SEP **TNM**

INSTITUTO TECNOLÓGICO DE TIJUANA

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

Métodos de representación, interpretación, búsqueda e inferencia basados en aprendizaje con redes neuronales, para analítica de datos a través de predicados de lógica difusa.

TRABAJO DE TESIS

Presentado por
CARLOS ERIC LLORENTE PERALTA

Para Obtener el Grado de
DOCTORADO EN CIENCIAS EN COMPUTACIÓN

Director de Tesis
DR. LAURA CRUZ REYES

Co-Director de Tesis
DR. RAFAEL A. ESPIN ANDRADE

Tijuana, BC, junio 2024

Tijuana, Baja California, 06/junio/2024
Oficio No. 101/DEPI/2024
Asunto: Autorización de Impresión de Tesis

MARÍA MAGDALENA SERRANO ORTEGA
JEFA DEL DEPARTAMENTO DE SERVICIOS ESCOLARES
PRESENTE

En lo referente al trabajo de tesis, "Métodos de representación, interpretación, búsqueda e inferencia basados en aprendizaje con redes neuronales, para analítica de datos a través de predicados de lógica difusa". Presentado por C. **Carlos Eric Llorente Peralta**, alumno del Doctorado en Ciencias en Computación con número de control **D20210006;** informo a usted que a solicitud del comité de tutorial, tengo a bien **Autorizar la Impresión de Tesis**, atendiendo las disposiciones de los Lineamientos para la Operación de Estudios de Posgrado del Tecnológico Nacional de México.

Sin más por el momento le envió un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica
Por una Juventud Integrada al Desarrollo de México

EDUCACIÓN
INSTITUTO TECNOLÓGICO DE TIJUANA
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

GUADALUPE HERNANDEZ ESCOBEDO
JEFE DE DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

ccp. Archivo
GHE/lap

Tijuana, B.C., 6 de Junio de 2024
Asunto: Se autoriza impresión de Trabajo de Tesis

C. DRA. MARÍA MAGDALENA SERRANO ORTEGA
JEFA DEL DEPTO. DE SERVICIOS ESCOLARES
PRESENTE.

En lo referente al trabajo de tesis escrito, con título **"Métodos de representación, interpretación, búsqueda e inferencia basados en aprendizaje con redes neuronales, para analítica de datos a través de predicados de lógica difusa"**, presentado por la ***C. LLORENTE PERALTA CARLOS ERIC*** alumno del Doctorado en Ciencias en Computación con número de control **D20210006**, informamos a usted que después de una minuciosa revisión, de acuerdo con lo establecido en el reglamento vigente para este caso, nuestro dictamen es: Se aprueba en todas sus partes, en virtud de reunir los requisitos de un trabajo de grado de Doctorado y a la vez se autoriza al interesado para que proceda de inmediato a la impresión del mismo.

ATENTAMENTE

DR. OSCAR CASTILLO LOPEZ
PRESIDENTE

DRA. DANIELA ADRIANA SANCHEZ VIZCARRA
SECRETARIO

DRA. CINTHIA PERAZA RAMIREZ
VOCAL

DRA. MARTHA PATRICIA OCHOA VELARDEZ
VOCAL

DRA. CYNTHIA IVETTE MIRAMONTES ORTEGA
VOCAL

c.c.p. Oficina de Titulación
c.c.p. División de Estudios de Posgrado e Investigación
c.c.p. Expediente
c.c.p. Interesado

EPMO/*inf

Tijuana, B.C., 6 de Junio de 2024
Asunto: Se autoriza impresión de Trabajo de Tesis

C. DR. GUADALUPE HERNÁNDEZ ESCOBEDO
JEFE DE DIVISIÓN DE ESTUDIOS DE POSGRADO E INV.
PRESENTE.

En lo referente al trabajo de tesis escrito, con título " **Métodos de representación, interpretación, búsqueda e inferencia basados en aprendizaje con redes neuronales, para analítica de datos a través de predicados de lógica difusa**", presentado por el ***C. LLORENTE PERALTA CARLOS ERIC,*** alumno del Doctorado en Ciencias en Computación con número de control **D20210006,** informamos a usted que se autoriza el escrito de tesis y se aprueba en todas sus partes, en virtud de reunir los requisitos de un trabajo de grado de Doctorado y a la vez se autoriza al interesado para que proceda de inmediato a la impresión del mismo y a presentar su examen de grado, ya que cumple con todos los requisitos.

ATENTAMENTE

DR. OSCAR CASTILLO LOPEZ
PRESIDENTE

DRA. DANIELA ADRIANA SANCHEZ VIZCARRA
SECRETARIO

DRA. CINTHIA PERAZA RAMIREZ
VOCAL

DRA. MARTHA PATRICIA OCHOA VELARDEZ
VOCAL

DRA. CYNTHIA IVETTE MIRAMONTES ORTEGA
VOCAL

c.c.p. Oficina de Titulación
c.c.p. División de Estudios de Posgrado e Investigación
c.c.p. Expediente
c.c.p. Interesado

EPMO/*inf

DECLARACIÓN DE ORIGINALIDAD

Tijuana, BC., 20 Mayo de 2024,

Yo, **Carlos Eric Llorente Peralta**, estudiante del Doctorado en Ciencias en Computación, en mi calidad de autor manifiesto que este documento de tesis es producto de mi trabajo original y que no infringe los derechos de terceros, tales como derechos de publicación, derechos de autor, patente y similaridad. Por lo tanto, la obra realizada es de mi exclusiva autoría y no infringí en copiar el texto o imágenes, de fuentes de información por lo cual soy responsable del escrito que aquí se presenta.

Así mismo, declaro que en las citas textuales que he incluido (las cuales aparecen entre comillas) y en los resúmenes que he realizado de publicaciones ajenas, indico explícitamente los datos de los autores y las publicaciones.

En caso de presentarse cualquier reclamación o acción por parte de terceros en cuanto a los derechos de autor sobre la obra en cuestión, acepto toda la responsabilidad de tal infracción y relevo de esta a mi director de tesis, así como al Tecnológico Nacional de México, al Instituto Tecnológico de Tijuana y a sus respectivas autoridades.

M.C.C. Carlos Eric Llorente Peralta
Estudiante de Doctorado en Ciencias en Computación

CARTA DECLARACION DE PROPIEDAD INTELECTUAL

Tijuana BC a 20 de Mayo de 2024

Yo **Carlos Eric Llorente Peralta** reconozco que el Trabajo de Tesis Doctoral que realice durante mis estudios en el Doctorado en Ciencias en Computación del Instituto Tecnológico de Tijuana fue parte del Proyecto de Investigación Titulado: **Analítica de Datos a través de predicados de Lógica Difusa Compensatoria y Arquimediano-Compensatoria**, que desarrollan mi Director de tesis la **Dra. Laura Cruz Reyes** y mi Codirector el **Dr. Rafael A. Espín Andrade**, y del cual son responsables del proyecto de investigación. Por esta razón, los métodos, modelos, algoritmos, y software realizados, así como datos y resultados obtenidos durante el desarrollo de mi tesis doctoral son propiedad intelectual de mi Director y Codirector de Tesis, del Tecnológico Nacional de México, Instituto Tecnológico de Tijuana y del Conacyt, y No podré utilizarlos por mi cuenta durante, Ni después de terminar mi beca o estudios, excepto a solicitud escrita para poder utilizarlos bajo una colaboración directa con mi Director y Codirector los cuales son responsable del proyecto de investigación. Por tanto, estoy de acuerdo en que No podré utilizar ni tomar modelos, ni datos utilizados en este proyecto de investigación y en el desarrollo de tesis para: presentaciones, publicaciones ni desarrollo de mi propia investigación que pudiera desarrollar una vez concluido mis estudios.
Atentamente

M.C.C Carlos Eric Llorente Peralta

Resumen

Este estudio presenta un nuevo modelo de red neuronal híbrida que incorpora una arquitectura de red neuronal de propagación hacia adelante. Este modelo reemplaza los conceptos clásicos de peso, sesgo, funciones lineales y de activación por conceptos generalizados de lógica difusa Arquimediana compensatoria. El enfoque propuesto se aplica a modelos de analítica de datos.

En la literatura, los modelos que permiten descubrir predicados lógicos a través del aprendizaje con redes neuronales presentan ventajas como su flexibilidad y alta capacidad de aprendizaje. Tales modelos ofrecen una descripción rica y detallada de los datos, aunque su interpretación puede ser difícil. Sin embargo, existen enfoques que facilitan esta interpretación y permiten el descubrimiento de relaciones desconocidas en los datos, además de ser altamente escalables.

Por otro lado, el enfoque de usar redes neuronales para aprender predicados que permitan resolver problemas de análisis de datos puede ser costoso computacionalmente y requiere una gran cantidad de datos para evitar el sobreajuste. A pesar de que existen métodos para mejorar la interpretabilidad, esta puede ser limitada y no permite comprender completamente por qué el modelo toma cada decisión particular. Además, el rendimiento del modelo depende en gran medida de la correcta configuración de los hiperparámetros, lo que puede resultar en una configuración general poco clara.

Dadas las ventajas y desventajas de estos modelos, se propone la red neuronal híbrida, que aprovecha las ventajas mencionadas anteriormente y combina una arquitectura de red neuronal de propagación hacia adelante con una teoría lógica difusa Arquimediana compensatoria.

La lógica difusa Arquimediana compensatoria es compatible con la manera clásica en que la lógica difusa trabaja con la interpretación expresada en lenguaje natural. Además, tiene una gran capacidad de modelado de datos y es adecuada para procesos de clasificación basados en la descripción de las relaciones entre las características de un conjunto de datos mediante predicados de identidad.

Para evaluar la red propuesta de manera objetiva, se utilizan diversos conjuntos de datos para el entrenamiento del modelo y la resolución de problemas de clasificación e inferencia. Se emplean características de la lógica difusa Arquimediana compensatoria, como una función que genera una familia de funciones de membresía llamada variable lingüística continua generalizada y un modificador lingüístico que intensifica la morfología de la función a la que se aplica. Además, se utilizan conceptos de conjunción y disyunción para crear una variedad de predicados lógicos.

Los resultados de los experimentos muestran que el modelo propuesto presenta una interpretabilidad no vista en modelos similares y permite expresar los resultados en lenguaje

natural mediante la traducción de la semántica de la red. A diferencia de otros modelos interpretables, el modelo propuesto no sacrifica la precisión y logra niveles competitivos de efectividad en procesos de clasificación e inferencia.

Además, el modelo aborda la opacidad característica de las arquitecturas de redes neuronales, lo que permite obtener un modelo justificable a partir de los resultados presentados. Esta característica es altamente valorada en procesos de análisis de datos.

En conclusión, el modelo propuesto introduce una teoría lógica altamente adaptable e interpretable que, junto con una arquitectura de propagación hacia adelante precisa y sencilla, permite encontrar soluciones a problemas de análisis de datos de manera interpretable y expresable en lenguaje natural a partir de predicados de lógica difusa Arquimediana compensatoria. Constituyendo una característica única en comparación con modelos similares estudiados, sin sacrificar la precisión, como es común en modelos interpretables.

Abstract

This study presents a new hybrid neural network model that incorporates a feed-forward neural network architecture. This model replaces the classical concepts of weight, bias, linear, and activation functions with generalized concepts of compensatory Archimedean fuzzy logic. The proposed approach is applied to data analytics models.

In the literature, models that allow logical predicates to be discovered through learning with neural networks have advantages such as flexibility and high learning capacity. These models provide a rich and detailed description of the data, although their interpretation can be difficult. However, there are approaches that facilitate this interpretation and allow the discovery of unknown relationships in the data, in addition to being highly scalable.

On the other hand, the approach of using neural networks to learn predicates to solve data analysis problems can be computationally expensive and requires a large amount of data to avoid overfitting. Although there are methods to improve interpretability, this can be limited and does not allow a complete understanding of why the model makes each particular decision. Furthermore, model performance is highly dependent on correct hyperparameter settings, which can result in unclear overall settings.

Given the advantages and disadvantages of these models, the hybrid neural network is proposed, which takes advantage of the aforementioned benefits and combines a feed-forward neural network architecture with the logic theory of compensatory Archimedean fuzzy logic.

Compensatory Archimedean fuzzy logic is compatible with the classical way in which fuzzy logic works, with interpretation expressed in natural language. Additionally, it has great data modeling capabilities and is suitable for classification processes based on the description of the relationships between the characteristics of a data set using identity predicates.

To evaluate the proposed network objectively, various data sets are used for training the model and solving classification and inference problems. Features of compensatory Archimedean fuzzy logic are used, such as a function that generates a family of membership functions called a generalized continuous linguistic variable and a linguistic modifier that intensifies the morphology of the function to which it is applied. Additionally, concepts of conjunction and disjunction are used to create a variety of logical predicates.

The results of the experiments show that the proposed model presents an interpretability not seen in similar models and allows the results to be expressed in natural language by translating the semantics of the network. Unlike other interpretable models, the proposed model does not sacrifice accuracy and achieves competitive levels of effectiveness in classification and inference processes.

Furthermore, the model addresses the opacity characteristic of neural network architectures, allowing a justifiable model to be obtained from the results presented. This feature is highly valued in data analysis processes.

In conclusion, the proposed model introduces a highly adaptable and interpretable logical theory that, together with a precise and simple feed-forward architecture, allows finding solutions to data analysis problems in an interpretable and expressible way in natural language from predicates of compensatory Archimedean fuzzy logic. This constitutes a unique characteristic compared to similar models studied, without sacrificing precision, as is common in interpretable models.

Tabla de contenido

Índice de figuras

Índice de tablas

Índice de ecuaciones

Capítulo 1: Introducción

En la actualidad, vivimos en un mundo tan interconectado que se genera una cantidad masiva de datos en el día a día. Se estima que para el año 2025, esta cifra alcance los 181 zettabytes. Surge así la necesidad de desarrollar modelos computacionales que puedan filtrar, analizar, modelar y extraer información importante, precisa y veraz, con el fin de ayudar en procesos de toma de decisiones, mejora de la eficiencia, la innovación y el desarrollo, entre otros aspectos. Es de tal forma que surge el enfoque de la analítica de datos.

El creciente estudio y desarrollo de modelos enfocados en el análisis de datos ha traído consigo grandes avances en áreas como la ingeniería, la medicina, los negocios y la seguridad, entre otras. Dentro de tales modelos, las redes neuronales han tenido un impacto significativo, gracias a sus resultados altamente precisos y su capacidad de aprendizaje y adaptación a los datos. Específicamente, las redes neuronales que implementan lógica difusa destacan, ya que permiten al modelo manejar la incertidumbre y datos vagos e incompletos, características comunes en problemas del mundo real.

Sin embargo, las redes neuronales de lógica difusa, o redes neuro difusas, todavía enfrentan algunas limitaciones. Por ejemplo, la implementación de estos modelos puede resultar conceptualmente difícil, lo que dificulta su comprensión e interpretación. Además, suelen requerir muchos recursos computacionales, tiempo de ejecución, tal que a menudo necesitan modelos de optimización especializados debido a la complejidad de los parámetros involucrados.

Por tanto, en el presente proyecto de tesis se presentan una serie de conceptos que buscan, a través del análisis de diversos enfoques, construir una red neuronal de lógica difusa Arquimediana compensatoria. Esta red, mediante el aprendizaje de predicados lógicos, pretende resolver problemas de analítica de datos a través de un modelo interpretable y preciso, evitando así el uso de recursos extra lógicos, lo cual facilita su interpretación semántica.

De esta manera, se contribuye a la creación de modelos no solo precisos en tareas de clasificación e inferencia, sino también interpretables y explicables en función del lenguaje natural.

1.1 Descripción del problema de investigación

El descubrimiento de conocimiento representa un gran desafío en los campos de la inteligencia artificial y el procesamiento de datos, dado su papel fundamental en la toma de decisiones y el manejo de datos no estructurados en contextos difusos, así como ambiguos. Bajo este contexto, se identifica la necesidad de superar limitaciones existentes en la representación y el procesamiento de datos difusos.

Para abordar este problema, se propone la hibridación de dos metodologías: la lógica difusa Arquimediana compensatoria y el aprendizaje con redes neuronales. Con esta combinación se busca extraer conocimiento a través de predicados de lógica difusa Arquimediana compensatoria, utilizando un sistema que demuestra una mayor capacidad para comprender y manipular datos de manera precisa, rápida y escalable, en comparación con las soluciones actualmente disponibles en la literatura científica.

La pregunta central que guía este estudio es la siguiente:

¿Cómo se puede hibridar la Lógica Difusa Arquimediana Compensatoria y el Aprendizaje con redes neuronales para lograr resultados competitivos con los de la literatura relacionada?

En particular se buscan buenos resultados en términos de:

1. Descubrimiento de Predicados:

Exhaustividad: El estudio busca desarrollar un algoritmo que descubra una amplia gama de predicados que reflejen con precisión el conocimiento presente en datos difusos.

Diversidad: Se aspira a generar predicados diversos que aborden diferentes aspectos del problema, evitando la redundancia y permitiendo la representación del conocimiento.

2. Inferencia de Datos:

Exactitud: El enfoque propuesto debe permitir la inferencia precisa de datos a través de los predicados identificados, minimizando los errores y mejorando la confiabilidad de los resultados.

1.2 Hipótesis

Sustituyendo en las redes neuronales artificiales, el uso tradicional de la suma ponderada, los pesos y las funciones de activación, por conjunciones o disyunciones, modificadores lingüísticos y una variable lingüística continua generalizada pertenecientes a una Lógica Difusa Arquimediana Compensatoria; es posible representar predicados de lógica difusa Arquimediana compensatoria como modelos de aprendizaje con redes neuronales resultando en un modelo competitivo con estado del arte en términos de:

1. Exhaustividad y diversidad en la búsqueda de predicados de Lógica Difusa.

2. Exactitud en la Inferencia de Datos a través de los predicados encontrados.

1.3 Justificación

En la actualidad, el campo de los modelos de aprendizaje con redes neuronales se encuentra en constante evolución y desarrollo. Sin embargo, enfrenta dos desafíos que requieren una atención especial. En primer lugar, el proceso de entrenamiento realizado por las redes neuronales es conocido por ser demasiado complejo, lo que implica una inversión significativa de tiempo y recursos computacionales. En segundo lugar, las redes neuronales tienen por lo general poca o nula interpretabilidad, lo cual significa que, si bien pueden ofrecer resultados muy cercanos al óptimo, su funcionamiento interno es oscuro y difícil de entender (Das et al., 2020).

En relación con el proceso de optimización, a pesar de los avances en las metodologías basadas en cálculos utilizadas para el entrenamiento de redes neuronales, ninguna de ellas ofrece una garantía en su proceso de optimización. Por otro lado, estas metodologías a menudo muestran sensibilidad al ruido en los datos y no son adecuadas para manejar datos heterogéneos o incompletos, lo que limita su aplicabilidad a situaciones del mundo real (Das et al., 2020).

Por otro lado, la interpretación de las decisiones tomadas por las redes neuronales es un tema de largo estudio entre los investigadores. A lo largo de décadas, se han propuesto diversos métodos, tanto teóricos como experimentales, para abordar este problema (Fan et al., 2021; Hakkoum et al., 2022; Vidal, 1994).

Una estrategia prometedora para enfrentar tales desafíos es la lógica difusa, ya que numerosas investigaciones han demostrado que puede mejorar el rendimiento de los modelos de aprendizaje con redes neuronales en situaciones donde los datos son ruidosos, heterogéneos, incompletos o vagos. (Das et al., 2020; Ojha et al., 2017).

Los sistemas difusos, que se basan en funciones de membresía, emplean dos métodos de aproximación para aprender estas funciones. El primero describe las funciones de membresía mediante parámetros que se optimizan durante el proceso de aprendizaje. El segundo método implica que la red neuronal aprenda a generar valores de membresía en función de las entradas dadas y utilizando un conjunto de datos de entrenamiento. A menudo se prefiere el primer enfoque debido a la falta de conocimiento explícito de las funciones de membresía en el segundo (Rivas-Asanza et al., 2018).

Por todas estas razones, se propone llevar a cabo un proyecto de investigación en el que se explora la hibridación de dos modelos cognitivos: las redes neuronales, en particular el modelo propagación hacia adelante debido a su simplicidad y desempeño en tareas de aprendizaje automático, y una innovadora teoría lógica conocida como lógica difusa Arquimediana compensatoria (de Campos Souza, 2020; Espín-Andrade et al., 2015).

Hasta la fecha, no se ha encontrado en las investigaciones publicadas, ninguna investigación que haga uso de la hibridación de ambos conceptos, y como resultado, no existe información disponible sobre el rendimiento que podrían lograr en conjunto.

La lógica difusa Arquimediana compensatoria es una teoría lógica que se caracteriza por su capacidad de adaptarse a datos diversos, al mismo tiempo que otorga interpretabilidad y precisión al sistema. Además, la lógica difusa Arquimediana compensatoria permite definir funciones de membresía para los datos de entrada a través de la optimización de un conjunto de parámetros, lo que resulta de particular interés, ya que los datos de entrada se tratan a través de una variable lingüística continua generalizada que permite crear una familia de funciones mediante la variación de los parámetros que la componen (Espín-Andrade et al., 2021).

Se espera que esta unión de conceptos permita aprovechar las ventajas de ambas metodologías: la capacidad de las redes neuronales de propagación hacia adelante para generar modelos interpretables y precisos y la adaptabilidad de la lógica difusa Arquimediana compensatoria a datos diversos y la inherente interpretabilidad de los modelos difusos. Además, esta combinación podría permitir la modelización de una amplia gama de funciones de membresía a través de la optimización de parámetros de una variable lingüística continua generalizada.

1.4 Objetivos

En esta sección se presenta el objetivo general del desarrollo de la siguiente investigación, así como los objetivos específicos planteados.

1.4.1 Objetivo general

Desarrollar un modelo híbrido y un algoritmo de aprendizaje con redes neuronales basado en lógica difusa Arquimediana compensatoria que incorpore estrategias de optimización relacionadas con la representación, búsqueda y uso de predicados en modelos de inferencia, con el objetivo de lograr resultados competitivos en términos de exhaustividad y diversidad en la búsqueda de predicados, así como en términos de exactitud e interpretabilidad en procesos de inferencia.

1.4.2 Objetivos específicos

1. Formular una lógica difusa Arquimediana compensatoria que sirva como base para el proceso de descubrimiento de datos.

2. Realizar un análisis de los modelos de aprendizaje con redes neuronales existentes en la literatura y seleccionar uno basado en sus ventajas.

3. Analizar las diferentes estructuras de representación de predicados existentes en la literatura.

4. Integrar la red neuronal artificial seleccionada en un algoritmo de búsqueda de predicados e inferencia de datos, utilizando lógica difusa Arquimediana compensatoria.

5. Evaluar la propuesta híbrida y comparar los resultados obtenidos con los resultados presentados en otros artículos que emplean estrategias similares.

1.5 Aportaciones

Las principales aportaciones presentadas en esta tesis son las siguientes:

I. **Sistema axiomático de predicados de lógica difusa Arquimediana compensatoria**
   a. Propone una teoría lógica denominada lógica difusa Arquimediana compensatoria.
   b. Permite descubrir predicados que resuelven problemas de clasificación en procesos de analítica de datos.
   c. Facilita la interpretación y explicación de los predicados de lógica difusa Arquimediana compensatoria descubiertos.
   d. Proporciona una base teórica robusta para el desarrollo de algoritmos basados en lógica difusa.

II. **Sistema axiomático para la interpretación de predicados de una lógica difusa Arquimediana compensatoria**
   a. Permite caracterizar una variable lingüística continua generalizada de una lógica difusa Arquimediana compensatoria.
   b. Facilita la interpretación de una variable lingüística continua generalizada en función de sus preimágenes.
   c. Permite expresar estas interpretaciones en lenguaje natural.
   d. Mejora la capacidad de comunicación de los resultados de modelos difusos a usuarios no expertos.

III. **Red neuronal híbrida de lógica difusa Arquimediana compensatoria**
   a. Introduce un modelo híbrido de red neuronal de propagación hacia adelante y lógica difusa Arquimediana compensatoria.
   b. Implementa el concepto de modificador lingüístico en una red neuronal difusa en lugar del concepto de peso.
   c. Propone la representación de un predicado en forma normal disyuntiva a través de la arquitectura de la red neuronal propuesta.
   d. Desarrolla una red neuronal interpretable en función de las preimágenes generadas para cada variable lingüística continua generalizada en la red neuronal propuesta.
   e. Propone una forma de expresar los predicados descubiertos en lenguaje natural.
   f. Aumenta la capacidad de la red neuronal para manejar datos imprecisos y ambiguos, mejorando la robustez y la adaptabilidad del modelo.

1.6 Producción científica

A continuación, se enlistan los artículos científicos generados por la investigación realizada en este trabajo de tesis:

Espín-Andrade, R. A., Cruz-Reyes, L., Llorente-Peralta, C., González-Caballero, E., Pedrycz, W., & Ruiz, S. (2021). Archimedean Compensatory Fuzzy Logic as a Pluralist Contextual Theory Useful for Knowledge Discovery. International Journal of Fuzzy Systems, 1–21. https://doi.org/10.1007/s40815-021-01150-6.

Ruiz, Susana., Espín-Andrade, R. Alejandro., Cruz-Reyes, Laura., & Llorente-Peralta, C. Eric. (s/f). Characterization of the Generalized Continuous Linguistic Variable defined in the Archimedean Compensatory Fuzzy Logic. *International Journal of Fuzzy Systems*. (En evaluación para su publicación)

Así mismo se enlistan los capítulos de libro desarrollados a partir de la siguiente tesis:

Llorente-Peralta, C. E., Cruz-Reyes, L., Espín-Andrade, R. A., & Padron-Tristan, J. F. (2023). Interpretability of an Archimedean Compensatory Fuzzy Logic in Data Analytics: Some Case Studies (pp. 237–252). https://doi.org/10.1007/978-3-031-28999-6_15

Llorente-Peralta, C. E., Cruz-Reyes, Laura., & Espín-Andrade, R. A. (2021). Knowledge Discovery Using an Evolutionary Algorithm and Compensatory Fuzzy Logic. In Oscar. Castillo & Patricia. Melin (Eds.), Fuzzy Logic Hybrid Extensions of Neural and Optimization Algorithms: Theory and Applications. (Studies in, Vol. 940, pp. 363–283). Springer, Cham. https://doi.org/https://doi.org/10.1007/978-3-030-68776-2_21

De igual forma se enlistan los diferentes congresos en los cuales se presentaron diversos avances y descubrimientos resultantes de los resultados y experimentos realizados para este trabajo de tesis:

"Archimedean compensatory fuzzy logic for data analytics: some cases of study" en International Seminar of Computational Intelligence. Agosto, 2022.

"Knowledge discovery through an Archimedean compensatory fuzzy logic neural network" en International Workshop on Numerical and Evolutionary Optimization. Noviembre 2022.

"Archimedean compensatory fuzzy logic neural network" en International Seminar of Computational Intelligence. Agosto, 2023.

"Archimedean compensatory fuzzy logic neural network" en International Workshop on Artificial Intelligence and Analytics. Noviembre, 2023.

"Aplicación de redes neuronales en la creación de reglas que ayudan en la toma de decisiones" en Seminario de Computo Inteligente. Octubre, 2023.

1.7 Organización del documento

La estructura del presente documento de tesis se describe a continuación:

Capítulo 2: Marco Teórico En este capítulo se presentan los conceptos básicos necesarios para comprender el proyecto de tesis propuesto. Entre os cuales se incluyen:

a) Analítica de Datos: Conceptos fundamentales.
b) Redes Neuronales: Principios y conceptos fundamentales.

c) Lógica Difusa: Fundamentos y principales vertientes.
d) Lógica Difusa Arquimediana Compensatoria: Introducción a la teoría propuesta.

Capítulo 3: Estado del Arte Este capítulo ofrece un estudio de los trabajos relacionados más relevantes, organizados de la siguiente manera:

a) Revisión de la Literatura: Análisis de estudios previos y su contribución.
b) Identificación de Problemas y Oportunidades: Problemas observados en la literatura y áreas de oportunidad.
c) Comparación y Originalidad del Proyecto: Características esenciales y únicas de cada trabajo analizado, destacando la originalidad del proyecto de esta tesis.

Capítulo 4: Métodos de Representación, Interpretación, Búsqueda e Inferencia de predicados Este capítulo detalla los principales desarrollos de la tesis, organizados por metodologías:

I. Lógica Difusa Arquimediana Compensatoria
    a) Descubrimiento de Predicados: Búsqueda exhaustiva e intensiva de predicados mediante un algoritmo genético para resolver problemas de clasificación en procesos de analítica de datos, representándolos como árboles basados en programación genética.
    b) Interpretación en Lenguaje Natural: Metodología para interpretar predicados utilizando funciones de membresía de la lógica difusa Arquimediana compensatoria.
II. Variable Lingüística Continua Generalizada
    a) Implementación en Redes de propagación hacia adelante: Uso como función de activación y evaluación de su impacto.
    b) Caracterización y Expresión: Caracterización de las variables lingüísticas continuas generalizadas y como consecuencia la capacidad de expresarse en lenguaje natural.
III. Red Neuronal Híbrida de Lógica Difusa Arquimediana Compensatoria
    a) Modelo Híbrido: Integración de la red neuronal de propagación hacia adelante con lógica difusa Arquimediana compensatoria.
    b) Representación de Predicados: Propuesta de representación en forma normal disyuntiva a través de la arquitectura de la red neuronal.
    c) Interpretabilidad y Expresión en Lenguaje Natural: Desarrollo de una red neuronal interpretable y expresión de predicados en lenguaje natural.
    d) Solución de Problemas de Analítica de Datos: Resolución de problemas mediante procesos de clasificación e inferencia, logrando una exactitud competitiva en términos de predicciones.

Capítulo 5: Conclusiones y Trabajo Futuro Este capítulo presenta las conclusiones alcanzadas y sugiere futuras líneas de investigación:

a) Conclusiones: Resumen de los resultados obtenidos y evaluación del cumplimiento de objetivos e hipótesis.
b) Trabajo Futuro: Propuestas para futuras investigaciones y desarrollo continuo del proyecto

Capítulo 2: Marco teórico

En este capítulo, se describen aquellos métodos y conceptos que forman parte del modelo de solución definido en capítulos posteriores, se espera que la información detallada permita facilitar la comprensión y asimilación del proyecto de manera fluida.

2.1 Analítica de datos

En la actualidad, vivimos en un mundo donde se genera un tráfico de información masivo a nivel mundial, esta información es generada a través del uso de dispositivos móviles, dispositivos inteligentes, sistemas de comunicación, de control, entre otros. De tal manera que en el año 2022 se generaron un total de 97 zettabytes de información, cifra que se espera crezca hasta los 181 zettabytes en 2025 (DOMO, 2022).

Esta gran cantidad de datos son usados para satisfacer las necesidades de información de los usuarios que las utilizan. Ya que cuando se obtiene un conocimiento preciso y objetivo, es posible realizar una mejor toma de decisiones basada en datos y en la experiencia.

Con el fin de manejar la gran cantidad de información generada en el mundo, constantemente se desarrollan modelos y herramientas que permiten gestionar los datos de manera sistémica y eficiente. Dentro de estos enfoques encontramos la *analítica de datos*, enfoque en el cual los datos son tratados y transformados en información útil y relevante, al mismo tiempo, se busca la metodología que presente una mayor eficacia en el proceso de análisis de los datos (Lugo Cabrera & López Herrera, 2018).

Valencia Asqui define a la analítica de datos como "*la ciencia de recogida, almacenamiento, extracción, limpieza, transformación, agregación y análisis de datos, con el fin de descubrir información y conocimiento*" (Valencia Asqui, 2017). Algunos de los conceptos clave de la analítica de datos, son los siguientes:

Big data: El termino, hace referencia a conjuntos de información masivos que, debido a su magnitud, resultan difíciles de gestionar y analizar mediante métodos convencionales. Estos conjuntos de datos suelen acumularse a lo largo del tiempo y presentan desafíos en su gestión a través de las herramientas tradicionales de bases de datos. Big Data no solo describe la información en sí, sino también las herramientas, procesos y procedimientos diseñados para organizar, crear, manipular y administrar eficientemente estas grandes cantidades de datos (Camargo-Vega et al., 2015).

Minería de datos: Se define como el proceso de seleccionar, explorar, modificar, modelizar y valorar almacenes de datos, con el fin de descubrir patrones desconocidos. Lo cual permite predecir nuevas perspectivas y pronosticar situaciones futuras (Diazaraque, 2009).

Aprendizaje automático: En este paradigma, el objetivo principal es generar modelos de manera automática, los cuales representan o describen el funcionamiento de un sistema o concepto a través de reglas, fórmulas o patrones. Tales modelos permiten construir cosas o partes de una cosa basándose en datos disponibles. El aprendizaje de estos modelos se lleva a cabo mediante la construcción de sistemas informáticos que mejoran automáticamente con la experiencia a partir de un modelo de aprendizaje (Oladipupo, 2010).

La analítica de datos puede ser clasificada según el proceso de análisis aplicado a la información. Lugo Cabrera y López Herrera proponen los siguientes enfoques (Lugo Cabrera & López Herrera, 2018):

Analítica descriptiva: Se examinan y analizan los datos históricos como un medio para afrontar los sucesos futuros, es decir, se examinan datos históricos con el fin de extraer información que permite entender los resultados alcanzados, esta información es representada mediante estimadores, gráficas y/o tablas y permite representar su comportamiento y tendencia. En términos de valor se dice que la analítica descriptiva tiene un valor mínimo, ya que requiere un conjunto básico de habilidades.

Diagnóstico de datos: Se utilizan técnicas y métodos que permiten describir la información y las relaciones entre sus elementos. En este proceso se determinan los datos necesarios y los métodos útiles para encontrar información relevante dentro de los datos. Mediante el diagnóstico de datos, se observan posibles problemas y sus consecuencias, se establecen prioridades, así como metas y objetivos. En términos de valor, el diagnóstico de datos es superior a la analítica descriptiva y hace uso de un conjunto de habilidades avanzadas.

Analítica predictiva: Hace uso de técnicas de minería de datos con el fin de reunir y tratar información disponible permitiendo realizar predicciones sobre sucesos futuros. El uso de las técnicas de minería de datos sobre datos históricos permite la construcción de modelos de inteligencia predictiva, descubriendo de esta manera tendencias y relaciones en conjuntos de datos tanto estructurados como no estructurados. En términos de valor este proceso es superior al diagnóstico de datos.

Análisis prescriptivo: Además de realizar un análisis predictivo, realiza sugerencias al tomador de decisiones con el fin de beneficiarse de las predicciones obtenidas. Mediante el análisis prescriptivo se sintetizan automáticamente grandes datos, aplica ciencias matemáticas y utiliza aprendizaje automático para hacer predicciones y luego sugiere opciones de decisión para aprovechar las predicciones. La analítica prescriptiva procesa de manera continua y automática nuevos datos, lo cual permite mejorar la precisión de la predicción y proporcionar mejores opciones de decisión. En términos de valor, este proceso tiene la escala más alta en la analítica de datos.

En la Figura 2.1 se observa de manera gráfica esta clasificación.

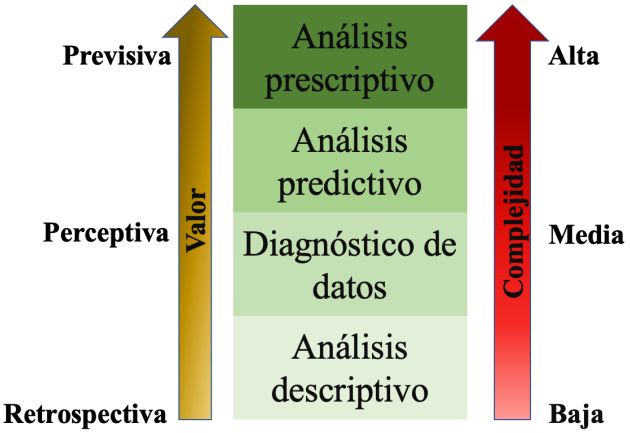


Figura 2.1: Representación gráfica de la clasificación de la analítica de datos

2.2 Redes neuronales artificiales

Durante décadas, se ha estudiado el funcionamiento del cerebro y la aplicación de este conocimiento a modelos computacionales. En la historia de este proceso, destacan investigadores como Alan Turing, quien en 1936 plantea estudiar el cerebro desde un punto de vista computacional, y McCulloch y Pitts, quienes en 1943 establecieron los primeros fundamentos de la computación neuronal. En la Figura 2.2 se muestra la neurona artificial definida por McCulloch y Pitts, cuyo desarrollo continuó con estudios de Hebb, Lashley, Rosenblatt, entre otros (Andrade Tepán, 2013).

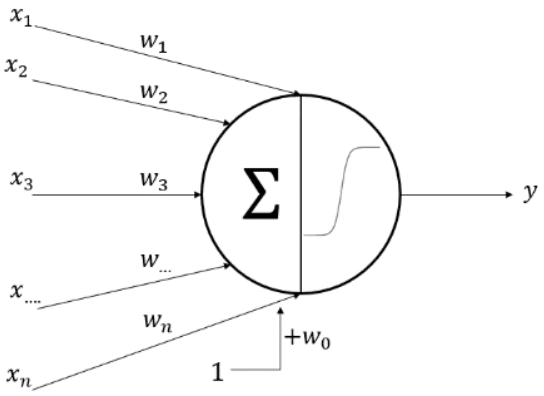


Figura 2.2: Representación de la neurona artificial concebido por McCulloch and Pitts

De acuerdo con Matich se puede definir a las *redes neuronales artificiales* (ANN por sus siglas en Inglés) como "*Redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico"* (Matich, 2001)

Dentro de las ventajas que representa el uso de las ANN se tienen el aprendizaje adaptativo, la autoorganización, la tolerancia a fallos, la operación en tiempo real y la fácil inserción dentro de la tecnología existente.

Entre las desventajas se encuentran la complejidad del proceso de aprendizaje, los tiempos prolongados necesarios para entrenar, la falta de capacidad para interpretar los resultados y el tiempo de convergencia. Es decir, se requiere un gran número de ejemplos en el proceso de entrenamiento (Rivas-Asanza et al., 2018).

Las ANN se clasifican de acuerdo con su estructura como *redes neuronales unidireccionales*, en las cuales las neuronas se agrupan en capas. Reciben señales que fluyen desde la capa de entrada hasta la capa de salida mediante conexiones de una sola dirección (Ferreyra, 2005). O como *redes neuronales recurrentes*, en las cuales las salidas de algunas de las neuronas retroalimentan a neuronas de la misma capa o a capas precedentes, siendo así que las señales fluyen en ambas direcciones (Ferreyra, 2005).

También es posible clasificarlas de acuerdo con el tipo de algoritmo de aprendizaje que es utilizado en el proceso de entrenamiento, dentro de los cuales se tienen el algoritmo de aprendizaje supervisado, el algoritmo de aprendizaje no supervisado y aprendizaje reforzado, entre otros.

Las ANN ha sido usados en diferentes procesos con resultados satisfactorios, algunos de los cuales son el reconocimiento del habla, el reconocimiento de sentimientos, análisis de imágenes, entre otros (Andrade Tepán, 2013).

La arquitectura de red neuronal de propagación hacia adelante es una de las más utilizadas en la literatura científica debido a su sencillez y eficacia. Su diseño permite una implementación rápida, lo que la hace atractiva para una variedad de aplicaciones. Además, la red de propagación hacia adelante ha demostrado un excelente desempeño en términos de precisión y velocidad de ejecución en comparación con otras arquitecturas presentes en la literatura. Esta combinación de simplicidad y rendimiento la convierte en una opción popular para una amplia gama de problemas de aprendizaje automático y análisis de datos .

2.2.1 Red neuronal de propagación hacia adelante

Las redes neuronales de propagación hacia adelante (Feed-Forward en Inglés), son uno de los enfoques más precisos y utilizados en el ámbito del análisis de datos. Estas consisten en el uso de la definición matemática de una neurona, la cual busca emular el funcionamiento de una neurona natural.

Estas neuronas están organizadas en una serie de capas. La primera capa es llamada capa de entrada o "input layer" en inglés, la última capa se le denomina capa de salida u "output layer" en inglés. A las capas existentes entre estas se les denominan capas ocultas o "hidden layer" en inglés. El término "propagación hacia adelante" se atribuye al hecho de que la información en esta estructura se mueve en una sola dirección, hacia adelante. Comenzando en la capa de

entrada, recorriendo la estructura hasta la capa de salida (Ketkar & Moolayil, 2021; Svozil et al., 1997).

Una red neuronal se puede definir como una función $f_\theta: x \to y$, donde $x \in \mathbb{R}^n$ representa un conjunto de entradas y $y \in \mathbb{R}^m$ es la salida producida. El comportamiento de esta función está parametrizado mediante $\theta \in \mathbb{R}^p$, que son los parámetros que controlan la estructura y el funcionamiento de la red, tales parámetros se ajustan durante el proceso de entrenamiento para minimizar la diferencia entre las salidas predichas y las salidas reales (Ketkar & Moolayil, 2021).

La unidad fundamental de este modelo es la neurona, la cual se describe formalmente como una función de mapeo *T*, que asigna a cada neurona *i* un subconjunto $T(i) \subseteq V$, donde *V* es el conjunto que consta de todos los ancestros de la neurona dada.

La conexión entre las neuronas *i-ésima* y *j-ésima* se caracteriza por el coeficiente de peso $w_{i,j}$, y la neurona *i-ésima* por el coeficiente de umbral b_i. El coeficiente de peso refleja el grado de importancia de la conexión dada en la red neuronal (Svozil et al., 1997).

El valor de salida *i-ésima* de la neurona, x_i, está determinado por las siguientes ecuaciones:

$$x_i = f(z_i) \tag{2.1}$$

$$z_i = \sum_{j \in T_i^{-1}} w_{i,j} x_j + b_i \tag{2.2}$$

En esta descripción, la *ec. 2.2* denota el potencial de la *i-ésima* neurona, mientras que la función $f(z_i)$ en *ec 2.1* conocida como la función de activación, introduce no linealidad al modelo. El coeficiente de umbral se define como el peso de la conexión con la neurona *j*, donde $x_j = 1$, tambien conocida como sesgo (Ketkar & Moolayil, 2021). En la Figura 2.3, se observa el funcionamiento de la neurona.

