261456 Introduction to Computer Intelligence

Computer Assignment 2

นาย ดรันภพ เป็งคำตา 580610642

Fuzzy Logic

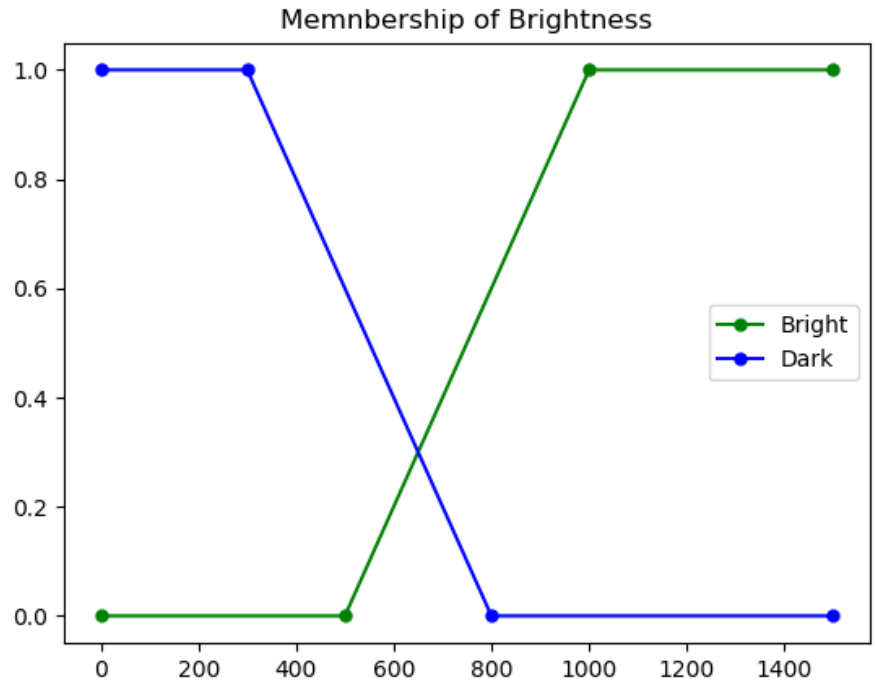
Cloth line controller

เครื่องควบคุมราวตากผ้า

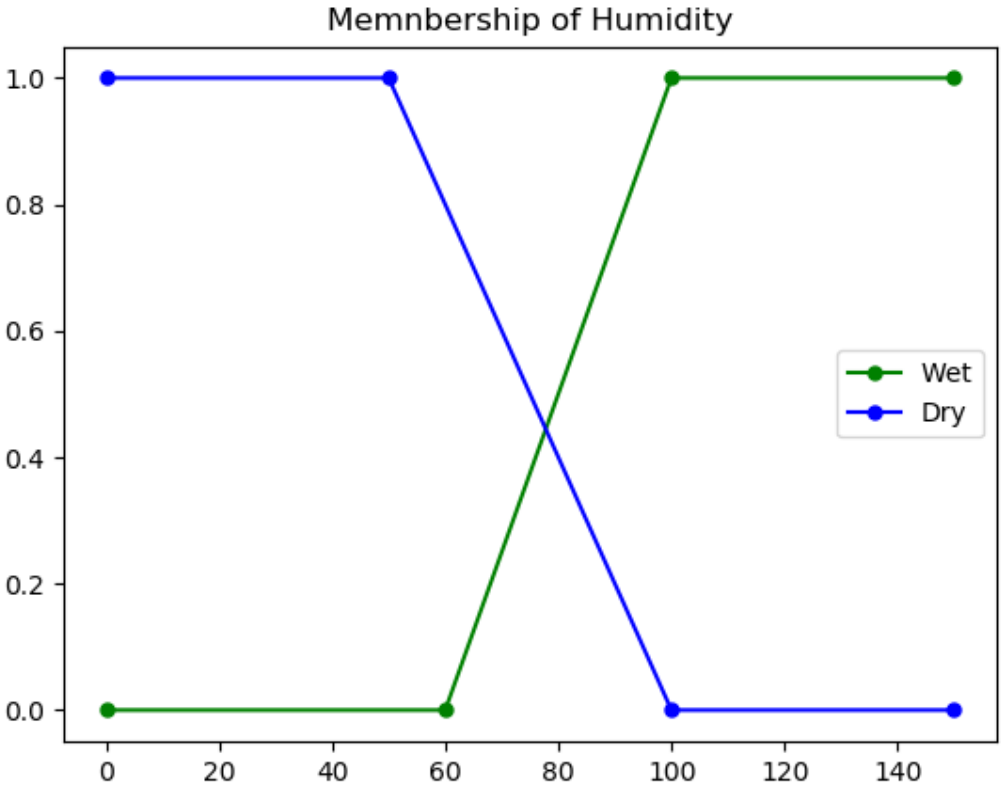
รายงานเป็นรายงานเกี่ยวกับการทดลองเรื่อง **Fuzzy Logic** ด้วยการทำการ implement โปรแกรมตามประบวนการทำงานของ Fuzzy logic เพื่อที่จะทำสร้างระบบควบคุมด้วย fuzzy ของเครื่องควบคุมราวตากผ้า โดยเครื่องควบคุมราวตากผ้านี้จะวัดค่า ความชื้นสัมพัทธ์ในอากาศ และความเข้มแสง แล้วนำค่าที่ได้ผ่านเข้าสู่ระบบฟัซซี่ประประเมินว่าฝนตกหรือไม่ หากจะตกจะทำการเก็บผ้าโดยอัตโนมัติ

