Assignment 2 in Machine Learning

Farid Musayev, Kristina Levina, and Wuhao Wang

17 November, 2021

Assignment 2

The data were scaled and divided into training and test data (60/40), as per the instructions. A linear regression model was computed from the training data without using an intercept. The estimated training and test MSEs are

The train MSE is 0.8731931

The test MSE is 0.9294911

In the model summary output, the most contributing parameters are marked by stars. The contribution of the terms is determined using the magnitude of p-values. The lower the p-values, the greater the contribution of the terms. In our case, the most contributing parameters (marked by three stars in the summary output) are Jitter(Abs), Shimmer:APQ5, Shimmer:APQ11, NHR, HNR, DFA, and PPE.

The obtained residual standard error: 0.9366 on 3509 degrees of freedom.

Next, the four function were constructed for optimisation (Loglikelihood, Ridge, RidgeOpt, and DF). These functions were constructed using the formula of loglikelihood, as per Eq. (3.20) from the course book (Andreas Lindholm (2021)):

$$ln(p(\mathbf{y}|\mathbf{X};\boldsymbol{\theta})) = -\frac{n}{2}ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\boldsymbol{\theta}^T\mathbf{x} - y_i)^2 \tag{1}$$

By using function RidgeOpt, optimal $\boldsymbol{\theta}$ parameters were calculated. The obtained parameters were then used to predict the motor_UPDRS values (target variable) for training and test data. Using different regularisation factors λ, the following results were obtained:

λ	MSE_test	MSE_tr	DF	σ
1	0.929	0.873	13.863	0.934
100	0.926	0.879	9.939	0.938
1000	0.948	0.916	5.643	0.957

As one can observe, the test value of the MSE is lowest for $\lambda = 100$. We can conclude that too large penalization does not contribute to better model. Optimum $\lambda = 100$. Regarding the degrees of freedom (Trevor Hastie (2017)), the larger the λ, the smaller the values of the diagonal elements of the hat matrix. The more degrees of freedom the model has, the better estimators should be obtained. However, we observe that for $\lambda = 1$, although the degrees of freedom are highest, the model performance is lower than when $\lambda = 100$. We can conclude that these two dependences (increase in degrees in freedom => better model and insufficient penalization => worse model) are balancing each other.

