512 Homework #7

Brody Vogel

3/29/2018

Preparation

```
set.seed(1234)
library(ISLR)
library(glmnet)
```

```
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-13
```

Problem 1 (6.8 #11)

a)

It looks like the lasso, trained on all the data, eliminates all the predictors except "rad". The lambda chosen by cross validation, in this case, was .061.

```
library(MASS)
attach(Boston)

x <- model.matrix(crim ~ .-crim, data = Boston)

lasso.mod <- cv.glmnet(x, Boston$crim, alpha = 1)

#plot(lasso.mod)

coef(lasso.mod)
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                     1
## (Intercept) 1.0894283
## (Intercept) .
## zn          .
## indus       .
## chas        .
## nox         .
## rm          .
## age         .
## dis         .
## rad         0.2643196
## tax         .
## ptratio     .
## black       .
## lstat       .
## medv        .
```

```r
bestlam <- lasso.mod$lambda.min

bestlam
```

```
## [1] 0.06179934
```

b)

When I used the lambda that minimized the training error, the lasso regressed *all* the predictor coefficients to 0, and so I used the 1-standard-error lambda. This regressed all the predictor coefficients to 0 except "rad". My proposed model, then, is:

$\hat{crim} = 2.75 + .109 \times rad$

Unfortunately, there are some massive outliers in the data, and so the MSE for the test data was still very high (51.05).

```r
train = sample(c(TRUE, FALSE), nrow(Boston), rep = TRUE)
test = (!train)

training <- Boston[train, ]
testing <- Boston[test, ]

x <- model.matrix(crim ~ .-crim, data = training)
x1 <- model.matrix(crim~ .-crim, data = testing)

lasso.mod1 <- cv.glmnet(x, training$crim, alpha = 1)

lasso.pred <- predict(lasso.mod1, s = lasso.mod1$lambda.1se, newx = x1)
mean((lasso.pred - testing$crim)^2)
```

```
## [1] 51.05103
```

```r
coef(lasso.mod1)
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                     1
## (Intercept) 2.7529073
## (Intercept) .
## zn          .
## indus       .
## chas        .
## nox         .
## rm          .
## age         .
## dis         .
## rad         0.1091921
## tax         .
## ptratio     .
## black       .
## lstat       .
## medv        .
```

c)

My proposed model does not contain all the predictors. The lasso regresses the coefficients for some predictors to 0, and in this case it regressed *all* of those coefficients to 0 except "rad".

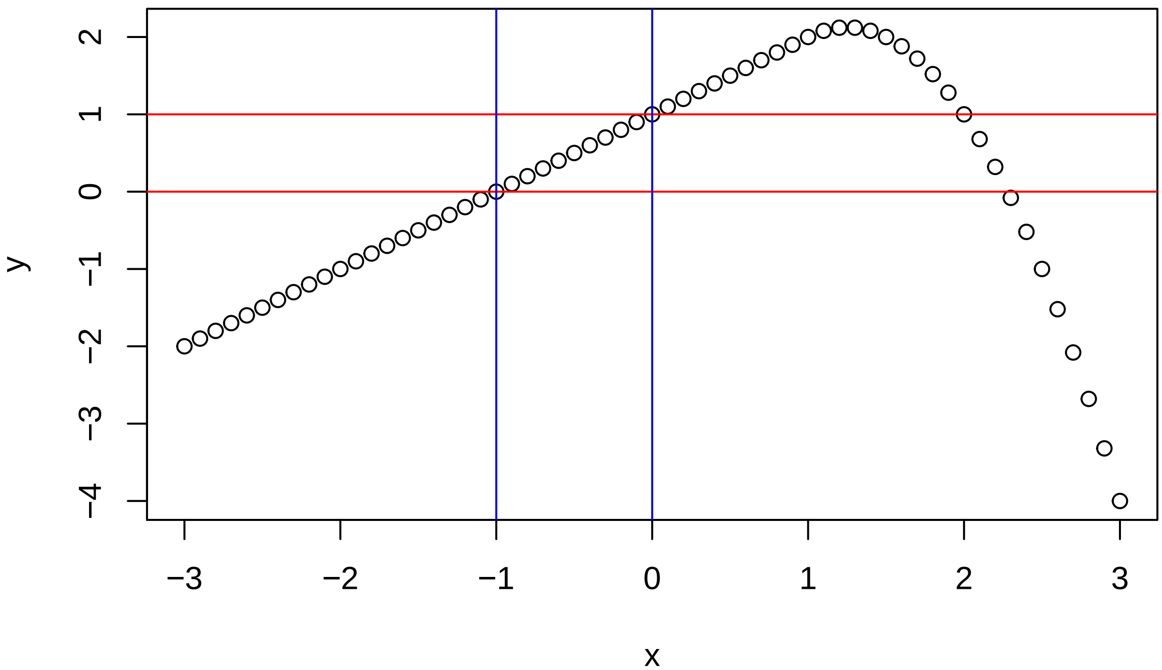
Problem 2 (7.9 #3)

