Linear Model Selection

Rishabh Vaish

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 <- lm(cmedv ~ ., data = BH)
```

The most significant variable from this full model with all features.

For this, I have calculated both statistically significant variable by p-value and most important variable by comparing coefficients of scaled model.

The statistically significant model can be figured using the lowest p-value for the coefficients -

```
summary(linear)
```

```
##
## Call:
## lm(formula = cmedv ~ ., data = BH)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.5831  -2.7643  -0.5994   1.7482  26.0822
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.350e+02  3.032e+02  -1.435 0.152029
## lon         -3.935e+00  3.372e+00  -1.167 0.243770
## lat          4.495e+00  3.669e+00   1.225 0.221055
## crim        -1.045e-01  3.261e-02  -3.206 0.001436 **
## zn           4.657e-02  1.374e-02   3.390 0.000755 ***
## indus        1.524e-02  6.175e-02   0.247 0.805106
## chas1        2.578e+00  8.650e-01   2.980 0.003024 **
## nox         -1.582e+01  4.005e+00  -3.951 8.93e-05 ***
## rm           3.754e+00  4.166e-01   9.011  < 2e-16 ***
## age          2.468e-03  1.335e-02   0.185 0.853440
## dis         -1.400e+00  2.088e-01  -6.704 5.61e-11 ***
## rad          3.067e-01  6.658e-02   4.607 5.23e-06 ***
## tax         -1.289e-02  3.727e-03  -3.458 0.000592 ***
## ptratio     -8.771e-01  1.363e-01  -6.436 2.92e-10 ***
```

```
## b            9.176e-03  2.663e-03   3.446 0.000618 ***
## lstat       -5.374e-01  5.042e-02 -10.660  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.7 on 490 degrees of freedom
## Multiple R-squared:  0.7458, Adjusted R-squared:  0.738
## F-statistic: 95.82 on 15 and 490 DF,  p-value: < 2.2e-16
```

By this logic the minimum P-value is for "rm".

If we want to find the most important model. We will standardize the variables in dataset. As the variables are now on the same scale, the coefficients of model on these values can be compared to get the most important variable.

```
#Scaling BH data
BH_scaled <- data.frame(scale(data.matrix(BH)))
#Fitting a LM model
lm_scaled <- lm(cmedv ~ ., data = BH_scaled)
#Extracting coefficients
coef <- lm_scaled$coefficients[2:length(lm_scaled$coefficients)]
# getting the highest coefficient and corresponding variable
which.max(abs(coef))
```

```
## lstat
##    15
```

```
max(abs(coef))
```

```
## [1] 0.4179728
```

The most important variable by this method is "lstat". I feel "lstat" is more important on the basis of results obtained in part d.

Starting from this full model, using stepwise regression with both forward and backward and BIC criterion to select the best model. We start with the basic lm full model and use step function to select best model. The names of variables removed are returned

```
#Using step function in both directions and setting k = log(n) to get BIC
lm_best <-
  step(linear,
       direction = "both",
       k = log(nrow(BH)),
       trace = 0)
#Colnames of full LM model
linear_colnames <- names(linear$coefficients)
#Colnames of selected best model
lm_best_colnames <- names(lm_best$coefficients)
#Printing removed columns
linear_colnames[!(linear_colnames %in% lm_best_colnames)]
```

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

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

```
library(leaps)
#Using regsubsets to get best model for all number of variables.
RSSleaps = regsubsets(cmedv ~ ., data = BH, nvmax = NULL)
#Storing and printing the summary
sumleaps <- summary(RSSleaps, matrix = T)
sumleaps
```