Appendix

```
################################################################################
# 1
################################################################################
data <- read.csv2("parkinsons.csv")

n <- dim(data)[1]

# Compute the number of columns
n_col <- length(unlist(strsplit(data[5,], ","))) #5 is just a random existing row

col_names <- strsplit(paste0("subject#,age,sex,test_time,motor_UPDRS,total_UPDRS,",
                      "Jitter(%),Jitter(Abs),Jitter:RAP,Jitter:PPQ5,Jitter:DDP,",
                      "Shimmer,Shimmer(dB),Shimmer:APQ3,Shimmer:APQ5,",
                      "Shimmer:APQ11,Shimmer:DDA,NHR,HNR,RPDE,DFA,PPE"), ",")

data <- lapply(data, function(i){
  i <- as.numeric(unlist(strsplit(i, ",")))
})

# Turn the obtained list into the matrix and to the data.frame.
data <- as.data.frame(matrix(unlist(data), nrow = n, ncol = n_col, byrow = TRUE))

names(data) <- unlist(col_names)

# Scale the data and use only the variables of interest in the upcoming analysis.
data <- scale(data[,c(5,7:22)])
n <- dim(data)[1]
data <- as.data.frame(data)

# Divide the data, as per the instructions
set.seed(12345)
id = sample(1:n, floor(n * 0.6))
train = data[id, ]
#print(dim(train)) #3525 entries

test = data[-id, ]

################################################################################
# 2
################################################################################

model <- lm(formula = motor_UPDRS ~ . - 1, data = train)

print(summary(model))
```

```
##
## Call:
## lm(formula = motor_UPDRS ~ . - 1, data = train)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -3.0119 -0.7270 -0.1018  0.7384  2.1959
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## `Jitter(%)`       0.181065   0.144249   1.255 0.209481
## `Jitter(Abs)`    -0.169830   0.040851  -4.157 3.30e-05 ***
## `Jitter:RAP`     -5.098809  18.184783  -0.280 0.779196
## `Jitter:PPQ5`    -0.071777   0.084701  -0.847 0.396816
## `Jitter:DDP`      5.079056  18.188164   0.279 0.780069
## Shimmer           0.590992   0.205286   2.879 0.004015 **
## `Shimmer(dB)`    -0.172860   0.139380  -1.240 0.214983
## `Shimmer:APQ3`   32.213852  77.012847   0.418 0.675759
## `Shimmer:APQ5`   -0.386846   0.113713  -3.402 0.000677 ***
## `Shimmer:APQ11`   0.310256   0.062270   4.982 6.58e-07 ***
## `Shimmer:DDA`   -32.529915  77.012630  -0.422 0.672761
## NHR              -0.186755   0.045741  -4.083 4.55e-05 ***
## HNR              -0.239777   0.036565  -6.558 6.27e-11 ***
## RPDE              0.003958   0.022611   0.175 0.861052
## DFA              -0.277038   0.019888 -13.930  < 2e-16 ***
## PPE               0.229006   0.033264   6.885 6.84e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9366 on 3509 degrees of freedom
## Multiple R-squared:  0.1212, Adjusted R-squared:  0.1172
## F-statistic: 30.25 on 16 and 3509 DF,  p-value: < 2.2e-16
```

```
pred_tr <- predict(object = model)
pred_test <- predict(object = model, newdata = test)

MSE_tr <- mean((train$motor_UPDRS - pred_tr)^2)
MSE_test <- mean((test$motor_UPDRS - pred_test)^2)

cat("The train MSE is ", MSE_tr, sep = "", "\n")
```

```
## The train MSE is 0.8731931
```

```
cat("The test MSE is ", MSE_test, sep = "", "\n")
```

```
## The test MSE is 0.9294911
```

```
################################################################################
# 3
################################################################################

# a

Loglikelihood <- function(par){ #par[1] is sigma and par[2:17] is theta vector
  X <- t(as.matrix(train[,-1]))
  n <- dim(train)[1]
  theta <- t(par[2:17])
```

```r
  result <- -n*log(2 * pi * par[1]^2)/2 -
    sum((theta %*% X - train$motor_UPDRS)^2)/(2 * par[1]^2)
  return(result)
}

Ridge <- function(par, lambda){
  result <- -Loglikelihood(par) + lambda * sum(par[2:17]^2)
  return(result)
}

RidgeOpt <- function(lambda){
  initial_sigma <- 0.9366
  initial_theta <- coef(model)
  result <- optim(par = c(initial_sigma, initial_theta), fn = Ridge,
                  lambda = lambda, method = "BFGS")
  return(result$par)
}

DF <- function(lambda){
  X <- as.matrix(train[,-1])
  # trace of the hat matrix
  result <- sum(diag(X %*% solve(t(X) %*% X + lambda * diag(16)) %*% t(X)))
  return(result)
}

##############################################################################
# 4
##############################################################################

lambda <- c(1, 100, 1000)
MSE_tr <- c()
MSE_test <- c()
DoF <- c()
sigma <- c()
for (l in lambda) {
  theta_lambda <- RidgeOpt(l)[2:17]
  X_tr <- t(as.matrix(train[,-1]))
  X_test <- t(as.matrix(test[,-1]))
  theta <- t(theta_lambda)
  pred_tr <- theta %*% X_tr
  pred_test <- theta %*% X_test
  MSE_tr <- c(MSE_tr, mean((train$motor_UPDRS - pred_tr)^2))
  MSE_test <- c(MSE_test, mean((test$motor_UPDRS - pred_test)^2))
  DoF <- c(DoF, DF(l))
  sigma <- c(sigma, RidgeOpt(l)[1])
}
df <- cbind.data.frame(lambda = lambda, MSE_test = MSE_test, MSE_tr = MSE_tr,
                       DoF = DoF, sigma = sigma)
print(round(df, digits = 3))
```

```
##   lambda MSE_test MSE_tr    DoF sigma
## 1      1    0.929  0.873 13.863 0.934
## 2    100    0.926  0.879  9.939 0.938
```

```
## 3   1000    0.948  0.916  5.643 0.957
```

Resources

Andreas Lindholm, et al. 2021. *Machine Learning. A First Course for Engineers and Scientists.* This material will be published by Cambridge University Press. This pre-publication version is free to view; download for personal use only.

Trevor Hastie, Jerome Friedma, Robert Tibshirani. 2017. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction.* Springer Series in Statistics.