评价类问题——TOPSIS

Introduction

TOPSIS（优劣解距离法）和AHP（层次分析法）一样是综合评价方法，用于解决评价类问题。

相对于TOPSIS，AHP有如下**局限性**：

- 决策层 n 不能太多，否则矩阵一致性较差（无法通过一致性检验）。
- 无法处理决策层数据已知问题，自行协调填写精确度不足，且数据利用率不足。

TOPSIS的核心思想为评分构造公式 $\frac{x-min}{max-min}$，多指标情况下为 $\frac{dis(x,min)}{dis(x,max)+dis(x,min)}$ （优劣解距离法名称的由来）。

Theory

以下为TOPSIS的理论分析及实现步骤。

Step 1 : 统一指标类型

评价类指标分类如下表：

指标类型	解释	参数
极大型（效益型）	数值越大越好	无
极小型（成本型）	数值越小越好	无
中间型	数值越靠近（中间）最优值越好	x_{best}
区间型	数值越靠近（中间）最优区间越好	x_{left}、x_{right}

统一指标类型的常用方法为**指标正向化**，即讲所有指标转换为**极大型**。

Case 1 : 极小型 -> 极大型

`Solution 1`（无前提，所有情况适用）

$$x_{ij} \Longrightarrow max_j - x_{ij}$$

`Solution 2`（**前提**：所有 x_{ij} 均为正数）

$$x_{ij} \Longrightarrow \frac{1}{x}$$

Case 2 : 中间型 -> 极大型

$$M = max\{|x_{ij} - x_{j\_best}|\}$$
$$\widetilde{x} \Longrightarrow 1 - \frac{|x_{ij} - x_{j\_best}|}{M}$$

Case 3 : 区间型 -> 极大型

$$M = max\{x_{left} - min\{x_j\}, max\{x_j\} - x_{right}\}$$
$$x \Longrightarrow \begin{cases} 1 - \frac{x_{left} - x}{M} & x < x_{left} \\ 1 & x_{left} \le x \le x_{right} \\ 1 - \frac{x - x_{right}}{M} & x > x_{right} \end{cases}$$

Step 2 : 标准化处理

标准化处理目的是**消去量纲差异**。已有正向化后的矩阵 X 如下：

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix}$$

对每一个元素除以其所在列平方和：

$$Z_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{n} x_{ij}^2}}$$

得到标准化矩阵 Z：

$$Z=\begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1m} \\ z_{21} & z_{22} & \cdots & z_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n1} & z_{n2} & \cdots & z_{nm} \end{bmatrix}$$

Step 3 : 加权处理

> 参考文章
>
> AHP法：**评价类算法：层次分析法笔记（附Python代码）*层次分析法python代码* 张张在努力_Lambda的博客-CSDN博客**
>
> 熵权法：**评价类算法：熵权法笔记（附Python代码）*熵权法python代码* 张张在努力_Lambda的博客-CSDN博客**

