

# Xi Liu

## Question 1

Two players take turns trying to kick a ball into the net in soccer. Player 1 succeeds with probability  $1/5$  and Player 2 succeeds with the probability  $1/4$ . Whoever succeeds first wins the game and the game is over. Assuming that Player 1 takes the first shot, what is the probability that Player 1 wins the game? Please derive your answer.

$p_1$  be probability of success of player 1,  $p_1 = 1/5$

$p_2$  be probability of success of player 2,  $p_2 = 1/4$

$$\begin{aligned}\sum_{i=1}^{\infty} p_1(1-p_1)^{i-1}(1-p_2)^{i-1} &= \sum_{i=1}^{\infty} \left(\frac{1}{5}\right) \left(1-\frac{1}{5}\right)^{i-1} \left(1-\frac{1}{4}\right)^{i-1} \\ &= \sum_{i=1}^{\infty} \left(\frac{1}{5}\right) \left(\frac{4}{5}\right)^{i-1} \left(\frac{3}{4}\right)^{i-1} \\ &= \frac{1}{5} \sum_{i=1}^{\infty} \left(\frac{4}{5}\right)^{i-1} \left(\frac{3}{4}\right)^{i-1} \\ &= \frac{1}{5} \sum_{i=1}^{\infty} \left(\frac{4}{5} \cdot \frac{3}{4}\right)^{i-1} \\ &= \frac{1}{5} \sum_{i=1}^{\infty} \left(\frac{3}{5}\right)^{i-1} \\ &= \frac{1}{5} \sum_{i=0}^{\infty} \left(\frac{3}{5}\right)^i\end{aligned}$$

$$/* \text{geometric series sum } \sum_{i=0}^n ar^i = a \left( \frac{1-r^{n+1}}{1-r} \right)$$

$$\text{if } |r| < 1 \quad \lim_{n \rightarrow \infty} \sum_{i=0}^n ar^i = \frac{a}{1-r} \quad */$$

$$\begin{aligned}&= \frac{1}{5} \cdot \frac{1}{1-3/5} \\ &= \frac{1}{5} \cdot \frac{1}{2/5} \\ &= \frac{1}{5} \cdot \frac{5}{2} \\ &= \boxed{\frac{1}{2}}\end{aligned}$$

## Question 2

You know that 1% of the population have COVID. You also know that 90% of the people who have COVID get a positive test result and 10% of people who do not have COVID also test positive. What is the probability that you have COVID given that you tested positive?

let  $A$  denote the event that you tested positive

$B$  be the event that you have covid

$$P(B) = 0.01$$

$$P(B^c) = 1 - P(B) = 1 - 0.01 = 0.99$$

$$P(A|B) = 0.9$$

$$P(A|B^c) = 0.1$$

$$\begin{aligned} P(A|B) &= \frac{P(A \cap B)}{P(B)}, & P(A|B^c) &= \frac{P(A \cap B^c)}{P(B^c)} \\ P(A \cap B) &= P(A|B)P(B), & P(A \cap B^c) &= P(A|B^c)P(B^c) \\ P(A) &= P(A \cap B) + P(A \cap B^c) = P(A|B)P(B) + P(A|B^c)P(B^c) \\ P(B|A) &= \frac{P(A \cap B)}{P(A)} \\ &= \frac{P(A|B)P(B)}{P(A|B)P(B) + P(A|B^c)P(B^c)} \\ &= \frac{(0.9)(0.01)}{(0.9)(0.01) + (0.1)(0.99)} \\ &= \frac{0.009}{0.108} \\ &= \frac{9}{108} \\ &= \boxed{\frac{1}{12}} \end{aligned}$$

### Question 3

Let the function  $f(x)$  be defined as:

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ \frac{1}{(1+x)} & \text{otherwise.} \end{cases} \quad (1)$$

Is  $f(x)$  a PDF? If yes, then prove that it is a PDF. If no, then prove that it is not a PDF.

