回归分析-第一次上机

南开大学统计与数据科学学院，马东升

2024 年 10 月 29 日

1 第一问

1.1 题目

检查复共线性

1.2 源代码（R）

```r
#导入数据
library(readxl)
data <- read_excel("longley.xlsx")
head(data)

# 提取前六列
X <- data[, 1:6]

# 中心化和标准化
X_scaled <- scale(X)

# 计算 X'X
XTX <- t(X_scaled) %*% X_scaled

# 计算特征值
eig_values <- eigen(XTX)$values

# 按从大到小排序
sorted_eig_values <- sort(eig_values, decreasing = TRUE)

#计算条件数
k=sorted_eig_values[1]/sorted_eig_values[6]

k
```

1.3 统计分析结论

根据 http://stat.ethz.ch/R-manual/R-devel/library/datasets/html/longley.html 的数据说明，我们以前六列为自变量，第七列为因变量。

1.2 中代码输出的结果为 X^TX 的条件数，即

$$k = \frac{\lambda_{max}}{\lambda_{min}} = 12220.0 > 1000$$

这说明了数据的复共线性非常严重。

2 第二问

2.1 题目

使用主成分回归解决复共线性，选择适当个数的主成分

2.2 源代码（R）

第一段

```r
# 提取前六列
X <- data[, 1:6]

# 对X进行PCA变换
pca_result <- prcomp(X, center = TRUE, scale. = TRUE)

# 查看PCA的结果，包括主成分得分和主成分载荷
summary(pca_result)
```

第二段

```r
Y <- as.numeric(data$Employed)

# 主成分得分（PCA变换后的数据）
pca_scores <- pca_result$x[, 1:2]

# 对Y进行回归分析，使用前两个主成分
regression_model <- lm(Y ~ pca_scores)

# 提取回归系数
coefficients <- regression_model$coefficients

# 恢复到原始变量空间
# 通过PCA载荷矩阵（主成分载荷矩阵）来恢复系数
pca_loadings <- pca_result$rotation[, 1:2]

# 计算标准化后的系数 beta
beta_standardized <- pca_loadings %*% coefficients[2:3]

# 提取原始数据的均值和标准差
X_means <- colMeans(X)
X_sds <- apply(X, 2, sd)

# 将系数恢复到原始变量空间
beta_original <- beta_standardized / X_sds

# 恢复截距项
original_intercept <- coefficients[1] - sum((X_means / X_sds) * beta_standardized)

# 输出恢复后的系数
list(
```

```
  original_intercept = original_intercept,
  original_coefficients = beta_original
)
```

2.3 统计分析结论

第一段代码，我们得出两个主成分的累计变差贡献率：

$$\frac{\lambda_1+\lambda_2}{\sum_{i=1}^{6}\lambda_i}=0.9631>0.85$$

后三者的特征根都比较小（小于 1），所以我们直接选择**两个**主成分。

第二段代码，我们用主成分做回归后，再恢复到原始系数，最终得到如下的经验回归方程：

$$\begin{aligned}\hat{Employed}=&-258.6257+0.0691GNP.deflator+0.0075GNP\\&+0.0029Unemployed+0.0090Armed.Forces\\&+0.1014Population+0.1529Year\end{aligned}$$

我们发现个别自变量的系数已经比较小了，这也符合我们做 PCA 的目的。

3 第三问

3.1 题目

使用岭回归解决复共线性，并采用不同方法估计岭参数

3.2 源代码（R）

第一段

```r
library(glmnet)
# 先定义 ridge_regression 函数
ridge_regression <- function(X, Y, k) {
  # 确保 Y 是数值型向量
  Y <- as.numeric(Y)

  # 对 X 进行中心化和标准化
  X_scaled <- scale(X)

  # 执行岭回归，alpha=0 表示岭回归（L2 正则化）
  ridge_model <- glmnet(X_scaled, Y, alpha = 0, lambda = k, standardize = FALSE)

  # 提取标准化后的系数（包含截距）
  beta_standardized <- coef(ridge_model)

  # 将系数转换为数值向量
  beta_standardized <- as.vector(beta_standardized)

  # 计算预测值
  Y_pred <- cbind(1, X_scaled) %*% beta_standardized  # 计算预测值
```

```
  # 计算残差平方和 (RSS)
  RSS <- sum((Y - Y_pred) ^ 2)

  # 恢复到原始变量空间的系数
  X_means <- attr(X_scaled, "scaled:center")
  X_sds <- attr(X_scaled, "scaled:scale")

  beta_original <- beta_standardized[-1] / X_sds  # 去掉截距项并进行恢复
  intercept_original <- beta_standardized[1] - sum(X_means * beta_original)

  # 返回结果
  return(list(
    beta_standardized = beta_standardized,
    beta_original = c(intercept_original, beta_original),
    RSS = RSS  # 返回 RSS 值
  ))
}
```

