Regularized Regression

In linear regression, regularization is a process of making the model more regular or simpler by shrinking the model coefficient to be closer to zero or absolute, ultimately to address over fitting.

The flow to conduct regularized regression:

1. Split Data: train - validate - test
2. Correlation plot on training data
3. Fit the model on training data
4. Choose the best lambda from the validation test
5. Evaluate the model on test data using MAE, MAPE, and RMSE

We need to load the data set first

```
library(readr)
boston <- read_csv("C:/Users/SU - NB0130/Desktop/bootcamp data science/week 9/day 20/dibimbing-m
aterials-main/boston.csv")
```

```
## Rows: 506 Columns: 14
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (14): crim, zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

The data is about predicting housing price (medv) in Boston city, features:

- Criminal rate (crim)
- Residential land zoned proportion (zn)
- Non-retail business acres proportion (indus)
- Is bounds with river (chas)
- Nitrogen oxides concentration (nox)
- Number rooms average (rm)
- Owner age proportion (age)
- Weighted distance to cities (dis)
- Accessibility index (rad)
- Tax rate (tax)

- Pupil-teacher ratio (ptratio)
- Black proportion (black)
- Percent lower status (lstat)

1. Split the data into train, validate, and test data

Pre-training and testing with comparison 80:20

```
library(caTools)
set.seed(123)
sample <- sample.split(boston$medv, SplitRatio = .80)
pre_train <- subset(boston, sample == TRUE)
sample_train <- sample.split(pre_train$medv, SplitRatio = .80)
```

After that, split pre-train data into train and validation (80:20)

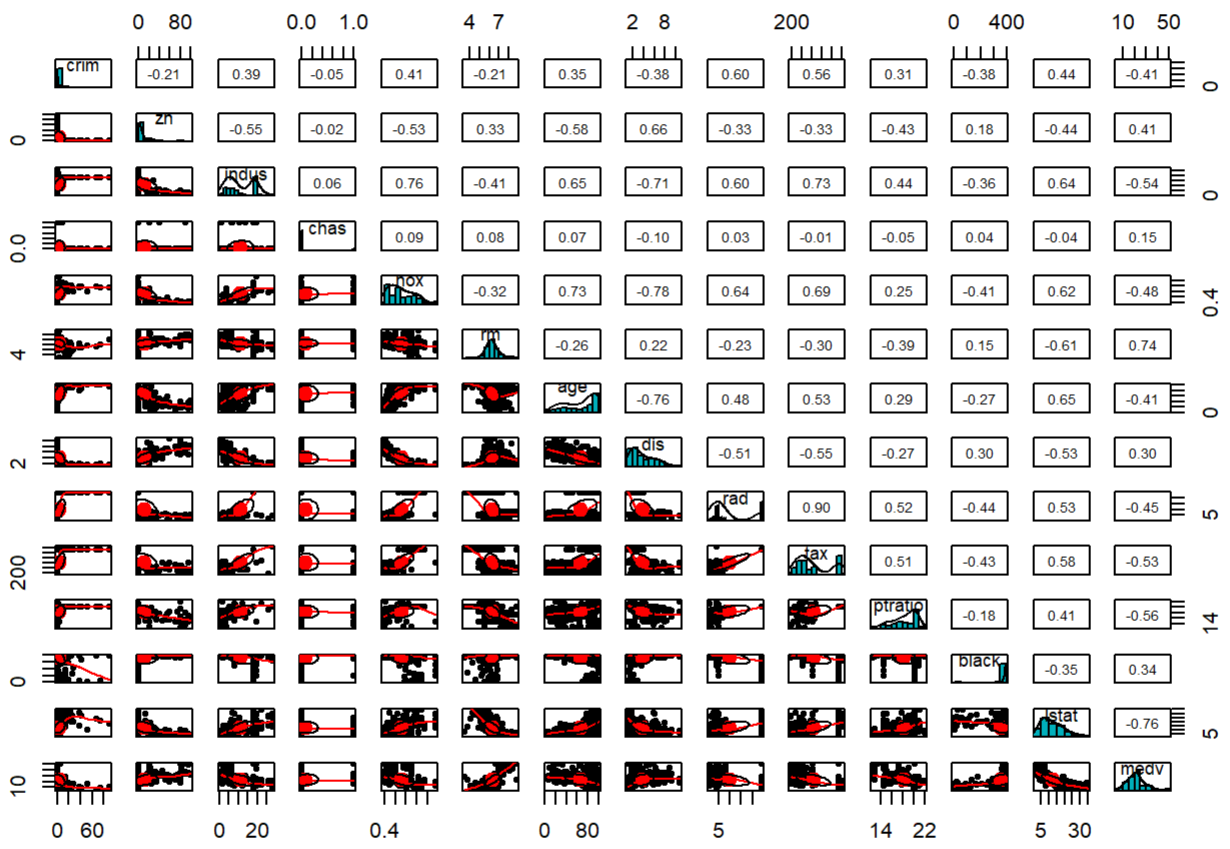
```
train <- subset(pre_train, sample_train == TRUE)
validation <- subset(pre_train, sample_train == FALSE)
```

Test data