let  $X$  be a continuous random variable, if  $func$  is a probability density function, then  $func$  has the property

$$P(-\infty \leq X \leq \infty) = \int_{-\infty}^{\infty} func(x) dx = 1$$

$$I := \int_{-\infty}^{\infty} f(x) dx = \int_0^{\infty} \frac{1}{1+x} dx$$

$$\begin{aligned} u &:= 1 + x, & du &= dx \\ \int \frac{1}{1+x} dx &= \int \frac{du}{u} \\ &= \ln(u) \\ &= \ln(1+x) \end{aligned}$$

$$\begin{aligned} I &= \lim_{a \rightarrow \infty} [\ln(1+x)]_0^a \\ &= \lim_{a \rightarrow \infty} \ln(a) - \ln(1) \\ &= \lim_{a \rightarrow \infty} \ln(a) - 0 \\ &= \lim_{a \rightarrow \infty} \ln(a) \\ &= \infty \end{aligned}$$

since the integral  $I$  is not equal to 1,  $f(x)$  is not a probability density function

**Question 4 (10 points)**

Assume that  $X$  and  $Y$  are two independent random variables and both have the same density function:

$$f(x) = \begin{cases} 2x & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

What is the value of  $\mathbb{P}(X + Y \leq 1)$ ?

$$\begin{aligned} P(X + Y \leq 1) &= \int_0^1 \int_0^{1-y} f_{X,Y}(x, y) dx dy \\ & \quad /* \text{ since } X, Y \text{ are independent, } f_{X,Y}(x, y) = f_X(x)f_Y(y) */ \\ &= \int_0^1 \int_0^{1-y} f_X(x)f_Y(y) dx dy \\ &= \int_0^1 \int_0^{1-y} (2x)(2y) dx dy \\ &= \int_0^1 \int_0^{1-y} 4xy \, dx dy \\ &= \int_0^1 [2x^2y]_{x=0}^{x=1-y} dy \\ &= 2 \int_0^1 (1-y)^2 y \, dy \\ &= 2 \int_0^1 (y - 2y^2 + y^3) dy \\ &= 2 \left[ \frac{y^2}{2} - \frac{2}{3}y^3 + \frac{y^4}{4} \right]_0^1 \\ &= 2 \left( \frac{1}{2} - \frac{2}{3} + \frac{1}{4} \right) \\ &= \frac{2}{12} \\ &= \boxed{\frac{1}{6}} \end{aligned}$$

### Question 5

Let  $X$  be a random variable which belongs to a Uniform distribution between 0 and 1:  $X \sim Unif(0, 1)$ . Let  $Y = g(X) = e^X$ . What is the value of  $\mathbb{E}(Y)$ ?

$a, b \in \mathbb{R}$ , for  $Unif(a, b)$

$$f(x) = \frac{1}{b-a}$$

for  $Unif(0, 1)$

$$f(x) = \frac{1}{1-0} = 1$$

$$E[X] = \int_{-\infty}^{\infty} x f(x) dx$$

$$E[Y] = E[g(X)] = \int_{-\infty}^{\infty} g(x) f(x) dx$$

$$= \int_{-\infty}^{\infty} e^x (1) dx$$

$$= \int_0^1 e^x dx$$

$$= [e^x]_0^1$$

$$= e^1 - e^0$$

$$= \boxed{e - 1}$$

### Question 6

Suppose that the number of errors per computer program has a Poisson distribution with mean 5. We have 125 program submissions. Let  $X_1, X_2, \dots, X_{125}$  denote the number of errors in the programs. What is the value of  $\mathbb{P}(\bar{X}_n < 5.5)$ ?

central limit theorem: let  $X_1, X_2, \dots, X_n$  be random variables, each  $X_i$  has expected value  $\mu$ , variance  $\sigma^2$  of each of the  $X_i$ .  $\forall n \geq 1$

$$Z_n := \frac{\bar{X}_n - E[\bar{X}_n]}{\sqrt{\text{Var}(\bar{X}_n)}} = \frac{\sum_{i=1}^n X_i - n\mu}{\sqrt{n\sigma^2}} = \frac{n\bar{X}_n - n\mu}{\sqrt{n}\sigma} = \sqrt{n} \frac{\bar{X}_n - \mu}{\sigma}$$

let  $\mu$  be mean,  $\sigma^2$  be variance  
for a poisson distribution,  $\mu = \sigma^2 = 5$

$$\begin{aligned} Z_n &= \sqrt{n} \frac{\bar{X}_n - \mu}{\sigma} \\ &= \sqrt{125} \frac{\bar{X}_n - 5}{\sqrt{5}} \\ &= \sqrt{5} \sqrt{25} \frac{\bar{X}_n - 5}{\sqrt{5}} \\ &= 5(\bar{X}_n - 5) \end{aligned}$$

