第二章的计算问题

音频文件计算

学习内容：(第二章课件PPT) p13：

以字节为单位，模拟波形声音被数字化后音频文件的存储量(假定未经压缩)为：

**存储量=采样频率×量化位数/8×声道数×时间**

　 课件上的例题：用44.1KHz的采样频率进行采样，量化位数选用16位，则录制1秒的立体声节目，其波形文件所需的存储量为：

　 44100×16／8×2×1=176400(字节)

**注意：**公式的变换计算，如果给出了其中任意的4个量，那么就能够计算出第5个量的值。

第三章的计算问题

信息熵信息量计算及压缩编码过程

本章教学要求学生：掌握信息熵的概念、计量及意义；掌握霍夫曼编码、LZ77算法、LZSS算法编码的基本步骤。

3.0 信息熵信息量计算

学习内容：(第三章课件PPT) p16—20 [这里略，请详见课件]

课件上的例题：40个像素组成的灰度图象，灰度为5级，ABCDE，出现每个灰度的像素个数不同，为：15、7、7、6、5，该图象的熵为多少？存储这40个像素需要的存储位数。

解：H（s）=(15/40)log2(40/15)+…=2.196；

40个像素需40×2.196=87.84位

说明：（1）灰度为5级，即有5个不同的灰度像素15、7、7、6、5，其和为40（个像素），所以使用了ppt上p18的公式

3.1 香农—范诺

Shannon-Fano的树是根据旨在定义一个有效的代码表的规范而建立的。实际的算法很简单：

1. 对于一个给定的符号列表，制定了概率相应的列表或频率计数，使每个符号的相对发生频率是已知。
2. 排序根据频率的符号列表，最常出现的符号在左边，最少出现的符号在右边。
3. 清单分为两部分，使左边部分的总频率和尽可能接近右边部分的总频率和。
4. 该列表的左半边分配二进制数字0，右半边是分配的数字1。这意味着，在第一半符号代都是将所有从0开始，第二半的代码都从1开始。
5. 对左、右半部分递归应用步骤3和4，细分群体，并添加位的代码，直到每个符号已成为一个相应的代码树的叶。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **符号** | **A** | **B** | **C** | **D** | **E** |
| 计数 | 15 | 7 | 6 | 6 | 5 |
| 概率 | 0.38461538 | 0.17948718 | 0.15384615 | 0.15384615 | 0.12820513 |

从左到右，所有的符号以它们出现的次数划分。在字母B与C之间划定分割线，得到了左右两组，总次数分别为22,17。这样就把两组的差别降到最小。通过这样的分割, A与B同时拥有了一个以0为开头的码字, C，D，E的码子则为1,如图b所示。随后，在树的左半边，于A，B间建立新的分割线，这样A就成为了码字为00的叶子节点，B的码子01。经过四次分割，得到了一个树形编码。如下表所示，在最终得到的树中，拥有最大频率的符号被两位编码，其他两个频率较低的符号被三位编码。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **符号** | **A** | **B** | **C** | **D** | **E** |
| 编码 | 00 | 01 | 10 | 110 | 111 |

根据A，B，C两位编码长度，D，E的三位编码长度，最终的平均码字长度是

\frac{2\,\text{Bit}\cdot(15+7+6) + 3\,\text{Bit} \cdot(6+5)}{39\, \text{Symbol}} \approx 

2.28\,\text{Bits per Symbol.}

3.2**霍夫曼编码**

教科书p67例2-14的解答过程有部分错误，这里重新讲解一下并补充树型图方法编码过程。

学习内容[这里略，请详见课件和教科书及网络资料]：

1.(第三章课件PPT) p25—28 2.教科书p67--68

课件p25:

霍夫曼编码采用从下到上的方法。

具体步骤：

（1）将事件出现的概率按由大到小的顺序排序

（2）合并概率最小的两个事件

（3）排序

（4）如果事件个数大于2则重复（2）和（3）

（5）赋值

（6）编码

说明：

p67例2-14题使用了如P68“图2-36霍夫曼编码过程”的图形方法没有错误，但计算过程出现了错误。

错误1，P67 最下面一行的计算式中乘以了7，应该是6，应将结果3.26改为2.98

错误2，P68 “信号源的信息熵为：”的计算式中负号后要全部用括号括起来，且将后面的乘以7改为乘以6，将结果3.24改为2.97

错误3，由于上述两个错误导致了“编码效率”计算需要修改，结果为99.66%

注意霍夫曼编码过程也可以使用树型图描述，书写起来更方便，教科书和课件上都没有涉及，这里讲解如下：

为了书写方便，将书上例题的待编码的符号写成X={e1,e2,e3,e4,e5,e6,e7,e8,e9}，其对应的概率p={0.22、0.22、0.14、0.07、0.07、0.07、0.07、0.07、0.07}

