Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Ульяновский государственный технический университет»
Кафедра «Вычислительная техника»

Системы искусственного интеллекта

Лабораторная работа №2

«Нечеткая логика»

Вариант №4

Выполнил:
студент группы ИВТАСбд-41
Рубцов Денис Алексеевич
Проверил:
Хайруллин И. Д.

Ульяновск
2025

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ 2
Постановка задачи 3
Ход работы 4
Тестирование 6
Заключение 8
Приложение А. 9

Постановка задачи

Общее задание:

Необходимо разработать программу на языке python, которая реализует предложенное вариантом задание. Предметную область можно выбрать из предложенного списка, либо выбрать свою.

Вариант 4: на языке Python разработайте скрипт, позволяющий выполнить операцию пересечения заданных пользователем нечетких множеств с трапециевидными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – пересечение данных нечетких множеств.

Предметная область – авиация.

Высота полета: низкая, средняя, высокая

Скорость полета: медленная, нормальная, быстрая

Ход работы

Первым делом в начале программы задаются универсумы (области определения) для входных переменных и нечёткие переменные (Рис.1)

```
self.height_universe = np.arange(0, 15001, 100)
self.speed_universe = np.arange(0, 1201, 10)

#нечеткие переменные
self.height = ctrl.Antecedent(self.height_universe, 'height')
self.speed = ctrl.Antecedent(self.speed_universe, 'speed')
```

Рис. 1.

Для каждой переменной определены три нечётких множества с использованием трапециевидных функций принадлежности (Рис.2)

```
#функции принадлежности
self.height['low'] = fuzz.trapmf(self.height.universe, [0, 0, 2000, 4000])
self.height['medium'] = fuzz.trapmf(self.height.universe, [2000, 4000, 6000, 8000])
self.height['high'] = fuzz.trapmf(self.height.universe, [6000, 8000, 15000, 15000])

self.speed['slow'] = fuzz.trapmf(self.speed.universe, [0, 0, 300, 500])
self.speed['normal'] = fuzz.trapmf(self.speed.universe, [300, 500, 700, 900])
self.speed['fast'] = fuzz.trapmf(self.speed.universe, [700, 900, 1200, 1200])
```

Рис. 2.

Это одна из самых часто используемых функций принадлежности в нечёткой логике. Она описывает плавный переход элемента из состояния «не принадлежит множеству» к состоянию «полностью принадлежит множеству» и обратно. Задается в виде следующей формулы (Рис.3)

$$\mu(x) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ 1, & b \le x \le c \\ \dfrac{d-x}{d-c}, & c \le x \le d \\ 0, & d \le x \end{cases}$$

Рис. 3.

Для введённых пользователем значений высоты и скорости выполняется вычисление степени принадлежности каждого значения всем нечётким

множествам. Результат выражается числом от 0 до 1 — степени принадлежности (Рис.4)

```
height_low = fuzz.interp_membership(self.height.universe, self.height['low'].mf, height_value)
height_medium = fuzz.interp_membership(self.height.universe, self.height['medium'].mf, height_value)
height_high = fuzz.interp_membership(self.height.universe, self.height['high'].mf, height_value)

speed_slow = fuzz.interp_membership(self.speed.universe, self.speed['slow'].mf, speed_value)
speed_normal = fuzz.interp_membership(self.speed.universe, self.speed['normal'].mf, speed_value)
speed_fast = fuzz.interp_membership(self.speed.universe, self.speed['fast'].mf, speed_value)
```

Рис. 4.

Далее для всех комбинаций термов (например, «низкая высота и высокая скорость») вычисляются степени пересечения с использованием операции минимума (Рис.5)

```
intersection_values = {
    'low_slow': min(height_low, speed_slow),
    'low_normal': min(height_low, speed_normal),
    'low_fast': min(height_low, speed_fast),
    'medium_slow': min(height_medium, speed_slow),
    'medium_normal': min(height_medium, speed_normal),
    'medium_fast': min(height_medium, speed_fast),
    'high_slow': min(height_high, speed_slow),
    'high_normal': min(height_high, speed_normal),
    'high_fast': min(height_high, speed_fast)
}
```

Рис. 5.

После вычисления всех пересечений программа выбирает комбинации, имеющие наибольшую степень пересечения.

Для лучшей комбинации строится график с наглядным положением четкого объекта на трапециевидных функциях принадлежности.

Тестирование

```
Введите высоту полета (м, 0-15000): 6700
Введите скорость полета (км/ч, 0-1200): 340
low height & slow speed: 0.000
low height & normal speed: 0.000
low height & fast speed: 0.000
medium height & slow speed: 0.650
medium height & normal speed: 0.200
medium height & fast speed: 0.000
high height & slow speed: 0.350
high height & normal speed: 0.200
high height & fast speed: 0.000
Лучшие комбинации:
- medium height & slow speed
```

Рис. 6. Высота полета 6700, скорость 340.

