ISyE 6501 HW8

October 9, 2019

Question 9.1

```
# reading data
data <- read.delim("uscrime.txt", header=TRUE)
attach(data)

# loading packages for stepAIC, glmnet
library(MASS)
library(glmnet)

# avoiding scientific notation for results and requiring 3 digits
options(scipen=4, digits = 3)

# setting seed for reproducibility
set.seed(118)
```

Data Preparation

```
# removing outliers
# we concluded in HW3 that obs 4 and 26 are likely to be outliers
# we want to be conservative and therefore we remove them from the data
data <- data[-c(4,26),]

# creating user-defined function calculating R-squared
# the function takes fitted values and actual values as args
R2 <- function(pred, y) {
SST <- sum((y-mean(y))^2)
SSM <- sum((pred-mean(pred))^2)
R2 <- SSM/SST
print(R2)
}
```

Stepwise Regression

```
# running the full model
model <- lm(Crime ~., data = data)

# training model with stepwise regression (backwards and forwards)
# the algorithm stops at iteration 8 with AIC = 484
# the predictors selected are M, Ed, Po1, Pop, U1, U2, Ineq, Prob
step <- stepAIC(model, direction="both", trace = FALSE)
```

The algorithm stops when it cannot decrease the AIC by removing any of the leftover predictors. The AIC is an information criterion that penalizes models with many regressors; differently than unadjusted R2, it

balances accuracy and simplicity. As expected, most of the regressors in the final model turned out to be significant. The R2 value informs us that the model explains 44.4% of the variation in the dependent variable Crime.

```
# running the final model
model <- lm(Crime ~ M + Ed + Po1 + Pop + U1 + U2 + Ineq + Prob, data = data)
summary(model)
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + Pop + U1 + U2 + Ineq + Prob,
##     data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -510.4 -101.7    7.8  134.0  561.5
##
## Coefficients:
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -4228.75    1017.82   -4.15   0.00019 ***
## M              84.11      35.05    2.40   0.02172 *
## Ed            182.21      51.71    3.52   0.00118 **
## Po1           106.30      20.03    5.31 0.0000059 ***
## Pop            -1.66       1.27   -1.30   0.20019
## U1          -4438.86    3040.70   -1.46   0.15301
## U2            164.55      72.18    2.28   0.02865 *
## Ineq           64.82      14.53    4.46 0.0000769 ***
## Prob        -4316.73    1554.42   -2.78   0.00865 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 198 on 36 degrees of freedom
## Multiple R-squared:  0.685,  Adjusted R-squared:  0.614
## F-statistic: 9.76 on 8 and 36 DF,  p-value: 0.00000044
```

```
R2(model$fitted.values, Crime)
```

```
## [1] 0.444
```

