Lógica Difusa

Curso de Especialización en Inteligencia Artificial y Big Data

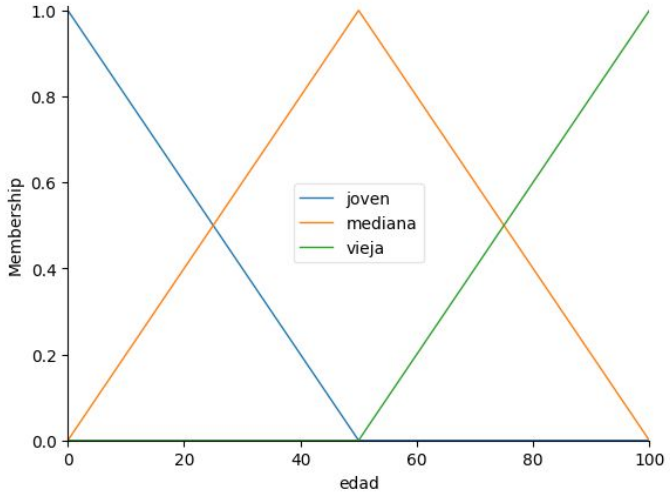
Modelos de Inteligencia Artificial

Introducción a la lógica difusa

La lógica difusa es una rama de la inteligencia artificial y la teoría de sistemas que se basa en el concepto de "verdad parcial" o "grado de verdad".

Lógica **clásica** → valores de verdad binarios (V, F)

Lógica **difusa** → manejo de la incertidumbre y la imprecisión.



¿Qué es la lógica difusa?

Tenemos un conjunto de los números naturales:

U = {1, 2, 3, 4, ...}

Mediante el predicado ‘impar’ (predicado preciso o clásico) se divide dicho conjunto en dos subconjuntos:

I = {1, 3, 5, 7, ...} → El predicado es verdadero

N = {2, 4, 6, 8, ...} → El predicado es falso

¿Qué es la lógica difusa?

¿Qué pasa si tenemos el predicado ‘alto’?

Este tipo de predicados no permiten realizar una división satisfactoria de una población de individuos en dos subconjuntos, aquéllos para los cuales ‘alto’ es cierto, y aquéllos para los que es falso → **Predicados vagos**.

Estos predicados protagonizan el **lenguaje ordinario**, lo dotan de flexibilidad y constituyen un elemento fundamental en la capacidad de abstracción del ser humano.

¡Un individuo puede ser al mismo tiempo alto, y no alto!

Otros ejemplos de predicados vagos: ‘velocidad moderada’, ‘bajas presiones’, ‘incremento súbito’.

¿Qué es la lógica difusa?

La lógica borrosa o difusa surge como la ciencia de los principios formales del razonamiento aproximado.

En la lógica difusa, las variables pueden tener múltiples grados de pertenencia a un conjunto, lo que se expresa en términos de un número entre cero y uno. Esto se logra mediante el uso de conjuntos difusos y funciones de pertenencia difusas. La inferencia difusa se utiliza para combinar las reglas lógicas difusas y producir una respuesta difusa.

Conjuntos borrosos

En el ejemplo anterior de los números naturales, aquellos que verifican el predicado ‘impar’ pertenecen al conjunto I.

En general, todo predicado preciso P, aplicado a una colección U , tiene asociado un conjunto preciso que denotaremos de la misma forma P ⊂ U , y que puede describirse mediante una función μP (u), cuyo valor viene determinado por la pertenencia a P de los distintos elementos u ∈ U , y definida como:

$$\mu_P(u) = \begin{cases} 0 & \text{si u}\notin \text{P} \\ 1 & \text{si u}\in \text{P} \end{cases}$$

