Name: Nachiket Erlekar

Roll no: 41434 Batch: Q4

Assignment 1

Title: Fuzzy Set Operations

Problem Statement:

Implement Union, Intersection, Complement and Difference operations on fuzzy sets. Also create fuzzy relation by Cartesian product of any two fuzzy sets and perform max-min composition on any two fuzzy relations.

Objectives:

1) To understand concept of fuzzy sets and fuzzy logic
2) To be able to implement Fuzzy-set operations

Outcome:

1) We will understand the concept of fuzzy logic
2) We will be able to implement set operations on fuzzy sets

Requirements:

1) Python 3.7+
2) Jupyter Notebook
3) Windows 64-bit OS

Theory:

In logic, fuzzy logic is a form of many-valued logic in which the truth value of variables may be any real number between 0 and 1 both inclusive. It is employed to handle the concept of partial truth, where the truth value may range between completely true and completely false.[1] By contrast, in Boolean logic, the truth values of variables may only be the integer values 0 or 1.

Fuzzy logic is based on the observation that people make decisions based on imprecise and non-numerical information. Fuzzy models or sets are mathematical means of representing vagueness and imprecise information (hence the term fuzzy). These models have the capability of recognizing, representing, manipulating, interpreting, and utilizing data and information that are vague and lack certainty. Fuzzy logic has been applied to many fields, from control theory to artificial intelligence.

A fuzzy set is a collection of ordered pairs of an element, and the degree of its membership. In classical set theory, the membership of elements in a set is assessed in binary terms according to a bivalent condition — an element either belongs or does not belong to the set. By contrast, fuzzy set theory permits the gradual assessment of the membership of elements in a set; this is described with the aid of a membership function valued in the real unit interval [0, 1]

Following are some set operations that are defined on fuzzy sets:

- For a given fuzzy set **A**, its **complement** **~A** is defined by the following membership function:

  $$\forall x \in U : u_{\sim A}(x) = 1 - u_A(x)$$

- For given fuzzy sets **A** and **B,** their intersection is defined by the following membership function

  $$\forall x \in U : u_{A \cap B}(x) = min(u_A(x), u_B(x))$$

- For given fuzzy sets **A** and **B,** their union is defined by the following membership function:

  $\forall x \in U : u_{A \sqcup B}(x) = max(u_A(x),\ u_B(x))$

- For given fuzzy sets **A** and **B,** their difference is defined by the following membership function:

  $\forall x \in U : u_{A-B}(x) = u_{A \sqcap \sim B}(x) = max(u_A(x),\ u_{\sim B}(x))$

- For given fuzzy sets **A** and **B,** their Cartesian product is defined by the following membership function

  $\forall x,\ y \in U : u_{A \times B}(x,\ y) = min(u_A(x),\ u_B(y))$

- For given fuzzy relations **R**(x, y) and **S**(y, z)**,** their min-max composition **T**(x, z) is defined by the following membership function

  $\forall x,\ y,\ z \in U : u_T(x,\ z) = max(u_R(x,\ y),\ u_S(y,\ z))$

<u>Test Cases</u>:

Consider two fuzzy sets as follows:

A: {'x1': 0.5, 'x2': 0.1, 'x3': 0.4, 'x4': 0.6}
B: {'x1': 0.2, 'x2': 0.3, 'x3': 0.5, 'x5': 0.7}

Union: A U B = {'x1': 0.5, 'x2': 0.3, 'x3': 0.5, 'x4': 0.6, 'x5': 0.7}

Intersection: {'x1': 0.2, 'x2': 0.1, 'x3': 0.4}

~A: {'x1': 0.5, 'x2': 0.9, 'x3': 0.6, 'x4': 0.4}

~B: {'x1': 0.8, 'x2': 0.7, 'x3': 0.5, 'x5': 0.3}

A – B: {'x1': 0.5, 'x2': 0.1, 'x3': 0.4}

A x B:

.	b1	b2	b3
a1	0.2	0.3	0.5
a2	0.1	0.1	0.1
a3	0.2	0.3	0.4
a4	0.2	0.3	0.5

For

A = {"a1": 0.5, "a2": 0.1, "a3": 0.4, "a4": 0.6}
B = {"b1": 0.2, "b2": 0.3, "b3": 0.5}
C = {"c1": 0.7, "c2": 0.4, "c3": 0.9, "c4": 0.3}

Min-max Composition:

.	c1	c2	c3	c4
a1	0.5	0.4	0.5	0.3
a2	0.1	0.1	0.1	0.1
a3	0.4	0.4	0.4	0.3
a4	0.5	0.4	0.5	0.3

Conclusion:

Thus I successfully implemented various fuzzy set-operations on sample fuzzy sets.