$Z_n$  converge to the probability density function of standard normal distribution  $\Phi$

$$\begin{aligned} P(\bar{X}_n < 5.5) &= P(5(\bar{X}_n - 5) < 5(5.5 - 5)) \\ &= P(5(\bar{X}_n - 5) < 2.5) \\ &\approx \Phi(2.5) \\ &= \boxed{0.9938} \end{aligned}$$

### Question 7 (10 points)

Let  $X_n = f(W_n, X_{n-1})$  for  $n = 1, \dots, P$ , for some function  $f()$ . Let us define the value of variable  $E$  as

$$E = \|C - X_P\|^2, \tag{3}$$

for some constant  $C$ . What is the value of the gradient  $\frac{\partial E}{\partial X_0}$ ?

$$\begin{aligned}
X_P &= f(W_P, X_{P-1}) \\
\frac{\partial E}{\partial X_0} &= \frac{\partial \|C - X_P\|^2}{\partial X_0} \\
&= \frac{\partial \|C - X_P\|^2}{\partial X_P} \frac{\partial X_P}{\partial X_0} \\
/* \text{ dot product of a vector } \mathbf{v} \text{ with itself is the square of } \mathbf{v}'\text{'s magnitude} \\
\text{since } \mathbf{v} \cdot \mathbf{v} &= |\mathbf{v}||\mathbf{v}| \cos(0) = |\mathbf{v}||\mathbf{v}|(1) = |\mathbf{v}|^2 */ \\
&= \frac{\partial ((C - X_P) \cdot (C - X_P))}{\partial X_P} \frac{\partial X_P}{\partial X_0} \\
/* \text{ derivative of a dot product of } \vec{a} + \vec{b} \text{ with respect to } a_i: \\
\frac{\partial ((\vec{a} + \vec{b}) \cdot (\vec{a} + \vec{b}))}{\partial b_i} &= \frac{\partial (\sum_{i=1}^n a_i^2 + 2a_i b_i + b_i^2)}{\partial b_i} = 2a_i + 2b_i = 2(\vec{a} + \vec{b})_i \\
\text{since } \|C - X_P\| &= \|(-1)(C - X_P)\| = \|-C + X_P\| \\
\text{let } \vec{a} &:= -C, \quad \vec{b} := X_P \\
\text{then } \frac{\partial ((-C + X_P) \cdot (-C + X_P))}{\partial X_P} &= 2(-C + X_P)^T = 2(X_P - C)^T */ \\
&= 2(X_P - C)^T \frac{\partial X_P}{\partial X_{P-1}} \frac{\partial X_{P-1}}{\partial X_{P-2}} \cdots \frac{\partial X_1}{\partial X_0} \\
&= \boxed{2(X_P - C)^T \prod_{i=1}^P D_i}
\end{aligned}$$

where  $D_i$  is the Jacobian matrix of  $f(W_i, X_{i-1})$

### Question 8 (10 points)

Let  $A$  be the matrix  $\begin{bmatrix} 2 & 6 & 7 \\ 3 & 1 & 2 \\ 5 & 3 & 4 \end{bmatrix}$  and let  $x$  be the column vector  $\begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$ . Let  $A^T$  and  $x^T$  denote the transpose of  $A$  and  $x$  respectively. Compute  $Ax$ ,  $A^T$  and  $x^T A$ .

$$\begin{aligned}
Ax &= \begin{bmatrix} 2 & 6 & 7 \\ 3 & 1 & 2 \\ 5 & 3 & 4 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 4 + 18 + 28 \\ 6 + 3 + 8 \\ 10 + 9 + 16 \end{bmatrix} = \begin{bmatrix} 50 \\ 17 \\ 35 \end{bmatrix} \\
A^T &= \begin{bmatrix} 2 & 3 & 5 \\ 6 & 1 & 3 \\ 7 & 2 & 4 \end{bmatrix} \\
x^T A &= \begin{bmatrix} 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 2 & 6 & 7 \\ 3 & 1 & 2 \\ 5 & 3 & 4 \end{bmatrix} = \begin{bmatrix} 4 + 9 + 20 & 12 + 3 + 12 & 14 + 6 + 16 \end{bmatrix} = \begin{bmatrix} 33 & 27 & 36 \end{bmatrix}
\end{aligned}$$

### Question 9

Find out if the following matrices are invertible. If yes, find the inverse of the matrix.

