Ramesh Chandra Soren

Enrollment No: 2022CSB086

Department: Computer Science and Technology

Fuzzy Numbers "Approximately Equals to 4" Using Different Membership Functions

1. Trapezoidal Membership Function

A trapezoidal membership function is defined by four parameters: `a`, `b`, `c`, and `d`. For the fuzzy number "approximately equals to 4", the shape could look like this:

We can define this function as follows:

[μ(x) = \begin{cases} 0, & x < a \ \frac{x - a}{b - a}, & a \leq x < b \ 1, & b \leq x \leq c \ \frac{d - x}{d - c}, & c < x \leq d \ 0, & x > d \end{cases}]

For example, using values `a = 2`, `b = 3.5`, `c = 4.5`, `d = 6`, the function is trapezoidal, where values between 3.5 and 4.5 have full membership.

2. Triangular Membership Function

A triangular membership function has a peak at the center and linearly decreases on both sides. It can be represented as:

For the fuzzy number "approximately equals to 4", the triangular membership function can be written as:

[\mu(x) = \begin{cases} 0, & x < a \ \frac{x - a}{b - a}, & a \leq x < b \ \frac{c - x}{c - b}, & b \leq x < c \ 0, & x > c \end{cases}]

For example, using values `a = 2.5`, `b = 4`, and `c = 5.5`, the membership function would peak at `x = 4`.

3. Gaussian Membership Function

A Gaussian membership function has the form:

μ(x) = exp(-(x - c)^2 / (2σ^2))

Where:

- `c` is the center of the fuzzy set (i.e., the value around which "approximately equals to 4" is centered).
- `σ` controls the width of the Gaussian curve.

For example, if `c = 4` and `σ = 1`, the Gaussian function would represent a smooth "approximately equals to 4" fuzzy number.

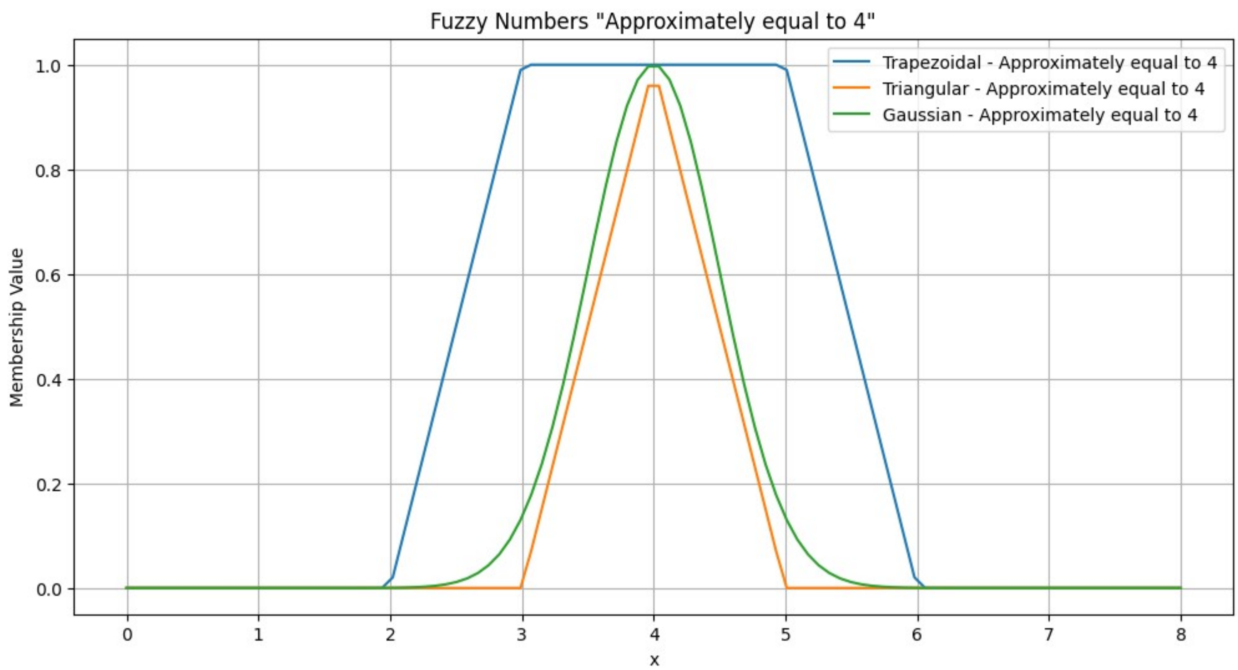
Graphical Representation of Each:

- **Trapezoidal**: Starts rising from 0 at `x = 2`, reaches full membership at `x = 3.5`, stays constant at 1 until `x = 4.5`, and then decreases to 0 at `x = 6`.
- **Triangular**: Peaks at `x = 4` with values tapering to 0 at `x = 2.5` and `x = 5.5`.
- **Gaussian**: A smooth bell curve centered at `x = 4` with spread determined by `σ`.

This captures the "approximately equals to 4" concept using different membership function shapes.

```
import numpy as np
import matplotlib.pyplot as plt

# Trapezoidal membership function
def trapezoidal(x, a, b, c, d):
    if x <= a or x >= d:
        return 0
    elif a < x < b:
        return (x - a) / (b - a)
    elif b <= x <= c:
        return 1
    else:
        return (d - x) / (d - c)

# Triangular membership function
def triangular(x, a, b, c):
    if x <= a or x >= c:
        return 0
    elif a < x <= b:
        return (x - a) / (b - a)
    else:
        return (c - x) / (c - b)

# Gaussian membership function
def gaussian(x, c, sigma):
    return np.exp(-((x - c) ** 2) / (2 * sigma ** 2))

# Generate x values
x = np.linspace(0, 8, 100)

# Calculate membership values
y_trapezoidal = np.array([trapezoidal(xi, 2, 3, 5, 6) for xi in x])
y_triangular = np.array([triangular(xi, 3, 4, 5) for xi in x])
y_gaussian = gaussian(x, 4, 0.5)

# Plotting the results
plt.figure(figsize=(12, 6))
```

```
plt.plot(x, y_trapezoidal, label='Trapezoidal - Approximately equal to 4')
plt.plot(x, y_triangular, label='Triangular - Approximately equal to 4')
plt.plot(x, y_gaussian, label='Gaussian - Approximately equal to 4')
plt.title('Fuzzy Numbers "Approximately equal to 4"')
plt.xlabel('x')
plt.ylabel('Membership Value')
plt.legend()
plt.grid(True)
plt.show()
```



Fuzzy Sets of Temperature in a City

Let **C(x)** and **D(x)** be two fuzzy sets representing temperature in a city, where:

- **C(x)**: "Cold"
- **D(x)**: "Warm"

Fuzzy Set C: "Cold"

The membership function for the fuzzy set **C(x)** (Cold) is defined as:

[\mu_C(x) = \begin{cases} 0 & \text{if } x < -10 \quad (\text{Definitely Cold}) \ \frac{x - (-10)}{0 - (-10)} & \text{if } -10 \leq x < 0 \quad (\text{Partially Cold}) \ 1 & \text{if } 0 \leq x < 5 \quad (\text{Cold}) \ 0 & \text{if } x \geq 5 \quad (\text{Not Cold}) \end{cases}]

Explanation:

- **$x < -10$**: Temperature is definitely cold (0 membership).
- **$-10 \leq x < 0$**: Temperature is partially cold, increasing linearly from 0 to 1.
- **$0 \leq x < 5$**: Temperature is fully cold (membership = 1).
- **$x \geq 5$**: Temperature is not considered cold (0 membership).

