Lab: 交叉验证

李文东

最后编译于 11/14/2022

在本lab中，我们实现交叉验证方法，其中的有些命令可能会需要一些时间来运行。

1 The Validation Set Approach

我们探索使用验证集方法来估计在 `Auto` 数据集上拟合各种线性模型所产生的测试错误率。

在开始之前，我们使用 `set.seed()` 函数来设置 `R` 的随机数生成器，这样大家将获得与下图完全相同的结果。 在执行包含随机元素的交叉验证等分析时，设置随机种子通常是一个好主意，以便以后可以精确地再现所获得的结果。

我们首先使用 `sample()` 函数将观察集分成两半，方法是从原始 392 观察中选择 196 观察的随机子集。我们将这些观察结果称为训练集。

```
library(ISLR2)
set.seed(1)
train <- sample(392, 196)
```

然后，我们使用 `lm()` 中的 `subset` 选项仅使用与训练集对应的数据来拟合线性回归。

```
lm.fit <- lm(mpg ~ horsepower, data = Auto, subset = train)
```

我们现在使用 `predict()` 函数来估计所有 392 个数据的响应，并且我们使用 `mean()` 函数来计算验证集中 196 个数据的 MSE。注意，下面的 `-train` 索引仅选择不在训练集中的数据。

```
attach(Auto)
mean((mpg - predict(lm.fit, Auto))[-train]^2)
```

```
## [1] 23.26601
```

因此，线性回归拟合的测试 MSE 的估计值为 23.27。 我们可以使用 `poly()` 函数来估计二次和三次回归的测试误差。

```
lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data = Auto, subset = train)
mean((mpg - predict(lm.fit2, Auto))[-train]^2)
```

```
## [1] 18.71646
```

```
lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = Auto, subset = train)
mean((mpg - predict(lm.fit3, Auto))[-train]^2)
```

```
## [1] 18.79401
```

这些错误率分别为 18.72 和 18.79 。 如果我们选择不同的训练集，那么我们将在验证集上获得一些不同的结果。

```
set.seed(2)
train <- sample(392, 196)
lm.fit <- lm(mpg ~ horsepower, subset = train)
mean((mpg - predict(lm.fit, Auto))[-train]^2)
```

```
## [1] 25.72651
```

```
lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data = Auto, subset = train)
mean((mpg - predict(lm.fit2, Auto))[-train]^2)
```

```
## [1] 20.43036
```

```
lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = Auto, subset = train)
mean((mpg - predict(lm.fit3, Auto))[-train]^2)
```

```
## [1] 20.38533
```

通过将观察结果拆分为训练集和验证集，我们发现具有线性、二次和三次项的模型的验证集MSE分别为25.73、20.43和 20.39。

这些结果与我们之前的发现一致：使用 `horsepower` 的二次函数预测 `mpg` 的模型比只涉及 `horsepower` 的线性函数的模型表现更好，而且几乎没有证据支持使用 `horsepower` 的三次函数的模型。

2 Leave-One-Out Cross-Validation

可以使用 `glm()` 和 `cv.glm()` 函数为任何广义线性模型自动计算 LOOCV 估计值。 在之前的实验中，我们使用 `glm()` 函数通过传入 `family = "binomial"` 参数来执行逻辑回归。但是如果我们使用 `glm()` 来拟合模型而不传入 `family` 参数，那么它会执行线性回归，就像 `lm()` 函数一样。例如：

```
glm.fit <- glm(mpg ~ horsepower, data = Auto)
coef(glm.fit)
```

```
## (Intercept)  horsepower
##  39.9358610  -0.1578447
```

以及

```
lm.fit <- lm(mpg ~ horsepower, data = Auto)
coef(lm.fit)
```

```
## (Intercept)  horsepower
##  39.9358610  -0.1578447
```

会得到同样的线性回归模型。在本实验中，我们将使用 `glm()` 函数而不是 `lm()` 函数，因为前者可以与 `cv.glm()` 一起使用。 `cv.glm()` 函数是 `boot` 库的一部分。

```
library(boot)
glm.fit <- glm(mpg ~ horsepower, data = Auto)
cv.err <- cv.glm(Auto, glm.fit)
cv.err$delta
```

```
## [1] 24.23151 24.23114
```