(a)

$$\begin{bmatrix} 6 & 2 & 3 \\ 3 & 1 & 1 \\ 10 & 3 & 4 \end{bmatrix} \quad (4)$$

(b)

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 2 \\ 1 & 4 & 5 \end{bmatrix} \quad (5)$$

a matrix is invertible if and only if its determinant is not 0

below calculate determinant using Laplace expansion

(a)

$$\begin{aligned} \begin{vmatrix} 6 & 2 & 3 \\ 3 & 1 & 1 \\ 10 & 3 & 4 \end{vmatrix} &= 6 \begin{vmatrix} 1 & 1 \\ 3 & 4 \end{vmatrix} - 2 \begin{vmatrix} 3 & 1 \\ 10 & 4 \end{vmatrix} + 3 \begin{vmatrix} 3 & 1 \\ 10 & 3 \end{vmatrix} \\ &= 6(4 - 3) - 2(12 - 10) + 3(9 - 10) = 6 - 4 - 3 = -1 \end{aligned}$$



determinant of (a) is 1, which is not 0, so the matrix is invertible

$$\begin{aligned}
 & \left[ \begin{array}{ccc|ccc} 6 & 2 & 3 & 1 & 0 & 0 \\ 3 & 1 & 1 & 0 & 1 & 0 \\ 10 & 3 & 4 & 0 & 0 & 1 \end{array} \right] \\
 R_1 & \leftarrow \frac{R_1}{6} \left[ \begin{array}{ccc|ccc} 1 & 1/3 & 1/2 & 1/6 & 0 & 0 \\ 3 & 1 & 1 & 0 & 1 & 0 \\ 10 & 3 & 4 & 0 & 0 & 1 \end{array} \right] \\
 R_2 & \leftarrow R_2 - 3R_1 \left[ \begin{array}{ccc|ccc} 1 & 1/3 & 1/2 & 1/6 & 0 & 0 \\ 0 & 0 & -1/2 & -1/2 & 1 & 0 \\ 10 & 3 & 4 & 0 & 0 & 1 \end{array} \right] \\
 R_3 & \leftarrow R_3 - 10R_1 \left[ \begin{array}{ccc|ccc} 1 & 1/3 & 1/2 & 1/6 & 0 & 0 \\ 0 & 0 & -1/2 & -1/2 & 1 & 0 \\ 0 & -1/3 & -1 & -5/3 & 0 & 1 \end{array} \right] \\
 R_2 & \leftrightarrow R_3 \left[ \begin{array}{ccc|ccc} 1 & 1/3 & 1/2 & 1/6 & 0 & 0 \\ 0 & -1/3 & -1 & -5/3 & 0 & 1 \\ 0 & 0 & -1/2 & -1/2 & 1 & 0 \end{array} \right] \\
 R_2 & \leftarrow -3R_2 \left[ \begin{array}{ccc|ccc} 1 & 1/3 & 1/2 & 1/6 & 0 & 0 \\ 0 & 1 & 3 & 5 & 0 & -3 \\ 0 & 0 & -1/2 & -1/2 & 1 & 0 \end{array} \right] \\
 R_3 & \leftarrow -2R_3 \left[ \begin{array}{ccc|ccc} 1 & 1/3 & 1/2 & 1/6 & 0 & 0 \\ 0 & 1 & 3 & 5 & 0 & -3 \\ 0 & 0 & 1 & 1 & -2 & 0 \end{array} \right] \\
 R_1 & \leftarrow R_1 - \frac{R_2}{3} \left[ \begin{array}{ccc|ccc} 1 & 0 & -1/2 & -3/2 & 0 & 1 \\ 0 & 1 & 3 & 5 & 0 & -3 \\ 0 & 0 & 1 & 1 & -2 & 0 \end{array} \right] \\
 R_2 & \leftarrow R_2 - 3R_3 \left[ \begin{array}{ccc|ccc} 1 & 0 & -1/2 & -3/2 & 0 & 1 \\ 0 & 1 & 0 & 2 & 6 & -3 \\ 0 & 0 & 1 & 1 & -2 & 0 \end{array} \right] \\
 R_1 & \leftarrow R_1 + \frac{R_3}{2} \left[ \begin{array}{ccc|ccc} 1 & 0 & 0 & -1 & -1 & 1 \\ 0 & 1 & 0 & 2 & 6 & -3 \\ 0 & 0 & 1 & 1 & -2 & 0 \end{array} \right]
 \end{aligned}$$

inverted matrix of (a) is  $\begin{bmatrix} -1 & -1 & 1 \\ 2 & 6 & -3 \\ 1 & -2 & 0 \end{bmatrix}$

