Homework 3 Problem 3

STAT 435 Spring 2024

Luke VanHouten

2024-05-15

Exercises

3. a. Here is my data:

```
conn <- dbConnect(RSQLite::SQLite(), dbname = "lahman_1871-2021.sqlite")

query <- "
SELECT p.playerID, W, L, p.G, p.GS, CG, SHO, SV, IPouts, H, ER, HR, BB, SO,
   BAOpp, ERA, IBB, p.WP, HBP, BK, BFP, GF, R, SH, SF, GIDP, PO, A, E, DP
FROM Pitching AS p, Fielding AS f
WHERE p.playerID = f.playerID
   AND p.yearID = f.yearID
   AND p.stint = f.stint
   AND p.teamID = f.teamID
   AND p.lgID = f.lgID"

data <- dbGetQuery(conn, query)

grouped_df <- data %>%
    group_by(playerID) %>%
    summarise(across(c(W, L, G, GS, CG, SHO, SV, IPouts, H, ER, HR, BB, SO, IBB,
                       WP, HBP, BK, BFP, GF, R, SH, SF, GIDP, PO, A, E, DP),
                     \(x) sum(x, na.rm = TRUE)),
              across(c(BAOpp, ERA), \(x) mean(x, na.rm = TRUE))) %>%
    filter(complete.cases(.))

war <- read.csv("war-pitchers.csv") %>%
    select(player_ID, year_ID, WAR, BIP) %>%
    filter(WAR != "NULL") %>%
    group_by(player_ID) %>%
    summarise(WAR = sum(as.numeric(WAR)), BIP = sum(as.numeric(BIP))) %>%
    filter(complete.cases(.))

predictors <- merge(war, grouped_df, by.x = "player_ID", by.y = "playerID") %>%
    select(-player_ID)

predictors <- predictors %>%
        mutate(id = rownames(predictors))
```

```r
head(predictors)
```

```
##     WAR  BIP  W   L   G  GS CG SHO SV IPouts    H  ER  HR  BB  SO IBB WP HBP BK
## 1  1.85  934 16  18 331   0  0   0 69   1011  296 160  41 183 340  22 12  16  1
## 2 15.09 3556 66  60 448  91 22   5 82   3328 1085 468  89 457 641  45 22   7  3
## 3  3.24 1083  8  29 400   6  0   0  2   1045  332 146  43 123 290  11 10  12  2
## 4 -0.89  320  0   0   7   1  0   0  0     52   20  13   7  11  12   0  2   0  0
## 5  5.26 4510 62  83 248 206 37   5  0   3858 1405 627 162 352 484  28 18  32  5
## 6 19.70 5670 87 108 263 254 31   6  0   5022 1779 791 154 620 888  30 53  32 11
##    BFP  GF   R SH SF GIDP  PO   A  E DP     BAOpp      ERA id
## 1 1475 141 169 17 11   21  11  29  3  2 0.2574444 5.194444  1
## 2 4730 235 503 50 34  106  67 135 13 10 0.2508462 3.493077  2
## 3 1481 101 155  7 12   28   7  40  2  2 0.2503636 4.219091  3
## 4   82   2  15  1  0    0   2   1  0  0 0.2860000 6.750000  4
## 5 5508  13 707 60 39  111 113 187 17 12 0.2785833 4.331667  5
## 6 7211   5 880 70 47  200  72 300  9 16 0.2803636 4.496364  6
```

```r
# Here is the number of predictors
print(ncol(predictors) - 2)
```

```
## [1] 30
```

```r
# Here is the number of data points
print(nrow(predictors))
```

```
## [1] 8779
```

I chose option I, and I used the Lahman Baseball Database, which includes basic baseball statistics at the player level from 1871 to 2021. I also used a dataset of the Wins Above Replacement (WAR) statistic for pitchers from Baseball-Reference.com. I am focusing on pitchers here, and my response variable is the aforementioned WAR, which is the preferred statistic for measuring general player value. My predictors are pitching statistics such as Earned Run Average (ERA), strikeouts, opposing batting average, and others. I also included fielding statistics. WAR is computed from many of these statistics, so a fair share of them should be good predictors for it (although this should vary). My data is at the player level per their careers, but does not include their names as I am only looking at numerical data.

b. We can split the data here using an 80/20 train/test split:

```r
set.seed(123)

train <- predictors %>%
    sample_frac(0.8)
test <- anti_join(predictors, train, by = "id") %>%
    select(-id)

train <- train %>%
    select(-id)
```

We obviously have data points than predictors. Now, we can do our forward subset selection using the `step` function (instead of `regsubsets`):