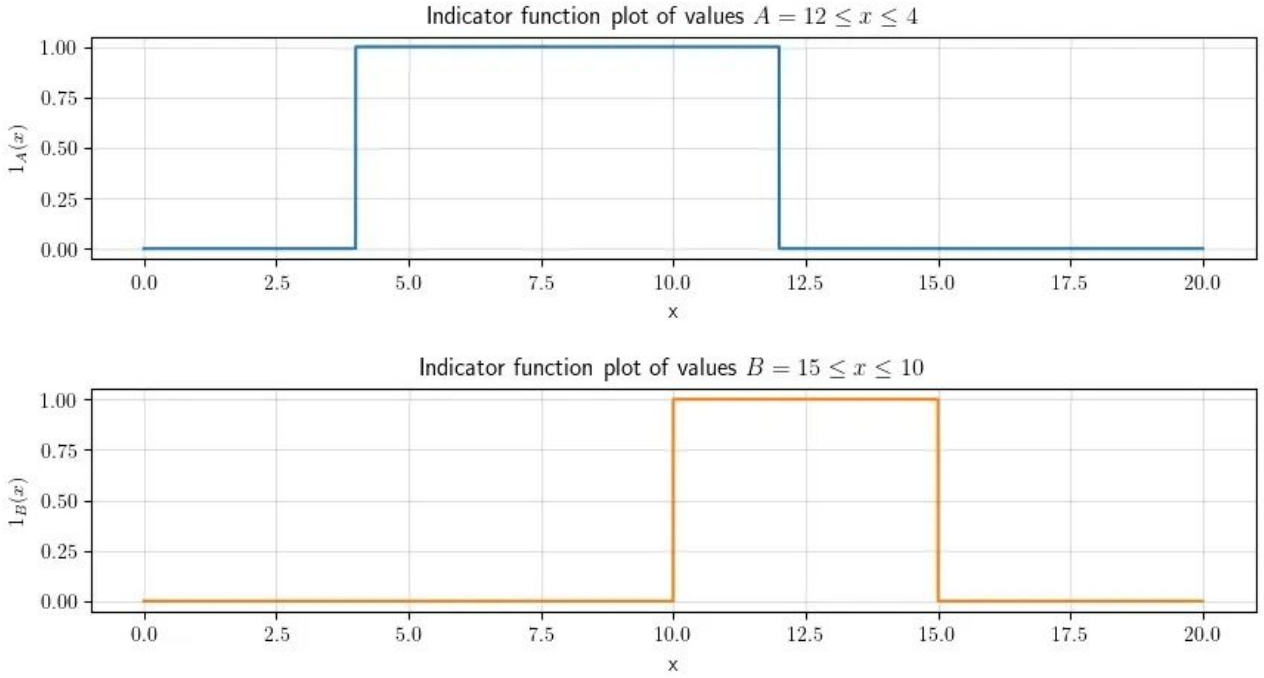
Conjuntos borrosos

Un predicado vago V, aplicado a la misma colección U , tiene asociado un conjunto borroso que denotaremos de la misma forma $V \subset U$, y que puede escribirse mediante una función μV (u), cuya representación de la pertenencia a V de los distintos elementos de U es una cuestión de grado, de modo que:

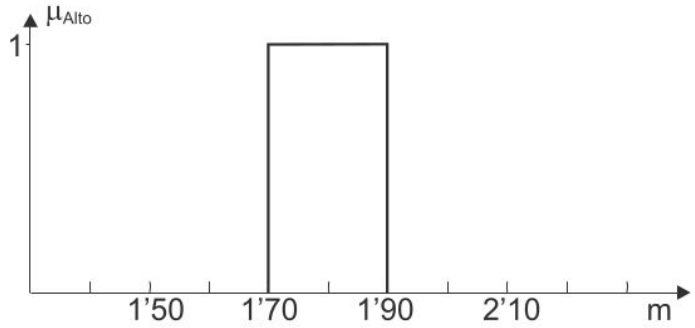
$$\mu_V : U \rightarrow [0, 1]$$

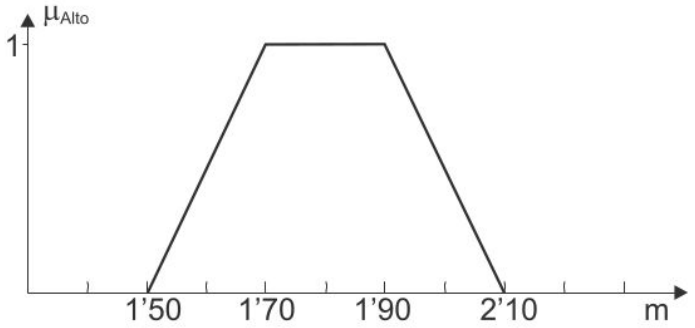
Además de aquellos elementos para los que el predicado V es cierto (μV (u) = 1), y aquéllos para los que V es falso (μV (u) = 0), existe un conjunto de elementos para los que el predicado V es cierto en un determinado grado $0 < \mu V (u) < 1$.

