Stat 432 Homework 5

Assigned: Feb 23, 2019; Due: 11:59pm Mar 2, 2019

Question 1 (linear regression)

[2 points] Sorry for the confusion, I meant to say page 21. You get 2 points for free for this problem. The irreducible error is $\text{E}[\|\mathbf{y}^* - \mathbf{X}\boldsymbol{\beta}\|^2] = n\sigma^2$, which comes from the model assumption and cannot be reduced further; the bias (of the estimator $\mathbf{X}\widehat{\boldsymbol{\beta}}$) is $\mathbf{X}\boldsymbol{\beta} - \text{E}[\mathbf{X}\widehat{\boldsymbol{\beta}}] = 0$, and the variance (of the estimator) is $\text{E}[\|\mathbf{X}\widehat{\boldsymbol{\beta}} - \text{E}[\mathbf{X}\boldsymbol{\beta}\|^2] = p\sigma^2$.

[2 points] This is actually written on page 15 of lecture note `penalized`. Note that in the ridge regression, we introduce some bias $\text{Bias}(\widehat{\beta}_j^{\text{ridge}}) = \frac{-\lambda}{1+\lambda}\beta_j$ to trade off for the reduced variance $\text{Var}(\widehat{\beta}_j^{\text{ridge}}) = \frac{1}{(1+\lambda)^2}\text{Var}(\widehat{\beta}_j^{\text{ols}})$. Therefore, the overall the prediction error will be smaller than that of OLS:

$$\text{Bias}^2(\widehat{\beta}_j^{\text{ridge}}) + \text{Var}(\widehat{\beta}_j^{\text{ridge}}) = \frac{\lambda^2}{(1+\lambda)^2}\beta_j^2 + \frac{1}{(1+\lambda)^2}\text{Var}(\widehat{\beta}_j^{\text{ols}}) < \text{Var}(\widehat{\beta}_j^{\text{ols}})$$

which is always satisfied for some λ because the quadratic curve of λ passes the origin. 1 point for specifying bias and variance in the ridge regression setting; 1 point for discussing the bias-variance trade-off.

Question 2 (model selection criteria)

The Boston Housing data is a classical dataset that models the median house values `medv` of different areas of Boston. Because a lot of variables exhibit an asymmetry, we will use some transformations.

```
data(Boston, package="MASS")
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98 24.0
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14 21.6
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03 34.7
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94 33.4
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33 36.2
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21 28.7
```

```
useLog = c(1,3,5,6,8,9,10,14)
Boston[,useLog] = log(Boston[,useLog])
Boston[,2] = Boston[,2] / 10
Boston[,7] = Boston[,7]^2.5 / 10^4
Boston[,11] = exp(0.4 * Boston[,11])/1000
Boston[,12] = Boston[,12] / 100
Boston[,13] = sqrt(Boston[,13])
```

part a)

[1 point]

We fit the following the linear model and summarize the outputs

```
lmfit = lm(medv~., Boston)
summary(lmfit)$coefficients
```

```
##                 Estimate  Std. Error    t value     Pr(>|t|)
## (Intercept)  4.176874035 0.379016727  11.0202895 2.158941e-25
## crim        -0.014606367 0.011650102  -1.2537545 2.105267e-01
```

```
## zn           0.001391943 0.005638624   0.2468585 8.051207e-01
## indus       -0.012709368 0.022311989  -0.5696206 5.691950e-01
## chas         0.109980144 0.036633710   3.0021569 2.817052e-03
## nox         -0.283111884 0.105340487  -2.6875885 7.440816e-03
## rm           0.421107840 0.110174650   3.8221845 1.492369e-04
## age          0.006403368 0.004863019   1.3167476 1.885362e-01
## dis         -0.183154286 0.036803637  -4.9765268 8.966317e-07
## rad          0.068361590 0.022473189   3.0419176 2.476295e-03
## tax         -0.201832385 0.048432167  -4.1673209 3.641214e-05
## ptratio     -0.040017441 0.008091477  -4.9456285 1.043293e-06
## black        0.044471934 0.011455971   3.8819872 1.177364e-04
## lstat       -0.262615094 0.016091240 -16.3203760 3.598708e-48
```

part b)

