Assignment

Fuzzy Logic in Real Life Scenarios

assignment: ให้นศ.ปรับปรุง Python Code สำหรับ Fuzzy Washing Machine Controller

จากตัวอย่างในบทเรียน (หรือนศ.สามารถ implement แบบจำลองระบบควบคุม ขึ้นใหม่เองก็ได้) ให้มีลักษณะดังนี้

1. ความสกปรก(Dirtiness) = 7 , ปริมาณผ้า(Load) = 5
2. เปลี่ยนแปลง Rule สำหรับการ inference ดังนี้

R1 : IF dirtiness IS low AND load IS small THEN wash_time IS short.
R2 : IF dirtiness IS medium AND load IS small THEN wash_time IS medium.
R3 : IF dirtiness IS high AND load IS small THEN wash_time IS long.
R4 : IF dirtiness IS low AND load IS large THEN wash_time IS medium.
R5 : IF dirtiness IS medium AND load IS large THEN wash_time IS long.
R6 : IF dirtiness IS high AND load IS large THEN wash_time IS long.

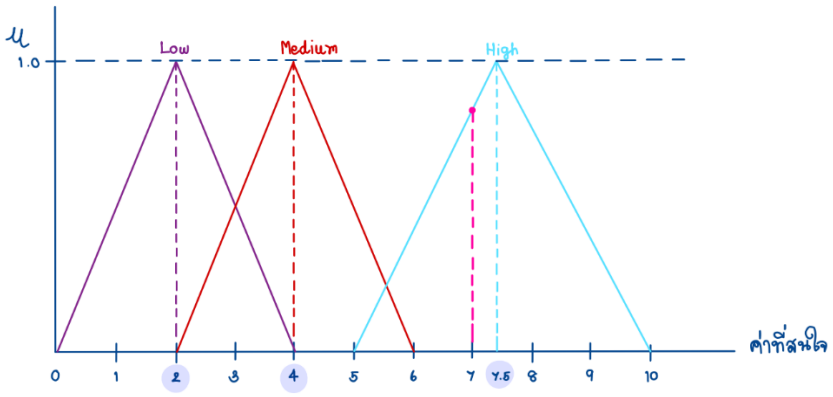
ทดสอบ run โปรแกรมเพื่อคำนวณผลลัพธ์ แล้วให้นศ.ทำการแสดงวิธีคำนวณด้วยมือโดยใช้ Mamdani's Method และ การเปรียบเทียบผลลัพธ์ทั้งสองเพื่อตรวจสอบความถูกต้อง

MF ที่ใช้ (จากเลคเชอร์) :

- Dirtiness : Low (0-4) , Medium (2-6) , High (5-10) Peak : 2 , 4 , 7.5
- Load : Small (0-5 , peak 2.5), Large (4-10 , peak 7)
- Wash Time : Short (0-30 , peak 15) , Medium (20-50 , peak 35) , Long (40-90 , peak 65)

