基于colab平台的信息论-信道容量迭代算算法（python实现版）

Copyright 2018 The BUPT Zhengyuan Zhu.

View all sources on GitHub

Affilication: BUPT

Author:824zzy(计算机学院-2018140455-朱正源)

References

- 《信息论基础（第二版）》
- 信道容量

- 算法原理
- 算法适用条件
- 求解结果

信道容量的迭代算法

书中第六章介绍的离散信道容量的计算智能处理某些特殊情况，因此本实验针对任意的离散信道的转移概率分布，使用迭代算法进行计算。

符号定义

- r: 输入符号集大小
- s:输出符号集大小
- ϵ: 很小的正数

- p_i:概率分布
- C:信道容量
- $q_{ji} = P_{X|Y}(a_i|b_j)$:反条件概率
- $I(x;y)$: p_i和q_{ji}的互信息
- u:中间变量，表示$\sum_i p_i \alpha_i$

信道容量的意义

在信息论中，信道容量（Channel capacity，又译通道容量）是指在一个信道中能够可靠地传送信息时可达速率的最小上界。所谓可靠传输指的是可以以任意小的错误率传递信息。根据有噪信道编码定理，信道容量是可以误差概率任意小地达到的给定信道的极限信息率。信道容量的单位为比特每秒、奈特每秒等等。

香农在第二次世界大战期间发展出信息论，为信道容量提了定义，并且提供了计算信道容量的数学模型。香农指出，信道容量是信道的输入与输出的互信息量的最大值，而相应的输入分布称为最佳输入分布。

