CS 598: Homework 2

Fall 2019, Apoorva (apoorva6)

Contents

Question 1 Linear Model Selection

We will use the Boston Housing data again. This time, we do not scale the covariate. We will still remove `medv`, `town` and `tract` from the data and use `cmedv` as the outcome. If you do not use R, you can download a '.csv' file from the course website.

```
library(mlbench)
data(BostonHousing2)
BH = BostonHousing2[, !(colnames(BostonHousing2) %in% c("medv", "town", "tract"))]
linear_model <- lm(cmedv~., data = BH)
```

Answer the following questions:

a. Report the most significant variable from this full model with all features.

```
p_value <- summary(linear_model)$coefficients[-1,4]
p_value
```

```
##          lon          lat         crim           zn        indus        chas1
## 2.437703e-01 2.210545e-01 1.435758e-03 7.551132e-04 8.051063e-01 3.023683e-03
##          nox           rm          age          dis          rad          tax
## 8.927787e-05 4.580106e-18 8.534405e-01 5.612294e-11 5.227040e-06 5.917649e-04
##      ptratio            b        lstat
## 2.922579e-10 6.177624e-04 5.274420e-24
```

At a significance level of 0.05 we fail to reject the Null Hypothesis that there is no relationship between some of the predictors and the response. The following set of 11 predictors are the ones that define the response the most: crim, zn, chas1, nox, rm, dis, rad, tax, ptratio, b, lstat

Out the this the most significant variable would be the **lstat** variable which has the lowest Pvalue among them all.

b. Starting from this full model, use stepwise regression with both forward and backward and BIC criterion to select the best model. Which variables are removed from the full model?

```
names(coef(linear_model))[!(names(coef(linear_model))%in% names(coef(stepwise_model)))]
```

```
## [1] "lon"   "lat"   "indus" "age"
```

The variables exculded from the model with the lowest BIC values are **lon, lat, indus, age**

c. Starting from this full model, use the best subset selection and list the best model of each model size.

```
library(leaps)
best_sub_selection <- regsubsets(BH$cmedv~., data = BH,nvmax =15)
summary(best_sub_selection)$outmat
```

```
##           lon lat crim zn  indus chas1 nox rm  age dis rad tax ptratio b   lstat
## 1  ( 1 )  " " " " " "  " " " "   " "   " " " " " " " " " " " " " "     " " "*"
## 2  ( 1 )  " " " " " "  " " " "   " "   " " "*" " " " " " " " " " "     " " "*"
## 3  ( 1 )  " " " " " "  " " " "   " "   " " "*" " " " " " " " " "*"     " " "*"
## 4  ( 1 )  " " " " " "  " " " "   " "   " " "*" " " "*" " " " " "*"     " " "*"
## 5  ( 1 )  " " " " " "  " " " "   " "   "*" "*" " " "*" " " " " "*"     " " "*"
## 6  ( 1 )  " " " " " "  " " " "   "*"   "*" "*" " " "*" " " " " "*"     " " "*"
## 7  ( 1 )  " " " " " "  " " " "   "*"   "*" "*" " " "*" " " " " "*"     "*" "*"
## 8  ( 1 )  " " " " " "  "*" " "   "*"   "*" "*" " " "*" " " " " "*"     "*" "*"
## 9  ( 1 )  " " " " " "  " " " "   "*"   "*" "*" " " "*" "*" "*" "*"     "*" "*"
## 10  ( 1 ) " " " " "*"  "*" " "   " "   "*" "*" " " "*" "*" "*" "*"     "*" "*"
## 11  ( 1 ) " " " " "*"  "*" " "   "*"   "*" "*" " " "*" "*" "*" "*"     "*" "*"
## 12  ( 1 ) " " "*" "*"  "*" " "   "*"   "*" "*" " " "*" "*" "*" "*"     "*" "*"
## 13  ( 1 ) "*" "*" "*"  "*" " "   "*"   "*" "*" " " "*" "*" "*" "*"     "*" "*"
## 14  ( 1 ) "*" "*" "*"  "*" "*"   "*"   "*" "*" " " "*" "*" "*" "*"     "*" "*"
## 15  ( 1 ) "*" "*" "*"  "*" "*"   "*"   "*" "*" "*" "*" "*" "*" "*"     "*" "*"
```