Fuzzifcation : หา μ ของ input

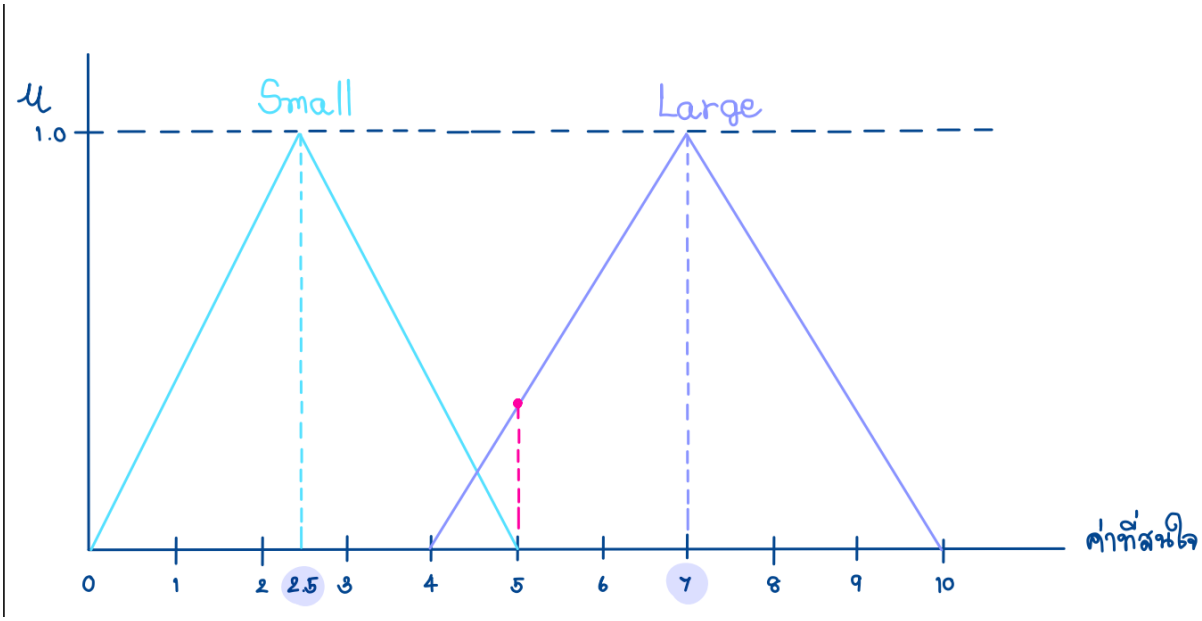
Dirtiness = 7



- Low : μ low = 0 เพราะพ้นช่วง 0-4

- Medium : μ medium = 0 เพราะเกินช่วง 2-6
- Large : μ(7) = X-X1 / X2-X1 = 7-5 / 7.5-5 = 2 / 2.5 = 0.8

Load = 5



- Small : μ(5) = X-X2 / X2-X1 = 5-5 / 5-2.5 = 0
- Large : μ(5) = X-X1 / X2-X1 = 5-4 /7-4 = 1 / 3 = 0.3

สรุปการทำ Fuzzifcation จะได้ดังนี้

- ค่าความเป็นสมาชิกของ Dirtiness : { Low = 0, Meduim = 0, High = 0.8 }
- ค่าความเป็นสมาชิกของ Load : { Small = 0, Large = 0.3 }

Rule Evaluation :

- Dirtiness : { Low = 0, Meduim = 0, High = 0.8 }
- Load : { Small = 0, Large = 0.3 }

R1 : IF dirtiness IS low AND load IS small THEN wash_time IS short.
R2 : IF dirtiness IS medium AND load IS small THEN wash_time IS medium.
R3 : IF dirtiness IS high AND load IS small THEN wash_time IS long.
R4 : IF dirtiness IS low AND load IS large THEN wash_time IS medium.
R5 : IF dirtiness IS medium AND load IS large THEN wash_time IS long.
R6 : IF dirtiness IS high AND load IS large THEN wash_time IS long.

R1 : ถ้า ความสกปรก**ต่ำ** และปริมาณผ้า**น้อย :** เวลาในการซัก **สั้น**

Firing = min (μlow , μsmall) = min(0 , 0) = 0 ไม่ถูกยิง

R2 : ถ้า ความสกปรก**ปานกลาง** และปริมาณผ้า**น้อย :** เวลาในการซัก **ปานกลาง**

Firing = min (μmedium , μsmall) = min(0 , 0) = 0 ไม่ถูกยิง

R3 : ถ้า ความสกปรก**สูง** และปริมาณผ้า**น้อย :** เวลาในการซัก **ยาว**

Firing = min (μhigh , μsmall) = min(0.8 , 0) = 0 ไม่ถูกยิง

R4 : ถ้า ความสกปรก**ต่ำ** และปริมาณผ้า**มาก :** เวลาในการซัก **ปานกลาง**

Firing = min (μlow , μlarge) = min(0 , 0.3) = 0 ไม่ถูกยิง

R5 : ถ้า ความสกปรก**ปานกลาง** และปริมาณผ้า**มาก :** เวลาในการซัก **ยาว**

Firing = min (μmedium , μlarge) = min(0 , 0.3) = 0 ไม่ถูกยิง

R2 : ถ้า ความสกปรก**สูง** และปริมาณผ้า**มาก :** เวลาในการซัก **ยาว**

Firing = min (μhigh , μlarge) = min(0.8 , 0.3) = 0.3 ถูกยิงที่ระดับ 0.3

Combine Output :

เนื่องจากมีกฎ R6 เท่านั้นที่มีการ firing ดังนั้น ผลลัพธ์ที่ได้ คือ ส่วนที่เป็น conclusion ของกฎ R6 จึงสรุปได้มา

ผลลัพธ์ของการอนุมาน คือ Long = 0.3

Defuzzification

- หลังจากที่ผ่านขั้นตอน Fuzzification, Infence และ Aggregation แล้วจะได้ผลลัพธ์ fuzzi output คือ Long μ = 0.3
- แปลงกลับเป็นค่าคมชัด
  ใช้ controid method $$g = \frac{\sum_i x_i \, u(x_i)}{\sum_i u(x_i)}$$

Respresentative peak : shot = 15 , Meduim = 35 , Long = 65

g = 0.3 x 65 / 0.3 = 65

ดังนั้น เครื่องซักผ้านี้จะใช้เวลาในการซัก คือ 65 นาที

จากโค๊ด :