```
test <- subset(boston, sample == FALSE)
```

2. Draw a pair plot to obtain a feature correlation and prevent multicolinearity

```
library(psych)
pairs.panels(train,
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = TRUE, # show density plots
             ellipses = TRUE) # show correlation ellipses
```



With 0.8 as the threshold, the correlated features are rad and tax. Therefore, we need to drop one of these columns.

The absolute value of correlation for rad and medv is 0.44

The absolute value of correlation for tax and medv is 0.53

From these 2 values, tax has highest correlation towards medv. Therefore, we keep tax column for our model and drop rad column.

Drop correlated column

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
drop_cols <- ("rad")

train <- train %>% select(-drop_cols)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(drop_cols)` instead of `drop_cols` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```r
validation <-  validation %>% select(-drop_cols)
test <- test %>% select(-drop_cols)
```

3. Train multiple models with different lambdas

We need to do pre processing to ensure we handle categorical features (transform categorical into numeric column).

```r
x <- model.matrix(medv ~ ., train)[,-1]
y <-  train$medv
```

Next, we can use two methods of regularized in linear regressions: Ridge and LASSO regressions

Ridge

```r
library(glmnet)
```

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

```
## Loaded glmnet 4.1-2
```

Ridge regression with lambda = 0.01

```r
ridge_reg_pointzeroone <- glmnet(x, y, alpha = 0, lambda = 0.01)
coef(ridge_reg_pointzeroone)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  2.807966e+01
## crim        -7.972347e-02
## zn           3.796482e-02
## indus       -4.106178e-02
## chas         2.893259e+00
## nox         -1.602703e+01
## rm           4.517287e+00
## age          5.679736e-03
## dis         -1.314253e+00
## tax         -2.421124e-04
## ptratio     -9.031044e-01
## black        6.572154e-03
## lstat       -4.779743e-01
```

Ridge regression with lambda = 0.1

```
ridge_reg_pointone <- glmnet(x, y, alpha = 0, lambda = 0.1)
coef(ridge_reg_pointone)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  2.720583e+01
## crim        -7.865999e-02
## zn           3.682165e-02
## indus       -4.208117e-02
## chas         2.888898e+00
## nox         -1.513326e+01
## rm           4.524625e+00
## age          5.018603e-03
## dis         -1.260076e+00
## tax         -4.973179e-04
## ptratio     -8.931536e-01
## black        6.639672e-03
## lstat       -4.709285e-01
```

Ridge regression with lambda = 1

```
ridge_reg_one <- glmnet(x, y, alpha = 0, lambda = 1)
coef(ridge_reg_pointone)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  2.720583e+01
## crim        -7.865999e-02
## zn           3.682165e-02
## indus       -4.208117e-02
## chas         2.888898e+00
## nox         -1.513326e+01
## rm           4.524625e+00
## age          5.018603e-03
## dis         -1.260076e+00
## tax         -4.973179e-04
## ptratio     -8.931536e-01
## black        6.639672e-03
## lstat       -4.709285e-01
```

Ridge regression with lambda = 10

```
ridge_reg_ten <- glmnet(x, y, alpha = 0, lambda = 10)
coef(ridge_reg_ten)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept) 21.798954932
## crim        -0.061446034
## zn           0.020349479
## indus       -0.081406215
## chas         2.105932529
## nox         -4.343751697
## rm           2.889836712
## age         -0.008076179
## dis         -0.190031642
## tax         -0.003586798
## ptratio     -0.571970624
## black        0.005615501
## lstat       -0.242577448
```

