Sistema de Inferencia de Lógica Difusa GFIS

Ariel Coto Santiesteban C412

May 17, 2019

1 Caractrísticas del Sistema

GFIS permite resolver cualquier problema de inferencia si desea implementarse un sistema de soluciones basado en Lógica Difusa, en *Python 3*. Para esto, se le ofrece al usuario dos clases fundamentales, **MemberFunct** y **DiffSystem**.

1.1 Funciones de Membresía

Un paso crucial para crear un sistema de soluciones basado en lógica difusa, es definirse los parámetros de entrada al sistema y todas las posibles clasificaciones de cada entrada. Por ejemplo, como parámetro de entrada, la *Temperatura* en grados celcius, y clasificaciones: *Alta*, *Baja* y *Media*. En **GFIS** para poder definirse estas funciones de membrecía, se utiliza la clase **MemberFunct**. Una limitación es que solo se pueden crear funciones trapezoidales y triangulares. Como entrada se le debe aclarar los nombres de a cuál parámetro define (*Temperatura* o *Presión*, u otra cosa), y el nombre de la función de membresía a la cual va a describir sobre ese parámetro (*Alta*, *Baja*, etc). Además, debe seleccionar el tipo de función que desea (trapezoidal o triangular) y los puntos que describen a esta. Se le puede pasar, por último, la función de membresía directamente, la cual debe recibir un entero y devolver otro, pero se recomienda pasarle *None* a esta entrada.

1.2 Sistema y operaciones

Luego de creadas las funciones de membresía, se pasa a configurar al sistema que incorpora todas las funciones. Para ello se ofrece la clase **DiffSystem**. A la hora de instanciar el sistema, se debe definir como entrada un diccionario que mapee de la forma *(<Nombre_Campo>,<Nombre_F_Mebrecía>): <función_de_membrecía>*. Ejemplo: *('Temperatura','Alta'): <objeto-función>*. Seguidamente, se pasa a definir las reglas del sistema.
GFIS le ofrece, en la clase **DiffSystem**, las operaciones *fuzzy_and*, *fuzzy_or* y *fuzzy_not*. Para obtener una secuencia de operaciones, el usuario debe componer, según el orden operacional que desee, estas funciones. Una entrada simple a una operación se define con un string con la forma: *<Nombre_del_campo><espacio><Nombre_f_membrecía>*, llamémosle *entrada simple*. Ejemplo: *Temperatura Alta*. Con las operaciones *and*, *or* y *not*, se crea el cuerpo del *if* de la regla. Para añadir la regla al sistema se utiliza la función *add_rule* de la clase **DiffSystem**, que recibe como primer

parámetro el cuerpo del *if* y como segundo parámetro un string como la *entrada simple*, que representa el cuerpo del *then* de la regla.

1.3 La salida

Para obetener la salida del sistema, se hace a través de la función *get_output* de la clase **DiffSystem**. Esta recibe como parámetros un diccionario que mapea de nombre de los campos medidos a medición, el nombre de la función de inferencia a utilizar, el nombre de la función de desifusificación, y si desea plotear la función resultante. La salida es un entero con el valor de la inferencia, y una lista con los resultados de las evaluaciones de las reglas. Los métodos de inferencia disponibles son: *mamdani* y *larsen*. Los métodos de desifusificación disponibles son: *lom*, *som*, *mom*, *centroid* y *bisec*.

2 Principales ideas sobre la implementación

2.1 Clase MemberFunct

Esta clase es la interfaz que se ofrece para crear las funciones de membresía de las características del sistema. La clase posee una función *evaluate*, que será la que devuelva la evaluación sobre la función. Inicialmente tiene la forma:

$$\max(\min(\frac{(x-a)}{(b-a)}, \frac{(c-x)}{(c-b)}), 0), \tag{1}$$

en el caso de ser triangular con puntos en a, b y c; y si es trapezoidal

$$f(x) = \begin{cases} \max(\min(\frac{(x-a)}{(b-a)}, 1, \frac{(d-x)}{(d-c)}), 0) & si \quad b-a \neq 0, \\ \\ \max(\min(1, \frac{(d-x)}{(d-c)}), 0) & si \quad b-a = 0, \end{cases} \tag{2}$$

puesto que d y c se asume que siempre serán distintos de cero. Esta clase tiene redefinido el método *__add__* para mezclar dos funciones de membresía, cuya implementación basicamente es devolver un nuevo objeto cuya función *evaluate* es

$$lambda\ x : \max(member\_function1.evaluate(x), member\_function2.evaluate(x)). \tag{3}$$

