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。

设服从如下的联合分布：[1]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

的边际分布为，的边际分布为。计算，，，，。

**解答：**

****

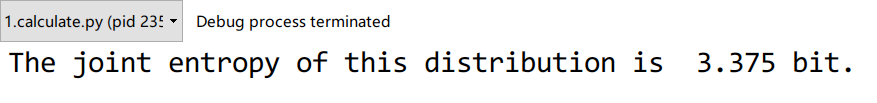
****

****

程序代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114 | # -\*- coding: utf-8 -\*-  """  Created on Sun Dec 10 21:52:41 2017  @author: Newton  """  """filename: Calculate.py"""  **from** math **import** log2 **as** log2  **class** **getEntropy:**  """This class aims to get the related entropy of a distribution.    The distribution should contain 3 or 2 inputs.    if 2:  A distribution  if 3:  A joint distribution    +----+----+  | | X |  +----|----+  | Y | |  +----+----+    """  **def** \_\_init\_\_**(**self**,** x\_value**,** dim **=** 1**):**  """H(X) = sum(p \* log(p))"""  self**.**distribution **=** x\_value # x's distribution  self**.**dim **=** dim  **if** dim **not** **in** **{**1**,** 2**}:**  **raise** ValueError**(**"dimension wrong!"**)**    **def** entropy**(**self**,** which**=**'X'**):**  """Calculate the Entropy."""  ans **=** 0    **if** which **==** 'X'**:**  **for** i **in** range**(**len**(**self**.**distribution**)):**  ans **+=** **-**1 **\*** self**.**distribution**[**i**]** **\*** log2**(**self**.**distribution**[**i**])**  **return** ans    **if** which **==** 'Y'**:**  **if** self**.**dim **!=** 2**:**  **raise** ValueError**(**"No Y distribution is imput!"**)**  row **=** len**(**self**.**distribution**)**  column **=** len**(**self**.**distribution**[**0**])**  distribution\_y **=** list**()**  **for** i **in** range**(**column**):**  tmp **=** list**()**  **for** j **in** range**(**row**):**  tmp**.**append**(**self**.**distribution**[**j**][**i**])**  distribution\_y**.**append**(**sum**(**tmp**))**  **for** i **in** range**(**len**(**distribution\_y**)):**  ans **+=** **-**1 **\*** distribution\_y**[**i**]** **+** log2**(**distribution\_y**)**  **return** ans    **if** which **==** 'joint'**:**  tmp **=** list**()** # to contain all the elements  **if** self**.**dim **!=** 2**:**  **raise** ValueError**(**"No Y distribution is imput!"**)**  **for** i **in** range**(**len**(**self**.**distribution**)):**  **for** j **in** range**(**len**(**self**.**distribution**[**0**])):**  tmp**.**append**(**self**.**distribution**[**i**][**j**])**  **for** i **in** tmp**:**  **if** i **==** 0**:**  log\_tmp **=** 0  **else:**  log\_tmp **=** log2**(**i**)**  ans **+=** **-**1 **\*** i **\*** log\_tmp  **return** ans    **def** condEntropy**(**self**,** which**=**'X|Y'**):**  """Default: H(X|Y)"""  ans **=** 0  **if** which **==** "X|Y"**:**  y\_distribution **=** list**()**  tmp **=** 0  **for** i **in** range**(**len**(**self**.**distribution**)):**  **for** j **in** range**(**len**(**self**.**distribution**[**0**])):**  tmp **+=** self**.**distribution**[**i**][**j**]**  y\_distribution**.**append**(**tmp**)**  **for** i **in** range**(**len**(**self**.**distribution**[**0**])):**  tmp **=** 0  **for** j **in** range**(**len**(**self**.**distribution**)):**  tmp\_cond **=** y\_distribution**[**i**]** **\*** self**.**distribution**[**i**][**j**]**  **if** tmp\_cond **==** 0**:**  cond\_partial **=** 0  **else:**  cond\_partial **=** tmp\_cond **\*** log2**(**tmp\_cond**)**  tmp **+=** cond\_partial    tmp **\*=** y\_distribution**[**i**]**  ans **+=** tmp  **return** ans  **if** \_\_name\_\_ **==** "\_\_main\_\_"**:**  tmp **=** **[[** 1**/**8**,** 1**/**16**,** 1**/**32**,** 1**/**32**],**\  **[**1**/**16**,** 1**/**8**,** 1**/**32**,** 1**/**32**],**\  **[**1**/**16**,** 1**/**16**,** 1**/**16**,** 1**/**16**],**\  **[** 1**/**4**,** 0**,** 0**,** 0**]]**    c **=** getEntropy**(**tmp**,** 2**)**  d **=** c**.**condEntropy**(**"X|Y"**)**  e **=** c**.**entropy**(**"joint"**)**  **print(**"The joint entropy of this distribution is "**,** e**,** "bit."**)**  #print("The conditional entropy of this distribution is ", d, "bit.") |

运行结果：



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。

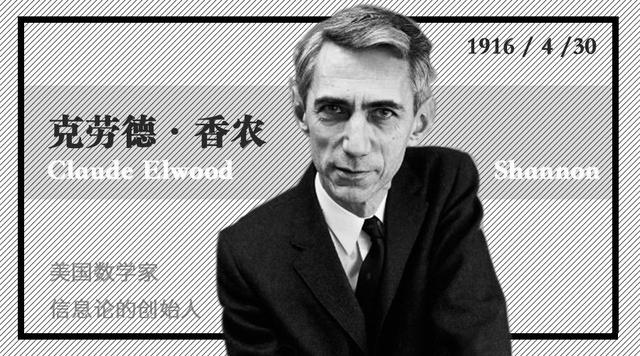
**解答：**

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意，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.