# STATS415hw6

*Yunguo Cai*

*3/2/2018*

1.

(a) Best option: (3)

As we increase s from 0, the number of variables included in the model will steadily increase because more $\beta s$ are incorporated in the model.

(b) Best option: (4)

As we increase s from 0, the training RSS will steadily decreases because the model becomes more flexible with the increasing s and thus $\beta_j$ is constrained and will be more and more close to the least squares estimate.

(c) Best option: (2)

As we increase s from 0, the test RSS will deacrease intially, and then eventually start increasing because $\beta_j$ is firstly constrained close to 0 for overfitting, resulting in decrease and coefficients are then removed from the model with the increasing of s, resulting in increase.

(d) Best option: (3)

As we increase s from 0, the variance of $\hat{\beta}$ will steadily increase because more $\beta s$ are incorporated in the model, which increases the variance.

(e) Best option: (4)

As we increase s from 0, the squared bias of $\hat{\beta}$ will steadily decrease because the model is highly biased when s = 0 and then the bias is decreased. The coefficients will increase to their least squares estimates and the model is becoming more and more flexible which provokes a steady decrease in bias.

2.(a)

```
library(ISLR)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-13
```

```
library(boot)
library(leaps)
library(SignifReg)
set.seed(23456)
data("College")
```

```
# Randomly pick observations from the data for the test data
test_id = sample(1:nrow(College),size=floor(0.30*length(1:nrow(College))))
College_train <- College[-test_id, ]
College_test <- College[test_id, ]
```

(b)

```
fit_linear <- lm(Apps ~ ., data = College_train)
mse = function(model, y, data) {
  yhat = predict(model, data)
```

```
  mean((y - yhat)^2)
}
training_err_linear = mse(fit_linear, College_train$Apps, College_train)
training_err_linear
```

```
## [1] 993164.6
```

```
test_err_linear = mse(fit_linear, College_test$Apps, College_test)
test_err_linear
```

```
## [1] 1300431
```

The training error is 993164.6, and the test error is 1300431.

(c)

```
#forward selection
regfit_fwd = SignifReg(Apps~., data = College_train, alpha = 0.05, direction = "forward",
                       criterion = "p-value", correction = "FDR")
summary(regfit_fwd)
```

```
##
## Call:
## lm(formula = reg, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5647.2  -445.2   -28.0   320.9  6877.5
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  532.67828  275.39506   1.934 0.053610 .
## Accept         1.66367    0.04262  39.038  < 2e-16 ***
## Top10perc     45.25051    6.41865   7.050 5.54e-12 ***
## Enroll        -0.74356    0.11826  -6.287 6.70e-10 ***
## PrivateYes  -785.61216  134.10855  -5.858 8.18e-09 ***
## Expend         0.04746    0.01236   3.841 0.000137 ***
## PhD          -10.54103    3.50697  -3.006 0.002773 **
## Top25perc    -11.86217    5.00593  -2.370 0.018159 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1029 on 536 degrees of freedom
## Multiple R-squared:  0.933,  Adjusted R-squared:  0.9322
## F-statistic:  1067 on 7 and 536 DF,  p-value: < 2.2e-16
```

```
training_err_fwd = mse(regfit_fwd, College_train$Apps, College_train)
training_err_fwd
```

```
## [1] 1043037
```

```
test_err_fwd = mse(regfit_fwd, College_test$Apps, College_test)
test_err_fwd
```

```
## [1] 1334782
```

```
#backward selection
regfit_bwd = SignifReg(Apps~., data = College_train, alpha = 0.05, direction = "backward",
```

```
                        criterion = "p-value", correction = "FDR")
summary(regfit_bwd)
```

```
##
## Call:
## lm(formula = reg, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5589.8  -440.1    -1.4   315.3  6658.0
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  182.17566  203.56862   0.895  0.37124
## PrivateYes  -403.41491  146.86794  -2.747  0.00622 **
## Accept         1.69288    0.04301  39.361  < 2e-16 ***
## Enroll        -0.83323    0.11921  -6.990 8.21e-12 ***
## Top10perc     45.82197    6.35006   7.216 1.84e-12 ***
## Top25perc    -12.12395    4.91888  -2.465  0.01402 *
## Outstate      -0.08479    0.01876  -4.519 7.65e-06 ***
## Expend         0.06782    0.01346   5.038 6.45e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1018 on 536 degrees of freedom
## Multiple R-squared:  0.9344, Adjusted R-squared:  0.9335
## F-statistic:  1091 on 7 and 536 DF,  p-value: < 2.2e-16
```

```
training_err_bwd = mse(regfit_bwd, College_train$Apps, College_train)
training_err_bwd
```

```
## [1] 1021693
```

```
test_err_bwd = mse(regfit_bwd, College_test$Apps, College_test)
test_err_bwd
```

```
## [1] 1355206
```

By forward selection, the variables PrivateYes, Accept, Enroll, Top10perc, Top25perc, PhD and Expend are recommended to include in the final model. The train error is 1043037, and the test error is 1334782.
By backward selection, the variables PrivateYes, Accept, Enroll, Top10perc, Top25perc, Outstate and Expend are recommended to include in the final model. The train error is 1021693, and the test error is 1355206.

