415 hw6 solution

April Cho

3/6/2018

Problem 1

(a)

(3): Number of variables included in the model will steadily increase as we increase s from 0. When s is 0, β_js all shrink to 0, which means there is no variable included in the model. As we increase s from 0, we penalize coefficients less and thus some of the β_j will start to have positive absolute values. Hence the number of variables in the model steadily increase.

(b)

(4): the training RSS will steadily decrease as we increase s from 0. Since the optimal solution of the problem with smaller s is always a feasible solution to the problem with larger s, the optimal solution of the latter couldn't be worse than the solution of the former.

(c)

(2): in general the test RSS will decrease initially, and then eventually start increasing in a U shape. Initially, when s equal 0, $\widehat{\beta}$ can only be 0. As s increases, the bias of estimating β decreases, so the test RSS decreases. As s continuously increases, the variance also increases. As a result of the bias variance trade-off, the test RSS will eventually increase.

(d)

(3): the variance will steadily increase. As s increases, the model tends to be more flexible, so the variance will increase.

(e)

(4): the bias will steady decrease. As s increases, the model tends to be more flexible, so the bias will decrease. When s is infinity, $\widehat{\beta}$ is the ordinary least square estimator.

Problem 2

(a)

We first load the ISLR package to access the dataset. We split the data into training and test dataset by sampling 30% of data as test set. (Sampling 70% of data as training set will have different results but is also acceptable.)

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.2.5
```

```
set.seed(23456)
test_id <- sample(1:nrow(College), size = trunc(0.3 * nrow(College)))
test <- College[test_id,]
train <- College[-test_id,]
```

(b)

We first fit the ordinary linear regression using the training dataset. We get the test MSE using the fitted model which is 1300431. The training MSE is 993164.

```
lm.fit <- lm(Apps~., data = train)
lm.train.err <- mean((lm.fit$residuals)^2)
lm.test.err <- mean((test$Apps - predict(lm.fit, test))^2)
lm.train.err; lm.test.err
```

```
## [1] 993164.6

## [1] 1300431
```

(c)

We perform forward and backward stepwise selection using p-value on the training dataset. The variables recommended by forward and backward selection is shown above. The training and test MSE from the model chosen by forward selection is 1043037 and 1334782. The training and test MSE from the final model chosen by backward selection is 1021693 and 1355206.

```
#install.packages("SignifReg")
library(SignifReg)
```

```
## Warning: package 'SignifReg' was built under R version 3.2.5
```

```
fw.fit <- SignifReg(Apps ~ ., train, alpha = 0.05, direction = "forward", criterion = "p-value")
fw.fit
```

```
##
## Call:
## lm(formula = reg, data = data)
##
## Coefficients:
## (Intercept)       Accept    Top10perc       Enroll   PrivateYes
##   532.67828      1.66367     45.25051     -0.74356   -785.61216
##      Expend          PhD    Top25perc
##     0.04746    -10.54103    -11.86217
```

```
train.err.fw <- mean((fw.fit$residuals)^2)
test.err.fw <- mean((test$Apps - predict(fw.fit, test))^2)

bw.fit <- SignifReg(Apps ~ ., train, alpha = 0.05, direction = "backward", criterion = "p-value")
bw.fit
```

```
##
## Call:
## lm(formula = reg, data = data)
##
## Coefficients:
## (Intercept)   PrivateYes       Accept       Enroll    Top10perc
```

```
##    182.17566   -403.41491      1.69288     -0.83323     45.82197
##    Top25perc     Outstate       Expend
##    -12.12395     -0.08479      0.06782
```

```
train.err.bw <- mean((bw.fit$residuals)^2)
test.err.bw <- mean((test$Apps - predict(bw.fit, test))^2)

train.err.fw; test.err.fw;
```

```
## [1] 1043037

## [1] 1334782
```

```
train.err.bw; test.err.bw;
```

```
## [1] 1021693

## [1] 1355206
```

(d)

Now we perform best subset selection using AIC and BIC. The variables recommended by AIC and BIC criterion are shown below.

