Introduction to Machine Learning (CSCI-UA.473): Homework 1

Instructor: Lerrel Pinto

September 6, 2022

Submission Instructions

You must typeset the answers using LATEX and compile them into a single PDF file. Name the pdf file as ⟨Your-NetID⟩_hw1.pdf and the notebook containing the coding portion as ⟨Your-NetID⟩_hw1.ipynb. The PDF file should contain solutions to both the theory portion and the coding portion. Submit the files through the following Google Form - https://forms.gle/Vqj9ry6o3mqim6Hm6 The due date is **September 20, 2022, 11:59 PM**. You may discuss the questions with each other but each student must provide their own answer to each question.

Questions

Probability and Calculus

Question 1 (10 points)

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.

Answer:

Assume player1 as A and player2 as B.
P(A succeeds) = $\frac{1}{5}$
P(A fails) = $1 - \frac{1}{5} = \frac{4}{5}$
P(B succeeds) = $\frac{1}{4}$
P(B fails) = $1 - \frac{1}{4} = \frac{3}{4}$
We know that A takes the 1st turn.

1st (A)	1st (B)	2nd (A)	2nd (B)	3rd (A)	3rd (B)	...	nth (A)
$\frac{1}{5}$							
$\frac{4}{5}$	$\frac{3}{4}$	$\frac{1}{5}$					
$\frac{4}{5}$	$\frac{3}{4}$	$\frac{4}{5}$	$\frac{3}{4}$	$\frac{1}{5}$			
$\frac{4}{5}$	$\frac{3}{4}$	$\frac{4}{5}$	$\frac{3}{4}$	$\frac{4}{5}$	$\frac{3}{4}$	...	$\frac{1}{5}$

P(A wins at his 1st turn) = $\frac{1}{5}$
P(A wins at his 2nd turn) = $\frac{4}{5} \times \frac{3}{4} \times \frac{1}{5} = (\frac{3}{5})(\frac{1}{5})$
P(A wins at his 3rd turn) = $(\frac{3}{5})^2(\frac{1}{5})$
P(A wins at his nth turn) = $(\frac{3}{5})^{n-1}(\frac{1}{5})$
Thus, the total probability of A wins the game is P(A wins the game) = $\sum_{n=1}^{\infty}(\frac{3}{5})^{n-1}(\frac{1}{5}) = (\frac{1}{5})\sum_{n=0}^{\infty}(\frac{3}{5})^n = (\frac{1}{5})(\frac{1}{1-\frac{3}{5}}) = (\frac{1}{5})(\frac{5}{2}) = \frac{1}{2}$

Question 2 (10 points)

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?

Answer:

P(covid) = 0.01
P(not covid) = 0.99
P(positive $\cap$ covid) = $0.9 \times 0.01 = 0.009$
P(positive $\cap$ not covid) = $0.1 \times 0.99 = 0.099$
P(positive) = 0.009 + 0.099 = 0.108
P(covid | positive) = $\frac{P(covid \cap positive)}{P(positive)} = \frac{0.009}{0.108} = \frac{1}{12} = 0.083$

Question 3 (10 points)

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

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

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

Answer:

$\int_{-\infty}^{\infty} f(x)\,dx = \int_{0}^{\infty} \frac{1}{1+x}\,dx = \lim_{u\to\infty} \int_{0}^{u} \frac{1}{1+x}\,dx = \lim_{u\to\infty} [ln(x+1)]_0^u = \lim_{u\to\infty} ln(u+1) = ln(\infty) = \infty$

Thus, f(x) is not a PDF, because it diverges.

Question 4 (10 points)

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

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

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

Answer:

$f_{X,Y}(x, y) = f_X(x) \times f_Y(y) = 2x2y = 4xy$
Since X + Y $\leq$ 1, then 0 $\leq$ x $\leq$ 1 and 0 $\leq$ (y=1-x) $\leq$ 1.
Thus, $P(X + Y \leq 1) = \int_0^1 \int_0^{1-x} 4xy\, dy\, dx = \int_0^1 2x^3 - 4x^2 + 2x\, dx = \frac{1}{6}$

Question 5 (10 points)

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 $\mathbb{E}(Y)$?

