PROYECTO 1 Sistemas Control Difuso problema de la propina

El "problema de la propina" se usa comúnmente para ilustrar el poder de los principios lógicos difusos para generar un comportamiento complejo a partir de un conjunto compacto e intuitivo de reglas expertas.

Si eres nuevo en el mundo de los sistemas de control difuso, es posible que quiera realizar una introducción de lógica difusa antes de leer este ejemplo funcional

El problema de las propinas

Creemos un sistema de control difuso que modele la forma en que usted decide dejar propina en un restaurante. A la hora de dejar la propina, se tiene en cuenta la calidad del servicio y de la comida valorada entre 0 y 10 puntos. A partir de ahí, se deja una propina entre el 0 y el 25%

Antecedentes (Entradas)

Servicio
- Universo (es decir, rango de valores nítidos): ¿Que tan bueno fue el servicio del personal de espera, en una escala de 0 a 10?
- Conjunto fuzzy (es decir, rango de valor difuso): pobre, aceptable, sorprendente

Calidad de la comida
- Universo: ¿Que tan sabrosa era la comida, en una escala de 0 a 10?
- Fuzzy set: malo, decente, genial

Consecuentes (Salidas)

Propina
- Universo: ¿Cuanto debemos dar de propina, en una escala del 0 al 25%?
- Fuzzy set: bajo, medio, alto

Reglas

If el servicio era bueno o la calidad de la comida era buena, **then** la propina será alta
If el servicio fue promedio, **then** la propina será media
If el servicio era pobre y la calidad de la comida era pobre, **then** la propina sera baja

Uso
si le digo a este controlador que califique el servicio como 9.8 y la calidad como 6.5

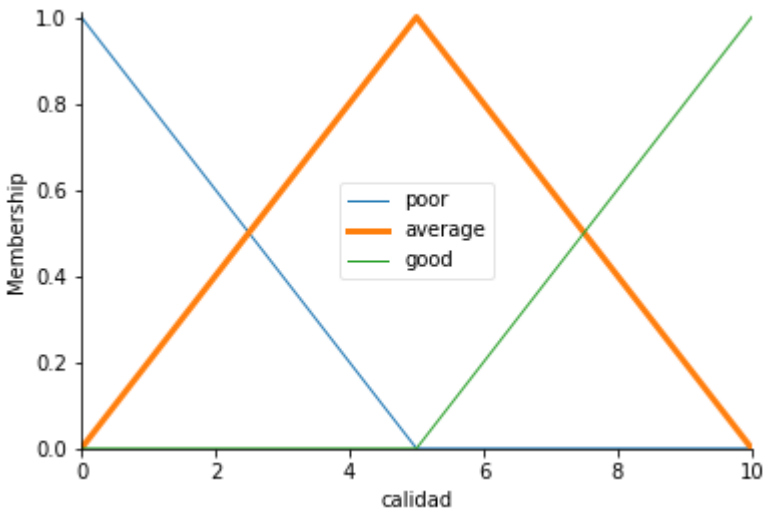
Recomendaría que dejara
Una propina de 20.2%

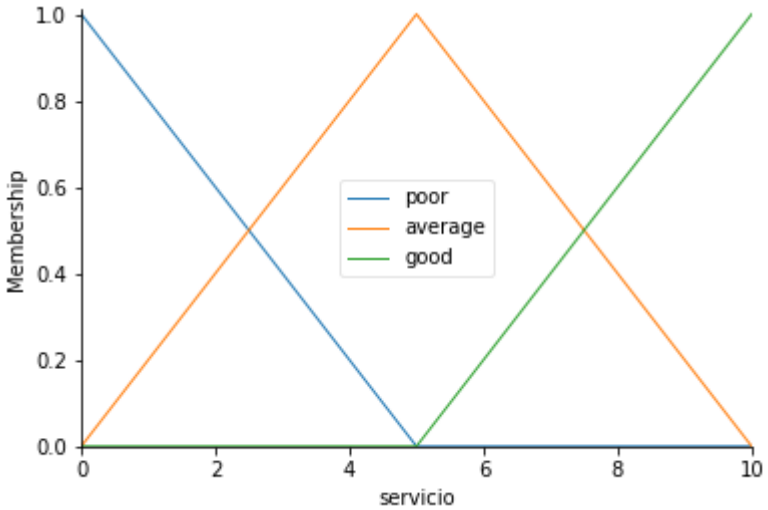
Definición de la forma de la función y captura de pantalla del sistema creado

