Lab 5 Resampling Methods

Problem Statement

Estimate the test MSE (mean squared error) by using

(I) Validation set approach

(II) `LOOCV`

(III) K-fold CV

Report which approach is the best.

Dataset

Use `Auto` dataset in the package of `ISLR`. Focus on the following pair of variables: `mpg` and `horsepower`.

Questions

(I) Validation set approach

1) Randomly pick half of the data as the training data. Remember to set a seed to make your result repeatable.

2) Build a linear regression model based the training data.

3) Estimate the test MSE based on the other half (as test data)

4) Now try to build polynomial regression of degree 2 and 3 using `lm(y~poly(x,i))`, where `y` is the response variable, `x` is the predictor variable and `i` is the highest degree of `x`. Compute the test MSE for the two models.

5) What conclusion could we draw from the above comparison of degree 1 (linear) and degree 2 (quadratic) and degree 3 (cubic) regression models?

6) Choose 10 different seeds. For each seed, calculate the test MSE for models of degree from 1 to 10. You may use a nested for-loop to do that. Plot the variability on the results. Can you obtain a similar plot as in Figure 1.

- Hint:

In order to do that you need to plot one curve first, and repeat the same procedure for another 9 times (using a for-loop) where each time a different seed is chosen.

In order to plot one curve, you need to obtain a vector of size 10, where each element of the vector records the test MSE of the model with degree `i (i = 1, 2, .10)`. This can be implemented by a for-loop to go through degree from 1 to 10.

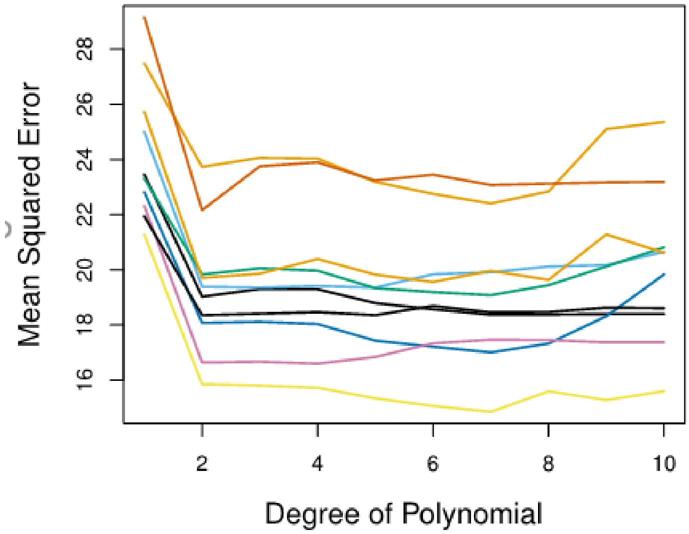

Figure 1: Validation Set Approach - MSE vs Degree of Polynomial

- Some examples on for-loop in R:

```
x <- c(2,5,3,9,8,11,6)
count <- 0
for (val in x) {
    if(val %% 2 == 0)  count = count+1  # %% returns the remainder of the division
}
print(count)
```

```
## [1] 3
```

The above program counts how many odd numbers there are in `x`.

Or

```
foo = seq(1, 100, by=2)
foo.squared = NULL

for (i in 1:50) {
    foo.squared[i] = foo[i]^2
}
```

See what happened to the vector `foo`.

(II) LOOCV

Function glm()

- In logistic regression: `glm(y~x,family="binomial",data=..)`
- In linear regression: `glm(y~x, data=..)`

```
> glm.fit=glm(mpg~horsepower,data=Auto)
> coef(glm.fit)
(Intercept)  horsepower
     39.936      -0.158
```

is the same as `lm(y ~ x, data = ...)`

```
> lm.fit=lm(mpg~horsepower,data=Auto)
> coef(lm.fit)
(Intercept)  horsepower
     39.936      -0.158
```

Function cv.glm() in boot library

- Produces a list with several components, including the cross-validation estimate for the test error: `delta`
- `cv.glm(data, glmfit, K)`

```
> library(boot)
> glm.fit=glm(mpg~horsepower,data=Auto)
> cv.err=cv.glm(Auto,glm.fit)
> cv.err$delta
    1     1
24.23 24.23
```

- `delta` is a vector of length two. For LOOCV, the two are the same.

7) Experiment on the LOOCV for increasingly complex polynomial fits. More specifically, write a for-loop to increase the degree `i`, as in `lm(y~poly(x,i))`, from 1 to 10 and record the LOOCV estimate for the test error for each degree.

8) Plot the result from 7) where x-axis is the degree `i` and y-axis is the LOOCV estimate for the test error. Can you plot a similar one as in Figure 2?

