Mathematical Foundations of Data Science
Assignment 4

Trimester 1, 2023

Chang Dong (a1807402)

Note: $E_{ij}(k)$ is the math symbol refers to Elememtary Row Transformation.

E_{ij}: Exchange two rows.
$E_i(k)$: Let row i times k.
$E_{ij}(k)$: Let k times row j add to row i.

1. A sports team plays in a competition that awards 3 points for a win, 1 point for a draw, and 0 points for a loss. For every draw, players are allocated 3 hours of extra training. For every loss, players are allocated 5 hours of extra training. At the end of the season, the team has 73 points, and has won four times as many games as they have lost. Its players were allocated 64 hours of extra training. Formulate a system of equations that would be used to calculate the number of wins, draws, and losses the team had during the season. Express your answer in matrix form $Ax = b$, clearly defining A, x, and b. Note, this question involves formulation only. You do not need to solve the system of equations.

Answer:

In this context, x_1 represents the number of wins, x_2 denotes the number of draws, and x_3 corresponds to the number of losses.

At the end of the season, the team has 73 points:

$3 \cdot x_1 + 1 \cdot x_2 + 0 \cdot x_3 = 73$

has won four times as many games as they have lost:

$x_1 = 4 \cdot x_3$

and its players were allocated 64 hours of extra training:

$3 \cdot x_2 + 5 \cdot x_3 = 64$

So:

$$Ax = b$$

$$\Longleftrightarrow \begin{bmatrix} 3 & 1 & 0 \\ 1 & 0 & -4 \\ 0 & 3 & 5 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 73 \\ 0 \\ 64 \end{bmatrix}$$

$$where\ A = \begin{bmatrix} 3 & 1 & 0 \\ 1 & 0 & -4 \\ 0 & 3 & 5 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad b = \begin{bmatrix} 73 \\ 0 \\ 64 \end{bmatrix}$$

2. Consider the following system of linear equations, in the variables x, y, and z, with some unknown parameter δ.

$$x + 2y = 6$$
$$x + \delta y = 5$$
$$y + \delta z = 4$$

(a) Write this system in matrix form $Ax = b$, clearly defining A, x, and b.

Answer:

$Ax = b$

$$\Longleftrightarrow \begin{bmatrix} 1 & 2 & 0 \\ 1 & \delta & 0 \\ 0 & 1 & \delta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 6 \\ 5 \\ 4 \end{bmatrix}$$

$$where\ A = \begin{bmatrix} 1 & 2 & 0 \\ 1 & \delta & 0 \\ 0 & 1 & \delta \end{bmatrix},\quad x = \begin{bmatrix} x \\ y \\ z \end{bmatrix},\quad b = \begin{bmatrix} 6 \\ 5 \\ 4 \end{bmatrix}$$

(b) Using matrix determinants, find the values of δ for which this system of equations has no solution.

Answer:

If $|A| \neq 0$, Ax = b should have one specific solution.

If $|A| = 0$, Ax = b can have two possible cases:

Case 1: have no solution;

Case 2: have infinite solution.

$$|A| = \begin{vmatrix} 1 & 2 & 0 \\ 1 & \delta & 0 \\ 0 & 1 & \delta \end{vmatrix} = 1 \cdot \begin{vmatrix} \delta & 0 \\ 1 & \delta \end{vmatrix} - 2 \cdot \begin{vmatrix} 1 & 0 \\ 0 & \delta \end{vmatrix} = \delta^2 - 2\delta = \delta(\delta - 2) = 0$$

In this case: $\delta = 0\ or\ 2$

when $\delta = 0$

$x + 2y = 6, x = 5, y = 4$

These three equations can not be true at the same time($x + 2y = 6 \neq 13$), system has no solution. So δ can be 0.(Case 1)

when $\delta = 2$

$x + 2y = 6, x + 2y = 5, y + 2z = 4$

These three equations can not be true at the same time($x + 2y = 6 \neq 5$), system has no solution. So δ can be 2.(Case 1)

(c) If there is a solution with $z = 0$, what are the possible values of δ?

