homework 03

Lee JongCheol

2024-11-21

202482123 이종철 과제3

```
library(tree)
library(ISLR)
data(Auto)
x <- scale(Auto[,3:7])
y <- Auto$mpg
```

```
summary(cbind(x,target=y))
```

```
##   displacement       horsepower          weight         acceleration
##  Min.   :-1.2080   Min.   :-1.5190   Min.   :-1.6065   Min.   :-2.73349
##  1st Qu.:-0.8544   1st Qu.:-0.7656   1st Qu.:-0.8857   1st Qu.:-0.64024
##  Median :-0.4149   Median :-0.2850   Median :-0.2049   Median :-0.01498
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.: 0.7773   3rd Qu.: 0.5594   3rd Qu.: 0.7501   3rd Qu.: 0.53778
##  Max.   : 2.4902   Max.   : 3.2613   Max.   : 2.5458   Max.   : 3.35597
##       year              target
##  Min.   :-1.62324   Min.   : 9.00
##  1st Qu.:-0.80885   1st Qu.:17.00
##  Median : 0.00554   Median :22.75
##  Mean   : 0.00000   Mean   :23.45
##  3rd Qu.: 0.81993   3rd Qu.:29.00
##  Max.   : 1.63432   Max.   :46.60
```

```
dim(cbind(x,target=y))
```

```
## [1] 392   6
```

- 5 predictors and 1 target variable.
- 392 samples

mpg(target): 연비
displacement: 배기량
horsepower: 마력
weight: 차량 무게 (파운드)
acceleration: 가속도 (0-60mph 도달 시간)
year: 제조 연도 (마지막 두 자리 숫자)

5-fold CV setting

```
set.seed(13579)
gr <- sample(rep(seq(5), length=length(y)))
```

```
table(gr)
```

```
## gr
##  1  2  3  4  5
## 79 79 78 78 78
```

R-squared function

```
R_squared <- function(y_pred, y_actual) {
  rss <- sum((y_actual - y_pred)^2)
  tss <- sum((y_actual - mean(y_actual))^2)
  rsq <- 1-(rss/tss)
  return(rsq)
}
```

1.

```
combs <- list()

for (i in 1:max(gr)) {
  combs[[i]] <- combn(colnames(x), i)
}
```

```
problem1 <- function(comb_mat, x, y, gr) {
  res <- matrix(0, nrow=31, ncol=2)
  for (K in 1:max(gr)) {
    train_x <- x[(gr!=K),]
    train_y <- y[(gr!=K)]
    test_x <- x[(gr==K),]
    test_y <- y[(gr==K)]

    Rsq_LM <- Rsq_RT <- c()
    idx <- 1
    dat <- NA

    for (i in 1:length(comb_mat)) {
      for (p in 1:ncol(comb_mat[[i]])) {
        predictors <- comb_mat[[i]][,p]

        train_dat <- data.frame(train_x[,predictors], target=train_y)
        colnames(train_dat) <- c(predictors, "target")
        LM <- lm(target ~., data=train_dat)
```

```
        RT <- tree(target ~., data=train_dat)

        test_dat <- data.frame(test_x[,predictors])
        colnames(test_dat) <- predictors
        LM_pred <- as.numeric(predict(LM, newdata = test_dat))
        RT_pred <- as.numeric(predict(RT, newdata = test_dat))

        Rsq_LM[idx] <- R_squared(LM_pred, test_y)
        Rsq_RT[idx] <- R_squared(RT_pred, test_y)
        idx <- idx+1
      }
    }
    res <- res + cbind(Rsq_LM, Rsq_RT)
  }
  res <- res/max(gr)
  res <- apply(res, 2, max)
  res <- data.frame(LM=res[1], RT=res[2])
  res <- t(res)
  colnames(res) <- "R-squared"
  return(res)
}

res1 <- problem1(combs, x, y, gr)
res1
```

```
##    R-squared
## LM 0.8041754
## RT 0.8025942
```