Fuzzy Set D: "Warm"

The membership function for the fuzzy set **D(x)** (Warm) is defined as:

[\mu_D(x) = \begin{cases} 1 & \text{if } x < 25 \quad (\text{Not Warm}) \ \frac{x - 25}{35 - 25} & \text{if } 25 \leq x < 35 \quad (\text{Partially Warm}) \ 1 & \text{if } 35 \leq x < 40 \quad (\text{Warm}) \ 0 & \text{if } x \geq 40 \quad (\text{Definitely Not Warm}) \end{cases}]

Explanation:

- **$x < 25$**: Temperature is not considered warm (1 membership).
- **$25 \leq x < 35$**: Temperature is partially warm, increasing linearly from 0 to 1.
- **$35 \leq x < 40$**: Temperature is fully warm (membership = 1).
- **$x \geq 40$**: Temperature is definitely not warm (0 membership).

```
def membership_cold(x):
    """Calculate membership value for fuzzy set C (Cold)"""
    if x < -10:
        return 0
    elif -10 <= x < 0:
        return (x + 10) / 10
    elif 0 <= x < 5:
        return 1
    else:
        return 0

def membership_warm(x):
    """Calculate membership value for fuzzy set D (Warm)"""
    if x < 25:
        return 1
    elif 25 <= x < 35:
        return (x - 25) / 10
    elif 35 <= x < 40:
        return 1
    else:
        return 0

# Test the membership functions for a range of temperatures
temperatures = [-15, -10, 0, 3, 10, 25, 30, 35, 38, 42]

print("Temperature  |  Cold (C)  |  Warm (D)")
print("-------------------------------------")
for temp in temperatures:
```

```
    cold_val = membership_cold(temp)
    warm_val = membership_warm(temp)
    print(f"{temp:12} |  {cold_val:8.3f}  |  {warm_val:8.3f}")
```

```
Temperature  |  Cold (C)  |  Warm (D)
-------------------------------------
         -15 |     0.000  |     1.000
         -10 |     0.000  |     1.000
           0 |     1.000  |     1.000
           3 |     1.000  |     1.000
          10 |     0.000  |     1.000
          25 |     0.000  |     0.000
          30 |     0.000  |     0.500
          35 |     0.000  |     1.000
          38 |     0.000  |     1.000
          42 |     0.000  |     0.000
```

Mamdani Fuzzy Inference System for Controlling Furnace FAN-SPEED

We will design a Mamdani FIS to control the **FAN-SPEED** of a furnace by taking the **TEMPERATURE** from a thermostat as input. The objective is to regulate the fan speed based on the input temperature using fuzzy logic.

1. Fuzzy Variables

Input: **TEMPERATURE**

The input linguistic variable **TEMPERATURE** is classified into the following fuzzy sets:

- **Risky** (High Temperature)
- **Average** (Moderate Temperature)
- **Excellent** (Low Temperature)

Output: **FAN-SPEED**

The output linguistic variable **FAN-SPEED** is classified into the following fuzzy sets:

- **Slow**
- **Moderate**
- **High**

2. Fuzzy Rules

The system's behavior is governed by the following If-Then fuzzy rules:

1. **If TEMPERATURE is Risky, then FAN-SPEED is High.**
2. **If TEMPERATURE is Average, then FAN-SPEED is Moderate.**
3. **If TEMPERATURE is Excellent, then FAN-SPEED is Slow.**
4. **If TEMPERATURE is Risky and exceeds a threshold, then increase FAN-SPEED to maximum (e.g., 4000 RPM).**
5. **If TEMPERATURE is Average but closer to Risky, then FAN-SPEED is between Moderate and High.**
6. **If TEMPERATURE is Excellent and decreases, then decrease FAN-SPEED gradually.**

3. Fuzzification of Input and Output

a. Membership Functions for **TEMPERATURE**

- **Risky (High)**: Trapezoidal membership function
- **Average (Moderate)**: Triangular membership function
- **Excellent (Low)**: Trapezoidal membership function

