دانشکده مهندسی کامپیوتر

گزارش پایانی درس: مبانی هوش محاسباتی

عنوان پژوهش: تمرین دوم بخش دوم (طراحی سیستم منطق فازی برای کنترل آبیاری)

ارائه دهنده:

نرگس سادات موسوی جد

شادی شاهی محمدی

استاد درس:

دکتر کارشناس

بهار ۱۴۰۴

فهرست:

۱–سیستم فازی برای کنترل آبیاری

این سیستک به‌طور کلی یک سیستم فازی برای کنترل آبیاری طراحی کرده است که از سیستم فازی و قوانین فازی برای پیش‌بینی میزان آبیاری بر اساس رطوبت خاک و شرایط آب و هوا استفاده می‌کند.

۱–۱– توابع عضویت :

توابع عضویت برای سه ویژگی اصلی تعریف شده‌اند:

تابع triangular(x, a, b, c) برای محاسبه مقدار عضویت در مجموعه‌های مثلثی مختلف استفاده می‌شود. این توابع عضویت به‌طور کلی برای مدلسازی ویژگی‌های فازی در سیستم‌های فازی استفاده می‌شوند.

```
#مثلثی عضویت تابع

def triangular(x, a, b, c):
    if x <= a or x >= c:
        return 0
    elif a < x < b:
        return (x - a) / (b - a)
    elif b <= x < c:
        return (c - x) / (c - b)
    elif x == b:
        return 1
```

رطوبت خاک (soil) با سه مجموعه فازی: خشک (soil_dry)، متوسط (soil_medium)، و مرطوب (soil_wet) تعریف می‌شود.

```
#خاک رطوبت
def soil_dry(x):
    return triangular(x, 0, 0, 50)

def soil_medium(x):
    return triangular(x, 30, 50, 70)

def soil_wet(x):
    return triangular(x,60, 100, 100)
```

شرایط آب و هوا (weather) با سه مجموعه فازی: آفتابی (weather_sunny)، ابری (weather_cloudy)، و بارانی (weather_rainy) تعریف می‌شود.

```
#هوا و اب
def weather_sunny(x):
    return triangular(x, 0, 0, 50)

def weather_cloudy(x):
    return triangular(x, 30, 50, 70)
```

```
def weather_rainy(x):
    return triangular(x, 60, 100, 100)
```

مقدار آبیاری (water) با چهار مجموعه فازی: بدون آبیاری (water_none)، کم (water_low)، متوسط (water_medium)، و زیاد (water_high) تعریف می‌شود

```
#ابیاری
def water_none(x):
    return triangular(x, 0, 0, 25)

def water_low(x):
    return triangular(x, 10, 30, 50)

def water_medium(x):
    return triangular(x, 40, 60, 80)

def water_high(x):
    return triangular(x, 70, 100, 100)
    return min(a, b)
```

در تابع rule_base(soil_val, weather_val), مجموعه‌ای از قوانین فازی تعریف می‌شود که ارتباط بین رطوبت خاک و شرایط آب و هوا را بررسی می‌کند. بر اساس این قوانین، برای ترکیب مقادیر عضویت از عملیات فازی AND استفاده می‌شود.

قوانین بکار رفته در این سامانه عبارتند از:

اگر خاک خشک است و هوا آفتابی است، آبیاری زی اد باشد.

- اگر خاک خشک و هوا ابر ی است، آب دهی متوسط باشد.
- اگر خاک خشک و هوا بارانی است، آب دهی کم.
- اگر خاک متوسط و هوا آفتابی است، مقدار آب متوسط.
- اگر خاک متوسط و هوا ابری است، مقدار آب کم.
- اگر خاک متوسط و هوا بارانی است، نیازی به آب نیست.
- اگر خاک مرطوب و هوا آفتابی است، مقدار آب کم.
- اگر خاک مرطوب و هوا ابری است، نیازی به آب نیست.
- اگر خاک مرطوب و هوا بارانی است، نیازی به آب نیست.

همچنین قوانین جدیدی نیز برای بهبود عملکرد به سیستم اضافه شده‌اند که عبارتند از:

اگر خاک بین خشک و متوسط باشد و هوا آفتابی باشد، آبیاری متوسط است.

اگر خاک بین متوسط و مرطوب و هوا آفتابی باشد، آبیاری کم.

اگر خاک متوسط و هوا بین ابری و بارانی باشد، آبیاری خیلی کم.