Assignment 1(SCOA)

Code

```
import numpy as np

class FuzzySet:
    def __init__(self, iterable: any):
        self.f_set = set(iterable)
        self.f_list = list(iterable)
        self.f_len = len(iterable)
        for elem in self.f_set:
            if not isinstance(elem, tuple):
                raise TypeError("No tuples in the fuzzy set")
            if not isinstance(elem[1], float):
                raise ValueError("Probabilities not assigned to
elements")

    def __or__(self, other):
        if len(self.f_set) != len(other.f_set):
            raise ValueError("Length of the sets is different")
        f_set = [x for x in self.f_set]
        other = [x for x in other.f_set]
        return FuzzySet([f_set[i] if f_set[i][1] > other[i][1] else
other[i] for i in range(len(self))])

    def __and__(self, other):
        if len(self.f_set) != len(other.f_set):
            raise ValueError("Length of the sets is different")
        f_set = [x for x in self.f_set]
        other = [x for x in other.f_set]
        return FuzzySet([f_set[i] if f_set[i][1] < other[i][1] else
other[i] for i in range(len(self))])

    def __invert__(self):
        f_set = [x for x in self.f_set]
        for indx, elem in enumerate(f_set):
            f_set[indx] = (elem[0], float(round(1 - elem[1], 2)))
        return FuzzySet(f_set)

    def __sub__(self, other):
        if len(self) != len(other):
            raise ValueError("Length of the sets is different")
        return self & ~other

    @staticmethod
    def max_min(array1: np.ndarray, array2: np.ndarray):
        tmp = np.zeros((array1.shape[0], array2.shape[1]))
        t = list()
        for i in range(len(array1)):
            for j in range(len(array2[0])):
                for k in range(len(array2)):
                    t.append(round(min(array1[i][k], array2[k][j]), 2))
                tmp[i][j] = max(t)
                t.clear()
        return tmp
```

```
    def __len__(self):
        self.f_len = sum([1 for i in self.f_set])
        return self.f_len

    def __str__(self):
        return f'{[x for x in self.f_set]}'

    def __getitem__(self, item):
        return self.f_list[item]

    def __iter__(self):
        for i in range(len(self)):
            yield self[i]


a = FuzzySet({('x1', 0.5), ('x2', 0.7), ('x3', 0.0)})
b = FuzzySet({('x1', 0.8), ('x2', 0.2), ('x3', 1.0)})
print(f'a -> {a}')
print(f'b -> {b}\n')
print(f'Fuzzy union: {a | b}')
print(f'Fuzzy intersection: {a & b}\n')
print(f"Fuzzy inversion of a:  {~a}")
print(f'Fuzzy inversion of b: {~b}')
print(f'Fuzzy Subtraction: {a - b}')

r = np.array([[0.6, 0.6, 0.8, 0.9], [0.1, 0.2, 0.9, 0.8], [0.9, 0.3,
0.4, 0.8], [0.9, 0.8, 0.1, 0.2]])
s = np.array([[0.1, 0.2, 0.7, 0.9], [1.0, 1.0, 0.4, 0.6], [0.0, 0.0,
0.5, 0.9], [0.9, 1.0, 0.8, 0.2]])
print(f"Max Min: of \n{r} \nand \n{s}\n:")
print(FuzzySet.max_min(r, s))
```

Output—

```
a -> [('x1', 0.5), ('x2', 0.7), ('x3', 0.0)]
b -> [('x1', 0.8), ('x2', 0.2), ('x3', 1.0)]

Fuzzy union: [('x1', 0.8), ('x2', 0.7), ('x3', 1.0)]
Fuzzy intersection: [('x1', 0.5), ('x2', 0.2), ('x3', 0.0)]

Fuzzy inversion of a:  [('x1', 0.5), ('x2', 0.3), ('x3', 1.0)]
Fuzzy inversion of b: [('x1', 0.2), ('x2', 0.8), ('x3', 0.0)]
Fuzzy Subtraction: [('x1', 0.2), ('x2', 0.7), ('x3', 0.0)]
Max Min: of
[[0.6 0.6 0.8 0.9]
 [0.1 0.2 0.9 0.8]
 [0.9 0.3 0.4 0.8]
 [0.9 0.8 0.1 0.2]]
and
[[0.1 0.2 0.7 0.9]
 [1.  1.  0.4 0.6]
 [0.  0.  0.5 0.9]
 [0.9 1.  0.8 0.2]]
:
[[0.9 0.9 0.8 0.8]
 [0.8 0.8 0.8 0.9]
 [0.8 0.8 0.8 0.9]
 [0.8 0.8 0.7 0.9]]

In [2]:
```