CO542 - Neural Networks and Fuzzy Systems
Lab 09: Fuzzy rule-based decision system

E/19/129
K.H. Gunawardana

Tasks

Task 01: Define the Rule Base

Fuzzy Rule Table

Look \ Personality	Bad	Okay	Good
Bad	Say No	Say No	Think Before Saying No
Okay	Say No	Think Before Saying No	Average
Good	Think Before Saying No	Average	Think Before Saying Yes

Linguistic Rules:
IF Look is Bad AND Personality is Bad THEN Matching is Say No
IF Look is Bad AND Personality is Okay THEN Matching is Say No
IF Look is Bad AND Personality is Good THEN Matching is Think before saying No
IF Look is Okay AND Personality is Bad THEN Matching is Say No
IF Look is Okay AND Personality is Okay THEN Matching is Think before saying No
IF Look is Okay AND Personality is Good THEN Matching is Average
IF Look is Good AND Personality is Bad THEN Matching is Think before saying No
IF Look is Good AND Personality is Okay THEN Matching is Average
IF Look is Good AND Personality is Good THEN Matching is Think before saying Yes

Task 02: Fuzzification and Rule Evaluation

Membership Function Definitions
The membership functions for the inputs Look and Personality, as well as the output Matching Percentage, were defined as piecewise linear functions based on the diagrams provided in the lab sheet.

Look and Personality
Each input has three linguistic terms:

Bad:

- 1 for $x \leq 0$
- $(40 - x)/40$ for $0 < x < 40$
- 0 for $x \geq 40$

Okay:

- 0 for $x \leq 10$ or $x \geq 90$
- $(x - 10)/40$ for $10 < x \leq 50$
- $(90 - x)/40$ for $50 < x < 90$

Good:

- 0 for $x \leq 60$
- $(x - 60)/40$ for $60 < x < 100$
- 1 for $x \geq 100$

Matching Percentage (Output)
The output variable has five linguistic terms:

- Say No (SN)
- Think Before Saying No (TBN)
- Average (AV)
- Think Before Saying Yes (TBY)
- Say Yes (SY)

Fuzzification of Input Values
We evaluate the system for two test cases:

- Boy A: Look = 40, Personality = 80
- Boy B: Look = 75, Personality = 40

Boy A
Look = 40

- Bad = 0
- Okay = (40 - 10)/40 = 0.75
- Good = 0

Personality = 80

- Bad = 0
- Okay = (90 - 80)/40 = 0.25
- Good = (80 - 60)/40 = 0.5

Boy B
Look = 75

- Bad = 0
- Okay = (90 - 75)/40 = 0.375
- Good = (75 - 60)/40 = 0.375

Personality = 40

- Bad = 0
- Okay = (40 - 10)/40 = 0.75
- Good = 0

Fuzzy Rule Evaluation (Mamdani Method)

We use the Min-Max Mamdani method. For each rule, the firing strength is the minimum of the two input memberships. The output fuzzy set is clipped at that value, and the final result is the maximum of all activated outputs.

Boy A
Activated rules:

- Rule 5: IF Look is Okay AND Personality is Okay → Output = Average
- Firing strength = min(0.75, 0.25) = 0.25
- Rule 6: IF Look is Okay AND Personality is Good → Output = Think Before Saying Yes
- Firing strength = min(0.75, 0.5) = 0.5

Final output for A:

- Average → clipped at 0.25
- Think Before Saying Yes → clipped at 0.5

2.3.2 Boy B
Activated rules:

- Rule 5: IF Look is Okay AND Personality is Okay → Output = Average
- Firing strength = min(0.375, 0.75) = 0.375
- Rule 8: IF Look is Good AND Personality is Okay → Output = Think Before Saying Yes
- Firing strength = min(0.375, 0.75) = 0.375

Final output for B:

- Average → clipped at 0.375
- Think Before Saying Yes → clipped at 0.375

Task 03: Defuzzification

Method Used – Centroid Method
To convert the aggregated fuzzy outputs into a crisp value (i.e., a final matching percentage), the centroid method was used. This method calculates the weighted average of all values in the output universe, using their respective membership degrees. It is one of the most commonly used defuzzification techniques due to its balance between accuracy and interpretability.