根据上述编码步骤，始终合并两个最小概论的事件。

观察X中的事件概率，最小的是事件e4、e5、e6、e7、e8、e9，将两两进行合并，即e4与e5合并，e6与e7合并，e8与e9合并 【这几个事件的概率一样大，所以无需排序了】

合并后的结果如下图：

将合并后的新的概率事件命名为 X1 、X2 、 X3

这时可以将待编码的符号和对应的概率改写成下列形式：

X={e1,e2,e3,x1,x2,x3} p={0.22、0.22、0.14、0.14、0.14、0.14}

观察它们的事件概率，最小的是事件e3、x1、x2、x3，将两两进行合并，即e3与x1合并，x2与x3合并，结果如下图：

将合并后的新的概率事件命名为 y1 、y2

这时可以将待编码的符号和对应的概率改写成下列形式：

X={ y1,y2,e1,e2} p={0.28、0.28、0.22、0.22} [注意由大到小排序]

观察它们的事件概率，最小的是事件e1和e2，将它们合并，结果如下图：

将合并后的新的概率事件命名为z1

这时可以将信号源和对应的概率改写成下列形式：

X={ z1,y1,y2} p={0.44、0.28、0.28}

观察它们的事件概率，最小的是事件y1和y2，将它们合并，结果如下图：

将合并后的新的概率事件命名为m

这时可以将待编码的符号和对应的概率改写成下列形式：

X={m, z1} p={0.56、0.44}

直接将它们合并，结果如下图：

最后，将上述图形合并到一起，得到如下图型：

将各个事件用概率表示，得到如下图型：

图型中，（1）叶子节点【即没有分支的节点】的值就是每个事件对应的概率值

（2）每个节点中，数值大的子节点排放在左边，标记为1，数值小的子节点排放在右边，标记为0；如果两个子节点的值一样大小即分不出大小的，则图型左边的为1，右边的为0.

（3）画最后的树型图时，总的根节点的值一定为1，否则计算过程有误。

对照上述两个图型可以得到每个事件的编码

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| e1 | e2 | e3 | e4 | e5 | e6 | e7 | e8 | 9e |
| 01 | 00 | 111 | 1101 | 1100 | 1011 | 1010 | 1001 | 1000 |

举例说明：符号e1的编码，在上述符号标识的图中可以看出其经历的是从z1到e1，z1在根节点1.00的右边，其标记为0，e1在节点z1的左边，其标记为1，所以e1的编码为01。同理，e2的编码为00

符号e4的编码，在上述符号标识的图中可以看出其经历的是从m—y1----x1---e4，m在根节点1.00的左边，其标记为1，y1在节点m的左边，其标记为1，x1在节点y1的右边，其标记为0，e4在节点x1的左边，其标记为1，所以e4的编码为1101。同理，e5的编码为1100

其它的各符号的编码也同理可得。

**3.3 LZ77编码过程**

这部分内容书本上没有，将课件上的内容详细讲解如下。(第三章课件PPT) p35—38

P36 **LZ77编码的基本流程**

步骤1）、从当前压缩位置开始，考察未编码的数据，并试图在滑动窗口中找出最长的匹配字符串，如果找到，则进行步骤 2，否则进行步骤 3。

步骤2）、输出三元符号组 ( off, len, c )。其中 off 为窗口中匹配字符串相对窗口边界的偏移，len 为可匹配的长度，c 为下一个字符，即不匹配的第一个字符。然后将窗口向后滑动 len + 1 个字符，继续步骤 1。

步骤3）、输出三元符号组 ( 0, 0, c )。其中 c 为正在考察的未匹配字符。然后将窗口向后滑动 1 个字符，继续步骤 1。

结合实例来讲解。

例题1：(第3章课件PPT38) 假设窗口中输入的数据流如下表：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 位置 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 字符 | A | A | B | C | B | B | A | B | C | A |

编码过程分析如下：

开始时，第1列位置上的字符是’A’, 它的左边没有与它相同的字符串[因为是第1列字符]，因此跳转到步骤3，off=0,len=0, 输出(0,0,A),这个括号里的’A’是上表中第1列的’A’；

接着回到步骤1，向右滑动窗口至下一个字符’A’[即第2列字符]，它与左边第1列字符相同，而第2列与后续的第3列’B’组成的字符串’AB’在前面没有相同的字符串了，因此这一步结束向右移动字符，跳转到步骤2, 第2列与第1列的位置偏移值为1，匹配字符串’A’长度为1， off=1,len=1, 输出(1,1,B) ,这个括号里的’B’是上表中第3列的’B’；