第二段

```
X <- data[, 1:6]

# 确保Y是数值型向量
Y <- as.numeric(data$Employed)

# 标准化 X
X_scaled <- scale(X)

# 对 X'X 进行特征值分解 (spectral decomposition)
XtX <- t(X_scaled) %*% X_scaled
eigen_decomp <- eigen(XtX)

# 提取特征向量矩阵 Phi 和特征值矩阵 Lambda
Phi <- eigen_decomp$vectors
Lambda <- diag(eigen_decomp$values)

# 计算新变量矩阵 Z = X * Phi
Z <- X_scaled %*% Phi

# 进行典则形式的回归 y = alpha_0 + Z * alpha + e
canonical_model <- lm(Y ~ ., data = as.data.frame(Z))

# 提取回归系数 alpha（去掉截距项）
alpha <- coef(canonical_model)[-1]

# 计算残差的方差 sigma^2
residuals <- residuals(canonical_model)
sigma_squared <- var(residuals)

# 选取 alpha 的最大值
max_alpha2 <- max(alpha**2)

```

```
# 根据 H-K 公式计算岭回归系数 k
k <- sigma_squared / max_alpha2
k
```

第三段

```

# 定义岭迹图绘制函数
ridge_trace <- function(X, Y, k_values) {
  # 创建一个矩阵以存储标准化系数
  beta_matrix <- matrix(0, nrow = length(k_values), ncol = ncol(X) + 1)

  # 计算每个 k 值下的标准化系数
  for (i in seq_along(k_values)) {
    k <- k_values[i]
    ridge_result <- ridge_regression(X, Y, k)

    # 将标准化系数存入矩阵
    beta_matrix[i, ] <- ridge_result$beta_standardized
  }

  # 将 k_values 转换为矩阵（确保是列矩阵）
  k_values_matrix <- matrix(k_values, ncol = 1)

  # 绘制岭迹图
  matplot(k_values_matrix, beta_matrix[, -1], type = "l", lty = 1, col = 1:ncol(X),
          xlab = "岭回归系数 k", ylab = "标准化系数",
          main = "岭迹图", ylim = range(beta_matrix[, -1]),lwd=2)
  legend("topright", legend = colnames(X), col = 1:ncol(X), lty = 1)
    abline(h = 0, col = "gray", lwd = 2)
}
k_values <- seq(0.01, 10, length.out = 10000)
ridge_trace(X = data[, 1:6], Y = data$Employed, k_values = k_values)
k_values <- seq(1, 4, length.out =3000 )
ridge_trace(X = data[, 1:6], Y = data$Employed, k_values = k_values)
```

第四段

```
ridge_regression(X = data[, 1:6], Y = data$Employed, k = 0)
ridge_regression(X = data[, 1:6], Y = data$Employed, k = 0.00115)
ridge_regression(X = data[, 1:6], Y = data$Employed, k = 3)
```

3.3 统计分析结论

第一段代码是岭回归的代码，第四段代码是最终做的岭回归；第二段代码为用 Horel-Kennard 公式确定 k，第三段代码为用岭迹图确定 k。

3.3.1 用 Horel-Kennard 公式确定 k

Horel-Kennard 公式确定的 k 值为：

$$\hat{k} = \frac{\hat{\sigma}^2}{max_i \hat{\alpha}_i^2}$$

计算得到 $\hat{k} = 0.00115$，这个值略小，这是因为 $\hat{\sigma}^2 = 0.0558$ 比较小，而且 $max_i \hat{\alpha}_i^2 = 48.4885$ 也比较大。最终得到的经验回归方程为：

$$
\begin{aligned}
\hat{Employed} = & -2429.5210 - 0.0137GNP.deflator - 0.0027GNP \\
& - 0.0153Unemployed - 0.0088Armed.Forces \\
& - 0.1608Population + 1.2910Year
\end{aligned}
$$