```
## Subset selection object
## Call: regsubsets.formula(cmedv ~ ., data = BH, nvmax = NULL)
## 15 Variables  (and intercept)
##         Forced in Forced out
## lon         FALSE      FALSE
## lat         FALSE      FALSE
## crim        FALSE      FALSE
## zn          FALSE      FALSE
## indus       FALSE      FALSE
## chas1       FALSE      FALSE
## nox         FALSE      FALSE
## rm          FALSE      FALSE
## age         FALSE      FALSE
## dis         FALSE      FALSE
## rad         FALSE      FALSE
## tax         FALSE      FALSE
## ptratio     FALSE      FALSE
## b           FALSE      FALSE
## lstat       FALSE      FALSE
## 1 subsets of each size up to 15
## Selection Algorithm: exhaustive
##           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 ) "*" "*" "*"  "*" "*"   "*"   "*" "*" "*" "*" "*" "*" "*"     "*" "*"
```

Use the Cp criterion to select the best model

```
#Using the precalculated Cp values to get best model
# The best model and its Cp:
#Finding minimum Cp value
which.min(sumleaps$cp)
```

```
## [1] 11
```

```
sumleaps$cp[which.min(sumleaps$cp)]
```

```
## [1] 10.68061
```

The best model is the one with lowest Cp i.e the model with 11 variables. The same four variables as that b part are removed ie - "lon" "lat" "indus" "age". To find the most significant variable, we will fit a linear regression using variables selected by leaps.

```
#fitting a linear model on variables selected by leaps to get the most significant and important variab
linear_best <-
  lm(cmedv ~ crim + zn + chas + nox + rm + dis + rad + tax + ptratio + b + lstat,
     data = BH)
summary(linear_best)
```

```
##
## Call:
## lm(formula = cmedv ~ crim + zn + chas + nox + rm + dis + rad +
##     tax + ptratio + b + lstat, data = BH)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.566  -2.686  -0.552   1.790  26.167
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.244827   5.022209   7.217 2.02e-12 ***
## crim         -0.106657   0.032487  -3.283 0.001099 **
## zn            0.047099   0.013402   3.514 0.000481 ***
## chas1         2.727209   0.846606   3.221 0.001360 **
## nox         -17.316823   3.503652  -4.943 1.06e-06 ***
## rm            3.778662   0.402685   9.384  < 2e-16 ***
## dis          -1.520270   0.184071  -8.259 1.35e-15 ***
## rad           0.296555   0.062836   4.720 3.08e-06 ***
## tax          -0.012077   0.003342  -3.613 0.000333 ***
## ptratio      -0.917035   0.127912  -7.169 2.77e-12 ***
## b             0.009202   0.002650   3.473 0.000561 ***
## lstat        -0.528441   0.047001 -11.243  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.694 on 494 degrees of freedom
## Multiple R-squared:  0.7444, Adjusted R-squared:  0.7387
## F-statistic: 130.8 on 11 and 494 DF,  p-value: < 2.2e-16
```

By looking at the P-value the most statistically important variable is "rm" and "lstat" because they have the lowest p-value.

To find the most important variable we will fit the linear model on scaled variables given by leaps and search for the highest beta coefficients

```r
#fitting a linear model on variables selected by leaps to get the most significant and important variab
linear_best <-
  lm(cmedv ~ crim + zn + chas + nox + rm + dis + rad + tax + ptratio + b + lstat,
     data = BH_scaled)
coef <- linear_best$coefficients[2:length(linear_best$coefficients)]
# getting the highest coefficient and corresponding variable
which.max(abs(coef))
```

```
## lstat
##    11
```

```r
max(abs(coef))
```

```
## [1] 0.410973
```

Thus, I feel "lstat" is the most significant variable. "rm" seems most significant initially due to unscaled data. but after removing some variables or scaling the data we find that lstat is the most significant value. This is also confirmed by the result of regsubsets as "lstat" is the variable selected in one variable model.

Cross-Validation for Model Selection

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

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

```r
# Reading data
library(tidyverse)
WalmartSales <- read_csv("train.csv")

#convert y to lag scale
WalmartSales$Item_Outlet_Sales <-
  log(WalmartSales$Item_Outlet_Sales)

