FUZZY LOGIC PROJECT



*Survival Chance Calculator for Tomato Plant (SCCforTP)*

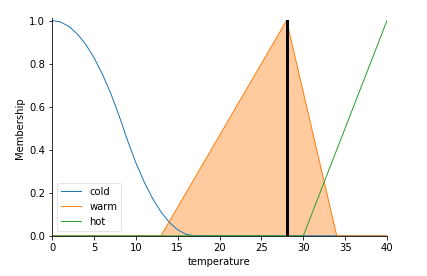
# Project Goal and Definition

The “*SCCforTP”* system is designed with fuzzy logic algorithms to calculate survival chance of tomato plants in an environment by looking at its weather temperature, soil moisture and lighting level.The system can be used more efficiently with an Arduino or Raspberry Pi with LDR, DHT11 and Soil Moisture sensors on it.

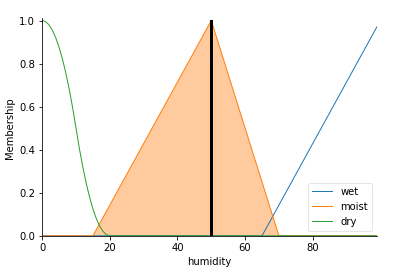
# Application Membership

|  |  |  |  |
| --- | --- | --- | --- |
| **INPUTS** | **INTERVAL** | **LINGUISTIC VARIABLES** | **PARAMETERS** |
| **Weather Temperature** | 0-41 °C | Cold | 0-17 |
| Warm | 13-34 |
| Hot | 30-41 |
| **Soil Moisture** | 0-100 % | Wet | 65-100 |
| Moist | 15-70 |
| Dry | 0-20 |
| **Lighting Level** | 0-1000 LDR-Unit | Low | 0-300 |
| Good | 250-900 |
| High | 800-1000 |
| **OUTPUT** | **INTERVAL** | **LINGUISTIC VARIABLES** | **PARAMETERS** |
| **Survival Chance** | 0-90 % | Very Low | 0–15 |
| Low | 13-30 |
| Pretty | 25-45 |
| Good | 40-70 |
| High | 65-90 |

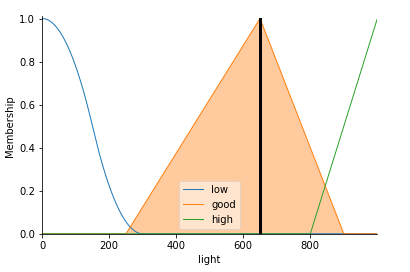
**Weather Temperature Membership Function Plot**



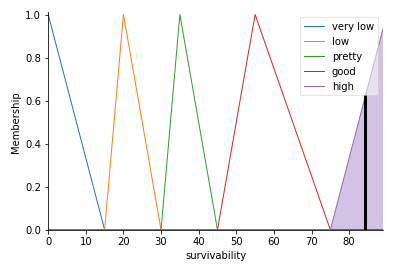
**Soil Moisture Membership Function Plot**

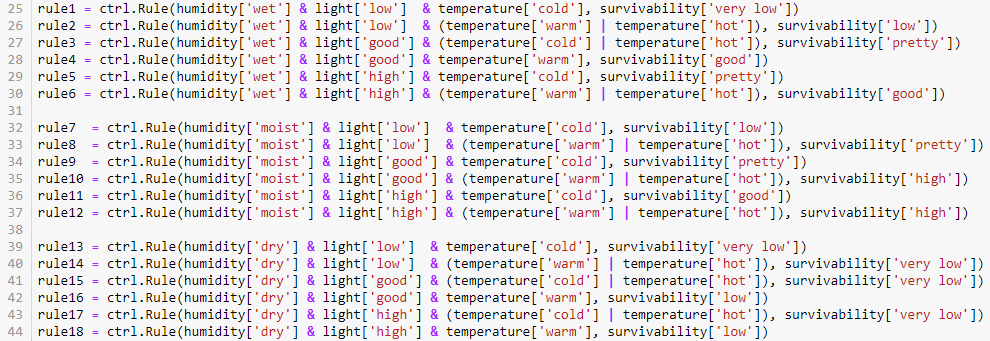


**Lighting Level Membership Function Plot**



**Survival Chance Membership Function Plot**



**Fuzzy Rules Of The System**

As shown above, The system has 18 rules to calculate survival chance of the tomato plant. This rules are based on 3 inputs which are Temperature, Soil Moisture (called as humidity in the code) and Lighting-level.

While ‘dry’ soil has lowest survival chance, ‘moist’ one has the highest and ‘wet’ one is in the middle of both others because tomato plants do not love high moisture.

While ‘low’ light level has the lowest survival chance, ‘good’ and ‘high’ both has the highest survival chance for tomato plants.

Lastly, while ‘cold’ temperature has the lowest survival chance, ‘warm’ one has the highest and ‘hot’ one has close chances with ‘warm’ one but still it’s in the middle of both others.

