数** 12](#_Toc123761116)

[**实验四 信道编码** 14](#_Toc123761117)

**实验一 信源与信息熵**

一、实验目的

1.掌握信源与信息熵相关知识

2.编程实现复现实践以加深理解

二、实验内容摘要

使用python语言，信源熵，条件互信息计算以及隐马尔可夫模型的实现。

难度系数为3，增加了应用。

三、实验步骤

def H(X):

    """

    计算信息熵

    H(X) = -sigma p(x)log p(x)

    :param X:

    :return:

    """

    x\_values = {}

    for x in X:

        x\_values[x] = x\_values.get(x, 0) + 1

    length = len(x\_values)

    ans = 0

    for x in X:

        p = x\_values.get(x) / length

        ans += p \* math.log2(p)

    return 0 - ans

def condition\_H(X, Y):

    """

    条件互信息计算

    H(X|Y) = Sigma\_Y p(y)\*H(X|Y=y)

    :param X:

    :param Y:

    :return:

    """

    y\_value = set(Y)

    y\_condition = {}

    for v in y\_value:

        y\_condition[v] = []

    for x, y in zip(X, Y):

        y\_condition[y].append(x)

    y\_counts = {}

    for y in Y:

        y\_counts[y] = y\_counts.get(y, 0) + 1

    ans = 0

    for k in y\_counts:

        ans += k \* H(y\_condition[k])

    return ans / len(y\_counts)

class HMM1:

    def \_\_init\_\_(self):

        self.A=[[0,1,0,0],[0.4,0,0.6,0],[0,0.4,0,0.6],[0,0,0.5,0.5]]       ##转移概率

        self.B=[[0.5,0.5],[0.3,0.7],[0.6,0.4],[0.8,0.2]]                   ##观测概率

        self.W=[0.25,0.25,0.25,0.25]

        ##初始状态

    def cal\_q(self,Q):

        """

        前向算法

        :param Q: 观测数据 1，0，1，0，1，0

        :return:

        """

        a=[[0 for i in range(len(self.B))] for j in range(len(Q))]

        for i in range(len(self.B)):

            a[0][i]=self.W[i]\*self.B[i][Q[0]]

        for t in range(1,len(Q)):

            for i in range(len(self.B)):

                a[t][i]=0

                for j in range(len(self.B)):

                    a[t][i]+=a[t-1][j]\*self.A[j][i]\*self.B[i][Q[t]]

        return sum(a[-1])

    def cal\_q2(self,Q):

        """

        后向算法

        :param Q: 观测数据 1，0，1，0，1，0

        :return:

        """

        b = [[0 for i in range(len(self.B))] for j in range(len(Q))]

        for i in range(len(self.B)):

            b[-1][i]=1

        for t in range(len(Q)-2,-1,-1):

            for i in range(len(self.B)):

                b[t][i]=0

                for j in range(len(self.B)):

                    b[t][i]+=b[t+1][j]\*self.A[i][j]\*self.B[j][Q[t+1]]

        q=0

        for i in range(len(self.B)):

            q+=b[0][i]\*self.W[i]\*self.B[i][Q[0]]

        return q

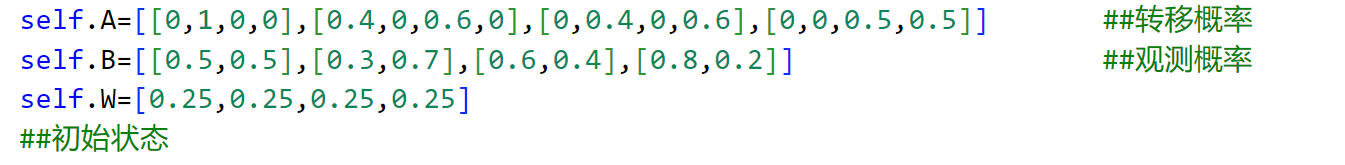
T=HMM1()

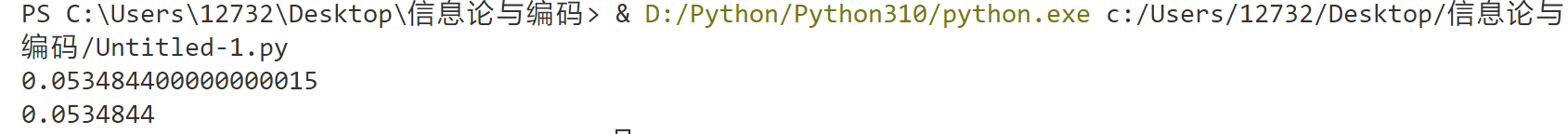
print(T.cal\_q([0,1,1,0]))

print(T.cal\_q2([0,1,1,0]))