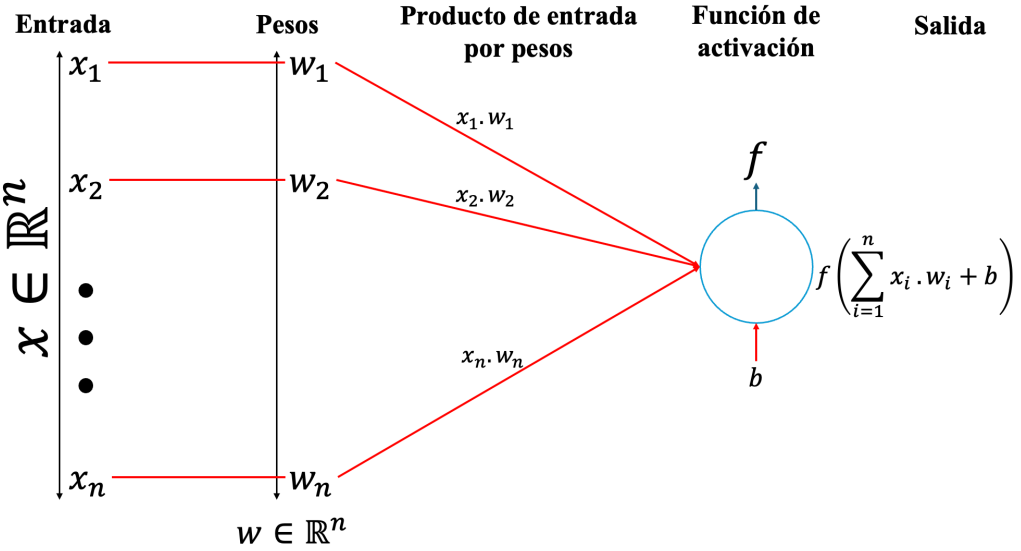


Figura 2.3: Representación de una neurona artificial

Existe una gran cantidad de funciones de activación definidas para una red neuronal de propagación hacia adelante, entre las cuales tenemos las siguientes:

ReLU: Esta función, definida como $ReLU(a) = max(0, a)$, tiene características positivas significativas:

- Alivia el problema del gradiente evanescente al no estar limitado en al menos una dirección;
- Facilita la codificación dispersa ya que el porcentaje de neuronas que están realmente activas al mismo tiempo suele ser muy bajo (Apicella et al., 2021).

Softplus: Puede verse como una aproximación suave de la función ReLU. La forma suave, así como la falta de puntos de no diferenciación podrían sugerir un mejor comportamiento y un entrenamiento más fácil como función de activación (Apicella et al., 2021).

Sigmoidal: Se pueden utilizar en la capa de salida junto con la entropía cruzada binaria para problemas de clasificación binaria. La salida de esta unidad puede modelar una distribución de Bernoulli sobre la salida y condicionada sobre *x* (Ketkar & Moolayil, 2021).

Softmax: Softmax normaliza las salidas de la capa anterior para que sumen uno. Normalmente, las unidades de la capa anterior modelan una puntuación no normalizada de la probabilidad de que la entrada pertenezca a una clase particular (Ketkar & Moolayil, 2021).

Como se mencionó anteriormente, la neurona, que es la unidad básica de la red neuronal, se organiza a través de una serie de capas, donde cada capa puede contener una o más neuronas.

La primera capa, también conocida como capa 0, es la capa de entrada de la red. Cada capa se conecta con la siguiente a través de un vector de pesos, los cuales se descubren de manera iterativa durante el proceso de entrenamiento.

El número de neuronas definidas para cada capa se conoce como el ancho de la capa. No es necesario que el ancho de cada capa sea el mismo, sin embargo, las dimensiones deben estar alineadas. El número de capas que forman parte de una red se conoce como profundidad de la red.

Cada capa que forma parte de la red toma como entrada la salida producida por la capa anterior. Excepto la primera capa, que hace uso de los datos de entrada, la salida obtenida en la última capa es la predicción generada en función de la entrada (Svozil et al., 1997).

Como se mencionó anteriormente una red neuronal puede ser definida como $f_\theta: x \rightarrow y$, donde siendo especifico θ es la agrupación de todos los pesos *w* que forman parte de la red. Definir la estructura de la red, significa establecer el número de capas y la anchura de cada capa. En la Figura 2.4 se observa un ejemplo de la arquitectura de una red (Ketkar & Moolayil, 2021).

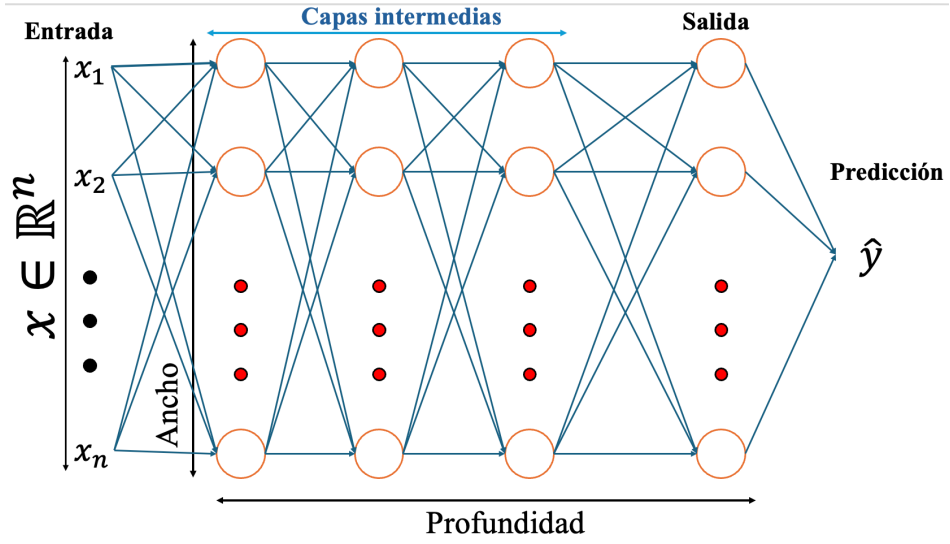


Figura 2.4: Estructura de una red neuronal de propagación hacia adelante

En la Figura 2.4 se observa cómo, para una entrada de datos, se obtiene una salida $\hat{y}$, la cual puede definirse como la predicción realizada por la red para una serie de datos de entrada *x*.

El siguiente paso sería evaluar qué tan precisa es la predicción de la red neuronal en función de una serie de etiquetas de datos y, que contiene los valores reales de cada entrada *x*. Es en este punto donde se hace necesaria un método que permita determinar la magnitud de las diferencias entre las predicciones y los valores reales, introduciendo así el concepto de una función de costo o pérdida (Ketkar & Moolayil, 2021; Svozil et al., 1997).

Esta función mide el desacuerdo existente entre y, $\hat{y}$, denotándose como *l*. Esta función de pérdida permite calcular el desacuerdo existente entre varios puntos de datos. En la Figura 2.5 se ejemplifica el funcionamiento de una función de costo.

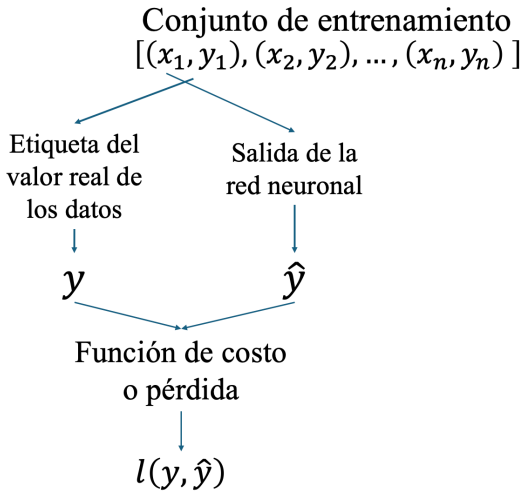


Figura 2.5: Función de costo o pérdida de la red

Algunas de las funciones de costo más usadas se muestran a continuación:

Error cuadrático medio (MSE por sus siglas en Inglés): Es la función de costo más popular en la literatura, debido a su tratabilidad matemática. Esta función es monotónicamente creciente, simétrico, homogéneo de grado 2 y diferenciable en todo su rango. Se calcula a partir de la siguiente fórmula (Jalil & Misas, 2007).

$$MSE = \frac{1}{2}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2 \quad (2.3)$$

Donde *n* es el número de observaciones realizadas, *y* es el valor real de la *i-ésima* observación y $\hat{y}$ es el valor predicho por la red para la *i-ésima* observación.

Error absoluto medio (MAE, por sus siglas en Inglés): Esta función de costo se define como monotónicamente creciente, simétrica, homogénea y diferenciable en todo su rango con la excepción de $|y - x|^2 = 0$. Se calcula a partir de la siguiente fórmula (Jalil & Misas, 2007).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}| \quad (2.4)$$

Error absoluto medio suavizado (SMAE por sus siglas en Inglés): en esta función de costo, cuando la diferencia existente entre la predicción y el valor real es menor a 1 se hace uso del cuadrado de la diferencia multiplicado por 0.5 como costo, lo cual equivale a usar el MSE. Por el contrario, cuando la diferencia es mayor que 1, se utiliza la diferencia absoluta menos 0.5 como costo. Esta función es diferenciable en todo punto, incluyendo los puntos de cambio. Esta propiedad ayuda a estabilizar el proceso de optimización en comparación con otras funciones de costo que no son diferenciables en puntos específicos (Jalil & Misas, 2007). Se calcula a partir de la siguiente fórmula.

$$SMAE = \begin{cases} \frac{1}{2}(x-y)^2 \;\; si\; |x-y| < 1 \\ |x-y| - 0.5 \;\; de\; lo\; contrario \end{cases} \quad (2.5)$$

Una vez definida la función de costo, estas pueden derivarse con el fin de estimar el nivel de desacuerdo permitiendo así actualizar los parámetros de peso para reducir la diferencia y mejorar el rendimiento del modelo.

Con este fin se hace uso del algoritmo de retropropagación, también llamado Backpropagation en inglés. Este algoritmo se utiliza para entrenar redes en problemas de aprendizaje supervisado. Funciona calculando el error total de la red para posteriormente retro propagarlo a las capas anteriores, de modo que, en cada paso de entrenamiento, la función de costo sea cercana a 0.

El algoritmo de retropropagación es un método que permite calcular los gradientes de pérdida con respecto a los pesos y sesgos mediante la implementación de la regla de la cadena. Durante el proceso de entrenamiento, la red neuronal calcula la predicción para un conjunto

de datos de entrada, la función de costo determina la diferencia entre el valor predicho y el valor objetivo real.

A través de este cálculo, se determina el gradiente de la función de pérdida con respecto a los pesos y sesgos de la red, lo que proporciona una descripción generalmente clara de cómo un pequeño cambio en los parámetros de peso o sesgo impacta en el valor de la pérdida total. Utilizando el cálculo de los gradientes, se actualizan los pesos y sesgos de manera iterativa, realizando pequeños ajustes en dirección opuesta al gradiente para moverse hacia los mínimos locales.

Este proceso se conoce como descenso de gradiente, su objetivo es reducir el error de la función de pérdida para encontrar el mínimo. De esta manera, la red neuronal aprende los valores que generan patrones que permiten predecir, para un conjunto de entradas, los valores con la menor diferencia respecto a los valores objetivo-reales. Existen múltiples variantes de este algoritmo, algunas de las cuales incluyen:

Gradiente descendente estocástico (SGD por sus siglas en Inglés): A diferencia de en el método de gradiente descendente, en el gradiente descendente estocástico, el número de gradientes a evaluar no depende de *n* si no que es constante. Y las iteraciones se calculan mediante la siguiente ecuación:

$$\theta_{k+1} = \theta_k - \frac{\alpha k}{n} \nabla \psi_{i_k}(\theta_k) \qquad (2.6)$$

Donde $i_k \in \{1,2,\dots,n\}$ es escogido aleatoriamente, el gradiente $\nabla \psi_{i_k}(\theta_k)$ es un estimador insesgado, de esta manera cada iteración es menos costosa pues solo se evalúa a un solo gradiente (Becerra Contreras, 2017).

RMSPROP: En este algoritmo el factor de entrenamiento es variable, es decir, que cambie en cada iteración el objetivo de esta variación es que el tamaño del paso aumente dependiendo del valor del gradiente en la iteración respectiva para posteriormente proceder a la actualización de los coeficientes en cada iteración. Con este fin se divide el gradiente de un mini lote entre un promedio de sus valores anteriores de tal manera que mini lotes adyacentes sean divididos por un valor similar, este valor es llamado *media cuadrada* y se calcula a través de la ecuación.

$$media - cuadrada(w,t) = 0.9 media - cuadrada(w,t-1) + 0.1\left(\frac{\partial E}{\partial w}\right)^2 \qquad (2.7)$$

Donde *t* es el mini lote actual y $\frac{\partial E}{\partial w}$ es el gradiente de la función de costo con respecto al vector de pesos *w* (Becerra Contreras, 2017).

ADAM: Adam utiliza una tasa de aprendizaje adaptativa para cada parámetro, basada en estimaciones de primer y segundo orden del gradiente. Lo cual permite ajustar la tasa de aprendizaje para cada parámetro de manera independiente, proporcionando una convergencia rápida y estable en comparación con los métodos de optimización tradicionales (Ketkar & Moolayil, 2021).

Adagrad: Se utilizan diferentes tasas de aprendizaje para las variables teniendo en cuenta el gradiente acumulado en cada una de ellas. Un problema de este optimizador es que, en ocasiones, puede ocurrir que la tasa de aprendizaje para una variable decrezca demasiado rápido (Becerra Contreras, 2017).

Adadelta: Es una extensión de Adagrad que busca reducir su tasa de aprendizaje agresiva y monótonamente decreciente. En lugar de almacenar la suma de gradientes cuadrados anteriores, la suma de gradientes se define recursivamente como un promedio decreciente de todos los gradientes cuadrados anteriores (Becerra Contreras, 2017).

Las redes neuronales de propagación hacia adelante han demostrado ser eficaces para modelar relaciones complejas, sin embargo, existen problemas en el análisis de datos donde la interpretabilidad es fundamental. Por tal motivo, algunos autores han propuesto modelos que incorporan redes neuronales con lógica difusa, surgiendo así las llamadas redes neuro difusas, las cuales combinan la capacidad de aprendizaje de las redes neuronales con la interpretación intuitiva de la lógica difusa, lo que las convierte en una herramienta valiosa para entender y tomar decisiones basadas en datos complejos.

2.3 Lógica difusa

La lógica es posiblemente la disciplina más antigua de la historia humana, una primera forma de tomar decisiones conforme a un razonamiento basado en la observación y la experiencia. En la historia se encuentran personajes notables que han realizado estudios de la metodología lógica, entre ellos se puede mencionar a Leibniz, Boole, Russell, Turing, entre muchos otros. Se puede definir a la lógica como el "método o razonamiento en el que las ideas o la sucesión de los hechos se manifiestan o se desarrollan de forma coherente sin que haya contradicciones entre ellas"

Es posible clasificar a la lógica como clásica o Aristotélica, en la cual los cálculos lógicos son bivalentes, lo que significa que mediante el cálculo de sus fórmulas se obtienen valores verdaderos o falsos, pero no pueden ser ambos a la vez. O lógica no clásica que mediante cálculos lógicos ofrece más valores de verdad que los de cierto o falso u hace uso de otros recursos expresivos (Muñoz, 2000).

En este caso nos enfocamos en la *lógica difusa* (FL por sus siglas en Inglés), esta es una disciplina propuesta en los años sesenta por Lotfi Zadeh, cuando se dio cuenta de lo que él llamo el principio de incompatibilidad "*conforme la complejidad de un sistema aumenta, nuestra capacidad para ser precisos y construir instrucciones sobre su comportamiento disminuye hasta el umbral más allá del cual, la precisión y el significado son características excluyentes*" (Pérez Pueyo, 2005). En la lógica difusa se hace uso de los conceptos de la lógica y de los conjuntos difusos (gobierna los principios de implicación y equivalencia en una lógica trivaluada) mediante la definición de grados de pertenencia (Pérez Pueyo, 2005).

Un *conjunto difuso* es una clase de objeto que presenta valores continuos de grados de pertenencia, se caracteriza por una función que asigna a cada objeto un grado de pertenencia que varía entre cero y uno, Las nociones de inclusión, unión, intersección, complemento,

relación, convexidad entre otros, se extienden a tales conjuntos, estableciendo diversas propiedades de estas nociones en el contexto de conjuntos difusos (Zadeh, 1965).

La lógica se estructura en cálculos, y bajo este criterio, es posible clasificarla según cómo se realizan estos cálculos. La lógica formal moderna puede concebirse como una especie de cebolla, en la que cada cálculo base sirve como fundamento para otro con más recursos expresivos, requiriendo nuevos elementos. Así, la estructura de la lógica se organiza de la siguiente manera (Muñoz, 2000):

Lógica proposicional: En esta lógica las fórmulas son proposiciones, oraciones o enunciados sin analizar internamente, la relación a estudiar es la que se establece entre oraciones que constituyen la unidad mínima de significancia lógica. Es un cálculo hipotético ya que la deducción se establece en una relación condicional entre las premisas y la conclusión, es decir si ocurren las premisas ocurre la conclusión.

Lógica de predicados: Se caracteriza por analizar las oraciones en sus componentes y se puede cuantificar sobre individuos. Es fundamentalmente una lógica de clases donde la relación que se estudia es la pertenencia a un conjunto o la posesión de propiedades por los distintos individuos de los que se está hablando. Se dice que es cuantificable debido a que esta lógica tiene recursos para hablar de todos o de algunos de los individuos que pertenecen a un conjunto.

Lógica de primer orden: Se le llama así a la unión del cálculo proposicional y de la lógica de predicados. En esta lógica existe la restricción de que solo se pueden usar los cuantificadores con algunos elementos individuales.

Lógica de 2°, 3°, …, n°: La lógica de segundo, tercer, y así sucesivamente hasta *n-ésimo* orden, se construye sobre la base de la lógica de primer orden. En niveles superiores, los cuantificadores pueden aplicarse a propiedades, predicados, o incluso a predicados de predicados, aumentando así el nivel de abstracción y complejidad de la lógica. En la Figura 2.6 se observa gráficamente esta clasificación:

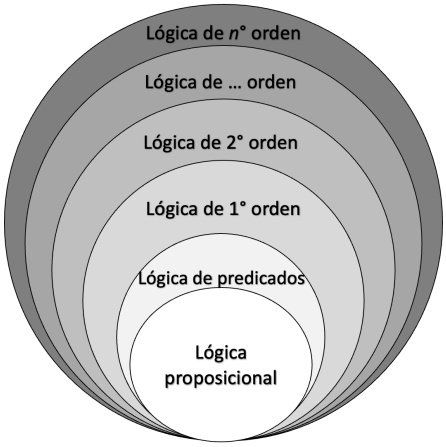


Figura 2.6: Clasificación de la lógica.

La FL tiene la capacidad de reproducir aceptablemente los modos usuales del razonamiento, considerando que la certeza de una proposición es cuestión de grado, donde la lógica clásica es considerado el caso limite. Por lo cual las características más atractivas de la lógica difusa son su flexibilidad, su tolerancia con la imprecisión, su capacidad para modelar problemas no lineales y su base en el lenguaje natural (Pérez Pueyo, 2005).

En las lógicas multivalentes los valores de verdad se corresponden con el intervalo [0,1], la lógica difusa compensatoria (CFL por sus siglas en Inglés) es un nuevo sistema multivalente que rompe con la axiomática tradicional de este tipo de sistemas para lograr un comportamiento semánticamente mejor a los sistemas clásicos (Montero et al., 2012).

2.3.1 Principales tendencias de investigación en lógica difusa

La lógica difusa (FL) representa una gran cantidad de sistemas especiales, que se distinguen entre sí por diversas propiedades, como el conjunto de grados de verdad empleados, su estructura algebraica, las funciones de verdad elegidas para las conexiones lógicas y otras propiedades (Bělohlávek et al., 2017). La FL tiene una historia relativamente pequeña de aproximadamente 50 años. A continuación, se describen dos tendencias de la comunidad académica que la han impulsado y que se complementan.

FL en sentido estricto (FLn, fuzzy logic in narrow sense), también llamada lógica difusa matemática, es básicamente el estudio de sistemas lógicos formales en los que la verdad es una cuestión de grado (Bělohlávek et al., 2017). FLn puede considerarse como una lógica con una noción comparativa de verdad: las oraciones pueden compararse de acuerdo con sus valores de verdad (por ejemplo: "verdadero", "casi verdadero", "no del todo falso", "completamente falso"), lo cual es bastante natural. Los comienzos de FLn generalmente están asociados con el segundo artículo clásico de Goguen convirtiéndose gradualmente en un área establecida de lógica matemática (Goguen, 1969). Como resultado, se han desarrollado varios cálculos lógicos, incluyendo lógicas proposicionales, predicadas, lógicas de orden superior, así como varios tipos de otras lógicas.

Hoy en día, FLn se ha desarrollado gracias a los resultados de un grupo internacional de matemáticos, entre ellos Hájek, Gottwald entre otros (Gottwald, 2000; Hájek, 1998).

La FL en sentido amplio (FLb, fuzzy logic in broad sense) está motivada por el objetivo final de desarrollar medios suficientemente expresivos para emular el razonamiento humano de sentido común en lenguaje natural y algunas otras capacidades únicas de los seres humanos (Bělohlávek et al., 2017). Durante un largo periodo, FLb fue llevada a cabo bajo el liderazgo de Zadeh, centrándose en conjuntos difusos en todas sus manifestaciones teóricas y aplicadas. Desde el punto de vista teórico, su objetivo general es generalizar la teoría de conjuntos intuitiva a la teoría de conjuntos difusos intuitiva mediante la sustitución de conjuntos clásicos por conjuntos difusos (Zadeh, 1965). FLn proporciona los fundamentos para la FLb (Ver Figura 2.7).

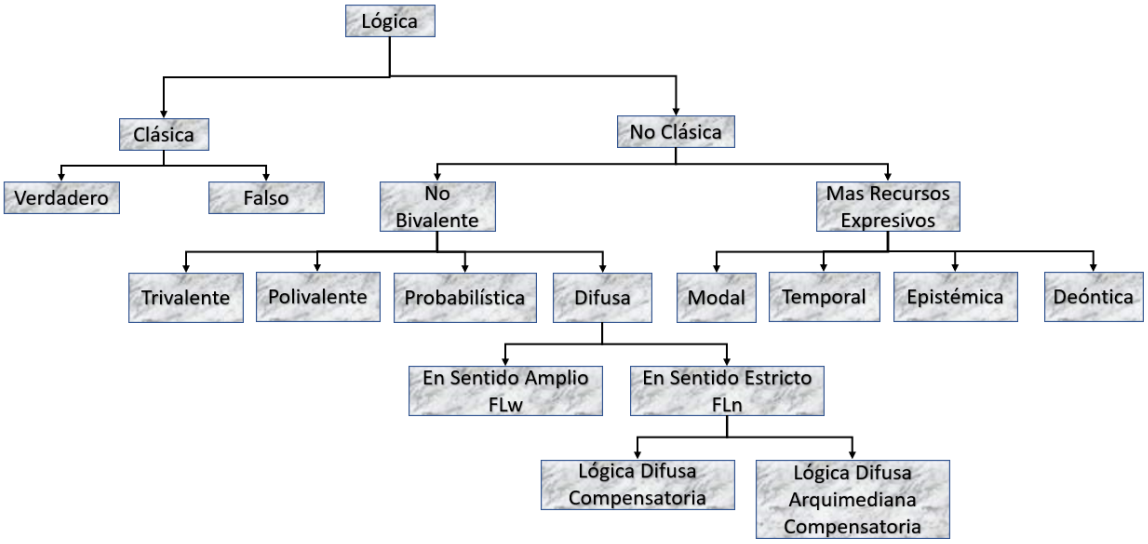


Figura 2.7: Esquema de la lógica.

2.4 Lógica difusa Arquimediana compensatoria

La *lógica difusa Arquimediana compensatoria* (ACFL por sus siglas en Inglés) es un tipo de lógica definida a partir da la unificación de dos enfoques diferentes estas son la lógica difusa Arquimediana y la lógica difusa compensatoria (Espín-Andrade et al., 2015).

La *lógica difusa Arquimediana* (AFL por sus siglas en Inglés) es un sistema lógico de t-norma y t-conorma, donde una t-norma $T: [0,1]^2 \rightarrow [0,1]$ se dice Arquimediana si y solo si satisface alguna de las siguientes condiciones (Espín-Andrade et al., 2015):

i. Para cada $x \in (0,1), T(x,x) < x$
ii. Para cada $x, y \in (0,1)$ hay un numero natural n tal que $T(x,x,\dots,x) < y$ n veces
iii. Para cada $x, y \in (0,1)$ hay un numero natural n tal que $T(x,x,\dots,x) = 0$ n veces

Espin-Andrade define a la AFL como "*un trío* (c,d,n) *de operadores, donde* c *es una t-norma de continua Arquimediana,* d *es su correspondiente t-conorma continua Arquimediana y* n *es el operador de negación"* (Espín-Andrade et al., 2015).

Esta tiene las siguientes propiedades (Espín-Andrade et al., 2015):

c) Toda t-norma $T: [0,1]^2 \rightarrow [0,1]$, satisface $T(x,y) \leq min\{x,y\}$
d) Toda t-conorma $S: [0,1]^2 \rightarrow [0,1]$ satisface $S(x,y) \geq max\{x,y\}$
e) $\min(x,y)$ es la única t-norma continua idempotente y la función máxima de la familia de operadores t-norma.
f) $max(x,y)$ es la única t-conorma continua idempotente y la función mínima de la familia de operadores t-conorma.

g) Como consecuencia de las propiedades precedentes se tiene que cada t-norma (t-conorma) pertenece solo a uno de dos subconjuntos, el singleton de la mínima t-norma (la máxima t-conorma) o el conjunto de t-normas no idempotentes (t-conormas).

h) Para cada t-norma continua Arquimediana existe una función continua no creciente $f:[0,1] \rightarrow [0,+\infty]$ satisfaciendo $f(1)=0$, tal que:

$$T(x_1, x_2, \dots, x_n) = f^{(-1)}\left(\sum_{i=1}^{n} f(x_i)\right) \quad (2.8)$$

a. Donde:

$$f^{(-1)}(z) = \begin{Bmatrix} f^{-1}(z), if\ z \in [0, f(0)] \\ 0\ if\ [f(0), +\infty] \end{Bmatrix} \quad (2.9)$$

i) Existe una propiedad equivalente para t-conormas.

Una *lógica difusa compensatoria* (CFL por sus siglas en Inglés) es aquella que se compone de una cuarteta de operadores continuos (c, d, o, n), c y d de $[0,1]^n$ en $[0,1]$, o de $[0,1]^2$ en $[0,1]$ y n un operador de negación, que satisface el siguiente grupo de axiomas (Espín Andrade et al., 2011; Espín-Andrade et al., 2015, 2016; Montero, J. C., 2011):

I. **Axioma de compensación:** $min(x_1, x_2, \dots, x_n) \leq c(x_1, x_2, \dots, x_n) \leq max(x_1, x_2, \dots, x_n)$

II. **Axioma de conmutatividad o simetría:** $c(x_1, x_2, \dots, x_i, x_j, x_n) = c(x_1, x_2, \dots, x_j, x_i, x_n)$

III. **Axioma de crecimiento estricto:** si $x_1 = y_1, x_2 = y_2, \dots, x_{i-1} = y_{i-1} = x_{i+1} = y_{i+1}, \dots, x_n = y_n$ son diferentes de cero, y $x_i > y_j$ entonces $c(x_1, x_2, \dots, x_n) > c(x_1, x_2, \dots, x_n)$

IV. **Axioma de veto:** Si x_i=0 para alguna i entonces $c(x) = 0$

V. **Axioma de reciprocidad difusa:** $o(x,y) = n[o(y,x)]$

VI. **Axioma de transitividad difusa: si** $o(x,y) \geq 0.5$ y $o(y,z) \geq 0.5$ entonces $o(x,y) \geq \max(o(x,y), o(y,z))$

VII. **Leyes de De Morgan:** $n(c(x_1, x_2, \dots, x_n)) = d(n(x_1), n(x_2), \dots, n(x_n))$

$$n(d(x_1, x_2, \dots, x_n)) = c(n(x_1), n(x_2), \dots, n(x_n))$$

Los operadores c y d son llamados de *conjunción* y *disyunción*, o es el operador de *orden estricto* y n el de *negación*. Una familia particular de la CFL, son aquellas basadas en medias quasi-aritmeticas, en las cuales se encuentran operadores de la forma:

$$M_f(x_1, x_{2,\dots,} x_n) = f^{-1}\left(\frac{\sum_{i=1}^{n} f(x_i)}{n}\right) \quad (2.10)$$

Donde f es una función estrictamente monótona y continua, que se extiende a puntos no definidos mediante su correspondiente limite (Espín-Andrade et al., 2015; Montero, J. C., 2011).

Se tiene que los operadores mencionados satisfacen los axiomas I-III mencionados anteriormente, además si se tiene que para todo $i \in (1, 2, \dots, n), lim_{x_i \rightarrow 0}$,

$M_f(x_1, x_2, \dots, x_n) = 0$ se satisface el axioma IV, además, a través de las siguientes ecuaciones:

$$d\left(x_1, x_{2,\dots,} x_n\right) = 1 - f^{-1}\left(\frac{\sum_{i=1}^{n} f(1-x_i)}{n}\right) \quad (2.11)$$

$$n(x) = 1 - x \quad (2.12)$$

$$o(x, y) = 0.5\,[c(x) - c(y)] + 0.5 \quad (2.13)$$

Se tiene una lógica compensatoria conocida como lógica compensatoria basada en medias cuasi-aritmeticas (CFLQMB por sus siglas en Inglés) (Espín Andrade et al., 2011; Espín-Andrade et al., 2015, 2016; Montero, J. C., 2011).

Espin-Andrade et al. (Espín-Andrade et al., 2015) define a la ACFL como "*El sexteto de operadores* $(c_t, c_c, d_t, d_c, o, n)$ *tales que* (c_t, d_t, n) *es una* AFL *y* (c_c, d_c, o, n) *es una* CFL*, de manera que, la* LDA *y la* LDC *son compatibles".* Además, se menciona que "*toda* AFL *y su correspondiente* CFL *son compatibles para cada fórmula proposicional formada recursivamente por predicados simples, operadores de conjunción, disyunción, negación, implicación y equivalencia*".

Para este caso una ACFL es obtenida a través de una lógica probabilística (PL por sus siglas en Inglés) presentada por Detyniecki y una lógica difusa compensatoria basada en la media geométrica (CFLGMB por sus siglas en Inglés) presentada por Espín Andrade (Detyniecki, 2001; Espín Andrade et al., 2011).

Ambos sistemas lógicos pueden ser definidos a través de las mismas funciones generadoras, $f(u) = -\ln(u)$ para la conjunción y $f(u) = -\ln(1-u)$ para la disyunción. De esta manera la ACFL generada a partir del par de funciones $-\ln(u)$ y $-\ln(1-u)$ son compatibles para calcular con PL o CFLGMB, esto con el fin de mantener el orden de los predicados. En la Tabla 2.1 se observan los operadores que conforman los predicados para una ACFL.

Tabla 2.1: Operadores de PL y CFLGMB

Operador	***PL***	***CFLGMB***
Conjunción	$c(x_1, x_2) = x_1 . x_2$	$c(x) = \sqrt[n]{\sum_{i=1}^{n} x_i}$
Disyunción	$d(x_1, x_2) = x_1 + x_2 - (x_1 . x_2)$	$d(x) = 1 - \sqrt[n]{\sum_{i=1}^{n}(1 - x_i)}$
Negación	$n(x) = 1 - x$	$n(x) = 1 - x$
Implicación	$i(x_1, x_2) = 1 - x_1 + x_1 . x_2$	$i(x_1, x_2) = \sqrt{x_1 - (x_1 . x_2)}$
Equivalencia	$e(x_1, x_2) = i(x_1, x_2) . i(x_2, x_1)$	$e(x_1, x_2) = c(i(x_1, x_2) . i(x_2, x_1))$

Además, en Espin-Andrade presenta lo que denominan un nuevo enfoque del cuantificador universal y existencial para una ACFL. En este una cuarteta de cuantificadores $(\forall_T, \forall_c, \exists_T, \exists_c)$ se define como sigue:

"Sea $X = \{x_1, x_2, \dots, x_n\} \subset [0,1]$ *el cuantificador universal para el conjunto finito X y la T-norma continua Arquimediana, definida por:* $\forall_T x_i \in X = c_T(x_1, x_2, \dots, x_n)$*. llamémosle cuantificador universal de no refutación"* (Espín-Andrade et al., 2015).

"Sea $X = \{x_1, x_2, \dots, x_n\} \subset [0,1]$ *el cuantificador universal para el conjunto finito X y el operador de conjunción de una CFL basada en ecuaciones cuasi-aritmeticas, definida por:* $\forall_c x_i \in X = c_c(x_1, x_2, \dots, x_n)$*. llamémosle cuantificador universal de la afirmación"* (Espín-Andrade et al., 2015).

"Sea $X = \{x_1, x_2, \dots, x_n\} \subset [0,1]$ *el cuantificador existencial para el conjunto finito X y la T-norma continua Arquimediana, definida por:* $\exists_T x_i \in X = d_T(x_1, x_2, \dots, x_n)$*. llamémosle cuantificador existencial de no refutación"* (Espín-Andrade et al., 2015)

"Sea $X = \{x_1, x_2, \dots, x_n\} \subset [0,1]$ *el cuantificador existencial para el conjunto finito X y la CFL, definida por:* $\exists_c x_i \in X = d_c(x_1, x_2, \dots, x_n)$*. llamémosle cuantificador existencial de la afirmación"* (Espín-Andrade et al., 2015).

Además, es mencionado que una ACFL tiene una propiedad de afirmación y refutación al mismo tiempo, las cuales son operaciones complementarias.

Denominando así una cuarteta de cuantificadores de predicados como sigue:

$\forall_T \boldsymbol{xp_T(x)}$**:** Se denomina el valor de la necesidad de no refutación de $p(x)$.
$\forall_c \boldsymbol{xp_c(x)}$**:** Se denomina el valor de la necesidad de la afirmación de $p(x)$.
$\exists_c \boldsymbol{xp_c(x)}$**:** Se denomina el valor de la posibilidad de la afirmación de $p(x)$.
$\exists_T \boldsymbol{xp_T(x)}$**:** Se denomina el valor de la posibilidad de no refutación de $p(x)$.

Estos cuatro operadores son enfoques de los conceptos de la necesidad y posibilidad observados en la lógica modal, también incluidos en la teoría de la lógica difusa de la necesidad.

Capítulo 3: Estado del arte

En este capítulo se analizan los artículos del estado del arte relacionados con procesos de analítica de datos mediante metodologías basadas en lógica difusa. El principal objetivo de esta revisión es observar cómo se han aplicado los modelos de lógica difusa para resolver problemas de analítica de datos, identificar las principales contribuciones, las ventajas que ofrecen los modelos propuestos y sus áreas de mejora. Además, se busca determinar cómo la integración de modelos de lógica difusa con redes neuronales aborda el proceso de interpretabilidad de sus resultados, así como sus alcances, limitaciones y principales contribuciones para generar un modelo que justifique los resultados a partir de expresiones lingüísticas similares al lenguaje humano.

3.1 Lógica difusa y su aplicación al análisis de datos

En esta sección, se recopilan y analizan algunos de los enfoques reportados en la literatura sobre la integración de la lógica difusa en problemas de analítica de datos. El objetivo es examinar la efectividad de tales enfoques en la resolución de diversas tareas de analítica de datos, como clasificación, predicción, clustering y toma de decisiones. Mediante la revisión de dichos trabajos, se busca identificar los logros, desafíos y áreas de mejora en la aplicación de la lógica difusa en el campo de la analítica de datos.

Los modelos de lógica difusa han adquirir la capacidad de aprender mediante su implementación a través de sistemas inteligentes de análisis de datos. Esta integración ha facilitado su comprensión y aplicación en contextos prácticos, lo cual resulta en soluciones efectivas y accesibles para una amplia gama de problemas.

Los modelos difusos se han aplicado exitosamente en problemas de clasificación, predicción, planificación, control y toma de decisiones, entre otros. En particular, los modelos de Mamdani y Sugeno, basados en reglas, son útiles, simples y fáciles de manejar (Kruse et al., 2013). Estas características los convierten en herramientas ampliamente aplicables en una variedad de áreas. Específicamente, aquellos modelos difusos basados en reglas han demostrado ser efectivos y fácilmente comprensibles al abordar problemas del mundo real.

En áreas como el aprendizaje automático, varios enfoques de aprendizaje supervisado se han utilizado para resolver problemas de clasificación y predicción, incluyendo algoritmos como árboles de decisión, vecinos cercanos, algoritmos genéticos, máquinas de soporte vectorial, redes bayesianas y redes neuronales. La lógica difusa se ha integrado comúnmente con dichos enfoques, resolviendo problemas en datos incompletos o imprecisos (Ahn et al., 2019).

En Ahn et al., presentan un algoritmo de aprendizaje automático basado en lógica difusa para realizar análisis de comportamiento empresarial. Para esto hace uso del algoritmo C4.5 usado por lo general en procesos de clasificación, regresión y extracción de reglas. A este algoritmo se le incorpora el concepto de lógica difusa. Como resultado, presenta un algoritmo que

acelera los procesos de minería y analítica de negocios además de altos valores de exactitud en procesos de predicción y clasificación (Ahn et al., 2019).

En Livieris et al., proponen un algoritmo de aprendizaje semi-supervisado conjunto que permite predecir el comportamiento de una serie de acciones en el promedio industrial Dow Jones. Los resultados mostrados indican que el algoritmo este algoritmo supera a los algoritmos de aprendizaje semi supervisados que lo componen, lo cual puede permitir desarrollar modelos de predicción confiables y robustos (Livieris et al., 2018).

En Vivek et al., proponen un modelo de evaluación basado en lógica difusa para analizar las percepciones de los clientes sobre diferentes alimentos. Este enfoque proporciona una evaluación objetiva de las percepciones de los consumidores, lo que permite comprender su actitud hacia los alimentos presentados, permitiendo prever su comportamiento de compra futuro. El estudio utiliza etiquetas lingüísticas para interpretar los resultados y definir las condiciones del modelo, lo que facilita la comprensión de las preferencias del consumidor (Vivek et al., 2020).

En Reddy et al., presentan un enfoque el cual implica la implementación de un algoritmo genético adaptativo que emplea reglas difusas para clasificar los tipos de enfermedades cardíacas en pacientes. Este modelo demuestra un rendimiento satisfactorio en términos de precisión, especificidad y sensibilidad. Además, destaca por su capacidad para trabajar eficazmente con conjuntos de datos incompletos y de gran escala (Reddy et al., 2020).

En Ren et al., introducen un algoritmo de máquina de soporte vectorial que utiliza reglas difusas para clasificar posturas humanas en la cama de manera eficiente. Los resultados obtenidos muestran una alta precisión en la identificación de patrones y clasificación. Se espera que este avance contribuya al desarrollo de robots autónomos para asistencia en entornos hospitalarios (Ren et al., 2020).

En Hong Lan et al., introducen nuevas nociones y algoritmos para un sistema de inferencia difuso complejo, empleando modelos de inferencia difusa de Mamdani, Sugeno y Tsukamoto. Se desarrolla un grafo de conocimiento difuso basado en reglas difusas definidas a partir de una variedad de etiquetas lingüísticas. Este enfoque logra reducir el costo computacional y alcanzar una precisión en las predicciones aceptable, además de obtener una capacidad de razonamiento aproximado para nuevos conjuntos de datos (Hong Lan et al., 2020).

En Hernández-Julio et al., se presenta un enfoque que combina el algoritmo K-Means con la lógica difusa para estimar con precisión las características necesarias para el desarrollo y reproducción de carnada viva en acuicultura. Este enfoque utiliza procesos de clusterización y tablas dinámicas para realizar predicciones altamente precisas sobre las condiciones óptimas para el cultivo de carnada viva (Hernández-Julio et al., 2020).

En Zhu et al., proponen una red bayesiana que incorpora conceptos de lógica difusa. El objetivo de esta propuesta es clasificar de manera precisa el grado de concentración de un acuanauta en la ejecución de su trabajo mediante la evaluación de las condiciones físicas y psicológicas del entorno. A partir de la información obtenida, se buscan optimizar las

condiciones necesarias para mantener un alto nivel de concentración del acuanauta. Como resultado, se obtienen factores tanto cuantitativos como cualitativos que permiten inferir y mejorar el estado de concentración del acuanauta (Zhu et al., 2021).

En Llorente-Peralta et al., introducen un algoritmo genético que permite descubrir predicados de lógica difusa compensatoria mediante programación genética. Los predicados son descubiertos a través del algoritmo genético, lo que facilita el descubrimiento de conocimiento sin la necesidad de intervención de un modelado experto. Tales predicados resultan útiles en procesos de clasificación y representan una forma eficiente de manejar datos complejos y variables (Llorente-Peralta et al., 2021).

Gündogdu, propone una serie de metodologías para seleccionar las mejores características en el proceso de optimización a través de un modelo de regresión lineal. Al mismo tiempo, se seleccionan las mejores características del conjunto de datos Dow Jones Index, lo que permite obtener una mayor precisión mediante la implementación de una red neuronal artificial (ANN). Como resultado, se obtiene un modelo altamente preciso y capaz de resolver problemas de estimación de acciones en la bolsa de valores (Gündogdu, 2022).

