

云南大学数学与统计学院
上机实践报告

课程名称：信息论基础实验	年级：2015 级	上机实践成绩：
指导教师：陆正福	姓名：刘鹏	
上机实践名称：熵的计算编程实验	学号：20151910042	上机实践日期：2017-11-16
上机实践编号：No.02	组号：	上机实践时间：0:13

一、实验目的

1. 给定分布，计算熵；
2. 给定原始数据，计算熵。

二、实验内容

1. 给定二维分布函数，计算联合熵、条件熵、互信息、各变量的熵。可选择课本例题 2.2.1 作为程序测试用例。
2. 自行设定原始数据（如一段文本、一幅图像、一个数据表等），按照频率计算符号的分布，进而计算有关的熵。

三、实验平台

Windows 10 1703 Enterprise 中文版；
Python 3.6.0；
Wing IDE Professional 6.0.5-1 集成开发环境。

四、实验记录与实验结果分析

1 题

给定如下分布，计算该概率密度分布的熵。

设 (X, Y) 服从如下的联合分布：^[1]

$X \backslash Y$	1	2	3	4
1	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{32}$
2	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{32}$	$\frac{1}{32}$
3	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$
4	$\frac{1}{4}$	0	0	0

X 的边际分布为 $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\right)$ ， Y 的边际分布为 $\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)$ 。计算 $H(X, Y)$ ， $H(X|Y)$ ， $H(Y|X)$ ， $H(X|y=3)$ ， $I(X; Y)$ 。

解答：

$$H(X, Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log(p(x, y))$$

$$H(X|Y) = - \sum_{y \in Y} P(y) \sum_{x \in X} P(x|y) \log(P(x|y))$$

$$I(X; Y) = H(X) - H(Y|X)$$

程序代码:

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Sun Dec 10 21:52:41 2017
4
5  @author: Newton
6  """
7
8  """filename: Calculate.py"""
9
10 from math import log2 as log2
11
12 class getEntropy:
13     """This class aims to get the related entropy of a distribution.
14
15     The distribution should contain 3 or 2 inputs.
16
17     if 2:
18         A distribution
19     if 3:
20         A joint distribution
21
22     +----+----+
23     |    | X |
24     +----+----+
25     | Y  |   |
26     +----+----+
27
28     """
29     def __init__(self, x_value, dim = 1):
30         """H(X) = sum(p * log(p))"""
31         self.distribution = x_value # x's distribution
32         self.dim = dim
33         if dim not in {1, 2}:
34             raise ValueError("dimension wrong!")
35
36     def entropy(self, which='X'):
37         """Calculate the Entropy."""
38         ans = 0
39
40         if which == 'X':
41             for i in range(len(self.distribution)):
42                 ans += -1 * self.distribution[i] * log2(self.distribution[i])

```

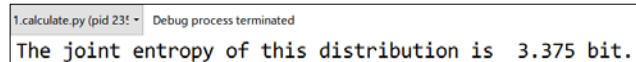
```

43         return ans
44
45     if which == 'Y':
46         if self.dim != 2:
47             raise ValueError("No Y distribution is input!")
48         row = len(self.distribution)
49         column = len(self.distribution[0])
50
51         distribution_y = list()
52
53         for i in range(column):
54             tmp = list()
55             for j in range(row):
56                 tmp.append(self.distribution[j][i])
57             distribution_y.append(sum(tmp))
58
59         for i in range(len(distribution_y)):
60             ans += -1 * distribution_y[i] * log2(distribution_y[i])
61         return ans
62
63     if which == 'joint':
64         tmp = list() # to contain all the elements
65         if self.dim != 2:
66             raise ValueError("No Y distribution is input!")
67         for i in range(len(self.distribution)):
68             for j in range(len(self.distribution[0])):
69                 tmp.append(self.distribution[i][j])
70         for i in tmp:
71             if i == 0:
72                 log_tmp = 0
73             else:
74                 log_tmp = log2(i)
75             ans += -1 * i * log_tmp
76         return ans
77
78     def condEntropy(self, which='X|Y'):
79         """Default: H(X|Y)"""
80         ans = 0
81
82         if which == "X|Y":
83             y_distribution = list()
84             tmp = 0
85             for i in range(len(self.distribution)):
86                 for j in range(len(self.distribution[0])):
87                     tmp += self.distribution[i][j]
88             y_distribution.append(tmp)
89
90             for i in range(len(self.distribution[0])):
91                 tmp = 0

```

```
92         for j in range(len(self.distribution)):
93             tmp_cond = y_distribution[i] * self.distribution[i][j]
94             if tmp_cond == 0:
95                 cond_partial = 0
96             else:
97                 cond_partial = tmp_cond * log2(tmp_cond)
98             tmp += cond_partial
99
100         tmp *= y_distribution[i]
101         ans += tmp
102     return ans
103
104 if __name__ == "__main__":
105     tmp = [[ 1/8, 1/16, 1/32, 1/32],\
106            [1/16, 1/8, 1/32, 1/32],\
107            [1/16, 1/16, 1/16, 1/16],\
108            [ 1/4, 0, 0, 0]]
109
110     c = getEntropy(tmp, 2)
111     d = c.condEntropy("X|Y")
112     e = c.entropy("joint")
113     print("The joint entropy of this distribution is ", e, "bit.")
114     #print("The conditional entropy of this distribution is ", d, "bit.")
```

运行结果:



```
1.calculate.py (pid 231 - Debug process terminated)
The joint entropy of this distribution is 3.375 bit.
```

代码分析:

这段程序是利用二维数组来进行联合熵的计算。

2 题

选择一幅 RGB 彩色图片，进行熵的计算。图片分辨率自选，要求用上 1 题的代码。

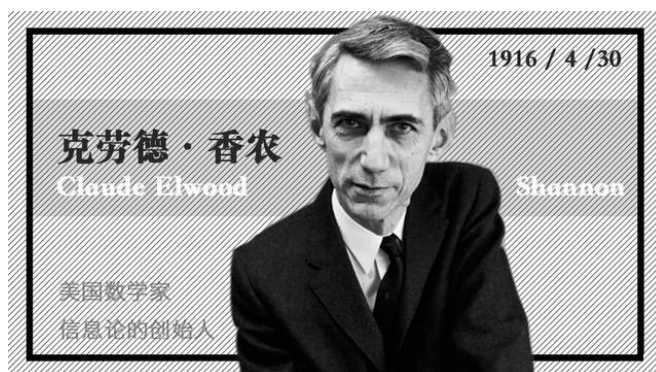
解答：

程序代码：

运行结果：

代码分析：

在这段代码里，为了向信息论鼻祖香农致敬，采用了它的一张图片进行熵的计算。由于香农生时，彩色照片还没有普及，所以这里采用了一张灰度图像。



计算需要调用之前用过的代码。核心思想就是把图片转为灰度矩阵，然后计算矩阵中的元素的频率。并且记录矩阵的行列数目，即图片分辨率。这样就可以了。

五、教材翻译

六、实验体会

经过这个实验，更加深刻地理解了熵的计算方法。以及相对熵与条件熵的计算方法。

七、参考文献

[1] Cover TM, Thomas JA. Elements of Information Theory[M]. Canada: John Wiley & Sons, I,c.; 2006.