Lasso Regression

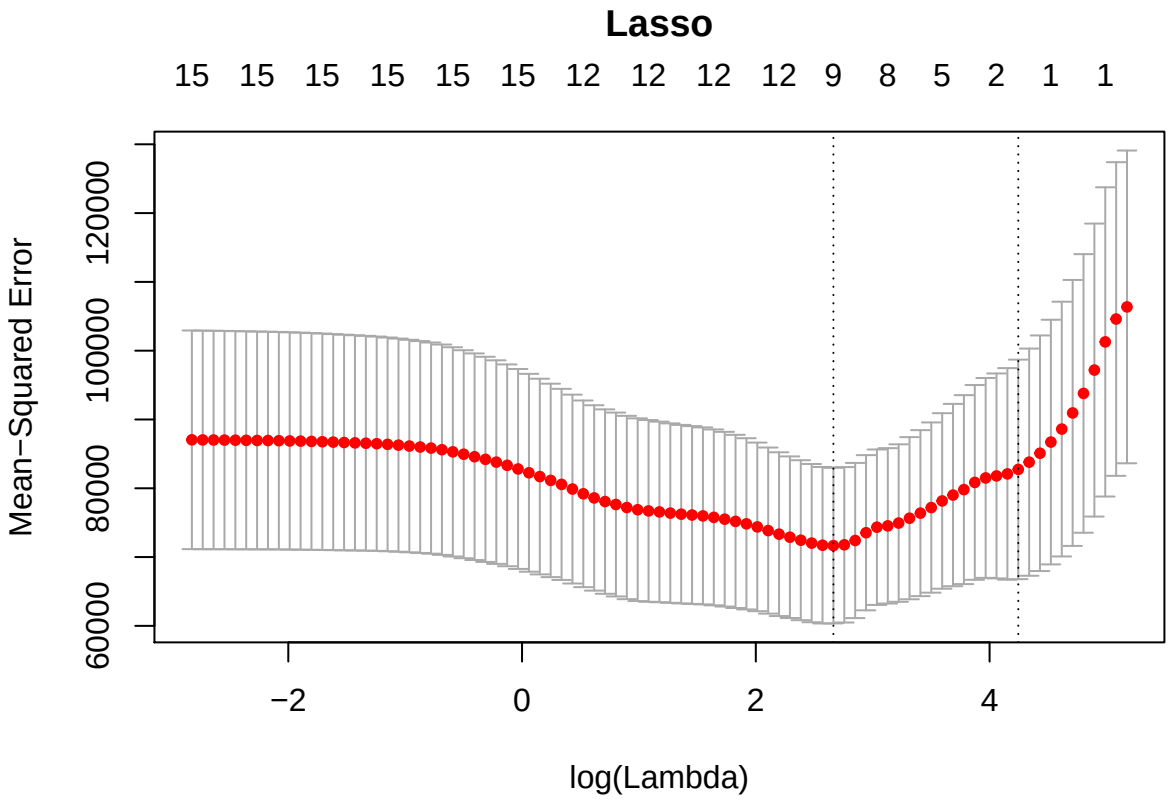
```
# the default in glmnet() standardizes the input data
# however, predictors have to be arranged in matrix format

x <- model.matrix(Crime~., data)[,-Crime]
y <- data$Crime
```

Other than a specific α, a value for λ needs be specified in glmnet(). λ is a multiplier specifying the degree of coefficient shrinkage. Two approaches might be used to choose the optimal λ with the cross-validation function cv.glmnet() training lasso using cross-validation (5-fold is default). The first option is to chose a value for λ such that the prediction error is minimized. The alternate route is to set λ such that two conditions are satisfied at the same time: *i)* the number of predictors is minimized; *ii)* the new λ lies within at most one SD of the optimal λ.

```
# plotting log(lambda) against associated MSE
lasso_cv <- cv.glmnet(x, y, alpha = 1, family = "gaussian")
```

```r
plot(lasso_cv)
title("Lasso", line = +2.5)
```




```r
# lambda minimizing the prediction error
lasso_cv$lambda.min
```

```
## [1] 14.4
```

```r
# lambda minimizing number of predictors while lying within one SD of optimal lambda
lasso_cv$lambda.1se
```

```
## [1] 69.8
```

Next, we run lasso regression with optimal λ, saved the fitted values and displayed R2 and coefficients. The non-zero predictors are M, So, Ed, Po1, M.F, NW, U2, Ineq, and Prob; the variation in Crime explained by the model reduces to 42.3%.

```r
lasso1 <- glmnet(x, y, family="gaussian", alpha=1, lambda = lasso_cv$lambda.min)
lasso1_pred <- predict(lasso1, s = lasso_cv$lambda.min, newx = x)
R2(lasso1_pred, y)
```

```
## [1] 0.423
```

```r
coef(lasso1)
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept) -2699.040
## (Intercept)     .
## M              48.250
## So             60.392
## Ed             77.164
```

```
## Po1            85.973
## Po2             .
## LF              .
## M.F             7.762
## Pop             .
## NW              0.272
## U1              .
## U2             21.470
## Wealth          .
## Ineq           35.395
## Prob        -3705.628
## Time            .
```

