
	Computación	Docente: Diego Quisi
	SISTEMAS EXPERTOS	Período Lectivo: Septiembre 2020 – Febrero 2021

			<b>FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES</b>		
<b>Jonathan Atancuri</b>					
<b>CARRERA:</b> INGENIERÍA DE SISTEMAS			<b>ASIGNATURA:</b> SISTEMAS EXPERTOS		
<b>NRO. PRÁCTICA:</b>	4	<b>TÍTULO PRÁCTICA:</b> Lógica Difusa. Introducción a los conjuntos difusos y modelado de sistemas difusos.			
<b>OBJETIVO:</b> Familiarizarse con las operaciones fundamentales de los conjuntos difusos y cómo desarrollar las 3 etapas fundamentales de modelado de un sistema difuso: <i>fuzzification, inference, defuzzification</i> .					
<b>INSTRUCCIONES:</b>		1. Revisar el contenido teórico del tema			
		2. Profundizar los conocimientos revisando los libros guías, los enlaces contenidos en los objetos de aprendizaje y la documentación disponible en fuentes académicas en línea			
		3. Deberá desarrollar los scripts que permitan implementar sistemas expertos básicos basados en reglas.			
		4. Se sugiere aplicar la logica difusa en Python.			
		5. Fecha de Entrega: <b>31/01/2021 – 23:55</b>			
		6. Subir un documento PDF de la resolución y pruebas del sistema a demas del código fuente dentro de repositorio personal de <b>GIT</b> .			
<b>ACTIVIDADES POR DESARROLLAR</b>					

En un galpón se tiene una **temperatura de 18 grados** centígrados, y una **humedad de aproximadamente 22 grados** centígrados. Según estos valores determinar cual es la velocidad que **debería estar funcionando** el motor.

Para revisar las reglas, función de pertinencia y el proceso revisar el siguiente link:  
<https://medium.com/@javierdiazarca/l%C3%B3gica-difusa-ejercicio-2-bases-de-la-ia-1a8ae594cc15>

En base a ello, desarrollar e implementar el sistema dentro de Python o Java en donde me permita modificar los valores de la **temperatura** y humedad, generando así un sistema experto basado en lógica difusa para obtener la velocidad del motor de aire acondicionado.

Este sistema deberá tener la opción de poder modificar los valores de la temperatura y humedad con un scroll bar y obtener la velocidad de giro. Además, deberá presentarme las gráficas de pertenencia de INPUT/OUTPUT del sistema difuso y como estas varían de acuerdo al cambio de las variables.

**Nota:** Esta práctica remplazará la segunda prueba con los siguientes criterios de evaluación:

- GUI: 40%
- FUZZY LOGIC: 40%
- informe y pruebas: 20%

Subir el informe en formato PDF y los códigos fuentes al repositorio Git personal.

#### **RESULTADO(S) OBTENIDO(S):**

Entender cómo representar y modelar el conocimiento a través de sistemas expertos basados en razonamiento incierto.

Temperatura	Humedad	RPM del Motor
Baja	Alta	Baja
Media	Alta	Media
Alta	Alta	Media
Baja	Media	Baja
Media	Media	Baja
Alta	Media	Media
Baja	Baja	Baja
Media	Baja	Baja
Alta	Baja	Alta

**Configuramos los rangos de la temperatura , humedad ,rpm del motor**

```
temperatura = ctrl.Antecedent(np.arange(1, 61, 1), 'temperatura')
humedad = ctrl.Antecedent(np.arange(0, 61, 1), 'humedad')
rpm = ctrl.Consequent(np.arange(0, 61, 1), 'rpm')
```

