FUZZY LOGIC

We must exploit our tolerance for imprecision.
Lotfi Zadeh, 1973
Professor, Systems Engineering, UC- Berkeley

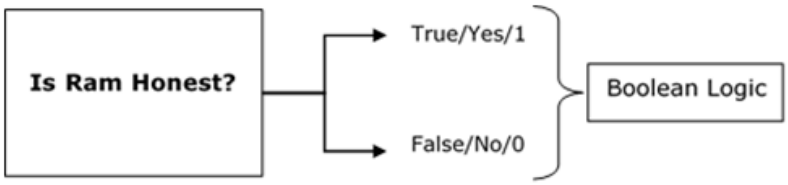
LA lógica difusa puede ser descrita como un sistema interpretativo, en el cual los objetos o elementos son relacionados con conjuntos de fronteras no nítidamente definidas, otorgándoles un grado de pertenecía relativa o graduada y no estricta como es de costumbre en la lógica tradicional.

¿Cuál es el beneficio de la lógica difusa?

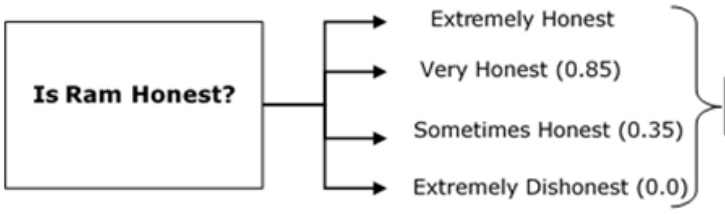
La lógica difusa ayuda a resolver un problema en particular después de considerar todos los datos disponibles y luego tomar la decisión adecuada.
El método de lógica difusa emula la forma humana de toma de decisiones, que considera todas las posibilidades entre valores digitales de Verdadero y Falso.

¿Se utiliza la lógica difusa en el aprendizaje automático?

Si, La lógica tradicional y clásica suele clasificar la información en patrones binarios como: sí/no, verdadero/falso o día/noche. En cambio, la lógica difusa se enfoca en caracterizar el espacio entre estos escenarios en blanco o negro.



En este caso en particular tenemos
un resultado concreto o es Falso o es Verdadero



Si hablamos de fuzzy logic podemos ver que hay
un ponderado a diferentes valores
con caracteristicas en diferentes grados.

- **PROCEDIMIENTO:**

En este ejemplo vamos a predecir los clientes en una heladería dada la temperatura de la ciudad.
Este ejemplo esta basado en toma de decisiones y procesamiento de Señales.

El sistema de lógica difusa consiste:
if premisa then conclusion

Considere una heladería, con las siguientes premisas:

- **Si la temperatura es calurosa entonces la heladería se llena**
- **Si la temperatura es moderada entonces la heladería esta ocupada.**
- **Si la temperatura es fría entonces la heladería esta tranquila.**

PRIMERO SE CREAN FUNCIONES DE PERTENENCIA DIFUSAS QUE REPRESENTEN EL GRADO DE VERDAD RELACIONADO CON UNA VARIABLE VARIABLE CONTINUA.

