PCA

2020 年 5 月 6 日

```
In [1]: import numpy as np
        import pandas as pd

In [2]: data=np.loadtxt('data16-1.txt')
        x=data

In [3]: pd.DataFrame(x)

Out[3]:      0    1    2    3    4    5
        0  2.0  3.0  3.0  4.0  5.0  7.0
        1  2.0  4.0  5.0  5.0  6.0  8.0
```

0.1 1. 标准化-求相关矩阵

```
In [4]: np.mean(x, axis=1)
        np.mean(x, axis=1)
        # pd.DataFrame()

Out[4]: array([4., 5.])

In [5]: pd.DataFrame(x.T)

Out[5]:      0    1
        0  2.0  2.0
        1  3.0  4.0
        2  3.0  5.0
        3  4.0  5.0
        4  5.0  6.0
        5  7.0  8.0

In [6]: # 样本矩阵减去每列特征的均值
        x_zero_mean=(x.T- np.mean(x, axis=1)).T
        pd.DataFrame(x_zero_mean)
```

```
Out[6]:      0    1    2    3    4    5
        0 -2.0 -1.0 -1.0  0.0  1.0  3.0
        1 -3.0 -1.0  0.0  0.0  1.0  3.0
```

```
In [7]: n=x.shape[1]
        n
```

```
Out[7]: 6
```

```
In [8]: # 计算协方差矩阵
        var=np.sum(x_zero_mean*x_zero_mean, axis=1)/(n-1)
        var
        # pd.DataFrame()
```

```
Out[8]: array([3.2, 4. ])
```

```
In [9]: # 对样本矩阵进行标准化
        x_std=(x_zero_mean.T/var**0.5).T
        pd.DataFrame(x_std)
```

```
Out[9]:          0         1         2    3         4         5
        0 -1.118034 -0.559017 -0.559017  0.0  0.559017  1.677051
        1 -1.500000 -0.500000  0.000000  0.0  0.500000  1.500000
```

```
In [10]: # 计算样本矩阵对应的相关矩阵 r
         r=x_std.dot(x_std.T)/(x_zero_mean.shape[1]-1)
         pd.DataFrame(r)
```

```
Out[10]:          0         1
         0  1.000000  0.950329
         1  0.950329  1.000000
```

0.2 2. 对相关矩阵进行对角化，求特征值特征向量

```
In [11]: #     对相关矩阵 r 进行对角化分解
         evalue, evector=np.linalg.eig(r)
```

```
In [12]: pd.DataFrame(evalue)
```

```
Out[12]:          0
         0  1.950329
         1  0.049671
```

```
In [13]: pd.DataFrame(evector)
```

```
Out[13]:          0         1
         0  0.707107 -0.707107
         1  0.707107  0.707107
```

0.3 3. 求第一、第二主成分对应的方差贡献率、因子负荷量

```
In [14]: #     计算方差贡献率
         contribution=evalue/np.sum(evalue)
         var_accumulative_percent=0.0
         for k in range(len(contribution)):
             var_accumulative_percent+=contribution[k]
             if var_accumulative_percent>=1:
                 break
         contribution
```

```
Out[14]: array([0.97516445, 0.02483555])
```

```
In [15]: #     计算因子负荷量 factor loadig
         n=x.shape[0]
         factor_loading = np.mat(np.zeros((n,n)), dtype=float)
         for i in range(len(evalue)):
             for j in range((len(evector))):
                 print(evalue[j], evector[i,j], var[i])
                 factor_loading[i, j] = evalue[j]**0.5*evector[i,j]/var[i]**0.5

         pd.DataFrame(factor_loading.T, index=["y1","y2"], columns=["x1","x2"])
```

```
1.9503288904374105 0.7071067811865475 3.2
0.049671109562589466 -0.7071067811865475 3.2
1.9503288904374105 0.7071067811865475 4.0
0.049671109562589466 0.7071067811865475 4.0
```

```
Out[15]:           x1        x2
         y1  0.552032  0.493752
         y2 -0.088097  0.078797
```