The centroid formula is given as:
Final Output = (Sum of x * μ(x)) / (Sum of μ(x))
Where:

x = values across the output range (e.g., 30 to 80)
μ(x) = aggregated membership value at x

Defuzzification for Boy A
From Task 2, the aggregated output fuzzy set for Boy A includes:

- "Average" clipped at 0.25
- "Think Before Saying Yes" clipped at 0.5

The final aggregated membership values (μ) at each x value are as follows:

x	Average Clipped	TBY Clipped	Aggregated μ(x)
35	0.25	0	0.25
40	0.25	0	0.25
45	0.25	0	0.25
50	0.25	0	0.25
55	0.25	0.333	0.333
60	0.25	0.5	0.5
65	0.25	0.5	0.5
70	0	0.5	0.5
75	0	0.333	0.333

Numerator = (35×0.25) + (40×0.25) + (45×0.25) + (50×0.25) + (55×0.333) + (60×0.5) + (65×0.5) + (70×0.5) + (75×0.333)
= 8.75 + 10 + 11.25 + 12.5 + 18.315 + 30 + 32.5 + 35 + 24.975 = 183.29
Denominator = Sum of all μ(x) = 0.25 + 0.25 + 0.25 + 0.25 + 0.333 + 0.5 + 0.5 + 0.5 + 0.333 = 3.166
Matching Percentage for Boy A = 183.29 / 3.166 ≈ 57.88%

Defuzzification for Boy B

For Boy B, the aggregated output fuzzy set includes:

- "Average" clipped at 0.375
- "Think Before Saying Yes" clipped at 0.375

x	Average Clipped	TBY Clipped	Aggregated μ(x)
35	0.25	0	0.25
40	0.375	0	0.375
45	0.375	0	0.375
50	0.375	0	0.375
55	0.375	0.333	0.375
60	0.375	0.375	0.375
65	0.25	0.375	0.375
70	0	0.375	0.375
75	0	0.333	0.333

Numerator = (35×0.25) + (40×0.375) + (45×0.375) + (50×0.375) + (55×0.375) + (60×0.375) + (65×0.375) + (70×0.375) + (75×0.333)
= 8.75 + 15 + 16.875 + 18.75 + 20.625 + 22.5 + 24.375 + 26.25 + 24.975 = 178.1
Denominator = 0.25 + 0.375×7 + 0.333 = 3.208
Matching Percentage for Boy B = 178.1 / 3.208 ≈ 55.52%

Candidate	Matching %
Boy A	57.88%
Boy B	55.52%

Based on the defuzzified outputs, Boy A is the better match for C, with a slightly higher matching percentage. The result is consistent with both manual and programmatic evaluations.

Task 04: Programming Implementation

The fuzzy logic-based decision system was implemented in Python using the scikit-fuzzy library. This implementation followed the same specifications and logic described in the previous tasks.

Input Variables

- Look: Linguistic values — Bad, Okay, Good
- Personality: Linguistic values — Bad, Okay, Good

Output Variable
Matching Percentage: Linguistic values — Say No, Think Before Saying No, Average, Think Before Saying Yes, Say Yes

Membership Functions

- All variables were defined using triangular membership functions via fuzz.trimf().
- The shapes and ranges were consistent with those specified in the lab sheet.

Rule Base
A total of 9 fuzzy IF-THEN rules were defined based on Task 01.
Mamdani-style inference with min-max operators was used for rule evaluation.