2.

```
problem2 <- function(comb_mat, x, y, gr) {

  set.seed(111)
  boot <- vector(mode="list", length=max(gr))
  for (k in 1:max(gr)) {
    mat <- matrix(1:sum(gr!=k), sum(gr!=k), 500)
    mat <- apply(mat, 2, function(t) sample(t, replace=TRUE))
    boot[[k]] <- mat
  }

  res <- rep(0, 31)
  for (K in 1:max(gr)) {
    test_x <- x[(gr==K),]
    test_y <- y[(gr==K)]

    fold_train_x <- x[(gr!=K),]
    fold_train_y <- y[(gr!=K)]

    grs <- boot[[K]]
    y_hat_fold <- matrix(0, nrow=31, ncol=nrow(test_x))
```

```
    Rsq_fold <- NULL
    for (i in 1:ncol(grs)) {
      train_x <- fold_train_x[grs[,i],]
      train_y <- fold_train_y[grs[,i]]

      idx <- 1
      RT_pred <- matrix(0, nrow=31, ncol=nrow(test_x))
      for (i in 1:length(comb_mat)) {
        for (p in 1:ncol(comb_mat[[i]])) {
          predictors <- comb_mat[[i]][,p]

          train_dat <- data.frame(train_x[,predictors], target=train_y)
          colnames(train_dat) <- c(predictors, "target")
          BG <- tree(target ~., data=train_dat)

          test_dat <- data.frame(test_x[,predictors])
          colnames(test_dat) <- predictors
          RT_pred[idx,] <- as.numeric(predict(BG, newdata = test_dat))

          idx <- idx+1
        }
      }
      y_hat_fold <- y_hat_fold + RT_pred # 31*n_samples
    }
    y_hat_fold <- y_hat_fold/ncol(grs) # 500     500
    Rsq_fold <- apply(y_hat_fold, 1, function(t) R_squared(t, test_y)) # 31  R^2
    res <- res + Rsq_fold
  }
  res <- res/max(gr) # 5          R^2
  res <- max(res)
  res <- data.frame(BG=res)
  res <- t(res)
  colnames(res) <- "R-squared"
  return(res)
}

res2 <- problem2(combs, x, y, gr)
res2
```

```
##    R-squared
## BG  0.846116
```

3.

```
D1_func <- function(t) 3/4*(1-t^2)*(abs(t)<1)
D2_func <- function(t) 71/80*(1-t^3)^3*(abs(t)<1)
D3_func <- function(t) 1/sqrt(2*pi)*exp(-1/2*t^2)
D4_func <- function(t) pi/4*cos(pi/2*t)*(abs(t)<1)

Ki0 <- function(lambda, test, train, weight_func) {
  l2_norm <- sqrt(sum((test-train)^2))
```

```
  res <- weight_func(l2_norm/lambda)
  return(res)
}
```

```
problem3 <- function(x, y, gr) {
  Rsq_fold1 <- Rsq_fold2 <- Rsq_fold3 <- Rsq_fold4 <- rep(0, 30)
  for (K in 1:max(gr)) {
    train_x <- x[(gr!=K),]
    train_y <- y[(gr!=K)]
    test_x <- x[(gr==K),]
    test_y <- y[(gr==K)]

    yhat_fold1 <- yhat_fold2 <- yhat_fold3 <- yhat_fold4 <-
      matrix(0, nrow = nrow(test_x), ncol = 30)
    for (j in 1:nrow(test_x)) {
      x0 <- test_x[j,]
      l2_norm_vec <- apply(train_x, 1, function(t) sqrt(sum((x0-t)^2)))
      percent_10 <- quantile(l2_norm_vec, probs = 0.1)
      maximum <- max(l2_norm_vec)

      lambda <- seq(from=percent_10, to=maximum, length.out=30)

      for (l in 1:30) {
        lam_value <- lambda[l]
        weights1 <- apply(train_x, 1, function(t) Ki0(lam_value, x0, t, D1_func))
        weights2 <- apply(train_x, 1, function(t) Ki0(lam_value, x0, t, D2_func))
        weights3 <- apply(train_x, 1, function(t) Ki0(lam_value, x0, t, D3_func))
        weights4 <- apply(train_x, 1, function(t) Ki0(lam_value, x0, t, D4_func))

        train_dat <- data.frame(train_x, target=train_y)
        colnames(train_dat) <- c(colnames(train_x), "target")
        LR1 <- lm(target ~., data=train_dat, weights = weights1)
        LR2 <- lm(target ~., data=train_dat, weights = weights2)
        LR3 <- lm(target ~., data=train_dat, weights = weights3)
        LR4 <- lm(target ~., data=train_dat, weights = weights4)

        test_dat <- data.frame(t(x0))
        colnames(test_dat) <- colnames(test_x)
        yhat_fold1[j,l] <- as.numeric(predict(LR1, newdata = test_dat))
        yhat_fold2[j,l] <- as.numeric(predict(LR2, newdata = test_dat))
        yhat_fold3[j,l] <- as.numeric(predict(LR3, newdata = test_dat))
        yhat_fold4[j,l] <- as.numeric(predict(LR4, newdata = test_dat))
      }
    }
    Rsq_fold1 <- Rsq_fold1 + apply(yhat_fold1, 2, function(t) R_squared(t, test_y))
    Rsq_fold2 <- Rsq_fold2 + apply(yhat_fold2, 2, function(t) R_squared(t, test_y))
    Rsq_fold3 <- Rsq_fold3 + apply(yhat_fold3, 2, function(t) R_squared(t, test_y))
    Rsq_fold4 <- Rsq_fold4 + apply(yhat_fold4, 2, function(t) R_squared(t, test_y))
  }
  Rsq_fold1 <- Rsq_fold1/max(gr)
  Rsq_fold2 <- Rsq_fold2/max(gr)
  Rsq_fold3 <- Rsq_fold3/max(gr)
  Rsq_fold4 <- Rsq_fold4/max(gr)
```

```
  res <- data.frame(LR1 = max(Rsq_fold1),
                    LR2 = max(Rsq_fold2),
                    LR3 = max(Rsq_fold3),
                    LR4 = max(Rsq_fold4))
  res <- t(res)
  colnames(res) <- "R-squared"
  return(res)
}

res3 <- problem3(x, y, gr)
res3
```

```
##     R-squared
## LR1 0.8746833
## LR2 0.8751856
## LR3 0.8629791
## LR4 0.8751375
```