若评价指标无重要性之分，则可直接跳转 `Step 4`。以下使用熵权法进行加权。

首先保证矩阵中不存在负数（否则可以在进行一次归一化）；

接着计算熵值：

$$e=-\frac{1}{\ln(n)}\times\sum_{i=1}^{n}p_i\times\ln(p_i)$$

其中 p 值为相应值除以一列中数值的和；

最后构建权重，用 $1-e$ 得到信息的**效用值**，再对权重进行归一化即可得到权重 w，将权重矩阵与数据矩阵**相乘**获得加权后的数据矩阵。

Step 4 : 计算得分

最大值向量 $Z^+=[Z_1^+,Z_2^+,\ldots,Z_m^+]$

最小值向量 $Z^-=[Z_1^-,Z_2^-,\ldots,Z_m^-]$

计算第 i 个对象与**最大值和最小值距离**，可以理解为多维距离公式：

$$D_i^+=\sqrt{\sum_{j=1}^{m}(Z_j^+-Z_{ij})^2}$$

$$D_i^-=\sqrt{\sum_{j=1}^{m}(Z_j^--Z_{ij})^2}$$

第 i 个对象未归一化的得分：

$$S_i=\frac{D_i^-}{D_i^++D_i^-}$$

Step 5 : 归一化

矩阵全局归一化。

Code

`Python` （只有py，鼠鼠不会Matlab /(ㄒoㄒ)/~~

```python
# -*- coding: utf-8 -*-
# @File    :   topsis.py
# @Time    :   2023/03/31 15:57:36
# @Author  :   HzzzQ
# @College :   Computer Science & Engineering, CEIE, Tongji University

import numpy as np

''' step 1  输入数据 '''

''' 样例数据：
5000 0.01 7.35 89
4500 0.2 7 63
4000 0.1 7.42 201
4400 0.0 7.10 60
5100 0.03 7.52 180
'''

# 数据输入和类型划分
print("请输入参评对象数目n：")
n = eval(input())
print()

print("请输入评价指标数目m：")
m = eval(input())
print()

print("请输入类型矩阵：1:极大型，2：极小型，3：中间型，4：区间型")
kind = input().split(" ")
print()

print("请输入矩阵：")
A = np.zeros(shape=(n, m))
for i in range(n):
    A[i] = input().split(" ")
    A[i] = list(map(float, A[i]))
print()
print("输入矩阵为：\n{}".format(A))
print()


''' step 2  统一指标类型 '''

# 极小型指标转化为极大型指标：
```

```python
def minTomax(maxx, x):
    x = list(x)
    ans = [[(maxx-e)] for e in x]       # 可选择（极小型 → 极大型）统一方式
    # ans = [list(1/e) for e in x]
    return np.array(ans)

# 中间型指标转化为极大型指标：
def midTomax(bestx, x):
    x = list(x)
    h = [abs(e-bestx) for e in x]
    M = max(h)
    if M == 0:
        M = 1
    ans = [[(1-e/M)] for e in h]
    return np.array(ans)

# 区间型指标转化为极大型指标：
def regTomax(lowx, highx, x):
    x = list(x)
    M = max(lowx-min(x), max(x)-highx)
    if M == 0:
        M = 1
    ans = []
    for i in range(len(x)):
        if x[i]<lowx:
            ans.append([(1-(lowx-x[i])/M)])
        elif x[i]>highx:
            ans.append([(1-(x[i]-highx)/M)])
        else:
            ans.append([1])
    return np.array(ans)

X = np.zeros(shape=(n, 1))
# 根据输入的每一列类型进入不同函数
for i in range(m):
    if kind[i]=="1":
        v = np.array(A[:, i])

    elif kind[i]=="2":
        maxA = max(A[:, i])
        v = minTomax(maxA, A[:, i])

    elif kind[i]=="3":
        print("类型三：请输入最优值：")
        bestA = eval(input())
        v = midTomax(bestA, A[:, i])
        print()

    elif kind[i]=="4":
        print("类型四：请输入区间[a, b]值a：")
        lowA = eval(input())
        print()
        print("类型四：请输入区间[a, b]值b：")
        highA = eval(input())
        print()
        v = regTomax(lowA, highA, A[:, i])
```

```
101
102         if i==0:
103             X = v.reshape(-1, 1)
104         else:
105             X = np.hstack([X, v.reshape(-1, 1)])
106
107     print("统一指标后矩阵为：\n{}".format(X),'\n')
108
109
110     ''' step 3    标准化处理 '''
111
112     X = X.astype('float')
113     for j in range(m):
114         X[:, j] = X[:, j]/np.sqrt(sum(X[:, j]**2))
115     print("标准化矩阵为：\n{}".format(X),'\n')
116
117
118     ''' step 4    获取权重指标 '''
119     # 熵权法
120     # 这里默认了矩阵中不存在负数；若存在负数，需要再次归一化
121     p = X                                       # 计算概率矩阵P
122     for j in range(m):
123         p[:, j] = X[:, j]/sum(X[:, j])
124
125     E = np.array(X[0, :])                       # 计算熵值
126     for j in range(m):
127         E[j] = -1/np.log(n)*sum(p[:, j]*np.log(p[:, j]+ 1e-5))
128
129     w = (1-E)/sum(1-E)                          # 计算熵权
130     print("权重矩阵为：\n{}".format(w),'\n')
131
132
133     '''step 5    最大值最小值距离'''
134
135     # 得到加权后的数据
136     R = X*w
137     print("权重后的数据:\n{}".format(R),'\n')
138     # 得到最大值最小值距离
139
140     r_max = np.max(R, axis=0)               # 每个指标的最大值
141     r_min = np.min(R, axis=0)               # 每个指标的最小值
142     d_z = np.sqrt(np.sum(np.square((R - np.tile(r_max, (n, 1)))), axis=1))      #
        D+向量
143     d_f = np.sqrt(np.sum(np.square((R - np.tile(r_min, (n, 1)))), axis=1))      #
        D-向量
144     print('每个指标的最大值:', r_max,'\n')
145     print('每个指标的最小值:', r_min,'\n')
146     print('d+向量:', d_z,'\n')
147     print('d-向量:', d_f,'\n')
148
149
150     ''' step 6    计算排名 '''
151
152     s = d_f/(d_z+d_f)
153     Score = 100*s/max(s)
154     for i in range(len(Score)):
```

```
155         print(f"第{i+1}个百分制得分为: {Score[i]}\n")
156
```