Answer:

To solve that in which case this system of equations has no solution, we can expand the matrix into the form:

$$[A|b] = \left[\begin{array}{ccc|c} 1 & 2 & 0 & 6 \\ 1 & \delta & 0 & 5 \\ 0 & 1 & \delta & 4 \end{array}\right] \xrightarrow{E_{21}(-1)} \left[\begin{array}{ccc|c} 1 & 2 & 0 & 6 \\ 0 & \delta-2 & 0 & -1 \\ 0 & 1 & \delta & 4 \end{array}\right] \xrightarrow{E_2(\frac{1}{\delta-2})(Assume\ \delta\neq2)}$$

$$\left[\begin{array}{ccc|c} 1 & 2 & 0 & 6 \\ 0 & 1 & 0 & -\frac{1}{\delta-2} \\ 0 & 1 & \delta & 4 \end{array}\right] \xrightarrow{E_{32}(-1)} \left[\begin{array}{ccc|c} 1 & 2 & 0 & 6 \\ 0 & 1 & 0 & -\frac{1}{\delta-2} \\ 0 & 0 & \delta & 4+\frac{1}{\delta-2} \end{array}\right]$$

According the the question, we have: $\delta \cdot z = \delta \cdot 0 = 4 + \frac{1}{\delta-2}$

So: $\delta = \frac{7}{4}$ ($\delta \neq 2$, assumption is true)

(d) Can the system of equations have a solution if $y = 0$? Justify your answer.

Answer:

$$[A|b] \xrightarrow{E_{21}(-1)} \left[\begin{array}{ccc|c} 1 & 2 & 0 & 6 \\ 0 & \delta-2 & 0 & -1 \\ 0 & 1 & \delta & 4 \end{array}\right]$$

According the the question, we should let: $(\delta - 2) \cdot y = (\delta - 2) \cdot 0 = -1$

While $(\delta - 2) \cdot 0 = 0 \neq -1 (\forall\ \delta\ \in\ R)$, so the system of equations have no solution if $y = 0$

3. Consider the following augmented matrix $[A|b]$:

$$\left[\begin{array}{cccc|c} 1 & 1 & 1 & 2 & 2 \\ 1 & 1 & -1 & 2 & -2 \\ 0 & 0 & 1 & 1 & 5 \end{array}\right]$$

Using elementary row operations, reduce this matrix to reduced row echelon form and determine the set of solutions corresponding to the system of linear equations $Ax = b$, where $x = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix}$. You must show all working, including stating the row operations that you use.

Answer:

$$[A|b] = \left[\begin{array}{cccc|c} 1 & 1 & 1 & 2 & 2 \\ 1 & 1 & -1 & 2 & -2 \\ 0 & 0 & 1 & 1 & 5 \end{array}\right] \xrightarrow{E_{21}(-1)} \left[\begin{array}{cccc|c} 1 & 1 & 1 & 2 & 2 \\ 0 & 0 & -2 & 0 & -4 \\ 0 & 0 & 1 & 1 & 5 \end{array}\right]$$

$$\xrightarrow{E_2(-\frac{1}{2})} \left[\begin{array}{cccc|c} 1 & 1 & 1 & 2 & 2 \\ 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 & 5 \end{array}\right] \xrightarrow{E_{12}(-1)} \left[\begin{array}{cccc|c} 1 & 1 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 & 5 \end{array}\right]$$

$$\xrightarrow{E_{32}(-1)} \left[\begin{array}{cccc|c} 1 & 1 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 & 3 \end{array}\right] \xrightarrow{E_{13}(-2)} \left[\begin{array}{cccc|c} 1 & 1 & 0 & 0 & -6 \\ 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 & 3 \end{array}\right]$$

So:
$z = 3$
$y = 2$
$w + x = -6,$

let $w = t$, then we have $x = -6 - t$ $(\forall\, t \in R)$

The solution set is $x = [w, x, y, z]^T = [t, -6 - t, 2, 3]^T$ **(**$\forall\, t \in R$**)**

MATHS7027 Mathematical Foundations of Data Science

Assignment 4 - Question 4

In this question, we will explore some properties of the transpose and the inverse of a matrix using `numpy` .

First, we need to import `numpy` .

In [1]:
```
import numpy as np
```

4(a)

We can use `numpy` to generate random numbers with the function `np.random.rand()` (to read about this function, [click here (https://numpy.org/doc/stable/reference/random/generated/numpy.random.rand.html)](https://numpy.org/doc/stable/reference/random/generated/numpy.random.rand.html)).

Use `np.random.rand()` to generate a random 4×4 matrix, A.

In [36]:
```
A_numbers = [np.random.rand() for i in range(16)]
A = np.array(A_numbers).reshape(4,4)
print(A)
```

```
[[0.44950705 0.10813919 0.49233686 0.19147725]
 [0.51854825 0.51374271 0.05151946 0.45871465]
 [0.84503145 0.40734015 0.34456353 0.59290289]
 [0.85739481 0.5454557  0.37002365 0.19048936]]
```

4(b)

We can also use `numpy` to easily calculate matrix powers. Using `np.linalg.matrix_power(A,n)` will calculate the n^{th} power of A (read more about `np.linalg.matrix_power()` here (https://numpy.org/doc/stable/reference/generated/numpy.linalg.matrix_power.html)).

