Experiment 7 - Implementation of Fuzzy Set Properties using Python

Roll No.	37
Name	Mikil Lalwani
Class	D20 B
Subject	Data Science Lab
LO Mapped	L3: Implement a fuzzy controller system

Aim: To implement Fuzzy Set Properties using python .

Fuzzy Set:

Set: A set is defined as a collection of objects, which share certain characteristics.
Classical set
Classical set is a collection of distinct objects. For example, a set of students passing grades.
Each individual entity in a set is called a member or an element of the set.
The classical set is defined in such a way that the universe of discourse is splitted into two groups members and non-members. Hence, In case of classical sets, no partial membership exists.
Let A is a given set. The membership function can be use to define a set A is given by:

$$\mu A(x)=\{\ 1 \text{ if } x \in A$$

$$0 \text{ if } x \notin A\ \}$$

Fuzzy set:
Fuzzy set is a set having degrees of membership between 1 and 0. Fuzzy sets are represented with tilde characters(~). For example, Number of cars following traffic signals at a particular time out of all cars present will have membership value between [0,1].
Partial membership exists when members of one fuzzy set can also be a part of other fuzzy sets in the same universe.
The degree of membership or truth is not the same as probability, fuzzy truth represents membership in vaguely defined sets.
A fuzzy set A~ in the universe of discourse, U, can be defined as a set of ordered pairs and it is given by

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) | x \in X\}$$

Fuzzy Set Properties:

Common Operations on fuzzy sets: Given two Fuzzy sets A~ and B~

- Union : Fuzzy set C~ is union of Fuzzy sets A~ and B~ :

$$\tilde{C} = \tilde{A} \cup \tilde{B},$$

$$\mu_{\tilde{C}}(x) = \max(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x))$$

- Intersection: Fuzzy set D~ is intersection of Fuzzy sets A~ and B~ :

$$\tilde{D} = \tilde{A} \cap \tilde{B}$$

$$\mu_{\tilde{D}}(x) = \min(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x))$$

- Complement: Fuzzy set E~ is complement of Fuzzy set A~ :

$$\tilde{E} = \mathbb{C}_{\tilde{A}}X$$

-

$$\mu_{\tilde{E}}(x) = 1 - \mu_{\tilde{A}}(x)$$

Some other useful operations on Fuzzy set:

- Algebraic sum:

$$\mu A+B(x) = \mu A(x) + \mu B(x) - \mu A(x).\ \mu B(x)$$

- Algebraic product:

$$\mu A.B(x) = \mu A(x).\ \mu B(x)$$

- Bounded sum:

$$\mu A \oplus B(x) = \min\{1, \mu A(x) + \mu B(x)\}$$

- Bounded difference:

$$\mu A \odot B(c) = \max\{0, \mu A(x) - \mu B(x)\}$$

Python Library Function Used:

1. NumPy (Numerical Python):
   NumPy is a core library for numerical and scientific computing in Python. It provides essential data structures, such as arrays and matrices, along with a collection of mathematical functions to perform operations on these arrays efficiently. NumPy is the foundation for many other scientific libraries and data analysis tools in Python. It's commonly used for tasks like data manipulation, linear algebra, statistical analysis, and numerical simulations.

2. deepcopy
   Deepcopy in Python refers to creating a new object that is a completely independent copy of another object, including all its nested components. This process ensures that changes made to the original object do not affect the copied one, and vice versa. The "copy" module's "deepcopy" function accomplishes this by recursively traversing the object's structure and creating duplicate instances of all objects it encounters.

Code and Observation:

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

[2]:
```
# Max-Min Composition given by Zadeh
def maxMin(x, y):
    z = []
    for x1 in x:
        for y1 in y.T:
            z.append(max(np.minimum(x1, y1)))
    return np.array(z).reshape((x.shape[0], y.shape[1]))
```

```
# Max-Product Composition given by Rosenfeld
def maxProduct(x, y):
    z = []
    for x1 in x:
        for y1 in y.T:
            z.append(max(np.multiply(x1, y1)))
    return np.array(z).reshape((x.shape[0], y.shape[1]))
```

```
# 3 arrays for the example
r1 = np.array([[1, 0, .7], [.3, .2, 0], [0, .5, 1]])
print("R1:\n" ,r1)
r2 = np.array([[.6, .6, 0], [0, .6, .1], [0, .1, 0]])
print("R2:\n" ,r2)
r3 = np.array([[1, 0, .7], [0, 1, 0], [.7, 0, 1]])
print("R3:\n" ,r3)
print("\n")
print("R1oR2 => Max-Min :\n" + str(maxMin(r1, r2)) + "\n")
print("R1oR2 => Max-Product :\n" + str(maxProduct(r1, r2)) + "\n\n")
print("R1oR3 => Max-Min :\n" + str(maxMin(r1, r3)) + "\n")
print("R1oR3 => Max-Product :\n" + str(maxProduct(r1, r3)) + "\n\n")
print("R1oR2oR3 => Max-Min :\n" + str(maxMin(r1, maxMin(r2, r3))) + "\n")
print("R1oR2oR3 => Max-Product :\n" + str(maxProduct(r1, maxProduct(r2, r3))) + "\n")
```

```
R1:
 [[1.  0.  0.7]
 [0.3 0.2 0. ]
 [0.  0.5 1. ]]
R2:
 [[0.6 0.6 0. ]
 [0.  0.6 0.1]
 [0.  0.1 0. ]]
R3:
 [[1.  0.  0.7]
 [0.  1.  0. ]
 [0.7 0.  1. ]]


R1oR2 => Max-Min :
[[0.6 0.6 0. ]
 [0.3 0.3 0.1]
 [0.  0.5 0.1]]

R1oR2 => Max-Product :
[[0.6  0.6  0.  ]
 [0.18 0.18 0.02]
 [0.   0.3  0.05]]


R1oR3 => Max-Min :
[[1.  0.  0.7]
 [0.3 0.2 0.3]
 [0.7 0.5 1. ]]

R1oR3 => Max-Product :
[[1.   0.   0.7 ]
 [0.3  0.2  0.21]
 [0.7  0.5  1.  ]]
```

```
R1oR2oR3 => Max-Min :
[[0.6 0.6 0.6]
 [0.3 0.3 0.3]
 [0.1 0.5 0.1]]

R1oR2oR3 => Max-Product :
[[0.6   0.6   0.42 ]
 [0.18  0.18  0.126]
 [0.035 0.3   0.05 ]]
```

```
from copy import deepcopy
def checkElementHelper(x, S):
    for e in S:
        if x==e[1]: return e[0]
    return 0

def union(setA, setB):
    X,Y = deepcopy(setA),deepcopy(setB)
    Z = []
    for i in X:
        mb = checkElementHelper(i[1],Y)
        Z.append( [max(mb, i[0]), i[1]] )
        if mb!=0:
            Y.remove([mb,i[1]])
    Z = Z + Y
    return Z

def intersection(setA, setB):
    X,Y = deepcopy(setA),deepcopy(setB)
    Z = []
    for i in X:
        mb = checkElementHelper(i[1],Y)
        if min(mb, i[0])!=0:
            Z.append( [min(mb, i[0]), i[1] ])
    return Z

def complement(setA):
```

```
    Z = deepcopy(setA)
    for i in Z:
        i[0] = 1-i[0]
        if i[0]==0:
            Z.remove(i)
    return Z

def fuzzy_sum(setA, setB):
    X,Y = deepcopy(setA),deepcopy(setB)
    Z = []
    for i in X:
        mb = checkElementHelper(i[1],Y)
        if mb!=0:
            m = round( i[0]+mb-(i[0]*mb), 3)
            Z.append( [m, i[1]] )
            Y.remove([mb, i[1]])
        else:
            Z.append(i)
    Z = Z + Y
    return Z

#Algebraic Product
def fuzzy_product(setA, setB):
    X,Y = deepcopy(setA),deepcopy(setB)
    Z = []
    for i in X:
        mb = checkElementHelper(i[1],Y)
        if mb!=0:
            m = i[0]*mb
            Z.append( [m, i[1]] )
            Y.remove([mb, i[1]])
    return Z

def bounded_sum(setA, setB):
    X,Y = deepcopy(setA),deepcopy(setB)
    Z = []
    for i in X:
        mb = checkElementHelper(i[1],Y)
        if mb!=0:
```

```
            m = i[0]+mb
            Z.append( [min(1,m), i[1]] )
            Y.remove([mb, i[1]])
        else:
            Z.append(i)
    Z = Z + Y
    return Z

def bounded_diff(setA, setB):
    X,Y = deepcopy(setA),deepcopy(setB)
    Z = []
    for i in X:
        mb = checkElementHelper(i[1],Y)
        if mb!=0:
            m = round( i[0]-mb, 3)
            Z.append([max(0,m), i[1]] )
            Y.remove([mb, i[1]])
        else:
            Z.append(i)
    return Z

def menu():
    print("------ Fuzzy Operations -------")
    print('')
    print("[0] Declare new A,B sets")
    print("[1] Union")
    print("[2] Intersection")
    print("[3] Complement A")
    print("[4] Complement B")
    print("[5] Algebraic Sum")
    print("[6] Algebraic Product")
    print("[7] Bounded Sum of A and B")
    print("[8] Bounded Difference of A and B")
