TOPSIS优劣解距离法

In [ ]:
```
import numpy as np
import pandas as pd
```

例题：

In [ ]:
```
#   例题数据

columns = ["经费", "失业率", "评分"]
Funding = [99, 100, 98, 97]
unemploymentRate = [0.010, 0.012, 0.040, 0.033]
scores = [0, 0, 0, 0]
X = np.array([Funding, unemploymentRate, scores]).T
X = np.around(X, 3)
df = pd.DataFrame(X, index=["北大", "清华", "上交", "浙大"], columns=columns)
df
```

Out[ ]:

	经费	失业率	评分
北大	99.0	0.010	0.0
清华	100.0	0.012	0.0
上交	98.0	0.040	0.0
浙大	97.0	0.033	0.0

先对数据进行正向化和标准化处理，之后进行打分。

In [ ]:
```
def Min2Max(X):
    """
    极小型转极大型
    """
    X = [max(X) - x_i for x_i in X]
    return X


new_rate = Min2Max(unemploymentRate)
columns = ["经费", "失业率", "正向化后的失业率", "评分"]
df = pd.DataFrame(
    np.array([Funding, unemploymentRate, new_rate, scores]).T,
    index=["北大", "清华", "上交", "浙大"],
    columns=columns,
)
```

In [ ]:
```
def calculate_Z_vector(x_row):
    """
    计算标准化后的矩阵

    参数是一维列向量
```

```
    """
    x_row = [x_i / (sum(np.square(x_row)) ** (1 / 2)) for x_i in x_row]
    return x_row


stand_funding = calculate_Z_vector(Funding)
stand_rate = calculate_Z_vector(new_rate)

columns = [
    "经费",
    "标准化后的经费",
    "失业率",
    "正向化后的失业率",
    "标准化后的失业率",
    "评分",
]

df = pd.DataFrame(
    np.array(
        [Funding, stand_funding, unemploymentRate, new_rate, stand_rate, scores]
    ).T,
    index=["北大", "清华", "上交", "浙大"],
    columns=columns,
)
df
```

Out[ ]:

	经费	标准化后的经费	失业率	正向化后的失业率	标准化后的失业率	评分
北大	99.0	0.502506	0.010	0.030	0.720646	0.0
清华	100.0	0.507582	0.012	0.028	0.672603	0.0
上交	98.0	0.497430	0.040	0.000	0.000000	0.0
浙大	97.0	0.492354	0.033	0.007	0.168151	0.0

- 方法一：距离法

$$\text{评分} = \frac{x-min}{max-min}$$

In [ ]:

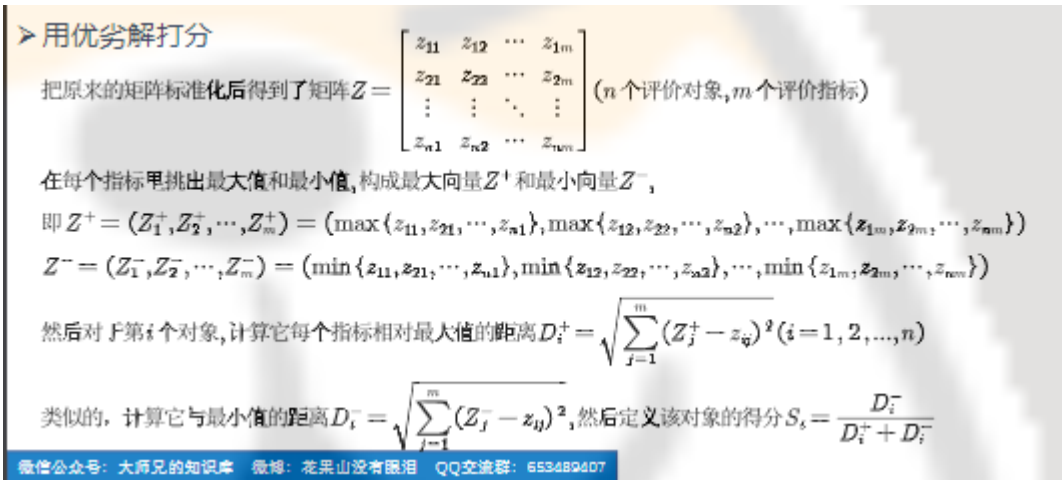
```
def calc_score_with_direction(X):
    """
    计算距离法评分
    """
    X = [(x_i - min(X)) / (max(X) - min(X)) for x_i in X]
    return X


score_funding = calc_score_with_direction(stand_funding)
score_rate = calc_score_with_direction(stand_rate)

columns = [
    "经费",
    "标准化经费",
    "经费打分",
    "失业率",
    "正向化失业率",
    "标准化失业率",
```

```
    "失业率打分",

]

df = pd.DataFrame(
    np.array(
        [
            Funding,
            stand_funding,
            score_funding,
            unemploymentRate,
            new_rate,
            stand_rate,
            score_rate,
        ]
    ).T,
    index=["北大", "清华", "上交", "浙大"],
    columns=columns,
)
df
```

Out[ ]:

	经费	标准化经费	经费打分	失业率	正向化失业率	标准化失业率	失业率打分
北大	99.0	0.502506	0.666667	0.010	0.030	0.720646	1.000000
清华	100.0	0.507582	1.000000	0.012	0.028	0.672603	0.933333
上交	98.0	0.497430	0.333333	0.040	0.000	0.000000	0.000000
浙大	97.0	0.492354	0.000000	0.033	0.007	0.168151	0.233333

- 方法二：优劣解



In [ ]:
```
Z = np.array([stand_funding,stand_rate]).T
Z
```

Out[ ]:
```
array([[0.5025, 0.7206],
       [0.5076, 0.6726],
       [0.4974, 0.    ],
       [0.4924, 0.1682]])
```

In [ ]:
```
Z_max = Z.max(axis=0)
Z_min = Z.min(axis=0)
```

In [ ]:
```
D_max = np.sqrt(np.sum(np.square(Z - Z_max), axis=1))
D_min = np.sqrt(np.sum(np.square(Z - Z_min), axis=1))
scores = D_min / (D_max + D_min)
```

> 此时的结果还没有进行归一化。

In [ ]:
```
scores
```

Out[ ]:
```
array([0.993 , 0.9333, 0.007 , 0.2333])
```

In [ ]:
```
# 归一化

final_scores = scores / sum(scores)
final_scores
```

Out[ ]:
```
array([0.4583, 0.4308, 0.0032, 0.1077])
```

In [ ]:
```
columns = [
    "标准化经费",
    "正向化失业率",
    "标准化失业率",
    "D+",
    "D-",
    "评分",
    "归一化评分",
]

df = pd.DataFrame(
    np.array(
        [stand_funding, new_rate, stand_rate, D_max, D_min, scores, final_scores
    ).T,
    index=["北大", "清华", "上交", "浙大"],
    columns=columns,
)
df
```

Out[ ]:

	标准化经费	正向化失业率	标准化失业率	D+	D-	评分	归一化评分
北大	0.502506	0.030	0.720646	0.005076	0.720717	0.993007	0.458322
清华	0.507582	0.028	0.672603	0.048043	0.672775	0.933349	0.430787
上交	0.497430	0.000	0.000000	0.720717	0.005076	0.006993	0.003228
浙大	0.492354	0.007	0.168151	0.552705	0.168151	0.233265	0.107664