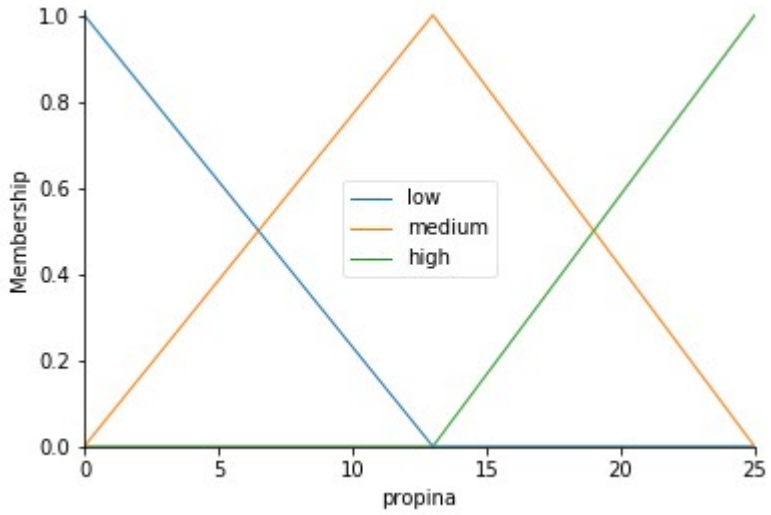
Como primera medida se consigna la captura de la pantalla con la definición y grafico triangular para después determinar como establecer las reglas, como notas adicionales recordar que el numero 13 se utiliza como un pico de inflexión entre las etiquetas low y medium, y entre medium y high, lo que quiere decir que hasta 13 es mas low y por encima es high [inicio, pico, final] trimf python. Recuerde que se utilizan etiquetas en ingles por que son nativas de las funciones membresía de la función o inclusive de la misma librería. Es la única razón por que los objetos y atributos obviamente se encuentran en español

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

# definicion de las variables difusas

# nuevos objetos antecedentes/ consecuentes variables hold y funciones de membresia

calidad = ctrl.Antecedent(np.arange(0, 11, 1), 'calidad')
servicio = ctrl.Antecedent(np.arange(0, 11, 1), 'servicio')
propina = ctrl.Consequent(np.arange(0, 26, 1), 'propina')

# la poblacion de la funcion de auto-membresia es posible con .automf(3,5,7)

calidad.automf(3)
servicio.automf(3)

# Las funciones de membresia personalizadas se pueden crear de forma interactiva con una API Python familar

propina['low'] = fuzz.trimf(propina.universe, [0, 0, 13])
propina['medium'] = fuzz.trimf(propina.universe, [0, 13, 25])
propina['high'] = fuzz.trimf(propina.universe, [13, 25, 25])

# ahora puede ver como se ven con .view()
calidad['average'].view()
servicio.view()
propina.view()
```

Las capturas correspondientes y la forma/ entendimiento del codigo descrito es:







Reglas difusas

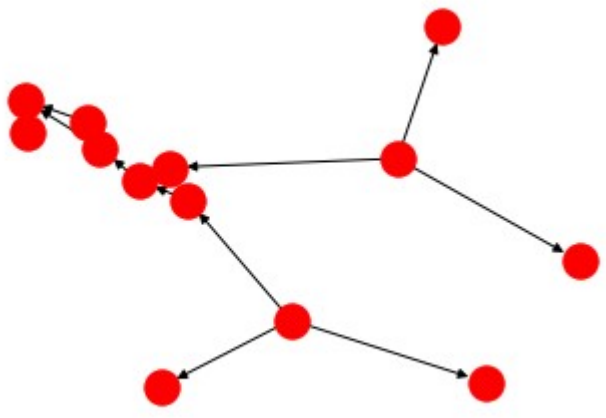
Ahora, para que estos triángulos sean útiles, definimos la relación difusa entre las variables de entrada y salida. Para los fines de nuestro ejemplo considere 3 reglas simples

1. If la comida es pobre OR el servicio es pobre, then la propina sera baja
2. If el servicio es promedio, then la propina sera media
3. If la comida es buena OR el servicio es bueno, then la propina sera alta

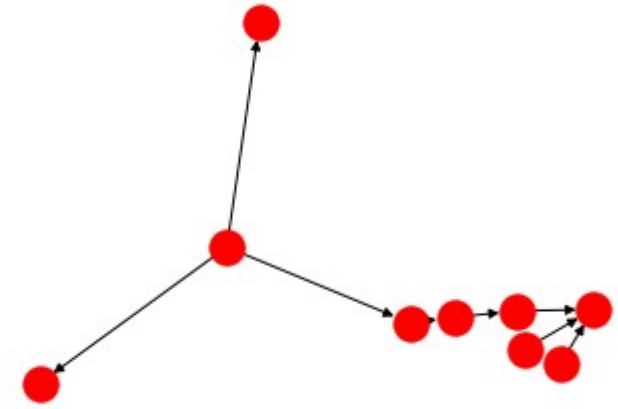
La mayoría de la gente estaría de acuerdo en estas reglas, pero las reglas son confusas. Convertir las reglas imprecisas en algo definido y procesable es todo un reto. Este es el tipo de tarea en el que se destaca la lógica difusa.

