

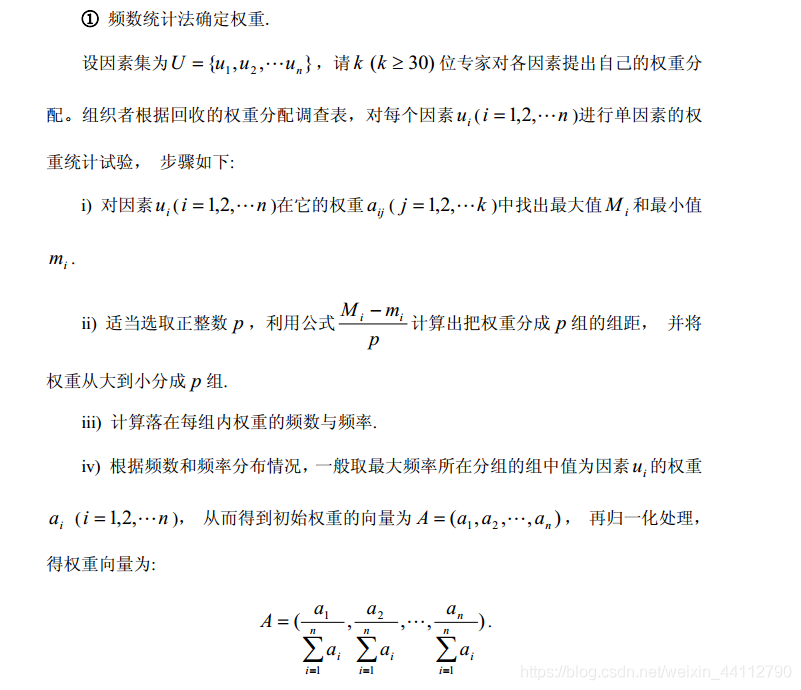
## 代码

# 具体实现

## 权重 ak 的确定

### 频数统计法确定权重

#### 算法理论



算法代码：

def frequency(matrix,p):

'''

频数统计法确定权重

:param matrix: 因素矩阵

:param p: 分组数

:return: 权重向量

'''

A = np.zeros((matrix.shape[0]))

for i in range(0, matrix.shape[0]):

## 根据频率确定频数区间列表

row = list(matrix[i, :])

maximum = max(row)

minimum = min(row)

gap = (maximum - minimum) / p

row.sort()

group = []

item = minimum

while(item < maximum):

group.append([item, item + gap])

item = item + gap

print(group)

## 初始化一个数据字典，便于记录频数

dataDict = {}

for k in range(0, len(group)):

dataDict[str(k)] = 0

## 判断本行的每个元素在哪个区间内，并记录频数

for j in range(0, matrix.shape[1]):

for k in range(0, len(group)):

if(matrix[k, j] >= group[k][0]):

dataDict[str(k)] = dataDict[str(k)] + 1

break

print(dataDict)

## 取出最大频数对应的key，并以此为索引求组中值

index = int(max(dataDict,key=dataDict.get))

mid = (group[index][0] + group[index][1]) / 2

print(mid)

