SCT Experiment No: 5

Name: Abhishek S Waghchaure

PRN: 1032221714

Dept: FY M Tech DSA(2022-24)

Aim:

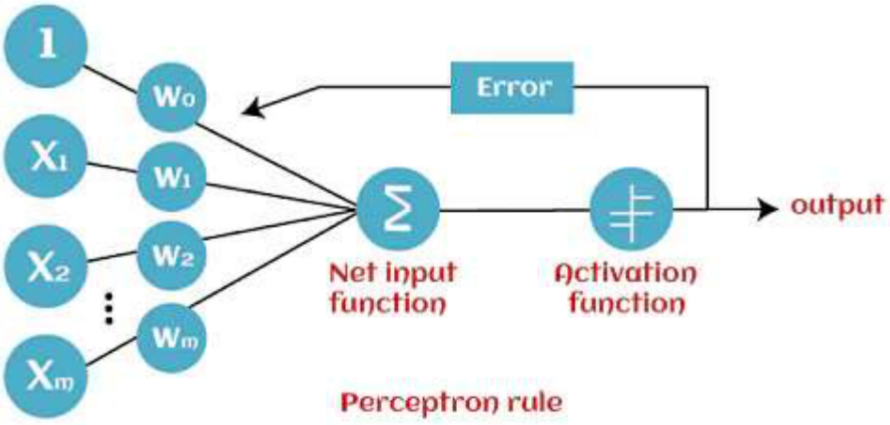
Implementing a fuzzy controller system for a real world application.

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 (ie, crisp value range): How good was the service of the wait staff, on a scale of 0 to 10?**
    - **Fuzzy set (ie, 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 19.84% tip.**



Perceptron rule

Code:

Creating the Tipping Controller Using the skfuzzy control API¶

We can use the skfuzzy control system API to model this. First, let's define fuzzy variables

In [16]:

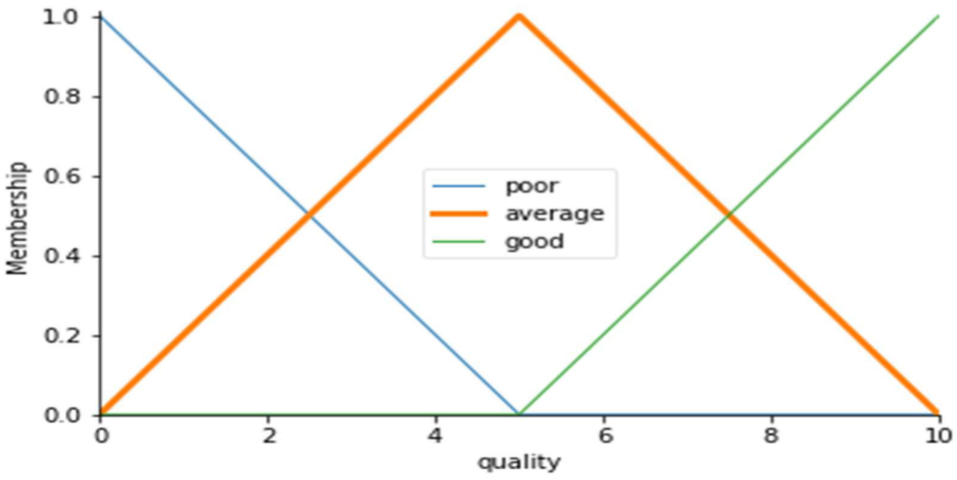
```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

#  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])
```

To help understand what the membership looks like, use the view methods.

In [17]:

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

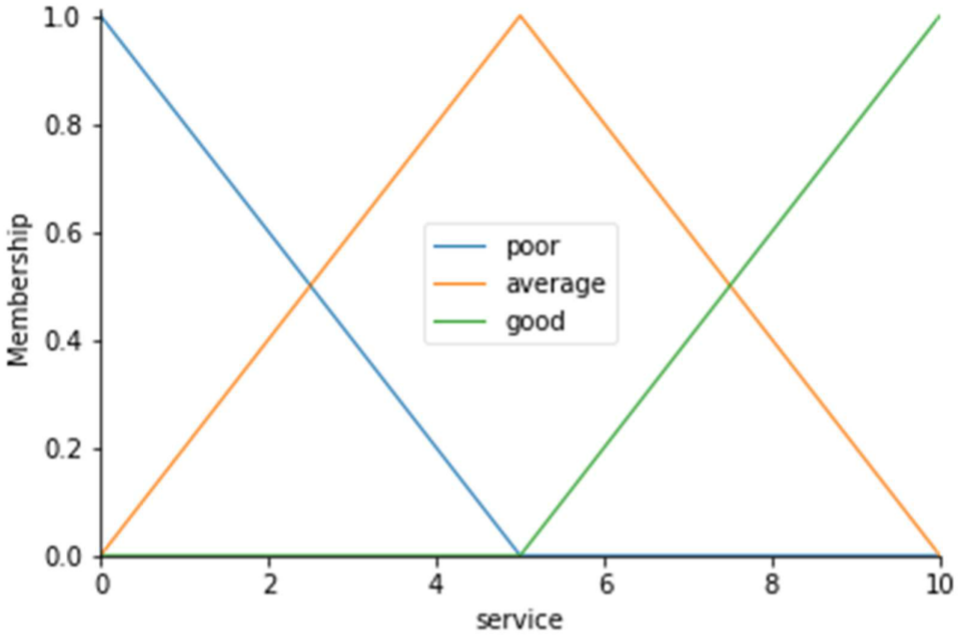
```
C:\Users\sunsp\anaconda3\lib\site-packages\skfuzzy\control\term.py:74: UserWarning: Matplotlib is currently using module://matplotlib_inline.backend _inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```



In [18]:

```
service.view()
```

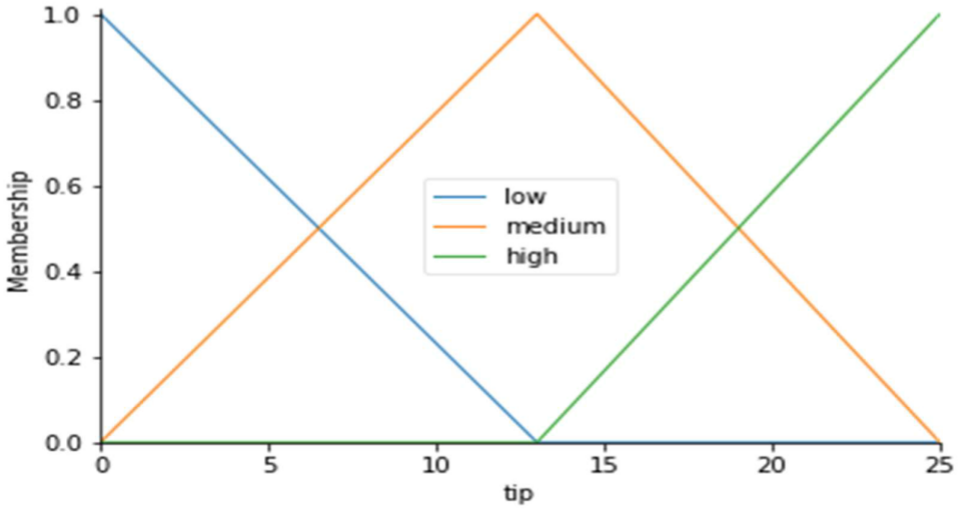
```
C:\Users\sunsp\anaconda3\lib\site-packages\skfuzzy\control\fuzzyvariable.p y:122:
UserWarning: Matplotlib is currently using module://matplotlib_inli
ne.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```



In [19]:

```
tip.view()
```

```
C:\Users\sunsp\anaconda3\lib\site-packages\skfuzzy\control\fuzzyvariable.p y:122:
UserWarning: Matplotlib is currently using module://matplotlib_inli
ne.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```



Fuzzy rules¶

Now, to make these triangles useful, we define the fuzzy relationship between input and output variables. For the purposes of our example, consider three simple rules:

1. **If the food is poor OR the service is poor, then the tip will be low**
2. **If the service is average, then the tip will be medium**
3. **If the food is good OR the service is good, then the tip will be high.**

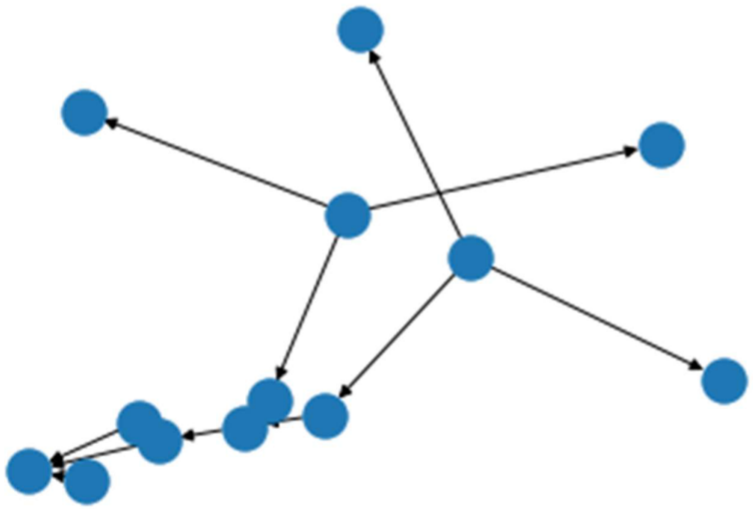
Most people would agree on these rules, but the rules are fuzzy. Mapping the imprecise rules into a defined, actionable tip is a challenge. This is the kind of task at which fuzzy logic excels.

In [20]:

```
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()
```

Out[20]:

(<Figure size 432x288 with 1 Axes>, <AxesSubplot:>)



Control System Creation and Simulation¶

Now that we have our rules defined, we can simply create a control system via:

In [21]:

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

In order to simulate this control system, we will create a ControlSystemSimulation. Think of this object representing our controller applied to a specific set of cirucmstances. For tipping,

this might be tipping Sharon at the local brew-pub. We would create another ControlSystemSimulation when we're trying to apply our tipping_ctrl for Travis at the cafe because the inputs would be different.

In [22]:

```
tipping = ctrl.ControlSystemSimulation(tipping_ctrl)
```

In order to simulate this control system, we will create a ControlSystemSimulation. Think of this object representing our controller applied to a specific set of cirucmstances. For tipping, this might be tipping Sharon at the local brew-pub. We would create another ControlSystemSimulation when we're trying to apply our tipping_ctrl for Travis at the cafe because the inputs would be different.

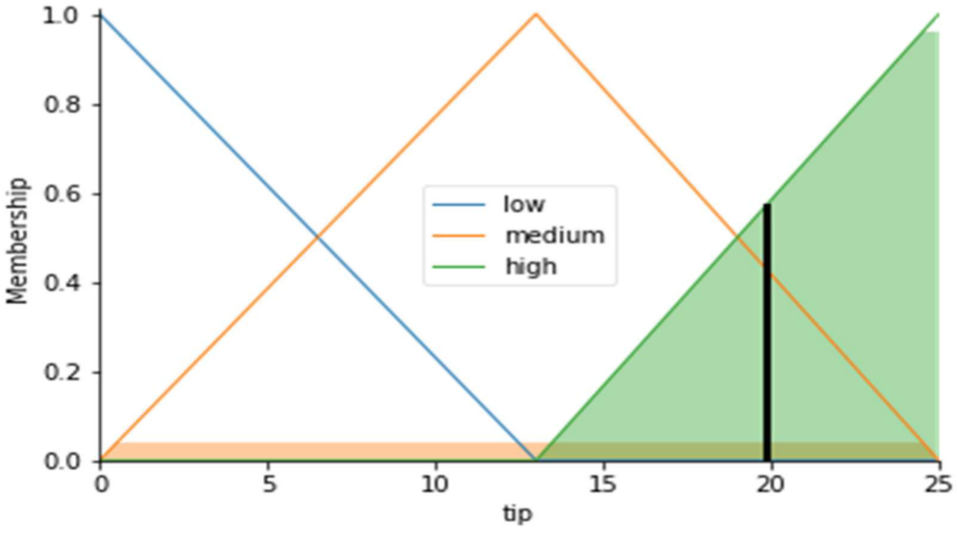
In [23]:

```
#  Pass inputs to the ControlSystem using Antecedent labels with Pythonic A PI

#  Note: if you like passing many inputs all at once, use .inputs(dict_of_d ata)

tipping.input['quality'] = 6.5
tipping.input['service'] = 9.8

#  Crunch the numbers
tipping.compute()
```

In [24]:

```
print(tipping.output['tip'])
tip.view(sim=tipping)
```

```
19.847607361963192

C:\Users\sunsp\anaconda3\lib\site-packages\skfuzzy\control\fuzzyvariable.p  y:122:
UserWarning: Matplotlib is currently using module://matplotlib_inli
ne.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```



Conclusion:

Fuzzy logic is applied with great success in various control application. Almost all the consumer products have fuzzy control. Some of the examples include controlling your room temperature with the help of air-conditioner, anti-braking system used in vehicles, control on traffic lights, washing machines, large economic systems, etc.

Reference:

https://www.tutorialspoint.com/fuzzy_logic/fuzzy_logic_control_system.htm

https://www.geeksforgeeks.org/fuzzy-logic-control-system/

https://www.researchgate.net/figure/Fuzzy-logic-controller-design-conclusion_tbl2_310835123

https://en.wikipedia.org/wiki/Fuzzy_control_system