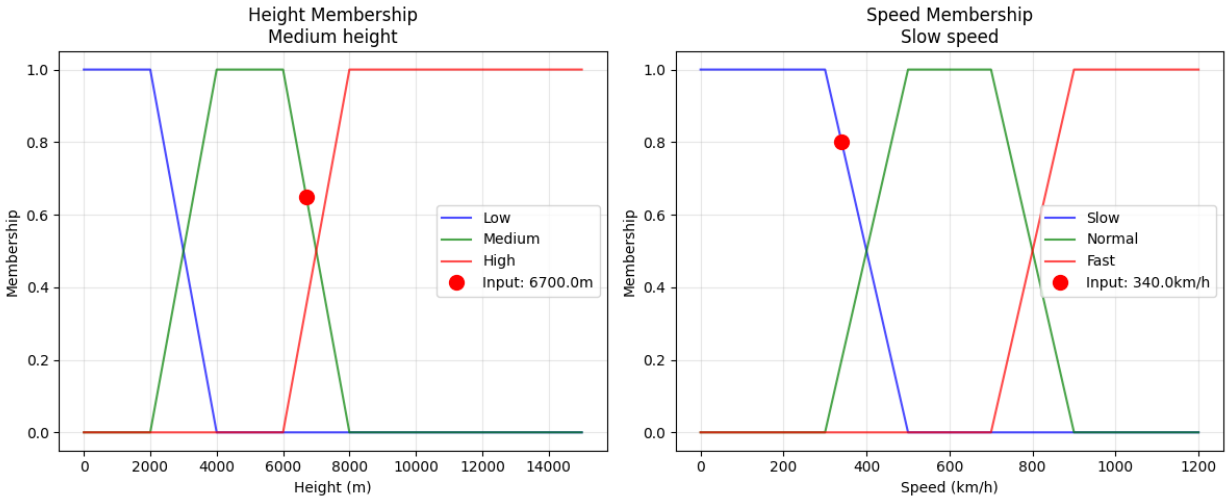

Рис. 7. График: высота полета 6700, скорость 340.

```
Введите высоту полета (м, 0-15000): 15000
Введите скорость полета (км/ч, 0-1200): 1200
low height & slow speed: 0.000
low height & normal speed: 0.000
low height & fast speed: 0.000
medium height & slow speed: 0.000
medium height & normal speed: 0.000
medium height & fast speed: 0.000
high height & slow speed: 0.000
high height & normal speed: 0.000
high height & fast speed: 1.000
Лучшие комбинации:
- high height & fast speed
```

Рис. 8. Высота полета 15000, скорость 1200.

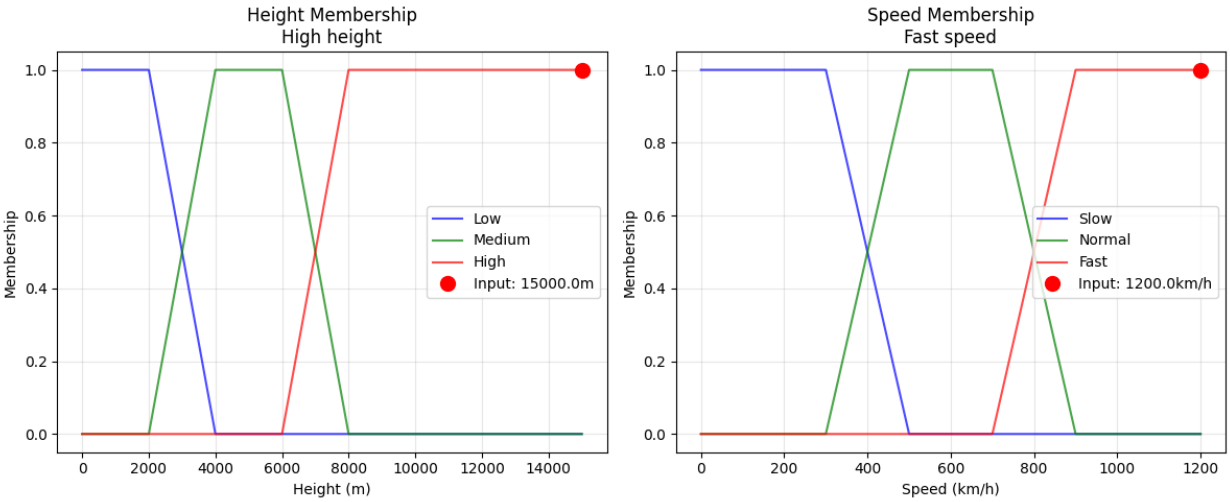

Рис. 9. График: высота полета 15000, скорость 1200.

Заключение

В рамках выполнения лабораторной работы были приобретены умения и навыки в реализации нечеткой логики на языке программирования python.

Также была написана программа, реализующая все пункты задания

Приложение А.

