Genetic Algorithm & Fuzzy Logic

Semester-5

Practical - 11

Name: Shivam Tawari

Roll no: A-58

Aim: Implementation of Fuzzy Logic Controller

Theory:

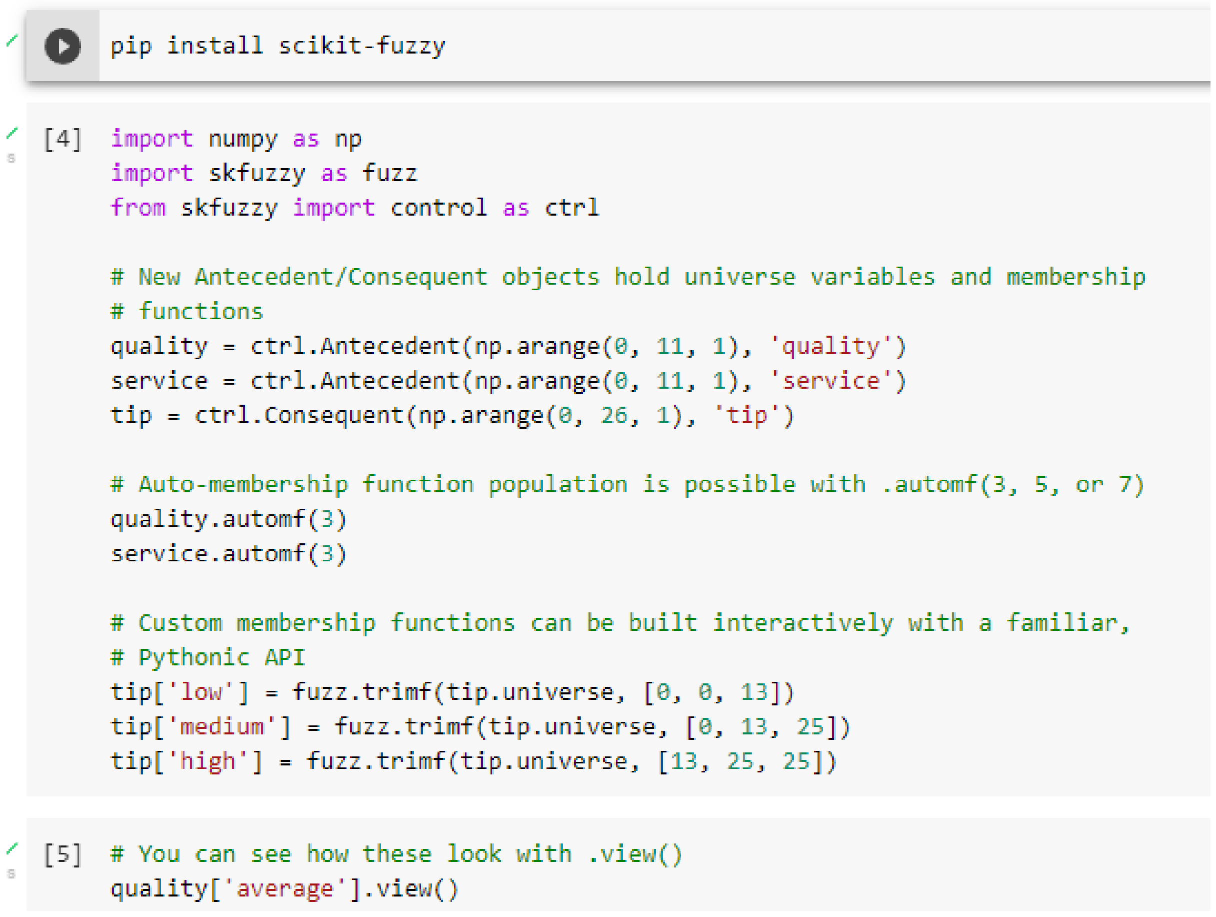
Fuzzy Control Systems: The Tipping Problem

Let's create a fuzzy control system which models how you might choose to tip at a restaurant. When tipping, you consider the service and food quality, rated between 0 and 10. You use this to leave a tip of between 0 and 25%.

We would formulate this problem as:

- **Antecednets (Inputs)**
  - **service**
    - Universe (i.e., crisp value range): How good was the service of the wait staff, on a scale of 0 to 10?
    - Fuzzy set (i.e., fuzzy value range): poor, acceptable, amazing
  - **food quality**
    - Universe: How tasty was the food, on a scale of 0 to 10?
    - Fuzzy set: bad, decent, great
- **Consequents (Outputs)**
  - **tip**
    - Universe: How much should we tip, on a scale of 0% to 25%
    - Fuzzy set: low, medium, high
- **Rules**
  - IF the *service* was good *or* the *food quality* was good, THEN the tip will be high.
  - IF the *service* was average, THEN the tip will be medium.
  - IF the *service* was poor *and* the *food quality* was poor THEN the tip will be low.
- **Usage**
  - **If I tell this controller that I rated:**
    - the service as 9.8, and
    - the quality as 6.5,
  - **it would recommend I leave:**
    - a 20.2% tip.