d. Use the Cp criterion to select the best model from part c). Which variables are removed from the full model? What is the most significant variable?

```r
lowest_cp <- which.min(summary(best_sub_selection)$cp)
best_pred <- names(coef(best_sub_selection,lowest_cp))
best_pred[-1]
```

```
##  [1] "crim"    "zn"      "chas1"   "nox"     "rm"      "dis"     "rad"     "tax"
##  [9] "ptratio" "b"       "lstat"
```

```r
# Building LM using the model chosen from best subset selection
best_model <- lm(BH$cmedv~ crim+zn+chas+nox+rm+dis+rad+tax+ptratio+b+lstat, data=BH)
coef(summary(best_model))[,4]
```

```
##  (Intercept)         crim           zn        chas1          nox           rm
## 2.017013e-12 1.099454e-03 4.813561e-04 1.360054e-03 1.058023e-06 2.317178e-19
##          dis          rad          tax      ptratio            b        lstat
## 1.354236e-15 3.082374e-06 3.330966e-04 2.766814e-12 5.608988e-04 2.855042e-26
```

- The variables removed from the Full model after best subset selection are : **lon, lat, indus, age**
- The most significant variable out of the 11 variables in the best subset model is **lstat** as it has the lowest P-value

Question 2 Code Your Own Lasso

For this question, we will write our own Lasso code. You are not allowed to use any built-in package that already implements Lasso. First, we will generate simulated data. Here, only X_1, X_2 and X_3 are important, and we will not consider the intercept term.

```r
library(MASS)
set.seed(1)
n = 200
p = 200
# generate data
V = matrix(0.2, p, p)
diag(V) = 1
X = as.matrix(mvrnorm(n, mu = rep(0, p), Sigma = V))
y = X[, 1] + 0.5*X[, 2] + 0.25*X[, 3] + rnorm(n)
# we will use a scaled version
```

```
X = scale(X)
y = scale(y)
```

As we already know, coordinate descent is an efficient approach for solving Lasso. The algorithm works by updating one parameter at a time, and loop around all parameters until convergence.

a. Hence, we need first to write a function that updates just one parameter, which is also known as the soft-thresholding function. Construct the function in the form of `soft_th <- function(b, lambda)`, where `b` is a number that represents the one-dimensional linear regression solution, and `lambda` is the penalty level. The function should output a scaler, which is the minimizer of

$$(x - b)^2 + \lambda|b|$$

```
soft_th <- function(b,lambda)
{
  soft_th_values <- lambda/2
  ifelse(b > soft_th_values, (b - soft_th_values) , ifelse( b < -(soft_th_values), (b+soft_th_values),0
}
```

b. Now lets pretend that at an iteration, the current parameter $\boldsymbol{\beta}$ value is given below (as `beta_old`, i.e., $\boldsymbol{\beta}^{\text{old}}$). Apply the above soft-thresholding function to update all p parameters sequencially one by one to complete one "loop" of the updating scheme. Please note that we use the Gauss-Seidel style coordinate descent, in which the update of the next parameter is based on the new values of previous entries. Hence, each time a parameter is updated, you should re-calculate the residual

$$\mathbf{r} = \mathbf{y} - \mathbf{X}^{\text{T}}\boldsymbol{\beta}$$

so that the next parameter update reflects this change. After completing this one enrire loop, print out the first 3 observations of $\mathbf{r}$ and the nonzero entries in the updated $\boldsymbol{\beta}^{\text{new}}$ vector. For this question, use `lambda` $= 0.7$ and

```
beta_old = rep(0, p)
 lambda = 0.7

 for(j in 1:p)
 {
   beta_star <- (t(X[,j]) %*% (y - (X[,-j] %*% beta_old[-j]))) / ( t(X[,j]) %*% X[,j])
   beta_old[j] <-  soft_th(beta_star,lambda)
 }

 r <- y - (X %*% beta_old)
```