接着回到步骤1，向右滑动窗口至下一个字符’C’[第4列字符]，它与左边的字符串中没有相同的字符，因此跳转到步骤3, off=0,len=0, 输出(0,0,C) ,这个括号里的’C’是上表中第4列的’C’；

接着回到步骤1，向右滑动窗口至下一个字符’B’[第5列字符]，同理，它前面第2个[即上表中第3列字符]与它相同，而当前’B’ [第5列字符]后续的第6列字符是’B’，组成的字符串’BB’在前面的字符串中没有相同的字符串，因此,这一步结束向右移动字符，跳转到步骤2,第5列与第3列的位置偏移值为2，匹配字符串’B’长度为1，得 off=2,len=1, 输出(2,1,B) ,这个括号里的’B’是上表中第6列的’B’；

接着回到步骤1，向右滑动窗口至下一个字符’A’[第7列字符]，它与后续第8、9列字符组成字符串’ABC’,与前面第2、3、4列组成的字符串相同，第7列与第2列的位置偏移值为5，匹配字符串’ABC’长度为3，因此, off=5,len=3, 输出(5,3,A)，编码结束。

将上述从开始到结束的步骤填入下表：

|  |  |  |  |
| --- | --- | --- | --- |
| **步骤** | **位置** | **匹配串** | **输出** |
| **1** | **1** | **-** | **（0,0,A）** |
| **2** | **2** | **A** | **（1,1,B）** |
| **3** | **4** | **-** | **(0,0,C)** |
| **4** | **5** | **B** | **（2,1,B）** |
| **5** | **7** | **ABC** | **（5,3,A）** |

例题2：假设窗口中输入的字符流为：abaeaaabac

用表格表示：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| a | b | a | e | a | a | a | b | a | c |

编码过程分析如下：

开始时，第1列位置上的字符是’a’, 它的左边没有与它相同的字符串[因为这是第1列字符]，因此跳转到步骤3，off=0,len=0, 输出(0,0,a) ,这个括号里的’a’是上表中第1列的’a’；

接着回到步骤1，向右滑动窗口至下一个字符’b’[即第2列字符]，这个’b’的左边没有与它相同的字符串[因为这时’b’的左边只有’a’]，因此跳转到步骤3, off=0,len=0, 输出(0,0,b) ,这个括号里的’b’是上表中第2列的’b’；

接着回到步骤1，向右滑动窗口至下一个字符’a’[第3列字符]，发现在这个’a’的左边第2个字符[第1列的字符]与它相同，接着从第3列字符’a’开始向右找，看一看有没有连续的字符串与它前面的字符串相同，找到最大长度的相同字符串为止。发现第3个字符之后是’e’[第4列字符]而与前面’a’的后续字符’b’不同[上表中第2列字符]，因此，这一步结束向右移动字符了，跳转到步骤2, 第3列与第1列的位置偏移值为2，匹配字符串’a’长度为1， off=2,len=1, 输出(2,1,e) ,这个括号里的’e’是上表中第4列的’e’；

接着回到步骤1，向右滑动窗口至下一个字符’a’[第5个字符]，同理，它前面第2个[即上表中第3列字符]与它相同，而当前’a’的后续字符是’a’[第6列字符]与第4列字符’e’不同，因此, 这一步结束向右移动字符了，跳转到步骤2，第5列与第3列的位置偏移值为2，匹配字符串’a’长度为1，off=2,len=1, 输出(2,1,a) ,这个括号里的’a’是上表中第6列的’a’；

接着回到步骤1，向右滑动窗口至下一个字符’a’[第7列个字符]，以及第8列、第9列字符’b’和’a’,组成字符串’aba’与第1、2、3位置的字符相同，满足相同长度最大的要求，因此, 跳转到步骤2，第7列与第1列的位置偏移值为6，匹配字符串’aba’长度为3，off=6,len=3, 输出(6,3,c)，编码结束。

将上述从开始到结束的步骤填入下表：

|  |  |  |  |
| --- | --- | --- | --- |
| **步骤** | **位置** | **匹配串** | **输出** |
| **1** | **1** | **-** | **（0,0,a）** |
| **2** | **2** | **-** | **（0,0,b）** |
| **3** | **3** | **a** | **(2,1,e)** |
| **4** | **5** | **a** | **(2,1,a）** |
| **5** | **7** | **aba** | **（6,3,c）** |

例3：假设窗口中输入的字符流为：ABBCABBBD

用表格表示：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| A | B | B | C | A | B | B | B | D |