EN SCIKIT-FUZZY, TODAS LAS FUNCIONES DE PERTENENCIA SON ARREGLOS NUMPY SIN PROCESAR. ESTO SIGNIFICA QUE EN MUCHAS FUNCIONES DEBE ESPECIFICAR MANUALMENTE TANTO LA VARIABLE DEL UNIVERSO (AQUÍ TEMP) COMO LA FUNCIÓN DE PERTENENCIA. LOS ARREGLOS DE VARIABLES DE UNIVERSO Y FUNCIONES DE PERTENENCIA DEBEN TENER UNA FORMA IDÉNTICA.

```
# FUNCIONES UNIVERSO PARA TEMPERATURA Y CLIENTES
temp = np.arange(30, 101, 1)
customers = np.arange(0, 36, 1)

# funciones de pertenencia para temperatura
t_hot = fuzz.trimf(temp, [65, 100, 100])
t_moderate = fuzz.trimf(temp, [30, 65, 100])
t_cool = fuzz.trapmf(temp, [20, 20, 30, 65])

# funciones de pertenencia de clientes en el local
c_crowded = fuzz.trimf(customers, [24, 35, 35])
c_busy = fuzz.trimf(customers, [0, 24, 35])
c_quiet = fuzz.trimf(customers, [0, 0, 24])



# visualizar funciones de temperatura por calor
fig, ax = plt.subplots()

ax.plot(temp, t_hot, 'r', temp, t_moderate, 'm', temp, t_cool, 'b')
ax.set_ylabel('Fuzzy ')
ax.set_xlabel('Temperatura ( En Farenheit)')
ax.set_ylim(-0.05, 1.05);

# visualizar las funciones graficadas
fig, ax = plt.subplots()

ax.plot(customers, c_quiet, 'c', customers, c_busy, 'm', customers, c_crowded, 'ForestGreen')
ax.set_ylabel('Fuzzy ')
ax.set_xlabel('Nùmero de clientes')
ax.set_ylim(-0.05, 1.05);


# Relacion Fuzzy
R1 = fuzz.relation_product(t_hot, c_crowded)
R2 = fuzz.relation_product(t_moderate, c_busy)
R3 = fuzz.relation_product(t_cool, c_quiet)
```

A CONTINUACIÓN, USAMOS LA IMPLICACIÓN DIFUSA A TRAVÉS DE LA RELACIÓN PRODUCTO (LARSEN), FUZZ.RELATION_PRODUCT. ESTO SE HACE POR SEPARADO PARA CADA DECLARACIÓN EN EL SISTEMA DIFUSO, POR LO QUE TENDREMOS 3 MATRICES DE RELACIONES DIFUSAS SEPARADAS.

NOTA: SCIKIT-FUZZY INCLUYE UNA SEGUNDA RELACIÓN DE IMPLICACIÓN DIFUSA, RELACIÓN MIN (MAMDANI), COMO FUZZ.RELATION_MIN

LARSEN METHOD

Case 1: Inputs are crisp and treated as fuzzy singletons.

$u = u_0,\ v = v_0$

$$\mu_{C_i'}(w) = [\mu_{A_i}(u_0)\ and\ \mu_{B_i}(v_0)] \rightarrow \mu_{C_i}(w)$$