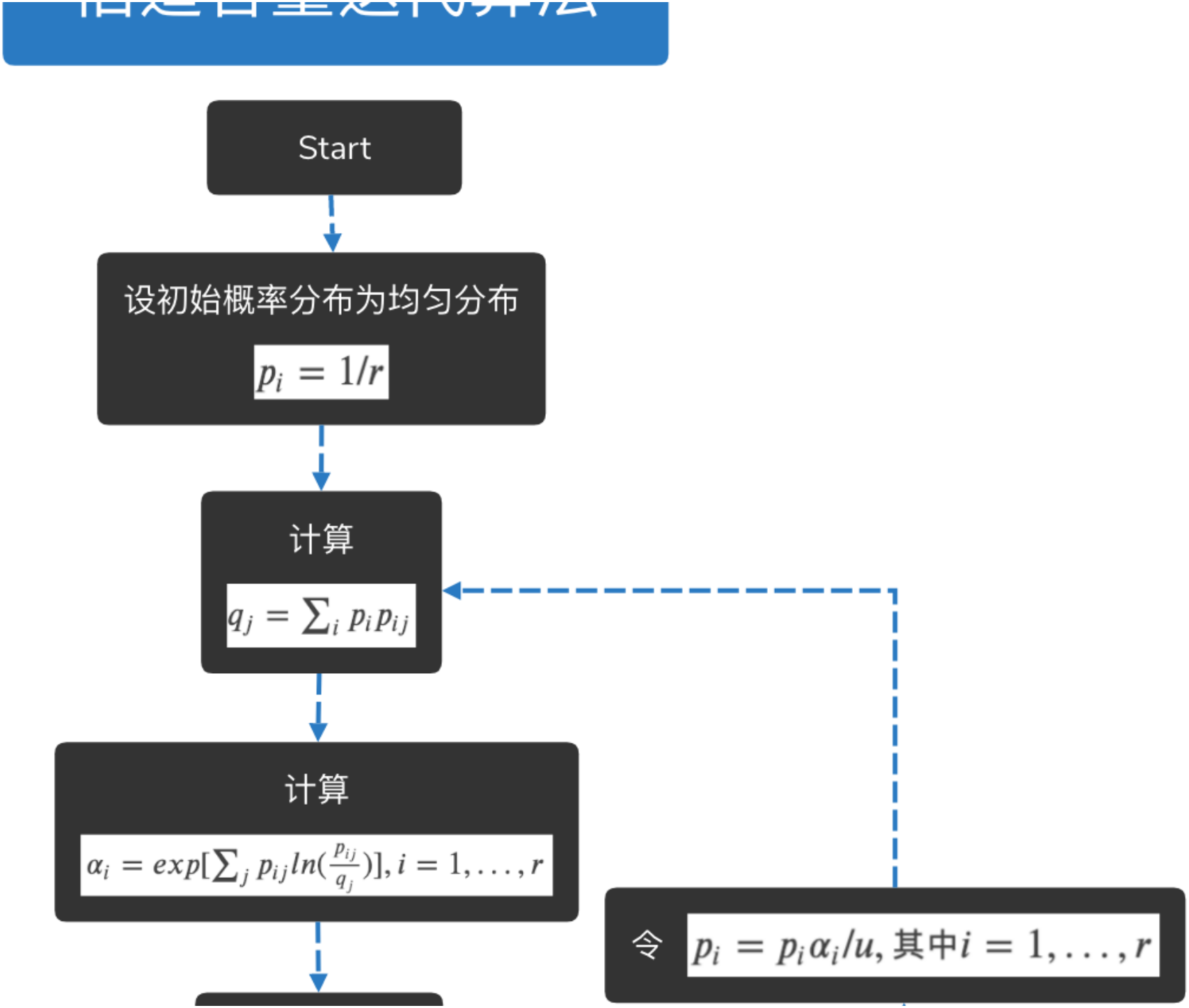
算法原理

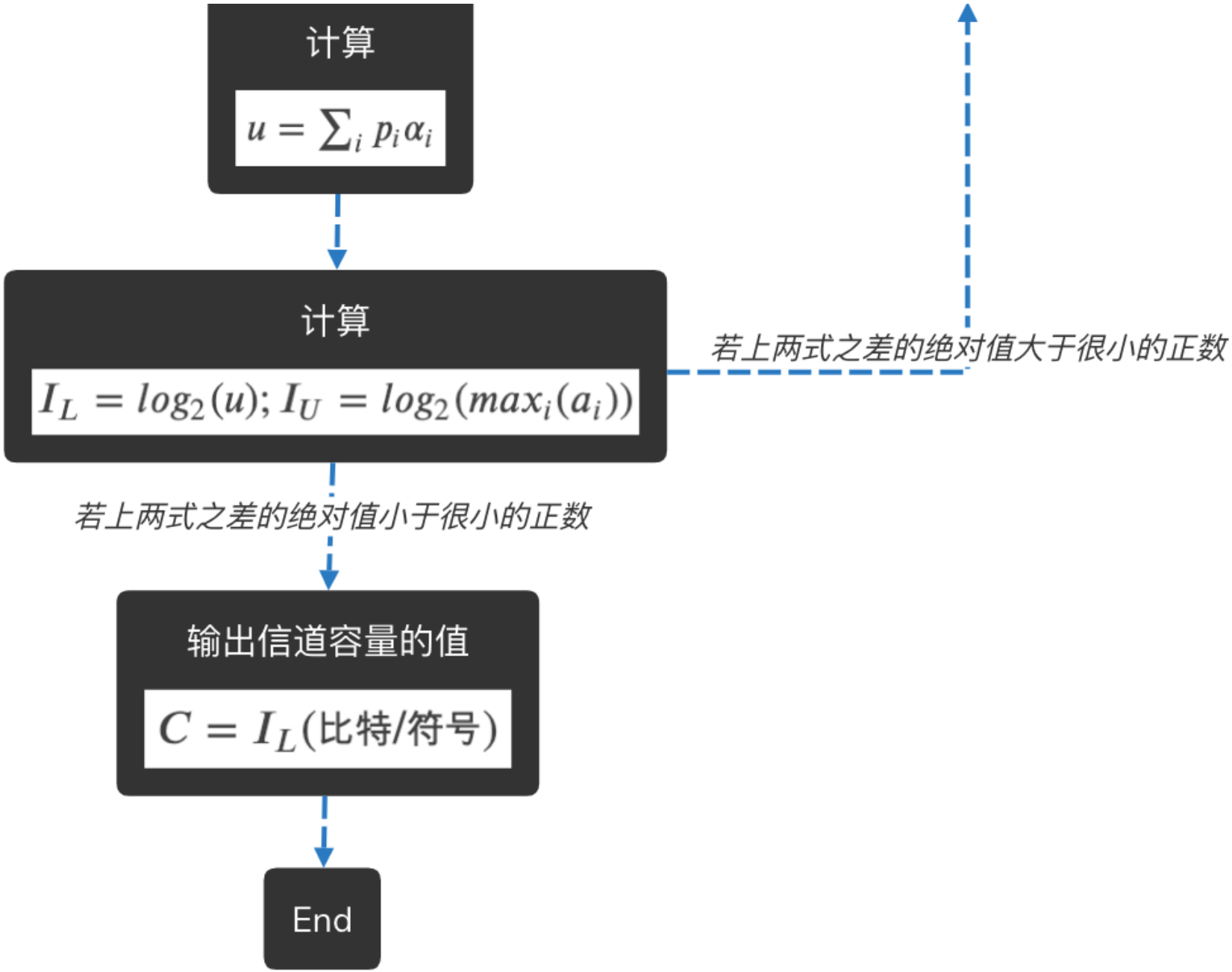
1. 在约束条件p_i通过迭代计算使$I(x;y) = C(n,n)$收敛于信道容量
2. 当信道固定时，把$I(X;Y)$看成p_i和q_{ij}的函数，进行信道容量计算的迭代。

算法适用条件

1. 离散信道的容量计算
2. 不适用于计算有约束信道的容量

算法流程图


Start
设初始概率分布为均匀分布
$p_i = 1/r$
计算
$q_j = \sum_i p_i p_{ij}$
计算
$\alpha_i = exp[\sum_j p_{ij} ln(\frac{p_{ij}}{q_j})], i = 1, \ldots, r$
令
$p_i = p_i \alpha_i / u$, 其中$i = 1, \ldots, r$


计算
$u = \sum_i p_i \alpha_i$
计算
$I_L = log_2(u); I_U = log_2(max_i(a_i))$
若上两式之差的绝对值大于很小的正数
若上两式之差的绝对值小于很小的正数
输出信道容量的值
$C = I_L$(比特/符号)
End