Answer:

Since $X \sim Unif(0,1)$, then the pdf of $f_X(x)$ is :

$\begin{cases} f_X(x)=1, if x \in [0,1] \\ f_X(x)=1, otherwise \end{cases}$

The cdf of $f_X(x)$ is :

$\begin{cases} F_X(x)=x, if x \in [0,1] \\ F_X(x)=0, otherwise \end{cases}$

Since $Y = g(X) = e^x$ and $X \in [0,1]$, Y lies in the interval [1, e].

Thus, $F_Y(y) = \mathrm{P}(\mathrm{Y} \leq \mathrm{y}) = P(e^x \leq y) = P(X \leq ln(y)) = F_X(lny) = ln(y)$

Then, the cdf of Y is:

$\begin{cases} F_Y(y)=ln(y), if y \in [1,e] \\ F_Y(y)=0, otherwise \end{cases}$

Since $f_Y(y) = \frac{d}{dy}(F_Y(y)) = \frac{d(lny)}{dy} = \frac{1}{y}$,the pdf of Y is:

$\begin{cases} f_Y(y)=\frac{1}{y}, if y \in [1,e] \\ f_Y(y)=0, otherwise \end{cases}$

Therefore, $\mathrm{E[Y]} = \int_1^e y \cdot \frac{1}{y}\, dy = [y]_1^e = \mathrm{e\text{-}1}$

Question 6 (10 points)

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

Answer:

Since a Poisson distribution with mean = 5, we know that $\mu = \mathrm{E[X]} = \lambda = 5$ and $\sigma^2 = \mathrm{Var(X)} = \lambda = 5$.

According to central limit theorem, $Z_n = \sqrt{n}\frac{\bar{X}_n - \mu}{\sigma}$.

$Z_n = \sqrt{125}\frac{\bar{X}_n - 5}{\sqrt{5}} = 5(\bar{X}_n - 5)$

$\bar{X}_n = \frac{1}{5}Z_n + 5$

Since $\bar{X}_n < 5.5$, then $\frac{1}{5}Z_n + 5 < 5.5$.

So $Z_n < 2.5$.

Thus, $\mathrm{P}(\bar{X}_n < 5.5) = \mathrm{P}(Z_n < 2.5) = 0.9938$

Question 7 (10 points)

Let $X_n = f(W_n, X_{n-1})$ for $n = 1, \ldots, P$, 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 $\frac{\partial E}{\partial X_0}$?

Answer:

Since $X_n = f(W_n, X_{n-1})$ for $n = 1, \ldots, P$, so
$X_p = f(W_p, X_{p-1})$
$= f(W_p, f(W_{p-1}, X_{p-2}))$
$= f(W_p, f(W_{p-1}, f(W_{p-2}, X_{p-3})))$
$= f(W_p, f(W_{p-1}, f(W_{p-2}, f(W_{p-3}, X_{p-4}))))$
$= ...$
$= f(W_p, f(W_{p-1}, f(W_{p-2}, f(W_{p-3}, ..., f(W_1, X_0)))))$

Since $E = ||C - X_P||^2 = (C - X_p)^T (C - X_p)$,
then $\frac{\partial E}{\partial X_p} = -2(C - X_p)$.

Thus, according to chain rule,
$\frac{\partial E}{\partial X_0} = \frac{\partial E}{\partial X_p} \frac{\partial X_p}{\partial X_{p-1}} \frac{\partial X_{p-1}}{\partial X_{p-2}} \frac{\partial X_{p-2}}{\partial X_{p-3}} \cdots \frac{\partial X_1}{\partial X_0}$
$= -2(C - X_p) \frac{\partial f(W_p, X_{p-1})}{\partial X_{p-1}} \frac{\partial f(W_{p-1}, X_{p-2})}{\partial X_{p-2}} \frac{\partial f(W_{p-2}, X_{p-3})}{\partial X_{p-3}} \cdots \frac{\partial f(W_1, X_0)}{\partial X_0}$

Linear Algebra

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 $x^T A$.

