Stat 432 Homework 6

Assigned: Mar 3, 2019; Due: Mar 8, 2019

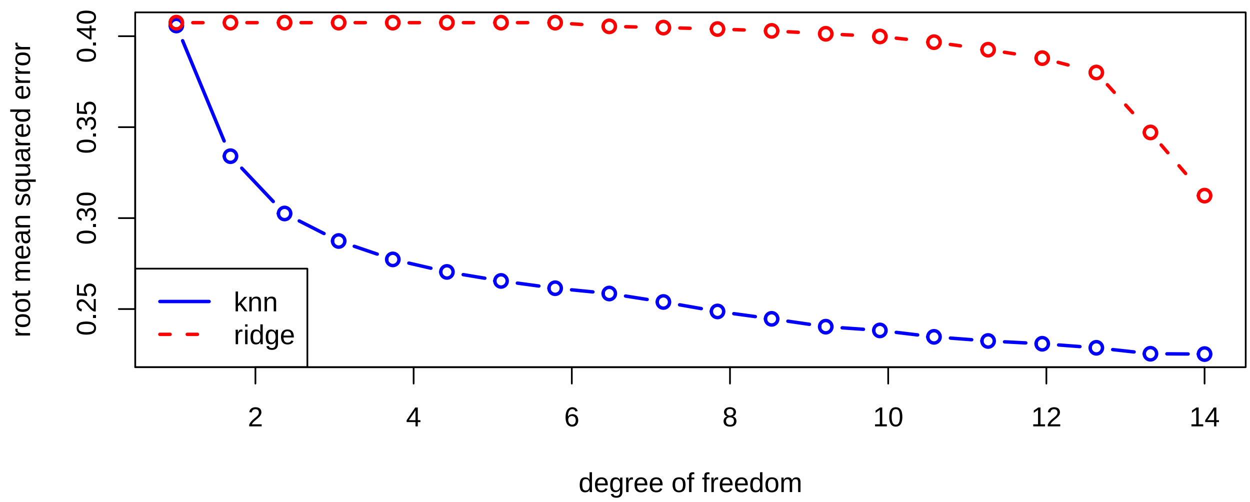
Question 1 (ridge regression) [5 points]

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

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])
```

We need to compare kNN with ridge regression on a range of degree of freedom (df) from 1 to 14. We choose k and λ such that their df's are matched (1 point). Then we use `caret` package to obtain the validation errors on testing subset in 10-fold cross validation (2 points, 1 point for each method; can use for loop). We plot these errors in the same figure as a function of df (1 point).

```
# preprocess data
n=dim(Boston)[1]; p=dim(Boston)[2]-1
Boston[,-c(4,p+1)]=scale(Boston[,-c(4,p+1)])
sv=svd(Boston[,-(p+1)])$d
# set df
ndf=20 # could be some other number as df does not have to be integer
dfs=seq(1,(p+1),length.out=ndf)
# choose k
ks=ceiling(n/dfs)
# choose lambda
lambdas=rep(NA,ndf)
df=function(lambda,d=sv)sum(d^2/(d^2+lambda))
for(i in 1:ndf){
  lambdas[i]=optim(0,function(lambda)(n/ks[i]-(df(lambda)+1))^2,method='Brent',lower=0,upper=1e8)$par}
# fit knn in CV
library(caret)
knnFit <- train(medv~., data=Boston, method = "knn",
                tuneGrid = data.frame("k" = pmin(floor(n*.9)-1,ks)),
                trControl = trainControl(method = "cv", number = 10))
library(glmnet)
# fit ridge regression in CV
ridgeFit <-train(Boston[,-(p+1)],Boston[,(p+1)], method = "glmnet",
                 tuneGrid = data.frame("lambda" =lambdas, "alpha"=0),
                 trControl = trainControl(method = "cv", number = 10))
# plot
plot(dfs,rev(knnFit$results$RMSE),type='b',col='blue',lwd=2,xlab='degree of freedom',ylab='root mean sq
points(dfs,rev(ridgeFit$results$RMSE),type='b',col='red',lwd=2,lty=2)
legend('bottomleft',legend=c('knn','ridge'),col=c('blue','red'),lwd=2,lty=1:2)
```



Note, with increasing degree of freedom (decreasing k), kNN has decreasing bias/increasing variance and tends to move away from under-fitting. Thus RMSE on the validation set decreases. On the other hand, ridge regression has decreasing penalty λ with increasing df, therefore it tends to have decreasing bias/increasing variance. RMSE on the validation set decreases to move away from under-fitting. In general kNN has lower RMSE than ridge regression on the validation set though they start with comparable prediction error when df$=1$. (1 point for comments; you lose it if there is no comment at all.)

Question 2 (Lasso regression) [5 points]

Now we compare Lasso with best subset selection. We first fit Lasso using `cv.glment` function in the `glmnet` package to tune the penalty parameter λ (2 points: 1 point for fitting, 1 point for reporting the best model.). Then we run `regsubsets` in `leaps` package to select the best model using AIC penalty (2 points: 1 point for fitting, 1 point for reporting the best model.). Then we campare their bias-variance trade-off.

