Linear Regression

Jieqi Tu (jt3098)

3/26/2019

```r
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------- tidy
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.7
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
## -- Conflicts ---------------------------------------------------------------------- tidyverse_c
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library("leaps")
```

#Load and tidy data

```r
#read data
rawdata <- read.csv("kc_house_data.csv", header = TRUE)

#inspect the structure of data
str(rawdata)
```

```
## 'data.frame':    21613 obs. of  21 variables:
##  $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
##  $ date         : Factor w/ 372 levels "20140502T000000",..: 165 221 291 221 284 11 57 252 340 306 .
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms     : int  3 3 2 4 3 4 3 3 3 3 ...
##  $ bathrooms    : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
##  $ sqft_living  : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890 ...
##  $ sqft_lot     : int  5650 7242 10000 5000 8080 101930 6819 9711 7470 6560 ...
##  $ floors       : num  1 2 1 1 1 1 2 1 1 2 ...
##  $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ condition    : int  3 3 3 5 3 3 3 3 3 3 ...
##  $ grade        : int  7 7 6 7 8 11 7 7 7 7 ...
##  $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ...
##  $ sqft_basement: int  0 400 0 910 0 1530 0 0 730 0 ...
##  $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003 ...
##  $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198 98146 98038 ...
```

```
##  $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390 ...
##  $ sqft_lot15   : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570 ...
```

`id`, `date`, `zipcode`, `lat`, `long` can be removed from the dataframe.
The cleaned dataset to be used in this project include one response variable `price` and additional 15 variables. `view`, `sqft_basement` and `yr_renovated` are continuous or integer variables in the original dataset. For the purpose of easy interpretation in later modelling, we convert these variables to be binary. Then these variables indicate whether the house has been viewed by potential buyers, whether the house has basement or not and whether the house has been renovated or not, respectively.

```r
#clean the rawdata; create a tidied dataset for analysis and modelling.
#subset data: only those with view >0.
housing =
  rawdata %>%
  select(-id, -date, -zipcode, -lat, -long) %>%
  filter(view > 0, bedrooms <30) %>%
  mutate(basement = ifelse(sqft_basement == 0, 0, 1),
         renovated = ifelse(yr_renovated == 0, 0, 1)) %>%
  filter(basement > 0) %>%
  select(-sqft_basement, -yr_renovated, -view, - sqft_living, -sqft_lot, -basement)


# create training data and testing data.
rowTrain <- createDataPartition(y = housing$price,
                                p=0.8, list = FALSE)
```

```r
# vector of response
y <- housing$price[rowTrain]
x <- model.matrix(price~.,housing)[rowTrain,-1]
```

Model Building

Linear model

```r
# build a least-square linear model
ctrl1 = trainControl(method = "repeatedcv", number = 10)
set.seed(1)

# stepwise elimination to select variables
lm_fit = lm(price~., data = housing)
step(lm_fit, direction = 'backward')
```

```
## Start:  AIC=37297.96
## price ~ bedrooms + bathrooms + floors + waterfront + condition +
##     grade + sqft_above + yr_built + sqft_living15 + sqft_lot15 +
##     renovated
##
##                 Df  Sum of Sq        RSS   AIC
## <none>                         2.3043e+14 37298
## - renovated      1 5.4697e+11 2.3098e+14 37299
## - bedrooms       1 1.4271e+12 2.3186e+14 37305
## - condition      1 2.9548e+12 2.3339e+14 37314
## - bathrooms      1 3.0299e+12 2.3346e+14 37315
```

```
## - floors         1 3.3559e+12 2.3379e+14 37317
## - sqft_living15  1 4.2095e+12 2.3464e+14 37322
## - sqft_lot15     1 4.3064e+12 2.3474e+14 37323
## - yr_built       1 1.3649e+13 2.4408e+14 37379
## - grade          1 2.4412e+13 2.5485e+14 37441
## - sqft_above     1 3.4475e+13 2.6491e+14 37497
## - waterfront     1 3.8522e+13 2.6896e+14 37519
```

```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + floors + waterfront +
##     condition + grade + sqft_above + yr_built + sqft_living15 +
##     sqft_lot15 + renovated, data = housing)
##
## Coefficients:
##   (Intercept)       bedrooms      bathrooms         floors     waterfront
##     7.594e+06     -3.907e+04      8.742e+04     -1.194e+05      6.675e+05
##     condition          grade     sqft_above       yr_built  sqft_living15
##     7.120e+04      1.736e+05      3.212e+02     -4.681e+03      1.000e+02
##    sqft_lot15      renovated
##    -2.143e+00      7.112e+04
```

