10182100359-郑佳辰-第八周实验练习题

2021 年 4 月 20 日

1 Week8 Multicollinearity-PCR

1.1 背景描述

数据集来源：Longley's(1967)

我们构造了 16 个观测的 6 个自变量，具体请见下表：

1.2 数据描述

变量名	变量含义	变量类型	变量取值范围
（自变量）X1	国民生产总值隐含价格评价指数 (1954=100)	continuous variable	$\mathbb{R}^+$
（自变量）X2	国民生产总值	continuous variable	$\mathbb{R}^+$
（自变量）X3	失业人数	continuous variable	$\mathbb{R}^+$
（自变量）X4	武装力量的规模	continuous variable	$\mathbb{R}^+$
（自变量）X5	14 岁及以上的非机构人口	continuous variable	$\mathbb{R}^+$
（自变量）X6	时间（年份）	continuous variable	$\mathbb{R}^+$
（因变量）Y	总就业人数	continuous variable	$\mathbb{R}^+$

1.3 问题

注：这里使用 $\alpha = 0.05$ 的显著性水平

1. 判断所给数据是否具有多重共线性.
2. 若具有多重共线性, 选择适当的主成分.
3. 对降维后的数据进行回归分析.

1.4 解决方案

Q1:

多重共线性是指自变量 $x_1, x_2, ..., x_p$ 之间不完全线性相关但是相关性很高的情况。此时，虽然可以得到最小二乘估计，但是精度很低。随着自变量之间相关性增加，最小二乘估计结果的方差会增大。

[1]:
```
# Import standard packages
import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
import math

# Import additional packages
from itertools import combinations
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.decomposition import PCA  # 进行主成分分析

alpha = 0.05
p = 6
n = 16

x = pd.read_csv('Project8.csv')
x.insert(0, 'intercept', np.ones(len(x)))
data = x.values * 1.0
df = pd.DataFrame(data)
print(df.head())

# 对数据进行分割
X = data[:,0:p+1]
Y = data[:,-1]
```

```
     0      1         2       3       4         5       6        7
0  1.0  830.0  234289.0  2356.0  1590.0  107608.0  1947.0  60323.0
1  1.0  885.0  259426.0  2325.0  1456.0  108632.0  1948.0  61122.0
2  1.0  882.0  258054.0  3682.0  1616.0  109773.0  1949.0  60171.0
```

```
3   1.0   895.0   284599.0   3351.0   1650.0   110929.0   1950.0   61187.0
4   1.0   962.0   328975.0   2099.0   3099.0   112075.0   1951.0   63221.0
```

[2]:

```
# Do the multiple linear regression——对原始数据
# OLS(endog,exog=None,missing='none',hasconst=None) (endog:因变量, exog= 自变量)
model = sm.OLS(Y, X).fit()
beta = model.params
model.summary()
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-
packages/scipy/stats/stats.py:1604: UserWarning: kurtosistest only valid for
n>=20 … continuing anyway, n=16
  "anyway, n=%i" % int(n))
```

[2]:

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.995
Model:                            OLS   Adj. R-squared:                  0.992
Method:                 Least Squares   F-statistic:                     330.3
Date:                Tue, 20 Apr 2021   Prob (F-statistic):           4.98e-10
Time:                        22:19:01   Log-Likelihood:                -109.62
No. Observations:                  16   AIC:                             233.2
Df Residuals:                       9   BIC:                             238.6
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -3.482e+06    8.9e+05     -3.911      0.004    -5.5e+06   -1.47e+06
x1             1.5062      8.491      0.177      0.863     -17.703      20.715
x2            -0.0358      0.033     -1.070      0.313      -0.112       0.040
x3            -2.0202      0.488     -4.136      0.003      -3.125      -0.915
x4            -1.0332      0.214     -4.822      0.001      -1.518      -0.549
x5            -0.0511      0.226     -0.226      0.826      -0.563       0.460
x6          1829.1515    455.478      4.016      0.003     798.788    2859.515
```