Membership functions (values are assumed for demonstration):

b. Membership Functions for **FAN-SPEED**

- **Slow**: Triangular membership function
- **Moderate**: Triangular membership function
- **High**: Trapezoidal membership function

Membership functions for FAN-SPEED (values are assumed for demonstration):

4. Aggregating the Rules Using Mamdani Model

The Mamdani FIS combines the rules using the **min-max** inference method:

- **Min Operation (AND)**: For each rule, compute the degree of fulfillment by taking the minimum value between the input fuzzy sets.
- **Max Operation (OR)**: After calculating the outputs from all rules, take the maximum of the results for aggregation.

5. Defuzzification Using Centroid Method

The output fuzzy set (FAN-SPEED) is defuzzified to produce a crisp value using the **Centroid Defuzzification Method**, which calculates the center of gravity of the aggregated output fuzzy set.

The crisp output is given by:

$$\text{Crisp Value of FAN-SPEED} = \frac{\int x \cdot \mu(x) dx}{\int \mu(x) dx}$$

Where:

- (x) represents the FAN-SPEED.
- (\mu(x)) is the aggregated membership function for the FAN-SPEED output.

6. Summary of Rules

Rule	IF TEMPERATURE is	THEN FAN-SPEED is
1	Risky	High
2	Average	Moderate
3	Excellent	Slow
4	Risky (Above Threshold)	Max (4000 RPM)
5	Average (Closer to Risky)	Moderate-High
6	Excellent (Decreasing)	Decrease FAN-SPEED

Conclusion

The Mamdani FIS uses these fuzzy rules, fuzzification, and defuzzification to dynamically control the FAN-SPEED based on temperature input, ensuring the furnace operates efficiently.

```
# Need to install fuzzy library
!pip install scikit-fuzzy
```

```
Collecting scikit-fuzzy
  Downloading scikit_fuzzy-0.5.0-py2.py3-none-any.whl.metadata (2.6
kB)
Downloading scikit_fuzzy-0.5.0-py2.py3-none-any.whl (920 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 920.8/920.8 kB 10.8 MB/s eta
0:00:00
```

```
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt

# Define the range for temperature and fan speed
temp_range = np.arange(0, 101, 1)
fan_speed_range = np.arange(0, 501, 1)

# Fuzzy membership functions for Temperature
temp_risky = fuzz.trapmf(temp_range, [70, 80, 100, 100])
temp_average = fuzz.trimf(temp_range, [40, 60, 80])
temp_excellent = fuzz.trapmf(temp_range, [0, 0, 30, 50])

# Fuzzy membership functions for Fan Speed
fan_slow = fuzz.trimf(fan_speed_range, [0, 100, 200])
fan_moderate = fuzz.trimf(fan_speed_range, [150, 250, 350])
fan_high = fuzz.trapmf(fan_speed_range, [300, 400, 500, 500])

# Plot the membership functions
plt.figure(figsize=(10, 8))
```

```
plt.subplot(2, 1, 1)
plt.plot(temp_range, temp_risky, 'r', label='Risky')
plt.plot(temp_range, temp_average, 'g', label='Average')
plt.plot(temp_range, temp_excellent, 'b', label='Excellent')
plt.title('Temperature Membership Functions')
plt.xlabel('Temperature (°C)')
plt.ylabel('Membership Degree')
plt.legend()
plt.subplot(2, 1, 2)
plt.plot(fan_speed_range, fan_slow, 'r', label='Slow')
plt.plot(fan_speed_range, fan_moderate, 'g', label='Moderate')
plt.plot(fan_speed_range, fan_high, 'b', label='High')
plt.title('Fan Speed Membership Functions')
plt.xlabel('Fan Speed (RPM)')
plt.ylabel('Membership Degree')
plt.legend()
plt.tight_layout()
plt.show()
```