Homework 2

Yiming Gao (NetID: yimingg2)

2017/2/5

Contents

Exercise 1

In this exercise, we investigate the bias-variance tradeoff when estimating the function f defined below.

```
f = function(x1, x2) {
  x1 ^ 3 + x2 ^ 3
}
```

The following code defines the data generating process and should we used to simulate data.

```
get_sim_data = function(f, sample_size = 100) {
  x1 = runif(n = sample_size, min = -1, max = 1)
  x2 = runif(n = sample_size, min = -1, max = 1)
  y = f(x1, x2) + rnorm(n = sample_size, mean = 0, sd = 0.5)
  data.frame(x1, x2, y)
}
```

Use simulation to investigate the bias and variance of *five* models at the point $\mathbf{x} = (x_1, x_2) = (0.75, 0.95)$. The five models are of the form

- `y ~ poly(x1, degree = k) + poly(x2, degree = k)`

for $k = 1, 2, 3, 4, 5$. Use 500 simulated samples each of size 200. Before performing the simulations, I set the seed as 650379994.

```
uin = 650379994
set.seed(uin)
```

```r
n_sims = 500
x01 = 0.75
x02 = 0.95
pred = matrix(0, nrow = n_sims, ncol = 5)

for (i in 1:n_sims){
  sim_data = get_sim_data(f, sample_size = 200)

  fit1 = lm(y ~ poly(x1, degree = 1) + poly(x2, degree = 1), data = sim_data)
  fit2 = lm(y ~ poly(x1, degree = 2) + poly(x2, degree = 2), data = sim_data)
  fit3 = lm(y ~ poly(x1, degree = 3) + poly(x2, degree = 3), data = sim_data)
  fit4 = lm(y ~ poly(x1, degree = 4) + poly(x2, degree = 4), data = sim_data)
  fit5 = lm(y ~ poly(x1, degree = 5) + poly(x2, degree = 5), data = sim_data)

  pred[i, ] <- c(
    predict(fit1, newdata = data.frame(x1 = x01, x2 = x02)),
    predict(fit2, newdata = data.frame(x1 = x01, x2 = x02)),
    predict(fit3, newdata = data.frame(x1 = x01, x2 = x02)),
    predict(fit4, newdata = data.frame(x1 = x01, x2 = x02)),
    predict(fit5, newdata = data.frame(x1 = x01, x2 = x02))
  )
}
```

We simulate 500 samples each of size 200 for building our models.

```r
# Tradeoff
eps = rnorm(n = n_sims, mean = 0, sd = 0.3)
y0 = f(x01, x02) + eps

# get bias
get_bias = function(estimate, truth) {
  mean(estimate - truth)
}

# get mean square error
get_mse = function(estimate, truth) {
  mean((estimate - truth) ^ 2)
}

# suppress scientific notation
```

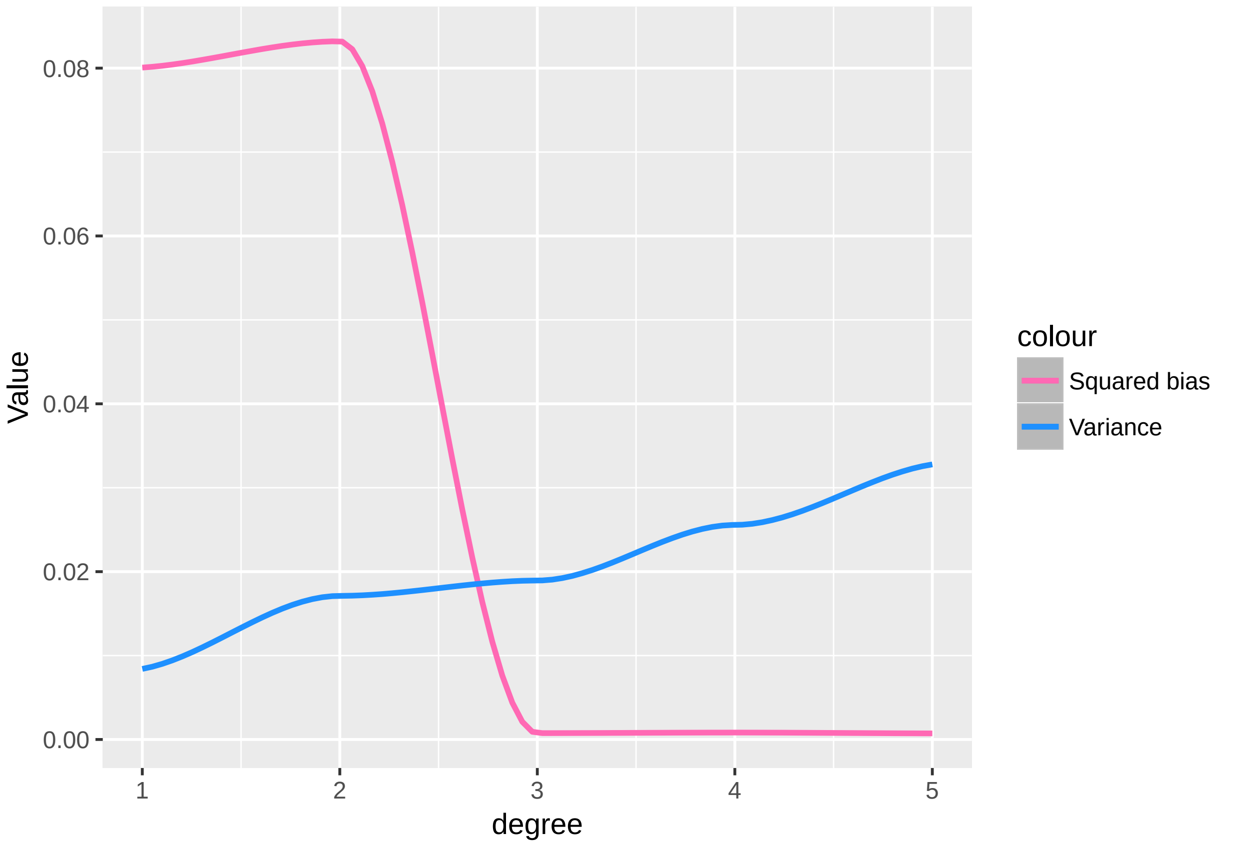
```r
options(scipen=999)
bias = apply(pred, 2, get_bias, y0)
variance = apply(pred, 2, var)
mse = apply(pred, 2, get_mse, y0)
```

