BHARATIYA VIDYA BHAVAN’S

SARDAR PATEL INSTITUTE OF TECHNOLOGY

(Empowered Autonomous Institute Affiliated to University of Mumbai)

[Knowledge is Nectar]

Department of Computer Engineering

COURSE :- ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING

UID	2024801005
Name	Vedant Kannurkar
Class and Batch	TE Computer Science and Engineering - Batch B2
Date	01-10-2025
Experiment No	8
Lab	ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING (412)
Aim	**To implement Fuzzy Sets and Fuzzy Relations for a given problem statement** Two fuzzy sets represent **Sunny (S)** and **Rainy (R)** days based on sensor confidence readings: $S = \{0.9/clear,\ 0.7/hazy,\ 0.3/cloudy,\ 0.1/rainy,\ 0.0/stormy\}$ $R = \{0.1/clear,\ 0.4/hazy,\ 0.7/cloudy,\ 0.9/rainy,\ 1.0/stormy\}$ Find: a) $S \cup R$ b) $S \cap R$ c) $\overline{S}$ d) $\overline{R}$ e) $\overline{S \cup R}$ f) $\overline{S \cap R}$ g) $R - S$ h) $S - R$ **Part 2 :**

A smart traffic controller adjusts signal timing based on **vehicle density (V)** and **road condition (R)**, which in turn affects **traffic delay (D)**.

Given fuzzy relations:

$$R_1(V,R) = \begin{bmatrix} 0.8 & 0.6 & 0.4 \\ 0.7 & 0.9 & 0.5 \\ 0.5 & 0.7 & 0.8 \end{bmatrix}, \quad R_2(R,D) = \begin{bmatrix} 0.9 & 0.6 & 0.3 \\ 0.5 & 0.8 & 0.7 \\ 0.4 & 0.6 & 0.9 \end{bmatrix}$$

Compute the **fuzzy composition**

$$R(V,D) = R_1(V,R) \circ R_2(R,D)$$

using **max–min** and **max–product** methods.
Interpret which method gives more realistic results for traffic delay prediction.

Theory

Fuzzy sets are an extension of classical sets that allow elements to have **partial membership**, rather than just being entirely in or out. Each element in a fuzzy set is associated with a **membership degree** between 0 and 1, which indicates how strongly the element belongs to the set. This degree is defined by a **membership function**.

Key Points:

- In classical sets, membership is binary: 0 (not in set) or 1 (in set).
- In fuzzy sets, membership values can range between 0 and 1, representing partial membership.
- Membership functions map elements from the universe of discourse to a real number in [0,1].

Example:
Consider the fuzzy set "Hot" over temperature:

- 40°C → membership 1 (definitely hot)
- 30°C → membership 0.6 (moderately hot)
- 20°C → membership 0 (not hot)

Operations on Fuzzy Sets:

- **Union:** Maximum of membership values for each element.
- **Intersection:** Minimum of membership values.
- **Complement:** 1 minus the membership value.

Fuzzy sets are useful for modeling **vague or uncertain information** in real-world systems, such as control systems, decision-making, and pattern recognition, allowing more flexible and human-like reasoning than classical sets.

Procedure

1. Define the fuzzy sets S and R with elements and their membership values.
2. Perform **union** by taking the maximum membership value for each element from both sets.
3. Perform **intersection** by taking the minimum membership value for each element from both sets.
4. Compute the **complement** of a set by subtracting each membership value from 1.
5. Compute the **difference** between two sets by subtracting membership values and taking 0 if the result is negative.
6. Perform combined operations like union and intersection with complements as needed.
7. Display all results of union, intersection, complement, and difference operations.