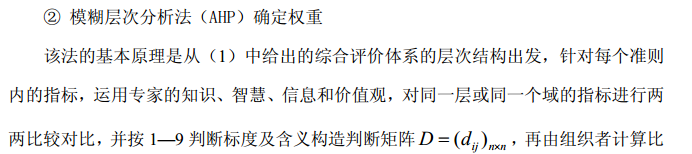
A[i] = mid

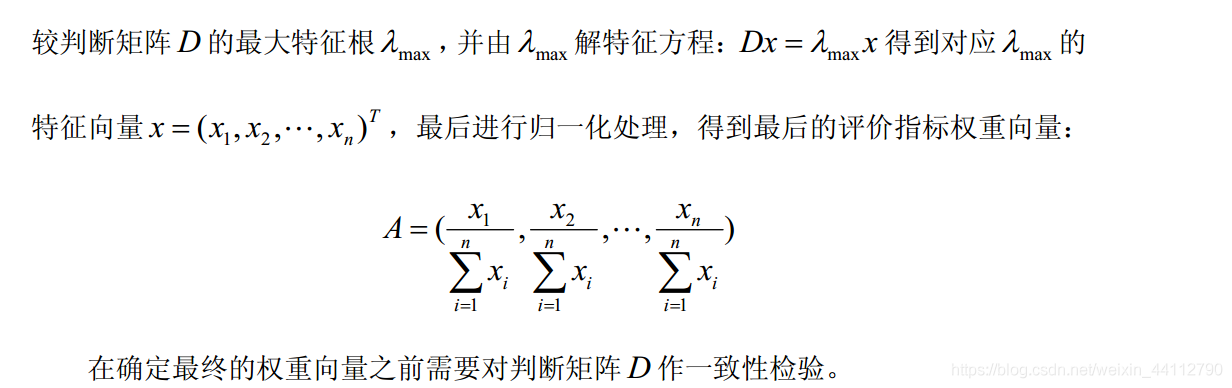
A = A / sum(A[:]) ## 归一化

return A

### 模糊层次分析法确定权重

#### 算法理论





def AHP(matrix):

if isConsist(matrix):

lam, x = np.linalg.eig(matrix)

return x[0] / sum(x[0][:])

else:

print("一致性检验未通过")

return None

def isConsist(matrix):

'''

:param matrix: 成对比较矩阵

:return: 通过一致性检验则返回true，否则返回false

'''

n = np.shape(matrix)[0]

a, b = np.linalg.eig(matrix)

maxlam = a[0].real

CI = (maxlam - n) / (n - 1)

RI = [0, 0, 0.58, 0.9, 1.12, 1.24, 1.32, 1.41, 1.45]

CR = CI / RI[n-1]

if CR < 0.1:

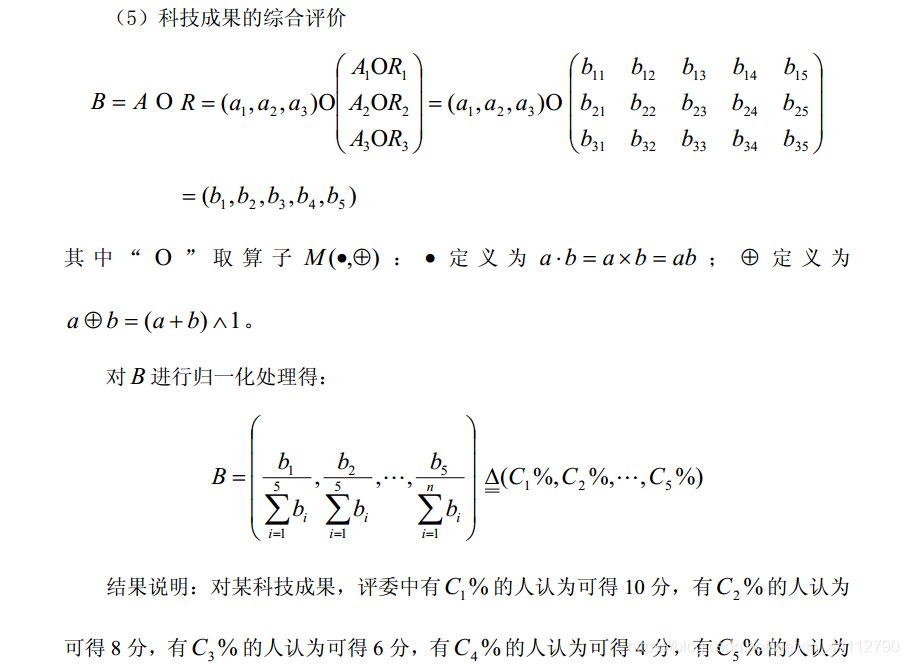
return True, CI, RI[n-1]

else:

return False, None, None

## 综合评价

### 算法理论



算法代码

def appraise(criterionMatrix, targetMatrixs, relationMatrixs):

'''

:param criterionMatrix: 准则层权重矩阵

:param targetMatrix: 指标层权重矩阵列表

:param relationMatrixs: 关系矩阵列表

:return:

'''

R = np.zeros((criterionMatrix.shape[1], relationMatrixs[0].shape[1]))

for index in range(0, len(targetMatrixs)):

row = mul\_mymin\_operator(targetMatrixs[index], relationMatrixs[index])

R[index] = row

B = mul\_mymin\_operator(criterionMatrix, R)

return B / sum(B[:])