(b)

$$\begin{aligned}
 \begin{vmatrix} 1 & 2 & 3 \\ 0 & 2 & 2 \\ 1 & 4 & 5 \end{vmatrix} &= 1 \begin{vmatrix} 2 & 2 \\ 4 & 5 \end{vmatrix} - 2 \begin{vmatrix} 0 & 2 \\ 1 & 5 \end{vmatrix} + 3 \begin{vmatrix} 0 & 2 \\ 1 & 4 \end{vmatrix} \\
 &= 1(10 - 8) - 2(0 - 2) + 3(0 - 2) = 2 + 4 - 6 = 0
 \end{aligned}$$

determinant of (b) is 0, so the matrix is not invertible

### Question 10

What is an Eigen Value of a matrix? What is an Eigen Vector of a matrix? Describe one method (any method) you would use to compute both of them. Use the above described

method to compute the Eigen Values of the matrix:

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & 0 \\ -2 & 2 & 1 \end{bmatrix} \quad (6)$$

let  $V$  be a vector space over the field  $\mathbb{C}$ ,  $A : V \rightarrow V$  be a linear map. An element  $v \in V$  is an eigenvector of  $A$  if  $\exists \lambda \in \mathbb{C}, Av = \lambda v$ , then  $\lambda$  is an eigenvalue of  $A$  associated with eigenvector  $v$

$$Av = \lambda v$$

$$Av - \lambda v = \mathbf{0}$$

$$Av - \lambda Iv = \mathbf{0}$$

$$(A - \lambda I)v = \mathbf{0}$$

$$\det(A - \lambda I) = 0$$

$$A - \lambda I = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & 0 \\ -2 & 2 & 1 \end{bmatrix} - \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix} = \begin{bmatrix} 1-\lambda & 0 & -1 \\ 1 & -\lambda & 0 \\ -2 & 2 & 1-\lambda \end{bmatrix}$$

$$\begin{aligned} \det(A - \lambda I) &= \begin{vmatrix} 1-\lambda & 0 & -1 \\ 1 & -\lambda & 0 \\ -2 & 2 & 1-\lambda \end{vmatrix} \\ &= (1-\lambda)((-\lambda)(1-\lambda) - 0) - (1)(2 - (-\lambda)(-2)) \\ &= (1-\lambda)(-\lambda + \lambda^2) - (2 - 2\lambda) \\ &= -\lambda + \lambda^2 + \lambda^2 - \lambda^3 - 2 + 2\lambda \\ &= -\lambda^3 + 2\lambda^2 + \lambda - 2 \\ &= -(\lambda - 1)(\lambda^2 - \lambda - 2) \\ &= -(\lambda - 1)(\lambda - 2)(\lambda + 1) \\ &= 0 \end{aligned}$$

$$\lambda_1 = 1$$

$$\lambda_2 = 2$$

$$\lambda_3 = -1$$

## Homework 1: Math Foundations for ML

This is the coding portion of Homework 1. The homework is aimed at testing the ability to code up mathematical operations using Python and the numpy library.

For each problem, we provide hints or example test cases to check your answers (see the assert statements below). Your full submission will be autograded on a larger batch of randomly generated test cases.

### Note on the autograding process

For this assignment, we are using nbgrader for autograding. We recommend that you use JupyterLab or Jupyter notebook to complete this assignment for compatibility.

The cells containing example test cases also serve as placeholders for the autograder to know where to inject additional random tests. Notice that they are always after your solution; moving/deleting them will cause the tests to fail, and we'd have to manually regrade it. They are marked with DO NOT MOVE/DELETE and set to read-only just in case.

The autograder tests will call the functions named `solve_system`, `split_into_train_and_test`, `closest_interval`. You may not change the function signature (function name and argument list), but otherwise feel free to add helper functions in your solution. You can also make a copy of the notebook and use that as a scratchpad.

