Cross-Validation Resampling Methods

Justin M Shea

Resampling Methods

We are going to use the Auto data from the `ISLR` package to illustrate various re-sampling methods.

```
library(ISLR)
data(Auto)
```

```
?Auto
```

```
dim(Auto)
```

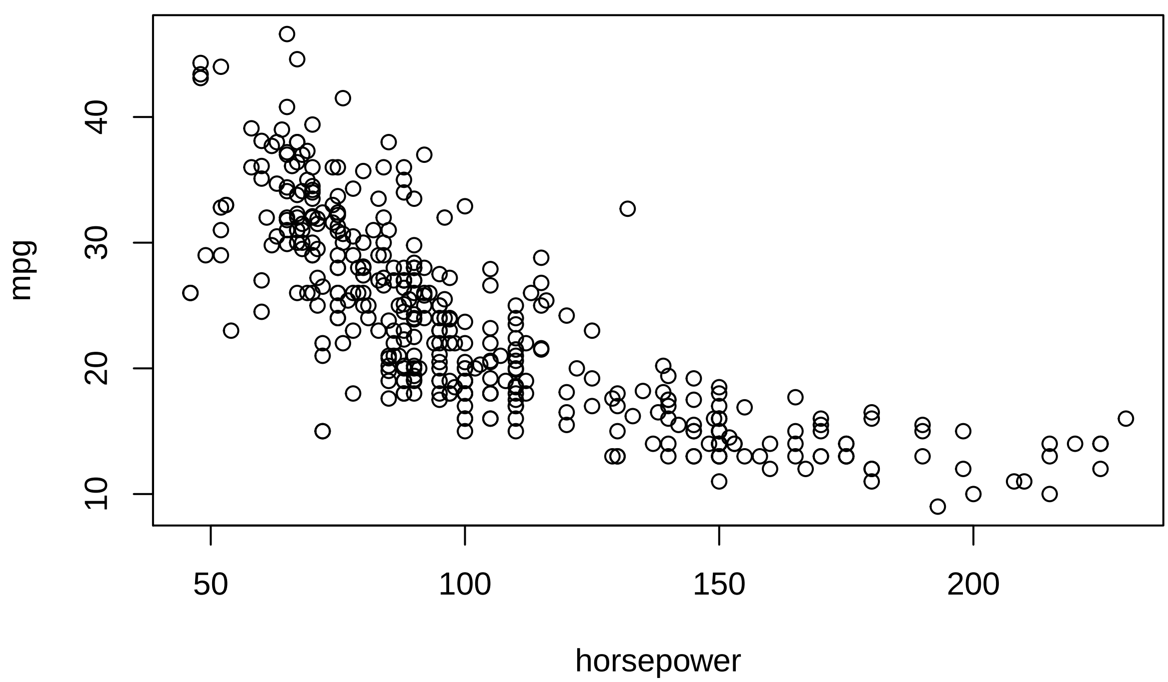
```
## [1] 392   9
```

```
names(Auto)
```

```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"
## [5] "weight"       "acceleration" "year"         "origin"
## [9] "name"
```

A plot is always a nice place to start with a new data set.

```
plot(mpg ~ horsepower, data = Auto)
```



The Leave-One-Out Cross-Validation (LCOOV) method.

First, lets run a `glm` model on the `Auto` data set.

```
glm_auto <- glm(mpg ~ horsepower, data = Auto)
```

Next, load the `boot` package and check out the documentation for the Cross-validation for Generalized Linear Models function, or `cv.glm`.

```
library(boot)
```

```
?cv.glm
```

Then, apply `cv.glm` function to the `Auto` data set, using `glm_auto` model, returning the delta parameter.

```
cv.glm(Auto, glm_auto)$delta
```

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

We can speed up the results by writing a function to use the formula displayed in section 5.2 (pg. 180) and then pass the `glm_auto` model to it.

```
loocv <- function(x){
          h <- lm.influence(x)$h
        mean((residuals(x)/(1-h))^2)
}
```

