512 Assignment #6

Brody Vogel

3/22/2018

Problem 1 (5.4 #8)

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

a)

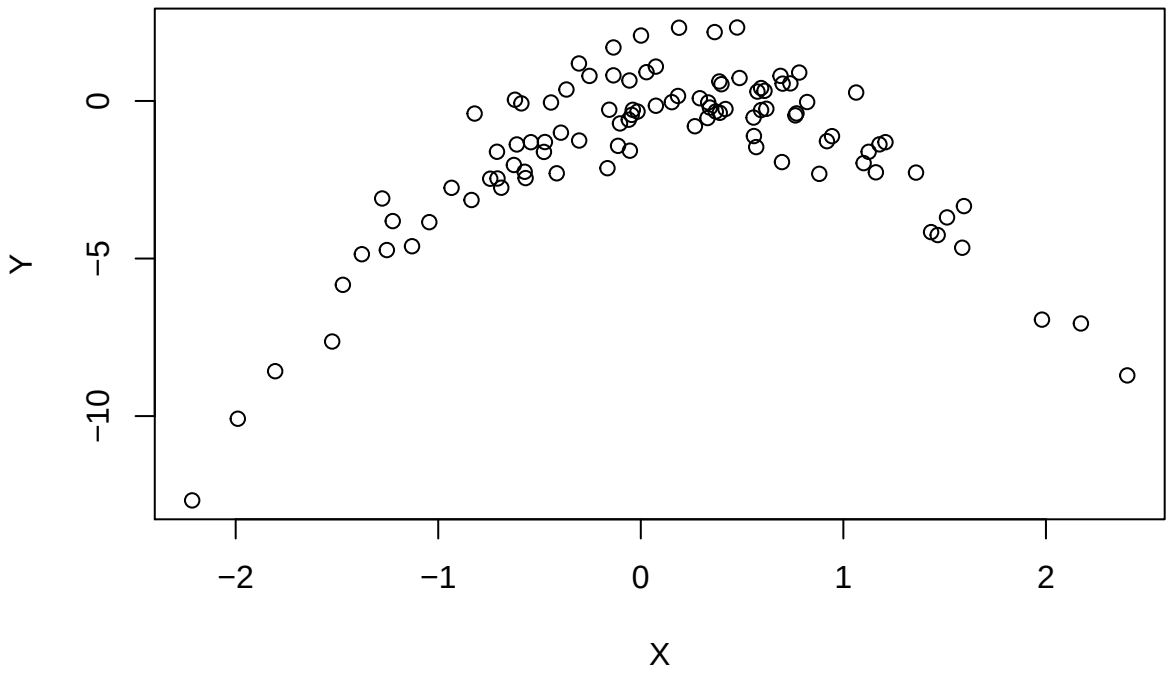
$n = 100$ random draws from the standard normal distribution

$p = 2 : X, -2X^2$

$Y = X - 2X^2 + \epsilon$

b)

```
plot(y~x, xlab = 'X', ylab = 'Y')
```



It's certainly not linear. It looks quadratic (because of the X^2 term), and appears to be limited by $x = (-3, 3)$ and $y = (-15, 5)$.

c)

```
library(boot)

set.seed(1234)

xy <- data.frame(x, y)

fit1 <- glm(y~x)
cv.glm(xy, fit1)$delta
```

```
## [1] 7.288162 7.284744
```

```
fit2 <- glm(y~poly(x, 2))
cv.glm(xy, fit2)$delta
```

```
## [1] 0.9374236 0.9371789
```

```
fit3 <- glm(y~poly(x, 3))
cv.glm(xy, fit3)$delta
```

```
## [1] 0.9566218 0.9562538
```

```
fit4 <- glm(y~poly(x, 4))
cv.glm(xy, fit4)$delta
```

```
## [1] 0.9539049 0.9534453
```

i. 7.29

ii. .94

iii. .96

iv. .95

It looks like the best, simplest fit comes from $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$.