```
==============================================================================
Omnibus:                        0.749   Durbin-Watson:                   2.559
Prob(Omnibus):                  0.688   Jarque-Bera (JB):                0.684
Skew:                           0.420   Prob(JB):                        0.710
Kurtosis:                       2.434   Cond. No.                     4.86e+09
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 4.86e+09. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

数据预处理:

[3]:
```python
# 对数据进行标准化
# 自变量 X 的均值
X_mean = []
for i in range(p):
    X_mean.append(np.mean(X[:, i+1]))

# 自变量 X 的标准差
X_L = []
for i in range(p):
    X_L.append(sum((X[:, i+1] - X_mean[i]) ** 2))

# 对自变量 X 标准化（截距项不用标准化）
X_std = X * 1.0
X_std[:,1:p+1] = (X[:,1:p+1] - X_mean) / np.sqrt(X_L)

# 对因变量 Y 标准化
Y_std = (Y - np.mean(Y)) / np.sqrt(sum((Y - np.mean(Y))**2))

df_std = pd.DataFrame(X_std)
df_std['Y'] = Y_std
print(df_std)
```

	0	1	2	3	4	5	6	Y
0	1.0	-0.446968	-0.398513	-0.231355	-0.377210	-0.364354	-0.406745	-0.367157
1	1.0	-0.315375	-0.333214	-0.239921	-0.426926	-0.326344	-0.352512	-0.308415
2	1.0	-0.322553	-0.336778	0.135028	-0.367563	-0.283992	-0.298279	-0.378332
3	1.0	-0.291449	-0.267822	0.043570	-0.354949	-0.241084	-0.244047	-0.303636
4	1.0	-0.131144	-0.152546	-0.302366	0.182657	-0.198546	-0.189814	-0.154097
5	1.0	-0.085685	-0.105725	-0.348509	0.366311	-0.154190	-0.135582	-0.123366
6	1.0	-0.064152	-0.057964	-0.365640	0.348873	-0.086486	-0.081349	-0.024114
7	1.0	-0.040226	-0.063868	0.106292	0.275783	-0.044728	-0.027116	-0.114397
8	1.0	-0.011514	0.025381	-0.079939	0.163735	-0.001336	0.027116	0.051611
9	1.0	0.069834	0.081780	-0.102596	0.092871	0.048625	0.081349	0.186740
10	1.0	0.160753	0.143057	-0.071097	0.070980	0.112134	0.135582	0.209678
11	1.0	0.218175	0.147673	0.411058	0.011246	0.167998	0.189814	0.087930
12	1.0	0.261242	0.246797	0.171224	-0.020290	0.220557	0.244047	0.245409
13	1.0	0.299524	0.298483	0.203828	-0.034389	0.294868	0.298279	0.312238
14	1.0	0.335413	0.338935	0.445597	-0.012870	0.387070	0.352512	0.295108
15	1.0	0.364124	0.434325	0.224827	0.081740	0.469807	0.406745	0.384802

做多元线性回归分析:

先后对未经标准化和已标准化的数据进行回归分析，得到的 $\hat{\beta}$ 分别如表所示。

[4]:
```python
# Do the multiple linear regression——对标准化后的数据
model_std = sm.OLS(Y_std, X_std).fit()
beta_std = model_std.params
model_std.summary()
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-
packages/scipy/stats/stats.py:1604: UserWarning: kurtosistest only valid for
n>=20 … continuing anyway, n=16
  "anyway, n=%i" % int(n))
```

[4]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.995
Model:                            OLS   Adj. R-squared:                  0.992
```