四、实验测试

对于HMM模型来说，当其转移概率和观测概率分别为如图所示时，得出结果如下。





**实验二 信源编码**

一、实验目的

1.掌握若干个信源编码相关知识

2.编程实现复现实践以加深理解

二、实验内容摘要

采用python语言分别编程复现了，哈夫曼、香农、lzw编码

难度系数为3，增加了实践结果

三、实验步骤

Huffman

class Huffmannode(object):

    def \_\_init\_\_(self):

        self.parent=0

        self.left=0

        self.right=0

        self.weight=0

#选择最小的结点下标

def select\_node(huffman):

    #俩个结点直接返回不需要找最小俩个结点

    if len(huffman)==2:

        return 0,1

    min=semin=inf#初始化成无穷大

    f=s=-1

    for i in range(len(huffman)):

        if huffman[i].parent==0:

            if min>huffman[i].weight:

                semin=min

                s=f

                min=huffman[i].weight

                f=i

            elif semin>huffman[i].weight:

                semin=huffman[i].weight

                s=i

    return f,s

LZW编码

string =input('请输入需要压缩的字符串：')

dictionary = {chr(i): i for i in range(1, 123)}#导入ASC码进入字典1

last = 256#新的编码重第256位开始

p = ""#定义前一个字符，开始为空

result1 = []#定义一个空数组作为编码输出

for c in string:  #c为后一个字符，如果c在字符串中执行循环，执行完一次后c指向下一个字符

    pc = p + c    #将前后两个字符组成一个新的字符用pc表示

    if pc in dictionary:  #如果pc在字典中，把pc作为前一个字符

        p = pc

    else:

        result1.append(dictionary[p])  #如果pc不在字典中，把前一个字符编码输出

        dictionary[pc] = last    #把pc编码并存入字典

        last += 1

        p = c   #P指向下一个字符

if p != '':   #处理最后一个字符

   result1.append(dictionary[p])

x2 = len(result1)   #计算编码长度

print('压缩后的编码为：',result1)  #输出编码

#译码

dictionary2 = {i: chr(i) for i in range(1, 123)}  #反向导入ASC码入字典2

last2 = 256

result2 = []

p = result1.pop(0)     #把编码1给p，并从输出数组中删除

result2.append(dictionary2[p])   #把编码1译出的字符存入译码数组中

for c in result1:  #因为编码1删除了，所以c从第二个编码开始

    if c in dictionary2:

        entry = dictionary2[c]

    result2.append(entry)   #将编码译出的字符存入译码数组中

    dictionary2[last2] = dictionary2[p] + entry[0]   #将前后两个码译出的字符组成新的字符存入字典2中

    last2 += 1

    p = c

print('译码结果为：')

print(''.join(result2)) #将译码结果输出为字符串形式

x1 = len(string)  #计算输入的字符串长度

x3 = (x2\*9)/(x1\*8)  #计算压缩比

print('字符串长度：',x1)

print('编码后长度：',x2)

print('LZW的压缩比：',x3)

Shammon 编码

n = int(input("please input xinyuan sign number"))   # 强制转换类型，如若输入其他字符将会报错

# 信源符号的输入

p = [double(n) for n in input("请输入信源符号：").split()]    # 强制转换类型

# 检测输入合法性，概率大于0小于1

for i in range(n):

    if p[i] >= 1 or p[i] <= 0:

        print("输入不合法！")

        exit(0)

# 将信源符号从大到小进行排序

p.sort(reverse=True)

# 累积概率

P = [0]

for i in range(n-1):

    P.append(P[i]+p[i])    # python列表只能追加

# 码字长度

k = []

for i in range(n):

    if int(-numpy.math.log(p[i], 2)) == -numpy.math.log(p[i], 2):

        k.append(int(-numpy.math.log(p[i], 2)))

    else:

        k.append(int(1-numpy.math.log(p[i], 2)))

# 香农编码表输出

print("-------香农编码表-------")

print("i\t概率\t累积\t码字\t")

for i in range(n):

    print(str(i+1)+'\t'+str(p[i])+'\t'+str(P[i])+'\t'+str(k[i]))