- The first 3 observations of $\mathbf{r}$: -0.0760434, 0.146774, 0.1562568
- The nonzero entries in the updated β^{new} vector : 0.3529634, 0.0902926

c. Now, let us finish the entire Lasso algorithm. We will write a function `myLasso(X, y, lambda, tol, maxitr)`. Set the tolerance level `tol` = 1e-5, and `maxitr` = 100 as the default value. Use the "one loop" code that you just wrote in the previous question, and integrate that into a grand for-loop that will continue updating the parameters up to `maxitr` runs. Check your parameter updates once in this grand loop and stop the algorithm once the ℓ_1 distance between $\boldsymbol{\beta}^{\text{new}}$ and $\boldsymbol{\beta}^{\text{old}}$ is smaller than `tol`. Use `beta_old = rep(0, p)` as the initial value, and `lambda` $= 0.3$. After the algorithm converges, report the following: i) the number of iterations took; ii) the nonzero entries in the final beta parameter estimate, and iii) the first three observations of the residual. Please write your algorithm as efficient as possible.

```
myLasso <- function(X, y, lambda, tol, maxitr)
{
```

```r
 iter <- 0
 beta_new = rep(0, p)
 beta_old = rep(0, p)
diff_l1 <- 10

while (diff_l1 > tol & iter < maxitr){
for(j in 1:p)
 {
   beta_star <- (t(X[,j]) %*% (y - (X[,-j] %*% beta_old[-j]))) / ( t(X[,j]) %*% X[,j])
   beta_old[j] <-  soft_th(beta_star,lambda)
 }

 diff_l1 <- sum(abs(beta_new-beta_old))
 beta_new <- beta_old
 iter <- iter + 1

}

 r <- y - (X %*% beta_old)
 return(list("iter" = iter, "r"= r[1:3], "nonzer_beta"= beta_new[beta_new!=0]))

}


#Calling function
out <- myLasso(X, y, 0.3,  1e-5, 100)
out
```

```
## $iter
## [1] 9
##
## $r
## [1] -0.1757378  0.2262848  0.1912103
##
## $nonzer_beta
## [1] 0.457802236 0.226116017 0.114399954 0.001018992 0.011551407 0.004669249
```

d. Now we have our own Lasso function, let's check the result and compare it with the `glmnet` package. Note that for the glmnet package, their `lambda` should be set as half of ours. Comment on the accuracy of the algorithm that we wrote. Please note that the distance of the two solutions should not be larger than 0.005.

```r
# Lasso using glmnet
library(glmnet)
lasso = glmnet(X, y,alpha =1,lambda = 0.15,thresh=1e-5)
nonzero_beta_lasso <-  lasso$beta[lasso$beta !=0]

# Checking the difference in the betas between myLasso and glmnet
distance_betas <- sum(abs(nonzero_beta_lasso-out[[3]]))
distance_betas
```

```
## [1] 0.001194457
```

The MSE of our model is 0.3758599. Our algorithm results are very close to the prediction values of coefficients that we get from the library function glmnet.

Question 3 Cross-Validation for Model Selection

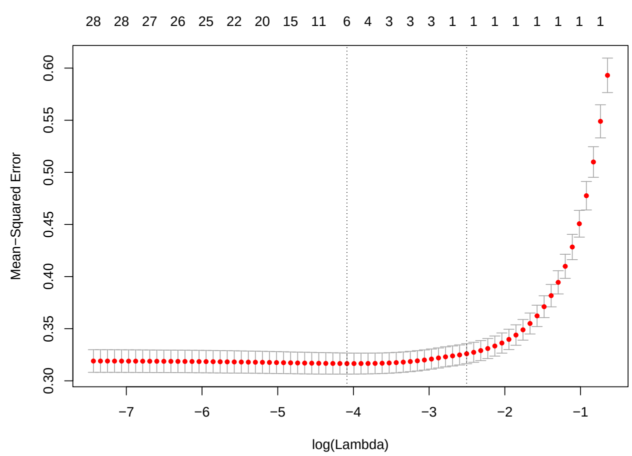
We will use the Walmart Sales data provided on Kaggle. For this question, we will use only the Train.csv file. The file is also available at here.