รูปแบบของฟัซซีที่ใช้

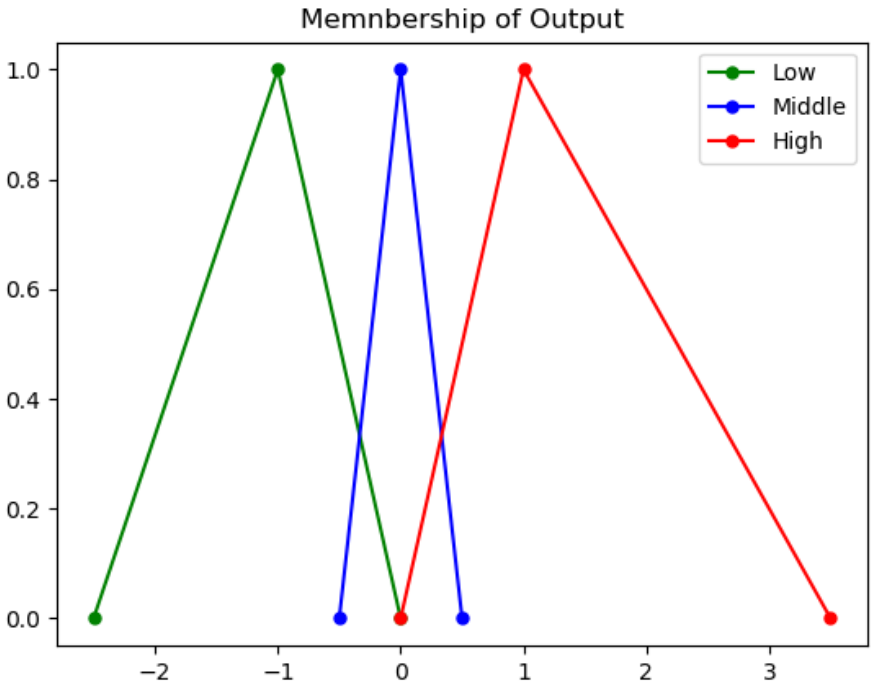
ฟัซซี่ของ ความสว่าง



ฟัซซี่ของ ความชื้น



ฟัซซี่ของฟังชั่นผลลัพธ์



ตารางแสดง กฎที่ใช้

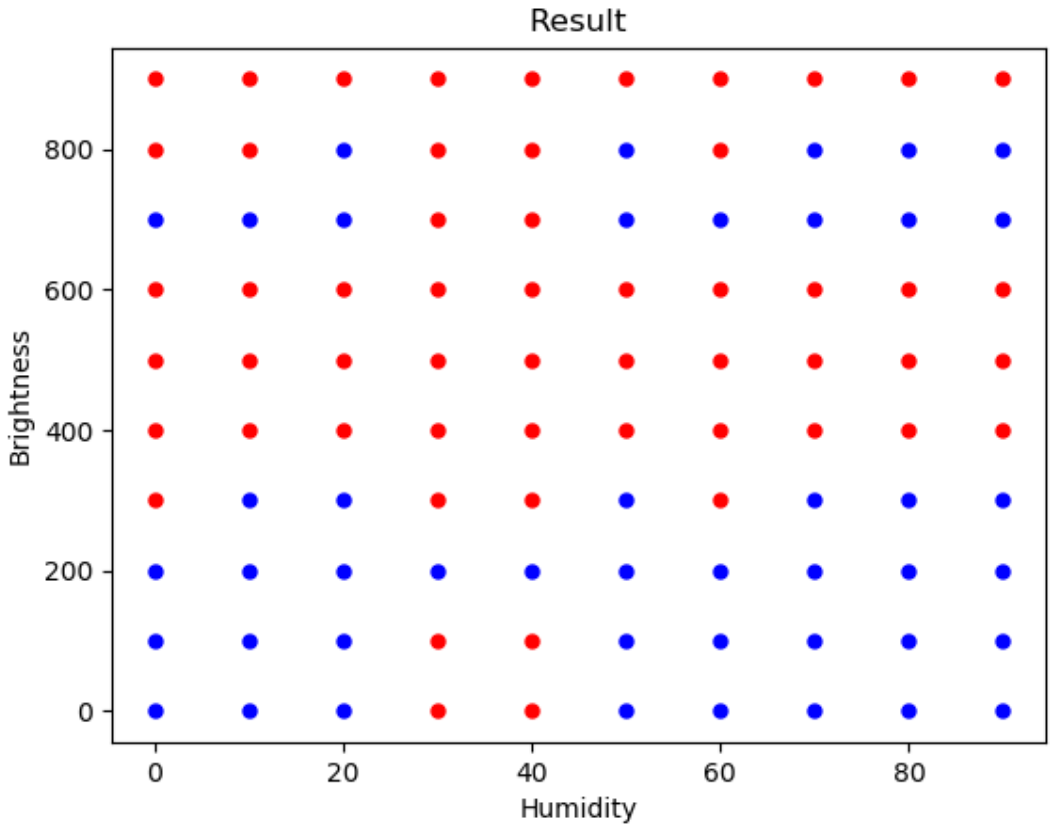
RULE	WET	DRY
BRIGHT	MIDDLE	LOW
DARK	HIGH	MIDDLE

