

Week 9 Lab

In this lab, we will implement a fuzzy inference system (FIS) to control an intelligent traffic light system. Here, we consider a simple intersection which is made of two streets, Main and Side, and no turning lanes. By default, the traffic lights are green for the Main street and red for the Side street. When there is some traffic on the Side, the FIS is activated. Street sensors provide information on the number of cars on the Main street (incoming traffic) and the number of cars on the Side street (waiting traffic). Based on this information, the FIS needs to make decisions on:

- waiting duration: the duration before the traffic lights on the Side street switch to green; and
- green duration: the duration for which the Side traffic lights stay green. Both durations are measured in seconds.

Below is a basic implementation for the traffic light FIS, followed by testing it on some test cases and some lab tasks.

A. Basic FIS implementation

```
In [2]: pip install -U scikit-fuzzy
```

```
Collecting scikit-fuzzy
  Downloading scikit_fuzzy-0.5.0-py2.py3-none-any.whl.metadata (2.6 kB)
  Downloading scikit_fuzzy-0.5.0-py2.py3-none-any.whl (920 kB)
    _____ 920.8/920.8 kB 20.0 MB/s eta 0:00:00
Installing collected packages: scikit-fuzzy
Successfully installed scikit-fuzzy-0.5.0
```

1- Inputs & outputs

```
In [25]: import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt
from skfuzzy import control as ctrl

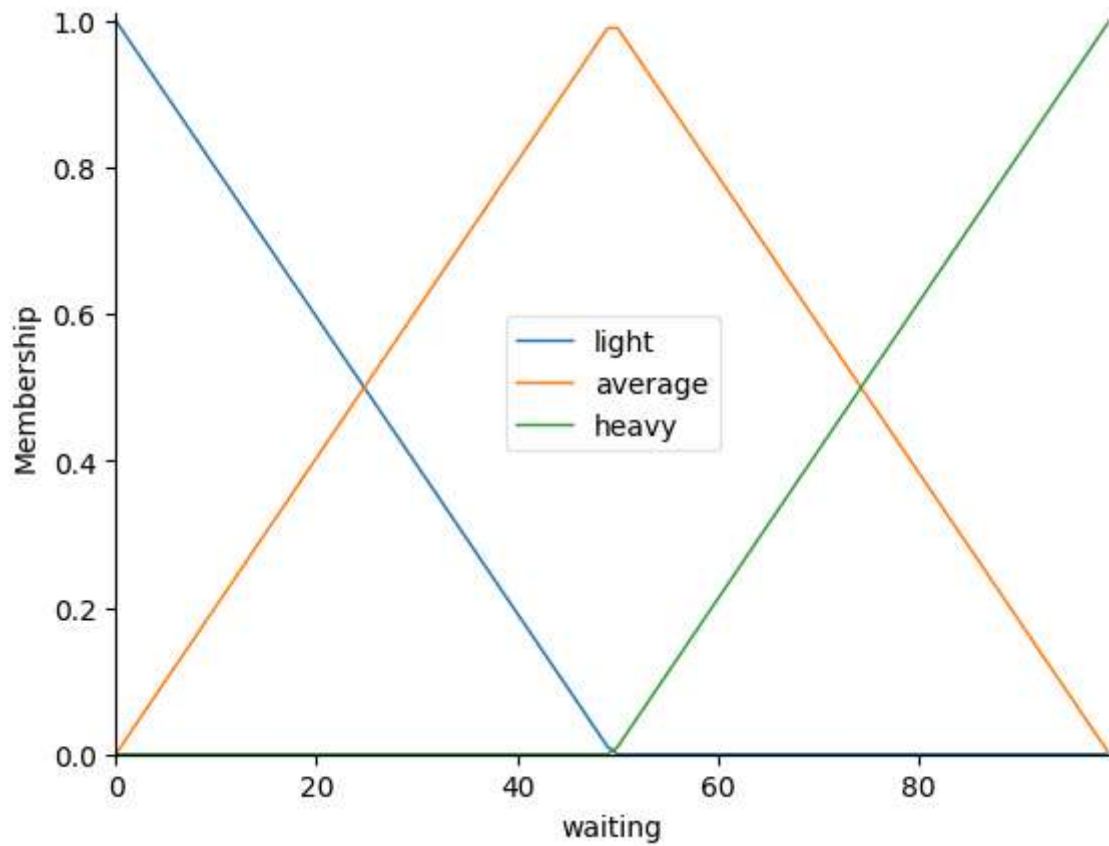
waitingTraffic = ctrl.Antecedent(np.arange(1, 100, 1), 'waiting')
incomingTraffic = ctrl.Antecedent(np.arange(0, 100, 1), 'incoming')

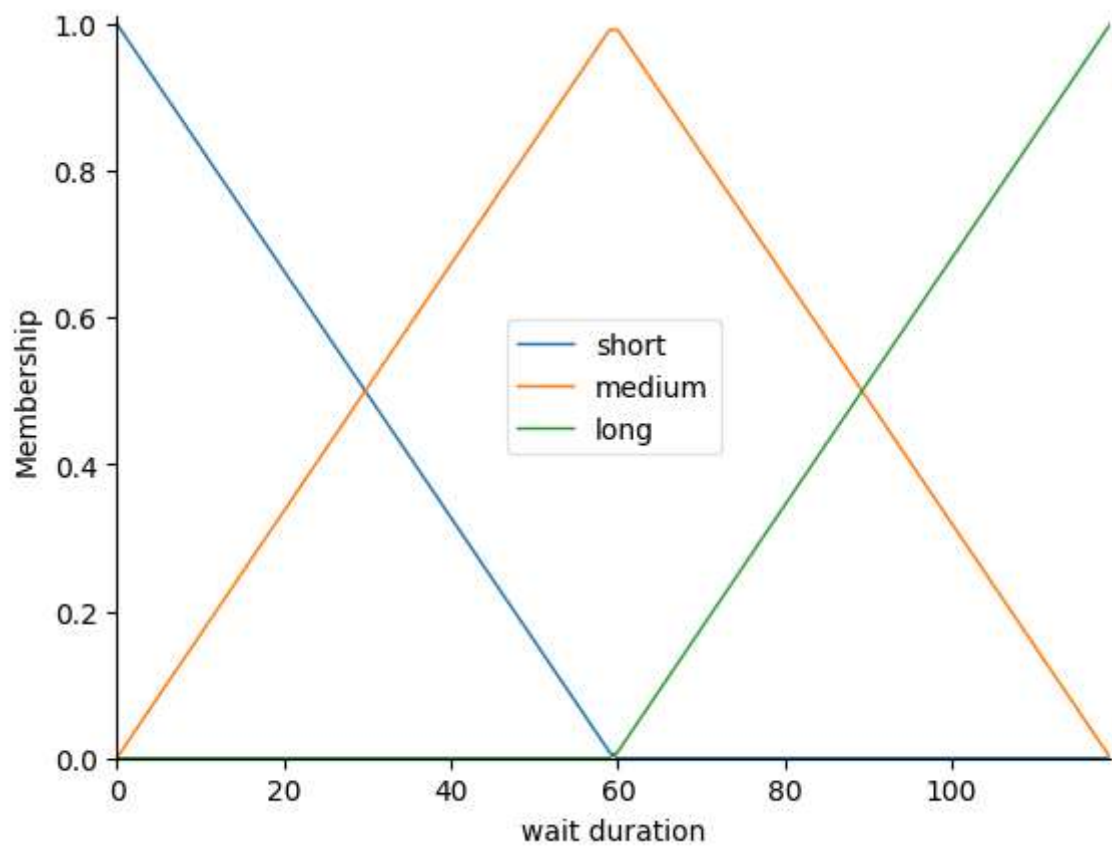
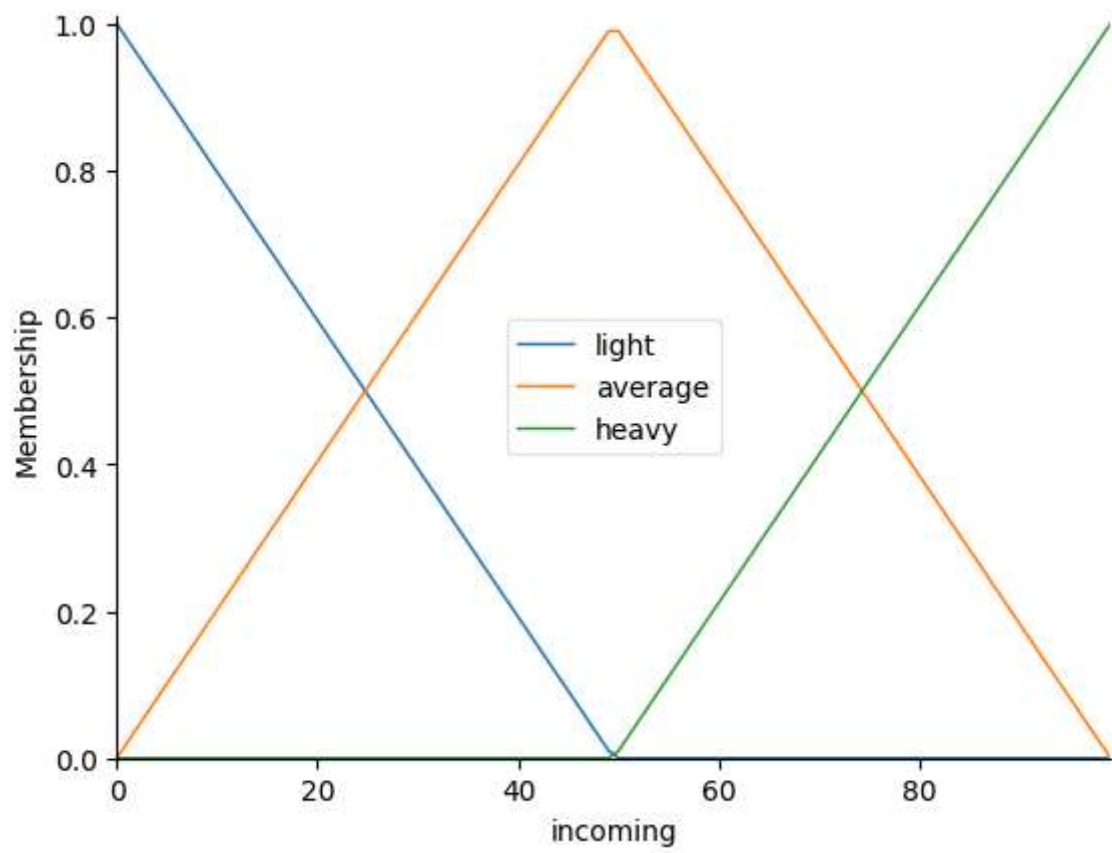
waitingDuration = ctrl.Consequent(np.arange(0, 120, 1), 'wait duration')
openDuration = ctrl.Consequent(np.arange(0, 120, 0.1), 'open duration')
```