4.

```
Ki0_gamma <- function(gamma, test_x, train_x) {
  l2_norm <- apply(train_x, 1, function(t) sqrt(sum((test_x-t)^2)))
  near_idx <- order(l2_norm, decreasing = F)[1:gamma]
  res <- rep(0, nrow(train_x))
  res[near_idx] <- 1
  res <- res/gamma
  return(res)
}

problem4 <- function(x, y, gr) {
  Rsq_fold <- rep(0, 40)
  for (K in 1:max(gr)) {
    train_x <- x[(gr!=K),]
    train_y <- y[(gr!=K)]
    test_x <- x[(gr==K),]
    test_y <- y[(gr==K)]

    yhat_fold <- matrix(0, nrow = nrow(test_x), ncol = 40)

    for (j in 1:nrow(test_x)) {
      x0 <- test_x[j,]

      for (g in 41:80) {
        weights <- Ki0_gamma(g, x0, train_x)

        train_dat <- data.frame(train_x, target=train_y)
        colnames(train_dat) <- c(colnames(train_x), "target")
        LR5 <- lm(target ~., data=train_dat, weights = weights)

        test_dat <- data.frame(t(x0))
        colnames(test_dat) <- colnames(test_x)
```

```
        yhat_fold[j,g-40] <- as.numeric(predict(LR5, newdata = test_dat))
      }
    }
    Rsq_fold <- Rsq_fold + apply(yhat_fold, 2, function(t) R_squared(t, test_y))
  }
  Rsq_fold <- Rsq_fold/max(gr)
  res = data.frame(LR5=max(Rsq_fold))
  res <- t(res)
  colnames(res) <- "R-squared"
  return(res)
}

res4 <- problem4(x, y, gr)
res4
```

```
##     R-squared
## LR5 0.8735301
```

