Homework Day 20

Author: Ahmad Ichsan Baihaqi

Email: ahmadichsanbaihaqi@gmail.com

In this task, we will use housing price in boston as the dataset. The objective of this task is to create linear regression model with regularization using Ridge and Lasso.

Load dataset

```
data = read.csv("boston.csv")

head(data)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

Determine data type of each feature

```
sapply(data, class)
```

```
##      crim        zn     indus      chas       nox        rm       age       dis
## "numeric" "numeric" "numeric" "integer" "numeric" "numeric" "numeric" "numeric"
##       rad       tax   ptratio     black     lstat      medv
## "integer" "integer" "numeric" "numeric" "numeric" "numeric"
```

Based on above observation, we can take insight: 1. There are 14 features in this dataset 2. Those features are: a. Criminal rate (crim) b. Residential land zoned proportion (zn) c. Non-retail business acres proportion (indus) d. Is bounds with river (chas) e. Nitrogen oxides concentration (nox) f. Number rooms average (rm) g. Owner age proportion (age) h. Weighted distance to cities (dis) i. Accessibility index (rad) l. Tax rate (tax) m. Pupil-teacher ratio (ptratio) n. Black proportion (black) o. Percent lower status (lstat) p. housing price (medv) 3. Our target feature is the housing price (medv) 4. No categorical feature, so we don't have to do encoding

Before we start, let's check if there is any missing value in this dataset.

Finding missing value

```
sapply(data, function(x) sum(is.na(x)))
```

```
##    crim      zn   indus    chas     nox      rm     age     dis     rad     tax
##       0       0       0       0       0       0       0       0       0       0
## ptratio   black   lstat    medv
##       0       0       0       0
```

Fortunately, there is no missing value in our dataset.

Splitting data

First thing we have to do is to split our dataset into three parts: a. 80% of original dataset will be devided into: 1. train: 80% from the 80% of the original dataset 2. validate: 20% from the 80% of the original dataset

b. 20% of original dataset will be devided into:

3. test: 100% from 20% of the original dataset

Note about validation set

The validation set is a set of data, separate from the training set, that is used to validate our model performance during training.

This validation process gives information that helps us tune the model's hyperparameters and configurations accordingly. It is like a critic telling us whether the training is moving in the right direction or not.

The model is trained on the training set, and, simultaneously, the model evaluation is performed on the validation set after every epoch.

The main idea of splitting the dataset into a validation set is to prevent our model from overfitting i.e., the model becomes really good at classifying the samples in the training set but cannot generalize and make accurate classifications on the data it has not seen before.

reference: https://www.v7labs.com/blog/train-validation-test-set

```
library(caTools)
```

```
# Split data into 3 parts
# train - validation - test
set.seed(123)

# split 80% (SAMPLE TRUE) : 20% (SAMPLE FALSE) from original dataset
sample = sample.split(data$medv, SplitRatio = .80)

# stored 80% (SAMPLE TRUE) of the dataset into one variable
# which later will be used as the total 100% of train and validation set
pre_train = subset(data, sample == TRUE)

# split 80% (SAMPLE TRAIN TRUE) : 20% (SAMPLE TRAIN FALSE) from the pre_train
sample_train = sample.split(pre_train$medv, SplitRatio = .80)
```