# 信源熵

h = 0

for i in range(n):

    h += -p[i]\*numpy.math.log(p[i], 2)

# 平均码长

K = 0

for i in range(n):

    K += k[i]\*p[i]

# 信息率

L = 1   # 单符号

m = 2   # 二进制

R = K/L\*numpy.math.log(2, m)

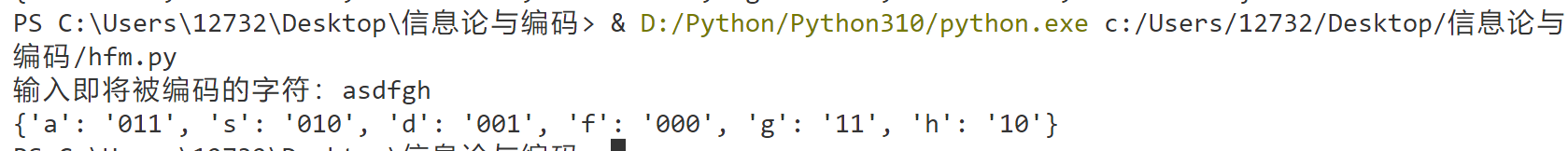
# 编码效率

e = h/R

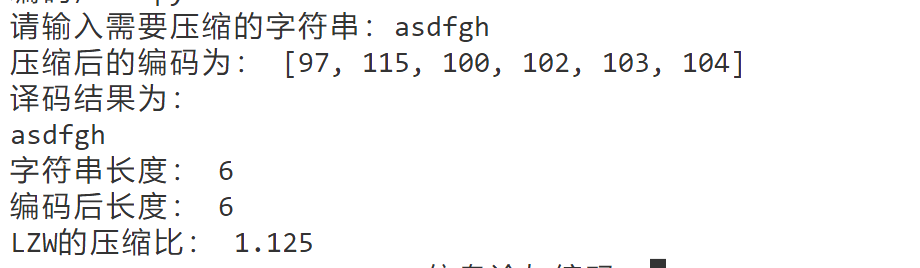
print("香农编码的编码效率="+str(round((e\*100), 2))+"%")      # 保留两位小数

四、实验测试

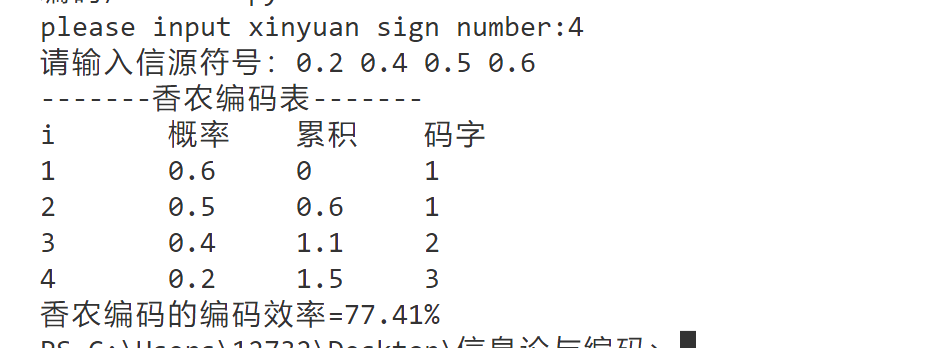
1.Huffman



2.LZW



3.Shammon



**实验三 信道与信息率失真函数**

一、实验目的

1.掌握信道与信息率失真函数相关知识

2.编程实现复现实践以加深理解

二、实验内容摘要

使用python语言，信道容量的计算的实现。

难度系数为3，增加了应用

三、实验过程记录

# coding:gbk

import numpy as np

import math

#输入信道矩阵

m = int(input("输入信道矩阵行数："))

n = int(input("输入信道矩阵列数："))

pyx = np.empty([m,n], dtype = float)

for i in range(m):

    for j in range(n):

        pyx[i][j] = float(input())

print("信道矩阵是\n",pyx)

print(pyx.shape)

e = float(input("输入精度："))

px = np.empty([1,m], dtype = float)

for i in range(m):

    px[0][i] = 1.0/m

py = np.empty([1,n], dtype = float)

py = np.dot(px,pyx)

pxy = np.empty([m,n], dtype = float)

IU =10

IL = 0

b = np.empty([m,n], dtype = float)

while abs(IU-IL)>e:

    for i in range(m):

        for k in range(n):

            a = pyx[i][k]\*math.log(pyx[i][k]/py[0][k])

            b[i][k] = float(a)