Answer:

$$\text{Ax} = \begin{bmatrix} 2 & 6 & 7 \\ 3 & 1 & 2 \\ 5 & 3 & 4 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 4+18+28 \\ 6+3+8 \\ 10+9+16 \end{bmatrix} = \begin{bmatrix} 50 \\ 17 \\ 35 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 2 & 3 & 5 \\ 6 & 1 & 3 \\ 7 & 2 & 4 \end{bmatrix}$$

$$x^T A = \begin{bmatrix} 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 2 & 6 & 7 \\ 3 & 1 & 2 \\ 5 & 3 & 4 \end{bmatrix} = \begin{bmatrix} 33 & 27 & 36 \end{bmatrix}$$

Question 9 (10 points)

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

(a)

$$\begin{bmatrix} 6 & 2 & 3 \\ 3 & 1 & 1 \\ 10 & 3 & 4 \end{bmatrix} \tag{4}$$

Answer: Given A = $\begin{bmatrix} 6 & 2 & 3 \\ 3 & 1 & 1 \\ 10 & 3 & 4 \end{bmatrix}$, so detA = 6(4-3) - 2(12-10) + 3(9-10) = -1 $\neq 0$ and A^{-1} exists.

Cofactor matrix C = $\begin{bmatrix} 1 & -2 & -1 \\ 1 & -6 & 2 \\ -1 & 3 & 0 \end{bmatrix}$

adjA = C^T = $\begin{bmatrix} 1 & 1 & -1 \\ -2 & -6 & 3 \\ -1 & 2 & 0 \end{bmatrix}$

$A^{-1} = \frac{adjA}{detA} = \begin{bmatrix} -1 & -1 & 1 \\ 2 & 6 & -3 \\ 1 & -2 & 0 \end{bmatrix}$

(b)

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 2 \\ 1 & 4 & 5 \end{bmatrix} \tag{5}$$

Answer: Given B = $\begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 2 \\ 1 & 4 & 5 \end{bmatrix}$, so detB = 1(10-8)-2(0-2)+3(0-2) = 0 and B^{-1} does not exist.

Question 10 (10 points)

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:

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & 0 \\ -2 & 2 & 1 \end{bmatrix} \tag{6}$$

Answer:

An eigenvector of an $n \times n$ matrix A is an nonzero vector $\vec{x}$, such that $A\vec{x} = \lambda\vec{x}$ for some scalar λ, known as the eigenvalue of A.

Method to compute eigenvector and eigenvalue:
Let A be an $n \times n$ matrix.
First, find the eigenvalue λ of A by solving equation $det(A - \lambda I) = 0$.
Then, for each λ, find the basic eigenvector $\vec{x} =\neq 0$ by finding the basic solutions to $(A - \lambda I)\vec{x} = \vec{0}$.

Given that A $= \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & 0 \\ -2 & 2 & 1 \end{bmatrix}$.

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

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

$(1 - \lambda)[-\lambda(1 - \lambda)] - 0 - (2 - 2\lambda) = 0$

$-\lambda + \lambda^2 + \lambda^2 - \lambda^3 - 2 + 2\lambda = 0$

$\lambda^3 - 2\lambda^2 - \lambda + 2 = 0$

$\lambda = -1$ or 1 or 2

Homework 1: Math Foundations for ML

The is the coding potion 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.

In [ ]:
```
import numpy as np
import re
import math
import random
```

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.