LASSO

LASSO regression with lambda = 0.01

```
lasso_reg_pointzeroone <- glmnet(x, y, alpha = 1, lambda = 0.01)
coef(lasso_reg_pointzeroone)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  2.782960e+01
## crim        -7.879101e-02
## zn           3.673332e-02
## indus       -3.849552e-02
## chas         2.864983e+00
## nox         -1.574647e+01
## rm           4.531757e+00
## age          4.411184e-03
## dis         -1.294476e+00
## tax         -2.439776e-04
## ptratio     -9.039250e-01
## black        6.556538e-03
## lstat       -4.764532e-01
```

LASSO regression with lambda = 0.1

```
lasso_reg_pointone <- glmnet(x, y, alpha = 1, lambda = 0.1)
coef(lasso_reg_pointone)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  2.472929e+01
## crim        -6.891240e-02
## zn           2.563768e-02
## indus       -1.728300e-02
## chas         2.590973e+00
## nox         -1.267687e+01
## rm           4.620427e+00
## age          .
## dis         -1.022117e+00
## tax         -5.088209e-04
## ptratio     -9.019518e-01
## black        6.368681e-03
## lstat       -4.677220e-01
```

LASSO regression with lambda = 1

```
lasso_reg_one <- glmnet(x, y, alpha = 1, lambda = 1)
coef(lasso_reg_pointone)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  2.472929e+01
## crim        -6.891240e-02
## zn           2.563768e-02
## indus       -1.728300e-02
## chas         2.590973e+00
## nox         -1.267687e+01
## rm           4.620427e+00
## age          .           
## dis         -1.022117e+00
## tax         -5.088209e-04
## ptratio     -9.019518e-01
## black        6.368681e-03
## lstat       -4.677220e-01
```

LASSO regression with lambda = 10

```
lasso_reg_ten <- glmnet(x, y, alpha = 1, lambda = 10)
coef(lasso_reg_ten)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                   s0
## (Intercept) 22.53775
## crim         0.00000
## zn           .      
## indus        .      
## chas         .      
## nox          .      
## rm           .      
## age          .      
## dis          .      
## tax          .      
## ptratio      .      
## black        .      
## lstat        .      
```

4. Choose the best lambda from the validation set

```
x_validation <- model.matrix(medv ~., validation)[,-1]
y_validation <- validation$medv
```

Ridge

RMSE Ridge for lambda = 0.01

```
RMSE_ridge_pointzeroone <- sqrt(mean((y_validation - predict(ridge_reg_pointzeroone, x_validatio
n))^2))
RMSE_ridge_pointzeroone
```

```
## [1] 4.3464
```

RMSE Ridge for lambda = 0.1

```
RMSE_ridge_pointone <- sqrt(mean((y_validation - predict(ridge_reg_pointone, x_validation))^2))
RMSE_ridge_pointone
```

```
## [1] 4.349494
```

RMSE Ridge for lambda = 1

```
RMSE_ridge_one <- sqrt(mean((y_validation - predict(ridge_reg_one, x_validation))^2))
RMSE_ridge_one
```

```
## [1] 4.422032
```

RMSE Ridge for lambda = 10

```
RMSE_ridge_ten <- sqrt(mean((y_validation - predict(ridge_reg_ten, x_validation))^2))
RMSE_ridge_ten
```

```
## [1] 5.342122
```

To have the best model's coefficients, we need to find the smallest RMSE on validation data. Hence, the best lambda is 0.01

```
coef(ridge_reg_pointzeroone)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  2.807966e+01
## crim        -7.972347e-02
## zn           3.796482e-02
## indus       -4.106178e-02
## chas         2.893259e+00
## nox         -1.602703e+01
## rm           4.517287e+00
## age          5.679736e-03
## dis         -1.314253e+00
## tax         -2.421124e-04
## ptratio     -9.031044e-01
## black        6.572154e-03
## lstat       -4.779743e-01
```

This implies that the best model for Ridge regression is:

medv = 2.807966e+01 -7.972347e-02 crim + 3.796482e-02 zn -4.106178e-02 indus + 2.893259e+00 chas -1.602703e+01 nox + 4.517287e+00 rm + 5.679736e-03 age -1.314253e+00 dis -2.421124e-04 tax -9.031044e-01 ptratio + 6.572154e-03 black -4.779743e-01 lstat

For example, an increase of 1 point in zn, while the other features are kept fixed, is associated with an increase of 3.796482e-02 point in medv.