算法实现（Python版本）

导入需要的库

```
import numpy as np
```

设置参数

```
# 很小的正数
e = 1e-7
```

设置输入概率转移矩阵

```
p_1 = np.array([[0.98, 0.02],
                [0.05, 0.95]])

p_2 = np.array([[0.6, 0.4],
                [0.01, 0.99]])

p_3 = np.array([[0.8, 0.15, 0.05],
                [0.05, 0.15, 0.8]])

# p_4 = np.array([[0.99, 0.01, 0.0000001],
#                 [0.005, 0.99, 0.0049999],
#                 [0.0000001 , 0.01, 0.99]])
p_4 = np.array([[0.99, 0.01, np.finfo(float).eps],
                [0.005, 0.99, 0.005],
                [np.finfo(float).eps , 0.01, 0.99]])
```

算法主体

```
def initiate_prob_distrib(p):
  p_x = np.transpose(np.ones((1, p.shape[0])) / p.shape[0])
  print("初始化概率分布为:\n", p_x)
```

```
  return p_x

p_x = initiate_prob_distrib(p_4)

def iteration(p_i, p_ij, k):
  q_j = np.sum(p_i * p_ij, axis=0)
  print("第 "+str(k)+" 次迭代的q_j为:\n", q_j)

  alpha_i = np.exp(np.sum(p_ij * np.log(p_ij / q_j), axis=1))
  alpha_i = np.expand_dims(alpha_i, axis=0)
  print("第 "+str(k)+" 次迭代的alpha_i为:\n", alpha_i)

  u = np.matmul(alpha_i, p_i)[0]
  print("第 "+str(k)+" 次迭代的u为:\n", u)

  I_L = np.log2(u)[0]
  print("第 "+str(k)+" 次迭代的I_L为:\n", I_L)

  I_U = np.log2(np.amax(alpha_i))
  print("第 "+str(k)+" 次迭代的I_U为:\n", I_U)

  if I_U - I_L < e:
    print("输出信道容量的值为:\n ", I_L)
    print("达到容量时的输入概率为:\n ", p_i)
    return True, I_L, p_i
  else:
    p_i = p_i * np.transpose(alpha_i) / u[0]
    print("第"+str(k)+"次更新后的概率分布为:\n", p_i)
    return False, _, p_i

flag, I_L, p_i = iteration(p_x, p_4, 1)
```

初始化概率分布为：
 [[0.33333333]
 [0.33333333]
 [0.33333333]]

迭代求解

```
ans_dict = dict()
for p in (p_1, p_2, p_3, p_4):
  flag = False
  p_x = initiate_prob_distrib(p)
  k = 1
  while not flag:
    flag, ans, tmp_p = iteration(p_x, p, k)
    if not flag:
      p_x = tmp_p
      k = k + 1
    else:
      ans_dict[ans] = tmp_p

    print("-----" * 10)
  print("*****" * 20)
```

```
 [[0.51116299]
 [0.48883701]]
--------------------------------------------------
第 2 次迭代的q_j为:
 [0.52538158 0.47461842]
第 2 次迭代的alpha_i为:
 [[1.72913904 1.71883751]]
第 2 次迭代的u为:
 [1.72410327]
第 2 次迭代的I_L为:
 0.7858461941715016
第 2 次迭代的I_U为:
 0.790053880984547
第2次更新后的概率分布为:
 [[0.512656]
 [0.487344]]
--------------------------------------------------
第 3 次迭代的q_j为:
 [0.52677008 0.47322992]
第 3 次迭代的alpha_i为:
 [[1.72477336 1.72340077]]
第 3 次迭代的u为:
 [1.72410443]
第 3 次迭代的I_L为:
 0.7858471659236256
第 3 次迭代的I_U为:
 0.7864067985148375
第3次更新后的概率分布为:
 [[0.5128549]
 [0.4871451]]
--------------------------------------------------
第 4 次迭代的q_j为:
 [0.52695506 0.47304494]
第 4 次迭代的alpha_i为:
 [[1.72419349 1.72401072]]
第 4 次迭代的u为:
 [1.72410446]
第 4 次迭代的I_L为:
 0.7858471831597305
```

```python
for k, v in ans_dict.items():
  print("概率转移矩阵的信道容量为: ", k)
  print("达到容量的输入概率为: \n", np.transpose(v)[0])
```

```
 print("*" * 50)
```

```
概率转移矩阵的信道容量为:  0.7858471834708413
达到容量的输入概率为:
 [0.51288544 0.48711456]
**************************************************
概率转移矩阵的信道容量为:  0.3687678777410564
达到容量的输入概率为:
 [0.42379041 0.57620959]
**************************************************
概率转移矩阵的信道容量为:  0.5756565849372116
达到容量的输入概率为:
 [0.5 0.5]
**************************************************
概率转移矩阵的信道容量为:  1.5008780579795749
达到容量的输入概率为:
 [0.33583498 0.32833005 0.33583498]
**************************************************
```