Міністерство освіти та науки України

Львівський національний університет імені Івана Франка

Факультет електроніки та

Комп’ютерних технологій

**Звіт**

Про виконання лабораторної роботи №2

“Побудова функцій приналежності нечіткої множини на основі попарних

порівнянь”

Виконав:

Студент групи ФеІ-44

Сапанюк М.І.

Перевірила:

Притула М.

Львів 2022

**Мета:**

Ознайомитися з непрямим методом побудови функцій приналежності нечіткої множини на основі попарних порівнянь елементів нечіткої множини експертом.

**Хід роботи:**

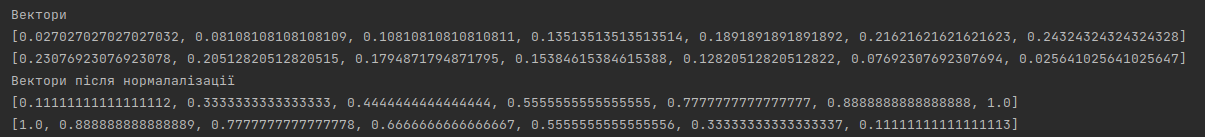
1. На основі опитування експерта побудувати матрицю попарних порівнянь елементів нечіткої множини «висока людина», які відповідають росту: 170, 175, 180, 185, 190, 195, 200 см, і нечіткої множини «низька людина»: 150, 155, 160, 165, 170, 175, 180 см. Зокрема, для кожної пари елементів універсальної множини Х оцінити перевагу одного елемента над іншим відносно певної властивості нечіткої множини.

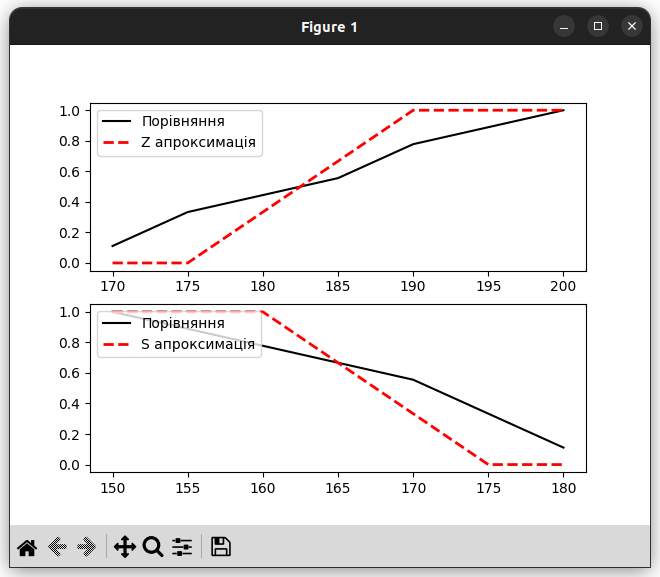
2. Визначити ступінь приналежності і-го елемента до нечіткої множини, який відповідає і-й координаті власного вектора матриці парних порівнянь.

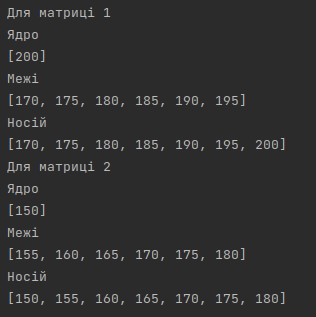
3. Створити програму, яка розраховує ступінь приналежності на основі матриці попарних порівнянь, нормалізує функцію приналежності, будує її графік і апроксимує однією з S-подібних або Z-подібних функцій.

4. Визначити носій нечіткої множини, її ядро та границі.

**Виконання завдання:**







**Висновок:**

Виконавши лабораторну роботу я навчився будувати непрямим методом побудови функцій приналежності нечіткої множини на основі попарних порівнянь елементів нечіткої множини експертом.

