Vishal_Assignment1

Vishal Arora

September 1, 2019

ASSIGNMENT 1 : IS 605 FUNDAMENTALS OF COMPUTATIONAL MATHEMATICS

Question 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

nd 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.

a) Calculate the dot product u:v where u = [0.5, 0.5] and v = [3,-4]

```
u <- c(0.5,0.5)
v <- c(3,-4)

dot_product <- dot(u,v)
dot_product
```

```
## [1] -0.5
```

b) What are the lengths of u and v?

```
length_u <- sqrt(dot(u,u))
length_u
```

```
## [1] 0.7071068
```

```
length_v <- sqrt(dot(v,v))
length_v
```

```
## [1] 5
```

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

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

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

d) What is the angle between u and v?

Hint angle <- acos((dot product of vectors)/((length of vector a) * (length of vector b)))

```
  angle <- acos(dot_product/(length_u * length_v))

angle
```

```
## [1] 1.712693
```

```
rad2deg(angle)
```

```
## [1] 98.1301
```

Question 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.

```
elimination_method <- function(A, b) {
  # vector to store the final outputs.
  x = numeric(3) # size of the vector
  # Combine into one matrix
  comb_matrix = cbind(A, b)

  # 1 in top left pivot
  if (comb_matrix[1,1] == 0) {
    #  switch with row that has non-zero in first column
    if (comb_matrix[2,2] != 0) {
    # switch rows 1 and 2 as row 2 has a non-zero in column 1
    comb_matrix = comb_matrix[c(2,1,3),]
    } else {
    # switch rows 3 and 1 since row 3 has a non-zero
      comb_matrix = comb_matrix[c(3,2,1),]
    }
  }  else if (comb_matrix[1,1] != 1) {
    # else divide 1st row by row 1 column 1 to get 1 in pivot
    comb_matrix[1,] <- comb_matrix[1,] / comb_matrix[1,1]
  }
  # Zero other entries in column 1
  if (comb_matrix[2,1] !=0) {
    # use value of row 2 column 1 for vector addition to zero the row
    vec <- comb_matrix[2,1] * comb_matrix[1,]
    # reduce row 2 using the new vec object
    comb_matrix[2,] <- -comb_matrix[2,] + vec
  }
  if (comb_matrix[3,1] !=0) {
    # use value of row 3 column 1 for vector addition to zero second row
    vec <- comb_matrix[3,1] * comb_matrix[1,]
```

```
    # now reduce row 2 using the vec
    comb_matrix[3,] <- -comb_matrix[3,] + vec
  }

  # check if row 2 column 2 are 0. If no, switch for row 3 to get non-zero
  if (comb_matrix[2,2] == 0) {
    # switch rows 2 and 3 as row 2 has 0 in column 2
    comb_matrix = comb_matrix[c(1,3,2),]
  }

  # divide second row by row 2 column 2 to get 1 in pivot
  comb_matrix[2,] <- comb_matrix[2,] / comb_matrix[2,2]
  # use value of row 3 column 2 for vector addition to zero row 3 column 2
  vec <- comb_matrix[3,2] * comb_matrix[2,]
  # now reduce row 3 using vec
  comb_matrix[3,] <- -comb_matrix[3,] + vec
  # solve for x with back subsitution
  x[3] <- comb_matrix[3,4] / comb_matrix[3,3]
  x[2] <- (comb_matrix[2,4] - comb_matrix[2,3]*x[3]) / comb_matrix[2,2]
  x[1] <- (comb_matrix[1,4] - comb_matrix[1,2]*x[2] - comb_matrix[1,3]*x[3]) / comb_matrix[1,1]


  result <- rbind(x[1], x[2], x[3])

  return (result)
}

matrix_A <- matrix(
    c(1, 2, -1, 1, -1, -2, 3, 5, 4),
    nrow=3,
    ncol=3)
constraint_B <- matrix(
    c(1, 2, 6),
    nrow=3,
    ncol=1)

# Test the output
elimination_method(matrix_A, constraint_B)
```

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