Homework 1: Matrix Algebra

Rick Farouni

August 22, 2016

Contents

Introduction

This section shows how to perform some matrix operations in R.

Creating a Matrix

In R, the basic data structure is a vector. A matrix is basically a vector with a dimension attribute, dim() . To create a matrix, we use the matrix() function. This function has the following arguments:

```
args(matrix)
```

```
## function (data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
## NULL
```

```
# fill matrix by column (default)
```

Here we populate the matrix with a vector

```
v <- c(1, 2, 3, 4, 5, 6)
v
```

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

```
M <-  matrix(v , nrow = 2, ncol = 3)
M
```

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

Matrix operations

1. Matrix transpose $\mathbf{M}^T$

```
Mt <-  t(M)
Mt
```

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

2. Matrix multiplication $\mathbf{M}\mathbf{M}^T$

```
MMt <-  M %*% Mt
MMt
```

```
##      [,1] [,2]
## [1,]   35   44
## [2,]   44   56
```

3. Matrix addition $\mathbf{M} + \mathbf{M}$

```
dM <-  M + M
dM
```

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    4    8   12
```

4. Matrix inverse $(\mathbf{M}\mathbf{M}^T)^{-1}$

```
MMt_inverse <-  solve(MMt)
MMt_inverse
```

```
##      [,1] [,2]
## [1,]  2.3 -1.8
## [2,] -1.8  1.5
```

Questions

Consider the simple regression model

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \text{ for } i = 1, 2, \ldots, n$$

In matrix form, the regression model can be represented as

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ \vdots & \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

or more concisely as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

It can be shown that the estimated parameters are

$$\hat{\beta} = \left(\mathbf{X}^{\mathsf{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{y}$$

and the fitted values are

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}\left(\mathbf{X}^{\mathsf{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{y}$$

The next three question involves working with the following data:

```
x <- c(10, 8, 13, 9, 11, 14, 6 , 4, 12, 7 ,5 )
y <-  c(8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82, 5.68 )
```

Question 1:

Compute the vector of estimated parameters $\hat{\boldsymbol{\beta}}$ for the simple regression problem given the data provided above.

Solution:

First we need to the column vector of ones. The cbind function binds the two vectors into a matrix.

```
X <- cbind(1, x)
Xt <- t(X)
beta_hat <- solve(Xt %*% X) %*% Xt %*% y
beta_hat
```

```
##   [,1]
##    3.0
## x  0.5
```

Let's compare our solution with the built-in linear model fit function. The results are identical.

```
model_fit <- lm.fit(X, y)
cbind(beta_hat, lmfit = model_fit$coefficients)
```

```
##        lmfit
##   3.0   3.0
## x 0.5   0.5
```

Question 2:

Obtain the fitted values, the vector $\hat{\mathbf{y}}$, by using matrix multiplication

Solution:

```r
y_hat <- X %*% beta_hat
y_hat
```

```
##       [,1]
##  [1,]  8.0
##  [2,]  7.0
##  [3,]  9.5
##  [4,]  7.5
##  [5,]  8.5
##  [6,] 10.0
##  [7,]  6.0
##  [8,]  5.0
##  [9,]  9.0
## [10,]  6.5
## [11,]  5.5
```

let's compare

```r
cbind(y_hat, lmfit = model_fit$fitted.values)
```

```
##            lmfit
##  [1,]  8.0   8.0
##  [2,]  7.0   7.0
##  [3,]  9.5   9.5
##  [4,]  7.5   7.5
##  [5,]  8.5   8.5
##  [6,] 10.0  10.0
##  [7,]  6.0   6.0
##  [8,]  5.0   5.0
##  [9,]  9.0   9.0
## [10,]  6.5   6.5
## [11,]  5.5   5.5
```

Question 3:

Obtain the vector of residuals $\hat{\boldsymbol{\epsilon}}$. Note that the vector of residuals has a hat on it, whereas the vector of errors $\boldsymbol{\epsilon}$ doesn't. Now you know the difference!

Solution:

```r
epsilon_hat <- y - y_hat
epsilon_hat
```

```
##         [,1]
##  [1,]  0.039
##  [2,] -0.051
##  [3,] -1.921
##  [4,]  1.309
##  [5,] -0.171
##  [6,] -0.041
```

```
##  [7,]  1.239
##  [8,] -0.740
##  [9,]  1.839
## [10,] -1.681
## [11,]  0.179
```

let's compare

```r
cbind(epsilon_hat, lmfit = model_fit$residuals)
```

```
##               lmfit
##  [1,]  0.039  0.039
##  [2,] -0.051 -0.051
##  [3,] -1.921 -1.921
##  [4,]  1.309  1.309
##  [5,] -0.171 -0.171
##  [6,] -0.041 -0.041
##  [7,]  1.239  1.239
##  [8,] -0.740 -0.740
##  [9,]  1.839  1.839
## [10,] -1.681 -1.681
## [11,]  0.179  0.179
```

Note: MSE (mean squared error) is just

```r
mean(epsilon_hat ^ 2)
```

```
## [1] 1.3
```

Question 4:

When you ever see a matrix, think *transformation*! In the equation for obtaining the fitted values above, we see that $\mathbf{y}$ gets multiplied (transformed) by several matrices to give us $\hat{\mathbf{y}}$. Let's denote the result of these multiplications by

$$\mathbf{H} = \mathbf{X}\left(\mathbf{X}^\intercal\mathbf{X}\right)^{-1}\mathbf{X}^\intercal$$

Obtain $\mathbf{H}$ for the data given above.

Solution:

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

```
##       [,1]  [,2]   [,3]  [,4]    [,5]    [,6]    [,7]   [,8]   [,9]
##  [1,] 0.100 0.082  0.127 0.091  1.1e-01  1.4e-01  0.0636  0.045  0.1182
##  [2,] 0.082 0.100  0.055 0.091  7.3e-02  4.5e-02  0.1182  0.136  0.0636
##  [3,] 0.127 0.055  0.236 0.091  1.6e-01  2.7e-01 -0.0182 -0.091  0.2000
##  [4,] 0.091 0.091  0.091 0.091  9.1e-02  9.1e-02  0.0909  0.091  0.0909
##  [5,] 0.109 0.073  0.164 0.091  1.3e-01  1.8e-01  0.0364  0.000  0.1455
```

```
##  [6,] 0.136 0.045  0.273 0.091  1.8e-01  3.2e-01 -0.0455 -0.136  0.2273
##  [7,] 0.064 0.118 -0.018 0.091  3.6e-02 -4.5e-02  0.1727  0.227  0.0091
##  [8,] 0.045 0.136 -0.091 0.091 -5.6e-17 -1.4e-01  0.2273  0.318 -0.0455
##  [9,] 0.118 0.064  0.200 0.091  1.5e-01  2.3e-01  0.0091 -0.045  0.1727
## [10,] 0.073 0.109  0.018 0.091  5.5e-02 -1.1e-16  0.1455  0.182  0.0364
## [11,] 0.055 0.127 -0.055 0.091  1.8e-02 -9.1e-02  0.2000  0.273 -0.0182
##       [,10]  [,11]
##  [1,] 0.073  0.055
##  [2,] 0.109  0.127
##  [3,] 0.018 -0.055
##  [4,] 0.091  0.091
##  [5,] 0.055  0.018
##  [6,] 0.000 -0.091
##  [7,] 0.145  0.200
##  [8,] 0.182  0.273
##  [9,] 0.036 -0.018
## [10,] 0.127  0.164
## [11,] 0.164  0.236
```

Now play with the interactive visualization here and try to figure out what kind of tranformation $\hat{\mathbf{H}}$ accomplished. The answer is one word only. Think about the dimension of $\mathbf{H}$ in relation with the dimension of the data vector $\mathbf{y}$

Solution:

H is a projection matrix with dimensions 11 by 11, which is also the number of observations we have. The matrix projects the data into points residing on a two-dimensional plane "determined" (spanned by the columns) of $\mathbf{X}$

By the way, the data is taken from Anscombe's_quartet.