In [ ]:
```
def solve_system(equations):
    """
    Takes in a list of strings for each equation.
    Returns a numpy array with a row for each equation value
    """
```

```
    # YOUR CODE HERE
    matrix = []
    result = []
    for eq in equations:
        num_list = []
        eq = eq.replace(" ","")
        if len(equations) == 4:
            info = re.findall('([+-]*\d*a)', eq)
            if info == []:
                num_list.append('0')
            else:
                for ele in info:
                    num_list.append(ele)
            info = re.findall('([+-]*\d*b)', eq)
            if info == []:
                num_list.append('0')
            else:
                for ele in info:
                    num_list.append(ele)
            info = re.findall('([+-]*\d*c)', eq)
            if info == []:
                num_list.append('0')
            else:
                for ele in info:
                    num_list.append(ele)
            info = re.findall('([+-]*\d*d)', eq)
            if info == []:
                num_list.append('0')
            else:
                for ele in info:
                    num_list.append(ele)
            for i in range(4):
                num = re.sub('[+abcd]','',num_list[i])
                if num !='' and num != '-':
                    num_list[i] = int(num)
                elif num != '-':
                    num_list[i] = 1
                else:
                    num_list[i] = -1
        elif len(equations) ==3:
            info = re.findall('([+-]*\d*a)', eq)
            if info == []:
                num_list.append('0')
            else:
                for ele in info:
                    num_list.append(ele)
            info = re.findall('([+-]*\d*b)', eq)
            if info == []:
                num_list.append('0')
            else:
                for ele in info:
                    num_list.append(ele)
            info = re.findall('([+-]*\d*c)', eq)
            if info == []:
                num_list.append('0')
            else:
                for ele in info:
                    num_list.append(ele)

            for i in range(3):
                num = re.sub('[+abcd]','',num_list[i])
                if num !='' and num != '-':
                    num_list[i] = int(num)
                elif num != '-':
```

```
                    num_list[i] = 1
                else:
                    num_list[i] = -1
        elif len(equations) == 2:
            info = re.findall('([+-]*\d*a)', eq)
            if info == []:
                num_list.append('0')
            else:
                for ele in info:
                    num_list.append(ele)
            info = re.findall('([+-]*\d*b)', eq)
            if info == []:
                num_list.append('0')
            else:
                for ele in info:
                    num_list.append(ele)
            for i in range(2):
                num = re.sub('[+abcd]','',num_list[i])
                if num !='' and num != '-':
                    num_list[i] = int(num)
                elif num != '-':
                    num_list[i] = 1
                else:
                    num_list[i] = -1
        else:
            info = re.findall('([+-]*\d*a)', eq)
            if info == []:
                num_list.append('0')
            else:
                for ele in info:
                    num_list.append(ele)
            for i in range(1):
                num = re.sub('[+abcd]','',num_list[i])
                if num !='' and num != '-':
                    num_list[i] = int(num)
                elif num != '-':
                    num_list[i] = 1
                else:
                    num_list[i] = -1
        matrix.append(num_list)
        result.append(int(eq.split('=')[1]))

    A = np.array(matrix)
    B = np.array(result)
    if len(equations) != 1:
        x = np.expand_dims(np.linalg.solve(A,B),axis=1)
    else:
        x = np.linalg.solve(A,B)
    return x

    raise NotImplementedError()
```

In [ ]:

```
# === 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.

In [ ]:
```
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-number
    '''
    # YOUR CODE HERE
    if seed is None:
        rng = np.random.RandomState()

    random.seed(seed)

    n_total_examples = x_all_LF.shape[0]
    n_train = math.floor((1-frac_test) * n_total_examples)
    n_test = math.ceil(frac_test * n_total_examples)

    pose_train =[]
    pose_test = random.sample(range(0,n_total_examples), n_test)
    for i in range(n_total_examples):
        if i not in pose_test:
            pose_train.append(i)
    test_NF = x_all_LF[pose_test]
    train_MF = x_all_LF[pose_train]

    return train_MF, test_NF

    raise NotImplementedError()
```

In [ ]:

```
# === 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 an 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.

In [ ]:
```python
def closest_interval(intervals):
    # YOUR CODE HERE

    result = []

    for i in range(len(intervals)):
        num = intervals[i][1]
        close = -1
        min_diff = np.inf
        for j in range(len(intervals)):
            diff = intervals[j][0]-num
            if diff > 0 and diff < min_diff:
                close = j
                min_diff = diff
        result.append(close)

    return result

    raise NotImplementedError()
```

In [ ]:
```python
# === 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)
```