[3 points, 1 point for each subproblem.] * We can use `leaps` package to calculate the Mallow's Cp statistics.

```r
# load leaps and calcualte the best subset fit
library(leaps)
RSSleaps=regsubsets(as.matrix(Boston[,-14]),Boston[,14],nvmax=13)
sumleaps=summary(RSSleaps,matrix=T)
msize=apply(sumleaps$which,1,sum)
n=dim(Boston)[1]
p=dim(Boston)[2]
# calcualte the Mallow's Cp
Cp=sumleaps$rss/(summary(lmfit)$sigma^2) + 2*msize - n
# Compare the results with the return by regsubsets function
cbind(msize, Cp, sumleaps$cp)
```

```
##    msize         Cp
## 1      2 166.568016 166.568016
## 2      3 110.304269 110.304269
## 3      4  90.220657  90.220657
## 4      5  61.263595  61.263595
## 5      6  45.547551  45.547551
## 6      7  27.940433  27.940433
## 7      8  21.385925  21.385925
## 8      9  14.968721  14.968721
## 9     10   9.737754   9.737754
## 10    11  10.346297  10.346297
## 11    12  10.584968  10.584968
## 12    13  12.060939  12.060939
## 13    14  14.000000  14.000000
```

- Now we calcualt AIC and BIC based on their definitions.

```r
# calcualt AIC and BIC based on their definitions
AIC = n*log(sumleaps$rss/n) + 2*msize
as.matrix(AIC)
```

```
##         [,1]
## 1  -1479.977
## 2  -1524.114
## 3  -1540.763
## 4  -1566.119
## 5  -1580.467
## 6  -1597.197
```

```
## 7  -1603.591
## 8  -1609.989
## 9  -1615.317
## 10 -1614.739
## 11 -1614.545
## 12 -1613.084
## 13 -1611.146
```

```
BIC = n*log(sumleaps$rss/n) + msize*log(n)
# compare it with the return by regsubsets function
cbind(BIC, sumleaps$bic+n*log(sum((Boston[,14] - mean(Boston[,14]))^2/n)))
```

```
##          BIC
## 1  -1471.524 -1471.524
## 2  -1511.434 -1511.434
## 3  -1523.857 -1523.857
## 4  -1544.986 -1544.986
## 5  -1555.107 -1555.107
## 6  -1567.611 -1567.611
## 7  -1569.779 -1569.779
## 8  -1571.950 -1571.950
## 9  -1573.051 -1573.051
## 10 -1568.247 -1568.247
## 11 -1563.827 -1563.827
## 12 -1558.139 -1558.139
## 13 -1551.975 -1551.975
```

- We select the best models using `step` based on AIC and BIC respectively. To use Mallow's Cp returned by `regsubsets` function, we select the model that has C_p closest to p, namely the 9th model from the first subproblem.

```
# backward with AIC
stepaic<-step(lmfit, direction="backward", trace=0)
paste(variable.names(stepaic),collapse = ' + ')
```

```
## [1] "(Intercept) + chas + nox + rm + dis + rad + tax + ptratio + black + lstat"
```

```
# backward with BIC
stepbic<-step(lmfit, direction="backward", k=log(n), trace=0)
paste(variable.names(stepbic),collapse = ' + ')
```

```
## [1] "(Intercept) + chas + nox + rm + dis + rad + tax + ptratio + black + lstat"
```

```
# paste(colnames(sumleaps$which)[sumleaps$which[which.min(sumleaps$bic),]],collapse=' + ')
# best result based on Mallow's Cp
paste(colnames(sumleaps$which)[sumleaps$which[9,]],collapse=' + ')
```

```
## [1] "(Intercept) + chas + nox + rm + dis + rad + tax + ptratio + black + lstat"
```