The y-intercept is at x = -1; the x-intercept is at y = 1.

The slope is 1 from x = -2 to x = 1; it then turns negative at x = 1 and decreases as x grows.

```r
x <- seq(-3, 3, .1)
y <- c()
for (num in x) {
  if (num < 1) {
    y <- c(y, 1 + num)
  }
  else {
    y <- c(y, 1 + num - 2*(num - 1)^2)
  }
}

plot(x, y)
abline(h = 0, col = 'red')
abline(h = 1, col = 'red')
abline(v = -1, col = 'blue')
abline(v = 0, col = 'blue')
```



Problem 3 (7.9 #9a-c)

a)

The regression model outputs:

$\hat{nox} = .934 - .18 \times dis + .022 \times dis^2 - .0008 \times dis^3$

each of the coefficients tested as statisticall significant. The plot suggests the fit is pretty good.

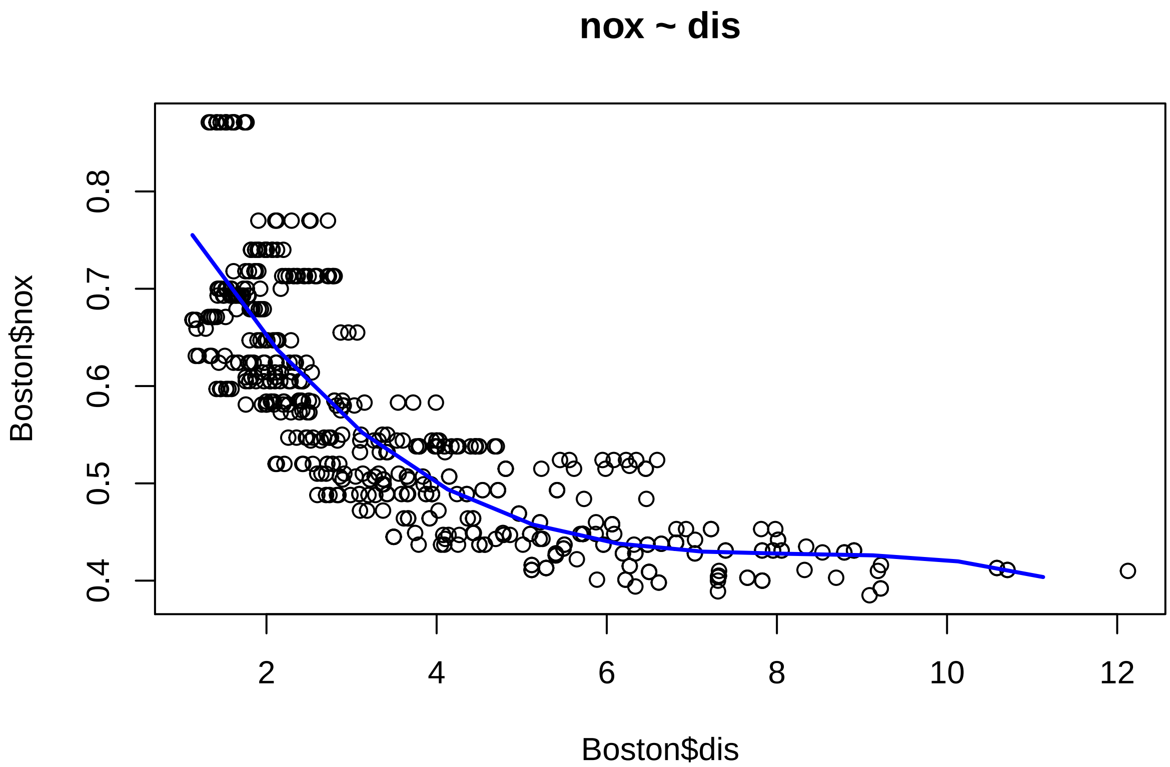
```
mod <- lm(nox~poly(dis, 3, raw = T), data = Boston)

summary(mod)
```

```