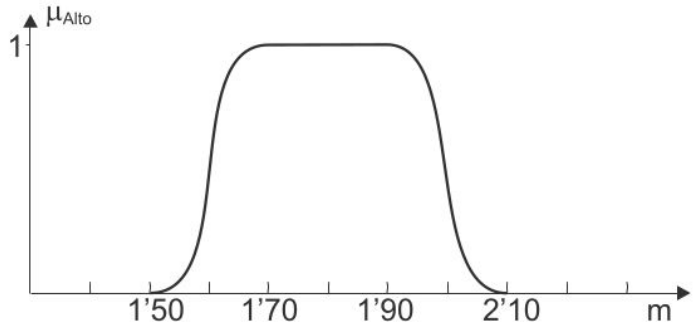
Conjuntos clásicos

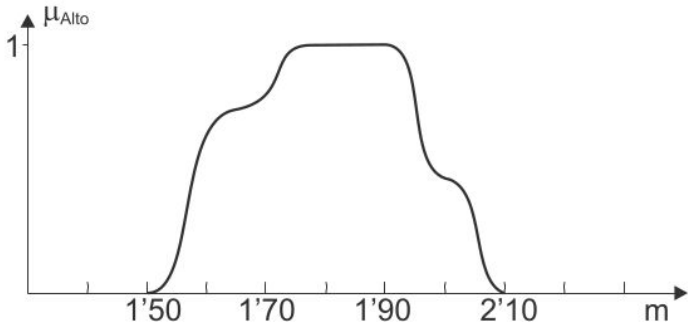


Conjuntos borrosos

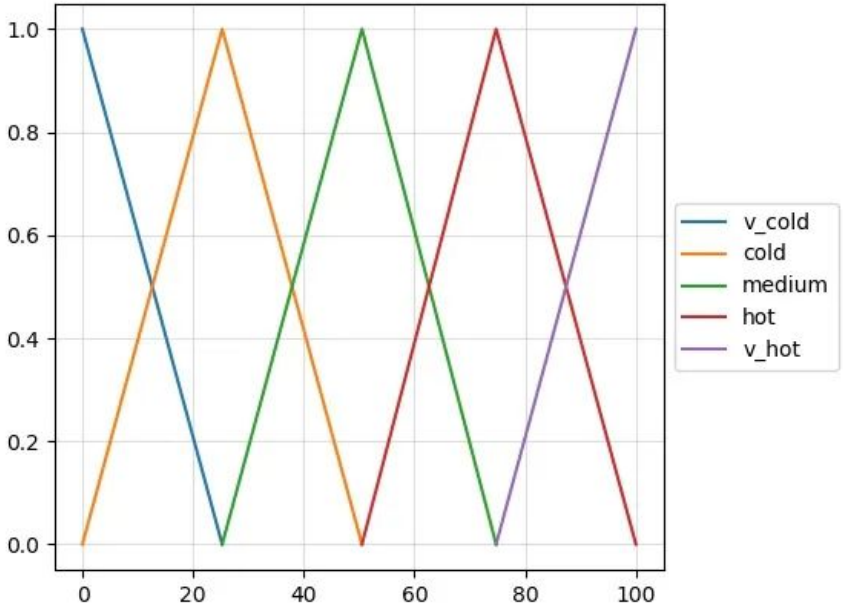




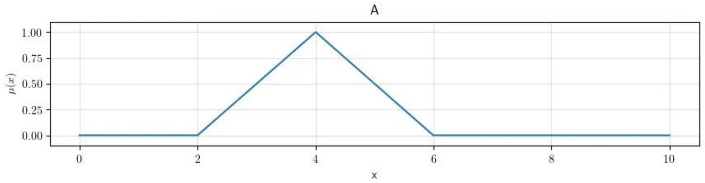


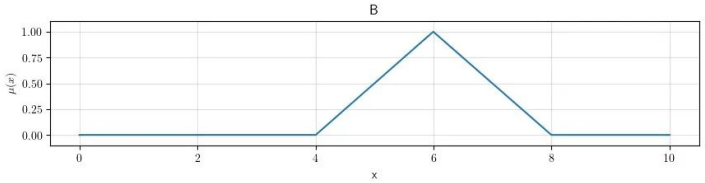


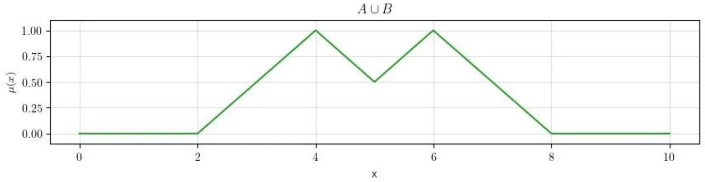
Conjuntos borrosos

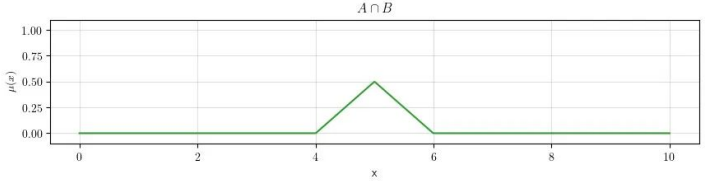


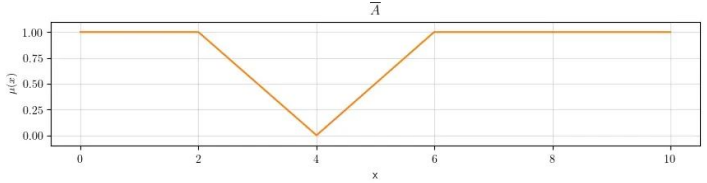
Unión, Intersección y Negación

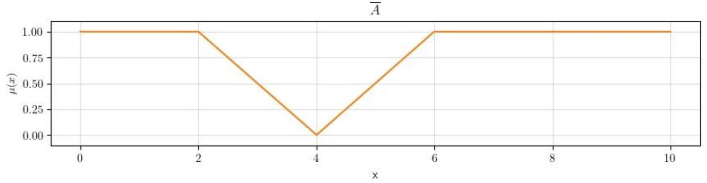