اگر خاک خیلی مرطوب است، آبیاری کاملا قطع باشد.

```
def rule_base(soil_val, weather_val):
    rules = []
    dry = soil_dry(soil_val)
    medium = soil_medium(soil_val)
    wet = soil_wet(soil_val)

    sunny = weather_sunny(weather_val)
    cloudy = weather_cloudy(weather_val)
    rainy = weather_rainy(weather_val)

    rules.append((fuzzy_and(dry, sunny), "High"))
    rules.append((fuzzy_and(dry, cloudy), "Medium"))
    rules.append((fuzzy_and(dry, rainy), "Low"))
    rules.append((fuzzy_and(medium, sunny), "Medium"))
    rules.append((fuzzy_and(medium, cloudy), "Low"))
    rules.append((fuzzy_and(medium, rainy), "None"))
    rules.append((fuzzy_and(wet, sunny), "None"))
    rules.append((fuzzy_and(wet, cloudy), "None"))
    rules.append((fuzzy_and(wet, rainy), "None"))

    transitional_dry_medium = min(dry, medium)
    rules.append((fuzzy_and(transitional_dry_medium, sunny), "High"))

    transitional_medium_wet = min(medium, wet)
    rules.append((fuzzy_and(transitional_medium_wet, sunny), "Low"))

    if soil_val >= 80:
        rules.append((1.0, "None"))

    return rules
```

Centroid: برای محاسبه مقدار دقیق خروجی از روش مرکزی (Centroid Method) استفاده می‌شود که میانگین وزنی مقدار خروجی را محاسبه می‌کند.

```
def centroid(aggregated_func, x_vals):
    numerator = sum(x * aggregated_func(x) for x in x_vals)
    denominator = sum(aggregated_func(x) for x in x_vals)
    return numerator / denominator if denominator != 0 else 0
```

irrigation_output() : این تابع بر اساس رطوبت خاک و شرایط آب و هوا میزان آبیاری را پیش‌بینی می‌کند. این تابع از قوانین فازی و عملیات Centroid استفاده می‌کند تا مقدار دقیق آبیاری (مقدار آب لازم) را محاسبه کند.

```
def irrigation_output(soil_val, weather_val):
    rules = rule_base(soil_val, weather_val)

    output_mfs = {
        "None": water_none,
        "Low": water_low,
        "Medium": water_medium,
        "High": water_high
    }

    clipped_funcs = []
    for degree, label in rules:
        if degree > 0:
            clipped_funcs.append(clipped_membership(output_mfs[label],
degree))

    def aggregated_output(x):
        return max([f(x) for f in clipped_funcs], default=0)

    x_vals = np.linspace(0, 100, 500)
    result = centroid(aggregated_output, x_vals)
    return result
```

تابع simulate_feedback_irrigation نیز برای شبیه‌سازی آبیاری در طی چند روز طراحی شده است. این تابع شرایط آب و هوای روزانه را می‌گیرد و مقدار آب مورد نیاز برای آبیاری را محاسبه می‌کند. سپس مقدار رطوبت خاک را بر اساس مقدار آبیاری و تأثیر شرایط آب و هوا به‌روزرسانی می‌کند. این روند طی ده روز ادامه می‌یابد.

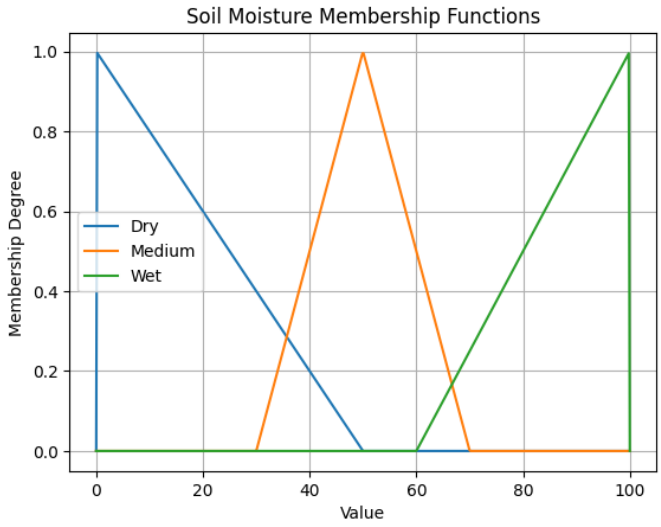
```
def simulate_feedback_irrigation(days=10, initial_soil=15, desired_range=(40,
60)):
    weather_list = ["Sunny", "Sunny", "Cloudy", "Rainy", "Sunny", "Cloudy",
"Rainy", "Sunny", "Cloudy", "Rainy"]
    weather_effect = {"Sunny": 5, "Cloudy": 2, "Rainy": -5}
    desired_mid = (desired_range[0] + desired_range[1]) / 2

    soil_history = [initial_soil]
    irrigation_history = []
    current_soil = initial_soil

    for day in range(days):
        weather = weather_list[day]
        weather_loss = weather_effect[weather]

        error = desired_mid - current_soil
        water_amount = irrigation_fuzzy_error(error)
        irrigation_history.append(water_amount)
```

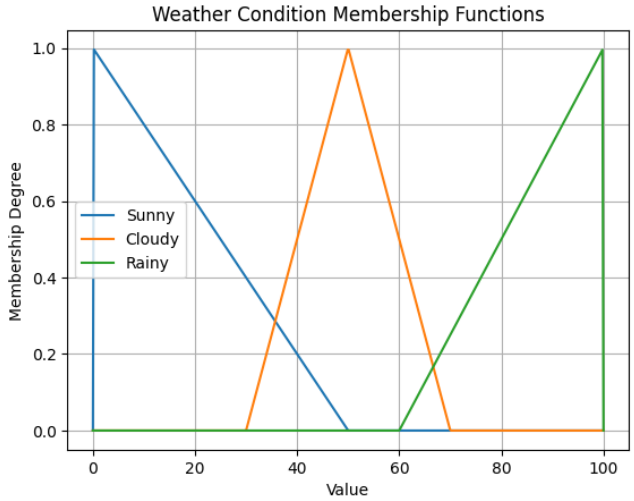
```
        new_soil = current_soil + (0.097 * water_amount) - weather_loss
        new_soil = max(0, min(100, new_soil))
        soil_history.append(new_soil)
        current_soil = new_soil

    days_list = list(range(days + 1))
    irrigation_days = list(range(days))
```

