Lab1_assignment2

Jin Yan

2022-11-18

Question 1

Divide it into training and test data (60/40) and scale it appropriately. In the coming steps, assume that motor_UPDRS is normally distributed and is a function of the voice characteristics, and since the data are scaled, no intercept is needed in the modelling.

The code is given in Appendix.

Question 2

Compute a linear regression model from the training data, estimate training and test MSE and comment on which variables contribute significantly to the model.

Training MSE is 0.87854 Test MSE is 0.93945

When we use summary function we can find that some P values of parameters are marked with three stars. That means these parameters contribute most to the result. They are shown in the picture in Appendix.They are respectively Jitter.Abs Shimmer.APQ5 Shimmer.APQ11 NHR HNR DFA PPE

Question 3

Implementing 4 functions, which are "Loglikelihood function", " Ridge function", "RidgeOpt function", "DF function".

Four functions are given in Appendix.

We create the "Loglikelihood function" based on the following formula: $\ln p(\mathbf{y}|\mathbf{X};\theta) = -\frac{n}{2}\ln(2\pi\sigma_\epsilon^2) - \frac{1}{2\sigma_\epsilon^2}\sum_{i=1}^{n}(\theta^{\mathrm{T}}\mathbf{x}_i - y_i)^2$

If we want to create "Ridge function", we need to add $\lambda||\theta||_2^2$ to the last formula.

Question 4

By using function RidgeOpt, compute optimal θ parameters for λ= 1, λ = 100 and λ = 1000. Use the estimated parameters to predict the motor_UPDRS values for training and test data and report the training and test MSE values. Which penalty parameter is most appropriate among the selected ones? Compute and compare the degrees of freedom of these models and make appropriate conclusions.

λ	$Training\_MSE$	$Test\_MSE$	DF
1	0.8786	0.9350	14.86
100	0.8844	0.9323	10.90
1000	0.9211	0.9539	6.42

By comparison we found that as the value of the λ increases, the value of DF decreases. At the same time, in general the predicted results would be less and less accurate. We also noticed that the optimum λ is 100, which means in order to get the relatively less errors, we should not use the extreme values of λ.