Is our new function faster? We can use the `system.time` function to compare both methods.

```
system.time(
cv.glm(Auto, glm_auto)$delta
)
```

```
##    user  system elapsed
##    1.22    0.00    1.22
```

```
system.time(
loocv(glm_auto)
)
```

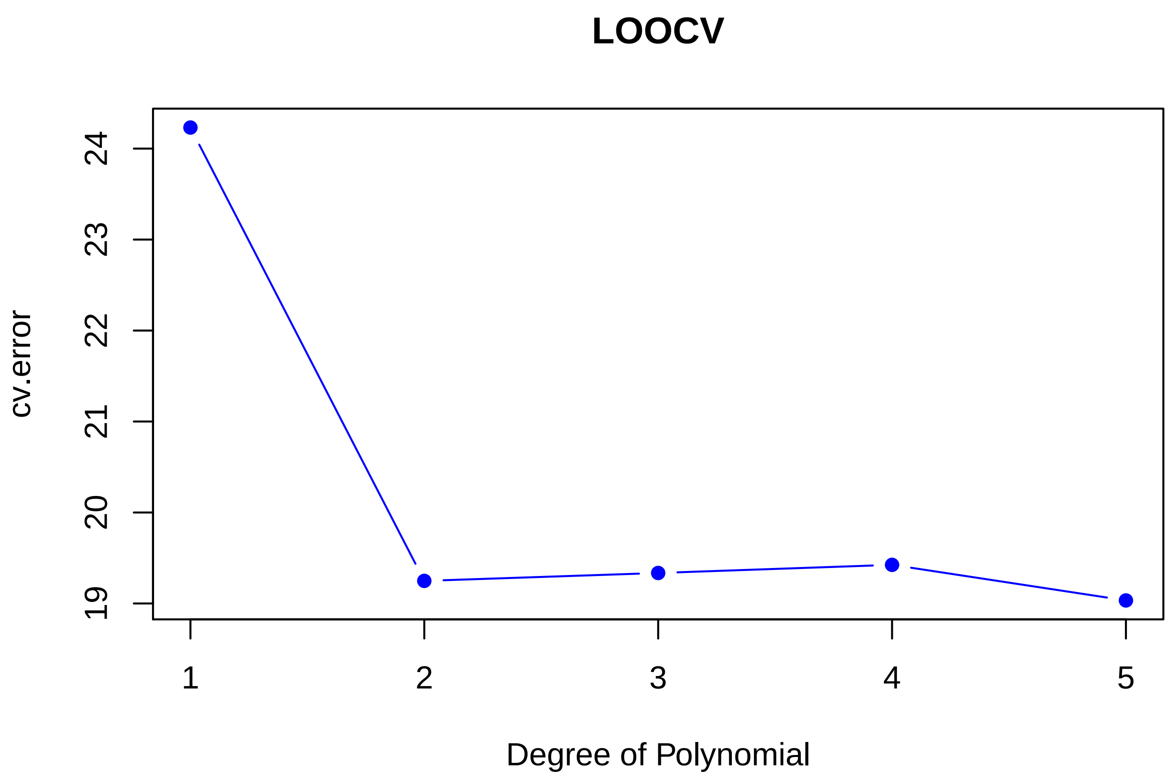
```
##    user  system elapsed
##       0       0       0
```

Next, lets use a `for` loop to efficiently create 5 new polynomial versions of the previous model, regressing `horsepower` against `mpg` and see if the results improve as polynomial order increases.

```
cv.error <- rep(0, 5)
degree <- 1:5

for(d in degree){
  glm.fit <- glm(mpg ~ poly(horsepower, d), data = Auto)
  cv.error[d] <- loocv(glm.fit)
}

plot(degree, cv.error, type = "b", col = "blue", pch = 16,
     main = "LOOCV", xlab = "Degree of Polynomial")
```



