Himanshu Singh, Yunis Ahmad Lone,- **Deep Neuro-Fuzzy Systems with Python** (With Case Studies and Applications from the Industry), - Apress, - 2020.

You can use the following line to install the **Scikit Fuzzy** package in the Python environment:

pipinstall scikit fuzzy

Figures 1-10 through 1-14 show the different types of membership functions and the curves that they represent.

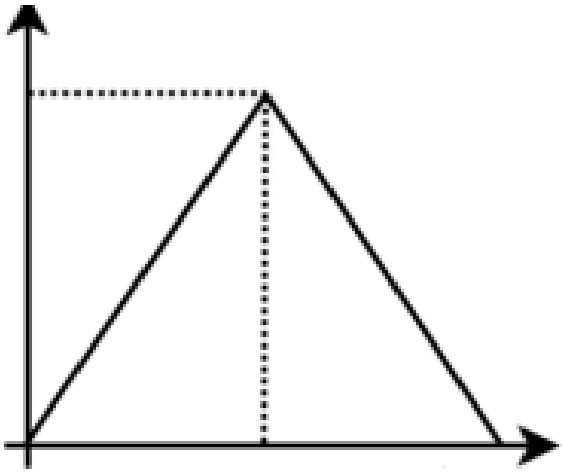


Figure 1-10. *Triangular membership function*

The graph in Figure 1-10 represents a triangular membership function, and you can use the `trimf` method from the `skfuzzy` package to find and plot the points.

Here is the sample code. The next chapter discusses this function in detail.

The following code takes an example where a person goes into a restaurant and tips a waiter. For tipping purposes, the quality of service is rated from 0 to 10. This example looks only at the service quality for now, but later it will discuss the actual tipping problem.

```
import numpy as np
import skfuzzy as sk

#Defining the Numpy array for Tip Quality
x_qual = np.arange(0, 11, 1)

#Defining the Numpy array for Triangular membership functions
qual_lo = sk.trimf(x_qual, [0, 0, 5])
```

Union

The union of two Fuzzy Sets A and B is a new Fuzzy Set, A ∪ B, also on X with a membership function defined as follows:

$$\mu_{(A\cup B)} = max\left(\mu_A(x), \mu_B(x)\right) = \mu_A(x) \wedge \mu_B(x)$$

∧ is called the *maximum operator*.

Here is the Python implementation:

```
import skfuzzy as sk
import numpy as np

#Defining the Numpy array for Tip Quality
x_qual = np.arange(0, 11, 1)

#Defining the Numpy array for two membership functions (Triangular)
qual_lo = sk.trimf(x_qual, [0, 0, 5])
qual_md = sk.trimf(x_qual, [0, 5, 10])

#Finding the Maximum (Fuzzy Or)
sk.fuzzy_or(x_qual,qual_lo,x_qual,qual_hi)
```

Intersection

An intersection of Fuzzy Sets A and B is a new Fuzzy Set, A ∩ B, also on X whose membership function is defined by:

$$\mu_{(A\cap B)} = min\left(\mu_A(x), \mu_B(x)\right) = \mu_A(x) \vee \mu_B(x)$$

is called the minimum operator.

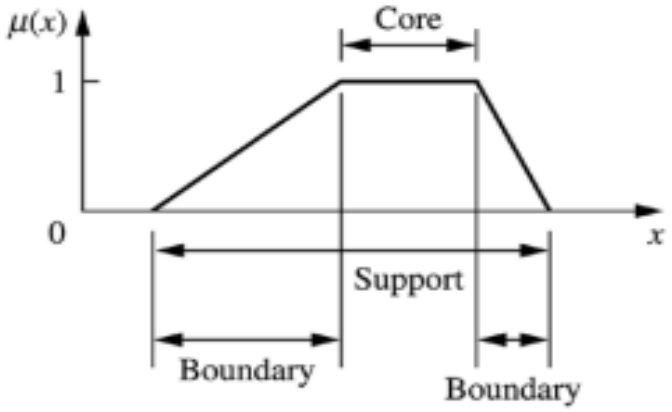
Here is the Python implementation:

```
import skfuzzy as sk
import numpy as np

#Defining the Numpy array for Tip Quality
x_qual = np.arange(0, 11, 1)

#Defining the Numpy array for two membership functions
(Triangular)
qual_lo = sk.trimf(x_qual, [0, 0, 5])
qual_md = sk.trimf(x_qual, [0, 5, 10])

#Finding the Minimum (Fuzzy AND)
sk.fuzzy_and(x_qual,qual_lo,x_qual,qual_hi)
```



Triangular Membership Function

Just as a triangle has three coordinates, a triangular membership function has three parameters: a, b, and c.

