HW4

Nathan Krieger

2026-02-17

Problem 1

(a)

Hypothesis testing is valuable here because while the summary shows us that the estimate is non-zero, it doesn't mean that the predictor X_3 actually influences Y. It would be rare to see an estimate that is truly zero due to noise.

We can use hypothesis testing to determine if the estimate is a real signal or just random luck.

In the case of X_3, the estimate is non-zero, but the P-value is 0.334 which is much higher than the $\alpha = 0.05$ threshold, therefore, X_3 is not a good predictor of Y

(b)

I disagree with this claim. Even if we knew the true $f(X)$, there is still error (ϵ) that we cannot predict.

$Y = f(X) + \epsilon$

(c)

False. The expected test MSE is evaluated based on the data (x_0, y_0) that's not part of the training set.

(d)

True. Sometimes a model that is too complex can overfit the noise. If we remove a predictor we may increase bias but also decrease the variance. This leads to a better test MSE.

(e)

False. The expected test MSE includes $Var(\epsilon)$ so theres no way it can be smaller than it.

(f)

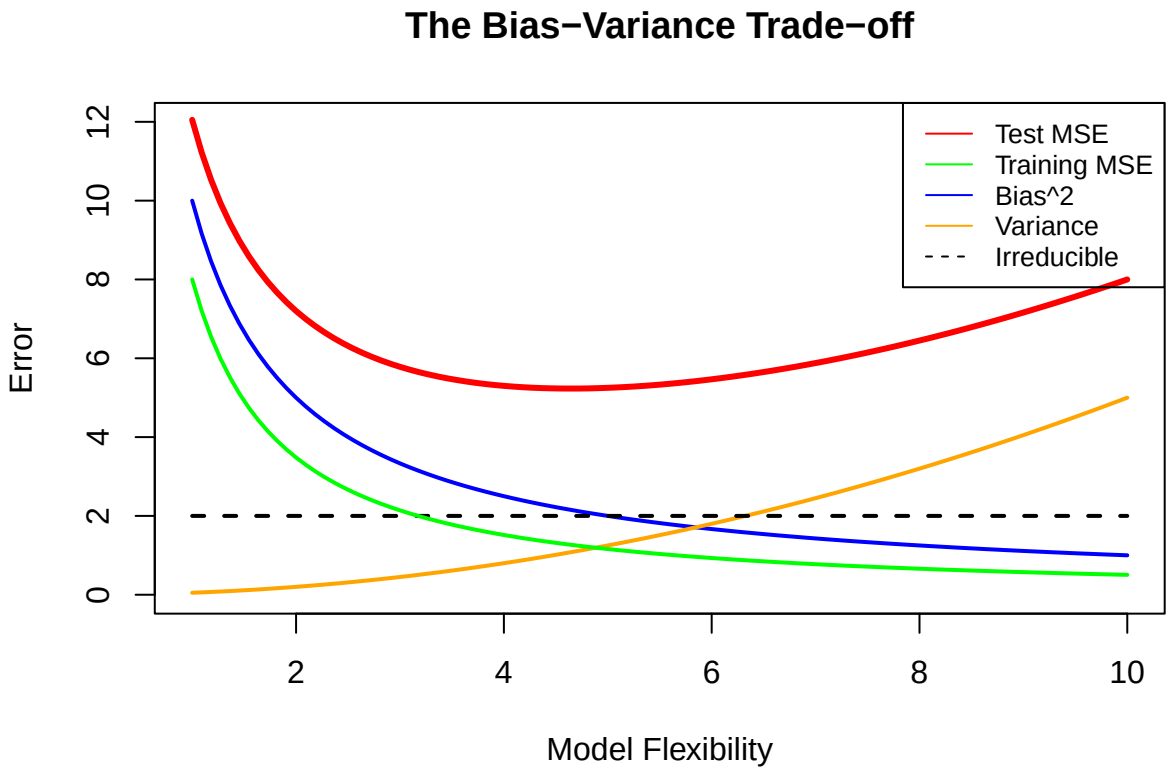
True. We could fit a model to be exactly the same (overfitting) which can get rid of the irreducable error.