```
Method:                 Least Squares   F-statistic:                     330.3
Date:                Tue, 20 Apr 2021   Prob (F-statistic):           4.98e-10
Time:                        22:19:01   Log-Likelihood:                 42.670
No. Observations:                  16   AIC:                            -71.34
Df Residuals:                       9   BIC:                            -65.93
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -1.128e-17      0.006  -2.01e-15      1.000      -0.013       0.013
x1             0.0463      0.261      0.177      0.863      -0.544       0.637
x2            -1.0137      0.948     -1.070      0.313      -3.158       1.130
x3            -0.5375      0.130     -4.136      0.003      -0.832      -0.244
x4            -0.2047      0.042     -4.822      0.001      -0.301      -0.109
x5            -0.1012      0.448     -0.226      0.826      -1.114       0.912
x6             2.4797      0.617      4.016      0.003       1.083       3.876
==============================================================================
Omnibus:                        0.749   Durbin-Watson:                   2.559
Prob(Omnibus):                  0.688   Jarque-Bera (JB):                0.684
Skew:                           0.420   Prob(JB):                        0.710
Kurtosis:                       2.434   Cond. No.                         206.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

求 $(X^*)'(X^*)$ 矩阵的特征值和特征向量:

如果求得的特征值至少一个非常接近 0，则可以认为存在多重共线性。

[5]:
```python
# (X*)'(X*) 矩阵等价于原始矩阵 X 样本相关矩阵
R = df.corr()
R = R.iloc[1:-1,1:-1]
```

```
# 求 (X*)'(X*) 矩阵，结果与样本相关矩阵一致
# R1 = np.dot(X_std.T,X_std)
# R1 = pd.DataFrame(R1[1:,1:])

R
```

[5]:

	1	2	3	4	5	6
1	1.000000	0.991589	0.620633	0.464744	0.979163	0.991149
2	0.991589	1.000000	0.604261	0.446437	0.991090	0.995273
3	0.620633	0.604261	1.000000	-0.177421	0.686552	0.668257
4	0.464744	0.446437	-0.177421	1.000000	0.364416	0.417245
5	0.979163	0.991090	0.686552	0.364416	1.000000	0.993953
6	0.991149	0.995273	0.668257	0.417245	0.993953	1.000000

[6]:

```
# 求特征值 & 特征向量
W, V = np.linalg.eig(R)
W_diag = np.diag(W)
V = V.T # 这里需要转置

print('特征值:', W)
# print(W_diag)
# print(sum(W)) # 验证特征值求和值为 p
# VV = np.dot(V,V.T)
# VV = pd.DataFrame(VV)
# print(VV) # 验证矩阵 V'V 结果为单位阵
```

```
特征值: [4.60337710e+00 1.17534050e+00 2.03425372e-01 1.49282587e-02
 2.55206576e-03 3.76708133e-04]
```

判断 X 矩阵是否具有多重共线性:

[7]:

```
# 定义"判断多重共线性"的函数
# 参数: (X_list: 设计矩阵 X, thres_vif: VIF 方法判断多重共线性的阈值, thres_kappa:
↪ 特征值方法判断多重共线性的阈值)
def judge_col(X_list, thres_vif, thres_kappa):
    var_num = X_list.shape[1]
    print('VIF 方法判断结果 (阈值为 %d): '% thres_vif)
    vif = [variance_inflation_factor(X_list, i) for i in range(var_num)]
```

```
    for i in range(var_num):
        if vif[i] >= thres_vif:
            print('设计矩阵 X 存在多重共线性.')
            break
        elif i == var_num-1:
            print('设计矩阵 X 不存在多重共线性.')

    print('\n特征值判定法判断结果 (阈值为 %d): '% thres_kappa)
    kappa = []
    for i in range(var_num):
        kappa.append(np.sqrt(max(W) / W[i]))
    if np.max(kappa) >= thres_kappa:
        print('设计矩阵 X 存在多重共线性, 其中 kappa 值为: %.4f'% np.max(kappa))
    else:
        print('设计矩阵 X 不存在多重共线性, 其中 kappa 值为: %.4f'% np.max(kappa))

# 判断多重共线性
judge_col(X_std[:,1:p+1], 5, 10)
```

```
VIF 方法判断结果 (阈值为 5):
设计矩阵 X 存在多重共线性.

特征值判定法判断结果 (阈值为 10):
设计矩阵 X 存在多重共线性, 其中 kappa 值为: 110.5442
```