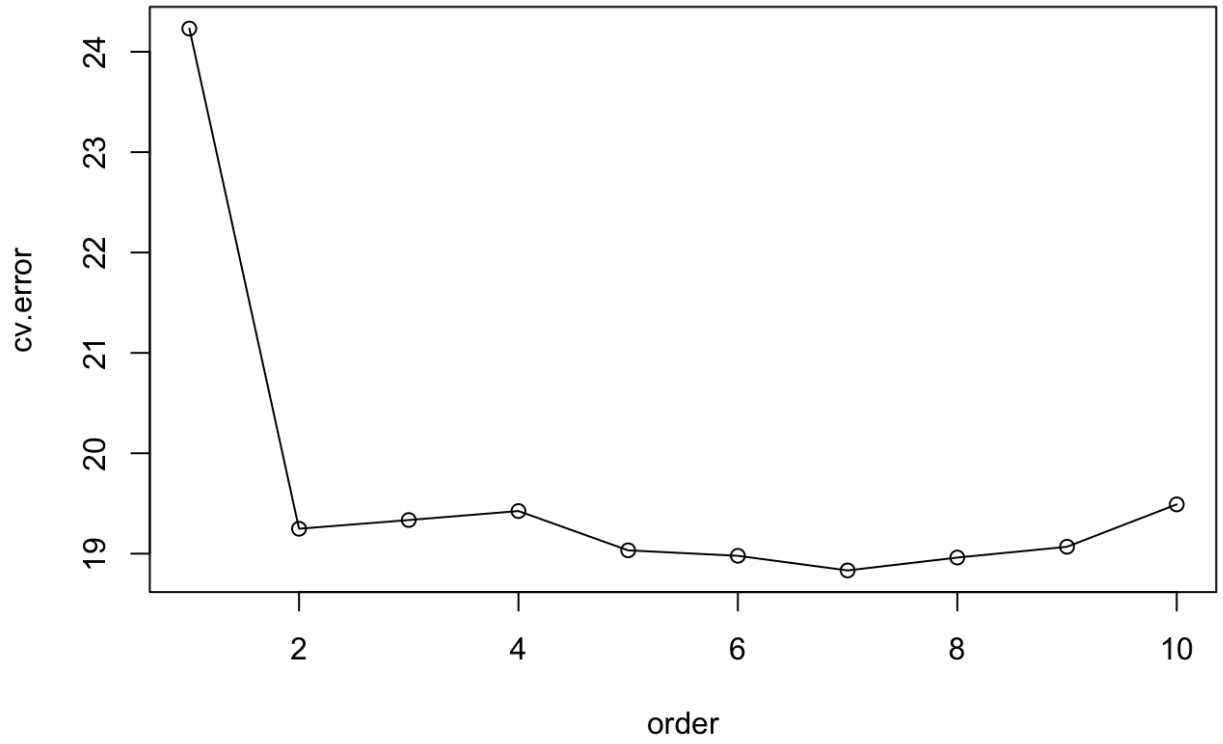
`cv.glm()` 函数生成一个包含多个元素的列表。 `delta` 向量中的两个数字包含交叉验证结果。 在这个例子中，两个数字是几乎相同的，并且都对应于LOOCV 统计量。 之后我们会讨论两个数字不同的情况。我们对测试误差的交叉验证估计约为 24.23。

我们可以对越来越复杂的多项式拟合重复这个过程。为了自动化这个过程，我们使用 `for()` 函数来启动一个迭代拟合多项式回归阶数从$i = 1$到$i = 10$，计算相关的交叉验证错误，并将其存储在向量 `cv.error` 的第i个元素中。 我们首先初始化向量。

```
cv.error <- rep(0, 10)
for (i in 1:10) {
  glm.fit <- glm(mpg ~ poly(horsepower, i), data = Auto)
  cv.error[i] <- cv.glm(Auto, glm.fit)$delta[1]
}
cv.error
```

```
##  [1] 24.23151 19.24821 19.33498 19.42443 19.03321 18.97864 18.83305 18.96115
##  [9] 19.06863 19.49093
```

```
plot(cv.error,type='o',xlab="order")
```