Problem 2

(a)

```
x <- seq(1, 10, length.out = 100)

bias_sq   <- 10 / x
variance  <- 0.05 * x^2
```

```r
irred     <- rep(2, 100)
test_mse  <- bias_sq + variance + irred
train_mse <- 8 / x^1.2

plot(x, test_mse, type = "n", ylim = c(0, 12),
     xlab = "Model Flexibility", ylab = "Error",
     main = "The Bias-Variance Trade-off")

lines(x, bias_sq,   col = "blue",   lwd = 2) # Squared Bias
lines(x, variance,  col = "orange", lwd = 2) # Variance
lines(x, irred,     col = "black",  lwd = 2, lty = 2) # Irreducible Error
lines(x, test_mse,  col = "red",    lwd = 3) # Test MSE
lines(x, train_mse, col = "green",  lwd = 2) # Training MSE

legend("topright", cex = 0.8,
       legend = c("Test MSE", "Training MSE", "Bias^2", "Variance", "Irreducible"),
       col = c("red", "green", "blue", "orange", "black"),
       lty = c(1, 1, 1, 1, 2))
```



(b)

Expected test MSE: A measurement of how well the model predicts new unseen data.

Training MSE: How well the model performs on the exact data that it used to learn.

Bias: The error that comes from using a model that is too simple to capture the true pattern. It consistently misses the mark in a predictable way.

Variance: How much the model's predictions change if you trained it on a different set of data.

Irreducible Error: The error that stays even if you have a perfect model. It is inherent to the problem itself.

(c)

TODO

Squared bias:

Variance: Increasing because a model that is very flexible follows the data very closely. This means that if you add new data the line would have to change its entire shape to accommodate them.

Expected test MSE:

Training MSE: This is decreasing because as the flexibility increases it is able to get closer to more of the datapoints and accommodate for the exact shape of the data.

Irreducible error: This is a flat line. Since it's the variance of the random noise (ϵ), and ϵ is inherent to the data and doesn't care about the model, it's a horizontal line.

(d)

TODO

```
n <- 100

x = seq(0, 10, length.out = n)
error = rnorm(n,0,1)

y1 <- x + error

m1 <- lm(y1 ~ x)

beta_0 <- 1
beta_1 <- 2
beta_2 <- 3
beta_3 <- 4

y2 <- beta_0 + beta_1 * x + beta_2 * x^2 + beta_3 * x^3 + error

m2 <- lm(y2 ~ poly(x, 3))

summary(m1)
```

```
##
## Call:
## lm(formula = y1 ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8243 -0.7342  0.1738  0.7983  3.0061
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.11259    0.21238    0.53    0.597
## x            0.97989    0.03669   26.70   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 1.07 on 98 degrees of freedom
## Multiple R-squared:  0.8792, Adjusted R-squared:  0.878
## F-statistic: 713.2 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
summary(m2)
```

```
##
## Call:
## lm(formula = y2 ~ poly(x, 3))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6181 -0.6500  0.1227  0.7628  2.7728
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.122e+03  1.064e-01 10536.8   <2e-16 ***
## poly(x, 3)1 1.146e+04  1.065e+00 10769.9   <2e-16 ***
## poly(x, 3)2 4.792e+03  1.065e+00  4501.5   <2e-16 ***
## poly(x, 3)3 7.786e+02  1.065e+00   731.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.064 on 96 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 4.56e+07 on 3 and 96 DF,  p-value: < 2.2e-16
```

(e)

Problem 3

Problem 4

(a)

(b)

(i)

(ii)