Model	Squared Bias	Variance	MSE
`fit1`	0.0800646	0.0084093	0.173899
`fit2`	0.0832004	0.0171158	0.1868814
`fit3`	0.0007602	0.0189415	0.1034944
`fit4`	0.000826	0.0255703	0.1114743
`fit5`	0.0007265	0.0327806	0.1168097

We summarize the table above, which contains **Model**, **squared bias**, **variance** and **MSE** of the estimates. We notice that `fit3` and `fit5` has the lowest squared bias. However, `fit5` has the highest variance among five models.

```r
mydata = data.frame("degree" = c(1:5), "Bias" = bias^2, "Variance" = variance)

library(ggplot2)
ggplot(mydata, aes(degree))+
  geom_smooth(aes(y = Bias, colour = "Squared bias"))+
  geom_smooth(aes(y = Variance, colour = "Variance")) +
  xlab("degree") + ylab(label = "Value") +
  # add legend
  scale_colour_manual(values = c("hotpink", "dodgerblue"))
```



We plot **Squared bias** and **Variance** values against the **degree** of the polynomials used. From the plot, we know that squared bias decreases significantly when degree increases to 3. The variance values appear to have a positive linear trend as degree increasing. So we may consider `model 3` achieves a balance between bias and variance.

Exercise 2

We need to find a model that satisfies:

- Train RMSE less than 1.08
- Test RMSE less than 1.01

The codes for building model and some relevant functions are as follows:

```
# user-defined function to calculate rmse
rmse = function(actual, predicted) {
  sqrt(mean((actual - predicted) ^ 2))
}

get_rmse = function(model, data, response) {
```

```r
  rmse(actual = data[, response],
       predicted = predict(model, data))
}

model = lm(y ~ . + x1:x2 + I(x3^2) + I(x4^3), data = train_data)

complexity = length(coef(model)) - 1
train_rmse = get_rmse(model, data = train_data, response = "y")
test_rmse = get_rmse(model, data = test_data, response = "y")
```

After a number of trials, we get our final model and its metrics:

- `y ~ . + x1:x2 + I(x3^2) + I(x4^3)`

Model	Train RMSE	Test RMSE	Number of parameters
`model`	1.03	0.978	7

Train RMSE is $1.030 < 1.08$. Test RMSE is $0.978 < 1.01$.

Exercise 3

We want to build a logistic regression model using `mpg` as the response.

We use the training data to train a classifier which achieves:

- Train Accuracy greater than 0.89
- Test Accuracy greater than 0.89

```r
library(caret)
model_glm = glm(mpg ~ ., data = auto_train_data, family = "binomial")

# obtain the predicted probabilities, use 0.5 as cutoff
glm_train_pred = ifelse(predict.glm(model_glm, newdata = auto_train_data,
                                    type = "response") > 0.5, 1, 0)
glm_test_pred = ifelse(predict.glm(model_glm, newdata = auto_test_data,
                                   type = "response") > 0.5, 1, 0)

train_accuracy = mean(glm_train_pred == auto_train_data$mpg) # 0.897
test_accuracy = mean(glm_test_pred == auto_test_data$mpg) # 0.924
```

```r
# confusion matrix for test data
test_tab = table(predicted = glm_test_pred, actual = auto_test_data$mpg)

# sensitivity and specificity
test_mat = confusionMatrix(test_tab, positive = "1")
```

We finally train the following classifier using training set:

- `mpg ~ cylinders + displacement + horsepower + weight + acceleration + year`

It has

- `Train Accuracy` $= 0.897 > 0.89$
- `Test Accuracy` $= 0.924 > 0.89$

The confusion matrix for test data is:

```
##          actual
## predicted  0  1
##         0 44  1
##         1  6 41
```

And other metrics for the test data are summarized in the following table:

Test sensitivity	Test specificity
0.976	0.88