WIFE, un entorno para crear modelos difusos

Juan Moreno-Garcia[1 [0000−0003−2430−145X]], David Muñoz-Valero[1 [0000−0002−2509−9911]], Enrique Adrian Villarrubia-Martin[2 [0009−0002−8006−5711]], Julio Alberto López-Gómez[2 [0000−0001−6291−6637]], and Luis Jimenez-Linares[1 [0000−0002−2737−5887]]

[1] Escuela de Ingeniería Industrial y Aeroespacial, Departamento de Tecnologías y Sistemas de Información, Universidad de Castilla-La Mancha, 45071, Toledo, España
juan.moreno,david.munoz@uclm.es
[2] Escuela Superior de Informática, Departamento de Tecnologías y Sistemas de Información, Universidad de Castilla-La Mancha, 13071, Ciudad Real, España
enrique.villarrubia,julioalberto.Lopez,luis.jimenez@uclm.es

Abstract. Se presenta un entorno software desarrollado en Python para la construcción, análisis y ejecución de modelos difusos. El entorno está operativo, pero todavía no cuenta con todas las funcionalidades deseadas. El sistema permite implementar modelos de inferencia clásicos basados en Mamdani y Takagi-Sugeno-Kang (TSK), integrando ambos enfoques dentro de una arquitectura común. Sin embargo, su principal aportación radica en su flexibilidad estructural, que posibilita la construcción de estructuras difusas más complejas que los modelos tradicionales, siendo adecuada tanto para investigación como para docencia.

Keywords: Inferencia difusa · Modelos difusos · Estructuras difusas

1 Introducción

La lógica difusa constituye una herramienta fundamental para el modelado de sistemas complejos caracterizados por incertidumbre, imprecisión o conocimiento heurístico. Tradicionalmente, los modelos difusos más utilizados en la literatura son los modelos de tipo Mamdani y los modelos de tipo TSK, aunque también es usual experimentar con otras estructuras adaptadas al problema tratado. Se propone un entorno software que no solo implementa estos modelos, sino que proporciona una base sólida para el desarrollo de nuevas estructuras difusas ofreciendo implementaciones de las componentes teóricos fundamentales. El entorno propuesto se fundamenta en tres ideas principales que, utilizadas conjuntamente, lo distinguen de otros entornos existentes en la actualidad:

(1) **Aprovechamiento de la potencia del lenguaje Python**: Una de las carácterísticas principales de Python es la denominada *First-class function*[3] [1]. Aunque el soporte de funciones como elementos de primer nivel no es exclusivo

[3] https://en.wikipedia.org/wiki/First-class_function

de Python y está presente en numerosos lenguajes modernos, este destaca especialmente por la amplitud de su ecosistema y su comunidad, que facilitan la reutilización y extensión del software en múltiples dominios. Asimismo, si bien su rendimiento en ejecución es inferior al de lenguajes compilados, su alta productividad y la posibilidad de apoyarse en librerías altamente optimizadas permiten obtener soluciones eficientes con un coste de desarrollo reducido. Referente a esto, también es destacable que los lenguajes interpretados facilitan el uso de archivos externos que contienen funciones o definiciones, permitiendo su carga dinámica dentro del programa sin necesidad de recompilación.

(2) **Diseño modular utilizando programación orientada a objetos**: Cada uno de los componentes fundamentales de la lógica difusa se encapsula en clases independientes pero interrelacionadas que hacen uso de la herencia. Este enfoque no solo favorece la modularidad y la reutilización del código, sino que también permite una clara separación de responsabilidades, facilitando tanto la comprensión como la extensión del sistema.

(3) **Evitar el uso de representaciones discretas**: En el diseño de WIFE se ha optado deliberadamente por evitar representaciones discretizadas en los modelos y estructuras difusas durante las fases de diseño y de inferencia de modelos y estructuras, priorizando el uso de representaciones continuas. Esta decisión responde a criterios de eficiencia computacional, ya que el uso de funciones de pertenencia definidas de forma continua permite evaluar directamente los grados de pertenencia sin necesidad de iteraciones extensivas.

En la siguiente sección se describe la arquitectura actual del sistema. La Sección 3 se describen los componentes que permiten crear los modelos junto con la inferencia, está centrado en modelos tipo Mamdani y TSK. Finalmente, en la Sección 4 se detallan las conclusiones y trabajo futuro.

2 Arquitectura del sistema

Cada elemento teórico se modela explícitamente para ser utilizado de manera flexible en las distintas estructuras difusas:

Operadores lógicos difusos: Se implementan operadores básicos como t-normas y t-conormas, definidos como funciones independientes y agrupados en un diccionario *Python* que permite seleccionar el operador adecuado mediante el uso de la clave, posibilitando su escalado.

Conjuntos difusos: El sistema define clases para representar conjuntos difusos mediante una clase con funciones de pertenencia parametrizables (triangulares, trapezoidales, etc.). Además, se incluyen estructuras alternativas para representar el conocimiento de forma discreta.

Variables lingüísticas: Se modelan como estructuras que contienen la información necesaria para modelar completamente su definición teórica.

Sistema de proposiciones: Se definen dos tipos de proposiciones, proposiciones atómicas y proposiciones compuestas (basadas en atómicas y otras compuestas), lo que permite construir expresiones lógicas complejas de manera estructurada y facilita el desarrollo de modelos no estándar.