2.2 Clase DiffSystem

Esta es la clase fundamental que se le ofrece al usuario para encajar todas las piezas del sistema. Todas las funciones de membresía se pasan en la inicialización de la clase. Simulando el patrón Factory, para acceder a una función de membresía se hace a través de la función *get_member_func*, la cual devuelve una nueva instancia de esa función. Su función principal es *get_output* que devuelve el resultado de inferencia del sistema junto a los resultados de evaluar cada regla. Esta función lo que hace basicamente es:

1. Guardar el *input_data* y resolver las funciones de inferencia y desifusificación.

2. Evaluar las reglas del sistema. Agrupar las que tengan igual consecuente y se escoge la de mayor valor de evaluación para cada consecuente distinto.

3. Se aplica la función de inferencia a cada regla, y se agregan todas las funciones.

4. A la resultante se le aplica el método de desifusificación escogido.

5. Se devuelve el resultado.

2.3 Las reglas

Para las operaciones *and*, *or* y *not* sobre las funciones de membresía, es obligado utilizar las funciones *fuzzy_and*, *fuzzy_or* y *fuzzy_not* de la clase **DiffSystem**. Estas son funciones que capturan los parámetros de entrada y devuelven otra función que será la que evalúe la regla. La evaluación se resuelve en profundidad.

2.4 Código

Las clases que se ofrecen se encuentran en el archivo *fuzzySys.py*. Si desea utilizarlas debe importarlas de allí. Además, se ofrece un ejemplo de cómo utilizar el sistema en el archivo *example.py*.

3 Propuesta de Problema a solucionar

Dado un televisor que detecta si hay ruido en el ambiente y el volumen de la señal, determina si subir o bajar el volumen. (Se desconocen las métricas reales del problema). Se desea saber en cada momento en qué estado dejar el volumen del televisor. Supongamos que el Ruido tiene 3 categorías:

1. Bajo (trapezoidal que comienza con (0,1), (50,1), (60,0))

2. Medio (trapezoidal que comienza con (50,0),(60,1),(80,1),(90,0))

3. Alto (trapezoidal que comienza con (80,0), (90,1), (∞,1)),

(ver figura 1), y que el volumen de la señal tenga como categorías

1. Bajo (trapezoidal que comienza con (0,1), (50,1), (60,0))

2. Medio (triangular que comienza con (50,0),(70,1),(90,0))

3. Alto (trapezoidal que comienza con (70,0), (90,1), (∞,1))

(ver figura 2).

El volumen del televisor tiene como categorías:

1. Bajo (trapezoidal que comienza con (0,1), (5,1), (10,0))

2. Medio (triangular que comienza con (5,0),(10,1),(15,0))

(ver figura 3).

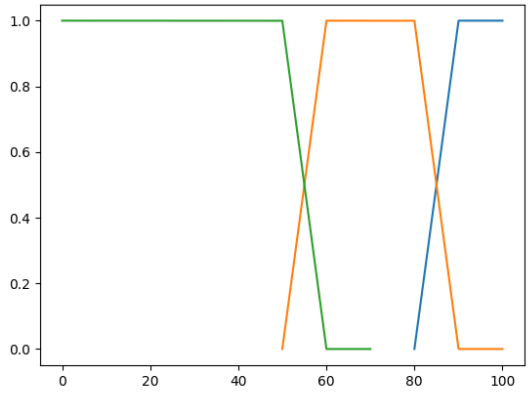

Figure 1: Categorías del Ruido: Bajo(verde), Medio(Naranja) y Alto(azul).