To double check your submission format, restart your kernel (Menu bar -> Kernel -> Restart Kernel); execute all cells from top to bottom, and see if you can pass the example test cases.

```
import numpy as np
```

### Part 1: Systems of linear equations

Given  $n$  equations with  $n$  unknown variables ( $n \leq 4$ ), write a function `solve_system` that can solve this system of equations and produce an output of value for each variable such that the system of equations is satisfied.

The system of equations will be provided as a list of strings as seen in `test_eq`.

You may assume that the variables are always in  $\{a, b, c, d\}$ , the system has a unique solution, and all coefficients are integers.

```
def solve_system(equations):  
    """  
    Takes in a list of strings for each equation.  
    Returns a numpy array with a row for each equation value  
    """  
    def strnum(s): # string to numeric coefficient
```

```

cur_num, is_pos, eq_seen = 0, 1, 0
n = len(s)
res = []
i = 0
while(i < n):
    while(i < n and '0' <= s[i] and s[i] <= '9'):
        cur_num = cur_num * 10 + ord(s[i]) - ord('0')
        i += 1
    if(eq_seen and cur_num):
        res.append(cur_num if is_pos else -cur_num)
    if(i == n):
        return res
    c = s[i]
    if(c == '+'):
        is_pos = 1
    elif(c == '-'):
        is_pos = 0
    elif('a' <= c <= 'd'):
        res.append(cur_num if is_pos else -cur_num)
        if(not res[-1]):
            res[-1] = 1
        cur_num = 0
    elif(c == '='):
        eq_seen = 1
    i += 1
return res
a, b = [], []
eq_n = len(equations)
for i in range(eq_n):
    _a = strnum(equations[i])
    len_a = len(_a)
    a.append(_a[:len_a - 1])
    for j in range(len_a - 1, eq_n):
        a[-1].append(0)
    b.append(_a[len_a] - 1])
sol = np.linalg.solve(a, b)
sol_n = len(sol)
res = []
for i in range(sol_n):
    res.append([sol[i]])
return res

```

*# === DO NOT MOVE/DELETE ===*

*# This cell is used as a placeholder for autograder script injection.*

```

def test_eq(sys_eq):
    results = solve_system(sys_eq)
    expected = np.array([[3],[5],[2],[4]])
    assert np.allclose(expected, results)

```

```
test_eq([
    '2 a + b - 3 c + d = 9',
    '-5 a + b - 4 c + d = -14',
    'a + 2 b - 10 c = -7',
    'a + 2 b = 13',
])
```

## Part 2: Split a dataset into test and train

(For this question, using an existing implementation (e.g. `sklearn.model_selection.train_test_split`) will give 0 points.)

In supervised learning, the dataset is usually split into a train set (on which the model is trained) and a test set (to evaluate the trained model). This part of the homework requires writing a function `split_into_train_and_test` that takes a dataset and the train-test split ratio as input and provides the data split as an output. The function takes a `random_state` variable as input which when kept the same outputs the same split for multiple runs of the function.

Note: if `frac_test` does not result in an integer test set size, round down to the nearest integer.

Hints:

- The input array `x_all_LF` should not be altered after the function call.
- Running the function with the same seed multiple times should yield the same results.
- Every element in the input array should appear either in the train or test set, but not in both.

```
def split_into_train_and_test(x_all_LF, frac_test=0.5, seed=None):
    ''' Divide provided array into train and test sets along first
        dimension
        User can provide random number generator object to ensure
        reproducibility.
        Args
        ----
        x_all_LF : 2D np.array, shape = (n_total_examples, n_features) (L,
F)
            Each row is a feature vector
        frac_test : float, fraction between 0 and 1
            Indicates fraction of all L examples to allocate to the "test"
set
            Returned test set will round UP if frac_test * L is not an
integer.
            e.g. if L = 10 and frac_test = 0.31, then test set has N=4
examples
        seed : integer or None
            If int, will create RandomState instance with provided value
as seed
            If None, defaults to current numpy random number generator
```

```

np.random.
    Returns
    -----
    x_train_MF : 2D np.array, shape = (n_train_examples, n_features)
    (M, F)
        Each row is a feature vector
        Should be a separately allocated array, NOT a view of any
input array
    x_test_NF : 2D np.array, shape = (n_test_examples, n_features) (N,
    F)
        Each row is a feature vector
        Should be a separately allocated array, NOT a view of any
input array
    Post Condition
    -----
    This function should be side-effect free. Provided input array
x_all_LF
    should not change at all (not be shuffled, etc.)
    Examples
    -----
    >>> x_LF = np.eye(10)
    >>> xcopy_LF = x_LF.copy() # preserve what input was before the
call
    >>> train_MF, test_NF = split_into_train_and_test(
    ...     x_LF, frac_test=0.201,
random_state=np.random.RandomState(0))
    >>> train_MF.shape
    (7, 10)
    >>> test_NF.shape
    (3, 10)
    >>> print(train_MF)
[[0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
 [0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]
 [0.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]
 [0.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [0.  0.  0.  0.  0.  0.  1.  0.  0.  0.]
 [0.  0.  0.  0.  0.  0.  0.  1.  0.  0.]]
    >>> print(test_NF)
[[0.  0.  0.  1.  0.  0.  0.  0.  0.  0.]
 [1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]]
    ## Verify that input array did not change due to function call
    >>> np.allclose(x_LF, xcopy_LF)
    True
    References
    -----
    For more about RandomState, see:
    https://stackoverflow.com/questions/28064634/random-state-pseudo-random-numberin-scikit-learn