    print("[9] Quit")
    print("")

def newInputs():
    global globalSetA
    global globalSetB
```

```
    del globalSetA[:]
    del globalSetB[:]
    globalSetA = []
    globalSetB = []
    print("----- Set A -----")
    n = int(input("[INPUT] Enter no. of values: \n"))
    print("\n----- Populating SET A -----")
    for i in range(n):
        n,d = map(float,input(f"[SET A INPUT - {i+1}] Enter values (0.1/1) -> (0.1 1)"))
        globalSetA.append([n,d])
    print("---- Set B -----")
    n = int(input("[INPUT] Enter no. of values: \n"))

    print("\n----- Populating SET B -----")
    for i in range(n):
        n,d = map(float,input(f"[SET B INPUT - {i+1}] Enter values (0.1/1) -> (0.1 1)"))
        globalSetB.append([n,d])
    print("\n[SUCCESS] A, B Sets UPDATED\n")
    print("[SET A] : ",globalSetA)
    print("[SET B] : ",globalSetB)

def fuzzyOperate(c):
    if (c==0):
        print("[NEW SETS] \n")
        newInputs()
    elif (c==1):
        print("[PERFORMING] Union Operation\n")
        print("[OUTPUT]:")
        print(union(globalSetA,globalSetB))
    elif (c==2):
        print("[PERFORMING] Intersection Operation\n")
        print("[OUTPUT]:")
        print(intersection(globalSetA,globalSetB))
    elif (c==3):
        print("[PERFORMING] Complement A Operation\n")
        print("[OUTPUT]:")
        print(complement(globalSetA))
    elif (c==4):
        print("[PERFORMING] Complement B Operation\n")
```

```
        print("[OUTPUT]:")
        print(complement(globalSetB))
    elif (c==5):
        print("[PERFORMING] Algebraic Sum Operation\n")
        print("[OUTPUT]:")
        print(fuzzy_sum(globalSetA,globalSetB))
    elif (c==6):
        print("[PERFORMING] Algebraic Product Operation\n")
        print("[OUTPUT]:")
        print(fuzzy_product(globalSetA,globalSetB))
    elif (c==7):
        print("[PERFORMING] Bounded Sum of A and B Operation\n")
        print("[OUTPUT]:")
        print(bounded_sum(globalSetA,globalSetB))
    elif (c==8):
        print("[PERFORMING] Bounded Difference of A and B Operation\n")
        print("[OUTPUT]:")
        print(bounded_diff(globalSetA,globalSetB))
    else:
        print("[ERROR] Invalid input!")

f= True
global globalSetA
globalSetA=[[0.1,0],[0.2,1],[0.3,2],[0.4,3],[0.5,4]]
global globalSetB
globalSetB=[[0.5,0],[0.4,1],[0.3,2],[0.2,3],[0.1,4]]

print("^^^^^***Fuzzy Set Operations***^^^^^\n")
print("")
print("[INITIALIZING]")

print("[SET A] : ",globalSetA)
print("[SET B] : ",globalSetB)
print("")

while f:
    menu()
    c = int(input("[INPUT] Enter your choice:\n"))
    print("\n------------------")
```

```
 if (c == 9) :
    f= False
    break
 else:
    fuzzyOperate(c)
 print("------------------\n")
```

```
^^^^^^^*Fuzzy Set Operations***^^^^^^


[INITIALIZING]
[SET A] :  [[0.1, 0], [0.2, 1], [0.3, 2], [0.4, 3], [0.5, 4]]
[SET B] :  [[0.5, 0], [0.4, 1], [0.3, 2], [0.2, 3], [0.1, 4]]

------ Fuzzy Operations -------

[0] Declare new A,B sets
[1] Union
[2] Intersection
[3] Complement A
[4] Complement B
[5] Algebraic Sum
[6] Algebraic Product
[7] Bounded Sum of A and B
[8] Bounded Difference of A and B
[9] Quit

[INPUT] Enter your choice:
 1


------------------
[PERFORMING] Union Operation

[OUTPUT]:
[[0.5, 0], [0.4, 1], [0.3, 2], [0.4, 3], [0.5, 4]]
------------------
```

```
------ Fuzzy Operations -------

[0] Declare new A,B sets
[1] Union
[2] Intersection
[3] Complement A
[4] Complement B
[5] Algebraic Sum
[6] Algebraic Product
[7] Bounded Sum of A and B
[8] Bounded Difference of A and B
[9] Quit

[INPUT] Enter your choice:
 2


------------------
[PERFORMING] Intersection Operation

[OUTPUT]:
[[0.1, 0], [0.2, 1], [0.3, 2], [0.2, 3], [0.1, 4]]
------------------
```

```
------ Fuzzy Operations -------

[0] Declare new A,B sets
[1] Union
[2] Intersection
[3] Complement A
[4] Complement B
[5] Algebraic Sum
[6] Algebraic Product
[7] Bounded Sum of A and B
[8] Bounded Difference of A and B
[9] Quit

[INPUT] Enter your choice:
 3


------------------
[PERFORMING] Complement A Operation

[OUTPUT]:
[[0.9, 0], [0.8, 1], [0.7, 2], [0.6, 3], [0.5, 4]]
------------------
```

Conclusion:

Thus we have successfully implemented Fuzzy Set Properties using Python.