Fuzzy Inference Workflow

- Inputs were fuzzified based on membership functions
- Rule activations were computed
- Outputs were aggregated and then defuzzified using the centroid method

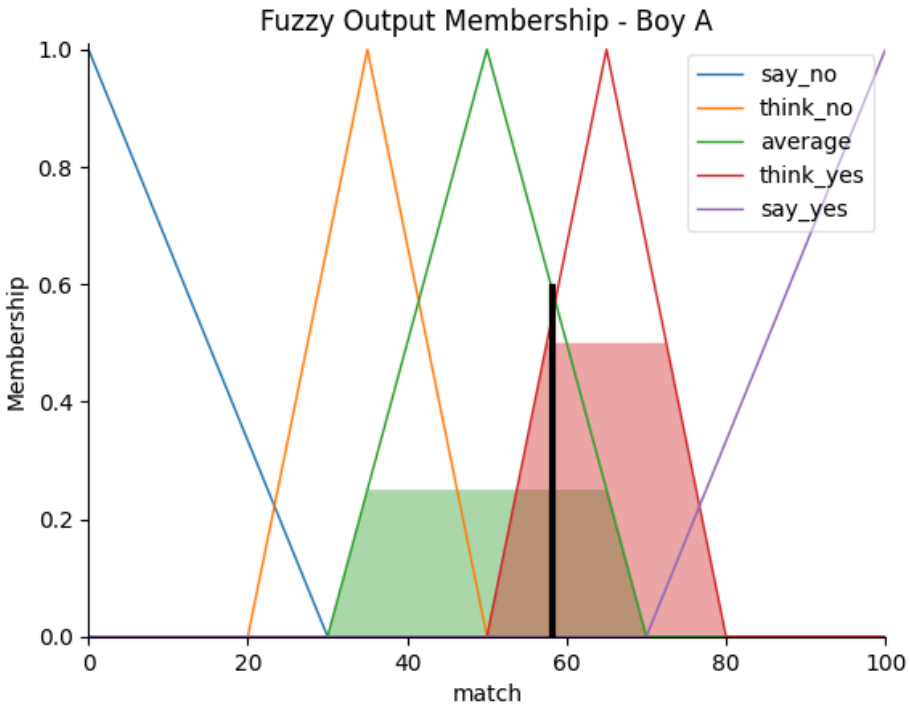
Code:

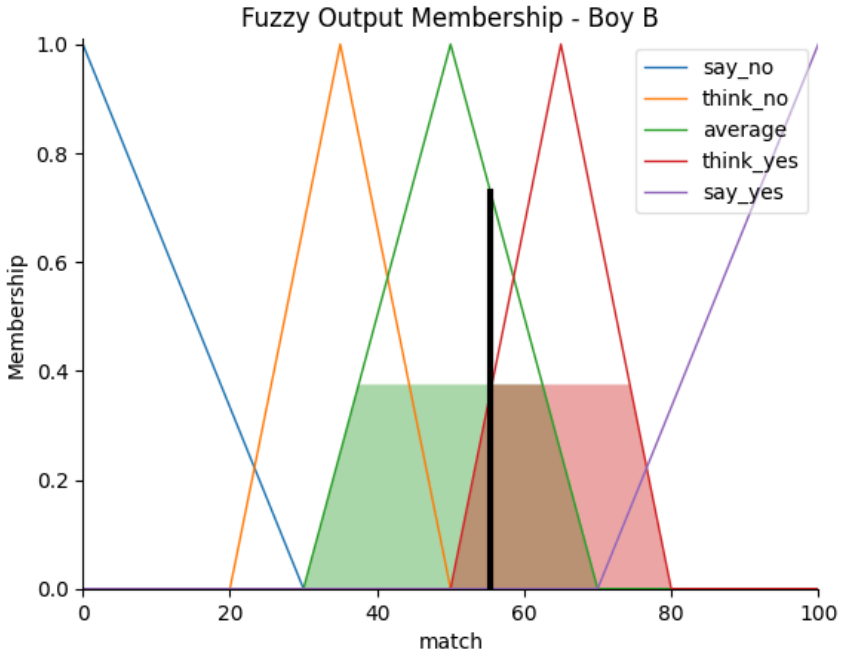
```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# ----------------------------
# 1. Define Input and Output Variables
# ----------------------------
look = ctrl.Antecedent(np.arange(0, 101, 1), 'look')
personality = ctrl.Antecedent(np.arange(0, 101, 1), 'personality')
match = ctrl.Consequent(np.arange(0, 101, 1), 'match')

# ----------------------------
# 2. Define Membership Functions
# ----------------------------
# Look
look['bad'] = fuzz.trimf(look.universe, [0, 0, 40])
look['okay'] = fuzz.trimf(look.universe, [10, 50, 90])
look['good'] = fuzz.trimf(look.universe, [60, 100, 100])

# Personality
personality['bad'] = fuzz.trimf(personality.universe, [0, 0, 40])
personality['okay'] = fuzz.trimf(personality.universe, [10, 50, 90])
personality['good'] = fuzz.trimf(personality.universe, [60, 100, 100])

# Matching Percentage
match['say_no'] = fuzz.trimf(match.universe, [0, 0, 30])
match['think_no'] = fuzz.trimf(match.universe, [20, 35, 50])
match['average'] = fuzz.trimf(match.universe, [30, 50, 70])
match['think_yes'] = fuzz.trimf(match.universe, [50, 65, 80])
match['say_yes'] = fuzz.trimf(match.universe, [70, 100, 100])

# ----------------------------
# 3. Define Fuzzy Rules
# ----------------------------
rules = [
    ctrl.Rule(look['bad'] & personality['bad'], match['say_no']),
    ctrl.Rule(look['bad'] & personality['okay'], match['think_no']),
    ctrl.Rule(look['bad'] & personality['good'], match['average']),
```

