Lab 5

Juan D Astudillo

11:59PM March 16, 2019

Load the Boston housing data frame and create the vector y (the median value) and matrix X (all other features) from the data frame. Name the columns the same as Boston except for the first name it "(Intercept)".

```
data(Boston, package = "MASS")
y = Boston$medv
X = as.matrix(cbind(1, Boston[, 1 : 13]))
colnames(X)[1] = "(Intercept)"
```

Run the OLS linear model to get b, the vector of coefficients. Do not use `lm`.

```
b = solve(t(X) %*% X) %*% t(X) %*% y
```

Find the hat matrix for this regression `H` and find its rank. Is this rank expected?

```
H = X %*% solve(t(X) %*% X) %*% t(X)
dim(H)
```

```
## [1] 506 506
```

```
pacman::p_load(Matrix)
rankMatrix(H)
```

```
## [1] 14
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 1.123546e-13
```

Verify this is a projection matrix by verifying the two sufficient conditions. Use the `testthat` library's `expect_equal(matrix1, matrix2, tolerance = 1e-2)`.

```
pacman::p_load(testthat)
expect_equal(H, t(H), tolerance = 1e-2)
expect_equal(H %*% H, H, tolerance = 1e-2)
```

Find the matrix that projects onto the space of residuals `H_comp` and find its rank. Is this rank expected?

```
I = diag(nrow(H))
H_comp = (I - H)
rankMatrix(H_comp)
```

```
## [1] 497
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 1.123546e-13
```

Verify this is a projection matrix by verifying the two sufficient conditions. Use the `testthat` library.

```r
expect_equal(H_comp, t(H_comp), tolerance = 1e-2)
expect_equal(H_comp %*% H_comp, H_comp, tolerance = 1e-2)
```

Calculate $\hat{y}$.

```r
yhat = H %*% y
#yhat
```

Calculate e as the difference of y and $\hat{y}$ and the projection onto the space of the residuals. Verify the two means of calculating the residuals provide the same results.

```r
e = y - yhat
e_2 = H_comp %*% y
expect_equal(e, e_2)
```

Calculate R^2 and RMSE.

```r
sse = sum(e^2)
sst = sum((y - mean(y))^2)

Rsquared = 1 - sse / sst
Rsquared
```

```
## [1] 0.7406427
```

```r
mse = sse / (nrow(X) - ncol(X))
rmse = sqrt(mse) #rmse is standard deviation of errors
rmse
```

```
## [1] 4.745298
```

Verify $\hat{y}$ and e are orthogonal.

```r
t(e) %*% yhat
```

```
##               [,1]
## [1,] -4.991142e-08
```

Verify $\hat{y} - \bar{y}$ and e are orthogonal.

```r
t(e) %*% (yhat - mean(y))
```

```
##              [,1]
## [1,] 2.832162e-09
```

Find the cosine-squared of $y - \bar{y}$ and $\hat{y} - \bar{y}$ and verify it is the same as R^2.

```r
y_minus_y_bar = y - mean(y)
yhat_minus_y_bar = yhat - mean(y)
len_y_minus_y_bar = sqrt(  sum(y_minus_y_bar^2)  )
len_yhat_minus_y_bar = sqrt(  sum(yhat_minus_y_bar^2)  )

theta = acos( (t(y_minus_y_bar) %*% yhat_minus_y_bar) / (len_y_minus_y_bar * len_yhat_minus_y_bar) )
#cos_theta * (180 / pi)
cos_theta_sqrd = cos(theta)^2
cos_theta_sqrd
```

```
##           [,1]
## [1,] 0.7406427
```

Verify the sum of squares identity which we learned was due to the Pythagorean Theorem (applies since the projection is specifically orthogonal).

```
len_y_minus_y_bar^2 - len_yhat_minus_y_bar^2 - sse
```

```
## [1] 5.666152e-09
```