    F = np.exp(b.sum(axis=1))

    h = np.vdot(F,px)

    x = sum(F)

    IL = math.log(h,2)

    g = max(F)

    IU = math.log(g,2)

    for i in range(m):

        px[0][i] = px[0][i] \* F[i] / h

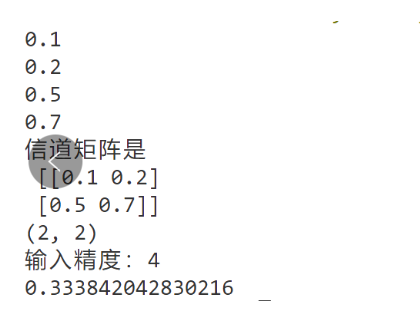
    py = np.dot(px,pyx)

    if abs(IU-IL)<=e:

        print(IL)

        break

四、实验测试



**实验四 信道编码**

一、实验目的

1.掌握信道编码相关知识

2.编程实现复现实践以加深理解

二、实验内容摘要

使用python语言，进行了卷积码、汉明码、循环码的实现。

难度系数为3，增加了应用。

三、实验步骤

**卷积码：**

def encode\_conv(x):

    # 存储编码信息

    y = []

    # 两个寄存器u\_1 u\_2初始化为0

    u\_2 = 0

    u\_1 = 0

    for j in x:

        c\_1 = (j + u\_1 + u\_2)%2

        c\_2 = (j+u\_2)%2

        y.append(c\_1)

        y.append(c\_2)

        # 更新寄存器

        u\_2 = u\_1

        u\_1 = j

    print(x,"编码为：",y)

    return y

**循环码：**

def encode\_cyclic(x):

    if not len(x) == 4:

        print("请输入4位信息比特")

        return

    y = np.dot(x,G)

    print(x,"编码为：",y)

    return y

# 译码，过程与汉明码一致

def decode\_cyclic(y):

    if not len(y) == 7:

        print("请输入7位信息比特")

        return

    x\_tmp = np.dot(y,H.T)%2

    if (x\_tmp!=0).any():

        for i in range(H.shape[1]):

            if (x\_tmp==H[:,i]).all():

                y[i] = (y[i]-1)%2

                break

    x = y[0:4]

    print(y,"解码为：",x)

    return x

**汉明码：**

def func(X):

    '''

    返回G \* H\_T

    '''

    H = np.array(

        [[0, 0, 0, 1, 1, 1, 1],

         [0, 1, 1, 0, 0, 1, 1],

         [1, 0, 1, 0, 1, 0, 1]])

    G1 = np.eye(4)

    G2 = [[X[0], X[1], X[2]]

        , [X[3], X[4], X[5]]

        , [X[6], X[7], X[8]]

        , [X[9], X[10], X[11]]]

    G = np.concatenate((G1, G2), axis=1)

    GH\_T = np.dot(G, np.transpose(H))

    m, n = GH\_T.shape[0], GH\_T.shape[1]

    for i in range(m):

        for j in range(n):

            GH\_T[i][j] %= 2                     #不进位

    GH\_T = np.array(GH\_T)

    GH\_T = GH\_T.reshape(1,12)

    return GH\_T

def shuzu(j,len):

    '''

    将字符串转换为数组

    '''

    C = np.zeros((len,))

    for k in range(0,len):

        C[k] = int(j[k])

    return C

def func2(e):

    '''

    返回伴随式s = e \* H\_T

    '''

    H = np.array(

        [[0, 0, 0, 1, 1, 1, 1],

         [0, 1, 1, 0, 0, 1, 1],

         [1, 0, 1, 0, 1, 0, 1]])

    s\_ = np.dot(e,np.transpose(H))

    k = s\_.shape[0]

    for i in range(k):

        s\_ %= 2             #不进位

    return s\_

def siranguji(e\_dataset):

    '''

    求每一个满足条件的e的概率

    '''

    p\_mat = np.array([[6/7,1/7]

                  ,[1/7,6/7]])

    P\_ = []

    length = len(e\_dataset)

    for i in range(0, length):

        P = 1

        for j in range(0,7):

            if e\_dataset[i][j] == 1:

                P = P \* p\_mat[0][1]

            elif e\_dataset[i][j] == 0:

                P = P \* p\_mat[0][0]

        P\_.append(P)

    return P\_

四、实验测试

