Instituto Tecnológico de Culiacán

Ingeniería en tecnologías de la información y comunicaciones

Tópicos de inteligencia artificial

Unidad 1

Tarea 2

“Lógica difusa y sistemas expertos”

Jesús Alberto Barraza Castro

Contenido:

Definición de Sistema Experto:

Un sistema experto es un tipo de software que utiliza conocimiento experto para resolver problemas complejos dentro de un dominio específico. Estos sistemas intentan emular el proceso de toma de decisiones de un experto humano, aplicando reglas y hechos que se encuentran almacenados en una base de conocimiento.

Componentes de un Sistema Experto:

Base de conocimiento: Esta es la parte central de un sistema experto. Consiste en un conjunto de hechos y reglas que definen el dominio del problema. Las reglas son de tipo "si-entonces", donde las premisas o condiciones se evalúan y, si son verdaderas, se ejecutan ciertas acciones o conclusiones.

Motor de inferencia: Es el componente que procesa las reglas y hechos en la base de conocimiento para generar conclusiones o decisiones. Dependiendo de la técnica utilizada, el motor de inferencia puede ser deductivo (donde se parte de hechos generales hacia conclusiones específicas) o inductivo (donde se generan conclusiones basadas en ejemplos o experiencias previas).

Interfaz de usuario: Permite la interacción del usuario con el sistema. El usuario puede ingresar datos sobre el problema o consulta, y el sistema responde con una solución o recomendación basada en las reglas y hechos almacenados.

Aplicaciones de los Sistemas Expertos:

Los sistemas expertos se aplican en diversos campos, tales como:

Diagnóstico médico: Los sistemas expertos pueden ayudar a los médicos a diagnosticar enfermedades basadas en los síntomas y antecedentes del paciente.

Asistencia en decisiones empresariales: Ayudan a tomar decisiones sobre ventas, inventarios, marketing, etc., en empresas.

Mantenimiento predictivo en la industria: Detectan fallas en equipos mediante el análisis de datos operativos.

Definición de Lógica Difusa

La lógica difusa es una extensión de la lógica booleana clásica que permite manejar la imprecisión y la incertidumbre. Mientras que la lógica clásica se basa en dos valores (verdadero o falso), la lógica difusa permite valores intermedios entre 0 y 1, lo que refleja más adecuadamente la complejidad y variabilidad del mundo real.

Componentes de la Lógica Difusa:

Conjuntos difusos: En la lógica clásica, un elemento pertenece completamente a un conjunto o no pertenece en absoluto. En contraste, los conjuntos difusos permiten que un elemento pertenezca parcialmente al conjunto, asignando un valor de pertenencia entre 0 y 1. Por ejemplo, la temperatura "caliente" puede tener un valor de pertenencia de 0.8 en un conjunto difuso de temperaturas, lo que indica que está cerca de ser caliente, pero no completamente.

Funciones de pertenencia: Cada conjunto difuso tiene una función de pertenencia, que es una función matemática que asigna un valor entre 0 y 1 a cada elemento del universo de discurso. Las funciones de pertenencia pueden tener varias formas, como triangulares, trapezoidales o gaussianas, dependiendo de cómo se modelen las relaciones difusas.

Operadores difusos: La lógica difusa también tiene operadores como la AND difusa (conjunción), OR difuso (disyunción), y NOT difuso (negación), que operan sobre los valores de pertenencia de los elementos. Estos operadores permiten combinar diferentes conjuntos difusos y tomar decisiones basadas en grados de pertenencia.

Control difuso: La lógica difusa se usa ampliamente en sistemas de control de procesos, como el control de temperatura en hornos, la velocidad en motores eléctricos o la estabilidad de vehículos autónomos.

Sistemas de clasificación: La lógica difusa se aplica en sistemas de clasificación y toma de decisiones, como los sistemas de diagnóstico o de recomendación.