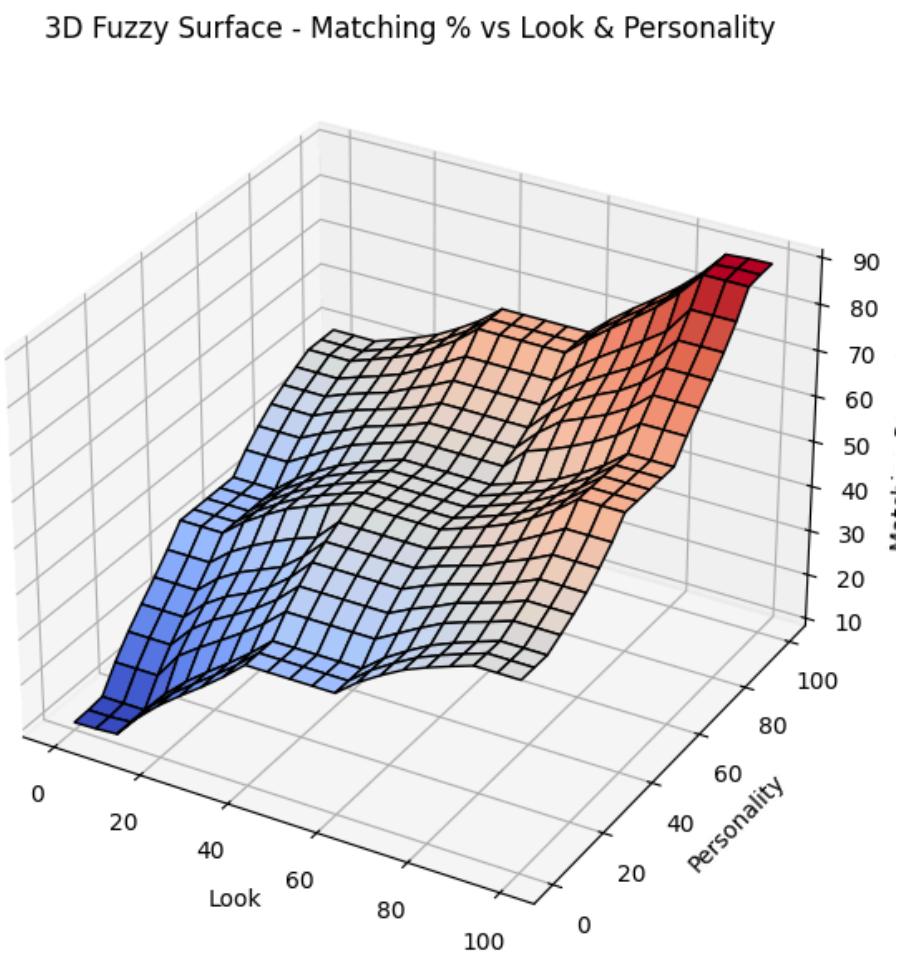
```
    ctrl.Rule(look['okay'] & personality['bad'], match['think_no']),
    ctrl.Rule(look['okay'] & personality['okay'], match['average']),
    ctrl.Rule(look['okay'] & personality['good'], match['think_yes']),
    ctrl.Rule(look['good'] & personality['bad'], match['average']),
    ctrl.Rule(look['good'] & personality['okay'], match['think_yes']),
    ctrl.Rule(look['good'] & personality['good'], match['say_yes'])
]

# ----------------------------
# 4. Build Control System
# ----------------------------
system = ctrl.ControlSystem(rules)
sim = ctrl.ControlSystemSimulation(system)

# ----------------------------
# 5. Test Case: Boy A (Look = 40, Personality = 80)
# ----------------------------
sim.input['look'] = 40
sim.input['personality'] = 80
sim.compute()
print("Boy A Matching Percentage:", round(sim.output['match'], 2))

match.view(sim=sim)
plt.title("Fuzzy Output Membership - Boy A")
plt.show()

# ----------------------------
# 6. Test Case: Boy B (Look = 75, Personality = 40)
# ----------------------------
sim.input['look'] = 75
sim.input['personality'] = 40
sim.compute()
print("Boy B Matching Percentage:", round(sim.output['match'], 2))

match.view(sim=sim)
plt.title("Fuzzy Output Membership - Boy B")
plt.show()

# ----------------------------
# 7. Optional: 3D Surface Plot of Output
```

```
# ------------------------------
look_range = np.arange(0, 101, 5)
personality_range = np.arange(0, 101, 5)
output_surface = np.zeros((len(personality_range), len(look_range)))

# Rebuild simulation for plotting
sim_surface = ctrl.ControlSystemSimulation(system)

for i, p in enumerate(personality_range):
    for j, l in enumerate(look_range):
        sim_surface.input['look'] = l
