Lab 4

Enoch Kim

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

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

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

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

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

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

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

```
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.

```
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`.

```
pacman::p_load(testthat)
expect_equal(H, t(H))
```

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

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

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

```
sum(diag(H))
```

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

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 $X_\perp$.

```
#TO-DO
```

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.

```r
y = iris$Petal.Length
y_hat = H %*% iris$Petal.Length
table(y_hat)
```

```
## y_hat
## 1.462  4.26 5.552
##    50    50    50
```

```r
I = diag(nrow(iris))
e = (I - H) %*% y
t(e) %*% y_hat
```

```
##               [,1]
## [1,] -2.2915e-13
```

```r
Matrix::rankMatrix(I - H)
```

```
## [1] 147
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 3.330669e-14
```

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

```r
SSE = t(e) %*% e
y_bar = mean(y)
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)

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 * (180 / pi)
```

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

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

```
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

#Note: There are not suppose to be equal, expect_equal(proj1 + proj2 + proj3, y_hat).
```

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

```
X2 <- cbind(as.numeric(iris$Species == "setosa"), iris$Species == "versicolor", iris$Species == "virgin
y = iris$Petal.Length
head(X2)
```

```
##      [,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.

```
H2 = X2 %*% solve(t(X2) %*% X2) %*% t(X2)
y_hat_2 = H2 %*% y

unique(y_hat)
```

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

```
unique(y_hat_2)
```

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

```
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
```

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).

```
pacman::p_load(testthat)
expect_equal(H, H2)
```

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

```
H_Y = H2 %*% y
expect_equal(H_Y, y_hat_2)
```

Convert this design matrix into Q, an orthonormal matrix.

```
qORTH = qr(X2)
Q = qr.Q(qORTH)
dim(Q)
```

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

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

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

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

```
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

y_hat_Q = Q %*% t(Q) %*% y
expect_equal(y_hat_Q, y_hat_2)
```

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?

```
mod_Q = lm(y ~ 0 + Q)
coef(mod_Q)
```

```
##         Q1        Q2        Q3
## -10.33790 -30.12275 -39.25857
```

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.

```
colnames(X2) <- c("setosa", "vericolor", "virginica")
mod_Q2 = lm(y ~ 0 + X2)
unique(predict(mod_Q2, data.frame(X2)))
```

```
## [1] 1.462 4.260 5.552
```

```
mod_Q3 = lm(y ~ 0 + Q)
unique(predict(mod_Q3, data.frame(Q)))
```

```
## [1] 1.462 1.462 4.260 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.

```
Boston <- MASS::Boston
y = MASS::Boston$medv
X = as.matrix(cbind(1, MASS::Boston[, 1: 13]))
n = nrow(X)
p_plus_one = ncol(X)

matrix <- matrix(NA, nrow = p_plus_one, ncol = p_plus_one, dimnames = list(NULL, colnames(X)))

for (j in 1:ncol(matrix)){
  b = array(NA, dim = ncol(matrix))
  X_star = X[,1:j]
  X_star = as.matrix(X_star)
  X_invert = solve(t(X_star) %*% X_star)
  b[1:j] = X_invert %*% t(X_star) %*% y
  matrix[j, ] <- b
}
```

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

The estimates are changing from row to row as you add in more predictors because each as you go through each row, you add in more features/data for the model. Also each of the rows represents a different model.

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

```
RSQ_Values = array(dim = p_plus_one)
y_bar = mean(y)
SST = sum((y - y_bar) ^ 2)
```

```r
for(i in 1:nrow(matrix)){
  b = c(matrix[i, 1:i], rep(0, nrow(matrix) - i))

  y_hat = X %*% b
  SSR = sum((y_hat - y_bar) ^2)
  RSQ = SSR/SST
  RSQ_Values[i] = RSQ
}
```

Is R^2 monotonically increasing? Why?

Yes, because you are increasing the amount of features(fitting parameters) in this model.