```
library(leaps)
regfit.full=regsubsets(Apps~., train, nvmax=17)
regsum <- summary(regfit.full)
which.min(regsum$cp) #Model size chosen by AIC
```

```
## [1] 10
```

```
coef(regfit.full,10) #Variables chosen by AIC
```

```
##    (Intercept)    PrivateYes        Accept        Enroll     Top10perc
##    92.77034574 -525.34275680    1.67339267   -0.78553299   46.44504933
##      Top25perc      Outstate    Room.Board           PhD        Expend
##   -11.82100842   -0.10008892    0.11938762   -7.40829931    0.06759232
##      Grad.Rate
##     5.00261295
```

```
which.min(regsum$bic) #Model size chosen by BIC
```

```
## [1] 6
```

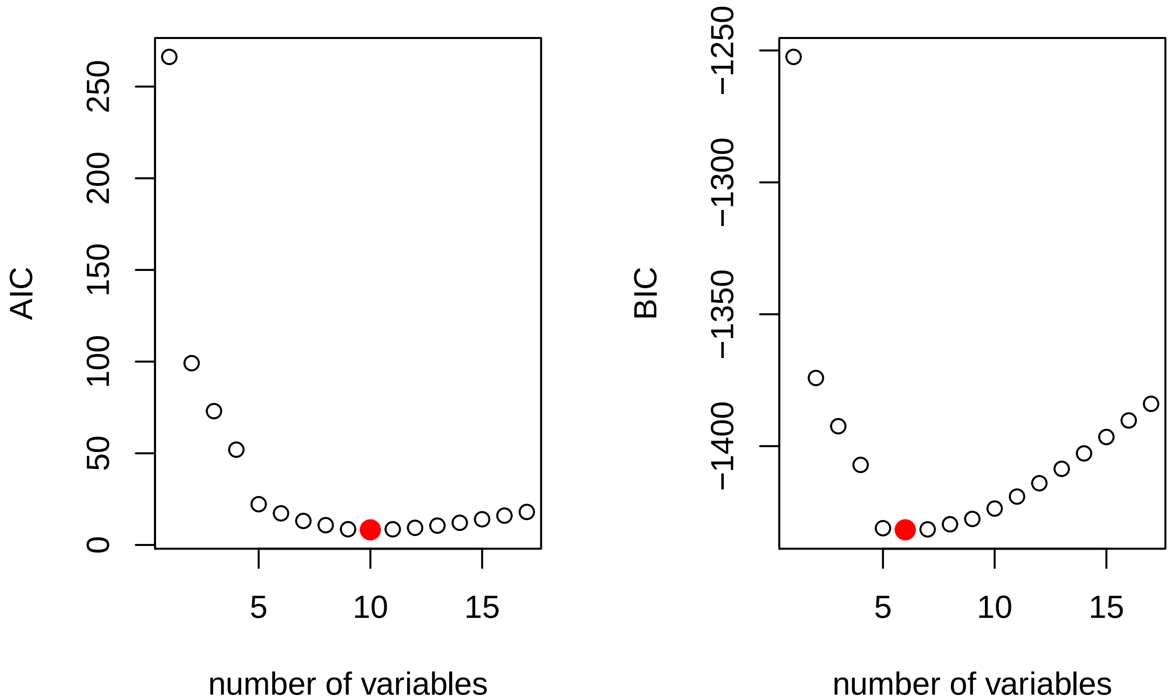
```
coef(regfit.full,6) #Variables chosen by BIC
```

```
##    (Intercept)    PrivateYes        Accept        Enroll     Top10perc
## -163.96646146 -386.22739630    1.68324007   -0.82769819   32.90344332
##       Outstate        Expend
##    -0.08889235    0.07474331
```

The plots show the AIC and BIC values for each model size. The red points are the final models chosen by each criterion. As shown in lab, we use the model.matrix() function to construct the matrix X and use it to caculate errors. The training and test error for the final model chosen by AIC are 1001215 and 1282321.The training and test error for the model chosen by BIC are 1033273 and 1380054.