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



class AviationSystem:
    def __init__(self):
        self.height_universe = np.arange(0, 15001, 100)
        self.speed_universe = np.arange(0, 1201, 10)

        #нечеткие переменные
        self.height = ctrl.Antecedent(self.height_universe, 'height')
        self.speed = ctrl.Antecedent(self.speed_universe, 'speed')

        #функции принадлежности
        self.height['low'] = fuzz.trapmf(self.height.universe, [0, 0, 2000, 4000])
        self.height['medium'] = fuzz.trapmf(self.height.universe, [2000, 4000, 6000, 8000])
        self.height['high'] = fuzz.trapmf(self.height.universe, [6000, 8000, 15000, 15000])

        self.speed['slow'] = fuzz.trapmf(self.speed.universe, [0, 0, 300, 500])
        self.speed['normal'] = fuzz.trapmf(self.speed.universe, [300, 500, 700, 900])
        self.speed['fast'] = fuzz.trapmf(self.speed.universe, [700, 900, 1200, 1200])

    def calculate_intersection(self, height_value, speed_value):
        #вычисление степеней принадлежности
        height_low = fuzz.interp_membership(self.height.universe, self.height['low'].mf, height_value)
        height_medium = fuzz.interp_membership(self.height.universe, self.height['medium'].mf, height_value)
        height_high = fuzz.interp_membership(self.height.universe, self.height['high'].mf, height_value)

        speed_slow = fuzz.interp_membership(self.speed.universe, self.speed['slow'].mf, speed_value)
        speed_normal = fuzz.interp_membership(self.speed.universe, self.speed['normal'].mf, speed_value)
        speed_fast = fuzz.interp_membership(self.speed.universe, self.speed['fast'].mf, speed_value)

        #операция пересечения (минимум) для всех комбинаций
        intersection_values = {
            'low_slow': min(height_low, speed_slow),
            'low_normal': min(height_low, speed_normal),
            'low_fast': min(height_low, speed_fast),
            'medium_slow': min(height_medium, speed_slow),
```

```
        'medium_normal': min(height_medium, speed_normal),
        'medium_fast': min(height_medium, speed_fast),
        'high_slow': min(height_high, speed_slow),
        'high_normal': min(height_high, speed_normal),
        'high_fast': min(height_high, speed_fast)
    }

    return intersection_values

def get_intersection_result(self, height_value, speed_value):
    results = self.calculate_intersection(height_value, speed_value)
    max_intersection = max(results.values())
    best_combinations = [comb for comb, value in results.items() if value == max_intersection]
    return best_combinations, results

def plot(self, height_value, speed_value, best_combination):
    height_term, speed_term = best_combination.split('_')

    #график с двумя подграфиками
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

    #Высота
    ax1.plot(self.height_universe, self.height['low'].mf, 'b', linewidth=1.5, label='Low', alpha=0.7)
    ax1.plot(self.height_universe, self.height['medium'].mf, 'g', linewidth=1.5, label='Medium', alpha=0.7)
    ax1.plot(self.height_universe, self.height['high'].mf, 'r', linewidth=1.5, label='High', alpha=0.7)

    #Точка для высоты
    height_membership = fuzz.interp_membership(self.height.universe, self.height[height_term].mf, height_value)
    ax1.plot(height_value, height_membership, 'ro', markersize=10, label=f'Input: {height_value}m')

    ax1.set_title(f'Height Membership\n{height_term.capitalize()} height')
    ax1.set_xlabel('Height (m)')
    ax1.set_ylabel('Membership')
    ax1.legend()
    ax1.grid(True, alpha=0.3)

    #Скорость
    ax2.plot(self.speed_universe, self.speed['slow'].mf, 'b', linewidth=1.5, label='Slow', alpha=0.7)
    ax2.plot(self.speed_universe, self.speed['normal'].mf, 'g', linewidth=1.5, label='Normal', alpha=0.7)
    ax2.plot(self.speed_universe, self.speed['fast'].mf, 'r', linewidth=1.5, label='Fast', alpha=0.7)

    # Точка для скорости
```

```
        speed_membership = fuzz.interp_membership(self.speed.universe, self.speed[speed_term].mf, speed_value)
        ax2.plot(speed_value, speed_membership, 'ro', markersize=10, label=f'Input: {speed_value}km/h')

        ax2.set_title(f'Speed Membership\n{speed_term.capitalize()} speed')
        ax2.set_xlabel('Speed (km/h)')
        ax2.set_ylabel('Membership')
        ax2.legend()
        ax2.grid(True, alpha=0.3)

        plt.tight_layout()
        plt.show()


def main():
    aviation_system = AviationSystem()
    try:
        height_val = float(input("Введите высоту полета (м, 0-15000): "))
        speed_val = float(input("Введите скорость полета (км/ч, 0-1200): "))

        best_combinations, all_results = aviation_system.get_intersection_result(height_val,speed_val)

        for combination, value in all_results.items():
            height_term, speed_term = combination.split('_')
            print(f"{height_term} height & {speed_term} speed: {value:.3f}")

        print("Лучшие комбинации:")
        for comb in best_combinations:
            height_term, speed_term = comb.split('_')
            print(f"- {height_term} height & {speed_term} speed")

        if best_combinations:
            aviation_system.plot(height_val, speed_val, best_combinations[0])

    except ValueError:
        print("Ошибка: введите корректные числовые значения.")
    except Exception as e:
        print(f"Произошла ошибка: {e}")


if __name__ == "__main__":
    main()
```