Lab 4

Hubert Majewski

11:59PM March 11, 2021

Load up the famous iris dataset. We are going to do a different prediction problem. Imagine the only input x is Species and you are trying to predict y which is Petal.Length. A reasonable prediction is the average petal length within each Species. Prove that this is the OLS model by fitting an appropriate `lm` and then using the predict function to verify.

```
data(iris)


Model <- lm (Petal.Length ~ Species, iris)

mean(iris$Petal.Length[iris$Species == "setosa"])
```

```
## [1] 1.462
```

```
mean(iris$Petal.Length[iris$Species == "versicolor"])
```

```
## [1] 4.26
```

```
mean(iris$Petal.Length[iris$Species == "virginica"])
```

```
## [1] 5.552
```

```
?predict
```

```
## starting httpd help server ... done
```

```
predict(Model, data.frame(Species = c("setosa")))
```

```
##     1
## 1.462
```

```
predict(Model, data.frame(Species = c("versicolor")))
```

```
##    1
## 4.26
```

```r
predict(Model, data.frame(Species = c("virginica")))
```

```
##     1
## 5.552
```

Construct the design matrix with an intercept, X, without using `model.matrix`.

```r
X <- cbind(1, iris$Species == "versicolor", iris$Species == "virginica")

head(X)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    1    0    0
## [3,]    1    0    0
## [4,]    1    0    0
## [5,]    1    0    0
## [6,]    1    0    0
```

Find the hat matrix H for this regression.

```r
H <- X %*% solve(t(X) %*% X) %*% t(X)

Matrix::rankMatrix(H)
```

```
## [1] 3
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 3.330669e-14
```

Verify this hat matrix is symmetric using the `expect_equal` function in the package `testthat`.

```r
pacman::p_load(testthat)

expect_equal(H, t(H))
```

Verify this hat matrix is idempotent using the `expect_equal` function in the package `testthat`.

```r
pacman::p_load(testthat)

expect_equal(H, H %*% H)
```

Using the `diag` function, find the trace of the hat matrix.

```r
?diag

trace <- sum(diag(H))
```

It turns out the trace of a hat matrix is the same as its rank! But we don't have time to prove these interesting and useful facts..

For masters students: create a matrix $X_\perp$.

```
#Masters Only
```

Using the hat matrix, compute the $\hat{y}$ vector and using the projection onto the residual space, compute the e vector and verify they are orthogonal to each other.

```
y <- iris$Petal.Length
y_hat <- H %*% y

e <- (diag(nrow(iris)) - H) %*% y

head(y_hat)
```

```
##       [,1]
## [1,] 1.462
## [2,] 1.462
## [3,] 1.462
## [4,] 1.462
## [5,] 1.462
## [6,] 1.462
```

```
head(e)
```

```
##        [,1]
## [1,] -0.062
## [2,] -0.062
## [3,] -0.162
## [4,]  0.038
## [5,] -0.062
## [6,]  0.238
```

```
#Orthogonal if approaches 0
expect_equal(t(e) %*% y_hat, as.matrix(0))
```

Compute SST, SSR and SSE and R^2 and then show that SST = SSR + SSE.

```
y_bar <- mean(y)

SSE <- t(e) %*% e
SST <- t(y - y_bar) %*% (y - y_bar)

Rsq <- 1 - SSE / SST
Rsq
```

```
##           [,1]
## [1,] 0.9413717
```

```r
SSR <- t(y_hat - y_bar) %*% (y_hat - y_bar)
SSR
```

```
##          [,1]
## [1,] 437.1028
```

```r
expect_equal(SSR + SSE, SST)
```

Find the angle θ between y - $\bar{y}1$ and $\hat{y} - \bar{y}1$ and then verify that its cosine squared is the same as the R^2 from the previous problem.

```r
theta <- acos( t(y - y_bar) %*% (y_hat - y_bar) / sqrt(SST * SSR) )
theta
```

```
##           [,1]
## [1,] 0.2445634
```

```r
theta * 180 / pi
```

```
##          [,1]
## [1,] 14.01245
```

```r
cos(theta) ^ 2
```

```
##           [,1]
## [1,] 0.9413717
```

```r
expect_equal(cos(theta)^2, Rsq)
```

Project the y vector onto each column of the X matrix and test if the sum of these projections is the same as yhat.

```r
proj1 <- ((X[, 1] %*% t(X[, 1])) / as.numeric((t(X[, 1]) %*% X[, 1]))) %*% y
proj2 <- ((X[, 2] %*% t(X[, 2])) / as.numeric((t(X[, 2]) %*% X[, 2]))) %*% y
proj3 <- ((X[, 3] %*% t(X[, 3])) / as.numeric((t(X[, 3]) %*% X[, 3]))) %*% y

#Not orthogonal, therefore, not equal.
#expect_equal(proj1 + proj2 + proj3, y_hat)
```