d)

```
set.seed(1)

xy <- data.frame(x, y)

fit1 <- glm(y~x)
cv.glm(xy, fit1)$delta
```

```
## [1] 7.288162 7.284744
```

```
fit2 <- glm(y~poly(x, 2))
cv.glm(xy, fit2)$delta
```

```
## [1] 0.9374236 0.9371789
```

```
fit3 <- glm(y~poly(x, 3))
cv.glm(xy, fit3)$delta
```

```
## [1] 0.9566218 0.9562538
```

```
fit4 <- glm(y~poly(x, 4))
cv.glm(xy, fit4)$delta
```

```
## [1] 0.9539049 0.9534453
```

Yes, they're the same. LOOCV performs k-folds CV with k = n, so there won't be any variation in the results because the groupings of train and test folds are the same.

e)

The quadratic model ($Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$) had the smallest error, barely. I'm a little surprised that the two highest-dimension models didn't accidentally capture more of the error term. This makes sense that the quadratic model produces the best fit, though, because the true data was generated with a quadratic function.

f)

In each of the *summary()* calls, the p-values for the X and X^2 terms tests as significant, while the X^3 and X^4 terms do not. This agrees with both our intuition and the cross-validation results: LOOCV told us the best model was quadratic, and we know the data truly was generated according to a quadratic function. Thus, the coefficients for the X and X^2 terms test as significant because they provide the best fit for the data.

```
#summary(fit1)
#summary(fit2)
#summary(fit3)
summary(fit4)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0550  -0.6212  -0.1567   0.5952   2.2267
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09591 -16.162  < 2e-16 ***
## poly(x, 4)1   6.18883    0.95905   6.453 4.59e-09 ***
## poly(x, 4)2 -23.94830    0.95905 -24.971  < 2e-16 ***
## poly(x, 4)3   0.26411    0.95905   0.275    0.784
## poly(x, 4)4   1.25710    0.95905   1.311    0.193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9197797)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  87.379  on 95  degrees of freedom
## AIC: 282.3
##
## Number of Fisher Scoring iterations: 2
```

Problem 2 (6.8 #1)

a)

The best subset model with k predictors will have the smallest *training* RSS, because it tests every possible model, whereas forward and backward stepwise do not.

b)

Hard to tell - there's no guarantee any of the three models with k predictors will do a good job on test data. Best subset might pick the best model (the one with the smallest test RSS) because it tries every possible model on the training data. But, becasue of their constraints, forward or backward stepwise stepwise might choose different models that end up performing better on test data than does that chosen by best subset.

c)

i. TRUE

ii. TRUE

iii. FALSE

iv. FALSE

v. FALSE

Problem 3 (6.8 #2a,b)

a)

iii. The lasso, relative to least squares, is less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.

The lasso simplifies a set of p predictors down to a smaller subset, and so is less flexible than least squares and introduces some bias into the model. But, without all the extra predictors, the lasso decreases the variance of the model, too. So, when that decrease in variance outweighs the increase in bias, the lassoed model will improve prediction accuracy.

b)

iii. Ridge Regression, relative to least squares, is less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.

Ridge Regression is less flexible than least squares because it shrinks the coefficients in the model; thus, with this restriction, ridge introduces some bias into the model. But, this restriction also decreases the model's variance when applied to different training data. Thus, again, when that decrease in variance outweighs the increase in bias, the ridge model will improve prediction accuracy.

Problem 4 (6.8 9a-d)

a)

```
set.seed(runif(1, 1, 1000))
x <- rnorm(100)
e <- rnorm(100)
```

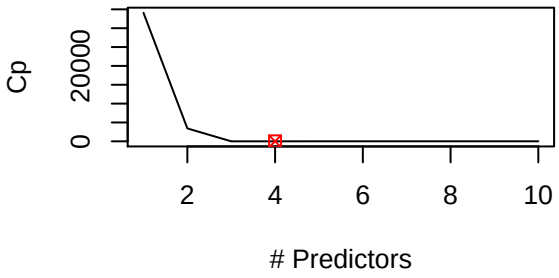
b)