Aplicación de la Lógica Difusa en los Sistemas Expertos:
La lógica difusa se integra en los sistemas expertos para manejar la incertidumbre inherente a muchos problemas del mundo real. En lugar de tomar decisiones con información precisa y clara, los sistemas expertos difusos pueden trabajar con información imprecisa o vaga, lo que les permite tomar decisiones más flexibles y realistas.

Cómo se Aplica la Lógica Difusa en los Sistemas Expertos:
Manejo de información imprecisa: En muchos casos, la información que se ingresa en un sistema experto no es exacta. Por ejemplo, en un diagnóstico médico, un paciente puede no tener un síntoma claro o un valor numérico preciso, entonces, usando lógica difusa, el sistema puede considerar "temperatura alta" como un valor de pertenencia en un conjunto difuso (por ejemplo, con una pertenencia de 0.7 en lugar de solo "sí" o "no").

Reglas difusas: Las reglas en un sistema experto difuso pueden ser del tipo "si-entonces", pero las premisas y las conclusiones son difusas. Por ejemplo: "Si la temperatura es un poco alta, entonces el paciente puede tener fiebre". Aquí, "un poco alta" no es una condición binaria, sino que es una condición difusa con un valor de pertenencia que puede variar dependiendo de los datos de entrada.

Motor de inferencia difuso: El motor de inferencia en un sistema experto difuso evalúa las reglas difusas y hace inferencias a partir de los valores de pertenencia. Puede utilizar métodos de inferencia difusa, como el método de Mamdani o el método de Sugeno, para procesar las reglas y generar conclusiones difusas.

Desfuzzificación: Una vez que el sistema experto difuso ha procesado las reglas y generado conclusiones difusas, estas conclusiones deben ser convertidas de nuevo a valores crispos (claros). Este proceso se llama desfuzzificación, y puede utilizar diferentes métodos, como el método del centro de gravedad.

Ejemplo Práctico: Sistema de Control Difuso en un Ordenador de Placa Reducida

Proyecto de titulación de Gustavo Valenzuela titulado "Desarrollo de un sistema de control basado en lógica difusa en un ordenador de placa reducida". Este proyecto tiene como objetivo la implementación de un sistema de control difuso utilizando un ordenador de placa reducida, comúnmente conocido como hardware embebido (por ejemplo, una Raspberry Pi o Arduino). Este tipo de hardware es ideal para proyectos de bajo costo, pero permite ejecutar algoritmos de control avanzados, como la lógica difusa, sin requerir un sistema de cómputo complejo.

Objetivo del Proyecto:
El propósito principal de este proyecto es implementar un sistema de control difuso para manejar un proceso físico en un entorno embebido. El proyecto está enfocado en la utilización de la lógica difusa como herramienta para la toma de decisiones en condiciones de incertidumbre o imprecisión, lo cual es común en sistemas de control reales. La lógica difusa es especialmente útil cuando las variables de entrada, como temperatura, presión o velocidad, no se pueden definir de forma exacta, sino que pueden tomar valores aproximados que necesitan ser gestionados adecuadamente.

Componentes del Sistema:
Hardware Embebido: El proyecto utiliza un ordenador de placa reducida como plataforma de hardware. Este tipo de sistemas es ideal para la implementación de sistemas de control a pequeña escala, y permite el manejo de sensores y actuadores de manera eficiente.

Lógica Difusa: Se emplea lógica difusa para modelar las variables de entrada y salida del sistema. En lugar de tener reglas de control exactas, la lógica difusa permite definir reglas aproximadas que pueden manejar valores vagos o inciertos. Por ejemplo, en lugar de decir que la temperatura debe ser "exactamente 30°C", se puede decir que la temperatura debe estar "baja" o "alta", usando rangos difusos.

Simulaciones: El proyecto incluye códigos de simulación que permiten probar el comportamiento del sistema de control sin necesidad de implementar directamente en el hardware. Estas simulaciones se realizan utilizando lenguajes como MATLAB o Python, lo que facilita la prueba y ajuste de los algoritmos antes de la implementación en un sistema real.