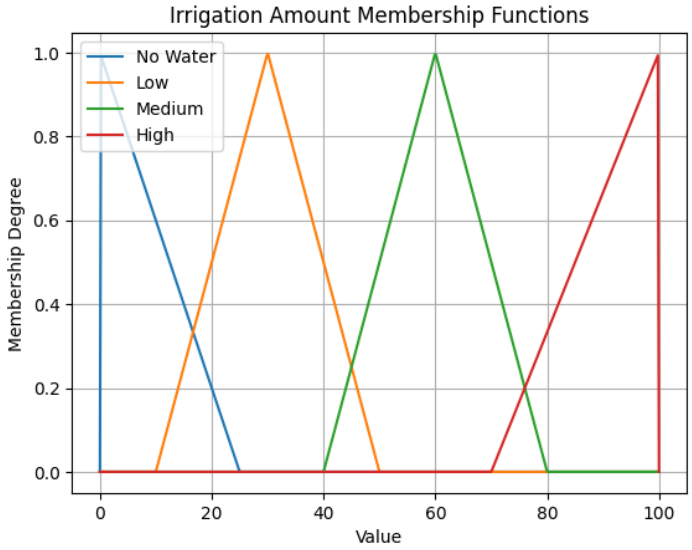
۲–نتایج:

۲–۱–توابع عضویت فازی:

توابع عضویت رطوبت خاک، آب‌وهوا و مقدار آبیاری به صورت زیر تعریف شده‌اند:



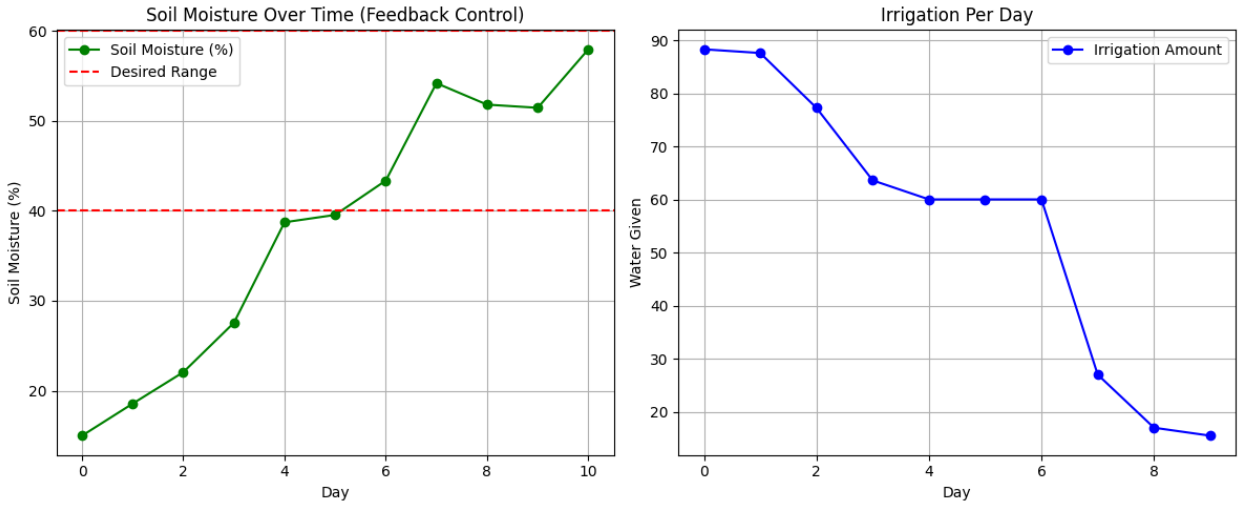




۲–۲–نتایج آبیاری طی ۱۰ روز:

طبق ورودی‌های داده شده نتایج آبیاری طی ده روز قابل مشاهده است:

```
weather_list = ["Sunny", "Sunny", "Cloudy", "Rainy", "Sunny", "Cloudy",
"Rainy", "Sunny", "Cloudy", "Rainy"]
```



هدف نگه‌داشتن سطح رطوبت خاک بین ۴۰ تا ۶۰ درصد بوده که همانطور که مشاهده می‌شود در پایان ۱۰ روز این اتفاق افتاده است همچنین با بالا رفتن میزان رطوبت خاک از میزان آبیاری کاسته شده است.

لینک کد:

https://colab.research.google.com/gist/tiyamti/ef8a5b2fd7a9c7b992590ea4782b504e/fuzzy1.ipynb