a. Do the following to process the data:
   - Read data into R
   - Convert character variables into factors
   - Remove `Item_Identifier`
   - Further convert all factors into dummy variables
b. Use all variables to model the outcome `Item_Outlet_Sales` in its *log* scale. First, we randomly split the data into two parts with equal size. Make sure that you set a random seed so that the result can be replicated. Treat one as the training data, and the other one as the testing data. For the training data, perform the following:
   - Use cross-validation to select the best Lasso model. Consider both `lambda.min` and `lambda.1se`. Provide additional information to summarize the model fitting result
   - Use cross-validation to select the best Ridge model. Consider both `lambda.min` and `lambda.1se`. Provide additional information to summarize the model fitting result
   - Test these four models on the testing data and report and compare the prediction accuracy

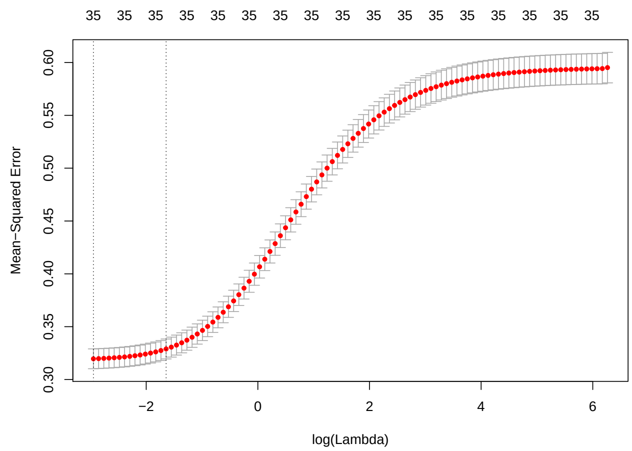
```
#install.packages("glmnet")
library(glmnet)
#Split data into test and train
set.seed(1)
train_ind <- sample(nrow(WalMartData), nrow(WalMartData)/2)

train <- WalMartData[train_ind, ]
test <- WalMartData[-train_ind, ]

########## Lasso model ###########
lasso_train = cv.glmnet(train[,-(ncol(train))], log(train[,ncol(train)]))
lasso_lam_min <- lasso_train$lambda.min
lasso_lam_1se <- lasso_train$lambda.1se
plot(lasso_train)
```




```
########## Ridge model ###########
ridge_train = cv.glmnet(train[,-(ncol(train))], log(train[,(ncol(train))]), alpha = 0)
ridge_lam_min <- ridge_train$lambda.min
ridge_lam_1se <- ridge_train$lambda.1se
plot(ridge_train)
```




```
#Predicting on test dataset
# Lasso with min lamda
lasso_test_lammin = predict(lasso_train, s =lasso_lam_min, newx= test[,-(ncol(test))])
acc_min_lasso <- mean((lasso_test_lammin - log(test[,ncol(test)]))^2)
# Lasso with 1se lamda
lasso_test_lam1se = predict(lasso_train, s =lasso_lam_1se, newx= test[,-(ncol(test))])
acc_1se_lasso <- mean((lasso_test_lam1se - log(test[,ncol(test)]))^2)
# Ridge with min lamda
ridge_test_lammin = predict(ridge_train, s =ridge_lam_min, newx= test[,-(ncol(test))])
acc_min_ridge <- mean((ridge_test_lammin - log(test[,ncol(test)]))^2)
# Ridge with 1se lamda
ridge_test_lam1se = predict(ridge_train, s =ridge_lam_1se, newx= test[,-(ncol(test))])
acc_1se_ridge <- mean((ridge_test_lam1se - log(test[,ncol(test)]))^2)

MSE_accuracies <- c(acc_min_lasso,acc_1se_lasso,acc_min_ridge,acc_1se_ridge)
names(MSE_accuracies) <- c("Lasso_minLambda","Lasso_1seLambda","Ridge_minLamda","Ridge_1seLamda")
MSE_accuracies
```

```
## Lasso_minLambda Lasso_1seLambda  Ridge_minLamda  Ridge_1seLamda
##       0.2893344       0.3015013       0.2903535       0.3000081
```

The Lasso model using min lamda performs the best among the four models above.