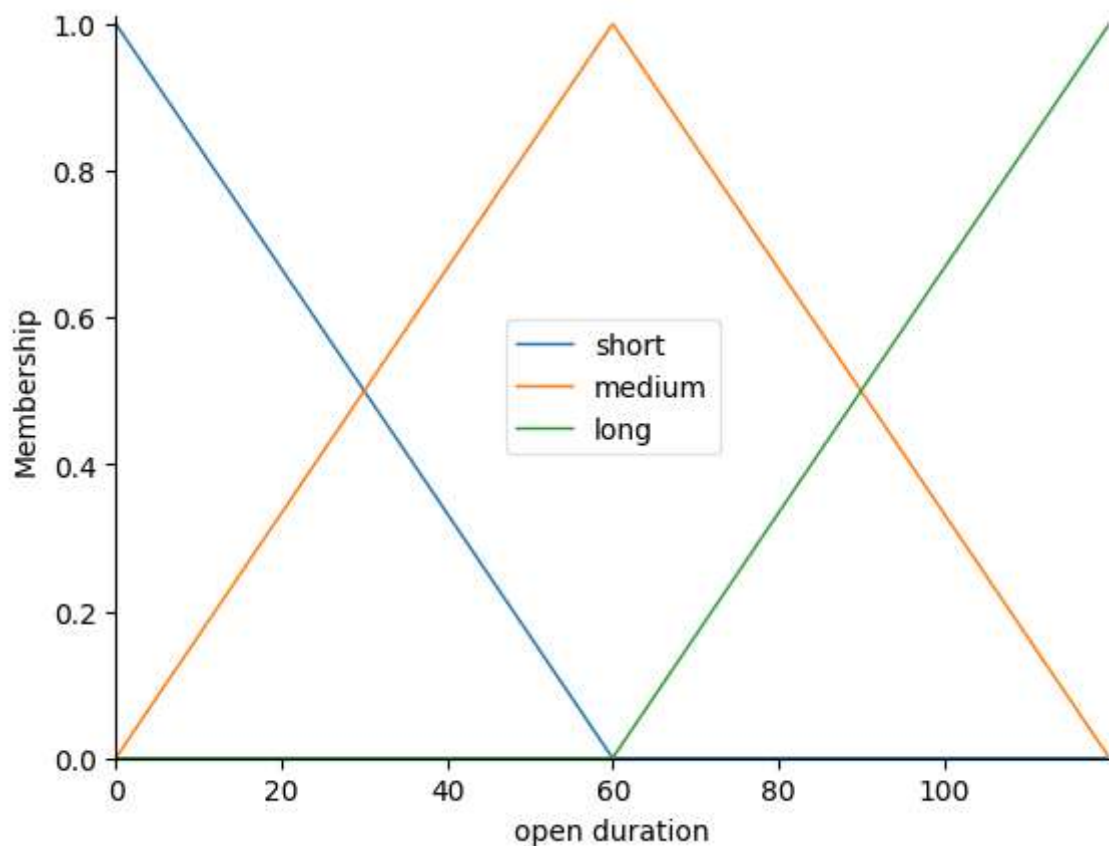
2- Fuzzification & membership functions

```
In [26]: fuzzySetNamesTraffic= ['light', 'average', 'heavy']
waitingTraffic.automf(names=fuzzySetNamesTraffic)
incomingTraffic.automf(names=fuzzySetNamesTraffic)
waitingTraffic.view()
incomingTraffic.view()

fuzzySetNamesDuration = ['short', 'medium', 'long']
waitingDuration.automf(names=fuzzySetNamesDuration)
openDuration.automf(names=fuzzySetNamesDuration)
waitingDuration.view()
openDuration.view()
```