##
## Call:
## lm(formula = nox ~ poly(dis, 3, raw = T), data = Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.121130 -0.040619 -0.009738  0.023385  0.194904
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.9341281  0.0207076  45.110  < 2e-16 ***
## poly(dis, 3, raw = T)1 -0.1820817  0.0146973 -12.389  < 2e-16 ***
## poly(dis, 3, raw = T)2  0.0219277  0.0029329   7.476 3.43e-13 ***
## poly(dis, 3, raw = T)3 -0.0008850  0.0001727  -5.124 4.27e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06207 on 502 degrees of freedom
## Multiple R-squared:  0.7148, Adjusted R-squared:  0.7131
## F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
dislims <- range(Boston$dis)
dis.grid <- seq(from=dislims[1],to=dislims[2])
preds <- predict(mod, newdata=list(dis=dis.grid), se=TRUE)

plot(Boston$dis, Boston$nox, xlim=dislims, main = "nox ~ dis")
lines(dis.grid, preds$fit, lwd=2, col="blue")
```



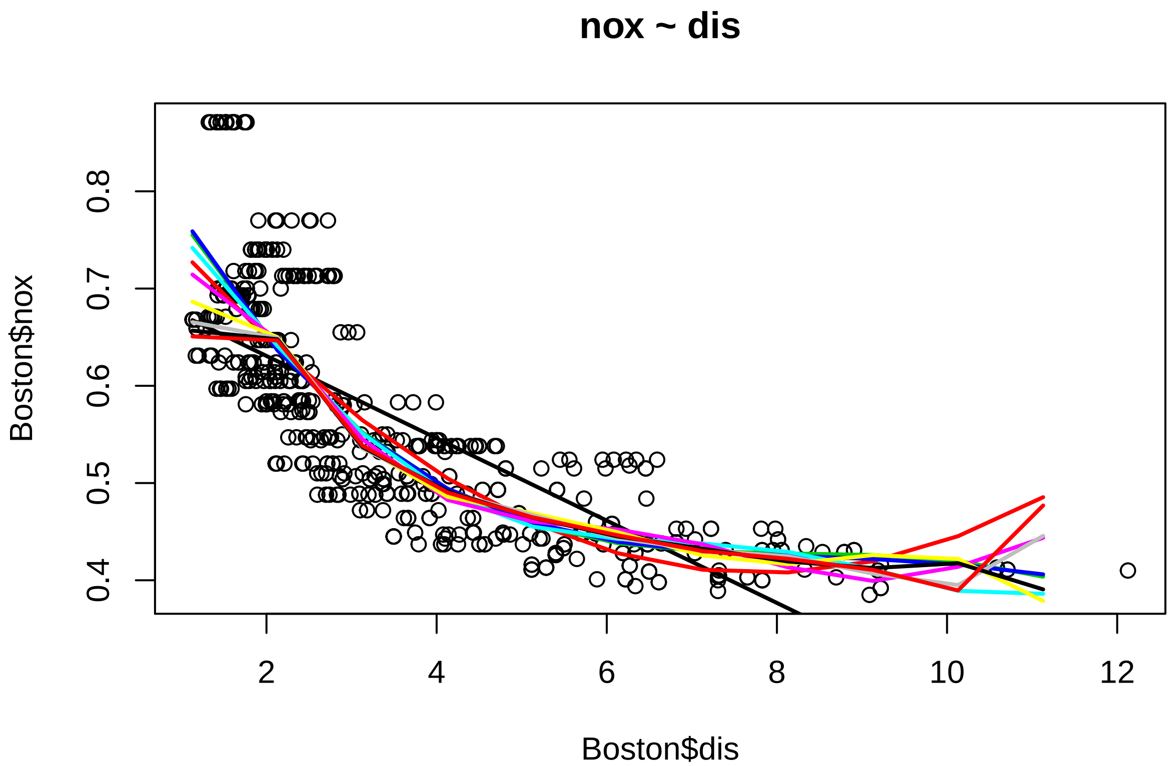
b)