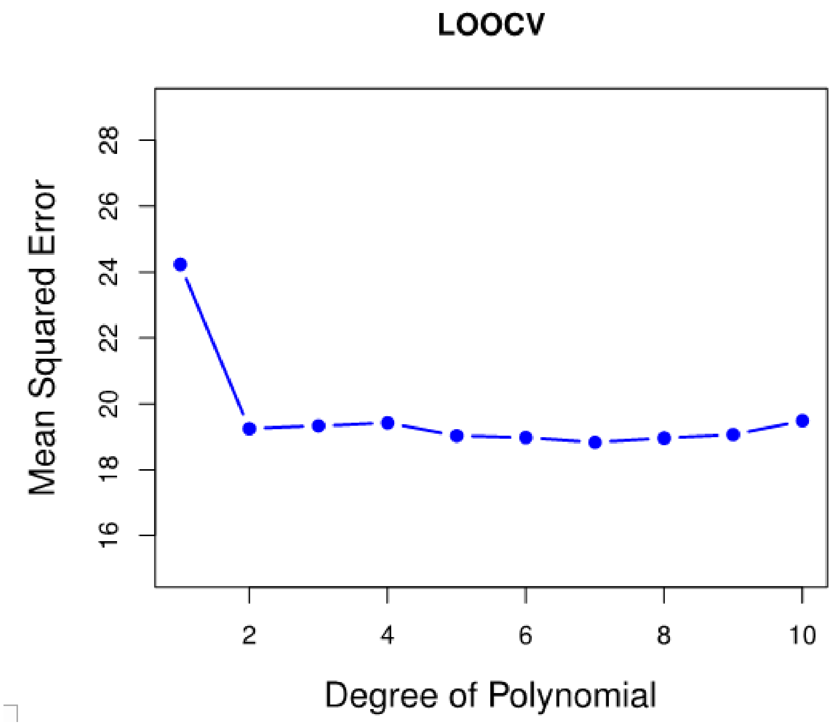

Figure 2: LOOCV - MSE vs Degree of Polynomial

(III) K-fold CV

Implement k-fold CV by passing the argument K in `cv.glm(data, glmfit, cost, K)`.

The errors are recorded in `delta`. There are two numbers associated with `delta`:

- The first number is the raw/standard CV estimate of prediction error.
- The second number is the adjusted CV estimate. The adjustment is designed to compensate for the bias introduced by not using leave-one-out cross-validation.

It is sufficient to report the raw CV error to estimate the test errors.

The following three questions can be answered by one chunk of code.

9) Set a seed. Write a for-loop to increase the degree i, as in lm(y~poly(x,i)), from 1 to 10 and record the 10-fold CV estimate for the test error for each degree.

10) Plot the result from 9) where x-axis is the degree i and y-axis is the 10-fold CV estimate for the test error.

11) Set 9 different seeds and repeat 9) and 10). Plot all the results into one plot like the one in Figure 3.

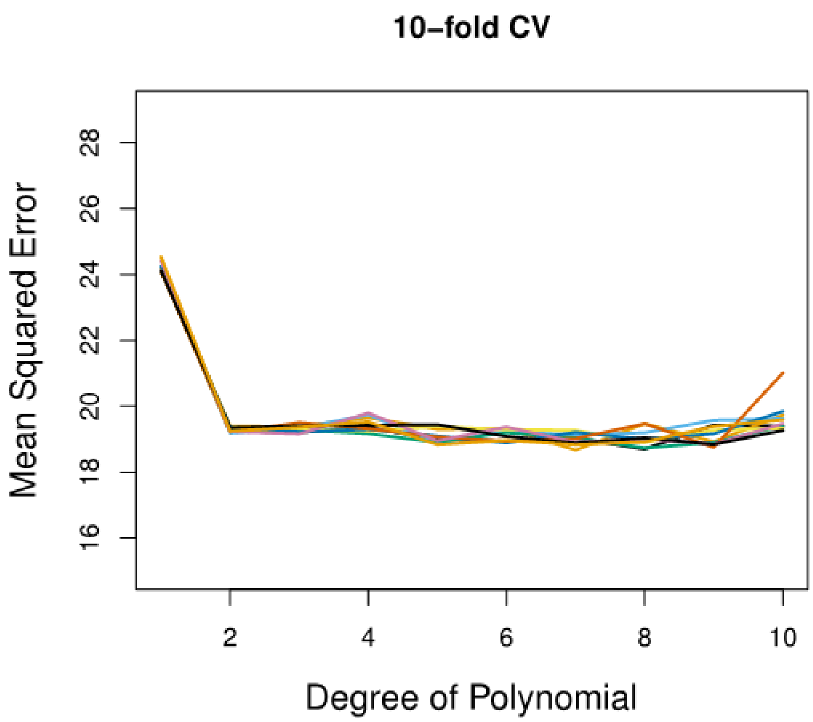

Figure 3: 10-fold CV - MSE vs Degree of Polynomial