**Source Code**

%matplotlib inline

import numpy as np

import skfuzzy as fuzz

import matplotlib.pyplot as plt

from skfuzzy import control as ctrl

temperature = ctrl.Antecedent(np.arange(0, 41, 1), 'temperature')

humidity = ctrl.Antecedent(np.arange(0, 100, 1), 'humidity')

light = ctrl.Antecedent(np.arange(0, 1000, 1), 'light')

survivability = ctrl.Consequent(np.arange(0, 90, 1), 'survivability')

temperature['cold'] = fuzz.zmf(temperature.universe, 0, 17)

temperature['warm'] = fuzz.trimf(temperature.universe, [13, 28, 34])

temperature['hot'] = fuzz.trimf(temperature.universe, [30, 40, 41])

humidity['wet'] = fuzz.trimf(humidity.universe, [65, 100, 100])

humidity['moist'] = fuzz.trimf(humidity.universe, [15, 50, 70])

humidity['dry'] = fuzz.zmf(humidity.universe, 0, 20)

light['low'] = fuzz.zmf(light.universe, 0, 300)

light['good'] = fuzz.trimf(light.universe, [250, 650, 900])

light['high'] = fuzz.trimf(light.universe, [800, 1000, 1000])

survivability['very low'] = fuzz.trimf(survivability.universe, [0, 0, 15])

survivability['low'] = fuzz.trimf(survivability.universe, [15, 20, 30])

survivability['pretty'] = fuzz.trimf(survivability.universe, [30, 35, 45])

survivability['good'] = fuzz.trimf(survivability.universe, [45, 55, 75])

survivability['high'] = fuzz.trimf(survivability.universe, [75, 90, 90])

rule1 = ctrl.Rule(humidity['wet'] & light['low'] & temperature['cold'], survivability['very low'])

rule2 = ctrl.Rule(humidity['wet'] & light['low'] & (temperature['warm'] | temperature['hot']), survivability['low'])

rule3 = ctrl.Rule(humidity['wet'] & light['good'] & (temperature['cold'] | temperature['hot']), survivability['pretty'])

rule4 = ctrl.Rule(humidity['wet'] & light['good'] & temperature['warm'], survivability['good'])

rule5 = ctrl.Rule(humidity['wet'] & light['high'] & temperature['cold'], survivability['pretty'])

rule6 = ctrl.Rule(humidity['wet'] & light['high'] & (temperature['warm'] | temperature['hot']), survivability['good'])

rule7 = ctrl.Rule(humidity['moist'] & light['low'] & temperature['cold'], survivability['low'])

rule8 = ctrl.Rule(humidity['moist'] & light['low'] & (temperature['warm'] | temperature['hot']), survivability['pretty'])

rule9 = ctrl.Rule(humidity['moist'] & light['good'] & temperature['cold'], survivability['pretty'])

rule10 = ctrl.Rule(humidity['moist'] & light['good'] & (temperature['warm'] | temperature['hot']), survivability['high'])

rule11 = ctrl.Rule(humidity['moist'] & light['high'] & temperature['cold'], survivability['good'])

rule12 = ctrl.Rule(humidity['moist'] & light['high'] & (temperature['warm'] | temperature['hot']), survivability['high'])

rule13 = ctrl.Rule(humidity['dry'] & light['low'] & temperature['cold'], survivability['very low'])

rule14 = ctrl.Rule(humidity['dry'] & light['low'] & (temperature['warm'] | temperature['hot']), survivability['very low'])

rule15 = ctrl.Rule(humidity['dry'] & light['good'] & (temperature['cold'] | temperature['hot']), survivability['very low'])

rule16 = ctrl.Rule(humidity['dry'] & light['good'] & temperature['warm'], survivability['low'])

rule17 = ctrl.Rule(humidity['dry'] & light['high'] & (temperature['cold'] | temperature['hot']), survivability['very low'])

rule18 = ctrl.Rule(humidity['dry'] & light['high'] & temperature['warm'], survivability['low'])

survivability\_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8, rule9, rule10,

rule11, rule12, rule13, rule14, rule15, rule16, rule17, rule18])

guess = ctrl.ControlSystemSimulation(survivability\_ctrl)

try :

print("(0-41 C) Temperature of the environment: ")

getTemp = int(input(""))

guess.input['temperature'] = getTemp

print("(0-100 %) Moisture of the soil: ")

getHum = int(input(""))

guess.input['humidity'] = getHum

print("(0-1000) Light level of the environment: ")

getLight = int(input(""))

guess.input['light'] = getLight

except(IOError, OSError, ValueError) as err :

print("Error : " + err)

guess.compute()

if getTemp <= 10 | getHum <=10 | getLight <= 10:

guess.output['survivability'] = 'lower than 10'

print("\nSurvival chance for this environment is " + str(guess.output['survivability']) + "%")

elif getTemp >=41:

guess.output['survivability'] = 'lower than 10'

print("\nSurvival chance for this environment is " + str(guess.output['survivability']) + "%")

else:

print("\nSurvival chance for this environment is " + str(guess.output['survivability']) + "%")

survivability.view(sim = guess)

temperature.view(sim = guess)

humidity.view(sim = guess)

light.view(sim = guess)

**REFERENCES**

1. **https://www.cs.princeton.edu/courses/archive/fall07/cos436/HIDDEN/Knapp/fuzzy004.htm**
2. **https://pythonhosted.org/scikit-fuzzy/api/skfuzzy.membership.html**
3. **http://www.inf.u-szeged.hu/~szepet/python/tip.py**
4. **https://www.researchgate.net/post/How\_to\_make\_sure\_the\_number\_of\_rules\_of\_fuzzy\_system**