如上图所示，我们看到估计的测试 MSE 在线性拟合和二次拟合之间急剧下降，但使用更高阶多项式并没有明显改善。

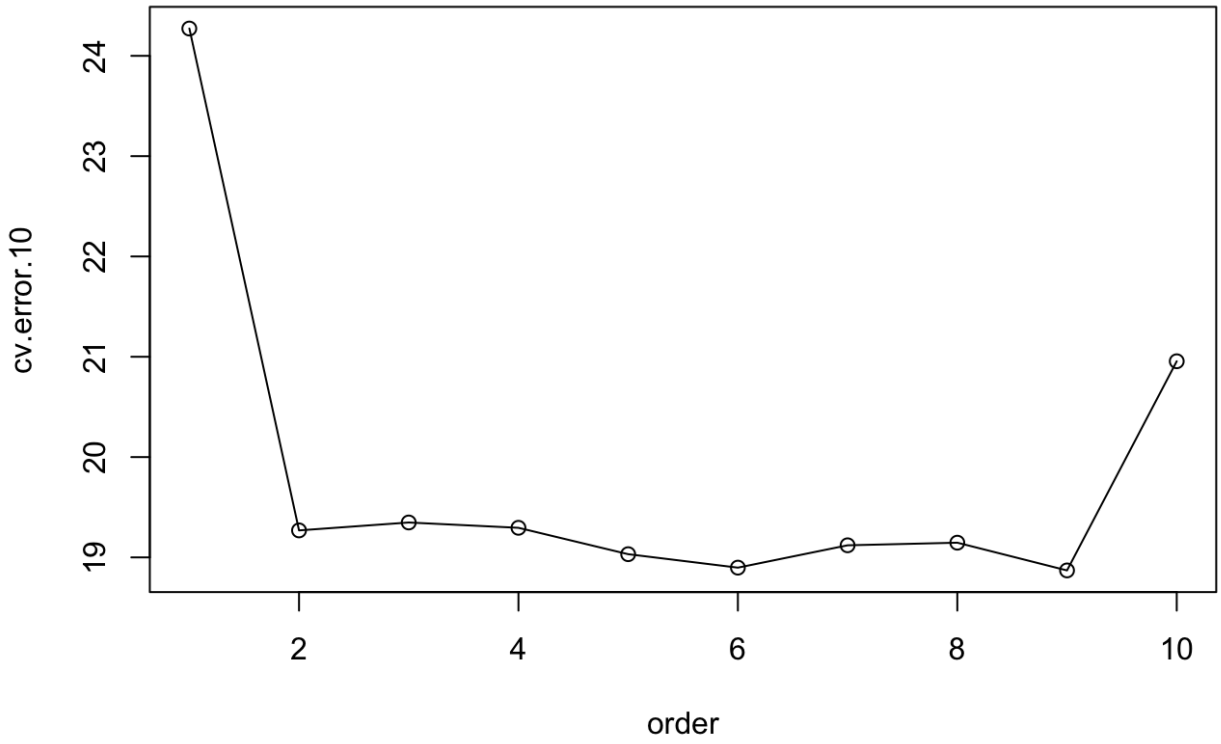
3 k-Fold Cross-Validation

`cv.glm()` 函数也可以用来实现k-fold CV。 下面我们在 `Auto` 数据集上使用 $k = 10$，这是 k 的常见选择。 我们再次设置一个随机种子并初始化一个向量，我们将在其中存储对应于 1 到 10 阶多项式拟合的 CV 误差。

```
set.seed(17)
cv.error.10 <- rep(0, 10)
for (i in 1:10) {
  glm.fit <- glm(mpg ~ poly(horsepower, i), data = Auto)
  cv.error.10[i] <- cv.glm(Auto, glm.fit, K = 10)$delta[1]
}
cv.error.10
```

```
##  [1] 24.27207 19.26909 19.34805 19.29496 19.03198 18.89781 19.12061 19.14666
##  [9] 18.87013 20.95520
```

```
plot(cv.error.10,type='o',xlab="order")
```



大家可以自己尝试，计算时间比 LOOCV 的要短。我们仍然很少看到证据表明使用三次或更高阶多项式项比简单地使用二次拟合会导致更低的测试误差。

练习

We saw that the cv.glm() function can be used in order to compute the LOOCV test error estimate. Alternatively, one could compute those quantities using just the glm() and predict.glm() functions, and a for loop. You will now take this approach in order to compute the LOOCV error for a simple logistic regression model on the Weekly data set.

a. Fit a logistic regression model that predicts Direction using Lag1 and Lag2.
b. Fit a logistic regression model that predicts Direction using Lag1 and Lag2 using all but the first observation.
c. Use the model from (b) to predict the direction of the first observation. You can do this by predicting that the first observation will go up if P(Direction="Up"|Lag1,Lag2) > 0.5. Was this observation correctly classified?
d. Write a for loop from i=1 to i=n, where n is the number of observations in the data set, that performs each of the following steps:

- Fit a logistic regression model using all but the ith observation to predict Direction using Lag1 and Lag2.
- Compute the posterior probability of the market moving up for the ith observation.
- Use the posterior probability for the ith observation in order to predict whether or not the market moves up.
- Determine whether or not an error was made in predicting the direction for the ith observation. If an error was made, then indicate this as a 1, and otherwise indicate it as a 0.

e. Take the average of the n numbers obtained in (d)iv in order to obtain the LOOCV estimate for the test error. Comment on the results.