En conclusión, la lógica difusa, que permite el manejo de datos incompletos o imprecisos, al combinarse con métodos de aprendizaje automático y sistemas de análisis de datos, ofrece soluciones robustas y precisas para una amplia variedad de problemas. Esta combinación ha mejorado la interpretabilidad y eficacia de los modelos de aprendizaje mediante expresiones lingüísticas similares al lenguaje humano

La lógica difusa presenta una serie de desafíos en su aplicación a problemas de analítica de datos. Entre los cuales se incluyen la complejidad computacional, que limita la escalabilidad y eficiencia de los sistemas, la dificultad en la interpretación de resultados, que complica la expresión de los hallazgos y la formulación de reglas difusas, que a menudo requiere expertos especializados. Además, la integración con otros modelos de analítica de datos presenta desafíos técnicos y la validación y verificación de los sistemas difusos pueden ser complicadas debido a la naturaleza subjetiva de las reglas y los conjuntos difusos.

Para abordar dichos desafíos, se observan las siguientes áreas de mejora: la optimización de modelos computacionales para manejar grandes conjuntos de datos y la automatización de la generación de reglas para reducir la dependencia de expertos. También es importante mejorar la interpretabilidad de los resultados mediante técnicas que permitan visualizar y explicar los hallazgos. Asimismo, fomentar la integración de la lógica difusa con otros enfoques de analítica de datos puede mejorar la precisión y la interpretabilidad de los modelos.

3.2 Analítica de datos mediante redes neuronales y lógica difusa

En la actualidad, la creciente cantidad de datos generados y la necesidad de extraer información relevante de ellos demanda la implementación de modelos inteligentes que apoyen el análisis de datos en diversos ámbitos del conocimiento humano (de Campos Souza, 2020).

Uno de los modelos inteligentes que ha destacado por su excelente desempeño en problemas de clasificación de patrones, regresión, estimación, análisis de datos, entre otros, son las redes neuronales artificiales (ANN) (de Campos Souza, 2020). No obstante, tales modelos enfrentan desafíos significativos, por ejemplo, requieren una cantidad adecuada de datos para el entrenamiento, lo que puede ser una tarea difícil. El tiempo necesario para el proceso de entrenamiento es significativo, donde una vez entrenados, los modelos están optimizados para tareas específicas, lo que significa que cualquier cambio en la tarea requiere repetir el proceso de entrenamiento.

Además, de necesitar una considerable potencia de cálculo. Las redes neuronales solo obtienen su mejor rendimiento con datos homogéneos, el manejo de datos con características heterogéneas exige la implementación de métodos de normalización adecuados. Otro desafío importante es que las soluciones generadas por los modelos de redes neuronales suelen ser tan complejas que resulta difícil explicarlas de manera comprensible a personas sin conocimientos especializados en el área (Das et al., 2020; de Campos Souza, 2020).

Teniendo en cuenta los aspectos observados, surge la propuesta de utilizar modelos de ANN que integran conceptos de lógica difusa (FL). La FL permite obtener la capacidad de manejar datos vagos, incompletos e imprecisos, lo que permite manejar la ambigüedad e incertidumbre inherentes a la información del mundo real. Además, la lógica difusa proporciona la capacidad de interpretar los modelos de resolución de problemas de manera comprensible y accesible (Das et al., 2020; de Campos Souza, 2020).

Así, la lógica difusa permite abordar los desafíos encontrados en los modelos de redes neuronales, creando una sinergia entre la capacidad de aprendizaje de las redes neuronales y la capacidad de representar problemas de manera interpretable. Esta combinación aprovecha lo mejor de ambos enfoques, dando lugar a modelos híbridos conocidos como modelos neuro difusos o modelos de redes neuronales difusas (Das et al., 2020; de Campos Souza, 2020).

Algunos de los artículos revisados que implementan estos modelos híbridos se mencionan en los párrafos siguientes.

En Ben Seghier et al., proponen una red neuronal de lógica difusa adaptativa (ANFIS) que descubre la configuración óptima de parámetros mediante los algoritmos de optimización de enjambre de partículas y un algoritmo genético, permitiendo así encontrar los valores adecuados de los parámetros de la ANFIS. Este enfoque permite obtener resultados precisos y eficientes en la estimación del factor de intensidad del estrés, superando los métodos tradicionales como la correlación estándar normal (Ben Seghier et al., 2020).

En Melin et al., presentan un enfoque que integra un modelo de red neuronal de conjuntos múltiples con agregación de respuesta difusa. En este método, la red neuronal de conjuntos utiliza una serie de módulos, donde cada módulo consiste en una red neuronal simple. A cada uno de los módulos predictores se aplica un modelo de lógica difusa para mejorar la precisión e interpretabilidad de las predicciones. Este modelo se utilizó para inferir el número de casos por estado y país de personas infectadas con COVID-19. Los resultados muestran que la red desarrollada permite predecir casos de manera muy precisa, manteniendo tiempos de ejecución aceptables (Melin et al., 2020).

En De Campos Souza et al., introducen una red neuro difusa evolutiva la cual logra alto grado de precisión en los procesos predictivos, así como en su capacidad de interpretación. Este algoritmo permite la creación de neuronas difusas mediante un proceso de fuzzificación adaptativa y un procedimiento de entrenamiento evolutivo. Además, el proceso de construcción permite combinar las partes antecedentes de las reglas con declaraciones AND y OR, lo que mejora la capacidad de las redes clásicas que solo permiten el uso de AND. Esta combinación de técnicas evolutivas y lógica difusa proporciona una alta capacidad de modelización y predicción de sistemas complejos, con la ventaja adicional de ser altamente interpretable (de Campos Souza & Lughofer, 2021).

En Xie et al., presentan un modelo neuronal difuso de Hammerstein-Wiener para predecir los precios de las acciones en el mercado bursátil. Este algoritmo incorpora el bloque dinámico lineal del modelo Hammerstein-Wiener como mecanismo de inferencia, junto con el modelo neuronal difuso de Mamdani. La implementación de este enfoque mejoró significativamente la precisión de la predicción de los valores de las acciones. Además, permite interpretar el conocimiento con significado semántico mediante el uso de reglas lingüísticas difusas (Xie et al., 2021).

En Nguyen Thu Hien et al., proponen un modelo neuro difuso basado en aprendizaje híbrido que combina las características del modelado difuso con la implementación de un mecanismo de atención. Este enfoque se aplica a conjuntos de datos para definir un diagnóstico de cáncer, logrando una alta precisión en este proceso. Además, las reglas si-entonces difusas generadas por el modelo pueden ser interpretadas, lo que valida y mejora la confiabilidad de los resultados del diagnóstico (Nguyen Thu Hien et al., 2022).

En Yang & Zhao, proponen una red neuronal que implementa también una red de picos, a través del cual hace posible extraer información espacial, temporal y espaciotemporal implícita en las muestras de entrenamiento, de tal manera que logra superar los algoritmos con los que se compara, además presenta una gran velocidad de convergencia y permite mejorar la capacidad de extracción de características (Yang & Zhao, 2023).

En Barraza et al., proponen un enfoque híbrido que combina una red neuronal competitiva con sistemas de inferencia difusa tipo 1. La principal contribución de este enfoque radica en la capacidad de diseñar automáticamente un sistema de inferencia difuso tipo 1. Este método fue aplicado en tareas de clasificación, donde demostró un desempeño superior en comparación con otros algoritmos utilizados en el estudio (Barraza et al., 2023).

3.3 Comentarios finales

De acuerdo con lo observado en la literatura, se pueden definir dos tipos principales de redes neuro-difusas: aquellas que presentan un modelo híbrido, donde la red neuronal incorpora conceptos de lógica difusa, y aquellas donde la red neuronal se encarga de optimizar los parámetros de un modelo difuso. Ambas metodologías han demostrado ser efectivas en procesos de clasificación e inferencia, resolviendo de manera óptima procesos de analítica de datos.

En las redes neuronales híbridas, la información se modela a partir de la definición de estados lingüísticos. Para cada entrada x_i, se asocia una función de membresía que establece su modelado difuso, y una etiqueta que define el estado de la función. Entre las etiquetas más comunes se encuentran Alto, Medio y Bajo. Estas etiquetas pasan a ser evaluadas mediante una serie de reglas difusas del tipo $If - Then$, un ejemplo de este tipo de regla se muestra a continuación: $If\ x\ is\ A\ and\ y\ is\ B\ Then\ z\ is\ C$. Estas reglas se integran en un modelo de cálculo para obtener la salida difusa. Entre los modelos de inferencia difusa más usados e implementados en estas redes se encuentran los modelos de Takagi-Sugeno-Kang (TSK).

Por otro lado, las redes neuronales que se utilizan para descubrir los parámetros que mejor definen el modelo difuso implementan algoritmos como los genéticos, enjambre de abejas, Hammerstein-Wiener, entre otros. En este caso, el objetivo es optimizar los parámetros para mejorar el modelo difuso. Generalmente, los modelos difusos implementados son los mismos mencionados anteriormente, como los modelos TSK. Estos enfoques permiten ajustar las funciones de membresía, las reglas difusas y otros parámetros del sistema para lograr un mejor desempeño.

Las redes neuronales híbridas combinan las capacidades de aprendizaje de las redes neuronales con la capacidad de razonamiento de los sistemas difusos. Este enfoque es particularmente útil cuando se requiere un sistema que pueda aprender a partir de datos y, al mismo tiempo, manejar incertidumbre e imprecisión. Además, la flexibilidad de las redes neuronales permite adaptar el sistema a diferentes contextos y tipos de datos.

En los modelos de optimización de redes neuronales, la red neuronal actúa como un optimizador que ajusta los parámetros del sistema difuso. Esto no solo mejora el rendimiento del sistema, sino que también permite una adaptación dinámica a cambios en el entorno o en los datos de entrada.

En los artículos listados anteriormente, se puede observar una diversidad de enfoques que permiten resolver problemas de análisis de datos. Por ejemplo, en Vivek et al., se evalúa las preferencias de un grupo de comensales, mientras que, en Melin et al., infieren el impacto de una pandemia en zonas específicas. Nguyen Thu Hien et al., se centran en definir el diagnóstico de una persona con cáncer, y Xie et al., aplican métodos difusos a modelos de negocios. Además, se encuentran aplicaciones en la ingeniería, como se observa en los trabajos de Campos Souza, Ben Seghier et al., Barraza et al., entre otros.

Por otro lado, una de las principales ventajas que ofrecen este tipo de modelos es la capacidad de definir modelos interpretables y expresables en lenguaje natural. De manera clásica, esta interpretación se realiza a través de la definición de una etiqueta lingüística enlazada a un atributo específico, el cual es modelado a través de una función de membresía específica. Por ejemplo, si se tiene el atributo *x*=altura y *y*=peso de un individuo, se puede establecer el siguiente predicado difuso: "Si *x* es alto y *y* es bajo, entonces el individuo es sano". De esta manera, la interpretación y su posterior expresión en lenguaje natural se vuelven fáciles de comprender.

Sin embargo, esta manera clásica de expresar los resultados está sujeta a la subjetividad y a las limitaciones propias del entorno de datos. Si bien un determinado predicado puede ser cierto para los elementos que forman parte de un registro, no necesariamente es objetivo cuando se evalúan elementos que no forman parte de este y que difieren en gran medida de los datos modelados. Además, la subjetividad en la definición de las etiquetas lingüísticas y las funciones de membresía puede llevar a interpretaciones variadas dependiendo de los expertos que definan estos parámetros.

En la Tabla 3.1, se presentan las principales características observadas en algunos de los artículos evaluados, permitiendo una comparación con el modelo de solución propuesto basado en lógica difusa Arquimediana compensatoria.

Tabla 3.1: Comparación de los modelos revisados en la literatura con el modelo propuesto

REF.	MODELO NEURO DIFUSO								INTERPRETABILIDAD			
	M	**MID**	**L**	**FM**	**EL.A**	**PL**	**W**	**ML**	**I**	**MI**	**ELN**	**EP**
(BEN SEGHIER ET AL., 2020)	2	6	1	3	3	1	S	N	S	N	N	N
(MELIN ET AL., 2020)	2	2	1	3	1	1	S	N	S	N	N	N
(DE CAMPOS SOUZA, 2020)	1	3	1	3	**2**	2	S	N	S	S	S	N
(XIE ET AL., 2021)	2	5	1	3	3	1	S	N	S	S	S	N
(NGUYEN THU HIEN ET AL., 2022)	1	6	1	3	1	1	S	N	S	S	S	N
(BARRAZA ET AL., 2023)	2	1 y 2	1	1,2 y 3	3	1	S	N	S	S	S	N
(LLORENTE-PERALTA ET AL., 2021)	1	**4**	**2**	**4**	**2**	**3**	N	**S**	S	S	S	**S**

Donde:

M: Se refiere al modelo de red difusa presentado de tal manera que:

1. Red neuronal híbrida de lógica difusa compensatoria
2. Red neuronal de lógica difusa que implementado junto a otros modelos algorítmicos

MID: Se refiere al modelo de inferencia difuso utilizado en la red de tal manera que:

1. Sugeno
2. Mamdani
3. Predicados If-Then
4. Predicado en forma normal disyuntiva (FND)
5. Centro medio
6. No definida

L: Se refiere al tipo de lógica implementada en el algoritmo:

1. Lógica difusa de Zadeh
2. Lógica difusa Arquimediana compensatoria

FM: Se refiere a la función o funciones de membresía implementadas por el modelo:

1. Trapezoidal
2. Triangular
3. Gaussiana
4. Variable lingüística continua generalizada (GCLV)

EL.A: Se refiere al número de estados lingüísticas asociadas a un atributo de entrada:

1. Tres etiquetas entre las cuales comúnmente son Alto, Medio, Bajo
2. Una, generalmente modificable a través de algún método.
3. No definida, pero más de dos.

PL: Se refiere al tipo de predicado lingüístico que se implementa en el modelo

1. Predicados del tipo If-Then conjuntivos $(If\ x\ is\ A\ and\ y\ is\ B\ Then\ Z\ is\ C)$
2. Predicados del tipo If-Then conjuntivos/disyuntivos
   $(If\ x\ is\ A\ and/or\ y\ is\ B\ Then\ Z\ is\ C)$
3. Predicados en forma normal disyuntiva
   $If\ (GCLV(x_1)\ and\ GCLV(y_1))\ or\ (GCLV(x_2)\ and\ GCLV(y_2))\ or\ \dots or$
   $(GCLV(x_n)\ and\ GCLV(y_n))$

W: Indica si en modelo utiliza un factor de peso como método de ponderación:

S: significa que utiliza un factor de peso W
N: Significa que no utiliza un factor de peso W

ML: Indica si el modelo hace uso de un modificador lingüístico para cambiar la morfología de una función de pertenencia.

S: indica que si se hace uso del modificador lingüístico
N: indica que no se hace uso de un modificador lingüístico

I: Indica que en documento se menciona que el modelo es interpretable

S: Indica que se menciona que el modelo es interpretable
N: Indica que en el modelo no se menciona que sea interpretable

MI: Indica si en el modelo se explica cómo se lleva a cabo la interpretabilidad

S: Demuestra la interpretabilidad del modelo
N: No muestra la interpretabilidad del modelo

ELN: Indica si los resultados son expresados en lenguaje natural
S: Los resultados son expresables en lenguaje natural
N: los resultados no son expresables en lenguaje natural

EP: Indica que los predicados son expresados en función de sus preimágenes:

De manera clásica, la interpretación de los predicados lógicos se realiza en función de etiquetas vinculadas a un determinado atributo. Por ejemplo, para el atributo $Peso$, se pueden definir las etiquetas $Bajo$, $Medio$ y $Alto$. Así, un predicado definido a partir del atributo y la etiqueta podría expresarse como: $Si\ Peso\ es\ Alto$, $Si\ Peso\ es\ Medio$, o $Si\ Peso\ es\ Bajo$.

Sin embargo, una desventaja de esta forma de expresar los predicados es que las etiquetas son subjetivas y dependen de la interpretación de un experto, lo cual puede agregar variabilidad y falta de consistencia en las interpretaciones, además de limitar la reproducibilidad y objetividad del modelo. Además, estas etiquetas están basadas en las magnitudes del conjunto de datos, lo que significa que aquellos con diferentes magnitudes no serán etiquetados correctamente.

La principal diferencia entre este tipo de expresión y la expresión basada en preimágenes es que la preimagen se interpreta como la magnitud en la que la pertenencia de un atributo a un conjunto es tan cierta como falsa (valor difuso de 0.5) y la magnitud en la que la pertenencia de un atributo a un conjunto es completamente cierta (valor difuso de 1.0). Esto permite definir predicados evaluados en función de sus magnitudes, eliminando la subjetividad y la necesidad de interpretación de un experto.

Capítulo 4: Métodos propuestos de representación, interpretación, búsqueda e inferencia

La lógica difusa se ha convertido en una herramienta importante para el análisis de datos en el ámbito de la inteligencia artificial. Su capacidad para modelar la vaguedad y la incertidumbre la hace versátil y con el potencial de revolucionar los procesos de analítica de datos. Este capítulo primero presenta dos sistemas axiomáticos de una lógica difusa Arquimediana compensatoria, que constituyen la base de la red neuronal híbrida propuesta. Finalmente, se presentan dos modelos de red neuronal resultantes que realizan clasificación y estimación, avanzando la analítica de datos en el aprendizaje automático.

4.1 Sistema axiomático de clasificación, representación y descubrimiento de predicados a partir de una lógica difusa Arquimediana compensatoria.

Esta sección inicia describiendo una nueva teoría lógica transdisciplinaria generalizada denominada *lógica difusa Arquimediana compensatoria* (ACFL por sus siglas en Inglés), la cual fue formulada por Rafael A. Espín Andrade y operacionalizada en esta tesis para el descubrimiento de conocimiento expresado a través de predicados lógicos e interpretables en términos de un lenguaje natural sencillo. La formulación y operacionalización se presentan en el artículo titulado *Archimedean Compensatory Fuzzy Logic as a Pluralist Contextual Theory Useful for Knowledge Discovery* (Espín-Andrade et al., 2021).

A diferencia de lo visto en la sección 2.3, esta sección se centra en la generalización de conceptos de una ACFL a través de la definición de una función generadora *f*. lo cual ofrece el aumento de nuevas perspectivas en el tratamiento de los datos al permitir generar funciones de pertenencia adaptativas y la capacidad de navegar entre diferentes lógicas.

Las funciones de pertenencia son un concepto fundamental de la lógica difusa, a través del estudio de diversos artículos científicos, se ha llegado a observar que, de alguna manera, este concepto es subestimado en el proceso del modelado de la información.

Dentro de las principales aportaciones que se generan a través de la aplicación de una ACFL generable, se tienen las siguientes:

I. A través de la definición de una función generadora *f*, se genera una ACFL, que permite la construcción de una función de membresía sigmoidal, la cual toma por nombre *función sigmoidal generalizada* (GSF por sus siglas en Inglés), donde a partir de la aplicación de esta GSF y un *modificador lingüístico generalizado* (LM por sus siglas

en Inglés) se genera una familia de funciones de membresía parametrizadas denominada *variable continua lingüística generalizada* (GCLV por sus siglas en Inglés) para cualquier ACFL especifica.

II. La GCLV permite generar una familia de funciones de membresía multiformes definibles a través de sus parámetros, lo cual resulta en una variable lingüística infinita, en contraste con un conjunto finito de valores lingüísticos. Esta familia incluye al menos tres formas básicas de funciones, que son funciones sigmoidales, sigmoidales inversas y convexas.
III. Una ACFL permite integrar en un solo marco teórico dos teorías lógicas difusas multivaluadas, funciones generadoras y herramientas semánticas esenciales, tales como modificadores, funciones de membresía y variables lingüísticas. Permitiendo a una ACFL expresar conocimientos contextuales específicos en una lógica determinada.
IV. La aplicación práctica de una familia de funciones de membresía generada a partir de una GCLV presenta ventajas frente a otras funciones de membresía. Por ejemplo, en procesos de *descubrimiento de conocimiento* (KD por sus siglas en Inglés), esta familia de funciones se ajusta según el conjunto de datos perteneciente al problema, lo cual permite resolver problemas de optimización en el espacio de parámetros continuos.
V. A través de la interpretación de las GCLV, los resultados son expresables en términos de lenguaje natural, donde estas expresiones incluyen los valores lingüísticos esenciales asociados a la función de membresía. Tal expresión se logra mediante el principio de representación de variables lingüísticas y algoritmos desarrollados que determinan etiquetas específicas definidas a partir de la variable lingüística predeterminada.

En todo proceso de análisis de datos, la interpretabilidad es un concepto esencial, a su vez es una propiedad inherente a la lógica difusa, es decir, son modelos fácilmente entendidos por los seres humanos, ya que su diseño refleja el razonamiento humano y captura la incertidumbre en muchos problemas de la vida real.

El concepto de interpretabilidad desde el punto de vista de una ACFL se refiere a la capacidad del modelo para establecer una relación clara y bidireccional entre los resultados obtenidos a través de los cálculos realizados por la teoría lógica y los significados de esos resultados en el contexto del problema que se está abordando. En otras palabras, se trata de la propiedad que cumple una ACFL, donde los resultados del cálculo realizado por la teoría se relacionan directamente con los objetos y operadores utilizados en ella, a su vez, los resultados tienen significado y relevancia en el contexto del problema. Permitiendo que el modelo sea representado y comprendido tanto en lenguaje natural como profesional, facilitando su interpretación.

La interpretación de los resultados obtenidos mediante una ACFL y su representación a través del lenguaje natural o profesional debe ser clara y directa. Sin embargo, no es necesario que coincidan punto por punto. Es decir, aunque no se correspondan exactamente en todos los aspectos, deben transmitir la misma idea de manera general. Esta flexibilidad interpretativa

es fundamental para que la ACFL sea una teoría lógica aplicable y comprensible en diferentes campos y disciplinas, facilitando su adaptación a diversos contextos y su aplicación práctica. Entonces, una ACFL es el resultado de combinar dos teorías lógicas interpretables diferentes.

Esto implica un tratamiento unificado de modificadores, funciones de membresía y variables lingüísticas continuas, lo que permite una mejora en la interpretación. Además, facilita la unión de funciones lógicas asociadas a conectivos Arquimedianos y compensatorios con elementos relevantes de la lógica difusa.

Lo cual permite construir predicados de lógica difusa, definidos a través de parámetros asociados a las funciones de membresía, la lógica difusa compensatoria y la lógica difusa Arquimediana específicas. Con estas características, se logra una manifestación práctica del pluralismo contextual de una ACFL, lo que posibilita la selección de una ACFL particular en consonancia con la proposición universal más verdadera de un predicado. De esta manera, se puede denominar a la ACFL como una teoría lógica difusa pluralista.

4.1.1 Generalización de conceptos de una ACFL

Una ACFL se define como una teoría lógica que consiste en una t-norma Arquimediana y su correspondiente operador compensatorio y una t-conorma Arquimediana y su correspondiente operador compensatorio, la cual es definida a través de una función generadora que permite la generalización de algunos conceptos difusos (González-Caballero et al., 2021)

Esta función genera una forma de S la cual es llamada función sigmoidal generalizada (GSF), así también un modificador lingüístico generalizado (LM) es definido.
Entonces a través de una ACFL que se denota por $L = (c_c, c_t, d_c, d_t, o, n)$ y una función generadora f un *modificador lingüístico generalizado* $m(x, L)$ es definido a través de la siguiente ecuación (González-Caballero et al., 2021):

$$x_L^a = f^{(-1)}\big(af(x)\big) \tag{4.1}$$

Donde $a \in \mathbb{R}^+$ y x_L^a denota $m(x, L)$

Además, cuando se sustituye $f(x)$ por $-g$ $(\ln(x))$ en *ec.* 4.1, se denomina a g como una *función generadora secundaria de L* (SGF). Además, cuando $a = 1$, $f(x) = 0$ y g es una función impar, entonces $x_L^a = f^{(-1)}(0) = e^{-g^{-1}(0)}$.

A partir de esto, teniendo una lógica L y g como su función generadora secundaria se define a:

$$S_g(x) = \frac{1}{1 + e^{-g^{(-1)}(x)}} \tag{4.2}$$

La cual es llamada función sigmoidal generalizada (GSF). Además, haciendo uso de los parámetros $\alpha,\ \gamma \in \mathbb{R},\ \alpha > 0$ en la *ec.* 4.2 se obtiene una función sigmoidal generalizada parametrizada (González-Caballero et al., 2021).

$$S_g(x;\alpha,\gamma) = \frac{1}{1+e^{-g^{(-1)}(\alpha(x-\gamma))}} \quad (4.3)$$

Es así como mediante una GSF g de una ACFL L y el uso de los parámetros $\alpha,\ \gamma \in \mathbb{R}, \alpha > 0$, y $m \in [0,1]$ se obtiene una familia de funciones parametrizada que genera funciones de diferentes formas tales como funciones sigmoideas, sigmoideas decrecientes y convexas. A esta familia se le da el nombre de variable continua lingüística generalizada (GCLV), que se define mediante la siguiente ecuación:

$$GCLV_L(x;\alpha,\gamma,m) = \frac{C_T\left(S_g(x;\alpha,\gamma)_L^m, \left(1-S_g(x;\alpha,\gamma)\right)_L^{1-m}\right)}{M} \quad (4.4)$$

Donde c_t es una t-norma continua Arquimediana en L y M es el máximo de $c_t\left(S_g(x;\ \alpha,\gamma)_L^m, \left(1-S_g(x;\ \alpha,\gamma)_L^{1-m}\right)\right)$ en $\mathbb{R}$ (González-Caballero et al., 2021).

El nombre GCLV es justificado debido a las diferentes formas que esta familia puede generar a través de sus parámetros, siendo así que puede representar una variable lingüística en lugar de un conjunto de valores lingüísticos. Funciones convexas y no simétricas forman también parte de esta familia dando como resultado que por primera vez una familia de funciones puede representar gran variedad de valores lingüísticos (González-Caballero et al., 2021).

Lo cual representa una ventaja en el uso de esta teoría, debido a que esta familia genera funciones que son capaces de ajustarse a un conjunto de datos modelando así de manera más exacta la información, permitiendo la resolución de problemas de optimización en el espacio de parámetros continuos, así también, permite expresar los resultados en función del lenguaje natural donde se tiene la capacidad de incluir todos los valor lingüísticos posibles o al menos los necesarios (González-Caballero et al., 2021).

Así también mediante una SGF g de una ACFL L, se calcula el cuarteto de operadores (c_t, c_c, d_t, d_c) mediante las siguientes ecuaciones:

$$\mathrm{c_t}(x_1,x_2) = e^{-g^{(-1)}\left(-g\left(ln(x_1)\right)-g\left(ln(x_2)\right)\right)} \quad (4.5)$$

$$\mathrm{c_c}(x_1,x_2,\dots,x_n) = e^{-g^{(-1)}\left(\frac{-\sum_{i=1}^{n} g\left(ln(x_i)\right)}{n}\right)} \quad (4.6)$$

$$\mathrm{d_T}(x_1,x_2) = 1 - e^{-g^{(-1)}\left(-g\left(ln(1-x_1)\right)-g\left(ln(1-x_2)\right)\right)} \quad (4.7)$$

$$\mathrm{d_c}(x_1,x_2,\dots,x_n) = 1 - e^{-g^{(-1)}\left(\frac{-\sum_{i=1}^{n} g\left(ln(1-x_i)\right)}{n}\right)} \quad (4.8)$$

Entonces, esta función generadora g y los demás términos basados en esta función, cobran sentido a través de las siguientes proposiciones.

Dado g^{-1}: $\mathbb{R} \to \mathbb{R}$ una función impar, ilimitada, estrictamente creciente, continua y cóncava en $\mathbb{R}^+$, tal que $g: [0,1] \to \mathbb{R}$ existe. Son condiciones suficientes y necesarias para las siguientes dos propiedades:

1. $S_g(x;\alpha,\gamma) = \frac{1}{1+e^{-g^{-1}(\alpha(x-\gamma))}}$ es una función sigmoidal $\lim_{x\to+\infty} S_g(x;\alpha,\gamma) = 1$, $\lim_{x\to-\infty} S_g(x;\alpha,\gamma) = 0$ es creciente en $\mathbb{R}$, concavo en $\mathbb{R}^+$, convexo en $\mathbb{R}^-$ y simétrico respecto a 0.
2. $f(x) = -g(\ln(x))$ es la inversa de $f^{-1}(x) = e^{-g^{-1}(x)}$ y generador de una t-norma continua Arquimediana.

Además de los operadores lógicos que se han descrito se hace uso también del operador de implicación. En Espín (Espín-Andrade et al., 2012) se mencionan los siguientes tipos:

- S-Implicación: $I_s(x,y) = d(n(x),y)$, donde *d* y *n* son los operadores de disyunción y negación, respectivamente.
- R-Implicación: $I_R(x,y) = sup\{z \in [0,1]: c(x,z) \leq y\}$, donde c es el operador de conjunción.
- QM-Implicación: también conocida como QL-Implicación $I_{QL}(x,y) = d\big(n(x), c(x,y)\big)$.
- Implicación: El operador satisface un grupo de axiomas, que lo asocian implícitamente con los operadores de conjunción, disyunción y negación.

Y entre estas se sugieren las siguientes:

- Implicación de Reichenbach (S-Implicación): $x \to y = 1 - x + xy$
- Implicación de Klir-Yuan: $x \to y = 1 - x + x^2 y$
- Implicación natural (S-Implicación): $x \to y = d(n(x), y)$
- Implicación generalizada de Zadeh (QM-Implicación): $x \to y = d\big(n(x), c(x,y)\big)$
- Implicación de Yager (A-Implicación): $x \to y = y^x$

Y se define al operador de equivalencia $e(x,y) = c\big(i(x,y), i(y,x)\big)$ que es válido para cualquier operador de implicación y de conjunción (Espín-Andrade et al., 2012).

Otras de las ecuaciones que es importante mencionar es el cuantificador universal y el cuantificador existencial que para un predicado *p* de una lógica *L* se calculan mediante las siguientes ecuaciones:

$$\forall_{x\in U} p(x) = \begin{cases} f^{-1}\left(\frac{1}{n}\sum_{x\in U} f\left(p(x)\right)\right) \; si \; p(x) \neq 0 \; para \; todo \; x \in U \\ 0 \; si \; algun \; x \; p(x) = 0 \end{cases} \quad (4.9)$$

$$\exists_{x\in U} p(x) = \begin{cases} 1 - f^{-1}\left(\frac{1}{n}\sum_{x\in U} f\left(1-p(x)\right)\right) \; si \; p(x) \neq 0 \; para \; todo \; x \in U \\ 0 \; si \; algun \; x \; p(x) = 0 \end{cases} \quad (4.10)$$

4.1.2 Lógica difusa Arquimediana compensatoria basada en una función logarítmica exponencial

Durante el proceso de desarrollo de esta investigación, se propusieron una serie de ACFL, s definidas a partir de una función generadora *f*, sin embargo, se usó principalmente la función generadora $f(x) = -log_b^n(x)$, a partir de la cual es posible definir una lógica difusa Arquimediana compensatoria basada en una función logarítmica-exponencial (ACFL-ELF por sus siglas en Inglés). Donde *b* permite definir la base logarítmica, *n* el valor exponencial y en conjunto determinan la lógica de evaluación. Además, partiendo de la función generadora $f(x)$, se calcula su inversa $f^{-1}(x) = b^{-\sqrt[n]{x}}$, y a su vez se define una función generadora secundaria $g = \frac{x^n}{\ln^n(b)}$ y su inversa $g^{-1} = ln_b \sqrt[n]{x}$.

A través de la función generadora y su inversa, se modifican las ecuaciones de la 4.5 a la 4.8 definiendo los operadores de conjunción y disyunción Arquimedianos y compensatorios basados en una función logarítmica exponencial y expresándolos mediante las siguientes ecuaciones:

$$c_T(x_1, x_2, \dots, x_n) = b^{\sqrt[n]{\sum_{i=1}^n log_b^n(x_i)}} \quad (4.11)$$

$$c_c\left(x_1, x_{2,\dots,} x_n\right) = b^{\sqrt[n]{\frac{\sum_{i=1}^n \log_b^n(x_i)}{n}}} \quad (4.12)$$

$$d_T(x_1, x_2, \dots, x_n) = 1 - b^{\sqrt[n]{\sum_{i=1}^n log_b^n(1-x_i)}} \quad (4.13)$$

$$d_c\left(x_1, x_{2,\dots,} x_n\right) = 1 - b^{\sqrt[n]{\frac{\sum_{i=1}^n \log_b^n(1-x_i)}{n}}} \quad (4.14)$$

Así como también un modificador lingüístico (*ec.* 4.1), una función sigmoidal generalizada parametrizada con α, γ, b y n como sus parámetros (*ec.* 4.3) y la variable lingüística continua generalizada (*ec.* 4.4) definidas mediante una función logarítmica exponencial y expresadas mediante las siguientes ecuaciones:

$$x_L^a = b^{-\sqrt[n]{a - log_b^n(x)}} \quad (4.15)$$

$$S_g(x; \alpha, \gamma) = \frac{1}{1 + b^{-\sqrt[n]{(\alpha(x-\gamma)}}} \quad (4.16)$$

$$GCLV_L(x;\alpha,\gamma,m) = \frac{b^{\sqrt[n]{\log_b^n(s_g(x;\alpha,\gamma)_L^m)+\log_b^n(1-s_g(x;\alpha,\gamma)_L^{1-m})}}}{\max_{x\in\mathbb{R}}\left[b^{\sqrt[n]{\log_b^n(s_g(x;\alpha,\gamma)_L^m)+\log_b^n(1-s_g(x;\alpha,\gamma)_L^{1-m})}}\right]} \quad (4.17)$$

Entonces, a través de una función sigmoidal generalizada (GSF), donde se definen los parámetros α, γ, b y n, siendo b y n los que definen la lógica L y α y γ la distribución y el punto medio de la función sigmoidea, respectivamente, es posible construir diversas funciones lógicas sigmoidales. La morfología de estas funciones se basa en la definición de sus parámetros. En la Figura 4.1, se muestra un ejemplo de esta construcción.

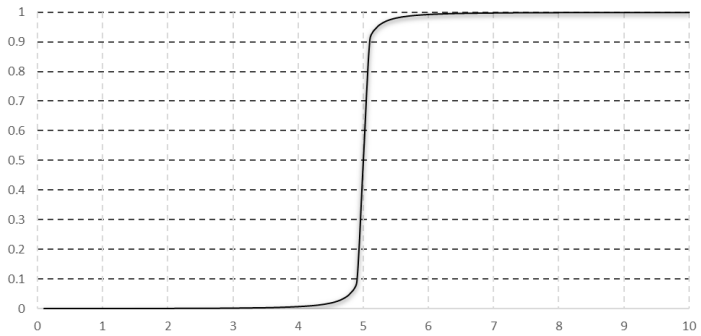


Figura 4.1: Gráfica de una GSF basada en $g(x) = \frac{x^n}{\ln^n(b)}$, donde $b = 10$, $n = 3$, $\alpha = 1$ y $\gamma = 5$.

A partir de una GSF que forma parte de una ACFL L y la definición de los parámetros α, γ y m, una GCLV es capaz de crear una familia de funciones de membresía parametrizadas.

Esta familia abarca desde una función sigmoidea hasta múltiples funciones convexas y una función sigmoidea inversa. En la Figura 4.2, se ejemplifica la construcción de estas funciones de membresía parametrizadas a partir de una GCLV.

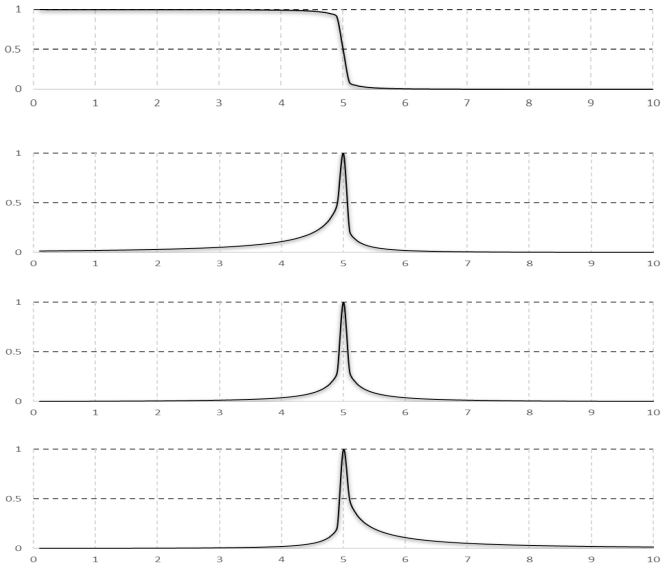


Figura 4.2: De la parte superior a la inferior, se muestran las funciones construidas mediante las siguientes GCLV. $GCLV_L(x; 10, 5, 0)$, $GCLV_L(x; 10, 5, 0.2)$, $GCLV_L(x; 10, 5, 0.5)$ y $GCLV_L(x; 10, 5, 0.8)$.

4.1.3 Aplicación de una ACFL-ELF en el descubrimiento de conocimiento a través de predicados

En esta sección, se aplica una ACFL-ELF al proceso de descubrimiento de conocimiento (KD por sus siglas en Inglés). La relación de cada ACFL-ELF denominada por simplicidad como L con modificadores y funciones de pertenencia permite enriquecer el descubrimiento de conocimiento y su interpretación lingüística.

Para llevar a cabo tales experimentos, se adapta el algoritmo GA-GP, originalmente diseñado para trabajar con una lógica difusa compensatoria (CFL) presentado en (Llorente-Peralta et al., 2021), para que pueda operar con una ACFL-ELF. Este algoritmo se detalla en (Espín-Andrade et al., 2021), y permite realizar simultáneamente el proceso de descubrimiento de conocimiento expresado a través de predicados lógicos y la optimización de las GCLV aplicadas a cada atributo del predicado.

En este caso, no se descubre ningún predicado. En su lugar, para un conjunto de datos se construye un predicado de la forma $(p \leftrightarrow q)$, su proceso de construcción se describe posteriormente. Además, para cada átomo de este predicado, se optimizan los parámetros de su correspondiente GCLV. Este proceso se lleva a cabo utilizando el algoritmo OGCLV modificado de GA-GP, conocido como Optimización de un GCLV definido mediante una Función Logarítmica Exponencial (OGCLV-ELF por sus siglas en Inglés).

La estructura de representación de un individuo en OGCLV-ELF se presenta de la forma siguiente. Se tienen un total de n atributos en p, más el atributo de clase q, que forma parte

del predicado. Cada atributo *i* requiere tres parámetros (α_i, γ_i, m_i) para definir su GCLV. En este caso, se emplea la lógica ACFL-ELF, que requiere dos segmentos para almacenar la base de una función logarítmica *b* y un valor exponencial *e*, utilizado en lugar del exponencial *n* para evitar confusiones. El último segmento se utiliza para contener el valor del cuantificador universal (*ec.* 4.9), nombrado por simplicidad como *valor de verdad* (*TV* por sus siglas en Inglés), por el resultado obtenido con el individuo descubierto sobre el universo de datos.

En la Figura 4.3, se muestra la construcción de un individuo con cuatro atributos. El subíndice de parámetros representa el atributo correspondiente; además, cada individuo de una población se construye seleccionando aleatoriamente los valores de $\alpha_{min}^{max}, \gamma_{min}^{max}$ y m_0^1.

α_1	γ_1	m_1	α_2	γ_2	m_2	α_3	γ_3	m_3	α_4	γ_4	m_4	f	b	e	TV

Figura 4.3: Estructura de representación de un individuo con tres GCLV para el algoritmo OGCLV.

El algoritmo 4.1 corresponde al OGCLV-ELF, que estima los parámetros del predicado de la misma manera que lo hace GA-GP. Sin embargo, la estructura de los individuos utilizados para la evolución y los operadores lógicos utilizados para evaluar un predicado en las líneas 12 y 4 están adaptados a una ACFL-ELF. Las configuraciones utilizadas en el algoritmo OGCLV para los operadores genéticos se mantienen en este algoritmo OGCLV-ELF.

Algoritmo 4.1: OGCLV-ELF

Entrada: *generations*, *max_population*, *predicate.*
Salida: $individual_0$, Contiene el individuo con el mayor *TV*.

```
1.  population = random_population( ); // Inicializa los parámetros del
    predicado
2.  evaluate(population);
3.  for i = 1 to generations
4.      sort(population); // Ordena del mayor al menor TV
5.      for j = 1 to (max_population * 0.95) //Cruza el 95% de la población
6.          if j ≤ (max_population * 0.1) // 10% de los mejores individuos son
            cruzados
7.              kids = cross_deck(individual_j, individual_j+1);
8.          else //Cruza el 85% de los individuos aleatoriamente
9.              kids = kids ∪ cross_deck(individual_rand, individual_rand);
10.     for j = 1 to (max_population * 0.05) // Muta el 5% de los individuos
11.         kids = kids ∪ mutation(individual_rand);
12.     evaluate(kids);
13.     replace(population, kids) //Remplaza los peores individuos con la
        nueva generación
14. return individual_0
```

El cálculo del valor de verdad de un predicado requiere los operadores lógicos descritos en las *ec.* 4.11 a la 4.14, así como un operador de equivalencia definido por la fórmula $e(x,y) = c\big(i(x,y), i(y,x)\big)$, Este operador de equivalencia se construye a través de la fórmula de implicación $I_s(x,y) = d(n(x),y)$, donde d y n son operadores de disyunción y negación respectivamente.

Además, el cuantificador universal, que permite evaluar el valor de verdad de un predicado p de una lógica L en todo el conjunto de datos, se expresa mediante la *ec.* 4.9 descrita en el Algoritmo 4.2.

En el Algoritmo 4.1 para cada individuo descubierto que contiene un conjunto de parámetros, la función de evaluación en la línea 12 invoca al Algoritmo 4.2 para calcular el valor de verdad del predicado de tipo $(p \leftrightarrow q)$. El Algoritmo 4.1 utiliza dichos parámetros y el conjunto de datos para ajustar los parámetros.

Algoritmo 4.2: *True Value*

Entrada: El set de datos con *n* atributos, un atributo de clase, y *r* filas; para cada atributo se incluye la magnitud de *x* y los parámetros (α, γ, m) requeridos para el cálculo de cada GCLV y el valor de verdad definida en la *ec.* 4.9. Los parámetros *e* y *b* de una ACFL-ELF.

Output: *TV*, que es el valor de evaluación de un predicado del tipo $p \leftrightarrow q$.