**Додаток:**

from matplotlib import pyplot as plt  
  
  
def aprocZ(x: int, a: int, b: int) -> float:  
 if x <= a:  
 return 1  
 elif a < x < b:  
 return (b - x) / (b - a)  
 else:  
 return 0  
  
  
def aprocS(x: int, a: int, b: int) -> float:  
 if x <= a:  
 return 0  
 elif a < x < b:  
 return (x - a) / (b - a)  
 else:  
 return 1  
  
  
def carrier(array\_label: list[int], array\_mu: list[float]):  
 temp: list[int] = []  
 for i in range(0, len(array\_mu)):  
 if array\_mu[i]:  
 temp.append(array\_label[i])  
 print("Носій")  
 print(temp)  
  
  
def core(array\_label: list[int], array\_mu: list[float]):  
 temp: list[int] = []  
 for i in range(0, len(array\_mu)):  
 if array\_mu[i] == 1:  
 temp.append(array\_label[i])  
 print("Ядро")  
 print(temp)  
  
  
def border(array\_label: list[int], array\_mu: list[float]):  
 temp: list[int] = []  
 for i in range(0, len(array\_mu)):  
 if 0 < array\_mu[i] < 1.0:  
 temp.append(array\_label[i])  
 print("Межі")  
 print(temp)  
  
  
def findvec(matrix: list[list[float]]) -> list[float]:  
 temp: list[float] = []  
 answer: list[float] = []  
 size: int = len(matrix)  
 temp\_sum: float = 0  
 for i in range(0, size):  
 temp1: float = 1  
 for x in matrix[i]:  
 temp1 \*= x  
 temp.append(temp1 \*\* (1/size))  
 temp\_sum += temp[-1]  
 for x in temp:  
 answer.append(x/temp\_sum)  
 return answer  
  
  
def normalvec(\_vec: list[float]) -> list[float]:  
 maxelem = max(\_vec)  
 answer: list[float] = [x/maxelem for x in \_vec]  
 return answer  
  
  
matrix1: list[list[float]] = []  
matrix2: list[list[float]] = []  
label\_array1: list[int] = [x for x in range(170, 205, 5)]  
label\_array2: list[int] = [x for x in range(150, 185, 5)]  
weight\_array1: list[int] = [1, 3, 4, 5, 7, 8, 9]  
weight\_array2: list[int] = [9, 8, 7, 6, 5, 3, 1]  
for i in weight\_array1:  
 matrix1.append([i/x for x in weight\_array1])  
for i in weight\_array2:  
 matrix2.append([i/x for x in weight\_array2])  
array\_mu1: list[float] = findvec(matrix1)  
array\_mu2: list[float] = findvec(matrix2)  
print("Вектори")  
print(array\_mu1)  
print(array\_mu2)  
print("Вектори після нормалалізації")  
array\_mu1 = normalvec(array\_mu1)  
array\_mu2 = normalvec(array\_mu2)  
print(array\_mu1)  
print(array\_mu2)  
print("Для матриці 1")  
core(label\_array1, array\_mu1)  
border(label\_array1, array\_mu1)  
carrier(label\_array1, array\_mu1)  
print("Для матриці 2")  
core(label\_array2, array\_mu2)  
border(label\_array2, array\_mu2)  
carrier(label\_array2, array\_mu2)  
array\_aprox1 = [aprocS(x, 175, 190) for x in label\_array1]  
array\_aprox2 = [aprocZ(x, 160, 175) for x in label\_array2]  
fig, ax = plt.subplots(2)  
ax[0].plot(label\_array1, array\_mu1, color='#000000', label='Порівняння')  
ax[0].plot(label\_array1, array\_aprox1, linestyle='--', color='#FF0000', linewidth=2, label='Z апроксимація')  
ax[0].legend(loc=2)  
ax[1].plot(label\_array2, array\_mu2, color='#000000', label='Порівняння')  
ax[1].plot(label\_array2, array\_aprox2, linestyle='--', color='#FF0000', linewidth=2, label='S апроксимація')  
ax[1].legend(loc=2)  
plt.show()