```r
# train-validation data

# stored 80% (SAMPLE TRAIN TRUE) of the dataset into one variable
train = subset(pre_train, sample_train == TRUE)

# stored 20% (SAMPLE TRAIN FALSE) of the dataset into one variable
validation = subset(pre_train, sample_train == FALSE)
```

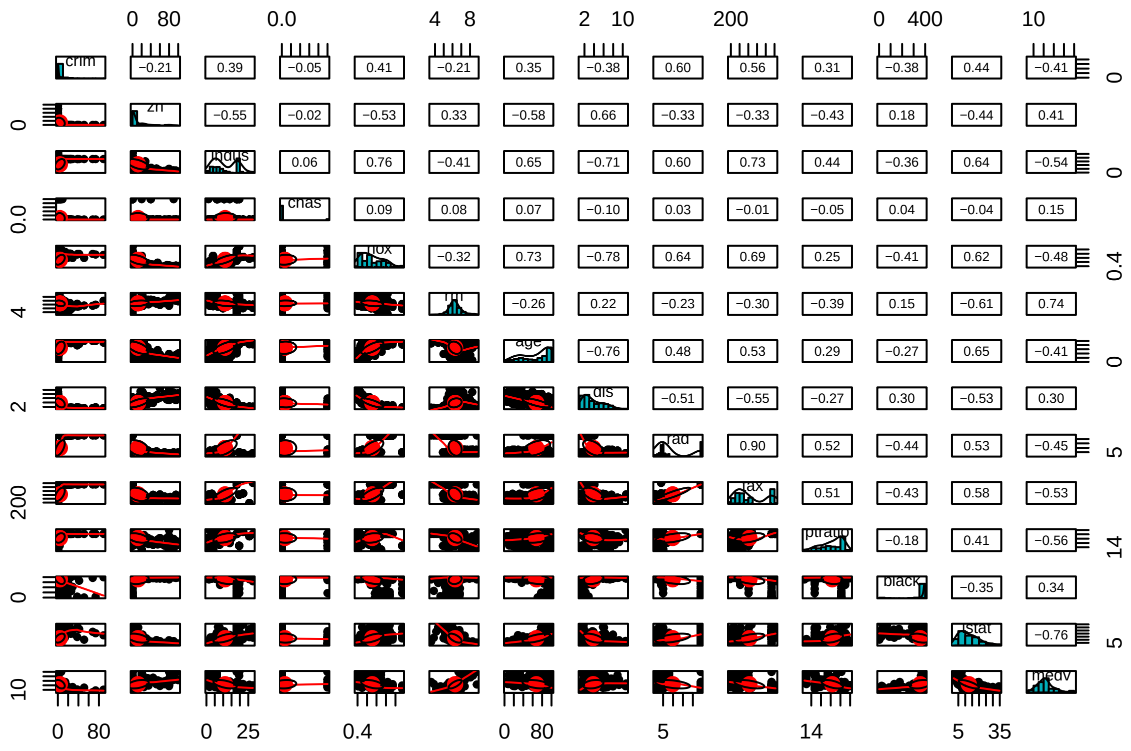
```r
# test data

# stored 20% (SAMPLE FALSE) of the dataset into one variable
test = subset(data, sample == FALSE)
```

Correlation features plot on training dataset

```r
library(psych)
```

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



Condition: 1. Threshold for high correlation is if the absolute(corr) >= 0.8

Insight: 1. The only feature correlation which satisfied our above condition is a correlation between rad and tax, which is 0.90. 2. To avoid multicollinearity, we need to drop one of this feature. 3. Between rad and medv, it has correlation coef -0.45 4. Between tax and medv, it has correlation coef -0.56 5. Feature that we should keep is a feature which has highest absolute(corr) with our target feature (medv). In this case, we should keep tax and drop rad feature.

Dropping feature

```
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 correlated columns
drop_cols = c('rad')
```

```r
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)
```

Fit models on training data

```r
# feature preprocessing
# to ensure we handle categorical features
x = model.matrix(medv ~ ., train)[,-1]
y = train$medv
```

```r
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-3
```

Ridge regression

```r
# fit multiple ridge regression with different lambda
# lambda = [0.01, 0.1, 1, 10]
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
```

```r
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
```

```r
ridge_reg_one = glmnet(x, y, alpha = 0, lambda = 1)
coef(ridge_reg_one)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept) 22.53185712
## crim        -0.07288298
## zn           0.02967177
## indus       -0.05282228
## chas         2.84365458
## nox         -9.91885353
## rm           4.44323173
## age          0.00101080
## dis         -0.90469612
## tax         -0.00195283
## ptratio     -0.82592673
## black        0.00688591
## lstat       -0.41785676
```

```r
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 regression

```
# fit multiple lasso regression with different lambda
# lambda = [0.01, 0.1, 1, 10]
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_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
```

```r
lasso_reg_one = glmnet(x, y, alpha = 1, lambda = 1)
coef(lasso_reg_one)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept) 13.987998322
## crim        -0.010183404
## zn           .
## indus        .
## chas         .
## nox          .
## rm           4.306536549
## age          .
## dis          .
## tax         -0.000775465
## ptratio     -0.710899295
## black        0.001632894
## lstat       -0.457861803
```

```r
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        .
```

Choose best lambda

The best lambda is when the RMSE value is the smallest among the other models.

```r
# Make predictions on the validation data
x_validation = model.matrix(medv ~., validation)[,-1]
y_validation = validation$medv
```

Choose best lambda with ridge regression

```
RMSE_ridge_pointzeroone = sqrt(mean((y_validation - predict(ridge_reg_pointzeroone, x_validation))^2))
RMSE_ridge_pointzeroone # 4.3464 => best
```

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

```
RMSE_ridge_pointone = sqrt(mean((y_validation - predict(ridge_reg_pointone, x_validation))^2))
RMSE_ridge_pointone # 4.349494
```

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

```
RMSE_ridge_one = sqrt(mean((y_validation - predict(ridge_reg_one, x_validation))^2))
RMSE_ridge_one # 4.422032
```