1: **for** *i* = 1 **to** *r* // Evalúa el predicado para cada registro en el set de datos

2: *antecedent* = $c_T\left(GCLV\left(attrib_{1_i}\right), \ldots, GCLV\left(attrib_{n_i}\right)\right)$ //Operación d conjunción del antecedente *p*

$$= b\sqrt[e]{\log_b^e\left(GCLV\left(attrib_{1_i}\right)\right) + \cdots + \log_b^e\left(GCLV\left(attrib_{n_i}\right)\right)}$$

3: *consequent* = $GCLV\left(attrib_{c_i}\right)$ //valor del consecuente *q* obtenido del atributo clase

4: *imp1* = $1 - b\sqrt[e]{\log_b^e(antecedent) + \log_b^e(consequent)}$ // Implicación de tipo (p → q)

5:

6: *imp2* = $1 - b\sqrt[e]{\log_b^e(consequent) + \log_b^e(antecedent)}$ // Implicación de tipo (q → p)

7: *equivalence* = $b\sqrt[e]{\log_b^e(imp1) + \log_b^e(imp2)}$ // Se calcula la equivalencia

8: *sum_TV* += $\log_b^e(equivalence)$ // Se suma el grado de pertenencia de cada registro mediante la conjunción

9: *TV* = $b\sqrt[e]{\frac{sum_{for_{all}}}{n}}$ // Se aplica la función inversa para calcular el valor de verdad

10: **return** *TV*

Capítulo 4: Métodos propuestos de representación, interpretación, búsqueda e inferencia

La aplicación de estos algoritmos conduce al descubrimiento de conocimiento expresado a través de predicados lógicos, los cuales son interpretables en términos de un lenguaje natural sencillo.

Para realizar el proceso de descubrimiento de conocimiento (KD), se hace uso de un conjunto de sets de datos usados en procesos de clasificación en diversas investigaciones publicadas.

La descripción de los conjuntos de datos se muestra en la Tabla 4.1.

Tabla 4.1: Descripción de los conjuntos de datos usados en el proceso de descubrimiento de conocimiento

Conjunto de datos	Clase Atributo	Numero de registros	Numero de atributos	Ubicación
Car	Nombre: class Tipo: string	1728	7	https://archive.ics.uci.edu/ml/datasets/Car+Evaluation
Dermatology	Nombre: class Tipo: string	366	35	https://archive.ics.uci.edu/ml/datasets/Dermatology
Logistic	Nombre: D-mest Tipo: real	60	29	https://www.dropbox.com/sh/w9e3glio3ngepcp/AAD5i0rbgWwZLGkwOGbK1hMda?dl=0
Glass	Nombre: class Tipo: string	214	10	https://archive.ics.uci.edu/ml/datasets/Glass+Identification
Bupa	Nombre: Drink Tipo: real	345	7	https://archive.ics.uci.edu/ml/datasets/liver+disorders
Indian	Nombre: class Tipo: integer	768	9	http://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/
Iris	Nombre: class Tipo: string	150	5	https://archive.ics.uci.edu/ml/datasets/Iris

Capítulo 4: Métodos propuestos de representación, interpretación, búsqueda e inferencia

La estructura general definida para el proceso de KD en cada conjunto de datos utiliza predicados con la forma "$p\ si\ y\ solo\ si\ q$" lo cual se simboliza como $p \leftrightarrow q$. En este contexto, la estructura de un predicado se describe de la siguiente manera:

$$\forall_{i \in \text{datos}} \left(attrib_i(x) \wedge attrib_{i+1}(x) \wedge \ldots \wedge attrib_n(x) \right) \leftrightarrow class(x).$$

La notación $attrib_i$ representa un atributo modelado mediante una GCLV, lo cual constituye una de las principales fortalezas de una ACFL. En términos simples, cuando hablamos de un "predicado descubierto", se hace referimos al conjunto de parámetros descubiertos que permiten modelar la interpretación de un predicado.

A través de estos experimentos, se busca demostrar que, para cada atributo dentro del conjunto de datos, una GCLV ajusta su forma de acuerdo con el contexto de los datos, lo que dota a la ACFL una alta sensibilidad.

Para cada GCLV se optimizan los parámetros α, γ, *m,* así como también los parámetros b y e que definen a la ACFL-ELF. De esta manera es posible maximizar el valor de verdad de cada predicado, modelando los datos a través de una GCLV explorando diversas lógicas de evaluación.

Con el objetivo de estandarizar los procesos, cada atributo de un conjunto de datos se normaliza en el intervalo $[0{,}100]$. Asimismo, la optimización de los parámetros de una GCLV se limita a los siguientes intervalos: $\alpha \in [0.05 - 3]$ que determina la distribución de la función; $\gamma \in [30 - 60] = 0.5$ que define el punto medio de la función; y $m \in [0{,}1]$ que determina la forma de la GCLV.

Además, se optimizan los datos *b* y *e* correspondientes a una ACFL-ELF, donde $b \in [0{,}7]$ y pertenece al conjunto de los números reales, y $e \in [0{,}15]$ que debe ser un numero impar.
El algoritmo para optimizar y descubrir los parámetros óptimos que modelan la información de los conjuntos de datos se resume en los siguientes pasos:

paso 1. Normalización del conjunto de datos dentro del intervalo [0,100].
paso 2. Resolución de un problema de optimización no lineal con la clase de tipo real:

$$P(\alpha, \gamma, m) = \max_{\alpha,\gamma,m} \Big\{ \forall_{x \in \text{datos}} \Big(GCLV_L\big(x_{\text{attrib}_1}; \alpha_{\text{attrib}_1}, \gamma_{\text{attrib}_1}, m_{\text{attrib}_1}\big) \wedge GCLV_L\big(x_{\text{attrib}_2}; \alpha_{\text{attrib}_2}, \gamma_{\text{attrib}_2}, m_{\text{attrib}_2}\big) \wedge \ldots \wedge GCLV_L\big(x_{\text{attrib}_n}; \alpha_{\text{attrib}_n}, \gamma_{\text{attrib}_n}, m_{\text{attrib}_n}\big) \Big) \leftrightarrow GCLV_L(x_{\text{class}}; \alpha_{\text{class}}, \gamma_{\text{class}}, m_{\text{class}}) \}$$

a. Para una clase entera, el GCLV es una función de membresía Singleton.
b. En esta representación, para cada atributo a optimizar y según el predicado propuesto, el GCLV está asociado a un ACFL-ELF. Se calcula un valor de α, γ, y *m* para cada atributo optimizado por un GCLV. Para la función generadora *f*, los parámetros *b* y *e* son optimizados.

c. Para cada elemento del conjunto de datos, el operador ∧ determina la t-norma asociada con *f*, es decir, la operación de conjunción.
d. El operador de equivalencia ↔ También es definida por f; además se puede observar que una GCLV es definida a través de f.

paso 3. Convertir el conjunto de datos a sus valores iniciales.

El algoritmo OGCLV-ELF lleva a cabo el proceso de optimización con 30 ejecuciones por conjunto de parámetros a descubrir, utilizando un porcentaje de cruce del 80% y una tasa de mutación del 20%.

En la Tabla 4.2 se observan los parámetros α, γ y m descubiertos por el algoritmo OGCLV-ELF, que permiten definir la función de membresía para cada atributo, generada mediante una $GCLV(atributo; \alpha, \gamma, m)$ (*ec.* 4.17). Esta función forma parte de un predicado $Si\ p\ entonces\ q$, donde q se refiere al campo clase y p a cada uno de los atributos que forman parte del conjunto de datos.

En la Tabla 4.3, se presentan los parámetros de la base logarítmica (b) y exponencial (e) descubiertos por el algoritmo OGCLV-ELF, que permiten definir los valores de la función generadora $f(x) = -log_b^e(x)$. Esta función establece la lógica de evaluación utilizada en un predicado $Si\ p\ entonces\ q$, donde q se refiere al campo clase. La calidad del predicado se evalúa a través del cuantificador de universalidad (ec. 4.9), el cual permite evaluar el valor de verdad de un predicado p de una lógica L en todo el conjunto de datos.

Tabla 4.2: Valores de los parámetros obtenidos tras el proceso de optimización.

Conjunto de datos Car

Buying			**Maint**			**Doors**			**Persons**			**Lug Boot**			**Safety**			**Class**
α	γ	m	α	γ	m	α	γ	m	α	γ	m	α	γ	m	α	γ	m	Singleton
0.93	50	0.62	1.52	60	0.41	1.31	48	0.45	0.58	47	0.26	1.43	64	0.54	1.32	74	0.61	Accesible Car (1)
1.11	51	0.66	1.61	48	0.65	1.00	48	0.37	1.06	45	0.54	0.25	59	0.36	1.79	64	0.39	Good car (2)
2.14	46	0.49	0.33	51	0.32	1.86	42	0.55	1.51	60	0.32	1.43	64	0.47	0.79	59	0.52	Unaccessible car (3)
0.45	62	0.47	2.40	55	0.47	0.08	62	0.48	0.65	61	0.36	1.05	62	0.68	2.33	63	0.59	Very good car (4)

Conjunto de datos Dermatology

Erythema			Scaling			Definite borders			Itching			Koebner Phenomenon			Polygonal Papules			Class
α	γ	m	α	γ	m	α	γ	m	α	γ	m	α	γ	m	α	γ	m	Singleton
1.18	48	0.71	0.32	46	0.84	1.39	78	0.55	1.33	77	0.11	1.96	40	0.00	1.72	43	0.03	Chronic dermatitis (1)

2.91	53	0.79	1.90	53	0.91	1.80	40	0.95	0.08	58	0.16	2.77	57	0.99	1.15	45	0.51	Lichen (2)	

Conjunto de datos Logistic

PIB1			PIB2			INF1			INF2			CPD1			CPD2			D Mest		
α	γ	m	α	γ	m	α	γ	m	α	γ	m	α	γ	m	α	γ	m	α	γ	m
1.27	77	0.80	0.19	73	0.97	1.00	55	0.92	1.13	49	0.82	0.80	68	0.00	2.17	57	0.24	0.81	59	0.00

Conjunto de datos Glass

Refractive			Na			Mg			Al			Si			K			Class
α	γ	m	α	γ	m	α	γ	m	α	γ	m	α	γ	m	α	γ	m	Singleton
2.19	75	0.13	0.64	66	0.22	0.71	50	0.68	2.56	70	0.28	0.50	64	0.65	0.41	41	0.18	Wind float (1)
0.54	74	0.56	0.74	70	0.64	1.14	67	0.17	2.95	75	0.46	2.86	79	0.50	2.24	60	0.67	Wind not float (2)
2.23	43	0.47	1.60	42	0.00	2.80	64	0.00	2.33	45	1.00	1.40	76	0.00	1.05	65	0.11	Container (3)
1.62	50	0.57	1.17	74	0.44	0.28	44	0.58	0.54	52	0.53	1.53	61	0.47	1.27	41	0.23	Headlamps (4)
1.99	64	0.46	1.72	57	0.99	3.02	57	0.00	0.75	51	0.11	1.04	65	0.48	0.78	71	0.52	Table ware (5)
0.73	77	0.68	0.78	77	0.79	0.81	47	0.59	0.30	69	0.24	2.74	51	0.37	1.10	59	0.15	Very wind float (6)

Conjunto de datos Bupa

Mcv			Alkphos			Sgpt			Sgot			Gammagt			Drinks		
α	γ	m	α	γ	m	α	γ	m	α	γ	m	α	γ	m	α	γ	m
2.23	76	1.00	1.49	48	0.33	1.44	41	0.31	0.92	40	0.00	0.23	48	0.21	3.03	74	0.68

Conjunto de datos Iris

Sepal lenght			Sepal Width			Petal Lenght			Petal Width			Class
α	γ	m	α	γ	m	α	γ	m	α	γ	m	Singleton
0.56	68	0.41	0.25	74	0.30	0.13	70	0.46	0.35	69	0.36	Setosa (1)
1.06	57	0.43	1.33	52	0.62	1.95	69	0.35	1.52	65	0.44	Virginica (2)
0.35	56	0.41	0.41	44	0.60	0.41	65	0.54	2.25	47	0.55	Versicolor (3)

Conjunto de datos Indian Pima

Times Pregnant			Glucose Concentration			Blood Pressure			Skin thickness			Insulin			BMI			Diabetes
α	γ	m	α	γ	m	α	γ	m	α	γ	m	α	γ	m	α	γ	m	Singleton
2.54	72	0.59	0.59	62	0.66	0.94	61	0.39	0.45	69	0.20	2.22	62	0.25	1.42	79	0.44	Diabetes (1)

Tabla 4.3: Valores de verdad obtenidos a través de la función logarítmica-exponencial ACFL-ELF con parámetros optimizados de la función generadora *f* (base logarítmica *b* y exponente *e*).

Conjunto de datos	Clase	Parámetro	Valor descubierto	Valor de verdad
Car	car accesible	*b*	2.01	0.9304522
		e	15	
	car good	*b*	2.09	0.91626229
		e	15	
	car unaccesible	*b*	2.04	0.92654275
		e	15	
	car very good	*b*	2.03	0.93132237
		e	15	
Dermatology	cronica	*b*	2.02	0.55443044
		e	9	
	lichen	*b*	2.04	0.55790857
		e	9	
Empresa	d mest	*b*	3.23	0.67023823
		e	3	
Glass	wind float	*b*	15	0.84416932
		e	3	
	wind not float	*b*	2.03	0.83581373
		e	15	
	container	*b*	2.01	0.65317164
		e	3	
	headlamps	*b*	2.02	0.8562308
		e	15	
	tableware	*b*	2.44	0.66706136
		e	3	
	v wind float	*b*	2.05	0.79473997
		e	11	
Bupa	drinks	*b*	3.71	0.99516633
		e	1	
Iris	setosa	*b*	2	0.85640946
		e	11	
	versicolor	*b*	2.02	0.87864615
		e	15	
	virginica	*b*	2.12	0.86550224
		e	15	
Indian pima	diabetes	*b*	2.06	0.84735278
		e	15	

La Tabla 4.3 muestra cómo la función generadora de ACFL-ELF se adapta a los datos de manera sensible, permitiendo obtener valores de verdad altos a través del cuantificador universal. A través de este proceso de optimización, los conocimientos obtenidos se pueden expresar en lenguaje natural.

Por ejemplo, para la conjunto de datos Bupa.txt, se utilizan los valores *b* y *e* observados en la Tabla 4.3 y se sustituyen en la función generadora, resultando en la siguiente ecuación:$f(x) = -log_{3.71}^{1}$. Ahora, haciendo uso de los parámetros α, γ y m de la Tabla 4.2, de la Tabla 4.2, se define la función de membresía generada para cada atributo a través de una GCLV (ec. 4.17), de tal manera que el predicado generado es:

$GCLV(Mcv; \alpha = 2.23, \gamma = 76, m = 1)$ y $GCLV(Alkphos; \alpha = 1.49, \gamma = 48, m = 0.33)$ y $GCLV(Sgpt; \alpha = 1.44, \gamma = 41, m = 0.31)$ y $GCLV(Sgot; \alpha = 0.92, \gamma = 40, m = 0)$ y $GCLV(Gammagt; \alpha = 0.23, \gamma = 48, m = 0.21)$ $equivale$ $GCLV(Drinks; \alpha = 3.03, \gamma = 74, m = 0.68)$

El valor de verdad obtenido es de 0.9951, y a través de este predicado se observa la relación existente entre los datos obtenidos a través de pruebas sanguíneas y el número de copas consumidas.

Una vez descubiertos los parámetros óptimos de GCLV y ACFL-ELF para todos los registros del conjunto de datos Bupa, podemos expresar el predicado de equivalencia interpretando los parámetros *m* y γ. La expresión del predicado en lenguaje natural se muestra a continuación. En la Figura 4.4, se observa cómo la GCLV ajusta los parámetros para cada uno de los atributos del conjunto de datos Bupa.

Por cada individuo que presente un volumen corpuscular medio (mcv) que supere (*m* =1) 94 unidades (γ =76) y una fosfatasa alcalina (alkpohos) que tienda (*m* =0.33) a 78 unidades (γ=48) y un alanina aminotransferasa (Sgpt) que tiende (*m* = 0.31) a 41 unidades (γ = 41) y una aspartato aminotransferasa (Sgot) que es inferior (*m* =0) a 36 unidades (γ=40) y una gamma-glutamil transpeptidasa (Gammagt) que tiende (*m* =0.21) a 147 unidades (γ= 48) equivale a una cantidad que tiende (*m* =0.68) a 15 bebidas alcohólicas (γ = 74).

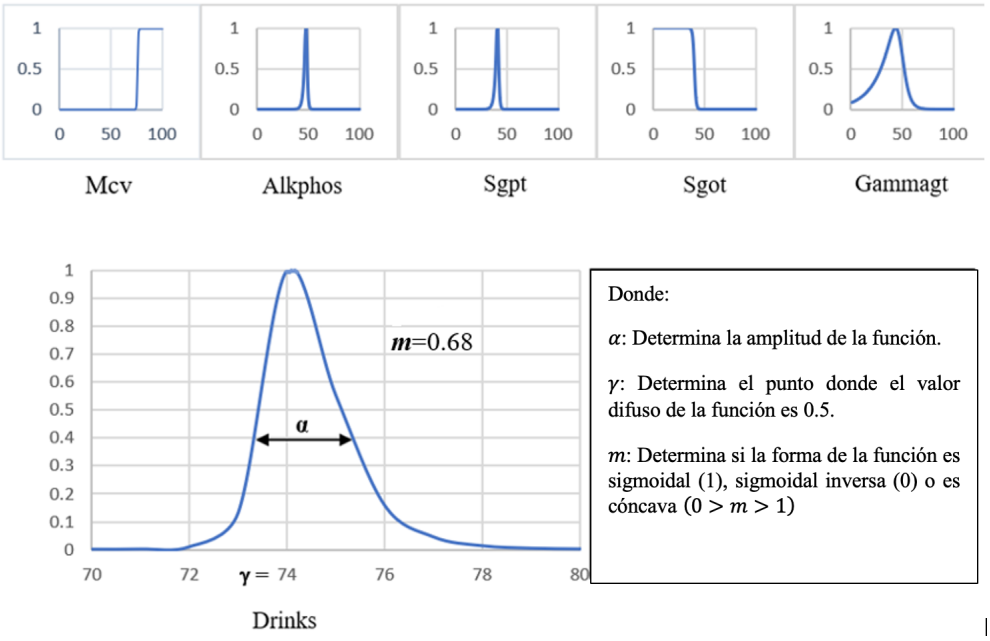


Figura 4.4: GCLVs descubiertas en la optimización de parámetros de un predicado para Bupa.

La Tabla 4.4 ofrece una interpretación de los resultados para cada conjunto de datos, siguiendo el ejemplo presentado anteriormente. Estas interpretaciones se derivan de los parámetros descubiertos y el análisis de los datos, tal como se ilustra con el conjunto de datos Bupa.

Tabla 4.4: Predicados del tipo $p \rightarrow q$ expresados en lenguaje natural a través de los parámetros obtenidos.

Conjunto de datos	**Parámetros de los atributos**											
Bupa	γ	m	γ	m	γ	m	γ	m	γ	m	γ	m
	76.00	1.00	48.00	0.33	41.00	0.31	40.00	0.00	48.00	0.21	74.00	0.68
	Interpretación											
	Por cada elemento que tenga un mcv que supere las 76 unidades y que su Alkphos tienda a 48 unidades y que su sgpt tienda a 41 unidades y que su sgot sea menor de 40 unidades y su gammagt tienda a 48 unidades, equivale a haber consumido alrededor de 74 unidades de alcohol.											
Car	γ	m	γ	m	γ	m	γ	m	γ	m	γ	m
	50.00	0.62	60.00	0.41	48.00	0.45	47.00	0.26	64.00	0.54	74.00	0.61
	Interpretación											

	Por cada elemento que se compra que tiende a 50 unidades y un mantenimiento que tiende a 60 unidades y puertas que tiende a 48 unidades y personas que tiende a 47 unidades y un maletero que tiende a 64 unidades y seguridad que tiende a 74 unidades es igual un coche accesible.
Dermatology	*γ* *m* *γ* *m* *γ* *m* *γ* *m* *γ* *m* *γ* *m* 48.00 0.71 46.00 0.84 78.00 0.55 77.00 0.11 40.00 0.00 43.00 0.03 Interpretación Por cada elemento que tiene eritema que tiende a 48 unidades y una descamación que tiende a 46 unidades y bordes definidos que tiende a 78 unidades y prurito que tiende a 77 unidades y fenómeno de Koebner menor a 40 unidades y pápulas poligonales menores a 43 unidades. equivale a tener dermatitis crónica.
Logistic	*γ* *m* *γ* *m* *γ* *m* *γ* *m* *γ* *m* *γ* *m* *γ* *m* 77.00 0.80 73.00 0.97 55.00 0.92 49.00 0.82 68.00 0.00 57.00 0.24 59.00 0.00 Interpretación Todo elemento que PIB1 tienda a 77 unidades y PIB2 supere 73 unidades y que INF1 supere 55 unidades y que INF2 tienda a 49 unidades y que CPD1 sea inferior a 68 unidades, y que CPD2 tienda a 57 unidades equivale a tener valores inferiores a 59 unidades.
Glass	*γ* *m* *γ* *m* *γ* *m* *γ* *m* *γ* *m* *γ* *m* 75.00 0.13 66.00 0.22 67.00 0.17 75.00 0.46 64.00 0.65 41.00 0.18 Interpretación Por cada elemento que tiene una refracción que tiende a 75 unidades y un Na que tiende a 66 unidades y un mg que tiende a 50 unidades y un ai que tiende a 70 unidades, y un sí que tiende a 64 unidades, y un k que tiende a 41 unidades equivale a ser un vidrio flotado eólico.
Iris	*γ* *m* *γ* *m* *γ* *m* *γ* *m* 68.00 0.41 74.00 0.30 70.00 0.46 69.00 0.36 Interpretación Todo elemento con una longitud de sépalo tiende a 68 unidades y un ancho de sépalo que tiende a 74 unidades y una longitud de pétalo que tiende a 70 unidades, y un ancho de pétalo que tiende a 69 unidades equivale a un iris setosa.
Indian pima	*γ* *m* *γ* *m* *γ* *m* *γ* *m* *γ* *m* *γ* *m* 72.00 0.59 62.00 0.66 61.00 0.39 69.00 0.20 62.00 0.25 79.00 0.44 Interpretación

	Por cada mujer que tiene tiempos de embarazo que tiende a 72 unidades y una concentración de glucosa que tiende a 62 unidades y una presión arterial que tiende a 61 unidades y un espesor de piel que tiende a 69 unidades e insulina que tiende a 62 unidades, y un IMC que tiende a 79 unidades equivale a tener diabetes.

A través de los resultados se establece una conexión entre la Lógica Difusa Arquimediana compensatoria(ACFL) y los principios básicos de la Lógica Difusa. Se introducen dos conceptos importantes, el Modificador Lingüístico Generalizado y la Función Sigmoidea Generalizada, utilizando la función generadora de ACFL. Tales conceptos permiten definir una familia de funciones de membresía parametrizadas, conocidas como GCLV, que son aplicables a cualquier ACFL. Se demuestra la utilidad de esta asociación en ejemplos prácticos, observando que diferentes lógicas ACFL y funciones GCLV se adaptan mejor a distintos conjuntos de datos.

La mejora teórica de ACFL para integrar conceptos fundamentales de la Lógica Difusa la convierte en una Lógica Pluralista Contextual. Lo cual permite la combinación de enfoques clásicos y de lógica difusa compensatoria en una sola teoría, facilitando la comparación de lógicas para expresar conocimiento contextual específico.

Se demuestra cómo esta teoría puede emplearse para obtener reglas de equivalencia interpretables y precisas, ajustando diferentes aspectos de la ACFL-ELF.

Las reglas expresadas mediante predicados de lógica difusa Arquimediana compensatoria, ofrecen la capacidad de encontrar y describir las relaciones entre cada uno de los elementos que componen el conjunto de datos analizado. A partir dichos predicados, se puede comprender las razones y consecuencias lógicas de estas relaciones. Estas capacidades están estrechamente relacionadas con la analítica descriptiva y el análisis de datos, ya que permite comprender y explicar de manera detallada como interactúan las diferentes variables en el conjunto de datos.

4.2 Sistema axiomático de representación e interpretación de predicados de lógica difusa Arquimediana compensatoria, a partir de las preimágenes de una variable lingüística continua generalizada.

En la sección 4.1, se abordó la lógica difusa Arquimediana compensatoria (ACFL), la cual demuestra ser útil en el proceso de descubrimiento de conocimiento (KD) al permitir la incorporación de una serie de parámetros que permiten generar la "mejor configuración lógica" para cada problema difuso, es decir la mejor definición de parámetros descubierta en cada problema difuso. Se define a la ACFL, como una lógica pluralista debido a su compatibilidad con el enfoque clásico de Normas y Conormas de la teoría de la Lógica Difusa

Compensatoria. ACFL a través de la definición de una función generadora permite generalizar los elementos de una lógica difusa, como funciones de membresía, operadores lógicos y modificadores lingüísticos.

En esta sección se presenta un modelo matemático para la interpretación de predicados de una ACFL en función de la caracterización de la variable lingüística continua generalizada. Este modelo fue formulado por Susana Ruiz y el modelo computacional se materializó en esta tesis. Ambos modelos se presentan en el artículo titulado Characterization of the Generalized Continuous Linguistic Variable defined in the Archimedean Compensatory Fuzzy Logic (Ruiz et al., s/f).

Dentro de la teoría de los conjuntos difusos propuesta por Zadeh, las funciones de membresía son esenciales para modelar la incertidumbre y la vaguedad presentes en muchos problemas del mundo real. A pesar del interés de diversos autores en estas funciones, en muchas ocasiones no se justifica adecuadamente el uso de una función de membresía específica para resolver un problema difuso. En lugar de ello, se tiende a utilizar las funciones de membresía más comunes de forma estándar, sin considerar si son las más apropiadas para el problema en cuestión. Esta práctica puede limitar el potencial de la teoría de conjuntos difusos, reduciendo la precisión y eficacia de las soluciones propuestas.

En la definición de una ACFL, se introduce la Variable Lingüística Continua Generalizada (GCLV), la cual es una función parametrizada que permite generar una amplia gama de funciones de membresía. La estructura de una GCLV se define mediante sus parámetros: α el cual corresponde al grado de dispersión de la función; γ que corresponde al valor 0.5 difuso y m que define la forma de función construida. Las GCLV pueden tomar tres formas principales: sigmoidal, sigmoidal inversa, así como varias funciones convexas. Esta versatilidad permite representar una diversidad de valores lingüísticos que se ajustan a las necesidades del problema difuso en cuestión.

En términos generales, la interpretación de funciones de membresía implica asignar valores lingüísticos a conjuntos de funciones de membresía asociadas con atributos de un conjunto de datos y etiquetas definidas por un experto. Sin embargo, en González-Caballero (2021), se presenta el principio de representación de variables lingüísticas, que establece requisitos para la interpretabilidad de las variables lingüísticas. Este principio implica asociar etiquetas directamente a funciones de membresía específicas, sin la intervención previa de expertos en la definición de un conjunto de funciones de membresía que representen las etiquetas.

Una de las ventajas de la GCLV es su capacidad de ser interpretadas a partir de expresiones algebraicas específicas, las cuales pueden traducirse en frases coherentes según la semántica algebraica. Esta interpretación está basada en proposiciones que se describen a través de funciones de membresía utilizando una serie de preimágenes obtenidas mediante los parámetros α, γ y m. Tal caracterización establece una relación uno a uno entre cada GCLV, las expresiones algebraicas y sus correspondientes frases específicas. Esta interpretación algebraica de una GCLV no depende de las observaciones de los tomadores de decisiones.

Capítulo 4: Métodos propuestos de representación, interpretación, búsqueda e inferencia

A partir de los conceptos involucrados en la ACFL y la definición de una GCLV, se enuncian y demuestran proposiciones para la caracterización de una GCLV, considerando diferentes situaciones según los valores de los parámetros asociados a una GCLV. En estas caracterizaciones se presta especial atención al cálculo de preimágenes para la familia de un GCLV.

Las ventajas presentadas a partir de este proceso de caracterización de una GCLV que forma parte de una teoría lógica de ACFL son las siguientes:

I. Permite expresar una GCLV en base a sus preimágenes, posibilitando su interpretación algebraica.
II. Las propiedades de interpretación de los operadores de ACFL y estas novedosas propiedades de las funciones de membresía generadas a partir de una GCLV, permiten una interpretación completa de cada predicado de ACFL utilizando un lenguaje casi natural.
III. Se define un algoritmo que permite calcular las preimágenes de una GCLV, lo cual permite expresarla en lenguaje natural.

4.2.1 Caracterización de una variable lingüística continua generalizada

Dada la función $g^{-1}: \mathbb{R} \to \mathbb{R}$ la cual es estrictamente creciente, continua, impar, además cóncava en $\mathbb{R}^+$, tal que $g: [0,\ 1] \to \mathbb{R}$ existe y es continua. Sea entonces $S_g(x; \alpha, \gamma) = \frac{1}{1+e^{-g^{-1}(\alpha(x-\gamma))}}$ una función sigmoidal parametrizada y $GCLV_L(x; \alpha, \gamma, m) = \frac{c_T\left(S_g(x;\alpha,\gamma)_L^m, \left(1-S_g(x;\alpha,\gamma)\right)_L^{1-m}\right)}{M}$ una variable lingüística continua generalizada.

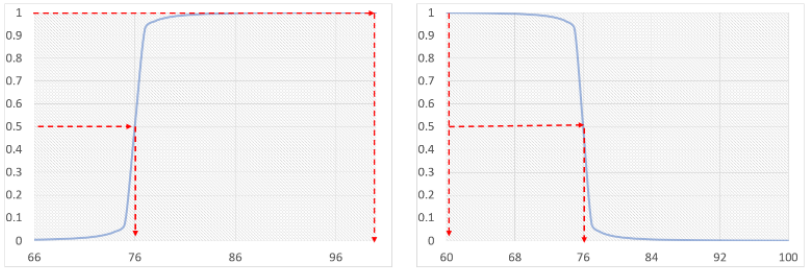
En esta definición, los parámetros $\alpha, \gamma \in \mathbb{R}$, $\alpha > 0$, $m \in [0, 1]$, una función generadora f y una función generadora secundaria g que forman parte de una lógica *L*, donde *M* es el valor máximo de $c_T\left(S_g(x;\alpha,\gamma)_L^m, \left(1-S_g(x;\alpha,\gamma)\right)_L^{1-m}\right)$ en $\mathbb{R}$ y un modificador linguistico generalizado (*ec.* 4.1):

$$x_L^a = f^{(-1)}\big(af(x)\big)$$

Donde sí:
$a = 0$, entonces $x_L^a = 1$
$a = 1$, entonces $x_L^a = f^{(-1)}\big(f(x)\big)$
Entonces para una $GCLV$ de una lógica *L* se pueden definir lo siguiente:

a) Para una $GCLV_L(x; \alpha, \gamma, m)$ donde $m = 0$ ó $m = 1$ (Ver Figura 4.5):
   I. Existe una y solo una preimagen de 0.5.

II. Para toda $\delta > 0$ se satisface que si $y \in (1-\delta, 1)$, entonces existe una sola preimagen de y para $GCLV_L(x; \alpha, \gamma, m)$.



$GCLV_{mcv}(x; \alpha = 2.23, \gamma = 76, m = 1.0)$ $GCLV_{mcv}(x; \alpha = 2.23, \gamma = 76, m = 0)$

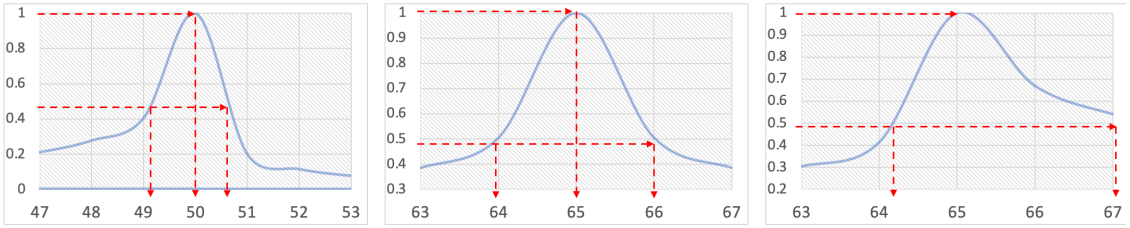
Figura 4.5: Función sigmoidal y sigmoidal inversa con $m = 1$ y $m = 0$.

b) Para $m = 0.5$ y $f^{(-1)} = f^{-1}$ derivable en todo su intervalo:

I. $c_T\left(S_g(x;\ \alpha = 1, \gamma = 0)_L^m,\ \left(1 - S_g(x;\ \alpha = 1, \gamma = 0)\right)_L^{1-m}\right)$ es una función par.

II. $c_T\left(S_g(x;\ \alpha = 1, \gamma = 0)_L^m,\ \left(1 - S_g(x;\ \alpha = 1, \gamma = 0)\right)_L^{1-m}\right)$ alcanza solo una vez el valor máximo donde M=0.5 cuando x=0.

III. $GCLV_L(x; \alpha = 1, \gamma = 0, m = 0.5)$ posee una y solo una preimagen del valor difuso 1.0, y también satisface que existen exactamente dos preimágenes del valor difuso 0.5 (Ver Figura 4.6).



$GCLV_{Drinks}(x; \alpha = 2, \gamma = 50, m = 0.25)$ $GCLV_{Drinks}(x; \alpha = 0.5, \gamma = 65, m = 0.5)$ $GCLV_{Drinks}(x; \alpha = 0.5, \gamma = 65, m = 0.75)$

Figura 4.6: Funciones convexas generadas para $0 < m < 1$

c) Sea $CT(x) = C_T\left(S_g(x;\ \alpha = 1, \gamma = 0)_L^m,\ \left(1 - S_g(x;\ \alpha = 1, \gamma = 0)\right)_L^{1-m}\right)$, $0 < m < 1$ y $f^{(-1)} = f^{-1}$ derivable en cualquier parte de su intervalo.

I. Si $m = 0.5 + \delta$ y $\delta \in (-0.5, 0.5)$, entonces para cada valor real de x, $C_T(x)$, es la imagen de un modelo de combinación lineal de una función continua e impar dada f^{-1}. Que es especificamente $CT(x) = f^{-1}\left(\frac{f\left(S_g(x; 1,0)\right) + f\left(S_g(-x; 1,0)\right)}{2} + 2\delta\frac{f\left(S_g(x; 1,0)\right) - f\left(S_g(-x; 1,0)\right)}{2}\right)$ tal que:

$$\frac{dCT}{dx}(x) = (f^{-1})'\left(\frac{f\left(S_g(x;1,0)\right)+f\left(S_g(-x;1,0)\right)}{2}+2\delta\frac{f\left(S_g(x;1,0)\right)-f\left(S_g(-x;1,0)\right)}{2}\right)\frac{S_g'(x;1,0)}{2}\left[f'\left(S_g(x;1,0)\right) - f'\left(S_g(-x;\ 1,0)\right)+2\delta\ \left(f'\left(S_g(x;1,0)\right)+f'\left(S_g(-x;1,0)\right)\right)\right] \quad (4.18)$$

II. Sea $\delta \in (-0.5,0.5)$. Si existe un valor real x_0 dado que $\frac{dCT}{dx}(x_0) = 0$ entonces; $\delta = -\frac{1}{2}\frac{f'\left(S_g(x_0;1,0)\right)-f'\left(S_g(-x_0;1,0)\right)}{f'\left(S_g(x_0;1,0)\right)+f'\left(S_g(-x_0;1,0)\right)}$

Como consecuencia de lo descrito y considerando que la definición de una $GCLV_L(x;\alpha = 1,\gamma = 0,m)$ es:

Si existe un valor real x_0, tal que; $GCLV_L(x_0;\alpha = 1,\gamma = 0,m) = 1$, entonces $= \frac{1}{2} - \frac{1}{2}\frac{f'\left(\boldsymbol{S_g(x_0;1,0)}\right)-f'\left(\boldsymbol{S_g(-x_0;1,0)}\right)}{f'\left(\boldsymbol{S_g(x_0;1,0)}\right)+f'\left(\boldsymbol{S_g(-x_0;1,0)}\right)}$.

Se demuestra que existe una relación entre el máximo de M en la definición de la función $GCLV_L(x_0;\alpha = 1,\gamma = 0,m) = 1$ y el valor del parametro m perteneciente a un modificador linguistico generalizado. Relación también observada entre la función $GCLV_L(x;\alpha,\gamma,m)$ y los parámetros $\alpha,\ \gamma \in \mathbb{R}$, $\alpha > 0$, y $m \in [0,1]$. En la Figura 4.7 es posible observar que en la expresión que define las funciones $GCLV_L(x;\alpha,\gamma,m) = \frac{c_T\left(S_g(x;\alpha,\gamma)_L^m,\left(1-S_g(x;\alpha,\gamma)\right)_L^{1-m}\right)}{M}$, el valor de M depende de los valores asignados al parámetro m.

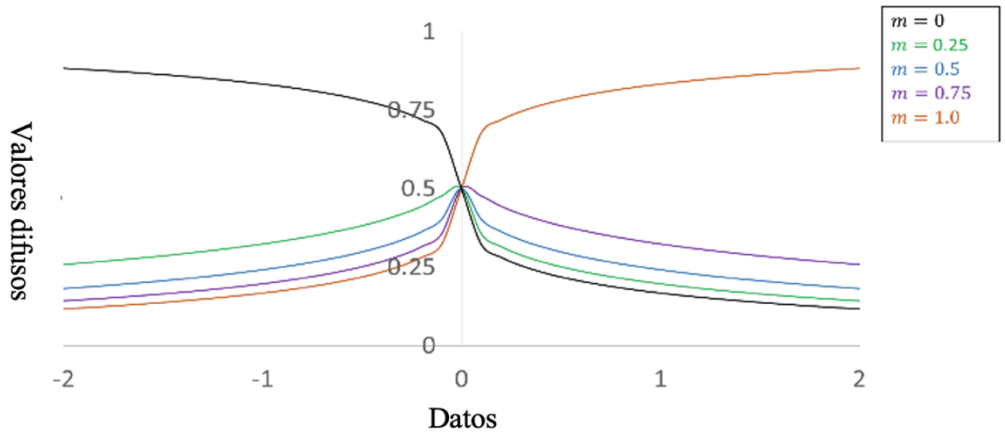


Figura 4.7: Gráfico de la función $C_T\left(S_g(x;\ \alpha,\gamma)_L^m,\ \left(1-S_g(x;\ \alpha,\gamma)\right)_L^{1-m}\right)$, donde: $\alpha = 1,\gamma = 0$ y algunos valores de mϵ[0,1], en este caso se considera $g(x) = \frac{x^n}{(\ln(b))^n}$ con los parámetros b=5 y n=3.

Capítulo 4: Métodos propuestos de representación, interpretación, búsqueda e inferencia

Para caracterizar una $GCLV_L$ asociada con una lógica de $ACFL$ es necesario analizar ciertas funciones y sus derivadas. En este caso, se considera la función $f(x) = -\left(log_b(x)\right)^n$ como función generadora y a la función $g(x) = \frac{x^n}{(\ln(b))^n}$ como la función generadora secundaria de la lógica de $ACFL$ L con $x, b \in \mathbb{R}^+$, $0 < x \leq 1$, $b > 1$ y n como un numero natural impar.

Entonces, dado $f(x) = -\left(log_b(x)\right)^n$ donde $x, b \in \mathbb{R}^+$, $0 < x \leq 1$, $b > 1$ y n es un numero natural impar, entonces se puede demostrar lo siguiente:

Cuando $n = 1$:

$\frac{df}{dx}(x) = -\frac{1}{x\ln(b)}$, $\frac{d^2f}{dx^2}(x) = \frac{1}{x^2\ln(b)}$ y $\frac{d^3f}{dx^3}(x) = -\frac{2}{x^3\ln(b)}$.

Cuando $n > 1$:

$\frac{df}{dx}(x) = -n\frac{\left(log_b(x)\right)^{n-1}}{x\ln(b)}$, $\frac{d^2f}{dx^2}(x) = \frac{-n(n-1)\left(log_b(x)\right)^{n-2} + n\left(log_b(x)\right)^{n-1}\ln(b)}{x^2(\ln(b))^2}$ y $\frac{d^3f}{dx^3}(x) = \frac{\left(-2n\left(log_b(x)\right)^{n-1}(\ln(b))^2 - n(n-1)(n-2)\left(log_b(x)\right)^{n-3} + 3n(n-1)\left(log_b(x)\right)^{n-2}\right)}{(x^3(\ln(b))^3)}$.

Estas funciones son continuas y derivables en $0 < x \leq 1$. Ademas cuando $b > 1$ y n es un numero natural impar, para $0 < x \leq 1$ se cumple que:

i. $\frac{df}{dx}(x) < 0$
ii. $\frac{d^2f}{dx^2}(x) > 0$
iii. $\frac{d^3f}{dx^3}(x) < 0$.

De lo cual se deduce que las funciones $f(x)$ y $\frac{d^2f}{dx^2}(x)$ son decrecientes en el intervalo $(0,1)$, mientras que $\frac{df}{dx}(x)$ es creciente dentro del mismo intervalo. Tambien se demuestra que:

i. $\lim_{x\to 0^+} f'(x) = -\infty$
ii. $\lim_{x\to 1} f'(x) = 0$.

Dada la función:

$$F(x) = -\frac{1}{2}\frac{f'\left(S_g(x;1,0)\right) - f'\left(S_g(-x;1,0)\right)}{f'\left(S_g(x;1,0)\right) + f'\left(S_g(-x;1,0)\right)} \quad (4.19)$$

Para $x\epsilon\mathbb{R}$. $F(x)$ es una función impar debido a que $F(x) = -F(-x)$ para toda $x\epsilon\mathbb{R}$. Entonces a partir de la continuidad de las funciones f' y S_g, y con la consideración de que $\frac{df}{dx}(x) < 0$ en $\mathbb{R}$. Es posible asegurar que $F(x)$ es continua en $\mathbb{R}$.