การทำงานของระบบนี้จะใช้ค่าความฟัซซี่ของความสว่างคือ สว่าง (Bright) และ มืด (Dark) ประกอบความชื้อ แห้ง (Wet) และ ชื้น (Wet) เพื่อทำนายว่าฝนจะตกหรือไม่ และฟังก์ชั่นฟัซซี่ output เอาไว้สำหรับกำหนดค่าความสำคัญจากกฎ

จะเห็นได้ว่าที่ มืด และ ชื้น จะได้ฟัซซี่ High ซึงมีคาในช่วง มากกว่า 0 หมายความว่าตามลักษณะนี้ มีโอกาศที่ฝนจะตก และจากกฎ แห้งและมืด กับ ชื้นและสว่าง ค่อนข้างครุมเครือ ให้ให้ผล MIDDLE เป็นฟัซซี่ค่าประมาณ 0 และเบี่ยงเบนเล็กน้อย ส่วนกฎ สว่าง และ แห้ง โอกาศเกิดฝนน้อย จึงได้ฟังก์ชั่น LOW ซ่งมีค่าในช่วงลบ

ในการจะประเมินว่าฝนจะตกหรือไม่จะทำการรวมผลทุกกฎไว้ด้วยกัน และค่าที่ได้จะเป็นผลตัดสิน โดนหากค่ามากกว่า 0 จะถือว่าฝนตก เครื่องควบคุมจะทำการเก็บผ้า และนอกจากนั้นจะถือว่าฝนไม่ตก

ตารางแสดงผลของการทำงาน



โดยที่จุดสีฟ้า แสดงถึงการทำนายว่าฝนตก และจุดสีแดง ทำนายว่าฝนไม่ตก จากผลลัพธ์ที่ได้เห็นว่า การตัดสินว่าฝนตกหรือไม่ไม่ไดแบ่งเป็นช่วงชัดเจน จะมีเป็นผลการทำนายออกเป็นกลุ่ม ๆ ซึ่งผลที่ได้อาจไม่ตรงกับความเข้าใจจริงๆนัก เนื่องจาก ฟังก์ชั่นฟัซซี่ที่ใช้ทำหมดเป็นฟังชั่นเส้นตรง ซึ่งไม่ค่อยมีความยืดหยุ่น และการออกแบบฟัซซี่ของฟังก์ชั่นเอ้าท์พุทนั้นก็มีความสำคัญมาก เพราะจะเป็นส่วนในการให้ความสำคัญต่อกฎแต่ละกฎให้สมเหตุสมผลและมีความถูกต้องยิ่งขึ้น

ภาคผนวก

Main.py

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

if __name__ == '__main__':
    fc = fuzzyClothline.Clothline()
    fBright = fuzzy.fuzzy([[0.0,0.0],[500.0,0.0],[1000.0,1.0]])
    fDark = fuzzy.fuzzy([[0.0,1.0],[400.0,1.0],[800.0,0.0]])
    fWet = fuzzy.fuzzy([[0.0,0.0],[60.0,0.0],[100.0,1.0]])
    fDry = fuzzy.fuzzy([[0.0,1.0],[50.0,1.0],[100.0,0.0]])

    fLow = fuzzy.fuzzy([[-2.5,0],[-1,1.0],[0.0,0.0]])
    fMedium = fuzzy.fuzzy([[-0.5,0],[0.0,1.0],[0.5,0.0]])
    fHigh = fuzzy.fuzzy([[0.0,0.0],[1.0,1.0],[3.5,0.0]])