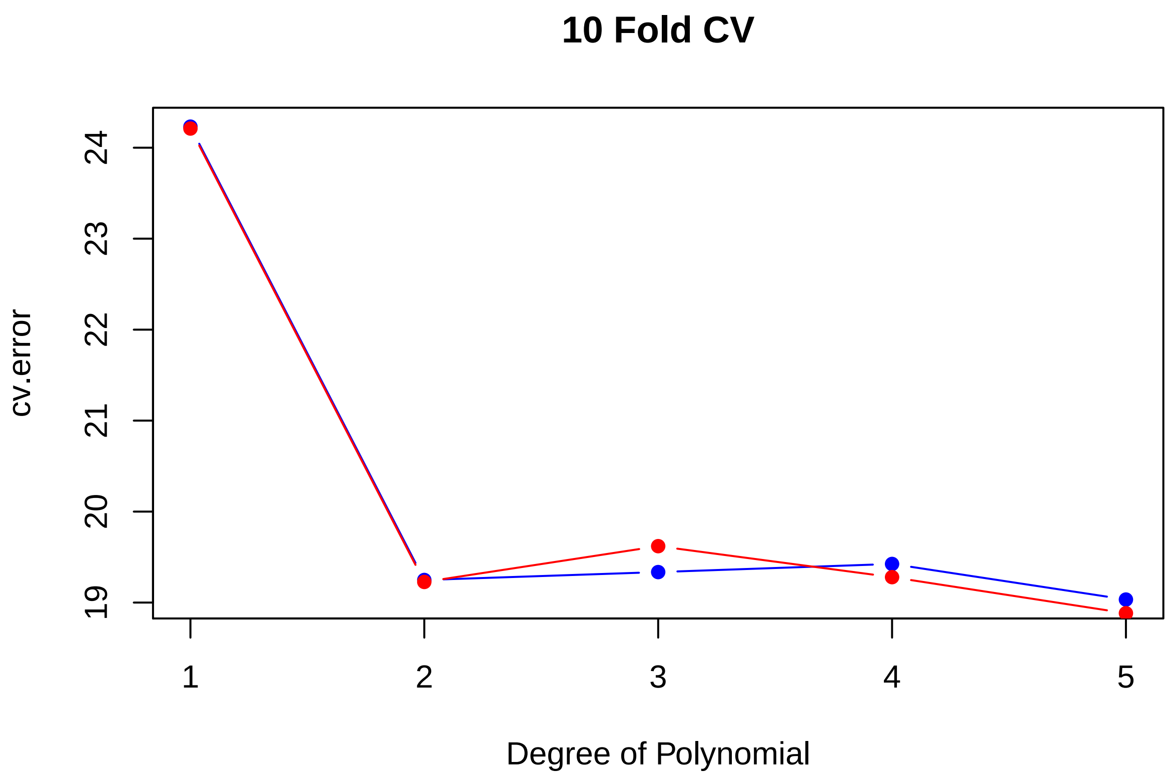
The10-fold Cross-Validation

First, initially the cv.error10 vector and ten run the loop.

```
cv.error10 <- rep(0, 5)

for(d in degree){
  glm.fit <- glm(mpg ~ poly(horsepower, d), data = Auto)
  cv.error10[d] <- cv.glm(Auto, glm.fit, K=10)$delta[1]
}

plot(degree, cv.error, type = "b", col = "blue", pch = 16,
     main = "10 Fold CV", xlab = "Degree of Polynomial")
lines(degree, cv.error10, type = "b", col = "red", pch = 16)
```



Bootstrap

Suppose that we wish to invest a fixed. sum of money in two financial assets that yield returns of X and Y, where X and Y are random quantities. We will invest a fraction of our money in X, and will invest the remaining $1 - \alpha$ in Y. We wish to choose α to minimize the total risk, or variance, of our investment. In other words, we want to minimize $Var(\alpha X + (1 - \alpha)Y)$. One can show that the value that minimizes the risk is given by

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$$

where $\sigma_X^2 = Var(X)$, $\sigma_Y^2 = Var(Y)$, and $\sigma_{XY} = Cov(X, Y)$.

However, the values of σ_X^2, σ_Y^2, and σ_{XY} are unkown. We can compute estimates for these quantities, $\hat{\sigma}_X^2$, $\hat{\sigma}_Y^2$, and $\hat{\sigma}_{XY}$, using a data set that contains measurments for X and Y.