Debido a que $\lim_{x\to-\infty} Sg(x;1,0) = 0$ y $\lim_{x\to+\infty} Sg(x;1,0) = 1$ lo cual equivale a que $\lim_{x\to+\infty} F(x) = \frac{1}{2}$ y $\lim_{x\to-\infty} F(x) = -\frac{1}{2}$.

En el caso donde $n = 1$:

$$F(x) = \frac{b^x - b^{-x}}{2(2+b^x+b^{-x})} \text{ y } F'(x) = \frac{\ln(b)}{2}\frac{\left((b^x+b^{-x})(b^x+b^{-x}+2)-(b^x-b^{-x})^2\right)}{(2+b^x+b^{-x})^2}$$

Y para los casos donde $n > 1$:

$$F(x) = \frac{\left(1+b^{\sqrt[n]{x}}\right)ln\left(\frac{1}{1+b^{\sqrt[n]{x}}}\right)-\left(1+b^{-\sqrt[n]{x}}\right)ln\left(\frac{1}{1+b^{-\sqrt[n]{x}}}\right)}{2\left(\left(1+b^{\sqrt[n]{x}}\right)ln\left(\frac{1}{1+b^{\sqrt[n]{x}}}\right)+\left(1+b^{-\sqrt[n]{x}}\right)ln\left(\frac{1}{1+b^{-\sqrt[n]{x}}}\right)\right)}$$

Una vez definido lo anterior y entonces, sea $S_g(x;1,0) = \frac{1}{1+e^{-g^{-1}(x)}}$ una función sigmoidal, donde la función generadora se define como $f(x) = -\left(log_b(x)\right)^n$ y una función generadora secundaria $g(x) = \frac{x^n}{(\ln(b))^n}$ que forman parte de una $ACFL\ L$ para las cuales $x, b \in \mathbb{R}^+$, $0 < x \leq 1$, $b > 1$ y n es un numero natural impar.

Se observa que $F(x) = -\frac{1}{2}\frac{f'\left(S_g(x;\mathbf{1},\mathbf{0})\right)-f'\left(S_g(-x;\mathbf{1},\mathbf{0})\right)}{f'\left(S_g(x;\mathbf{1},\mathbf{0})\right)+f'\left(S_g(-x;\mathbf{1},\mathbf{0})\right)}$ para cada $x \in \mathbb{R}$. Entonces para cualquier $\delta\epsilon(-0.5,0.5)$, existe un valor real en x_0 tal que $F(x_0) = \delta$.

Ahora, si $S_g(x;1,0) = \frac{1}{1+e^{-g^{-1}(x)}}$ es una función sigmoidal y $GCLV_L(x;1,0,m) = \frac{c_T\left(S_g(x;1,0)_L^m,\left(1-S_g(x;1,0)\right)_L^{1-m}\right)}{M}$ una variable lingüística continua generalizada definida mediante $m\epsilon(0,1)$, una función generadora $f(x) = -\left(log_b(x)\right)^n$ y una función generadora secundaria $g(x) = \frac{x^n}{(\ln(b))^n}$ que forman parte de una $ACFL\ L$ para las cuales $x, b \in \mathbb{R}^+$, $0 < x \leq 1$, $b > 1$ y n es un numero natural impar, una C_T norma arquimadiana generada por f, donde M es el valor maximo de $c_T\left(S_g(x;1,0)_L^m,\left(1-S_g(x;1,0)\right)_L^{1-m}\right)$ en $\mathbb{R}$ y donde:
$x_L^a = f^{(-1)}\left(af(x)\right)$, donde $a \in \mathbb{R}^+, a \neq 1\ y\ a \neq 0$

Ahora sea $F(x) = -\frac{1}{2}\frac{f'\left(S_g(x;1,0)\right)-f'\left(S_g(-x;1,0)\right)}{f'\left(S_g(x;1,0)\right)+f'\left(S_g(-x;1,0)\right)}$ para cualquier $x\epsilon\mathbb{R}$.

Después, para cada valor de $m \in (0,1)$, la función continua $c_T\left(S_g(x;1,0)_L^m, \left(1-S_g(x;1,0)\right)_L^{1-m}\right)$ alcanza su valor maximo M solamente una vez en un numero real x_0, tal que:

$$1-2m-\frac{f'\left(S_g(x_0;1,0)\right)-f'\left(S_g(-x_0;1,0)\right)}{f'\left(S_g(x_0;1,0)\right)+f'\left(S_g(-x_0;1,0)\right)}=0.$$

Una consecuencia inmediata de lo anteriormente descrito es que para cada valor $m \in (0,1)$, la función continua $GCLV_L(x;1,0,m)=1$ tiene un unico valor real x_0 tal que:

$$-2m-\frac{f'\left(S_g(x_0;1,0)\right)-f'\left(S_g(-x_0;1,0)\right)}{f'\left(S_g(x_0;1,0)\right)+f'\left(S_g(-x_0;1,0)\right)}=0.$$

De esta manera es posible mostrar el análisis de los resultados donde:

$GCLV_L(x;1,0,m)=\frac{C_T\left(S_g(x;1,0)_L^m,\left(1-S_g(x;1,0)\right)_L^{1-m}\right)}{M}=0.5$, considerando que $m \in (0,1)$.

Sea $m=0.5+\delta$ and $\delta \in (-0.5,0.5)$. Entonces$GCLV_L(x;1,0,m)=0.5$ si y solo si $C_T\left(S_g(x;\,1,0)_L^m,\,\left(1-S_g(x;\,1,0)\right)_L^{1-m}\right)=\frac{M}{2}$.

De acuerdo con la definición para una *GCLV* de una lógica *L*:

$GCLV_L(x;1,0,m)=0.5$ si y solo si $\frac{f\left(S_g(x;1,0)\right)+f\left(S_g(-x;1,0)\right)}{2}+2\delta\frac{f\left(S_g(x;1,0)\right)-f\left(S_g(-x;1,0)\right)}{2}=f\left(\frac{M}{2}\right)$ sí y solo si $f\left(S_g(x;\,1,0)\right)=\frac{2f\left(\frac{M}{2}\right)}{1+2\delta}-\frac{f\left(S_g(-x;1,0)\right)}{\left(\frac{1+2\delta}{1-2\delta}\right)}$.

Si x_o es tal que $F(x_0)=\delta$ (ver (*ec.* 4.19)), Entonces $CT(x_0)=M$. Ademas, $GCLV_L\left(x;1,0,0.5-F(x_0)\right)=0.5$ si y solo si:

$$f\left(S_g(x;\,1,0)\right)=\frac{2f\left(\frac{CT(x_0)}{2}\right)}{1+2F(x_0)}-\frac{f\left(S_g(-x;1,0)\right)}{\left(\frac{1+2F(x_0)}{1-2F(x_0)}\right)}$$

4.2.2 Modelo matemático para el cálculo de preimágenes

Para realizar el cálculo de la preimagen de 1.0 que existe dentro de una función de membresía generada a partir de una $GCLV$ (*ec* 4.4) de una $ACFL\ L$, es necesario hacer uso de su definición:

$$GCLV_L(x;\alpha,\gamma,m) = \frac{c_T\left(S_g(x;\alpha,\gamma)_L^m,\left(1-S_g(x;\alpha,\gamma)\right)_L^{1-m}\right)}{M}$$

Donde para los parámetros α y γ se definen valores fijos y el valor de $m \in (0,1)$, de tal manera que se reescribe la ecuación de una $GCLV$:

$$GCLV_L(x;\alpha,\gamma,m) = \frac{c_T(\,x_1(x),\ x_2(x))}{M} \qquad (4.20)$$

Donde:

$x_1(x) = S_g(x;\ \alpha,\gamma)_L^m,$
$x_2(x) = \left(1 - S_g(x;\ \alpha,\gamma)\right)_L^{1-m}$
$M = max_{x\in\mathbb{R}}\,(c_T(\,x_1(x),\ x_2(x))).$

Para este caso se define a $\alpha_z = 1,\ \gamma_z = 0$, y $z = \alpha_z(x-\gamma_z)$, donde $x \in X$. La variable x es normalizada y representada mediante z, entonces una $GCLV\ L$ se reescribe de la siguiente forma:

$$GCLV_L(\mathrm{z} = \alpha_z(x-\gamma_z);\alpha_z = 1,\ \gamma_z = 0,m) = \frac{c_T(\,x_1(z),\ x_2(z))}{M} \qquad (4.21)$$

Donde:
$x_1(z) = S_g(z;\ \alpha_z = 1,\ \gamma_z = 0)_L^m,$
$x_2(z) = \left(1 - S_g(z;\ \alpha_z = 1,\ \gamma_z = 0)\right)_L^{1-m},$
$M = max_{z\in\mathbb{R}}\,(c_T(\,x_1(z),\ x_2(z))).$

En la sección 4.20, se expresa la *ec.* 4.11 en el dominio de z. Esta ecuación define la T-norma como $c_T(x_1,x_2) = e^{-g^{(-1)}\left(-g(ln(x_1))-g(ln(x_2))\right)}$, donde g es la función generadora secundaria.

$$c_T(x_1(z),x_2(z)) = e^{-g^{(-1)}\left(-g\left(ln(x_1(z))\right)-g\left(ln(x_2(z))\right)\right)}. \qquad (4.22)$$

Si $x_1 = x_1(z)$ y $x_2 = x_2(z)$ son funciones continuas y diferenciables, podemos expresar la *ec.* 4.18 en el dominio de z.

$$\frac{d}{dz}\left(c_T(x_1(z), x_2(z))\right) = \frac{d}{dz}\left(e^{-g^{(-1)}\left(-g\left(ln(x_1(z))\right)-g\left(ln(x_2(z))\right)\right)}\right) = e^{-g^{(-1)}\left(-g\left(ln(x_1(z))\right)-g\left(ln(x_2(z))\right)\right)}.\left[-(g^{(-1)})'\left(-g\left(ln(x_1(z))\right) - g\left(ln(x_2(z))\right)\right)\right].\left[-g'\left(ln(x_1(z))\right).\frac{x_1{}'(z)}{x_1(z)} - g'\left(ln(x_2(z))\right).\frac{x_2{}'(z)}{x_2(z)}\right]. \quad (4.23)$$

Considerando que se cumple la propiedad de monotonía de las funciones $g^{(-1)}$, entonces el valor maximo M es igual a cero en la primera función derivada de C_T. Se determina de manera analitica que cuando esta derivada es cero, equivale a considerar solo la primera derivada del argumento $-g\left(ln(x_1(z))\right) - g\left(ln(x_2(z))\right)$, que aparece en la *ec.* 4.22, ya que esta se cancela, como se muestra en la siguiente ecuación.

$$\frac{d}{dz}\left(c_T(x_1(z), x_2(z))\right) = 0$$

$$\text{If } -g'\left(ln(x_1(z))\right).\frac{x_1'(z)}{x_1(z)} - g'\left(ln(x_2(z))\right).\frac{x_2'(z)}{x_2(z)} = 0 \quad (4.24)$$

Para determinar la preimagen de 1, es necesario encontrar el valor de z donde $c_T(x_1(z), x_2(z))$ alcanza su valor maximo M. Con este propósito, primero se encuentra aproximadamente la raíz del antecedente en la *ec.* 4.24, en el que los parámetros son definidos para $\alpha_z = 1,\ \gamma_z = 0$, $0 < m < 1$ y z= $\alpha_z(x - \gamma_z)$, y luego se transforma z nuevamente al dominio x. Lo cual equivale a encontrar el valor de z para el cual $GCLV_L(x = z; \alpha_z = 1, \gamma_z = 0, m) = \frac{M}{M} = 1$. Donde z representa la preimagen del valor difuso 1.

Una vez determinado el valor donde z_0 que corresponde a la preimagen de 1, es decir, donde se alcanza el valor extremo de la función M, este valor se usa en C_T para determinar las preimagenes del valor 0.5 difuso.

$$M = max_{z\in\mathbb{R}}\, C_T\left(S_g(z_0;\ 1, 0)_L^m\ ,\ \left(1 - S_g(z_0;\ 1, 0)\right)_L^{1-m}\right) \quad (4.25)$$

En la *ec.* 4.25 se define un grupo de Ecuaciones equivalentes:

$$GCLV_L(x; 1, 0, m) = 0.5$$

$$\Leftrightarrow \frac{c_T\left(S_g(z;1,0)_L^m, \left(1-S_g(z;1,0)\right)_L^{1-m}\right)}{M} = 0.5$$

$$\Leftrightarrow \frac{c_T\left(S_g(z;1,0)_L^m, \left(1-S_g(z;1,0)\right)_L^{1-m}\right)}{c_T\left(S_g(z_0;1,0)_L^m, \left(1-S_g(z_0;1,0)\right)_L^{1-m}\right)} = 0.5$$

$$\Leftrightarrow \frac{c_T\left(S_g(z;1,0)_L^m,\left(1-S_g(z;1,0)\right)_L^{1-m}\right)}{c_T\left(S_g(z_0;1,0)_L^m,\left(1-S_g(z_0;1,0)\right)_L^{1-m}\right)} - 0.5 = 0 \quad (4.26)$$

En la *ec.* 4.26 tenemos que: z_0 corresponde al valor extremo M de la función $c_T\left(S_g(z_0;1,0)_L^m,\left(1-S_g(z_0;1,0)\right)_L^{1-m}\right)$. Y de la *ec.* 4.26 se tiene que z corresponde al valor de x, para el cual $GCLV_L(x;1,0,m) = 0.5$.

Ahora bien, si z_1 es equivalente a una de las preimagenes de 0.5, esto se define mediante la siguiente ecuación:

$$\frac{c_T\left(S_g(z_1;1,0)_L^m,\left(1-S_g(z_1;1,0)\right)_L^{1-m}\right)}{M} - 0.5 = 0 \quad (4.27)$$

donde $z_1 \in \left[\underline{z_1}, \overline{z_1}\right]$. Tal que $\underline{z_1} = \alpha\left((\min\{X\}) - \gamma\right)$ y $\overline{z_1} = \alpha\left(\left(x_\gamma \in X\right) - \gamma\right)$

De la misma manera, si z_2 equivale a la segunda preimagen del vaor difuso 0.5, esto se define mediante la siguiente ecuación:

$$\frac{c_T\left(S_g(z_2;1,0)_L^m,\left(1-S_g(z_2;1,0)\right)_L^{1-m}\right)}{M} - 0.5 = 0 \quad (4.28)$$

donde $z_2 \in \left[\underline{z_2}, \overline{z_2}\right]$. Tal que $\underline{z_2} = \alpha\left(\left(x_\gamma \in \boldsymbol{X}\right) - \gamma\right)$ y $\overline{z_2} = \alpha\left((\max\{\boldsymbol{X}\}) - \gamma\right)$.

4.2.3 Algoritmos que calculan las preimágenes de una GCLV

En la sección 4.2.1 se estableció, a partir de una serie de ecuaciones matemáticas, que en función del parámetro m de una $GCLV$, existe una unica preimagen de y, donde $y \in (1.0 - \delta, 1.0)$, y una única preimagen de 0.5 cuando $m = 0$ o $m = 1$. Asi como tambien existe una unica preimagen de 1.0 y exactamente dos preimagenes del valor 0.5 difuso cuando $0 > m > 1$.

Por lo consiguiente se presentan un par de algoritmos que, en función del modelo axiomático formulado, permiten calcular las preimágenes de 1.0 y 0.5 existentes para cada $GCLV_L$ definida a partir de sus parámetros α, γ y m.

A través del Algoritmo 4.3es posible estimar la preimagen del valor difuso 1.0 para una función de membresía generada por una $GCLV_L$. Este algoritmo requiere como entrada los parametros α, γ y m que definen la morfología de una *GCLV*, Así como también los parámetros b y n pertenencientes a la función generadora $f(x) = -\left(log_b(x)\right)^n$ de una ACFL-ELF. Además, el algoritmo recibe el atributo modelado mediante la *GCLV*, es decir, la columna de datos x pertenecientes a un conjunto de datos X.

Como resultado, el Algoritmo 4.3 devuelve el valor real x_0 equivalente al valor difuso 1.0, es decir, el valor preimagen de 1.0. específicamente de una $GCLV(x_0; \alpha, \gamma, m) = 1.0$.

Algoritmo 4.3: Preimágenes de 1.0

Entrada: α, γ, m. b y n son las variables de una GCLV para una ACFL especifica; X es el conjunto de datos que define una $GCLV(x; \alpha, \gamma, m)$ para un atributo *x.*

Salida: x_0 que es la preimagen de 1.0

1. //Establecer parámetros para la función normalizada z= $\alpha_z(x - \gamma_z)$, de tal manera que *z* sea representada por *x*
   $\alpha_z = 1, \gamma_z = 0$
2. //Calcular los límites inferior y superior de $\boldsymbol{Z} = [\underline{z}, \overline{z}]$ para la función raíz
   $\underline{z} = \text{norm}(\min\{x\}); \overline{z} = \text{norm}(\max\{x\})$
   // Obtener *z* encontrando la raíz de la *ec.* 4.24, detallada en la Tabla 1.
   $\text{z} \leftarrow \text{root}\left(\left(-g'\left(ln(x_1(z))\right).\frac{x_1'(z)}{x_1(z)} - g'\left(ln(x_2(z))\right).\frac{x_2'(z)}{x_2(z)}\right), \boldsymbol{Z}\right), \text{z} \in \mathbf{Z}$
3. // Determina la preimagen de 1.0
   $x_0 = \frac{z}{\alpha_z} + \gamma_z$
4. **return** (x_0)

El Paso en la línea 1 implica la normalización del atributo x, el cual es un proceso de transformación utilizado en la *ec.* 4.22 para simplificar la formulación del modelo matemático estimado. El siguiente ejemplo ilustra esta transformación. Supongamos que recibimos la $GCLV(x; \alpha = 0.5, \gamma = 5, m = 0.5)$. Necesita ser normalizada en la siguiente función equivalente $GCLV(z;\ \alpha_z = 1, \gamma_z = 0, m)$ (ver Figura 4.8), donde α se transforma en $\alpha_z = 1$, γ se transforma en $\gamma_z = 0$, m permanece sin cambios, y $z = \alpha_z(x - \gamma_z)$. Este proceso de normalización se aplica a cualquier función de membresía generada utilizando una $GCLV$. En la Figura 4.8, el dominio de la función cambia de $[0,10]$ en la primera función al dominio $[-2.5,2.5]$ en la segunda función. Además, el centro de la función se desplaza de $\gamma = 5$ a $\gamma_z = 0$.

La línea 2 calcula los límites inferiores y superiores de z, encontrando el valor mínimo y máximo normalizado de x, respectivamente.

a) $GCLV(x; \alpha = 0.5, \gamma = 5, m = 0.5)$ b) $GCLV(z = \alpha_z(x - \gamma_z);\ \alpha_z = 1, \gamma_z = 0, m = 0.5)$

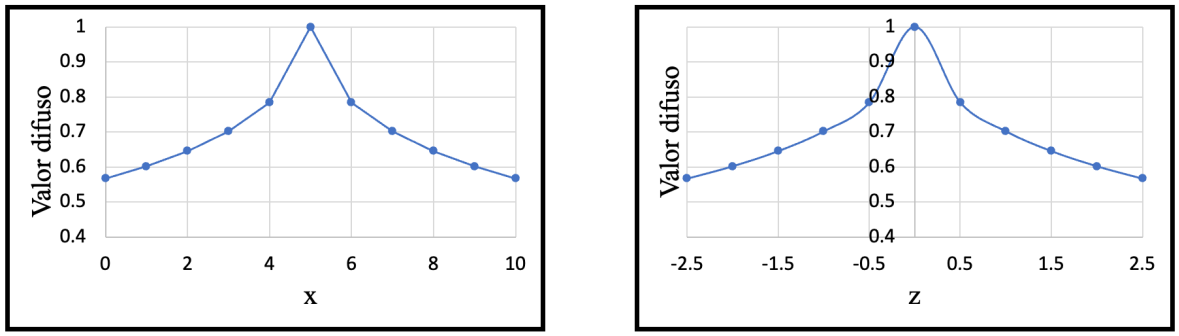


Figura 4.8: Gráficos de la función inicial (a) y después de la normalización de z (b)

Posteriormente, en la línea 3, se determina la preimagen de 1.0 dentro del dominio z calculando la raíz de la *ec.* 4.24, específicamente enfocándose en la parte antecedente de la regla *if*. En la Tabla 4.5 se proporciona un desglose detallado de la parte antecedente de la *ec.* 4.24 para referencia práctica. Una vez obtenido el valor de z, en la línea 5 se convierte de nuevo a su magnitud absoluta utilizando la ecuación $x_0 = \frac{z}{\alpha_z} + \gamma_z$.

Una vez determinada la preimagen de 1, se procede a calcular la preimagen del valor difuso 0.5 a través del Algoritmo 4.4. Este algoritmo recibe la misma información que el Algoritmo 4.3.

En la línea 1, el atributo x se normaliza en el dominio z. La línea 2 calcula tres límites de z: Z se utiliza para determinar el punto que determina el valor extremo de M de una función $GCLV$, Z_1 se utiliza para encontrar la primera preimagen en el lado izquierdo de una GCLV delimitada por γ_z, y Z_2 se utiliza para encontrar la segunda preimagen en el lado derecho de una GCLV delimitada por γ_z; la Figura 4.8 (b) ilustra tales intervalos.

La línea 3 obtiene el punto máximo (M) de la función C_T calculando la raíz de la *ec.* 4.25 dentro del intervalo Z de una $GCLV([-2.5,2.5])$. La línea 4 estima la primera preimagen normalizada z_1 de 0.5 dentro del intervalo negativo Z_1 de una $GCLV([-2.5,0])$, calculando la raíz de la *ec.* 4.27.

La línea 5 verifica si la función $GCLV$ es una función sigmoide o sigmoide inversa; en ese caso, devuelve solo una preimagen de 1.0 en x_1 convirtiendo z_1 a su magnitud absoluta con la ecuación $x_1 = \frac{z_1}{\alpha_z} + \gamma_z$.. De lo contrario, si la GCLV es una función convexa, la línea 6 estima las segundas preimágenes normalizadas z_2 de 0.5 dentro del intervalo positivo Z_2 de la $GCLV([0,2.5])$,, calculando la raíz de la *ec.* 4.28.

Finalmente, en la línea 7, z_1 y z_2 se convierten de vuelta a su magnitud absoluta x_1 y x_2 para devolver tales valores como las preimágenes de 0.5. Las ecuaciones utilizadas en los Algoritmos 4.3 y 4.4 se detallan en la Tabla 4.5.

Algoritmo 4.4: Preimágenes de 0.5

Entrada: *m* es un parámetro de la $GCLV$ que define si la función es convexa o sigmoidea α, γ, *b*, *n*, y *m* son variables de una GCLV para una $ACFL$ especifica.
X es un conjunto de datos que se define mediante una $GCLV(x; \alpha, \gamma, m)$ para un atributo *x* normalizado entre su mínimo y máximo valor.

Salida: x_1 y x_2 son las preimágenes de 0.5

1. // Set de parámetros para la función z= $\alpha_z(x - \gamma_z)$, tal que *z* representa *x*
   $\alpha_z = 1, \gamma_z = 0$
2. //Calcula los limites superior e inferiores de $\boldsymbol{Z} = \left[\underline{z}, \overline{z}\right]$; $\boldsymbol{Z}_1 = \left[\underline{z_1}, \overline{z_1}\right]$; $\boldsymbol{Z}_2 = \left[\underline{z_2}, \overline{z_2}\right]$
   $\underline{z} = \text{norm}(\min\{x\})$; $\overline{z} = \text{norm}(\max\{x\})$;
   $\underline{z_1} = \text{norm}(\min\{x\})$; $\overline{z_1} = \gamma_z$;
   $\underline{z_2} = \gamma_z$; $\overline{z_2} = \text{norm}(\max\{x\})$
3. // El valor máximo de la función C_T denotado por M en el dominio de *z* es calculado mediante la *ec.* 4.25.
   $$M = root\left(C_T\left(S_g(\text{z};\ 1, 0)_L^m,\quad \left(1 - S_g(\text{z};\ 1, 0)\right)_L^{1-m}\right), \boldsymbol{Z}\right), z \in \boldsymbol{Z}$$
4. // La preimagen del valor difuso 0.5 en el dominio de *z* es calculado mediante la *ec.* 4.27.
   $$z_1 \leftarrow root\left(\frac{C_T\left(S_g(z_1;\ 1, 0)_L^m,\ \left(1 - S_g(z_1;\ 1, 0)\right)_L^{1-m}\right)}{M} - 0.5, \boldsymbol{Z}_1\right), z_1 \in \boldsymbol{Z}_1$$
5. **if** *m*=0 or *m*=1 // La función $GCLV$ es sigmoidal.
   return $(x_1 = \frac{z_1}{\alpha_z} + \gamma_z)$ //Retorna solo la preimagen de 0.5
6. // De lo contrario, estima la segunda preimagen de 0.5 en el dominio de *z* usando la *ec.* 4.28.
   $$z_2 \leftarrow root\left(\frac{C_T\left(S_g(z_2;\ 1, 0)_L^m,\ \left(1 - S_g(z_2;\ 1, 0)\right)_L^{1-m}\right)}{M} - 0.5, \boldsymbol{Z}_2\right), z_2 \in \boldsymbol{Z}_2$$
7. // Para una función convexa convierte z_1 y z_2 de regreso a su magnitud absoluta
   $x_1 = \frac{z_1}{\alpha_z} + \gamma_z$
   $x_2 = \frac{z_2}{\alpha_z} + \gamma_z$
8. **return** (x_1, x_2); //Retorna dos preimágenes de 0.5

Tabla 4.5: Ecuaciones para caracterizar una $GCLV$ de una $ACFL - ELF$ permitiendo el cálculo de sus preimágenes

Función generadora de una ACFL	**Derivada de la función generadora de una ACFL**
$\boldsymbol{f(x) = -log_b^n(x)}$	$f'(x) = -\frac{n(log_b(x))^{n-1}}{x \ln(b)}$
Inversa de la función generadora	**Derivada de la inversa de la función generadora**
$\boldsymbol{f^{-1}(x) = b^{\sqrt[n]{x}}}$	$f^{-1\prime}(x) = -\frac{\ln(b)\, x^{1/n-1}\, b^{-\sqrt[n]{x}}}{n}$
Función generadora secundaria de una ACLF	**Derivada de la función generadora secundaria**
$\boldsymbol{g(x) = \frac{x^n}{\ln(b)^n}}$	$g'(x) = nx^{n-1} \ln(b)^{-n}$
Inversa de la función generadora secundaria	**Derivada de la inversa de la función generadora secundaria**
$\boldsymbol{g^{-1}(x) = \ln(b)\sqrt[n]{x}}$	$g^{-1\prime}(x) = \frac{\ln(b)\, x^{1/x-1}}{n}$
Función sigmoidea generalizada	**Derivada de la función sigmoidea generalizada**
$\boldsymbol{Sg(x) = \frac{1}{1+e^{-g^{-1}(x)}}}$	$Sg'(x) = \frac{e^{-g^{-1}(x)}}{1+e^{-g^{-1}(x)}}\, g^{-1\prime}(x)$

Ecuaciones que definen tres intervalos normalizados de una $\boldsymbol{GCLV}$ en los dominios: $\boldsymbol{Z}$, $\boldsymbol{Z_1}$ y $\boldsymbol{Z_2}$. $\boldsymbol{Z} = [\underline{z}, \overline{z}]$ (Dominio entero); $\boldsymbol{Z_1} = [\underline{z_1}, \overline{z_1}]$ (Dominio izquierdo); $\boldsymbol{Z_2} = [\underline{z_2}, \overline{z_2}]$ (Dominio derecho)

Función normalizada: z= $\boldsymbol{\alpha_z(x - \gamma_z)}$ donde $\boldsymbol{\alpha_z = 1, \gamma_z = 0}$
$\underline{z} = \mathbf{norm}(\mathbf{min}\{x\})$; $\overline{z} = \mathbf{norm}(\mathbf{max}\{x\})$;
$\underline{z_1} = \mathbf{norm}(\mathbf{min}\{x\})$; $\overline{z_1} = \gamma_z$;
$\underline{z_2} = \gamma_z$; $\overline{z_2} = \mathbf{norm}(\mathbf{max}\{x\})$

Función $\boldsymbol{GCLV}$ definida en el intervalo normalizado derecho $\boldsymbol{Z_2}$, donde $\boldsymbol{z \in Z_2}$

$\boldsymbol{x_1(z) = f^{-1}\left(-\left(m\, f(Sg(z))\right)\right)}$ (Forma parte de Ecuaciones 4.22, 4.27, y 4.28)

Derivada de la función que define el intervalo derecho normalizado $\boldsymbol{Z_2}$

$\boldsymbol{x_1'(z) = f^{-1\prime}\left(m\, f(Sg(z))\right)\, m\, f'(Sg(z))Sg'(z)}$

Función $\boldsymbol{GCLV}$ definida en el interval normalizado izquierdo $\boldsymbol{Z_1}$, donde $\boldsymbol{z \in Z_1}$

$$x_2(z) = f^{-1}\left(-\left((1-m)\, f(1-Sg(z))\right)\right)$$ (Forma parte de Ecuaciones 4.22, 4.27, y 4.28)

Derivada de la función que define el intervalo normalizado izquierdo Z_1

$$x_2'(z) = f^{-1'}\left((1-m)\, f(1-Sg(z))\right)\,(1-m)\, f'(1-Sg(z))\,(-1\, Sg'(z))$$

M es el valor máximo observado en el intervalo $[0,1]$ de una T-Norma Arquimediana C_T

$$M = C_T\left(S_g(z_0;\, 1,0)_L^m\,,\ \left(1-S_g(z_0;\, 1,0)\right)_L^{1-m}\right)$$ (Ver ec. 4.25)

Ecuaciones que estiman la preimagen de 1.0.

$$x_0 = \frac{z}{\alpha_z} + \gamma_z$$ (Ecuación que transforma el dominio para $\alpha_z = 1, \gamma_z = 0$)

$$\mathbf{z} \leftarrow \mathbf{root}\left(\left(-g'\left(ln(x_1(\bar{z}))\right).\frac{x_1'(z)}{x_1(\bar{z})} - g'\left(ln(x_2(z))\right).\frac{x_2'(z)}{x_2(z)}\right), Z\right), \mathbf{z} \in \mathbf{Z}$$ (Raíz de la *ec.* 4.24)

Ecuaciones que estiman la primera preimagen de 0.5.

$$x_1 = \frac{z_1}{\alpha_z} + \gamma_z$$ (Ecuación que transforma el dominio para $\alpha_z = 1, \gamma_z = 0$)

$$z_1 \leftarrow root\left(\frac{C_T\left(S_g(z_1;\, 1,0)_L^m\,,\left(1-S_g(z_1;\, 1,0)\right)_L^{1-m}\right)}{M} - 0.5, Z_1\right), z_1 \in Z_1$$ (Raíz de la *ec.* 4.27)

Ejemplo:

En el siguiente ejemplo se muestra cómo la caracterización de una GCLV permite interpretar predicados de ACFL en un lenguaje pseudo-natural, lo que significa que facilita la modelación del conocimiento de una manera que se asemeja a la expresividad humana. Para este propósito, se utiliza el siguiente predicado ACFL:

$$GCLV(sepal\ length)\ y\ GCLV(sepal\ width)\ y\ GCLV(petal\ length)\ y\ GCLV(petal\ width) \Longrightarrow iris\ setosa$$

Mediante el uso de un algoritmo genético, se determinaron los parámetros de una GCLV que mejor modelan cada atributo del predicado, los cuales se presentan en la Tabla 4.6. Además, este predicado es evaluado a través de la función generadora $f(x) = log_{2.81}^{5}(x)$.

Tabla 4.6: Parámetros descubiertos para el predicado de ACFL a evaluar.

Sepal length			**Sepal width**			**Petal length**			**Petal width**			**Clase**
α	γ	m	α	γ	m	α	γ	m	α	γ	m	
0.56	68	0.41	0.25	74	0.30	0.13	70	0.46	0.35	69	0.36	Setosa (1)

En la Figura 4.9 se observa de manera gráfica las funciones de membresía generadas a partir de una GCLV, modelada a través de sus parámetros α, γ y m, para cada atributo que forma parte del predicado de ACFL.

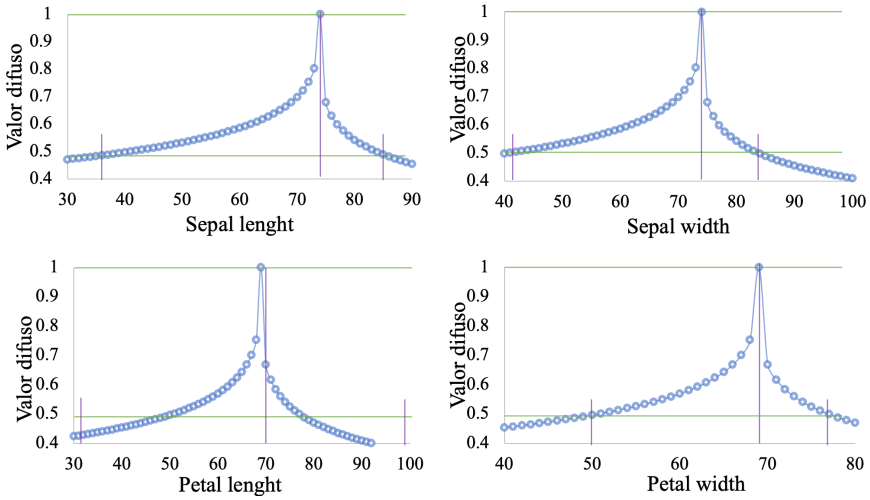


Figura 4.9: Funciones de membresía generadas mediante una GCLV para cada atributo del predicado a evaluar

Una vez determinada esta información, se hace uso de los Algoritmos 4.1 y 4.2, para calcular las preimágenes del valor difuso 1 y 0.5 correspondientes a cada atributo del predicado de ACFL. Los resultados se presentan en la Tabla 4.7.

Tabla 4.7: Preimágenes del valor difuso 1 y 0.5 determinados para cada atributo del predicado de ACFL evaluado

$GCLV_{sepal-length}$			$GCLV_{sepal-width}$		
$\mathbf{1-\delta}$	0.5	0.5	$1-\delta$	0.5	0.5
68	58.43	73.61	74	40.99	83.60

$GCLV_{petal-length}$			$GCLV_{petal-width}$		
$\mathbf{1-\delta}$	0.5	0.5	$1-\delta$	0.5	0.5
70	35.05	97.60	69	50.63	76.94

Los resultados observados para el predicado de ACFL evaluado se expresan de forma matemática (I), posteriormente se traducen a términos de lenguaje natural (II). Esto demuestra las ventajas del proceso de caracterización de una GCLV que forma parte de una ACFL.

I. $\forall\ (sepal\ lenght \cong 68; 58.43 \leq sepal\ lenght \leq 73.61)\ \wedge (sepal\ width \cong 74; 40.99 \leq sepal\ width \leq 83.6)\ \wedge (petal\ length\ \cong 70; 35.05 \leq$

$petal\ lenght \leq 97.6) \wedge (petal\ width \cong 69; 50.63 \leq petal\ width \leq 76.94) \implies iris\ setosa$

II. Para cada flor cuya longitud del sépalo es aproximadamente igual a 68 o está en el intervalo de 58.43 a 73.61, y cuyo ancho del sépalo es aproximadamente igual a 74 o está en el intervalo de 40.99 a 83.6, y cuya longitud del pétalo es aproximadamente igual a 70 o está en el intervalo de 35.05 a 97.6, y cuyo ancho del pétalo es aproximadamente igual a 69 o está en el intervalo de 50.63 a 76.94, implica que es una flor de iris de setosa.

Como se observa, se ha propuesto un sistema axiomático para la caracterización de GCLV. Este modelo propone definir las funciones de membresía generadas a partir de una GCLV en función de las preimágenes de los valores difusos 1.0 y 0.5, lo que facilita su interpretación.

Además, se ha aprovechado la capacidad de interpretación de los operadores lógicos, junto con las propiedades descritas en este documento para las funciones de membresía de una ACFL, lo que permite traducir el modelo matemático extraído a un lenguaje cercano al natural.

Asimismo, se han implementado dos algoritmos para determinar las preimágenes de las funciones de membresía generadas a partir de una GCLV. Dichos algoritmos se han probado en un ejemplo práctico y han permitido expresar el conocimiento descubierto a través de un algoritmo genético en términos de su interpretación natural.

4.3 Modelos de redes neuronales de lógica difusa Arquimediana compensatoria

Durante el desarrollo del presente proyecto, se exploraron diversos modelos y estrategias para la construcción de una red neuronal híbrida basada en lógica difusa Arquimediana compensatoria (ACFL). De tales planteamientos, se presentan dos modelos resultantes que, mediante diferentes paradigmas, llevan a cabo procesos de clasificación y estimación, contribuyendo así al avance de la analítica de datos en el ámbito del aprendizaje automático.

El primer modelo expone una red neuronal del tipo propagación hacia adelante, donde se incorpora una función de activación basada en una variable lingüística continua generalizada (GCLV), un concepto propio de una ACFL. A diferencia de las funciones de activación convencionales, como las sigmoidales, gaussianas o z, que están limitadas por su forma y definición, las GCLV pueden generar una familia de funciones que se ajustan a las particularidades del problema. Este enfoque dota a la red de una sensibilidad única, aspecto que se ha abordado de manera limitada en estudios relacionados.

En el segundo modelo, se propone una red neuronal híbrida basada en ACFL, empleando la estructura de una red neuronal de propagación hacia adelante. En este caso, se sustituyen los

conceptos de suma ponderada, pesos y función de activación por los de GCLV, modificador lingüístico, conjunción y disyunción de una ACFL. Esta implementación posibilita la obtención de un predicado interpretable de ACFL en cada uno de los nodos, lo que permite procesos de clasificación e inferencia interpretables.

El objetivo de los modelos propuestos es descubrir predicados de lógica difusa Arquimediana compensatoria que cumplan con los criterios mencionados en la descripción del problema de investigación.

Dentro de estos criterios, se espera que los predicados sean descubiertos de manera exhaustiva, lo cual significa que el modelo propuesto realizará una búsqueda profunda en el universo de predicados para identificarlos. Además, se espera que los predicados descubiertos sean diversos en su construcción.

Entonces a través de los predicados descubiertos en el proceso de entrenamiento, el modelo de la red neuronal híbrida debe ser capaz de llevar a cabo procesos de inferencia con un buen nivel exactitud. Lo cual significa que las predicciones realizadas a partir de dichos predicados podrán proporcionar resultados confiables.

En resumen, esta sección introduce un enfoque novedoso que combina lo mejor de las redes neuronales de propagación hacia adelante con la lógica difusa Arquimediana compensatoria. El objetivo del modelo propuesto es lograr un descubrimiento exhaustivo y diverso de predicados, facilitando procesos de inferencia eficientes e interpretables.

4.3.1 Red neuronal que implementa una variable lingüística continua generalizada (GCLV) como función de activación

Este modelo propone la aplicación de la topología de una red neuronal tipo propagación hacia adelante. Donde, para cada neurona, se emplea la función de combinación lineal clásica: $z = w_1.x_1 + w_2.x_2 + \cdots + w_n.x_n + b$, junto con una $GCLV$ (ec. 4.4) como función de activación, la cual se denomina con el nombre de NN-GCLV. La arquitectura de esta propuesta se ilustra en la Figura 4.10.

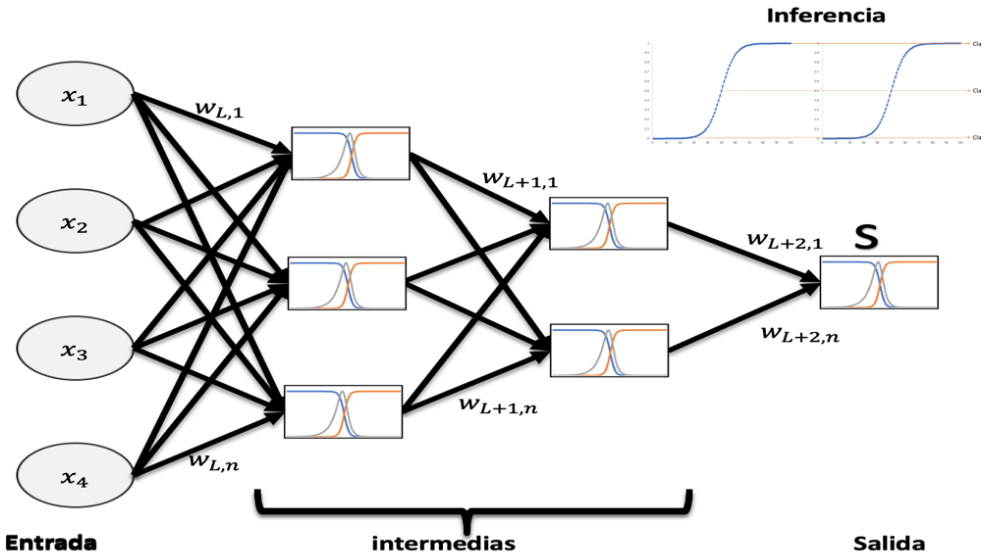


Figura 4.10: Arquitectura de una red neuronal de variables lingüísticas continuas generalizadas.

Tal como se ha definido previamente en la sección 4.1, la GCLV es un concepto propio de la lógica difusa Arquimediana compensatoria, cuyos principios se derivan de una función generadora, denotada como f. Debido a lo cual, es posible generar una familia de funciones (Ver Figura 4.11.) que abarcan desde sigmoidales positivas, pasando por una variedad de funciones convexas, hasta sigmoidal inversa.

Esta familia se genera ajustando los parámetros de la GCLV, lo que posibilita:

$\boldsymbol{\alpha}$**:** Ajustar la concentración de los valores de la función alrededor de los limites [0,1] (sharpness).

$\boldsymbol{\gamma}$**:** Determinar el punto central de una función generada.

$\boldsymbol{m}$**:** Determina el perfil de la función, que puede ser convexa ($m \in (0,1)$), sigmoidal ($m = 1$) o sigmoidal inversa ($m = 0$).