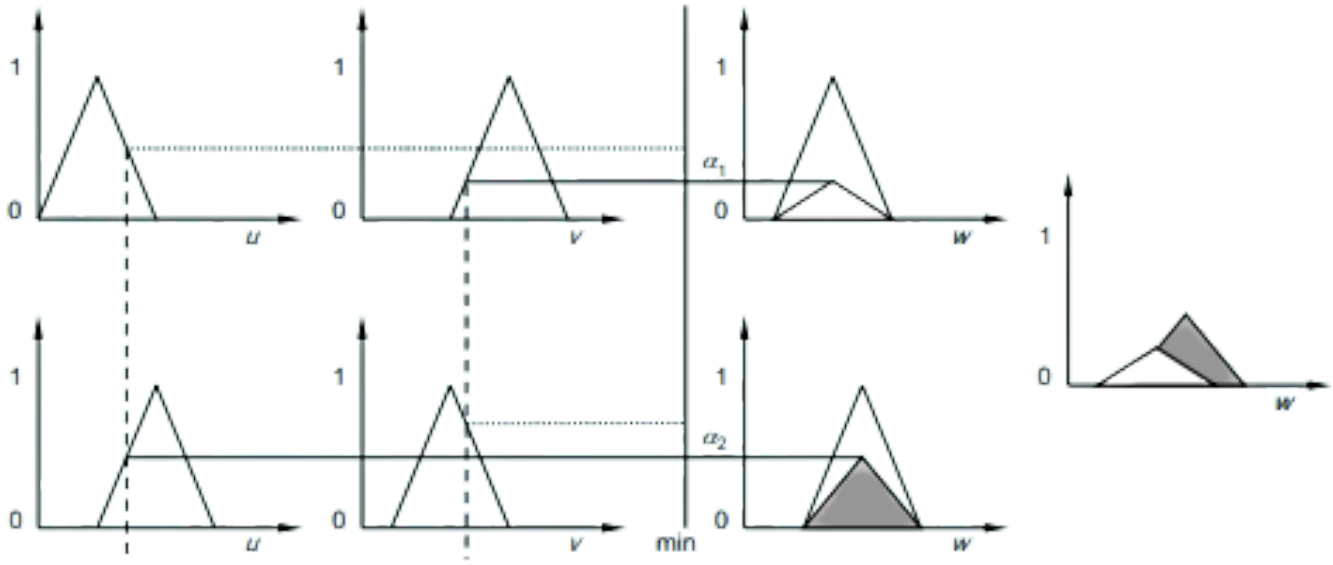
Inference result — if … — then …

$$= [\mu_{A_i}(u_0) \wedge \mu_{B_i}(v_0)] \bullet \mu_{C_i}(w)$$

$$= \alpha_i \bullet \mu_{C_i}(w) \quad where\ \alpha_i = \mu_{A_i}(u_0) \wedge \mu_{B_i}(v_0)$$

For multiple rules: $\mu_{C'}(w) = \overset{n}{\underset{i=1}{V}}[\alpha_i \bullet \mu_{C_i}(w)] = \overset{n}{\underset{i=1}{V}} \mu_{C_i'}(w)$

$$C' = \overset{n}{\underset{i=1}{U}} C_i'$$



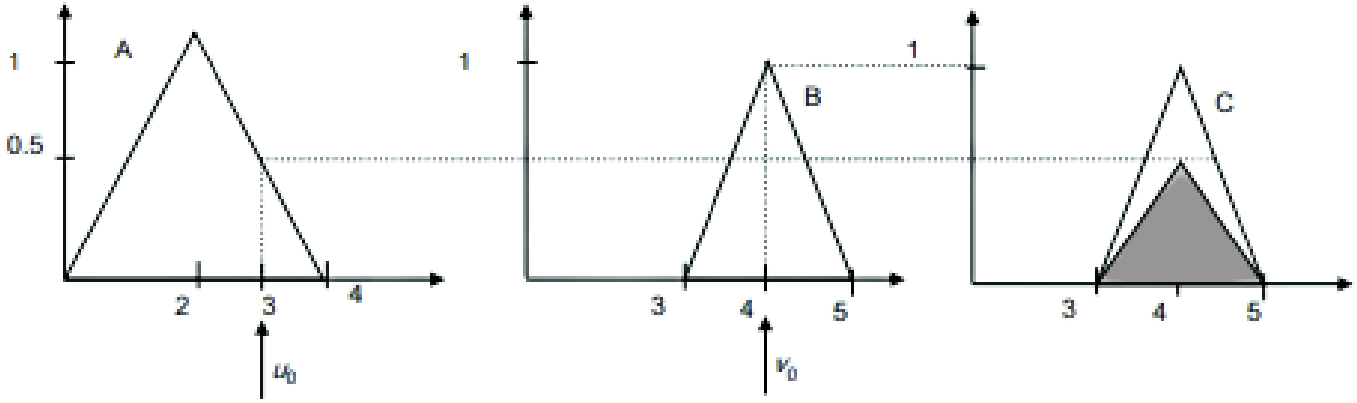
the fuzzy rule base consist of one rule:

R: If ***u*** is A and ***v*** is B then ***w*** is C

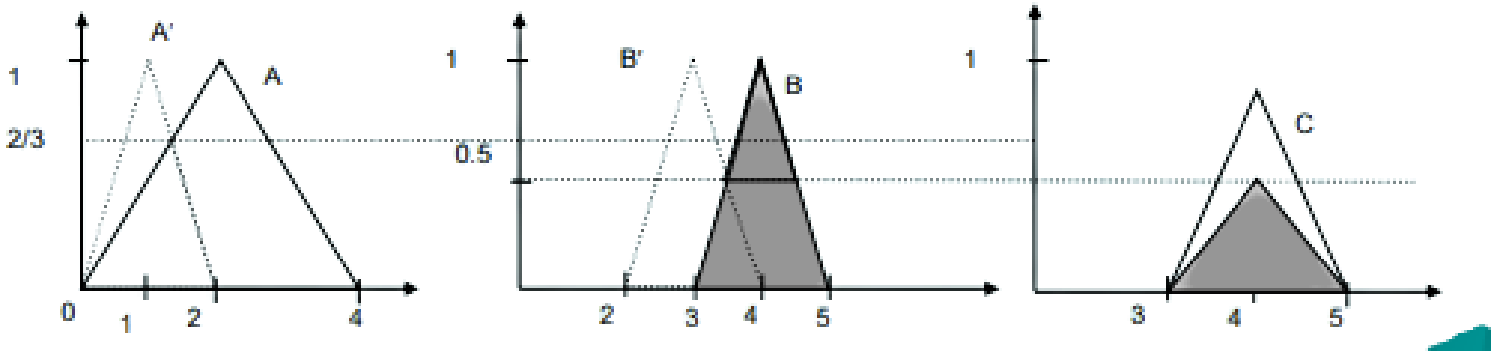
A=(0, 2, 4), B=(3, 4, 5) and C=(3,4,5) triangular fuzzy sets

What is the output C' if the inputs are s u_0=3, v_0=4?

What is the output C' if the inputs are sets A'=(0, 1, 2) **and** B'=(2,3,4)?



Larsen method with input u_0=3, v_0=4



LA FUNCION FMAX NOS RETORNA LOS MAYORES VALORES ENTRE 2 ARREGLOS.

```
IMPORT NUMPY AS NP

ARR1 = NP.ARRAY([1.5,3.5,4,NP.NAN])
ARR2 = NP.ARRAY([NP.NAN,6.5,2,NP.NAN])
PRINT(NP.FMAX(ARR1,ARR2))
```

```
[1.5 6.5 4.  nan]
```

```
#Combina la relacion fuzzy a una relacion agregada
R_combined = np.fmax(R1, np.fmax(R2, R3))

# Visualizar
plt.imshow(R_combined)
cbar = plt.colorbar()
cbar.set_label('Fuzzy pertenencia')
plt.yticks([i * 10 for i in range(8)], [str(i * 10 + 30) for i in range(8)]);
plt.ylabel('Temp')
plt.xlabel('Customers');
```

Las matrices de relaciones difusas muestran cómo responde el sistema para todas las temperaturas y todos los clientes. Pero necesitamos hacer una predicción específica para una temperatura específica. Necesitamos volver del dominio fuzzy a la lógica . Esto se conoce como defuzzificación.

Digamos que hace 75 grados afuera . ¿Cuántos clientes debe esperar la heladería?

```
fuzz.defuzz(customers, R_combined[temp == 75], 'centroid')
```

Ponostica unas 20 personas. ¿Parece razonable? Claro, pero no nos dice mucho. Eliminemos toda la relación, para tener una mejor idea intuitiva de lo que está pasando.