3- Knowledge base

```
In [28]: rule1 = ctrl.Rule(incomingTraffic['light'] & waitingTraffic['light'], [waitingDuration, openDuration])
rule2 = ctrl.Rule(incomingTraffic['light'] & waitingTraffic['average'], [waitingDuration, openDuration])
rule3 = ctrl.Rule(incomingTraffic['light'] & waitingTraffic['heavy'], [waitingDuration, openDuration])

rule4 = ctrl.Rule(incomingTraffic['average'] & waitingTraffic['light'], [waitingDuration, openDuration])
rule5 = ctrl.Rule(incomingTraffic['average'] & waitingTraffic['average'], [waitingDuration, openDuration])
rule6 = ctrl.Rule(incomingTraffic['average'] & waitingTraffic['heavy'], [waitingDuration, openDuration])

rule7 = ctrl.Rule(incomingTraffic['heavy'] & waitingTraffic['light'], [waitingDuration, openDuration])
rule8 = ctrl.Rule(incomingTraffic['heavy'] & waitingTraffic['average'], [waitingDuration, openDuration])
rule9 = ctrl.Rule(incomingTraffic['heavy'] & waitingTraffic['heavy'], [waitingDuration, openDuration])
```

4- Inference Engine

```
In [ ]: # add any code if needed
```

5- Defuzzification

```
In [50]: waitingDuration.defuzzify_method = 'centroid'
openDuration.defuzzify_method = 'centroid'
```