- Using lambda.min as the best lambda for lasso model, gives the following regression coefficients:

```
coef(lasso_train, lasso_train$lambda.min)
```

```
## 41 x 1 sparse Matrix of class "dgCMatrix"
##                                          1
## (Intercept)                    6.359462429
## Item_Weight                    .
## Item_Fat_ContentLF             .
## Item_Fat_Contentlow fat        .
## Item_Fat_ContentLow Fat        .
## Item_Fat_Contentreg           -0.065794996
## Item_Fat_ContentRegular        .
## Item_Visibility                .
## Item_TypeBreads                .
## Item_TypeBreakfast            -0.041866185
## Item_TypeCanned                .
## Item_TypeDairy                 .
## Item_TypeFrozen Foods          .
```

```
## Item_TypeFruits and Vegetables  .
## Item_TypeHard Drinks             .
## Item_TypeHealth and Hygiene      .
## Item_TypeHousehold               .
## Item_TypeMeat                    .
## Item_TypeOthers                  .
## Item_TypeSeafood                 .
## Item_TypeSnack Foods             0.007166267
## Item_TypeSoft Drinks             .
## Item_TypeStarchy Foods           .
## Item_MRP                         0.008086092
## Outlet_IdentifierOUT013          .
## Outlet_IdentifierOUT017          .
## Outlet_IdentifierOUT018         -0.080991995
## Outlet_IdentifierOUT019          .
## Outlet_IdentifierOUT027          .
## Outlet_IdentifierOUT035          .
## Outlet_IdentifierOUT045          .
## Outlet_IdentifierOUT046          .
## Outlet_IdentifierOUT049          .
## Outlet_Establishment_Year        .
## Outlet_SizeMedium                .
## Outlet_SizeSmall                 .
## Outlet_Location_TypeTier 2       .
## Outlet_Location_TypeTier 3      -0.047845583
## Outlet_TypeSupermarket Type1     .
## Outlet_TypeSupermarket Type2     .
## Outlet_TypeSupermarket Type3     .
```

- Using lambda.1se as the best lambda for lasso model, gives the following regression coefficients:

```
coef(lasso_train, lasso_train$lambda.1se)
```

```
## 41 x 1 sparse Matrix of class "dgCMatrix"
##                                           1
## (Intercept)                     6.468405521
## Item_Weight                     .
## Item_Fat_ContentLF              .
## Item_Fat_Contentlow fat         .
## Item_Fat_ContentLow Fat         .
## Item_Fat_Contentreg             .
## Item_Fat_ContentRegular         .
## Item_Visibility                 .
## Item_TypeBreads                 .
## Item_TypeBreakfast              .
## Item_TypeCanned                 .
## Item_TypeDairy                  .
## Item_TypeFrozen Foods           .
## Item_TypeFruits and Vegetables .
## Item_TypeHard Drinks            .
## Item_TypeHealth and Hygiene     .
## Item_TypeHousehold              .
## Item_TypeMeat                   .
## Item_TypeOthers                 .
## Item_TypeSeafood                .
```

```
## Item_TypeSnack Foods           .
## Item_TypeSoft Drinks           .
## Item_TypeStarchy Foods         .
## Item_MRP                       0.007071194
## Outlet_IdentifierOUT013        .
## Outlet_IdentifierOUT017        .
## Outlet_IdentifierOUT018        .
## Outlet_IdentifierOUT019        .
## Outlet_IdentifierOUT027        .
## Outlet_IdentifierOUT035        .
## Outlet_IdentifierOUT045        .
## Outlet_IdentifierOUT046        .
## Outlet_IdentifierOUT049        .
## Outlet_Establishment_Year      .
## Outlet_SizeMedium              .
## Outlet_SizeSmall               .
## Outlet_Location_TypeTier 2     .
## Outlet_Location_TypeTier 3     .
## Outlet_TypeSupermarket Type1   .
## Outlet_TypeSupermarket Type2   .
## Outlet_TypeSupermarket Type3   .
```