3.1 Reglas del problema

Se propuso al problema el siguiente conjunto de reglas:

- **IF** Ruido Alto **OR** VolumenDeLaSeñal Alto **THEN** VolumenDeLaTV Bajo
- **IF** Ruido Alto **AND** (**NOT VolumenDeLaSeñal Alto**) **THEN** VolumenDeLaTV Medio
- **IF** Ruido Medio **AND** (**NOT VolumenDeLaSeñal Alto**) **THEN** VolumenDeLaTV Medio
- **IF** Ruido Bajo **AND** VolumenDeLaSeñal Medio **THEN** VolumenDeLaTV Bajo
- **IF** Ruido Bajo **AND** VolumenDeLaSeñal Bajo **THEN** VolumenDeLaTV Medio

3.2 Resultados del problema

Este problema está modelado en el archivo *example.py*. Si desea ejecutarlo corra el comando *python3 example.py*. Para validar el sistema se probó como entrada $Ruido = 87$ y $VolumenDeLaSig = 80$, *mandani* como método de inferencia y *mom* de desifusificador. Se obtuvo como resultado 7.5 y como función resultante la figura 4. Se interpreta que con un 87 de ruido, que se considera alto, y un 80 de volumen de la señal, que también se considera alto, se debe poner el volumen de la TV a 8, redondeando por exceso, dado que la unidad de volumen de un televisor es discreto.

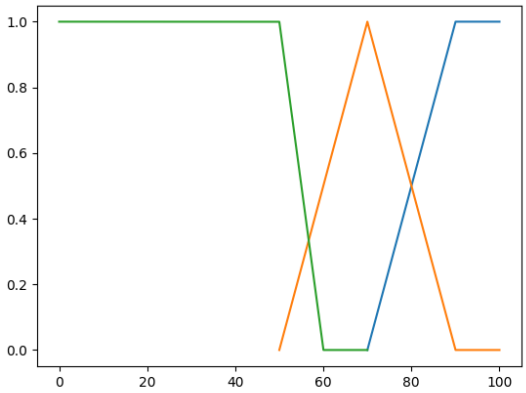

Figure 2: Categorías del Volumen de la Señal: Bajo(verde), Medio(Naranja) y Alto(azul).

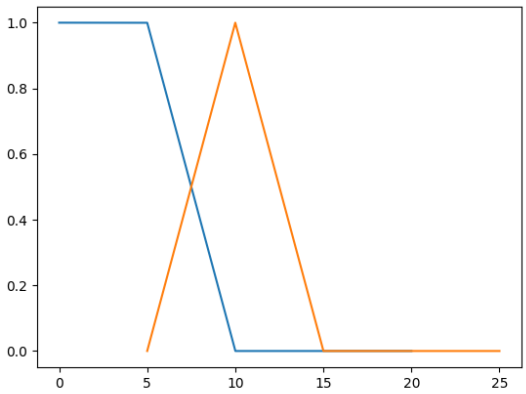

Figure 3: Categorías del Volumen de la TV: Bajo(azul) y Medio(Naranja).

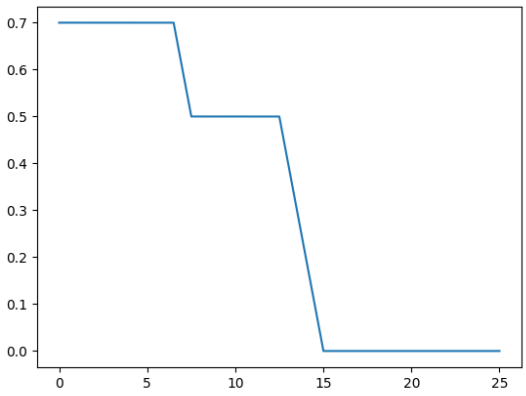

Figure 4: Función resultante con parámetros 87 y 80 de ruido y volumen de la señal, respectivamente, usando *mamdani* y *mom*. Se obtuvo como resultado 7.5.