lab1-asn2

Lepeng Zhang

2023-11-13

Assignment 2. Linear regression and ridge regression

1

```
rawdata <- read.csv("parkinsons.csv")
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
```

```
rawdata <- rawdata %>% select(-c(1:4,6))
n <- nrow(rawdata)
set.seed(12345)
id <- sample(1:n,floor(n*0.6))
train <- rawdata[id,]
test <- rawdata[-id,]

library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
param <- preProcess(train)
train_scaled <- predict(param,train)
test_scaled <- predict(param,test)
```

2

```r
mse <- function(true_value, predict_value){
  mean((true_value - predict_value)^2)
}

model <- lm(motor_UPDRS ~ ., data = train_scaled)

train_mse <- mse(train_scaled$motor_UPDRS, predict(model, train_scaled))
cat(paste0("Training MSE is: ",train_mse,".\n"))
```

```
## Training MSE is: 0.878543102826275.
```

```r
test_mse <- mse(test_scaled$motor_UPDRS, predict(model, test_scaled))
cat(paste0("Test MSE is: ",test_mse,".\n"))
```

```
## Test MSE is: 0.935447712156712.
```

```r
summary(model)
```

```
##
## Call:
## lm(formula = motor_UPDRS ~ ., data = train_scaled)
##
## 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
```

The significant contributors are Jitter.Abs., Shimmer.APQ5, Shimmer.APQ11, NHR, HNR, DFA, and PPE, which are marked $***$ in summary output.

3

```
Loglikelihood <- function(theta_vec, sigma){
  y <- train_scaled[,1]
  x <- as.matrix(train_scaled[,-1])
  n <- nrow(train_scaled)
  value <- -(n/2*log(sigma^2*2*pi)+sum((y-x%*%theta_vec)^2)/(2*sigma^2))
  return(value)
}

Ridge <- function(theta_vec, sigma, lambda){
  -Loglikelihood(theta_vec, sigma)+lambda*sum(theta_vec^2)
}

RidgeOpt <- function(lambda){
  init_theta <- rep(0, ncol(train_scaled)-1)
  init_sigma <- 0.9

  objective_function <- function(params) {
    theta_vec <- params[-length(params)]
    sigma <- params[length(params)]
    return(Ridge(theta_vec, sigma, lambda))
  }

  rlt <- optim(c(init_theta, init_sigma), fn = objective_function, method = "BFGS")

  optimal_theta_vec <- rlt$par[-length(rlt$par)]
  optimal_sigma <- rlt$par[length(rlt$par)]

  return(list(theta_vec = optimal_theta_vec, sigma = optimal_sigma))
}

DF <- function(lambda){
  x <- as.matrix(train_scaled[,-1])
  df <- sum(diag(x%*%solve(t(x)%*%x+lambda*diag(ncol(x)))%*%t(x)))
  return(df)
}
```

4

```
train_x <- as.matrix(train_scaled[,-1])
train_y <- train_scaled[,1]
test_x <- as.matrix(test_scaled[,-1])
test_y <- test_scaled[,1]
```

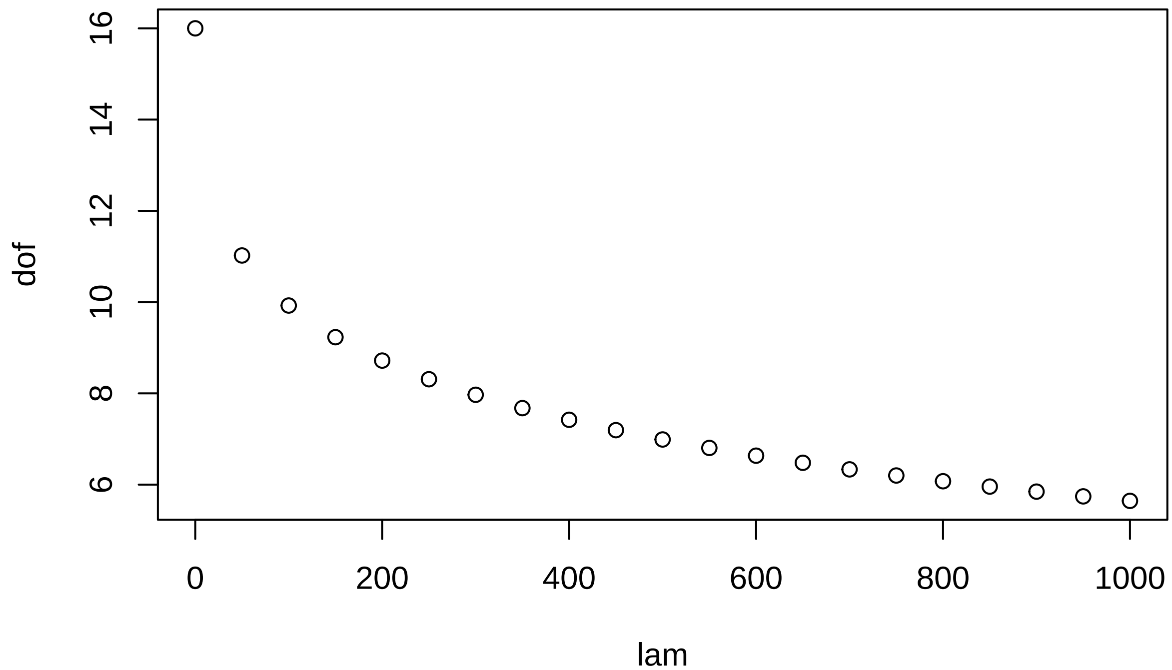
```
Lambda <- c(1,100,1000)
store_list <- list()
```

```r
for (i in 1:length(Lambda)){
  store_list$lambda[i] <- Lambda[i]
  optimal_theta_vec <- RidgeOpt(Lambda[i])$theta_vec
  store_list$train_MSE[i] <- mse(train_y, train_x%*%optimal_theta_vec)
  store_list$test_MSE[i] <- mse(test_y, test_x%*%optimal_theta_vec)
  store_list$DoF[i] <- DF(Lambda[i])
  store_list$sigma[i] <- RidgeOpt(Lambda[i])$sigma
}
store_df <- as.data.frame(store_list)
print(store_df)
```

```
##   lambda train_MSE  test_MSE       DoF     sigma
## 1      1 0.8786272 0.9350013 13.860736 0.9373538
## 2    100 0.8844134 0.9323346  9.924887 0.9404627
## 3   1000 0.9211180 0.9539465  5.643925 0.9597496
```

As the table shown, train_MSE increases slightly as λ increases, while test_MSE reaches minimum at $\lambda = 100$, which is regarded as the optimal penalty parameter among these three.
Larger λ results to smaller DoF. Therefore, effects of DoF on models can be viewed as the opposite of effects of λ on them. When λ increases (DoF decreases), model becomes simpler; when λ decreases (DoF increases), model becomes more complex. Since there is a trade-off of model complexity for the best model, there exists optimal values of λ and DoF.

```r
# Just for showing the inversely proportional relationship between lambda and DoF.
lam <- seq(0,1000,by=50)
dof <- c()
for (i in 1:length(lam)){
  dof[i] <- DF(lam[i])
}
plot(lam,dof)
```


16
14
12
10
8
6
dof
0
200
400
600
800
1000
lam