Construct the design matrix without an intercept, X, without using `model.matrix`.

```r
xOld <- X
HOld <- H

X <- cbind((iris$Species == "setosa"),
                     as.numeric(iris$Species == "versicolor"),
                     (iris$Species == "virginica"))

y <- iris$Petal.Length

head(X)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    1    0    0
## [3,]    1    0    0
## [4,]    1    0    0
## [5,]    1    0    0
## [6,]    1    0    0
```

Find the OLS estimates using this design matrix. It should be the sample averages of the petal lengths within species.

```r
#Project matrix
H <- X %*% solve(t(X) %*% X) %*% t(X)

yHat <- H %*% y

unique(yHat)
```

```
##       [,1]
## [1,] 1.462
## [2,] 4.260
## [3,] 5.552
```

```r
mean(iris$Petal.Length[iris$Species == "setosa"])
```

```
## [1] 1.462
```

```r
mean(iris$Petal.Length[iris$Species == "versicolor"])
```

```
## [1] 4.26
```

```r
mean(iris$Petal.Length[iris$Species == "virginica"])
```

```
## [1] 5.552
```

Verify the hat matrix constructed from this design matrix is the same as the hat matrix constructed from the design matrix with the intercept. (Fact: orthogonal projection matrices are unique).

```r
pacman::p_load(testthat)

expect_equal(H, HOld)
```

Project the y vector onto each column of the X matrix and test if the sum of these projections is the same as yhat.

```r
Hy <- H %*% y

expect_equal(Hy, yHat)
```

Convert this design matrix into Q, an orthonormal matrix.

```r
q <- qr(xOld)
Q <- qr.Q(q)
R <- qr.R(q)

dim(Q)
```

```
## [1] 150   3
```

```r
dim(R)
```

```
## [1] 3 3
```

```r
Matrix::rankMatrix(Q)
```

```
## [1] 3
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 3.330669e-14
```

```r
Matrix::rankMatrix(R)
```

```
## [1] 3
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 6.661338e-16
```

```r
# Therefore it is orthonormal
```

Project the y vector onto each column of the Q matrix and test if the sum of these projections is the same as yhat.

```r
# Same as above question

proj1 <- ((Q[, 1] %*% t(Q[, 1])) / as.numeric((t(Q[, 1]) %*% Q[, 1]))) %*% y
proj2 <- ((Q[, 2] %*% t(Q[, 2])) / as.numeric((t(Q[, 2]) %*% Q[, 2]))) %*% y
proj3 <- ((Q[, 3] %*% t(Q[, 3])) / as.numeric((t(Q[, 3]) %*% Q[, 3]))) %*% y

# Equal therefore these projections are orthognal
expect_equal(proj1 + proj2 + proj3, yHat)
```

Find the $p = 3$ linear OLS estimates if Q is used as the design matrix using the `lm` method. Is the OLS solution the same as the OLS solution for X?

```r
ModelX <- lm(y ~ xOld, iris)
ModelX
```

```
##
## Call:
## lm(formula = y ~ xOld, data = iris)
##
## Coefficients:
## (Intercept)        xOld1        xOld2        xOld3
##       1.462           NA        2.798        4.090
```

```r
# Set intercept to 0
ModelQ <- lm(Petal.Length ~ 0 + Q, iris)
ModelQ
```

```
##
## Call:
## lm(formula = Petal.Length ~ 0 + Q, data = iris)
##
## Coefficients:
##      Q1       Q2       Q3
## -46.026    4.347   20.450
```

Use the predict function and ensure that the predicted values are the same for both linear models: the one created with X as its design matrix and the one created with Q as its design matrix.

```r
predict(ModelQ, data.frame(Q))
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12    13
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    14    15    16    17    18    19    20    21    22    23    24    25    26
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    27    28    29    30    31    32    33    34    35    36    37    38    39
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    40    41    42    43    44    45    46    47    48    49    50    51    52
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 4.260 4.260
##    53    54    55    56    57    58    59    60    61    62    63    64    65
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##    66    67    68    69    70    71    72    73    74    75    76    77    78
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##    79    80    81    82    83    84    85    86    87    88    89    90    91
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##    92    93    94    95    96    97    98    99   100   101   102   103   104
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 5.552 5.552 5.552 5.552
##   105   106   107   108   109   110   111   112   113   114   115   116   117
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552
##   118   119   120   121   122   123   124   125   126   127   128   129   130
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552
##   131   132   133   134   135   136   137   138   139   140   141   142   143
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552
##   144   145   146   147   148   149   150
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552
```

```
predict(ModelX, data.frame(xOld[1]))
```

```
## Warning: 'newdata' had 1 row but variables found have 150 rows

## Warning in predict.lm(ModelX, data.frame(xOld[1])): prediction from a rank-
## deficient fit may be misleading

##     1     2     3     4     5     6     7     8     9    10    11    12    13
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    14    15    16    17    18    19    20    21    22    23    24    25    26
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    27    28    29    30    31    32    33    34    35    36    37    38    39
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    40    41    42    43    44    45    46    47    48    49    50    51    52
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 4.260 4.260
##    53    54    55    56    57    58    59    60    61    62    63    64    65
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##    66    67    68    69    70    71    72    73    74    75    76    77    78
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##    79    80    81    82    83    84    85    86    87    88    89    90    91
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##    92    93    94    95    96    97    98    99   100   101   102   103   104
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 5.552 5.552 5.552 5.552
##   105   106   107   108   109   110   111   112   113   114   115   116   117
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552
##   118   119   120   121   122   123   124   125   126   127   128   129   130
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552
##   131   132   133   134   135   136   137   138   139   140   141   142   143
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552
##   144   145   146   147   148   149   150
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552
```