**Definición de los descriptores de las variables de entrada y salida**

```
temperatura['Baja'] = fuzz.trapmf(temperatura.universe, [0, 0, 10, 20])
temperatura['Media'] = fuzz.trimf(temperatura.universe, [20, 35, 50])
temperatura['Alta'] = fuzz.trapmf(temperatura.universe, [40, 60, 70, 70])

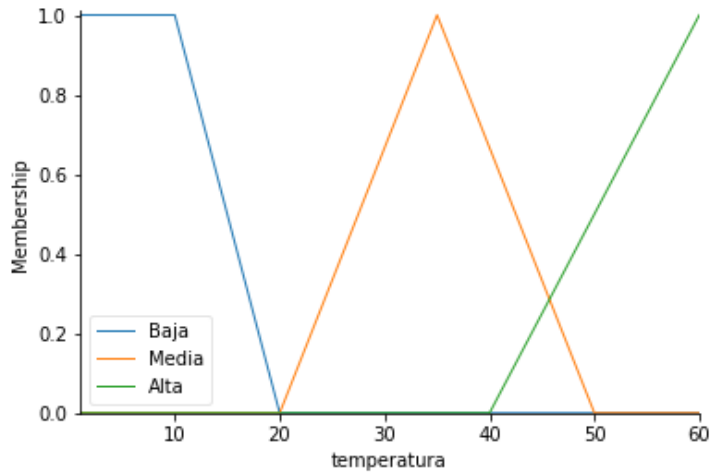
humedad['Baja'] = fuzz.trapmf(humedad.universe, [0, 0, 10, 20])
humedad['Media'] = fuzz.trimf(humedad.universe, [10, 40, 60])
humedad['Alta'] = fuzz.trapmf(humedad.universe, [50, 70, 100, 100])

rpm['Baja'] = fuzz.trapmf(rpm.universe, [0, 0, 10, 20])
rpm['Media'] = fuzz.trimf(rpm.universe, [10, 30, 45])
rpm['Alta'] = fuzz.trapmf(rpm.universe, [40, 55, 100, 100])
```

## Representación gráfica de la función de pertenencia de Entrada

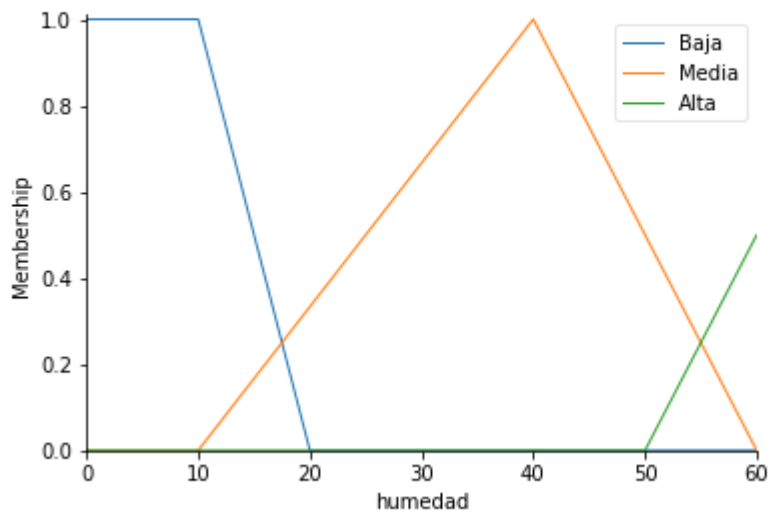
### Temperatura

```
temperatura.view()
```