Next, we run lasso regression with the alternate λ. As expected, dimensionality reduces substantially with only predictors Po1 and Prob being non-zero. Consistently, the R2 value plummeted to 11.7%.

```
lasso2 <- glmnet(x, y, family="gaussian", alpha=1, lambda = lasso_cv$lambda.1se)
lasso2_pred <- predict(lasso2, s = lasso_cv$lambda.1se, newx = x)
R2(lasso2_pred, y)
```

```
## [1] 0.117
```

```
coef(lasso2)
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                  s0
## (Intercept)  570.5
## (Intercept)    .
## M              .
## So             .
## Ed             .
## Po1           38.7
## Po2            .
## LF             .
## M.F            .
## Pop            .
## NW             .
## U1             .
## U2             .
## Wealth         .
## Ineq           .
## Prob        -626.2
## Time           .
```

Elastic Net Regression

We saw the case where α is set to 1 (i.e lasso regression). The case where α is set to 0 in glmnet() generates ridge regression whereas values between 0 and 1 produce elastic net regressions. We tried out 0.1 increments of $[0.1, 0.9]$ using the optimal λ.

```
# setting up destination vectors for the for loop
# creating destination vectors for alpha and R2
alpha <- c(length(9))
R_squared <- c(length(9))
```

```
# iterating glmnet in 0.1 increments of [0.1,0.9]
for (i in 0:9){
```

```
en_cv <- cv.glmnet(x, y, family="gaussian", alpha=i/10)
en <- glmnet(x, y, family="gaussian", alpha= i/10, lambda = en_cv$lambda.min)
en_pred <- predict(en, s = en_cv$lambda.min, newx = x)
SST <- sum((y-mean(y))^2)
SSM <- sum((en_pred-mean(en_pred))^2)
R2 <- SSM/SST
alpha[i]  <- i/10
R_squared[i]  <- R2
}
```

We saved the estimation results in a dataframe. We notice that $\alpha = 0.5$ returns the highest R2 value at 56.8%.

```
as.data.frame(cbind(alpha, R_squared))
```

```
##   alpha R_squared
## 1   0.1     0.432
## 2   0.2     0.373
## 3   0.3     0.427
## 4   0.4     0.352
## 5   0.5     0.568
## 6   0.6     0.383
## 7   0.7     0.245
## 8   0.8     0.309
## 9   0.9     0.537
```

The takeawys from this homework assignment are that lasso regression is a way to reduce the dimensionality of a modeling problem whereas stepwise regression takes care of predictive power. Furthermore, lasso regression (and so elastic net regression) requires some fine-tuning of the λ parameter, and we see how estimation results vary substantially with different values of λ. This occurrence might be driven by the small sample size we are working on, therefore all of our results need be taken with some doses of caution. Stepwise regression is a way to maximize the predictive power of a model, and specifically, to increase the chance of seeing significant entries in the final regression output. Neither lasso regression nor stepwise regression take into account what social scientists term the "proper specification" of a model, or Data Generating Process (DGP) for the outcome of interest. Is this a real problem though? It is, indeed, because properly specified models are more likely to be replicable with different datasets by different researchers at different points in time. Instead, stepwise and lasso regression only (albeit powerfully) take care of the predictive power and simplicity of the model respectively, and possibly include predictors that have little to do with the DGP of the outcome of interest and might be heavily contingent on a given dataset. Much in the same way, they might end up excluding predictors that properly specify the DGP of the outcome variable. We cannot expect crime records coming out of countries/regions/cities from all over the world to report variables such as LF and U2, therefore we always sacrifice generalizability whenever we focus on predictive performance (stepwise regression) and dimensionality reduction (lasso) alone. Again, the overall caveat to all of our statements is that we are operating on a (very) small sample size, and they are tentative and indicative in nature.