#check for character datatype
sapply(WalmartSales, class) == "character"
```

```
##           Item_Identifier                 Item_Weight          Item_Fat_Content
##                      TRUE                     FALSE                      TRUE
##           Item_Visibility                   Item_Type                  Item_MRP
##                     FALSE                      TRUE                     FALSE
##         Outlet_Identifier Outlet_Establishment_Year               Outlet_Size
##                      TRUE                     FALSE                      TRUE
##      Outlet_Location_Type                 Outlet_Type         Item_Outlet_Sales
##                      TRUE                      TRUE                     FALSE
```

```r
char <-
  c(colnames(WalmartSales[sapply(WalmartSales, class) == "character"]))
#convert character to factors
WalmartSales[char] = lapply(WalmartSales[char], factor) ## Convert to factor
```

```r
#remove item identifier
WalmartSales$Item_Identifier <- NULL

# convert factors into dummies
factor <-
  c(colnames(WalmartSales[sapply(WalmartSales, class) == "factor"]))

WalMartData <- model.matrix(
  ~ . - 1,
  data = WalmartSales,
  contrasts.arg = lapply(WalmartSales[factor], contrasts, contrasts =
                           FALSE)
)

colnames(WalMartData)
```

```
##  [1] "Item_Weight"                    "Item_Fat_ContentLF"
##  [3] "Item_Fat_Contentlow fat"        "Item_Fat_ContentLow Fat"
##  [5] "Item_Fat_Contentreg"            "Item_Fat_ContentRegular"
##  [7] "Item_Visibility"                "Item_TypeBaking Goods"
##  [9] "Item_TypeBreads"                "Item_TypeBreakfast"
## [11] "Item_TypeCanned"                "Item_TypeDairy"
## [13] "Item_TypeFrozen Foods"          "Item_TypeFruits and Vegetables"
## [15] "Item_TypeHard Drinks"           "Item_TypeHealth and Hygiene"
## [17] "Item_TypeHousehold"             "Item_TypeMeat"
## [19] "Item_TypeOthers"                "Item_TypeSeafood"
## [21] "Item_TypeSnack Foods"           "Item_TypeSoft Drinks"
## [23] "Item_TypeStarchy Foods"         "Item_MRP"
## [25] "Outlet_IdentifierOUT010"        "Outlet_IdentifierOUT013"
## [27] "Outlet_IdentifierOUT017"        "Outlet_IdentifierOUT018"
## [29] "Outlet_IdentifierOUT019"        "Outlet_IdentifierOUT027"
## [31] "Outlet_IdentifierOUT035"        "Outlet_IdentifierOUT045"
## [33] "Outlet_IdentifierOUT046"        "Outlet_IdentifierOUT049"
## [35] "Outlet_Establishment_Year"      "Outlet_SizeHigh"
## [37] "Outlet_SizeMedium"              "Outlet_SizeSmall"
## [39] "Outlet_Location_TypeTier 1"     "Outlet_Location_TypeTier 2"
## [41] "Outlet_Location_TypeTier 3"     "Outlet_TypeGrocery Store"
## [43] "Outlet_TypeSupermarket Type1"   "Outlet_TypeSupermarket Type2"
## [45] "Outlet_TypeSupermarket Type3"   "Item_Outlet_Sales"
```

```r
dim(WalMartData)
```

```
## [1] 4650   46
```

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

Answer

By fitting the cv.glmnest we recieve two optimized lambda values. Lamnbda.min and lambda.1se. Lambda.min gives the lambda corresponding to minimum training error and lambda.1se corresponds to the value one standard error away from minimum error.