Clear the workspace and load the boston housing data and extract X and y. The dimensions are $n = 506$ and $p = 13$. Create a matrix that is $(p+1) \times (p+1)$ full of NA's. Label the columns the same columns as X. Do not label the rows. For the first row, find the OLS estimate of the y regressed on the first column only and put that in the first entry. For the second row, find the OLS estimates of the y regressed on the first and second columns of X only and put them in the first and second entries. For the third row, find the OLS estimates of the y regressed on the first, second and third columns of X only and put them in the first, second and third entries, etc. For the last row, fill it with the full OLS estimates.

```
#Clear
rm(list=ls())

#Import
D <- MASS::Boston

X <- as.matrix(cbind(1, D[, 1:13]))
y <- as.matrix(D[, ncol(D)])

#Output
M <- matrix(data = NA, nrow = ncol(D), ncol = ncol(D))
```

```r
#Copy over column names from original data (not label rows)
colnames(M) <- c(colnames(X))

for (i in 1 : ncol(D)) {

  B <- array(data = NA, dim = ncol(D))

  XStar <- X[, 1:i]
  XStar <- as.matrix(XStar) # Have to convert it as R doesnt seem to like Matricies

  B[1:i] <- solve(t(XStar) %*% XStar) %*% t(XStar) %*% D$medv

  M[i, ] <- B
}

head(M)
```

```
##             1       crim         zn      indus     chas       nox rm age dis
## [1,] 22.53281         NA         NA         NA       NA        NA NA  NA  NA
## [2,] 24.03311 -0.4151903         NA         NA       NA        NA NA  NA  NA
## [3,] 22.48563 -0.3520783 0.11610909         NA       NA        NA NA  NA  NA
## [4,] 27.39465 -0.2486283 0.05850082 -0.4155778       NA        NA NA  NA  NA
## [5,] 27.11280 -0.2287981 0.05928665 -0.4403251 6.894059        NA NA  NA  NA
## [6,] 29.48994 -0.2185190 0.05511047 -0.3834805 7.026223 -5.424659 NA  NA  NA
##      rad tax ptratio black lstat
## [1,]  NA  NA      NA    NA    NA
## [2,]  NA  NA      NA    NA    NA
## [3,]  NA  NA      NA    NA    NA
## [4,]  NA  NA      NA    NA    NA
## [5,]  NA  NA      NA    NA    NA
## [6,]  NA  NA      NA    NA    NA
```

Why are the estimates changing from row to row as you add in more predictors?

The collection of rows above represents a new model with a different set of features. We can see that as we add more features to each model (not represented by NA), the y intercept (1 column) will increase while the coefficients will be assigned appropriate weights for each attribute (for each additional feature).

Create a vector of length $p + 1$ and compute the R^2 values for each of the above models.

```r
R2s <- array(dim = ncol(D))
yBar <- mean(y)
SST <- sum((y - yBar) ^ 2)


for(i in 1: nrow(R2s) ) {

  b <- as.matrix(c(M[i, 1:i], rep(0, nrow(M) - i)) )

  yHat <- X %*% b
  SSR <- sum((yHat - yBar) ^ 2)
  RSQ <- SSR / SST

  R2s[i] <- RSQ
```

```
}

R2s
```

```
##  [1] 5.382448e-30 1.507805e-01 2.339884e-01 2.937136e-01 3.295277e-01
##  [6] 3.313127e-01 5.873770e-01 5.894902e-01 6.311488e-01 6.319479e-01
## [11] 6.396628e-01 6.703141e-01 6.842043e-01 7.406427e-01
```

Is R^2 monotonically increasing? Why?

We are continuously trying to fit for a better model with each additional attribute/feature. Therefore, for each feature we add in, it makes sense that the R^2 value is increasing as we are starting fit the data, therefore, increasing R^2 as it is monotonic in this case. (Features conform to data)