虽然这是由 Horel-Kennard 公式确定的 k，但是很明显，该回归方程不符合经济学意义，$Employed$ 至少应该和 GNP 及 $GNP.deflator$ 正相关（经济越好或越热，就业越好）。所以我个人认为，这个 k 不是一个特别好的 k 值。

3.3.2 用岭迹图确定 k

我们用第三段代码来画出岭迹图，注意这里是标准化后的系数。

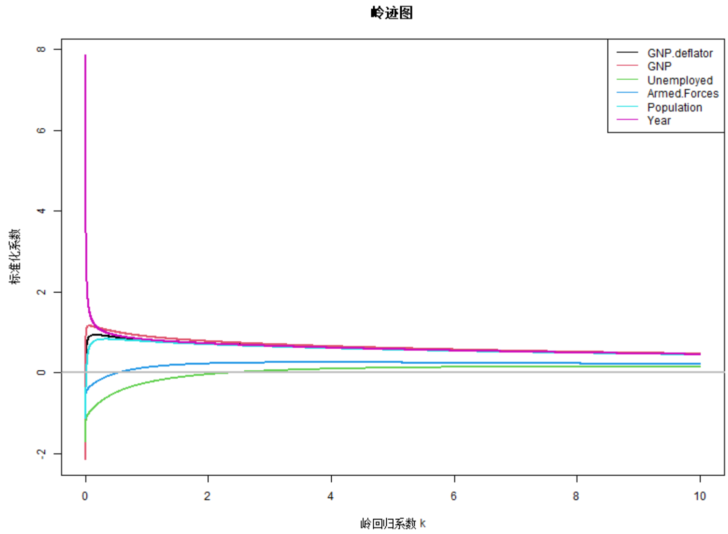

图 1: $k \in [0, 10]$的岭迹图

从图 1 可以看出 $k \in [0, 1]$ 这个区间确实不是一个岭回归的好区间，原因是系数尚不稳定，所以我们缩小区间到 $k \in [1, 4]$，得到图 2。

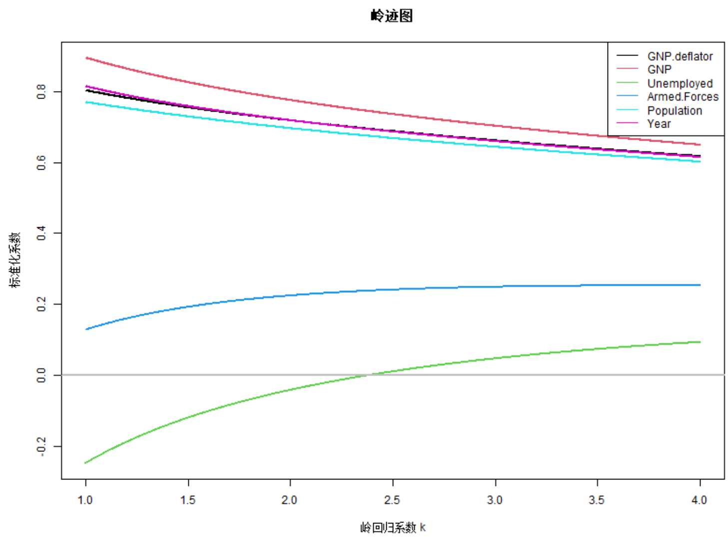

图 2: $k \in [1, 4]$的岭迹图

为了使系数符号稳定，虽然会使得 RSS 较大，但我们也不得不选择较大的 k，比如 $k=3$。

$k=3$ 时，我们得到经验回归方程：

$$
\begin{aligned}
\hat{Employed} = &-226.5122 + 0.0613GNP.deflator + 0.0070GNP \\
&+ 0.0005Unemployed + 0.0035Armed.Forces \\
&+ 0.0925Population + 0.1386Year
\end{aligned}
$$

我们发现这一方程和 PCA 得到的方程很相似，虽然 $RSS_{k=3}=15.1309$（相较于 $RSS_{k=0}=0.8542$）确实较大，但是至少系数的经济意义是相对正确的。所以我个人认为，$k=3$ 是一个较好的值。