Code and Output:



```
pip install scikit-fuzzy
```

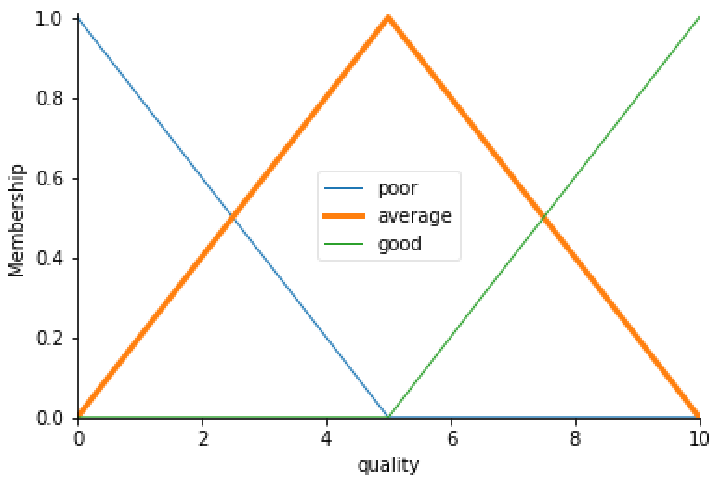
[4]

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

# New Antecedent/Consequent objects hold universe variables and membership
# functions
quality = ctrl.Antecedent(np.arange(0, 11, 1), 'quality')
service = ctrl.Antecedent(np.arange(0, 11, 1), 'service')
tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')

# Auto-membership function population is possible with .automf(3, 5, or 7)
quality.automf(3)
service.automf(3)

# Custom membership functions can be built interactively with a familiar,
# Pythonic API
tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])
tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])
tip['high'] = fuzz.trimf(tip.universe, [13, 25, 25])
```

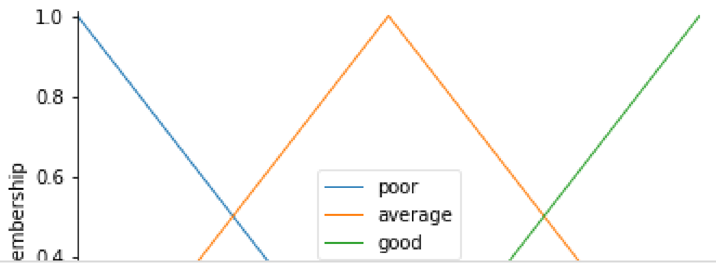
[5]

```
# You can see how these look with .view()
quality['average'].view()
```

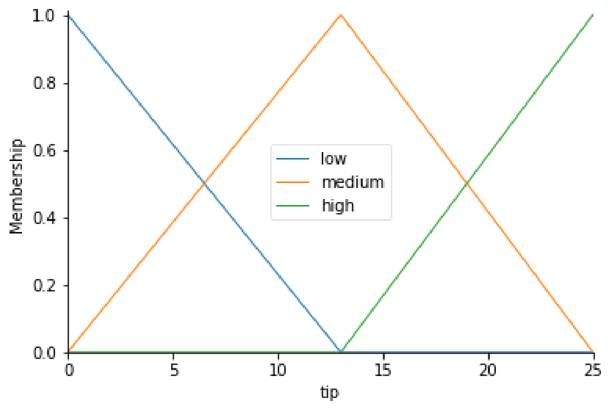
```
# You can see how these look with .view()
quality['average'].view()
```



```
[6] service.view()
```



```
[7] tip.view()
```



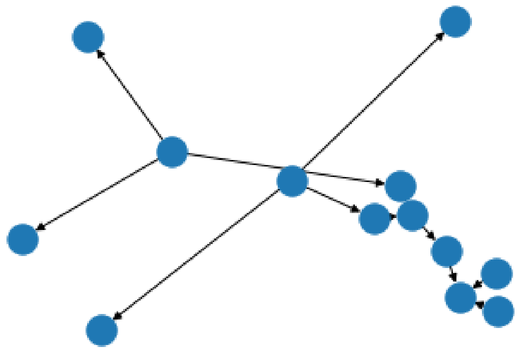
```
[8] rule1 = ctrl.Rule(quality['poor'] | service['poor'], tip['low'])
    rule2 = ctrl.Rule(service['average'], tip['medium'])
    rule3 = ctrl.Rule(service['good'] | quality['good'], tip['high'])

    rule1.view()
```

```
(<Figure size 432x288 with 1 Axes>,
 <matplotlib.axes._subplots.AxesSubplot at 0x7f346cb98950>)
```

+ Code + Text

```
rule1 = ctrl.Rule(quality['poor'] | service['poor'], tip['low'])
rule2 = ctrl.Rule(service['average'], tip['medium'])
rule3 = ctrl.Rule(service['good'] | quality['good'], tip['high'])

rule1.view()
```

```
(<Figure size 432x288 with 1 Axes>,
 <matplotlib.axes._subplots.AxesSubplot at 0x7f346cb98950>)
```



```
[9] tipping_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
```

```
[10] tipping = ctrl.ControlSystemSimulation(tipping_ctrl)
```

Conclusion: Hence, Implementation of Fuzzy Logic Controller