Q2:

本题需要构造出主成分矩阵，然后利用累计贡献率和特征值选择出适当的要保留的主成分的数量，并选择出需要保留的主成分。

构造主成分矩阵 Z:

[8]:
```
# 构造主成分矩阵 Z
Z = np.dot(X_std[:,1:p+1],V.T)
# ZZ = np.dot(Z.T,Z)
# ZZ = pd.DataFrame(ZZ)
# print(ZZ)  # 验证主成分矩阵 Z 各列之间正交, 主对角线元素对应的是特征值
```

[9]:
```
D = np.linalg.det(R)
print('(X*)\'X* 的行列式: ', D)
```

```
(X*)'X* 的行列式:  1.5796154862477436e-08
```

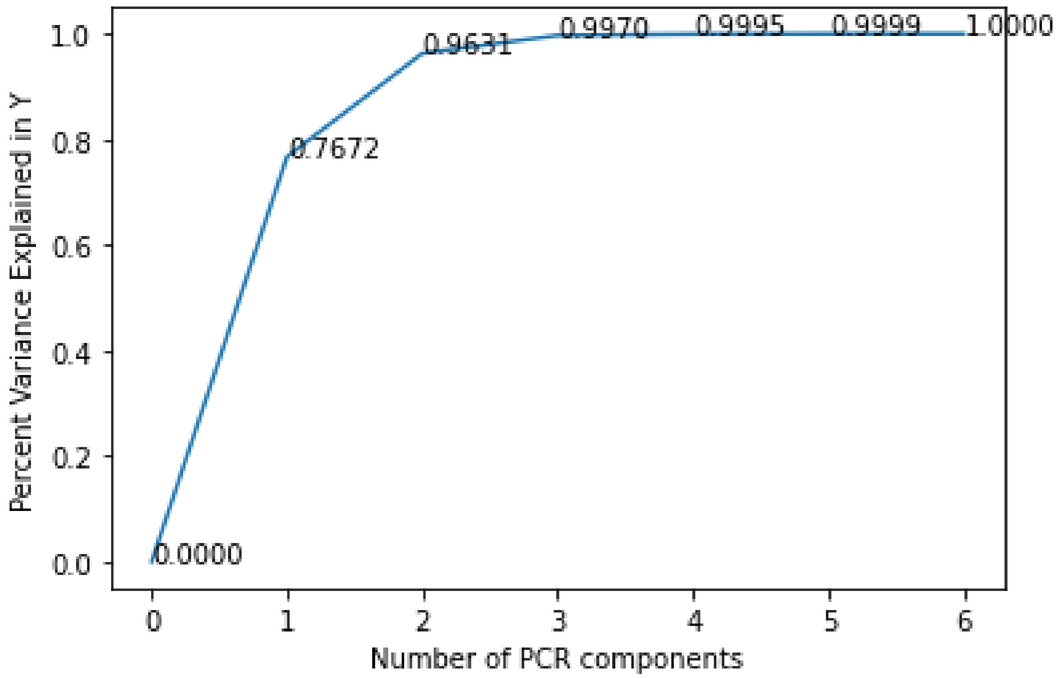
由于 $|(X^*)'(X^*)| \approx 0$, 则存在一个 k, 使得 $\lambda_{k+1}, \cdots, \lambda_p$ 均近似为 0. 因此 $\mathbf{z}_{k+1}, \cdots, \mathbf{z}_p$ 近似为 $\mathbf{0}$

选主成分:

[10]:
```
# 对特征值按降序排序
W_srt = W.tolist()
W_srt.sort(reverse=True)
W_idx = np.argsort(-W) # 返回的是元素值降序排序后的索引值的数组
print('特征值为: ', W_srt)
print('排序后特征值对应的原索引值: ', W_idx)
```

```
特征值为:  [4.603377095768392, 1.1753404992571477, 0.20342537240143493,
0.014928258677276958, 0.002552065763074821, 0.0003767081326775469]
排序后特征值对应的原索引值:  [0 1 2 3 4 5]
```

[11]:
```
# 绘制主成分的累计贡献率 (响应变量中解释的方差百分比) 与组件数量的碎石图
comp = range(0, p+1)
# 主成分的累计贡献率 (计算方差百分比)
summ = 0
W_sum = [0]
for i in range(p):
    summ += W_srt[i]
    W_sum.append(summ / p)
plt.plot(comp, W_sum)
plt.xlabel('Number of PCR components')
plt.ylabel('Percent Variance Explained in Y')
for i,j in zip(comp, W_sum):
    plt.text(i, j, '%.4f' % float(j))
```



[12]:

```python
# 保留累计贡献率比重大的主成分
c_pc = 0.8
cnt = True
thres = p * c_pc
while cnt:
    W_sum = 0
    W_summ = W_srt[0]
    for i in range(p-1):
        k1 = i + 1
        W_sum += W_srt[i]
        W_summ += W_srt[i+1]
        # print(i,W_sum,W_summ,thres)
        if (W_sum < thres) & (W_summ >= thres):
            cnt = False
            break
        elif i == p - 2:
            cnt = False
            k1 = i + 1
```

```
            break
k1 = k1 + 1
print('保留变量个数为: ', k1)
```

保留变量个数为: 2

[13]:
```
# 删除特征值接近于零的主成分
for i in range(p):
    if W_srt[i] < 1:
        k2 = i
        break
print('保留变量个数为: ', k2)
```

保留变量个数为: 2

[14]:
```
# 均方误差确定 k
mse = 0
for i in range(p):
    k3 = p
    mse += 1 / W_srt[i]
    print(5 * (i + 1), mse)
    if mse > 5 * (i + 1):
        k3 = i
        break
    elif i == p-1:
        k3 = p
        break
print('保留变量个数 <=', k3)
```

```
5 0.21723182333231833
10 1.0680490976728447
15 5.983856738553664
20 72.97090604003378
保留变量个数 <= 3
```

根据选择的累计贡献率，特征值，均方误差三个指标，综上，我们选择保留变量的个数为 2.

Q3:

在设计矩阵 X 具有多重共线性时，选择合适的 k，可以降低回归系数的均方误差。在进行主成分回归时，代码有两种编写方法：一为利用矩阵拆分的原理进行主成分回归，二为直接调用 PCA 库进行主成分分析。以下依次是两种方法的代码。

[15]:
```
# 矩阵拆分
k = k1
list_var1 = W_idx[0:k] # 记录降序排序后的前 k 个主成分
list_var2 = W_idx[k:]
# list_var1 = [0,2]
# list_var2 = [1,3]

Z_1 = Z[:,list_var1]
Z_2 = Z[:,list_var2]

W_diag_1 = np.diag(W_diag[list_var1,list_var1])
W_diag_2 = np.diag(W_diag[list_var2,list_var2])

# 按行进行拆分
V_1 = V[list_var1,:]
V_2 = V[list_var2,:]

#   的估计
# alpha_hat = np.linalg.inv(W_diag) @ Z.T @ Y_std
alpha1_hat = np.linalg.inv(W_diag_1) @ Z_1.T @ Y_std
print('系数:', alpha1_hat)

# 主成分估计
# beta_pc = np.dot(V_1.T,alpha1_hat)
# print(beta_pc)
# print(V_1.T @ V_1 @ beta_std[1:]) # 验证 PPT 99 页的性质 1
```

```
系数: [0.44565109 0.11156928]
```