(c)

```
set.seed(1)
x = rnorm(100)
error = rnorm(100)
y = x - 2*x^2 + error
```

(d)

```
set.seed(1)

Data = data.frame(x, y)

n <- nrow(Data)
```

```r
MSE_M1 = rep(0,n) #vector
MSE_M2 = rep(0,n) #vector
MSE_M3 = rep(0,n) #vector
MSE_M4 = rep(0,n) #vector

for(i in 1:n){
  test = Data[i, ]
  train = Data[-i, ]

  M1 <- lm(y ~ x, data = train)

  M2 <- lm(y ~ poly(x, degree = 2, raw = TRUE), data = train)

  M3 <- lm(y ~ poly(x, degree = 3, raw = TRUE), data = train)

  M4 <- lm(y ~ poly(x, degree = 4, raw = TRUE), data = train)

  M1_test = predict(M1, newdata = test)
  M2_test = predict(M2, newdata = test)
  M3_test = predict(M3, newdata = test)
  M4_test = predict(M4, newdata = test)


  MSE_M1[i] = (test$y - M1_test)^2
  MSE_M2[i] = (test$y - M2_test)^2
  MSE_M3[i] = (test$y - M3_test)^2
  MSE_M4[i] = (test$y - M4_test)^2

}

LOOCV_MSE_M1 = mean(MSE_M1)
LOOCV_MSE_M2 = mean(MSE_M2)
LOOCV_MSE_M3 = mean(MSE_M3)
LOOCV_MSE_M4 = mean(MSE_M4)

LOOCV_MSE_M1
```

```
## [1] 7.288162
```

```r
LOOCV_MSE_M2
```

```
## [1] 0.9374236
```

```r
LOOCV_MSE_M3
```

```
## [1] 0.9566218
```

```r
LOOCV_MSE_M4
```

```
## [1] 0.9539049
```

(e)

```r
set.seed(10101)
```

```r
Data = data.frame(x, y)

n <- nrow(Data)

MSE_M1 = rep(0,n) #vector
MSE_M2 = rep(0,n) #vector
MSE_M3 = rep(0,n) #vector
MSE_M4 = rep(0,n) #vector

for(i in 1:n){
  test = Data[i, ]
  train = Data[-i, ]

  M1 <- lm(y ~ x, data = train)

  M2 <- lm(y ~ poly(x, degree = 2, raw = TRUE), data = train)

  M3 <- lm(y ~ poly(x, degree = 3, raw = TRUE), data = train)

  M4 <- lm(y ~ poly(x, degree = 4, raw = TRUE), data = train)

  M1_test = predict(M1, newdata = test)
  M2_test = predict(M2, newdata = test)
  M3_test = predict(M3, newdata = test)
  M4_test = predict(M4, newdata = test)


  MSE_M1[i] = (test$y - M1_test)^2
  MSE_M2[i] = (test$y - M2_test)^2
  MSE_M3[i] = (test$y - M3_test)^2
  MSE_M4[i] = (test$y - M4_test)^2

}

LOOCV_MSE_M1 = mean(MSE_M1)
LOOCV_MSE_M2 = mean(MSE_M2)
LOOCV_MSE_M3 = mean(MSE_M3)
LOOCV_MSE_M4 = mean(MSE_M4)

LOOCV_MSE_M1
```

```
## [1] 7.288162
```

```r
LOOCV_MSE_M2
```

```
## [1] 0.9374236
```

```r
LOOCV_MSE_M3
```

```
## [1] 0.9566218
```

```r
LOOCV_MSE_M4
```

```
## [1] 0.9539049
```

The results for part e remained the exact same as part d even with a different seed value. This is because LOOCV is deterministic and works by leaving just one data point out at a time and doing it n times. This

means that each data point is left out exactly once regardless of the seed number.

(f)

M2 (degree 2) had the smallest LOOCV error. This is what I expected because $f(X)$ is an equation with a degree of 2.

(g)

LOOCV trains the model on n - 1 observations so it gives a better, more accurate estimate of the test error when comparing it to a standard test, training set approach.

(h)

As k increases, the bias decreases. This is because the training data gets larger (closer to the size of the full data set) as k gets larger.

(i)

TODO

As k increases, the variance increases. This is because as k increases, the size of the testing data decreases which increases variance.