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.

```
M = matrix(NA, nrow = ncol(X), ncol = ncol(X))
colnames(M) = colnames(X)
X_j = X[, 1, drop = FALSE]
b = solve(t(X_j) %*% X_j) %*% t(X_j) %*% y
M[1, 1] = b
X_j_2 = X[ , 1:2]
b = solve(t(X_j_2) %*% X_j_2) %*% t(X_j_2) %*% y
b
```

```
##                    [,1]
## (Intercept) 24.0331062
## crim        -0.4151903
```

```
for(j in 1 : ncol(M)){
  X_j = X[, 1 : j, drop = FALSE]
  b = solve(t(X_j) %*% X_j) %*% t(X_j) %*% y
  M[j, 1:j] = b
}
round(M, 2)
```

```
##       (Intercept)  crim   zn indus chas    nox   rm   age   dis   rad
##  [1,]       22.53    NA   NA    NA   NA     NA   NA    NA    NA    NA
##  [2,]       24.03 -0.42   NA    NA   NA     NA   NA    NA    NA    NA
##  [3,]       22.49 -0.35 0.12    NA   NA     NA   NA    NA    NA    NA
##  [4,]       27.39 -0.25 0.06 -0.42   NA     NA   NA    NA    NA    NA
##  [5,]       27.11 -0.23 0.06 -0.44 6.89     NA   NA    NA    NA    NA
##  [6,]       29.49 -0.22 0.06 -0.38 7.03  -5.42   NA    NA    NA    NA
##  [7,]      -17.95 -0.18 0.02 -0.14 4.78  -7.18 7.34    NA    NA    NA
##  [8,]      -18.26 -0.17 0.01 -0.13 4.84  -4.36 7.39 -0.02    NA    NA
##  [9,]        0.83 -0.20 0.06 -0.23 4.58 -14.45 6.75 -0.06 -1.76    NA
## [10,]        0.16 -0.18 0.06 -0.21 4.54 -13.34 6.79 -0.06 -1.75 -0.05
## [11,]        2.99 -0.18 0.07 -0.10 4.11 -12.59 6.66 -0.05 -1.73  0.16
## [12,]       27.15 -0.18 0.04 -0.04 3.49 -22.18 6.08 -0.05 -1.58  0.25
## [13,]       20.65 -0.16 0.04 -0.03 3.22 -20.48 6.12 -0.05 -1.55  0.28
## [14,]       36.46 -0.11 0.05  0.02 2.69 -17.77 3.81  0.00 -1.48  0.31
##         tax ptratio black lstat
##  [1,]    NA      NA    NA    NA
##  [2,]    NA      NA    NA    NA
##  [3,]    NA      NA    NA    NA
##  [4,]    NA      NA    NA    NA
##  [5,]    NA      NA    NA    NA
##  [6,]    NA      NA    NA    NA
##  [7,]    NA      NA    NA    NA
```

```
##  [8,]    NA      NA    NA    NA
##  [9,]    NA      NA    NA    NA
## [10,]    NA      NA    NA    NA
## [11,] -0.01      NA    NA    NA
## [12,] -0.01   -1.00    NA    NA
## [13,] -0.01   -1.01  0.01    NA
## [14,] -0.01   -0.95  0.01 -0.52
```

Examine this matrix. Why are the estimates changing from row to row as you add in more predictors? As we add more predictors the estimates change because of the different associations between the variables, its effect makes the stimators to chance as we add more and more.

Clear the workspace and load the diamonds dataset.

```r
pacman::p_load(ggplot2)
data(diamonds, package = "ggplot2")
```

Extract y, the price variable and "c", the nominal variable "color" as vectors.

```r
summary(diamonds)
```

```
##      carat               cut        color       clarity
##  Min.   :0.2000   Fair     : 1610   D: 6775   SI1    :13065
##  1st Qu.:0.4000   Good     : 4906   E: 9797   VS2    :12258
##  Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194
##  Mean   :0.7979   Premium  :13791   G:11292   VS1    : 8171
##  3rd Qu.:1.0400   Ideal    :21551   H: 8304   VVS2   : 5066
##  Max.   :5.0100                     I: 5422   VVS1   : 3655
##                                     J: 2808   (Other): 2531
##      depth           table           price             x
##  Min.   :43.00   Min.   :43.00   Min.   :  326   Min.   : 0.000
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710
##  Median :61.80   Median :57.00   Median : 2401   Median : 5.700
##  Mean   :61.75   Mean   :57.46   Mean   : 3933   Mean   : 5.731
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540
##  Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :10.740
##
##        y                z
##  Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 4.720   1st Qu.: 2.910
##  Median : 5.710   Median : 3.530
##  Mean   : 5.735   Mean   : 3.539
##  3rd Qu.: 6.540   3rd Qu.: 4.040
##  Max.   :58.900   Max.   :31.800
##
```

```r
y = diamonds$price
c = diamonds$color
table(c)
```

```
## c
##     D     E     F     G     H     I     J
##  6775  9797  9542 11292  8304  5422  2808
```

Convert the "c" vector to X which contains an intercept and an appropriate number of dummies. Let the color G be the refernce category as it is the modal color. Name the columns of X appropriately. The first should be "(Intercept)". Delete c.

```r
X = rep(1, nrow(diamonds))
X = cbind(X, diamonds$color == 'D')
X = cbind(X, diamonds$color == 'E')
X = cbind(X, diamonds$color == 'F')
X = cbind(X, diamonds$color == 'H')
X = cbind(X, diamonds$color == 'I')
X = cbind(X, diamonds$color == 'J')
colnames(X) = c("Intercept", "is_D", "is_E", "is_F", "is_H", "is_I", "is_J")
head(X)
```

```
##      Intercept is_D is_E is_F is_H is_I is_J
## [1,]         1    0    1    0    0    0    0
## [2,]         1    0    1    0    0    0    0
## [3,]         1    0    1    0    0    0    0
## [4,]         1    0    0    0    0    1    0
## [5,]         1    0    0    0    0    0    1
## [6,]         1    0    0    0    0    0    1
```