- a is the lower boundary
- b is the center
- c is the upper boundary

The following equation depicts the triangular membership function:

$$f(x;a,b,c)=\begin{cases} 0, & x\le a \\ \dfrac{x-a}{b-a}, & a\le x\le b \\ \dfrac{c-x}{c-b}, & b\le x\le c \\ 0, & c\le x \end{cases}$$

Alternatively, this can also be represented as follows:

$$f(x;a,b,c)=\max\left(\min\left(\frac{x-a}{b-a},\frac{c-x}{c-b}\right),o\right)$$

You can understand the triangular membership functions with the help of an example. This example uses the triangular membership function with a soccer example. Suppose the shooter can take four kinds of penalty shots:

- Full speed straight shot
- Medium powered curvy shot

- Slow straight shot
- Medium fast left shot

On average, the top speed at which a shooter takes a penalty kick is 80 mph. Therefore, there is no way you can say that this speed is slow. Hence you assign a 0% membership to 80mph. Similarly, a speed of 60 mph can be considered 70% fast and 30% medium. Likewise, you can assign different memberships to different speeds.

If you use a triangular membership function, it contains three limits: lower, full, and upper. The lower and upper bounds have a membership of 0% while the full value is 100%. The remaining values tread linearly. You can assign the following triangular membership functions to these categories:

- Full speed as [60, 80, 80]
- Medium powered as [40, 50, 70]
- Slow as [20, 20, 45]
- Medium fast as [50, 60, 80]

For example, if you defined the triangular membership function for "medium powered" as [40, 50, 70], the membership would be 0% at 40 mph, which linearly increases to 100% at 50 mph, and linearly decreases to 0% at 70 mph. The following Python code shows the execution of these triangular membership functions. Figure 2-6 shows the result.

```
#Importing Necessary Packages
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt
%matplotlib inline

#Defining the Fuzzy Range from a speed of 30 to 90
x = np.arange(30, 80, 0.1)
```

```
#Defining the triangular membership functions
slow = fuzz.trimf(x, [30, 30, 50])
medium = fuzz.trimf(x, [30, 50, 70])
medium_fast = fuzz.trimf(x, [50, 60, 80])
full_speed = fuzz.trimf(x, [60, 80, 80])

#Plotting the Membership Functions Defined
plt.figure()
plt.plot(x, full_speed, 'b', linewidth=1.5, label='Full Speed')
plt.plot(x, medium_fast, 'k', linewidth=1.5, label='Medium Fast')
plt.plot(x, medium, 'm', linewidth=1.5, label='Medium Powered')
plt.plot(x, slow, 'r', linewidth=1.5, label='Slow')
plt.title('Penalty Kick Fuzzy')
plt.ylabel('Membership')
plt.xlabel("Speed (Miles Per Hour)")
plt.legend(loc='center right', bbox_to_anchor=(1.25, 0.5),
ncol=1, fancybox=True, shadow=True)
```