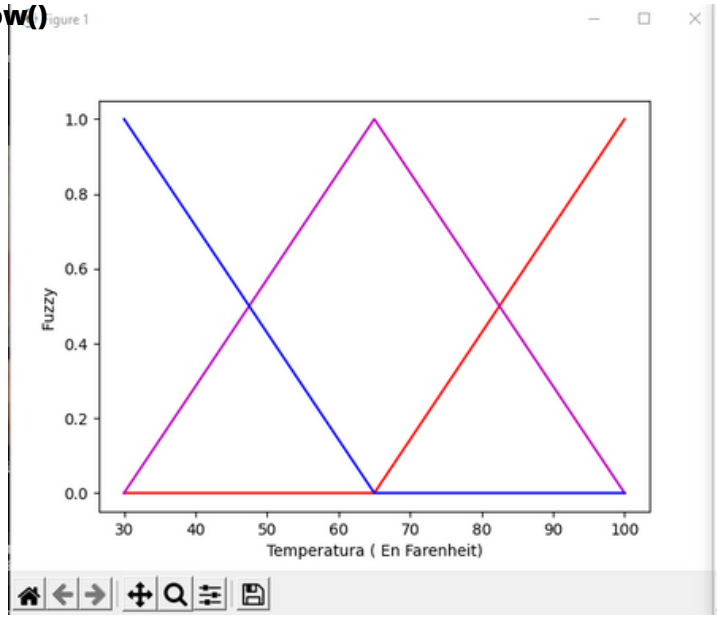
El resultado de este paso es efectivamente una tabla de consulta, que solo necesita consultarse, ¡no volver a calcularse! - en días futuros, a menos que se hayan ajustado las funciones de membresía del modelo.

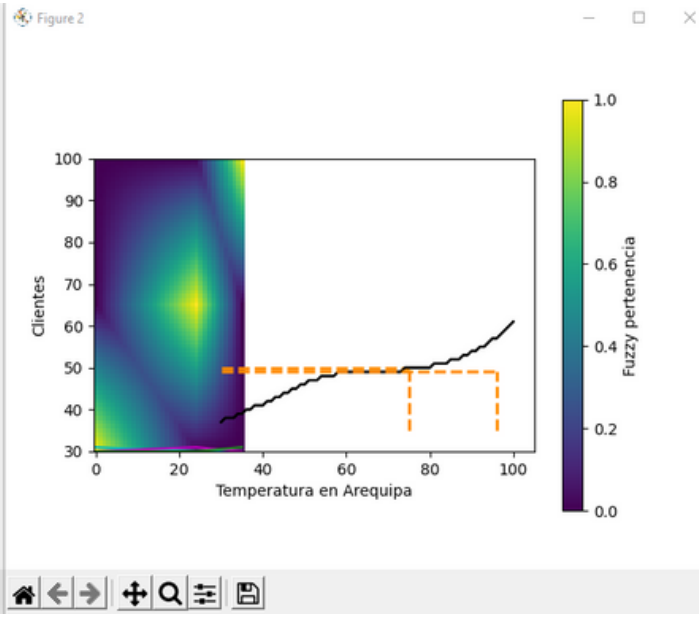
```
predicted_customers = np.zeros_like(temp)

for i in range(len(predicted_customers)):
    predicted_customers[i] = fuzz.defuzz(customers, R_combined[i, :], 'centroid')
```

Veamos cómo se ve la relación defusificada predecida.

```
# Numero de clientes en un caso hipotetico de 75 grados fahrenheit
plt.plot(temp, predicted_customers, 'k')
plt.vlines(75, 5, predicted_customers[temp == 75], color='DarkOrange', linestyle='dashed', lw=2)
plt.hlines(predicted_customers[temp == 75], 30, 75, color='DarkOrange', linestyle='dashed', lw=2)
plt.xlabel('Temperature')
plt.ylabel('Customers');
```

```
# Numero de clientes en Arequipa a 18 grados celsius a fahrenheit: 96 grados
plt.plot(temp, predicted_customers, 'k')
plt.vlines(96, 5, predicted_customers[temp == 64], color='DarkOrange',
linestyle='dashed', lw=2)
plt.hlines(predicted_customers[temp == 64], 30, 96, color='DarkOrange',
linestyle='dashed', lw=2)
plt.xlabel('Temperatura en Arequipa')
plt.ylabel('Clientes');
plt.show()
```





Bibliografía:

https://pythonhosted.org/scikit-fuzzy/auto_examples/plot_tipping_problem.html
https://numpy.org/doc/stable/reference/generated/numpy.fmax.html
https://github.com/JDWarner/scikit-fuzzy/blob/scipy2013/scikit-fuzzy_demo.ipynb