```
# grafico de las reglas que se consideran aceptadas por la "mayoria"
#If la comida es pobre OR el servicio es pobre, then la propina sera baja
#If el servicio es promedio, then la propina sera media
#If la comida es buena OR el servicio es bueno, then la propina sera alta

regla1 = ctrl.Rule(calidad['poor'] | servicio['poor'], propina['low'])
regla2 = ctrl.Rule(servicio['average'], propina['medium'])
regla3 = ctrl.Rule(servicio['good'] | calidad['good'], propina['high'])

regla1.view()
```

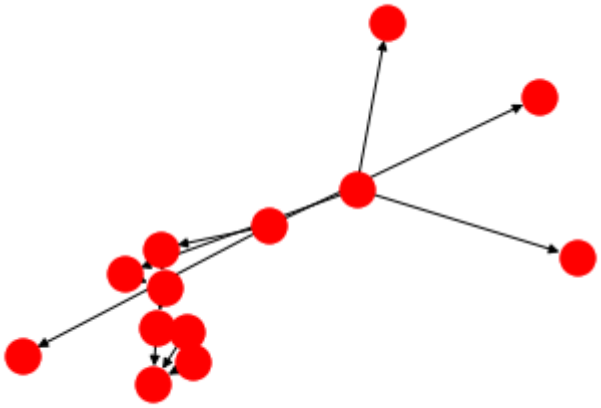
Da como consecuencia este grafico para la regla 1



Luego la regla 2



Luego la regla 3



Creación y simulación del sistema de control

Ahora que tenemos nuestras reglas definidas, simplemente podemos crear un sistema de control a través de:

Ahora que tenemos nuestras reglas definidas, simplemente podemos crear un sistema de control a través de:

propina_ctrl = ctrl.ControlSystem([regla1, regla2, regla3])

Para simular este sistema de control, crearemos un ControlSystemSimulation. Piense que este objeto representa nuestro controlador aplicado a un conjunto especifico de circunstancias. En el