```
set.seed(1)
index <- sample(nrow(WalMartData), 0.5 * nrow(WalMartData))

#Split into test and train
train <- WalMartData[index, ]
test <- WalMartData[-index, ]

library(glmnet)
# Fitting lasso cv to get lambda.min and lambda.1se
lasso_cv <-
  cv.glmnet(train[, -ncol(train)], train[, ncol(train)], alpha = 1, nfolds = 10)
#Fitting lasso with lambda.min
lasso_min <-
  glmnet(train[, -ncol(train)],
         train[, ncol(train)],
         lambda = lasso_cv$lambda.min,
         alpha = 1)
#Fitting lasso with lambda.1se
lasso_1se <-
  glmnet(train[, -ncol(train)],
         train[, ncol(train)],
         lambda = lasso_cv$lambda.1se,
         alpha = 1)

#Fit ridge cv to get ridge.min and ridge.1se
ridge_cv <-
  cv.glmnet(train[, -ncol(train)], train[, ncol(train)], alpha = 0, nfolds = 10)
# Fitting ridge with lambda min
ridge_min <-
  glmnet(train[, -ncol(train)],
         train[, ncol(train)],
         lambda = lasso_cv$lambda.min,
         alpha = 0)
#Fitting ridge with lamda 1se
ridge_1se <-
  glmnet(train[, -ncol(train)],
         train[, ncol(train)],
         lambda = lasso_cv$lambda.1se,
         alpha = 0)
```

The lambda.min values are smaller than lambda.1se values. Hence applying less penalty will allow more variables in lasso and less shrinkage in ridge. Hence the lambda.min model is more complex as compared to lambda.1se.

```
#making predictions
y_lasso_min <- predict(lasso_min, newx = test[, -ncol(test)])
y_lasso_1se <- predict(lasso_1se, newx = test[, -ncol(test)])
y_ridge_min <- predict(ridge_min, newx = test[, -ncol(test)])
```

```r
y_ridge_1se <- predict(ridge_1se, newx = test[, -ncol(test)])

#Calculating RMSE
rmse_lasso_min <-
  sqrt(mean((exp(test[, ncol(test)]) - exp(y_lasso_min)) ^ 2))
rmse_lasso_1se <-
  sqrt(mean((exp(test[, ncol(test)]) - exp(y_lasso_1se)) ^ 2))
rmse_ridge_min <-
  sqrt(mean((exp(test[, ncol(test)]) - exp(y_ridge_min)) ^ 2))
rmse_ridge_1se <-
  sqrt(mean((exp(test[, ncol(test)]) - exp(y_ridge_1se)) ^ 2))

# RMSE of 4 models Lasso_1se, lasso_min, ridge_1se, ridge_min respectively is:
rmse_lasso_1se
```

```
## [1] 1169.603
```

```r
rmse_lasso_min
```

```
## [1] 1148.435
```

```r
rmse_ridge_1se
```

```
## [1] 1157.197
```

```r
rmse_ridge_min
```

```
## [1] 1152.9
```

```r
#calculating rquared
rsq_lasso_min <- cor(exp(test[, ncol(test)]), exp(y_lasso_min)) ^ 2
rsq_lasso_1se <- cor(exp(test[, ncol(test)]), exp(y_lasso_1se)) ^ 2
rsq_ridge_min <- cor(exp(test[, ncol(test)]), exp(y_ridge_min)) ^ 2
rsq_ridge_1se <- cor(exp(test[, ncol(test)]), exp(y_ridge_1se)) ^ 2

#R-Squared of 4 models Lasso_1se, lasso_min, ridge_1se, ridge_min respectively is:
rsq_lasso_1se
```

```
##             s0
## [1,] 0.4539839
```

```r
rsq_lasso_min
```

```
##             s0
## [1,] 0.4542448
```

```r
rsq_ridge_1se
```

```
##             s0
## [1,] 0.4534846
```

```
rsq_ridge_min
```

```
##            s0
## [1,] 0.449452
```

Lambda.min models give lower rmse in both ridge and lasso. On the basis of RMSE and Rsquared, the Lasso model with lambda.min is the best fit. Lowest RMSE and highest R-squared. The difference is RMSE and R-squared is not much for all the four models hence we can use lambda.1se if we want a less complex model depending on less ariables. This can improve our understanding of model with some compromise on goodness of fit.