From the RSS, it looks like the performance of the regression levels out with the inclusion of the cubic term. The quadratic formula doesn't perform markedly worse than the higher-order formulas, either. Formally, the RSS for polynomials of degrees 1 through 10 were:

2.768563, 2.035262, 1.934107, 1.932981, 1.915290, 1.878257, 1.849484, 1.835630, 1.833331, 1.832171

```
plot(Boston$dis, Boston$nox, xlim=dislims, main = "nox ~ dis")
RSS <- c()

for (x in 1:10) {
  mod <- glm(nox~poly(dis, x, raw = T), data = Boston)
  preds <- predict(mod, newdata=list(dis=dis.grid), se=TRUE)
  lines(dis.grid, preds$fit, lwd=2, col=x)
  RSS <- c(RSS, sum((mod$residuals)^2))
}
```




```
RSS
```

```
## [1] 2.768563 2.035262 1.934107 1.932981 1.915290 1.878257 1.849484
## [8] 1.835630 1.833331 1.832171
```

c)

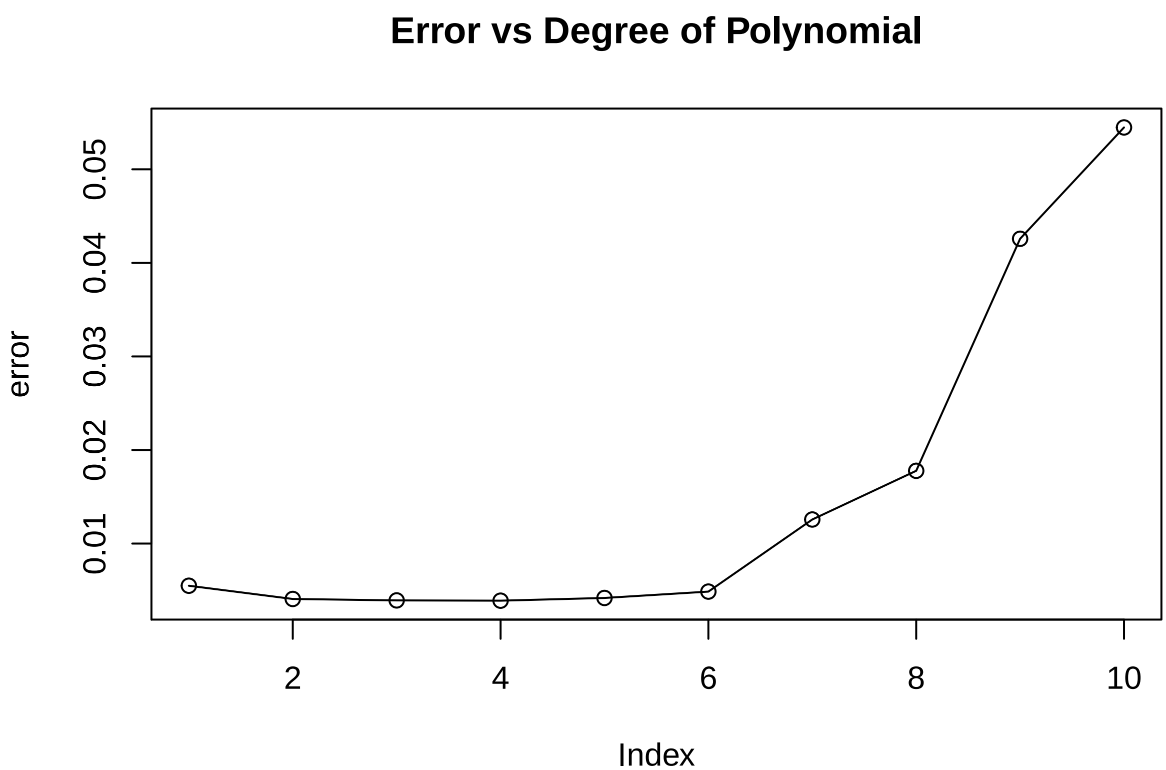
Based on the results of cross-validation, the optimal degree of the polynomial is 4. That is, the raw cross-validation estimate of the prediction error is lowest when the polynomial is of degree 4. However, polynomials of degree 2, 3, and 5 perform pretty well, too, so I'd go with a degree-2 polynomial.

```
library(boot)
error <- c()
for (x in 1:10) {
  error <- c(error, cv.glm(Boston, glm(nox~poly(dis, x), data=Boston), K=5)$delta[1])
}

which.min(error)
```

```
## [1] 4
```

```
plot(error, type = 'o', main = "Error vs Degree of Polynomial")
```




```
error
```

```
##  [1] 0.005495203 0.004078837 0.003926799 0.003898401 0.004188813
##  [6] 0.004868291 0.012572455 0.017778228 0.042576055 0.054474664
```