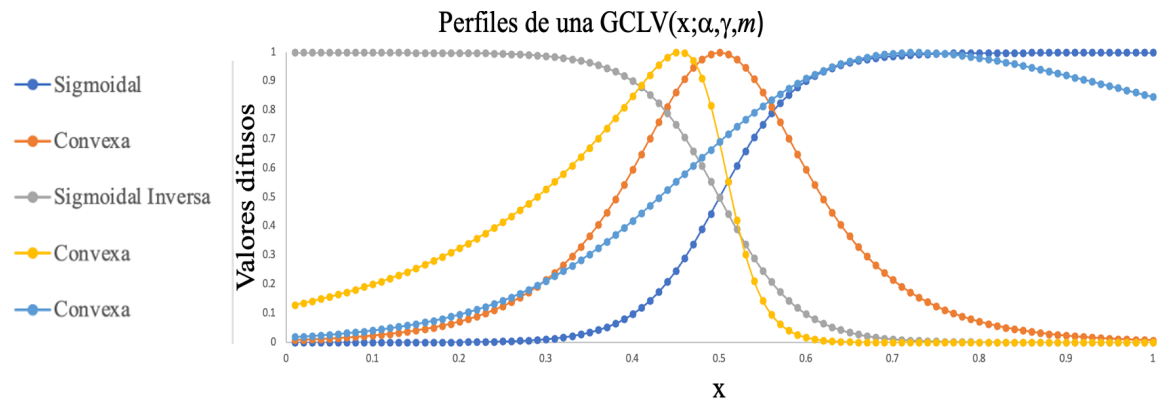


Figura 4.11: Familia de funciones generadas por una *GCLV*.

Dentro del ámbito de las redes neuronales, las funciones de activación permiten la introducción de no linealidades en el modelo. Esta característica dota a la red neuronal de la capacidad de aprender patrones y representaciones complejas en los datos. En la literatura especializada, se encuentran diversas funciones de activación, entre las cuales se mencionan:

I. **Sigmoidal:** Definida principalmente para modelos de clasificación binaria debido a su capacidad para generar salidas en el rango [0, 1].
II. **ReLU** (Rectified Linear Unit): Utilizada en problemas de clasificación y regresión, esta función establece que, si la entrada es positiva, se devuelve tal cual; de lo contrario, se devuelve cero.
III. **Softmax:** Eficiente en problemas de clasificación multiclase, esta función calcula la probabilidad de pertenencia de cada clase.

Debido a que cada una de estas funciones está limitada a su perfil específico se vuelve indispensable elegir la función adecuada para el propósito y la naturaleza particular del problema que se está abordando.

No obstante, al emplear la GCLV como función de activación, la necesidad de seleccionar una función específica disminuye significativamente. Debido a que, dada la naturaleza de la GCLV que no está limitada a un perfil predefinido, ofrece una mayor flexibilidad que la observada en otras funciones de activación. Una vez mencionadas estas consideraciones en relación con la GCLV, se comienza con la descripción de la arquitectura de la red observada en la Figura 4.10.

Capa inicial (0): Generalmente conocida como la capa de entrada de datos, en esta etapa, el conjunto de datos destinado a ser utilizado como modelo de prueba en el proceso de inferencia o clasificación es recibido (ver Figura 4.12). Cada característica del conjunto de datos se transforma al intervalo [0,1] (ver Tabla 4.8) esta operación evita que la variabilidad de las magnitudes influya en el proceso de entrenamiento, mejora el proceso de convergencia, proporciona una mayor estabilidad al algoritmo y evita problemas de sensibilidad en los datos

(Géron, 2023), además Llorente Peralta se hace la observación de que normalizar las propiedades de un conjunto de datos a un determinado intervalo, permite homogeneizar la definición de los parámetros a descubrir para una lógica difusa compensatoria (Llorente-Peralta et al., 2021), el procedimiento de manejo de la entrada de datos se describe en detalle en el Anexo A.

Tabla 4.8: Normalización de datos de entrada al intervalo [0,1].

	Datos de entrada			**Normalización [0,1]**		
No. Registro	**Atributo 1**	**Atributo 2**	**Atributo 3**	**N(Atributo 1)**	**N(Atributo 2)**	**N(Atributo 3)**
1	1	0.1	1000	0.0	0.0	0.0
2	2	0.2	2000	0.1	0.1	0.1
3	3	0.3	3000	0.2	0.2	0.2
4	4	0.4	4000	0.3	0.3	0.3
5	5	0.5	5000	0.4	0.4	0.4
6	6	0.6	6000	0.6	0.6	0.6
7	7	0.7	7000	0.7	0.7	0.7
8	8	0.8	8000	0.8	0.8	0.8
9	9	0.9	9000	0.9	0.9	0.9
10	10	1	10000	1.0	1.0	1.0

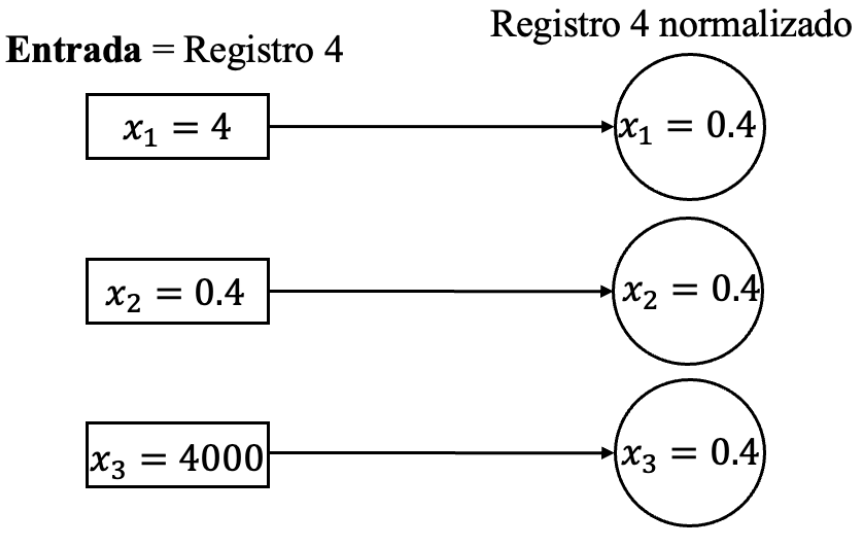


Figura 4.12: Ejemplo de normalización de datos al intervalo [0,1].

Capas intermedias: En estas capas, la información de entrada de cada nodo en la capa L, proviene de la capa anterior $L-1$. Donde cada nodo en la capa $L-1$ es afectado por un factor de peso denominado como w, el cual determina el impacto que ese nodo tiene en el modelo específico. Además, todos los nodos son influenciados por un factor de sesgo (b), el resultado de esta combinación se expresa mediante la *ec.* 2.2:

$z_i = \sum_{j \in T_i^{-1}} w_{i,j} x_j + b_i$

Esta ecuación representa un modelo lineal que genera una frontera para dividir la información en sectores.

Una vez completado el proceso de definición del modelo lineal, se introduce una función que añade no linealidad al modelo. Esta función es comúnmente conocida como función de activación, la cual se selecciona de acuerdo con las necesidades específicas del problema.

Algunas de las funciones de activación más comunes son ReLU, softmax, sigmoidal, entre otras. Sin embargo, en este modelo se propone implementar una GCLV (*ec.* 4.17) perteneciente a una ACFL-ELF como función de activación. Lo que permite no solo optimizar el modelo lineal y algunos atributos de la función de activación, sino también generar la función de activación que mejor se adapte al problema, sin limitaciones en la generación de su perfil (ver Tabla 4.9). Un ejemplo de esta característica se observa en la Figura 4.13.

Tabla 4.9: Ejemplo del cálculo de una neurona en una NN-GCLV.

Normalización [0,1]			**Modelo lineal(z)**			**Función de activación ($GCLV$)**		
N(Atributo 1)	**N(Atributo 2)**	**N(Atributo 3)**	N_1	N_2	N_3	N_1	N_2	N_3
0.0	0.0	0.0	0.900	0.200	0.600	0.000	0.041	1.000
0.1	0.1	0.1	1.011	0.378	0.756	0.001	0.097	0.996
0.2	0.2	0.2	1.122	0.556	0.911	0.004	0.227	0.979
0.3	0.3	0.3	1.233	0.733	1.067	0.028	0.509	0.907
0.4	0.4	0.4	1.344	0.911	1.222	0.138	0.913	0.681
0.6	0.6	0.6	1.456	1.089	1.378	0.406	0.913	0.320
0.7	0.7	0.7	1.567	1.267	1.533	0.708	0.509	0.094
0.8	0.8	0.8	1.678	1.444	1.689	0.892	0.227	0.022
0.9	0.9	0.9	1.789	1.622	1.844	0.971	0.097	0.005
1.0	1.0	1.0	1.900	1.800	2.000	1.000	0.041	0.001

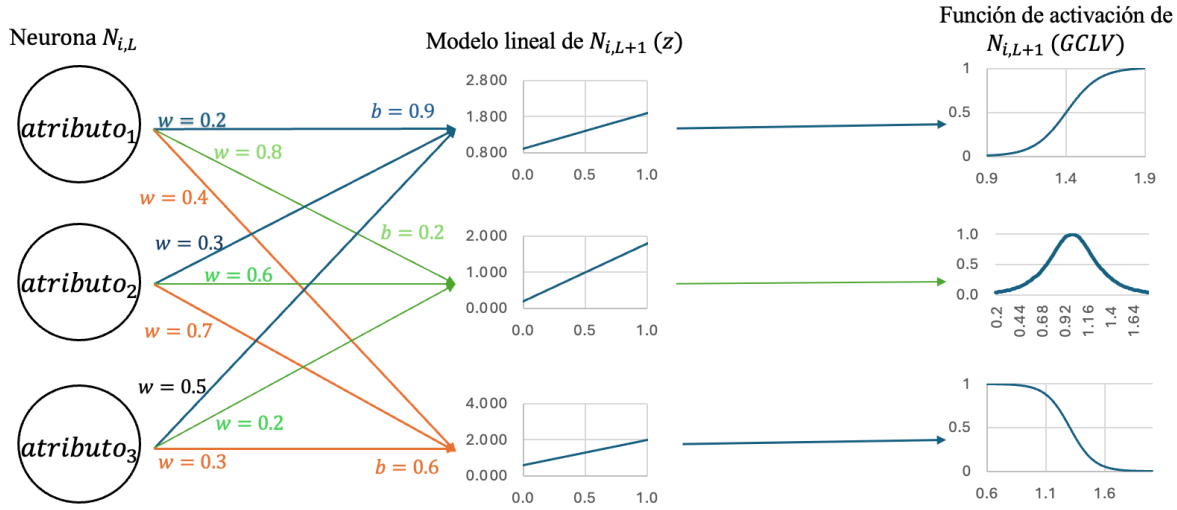


Figura 4.13: Ejemplo de funcionamiento de las neuronas de las capas intermedias.

Capa final: Durante el proceso de tratamiento de datos, el campo normalizado [0,1] que corresponde a las clases a inferir, se modela a través de la siguiente $GCLV(clase, \alpha = 1, \gamma = 0.5, m = 1)$, lo cual define una función sigmoidal con centro en el valor 0.5 y una amplitud igual a la unidad, lo que permite generar una GCLV de clases.

La capa de salida genera una neurona que busca encontrar una $GCVL(x, \alpha = ?, \gamma = ?, m = ?)$, que permite la inferencia de esa función de salida. Es decir, a través del modelado de esta neurona mediante el descubrimiento de los parámetros α, γ y m, se genera una función de salida que se asemeja a la función de clases. Con base a esta función, se logra realizar predicciones e inferencia de clases. En la Figura 4.10 se puede observar de manera gráfica un ejemplo.

Una vez explicado el funcionamiento básico de la arquitectura propuesta, en el Algoritmo 4.5 se presenta la implementación de una red neuronal artificial que utiliza variables lingüísticas continuas generalizadas como funciones de activación. Este modelo surge de pruebas previas en la construcción de un modelo híbrido de red neuronal con una ACFL-ELF como modelo de evaluación.

Algoritmo 4.5: NN-GCLV

Entrada: *Dataset* // Conjunto de datos a evaluar
Salida: *Inferencia* // Conjunto de datos predichos, reales y su diferencia absoluta
Parámetros //Conjunto de parámetros descubiertos en el proceso de optimización
Métricas // Conjunto de métricas de evaluación de rendimiento del algoritmo

1. *Data*, *Mínimo*, *Máximo* = read_dataset("*Dataset*.csv") //Normaliza los datos del archivo csv al intervalo [0,1] y devuelve el valor máximo y mínimo de la clase

```
2. inputs ← fila[:-1] para cada fila en data //Guarda en inputs los datos usados en el entrenamiento
3. labels ← fila[-1] para cada fila en data //Guarda en labels los valores reales de los registros
4. network ←Net() //Crea una instancia del modelo de red neuronal
5. criterio ← Funcion_error() // Define la función de error
6. optimizer ← Algoritmo_optimizacion(network.parameters) //Se establece el algoritmo de retropropagación usado en la optimización
7. Para cada época en épocas
8.     outputs = network(inputs) //Devuelve los valores inferidos por la red
9.     loss = criterio(outputs, labels) //Calcula el error de la red
10.    optimizer.gradienteszero //Reinicia los gradientes
11.    loss.backward() //Retro propaga el error
12.    optimizer.actualizar() //Calcula los nuevos valores de los parámetros
13. imprimir_resultados(network(inputs), labels, Maximo, Minimo)
14. params = network.state_dict()
15. imprimir_parametros(params)
```

El algoritmo 4.5, permite la implementación de una NN-GCLV, una red neuronal adaptativa basada en funciones GCLV (*ec.* 4.17). Esta red ofrece una alta adaptabilidad durante los procesos de entrenamiento, ya que no está definida por un perfil preestablecido de funciones de activación. En su lugar, permite descubrir la función de activación óptima que mejor modela la arquitectura de la solución en cuestión.

El Algoritmo 4.5 ha sido desarrollado en el lenguaje de programación Python, utilizando las bibliotecas PyTorch y Scikit-learn. Además, se han implementado las bibliotecas ActivationsFunctions, datareader y resultados en este desarrollo. Conceptos importantes de la arquitectura, como la descripción de las bibliotecas, los métodos implementados y las métricas de evaluación, entre otros aspectos, se detallan en el Anexo A.

En la línea 1 del código, se recibe el conjunto de datos en formato CSV. Este archivo es procesado mediante la función read_dataset, la cual pertenece a la biblioteca datareader. Esta función, detallada en el Anexo A, se encarga de normalizar los datos de cada campo del conjunto de datos al intervalo [0,1] y guarda los resultados en *Data*. Además, devuelve los valores máximos (*Máximo*) y mínimo (*Mínimo*) del campo clase, los cuales serán necesarios para la evaluación de los resultados al final del proceso.

En la línea 2 del algoritmo, se genera una matriz llamada *inputs* donde se guardan los campos desde 1 hasta *n-1* de la información contenida en *Data*. Estos campos corresponden a los atributos que serán utilizados en el proceso de optimización llevado a cabo mediante la NN-GCLV.

En la línea 3, al igual que la anterior, se genera una matriz denominada *labels*, donde se almacenan los datos del campo *n* almacenados en *Data* que corresponden a las clases a inferir. Tales datos se utilizan para calcular el error de la NN-GCLV y permiten realizar el proceso de optimización a partir del cálculo del error.

En la línea 4 se instancia la clase Net y se asigna a la variable network, que representa la arquitectura de la red neuronal propuesta. Esta arquitectura es utilizada en el proceso de entrenamiento y evaluación del modelo de aprendizaje automático. La clase Net encapsula la arquitectura de una red neuronal en la que pueden definirse dos o más capas lineales y funciones de activación GCLV.

En la línea 5, se define la función de pérdida que se utilizará para calcular el error entre las predicciones del modelo y las etiquetas reales. Este valor calculado se usará posteriormente para ajustar el modelo.

La línea 6 permite definir el algoritmo de optimización que se utilizará para actualizar los parámetros w y b de las capas lineales, así como los parámetros α, γ y m de una GCLV. Tales parámetros permiten modelar la arquitectura de la red para llevar a cabo el proceso de aprendizaje automático.

En la línea 7 se define el número de épocas que se utilizarán para resolver cada problema. Este valor es propuesto por el usuario y puede ajustarse según las necesidades específicas del problema en cuestión. Por lo general, para conjuntos de datos pequeños, mil épocas suelen ser suficientes. Sin embargo, para problemas más complejos o conjuntos de datos más grandes, puede ser necesario aumentar la cantidad de épocas para obtener resultados óptimos. Es importante ajustar este valor según la complejidad y el tamaño del conjunto de datos para garantizar un entrenamiento adecuado del modelo.

La línea 8 corresponde a la etapa de propagación hacia adelante (forward propagation) correspondiente a una red neuronal tipo propagación hacia adelante en el proceso de entrenamiento de la red neuronal. En esta línea, se pasa el lote de datos de entrenamiento *inputs* a través de la red neuronal network. Cada vez que se realiza esta operación, la red neuronal produce predicciones para el lote de datos de entrada de tamaño igual a *inputs*, las cuales se almacenan en la variable *outputs*.

En la línea 9, se calcula el error de las predicciones de la red comparando las predicciones almacenadas en *outputs* con los valores reales de las etiquetas de los datos de entrenamiento en *labels*, utilizando la función de pérdida definida anteriormente. Este cálculo de error es necesario para evaluar la eficiencia del proceso de aprendizaje aplicado a la red neuronal y es utilizado para ajustar los parámetros del modelo durante la etapa de retropropagación.

En la línea 10, se utiliza el método zero_grad() proporcionado por la biblioteca PyTorch para restablecer todos los gradientes de los parámetros del modelo a cero. Esta operación es

necesaria antes de realizar la retropropagación del error y actualizar los pesos de la red neuronal durante el proceso de entrenamiento.

En la línea 11, se llama al método backward() perteneciente a la biblioteca Pytorch, realiza el proceso de retropropagación del error a través de la red neuronal. Durante la retropropagación, se calculan los gradientes de la función de pérdida con respecto a los parámetros del modelo, lo que permite ajustar dichos parámetros para minimizar la pérdida en futuras iteraciones del proceso de entrenamiento.

En la línea 12, se llama al método step() del objeto *optimizer*. Este método perteneciente a la biblioteca Pytorch, actualiza los parámetros del modelo utilizando los gradientes calculados durante la retropropagación del error. Es decir, actualiza los pesos de la red neuronal de acuerdo con la tasa de aprendizaje y los gradientes de la función de pérdida para mejorar el rendimiento del modelo en la siguiente iteración del proceso de entrenamiento.

El segmento que corresponde a las líneas 8 al 12 del Algoritmo 4.5, representan el núcleo del proceso de aprendizaje automático de la arquitectura propuesta. En estas líneas se calcula el error entre las predicciones de la red y las etiquetas reales de los datos de entrenamiento, luego se realiza la retropropagación de este error para ajustar los parámetros de la red neuronal y mejorar su desempeño en la predicción. Este proceso se repite durante un número definido de épocas para permitir que el modelo aprenda de manera incremental y progresiva.

En la línea 13, se utiliza la función imprimir_resultados, la cual forma parte de la biblioteca personalizada resultados, que se presenta en el Anexo A. Esta función recibe como argumentos la predicción de la red neuronal sobre el conjunto de datos de prueba *inputs*, las etiquetas reales correspondientes a los datos *labels*, los valores *Máximo* y *Mínimo* que se utilizan para transformar los valores difusos a su magnitud real, lo que permite compararlos con la magnitud original del campo de clase.

En la línea 14 se obtienen los parámetros descubiertos en el proceso de optimización de la red neuronal almacenándose en la variable *params*. Lo cual se logra utilizando el método state_dict() perteneciente a la biblioteca Pytorch.

En la línea final 15, se utiliza la función imprimir_parametros, que forma parte de la biblioteca personalizada resultados. Esta función recibe un único argumento la información contenida en la variable *params*. Su objetivo es imprimir los parámetros descubiertos por el modelo en un formato predefinido, lo que facilita su análisis y evaluación.

El algoritmo 4.6 define en la clase Net la arquitectura de la red neuronal. Este algoritmo se compone de dos funciones principales las cuales corresponden a la arquitectura mostrada en la Figura 4.10, particularmente a las capas intermedias y la capa final.

Algoritmo 4.6 Clase Net //arquitectura de la red neuronal
Función Init (self) // Inicialización de la clase

1. Definir el modelo lineal que pasa de la capa L a la capa L+1
   Fc1← nn.Linear(*Variables_entrada*, N*odos_salida*)
2. Declarar tantas capas lineales, como capas existan. En la capa final, el número n de nodos se reduce a un único nodo de salida
   Fcn ← nn.Linear(*nodos_entrada*, 1)
3. Generar una GCLV para cada nodo de salida del modelo lineal. Se declaran tantas capas de funciones, como capas lineales existan, donde estas funciones son iguales al número de nodos de salida
   Activation1← ActivationsFunctions.GCLV(*Nodos*, *Base_logaritmica*, *Exponente*)
4. Se declaran tantas capas de funciones, como capas lineales existan, donde estas funciones son iguales al número de nodos de salida de cada nn.Linear()
5. Generar un nodo de salida en la última capa
   Activation← ActivationsFunctions.GCLV(1, *Base_logaritmica*, *Exponente*)

Función forward(*train_set*) // Genera la estructura de la red neuronal

1. Generar la estructura de la red neuronal capa por capa
   train_set ← Fc1(*train_set*)
   train_set ← activation1(*train_set*)
2. Definir las funciones lineales y de activación que actúan sobre cada capa
   train_set ← Fcn(*train_set*)
   train_set ← activationn(*train_set*)
3. Devolver *train_set*

Función init: Este método inicializa la red neuronal definiendo las capas lineales que conforman la arquitectura propuesta. Cada capa lineal se crea utilizando la función nn.Linear(*neuronas_entrada*, *neuronas_salida*) de PyTorch, donde se especifica el número de neuronas de entrada y de salida de cada capa. En este caso, las capas lineales se denominan fc1 hasta fcn, donde fc1 recibe como entrada la cantidad de atributos del problema y la capa fcn tiene un solo nodo de salida que proporciona las predicciones de la red. Se pueden definir tantas capas y nodos como sean necesarios para resolver el problema específico. Después de declarar las capas lineales, se define la función de activación para las neuronas de cada capa como una variable lingüística continua generalizada (GCLV). Esto se logra utilizando la función ActivationsFunctions.GCLV(nodos, base, exponente) de la biblioteca propia ActivationsFunction, la cual se describe en detalle en el Anexo A. Esta función toma el número de neuronas de salida definidas en cada capa *fc* para generar cada GCLV. Además, se deben especificar los valores de la base logarítmica y del exponente aplicado a la lógica de ACFL-ALF.

Función forward: Este método define cómo se propagan los datos a través de la red. Aquí se especifica la secuencia de operaciones que se aplican a los datos de entrada del *train_set*. Primero, los datos pasan a través de cada capa lineal (self.*fc1*, self.f*c2*, ..., self.f*cn*) y luego se aplican las funciones de activación correspondientes (self.*activation1*, self.*activation2,* …,

self.*activationn*). Finalmente, el resultado de la última capa lineal se devuelve como salida de la red. El número de capas y nodos por capa se propone por el usuario y este debe ajustarse a las necesidades específicas del problema. Sin embargo, basado en observaciones realizadas, se ha notado que una sola capa intermedia con un número de nodos que aumenta de acuerdo con la complejidad del problema ofrece una buena capacidad de adaptación y entrenamiento del modelo. Esto significa que, para problemas simples, es posible implementar un menor número de nodos, mientras que, en problemas más complejos, se hace necesario añadir más nodos en la capa intermedia. Aun así, es necesario experimentar con diferentes configuraciones de capas y nodos de capa para encontrar la combinación óptima que mejor se adapte al problema en cuestión.

4.3.2 Experimentación con una NN-GCLV

Una vez explicada la arquitectura de la red neuronal propuesta y el Algoritmo 4.5 que permite su implementación, se procede a realizar una serie de experimentos con el fin de evaluar aspectos clave de la configuración de la red neuronal. Posteriormente, se lleva a cabo un experimento de evaluación utilizando diversos conjuntos de datos, lo que permite poner a prueba la efectividad y robustez del modelo en diferentes escenarios y condiciones.

Los experimentos se realizaron utilizando una MacBook con las siguientes especificaciones:

- Procesador: Apple M1.
- Memoria RAM: 8 GB.
- Sistema Operativo: macOS Sonoma v14.4.1.

Experimento 1: Influencia de la base logarítmica *b* y exponencial *n* de una ACFL-ELF en el proceso de entrenamiento de una NN-GCLV.

En este primer experimento se observa como los valores de la base logarítmica y el valor exponencial impactan en el proceso de entrenamiento e inferencia de una NN-GCLV. Con este fin, se hace uso de los conjuntos de datos listados en la Tabla 4.10. los detalles pueden observarse en el Anexo B.

Tabla 4.10: Descripción de los sets de datos usados en las pruebas.

Conjunto de datos	Atributos	Registros
Bupa	6	345
Iris	5	150
Pima	9	768
Tinto	13	1599

La NN-GCLV hace uso de la función de error Mean Square Error y como algoritmo de optimización gradiente descendiente estocástico mencionados en la sección 2.2.1. La configuración de la NN-GCLV para cada conjunto de datos se observa en la Tabla 4.11:

Tabla 4.11: Configuración de una NN-GCLV.

Conjunto de datos	Capas	Nodos	Épocas
Bupa	3	$L_0 = 5$, $L_1 = 10$, $L_2 = 1$	1500
Iris	3	$L_0 = 4$, $L_1 = 5$, $L_2 = 1$	1000
Pima	3	$L_0 = 8$, $L_1 = 10$, $L_2 = 1$	2500
Tinto	3	$L_0 = 4$, $L_1 = 8$, $L_2 = 1$	2500

De acuerdo con las descripciones proporcionadas, se observa que para cada capa inicial o de índice cero, el número de nodos es igual al número de atributos del conjunto de datos menos uno. Debido a que el último atributo corresponde al campo de la clase objetivo y no se utiliza como entrada para la red neuronal.

Posteriormente, se define una única capa intermedia para cada prueba, con un número variable de nodos. Esta configuración se basa en el tamaño de los atributos del conjunto de datos y el número de registros a evaluar. Por lo tanto, el número de nodos en esta capa intermedia se ajusta de acuerdo con la magnitud del problema.

Para la capa de salida, se utiliza una única neurona que realiza las predicciones correspondientes al campo de la clase objetivo.

Por último, el número de épocas varía de acuerdo con la extensión del conjunto de datos y la complejidad del problema. Esta variación permite adaptar el proceso de entrenamiento a la cantidad de datos disponibles y a la dificultad de la tarea de predicción.

En las Tablas 4.12 y Tabla 4.13 se presentan los resultados de los experimentos, donde se variaron los valores de la base logarítmica (b) y el valor exponencial (n), analizando su impacto en el modelo de aprendizaje automático. Así también la Figura 4.14, permite observar la evolución del proceso de aprendizaje para el conjunto BUPA.

Tabla 4.12: Importancia de la base logarítmica en el proceso de entrenamiento.

Conjunto de datos	$b = e$ Y $n = 1$		$b = e$ Y $n = 1$		$b = e$ Y $n = 1$	
	MSE	**% Aciertos**	**MSE**	**% Aciertos**	**MSE**	**% Aciertos**
Bupa	8.5049	0.20289	8.1666	0.20289	7.8162	0.24347
Iris	0.0467	1	0.0224	1	**0.0152**	**1**
Pima	0.1537	0.64843	0.1408	0.70703	0.1346	0.71614
Tinto	0.4398	0.89118	0.4289	0.89368	0.4208	0.89493

Tabla 4.13: Importancia del exponente en el proceso de entrenamiento.

Conjunto de datos	$b = 25$ Y $n = 1$		$b = 25$ Y $n = 3$		$b = 25$ Y $n = 5$	
	MSE	**% Aciertos**	**MSE**	**% Aciertos**	**MSE**	**% Aciertos**
Bupa	7.8162	0.24347	45.258	0.03478	104.94	0.01159
Iris	**0.0152**	**1**	0.3516	0.98666	0.6778	0.66666
Pima	0.1346	0.71614	0.2856	0.41067	0.3967	0.34635
Tinto	0.4208	0.89493	1.0718	0.68918	3.0997	0.27954

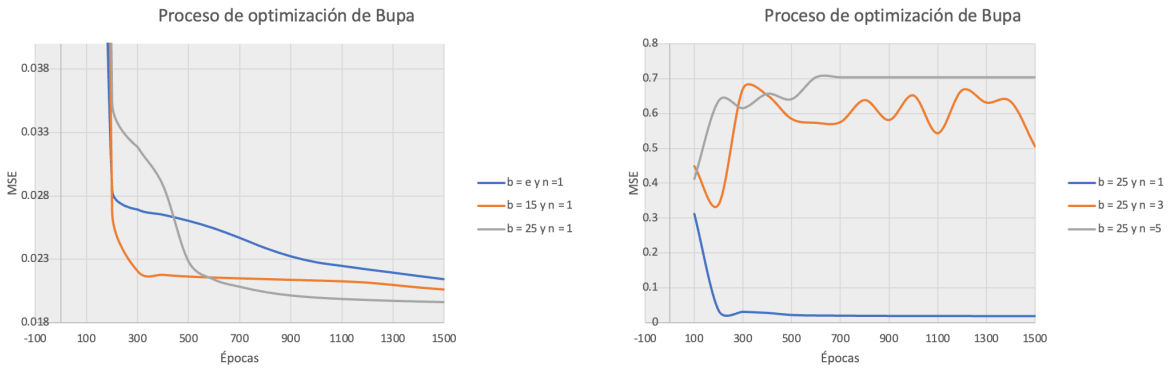


Figura 4.14: Representación gráfica del cálculo del error para Bupa.

Entonces, a partir de este experimento se evalúa la influencia de la base logarítmica (b) y el exponente (n) en el proceso de entrenamiento de una red neuronal. Observando los resultados de las pruebas utilizando diferentes valores de b y n en los conjuntos de datos listados en la Tabla 4.10 se concluye lo siguiente:

Se observó que el valor de b tiene un impacto significativo en la efectividad de la clasificación. En general, se encontró que a medida que el valor de b aumenta, aumenta el porcentaje de aciertos en la clasificación. Por ejemplo, en el conjunto de datos Bupa, se obtuvieron porcentajes de aciertos más altos con valores de b mayores.

Por otro lado, en cuanto al exponente n, se encontró que su valor influye en gran manera al proceso de entrenamiento. Se observó que a medida que el valor de n aumenta, la efectividad

de la inferencia disminuye, en algunos casos, el algoritmo no logra converger, dejando el proceso inconcluso.

Los resultados observados permiten determinar que la selección adecuada de los valores de *b* y *n* es esencial en el rendimiento óptimo del modelo de aprendizaje automático.

Experimento 2: Impacto de la función de costo y del algoritmo de optimización basado en el gradiente en el proceso de aprendizaje automático de una NN-GCLV.

El objetivo de este experimento es evaluar el impacto de las funciones de costo y los métodos de optimización basados en gradiente en el proceso de aprendizaje de una NN-GCLV. Se espera determinar cuál de los métodos implementados se adapta mejor al modelo propuesto, lo que permitirá tomar mejores decisiones con respecto a la configuración final de la NN-GCLV.

Para este análisis, se consideran las funciones de costo y métodos de optimización listados en la Tabla 4.14, los cuales han sido detallados en la sección 2.2.1. El análisis comparativo se centra en determinar cómo estas funciones afectan la convergencia del modelo.

Se espera poder identificar el método de optimización y función de costo que maximice el rendimiento de la NN-GCLV en la solución de diferentes problemas. Esperando que esta experimentación proporcione una comprensión profunda en el proceso de selección de las funciones de costo y los métodos de optimización que permitan mejorar el desempeño del modelo en tareas específicas de aprendizaje automático.

Tabla 4.14: Funciones de costo y optimización basados en el gradiente usados en el proceso de evaluación del entrenamiento en una NN-GCLV.

	Función de optimización	Función de costo
1	SGD	MSE
2	ADAM	MAE
3	AdaGrad	SMAE
4	RMSprop	
5	AdaDelta	

En el siguiente experimento, se mantienen fijos los valores de la base logarítmica y exponencial en 25 y 1, respectivamente, basándose en los resultados del experimento anterior que indica su eficiencia. La arquitectura de la NN-GCLV se mantiene igual que la descrita en la Tabla 4.11. Este experimento permite comparar cómo los diferentes métodos de optimización y funciones de costo afectan el rendimiento del modelo. Los resultados se muestran de la Tabla 4.15 a la Tabla 4.18. Así también en la Figura 4.15 se observa de manera gráfica los resultados de los experimentos para el conjunto de datos Bupa.

Tabla 4.15: Resultado del entrenamiento de la NN-GCLV con Bupa.

Bupa	MSE	% Aciertos	MAE	% Aciertos	SMAE	% Aciertos
SGD	10.333	0.17101	3.087	0.16811	1.8377	0.19130
ADAM	8.0554	0.19130	2.1014	0.33333	1.7379	**0.25797**
ADAGRAD	7.6685	0.21739	2.1504	**0.35362**	1.7703	0.20869
RMSPROP	8.5069	0.18260	2.2153	0.29565	1.8804	0.21449
ADADELTA	9.9846	**0.22028**	2.2509	0.25797	1.9093	0.23188

Tabla 4.16: Resultado del entrenamiento de la NN-GCLV con Iris.

Iris	MSE	% Aciertos	MAE	% Aciertos	SMAE	% Aciertos
SGD	0.032	1	0.383	1	0.011	1
ADAM	0.025	1	0.132	1	0.121	1
ADAGRAD	0.027	1	0.131	1	0.018	1
RMSPROP	0.034	1	0.125	0.99	0.029	1
ADADELTA	0.023	1	0.163	1	0.009	1

Tabla 4.17: Resultado del entrenamiento de la NN-GCLV con Pima.

Pima	MSE	% Aciertos	MAE	% Aciertos	SMAE	% Aciertos
SGD	0.163	0.66145	0.336	0.43576	0.076	0.69921
ADAM	0.139	**0.78833**	0.2380	**0.76171**	0.069	**0.70833**
ADAGRAD	0.144	0.68619	0.288	0.69270	0.073	0.68619
RMSPROP	0.143	0.69270	0.260	0.74348	0.072	0.69010
ADADELTA	0.144	0.73046	0.352	0.65104	0.077	0.68098

Tabla 4.18: Resultado del entrenamiento de la NN-GCLV con Tinto.

Tinto	MSE	% Aciertos	MAE	% Aciertos	SMAE	% Aciertos
SGD	0.456	0.84989	1.081	0.52345	0.216	0.86741
ADAM	0.428	**0.89429**	0.812	0.81238	0.194	**0.87616**
ADAGRAD	0.477	0.85115	0.814	0.81488	0.199	0.84427
RMSPROP	0.461	0.85490	0.815	**0.81550**	0.217	0.85803
ADADELTA	0.483	0.79174	0.398	0.39899	0.224	0.77923

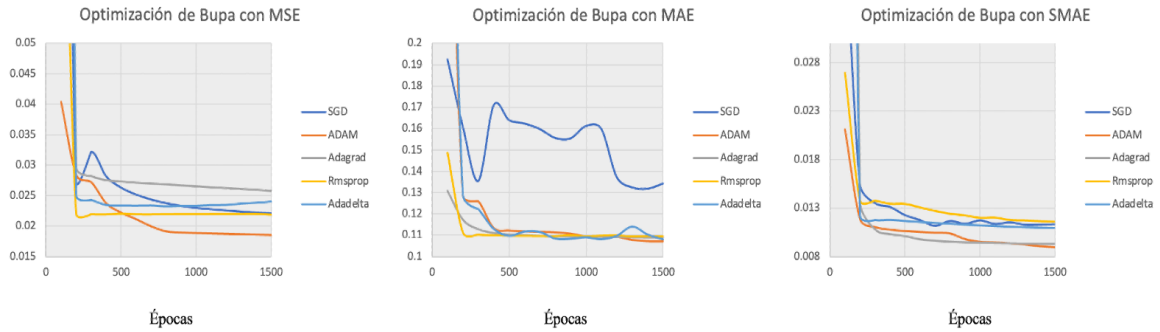


Figura 4.15: Representación gráfica del cálculo del error para Bupa.

Los resultados de las pruebas realizadas con diferentes funciones de costo y métodos de optimización para la red neuronal NN-GCLV permiten visualizar cómo cada combinación de funciones afecta el proceso de entrenamiento del modelo para diferentes conjuntos de datos. Esta información, permite observar que no existe una única solución óptima aplicable a todos los casos, en cambio, cada conjunto de datos parece tener una combinación adecuada.

Tomando como ejemplo el conjunto de datos Bupa, se observa que el método de optimización AdaGrad obtiene el MSE más bajo y el mayor porcentaje de aciertos al compararlo con los demás métodos. Sin embargo, para Pima, el método ADAM muestra un rendimiento superior tanto en MSE como en porcentaje de aciertos. Por otro lado, en Tinto, ADAM y RMSprop muestran un rendimiento similar, con MSE bajo y altos porcentajes de aciertos.

Los resultados obtenidos hacen notar la importancia de adaptar la elección de la función de costo y el método de optimización de acuerdo con las características específicas de cada conjunto de datos dado. Por lo tanto, es importante considerar aquellos factores que forman parte del conjunto de datos como son la complejidad del problema, la distribución de los datos y la cantidad de registros disponibles para realizar la configuración adecuada de la red neuronal.

Experimento 3: Influencia de los parámetros α, γ y m de una GCLV perteneciente a una ACFL-ELF en el proceso de aprendizaje automatico.

En este experimento, se examina el impacto individual de los parámetros α, γ y m de una GCLV que actúa como una función de activación en el proceso de aprendizaje de una NN-GCLV. El principal objetivo es entender cómo los parámetros inciden en la inferencia de datos y cómo afectan el rendimiento del modelo.

Los experimentos se llevan a cabo utilizando los mismos conjuntos de datos mencionados en experimentos anteriores, y la arquitectura de la red neuronal es especificada en la Tabla 4.19. Los valores de b y n se mantienen fijos en 25 y 1 respectivamente, mientras que la función de costo se establece como MSE y el algoritmo de optimización como ADAM.

Para evaluar el proceso de entrenamiento de la NN-GCLV para cada conjunto de datos evaluado, se llevan a cabo una serie de 30 experimentos. Donde los resultados presentados se basan en el promedio de esta serie de experimentos, excepto en los casos en el que se presentan los resultados utilizan gráficos, los cuales muestran un único proceso de entrenamiento para ejemplificar de manera sencilla y concisa el proceso de optimización mediante el cálculo de MSE en cada época.

Tabla 4.19: Configuración de una NN-GCLV

Conjunto de datos	Capas	Nodos	Épocas
Bupa	3	$L_0 = 5,$ $L_1 = 10,$ $L_2 = 1$	5000
Iris	3	$L_0 = 4,$ $L_1 = 5,$ $L_2 = 1$	1000
Pima	3	$L_0 = 8,$ $L_1 = 10,$ $L_2 = 1$	3000
Tinto	3	$L_0 = 4,$ $L_1 = 8,$ $L_2 = 1$	4000

Si bien, no existe una combinación de valores única que sea óptima para todos los problemas, se observa que esta combinación proporciona resultados satisfactorios en la mayoría de los casos. Y debido a que explorar variantes de los parámetros y funciones puede ser complicado y no necesariamente ofrecer buenos resultados.

Experimento con el conjunto de datos Iris: Se descubren los parámetros α, γ y m de una GCLV, la cual se usa como función de activación (*ec.* 2.1) en una red neuronal de propagación hacia adelante.

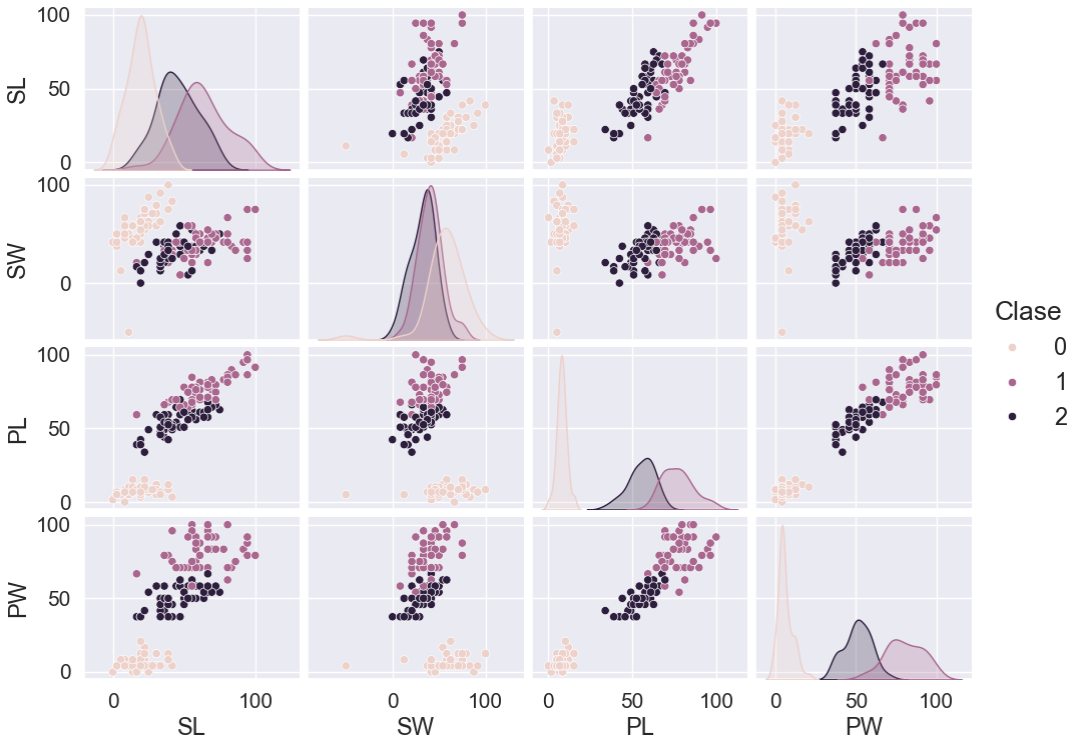


Figura 4.16: Diagrama de correlación entre los datos de Iris.