caso de la propina, podría ser la propina de "sharon en el bar de cerveza local". Crearíamos otro ControlSystemSimulation cuando aplicáramos nuestro tippin_ctrl (control de propinas) para "travis en la cafetería", por que las entradas serian diferentes

propin_a = ctrl.ControlSystemSimulation(propina_ctrl)

Ahora podemos simular nuestro sistema de control simplemente especificando las entradas y llamando al método de calculo. Supongamos que calificamos la calidad en 6.5 de 10 y el servicio como 9.8 de 10

Pase las entradas al sistema de control utilizando etiquetas antecedentes con API python

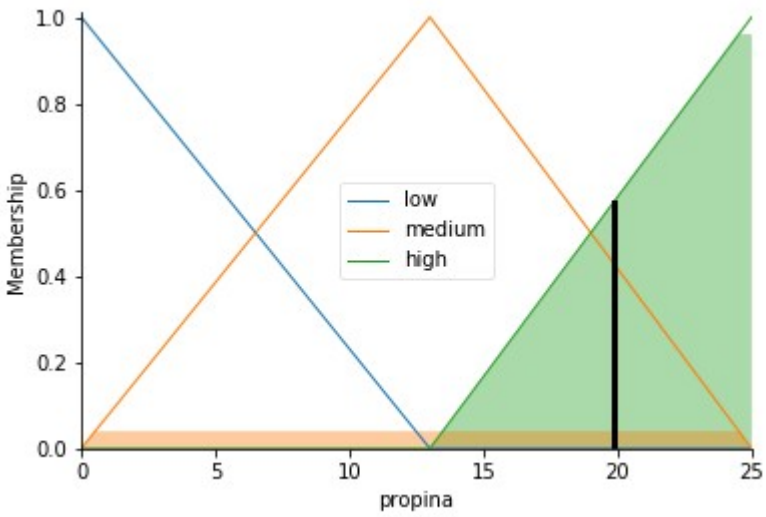
propin_a.input['calidad'] = 6.5
propin_a.input['servicio'] = 9.8

#poner y computar los numeros

propin_a.compute()

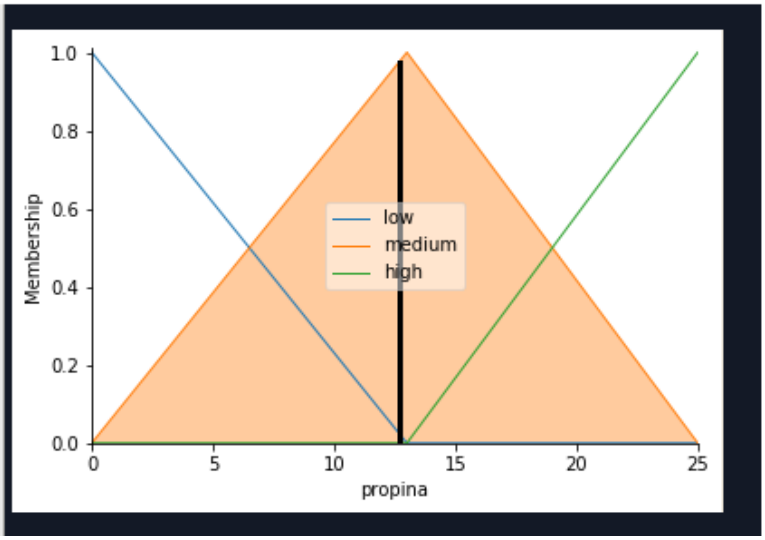
Una vez calculado el resultado podemos visualizarlo

print propin_a.output['propina']
propina.view(sim=propin_a)

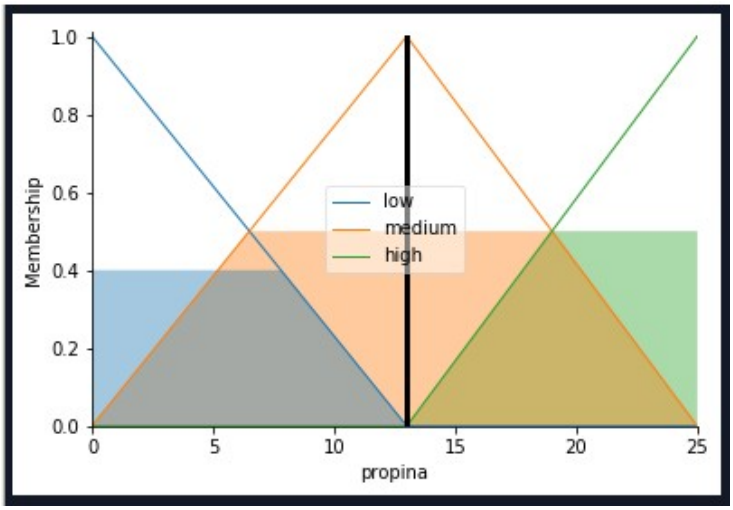


Ahora probaremos con otros resultados

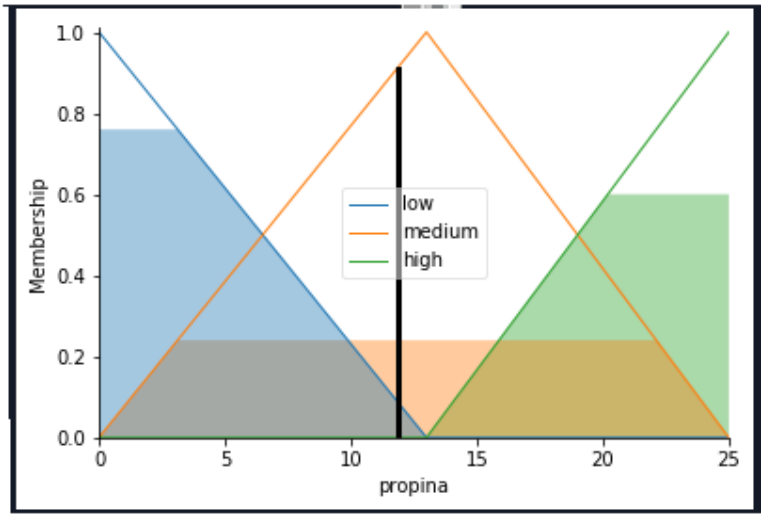
```
# Pasas todas las entradas al sistema
#tipping.inputs(input_data)
# Pase las entradas al sistema de cont