5.

```
Ki0_q_gamma <- function(gamma, q, test_x, train_x, dist_func) {
  l2_norm <- apply(train_x, 1, function(t) sqrt(sum((test_x-t)^2)))
  l2_near_idx <- order(l2_norm, decreasing = F)[1:gamma]
  indicat <- rep(FALSE, nrow(train_x))
  indicat[l2_near_idx] <- TRUE

  train_x_N <- train_x[indicat,]
  lq_norm <- apply(train_x_N, 1, function(t) (sum((abs(test_x-t))^q))^(1/q))
  lambda <- max(lq_norm)

  res <- rep(0, nrow(train_x))
  weights <- sapply(lq_norm, function(z) dist_func(z/lambda))
  res[indicat] <- weights
  return(res)
}

problem5 <- function(x, y, gr) {
  Rsq_fold1 <- Rsq_fold2 <- Rsq_fold3 <- Rsq_fold4 <-
    matrix(0, nrow=5, ncol=40)

  for (K in 1:max(gr)) {
    train_x <- x[(gr!=K),]
    train_y <- y[(gr!=K)]
    test_x <- x[(gr==K),]
    test_y <- y[(gr==K)]
    Rsq_each_gq1 <- Rsq_each_gq2 <- Rsq_each_gq3 <- Rsq_each_gq4 <-
      matrix(0, nrow=5, ncol=40)

    for (q in 1:5) {
      yhat_fold1 <- yhat_fold2 <- yhat_fold3 <- yhat_fold4 <-
        matrix(0, nrow = nrow(test_x), ncol = 40)
```

```
      for (g in 41:80) {
        for (j in 1:nrow(test_x)) {
          x0 <- test_x[j,]
          weights1 <- Ki0_q_gamma(g, q, x0, train_x, D1_func)
          weights2 <- Ki0_q_gamma(g, q, x0, train_x, D2_func)
          weights3 <- Ki0_q_gamma(g, q, x0, train_x, D3_func)
          weights4 <- Ki0_q_gamma(g, q, x0, train_x, D4_func)

          train_dat <- data.frame(train_x, target=train_y)
          colnames(train_dat) <- c(colnames(train_x), "target")
          LR6 <- lm(target ~., data=train_dat, weights = weights1)
          LR7 <- lm(target ~., data=train_dat, weights = weights2)
          LR8 <- lm(target ~., data=train_dat, weights = weights3)
          LR9 <- lm(target ~., data=train_dat, weights = weights4)

          test_dat <- data.frame(t(x0))
          colnames(test_dat) <- colnames(test_x)
          yhat_fold1[j,g-40] <- as.numeric(predict(LR6, newdata = test_dat))
          yhat_fold2[j,g-40] <- as.numeric(predict(LR7, newdata = test_dat))
          yhat_fold3[j,g-40] <- as.numeric(predict(LR8, newdata = test_dat))
          yhat_fold4[j,g-40] <- as.numeric(predict(LR9, newdata = test_dat))
        }
      }
      Rsq_each_gq1[q,] <- apply(yhat_fold1, 2, function(t) R_squared(t, test_y))
      Rsq_each_gq2[q,] <- apply(yhat_fold2, 2, function(t) R_squared(t, test_y))
      Rsq_each_gq3[q,] <- apply(yhat_fold3, 2, function(t) R_squared(t, test_y))
      Rsq_each_gq4[q,] <- apply(yhat_fold4, 2, function(t) R_squared(t, test_y))
    }

    Rsq_fold1 <- Rsq_fold1 + Rsq_each_gq1
    Rsq_fold2 <- Rsq_fold2 + Rsq_each_gq2
    Rsq_fold3 <- Rsq_fold3 + Rsq_each_gq3
    Rsq_fold4 <- Rsq_fold4 + Rsq_each_gq4
  }
  Rsq_fold1 <- Rsq_fold1/max(gr)
  Rsq_fold2 <- Rsq_fold2/max(gr)
  Rsq_fold3 <- Rsq_fold3/max(gr)
  Rsq_fold4 <- Rsq_fold4/max(gr)
  res <- data.frame(LR6 = max(Rsq_fold1),
                    LR7 = max(Rsq_fold2),
                    LR8 = max(Rsq_fold3),
                    LR9 = max(Rsq_fold4))
  res <- t(res)
  colnames(res) <- "R-squared"
  return(res)
}

res5 <- problem5(x, y, gr)
res5
```

```
##     R-squared
## LR6 0.8771152
## LR7 0.8776815
```

```
## LR8 0.8753583
## LR9 0.8774279
```

6.

```
set.seed(54321)
gr10 <- matrix(rep(seq(5), length=length(y)), length(y), 10)
gr10 <- apply(gr10, 2, sample)

problem6 <- function(comb_mat, x, y, gr_mat) {
  res1 <- matrix(0, nrow=2, ncol=10)
  res2 <- matrix(0, nrow=1, ncol=10)
  res3 <- matrix(0, nrow=4, ncol=10)
  res4 <- matrix(0, nrow=1, ncol=10)
  res5 <- matrix(0, nrow=4, ncol=10)

  for (r in 1:ncol(gr_mat)) {
    res1[,r] <- as.numeric(problem1(comb_mat, x, y, gr_mat[,r]))
    res2[,r] <- as.numeric(problem2(comb_mat, x, y, gr_mat[,r]))
    res3[,r] <- as.numeric(problem3(x, y, gr_mat[,r]))
    res4[,r] <- as.numeric(problem4(x, y, gr_mat[,r]))
    res5[,r] <- as.numeric(problem5(x, y, gr_mat[,r]))
  }

  res1 <- apply(res1, 1, mean)
  res2 <- apply(res2, 1, mean)
  res3 <- apply(res3, 1, mean)
  res4 <- apply(res4, 1, mean)
  res5 <- apply(res5, 1, mean)

  res <- data.frame(LM = res1[1], RT = res1[2],
                    BG = res2, LR1 = res3[1],
                    LR2 = res3[2], LR3 = res3[3],
                    LR4 = res3[4], LR5 = res4,
                    LR6 = res5[1], LR7 = res5[2],
                    LR8 = res5[3], LR9 = res5[4]
                    )
  res <- t(res)
  colnames(res) <- "R-squared"
  return(res)
}

final_res <- problem6(combs, x, y, gr10)
```

```
final_res
```

```
##     R-squared
## LM  0.8049224
## RT  0.7965895
## BG  0.8437017
## LR1 0.8779391
```

```
## LR2 0.8779272
## LR3 0.8631833
## LR4 0.8784493
## LR5 0.8721175
## LR6 0.8789374
## LR7 0.8807941
## LR8 0.8747680
## LR9 0.8795372
```

```
rownames(final_res)[which.max(as.numeric(final_res))]
```

```
## [1] "LR7"
```