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


# ---------- helper: triangular membership ----------
def trimf(x, a, b, c):
    """Triangular MF over x (array or scalar)."""
    x = np.asarray(x, dtype=float)
    left = np.where(b != a, (x - a) / (b - a), 0.0)
    right = np.where(c != b, (c - x) / (c - b), 0.0)
    y = np.maximum(np.minimum(left, right), 0.0)
    y = np.where(x == b, 1.0, y)  # peak = 1
    return y


# ---------- universes ----------
d_univ = np.linspace(0, 10, 501)    # dirtiness 0..10
l_univ = np.linspace(0, 10, 501)    # load 0..10
t_univ = np.linspace(0, 90, 901)    # wash time 0..90 minutes


# ---------- membership functions ----------
# Dirtiness: Low(0,2,4), Medium(2,4,6), High(5,7.5,10)
d_low = trimf(d_univ, 0.0, 2.0, 4.0)
d_med = trimf(d_univ, 2.0, 4.0, 6.0)
d_high = trimf(d_univ, 5.0, 7.5, 10.0)
```

```
# Load: Small(0,2.5,5), Large(4,7,10)
l_small = trimf(l_univ, 0.0, 2.5, 5.0)
l_large = trimf(l_univ, 4.0, 7.0, 10.0)

# Wash Time: Short(0,15,30), Medium(20,35,50), Long(40,65,90)
t_short = trimf(t_univ, 0.0, 15.0, 30.0)
t_med = trimf(t_univ, 20.0, 35.0, 50.0)
t_long = trimf(t_univ, 40.0, 65.0, 90.0)

# ---------- rule base ----------
rules = [
   ('low',    'small', 'short'),   # R1
   ('medium', 'small', 'medium'),  # R2
   ('high',   'small', 'long'),    # R3
   ('low',    'large', 'medium'),  # R4
   ('medium', 'large', 'long'),    # R5
   ('high',   'large', 'long'),    # R6
]

# mapping
d_mfs = {'low': d_low, 'medium': d_med, 'high': d_high}
l_mfs = {'small': l_small, 'large': l_large}
t_mfs = {'short': t_short, 'medium': t_med, 'long': t_long}

# ---------- fuzzy evaluation ----------
def fuzzy_wash_time(dirtiness_val, load_val, plot=True):
   # fuzzify crisp inputs
   d_vals = {k: float(trimf(dirtiness_val, *params)) for k, params in
           zip(['low','medium','high'], [(0,2,4),(2,4,6),(5,7.5,10)])}
```

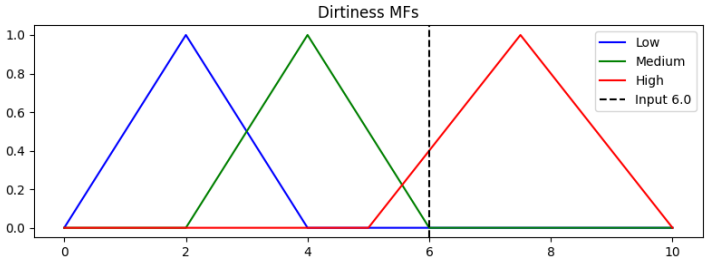
```
l_vals = {k: float(trimf(load_val, *params)) for k, params in
        zip(['small','large'], [(0,2.5,5),(4,7,10)])}
print("Fuzzified input degrees:")
print(" Dirtiness:", d_vals)
print(" Load     :", l_vals)

aggregated = np.zeros_like(t_univ)

# apply rules
for (d_label, l_label, t_label) in rules:
   firing = min(d_vals[d_label], l_vals[l_label])
   print(f"Rule ({d_label} & {l_label} -> {t_label}) firing = {firing:.3f}")
   if firing <= 0:
      continue
   consequent_mf = t_mfs[t_label]
   implied = np.minimum(consequent_mf, firing)
   aggregated = np.maximum(aggregated, implied)

# defuzzify (centroid)
area = np.trapezoid(aggregated, t_univ)
if area == 0:
   crisp = 35.0  # default medium
   print("Warning: no rules fired, using default =", crisp)
else:
   moment = np.trapezoid(t_univ * aggregated, t_univ)
   crisp = moment / area

print(f"Defuzzified wash time (centroid) = {crisp:.2f} minutes")
```

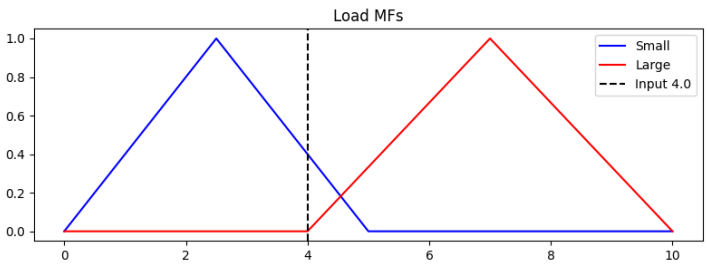
```
# plotting
if plot:
    fig, axs = plt.subplots(3, 1, figsize=(8, 9))