编码过程分析如下：

开始时，第1列位置上的字符是’A’, 它的左边没有与它相同的字符串[因为这是第1列字符]，因此跳转到步骤3，off=0,len=0, 输出(0,0,A),这个括号里的’A’是上表中第1列的’A’；

接着回到步骤1，向右滑动窗口至下一个字符’B’[即第2列字符]，左边没有与它相同的字符串[因为这时’B’的左边只有’A’]，因此跳转到步骤3, off=0,len=0, 输出(0,0,B) ,这个括号里的’B’是上表中第2列的’B’；

接着回到步骤1，向右滑动窗口至下一个字符’B’[第3列字符]，它的左边第1列字符与它相同，因此跳转到步骤2, 第3列与第2列的位置偏移值为1，匹配字符串’B’长度为1， off=1,len=1, 输出(1,1,C) ,这个括号里的’C’是上表中第4列的’C’；

接着回到步骤1，向右滑动窗口至下一个字符’A’[第5个字符]，以及与第6、7列字符’BB’组成的字符串’ABB’与前面第1、2、3列组成的字符串相同，因此, 第5列与第1列的位置偏移值为4，匹配字符串’ABB’长度为3，off=4,len=3, 输出(4,3,B) ,这个括号里的’B’是上表中第8列的’B’；

接着回到步骤1，向右滑动窗口至下一个字符’D’[第9列字符]，它与前面没有相同字符，因此, off=0,len=0, 输出(0,0,D)，编码结束。

将上述从开始到结束的步骤填入下表：

|  |  |  |  |
| --- | --- | --- | --- |
| **步骤** | **位置** | **匹配串** | **输出** |
| **1** | **1** | **-** | **（0,0,A）** |
| **2** | **2** | **-** | **（0,0,B）** |
| **3** | **3** | **B** | **(1,1,C)** |
| **4** | **5** | **ABB** | **（4,3,B）** |
| **5** | **9** | **-** | **（0,0,D）** |

**3.4 LZSS编码过程**

这部分内容书本上没有，将课件上的内容详细讲解如下。(第三章课件PPT) p39—41

LZSS算法思想：

if (匹配串长度 > 指针长度) then

输出指针

else

输出真实字符。

LZSS编码主要是解决LZ77中不足：（1）空指针；（2）编码器可能输出额外的字符。

**LZSS编码过程**

步骤1） 从当前压缩位置开始，考察未编码的字符，并试图在滑动窗口中找出最长的匹配字符串，如果匹配串长度len大于等于最小匹配串长度（len >= MIN\_LENGTH），则进行步骤 2，否则进行步骤 3。

步骤2） 输出指针二元组 ( off, len)。其中 off 为窗口中匹配字符串相对窗口边界的偏移，len 为匹配串的长度，然后将窗口向后滑动 len 个字符，继续步骤 1。

步骤3） 输出当前字符c，然后将窗口向后滑动 1 个字符，继续步骤 1。

LZSS编码举例：

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **位置** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
| **字符** | **A** | **A** | **B** | **B** | **C** | **B** | **B** | **A** | **A** | **B** | **C** |

编码过程 MIN\_LEN =2

|  |  |  |  |
| --- | --- | --- | --- |
| **步骤** | **位置** | **匹配串** | **输出** |
| **1** | **1** | **－－** | **A** |
| **2** | **2** | **A** | **A** |
| **3** | **3** | **－－** | **B** |
| **4** | **4** | **B** | **B** |
| **5** | **5** | **－－** | **C** |
| **6** | **6** | **BB** | **（3，2）** |
| **7** | **8** | **AAB** | **（7，3）** |
| **8** | **11** | **C** | **C** |

**通过完成第三章课件最后的作业题，加深对上述内容的掌握。**

第四章的计算问题

学习内容：(第三章课件PPT) p35：

用字节表示图像文件大小时，一幅未经压缩的数字图像的数据量大小计算如下：

**图像数据量大小 = 像素总数×图像深度÷8**

课件上的例题：一幅屏幕 640×480 的 256 色图像大小

640×480×8／8 = 307200 字节

**注意：（1）**公式中是计算“像素总数”的图像大小值，也可以计算出图像的单个像素的大小值，即 单个像素的大小值=图像深度÷8 ，该例题中单个像素的大小值=8÷8=1字节

**（2）**像素总数即为该图像的水平与垂直方向的像素个数，上例中的640×480；

特别要记住的是图像深度与图像色彩数的对应关系[参见第三章课件P33]，公式中用的是“图像深度”数而不是“图像颜色”数！例如，颜色色数为16的图像深度是4、颜色色数为256的图像深度是8、颜色色数为65536的图像深度是16

另外，公式计算得到图像大小的是“字节”，最好要转换成“KB”或“MB”[千字节、兆字节]等