PS4 Solutions

Teaching Staff

February 2023

Question 1

1

```
dat = read.csv("../data/vote92plus.csv")
set.seed(5000)
k = sample(1:nrow(dat), round(nrow(dat)*2/3), replace = FALSE)
train.dat = dat[k,]
test.dat = dat[-k,]
```

2-5

First write a helper function for mean square error.

```
mse = function(x, y){
  ## Calculate mean square error
  return(mean((x - y)^2))
}
```

Next, we can write a wrapper function that will run our model fit, calculate the train and test mean square error, and return both as a list.

```
model = function(formula, outcome, train, test){
  m = lm(as.formula(formula), data = train)
  trainMSE = mse(train[[outcome]],predict(m, train))
  testMSE = mse(test[[outcome]], predict(m, test))
  return(list(trainMSE = trainMSE, testMSE=testMSE))
}
```

Next, we write up the formulas as specified by the problem.

```
f ="clintondis~vote + dem + rep + female + persfinance + natlecon"
f2 = paste(f,"+", paste0("fake", 1:5, collapse = "+"))
f3 = paste(f, "+",paste0("poly(", paste0("fake", 1:5),",
                         degree = 2, raw = TRUE)",
                         collapse = "+"))
f4 = paste(f, "+",paste0("poly(", paste0("fake", 1:5),",
                         degree = 3, raw = TRUE)",
                         collapse = "+"))

formulas = c(f, f2, f3, f4)
```

Now we can simply fit the models and store the results in a list.

```
models = list()
for(i in 1:length(formulas)){
```

```
  models[[i]] = model(formulas[i], "clintondis", train.dat, test.dat)
}
```

6-8

While not required by the problem, we can write a little helper function to make our table.

```
makeTable = function(l){
  table = data.frame(Model = LETTERS[1:4],
                     matrix(unlist(l),
                            nrow = length(l), byrow = T))
  names(table) = c("Model","Train MSE", "Test MSE")
  return(table)
}
```

Model	Train MSE	Test MSE
A	11.34638	13.02046
B	11.31480	13.05915
C	11.18850	13.22988
D	11.12317	13.33001

The pattern that we see is that the training MSE goes down mechanically as we add more variables. The test MSE goes up when we include fake variables and their associated squared and cubic terms. Both are explained at a high level by the fact that the model is overfitting on the training data when we include fake data.

9

```
set.seed(202)
k2 = sample(1:nrow(dat), round(nrow(dat)*2/3), replace = FALSE)
train.dat2 = dat[k2,]
test.dat2 = dat[-k2,]

## An alternative way to the for loop
models2 = lapply(formulas, FUN = model,
                 outcome = "clintondis",
                 train = train.dat2,
                 test = test.dat2)
```

Model	Train MSE	Test MSE
A	10.74963	14.22471
B	10.62579	14.66784
C	10.48770	14.91585
D	10.36466	15.23320

The specific values of the results are different, which makes sense because we have taken a different random sample and unless every value is the same we should expect some differences between samples. The overall takeaway from the results is the same.

Question 2

1

Choice: Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.

Reason: By using LASSO, we put some contraints on the total size of coefficients, thus making our model less flexible, decreasing model variance, and also introducing some bias in coefficient estimates. LASSO will do us better in terms of prediction accuracy when the increased bias (squared term) is smaller than that the decreased variance.

2

a *iii* is correct. A standard OLS model is fit according to the objective criterion of minimizing the RSS of the data used in the fit (i.e. the training data). As a result, any deviation from OLS's standard fit will lead to a larger RSS. A positive λ puts a penalty on OLS estimated coefficients being non-zero, so assuming the standard OLS estimates are indeed non-zero, increasing the size of λ will increase that penalty and increasingly induce the coefficients to get further away from their standard OLS estimated values. By inducing the solutions to be increasingly further away from the standard OLS solutions, the training RSS will steadily increase.

b *ii* is correct (or at least, most likely). The magic of ridge regression is that, even though it makes the training RSS increase steadily, it can make the test RSS decrease, at least initially. The reason is that putting a penalty on large coefficients ends up reducing the variance of the model's predictions. It does, also, increase the squared bias, but typically, the variance will decrease at first at a faster rate than will the squared bias increase. However, as we have seen with the introduction/reduction of flexibility in all of our methods/models, the gradual reduction of flexibility (which is what the increasing of λ does) will eventually make it such that the squared bias increases at a faster rate than the variance decreases. The result of all of this is that that test RSS will at first decrease, but then increase, as λ increases (i.e. as the model is made less flexible). However, it is also possible for (iii) to be correct. The OLS model may indeed be the best for the test error for certain datasets, which mean test error will simply increase.

c *iv* is correct. As explained above, putting a penalty (λ) on the regression coefficients shrinks them toward zero, which consequently decreases the flexibility of the model and decreases the variance.

d *iii* is correct. As explained above, putting a penalty (λ) on the regression coefficients shrinks them toward zero, which consequently decreases the flexibility of the model and increases the (squared) bias.

e *v* is correct. The irreducible error is irreducible and does not depend upon the model. It cannot be modeled and it does not change.