(d)

```
regfit.full = regsubsets(Apps~., data = College_train, nvmax = 18)
reg.summary = summary(regfit.full)
#AIC
id_AIC = which.min(reg.summary$cp)
coef(regfit.full, id_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
```

```
names(coef(regfit.full, id_AIC))
```

```
##  [1] "(Intercept)" "PrivateYes"  "Accept"      "Enroll"      "Top10perc"
##  [6] "Top25perc"   "Outstate"    "Room.Board"  "PhD"         "Expend"
## [11] "Grad.Rate"
```

```
model_AIC = lm(Apps ~ Private+Accept+Enroll+Top10perc+Top25perc+Outstate+Room.Board+PhD+Expend
               +Grad.Rate, data = College_train)
training_err_AIC = mse(model_AIC, College_train$Apps, College_train)
training_err_AIC
```

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

```
test_err_AIC = mse(model_AIC, College_test$Apps, College_test)
test_err_AIC
```

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

```
#BIC
id_BIC = which.min(reg.summary$bic)
coef(regfit.full, id_BIC)
```

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

```
names(coef(regfit.full, id_BIC))
```

```
## [1] "(Intercept)" "PrivateYes"  "Accept"      "Enroll"      "Top10perc"
## [6] "Outstate"    "Expend"
```

```
model_BIC = lm(Apps ~ Private+Accept+Enroll+Top10perc+Outstate+Expend, data = College_train)
training_err_BIC = mse(model_BIC, College_train$Apps, College_train)
training_err_BIC
```

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

```
test_err_BIC = mse(model_BIC, College_test$Apps, College_test)
test_err_BIC
```

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

By AIC criterian, the variables PrivateYes, Accept, Enroll, Top10perc, Top25perc, Outstate, Room.Board, PhD, Expend, and Grad.Rate are recommended to include in the final model. The train error is 1001215, and the test error is 1282321.

By BIC criterian, the variables PrivateYes, Accept, Enroll, Top10perc, Outstate, and Expend are recommended to include in the final model. The train error is 1033273, and the test error is 1380054.

(e)

```
X = model.matrix(Apps~., College_train)[, -1]
y = College_train$Apps
grid = 10^seq(10, -2, length = 100)
ridge.mod = glmnet(X, y, alpha = 0, lambda = grid)
cv.out_ridge = cv.glmnet(X, y, alpha = 0)
minlam_ridge = cv.out_ridge$lambda.min
minlam_ridge
```

```
## [1] 411.4072
```

```
ridge.pred_train = predict(ridge.mod, s = minlam_ridge, newx = X)
training_err_ridge = mean((ridge.pred_train - y)^2)
training_err_ridge
```

```
## [1] 1384811
```

```
ridge.pred_test = predict(ridge.mod, s = minlam_ridge, newx = model.matrix(Apps~., College_test)[, -1])
test_err_ridge = mean((ridge.pred_test - College_test$Apps)^2)
test_err_ridge
```

```
## [1] 1223126
```

The value of $\lambda$ chosen by smallest cross-validation error is 411.4072. The train error is 1384811, and the test error is 1223126.

(f)

```
set.seed(23456)
lasso.mod = glmnet(X, y, alpha = 1, lambda = grid)
cv.out_lasso = cv.glmnet(X, y, alpha = 1)
minlam_lasso = cv.out_lasso$lambda.min
minlam_lasso
```

```
## [1] 3.495947
```

```
lasso.pred_train = predict(lasso.mod, s = minlam_lasso, newx = X)
training_err_lasso = mean((lasso.pred_train - y)^2)
training_err_lasso
```

```
## [1] 994152.8
```

```
lasso.pred_test = predict(lasso.mod, s = minlam_lasso, newx = model.matrix(Apps~., College_test)[, -1])
test_err_lasso = mean((lasso.pred_test - College_test$Apps)^2)
test_err_lasso
```

```
## [1] 1292839
```

The value of $\lambda$ chosen by smallest cross-validation error is 3.495947. The train error is 994152.8, and the test error is 1292839.

(g) The test errors of different methods range from 1223126 to 1380054, with mean 1310268 and standard deviation 51908. We can predict the number of college applications received most accurately by ridge regression. For prediction, I recommend the ridge regression method because it has the smallest test error from the prediction model. For interpretation, I recommend the AIC criterion method because it uses fewer variables than the other with similar test errors.