### Humedad

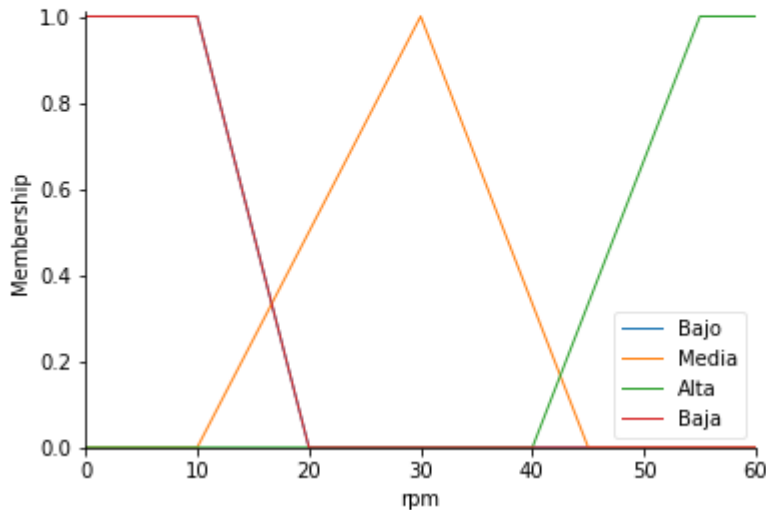
```
humedad.view()
```



## Funciones de Pertinencia de Salida

### RPM del Motor

```
rpm.view()
```



## Formando la Base de Reglas

```
rule1 = ctrl.Rule(temperatura['Baja'] & humedad['Alta'], rpm['Baja'])
rule2 = ctrl.Rule(temperatura['Media'] & humedad['Alta'], rpm['Media'])
rule3 = ctrl.Rule(temperatura['Alta'] & humedad['Alta'], rpm['Media'])
rule4 = ctrl.Rule(temperatura['Baja'] & humedad['Media'], rpm['Baja'])
rule5 = ctrl.Rule(temperatura['Media'] & humedad['Media'], rpm['Baja'])
rule6 = ctrl.Rule(temperatura['Alta'] & humedad['Media'], rpm['Media'])
rule7 = ctrl.Rule(temperatura['Baja'] & humedad['Baja'], rpm['Baja'])
rule8 = ctrl.Rule(temperatura['Media'] & humedad['Baja'], rpm['Baja'])
rule9 = ctrl.Rule(temperatura['Alta'] & humedad['Baja'], rpm['Alta'])

rule1.view()
```

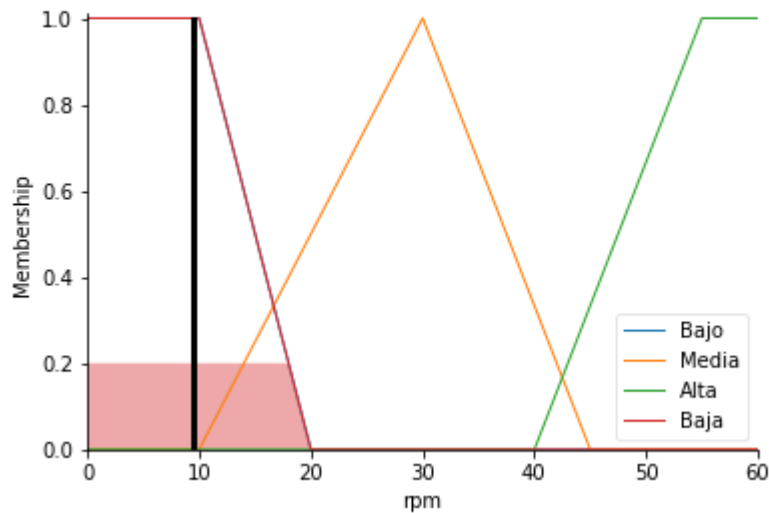
Ingresamos como los parámetros de entrada la temperatura de 18 y la humedad de 22

```
tipping_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8, rule9])
tipping = ctrl.ControlSystemSimulation(tipping_ctrl)

tipping.input['temperatura'] = 18
tipping.input['humedad'] = 22
print ('La velocidad que funciona el motor es : ')
tipping.compute()
print (tipping.output['rpm'])
#s=float(tipping.output['riesgo'])
#session.write_transaction(neo4j.crear_nodo, s, )
rpm.view(sim=tipping)
```