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

```
RMSE_ridge_ten = sqrt(mean((y_validation - predict(ridge_reg_ten, x_validation))^2))
RMSE_ridge_ten # 5.342122
```

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

Insight: 1. Best lambda using ridge regression is 0.01 with RMSE value 4.3464 2. The model based on the best lambda for predicting the housing price with ridge regression is:

medv = 27.82960 - 0.07879101 crim + 0.03673332 zn - 0.03849552 indus + 2.864983 chas - 0.1574647 nox + 4.531757 rm + 0.004411184 age - 1.294476 dis - 0.0002439776 tax - 0.9039250 ptratio + 0.006556538 black - 0.4764532 lstat

Model interpretation: 1. If the value for all predictors is zero, the mdev value is equal to the intercept of the model, which is 27.82960 2. An increase of 1 point in crim, while the other features are kept fixed, is associated with a decrease of 0.07879101 in medv. This is make sense since crim represent criminal rate. Higher criminal rate will decrease the housing price (medv) 3. An increase of 1 point in chas, while the other features are kept fixed, is associated with an increase of 2.864983 in medv. This indicate that house which bound with river will probably has higher housing price. 4. Based on interpretation on point 2 and 3, if we take a little conclusion based on some of the feature (crim and chas), a house will have higher price if the house located in a low criminal rate area and bound with a river.

Choose best lambda with lasso regression

```
RMSE_lasso_pointzeroone = sqrt(mean((y_validation - predict(lasso_reg_pointzeroone, x_validation))^2))
RMSE_lasso_pointzeroone # 4.340783 => best
```

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

```
RMSE_lasso_pointone = sqrt(mean((y_validation - predict(lasso_reg_pointone, x_validation))^2))
RMSE_lasso_pointone # 4.352728
```

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

```r
RMSE_lasso_one = sqrt(mean((y_validation - predict(lasso_reg_one, x_validation))^2))
RMSE_lasso_one # 4.937774
```

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

```r
RMSE_lasso_ten = sqrt(mean((y_validation - predict(lasso_reg_ten, x_validation))^2))
RMSE_lasso_ten # 9.371755
```

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

Insight: 1. Best lambda using lasso regression is 0.01 with RMSE value 4.340783 (pretty closed with ridge regression) 2. The model based on the best lambda for predicting the housing price with ridge regression is:

medv = 27.82960 - 0.07879101 crim + 0.03673332 zn - 0.03849552 indus + 2.864983 chas - 0.1574647 nox + 4.531757 rm + 0.004411184 age - 1.294476 dis - 0.0002439776 tax - 0.9039250 ptratio + 0.006556538 black - 0.4764532 lstat

Model interpretation: 1. despite of having different RMSE value between lasso and ridge with the same lambda (0.01), they both have same coeff for each feature. Thus, they both have same model with lambda 0.01.

Evaluate the best models on the test data

Create the test value

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

Evaluate with ridge

```r
# RMSE
RMSE_ridge_best = sqrt(mean((y_test - predict(ridge_reg_pointzeroone, x_test))^2))
RMSE_ridge_best
```

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

```r
# MAE
MAE_ridge_best = mean(abs(y_test-predict(ridge_reg_pointzeroone, x_test)))
MAE_ridge_best
```

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

```r
# MAPE
MAPE_ridge_best = mean(abs((predict(ridge_reg_pointzeroone, x_test) - y_test))/y_test)
MAPE_ridge_best
```

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

Interpretation: 1. Best result for RMSE with train data using ridge (`ridge_reg_pointzeroone`) is 4.3464. Meanwhile, when we are using the test dataset, the RMSE value is quite far, which is 6.82 (the distance is two points). Is this indicate that our model overfitting? After googling for a while, unfortunately I couldn't find a reference which declared the treshold for delta to be considered as overfit. The only reference I found is this discussion https://stats.stackexchange.com/questions/497050/how-big-a-difference-for-test-train-rmse-is-considered-as-overfit#comment919735_497050 which said that there are no treshold to be considered as overfit.

Evaluate with lasso

```r
# RMSE
RMSE_lasso_best = sqrt(mean((y_test - predict(lasso_reg_pointzeroone, x_test))^2))
RMSE_lasso_best
```

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

```r
# MAE
MAE_lasso_best = mean(abs(y_test-predict(lasso_reg_pointzeroone, x_test)))
MAE_lasso_best
```

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

```r
# MAPE
MAPE_lasso_best = mean(abs((predict(lasso_reg_pointzeroone, x_test) - y_test))/y_test)
MAPE_lasso_best
```

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