Cross Validation and Bootstrap

We use the `set.seed()` function in order to set a seed for `R`'s random number generator, so that the results are reproducible.

```
library(ISLR2)
library(boot)
set.seed(1)
```

Cross validation

The LOOCV estimate can be automatically computed for any generalized linear model using the `glm()` and `cv.glm()` functions. If we use `glm()` to fit a model without passing in the `family` argument, then it performs linear regression, just like the `lm()` function. `glm()` can be used together with `cv.glm()`. The `cv.glm()` function is part of the `boot` library. The `cv.glm()` function produces a list with several components. The two numbers in the `delta` vector contain the LOOCV estimates of the test error. The latter is bias corrected version. In the case of LOOCV, these two numbers should be very close to each other.

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

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

We can repeat this procedure for increasingly complex polynomial fits. We use `for` loop and the vector `cv.error` is created to store the estimates.

```
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
```

We see a sharp drop in the estimated test MSE between the linear and quadratic fits, but then no clear improvement from using higher-order polynomials.

The `cv.glm()` function can also be used to implement k-fold CV.

```
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
```

We still see little evidence that using cubic or higher-order polynomial terms leads to lower test error than simply using a quadratic fit. The two numbers associated with `delta` are essentially the same when LOOCV is performed. When we instead perform k-fold CV, then the two numbers associated with `delta` differ slightly.

Bootstrap

The bootstrap approach can be used to assess the variability of the coefficient estimates and predictions from a statistical learning method. Here we use the bootstrap approach in order to assess the variability of the estimates for β_0 and β_1, the intercept and slope terms for the linear regression model that uses `horsepower` to predict `mpg` in the `Auto` data set.

We first create a simple function, `boot.fn()`, which takes in the `Auto` data set as well as a set of indices for the observations, and returns the intercept and slope estimates for the linear regression model. We then apply this function to the full set of 392 observations in order to compute the estimates of β_0 and β_1 on the entire data set.

```
boot.fn <- function(data, index)
  coef(lm(mpg ~ horsepower, data = data, subset = index))
boot.fn(Auto, 1:392)
```

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

Next, we use the `boot()` function to compute the standard errors of 1,000 bootstrap estimates for the intercept and slope terms.

```
set.seed(1)
boot(Auto, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Auto, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##       original        bias    std. error
## t1* 39.9358610  0.0553942585 0.843931305
## t2* -0.1578447 -0.0006285291 0.007367396
```

This indicates that the bootstrap estimate for $\mathrm{SE}(\hat{\beta}_0)$ is 0.84, and that the bootstrap estimate for $\mathrm{SE}(\hat{\beta}_1)$ is 0.0073. We compare them with results from standard formulas.

```
summary(lm(mpg ~ horsepower, data = Auto))$coef
```

```
##               Estimate  Std. Error   t value      Pr(>|t|)
## (Intercept) 39.9358610 0.717498656  55.65984 1.220362e-187
## horsepower  -0.1578447 0.006445501 -24.48914  7.031989e-81
```

The standard error estimates using the standard formulas are 0.717 for the intercept and 0.0064 for the slope. Interestingly, these are somewhat different from the estimates obtained using the bootstrap. Indeed, bootstrap standard errors are automatically heteroskedasticity-robust.

To get White heteroskedasticity robust standard errors, use package `sandwich`. Construct the robust variance covariance matrix `rvcv`. The robust standard errors are on the diagonal.

```
library(sandwich)
lm.fit = lm(mpg ~ horsepower, data = Auto)
rvcv=vcovHC(lm.fit,type="HC")
sqrt(diag(rvcv))
```

```
## (Intercept)  horsepower
## 0.853642884 0.007367528
```

To get t-statistics and p-values:

```
library(lmtest)
```

```
##      zoo
```

```
##
##    'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
coeftest(lm.fit, vcov=rvcv)
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 39.9358610  0.8536429  46.783 < 2.2e-16 ***
## horsepower  -0.1578447  0.0073675 -21.424 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```