En la Figura 4.16, se observa la relación existente entre cada par de atributos con respecto a la clase a la que pertenecen, donde a la especie iris setosa se asigna el valor de 0, iris virginica corresponde al valor de 1 e iris versicolor corresponde al valor de 2. Al observar este diagrama se puede observar que la clase "iris setosa" es fácil de diferenciar en comparación con las otras dos clases.

Su distribución muestra una separación clara respecto a las otras dos clases, lo que facilita su distinción. Por otro lado, las clases "iris virginica" e "iris versicolor" presentan una mayor superposición y similitud en su distribución, lo que las hace difíciles de distinguir entre sí.

En la Figura 4.17 se observan los resultados del entrenamiento a través de la optimización de los parámetros de una GCLV, lo cual indica la evolución del MSE a lo largo del proceso de optimización.

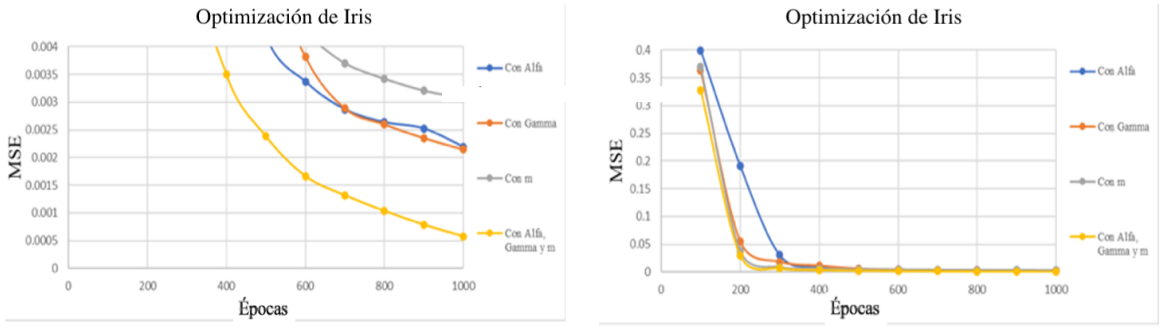


Figura 4.17: Gráficas del proceso de entrenamiento del conjunto de datos iris en relación con los parámetros α, γ y m.

De la misma manera, la Tabla 4.20 muestra los resultados promedio obtenidos para cada experimento en iris realizado con la NN-GCLV.

Tabla 4.20: Resultados del entrenamiento de iris con una NN-GCLV

Métricas	$\boldsymbol{\alpha}$	$\boldsymbol{\gamma}$	$\boldsymbol{m}$	$\boldsymbol{\alpha, \gamma}$ Y $\boldsymbol{m}$
MSE	0.00824	0.008	0.0116	**0.0018**
% Aciertos	0.992	0.9906	0.9853	**1**
% Error	0.008	0.00933	**0.01466**	0

A través los resultados, se puede decir que para el conjunto de datos iris, se observa que el peor desempeño se registra al optimizar el parámetro m. Por otro lado, los mejores resultados se obtienen al optimizar los parámetros α, γ y m ". La optimización a través de los parámetros observados proporciona la menor distancia cuadrada entre los valores reales y predichos, así como el mayor porcentaje de aciertos en la clasificación de datos. No obstante, otros procesos también muestran una precisión cercana a los resultados observados.

Experimento con el conjunto de datos Bupa: Se descubren los parámetros α, γ y m de una GCLV, la cual se usa como función de activación (*ec.* 2.1) en una red neuronal de propagación hacia adelante.

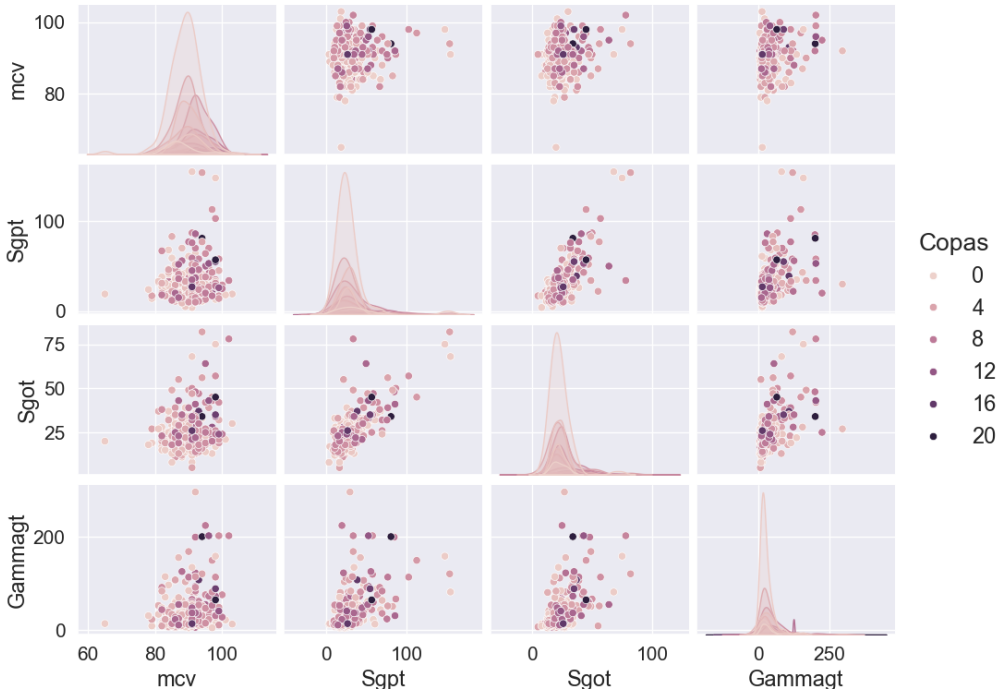


Figura 4.18: Diagrama de correlación entre los datos de Bupa.

En la Figura 4.18 se observa la relación entre los atributos de bupa y su relación en función del número de bebidas alcohólicas consumidas por día. En la gráfica presentada, se evidencia que es difícil determinar qué atributo refleja una diferenciación clara entre las clases, ya que todos ellos muestran una fuerte relación entre sí. No se aprecia una separación clara o un patrón distintivo que permita una fácil discriminación de las clases.

En el gráfico presentado en la Figura 4.19 se aprecia el proceso de optimización del conjunto de datos bupa con respecto al cálculo del MSE a lo largo de las épocas evaluadas.

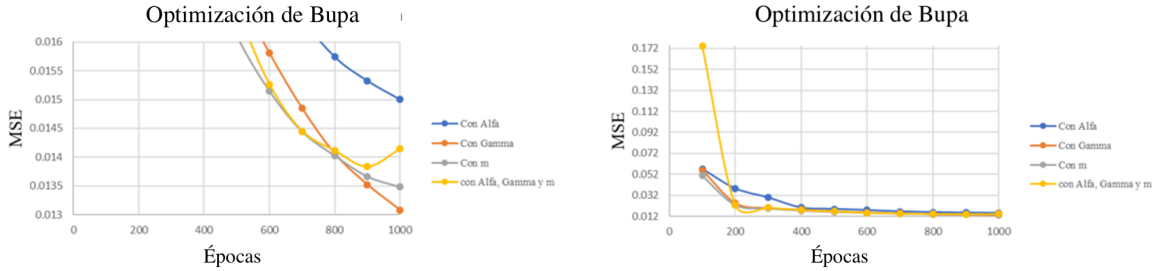


Figura 4.19: Gráficas del proceso de entrenamiento de Bupa en relación con los parámetros α, γ y m.

De la misma manera, la Tabla 4.21 muestra los resultados promedio obtenidos para cada experimento en Bupa realizado con la NN-GCLV.

Tabla 4.21: Resultados del entrenamiento de Bupa con una NN-GCLV.

Métricas	$\boldsymbol{\alpha}$	$\boldsymbol{\gamma}$	$\boldsymbol{m}$	$\boldsymbol{\alpha, \gamma}$ Y $\boldsymbol{m}$
MSE	5.8778	**5.0959**	5.3023	5.3602
% Aciertos	0.1605	0.1768	0.1692	**0.1785**
% Error	**0.8394**	0.8231	0.8307	0.8214

Para Bupa, se destaca que el mejor desempeño en términos de error durante las iteraciones se logra al optimizar el parámetro γ, mientras que el peor desempeño se observa en la optimización del parámetro α.

En cuanto a la distancia entre el valor real y el predicho, se obtiene el mejor resultado con la optimización de γ. Sin embargo, el mayor porcentaje de aciertos se alcanza con la optimización de los parámetros α, γ y m, aunque la diferencia final no es significativa.

Experimento con el conjunto de datos Pima: Se descubren los parámetros α, γ y m de una GCLV, que es usada como función de activación (*ec.* 2.1) en una red neuronal de propagación hacia adelante.

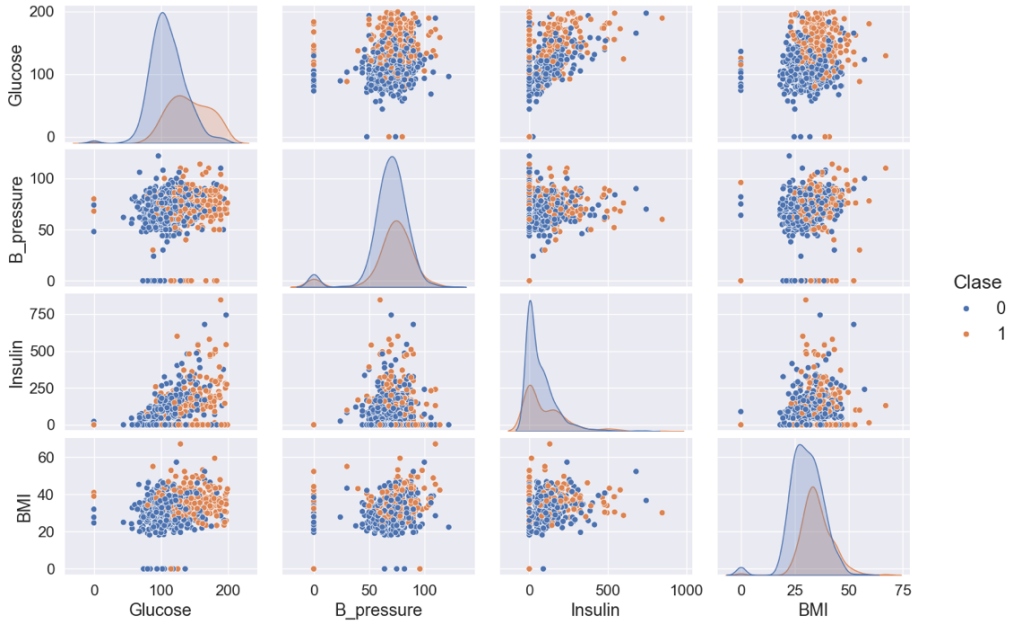


Figura 4.20: Diagrama de correlación entre los datos de Pima.

En la Figura 4.20 se muestra que la relación entre los atributos es compleja y no permite una diferenciación clara entre las clases. Lo cual indica que la tarea de clasificación de los

registros representados en el gráfico es difícil. La superposición y la falta de separación clara entre los conjuntos de datos dificultan la tarea de clasificación.

En la Figura 4.21 se presentan los resultados del proceso de optimización del conjunto de datos Pima en relación con los parámetros de una GCLV, es decir, se observa cómo varía el MSE calculado en cada época del proceso de entrenamiento. Este gráfico proporciona una visualización clara de cómo los diferentes valores de los parámetros de la GCLV afectan la convergencia del modelo durante el entrenamiento. Mediante esta representación, es posible identificar qué combinación de parámetros conduce a una optimización efectiva del modelo en términos de reducción del error

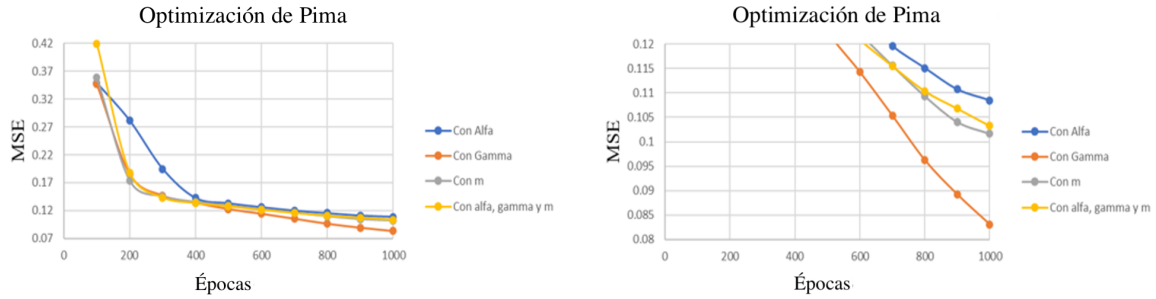


Figura 4.21: Gráficas del proceso de entrenamiento de Pima en relación con los parámetros α, γ y m.

La Tabla 4.22 presenta los resultados promedio obtenidos para cada experimento realizado con la NN-GCLV en el conjunto de datos Pima. En esta Tabla se resumen las métricas de rendimiento, como el MSE y el porcentaje de aciertos, los cuales son calculados a partir de múltiples ejecuciones del experimento. Los resultados obtenidos proporcionan una comprensión general del desempeño del modelo en términos de precisión.

Tabla 4.22: Resultados del entrenamiento de Pima con una NN-GCLV.

Métricas	$\boldsymbol{\alpha}$	$\boldsymbol{\gamma}$	$\boldsymbol{m}$	$\boldsymbol{\alpha, \gamma}$ Y $\boldsymbol{m}$
MSE	0.056	**0.0422**	0.0661	0.0602
% Aciertos	0.8492	**0.9583**	0.8593	0.9309
% Error	**0.1507**	0.0416	0.1406	0.0690

Según lo observado, el conjunto de datos Pima muestra un comportamiento similar al de la Bupa, donde la mayor pérdida se produce al optimizar el parámetro γ, y el peor resultado se obtiene al ajustar el parámetro α.

En resumen, los mejores resultados en términos de distancia media cuadrada y porcentaje de aciertos se alcanzan al optimizar el parámetro γ. Por otro lado, el proceso de optimización más rápido corresponde al ajuste del parámetro α.

Experimento con el conjunto de datos Tinto: Se descubren los parámetros α, γ y m de una GCLV, que es usada como función de activación (*ec.* 2.1) en una red neuronal de propagación hacia adelante.

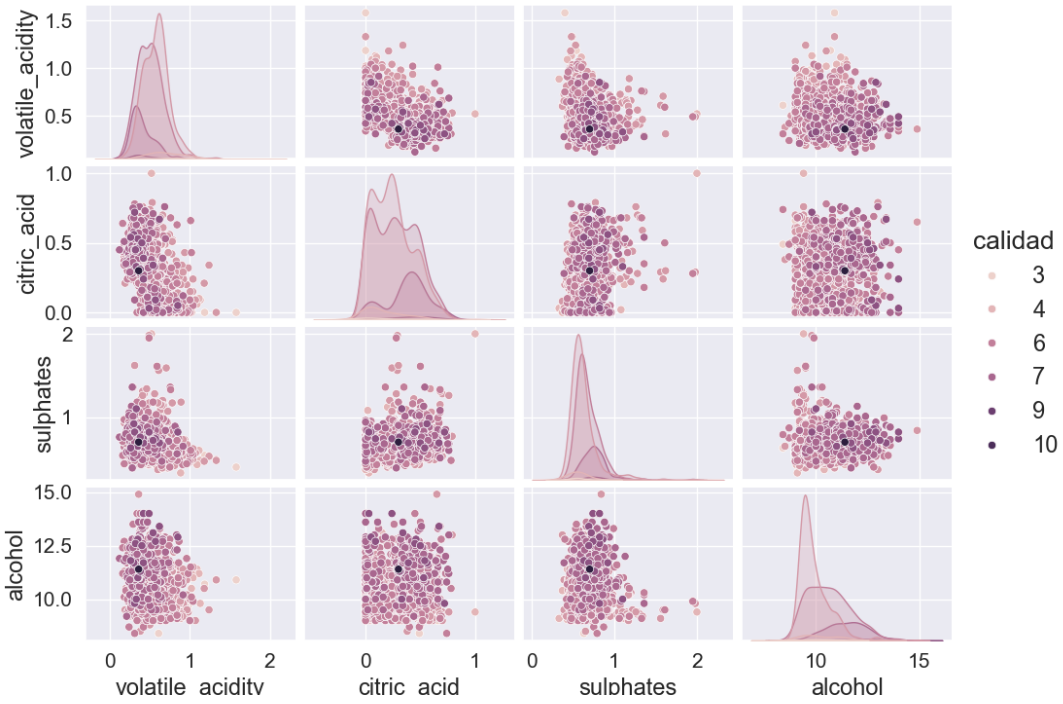


Figura 4.22: Diagrama de correlación entre los datos de Tinto.

La relación entre los datos muestra una baja diferenciación entre las clases, lo que dificulta la identificación de una clara división entre una clase y otra. La superposición y la falta de separación evidente entre los grupos de datos hacen que sea complicado establecer límites definidos entre las clases.

En la Figura 4.23 se muestra el comportamiento de los experimentos de aprendizaje aplicados a los parámetros α, γ y m de forma individual y optimizándolos simultáneamente. Los experimentos planteados, se basan en la minimización del MSE y permiten observar cuál de los parámetros proporciona a la función de activación una mayor capacidad de adaptación y el mejor ajuste al modelo evaluado.

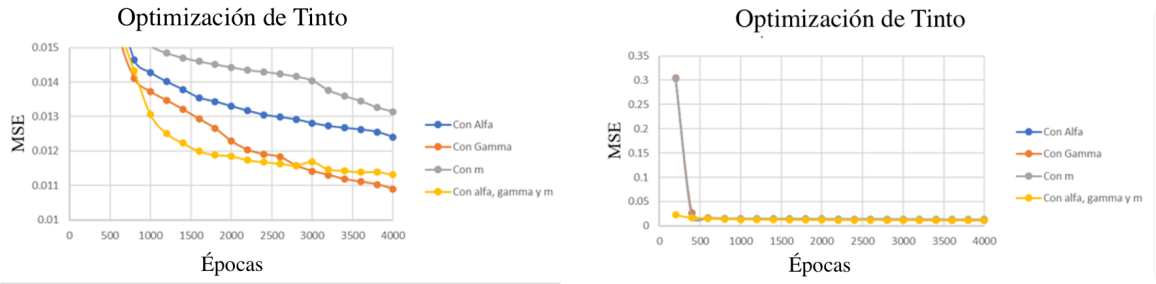


Figura 4.23: Gráficas del proceso de entrenamiento de Tinto en relación con los parámetros α, γ y m.

La Tabla 4.23 muestra los resultados promedio obtenidos para cada aprendizaje realizado con una NN-GCLV para el conjunto de datos tinto. Esta Tabla ofrece métricas de rendimiento como el MSE, el porcentaje de aciertos, entre otros, calculados a partir de múltiples ejecuciones del experimento. Los resultados obtenidos, permiten observar cómo influyen los parámetros en el desempeño del modelo en términos de precisión y adaptabilidad

Tabla 4.23: Resultados del entrenamiento del conjunto de datos Tinto con una NN-GCLV.

Métricas	$\boldsymbol{\alpha}$	$\boldsymbol{\gamma}$	$\boldsymbol{m}$	$\boldsymbol{\alpha, \gamma}$ Y $\boldsymbol{m}$
MSE	0.3083	**0.2685**	0.3261	0.2835
% Aciertos	0.5803	**0.6447**	0.5434	0.6197
% Error	0.4196	0.3552	**0.4565**	0.3802

De acuerdo con los resultados gráficos observados, se puede concluir que la optimización del parámetro γ logra el mejor desempeño basado en la función de pérdida.

En la Tabla 4.23, se presenta que el mejor rendimiento en la diferencia entre los valores predichos y reales, así como el mayor porcentaje de aciertos, se alcanzan con la optimización del parámetro γ.

Una vez realizados los experimentos se comparan con resultados encontrados en artículos que realizan un proceso de aprendizaje automático mediante el uso de diferentes modelos, los resultados se muestran en la Tabla 4.24.

Tabla 4.24: Tabla comparativa de los resultados de una NN-GCLV y la Literatura.

Conjunto de datos	**NN-GCLV**	**Literatura**	
Iris	**1**	97	(Yang & Zhao, 2023)
BUPA	0.3072	**0.5855**	(Gao, 2023)
Pima	**0.9583**	0.7454	(Yang & Zhao, 2023)
Tinto	**0.6447**	0.5331	(Gao, 2023)

Al comparar los resultados con los encontrados en (Gao, 2023; Yang & Zhao, 2023), se observa que la NN-GCLV logra resultados competitivos. Sin embargo, también es importante notar que aún hay margen para mejorar los resultados obtenidos. En otras palabras, no se ha llegado a una conclusión definitiva sobre qué configuración de parámetros produce de manera consistente los mejores resultados. Sin embargo, se han obtenido resultados prometedores y competitivos en comparación con otros estudios de vanguardia

4.3.3 Red neuronal de lógica difusa Arquimediana compensatoria (NN-ACFL)

En esta sección se establece la arquitectura principal de la presente tesis, que consiste en una red neuronal de tipo propagación hacia adelante cuyos conceptos clásicos se reemplazan por los de una ACFL, específicamente una ACFL-ELF. Esta lógica se detalla en la sección 4.1 de la investigación.

El resultado de esta combinación es una red neuronal artificial híbrida de lógica difusa Arquimediana compensatoria y la arquitectura de una red tipo propagación hacia adelante. Lo cual significa que cada arquitectura construida puede ser evaluada mediante una lógica difusa Arquimediana y una lógica compensatoria. Lo cual permite introducir los conceptos de la necesidad y la posibilidad ampliamente relacionados con la lógica modal. Además, este modelo evita el uso de recursos extra lógicos, por lo cual es interpretable a partir de los conceptos de la ACFL.

Como se ha demostrado anteriormente, una ACFL es una lógica interpretable, que como se muestra en la sección 4.2, es expresable en lenguaje natural a partir de sus interpretaciones algebraicas y la definición de las preimágenes de 1, 0.5 y 1-δ según la definición de su perfil.

Dentro de las ventajas que presenta este modelo se tienen las siguientes:

I. Cada atributo de entrada se modela a través de una GCLV(x, α, γ, m), la cual, como se ha demostrado, es una función que se adapta a la información. Debido a su capacidad para adaptar su morfología, permite múltiples interpretaciones de los datos, así como la posibilidad de expresar sus resultados en lenguaje natural.
II. El uso de conceptos de una lógica difusa Arquimediana compensatoria convierte la arquitectura en un predicado interpretable y expresable en lenguaje natural. Este enfoque es altamente atractivo en el presente modelo, ya que permite justificar y entender los resultados presentados. Esta ventaja es poco común en las redes neuronales tradicionales.
III. El uso de operadores conjuntivos y disyuntivos tanto Arquimedianos como compensatorios, permiten generar predicados en Forma Normal Conjuntiva (FNC) y Forma Normal Disyuntiva (FND) los cuales, de acuerdo con la lógica proposicional, cualquier predicado construido tiene su correspondiente equivalente en FNC y FND.

IV. Los resultados obtenidos, son competitivos al compararlos en función de la efectividad, robustez con los resultados reportados en el estado del arte. Añadiendo las capacidades inherentes a un modelo lógico

En la Figura 4.24 se presenta la arquitectura de una red neuronal híbrida de tipo propagación hacia adelante y ACFL, denominada NN-ACFL.

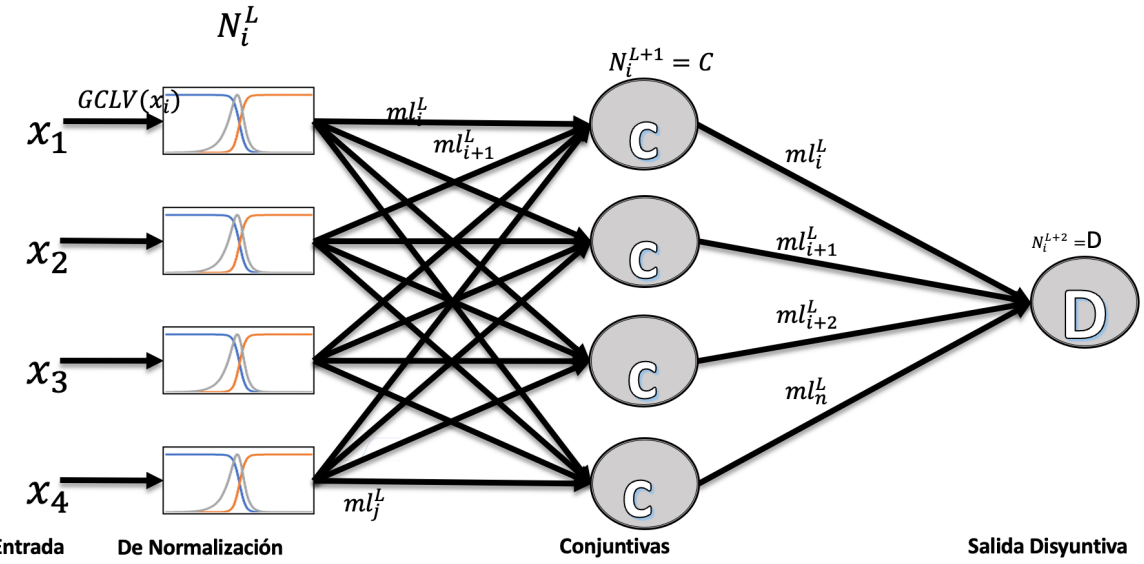


Figura 4.24: Arquitectura de una red híbrida de propagación hacia adelante y ACFL denominada NN-ACFL.

En la arquitectura propuesta, se emplean al menos dos capas intermedias además de la inicial y de salida. En la primera de estas capas, los datos de entrada transformados al intervalo [0,1] son modelados a través de una GCLV, la cual se describe detalladamente en la sección anterior mostrándose en la Figura 4.11. Esta GCLV a diferencia de la arquitectura anterior, no se considera como una función de activación, sino más bien como una función que modela la información mediante un sistema lógico difuso. A diferencia de la lógica difusa clásica, donde se asigna una etiqueta a una función y es interpretada de acuerdo con estas definiciones, en este caso, la función se interpreta en base a sus preimágenes, esta interpretación se expresa tal y como se observa en el tema 4.2.

Luego, estas variables lingüísticas, así denominadas por su capacidad de generar múltiples funciones de membresía interpretables de manera lingüística, pasan a la siguiente capa, conocida como capa conjuntiva. Esta capa puede ser evaluada mediante una lógica Arquimediana (AFL) o Compensatoria (CFL), definidas a partir de las ecuaciones 4.11 y 4.12 respectivamente. Este operador lógico conjuntivo, permite expresar las entradas en forma de un predicado conjuntivo, es decir, $Si\ x_1\ y\ x_2\ y \ldots y\ x_n$. Si bien pueden definirse

tantas capas conjuntivas como sea necesario para resolver el problema, tener menos capas conjuntivas facilita la comprensión del modelo.

Finalmente, en la capa final se utiliza una función disyuntiva, que se evalúa mediante un operador Arquimediano o compensatorio. A través del cual se genera un predicado en forma normal disyuntiva, es decir un predicado de la forma $si\ p_1 o\ p_2\ o \ldots o\ p_n$, donde $p_i$es un predicado conjuntivo de un nodo i en la capa $L-1$.
La descripción de la NN-ACFL se presenta a continuación.

Capa inicial (0): Esta capa funciona exactamente igual que la capa descrita en el modelo anterior de una NN-GCLV. En esta los datos son transformado al intervalo [0,1] usando el algoritmo descrito en el Anexo A.

Capa de GCLV (1): En esta capa, lo datos de entrada los cuales son los atributos y campo clase denominados en esta ocasión como x_i, pasan a través de una $GCLV_L(ec.\ 4.17)$, tal que los valores sufren una normalización lógica multivalente en el intervalo [0,1]. Tal que las funciones de membresía generadas pueden tener diferentes interpretaciones en función del perfil definido mediante los parámetros α, γ y m.

$$GCLV_L(x_i;\alpha,\gamma,m)=\frac{{}^{n}\sqrt{log_b^n\left(s_g(x_i;\alpha,\gamma)_L^m\right)+log_b^n\left(1-s_g(x_i;\alpha,\gamma)_L^{1-m}\right)}}{max_{x_i\in\mathbb{R}}\left[{}^{n}\sqrt{log_b^n\left(s_g(x_i;\alpha,\gamma)_L^m\right)+log_b^n\left(1-s_g(x_i;\alpha,\gamma)_L^{1-m}\right)}\right]}$$

En la Figura 4.25, se observa el proceso de transformación de los valores de entrada mediante el uso de una *GCLV*, la cual puede generar diversas funciones de membresía, adaptándose a los datos.

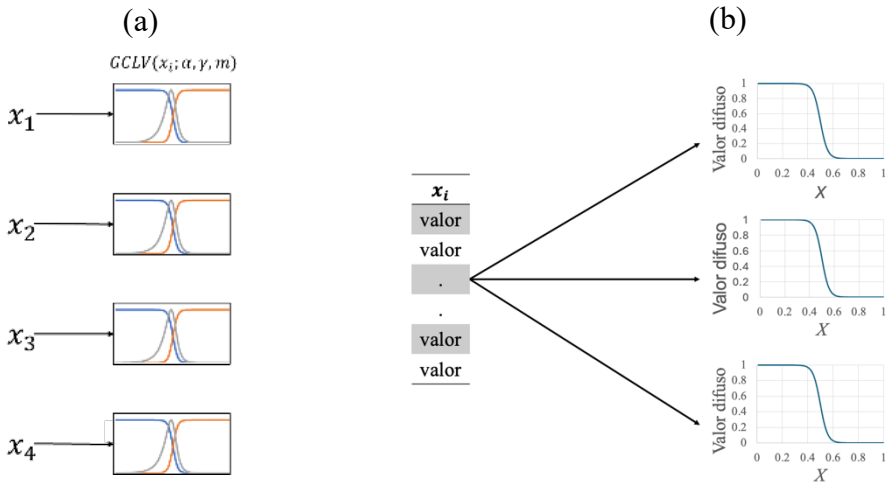


Figura 4.25: Ejemplos de la transformación de valores de entrada a valores difusos mediante una GCLV.

En la Figura 4.25 (a) se puede observar de manera gráfica, como cada valor $x_1$del conjunto de datos es transformado mediante una GCLV a múltiples funciones de membresía, así mismo en (b) se ejemplifica este proceso a partir del uso del atributo x_i, donde se observa que para este puede ser generada una gran cantidad de funciones de membresía basadas en los valores de sus parámetros. Aunque si bien esto es cierto para los parámetros, los campos clases son modelados a partir de una $GCLV(x_{clase}; \alpha = 1, \gamma = 0.5, m = 1)$.

Además, es importante comentar que cada uno de los atributos forma la base interpretable que construye el predicado conjuntivo de la capa subsecuente.

Capa conjuntiva: En esta capa, cada $GCLV_i$ generada en la capa anterior, es afectada por un modificador lingüístico $ml_L^w(GCLV_i)$, el cual ajusta la función de pertenencia mediante la inserción del parámetro w que define un modificador lingüístico. Permitiendo variar la interpretación del nodo al pasar al nodo de la segunda capa haciendo más estricta la evaluación o generando su inversa modificada. Este modificador se calcula a partir de la fórmula:

$$ml_L^w = \begin{cases} f^{-1}\big(w.f(x)\big) \ \ si\ w > 0 \\ f^{-1}\big(|w|.f(1-x)\big)\ si\ w < 0 \end{cases} \qquad (4.29)$$

En la *ec.* 4.29, los valores w pueden ser tanto positivos como negativos y tiene su equivalencia en el concepto del uso de pesos en las redes neuronales clásicas, un ejemplo de su funcionamiento puede ser apreciado en la Figura 4.26.

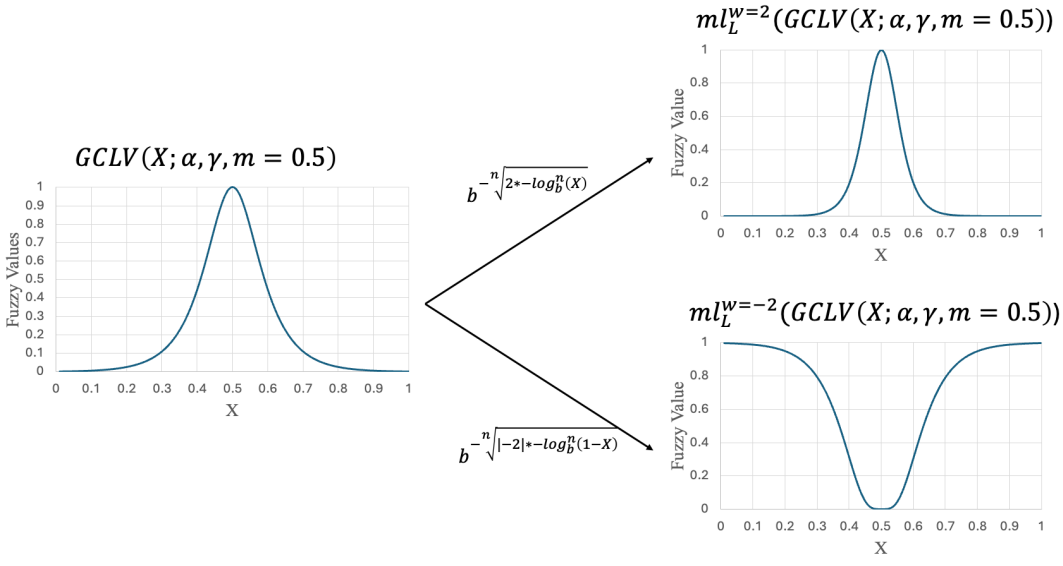


Figura 4.26: Ejemplo del funcionamiento de un modificador lingüístico.

Como se observa en la Figura 4.26, la GCLV genera una función de membresía convexa definida a través de $m = 0.5$, con centro en $x = 0.5$ el cual corresponde al valor difuso 1.0. esta función es afectada por un modificador lingüístico 2 y -2, en ambos casos la función inicial es modificada, cambiando así los grados de pertenencia de cada *X* para la función, en el caso donde *w* es negativo, construye su inversa modificada, obteniendo de esta manera funciones convexas Entonces, una vez modificadas las funciones de membresía, en la segunda capa los valores de cada $GCLV_i$ afectados por el modificador lingüístico $(ml_L^w(GCLV_i))$, son tratados mediante la operación de conjunción lógica c_c (*ec.* 4.12) o c_t (*ec.* 4.11):

$$c_T(x_1, x_2, \dots, x_n) = b^{\sqrt[n]{\sum_{i=1}^{n} log_b^n(x_i)}}$$

$$c_c(x_i, \dots, x_n) = f^{-1}\left(\sum_{i=1}^{n} \frac{f\left(ml_L^w(GCLV_i)\right)}{n}\right)$$

Lo cual define que para cada nodo conjuntivo se defina un predicado del tipo $si\ \left(ml_L^w(GCLV_1)\right)\ y\ \left(ml_L^w(GCLV_2)\right)\ y \dots y\ (ml_L^w(GCLV_n))$ el cual se interpreta como: si función de membresía 1 y función de membresía 2 y … y función de membresía *n* todas ellas afectadas por el modificador *w*. El cual puede ser evaluado tanto por c_c, como por c_t. Lo cual se observa de manera gráfica en la Figura 4.27.

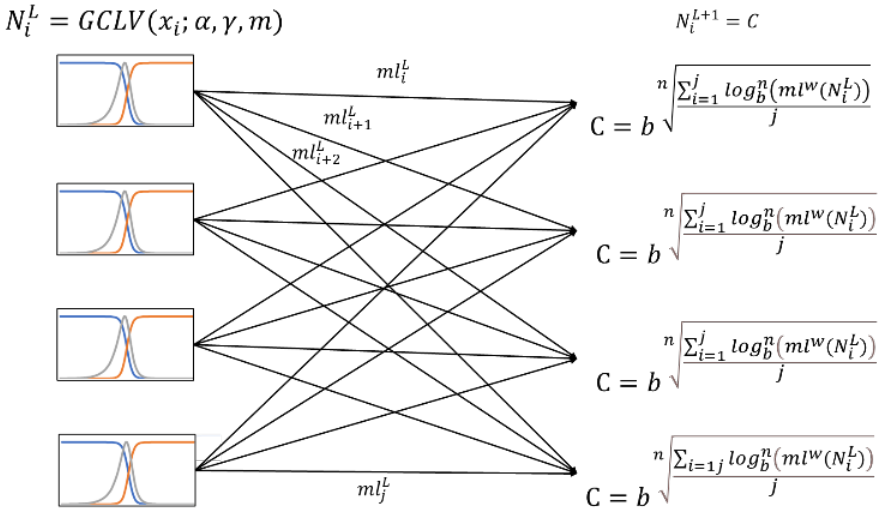


Figura 4.27: Representación gráfica del paso de la primera capa a la segunda capa.

Un ejemplo de este proceso se presenta mediante la Tabla 4.25 y la Figura 4.28, donde se utilizan los valores de los atributos del conjunto de datos iris afectadas por un modificador y evaluadas a través de la ecuación de conjunción compensatoria.

Tabla 4.25: Calculo de la conjunción.

Registro	Sepal length	Sepal width	Petal lenght	Petal width	Conjunción
1	0.004	0.905	0.254	0.246	0.453
2	0.735	0.390	0.981	0.339	0.335
3	0.609	0.937	0.343	0.268	0.422
4	0.503	0.877	0.127	0.738	0.093
5	0.401	0.677	0.188	0.319	0.358
6	0.999	0.612	0.156	0.747	0.090
7	0.234	0.028	0.981	0.200	0.526
8	0.646	0.620	0.167	0.466	0.223

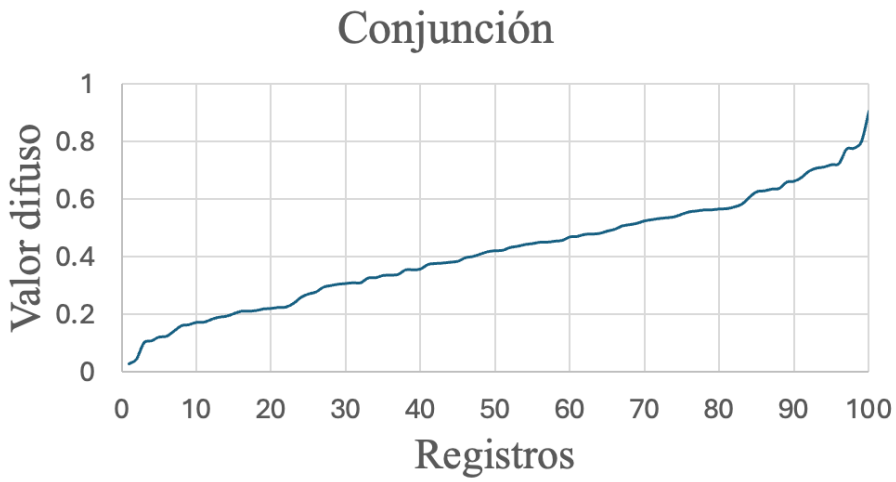


Figura 4.28: Representación de la función generada a través de una c_c.

En este contexto, *w* opera de manera similar a lo que se conoce tradicionalmente como peso. Sin embargo, este parámetro permite realizar una interpretación lingüística de la función resultante. En otras palabras, modifica la función de membresía inicial, alterando su interpretación por otra.

Dado que, para cada nodo conjuntivo, el valor de entrada es una función de membresía afectada por un parámetro *w*, facilitando la búsqueda exhaustiva y profunda de predicados. Gracias a sus múltiples posibilidades, la creación de predicados es amplia y adaptable.

Aunque es posible definir múltiples capas conjuntivas de una lógica AFL o CFL, se recomienda utilizar solo una. Lo cual permite obtener un resultado interpretable y explicable. El uso de múltiples capas puede hacer que la interpretación sea compleja debido a la cantidad de información presente, lo que dificulta la comprensión del resultado final.

Capa disyuntiva: También definida como capa de salida, en esta capa, todos los predicados conjuntivos generados en la capa anterior son alterados mediante un modificador lingüístico (ml_L^w) Este modificador funciona tal como se explicó anteriormente, es decir, mediante la definición de un parámetro *w* que permite reinterpretar una función definida previamente.

Los predicados modificados, son posteriormente evaluados por un operador de disyunción d_T (*ec.*4.13) o d_c (*ec.*4.14):

$$d_T(x_1, x_2, \dots, x_n) = 1 - \sqrt[n]{\sum_{i=1}^{n} log_b^n(1-x_i)}$$

$$d_c(x_1, x_{2,\dots,} x_n) = 1 - \sqrt[n]{\frac{\sum_{i=1}^{n} \log_b^n(1-x_i)}{n}}$$

Lo cual nos permite generar como salida un predicado en forma normal disyuntiva, es decir, $si\ \left(ml_L^w(p_{1,L-1})\right)\ o\ \left(ml_L^w(p_{2,L-1})\right)\ o \dots o\ \left(ml_L^w(p_{n,L-1})\right)$, el cual puede interpretarse como: si predicado 1 de la capa *L-1* o predicado 2 de la capa *L-1* o … o predicado *n* de la capa *L-1*. Predicado el cual puede ser evaluado tanto por una AFL como una CFL.

En la Figura 4.29 se observa el proceso del paso de la segunda a la tercera capa de forma gráfica.