```
par(mfrow=c(1,2))
plot(regsum$cp, xlab="number of variables", ylab="AIC")
points(10, regsum$cp[10], col="red", cex=2, pch=20)
```

```
plot(regsum$bic, xlab="number of variables", ylab="BIC")
points(6, regsum$bic[6], col="red", cex=2, pch=20)
```




```
train.mat = model.matrix(Apps ~., data = train)
test.mat = model.matrix(Apps ~., data = test)
coefi.aic = coef(regfit.full, id=10)
train.err.aic = mean((train$Apps - train.mat[,names(coefi.aic)]%*%coefi.aic)^2)
test.err.aic = mean((test$Apps - test.mat[,names(coefi.aic)]%*%coefi.aic)^2)

coefi.bic = coef(regfit.full, id=6)
train.err.bic = mean((train$Apps - train.mat[,names(coefi.bic)]%*%coefi.bic)^2)
test.err.bic = mean((test$Apps - test.mat[,names(coefi.bic)]%*%coefi.bic)^2)

train.err.aic; test.err.aic
```

```
## [1] 1001215
```

```
## [1] 1282321
```

```
train.err.bic; test.err.bic
```

```
## [1] 1033273
```

```
## [1] 1380054
```

(e)

We fit the ridge regression model using the glmnet() function and choose λ by crossvalidation using cv.glmnet(). Finally, we refit our ridge regression using the selected best λ. The training MSE is 1385774. The test MSE is 1223317.

```
x=model.matrix(Apps~.,College)[,-1]
y=College$Apps
```

```r
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.2.4

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-5
```

```r
grid=10^seq(10,-2,length=100)
ridge.mod=glmnet(x[-test_id,],y[-test_id],alpha=0,lambda=grid,thresh=1e-12)
cv.out=cv.glmnet(x[-test_id,],y[-test_id],alpha=0)
bestlam=cv.out$lambda.min

ridge.pred.train = predict(ridge.mod,s=bestlam,newx=x[-test_id,])
rid.train.err = mean((y[-test_id]-ridge.pred.train)^2)
ridge.pred.test = predict(ridge.mod,s=bestlam,newx=x[test_id,])
rid.test.err = mean((y[test_id]-ridge.pred.test)^2)

rid.train.err; rid.test.err;
```

```
## [1] 1385774

## [1] 1223317
```

(f)

Using the glmnet() function with alpha=1, we fit the lasso model. Again the best λ is chosen by cross-validation. The variables chosen in the model are shown above. The trainint MSE is 993997 and test MSE is 1293278.

```r
lasso.mod=glmnet(x[-test_id,],y[-test_id],alpha=1,lambda=grid)
cv.out=cv.glmnet(x[-test_id,],y[-test_id],alpha=1)
bestlam=cv.out$lambda.min
lasso.coef=predict(lasso.mod,type="coefficients",s=bestlam)
lasso.coef[1:18,]
```

```
##   (Intercept)    PrivateYes        Accept        Enroll     Top10perc
## -2.179788e+02 -4.947710e+02  1.658853e+00 -7.513941e-01  4.383458e+01
##     Top25perc   F.Undergrad   P.Undergrad      Outstate    Room.Board
## -9.640729e+00 -5.208176e-03  2.716643e-02 -8.952296e-02  1.045607e-01
##         Books      Personal           PhD      Terminal     S.F.Ratio
##  2.914315e-01  7.488261e-02 -4.355142e+00 -3.862260e+00 -2.663637e-02
##   perc.alumni        Expend     Grad.Rate
## -6.691672e-01  6.467427e-02  5.986592e+00
```

```r
lasso.pred.train = predict(lasso.mod,s=bestlam,newx=x[-test_id,])
lasso.train.err = mean((lasso.pred.train-y[-test_id])^2)
lasso.pred.test = predict(lasso.mod,s=bestlam,newx=x[test_id,])
lasso.test.err = mean((lasso.pred.test-y[test_id])^2)

lasso.train.err; lasso.test.err;
```

```
## [1] 993997

## [1] 1293278
```

(g)

As a benchmark, we also compute the MSE when not using any of the prediction variables but simply using the mean of the training data. The test MSE is 13524121, which is about ten times of the test MSE of the ordinary least square regression, and it implies that these models we have built can explain about 90% of the variance in the testing data.

```
mean((mean(College$Apps[-test_id]) - College$Apps[test_id])^2)
```

```
## [1] 13524121
```

The test error is smallest for the ridge regression, followed by AIC method and Lasso. This suggests that ridge regression model performs best on our test data and we choose ridge regression.