A-5.R

pradyuth

Sat Mar 31 19:15:49 2018

```
#Name : Pradyuth Vangur

#Problem 1: Solving a set of equations using solve rather than inverting
#Generating 3X4 matrix
X <- matrix(rnorm(12), nrow = 3, ncol = 4)

#Generating 4X1 matrix
y <- matrix(rnorm(4), nrow = 4, ncol = 1)

#Solving using solve
solve(X %*% t(X), X %*% y)
```

```
##           [,1]
## [1,] 0.5583044
## [2,] 0.7072937
## [3,] 1.0243259
```

```
#Problem 2:
#Solving for this result isn't possible as the error reports the matrix to
#be singular which means that the inverse cannot be found. A way to solve it
#is to change the tolerance in the solve function.

#Problem 3:
lambda = 5
#We observe that a solution is obtained with value of coefficients reduced
solve(X %*% t(X) + lambda * diag(3), X %*% y)
```

```
##           [,1]
## [1,] 0.1119316
## [2,] 0.1347710
## [3,] 0.1843435
```

```
#We see that the matrix can be inverted by adding a small lambda value and coefficients
#are obtained without changing the tolerance level
Xnew <- matrix(rnorm(12), nrow = 4, ncol = 3)
ynew <- matrix(rnorm(3), nrow = 3, ncol = 1)
solve(Xnew %*% t(Xnew) + lambda * diag(4), Xnew %*% ynew)
```

```
##            [,1]
## [1,] -0.2154971
## [2,]  0.4183772
## [3,] -0.1892315
## [4,]  0.4169359
```

```
#Problem 4:
#Function to calculate the ridge regression solution
train.ridge <- function(ip_data, lambda){
  num <- nrow(ip_data[[1]])
  return(solve(ip_data[[1]] %*% t(ip_data[[1]]) + lambda * diag(num), ip_data[[1]] %*% ip_data[[2]]))
```

```
}

#Problem 5:
#X and y are taken from problem 1
a <- list(X,y)
#We observe that the values of train.ridge and problem 3.a are same
train.ridge(a, lambda = 5)
```

```
##           [,1]
## [1,] 0.1119316
## [2,] 0.1347710
## [3,] 0.1843435
```

```
#Problem 6:
#Defining the class of list mentioned in above problem
class(a) <- "ridge"

#Defining generic function train
train <- function(x, ...){
  UseMethod("train")
}

#Running train function on a
train(a, 5)
```

```
##           [,1]
## [1,] 0.1119316
## [2,] 0.1347710
## [3,] 0.1843435
```