[16]:
```
# 使用拆分后的数据用线性回归模型进行建模
X_pc = Z_1
model_pc = sm.OLS(Y_std, X_pc).fit()
model_pc.summary()
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-
packages/scipy/stats/stats.py:1604: UserWarning: kurtosistest only valid for
n>=20 … continuing anyway, n=16
  "anyway, n=%i" % int(n))
```

[16]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                      y   R-squared (uncentered):
0.929
Model:                            OLS   Adj. R-squared (uncentered):
0.919
Method:                 Least Squares   F-statistic:
91.43
Date:                Tue, 20 Apr 2021   Prob (F-statistic):
9.20e-09
Time:                        22:19:03   Log-Likelihood:
20.625
No. Observations:                  16   AIC:
-37.25
Df Residuals:                      14   BIC:
-35.71
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             0.4457      0.033     13.416      0.000       0.374       0.517
x2             0.1116      0.066      1.697      0.112      -0.029       0.253
==============================================================================
Omnibus:                        0.210   Durbin-Watson:                   1.919
Prob(Omnibus):                  0.900   Jarque-Bera (JB):                0.371
Skew:                          -0.201   Prob(JB):                        0.831
Kurtosis:                       2.371   Cond. No.                         1.98
==============================================================================
```

```
Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[17]:
```python
# 判断多重共线性【k > 1 时才可能存在多重共线性问题】
judge_col(X_pc, thres_vif=5, thres_kappa=10)
```

```
VIF 方法判断结果 (阈值为 5):
设计矩阵 X 不存在多重共线性.

特征值判定法判断结果 (阈值为 10):
设计矩阵 X 不存在多重共线性, 其中 kappa 值为: 1.9790
```

[18]:
```python
# 创建 pca 模型
pca = PCA(n_components=2)

# 对模型进行训练
X_pc_ = X_std * 1.0
pca.fit(X_pc_)

# 返回降维后数据
X_pc_ = pca.transform(X_pc_)

# 使用返回后的数据用线性回归模型进行建模
model_pc_ = sm.OLS(Y_std, X_pc_).fit()
model_pc_.summary()
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-
packages/scipy/stats/stats.py:1604: UserWarning: kurtosistest only valid for
n>=20 … continuing anyway, n=16
  "anyway, n=%i" % int(n))
```

[18]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                      y   R-squared (uncentered):
0.929
Model:                            OLS   Adj. R-squared (uncentered):
0.919
Method:                 Least Squares   F-statistic:
91.43
Date:                Tue, 20 Apr 2021   Prob (F-statistic):
9.20e-09
Time:                        22:19:03   Log-Likelihood:
20.625
No. Observations:                  16   AIC:
-37.25
Df Residuals:                      14   BIC:
-35.71
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -0.4457      0.033    -13.416      0.000      -0.517      -0.374
x2             0.1116      0.066      1.697      0.112      -0.029       0.253
==============================================================================
Omnibus:                        0.210   Durbin-Watson:                   1.919
Prob(Omnibus):                  0.900   Jarque-Bera (JB):                0.371
Skew:                          -0.201   Prob(JB):                        0.831
Kurtosis:                       2.371   Cond. No.                         1.98
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
```

```
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[19]:
```python
# 判断多重共线性【k 值大于 1 时才可能存在多重共线性问题】
judge_col(X_pc_, thres_vif=5, thres_kappa=10)
```

```
VIF 方法判断结果 (阈值为 5):
设计矩阵 X 不存在多重共线性.

特征值判定法判断结果 (阈值为 10):
设计矩阵 X 不存在多重共线性, 其中 kappa 值为: 1.9790
```

可以看出，选择合适的 k 用于进行主成分回归，可以使得到的设计矩阵 X 不再具有多重共线性。故主成分回归可用于数据具有多重共线性的情形。