

The background of the slide is a complex, abstract pattern composed of numerous triangles in various shades of purple, blue, and black. The triangles are arranged in a way that creates a sense of depth and movement, with some triangles appearing to point towards the viewer and others away. The overall effect is a modern, digital aesthetic.

AI LAB WEEK 9

Dr. Mubashir Ahmad

APPLY FUZZY LOGIC

- It is necessary to compare two sensors based upon their detection levels and gain settings. The table of gain settings and sensor detection levels with a standard item being monitored providing typical membership values to represent the detection levels for each sensor is given in Table.

| Gain | Detection Level Sensor 1 | Detection Level Sensor 2 |
|------|-----------------------------|-----------------------------|
| 0 | 0 | 0 |
| 10 | 0.2 | 0.35 |
| 20 | 0.35 | 0.25 |
| 30 | 0.65 | 0.8 |
| 40 | 0.85 | 0.95 |
| 50 | 1 | 1 |

APPLY FUZZY LOGIC

- Now given the universe of discourse $X = \{0, 10, 20, 30, 40, 50\}$ and the membership functions for the two sensors in discrete form as

$$\tilde{D}_1 = \left\{ \frac{0}{0} + \frac{0.2}{10} + \frac{0.35}{20} + \frac{0.65}{30} + \frac{0.85}{40} + \frac{1}{50} \right\}$$
$$\tilde{D}_2 = \left\{ \frac{0}{0} + \frac{0.35}{10} + \frac{0.25}{20} + \frac{0.8}{30} + \frac{0.95}{40} + \frac{1}{50} \right\}$$

| Gain | Detection Level Sensor 1 | Detection Level Sensor 2 |
|------|-----------------------------|-----------------------------|
| 0 | 0 | 0 |
| 10 | 0.2 | 0.35 |
| 20 | 0.35 | 0.25 |
| 30 | 0.65 | 0.8 |
| 40 | 0.85 | 0.95 |
| 50 | 1 | 1 |





INTERSECTION?



DIFFERENCE?

The background features a vibrant red sunburst pattern with numerous black lines radiating from a central point. Overlaid on this is a complex, multi-colored wireframe sphere. The sphere is composed of intersecting lines in shades of green, blue, and yellow, creating a mesh-like structure. The word "COMPLEMENT?" is centered in white, bold, sans-serif capital letters, with a subtle drop shadow, positioned over the lower half of the wireframe sphere.

COMPLEMENT?



MAX MIN COMPOSITION?



MAX PRODUCT COMPOSITION?

```
    for object to mirror  
    mirror_mod.mirror_object =  
    operation == "MIRROR_X":  
    mirror_mod.use_x = True  
    mirror_mod.use_y = False  
    mirror_mod.use_z = False  
    operation == "MIRROR_Y":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = True  
    mirror_mod.use_z = False  
    operation == "MIRROR_Z":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = False  
    mirror_mod.use_z = True
```

```
    selection at the end -add  
    mirror_ob.select= 1  
    modifier_ob.select=1  
    context.scene.objects.active  
    ("Selected" + str(modifier_ob))  
    mirror_ob.select = 0  
    bpy.context.selected_objects  
    = bpy.context.selected_objects  
    if len(selected) > 0:  
        please select the  
        OPERATOR CLASS
```

```
types.Operator):  
    X mirror to the selected  
    object.mirror_mirror_x"  
    ror X"
```

PYTHON CODE

TASK 1, 2, 3 AND 4

```
import numpy as np
# Membership values (gain settings)
membership_values = [0, 10, 20, 30, 40, 50]
# Detection levels for Sensor 1 and Sensor 2
sensor1 = [0, 0.2, 0.35, 0.65, 0.85, 1]
sensor2 = [0, 0.35, 0.25, 0.8, 0.95, 1]
# Union
union = np.maximum(sensor1, sensor2)
print("Union: ", dict(zip(membership_values, union)))
```


TASK 1, 2, 3 AND 4

Intersection

```
intersection = np.minimum(sensor1, sensor2)
```

```
print("Intersection: ", dict(zip(membership_values, intersection)))
```

Difference (Sensor1 - Sensor2)

```
difference = np.maximum(0, np.array(sensor1) - np.array(sensor2))
```

```
print("Difference (Sensor1 - Sensor2): ",  
dict(zip(membership_values, difference)))
```

TASK 1, 2, 3 AND 4

Complement

```
complement_sensor1 = [1 - d for d in sensor1]
```

```
complement_sensor2 = [1 - d for d in sensor2]
```

```
print("Complement of Sensor 1: ", dict(zip(membership_values,  
complement_sensor1)))
```

```
print("Complement of Sensor 2: ", dict(zip(membership_values,  
complement_sensor2)))
```

TASK 5 AND 6

```
import numpy as np

# Define sets A and B
A = np.array([[0.6, 0.3], [0.2, 0.9]])
B = np.array([[1, 0.5, 0.3], [0.8, 0.4, 0.7]])

# Max-Min Composition
maxmin_min_composition = np.zeros((A.shape[0], B.shape[1]))
```


TASK 5 AND 6

```
for i in range(A.shape[0]):  
    for j in range(B.shape[1]):  
        min_values = [min(A[i, k], B[k, j]) for k in range(A.shape[1])]  
        max_min_composition[i, j] = max(min_values)  
  
print("Max_Min_Composition:")  
print(max_min_composition)
```

TASK 5 AND 6

```
# Max-Product Composition
max_product_composition = np.zeros((A.shape[0], B.shape[1]))
for i in range(A.shape[0]):
    for j in range(B.shape[1]):
        product_values = [A[i, k] * B[k, j] for k in range(A.shape[1])]
        max_product_composition[i, j] = max(product_values)
print("Max-Product Composition:")
print(max_product_composition)
```

TASKS

Task#1: Find the union according to the Gain value of sensor1 and 2 detection values.

Task#2: Find the intersection.

Task#3: Find the difference

Task#4: Find the Complement.

Task#5: Min max composition.

Task#6: Max product of two fuzzy sets.