LASSO

RMSE LASSO for lambda = 0.01

```
RMSE_lasso_pointzeroone <- sqrt(mean((y_validation - predict(lasso_reg_pointzeroone, x_validatio
n))^2))
RMSE_lasso_pointzeroone
```

```
## [1] 4.340783
```

RMSE LASSO for lambda = 0.1

```
RMSE_lasso_pointone <- sqrt(mean((y_validation - predict(lasso_reg_pointone, x_validation))^2))
RMSE_lasso_pointone
```

```
## [1] 4.352728
```

RMSE LASSO for lambda = 1

```
RMSE_lasso_one <- sqrt(mean((y_validation - predict(lasso_reg_one, x_validation))^2))
RMSE_lasso_one
```

```
## [1] 4.937774
```

RMSE LASSO for lambda = 10

```
RMSE_lasso_ten <- sqrt(mean((y_validation - predict(lasso_reg_ten, x_validation))^2))
RMSE_lasso_ten
```

```
## [1] 9.371755
```

Best model’s coefficients of LASSO regression is when the lambda is 0.01 (its RMSE is the smallest)

```
coef(lasso_reg_pointzeroone)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept)  2.782960e+01
## crim        -7.879101e-02
## zn           3.673332e-02
## indus       -3.849552e-02
## chas         2.864983e+00
## nox         -1.574647e+01
## rm           4.531757e+00
## age          4.411184e-03
## dis         -1.294476e+00
## tax         -2.439776e-04
## ptratio     -9.039250e-01
## black        6.556538e-03
## lstat       -4.764532e-01
```

This implies that:

medv = 2.782960e+01 -7.879101e-02 crim + 3.673332e-02 zn -3.849552e-02 indus + 2.864983e+00 chas -1.574647e+01 nox + 4.531757e+00 rm + 4.411184e-03 age -1.294476e+00 dis -2.439776e-04 tax -9.039250e-01 ptratio + 6.556538e-03 black -4.764532e-01 lstat

For instance, an increase of 1 point in zn, while the other features are kept fixed, is associated with an increase of 3.673332e-02 point in medv.

5. Evaluate the best models on the test data

```
x_test <- model.matrix(medv ~ ., test)[,-1]
y_test <-  test$medv
```

Ridge

##RMSE

```
RMSE_ridge_best <- sqrt(mean((y_test - predict(ridge_reg_pointzeroone, x_test))^2))
RMSE_ridge_best
```

```
## [1] 6.820639
```

The standard deviation of prediction errors to the regression line is 6.820639

##MAE

```
MAE_ridge_best <- mean(abs(y_test-predict(ridge_reg_pointzeroone, x_test)))
MAE_ridge_best
```

```
## [1] 3.896186
```

Our prediction deviates from the actual data (the true housing price (medv)) as much as 3.896186

##MAPE

```
MAPE_ridge_best <- mean(abs((predict(ridge_reg_pointzeroone, x_test) - y_test))/y_test)
MAPE_ridge_best
```

```
## [1] 0.1710101
```

MAE value of 3.896186 is equivalent to 17.1% deviation relative to the true housing price

LASSO

##RMSE

```
RMSE_ridge_best <- sqrt(mean((y_test - predict(lasso_reg_pointzeroone, x_test))^2))
RMSE_ridge_best
```

```
## [1] 6.823445
```

The standard deviation of prediction errors to the regression line is 6.823445

##MAE

```
MAE_ridge_best <- mean(abs(y_test-predict(lasso_reg_pointzeroone, x_test)))
MAE_ridge_best
```

```
## [1] 3.888415
```

Our prediction deviates from the the true housing price as much as 3.888415

##MAPE

```
MAPE_ridge_best <- mean(abs((predict(lasso_reg_pointzeroone, x_test) - y_test))/y_test)
MAPE_ridge_best
```

```
## [1] 0.1707025
```

3.888415 in MAE is equivalent to 17.07% deviation relative to the true housing price