Implementation Code

Part 1

```
S = {"clear": 0.9, "hazy": 0.7, "cloudy": 0.3, "rainy": 0.1, "stormy": 0.0}
R = {"clear": 0.1, "hazy": 0.4, "cloudy": 0.7, "rainy": 0.9, "stormy": 1.0}

def union(set1, set2):
    result = {}
```

```
    all_keys = set1.keys() | set2.keys()
    for key in all_keys:
        result[key] = max(set1.get(key, 0), set2.get(key, 0))
    return result

def intersection(set1, set2):
    result = {}
    all_keys = set1.keys() | set2.keys()
    for key in all_keys:
        result[key] = min(set1.get(key, 0), set2.get(key, 0))
    return result

def complement(set1):
    result = {}
    for key, value in set1.items():
        result[key] = round(1 - value,2)
    return result

def difference(set1, set2):
    result = {}
    for key in set1.keys():
        result[key] = round(max(set1.get(key, 0) - set2.get(key, 0),
0),2)
    return result

print(f"S ∪ R: {union(S, R)}")
print(f"S ∩ R: {intersection(S, R)}")
print(f"S̄: {complement(S)}")
print(f"R̄: {complement(R)}")
print(f"S ∪ R̄: {union(S, complement(R))}")
print(f"S ∩ R̄: {intersection(S, complement(R))}")
print(f"R - S: { difference(R, S)}")
print(f"S - R: { difference(S, R)}")
```

Part 2

```
import numpy as np

print("Part 2: Fuzzy Relation Composition")

V_R = np.array([
    [0.8, 0.6, 0.4],
```

	`` [0.7, 0.9, 0.5], [0.5, 0.7, 0.8]]) R_D = np.array([[0.9, 0.6, 0.3], [0.5, 0.8, 0.7], [0.4, 0.6, 0.9]]) def max_min_comp(A, B): rA, cA = A.shape rB, cB = B.shape if cA != rB: raise ValueError("Incompatible matrices for composition") C = np.zeros((rA, cB)) for i in range(rA): for j in range(cB): C[i, j] = np.max(np.minimum(A[i, :], B[:, j])) return C def max_prod_comp(A, B): rA, cA = A.shape rB, cB = B.shape if cA != rB: raise ValueError("Incompatible matrices for composition") C = np.zeros((rA, cB)) for i in range(rA): for j in range(cB): C[i, j] = np.max(A[i, :] * B[:, j]) return C R_vm_min = max_min_comp(V_R, R_D) print("R(V,D) MAX-MIN:") print(R_vm_min) R_vm_prod = max_prod_comp(V_R, R_D) print("\nR(V,D) MAX-PROD:") print(R_vm_prod) ```
Output	**Output:**

Part 1

```
S ∪ R: {'cloudy': 0.7, 'clear': 0.9, 'rainy': 0.9, 'stormy': 1.0, 'hazy': 0.7}
S ∩ R: {'cloudy': 0.3, 'clear': 0.1, 'rainy': 0.1, 'stormy': 0.0, 'hazy': 0.4}
S̄: {'clear': 0.1, 'hazy': 0.3, 'cloudy': 0.7, 'rainy': 0.9, 'stormy': 1.0}
R̄: {'clear': 0.9, 'hazy': 0.6, 'cloudy': 0.3, 'rainy': 0.1, 'stormy': 0.0}
S ∪ R̄: {'cloudy': 0.3, 'clear': 0.9, 'rainy': 0.1, 'stormy': 0.0, 'hazy': 0.7}
S ∩ R̄: {'cloudy': 0.3, 'clear': 0.9, 'rainy': 0.1, 'stormy': 0.0, 'hazy': 0.6}
R - S: {'clear': 0, 'hazy': 0, 'cloudy': 0.4, 'rainy': 0.8, 'stormy': 1.0}
S - R: {'clear': 0.8, 'hazy': 0.3, 'cloudy': 0, 'rainy': 0, 'stormy': 0}
```

Part 2.

```
Part 2: Fuzzy Relation Composition
R(V,D) MAX-MIN:
[[0.8 0.6 0.6]
 [0.7 0.8 0.7]
 [0.5 0.7 0.8]]

R(V,D) MAX-PROD:
[[0.72 0.48 0.42]
 [0.63 0.72 0.63]
 [0.45 0.56 0.72]]
```

Conclusion	Implemented Fuzzy Sets and Fuzzy Relations for weather conditions. Performed fuzzy set operations on sunny and rainy days and computed fuzzy compositions for smart traffic controller using Max-Min and Max-Product methods. Gained practical understanding of how membership values combine, how fuzzy relations propagate uncertainty, and the differences between composition methods.