    fc.addRule(fBright,fWet,fMedium)
    fc.addRule(fBright,fDry,fLow)
    fc.addRule(fDark,fWet,fHigh)
    fc.addRule(fDark,fDry,fMedium)


    w = np.arange(10,110, 10)
    l = np.arange(100,1100, 100)
    res = []

    fig = plt.figure(1)
    plt.title('Rule table')
    plt.xlabel('Humidity')
    plt.ylabel('Brightness')
    for i in range(10):
        for j in range(10):
            fc.doInference(l[i],w[j])
            # res.append(fc.doAction())
            if fc.doAction() == 'Raining':
                plt.plot(j*10, i*100, 'ro', ms=5)
            else:
                plt.plot(j*10, i*100, 'bo', ms=5)

    fig.savefig('exp1,'+str((i+1))+'.png')
```

fuzzyClothline.py

```
import fuzzy

class Clothline:
    class Rule:
        def __init__(self, _f1, _f2, _o):
            self.f1 = _f1
            self.f2 = _f2
            self.fo = _o

        def getA(self, _i1, _i2):
            return self.fo.getAlphaCut(min(self.f1.getMemberness(_i1), self.f2.getMemberness(_i2)))

    def __init__(self):
        self.ruleSet = []
        self.value = 0


    def addRule(self, f1, f2, of):
        self.ruleSet.append(self.Rule(f1,f2,of))

    def doInference(self, _i1, _i2):
        c = []
        for r in self.ruleSet:
            a = r.getA(_i1, _i2)
            c.append(r.f1.centroid(a))



        sum = [0,0]
        for j in c:
            sum[0] += j[0]*j[1]
            sum[1] += j[1]

        print(sum[0] , sum[1])
        if int(sum[1]) == 0:
            return 0
        self.value = int(sum[1]) == 0 and 0 or (sum[0] / sum[1])
        return self.value

    def doAction(self):
        return self.value > 0.0 and 'Raining' or 'Not Rain'
```

fuzzy.py

```
import numpy as np

class fuzzy:
    def __init__(self, _membership=[]):
        self.membership = _membership

    def getMemberness(self, val):
        res = 0
        for i in range(len(self.membership)):
            if val <= self.membership[i][0]:
                if i == 0:
                    res = self.membership[i][1]
                    break
                else:

                    res = (self.membership[i][0] > self.membership[i-1][0] and 0 or 1) - (val - self.membership[i-1][0])/(self.membership[i][0]-
self.membership[i-1][0])
                    break
            elif i == len(self.membership[i])-1:
                res = self.membership[i][1]
        return res

    def getAlphaCut(self, val):
        res = []
        for i in range(len(self.membership)-1):
            if self.membership[i][1] < val and self.membership[i+1][1] > val:
                res.append(self.membership[i])
                res.append([val*(self.membership[i+1][0]-self.membership[i][0])+self.membership[i][0],val])
                # print('A')

            elif self.membership[i][1] > val and self.membership[i+1][1] < val:
                res.append([self.membership[i][0],val])
                res.append([(1-val)*(self.membership[i+1][0]-self.membership[i][0])+self.membership[i][0], val])
                # print('b')

            elif self.membership[i][1] < val:
                res.append(self.membership[i])
                # print('c')
            elif self.membership[i][1] >= val:
                res.append([self.membership[i][0],val])
                # print('d')

        if self.membership[-1][1] < val:
            res.append(self.membership[-1])
            # print('e')
        else:
```

```
            res.append([self.membership[-1][0],val])
            # print('f')

        return res

    def getCentroid(self):
        return self.centroid(self.membership)

    def centroid(self, _mem):
        c = []
        matter = 0.0
        for i in range(len(_mem)-1):
            if _mem[i][0] < _mem[i+1][0]:
                tmp = [_mem[i][0]+((_mem[i+1][0]-_mem[i][0])*(2.0/3.0)), _mem[i+1][1]/3.0]
                c.append(tmp)
                # matter += 0.5*(_mem[i+1][0]-_mem[i][0])*_mem[i+1][1]
            elif _mem[i][0] > _mem[i+1][0]:
                tmp = [_mem[i][0]+((_mem[i+1][0]-_mem[i][0])*(1.0/3.0)), _mem[i+1][1]/3.0]
                c.append(tmp)
                # matter += 0.5*(_mem[i+1][0]-_mem[i][0])*_mem[i][1]
            else:
                tmp = [_mem[i][0]+((_mem[i+1][0]-_mem[i][0])/2.0), _mem[i][1]/2.0]
                # matter += (_mem[i+1][0]-_mem[i][0])*_mem[i+1][1]

        sum = [0,0]
        for j in c:
            sum[0] += j[0]*j[1]
            sum[1] += j[1]
            print(j[0], j[1])
            matter += j[1]

        return sum[1]==0 and [0,0] or [sum[0] / sum[1], matter]
```