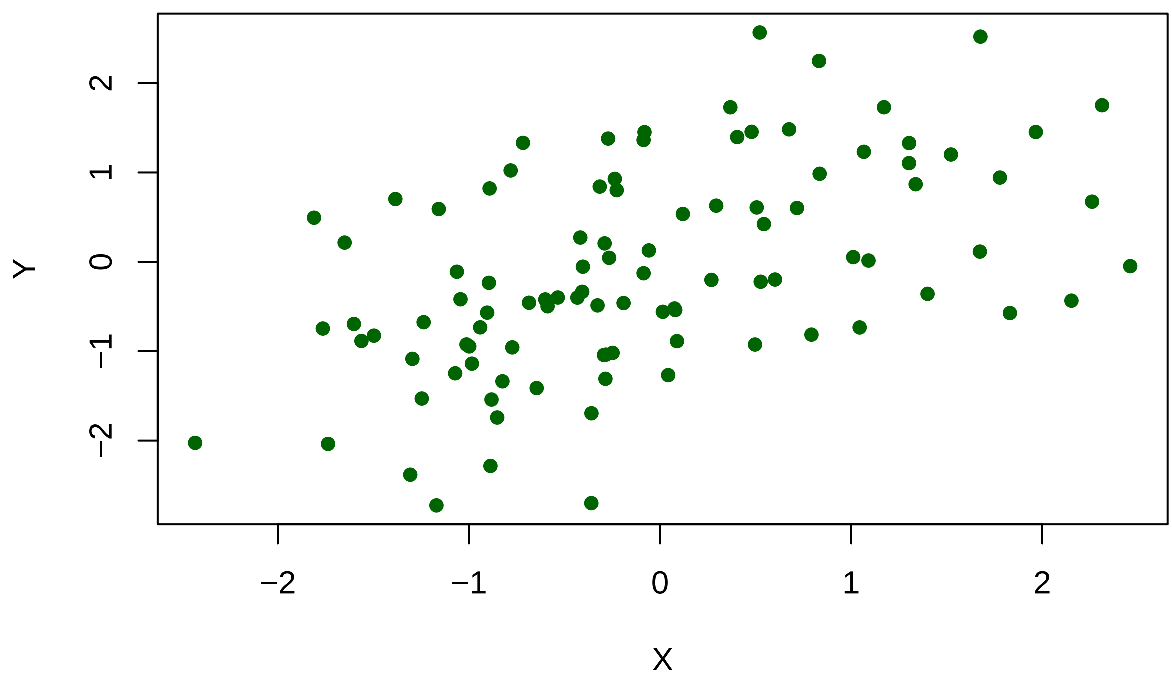
We can then estimate the value of α that minimizes the variance of our investment using:

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}$$

Load the `Portfolio` data set from the `ISLR` package, containing 100 returns for two assets, X and Y.

```
data("Portfolio")
```

```
plot(Y ~ X, data = Portfolio, col = "darkgreen", type = "p", pch = 16)
```




```
alpha <- function(x, y) {

          var_x <- var(x)
          var_y <- var(y)
          cov_xy <- cov(x, y)

(var_y - cov_xy)/(var_x + var_y - 2 *cov_xy)

}

alpha(Portfolio$X, Portfolio$Y)
```

```
## [1] 0.5758321
```

So what is the standard error of alpha? First, lets make a wrapper function

```r
alpha2 <- function(data, index){
                  with(data[index, ], alpha(X, Y))
}

alpha2(Portfolio, 1:100)
```

```
## [1] 0.5758321
```

```r
set.seed(1)
alpha2(Portfolio, sample(1:100, 100, replace = TRUE))
```

```
## [1] 0.5963833
```

```r
boot.out <- boot(Portfolio, alpha2, R = 1000)

boot.out$t0
```

```
## [1] 0.5758321
```

Finally, plot the bootstrap model object.

```r
plot(boot.out)
```