Calculate the tenth power of your randomly generated matrix, A, i.e., calculate A^{10}.

In [37]:
```
A_power_10 = np.linalg.matrix_power(A,n=10)
print(A_power_10)
```

```
[[45.39724443 24.77626148 24.9170542  23.9018814 ]
 [50.18706707 27.39038687 27.54602708 26.42374802]
 [74.42049762 40.61616149 40.84696094 39.18277264]
 [64.39191807 35.14290651 35.34260374 33.90267411]]
```

4(c)

An interesting property of matrices is that $(A^n)^T = (A^T)^n$.

To test this, calculate the tenth power of the *transpose* of your matrix, A, and compare it to the transpose of the tenth power of your matrix, A^{10}.

In [58]:
```
A_10_T = A_power_10.transpose()
A_T_10 = np.linalg.matrix_power(A.transpose(), n= 10)

print("(A^10)^T = \n",A_10_T,"\n")
print("(A^T)^10 = \n",A_T_10,"\n")
```

```
(A^10)^T = 
 [[45.39724443 50.18706707 74.42049762 64.39191807]
 [24.77626148 27.39038687 40.61616149 35.14290651]
 [24.9170542  27.54602708 40.84696094 35.34260374]
 [23.9018814  26.42374802 39.18277264 33.90267411]] 

(A^T)^10 = 
 [[45.39724443 50.18706707 74.42049762 64.39191807]
 [24.77626148 27.39038687 40.61616149 35.14290651]
 [24.9170542  27.54602708 40.84696094 35.34260374]
 [23.9018814  26.42374802 39.18277264 33.90267411]] 
```

By comparing them from the print result, we can prove $(A^n)^T = (A^T)^n$

In [51]:
```
A_10_T == A_T_10
```

Out[51]:
```
array([[ True, False, False, False],
       [False,  True, False, False],
       [False,  True, False,  True],
       [ True,  True,  True, False]])
```

If we use `A_10_T == A_T_10` we will not get 4×4 "True" Matrix, because the computer can not store the all the decimals in a float value, so there will some precision loss during calculation.

In [81]:
```
A_10_T - A_T_10
```

Out[81]:
```
array([[ 0.00000000e+00, -2.13162821e-14, -1.42108547e-14,
        -1.42108547e-14],
       [-3.55271368e-15,  0.00000000e+00, -7.10542736e-15,
        -7.10542736e-15],
       [-7.10542736e-15,  0.00000000e+00, -7.10542736e-15,
         0.00000000e+00],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        -7.10542736e-15]])
```

But we can calculate the `A_10_T - A_T_10`, then we can get the subtractions are very small, acutually `A_10_T` and `A_T_10` are identical.

4(d)

Finally, we would like to see if the inverse of the transpose of a matrix is equal to the transpose of its inverse.

Calculate the transpose of the inverse of your matrix, $(A^{-1})^T$, and check that it is the inverse of the transpose of your matrix by multiplying together $(A^{-1})^T$ and A^T.

Hint: There is a chance that your random matrix is not invertible. If this is the case, you can generate a new matrix by re-running the code in this notebook.

In [79]:
```
A_T = A.transpose()
A_T_inv = np.linalg.inv(A_T)
A_inv = np.linalg.inv(A)
A_inv_T = np.linalg.inv(A).transpose()

np.matmul(A_inv_T, A_T)
```

Out[79]:
```
array([[ 1.00000000e+00, -2.22044605e-16, -4.44089210e-16,
         0.00000000e+00],
       [-2.22044605e-16,  1.00000000e+00, -8.88178420e-16,
        -4.44089210e-16],
       [ 2.22044605e-16, -4.44089210e-16,  1.00000000e+00,
         0.00000000e+00],
       [-1.11022302e-16,  0.00000000e+00,  2.22044605e-16,
         1.00000000e+00]])
```

Do you notice anything strange about the output of your calculation?

Think about the result you expect to see if you multiply a matrix by its inverse, and then think about the result you have obtained above. Is your result exactly the same as what you expected, or is it slightly different? What might explain the differences?

Computer have precision limit, it can not store all the decimals of float number, so we can not get a perfect matrix I. There are still some small non-zero values in non-diagnal line.

In [80]:
```
np.matmul(A_inv_T, A_T).round(10)
```

Out[80]:
```
array([[ 1., -0., -0.,  0.],
       [-0.,  1., -0., -0.],
       [ 0., -0.,  1.,  0.],
       [-0.,  0.,  0.,  1.]])
```

We round to 10 decimals(a very high precission standard), we can get see a perfect diagonal I matrix. So actually `A_inv_T == A_T_inv`