```r
set.seed(2019)
# fit Lasso
lassoFit <-  cv.glmnet(data.matrix(Boston[,-(p+1)]),Boston[,(p+1)],nfolds=10) # default alpha=1 Lasso
# report the best model by lasso
coef(lassoFit,s='lambda.min')
```

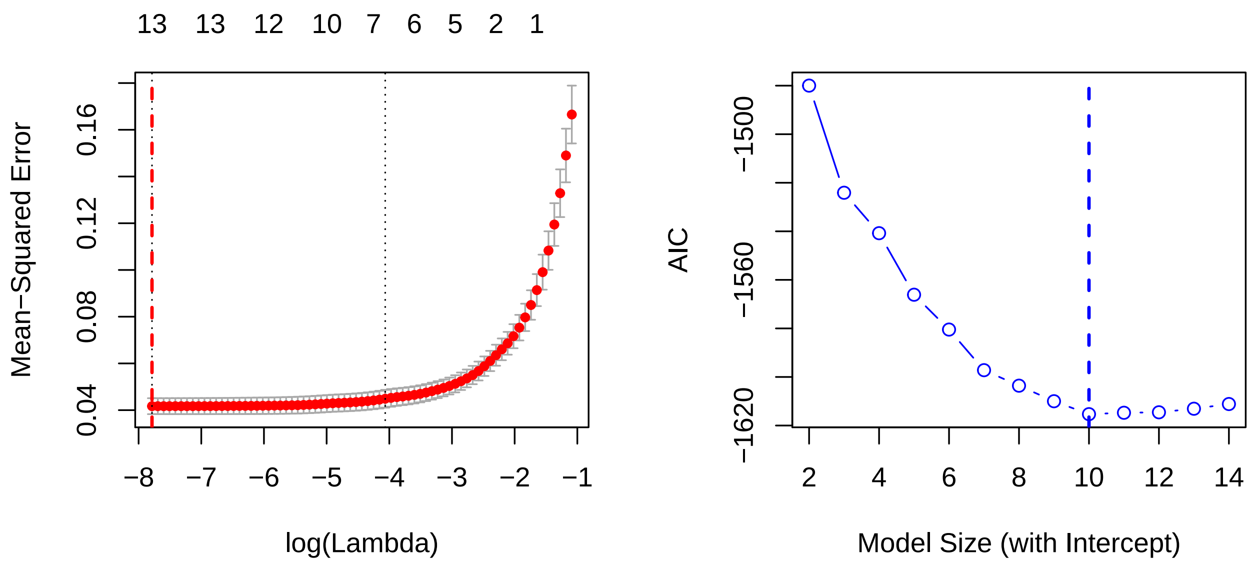
```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  3.026952812
## crim        -0.027905834
## zn           0.002345714
## indus       -0.009809494
## chas         0.109296906
## nox         -0.054375816
## rm           0.047799450
## age          0.020447990
## dis         -0.095718598
## rad          0.055015323
## tax         -0.077821427
## ptratio     -0.054454135
## black        0.040596485
## lstat       -0.258645964
```

```r
# best subset selection
library(leaps)
bestsubset <- regsubsets(as.matrix(Boston[,-(p+1)]),Boston[,(p+1)],nvmax=13)
```

```r
sumysub = summary(bestsubset)
# report the best model by AIC
msize=apply(sumysub$which,1,sum)
AIC=n*log(sumysub$rss/n) + 2*msize
as.matrix(coef(bestsubset,which.min(AIC)))
```

```
##                    [,1]
## (Intercept)  3.02699723
## chas         0.10865477
## nox         -0.06156124
## rm           0.05243412
## dis         -0.10010582
## rad          0.04302758
## tax         -0.08307596
## ptratio     -0.05598859
## black        0.04394884
## lstat       -0.25545191
```

```r
# plot bias-variance trade-off
par(mfrow = c(1, 2))
plot(lassoFit)
abline(v=log(lassoFit$lambda.min),col='red',lwd=2,lty=2)
plot(msize,AIC,type='b',col='blue',xlab="Model Size (with Intercept)", ylab="AIC")
abline(v=1+which.min(AIC),col='blue',lwd=2,lty=2)
```



Lasso trades off the bias and variance through the ℓ_1 penalty: the error on validation set increases with larger penalty (λ), indicating larger bias and smaller variance (moving towards under-fitting). While the best subset trades off the bias and variance through AIC with the numnber of parameters p which can be viewed as ℓ_0 penalty: with larger number of parameters, the model tends to be lower in bias and higher in variance (moving from under-fitting towards over-fitting). (1 point: for comparing on their trade-offs in bias and variance; you lose 0.5 if you do not include the **left** panel of the figure; and lose 0.5 if there is no comment at all.)

Extra-Credit Question [4 points]

Denote the probability of getting head as θ. First, by calculation of the first statement, you can see that the one-side hypothesis of the fair coin is as follows (1 point: you lose 0.5 if the alternative is wrong.)

$$H_0 : \theta = \frac{1}{2}, \quad H_a : \theta < \frac{1}{2}$$

Denote Y as the number of heads observed. For the first statement, we use the Binomial model $Y \sim \text{binom}(n, \theta)$. The p-value is calculated (1.5 points: 1 point for correct formula, 0.5 point for verifying the p-value.)

$$p_1 = P[Y \leq 3 | H_0] = \text{pbinom}(3, 12, 0.5) = 0.072998$$

For the second statement, we use the negative binomial distribution. Note, there are different definitions of negative binomial depending on the focus. If you use `R` built-in function `pnbinom`, you need to pay attent to that $Y \sim \text{negbinom}(n, \theta)$ denotes the number of **failures** needed before n **successes**. Then the corresponding p-value is calculated (1.5 points: 1 point for correct formula, 0.5 point for verifying the p-value.)

$$p_2 = P[Y \geq 9 | H_0] = \text{pnbinom}(8, 3, 0.5, \text{lower.tail} = F) = 0.0327148$$