Unión, Intersección y Negación

Negación:

$$!A = \{u \in U : !(u \in A)\}$$

Intersección:

$$A \cap B = \{u \in U : u \in A \text{ y } u \in B\}$$

Unión:

$$A \cup B = \{u \in U : u \in A \text{ o } u \in B\}$$

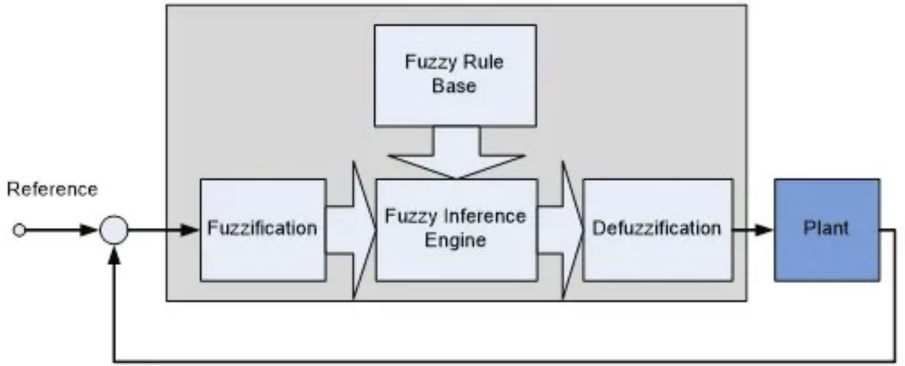
¿Cómo funciona la lógica difusa?

Un sistema difuso está formado por:

- Colección de **funciones** de pertenencia difusas
- Conjunto de **reglas** de lógica difusa

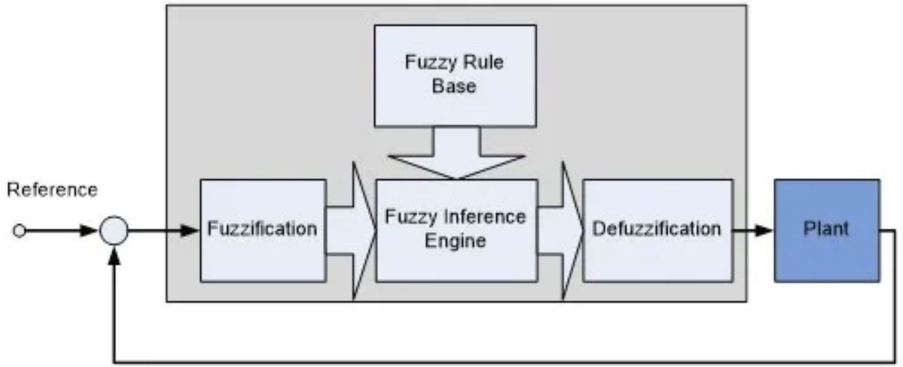
Ejemplo de regla:

IF (condiciones) **THEN** (consecuencias)



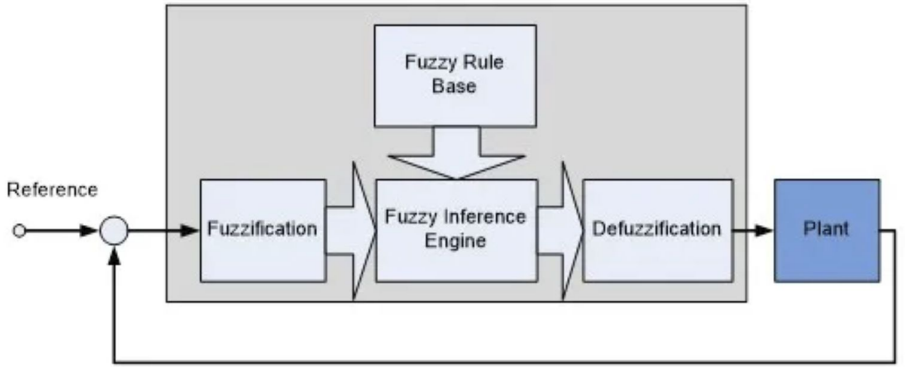
¿Cómo funciona la lógica difusa?

1. Fuzzifier: convierte la entrada precisa o numérica en un valor difuso, que representa la incertidumbre o imprecisión de la entrada en términos de pertenencia a un conjunto difuso.



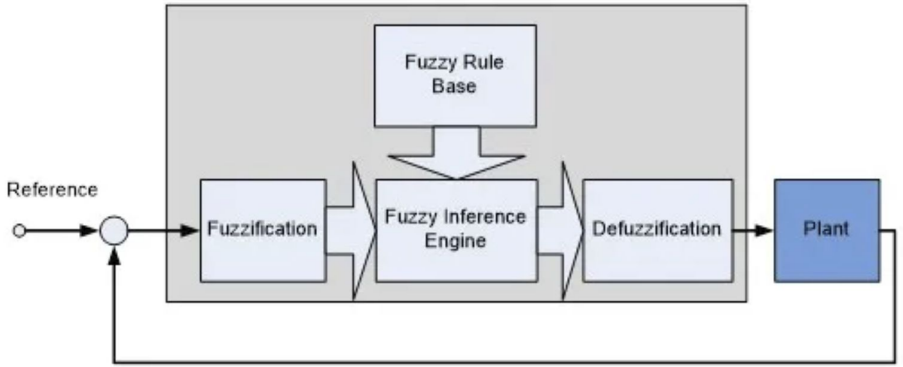
¿Cómo funciona la lógica difusa?

2. Base de conocimiento: conocimiento del dominio de aplicación y los objetivos de control correspondientes.



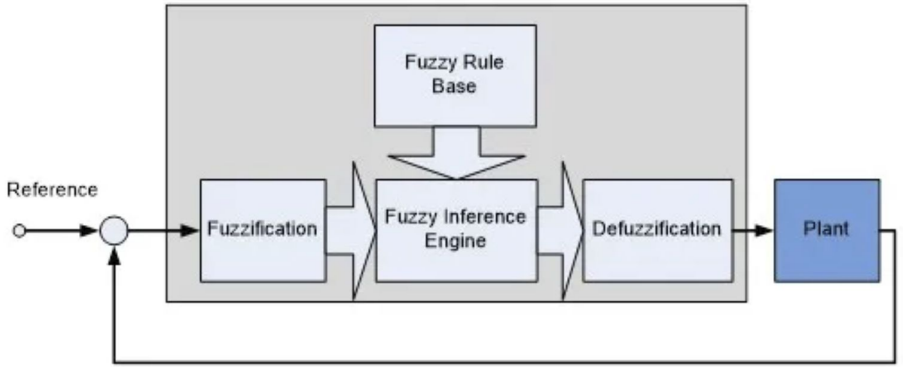
¿Cómo funciona la lógica difusa?

3. Motor de inferencia: deduce las acciones de control difuso empleando implicaciones difusas y reglas difusas de inferencia.



¿Cómo funciona la lógica difusa?

4. Defuzzification: convierte los valores de control difusos en valores no difusos, es decir, vincula un punto único a un conjunto difuso.