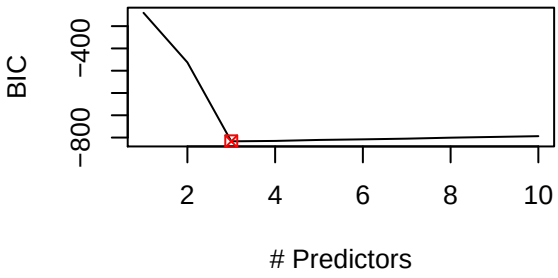
```
y = 5 + 10*x + 15*x^2 + 20*x^3 + e
```

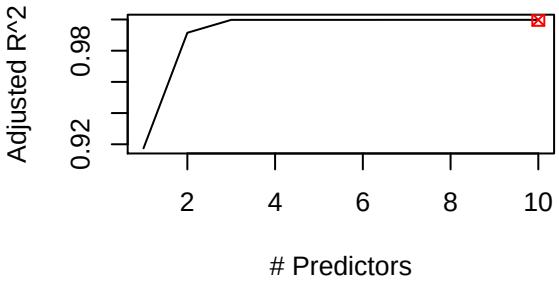
c)

```
library(leaps)

xy <- data.frame(y = y, x = x)

regfit <- regsubsets(y~poly(x, 10, raw = T), xy, nvmax = 10)

cp <- summary(regfit)$cp
bic <- summary(regfit)$bic
ar2 <- summary(regfit)$rsq

par(mfrow = c(2,2))

plot(cp, ylab = 'Cp', xlab = '# Predictors', type = 'l')
points(which.min(cp), cp[which.min(cp)], col = 'red', pch = 7)

plot(bic, ylab = 'BIC', xlab = '# Predictors', type = 'l')
points(which.min(bic), bic[which.min(bic)], col = 'red', pch = 7)

plot(ar2, ylab = 'Adjusted R^2', xlab = '# Predictors', type = 'l')
points(which.max(ar2), ar2[which.max(ar2)], col = 'red', pch = 7)

coef(regfit, id = 3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##              4.962871              9.950383             15.038577
## poly(x, 10, raw = T)3
##             20.036856
```







According to BIC, the best model is the one with three parameters (Cp chose the model with four and Adjusted R^2 that with ten because they had *slightly* lower/higher values than that with 3). When restricted to three parameters, those chosen in each case turn out to be those of the cubic model. This is what we'd expect, considering the true form of our generated data is cubic.

The raw coefficients for the three-predictor model came out accordingly:

$Y = 4.9 + 10.2X + 14.9X^2 + 19.9X^3$

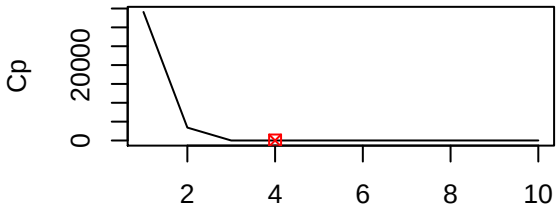
which are very close to the true parameters.

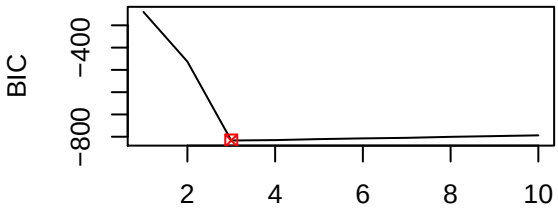
d)

