Práctica 4

Diseño de un control difuso

Cálculo de propinas en un restaurante

> Caso práctico

Un problema típico en un restaurante es saber cuánta propina deberíamos dejar al camarero/a. En esta práctica daremos solución a este problema a través del diseño de un control difuso que se apoye en lógica difusa. Usaremos para ello las librerías de Python **Scikit-fuzzy**[1] y **Numpy**.

A lo largo de la práctica revisaremos todos los pasos que comprende el diseño de un sistema de control difuso.

Antes de empezar, prepararemos el entorno de trabajo. Abre para ello un cuaderno en Google Colab o Jupyter, y empieza importando las siguiente librerías.

```
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt
```

Si la importación de la librería `skfuzzy` genera un error, importa el paquete de Python correspondiente la siguiente forma:

```
!pip install -U scikit-fuzzy
```

[1] En lugar de la librería estándar scikit-fuzzy podríamos usar igualmente "fuzzy.py", tal y como hemos hecho en clase.

> Procedimiento para el diseño del control difuso

Tal y como hemos visto en las sesiones de clase, el diseño de un control difuso empieza con la definición de determinadas variables lingüísticas como paso previo a plantear el método de inferencia que se usará para extraer conclusiones.

Vamos paso a paso …

1. Variables, universo de discurso y valores lingüísticos

En los ejemplos expuestos en clase (por ejemplo, el del mecanismo de una impresora) hemos manejado dos funciones dentro del sistema de control difuso: una de entrada (en aquel ejemplo, el voltaje del motor) y otra de salida (la nueva posición del cabezal). En este caso vamos a usar 3 variables: la **calidad de la comida** y la **calidad del servicio**, por un lado, que actuarán como funciones de **entrada**, y el **porcentaje de propina**, por otro lado, que será lo que calcule nuestro control difuso actuando como función de **salida**.

> Variables y universo de discurso asociado:

- **entrada**
  - **calidad del servicio** = [0, 10] # sería la puntuación otorgada
  - **calidad de la comida** = [0, 10] # sería la puntuación otorgada
- **salida**
  - **porcentaje de la propina** = [5, 25] # entre el 5% y el 25%

A continuación, introduce las siguientes líneas para dar forma a lo anterior:

```
# Rango de la calidad de la comida
x_cal = np.arange(0, 11, 1)

# Rango de la calidad del servicio
x_serv = np.arange(0, 11, 1)

# Rango del porcentaje de propina (del 5% al 25%)
```

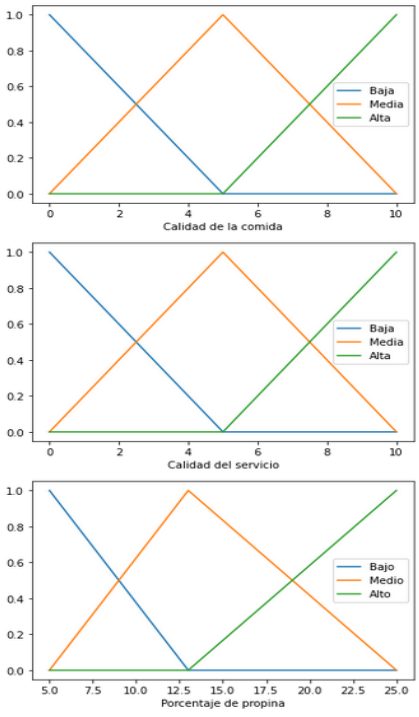
```
x_prop = np.arange(5, 26, 1)
```

Además, podemos definir los diferentes estados asociados a cada variable:

> Valores lingüísticos (estados):

- **calidad del servicio** = [bajo, medio, alto]
- **calidad de la comida** = [bajo, medio, alto]
- **porcentaje de la propina** = [bajo, medio, alto]

Los valores lingüísticos pueden ser modelizados mediante una **función de membresía** de tipo **triangular**.



> Ejercicio 1

Codifica en Python las 3 funciones anteriores, para lo cual puedes usar el método "trimf" del objeto "fuzz" anteriormente declarado. Además, representa las funciones por pantalla.

2. Diseño del controlador difuso

A continuación, calcularemos el grado de membresía que ciertos valores de entrada tienen para los valores lingüísticos declarados. El siguiente código realiza esa tarea para la variable **calidad de comida** apoyándose en el método *interp_membership* del objeto *fuzz*, generando la gráfica correspondiente.