```

```

...
if seed is None:
    rng = np.random.RandomState()
    rng = np.random.default_rng(seed)
    copy = x_all_LF.copy()
    rng.shuffle(copy)
    L = np.shape(x_all_LF)[0]
    import math
    N = math.floor(frac_test * L) # n_test_examples
    x_test_NF, x_train_MF = [], []
    for i in range(N):
        x_test_NF.append(copy[i].copy())
    for i in range(N, L):
        x_train_MF.append(copy[i].copy())
    return np.array(x_train_MF), np.array(x_test_NF)

# === DO NOT MOVE/DELETE ===
# This cell is used as a placeholder for autograder script injection.

N = 10
x_LF = np.eye(N)
xcopy_LF = x_LF.copy() # preserve what input was before the call
train_MF, test_NF = split_into_train_and_test(x_LF, frac_test=0.2,
seed=0)

```

### Part 3: Solving a Search Problem

Given a list of  $N$  intervals, for each interval  $[a, b]$ , we want to find the closest non-overlapping interval  $[c, d]$  greater than  $[a, b]$ .

An interval  $[c, d]$  is greater than a non-overlapping interval  $[a, b]$  if  $a < b < c < d$ .

The function `closest_interval` takes in the list of intervals, and returns a list of indices corresponding to the index of the closest non-overlapping interval for each interval in the list. If a particular interval does not have a closest non-overlapping interval in the given list, return -1 corresponding to that element in the list.

```

from typing import List
from dataclasses import dataclass
from functools import cmp_to_key

```

```
@dataclass
```

```
class pair:
    a: int
    b: int

```

```
def cmp(p1: pair, p2: pair) -> int:
    return p1.a - p2.a

```

```
def lower_bound(a: List[pair], x: pair, n: int, cmp) -> int:

```

```

low, high = 0, n
while(low < high):
    mid = low + (high - low) // 2
    if(cmp(x, a[mid]) < 0):
        high = mid
    else:
        low = mid + 1
if(low < n and cmp(a[low], x) < 0):
    low += 1
return low

```

```

def closest_interval(intervals):
    n = len(intervals)
    sort_a = [pair(0, 0) for _ in range(n)]
    for i in range(n):
        sort_a[i].a = intervals[i][0]
        sort_a[i].b = i
    sort_a = sorted(sort_a, key = cmp_to_key(cmp))
    res = n * [-1]
    for i in range(n):
        j = lower_bound(sort_a, pair(intervals[i][1], 0), n, cmp)
        if(j != n):
            res[i] = sort_a[j].b
    return res

```

*# === DO NOT MOVE/DELETE ===*

*# This cell is used as a placeholder for autograder script injection.*

```

intervals = np.array([
    [1, 4],
    [2, 5],
    [8, 9],
    [6, 8],
    [9, 10],
    [3, 4],
    [7, 9],
    [5, 7],
])

```

```

expected_closest_intervals = closest_interval(intervals)

```

*# Evaluate*

```

results = np.array([7, 3, -1, 4, -1, 7, -1, 2])
assert np.allclose(expected_closest_intervals, results)

```