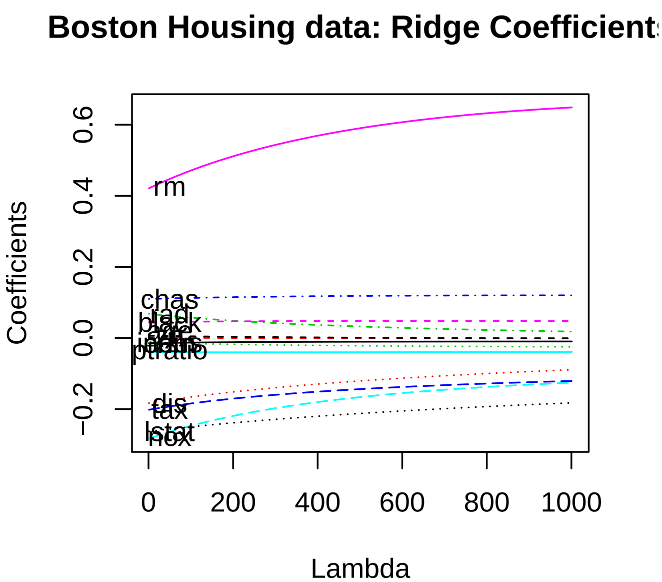
In this example, all the three selection criteria give the same conclusion (0.5 points off for missing conclusion.).

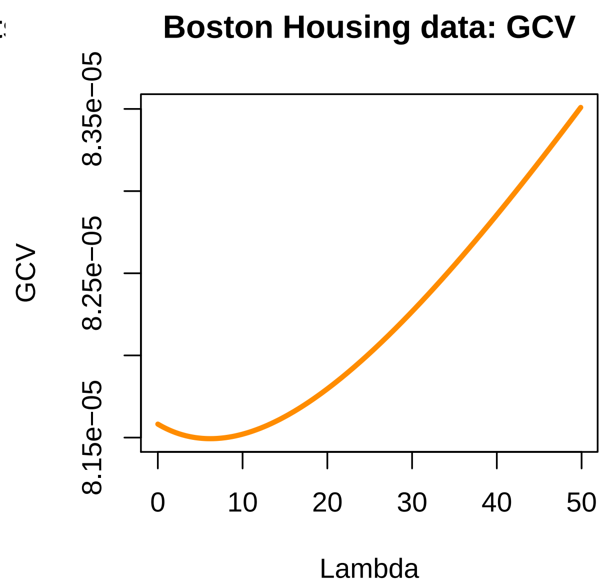
Question 3 (ridge regression)

[2 points] Now use `lm.ridge` function to fit the same dataset. We choose λ in a sequence between 0 and 100 and use geneneralized cross-validiation (GCV) to choose the best value (1 point). We plot shrinking coefficients and the GCV values as below.

```
library(MASS)
# ridge regression
```

```r
ridge.fit = lm.ridge(medv~., Boston, lambda=seq(0,100,by=0.1))
# plot shrinking coefficients and the GCV values
par(mfrow=c(1,2))
matplot(coef(ridge.fit)[, -1], type = "l", xlab = "Lambda", ylab = "Coefficients")
text(rep(50, p-1), coef(ridge.fit)[1,-1], colnames(Boston)[1:p-1])
title("Boston Housing data: Ridge Coefficients")
plot(ridge.fit$lambda[1:500], ridge.fit$GCV[1:500], type = "l", col = "darkorange",
     ylab = "GCV", xlab = "Lambda", lwd = 3)
title("Boston Housing data: GCV")
```





Now we use GCV to choose the best λ and report the penalized regression coefficients (1 point).

```r
# use GCV to select the best lambda
ridge.fit$lambda[which.min(ridge.fit$GCV)]
```

```
## [1] 6.2
```

```r
# ridge regression coefficients
round(coef(ridge.fit)[which.min(ridge.fit$GCV), ], 4)
```

```
## crim zn indus chas nox rm age dis rad tax ptratio black lstat
## 4.0428 -0.0134 0.0009 -0.0147 0.1118 -0.2594 0.4538 0.0053 -0.1719 0.0610 -0.1902
-0.0401 0.0454 -0.2544
```