Las dos primeras líneas definen dos valores de prueba (es decir, el x_0) para las variables **calidad de comida** y **calidad del servicio**. Tales valores funcionarán como los datos de entrada de nuestro sistema de control difuso. Revisa el código línea a línea y asegúrate de que entiendes cómo funciona.

```
# Introducimos valores para las variables de entrada
cal_comida = 8
cal_servicio = 7

# Calculamos el grado de membresía de cierto valor de entrada para la
variable 'calidad de comida'

cal_nivel_b = fuzz.interp_membership(x_cal, cal_b, cal_comida)
cal_nivel_m = fuzz.interp_membership(x_cal, cal_m, cal_comida)
cal_nivel_a = fuzz.interp_membership(x_cal, cal_a, cal_comida)

print("Calidad comida baja:", cal_nivel_b)
print("Calidad comida media:", cal_nivel_m)
print("Calidad comida alta:", cal_nivel_a)

plt.plot(x_cal, cal_b, label="Baja")
plt.plot(x_cal, cal_m, label="Media")
plt.plot(x_cal, cal_a, label="Alta")
plt.plot([cal_comida, cal_comida], [0.0, 1.0], linestyle="--")
plt.plot(cal_comida, cal_nivel_b, 'x')
plt.plot(cal_comida, cal_nivel_m, 'x')
plt.plot(cal_comida, cal_nivel_a, 'x')
plt.legend(loc='best')
plt.xlabel('x = Calidad de la comida')
plt.ylabel('$\mu (x)$')
plt.show()
```

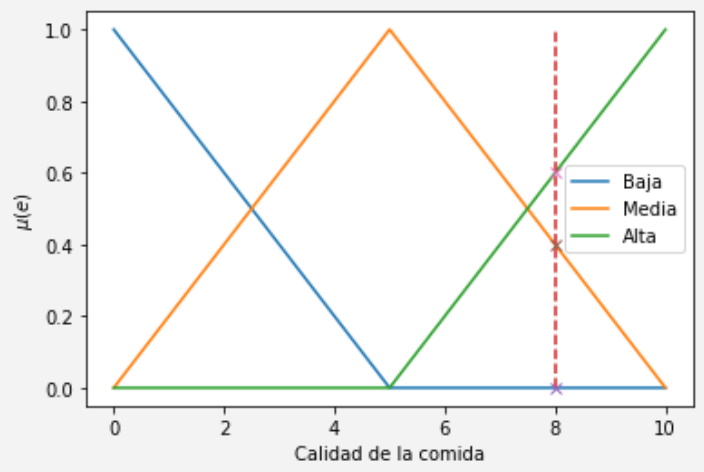
```
# Calculamos el grado de membresía de cierto valor de entrada para la
variable 'calidad del servicio'

...
```

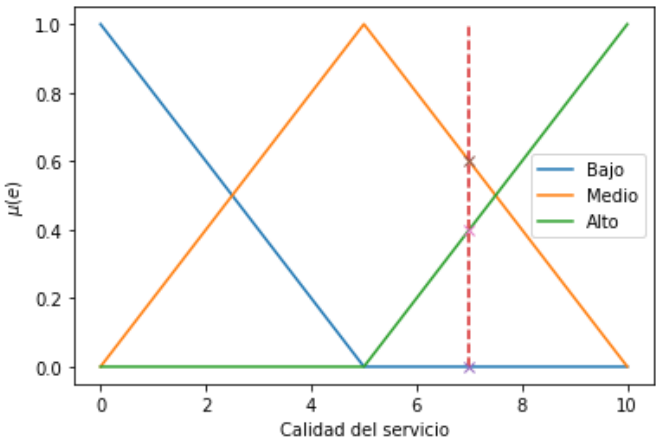
> Ejercicio 2

Completa el código que falta para la variable **calidad del servicio.**

Como se puede comprobar en las gráficas resultantes, para un valor de entrada para la variable "calidad de la comida" de por ejemplo **8,** los grados de pertenencia a los conjuntos "baja", "media" y "alta" son **0, 0'4 y 0'6,** respectivamente, lo cual parece razonable.



De forma similar, para una "calidad de servicio" de **7,** los grados de pertenencia a los conjuntos "baja", "media" y "alta" son **0, 0'6 y 0'4,** respectivamente.



Definición de las reglas

El siguiente paso comporta definir las reglas que integran el **método de inferencia del control difuso.**

Dado que, a diferencia del ejemplo expuesto en clase en relación al caso de la impresora, aquí contamos ahora con dos variables de entrada en lugar de una (allí era únicamente la posición del cabezal), nos encontramos con múltiples combinaciones de las mismas que cabría asociar a diferentes niveles de propina según nos interesara, siendo todo ello parte del sistema difuso que estamos construyendo.