#propin_a.input['calidad'] = 6.5
#propin_a.input['servicio'] = 9.8

propin_a.input['calidad'] = 5
propin_a.input['servicio'] = 5



#poner y computar  los numeros
propin_a.compute()

print propin_a.output['propina']
propina.view(sim=propin_a)
```



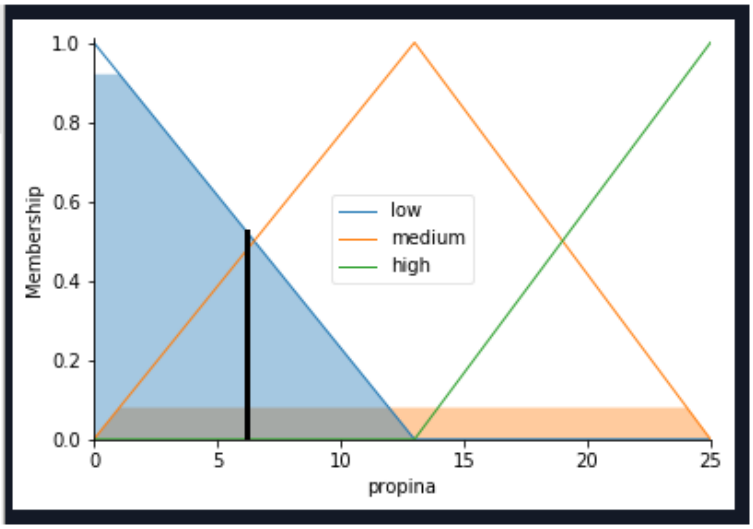
```
#tipping.inputs(input_data)
# Pase las entradas al sistema de cont

#propin_a.input['calidad'] = 6.5
#propin_a.input['servicio'] = 9.8

propin_a.input['calidad'] = 3
propin_a.input['servicio'] = 7.5



#poner y computar  los numeros
propin_a.compute()

print propin_a.output['propina']
propina.view(sim=propin_a)
```




```
#tipping.inputs(input_data)
# Pase las entradas al sistema de cont

#propin_a.input['calidad'] = 6.5
#propin_a.input['servicio'] = 9.8

propin_a.input['calidad'] = 8
propin_a.input['servicio'] = 1.2



#poner y computar  los numeros
propin_a.compute()

print propin_a.output['propina']
propina.view(sim=propin_a)
```




```
#tipping.inputs(input_data)
# Pase las entradas al sistema de contr

#propin_a.input['calidad'] = 6.5
#propin_a.input['servicio'] = 9.8

propin_a.input['calidad'] = 1.1
propin_a.input['servicio'] = 0.4



#poner y computar  los numeros
propin_a.compute()

print propin_a.output['propina']
propina.view(sim=propin_a)
```



Pensamientos (autor del blog)

El poder de los sistemas difusos esta permitiendo un comportamiento complicado e intuitivo basado en un sistema escaso de reglas con una sobre carga mínima. Tenga en cuenta que nuestros universos de función de membresía eran gruesos, solo definidos en los enteros, pero Fuzz.Inerp_Membership permitio que la resolución efectiva aumente la demanda. Este sistema puede responder a cambios arbitrariamente pequeños en las entradas, y la carga de procesamiento es mínima

Pensamientos propios

Triángulos: Cada triángulo representa un conjunto difuso diferente. La altura del triángulo en cualquier punto a lo largo del eje X indica el grado de membresía de esa entrada al conjunto difuso.
Área sombreada: Esta área bajo el/los triángulos podrían estar indicando un área específica de interés o actividad.

Ahora queda implementar el anterior ejemplo de forma manual existe un recurso en Matlab que explica la fuzzy logic video, esto para complementar lo aquí descrito no descuidar la complejidad real de los sistemas difusos (Reznik)