        sim_surface.input['personality'] = p
        sim_surface.compute()
        output_surface[i, j] = sim_surface.output['match']

# Plotting the 3D surface
from mpl_toolkits.mplot3d import Axes3D

X, Y = np.meshgrid(look_range, personality_range)
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, output_surface, cmap='coolwarm', edgecolor='k')
ax.set_xlabel('Look')
ax.set_ylabel('Personality')
ax.set_zlabel('Matching Percentage')
plt.title("3D Fuzzy Surface - Matching % vs Look & Personality")
plt.tight_layout()
plt.show()
```

Output:
Boy A Matching Percentage: 58.13
Boy B Matching Percentage: 55.45


Fuzzy Output Membership - Boy A
Membership
match
say_no
think_no
average
think_yes
say_yes
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100


Fuzzy Output Membership - Boy B
Membership
match
say_no
think_no
average
think_yes
say_yes
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100



Results and Validation
The Python system produced the following results:

Candidate	Matching % (Program)	Interpretation
Boy A	58.13%	Average / Think Yes
Boy B	55.45%	Average / Think Yes

These results were compared with the manually calculated values from Task 03:

Candidate	Manual Calculation	Python Result	Difference
Boy A	57.88%	58.13%	0.25%
Boy B	55.52%	55.45%	0.07%

The small differences are due to:

- Discrete sampling used in the program (step size = 1)
- Slight variations in interpolation and centroid calculation

The implementation successfully replicated the expected fuzzy logic behavior. The results closely match manual calculations and demonstrate the system's ability to make meaningful decisions based on imprecise inputs. The model proves effective for scenarios that require soft decision boundaries and human-like reasoning.