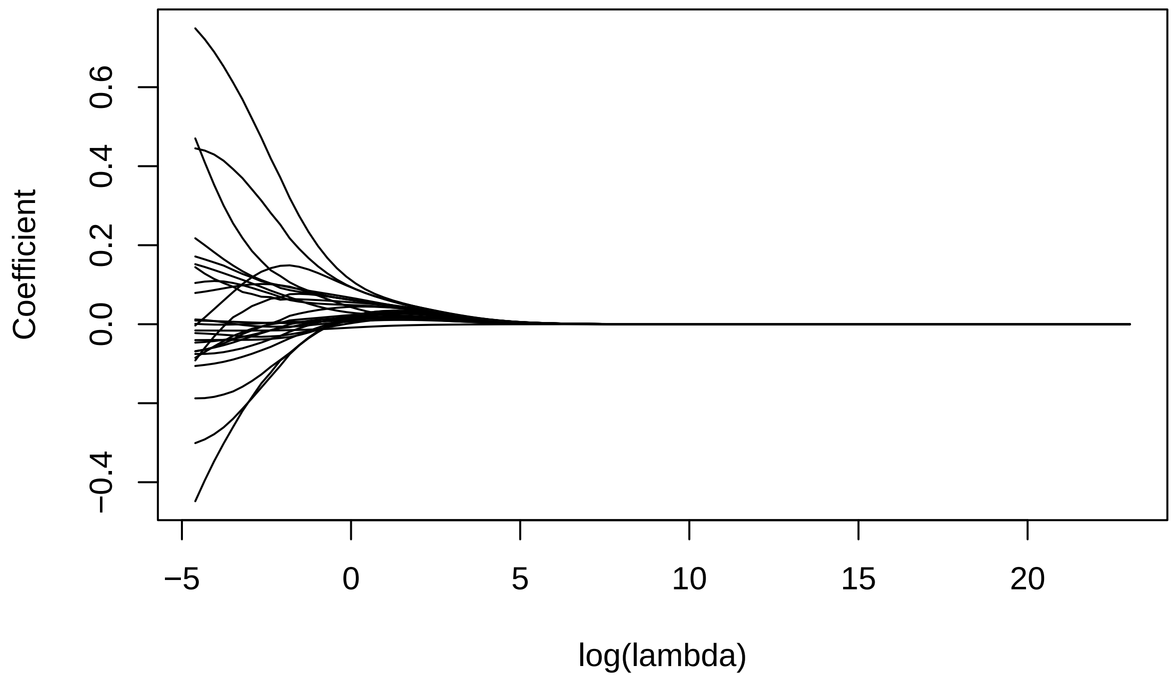
```r
base_model <- lm(WAR ~ 1, data = train)
full_model <- lm(WAR ~ ., data = train)

reduced_model <- step(base_model, scope = list(upper = full_model, lower = ~1),
                      direction = "forward", k = 2)
```

```r
step_predictions <- predict(reduced_model, newdata = test)

num_predictors <- length(coef(reduced_model)) - 1
step_error <- round(mean((test$WAR - unname(step_predictions)) ^ 2), 4)
```

There are 26 predictors used in the reduced model and we have a test error of 6.817.

c. Here is the ridge regression model:

```r
grid <- 10 ^ seq(10, -2, length = 100)

scaled_train <- scale(train[, c("WAR", names(reduced_model$coefficients)[-1])])

ridge <- glmnet(scaled_train[, -1], scaled_train[, 1], alpha = 0, lambda = grid)
cv_ridge <- cv.glmnet(scaled_train[, -1], scaled_train[, 1], alpha = 0,
                      type.measure = "mse", nfolds = 10, lambda = grid)

coefficients <- matrix(NA, nrow = num_predictors, ncol = 100)
for (i in 1:num_predictors) {
    for (j in 1:100) {
        coefficients[i, j] <- coef(ridge)[, j][i + 1]
    }
}

plot(log(grid), coefficients[1, ], type = "l",
     xlab = "log(lambda)", ylab="Coefficient",
     ylim = c(min(coefficients), max(coefficients)))

for (i in 2:num_predictors) {
    points(log(grid), coefficients[i, ], type = "l")
}
```



d. Here we can find the λ that gives us the smalles cross-validation error:

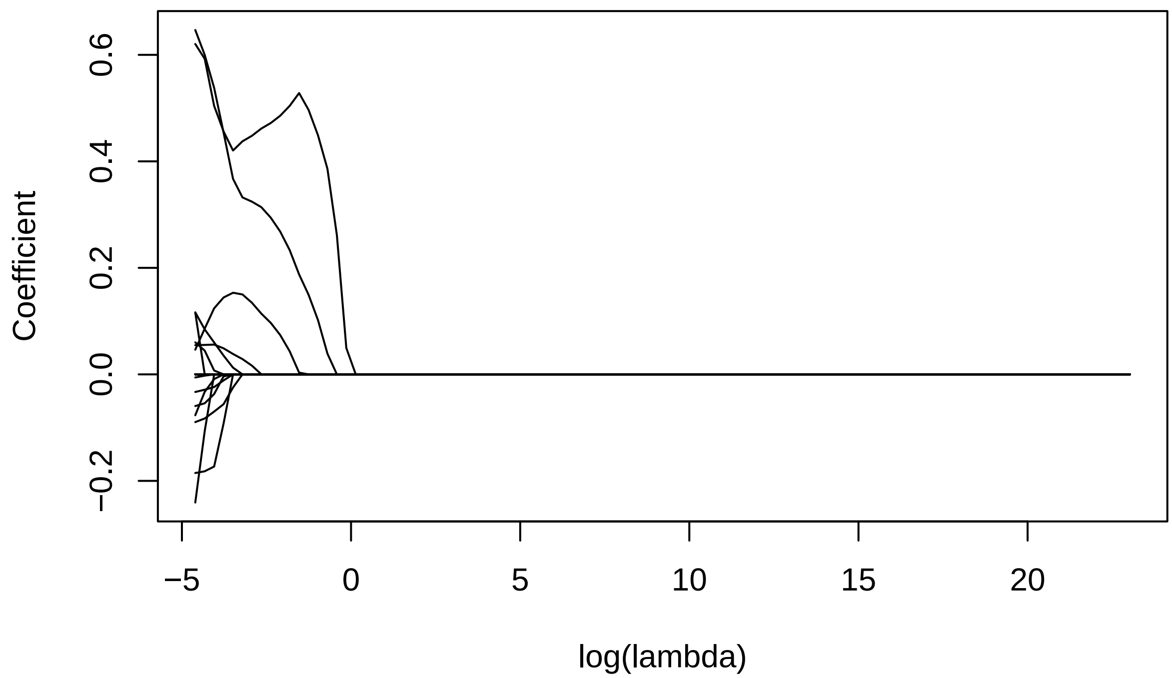
```
lambda <- cv_ridge$lambda.min

test_scaled <- scale(test[, c("WAR", names(reduced_model$coefficients)[-1])])

predictions <- predict(ridge, s = lambda, newx = test_scaled[, -1])

test_error <- round(mean((test_scaled[, 1] - predictions) ^ 2), 4)
```

The value of λ that gives the smallest CV error is $\lambda = 0.01$. The test error of this model is 0.0765. This looks a lot different than the error in part b., because we standardized the data before performing ridge regression.

e. Here, we can fit a LASSO model:

```
lasso <-  glmnet(scaled_train[, -1], scaled_train[, 1], alpha = 1,
                 lambda = grid)
cv_lasso <- cv.glmnet(scaled_train[, -1], scaled_train[, 1], alpha = 1,
                      type.measure = "mse", nfolds = 10, lambda = grid)

coefficients_lasso <- matrix(NA, nrow = num_predictors, ncol = 100)
for (i in 1:num_predictors) {
    for (j in 1:100) {
```

```r
        coefficients_lasso[i, j] <- coef(lasso)[, j][i + 1]
    }
}

plot(log(grid), coefficients_lasso[1, ], type = "l",
     xlab = "log(lambda)", ylab="Coefficient",
     ylim = c(min(coefficients_lasso), max(coefficients_lasso)))

for (i in 2:num_predictors) {
    points(log(grid), coefficients_lasso[i, ], type = "l")
}
```




```r
lambda_lasso <- cv_lasso$lambda.min

predictions_lasso <- predict(lasso, s = lambda, newx = test_scaled[, -1])

test_error_lasso <- round(mean((test_scaled[, 1] - predictions_lasso) ^ 2), 4)
```

The value of λ that gives the smallest CV error is $\lambda = 0.01$. The test error of this model is 0.0994. Here are the features that have a non-zero estimation coefficient:

```r
nonzero_coefs <- names(train[, -1])[which(rowSums(coefficients_lasso) != 0)]
nonzero_coefs
```

```
##  [1] "BIP"   "W"     "L"     "G"     "GS"    "CG"    "SHO"   "ER"    "HR"    "IBB"
```

```
## [11] "WP"   "BFP"  "GF"   "GIDP"
```

There are 14 of these. A lot of these being included make sense, because WAR is an accumulative baseball statistic; pitchers with longer careers are going to have a higher WAR. So the pitchers with a high WAR are going to have more wins, losses, games started, home runs given up, etc. than other pitchers with smaller careers and less WAR.

I received full points on this problem.