    # Dirtiness
    axs[0].plot(d_univ, d_low, 'b', label='Low')
    axs[0].plot(d_univ, d_med, 'g', label='Medium')
    axs[0].plot(d_univ, d_high, 'r', label='High')
    axs[0].axvline(dirtiness_val, color='k', linestyle='--', label=f'Input {dirtiness_val}')
    axs[0].set_title("Dirtiness MFs")
    axs[0].legend()

    # Load
    axs[1].plot(l_univ, l_small, 'b', label='Small')
    axs[1].plot(l_univ, l_large, 'r', label='Large')
    axs[1].axvline(load_val, color='k', linestyle='--', label=f'Input {load_val}')
    axs[1].set_title("Load MFs")
    axs[1].legend()

    # Wash time
    axs[2].plot(t_univ, t_short, '--', label='Short')
    axs[2].plot(t_univ, t_med, '--', label='Medium')
    axs[2].plot(t_univ, t_long, '--', label='Long')
    axs[2].fill_between(t_univ, aggregated, alpha=0.6, label='Aggregated Output')
    axs[2].axvline(crisp, color='black', linestyle='--', label=f'Crisp = {crisp:.2f} min')
    axs[2].set_title("Wash Time Output")
    axs[2].legend()

    plt.tight_layout()
```

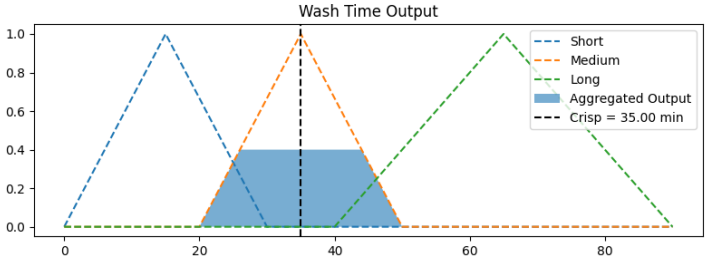
```
    plt.show()

    return crisp, aggregated

# ---------- example run ----------
if __name__ == "__main__":
    dirtiness_in = 7.0   # ตามรูป
    load_in = 5.0        # ตามรูป
    wash_time_crisp, agg_mu = fuzzy_wash_time(dirtiness_in, load_in, plot=True)
```

ผลลัพธ์ที่ได้จากการรันโค๊ด :

```
Fuzzified input degrees:
 Dirtiness: {'low': 0.0, 'medium': 0.0, 'high': 0.8}
 Load     : {'small': 0.0, 'large': 0.3333333333333333}
Rule (low & small -> short) firing = 0.000
Rule (medium & small -> medium) firing = 0.000
Rule (high & small -> long) firing = 0.000
Rule (low & large -> medium) firing = 0.000
Rule (medium & large -> long) firing = 0.000
Rule (high & large -> long) firing = 0.333
Defuzzified wash time (centroid) = 65.00 minutes
```







เปรียบเทียบผลลัพธ์ที่ได้จากการคำนวณเอง และจากการรันโค๊ด

- ผลลัพธ์ที่ได้จากการคำนวณ :
    - Fuzzified input degrees

      Dirtiness = 7.0 ⟶ {low: 0.0, medium: 0.0, high: 0.8}

      Load = 5.0 ⟶ {small: 0.0, large: 0.3}

    - Rule firing (Mamdani, min):

      R1 (low & small → short) = 0.000

      R2 (medium & small → medium) = 0.000

      R3 (high & small → long) = 0.000

      R4 (low & large → medium) = 0.000

      R5 (medium & large → long) = 0.000

      R6 (high & large → long) = 0.300

    - หลัง defuzzify ได้ เวลาซัก = 65 นาที

- ผลลัพธ์ที่ได้จากการรันโค๊ด :
    - Dirtiness = สูง (0.8)
    - Load = ใหญ่เล็กน้อย (0.333)
    - มีกฎเดียวที่ทำงานคือ: (High Dirtiness & Large Load → Long Wash)
    - หลัง defuzzify ได้ เวลาซัก = 65 นาที

สรุปผลลัพธ์ทั้งสองที่ได้จากการคำนวณเอง และการรันโค๊ด มีค่าเท่ากัน คือ 65 นาที