Dichas combinaciones podrían expresarse en una tabla como la que sigue, en la que asociamos un valor lingüístico (bajo, medio o alto) de propina otorgada resultado de combinar los posibles valores de las variables comida y servicio. De esta forma, podríamos convenir, por ejemplo, que si la calidad del servicio es baja, pero la del servicio es en cambio alta, se tendría que ofrecer un nivel de propina medio (¡en eso consiste precisamente el razonamiento aproximado!).

NIVEL DE PROPINA OFRECIDO		CALIDAD DEL SERVICIO		
		Baja	Media	Alta
CALIDAD DE LA COMIDA	Baja	BAJO	BAJO	MEDIO
	Media	MEDIO	MEDIO	ALTO
	Alta	ALTO	ALTO	ALTO

Sin embargo, en lugar de basarnos en todas esas combinaciones, y a fin de simplificar nuestra solución, nuestro sistema se regirá únicamente por 3 de esas reglas con las que se determinará de forma resumida el nivel de propina a dar.

Estas 3 reglas pueden ser enunciadas en lenguaje cotidiano de la siguiente forma:

> *REGLA 1: "Si la calidad de la comida es mala (es decir, "baja") o la de calidad del servicio es baja entonces daremos poca propina (es decir, "baja") "*
>
> *REGLA 2: "Si la calidad del servicio es aceptable (es decir, "media") entonces daremos una propina razonable (es decir, "media") "*
>
> *REGLA 3: "Si la calidad de la comida es alta o la calidad del servicio es alta entonces daremos mucha propina (es decir, "alta") "*

Como hemos visto a nivel teórico, es posible reducir el método de inferencia a un conjunto de operaciones de cálculos de máximos y mínimos que, como sabemos, tienen su equivalente en el cálculo de productos cartesianos y operaciones de composición.

Por ejemplo, el siguiente código corresponde a la aplicación de la **regla 1** siguiendo la operación ***modus ponens***.

```
# REGLA 1: "Si la calidad de la comida es baja o la de calidad del
servicio es bajo entonces la propina es baja"

# La regla 1 tiene como antecedente "Si la calidad de la comida está baja
# o la de calidad del servicio es bajo"

antecedente_1 = np.fmax(cal_nivel_b, serv_nivel_b)
regla_1 = np.fmin(antecedente_1, prop_b)
...
```

Como se puede observar, primero se realiza el cálculo de unión entre las variables que forman parte del antecedente de la regla en cuestión, y después se busca la intersección de antecedente (`"Si la calidad de la comida está baja o la de calidad del servicio es bajo"`) y consecuente (`"entonces la propina es baja"`), usando para ello las funciones fmax y fmin,

respectivamente, incluidas en la librería *numpy*:

> Ejercicio 3

¿Cómo podríamos representar las reglas 2 y 3? Introduce el código en tu cuaderno para las reglas 2 y 3 a continuación del código anterior.

3. Fusificación

Para completar la fase de **fusificación** iniciada en el apartado previo necesitamos generar la unión de las funciones resultantes de aplicar las reglas teniendo en cuenta los "valores de corte" representados por las variables `cal_comida` y `cal_servicio`.

Añade la siguiente línea a fin de completar la operación de "modus ponens", en la que se calcula la función resultante de combinar, en dos pasos, las 3 funciones correspondiente a las variables en juego:

```
# Se hace la unión de los conjuntos resultantes de la aplicación de reglas
union = np.fmax(regla_1, np.fmax(regla_2, regla_3))
```

Imprime por pantalla el contenido de la variable "union". **¿Qué representan los valores mostrados?**

4. Defusificación

Finalmente nos queda acometer la fase de **defusificación**. Con la defusificación se pretende calcular un valor de salida del sistema difuso, que en nuestro caso corresponderá a cierto porcentaje de propina. Como hemos visto en clase, existen diferentes métodos para obtener ese valor de salida.

> Ejercicio 4

Utiliza el método "**defuzz**" del objeto "fuzz" a fin de aplicar la defusificación mediante el método del "centroide" y genera la línea correspondiente en tu cuaderno que imprima el valor de la propina resultante por pantalla. Si lo necesitas, además del cuaderno con los ejemplos usados en clase, puedes consultar la sintaxis del método "defuzz" en la página web de "scikit-fuzz".