Y nos devuelve un valor de la velocidad que funciona el motor

La velocidad que funciona el motor es :  
9.508771929824562



Ahora procedemos a calcular el centroide de manera manual

Cálculo de defuzzyfication con área

#### Centroides

$$C1 : (18 / 2) = 9$$

$$C2 : [(20 - 18) / 3] + 18 = 18.66$$

#### Áreas Parciales

$$A1 : B \times A$$

$$A1 : 18 \times 0.3 = 5.4$$

$$A2 : (B \times A) / 2$$

$$A2 : (2 \times 0.3) / 2 = 0.3$$

#### Área Total

$$AT = A1 + A2$$

$$AT = 5.4 + 0.3 = 5.7$$

#### Cálculo del centroide

$$Centroid = \frac{C1 \cdot A1 + C2 \cdot A2}{A(total)} \quad \text{---} \quad Centroid = \frac{(9 \cdot 5.4) + (18.66 \cdot 0.3)}{5.7}$$

$$Centroid = 9.51$$

Nos devuelve un valor de 9.51

```
tipping.compute()
prediccion_final = tipping.compute()
prediccion_final = tipping.output['rpm']
print ("Cálculo de resultados computacionales = %.2f" % prediccion_final)
```

Cálculo de resultados computacionales = 9.51

Procedemos a realizar el calculo con la precisión y el error

## Cálculo de precisión y error para comparar el conteo manual con la computación

```
outputmanual = 9.51
if prediccion_final < outputmanual:
    precision = prediccion_final*100/outputmanual ## akurasi jika nilai komp
else:
    precision=(outputmanual/prediccion_final)*100
error=100-precision
print ("Precision = %.2f" % precision + " %")
print ("Error = %.2f" % error + " %")
```

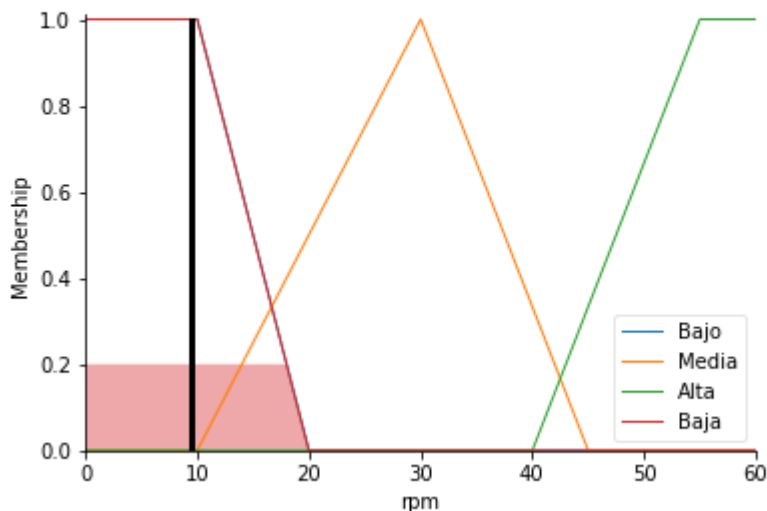
Precision = 99.99 %

Error = 0.01 %

Finalmente visualizamos la predicción final

## Visualización de predicción final

```
rpm.view(sim=tipping)
```



Link repositorio :


<https://github.com/JonathanAtancuri3218/SistemasExpertos/tree/master/Trabajos/2do%20Interciclo/Pruebas/Prueba2-logicadifusa>

## CONCLUSIONES:

- Los estudiantes identifican las principales estructuras, esquema de modelado y etapas principales para representar el conocimiento y desarrollar sistemas expertos basados en razonamiento incierto.

## RECOMENDACIONES:

- Revisar la información proporcionada por el docente previo a la práctica.
- Haber asistido a las sesiones de clase.
- Consultar con el docente las dudas que puedan surgir al momento de realizar la práctica.

	Computación	Docente: Diego Quisi
	SISTEMAS EXPERTOS	Período Lectivo: Septiembre 2020 – Febrero 2021

**Firma:** Jonathan Atancuri