Repeat the iterative exercise above we did for Boston here.

```r
b = solve(t(X) %*% X) %*% t(X) %*% y


M = matrix(NA, nrow = ncol(X), ncol = ncol(X))
colnames(M) = colnames(X)
X_j = X[, 1, drop = FALSE]
b = solve(t(X_j) %*% X_j) %*% t(X_j) %*% y
M[1, 1] = b
X_j_2 = X[ , 1:2]
b = solve(t(X_j_2) %*% X_j_2) %*% t(X_j_2) %*% y
b
```

```
##                [,1]
## Intercept 4042.3784
## is_D      -872.4243
```

```r
for(j in 1 : ncol(M)){
  X_j = X[, 1 : j, drop = FALSE]
  b = solve(t(X_j) %*% X_j) %*% t(X_j) %*% y
  M[j, 1:j] = b
}
round(M, 2)
```

```
##      Intercept     is_D     is_E    is_F   is_H    is_I    is_J
## [1,]   3932.80       NA       NA      NA     NA      NA      NA
## [2,]   4042.38  -872.42       NA      NA     NA      NA      NA
## [3,]   4295.54 -1125.59 -1218.79      NA     NA      NA      NA
## [4,]   4491.23 -1321.28 -1414.48 -766.34     NA      NA      NA
## [5,]   4493.17 -1323.22 -1416.42 -768.28  -6.50      NA      NA
## [6,]   4262.94 -1092.99 -1186.19 -538.06 223.72  828.93      NA
## [7,]   3999.14  -829.18  -922.38 -274.25 487.53 1092.74 1324.68
```

Why didn't the estimates change as we added more and more features?

TO-DO

Create a vector y by simulating $n = 100$ standard iid normals. Create a matrix of size 100 x 2 and populate the first column by all ones (for the intercept) and the second column by 100 standard iid normals. Find the R^2 of an OLS regression of `y ~ X`. Use matrix algebra.

```
y = rnorm(100, mean = 0, sd = 1)
intercept = rep(1, 100)
X = cbind(intercept, rnorm(100, mean = 0, sd = 1))
head(X)
```

```
##      intercept
## [1,]         1 -1.66074889
## [2,]         1  0.51295296
## [3,]         1 -1.67630958
## [4,]         1  0.09649496
## [5,]         1  1.37781493
## [6,]         1  1.09246027
```

```
H = X %*% solve(t(X) %*% X) %*% t(X)
yhat = H %*% y


e = y - yhat

sse = sum(e^2)
sst = sum((y - mean(y))^2)

Rsquared = 1 - sse / sst
Rsquared
```

```
## [1] 0.008503835
```

```
reg = lm(y ~ X )
```

from the last problem. Find the R^2 of an OLS regression of `y ~ X`. You can use the `summary` function of an `lm` model.

Write a for loop to each time bind a new column of 100 standard iid normals to the matrix X and find the R^2 each time until the number of columns is 100. Create a vector to save all R^2's. What happened??

```
n = 100
N = 100 - 2
v = rep(NA, 98)
for (i in 1 : N ){
  colm = rnorm(100, mean = 0, sd = 1)
  X = cbind(X, colm)
  v[i] = summary(lm(y ~ X))$r.squared
}
#head(X)
v
```

```
##  [1] 0.01383835 0.01385991 0.03577735 0.04105826 0.04549885 0.05058290
##  [7] 0.05100189 0.05105692 0.06681203 0.08915309 0.08940169 0.09102598
## [13] 0.09780512 0.10047727 0.11274099 0.15662943 0.15751082 0.15756278
## [19] 0.15758803 0.16113063 0.16183604 0.16597154 0.16608310 0.18643102
## [25] 0.21324908 0.21433170 0.22104419 0.23411068 0.25626521 0.26335643
## [31] 0.27022129 0.30848908 0.31191956 0.32994215 0.33951773 0.33966708
```

```
## [37] 0.34962502 0.34993271 0.35946771 0.36687123 0.37567581 0.37756223
## [43] 0.38237060 0.41564266 0.44811207 0.44879482 0.45013788 0.45989823
## [49] 0.48033404 0.48054209 0.48122316 0.48815230 0.49153052 0.51384314
## [55] 0.51554800 0.55312335 0.55532299 0.55590615 0.56768176 0.57594817
## [61] 0.57709206 0.60321189 0.60899763 0.61329779 0.63585036 0.66232084
## [67] 0.66232758 0.69135509 0.69165978 0.69635956 0.75480313 0.76186222
## [73] 0.76497154 0.77540703 0.78487752 0.78539204 0.78543561 0.78775387
## [79] 0.79749909 0.80248751 0.81382792 0.84392093 0.84417039 0.88196292
## [85] 0.88614137 0.92665087 0.93847845 0.94239110 0.94451417 0.94597553
## [91] 0.96298585 0.96749943 0.96947658 0.96951561 0.97237536 0.97459568
## [97] 0.97584503 1.00000000
```

```r
dim(X)
```

```
## [1] 100 100
```

Add one final column to X to bring the number of columns to 101. Then try to compute R^2. What happens and why?

```r
newX = cbind(X, rnorm(100, mean = 0, sd = 1) )
summary(lm(y ~ newX))$r.squared
```

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

As we add more and more p features to the matrix R^2 becomes 1. Little by little p+1 gets closer and closer to n. We are overfitting.