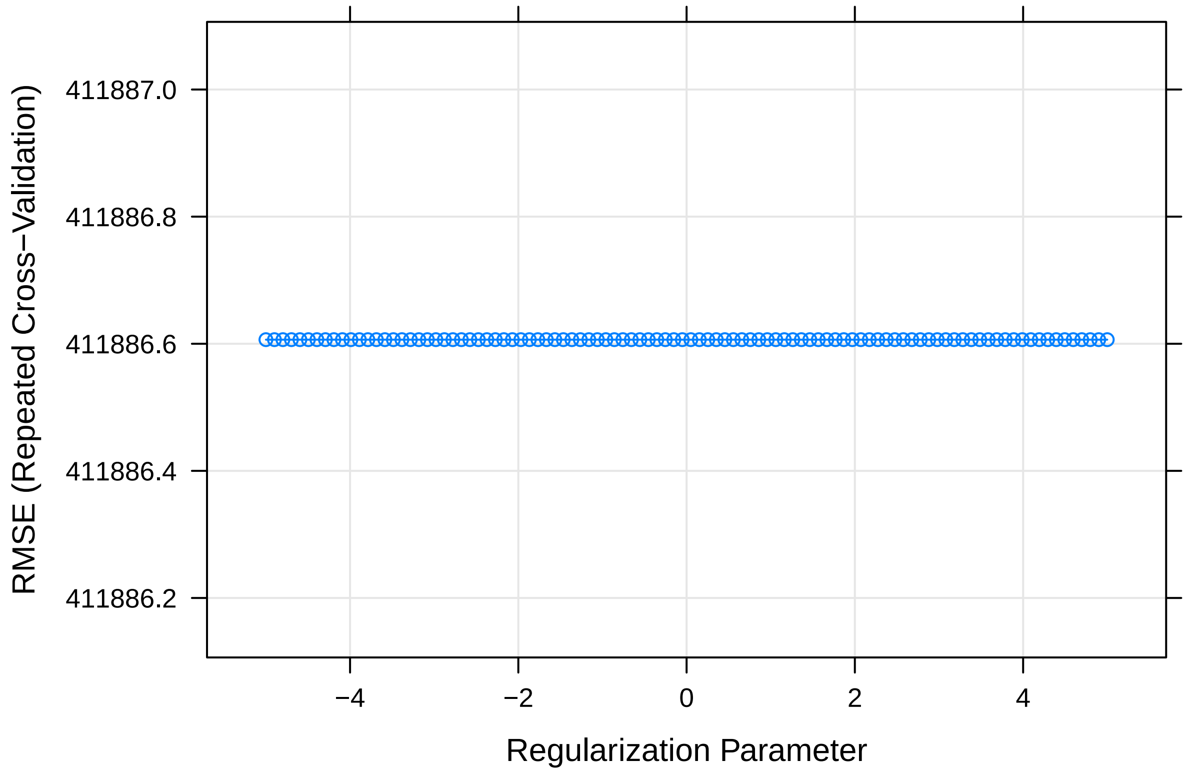
```
model_ls = train(x, y,
                 method = "lm",
                 preProcess = c("center", "scale"),
                 trControl = ctrl1)

#obtain coefficients
coef_ls = model_ls$finalModel$coefficients %>% as.data.frame(); coef_ls
```

```
##                        .
## (Intercept)   1012794.52
## bedrooms       -38450.54
## bathrooms       95056.56
## floors         -67442.90
## waterfront     164874.24
## condition       47471.64
## grade          232605.76
## sqft_above     329860.51
## yr_built      -136518.45
## sqft_living15   63046.00
## sqft_lot15     -59753.05
## renovated       29810.81
```

```
set.seed(1)
# fit a ridge model using caret package
ridge.fit = train(x, y,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = 0,
                                         lambda = exp(seq(-5, 5, length = 100))),
                  preProcess = c("center", "scale"),
                  trControl = ctrl1)

# plot the RMSE by log(lambda)
plot(ridge.fit, xTrans = function(x) log(x))
```




```r
# find the optimal lambda
ridge.fit$bestTune
```

```
##     alpha   lambda
## 100     0 148.4132
```

```r
# obtain the coefficients of ridge model
coef_ridge = coef(ridge.fit$finalModel, ridge.fit$bestTune$lambda); coef_ridge
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)    1012794.52
## bedrooms        -26090.91
## bathrooms       101676.82
## floors          -41423.07
## waterfront      157641.12
## condition        47965.98
## grade           215393.59
## sqft_above      280409.43
## yr_built       -120489.68
## sqft_living15    80850.45
## sqft_lot15      -52284.67
## renovated        34238.54
```