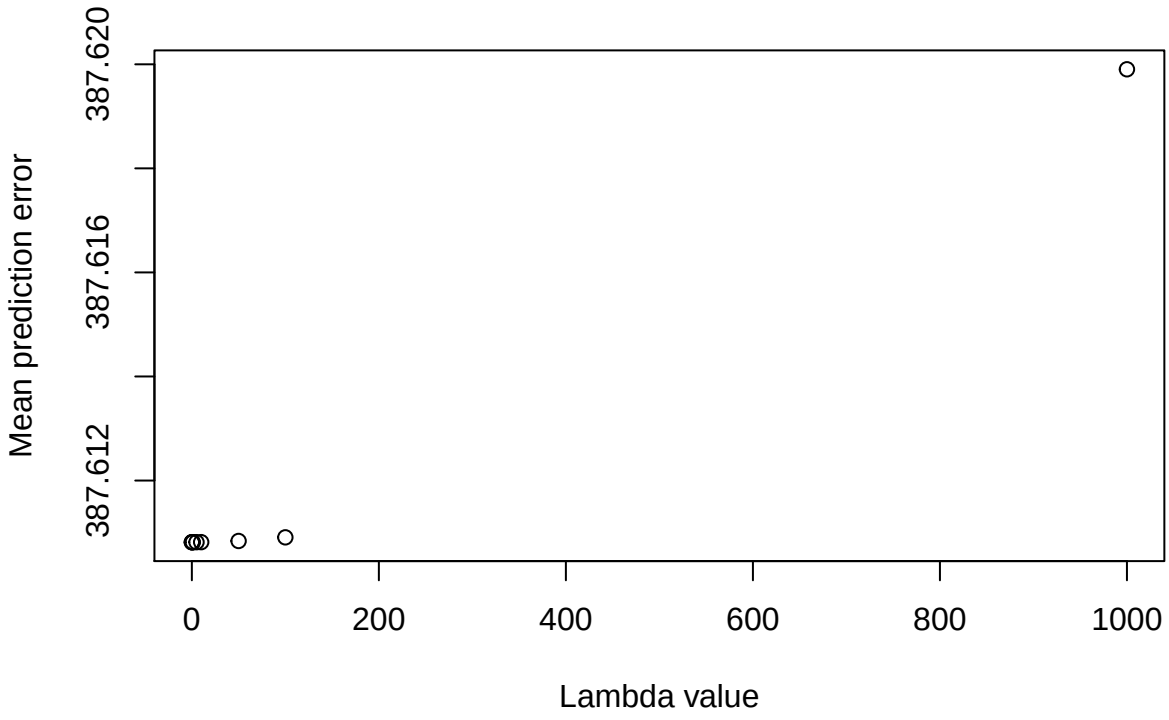
```
#Problem 7:
#Function to find the prediction error
pred_err.ridge <- function(w, cl_rd){
  Ypred <- (t(cl_rd[[1]]) %*% w)
  pred_error <- (Ypred - cl_rd[[2]])
  return(sqrt(sum(pred_error^2) / length(pred_error^2)))
}

pred_err <- function(x, y, ...){
  UseMethod("pred_err")
}

#Problem 8:
crossval1 <- function(cl_rd, lambdas, k){
  X <- as.matrix(cl_rd[[1]])
  row_1 <- nrow(X)
  x_fold <- row_1/k
  n_start <- 1
  n_end <- x_fold
  j <- 1
  x_train <- c(rep(0, k))
  pred_error <- matrix(0, nrow = k, ncol = length(lambdas))
  for(i in 1:k){
    x_train[j] <- list(X[n_start:n_end, ])
    j <- j + 1
```

```
    n_start <- n_end + 1
    n_end <- n_end + x_fold
  }
  for(m in 1:k){
    for(l in 1:length(lambdas)){
      data <- train.ridge(list(t(as.matrix(x_train[[m]])), cl_rd[[2]]), lambdas[l])
      pred_error[m, l] <- pred_err.ridge(data, list(t(as.matrix(x_train[[m]])), cl_rd[[2]]))
    }
  }
  colnames(pred_error) <- paste('lambda =',lambdas)
  rownames(pred_error) <- paste('Set',1:k)
  return(pred_error)
}

#crossval <- function(cl_rd, lambdas, k){
#  X <- cl_rd[[1]]
#  row_1 <- nrow(X)
#  print(row_1)
#  print(x_fold <- row_1/k)
#  j <- 1
#  x_train <- c(rep(0, x_fold))
#  pred_error <- matrix(0, nrow = k, ncol = length(lambdas))
#  for(i in seq(1, row_1 ,x_fold)){
#    x_train[j] <- list(X[i:(i+1), ])
#    j <- j + 1
#  }
#  print(x_train)
#  for(m in 1:k){
#    for(l in 1:length(lambdas)){
#      data <- train.ridge(c(x_train[m], a[2]), lambdas[l])
#      pred_error[m, l] <- pred_err.ridge(data, c(x_train[m], cl_rd[2]))
#    }
#  }
#  colnames(pred_error) <- paste('lambda =',lambdas)
#  rownames(pred_error) <- paste('Set',1:k)
#  return(pred_error)
#}

#Problem 9:
Credit_csv <- read.table("C:\\Users\\pradyuth\\STAT 598 assignment\\Credit dataset.csv", sep = ',',
                         skip = 1)
#colnames(Credit_csv) <- c("ID", "Income",  "Limit","Ratings",  "Cards",    "Age",
#                          "Education",  "Gender",   "Student",  "Married",  "Ethnicity",    "Balance")
y <- as.matrix(Credit_csv[, c(12)])
X <- t(as.matrix(Credit_csv[, c(2:4,6:7)]))

my_credit <- list(X, y)
#class(my_credit) <- "ridge"

#Problem 10:
crossval1(my_credit, c(0, 0.1, 0.5, 1, 5, 10, 50, 100, 1000), 5)
```

```
##       lambda = 0 lambda = 0.1 lambda = 0.5 lambda = 1 lambda = 5
## Set 1   434.1101     434.1101     434.1101   434.1101   434.1101
```

```
## Set 2   265.9182     265.9182     265.9182   265.9182   265.9182
## Set 3   279.1584     279.1584     279.1584   279.1584   279.1584
## Set 4   483.9823     483.9823     483.9823   483.9823   483.9823
## Set 5   474.8851     474.8851     474.8851   474.8851   474.8851
##       lambda = 10 lambda = 50 lambda = 100 lambda = 1000
## Set 1    434.1101    434.1101     434.1101      434.1103
## Set 2    265.9182    265.9182     265.9182      265.9182
## Set 3    279.1584    279.1584     279.1584      279.1584
## Set 4    483.9823    483.9823     483.9823      483.9824
## Set 5    474.8851    474.8852     474.8856      474.9302
```

```r
#Problem 11:
ab <- as.matrix(crossval1(my_credit, c(0, 0.1, 0.5, 1, 5, 10, 50, 100, 1000), 5))
colnames(ab) <- paste("lambda", c(0, 0.1, 0.5, 1, 5, 10, 50, 100, 1000))
mean_vec <- rep(0, ncol(ab))
for(u in 1:ncol(ab)){
  mean_vec[u] <- mean(ab[,u])
}
plot(c(0, 0.1, 0.5, 1, 5, 10, 50, 100, 1000), mean_vec, xlab = "Lambda value", ylab = "Mean prediction
```




```r
#Problem 12:
#Looking at the plot above, we conclude that value of lambda taken is 50
#The value of ridge coefficients is found by train.ridge function
train.ridge(my_credit, 50)
```

```
##           [,1]
## V2  -7.0794640
```

```
## V3   0.2602672
## V4  -0.1578010
## V6  -2.6999739
## V7 -13.0702307
```