B. Model Usage

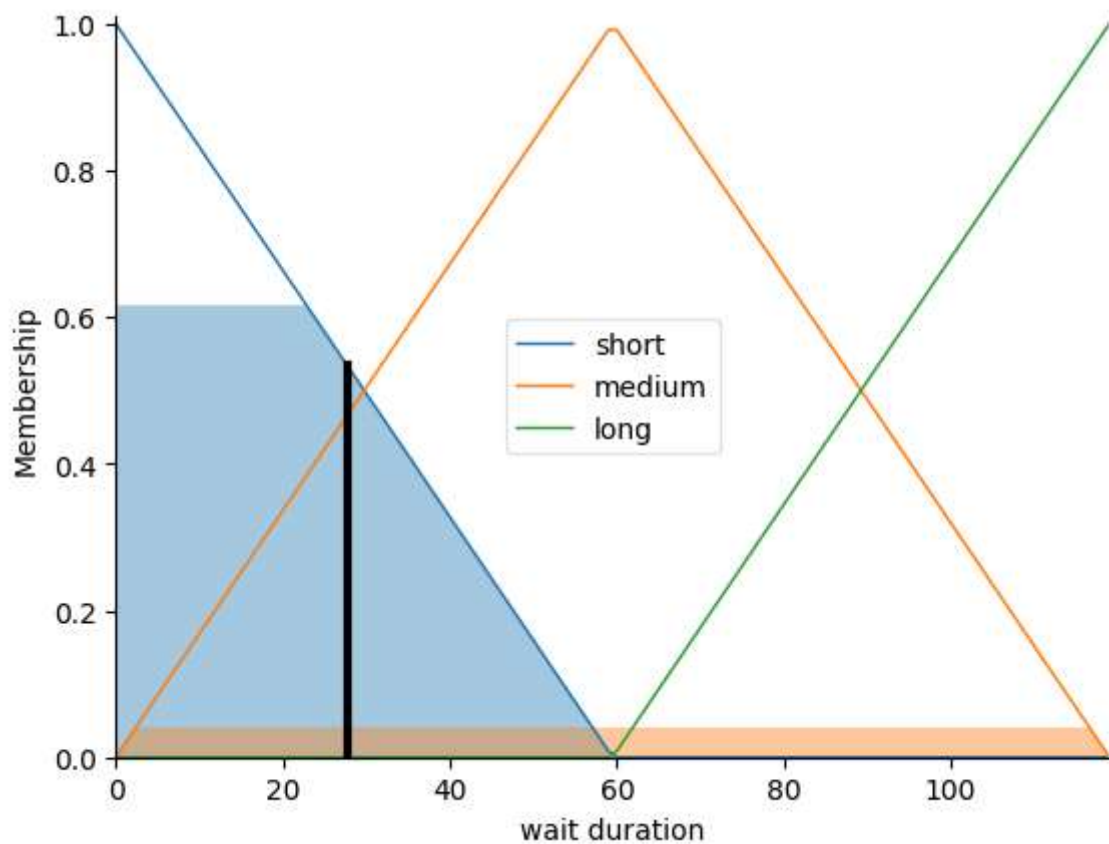
Testing on a single test case and visualising the results.