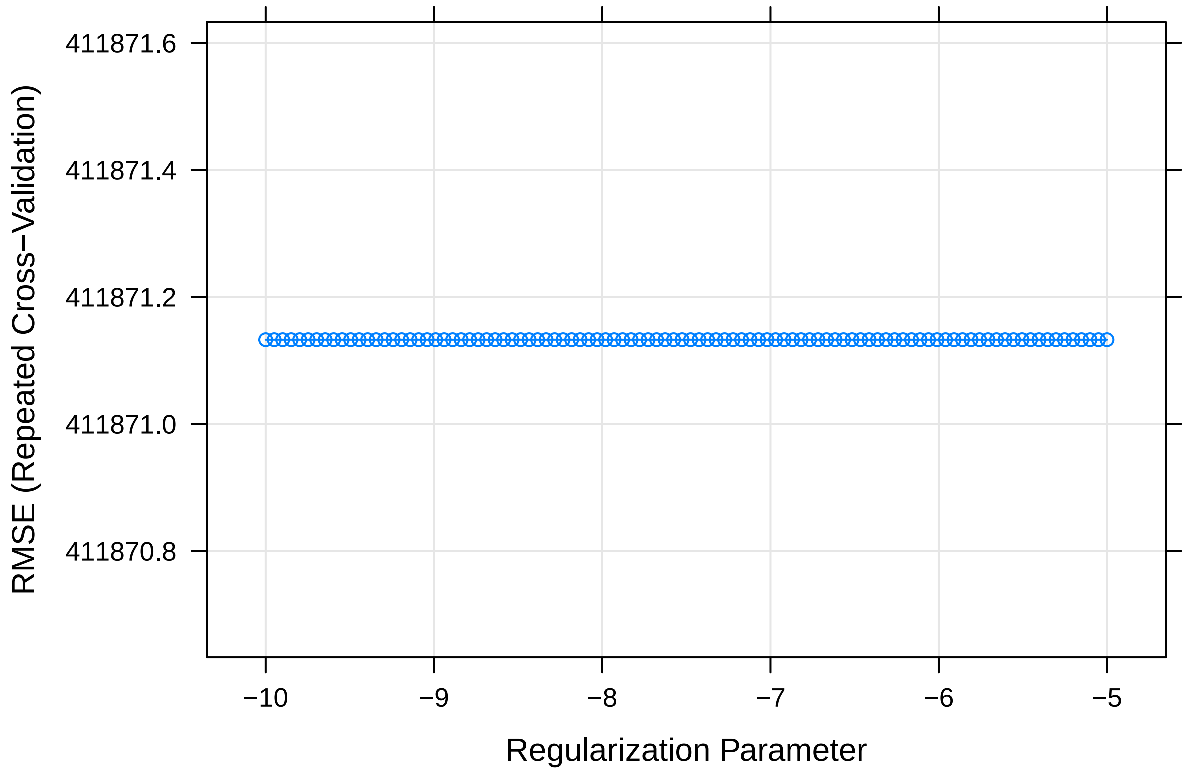
```r
summary(ridge.fit)
```

```
##             Length Class      Mode
## a0           100   -none-     numeric
```

```
## beta        1100   dgCMatrix  S4
## df           100   -none-     numeric
## dim            2   -none-     numeric
## lambda       100   -none-     numeric
## dev.ratio    100   -none-     numeric
## nulldev        1   -none-     numeric
## npasses        1   -none-     numeric
## jerr           1   -none-     numeric
## offset         1   -none-     logical
## call           5   -none-     call
## nobs           1   -none-     numeric
## lambdaOpt      1   -none-     numeric
## xNames        11   -none-     character
## problemType    1   -none-     character
## tuneValue      2   data.frame list
## obsLevels      1   -none-     logical
## param          0   -none-     list
```

```r
# fit a lasso model using caret
set.seed(1)
lasso.fit = train(x, y,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = 1,
                                         lambda = exp(seq(-10, -5, length = 100))),
                  preProcess = c("center", "scale"),
                  trControl = ctrl1)

# plot the RMSE by log(lambda)
plot(lasso.fit, xTrans = function(x) log(x))
```




```r
# obtain the optimal lambda
lasso.fit$bestTune
```

```
##     alpha      lambda
## 100     1 0.006737947
```