Desarrollo del Algoritmo de Control Difuso: El núcleo del proyecto es la implementación de un algoritmo de control difuso que procesa las lecturas de los sensores y genera señales de salida para controlar los actuadores del sistema. Este algoritmo toma decisiones basadas en un conjunto de reglas difusas que determinan las respuestas del sistema bajo diferentes condiciones.

Codigo de ejemplo:

```
######################################################################
#                  Escrito por: Gustavo Valenzuela                   #
#                  gustavo.valenzuela.ing@gmail.com                  #
######################################################################

"""
  En este código se simula el sistema de control difuso PD+I utilizando
  una LookUp-Table.
"""

import math
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import RectBivariateSpline
from modelo_planta import salida

# Parámetros de la planta
a = 1.00151e-4
b = 8.67973e-3
g = 40
Y0 = 25 # Temperatura ambiente
Ts = 25 # Tiempo de muestreo
aTs = math.exp(-a*Ts)
bTs = (b/a)*(1-math.exp(-a*Ts))

# Importar lookup table
LookUpTableData = np.loadtxt(open("LookUpTableData.csv"), delimiter=",")
E = np.linspace(-1,1,np.size(LookUpTableData,0))  # Discretización de filas
CE = np.linspace(-1,1,np.size(LookUpTableData,1)) # Discretización de columnas
interp = RectBivariateSpline(E,CE,LookUpTableData,kx=3,ky=3) # Crear clase interpolación

# Ganancias controlador
x_1 = np.array([0.065751091003418,4.499999918447061,3.017834563310598e-05,1.251905164602053e-05])
x_2 = np.array([0.065999962502552,3.884999892217342,4.994046780049530e-05,0.009999738029963])

Setpoint = np.array([65,80])  # Salida deseada (°C)
hr = 2                        # Horas
Time = 3600*hr                # Tiempo total de simulacion (s)
n = round(Time/Ts)            # Numero de muestras

```

```
41  # Pre-asignar todas las matrices para optimizar el tiempo de simulacion
42  t = np.arange(0,Time,Ts)
43  u = np.zeros((n,1))
44  y = np.zeros((n,1))
45  y[0] = 50
46  e = np.zeros((n,1))
47  de = np.zeros((n,1))
48  ie = np.zeros((n,1))
49  efuzz1 = np.zeros((n,1))
50  efuzz2 = np.zeros((n,1))
51  r = np.zeros((n,1))
52
53  # Cambio de setpoint
54  for i in range(2):
55      if i == 0:
56          r[0:n//2] = Setpoint[i]
57      else:
58          r[n//2:] = Setpoint[i]
59
60  # Bucle de control
61  for k in range(n-1):
62      # Asignación de ganancias
63      if r[k] == Setpoint[0]:
64          x = x_1
65      else:
66          x = x_2
67      GE = x[0]
68      GU = x[1]
69      GIE = x[2]
70      GCE = x[3]
71      # Cálculos controlador
72      e[k] = r[k] - y[k]             # Error actual
73      if k == 0:
74          de[k] = e[k]/Ts            #Cáculos para primera iteración
75          ie[k] = 0
76      else:
77          de[k] = (e[k] - e[k-1])/Ts # Derivada error (euler hacia atrás)
78          ie[k] = e[k-1]*Ts          # Integral error (euler hacia adelante)
79      I = np.sum(ie)                 # Suma integral error
80      efuzz1[k] = GE*e[k]            # Multiplicar error por ganancia GE
81      efuzz2[k] = GCE*de[k]          # Multiplicar derivada por ganancia GCE
82      u[k] = GU*(interp.ev(efuzz1[k],efuzz2[k]) + GIE*I) # Cálculo acción de control
83      if k < n:
84          y[k+1] = salida(y[k],u[k],aTs,bTs,g,Y0)        # Cálculo de salida de la planta
85
```

Explicacion:

Este código simula el comportamiento de un sistema de control de temperatura utilizando un controlador difuso PD+I. Lo que hace este sistema es tratar de mantener la temperatura en un valor deseado (setpoint), ajustando la entrada al sistema de forma inteligente utilizando lógica difusa.

1. ¿Qué es un controlador PD+I?