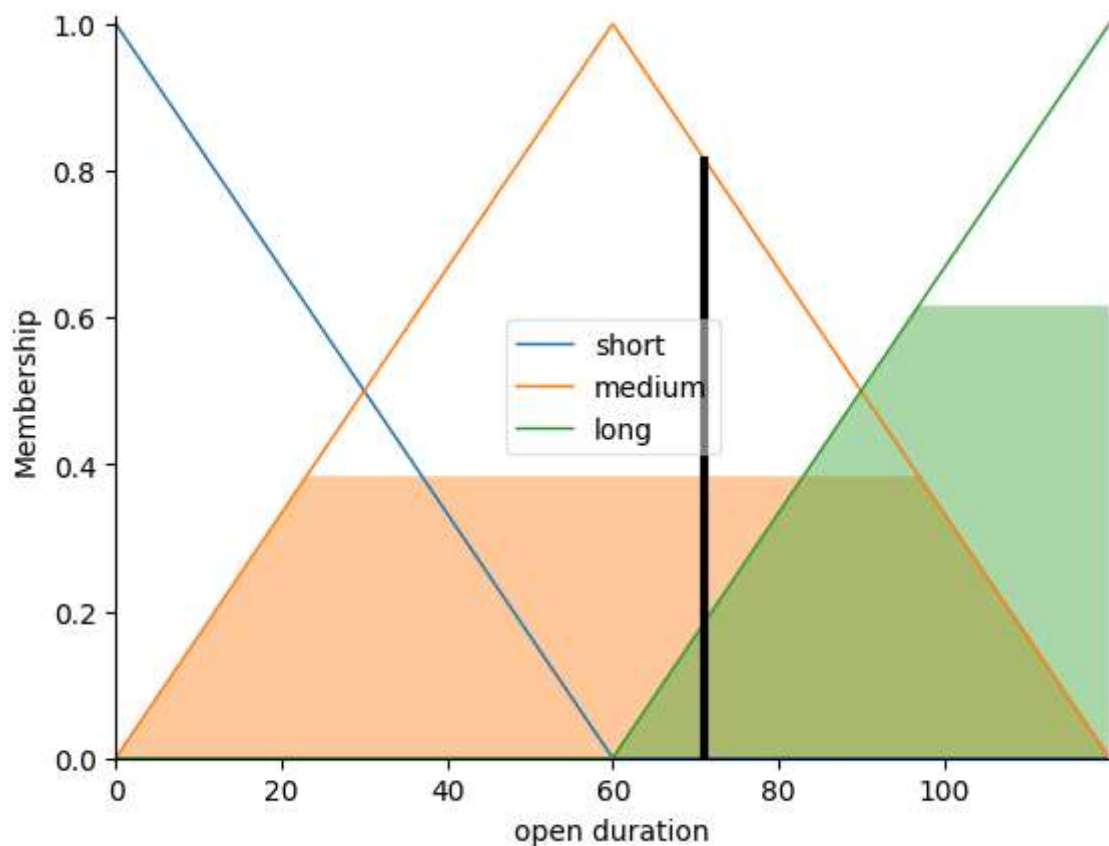
```
In [52]: fis = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8, rule9])
fis_simulator = ctrl.ControlSystemSimulation(fis)

# Now, test your FIS with some inputs & check the outputs
fis_simulator.input['incoming'] = 2
fis_simulator.input['waiting'] = 80
fis_simulator.compute()
print(round(fis_simulator.output['wait duration'],0))
waitingDuration.view(sim= fis_simulator)
print(round(fis_simulator.output['open duration'],0))
openDuration.view(sim= fis_simulator)
```

28.0

71.0





Testing on multiple test cases

```
In [40]: incoming_test = [0,15, 20, 40 , 12, 8, 59]
         waiting_test  = [2, 1, 25 , 5 , 29, 8, 5]

         for i in range(incoming_test.__len__()):
             fis_simulator.input['incoming'] = incoming_test[i]
             fis_simulator.input['waiting'] = waiting_test[i]
             fis_simulator.compute()
             print( ' waiting: ' + str(waiting_test[i]) + ', incoming: ' + str(incoming_test[i])

incoming: 0, waiting: 2, wait duration: 0.0, open duration: 25.0
incoming: 15, waiting: 1, wait duration: 0.0, open duration: 24.0
incoming: 20, waiting: 25, wait duration: 0.0, open duration: 53.0
incoming: 40, waiting: 5, wait duration: 48.0, open duration: 32.0
incoming: 12, waiting: 29, wait duration: 0.0, open duration: 55.0
incoming: 8, waiting: 8, wait duration: 0.0, open duration: 37.0
incoming: 59, waiting: 5, wait duration: 48.0, open duration: 32.0
```

C. Lab tasks

- Examine the performance of the FIS under the test cases provided in the previous section. You can also provide additional test cases.
- What are the key performance issues?
- Suggest some fixes that could improve these issue.
- Modify the above code to implement the suggested fixes
- Compare the test performance before & after changes.

