Assignment 2

Martynas Lukosevicius, Alejo Perez Gomez, Shwetha Vandagadde Chandramouly

07/11/2020

Assignment 2

1.

Model:
$\hat{y} \sim N(w_0 + w^t X_i, \sigma^2)$ where $w \sim N(0, \frac{\sigma^2}{\lambda})$

- w - weights
- X - features
- λ - regularization penalty
- σ - variance

2.

Scaling data:

```
library(readr)
parkinsons <- read_csv("parkinsons.csv")
cleaned <- parkinsons[c(-1:-4, -6)]
parkinsons.scaled <- scale(cleaned)
set.seed(12345)
n <- dim(parkinsons.scaled)[1]
id=sample(1:n, floor(n*0.6))
train=parkinsons.scaled[id,]
test=parkinsons.scaled[-id,]
```

3.

As we will be optimizing σ and w, likelihood and prior should contain all σ, even if data is scaled (so it means that $\sigma \sim 1$):

$$log(posterior) = log(likelihood * prior) = log(likelihood) + log(prior)$$

a) loglikelihood:

$$log(p(D|w)) = -\frac{n}{2} log(2\pi\sigma^2) - \sum_{i=1}^{n} \frac{(y_i - w^T X_i)^2}{2\sigma^2}$$

```
loglikelihood <- function(w, sigma){
  n <- dim(train)[1]
  part1 <- -(n/ 2) * log(2 * pi*(sigma^2))
  sum <- 0
  for (i in 1:n) {
    y <- train[i, 1]
```

```
    x <- train[i, -1]
    temp <- (y - (t(w) %*% x))^2
    sum <- sum + as.vector(temp)
  }
  return(part1 - (sum/(2*(sigma)^2)))
}
```

b) Ridge part $\sim$ log prior, where $\tau = \frac{\sigma^2}{\lambda}$:

$$log(prior) = -\frac{1}{2}log(2\pi\tau) - \frac{(w)^2}{2\tau}$$

function returns $-log(posterior)$

```
ridge <- function(x, lambda){
  w <- x[1:16]
  sigma <- x[17]
  tau <- sigma^2 / lambda
  part1 <-  (-1/2) * log(2* pi * tau)
  part2 <- (w %*% w) / (2* tau)
  ridge <- part1 - part2
  return( - (loglikelihood(w,sigma) + ridge))
}
```

c) function to predict weights (w) and σ

```
ridgeOpt <- function(lambda){
  x <- rep(1,17)
  a <- optim(x ,ridge, method = "BFGS", lambda = lambda)
  w <- a$par[1:16]
  sigma <- a$par[17]
  return(a)
}
```

d) function to calculate degrees of freedom

```
DF <- function(lambda){
  m <- as.matrix(train[ ,-1])
  part1 <- t(m) %*% m + (lambda * diag(16))
  part2 <- m %*% solve(part1) %*% t(m)
  return(sum(diag(part2)))
}
```

4.

	MSE train	MSE test
lambda = 1	0.8872145	0.6342994
lambda = 100	0.8769148	0.6173668
lambda = 1000	0.9019595	0.6233459

$\lambda = 100$ is better than others because MSE for train set and for test set is lowest. MSE is good loss function because it comes from model's MLE.

5.

	AIC
lambda = 1	9610.013
lambda = 100	9562.313
lambda = 1000	9655.555

The optimal model is with lowest AIC score, in this case its a model with $\lambda = 100$. Hold out method requires to divide data into 3 parts, which wont allow to use all data for training, its not the case with AIC.

Appendix

```
##Assignment 2

library(readr)
parkinsons <- read_csv("parkinsons.csv")
cleaned <- parkinsons[c(-1:-4, -6)]
parkinsons.scaled <- scale(cleaned)
set.seed(12345)
n <- dim(parkinsons.scaled)[1]
id=sample(1:n, floor(n*0.6))
train=parkinsons.scaled[id,]
test=parkinsons.scaled[-id,]

loglikelihood <- function(w, sigma){
  n <- dim(train)[1]
  part1 <- -(n/ 2) * log(2 * pi*(sigma^2))
  sum <- 0
  for (i in 1:n) {
    y <- train[i, 1]
    x <- train[i, -1]
    temp <- (y - (t(w) %*% x))^2
    sum <- sum + as.vector(temp)
  }
  return(part1 - (sum/(2*(sigma)^2)))
}

ridge <- function(x, lambda){
  w <- x[1:16]
  sigma <- x[17]
  tau <- sigma^2 / lambda
  part1 <-  (-1/2) * log(2* pi * tau)
  part2 <- (w %*% w) / (2* tau)
  ridge <- part1 - part2
  return( - (loglikelihood(w,sigma) + ridge))
}

ridgeOpt <- function(lambda){
  x <- rep(1,17)
  a <- optim(x ,ridge, method = "BFGS", lambda = lambda)
  w <- a$par[1:16]
  sigma <- a$par[17]
```

```
  return(a)
}

DF <- function(lambda){
  m <- as.matrix(train[ ,-1])
  part1 <- t(m) %*% m + (lambda * diag(16))
  part2 <- m %*% solve(part1) %*% t(m)
  return(sum(diag(part2)))
}

task4 <- function(lambda){
  a <- ridgeOpt(lambda)
  n <- dim(train)[1]

  w <- a$par[1:16]

  predict <- train[ , -1] %*% w
  y <- train[ ,1]

  MSEtrain <- (1/n) * t((y-predict)) %*% (y-predict)
  predict <- test[ , -1] %*% w
  y <- test[ ,1]

  MSEtest <- (1/n) * t((y-predict)) %*% (y-predict)
  result <- list(w = w, sigma = a$par[17], testMSE = MSEtest, trainMSE = MSEtrain)
  return(result)

}

lambda1 <- task4(1)
lambda2 <- task4(100)
lambda3 <- task4(1000)
mm <- rbind(c(lambda1$trainMSE, lambda1$testMSE),
            c(lambda2$trainMSE, lambda2$testMSE),
            c(lambda3$trainMSE, lambda3$testMSE))
row.names(mm) <- c("lambda = 1", "lambda = 100", "lambda = 1000")
colnames(mm) <- c("MSE train", "MSE test")

DF <- function(lambda){
  m <- as.matrix(parkinsons.scaled[ ,-1])
  part1 <- t(m) %*% m + (lambda * diag(16))
  part2 <- m %*% solve(part1) %*% t(m)
  return(sum(diag(part2)))
}

AIC1 <- -2 * loglikelihood(lambda1$w, lambda1$sigma) + 2 * DF(1)
AIC2 <- -2 * loglikelihood(lambda2$w, lambda2$sigma) + 2 * DF(100)
AIC3 <- -2 * loglikelihood(lambda3$w, lambda3$sigma) + 2 * DF(1000)
#print(paste("AIC1:", AIC1, "AIC2:", AIC2, "AIC3:", AIC3))

AIC <- c(AIC1 , AIC2 , AIC3)
names(AIC) <- c("lambda = 1", "lambda = 100", "lambda = 1000")
knitr::kable(as.matrix(AIC, nrow= 1), row.names = TRUE,  col.names = c("AIC") )
res <- which.min(AIC)
```

```
res <- ifelse(res == 1, "1", ifelse(res == 2, 100, 1000))
```