霍夫曼编码实验报告

杨坤泽
2023210799

2023 年 12 月 18 日

1 问题描述与基本原理

根据要求分析，发现不同的问题都需要实现在给定概率分布下的霍夫曼编码。因此考虑代码结构，可以将整体框架分为概率分布生成与霍夫曼编码两个函数实现。

对于问题一的概率分布为伯努利分布，对于问题二其概率分布则为泊松分布。而考虑到霍夫曼编码的具体实现，基本原理为在每次编码时先合并两个概率分布最小的项。为了方便处理，可以将概率分布进行排序，每次合并前两项；需要注意的是不仅要合并概率值，还需要合并下标值并记录。在 Python 中可以利用列表对下标值记录和合并，将合并后的列表整体以及概率分别重新插入序号列表和概率数组中，记录概率和下标的对应关系。重复操作，直到最终概率数组和下标列表仅剩余 2 个元素则实现了霍夫曼编码的合并过程。

之后进行编码的过程，这里考虑利用列表的特性，由于合并时每两项通过列表重新组合，每个括号项与编码时是否需要加入 0 或 1 对应，即通过列表实现了霍夫曼树的结构。如对应于序列长度为 2 的伯努利分布，其霍夫曼编码的合并后序号列表为

$$idx = [\,[\,0, 1\,], [\,2, 3\,]\,]$$

同时发现由于霍夫曼树是二叉树，每个列表内一定仅含有两项；这两项也一定是列表或某个下标数值。因此采用递归求解，编写函数处理对应初始下标的码本，即可实现霍夫曼编码的整个过程。

2 算法实现

对于伯努利概率分布的生成，可以先通过 pow 函数计算概率分布的总项数；之后对于每个项数化为 2 进制序列，通过判断其中 1 的数量计算概率 $p_i = p^i(1-p)^{(n-i)}$，则得到伯努利分布的结果。

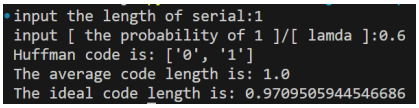
对于泊松分布概率的生成与伯努利概率类似，需要注意的是序列并不是无限长的，需要确定项数，即对分布进行截断即可。

对于霍夫曼编码的具体流程，分为 huffmancode 与 coding 两个函数处理。其中 huffmancode 主要完成对给定概率分布合并与记录下标的工作，下标利用列表记录与处理，概率分布采用 numpy 数组进行处理，方便计算。在确定循环次数时，每次合并会减少一项，直到最终仅剩两项，则终止条件为 $len(idx) = 2$。在编写代码的时候需要注意一些细节，如通过 np.where 寻找到的下标为 tuple 元组，需要提取对应的下标；插入合并后的下标列表组合时需要额外添加中括号记录二叉树的合并过程。

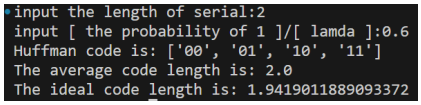
在得到下标列表表示的哈夫曼树结构后，由于列表中一定仅有两项，且这两项只可能是下标数值或者含有两项的列表，因此考虑利用递归解决编码问题。代码中为 coding 函数，输入参数包括下标列表 idx，最终霍夫曼码本 codebook 与已有的码字 char。其中 codebook 为根据初始概率分布总数初始化的字符串列表，每一项为”。当输入列表仅有一项时，返回对应的 codebook[i]=codebook[i] + char，即为对应概率的编码后的码字。否则对列表内的两项递归处理，其中的 char 参数分别赋值为 char + ’0’ 与 char + ’1’。则完成整个霍夫曼编码的过程。

3 仿真结果

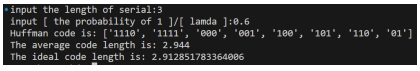
为方便比较结果与理解霍夫曼编码的特点，计算编码后的平均码长，与根据概率分布计算的熵进行比较。利用 Python 仿真在 Bernoulli 分布下，序列长度分别为 1 到 5 时的编码结果：



(a) n=1



(b) n=2



(c) n=3

(d) n=4

(e) n=5

可见程序能够正确实现霍夫曼编码，计算得到的编码长度也接近于理想长度，具有较好的性能。类似的仿真参数为 0.5 下的泊松分布的霍夫曼编

码，截断后的长度分别为 2、3、5、10、20 的结果如下：

```
input the length of serial:2
input [ the probability of 1 ]/[ lamda ]:0.5
Huffman code is: ['0', '1']
The average code length is: 0.9097959895689501
The ideal code length is: 0.9595444110422124
```

(a) k=2

```
input the length of serial:3
input [ the probability of 1 ]/[ lamda ]:0.5
Huffman code is: ['00', '01', '1']
The average code length is: 1.8954083116019795
The ideal code length is: 1.241683331866608
```

(b) k=3

```
input the length of serial:5
input [ the probability of 1 ]/[ lamda ]:0.5
Huffman code is: ['0000', '0001', '001', '01', '1']
The average code length is: 3.893484573415733
The ideal code length is: 1.3360695958680344
```

(c) k=5

(d) k=10

(e) k=20

注意到由于对概率进行了截断处理，余项的概率和并不是 1，因此会出现在长度为 2 时理想长度大于实际长度的情况。同时由于泊松分布随着项数的增加，概率分布快速下降，导致霍夫曼树的分布非常不均匀，无法取得较好的结果。因此霍夫曼编码在此处并不适用。

对于问题三，霍夫曼编码是期望意义上的最优编码，然而对于某些特定序列可能并不是最优的。例如对于概率分布为 $\mathrm{P} = [\,0.1, 0.2, 0.3, 0.4\,]$，分别对应字符为 A，B，C，D 的情况下，一种霍夫曼编码为 A-001，B-101，C-11，D-0，平均编码长度为 1.9，逼近理论码长 1.85。然而对于序列 ABABABCD 而言，若采用霍夫曼编码，总码长为 21；若采用等长编码 A-11，B-10，C-01，D-00，其总码长为 16，小于 21。因此期望意义的最小码长对于特定序列并不一定最优。

4 代码说明

代码文件为 huffman.py，需要使用的库为 numpy 与 math。运行代码后可在控制台输入序列长度/截断参数和 1 概率/λ 参数，调用不同的概率分布需要修改 main 函数，采用 Pprob/Eprob 函数生成对应分布。