Es un tipo de controlador que tiene tres partes:

P (Proporcional): El controlador ajusta la salida en función de cuán grande es el error (la diferencia entre la temperatura real y la deseada).

D (Derivativo): El controlador ajusta la salida basándose en la velocidad de cambio del error.
I (Integral): El controlador ajusta la salida tomando en cuenta todo el error acumulado a lo largo del tiempo.
Este código usa un control difuso (basado en lógica difusa) para ajustar cómo se controla el error de manera imprecisa pero eficiente, a diferencia de los controladores convencionales que son más rígidos.

2. La planta:
La planta en este caso es el sistema que se quiere controlar, que es una caja negra que transforma la señal de entrada (acción de control) en una salida (temperatura).

En el código, los parámetros como a, b, g, Y0 y Ts representan características de cómo se comporta la planta, es decir, cómo reacciona la temperatura cuando se le aplica una acción de control. En este caso, la planta tiene que ver con un control de temperatura.

3. LookUp Table (Tabla de Búsqueda):
En el código, se usa una tabla de búsqueda (LookUp Table), que es una especie de "método rápido" para obtener los resultados de un cálculo sin hacerlo desde cero. Aquí, la tabla contiene valores predefinidos (calculados con anterioridad) que se utilizan para obtener las acciones de control difusas.

4. Ganancias del controlador:
Las ganancias son valores que determinan cómo se ajustan las diferentes partes del controlador (proporcional, derivativo, integral). Los valores x_1 y x_2 se usan para asignar diferentes ganancias dependiendo del setpoint (la temperatura que se quiere alcanzar). En otras palabras, cuando el setpoint cambia (por ejemplo, de 65°C a 80°C), se usan diferentes valores de ganancias para ajustar mejor el control.

5. Simulación:

El código simula el comportamiento del sistema durante un período de tiempo (en este caso, 2 horas) y va calculando la temperatura en cada instante de tiempo. Para esto, el controlador calcula el error en cada paso (la diferencia entre la temperatura deseada y la real) y ajusta la salida de la planta utilizando los valores de la tabla de búsqueda y las ganancias.

6. ¿Cómo funciona el cálculo?
Cada vez que pasa un intervalo de tiempo (definido por Ts), el controlador realiza lo siguiente:

Calcula el error: La diferencia entre el setpoint y la temperatura real.
Deriva y acumula el error: Usa la velocidad de cambio del error y la acumulación de los errores pasados para ajustar la entrada al sistema.
Usa la tabla de búsqueda: El controlador usa la tabla de búsqueda para obtener un valor difuso que ajusta la entrada al sistema.
Calcula la nueva temperatura: Aplica el valor de control al sistema para calcular la nueva temperatura.
7. Visualización de los resultados:
Al final, el código grafica cómo cambia la temperatura a lo largo del tiempo. La línea roja muestra el setpoint (la temperatura que queremos), y la línea azul muestra la temperatura real que alcanza el sistema. Si todo va bien, la temperatura real debe seguir de cerca el setpoint.

Resumen en pocas palabras:
Este código simula un controlador difuso PD+I para mantener la temperatura de un sistema en un valor deseado. Utiliza una tabla de búsqueda para obtener resultados rápidamente y ajustar la acción de control, de modo que el sistema pueda reaccionar de manera más flexible a los cambios en la temperatura. El objetivo es que, al final de la simulación, la temperatura real se acerque lo más posible a la temperatura deseada.

Referencias:

Zadeh, L. A. (1965). Fuzzy Sets. Information and Control, 8(3), 338-353.
Mendel, J. M. (2001). Fuzzy Logic Systems for Engineering: A Tutorial. Proceedings of the IEEE, 90(9), 1413-1427.
Passino, K. M., & Yurkovich, S. (1998). Fuzzy Control. Addison-Wesley Longman Publishing Co., Inc.
Kuo, B. C., & Golnaraghi, F. (2003). Automatic Control Systems (8th Edition). John Wiley & Sons, Inc.

Repositorio del codigo: https://github.com/g-valenzuela/Fuzzy_controller