Lab 4

Meihe Liu

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 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))
#no need of tolerance
```

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

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

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

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

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

```r
#sum of trace is the rank
```

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 $X_\perp$.

```r
#ec
```

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
I = diag(nrow(iris))
y = iris$Petal.Length
yhat = H %*% y
e = (I-H) %*% y
t(e) %*% yhat
```

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

```r
head(e)
```

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

```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 = e %*% t(e) this gives n x n matrix with rank 1.
SSE = t(e) %*% e
ybar = mean(y)
SST = t(y - ybar) %*% (y - ybar)
Rsq = 1 - SSE/SST
SSR = t(yhat - ybar) %*% (yhat - ybar)
expect_equal(SSR+SSE,SST)
#var(y)
#var(e)
```

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 - ybar) %*% (yhat - ybar) / sqrt(SST*SSR))
theta = (180/pi)
```

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
#expect_equal(proj1+proj2+proj3, yhat)
#not equal
```

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

```r
X2 <- cbind(as.numeric(iris$Species=="setosa"), iris$Species=="versicolor", iris$Species=="virginica" )
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.

```r
H2 = X2 %*% solve(t(X2) %*% X2) %*% t(X2)
yhat2 = H2 %*% y
unique(yhat)
```

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

```r
unique(yhat2)
```

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

```r
mean(y[iris$Species == "setosa"])
```

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

```r
mean(y[iris$Species == "versicolor"])
```

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

```r
mean(y[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).

```
hat = X2 %*% solve(t(X2) %*% X2) %*% t(X2)

expect_equal(hat,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.

```
proj1 = ((X2[,1] %*% t(X2[,1]))/as.numeric(t(X2[,1]) %*% X2[,1])) %*% y
proj2 = ((X2[,2] %*% t(X2[,2]))/as.numeric(t(X2[,2]) %*% X2[,2])) %*% y
proj3 = ((X2[,3] %*% t(X2[,3]))/as.numeric(t(X2[,3]) %*% X2[,3])) %*% y
yhat = H %*% y
expect_equal(proj1+proj2+proj3 , yhat)
```

Convert this design matrix into Q, an orthonormal matrix.

```
Q = qr.Q(qr(X2))
head(Q)
```

```
##            [,1] [,2] [,3]
## [1,] -0.1414214    0    0
## [2,] -0.1414214    0    0
## [3,] -0.1414214    0    0
## [4,] -0.1414214    0    0
## [5,] -0.1414214    0    0
## [6,] -0.1414214    0    0
```

```
#verification
sum(Q[, 1]^2)
```

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

```
sum(Q[, 2]^2)
```

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

```
sum(Q[, 3]^2)
```

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

```
Q[, 1] %*% Q[, 2]
```

```
##      [,1]
## [1,]    0
```

```r
Q[, 1] %*% Q[, 3]
```

```
##      [,1]
## [1,]    0
```

```r
Q[, 2] %*% Q[, 3]
```

```
##      [,1]
## [1,]    0
```

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
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
yhat_Q = Q %*% t(Q) %*% y
expect_equal(yhat_Q , yhat2)
```

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
mod2 = lm(y~0+Q)
coef(mod2)
```

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

```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
```

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
colnames(X2)<-c("setosa", "versicolor", "virginica")
mod3 = lm(y~0+X2)
unique(predict(mod3, data.frame(X2)))
```

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

```r
mod4 = lm(y~0+Q)
unique(predict(mod4, 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.

```r
rm(list=ls())
Boston <- MASS::Boston
one_vec = rep(1,nrow(Boston))
X = as.matrix(cbind(one_vec,Boston[,1:13]))
y = Boston[,14]
n=nrow(X)
df = ncol(X)
Matrix <- matrix(NA,nrow = df, ncol=df,dimnames = list(NULL,colnames(X)))

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

```
##           one_vec       crim         zn       indus     chas        nox
##  [1,]  22.5328063         NA         NA          NA       NA         NA
##  [2,]  24.0331062 -0.4151903         NA          NA       NA         NA
##  [3,]  22.4856281 -0.3520783 0.11610909          NA       NA         NA
##  [4,]  27.3946468 -0.2486283 0.05850082 -0.41557782       NA         NA
##  [5,]  27.1128031 -0.2287981 0.05928665 -0.44032511 6.894059         NA
##  [6,]  29.4899406 -0.2185190 0.05511047 -0.38348055 7.026223  -5.424659
##  [7,] -17.9546350 -0.1769135 0.02128135 -0.14365267 4.784684  -7.184892
##  [8,] -18.2649261 -0.1727607 0.01421402 -0.13089918 4.840730  -4.357411
##  [9,]   0.8274820 -0.1977868 0.06099257 -0.22573089 4.577598 -14.451531
## [10,]   0.1553915 -0.1780398 0.06095248 -0.21004328 4.536648 -13.342666
## [11,]   2.9907868 -0.1795543 0.07145574 -0.10437742 4.110667 -12.591596
## [12,]  27.1523679 -0.1840321 0.03909990 -0.04232450 3.487528 -22.182110
## [13,]  20.6526280 -0.1599391 0.03887365 -0.02792186 3.216569 -20.484560
## [14,]  36.4594884 -0.1080114 0.04642046  0.02055863 2.686734 -17.766611
##             rm          age       dis         rad          tax    ptratio
##  [1,]       NA           NA        NA          NA           NA         NA
##  [2,]       NA           NA        NA          NA           NA         NA
##  [3,]       NA           NA        NA          NA           NA         NA
##  [4,]       NA           NA        NA          NA           NA         NA
##  [5,]       NA           NA        NA          NA           NA         NA
```

```
## [6,] NA NA NA NA NA NA
## [7,] 7.341586 NA NA NA NA NA
## [8,] 7.386357 -0.0236248493 NA NA NA NA
## [9,] 6.752352 -0.0556354540 -1.760312 NA NA NA
## [10,] 6.791184 -0.0562612189 -1.748296 -0.04529059 NA NA
## [11,] 6.664084 -0.0546675064 -1.727933 0.15926305 -0.01434060 NA
## [12,] 6.075744 -0.0451880522 -1.583852 0.25472196 -0.01221262 -0.9962062
## [13,] 6.123072 -0.0459320518 -1.554912 0.28157503 -0.01173838 -1.0142228
## [14,] 3.809865 0.0006922246 -1.475567 0.30604948 -0.01233459 -0.9527472
## black lstat
## [1,] NA NA
## [2,] NA NA
## [3,] NA NA
## [4,] NA NA
## [5,] NA NA
## [6,] NA NA
## [7,] NA NA
## [8,] NA NA
## [9,] NA NA
## [10,] NA NA
## [11,] NA NA
## [12,] NA NA
## [13,] 0.013620833 NA
## [14,] 0.009311683 -0.5247584
```

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

Because every row is a different model with different number of features i.e. the first row has 0 features with intercept, the second row has one feature with the intercept,the third row has two features with the intercept. . . . We find that the values of the weights of a single features may vary when we change the number of features we fit the model on.

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

```r
Rsq_array = array(dim = 14)
ybar = mean(y)
SST = sum((y-ybar)^2)
for(i in 1:nrow(Matrix)){
  b = c(Matrix[i,1:i], rep(0, nrow(Matrix)-i))
  yhat = X %*% b
  SSR = sum((yhat - ybar)^2)
  Rsq = SSR / SST
  Rsq_array[i] = Rsq
}
Rsq_array
```

```
## [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?

Yes. Beacuse as we are adding more features the model will fit better on data.