Lab 4

Elizabeth McHugh

11:59PM March 10, 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 for the previous linear model with an intercept, X, without using `model.matrix`.

```
X = cbind(1, iris$Species == "versicolor", iris$Species == "virginica")     #using setosa as reference
```

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 $X_\perp$.

```
##Come back when you have time to think through.

dim(H)
dim(X)

I = diag(nrow(H))
X_perp =  (I - H) %*% X  # n x n-(p+1) matrix which will span the space of e   Binding X and X_perp wil

X_bind = cbind(X, X_perp)
X_bind
```

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 %*% y

e = (diag(nrow(iris)) - H) %*% y
```

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

```r
y_bar = mean(y)

SSE = t(e) %*% e
SSR = t(y_hat - y_bar) %*% (y_hat - y_bar)
SST = t(y - y_bar) %*% (y - y_bar)
R_squared = 1 - SSE/SST

expect_equal(SST, SSR + SSE)
```

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
expect_equal(cos(theta)*cos(theta), R_squared)
```

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
proj_1 = ((X[,1] %*% (t(X[,1]))) / as.numeric(t(X[,1]) %*% X[,1])) %*% y
proj_2 = ((X[,2] %*% (t(X[,2]))) / as.numeric(t(X[,2]) %*% X[,2])) %*% y
proj_3 = ((X[,3] %*% (t(X[,3]))) / as.numeric(t(X[,3]) %*% X[,3])) %*% y
proj_y_hat = ((X[,3] %*% (t(X[,3]))) / as.numeric(t(X[,3]) %*% X[,3])) %*% y_hat

#expect_equal(proj_1 + proj_2 + proj_3, y_hat)
#Not equal...
```

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

```r
X_no_int = as.matrix( cbind( as.numeric( iris$Species == "setosa"), as.numeric( iris$Species == "virgin
```

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

```r
H_again = X_no_int %*% solve(t(X_no_int) %*% X_no_int) %*% t(X_no_int)

yhat = H_again %*% y
```

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).

```
expect_equal(H * y, H_again * y)
```

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

```
X_proj = H_again %*% y

expect_equal(X_proj, y_hat)
```

Convert this design matrix into Q, an orthonormal matrix.

```
Q = qr.Q(qr(X))
```

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)

# So far, so good.
```

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?

No, the OLS solutions are not the same for X and Q, nor should they be, as X and Q have different values, and thus certainly different OLS solutions, but should produce the same predictions given the same observation values.

```
mod = lm(Petal.Length ~ Q, iris)
mod
```

```
##
## Call:
## lm(formula = Petal.Length ~ Q, data = iris)
##
## Coefficients:
## (Intercept)           Q1           Q2           Q3
##       3.758           NA        4.347       20.450
```

```
modX = lm(Petal.Length ~ X, iris)
modX
```

```
##
```

```
## Call:
## lm(formula = Petal.Length ~ X, data = iris)
##
## Coefficients:
## (Intercept)           X1           X2           X3
##       1.462           NA        2.798        4.090
```

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.

```
pred_q = predict(mod, data.frame(X))
```

```
## Warning in predict.lm(mod, data.frame(X)): prediction from a rank-deficient fit
## may be misleading
```

```
pred_X = predict(modX, data.frame(X))
```

```
## Warning in predict.lm(modX, data.frame(X)): prediction from a rank-deficient fit
## may be misleading
```

```
expect_equal(pred_q, pred_X)

#Not quite.
```

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.

```
rm(list = ls())

pacman::p_load(MASS)
data(Boston)

X = Boston[ , 1:13]
X_1 = cbind(1, X)

y = Boston[,14]

X_reg = matrix(data = NA, nrow = 14, ncol = 14)

colnames(X_reg) = c(colnames(X_1))

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

  b = array(NA, dim = ncol(X_reg))
  X_reg = as.matrix(X_reg[1, i])
  comp_matrix = solve(t(X_1) %*% X_1)
  b[1, i] = comp_matrix %*% t(X_1) %*% y

  X_reg[i, ] = b
}

X
```

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

Well, in my code they actually didn't… do anything, but had they I imagine this would be due to the fact that X is regressed on different features.

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

```
y_bar = mean(y)
e = y_hat - y
SSE = t(e) %*% e
SSR = t(y_hat - y_bar) %*% (y_hat - y_bar)
SST = t(y - y_bar) %*% (y - y_bar)
R_squared = 1 - SSE/SST
```

Is R^2 monotonically increasing? Why?

I'm not sure, since the regression didn't work out, neither did the R^2 estimations. Though, I don't expect it to necessarily be monotone increasing. As the regression cycles through the factor of regression, I expect R^2 to change, I'm just not quite sure how.