- Using lambda.min as the best lambda for ridge model, gives the following regression coefficients:

```r
coef(ridge_train, ridge_train$lambda.min)
```

```
## 41 x 1 sparse Matrix of class "dgCMatrix"
##                                           1
## (Intercept)                     7.0619225780
## Item_Weight                     0.0015733365
## Item_Fat_ContentLF              0.0210537501
## Item_Fat_Contentlow fat         0.1110749888
## Item_Fat_ContentLow Fat        -0.0025879914
## Item_Fat_Contentreg            -0.1948953571
## Item_Fat_ContentRegular         0.0067128831
## Item_Visibility                -0.1387042818
## Item_TypeBreads                 0.0570176014
## Item_TypeBreakfast             -0.1967047732
## Item_TypeCanned                 0.0301037716
## Item_TypeDairy                 -0.0489278817
## Item_TypeFrozen Foods          -0.0244916880
## Item_TypeFruits and Vegetables -0.0118785826
## Item_TypeHard Drinks            0.0165472032
## Item_TypeHealth and Hygiene    -0.0069530378
## Item_TypeHousehold             -0.0256674341
## Item_TypeMeat                   0.0101035189
## Item_TypeOthers                -0.0548302218
## Item_TypeSeafood                0.1203989909
## Item_TypeSnack Foods            0.0423314498
## Item_TypeSoft Drinks           -0.0022057267
## Item_TypeStarchy Foods         -0.1029120522
## Item_MRP                        0.0078213761
## Outlet_IdentifierOUT013        -0.0122794993
## Outlet_IdentifierOUT017         .
## Outlet_IdentifierOUT018        -0.0357617366
```

```
## Outlet_IdentifierOUT019         .
## Outlet_IdentifierOUT027         .
## Outlet_IdentifierOUT035         0.0172083486
## Outlet_IdentifierOUT045         .
## Outlet_IdentifierOUT046        -0.0025541603
## Outlet_IdentifierOUT049         0.0329803813
## Outlet_Establishment_Year      -0.0003615495
## Outlet_SizeMedium              -0.0019856815
## Outlet_SizeSmall                0.0104647528
## Outlet_Location_TypeTier 2      0.0178140191
## Outlet_Location_TypeTier 3     -0.0321985414
## Outlet_TypeSupermarket Type1    0.0358361622
## Outlet_TypeSupermarket Type2   -0.0355652589
## Outlet_TypeSupermarket Type3    .
```

- Using lambda.1se as the best lambda for ridge model, gives the following regression coefficients:

```
coef(ridge_train, ridge_train$lambda.1se)
```

```
## 41 x 1 sparse Matrix of class "dgCMatrix"
##                                            1
## (Intercept)                     7.2063170115
## Item_Weight                     0.0018025775
## Item_Fat_ContentLF              0.0217151792
## Item_Fat_Contentlow fat         0.0996965390
## Item_Fat_ContentLow Fat        -0.0051052607
## Item_Fat_Contentreg            -0.1745166089
## Item_Fat_ContentRegular         0.0081946895
## Item_Visibility                -0.0962042690
## Item_TypeBreads                 0.0540248903
## Item_TypeBreakfast             -0.1691457728
## Item_TypeCanned                 0.0254323601
## Item_TypeDairy                 -0.0285008247
## Item_TypeFrozen Foods          -0.0128309973
## Item_TypeFruits and Vegetables -0.0034422448
## Item_TypeHard Drinks            0.0179559790
## Item_TypeHealth and Hygiene    -0.0043370840
## Item_TypeHousehold             -0.0054497938
## Item_TypeMeat                   0.0158613146
## Item_TypeOthers                -0.0387891085
## Item_TypeSeafood                0.1157664315
## Item_TypeSnack Foods            0.0496777058
## Item_TypeSoft Drinks           -0.0018542348
## Item_TypeStarchy Foods         -0.0711336008
## Item_MRP                        0.0066667355
## Outlet_IdentifierOUT013        -0.0118631592
## Outlet_IdentifierOUT017         .
## Outlet_IdentifierOUT018        -0.0343439333
## Outlet_IdentifierOUT019         .
## Outlet_IdentifierOUT027         .
## Outlet_IdentifierOUT035         0.0172789177
## Outlet_IdentifierOUT045         .
## Outlet_IdentifierOUT046         0.0010931100
## Outlet_IdentifierOUT049         0.0277813058
## Outlet_Establishment_Year      -0.0003577309
```

```
## Outlet_SizeMedium              -0.0042716563
## Outlet_SizeSmall                0.0123294077
## Outlet_Location_TypeTier 2      0.0175271566
## Outlet_Location_TypeTier 3     -0.0309262491
## Outlet_TypeSupermarket Type1    0.0343985473
## Outlet_TypeSupermarket Type2   -0.0342345281
## Outlet_TypeSupermarket Type3    .
```

This shows that the lasso regression eliminates more predictors by setting them to 0.