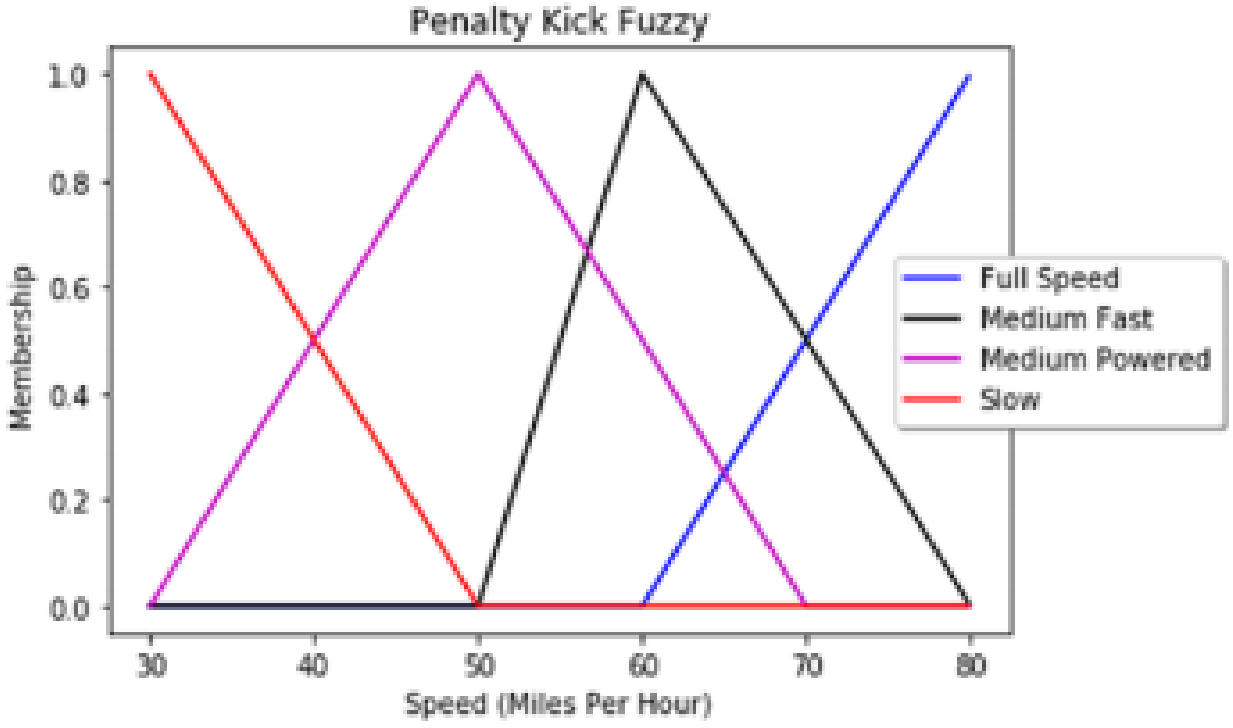


Figure 2-6. *Triangular membership of the soccer example*

To make a Fuzzy Inference System in Python, you have a library named FuzzyLite. To install this package on your system, execute the following command:

```
pip install pyfuzzylite
```

Fuzzylite is a free and open source Fuzzy Logic control library programmed in C++ for multiple platforms (e.g., Windows, Linux, Mac, and iOS). The goal of the FuzzyLite libraries is to easily design and efficiently operate Fuzzy Logic controllers following an object-oriented programming model, without relying on external libraries. For detailed exploration of this library, clone the GitHub page:

https://github.com/fuzzylite/pyfuzzylite.git

You can see the application of the Mamdani FIS in Python using this package:

```
import fuzzylite as fl
#Declaring and Initializing the Fuzzy Engine
engine = fl.Engine(
      name="SimpleDimmer",
       description="Simple Dimmer Fuzzy System which dims light
       based upon Light Conditions"
)
#Defining the Input Variables (Fuzzification)
engine.input_variables = [
       fl.InputVariable(
       name="Ambient",
       description="",
       enabled=True,
       minimum=0.000,
       maximum=1.000,
       lock_range=False,
       terms=[
       fl.Triangle("DARK", 0.000, 0.250, 0.500), #Triangular
       Membership Function defining "Dark"
       fl.Triangle("MEDIUM", 0.250, 0.500, 0.750), #Triangular
```

```
        Membership Function defining "Medium"
        fl.Triangle("BRIGHT", 0.500, 0.750, 1.000) #Triangular
        Membership Function defining "Bright"
        ]
        )
]
#Defining the Output Variables (Defuzzification)
engine.output_variables = [
        fl.OutputVariable(
        name="Power",
        description="",
        enabled=True,
        minimum=0.000,
        maximum=1.000,
        lock_range=False,
        aggregation=fl.Maximum(),
        defuzzifier=fl.Centroid(200),
        lock_previous=False,
        terms=[
        fl.Triangle("LOW", 0.000, 0.250, 0.500), #Triangular
        Membership Function defining "LOW Light"
        fl.Triangle("MEDIUM", 0.250, 0.500, 0.750), #Triangular
        Membership Function defining "MEDIUM light"
        fl.Triangle("HIGH", 0.500, 0.750, 1.000) #Triangular
        Membership Function defining "HIGH Light"
        ]
        )
]
#Creation of Fuzzy Rule Base
engine.rule_blocks = [
        fl.RuleBlock(
        name="",
        description="",
        enabled=True,
        conjunction=None,
        disjunction=None,
        implication=fl.Minimum(),
        activation=fl.General(),
```

```
        rules=[
        fl.Rule.create("if Ambient is DARK then Power is HIGH",
        engine),
        fl.Rule.create("if Ambient is MEDIUM then Power is
        MEDIUM", engine),
        fl.Rule.create("if Ambient is BRIGHT then Power is LOW",
        engine)
        ]
        )
]
```

You can see in this code that the Defuzzifier is called `Centroid`. `FuzzyLite` provides different kinds of Fuzzifiers, as listed here. All you need to do is replace them in the previous code:

- `fl.Centroid()`
- `fl.LargestOfMaximum()`
- `fl.MeanOfMaximum()`
- `fl.SmallestOfMaximum()`
- `fl.WeightedAverage()`
- `fl.Weighted Sum()`