Appendix

```
my_data = read.csv("parkinsons.csv")
set.seed(12345)
n = nrow(my_data)
id = sample(1:n, floor(n*0.6))
train = my_data[id,]
test = my_data[-id,]


scaler = preProcess(train)
trainS = predict(scaler, train)
testS = predict(scaler, test)


# train MSE
trainS_1 = trainS %>% select(motor_UPDRS, 7:22)# what should we do if name of one column contains #
ml = lm(motor_UPDRS~.,trainS_1)
summary(ml)
```

```
##
## Call:
## lm(formula = motor_UPDRS ~ ., data = trainS_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0255 -0.7363 -0.1087  0.7333  2.1960
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    6.575e-15  1.583e-02   0.000 1.000000
## Jitter...      1.869e-01  1.496e-01   1.250 0.211496
## Jitter.Abs.   -1.696e-01  4.081e-02  -4.156 3.32e-05 ***
## Jitter.RAP    -5.270e+00  1.884e+01  -0.280 0.779688
## Jitter.PPQ5   -7.457e-02  8.778e-02  -0.850 0.395659
## Jitter.DDP     5.250e+00  1.884e+01   0.279 0.780541
## Shimmer        5.924e-01  2.060e-01   2.876 0.004055 **
## Shimmer.dB.   -1.727e-01  1.393e-01  -1.239 0.215380
## Shimmer.APQ3   3.207e+01  7.717e+01   0.416 0.677738
## Shimmer.APQ5  -3.875e-01  1.138e-01  -3.405 0.000669 ***
## Shimmer.APQ11  3.055e-01  6.124e-02   4.989 6.37e-07 ***
## Shimmer.DDA   -3.239e+01  7.717e+01  -0.420 0.674739
## NHR           -1.854e-01  4.557e-02  -4.068 4.85e-05 ***
## HNR           -2.385e-01  3.640e-02  -6.553 6.45e-11 ***
## RPDE           4.068e-03  2.267e-02   0.179 0.857576
## DFA           -2.803e-01  2.014e-02 -13.919  < 2e-16 ***
## PPE            2.265e-01  3.289e-02   6.886 6.75e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.9396 on 3508 degrees of freedom
## Multiple R-squared:  0.1212, Adjusted R-squared:  0.1172
## F-statistic: 30.24 on 16 and 3508 DF,  p-value: < 2.2e-16
```

```
Preds = predict(ml)
MSE_training = mean((trainS_1$motor_UPDRS - Preds)^2)

# test MSE
testS_1 = testS %>% select(motor_UPDRS, 7:22)
Preds_2 = predict(ml,testS_1)


MSE_test = mean((testS_1$motor_UPDRS - Preds_2)^2)



# The third question.

#a
n <- nrow(trainS)
trainS_1 <- tibble(trainS_1)
one <- tibble(one = rep(1,n))

trainS_1_new <- trainS_1%>%
  select(-motor_UPDRS)%>%
  mutate(one, .)


likelihood <- function(xita,sigma){
  n <- nrow(trainS)
  y_hat <- difference <- as.matrix(trainS_1_new) %*% as.matrix(xita)
  difference <- trainS_1$motor_UPDRS - y_hat
  square_difference <- difference^2
  sum <- sum(square_difference)
  return((-n/2) * log(2 * pi * sigma^2 ) - (1 / (2 * sigma^2)) * sum)

}

#b
Ridge <- function(lambda, xita, sigma){
  -likelihood(xita,sigma) + lambda * sum(xita^2)
}


#c
RidgeOpt <- function(lambda){

  to_optimize <- function(x){
    x1 = x[1:17]
    x2 = x[18]
    return(Ridge(lambda, x1, x2))
  }
  optim(rep(1, times = 18), fn = to_optimize, method = "BFGS")
}
```

```r
#d
DF <- function(lambda){
  X <- as.matrix(trainS_1_new)
  I <- diag(ncol(trainS_1))
  element_I <- diag(X %*% solve((t(X) %*% X + lambda * I)) %*% t(X))
  df <- sum(element_I)
  return(df)
}
df_1 <- DF(1)
df_100 <- DF(100)
df_1000 <- DF(1000)

#e

#when lambda equals 1, theta
theta_sigma_1 <- RidgeOpt(1)
theta_1 <- theta_sigma_1$par[1:17]

##MSE for training data
y_hat_tr_1 <- as.matrix(trainS_1_new) %*% as.matrix(theta_1)
MSE_train_1 <-mean((y_hat_tr_1 - trainS_1$motor_UPDRS)^2)

##MSE for test data
n_2 <- nrow(testS)
testS_1 <- tibble(testS_1)
one_2 <- tibble(one = rep(1,n_2))

testS_1_new <- testS_1%>%
  select(-motor_UPDRS)%>%
  mutate(one_2, .)

y_hat_test_1 <- as.matrix(testS_1_new) %*% as.matrix(theta_1)
MSE_test_1 <- mean((y_hat_test_1 - testS_1$motor_UPDRS)^2)





#when lambda equals 100, theta
theta_sigma_2 <- RidgeOpt(100)
theta_2 <- theta_sigma_2$par[1:17]


## MSE for training data

y_hat_tr_2 <- as.matrix(trainS_1_new) %*% as.matrix(theta_2)
MSE_train_2 <-mean((y_hat_tr_2 - trainS_1$motor_UPDRS)^2)

## MSE for test data
y_hat_test_2 <- as.matrix(testS_1_new) %*% as.matrix(theta_2)
MSE_test_2 <- mean((y_hat_test_2 - testS_1$motor_UPDRS)^2)
```

```r
#when lambda equals 1000, theta
theta_sigma_3 <- RidgeOpt(1000)
theta_3 <- theta_sigma_3$par[1:17]

## MSE for training data

y_hat_tr_3 <- as.matrix(trainS_1_new) %*% as.matrix(theta_3)
MSE_train_3 <-mean((y_hat_tr_3 - trainS_1$motor_UPDRS)^2)

## MSE for test data
y_hat_test_3 <- as.matrix(testS_1_new) %*% as.matrix(theta_3)
MSE_test_3 <- mean((y_hat_test_3 - testS_1$motor_UPDRS)^2)
```