myPCA

2020 年 5 月 6 日

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


        def myPCA(x, var_contribution_threshold=0.8):
            '''
            自编程实现主成分分析
            x 样本矩阵
            var_contribution_threshold 方差贡献率阈值
            '''
        # 样本矩阵减去每列特征的均值
            x_zero_mean=(x.T- np.mean(x, axis=1)).T
        # 计算协方差矩阵
            n=x_zero_mean.shape[1]
            var=np.sum(x_zero_mean*x_zero_mean, axis=1)/(n-1)
        # 对样本矩阵进行标准化
            x_std=(x_zero_mean.T/var**0.5).T
        # 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)
        #     对相关矩阵 r 进行对角化分解
            evalue, evector=np.linalg.eig(r)

        #     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>=var_contribution_threshold:
                    break
        #     计算因子载荷量
            n=x.shape[0]
```

```
    factor = 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[i, j] = evalue[j]**0.5*evector[i,j]/var[i]**0.5

#     对数据进行降维，取前 k 个
    evalue=evalue[0:k+1]
    evector=evector[:, 0:k+1]
    contribution=contribution[0:k+1]
    factor=factor[0:k+1, :]
    y=evector.T.dot(x_std)

    return y, evalue, evector, contribution, factor
```

In [2]:

```
np.random.seed(5)
a = 10*np.random.rand(6,6)
a
```

Out[2]:

```
array([[2.21993171, 8.70732306, 2.06719155, 9.18610908, 4.88411189,
        6.11743863],
       [7.65907856, 5.18417988, 2.96800502, 1.87721229, 0.80741269,
        7.38440296],
       [4.41309223, 1.58309868, 8.79937031, 2.74086462, 4.14235019,
        2.96079933],
       [6.28787909, 5.7983781 , 5.99929197, 2.65819118, 2.84685881,
        2.53588206],
       [3.27563948, 1.44164301, 1.65612861, 9.63930529, 9.60226715,
        1.88414656],
       [0.24306562, 2.04555546, 6.99843614, 7.79514586, 0.22933092,
        5.77662858]])
```

In [3]:

```
x=np.loadtxt('data16-1.txt', dtype=float)
x
```

Out[3]:

```
array([[2., 3., 3., 4., 5., 7.],
       [2., 4., 5., 5., 6., 8.]])
```

In [4]:

```
var_contribution_threshold=1
y, evalue, evector, contribution, factor=myPCA(x, var_contribution_threshold)
print("样本主成分值 (方差贡献率前%d%%): " %(var_contribution_threshold*100))
print(y)
```

```
        print("特征值(方差贡献率前%d%%): " %(var_contribution_threshold*100))
        print(evalue)
        print("单位向量和主成分的方差贡献率%d%%: " %(var_contribution_threshold*100))
        print(np.vstack((evector, contribution)).T)
        print("主成分的因子负载量(方差贡献率前%d%%): " %(var_contribution_threshold*100))
        print(factor)
```

```
1.9503288904374105 0.7071067811865475 3.2
0.049671109562589466 -0.7071067811865475 3.2
1.9503288904374105 0.7071067811865475 4.0
0.049671109562589466 0.7071067811865475 4.0
样本主成分值(方差贡献率前100%):
[[-1.85122959 -0.7488381  -0.39528471  0.          0.7488381   2.24651429]
 [-0.27009076  0.04173132  0.39528471  0.         -0.04173132 -0.12519395]]
特征值(方差贡献率前100%):
[1.95032889 0.04967111]
单位向量和主成分的方差贡献率100%:
[[ 0.70710678  0.70710678  0.97516445]
 [-0.70710678  0.70710678  0.02483555]]
主成分的因子负载量(方差贡献率前100%):
[[ 0.5520316  -0.08809717]
 [ 0.49375207  0.0787965 ]]
```