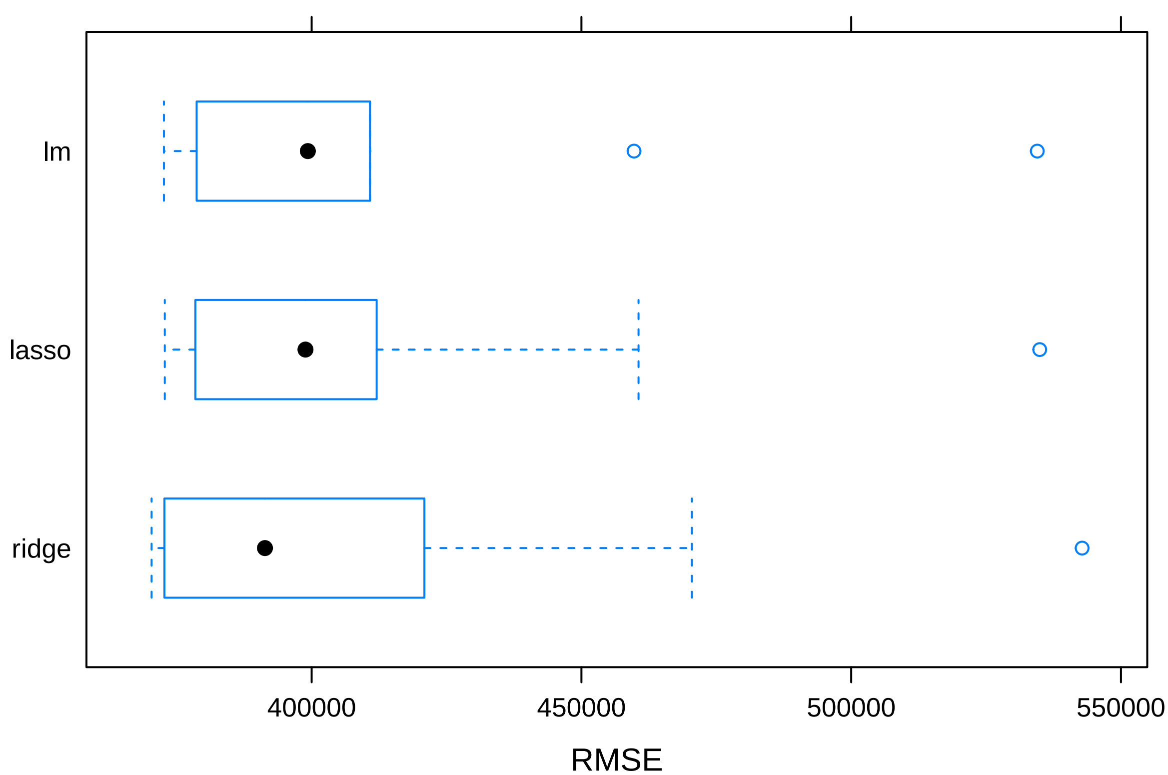
```r
# check the coefficients for each predictors
coef(lasso.fit$finalModel, lasso.fit$bestTune$lambda)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)   1012794.52
## bedrooms       -35677.09
## bathrooms       92535.01
## floors         -64501.67
## waterfront     164392.82
## condition       46778.94
## grade          232262.10
## sqft_above     327582.27
## yr_built      -135342.01
## sqft_living15   62855.33
## sqft_lot15     -58184.79
## renovated       29265.50
```

```r
set.seed(1)
resamp = resamples(list(lasso = lasso.fit, ridge = ridge.fit, lm = model_ls))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lasso, ridge, lm
## Number of resamples: 10
##
## MAE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso 250921.9 267868.9 276437.4 281568.8 295878.9 320091.8    0
## ridge 247093.0 266801.1 272754.0 277753.8 285230.9 317989.7    0
## lm    251699.7 267883.2 276566.1 282000.6 296928.8 320875.8    0
##
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso 372779.6 378794.6 398851.2 411871.1 411402.5 534939.3    0
## ridge 370330.3 374354.8 391341.8 411886.6 417515.5 542801.8    0
## lm    372621.6 379129.2 399292.1 411890.6 410734.6 534485.2    0
##
## Rsquared
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lasso 0.5531516 0.6627038 0.6750201 0.6733862 0.6853691 0.7676324    0
## ridge 0.5641182 0.6608677 0.6712183 0.6739028 0.6907061 0.7653062    0
## lm    0.5519036 0.6619972 0.6754908 0.6734033 0.6858197 0.7686745    0
```

```r
bwplot(resamp, metric = "RMSE")
```



Test Performance

```r
# linear regression
mean((predict(model_ls, housing[-rowTrain,]) - housing[-rowTrain,]$price)^2)
```

```
## [1] 145544753294
```

```r
# ridge
mean((predict(ridge.fit, housing[-rowTrain,]) - housing[-rowTrain,]$price)^2)
```

```
## [1] 1.45152e+11
```

```r
# lasso
mean((predict(lasso.fit, housing[-rowTrain,]) - housing[-rowTrain,]$price)^2)
```

```
## [1] 145402121651
```