Problem 4 (7.9 #10ab)

a)

When letting it run without bound (no nvmax set), the function returns a model with 8 predictors:

PrivateYes, Accept, Enroll, Room.Board, Terminal, perc.alumni, Expend, and Grad.Rate

From the adjusted R^2, it looks like we could go down to 5 or 6 predictors without losing much explanatory power. But, because the function turned out 8 predictors, I'll stick with that number.

```
attach(College)
library(leaps)

train = sample(c(TRUE, FALSE), nrow(College), rep = TRUE)
test = (!train)

training <- College[train, ]
testing <- College[test, ]

regfit.fwd <- regsubsets(Outstate ~ ., data=training, method ="forward")

results <- summary(regfit.fwd)

results$adjr2
```

```
## [1] 0.4652375 0.6188901 0.6844875 0.7304646 0.7461381 0.7613125 0.7650013
## [8] 0.7688115
```

```r
#results
```

b)

I wasn't sure what to do about the degrees of freedom for each predictor, so I went with a uniform 3 for each. From the plots, it looks like Private, Accept, and Room.Board are positively correlated with Outstate, while Enroll is negatively correlated; it's a bit ambiguous what the relationship between predictor and response is for Terminal, perc.alumni, Expend, and Grad.Rate. I would guess that at least a few of the last group could be dropped without sacrificing much from the model.

```r
library(gam)
```

```
## Loading required package: splines
```

```
## Loaded gam 1.15
```

```r
gam.fit <- gam(Outstate ~ Private + s(Accept, 3) + s(Enroll, 3) + s(Room.Board, 3) + s(Terminal, 3) + s

par(mfrow = c(3,3))
plot(gam.fit, se = T, col = 'red')
```