3 Creación y uso de modelos difusos

En esta sección se ilustra cómo, a partir de los conceptos básicos previamente definidos, es posible construir modelos difusos completos y utilizar dichos modelos para realizar procesos de inferencia. Se describe, por tanto, no solo la creación de las distintas estructuras —variables, conjuntos y reglas—, sino también su integración en sistemas funcionales capaces de procesar entradas y generar salidas, poniendo de manifiesto el flujo completo desde la definición teórica hasta su aplicación práctica. Se dispone de las clases que permiten trabajar con los modelos tipo Mamdani y TSK, concretamente, el entorno incluye clases específicas para: (1) Creación e inferencia con modelos tipo Mamdani y; (2) Diseño e inferencia con modelos tipo TSK.

Las reglas difusas se modelan como objetos independientes que combinan un antecedente lógico estructurado y un consecuente —difuso o funcional—, lo que permite mantener un mecanismo de inferencia uniforme independientemente del modelo. En este contexto, dado que la diferencia fundamental entre los enfoques reside en el tratamiento del consecuente, se ha optado por emplear conjuntos difusos en Mamdani y funciones dependientes de las entradas en TSK. Sea la regla:

IF ((T is *Cold*) OR (T is *Medium*)) AND (H is *Low*)
THEN S is *Comfortable*

donde T, H y S son las variables lingüísticas que modelan la temperatura (T), la humedad (H) y la sensación corporal (S). Además, *Cold* y *Medium* son conjuntos difusos de la variable T, *Low* es un conjunto difuso de la variable H y *Comfortable* es un conjunto difuso de la variable S.

El siguiente código muestra cómo se crean los distintos operadores definiendo una función por cada operador y añadiendo la entrada correspondiente en el diccionario *functions* (líneas 1-11), accediendo al elemento deseado con etiqueta de la función. De las líneas 13-25 se muestra cómo se crea una regla. Se hace uso de la proposición compuesta que define el antecedente (líneas 13-15) junto con las variables lingüísticas *temp_vl*, *hum_vl* y *sensa_vl*. Se crea el consecuente como la proposición atómica S is *Comfortable*, al ser un modelo tipo Mandani, utilizando el constructor *FuzzyAP* (líneas 16 y 19). Se crea la regla mediante el constructor *MandaniRule()* que cuenta con el identificador de la regla (línea 18), la proposición difusa compuesta se crea utilizando el constructor de ésta, *PDC(ant,functions)* (línea 19), y, finalmente, la proposición difusa que modela el consecuente mediante el constructor *FuzzyAP()* (líneas 20-25). Finalmente, se crea el modelo invocando al constructor *MandaniFuzzyModel()*. La clase *MandaniFuzzyModel()* tiene implementado el proceso de inferencia. También existe la versión para modelos tipo TSK.

```
1 def and_t_norm(val1:float, val2:float):
2     return min(val1,val2)
3 def or_t_conorm(val1:float, val2:float):
4     return max(val1,val2)
5 def not_operator(val1:float):
6     return 1-val1
```

```
functions = {
    'AND': and_t_norm,
    'OR': or_t_conorm,
    'NOT': not_operator
}
ant = [
    [[temp_vl, 'Cold'], 'OR', [temp_vl, 'Medium']],
    'AND', [hum_vl, 'Low'] ]
cons = [sensa_vl, 'Comfortable']
R1 = MandaniRule(
    'R1',
    PDC(ant, functions),  # antedent
    FuzzyAP(  # consequent
        LinguisticV(sensa_vl['name'], sensa_vl['position'],
                    sensa_vl['support'], sensa_vl['sets']),
        sensa_vl['sets'][cons[1]]
    )
)
modelo = MandaniFuzzyModel([R1, R2, R3, R4, R5, R6])
```

Código 1.1. Definiendo los operadores

4 Conclusiones y trabajos futuros

Se ha presentado un entorno software para lógica difusa que integra modelos tipo Mamdani y TSK, cuya principal aportación es su carácter abierto y extensible. No impone una estructura fija, permitiendo la construcción de nuevas arquitecturas difusas a partir de componentes básicos bien definidos. Destaca su modularidad y reutilización de componentes y su adecuación para investigación y la docencia. Como trabajo futuro se añadirán nuevos modelos de inferencia, como variantes avanzadas de TSK o enfoques híbridos basados en aprendizaje automático, que permitan ajustar automáticamente los parámetros de los conjuntos difusos y las reglas a partir de datos. También se podrían mejorar las clases desarrolladas y los mecanismos de visualización e interpretación de resultados.

Acknowledgments. Los autores agradecen al departamento de Tecnologías y Sistemas de Información (TSI) de la Universidad de Castilla la Mancha (UCLM) y al proyecto ATENEA: Aprendizaje Profundo, Metaheurísticas y Conocimiento Experto para Sistemas Multiagente en Entornos Ferroviarios Liberalizados Parcialmente Observables (Referencia 2025-GRIN-38352), cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER), la financiación necesaria para la difusión de este trabajo. Además, se realizó cuando Enrique Adrian Villarrubia-Martin era becario predoctoral de la UCLM financiado por el Fondo Social Europeo Plus (FSE+).

References

1. Abelson, H., Sussman, G.J., Sussman, J.: Structure and Interpretation of Computer Programs, Second edition, MIT Press, 2010