Paquete scikit-fuzzy en python

Se trata de un paquete de software libre cuyo código podemos acceder en el repositorio:

https://github.com/scikit-fuzzy/scikit-fuzzy

También se puede acceder a la documentación en:

https://pythonhosted.org/scikit-fuzzy/

Paquete scikit-fuzzy en python

Instalación:

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

Imports:

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

Paquete scikit-fuzzy en python

Variables de entrada/salida:

Entrada:

```
edad = ctrl.Antecedent(np.arange(0, 101, 1), 'edad')
```

Salida:

```
felicidad = ctrl.Consequent(np.arange(0, 11, 1), 'felicidad')
```

Paquete scikit-fuzzy en python

Funciones de membresía:

```
edad['joven'] = fuzz.trimf(edad.universe, [0, 0, 50])
```

`skfuzzy.membership.dsigmf`(x, b1, c1, b2, c2)	Difference of two fuzzy sigmoid membership functions.
`skfuzzy.membership.gauss2mf`(x, mean1, ...)	Gaussian fuzzy membership function of two combined Gaussians.
`skfuzzy.membership.gaussmf`(x, mean, sigma)	Gaussian fuzzy membership function.
`skfuzzy.membership.gbellmf`(x, a, b, c)	Generalized Bell function fuzzy membership generator.
`skfuzzy.membership.piecemf`(x, abc)	Piecewise linear membership function (particularly used in FIRE filters).
`skfuzzy.membership.pimf`(x, a, b, c, d)	Pi-function fuzzy membership generator.
`skfuzzy.membership.psigmf`(x, b1, c1, b2, c2)	Product of two sigmoid membership functions.
`skfuzzy.membership.sigmf`(x, b, c)	The basic sigmoid membership function generator.
`skfuzzy.membership.smf`(x, a, b)	S-function fuzzy membership generator.
`skfuzzy.membership.trapmf`(x, abcd)	Trapezoidal membership function generator.
`skfuzzy.membership.trimf`(x, abc)	Triangular membership function generator.
`skfuzzy.membership.zmf`(x, a, b)	Z-function fuzzy membership generator.

Paquete scikit-fuzzy en python

Reglas difusas:

```
regla1 = ctrl.Rule(edad['joven'] & ingreso['bajo'], felicidad['bajo'])
regla2 = ctrl.Rule(edad['joven'] & ingreso['medio'], felicidad['medio'])
regla3 = ctrl.Rule(edad['joven'] & ingreso['alto'], felicidad['alto'])
regla4 = ctrl.Rule(edad['mediana'] & ingreso['bajo'], felicidad['muy_bajo'])
regla5 = ctrl.Rule(edad['mediana'] & ingreso['medio'], felicidad['medio'])
regla6 = ctrl.Rule(edad['mediana'] & ingreso['alto'], felicidad['alto'])
regla7 = ctrl.Rule(edad['vieja'] & ingreso['bajo'], felicidad['muy_bajo'])
regla8 = ctrl.Rule(edad['vieja'] & ingreso['medio'], felicidad['bajo'])
regla9 = ctrl.Rule(edad['vieja'] & ingreso['alto'], felicidad['medio'])
```

Paquete scikit-fuzzy en python

Sistema de control:

```
sistema_ctrl = ctrl.ControlSystem([regla1, regla2, regla3, regla4,
regla5, regla6, regla7, regla8, regla9])

sistema = ctrl.ControlSystemSimulation(sistema_ctrl)
```

Computación de entrada:

```
sistema.input['edad'] = 30
sistema.input['ingreso'] = 60000
```

Aplicaciones de la lógica difusa

La **lógica difusa** se utiliza en una amplia variedad de aplicaciones:

- Control de procesos industriales → temperatura en un horno industrial
- Toma de decisiones en sistemas expertos
- Minería de datos
- Reconocimiento de patrones
- Robótica → agarre y manipulación de objetos con brazo robótico
- Ingeniería de sistemas
- Economía → relación entre el precio de un producto y su demanda
- Incertidumbre en el razonamiento humano

Ventajas y desventajas de la lógica difusa

Ventajas:

- Ofrece una forma de representar y manejar la incertidumbre
- Útil en sistemas complejos y no lineales
- Fácil de entender e interpretar → fácil representación gráfica
- Flexible y adaptable

Limitaciones:

- Subjetividad en la selección de las funciones de pertenencia
- Complejidad computacional en algunos casos → Limitación en tiempo real
- Los resultados pueden ser menos precisos
- Difícil de ajustar y optimizar → conocimiento experto + prueba y error