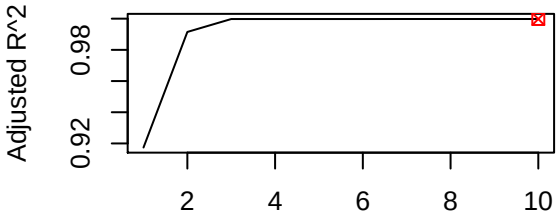
```
regfit1 <- regsubsets(y~poly(x, 10, raw = T), xy, nvmax = 10, method = 'forward')

cp <- summary(regfit1)$cp
bic <- summary(regfit1)$bic
ar2 <- summary(regfit1)$rsq

par(mfrow = c(2,2))

plot(cp, ylab = 'Cp', xlab = '# Predictors', type = 'l')
points(which.min(cp), cp[which.min(cp)], col = 'red', pch = 7)

plot(bic, ylab = 'BIC', xlab = '# Predictors', type = 'l')
points(which.min(bic), bic[which.min(bic)], col = 'red', pch = 7)

plot(ar2, ylab = 'Adjusted R^2', xlab = '# Predictors', type = 'l')
points(which.max(ar2), ar2[which.max(ar2)], col = 'red', pch = 7)

coef(regfit1, id = 3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##              4.962871              9.950383             15.038577
## poly(x, 10, raw = T)3
##             20.036856
```

```
regfit2 <- regsubsets(y~poly(x, 10, raw = T), xy, nvmax = 10, method = 'backward')

cp <- summary(regfit2)$cp
bic <- summary(regfit2)$bic
ar2 <- summary(regfit2)$rsq

par(mfrow = c(2,2))
```

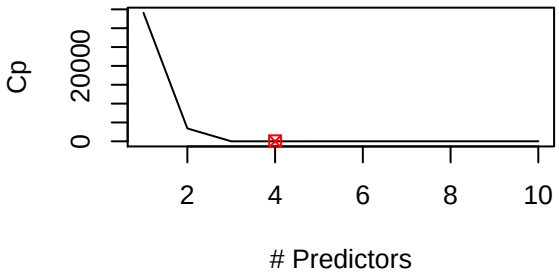


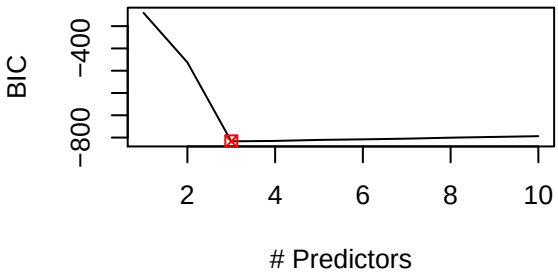


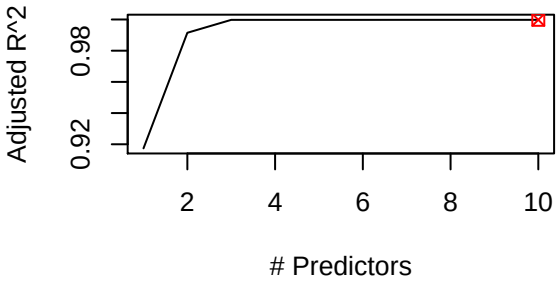



```
plot(cp, ylab = 'Cp', xlab = '# Predictors', type = 'l')
points(which.min(cp), cp[which.min(cp)], col = 'red', pch = 7)

plot(bic, ylab = 'BIC', xlab = '# Predictors', type = 'l')
points(which.min(bic), bic[which.min(bic)], col = 'red', pch = 7)

plot(ar2, ylab = 'Adjusted R^2', xlab = '# Predictors', type = 'l')
points(which.max(ar2), ar2[which.max(ar2)], col = 'red', pch = 7)

coef(regfit2, id = 3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##              4.962871              9.950383             15.038577
## poly(x, 10, raw = T)3
##             20.036856
```







My results were *exactly* the same as those from best subset. Again, according to BIC, the best model was the one with three parameters, while Cp chose the model with four and Adjusted R^2 that with ten because they had *slightly* lower/higher values than that with 3. When restricted to three parameters, those chosen in each case again turned out to be those of the cubic model.

The raw coefficients, in both casesl for the three-predictor model came out identically to best subset:

$Y = 4.9 + 10.2X + 14.9X^2 + 19.9X^3$

which are still very close to the true parameters.