Assign_1(Vectors and Matrices)

Salma Elshahawy

1/29/2020

```r
if (!require("pacman")) install.packages("pacman")
pacman::p_load(knitr, rmdformats, REdaS)
```

Problem set-1

You can think of vectors representing many dimensions of related information. For instance, Netflix might store all the ratings a user gives to movies in a vector. This is clearly a vector of very large dimensions (in the millions) and very sparse as the user might have rated only a few movies. Similarly, Amazon might store the items purchased by a user in a vector, with each slot or dimension representing a unique product and the value of the slot, the number of such items the user bought. One task that is frequently done in these settings is to find similarities between users. And, we can use dot-product between vectors to do just that. As you know, the dot-product is proportional to the length of two vectors and to the angle between them. In fact, the dot-product between two vectors, normalized by their lengths is called as the cosine distance and is frequently used in recommendation engines.

1. Calculate the dot product u.v where u = [0.5, 0.5] and v = [3, -4]

```r
u <- c(0.5, 0.5)
v <- c(3, -4)
dot_prod = u %*% v
dot_prod
```

```
     [,1]
[1,] -0.5
```

2. What are the lengths of u and v? Please note that the mathematical notion of the

length of a vector is not the same as a computer science definition.

```r
v_length <- sqrt((v[1]) * (v[1]) + (v[2]) * (v[2]))
v_length
```

```
[1] 5
```

```r
u_length <- sqrt((u[1]) * (u[1]) + (u[2]) * (u[2]))
u_length
```

```
[1] 0.7071068
```

3. What is the linear combination: 3u - 2v?

```
linear_comb <- (3 * u) - (2 * v)
linear_comb
```

```
[1] -4.5  9.5
```

4. What is the angle between u and v?

```
cos_theta <- dot_prod/(u_length * v_length)
rad2deg(acos(cos_theta))
```

```
        [,1]
[1,] 98.1301
```

Problem set-2

Set up a system of equations with 3 variables and 3 constraints and solve for x. Please write a function in R that will take two variables (matrix A & constraint vector b) and solve using elimination. Your function should produce the right answer for the system of equations for any 3-variable, 3-equation system. You don't have to worry about degenerate cases and can safely assume that the function will only be tested with a system of equations that has a solution. Please note that you do have to worry about zero pivots, though. Please note that you should not use the built-in function solve to solve this system or use matrix inverses. The approach that you should employ is to construct an Upper Triangular Matrix and then back-substitute to get the solution. Alternatively, you can augment the matrix A with vector b and jointly apply the Gauss Jordan elimination procedure.

Please test it with the system below and it should produce a solution x = $[-1.55, -0.32, 0.95]$

$$\begin{bmatrix} 1 & 1 & 3 \\ 2 & -1 & 5 \\ -1 & -2 & 4 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 6 \end{bmatrix}$$

```
calculate_gauss <- function(a, b) {
    n <- nrow(a)
    # search for the position of the maximum element in the matrix for each
    # column diagonally.
    for (i in seq_len(n - 1)) {
        j <- which.max(a[i:n, i]) + i - 1  # pointer to the new index of max. element in a column
        if (j != i) {
            a[c(i, j), i:n] <- a[c(j, i), i:n]
            # swap the two elements
            b[c(i, j), ] <- b[c(j, i), ]
        }
        # find the multiplier to eliminate - iterate over the submatrix
        k <- seq(i + 1, n)
        for (j in k) {
            # find the multiplier
            s <- a[[j, i]]/a[[i, i]]
            # multiply s to the 2nd row then subtract from the first -> substitute in
            # the second
```

```r
            a[j, k] <- a[j, k] - s * a[i, k]
            b[j, ] <- b[j, ] - s * b[i, ]
        }
    }

    # backword solve
    for (i in seq(n, 1)) {
        if (i < n) {
            for (j in seq(i + 1, n)) {
                b[i, ] <- b[i, ] - a[[i, j]] * b[j, ]
            }
        }
        b[i, ] <- b[i, ]/a[[i, i]]
    }

    return(x = b)
}
a <- matrix(c(1, 2, -1, 1, -1, -2, 3, 5, 4), nrow = 3, ncol = 3)
b <- matrix(c(1, 2, 6), nrow = 3, ncol = 1)

val <- calculate_gauss(a, b)
val
```

```
           [,1]
[1,] -1.5454545
[2,] -0.3181818
[3,]  0.9545455
```