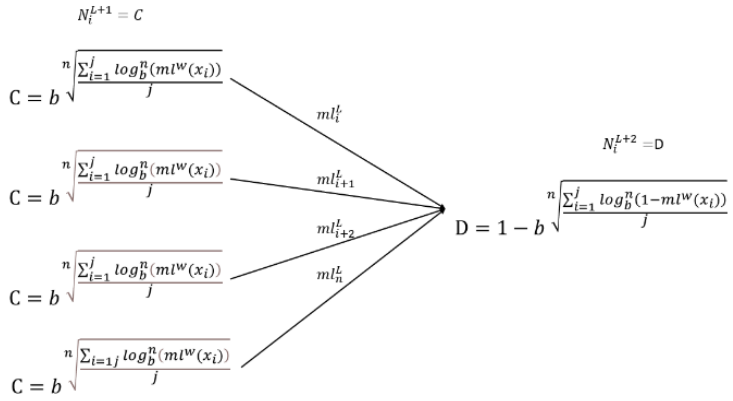


Figura 4.29: Paso de la segunda capa o capa conjuntiva hacia la tercera capa o capa disyuntiva.

Tal como se puede observar en esta Figura, los predicados conjuntivos generados en cada nodo i (N_i) de la capa $L+1$ son afectados por un modificador lingüístico en su paso a la capa de salida o capa disyuntiva, lo cual nos permite que a través de la estructura de la red se pueda obtener un predicado interpretable y explicable. Lo cual además de brindar una solución al problema evaluada, nos permite justificar el resultado expresando cada uno de sus conceptos en un lenguaje cercano al lenguaje natural.

En la Tabla 4.26 y la Figura 4.30, se observa un ejemplo de cómo la función generada a través de un operador de disyunción compensatoria (d_c).

Tabla 4.26: Calculo de la disyunción.

Registro	Sepal length	Sepal width	Petal lenght	Petal width	Disyunción
1	0.004	0.905	0.254	0.246	0.0502331
2	0.735	0.390	0.981	0.339	0.18381292
3	0.609	0.937	0.343	0.268	0.24480091
4	0.503	0.877	0.127	0.738	0.27273964
5	0.401	0.677	0.188	0.319	0.33677759
6	0.999	0.612	0.156	0.747	0.34964117
7	0.234	0.028	0.981	0.200	0.36101791
8	0.646	0.620	0.167	0.466	0.36427398

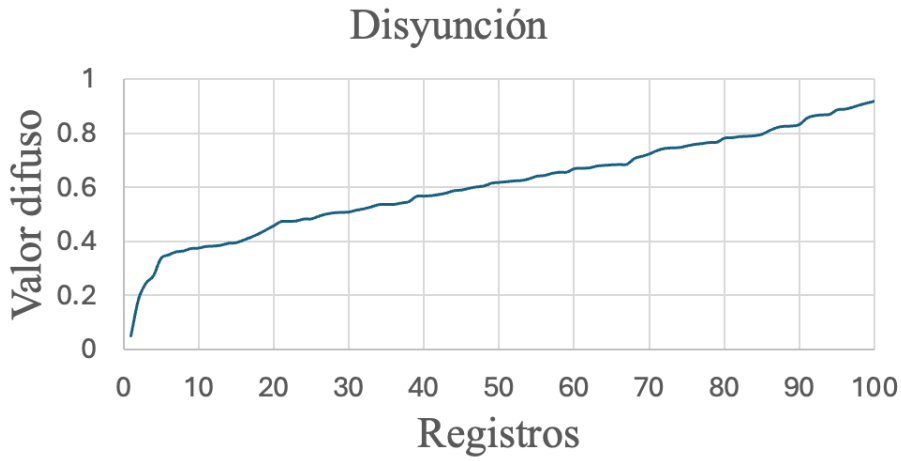


Figura 4.30: Ejemplo de la función generada a través de un operador de disyunción compensatoria (d_c).

El nodo disyuntivo de salida busca construir una función que permita ser equivalente a una función de clases la cual se modela inicialmente a través de la siguiente $GCLV(clase, \alpha = 1, \gamma = 0.5, m = 1)$, lo cual define una función sigmoidal con centro en el valor 0.5 y una amplitud igual a la unidad, lo que permite generar una GCLV de clases.

Y donde si la función de disyunción *D* tiene un valor de salida difuso igual a 1.0, el valor de la función de clases será igual a 1.0, si el valor de salida de *D* es igual a 0.5, el valor de la clase será 0.5, entre algunos ejemplos. En la Figura 4.31 se observa este comportamiento.

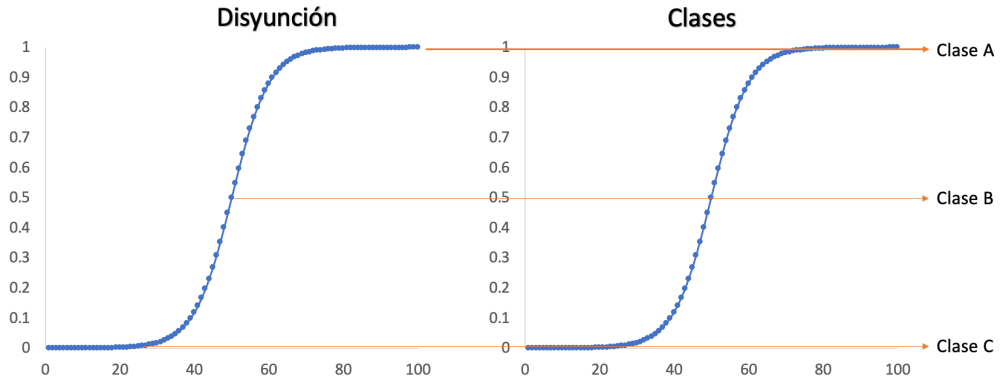


Figura 4.31: Ejemplo de la salida de la red en el proceso de inferencia de clase.

Una vez explicadas las capas que forman parte de una arquitectura de red neuronal híbrida de lógica difusa Arquimediana compensatoria definida como NN-ACFL, en el Algoritmo 4.7 se presenta la implementación de esta red neuronal artificial que usa una arquitectura de red tipo propagación hacia adelante, donde se sustituye los conceptos clásicos de las redes neuronales tales como peso (w), sesgo (b), funciones de activación, por operadores lógicos (c_c, c_t, d_c, d_t), modificador lingüístico (w) y variables lingüísticas continuas generalizadas ($GCLV$) .

Algoritmo 4.7: NN-ACFL

Entrada: *Dataset* // Conjunto de datos a evaluar
Salida: *Inferencia* // Conjunto de datos predichos, reales y su diferencia absoluta
Parámetros //Conjunto de parámetros descubiertos en el proceso de optimización
Métricas // Conjunto de métricas de evaluación de rendimiento del algoritmo

1. *Data*, *Mínimo*, *Máximo* = read_dataset("*Dataset*.csv") //Normaliza los datos del archivo csv al intervalo [0,1] y devuelve el valor máximo y mínimo de la clase
2. *inputs* ← *fila*[:-1] **para cada** *fila* **en** *data* //Guarda en inputs los datos usados en el entrenamiento
3. *labels* ← *fila*[-1] **para cada** *fila* **en** *data* //Guarda en labels los valores reales de los registros
4. *kf* ← KFold(*numero_folds*, *aleatorio*=true, *semilla*) // Genera la arquitectura de cross fold.
5. **Para cada** *fold* **en** *kf.folds*
6. *Train_set* ← *kf*.split(*inputs*), *kf*.split(*labels*) // Divide en conjuntos de entrenamiento los datos
7. *Test_set* ← *kf*.split(*inputs*), *kf*.split(*labels*) //Divide en conjuntos de prueba los datos
8. *Train_loader* ← DataLoader(*Train_set*, *batch_size*, *aleatorio* = true) //Configura el conjunto de entrenamiento
9. *Test_loader* ← DataLoader(*Test_set*, *batch_size*, *aleatorio* = true) //Configura el conjunto de prueba
10. network ←Net() //Crea una instancia del modelo de red neuronal (Algoritmo 4.8)
11. criterio ← MSE_Loss() // Define la función de error

```
12.  optimizer ← optimizer.ADAM(network.parameters) //Se establece el algoritmo de
     retropropagación usado en la optimización
13.  Para cada época en épocas
14.     Para cada train_inputs, train_labels en Train_loader
15.        outputs = network(train_inputs) //Devuelve los valores inferidos por la red
16.        loss = criterio(outputs, train_labels) //Calcula el error de la red
17.        optimizer.gradienteszero //Reinicia los gradientes
18.        loss.backward() //Retro propaga el error
19.        optimizer.actualizar() //Calcula los nuevos valores de los parámetros
20.  Para cada test_inputs, test_label en test_loader
21.  imprimir_resultados(network(test_inputs), test_labels, Maximo, Minimo)
22.  params = network.state_dict()
23.  imprimir_parametros(params)
```

El código generado para el Algoritmo 4.7 ha sido desarrollado en el lenguaje de programación Python, el cual hace uso de las siguientes bibliotecas Pytorch y Sklearn que pueden ser descargadas directamente desde el gestor de Python, así como de las bibliotecas ActivationsFunctions, datareader y resultados las cuales fueron implementadas en el siguiente desarrollo.

Conceptos importantes de la arquitectura como la descripción de las bibliotecas, los métodos implementados, métricas de evaluación, entre otras cosas están detalladas en el Anexo A.

En la línea 1 del código, se recibe el conjunto de datos en formato CSV. Este archivo es procesado mediante la función read_dataset, la cual pertenece a la biblioteca datareader. Esta función, detallada en el Anexo A, se encarga de normalizar los datos de cada campo del conjunto de datos al intervalo [0,1] y guarda los resultados en *Data*. Además, devuelve los valores máximos (*Máximo*) y mínimos (*Mínimo*) del campo clase, los cuales serán necesarios para la evaluación de los resultados al final del proceso.

En la línea 2 del algoritmo, se genera una matriz llamada *inputs* donde se guardan los campos desde 1 hasta *n-1* de la información contenida en *Data*. Dichos campos corresponden a los atributos que serán utilizados en el proceso de optimización llevado a cabo mediante la NN-GCLV.

En la línea 3, al igual que la anterior, se genera una matriz denominada *labels*, donde se almacenan los datos del campo *n* almacenados en *Data* que corresponden a las clases a inferir. Estos datos se utilizan para calcular el error de la NN-GCLV y permiten realizar el proceso de optimización a partir del cálculo del error.

En la línea 4, se configura el algoritmo de validación cruzada (cross fold validation) a través de la implementación del método KFolds perteneciente a la biblioteca Sklearn, en este se define el número de folds en que se dividirá el algoritmo, permitiendo también mezclar los

datos de forma aleatoria antes de su división. Detallando los datos más importantes tenemos lo siguiente:

a. **kf**: Variable que representa el objeto generado por el método KFold(). Este objeto es usado para dividir los datos en conjuntos de entrenamiento y prueba en cada iteración de la validación cruzada.
b. **KFold**: Es el método utilizado para realizar la validación cruzada mediante el esquema KFold. Este método divide el conjunto de datos en *k* folds y utiliza *k-1* folds para entrenamiento y el fold restante es usado en el proceso de evaluación. Este proceso de entrenamiento y evaluación se repite *k* veces, utilizando cada fold como conjunto de prueba exactamente una vez.
c. **numero_folds**: Este parámetro indica el número de folds en los que se dividirá el conjunto de datos. Donde para estas pruebas se ha establecido un total de 5 folds.
d. **aleatorio**: Este parámetro permite realizar el particionamiento de los datos de forma aleatoria antes de dividirlos en folds. En este algoritmo se establece aleatorio como True, lo cual significa que los datos se mezclarán aleatoriamente antes de la división. Si se cambia aleatorio a False, los datos se dividirán en folds ordenados de manera secuencial.
e. **semilla**: Este parámetro especifica la semilla utilizada para la generación de números aleatorios que son usados en el proceso de división aleatoria de los datos. Es este caso se establece una semilla fija igual a 42, lo cual es útil debido a que permite reproducir los mismos resultados en cada ejecución del código.

La línea 5 indica que se llevarán a cabo un total de *folds* pruebas, que en este caso está configurado en cinco, es decir, que el número de registros por fold será igual a $Total\_registros/5$. Entonces, para cada *fold* de prueba dentro del objeto *kf* se ejecutarán las líneas del 6 al 23 del algoritmo.

La línea 6 y 7 se crean los conjuntos de datos correspondientes al entrenamiento *Train_set* y al de prueba *Test_set*, a través de la selección de los índices de las filas correspondientes a los conjuntos de datos *inputs* que corresponde a los atributos y *labels* que corresponde a las clases correspondientes.

En la línea 8 se genera un objeto DataLoader perteneciente a la biblioteca Sklearn es contenido en la variable *Train_loader.* Este objeto se utiliza para cargar los datos de entrenamiento contenidos en *Train_set* y segmentarlos en lotes de tamaño *Batch_size.* En el presente modelo, se utiliza un tamaño de lote igual al número de registros contenidos en *Train_loader* divididos entre cinco. Además, el parámetro $aleatorio = True$ indica que los datos contenidos en *Train_loader* se mezclarán aleatoriamente en cada iteración de entrenamiento, lo que permite evitar el sobreajuste y mejorar la generalización del modelo.

En la línea 9 se realiza un proceso similar al de la línea 8 para definir el conjunto de prueba denominado *Test_loader*. Al igual que en la línea 8, se utiliza un objeto DataLoader para cargar los datos de prueba contenidos en *Test_set* y segmentarlos en lotes de tamaño

Batch_size. Sin embargo, a diferencia del proceso anterior, el tamaño del lote en este caso es igual al número de registros contenidos en *Test_set,* lo que permite procesar todos los datos de prueba en un solo lote. A través de este proceso se evalúa el rendimiento del modelo de forma completa en el conjunto de prueba.

En la línea 10, se instancia la clase Net asignando esta instancia a la variable network. Esta clase representa la arquitectura de la red neuronal propuesta, la cual se utiliza en el proceso de entrenamiento y evaluación del modelo de aprendizaje automático.

En la línea 11, se define la función de pérdida que se utilizará para calcular el error entre las predicciones del modelo y las etiquetas reales. Este valor calculado se usará posteriormente para ajustar el modelo. Dentro de la biblioteca de PyTorch, se encuentran implementadas una amplia gama de funciones de error. Para este caso se ha definido usar la función de error cuadrático medio (MSE) en función de su desempeño en experimentos anteriores

La línea 12 permite definir el algoritmo de optimización que se utilizará para actualizar los parámetros w de las capas conjuntivas y disyuntivas, así como los parámetros α, γ y m de una GCLV. Los parámetros mencionados permiten modelar la arquitectura de la red y llevar a cabo el proceso de aprendizaje automático. Para este caso se implementa el algoritmo ADAM como optimizador del modelo, debido al desempeño observado en experimentos anteriores.

En la línea 13 se define el número de épocas que se utilizarán para resolver cada problema. Este valor es propuesto por el usuario y puede ajustarse según las necesidades específicas del problema en cuestión. Por lo general, para conjuntos de datos pequeños, mil épocas suelen ser suficientes. Sin embargo, para problemas mayor complejidad o conjuntos de datos grandes, puede ser necesario aumentar la cantidad de épocas para obtener resultados óptimos. Es importante ajustar este valor según la complejidad y el tamaño del conjunto de datos para garantizar un entrenamiento adecuado del modelo.

En la línea 14, los datos de entrenamiento se almacenan en la variable *train_inputs* y las etiquetas de las clases correspondientes se almacenan en la variable *train_labels,* siguiendo la estructura definida en el objeto *train_loader*. Para cada iteración, se extrae un set de datos de un tamaño definido por *batch_size*, que para este caso es igual al número total de registros en *train_loader* dividido entre cinco, por lo cual, se realiza un total de cinco iteraciones por proceso de entrenamiento. Lo cual significa que, durante cada iteración, la arquitectura de la red se entrena con un subconjunto de datos equivalente al tamaño del lote.

La línea 15 corresponde a la etapa de propagación hacia adelante (forward propagation) correspondiente a una red neuronal tipo propagación hacia adelante en el proceso de entrenamiento de la red neuronal. En esta línea, se pasa el lote de datos de entrenamiento *train_inputs* a través de la red neuronal network. Cada vez que se realiza esta operación, la

red neuronal produce predicciones para el lote de datos de entrada de tamaño igual a *Batch_size*, las cuales se almacenan en la variable *outputs*.

En la línea 16, se calcula el error de las predicciones de la red comparando las predicciones almacenadas en *outputs* con los valores reales de las etiquetas de los datos de entrenamiento en *train_labels*, utilizando la función de pérdida definida anteriormente. Este cálculo de error es necesario para evaluar la eficiencia del proceso de aprendizaje aplicado a la red neuronal y es utilizado para ajustar los parámetros del modelo durante la etapa de retropropagación.

En la línea 17, se utiliza el método zero_grad() proporcionado por la biblioteca PyTorch para restablecer todos los gradientes de los parámetros del modelo a cero. Esta operación es necesaria antes de realizar la retropropagación del error y actualizar los pesos de la red neuronal durante el proceso de entrenamiento.
En la línea 18, se llama al método backward() del objeto *loss*. Este método perteneciente a la biblioteca Pytorch, realiza el proceso de retropropagación del error a través de la red neuronal. Durante la retropropagación, se calculan los gradientes de la función de pérdida con respecto a los parámetros del modelo, lo que permite ajustar dichos parámetros para minimizar la pérdida en futuras iteraciones del proceso de entrenamiento.

En la línea 19, se llama al método step() del objeto *optimizer*. Este método perteneciente a la biblioteca Pytorch, actualiza los parámetros del modelo utilizando los gradientes calculados durante la retropropagación del error. Es decir, actualiza los pesos de la red neuronal de acuerdo con la tasa de aprendizaje y los gradientes de la función de pérdida para mejorar el rendimiento del modelo en la siguiente iteración del proceso de entrenamiento.

El segmento que corresponde a las líneas 13 al 19 del Algoritmo 4.7, representan el núcleo del proceso de aprendizaje automático de la arquitectura propuesta. En estas líneas se calcula el error entre las predicciones de la red y las etiquetas reales de los datos de entrenamiento, luego se realiza la retropropagación de este error para ajustar los parámetros de la red neuronal y mejorar su desempeño en la predicción. Este proceso se repite durante un número definido de épocas para permitir que el modelo aprenda de manera incremental y progresiva.

En la línea 20, se inicia un ciclo que itera sobre los datos de prueba cargados en el objeto Test_loader. Cada iteración de este ciclo toma un lote de datos de prueba del tamaño definido por *Batch_size*, el cual es igual al número de registros utilizados para el conjunto de prueba del fold actual. Las variables *test_inputs* y *test_labels* representan los datos de entrada y las etiquetas reales que se utilizan para evaluar el rendimiento del modelo en el conjunto de prueba.

En la línea 21, se utiliza la función imprimir_resultados, la cual forma parte de la biblioteca personalizada resultados detallada en el Anexo A. Esta función recibe como argumentos la predicción de la red neuronal sobre el conjunto de datos de prueba *test_inputs*, las etiquetas reales correspondientes a dichos datos *test_labels*, los valores *Máximo* y *Mínimo* que se

utilizan para transformar los valores difusos a su magnitud real, lo que permite compararlos con la magnitud original del campo de clase.

En la línea 22 se obtienen los parámetros descubiertos en el proceso de optimización de la red neuronal y se almacenan en la variable *params*. Lo cual se logra utilizando el método state_dict() perteneciente a la biblioteca Pytorch.

En la línea final 23, se utiliza la función imprimir_parametros, que forma parte de la biblioteca personalizada resultados. Esta función recibe un único argumento la información contenida en la variable *params*. Su objetivo es imprimir los parámetros descubiertos por el modelo en un formato predefinido, lo que facilita su análisis y evaluación.

El algoritmo 4.8 define en la clase Net la arquitectura de la red neuronal. Este algoritmo se compone de dos funciones principales las cuales corresponden a la arquitectura mostrada en la Figura 4.24, particularmente a las capas intermedias y la capa final.

Algoritmo 4.8 Clase Net //arquitectura de la red neuronal de ACFL
Función __init__(self) //Inicialización de la clase

1. Define una capa conjuntiva que pasa de la capa L a la capa $L + 1$
   Fc1← ActivationsFunctions.ConjunctionCFL(*Nodos_entrada*, N*odos_salida, base, exponente*)
2. Se declaran tantas capas conjuntivas como capas sean necesarias. En la capa final entra un numero n de nodos y entra a un único nodo de salida
   Fcn ← ActivationsFunctions.DisjunctionCFL(*Nodos_entrada*, 1*, base, exponente*)
3. Genera una GCLV para cada atributo que es definido en la capa 0
   Activation1← ActivationsFunctions.GCLV(*Nodos*, *Base_logaritmica*, *Exponente*)

Función forward(*train_set*)

1. Se modelan los datos mediante una GCLV
   train_set ← Activation1(*train_set*)
2. Se modifican las GCLV mediante el parámetro w y son definidos como predicados evaluados mediante el operador de conjunción. Se generan tantas capas conjuntivas como se requieran, se recomienda solo una
   train_set ← Fc1(*train_set*)
3. Los predicados conjuntivos se modifican mediante el parámetro w y son evaluados mediante el operador de disyunción.
   train_set ← Fcn(*train_set*)
   Devolver *train_set*

La clase Net encapsula la arquitectura de la red neuronal, que incluye las siguientes capas:

i. Capa de GCLV. En esta a los datos de la capa 0, se les realiza una normalización lógica multivalente definida a partir de GCLV.

ii. Una o más capas conjuntivas, donde cada nodo representa un predicado conjuntivo. Cada función GCLV, c_c o c_t que componen los predicados se alteran mediante un modificador lingüístico (w).
iii. Capa disyuntiva de salida, donde cada predicado construido en la capa anterior, definido a través de un operador de conjunción, se altera mediante un modificador lingüístico (w). La salida de esta capa es un predicado en forma normal disyuntiva.

La interpretación de las GCLV que conforman cada predicado conjuntivo en la estructura disyuntiva permite inferir o clasificar cada uno de los elementos del conjunto de datos evaluado.

La clase Net se compone de dos funciones principales las cuales corresponden a la arquitectura mostrada en la Figura 4.24 y a la explicación de las capas que a conforman, las cuales se detallan a continuación:

Función init: Este método inicializa la red neuronal definiendo las capas que conforman la arquitectura propuesta. A diferencia de la propuesta anterior, en esta arquitectura no existe el concepto de función lineal, todos los datos son tratados específicamente a partir de los conceptos de una ACFL-ELF.

Para iniciar la arquitectura, todo atributo del conjunto de datos normalizado al intervalo $[0,1]$ exepto el de clases, es tratado mediante la función ActivationsFunctions.GCLV(*Nodos*, *Base_logaritmica*, *Exponente*) la cual se detalla en el Anexo A, en esta función, *Nodos* define el número de atributos para los cuales se generará una variable lingüística continua generalizada donde los parámetros de una GCLV se inician con $\alpha\epsilon[4,10]$, $\gamma = 0.5$ y $m\epsilon[0,1]$. Para esta función se definen también los parámetros de la ACFL-ELF base logarítmica y valor exponencial. El número de GCLV es igual al número de atributos de entrada.

Posteriormente se hace uso de la función ConjunctionCFL(*Nodos_entrada*, N*odos_salida, base, exponente* la cual toma como *Nodos_entrada* la cantidad de GCLV generados en la capa anterior, y se define como *Nodos_salida* la cantidad de predicados conjuntivos que se deseen usar en el proceso de entrenamiento, cada predicado conjuntivo contiene cada una de las GCLV de la capa anterior dentro de su estructura, cada una de estas es modificada para cada predicado mediante el uso de un modificador lingüístico *w* de tal modo que si bien cada predicado tiene como base la misma GCLV, esta no opera de la misma forma en cada predicado. Los parámetros w se generan de manera aleatoria usando la función Kaiming_uniform de Pytorch. Si se desea evaluar la red usando una AFL, solo es necesario cambiar ConjunctionCFL por ConjunctionAFL.

Si se definen otras capas conjuntivas, cada nodo existente se tomará como la conjunción de los predicados generados en la capa anterior, si bien esto es posible, no se recomienda ya

que, a mayor cantidad de capas conjuntivas, más difícil es comprender los resultados obtenidos.

Finalmente la salida se obtiene a partir de la función DisjunctionCFL(*Nodos_entrada*, 1*, base, exponente*) donde *Nodos_entrada* define la cantidad de predicados conjuntivos existentes en la estructura del predicado en forma normal disyuntiva, los predicados son alterados mediante el uso de un modificador lingüístico *w*, con lo cual se busca generar como única salida una función que busca emular una función de clases, permitiendo de esta manera realizar procesos de clasificación e inferencia de datos.

Función forward: Este método define cómo se propagan los datos a través de la red. Aquí se especifica la secuencia de operaciones que se aplican a los datos de entrada del *train_set*. Primero, los datos pasan a través cada concepto de una ACFL-ELF, inicialmente modelando los datos de los atributos mediante una GCLV usando Activations1, posteriormente cada GCLV que forma parte de activations1 es usada en la construcción de tantos predicados conjuntivos como *Nodos_salida* en *fc1* hayan sido definidos. Por último, los predicados conjuntivos son tratados mediante *fcn* la cual toma los predicados construidos, entregando una única función de salida.

El número de capas conjuntivas y nodos por capa se propone por el usuario y este debe ajustarse a las necesidades específicas del problema. Sin embargo, basado en observaciones realizadas, se ha notado que una sola capa intermedia con un número de nodos que aumenta de acuerdo con la complejidad del problema ofrece una buena capacidad de adaptación y entrenamiento del modelo. Lo cual significa que, para problemas simples, es posible implementar un menor número de nodos, mientras que, en problemas complejos, se hace necesario añadir más nodos en la capa intermedia. Aun así, es necesario experimentar con diferentes configuraciones de capas y nodos de capa para encontrar la combinación óptima que mejor se adapte al problema en cuestión. Además, la interpretación y comprensión de una única capa conjuntiva es más sencilla.

4.3.4 Experimentación con una NN-ACFL

Una vez que se ha detallado la arquitectura de la NN-ACFL y definido su implementación a través del Algoritmo 4.7, se procede a realizar una serie de experimentos para evaluar su robustez, precisión y eficacia en tareas de clasificación e inferencia. Los experimentos planteados también buscan confirmar la capacidad del modelo para representar predicados de lógica difusa Arquimediana compensatoria, así como observar su capacidad para interpretar y explicar los resultados obtenidos.

Los experimentos se realizaron utilizando una MacBook con las siguientes especificaciones:

- Procesador: Apple M1.
- Memoria RAM: 8 GB.
- Sistema Operativo: macOS Sonoma v14.4.1.

Como técnica de evaluación del modelo se hace uso de la técnica de validación cruzada, la cual es ampliamente usado en la literatura para evaluar el rendimiento de modelos predictivos. La configuración de este algoritmo se detalla en la sección 4.3.1 durante la explicación de las líneas 4 y 5 del algoritmo 4.5.

Como técnica de evaluación del modelo, se emplea la validación cruzada, ampliamente utilizada en la literatura para evaluar el rendimiento de modelos predictivos. La configuración de este algoritmo se detalla en la sección 4.3.1 durante la explicación de las líneas 4 y 5 del algoritmo 4.5.

Se establece un total de *folds* de prueba, configurado en cinco en este caso, lo que implica que el número de registros por *fold* será igual a $total\_registros/5$. Para cada *fold* de prueba dentro del objeto kf, se ejecutan las líneas del 6 al 23 del Algoritmo 4.7.

Basándonos en los resultados observados en experimentos anteriores, se define como función de costo la función de error cuadrado mínimo y como método de optimización el algoritmo ADAM. Las metodologías planteadas, permiten determinar la distancia del error entre los valores predichos y los reales, así como retro propagar el error entre cada una de las capas. Además, mediante ADAM se optimiza el proceso de entrenamiento, lo que permite descubrir la mejor configuración de parámetros para ajustar el modelo de predicción y clasificación.

Además, se definen los valores base y exponente en 25 y 1, respectivamente, definiendo la función generadora de la siguiente forma: $f(x) = log_{25}^{1}(x)$. A través de esta lógica, se definen los operadores c_c, c_t, d_c, d_t, una variable lingüística continua generalizada (GCLV) y un modificador lingüístico (ml), describiéndolos de la siguiente manera:

$$c_T(x_1, x_2, \dots, x_n) = 25^{\sqrt[1]{\sum_{i=1}^{n} log_{25}^{1}(x_i)}} \quad (4.30)$$

$$c_c(x_1, x_{2,\dots,} x_n) = 25^{\sqrt[1]{\frac{\sum_{i=1}^{n} \log_{25}^{1}(x_i)}{n}}} \quad (4.31)$$

$$d_T(x_1, x_2, \dots, x_n) = 1 - 25^{\sqrt[1]{\sum_{i=1}^{n} log_{25}^{1}(1-x_i)}} \quad (4.32)$$

$$d_c(x_1, x_{2,\dots,} x_n) = 1 - 25^{\sqrt[1]{\frac{\sum_{i=1}^{n} \log_{25}^{1}(1-x_i)}{n}}} \quad (4.33)$$

$$ml_L^w(x) = 25^{-\sqrt[1]{w.-log_{25}^{1}(x)}} \text{ si } w > 0 \ \mathbf{O} \ ml_L^w(x) = 25^{-\sqrt[1]{|w|.-log_{25}^{1}(x)}} \text{ si } w < 0 \quad (4.34)$$

$$GCLV_L(x;\alpha,\gamma,m) = \frac{25^{\sqrt[1]{\log_{25}^{1}(s_g(x;\alpha,\gamma)_L^m)+\log_{25}^{1}(1-s_g(x;\alpha,\gamma)_L^{1-m})}}}{\max_{x\in\mathbb{R}}\left[25^{\sqrt[1]{\log_{25}^{1}(s_g(x;\alpha,\gamma)_L^m)+\log_{25}^{1}(1-s_g(x;\alpha,\gamma)_L^{1-m})}}\right]} \quad (4.35)$$

Experimento: Evaluación de desempeño de una NN-ACFL con diversos conjuntos de datos, en función de su precisión y eficacia en procesos de clasificación e inferencia de datos y su capacidad de ser interpretable y explicable.

En el siguiente experimento se hace uso de los conjunto de datos mostrados en la Tabla 4.27, donde se detallan características importantes usadas durante la experimentación.

Tabla 4.27: Descripción de los conjuntos de datos usados en el experimento.

Conjunto de datos	Atributos	Registros	Clase	Experimento
Daily Orders	13	60	Continua	Estimación
Dow Jones	16	750	Continua	Estimación
Iris	5	150	Discreta	Clasificación
Pima	9	768	Discreta	Clasificación

De la misma manera en la Tabla 4.28, se detallan las configuraciones de la arquitectura en una NN-ACFL para cada uno de los conjuntos de datos usados en la experimentación.

Tabla 4.28: Configuración de una NN-GCLV

Conjunto de datos	Capas	Nodos	Épocas
Daily orders	3	$L_0 = 12$ $L_1 = 6$ $L_2 = 1$	1000
Dow jones	3	$L_0 = 15$ $L_1 = 5,$ $L_2 = 1$	1000
Iris	3	$L_0 = 4,$ $L_1 = 8,$ $L_2 = 1$	1000
Pima	3	$L_0 = 8$ $L_1 = 5$ $L_2 = 1$	1000

Como se mencionó anteriormente, el método de evaluación usado es el de validación cruzada, para el cual se realizaron un total de cinco ejecuciones por cada uno de los conjuntos de datos del experimento, resultando así un total de 25 procesos de entrenamiento e inferencia, presentando de esta manera el resultado promedio de cada una de las métricas de evaluación usadas en el proceso de entrenamiento de la red.

En la Tabla 4.29 se observan los resultados obtenidos por cada una de las métricas de evaluación usadas para evaluar el entrenamiento en cada uno de los conjuntos de datos:

Tabla 4.29: Resultados del proceso de entrenamiento en función de las métricas de evaluación.

Conjunto de datos	MSE	MAE	R^2	Precisión con 0.25	Precisión con 0.5	Precisión con 1
Daily	2373.70	29.1940	0.6600	-	-	-
Dow	0.01543	0.07980	0.84135	57.92	79.413	82.253
Iris	0.53716	0.17747	0.91870	76.533	96.333	100
Pima	0.16166	0.34250	0.33302	40.909	77.402	-

De acuerdo con los resultados observados, se puede expresar lo siguiente:

Para el conjunto de datos Daily Orders, la cual está destinada a estimar el número total de órdenes diarias recibidas por una empresa de logística. La Tabla 4.29 revela un MSE (Error Cuadrático Medio) de 2373.70, indicando una alta variabilidad entre las predicciones y los valores reales. Además, se registra un MAE (Error Absoluto Medio) de 29.1940, un valor intermedio que sugiere cierta precisión en las predicciones. El coeficiente R^2 de 0.66 señala que se puede explicar el 66% de la variabilidad de la variable dependiente con las variables independientes, lo que representa una capacidad moderada de explicación.

Esto indica, que los resultados obtenidos, si bien no son los mejores, si presentan una aceptable capacidad predictiva (ver Tabla 4.30), la cual se observa en la Figura 4.32, aun así, es posible trabajar en métodos que mejoren estos resultados. La Figura 4.32 proporciona una representación gráfica de los resultados predichos por el modelo, comparados con los valores reales, lo que facilita la visualización y la comprensión de la efectividad del modelo.

Tabla 4.30: Resultados del proceso de inferencia con el conjunto de datos Daily Orders.

Registro	Predichos	Reales	Diferencia	Registro	Predichos	Reales	Diferencia
1	250.80	248.43	2.37	**7**	227.20	235.60	8.40
2	237.68	243.57	5.89	**8**	228.55	242.11	13.57
3	251.73	246.99	4.74	**9**	272.04	298.46	26.42
4	331.04	308.88	22.16	**10**	581.13	616.45	35.33
5	231.05	231.04	0.01	**11**	350.60	316.85	33.75
6	232.52	238.83	6.31	**12**	309.83	286.41	23.42

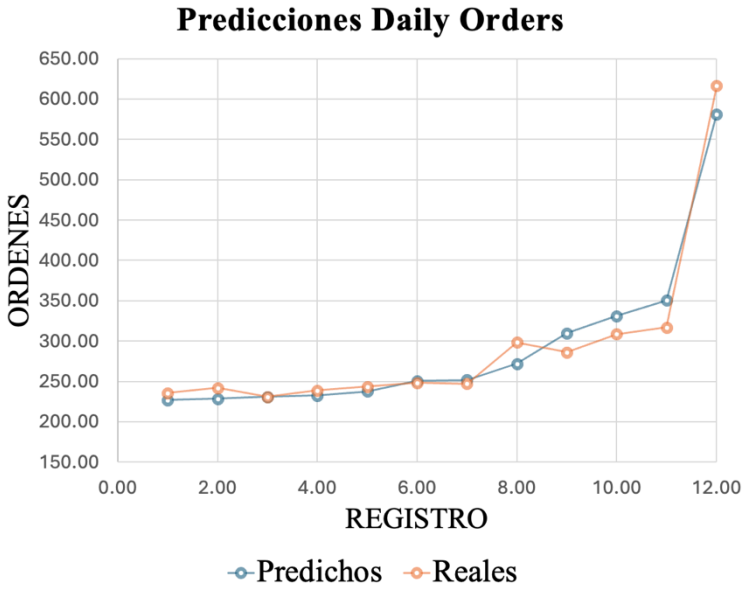


Figura 4.32 Representación gráfica de valores predichos contra valores reales para Daily Order.

Para el conjunto de datos Dow Jones la cual permite predecir el porcentaje de ganancias en la compra de acciones de la bolsa bursátil, los resultados del modelo son alentadores. Un MSE y un MAE bajos como 0.01543 y 0.07980, respectivamente, indican que las predicciones del modelo están cerca de los valores reales, lo que sugiere una alta precisión en las predicciones. Además, el valor de R2 de 0.8413 indica que el modelo puede explicar el 84.13% de la variabilidad de la variable dependiente, lo cual es un buen indicador de la capacidad explicativa del modelo (ver Tabla 4.31).

Por otro lado, los valores de aceptación de los umbrales mencionados en la Tabla 4.29, para este conjunto de datos no corresponden a los valores reales. Es decir, la precisión de la predicción con un intervalo de confianza de 0.25 es realmente de 0.025% el cual es igual a 57.92, la diferencia con un intervalo de confianza de 0.5 es realmente de 0.05% el cual es igual a 79.413 y la diferencia en el intervalo de confianza de 1.0 es realmente de 0.10% el cual es igual a 82.253. Un intervalo de confianza se define como la diferencia entre el valor real y el predicho, para el intervalo de confianza de 0.025 la diferencia debe ser menor al 2.5 % para ser tomado como correcto, es decir $|valor\ predicho - valor\ real\ | \leq 2.5$.

En la Figura 4.33 se presenta un ejemplo gráfico de la diferencia entre los valores reales y los valores predichos por el modelo. Esta representación gráfica proporciona una perspectiva visual de la calidad de las predicciones realizadas por el modelo en comparación con los valores verdaderos. Observar estas diferencias en un gráfico puede ayudar a comprender mejor la efectividad del modelo.

Tabla 4.31: Resultados de la inferencia con el conjunto de datos Dow Jones .

Registro	Predichos	Reales	Diferencia	Registro	Predichos	Reales	Diferencia
1	0.181	0.190	0.009	**11**	0.460	0.431	0.029
2	0.194	0.180	0.014	**12**	0.345	0.286	0.059
3	0.178	0.186	0.008	**13**	0.311	0.287	0.025
4	0.433	0.395	0.038	**14**	0.079	0.321	0.242
5	0.555	0.575	0.020	**15**	0.092	0.318	0.226
6	0.545	0.581	0.036	**16**	0.338	0.334	0.003
7	0.537	0.586	0.049	**17**	0.680	0.790	0.109
8	0.087	0.074	0.014	**18**	0.674	0.741	0.068
9	0.523	0.468	0.055	**19**	0.714	0.747	0.032
10	0.435	0.416	0.020	**20**	0.678	0.739	0.062

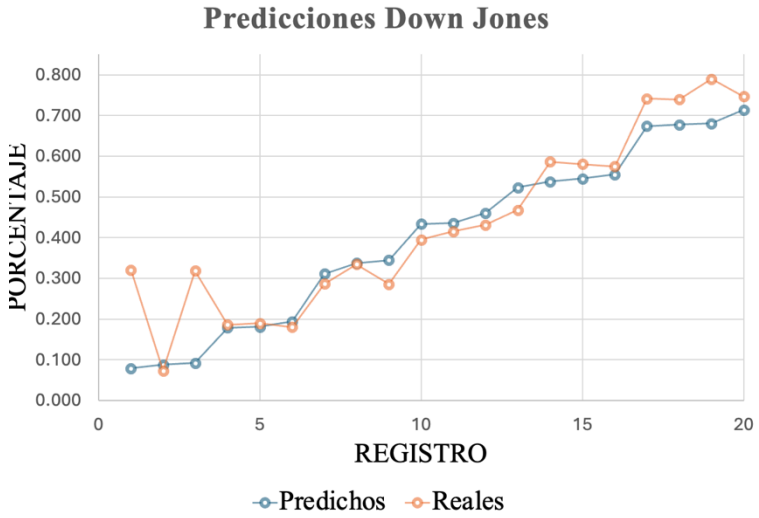


Figura 4.33: Representación gráfica de valores predichos contra valores reales para Dow Jones.

Para el conjunto de datos Iris Flowers los resultados del entrenamiento con una NN-ACFL son los siguientes: el MSE y el MAE son 0.537 y 0.177 respectivamente, valores que indican predicciones cercanas a los valores reales. El valor de R2 es 0.91870, lo que sugiere que el modelo explica el 91.87% de la variabilidad de las variables dependientes, mostrando una capacidad alta de explicación. Los intervalos de confianza de 0.25, 0.5 y 1.0 muestran precisión del 76.533%, 96.333% y 100% respectivamente, lo que indica una gran capacidad para inferir la clase correcta (ver Tabla 4.32).

La Figura 4.34 proporciona un ejemplo gráfico del proceso de predicción y su comparación con los valores de clase reales del modelo. Se destaca que el valor 0 corresponde a la clase

iris setosa, el valor 1 a la clase iris virginica y el valor 2 a la clase iris versicolor. Este análisis visual permite evaluar el desempeño del algoritmo NN-ACFL en procesos de clasificación.

Tabla 4.32: Resultados de la inferencia con el conjunto de datos Iris.

Registro	Predichos	Reales	Diferencia	Registro	Predichos	Reales	Diferencia
1	0.0766	0	0.0766	**11**	1.1374	1	0.1374
2	1.1543	1	0.1543	**12**	0.0726	0	0.0726
3	1.0793	1	0.0793	**13**	1.9997	2	0.0003
4	1.9998	2	0.0002	**14**	0.0844	0	0.0844
5	0.1344	0	0.1344	**15**	0.1129	0	0.1129
6	0.9359	1	0.0641	**16**	1.9998	2	0.0002
7	1.0361	1	0.0361	**17**	0.0477	0	0.0477
8	0.0801	0	0.0801	**18**	1.1292	1	0.1292
9	0.8260	1	0.1740	**19**	0.0982	0	0.0982
10	1.7972	2	0.2028	**20**	0.0869	0	0.0869

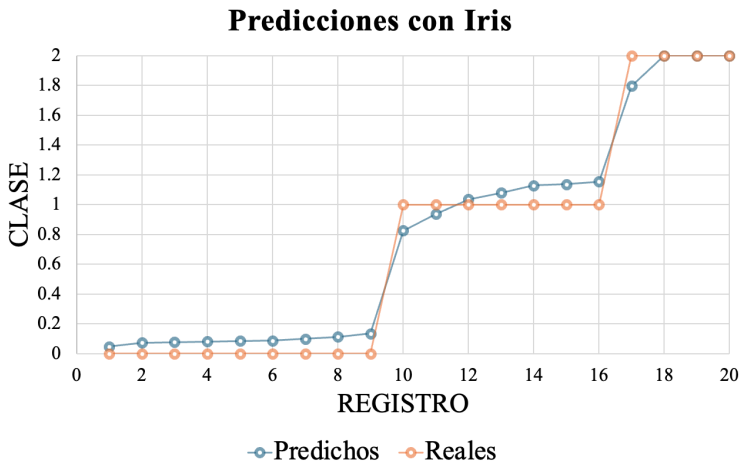


Figura 4.34: Representación gráfica de valores predichos contra valores reales para Iris Flowers.

Los resultados del algoritmo NN-ACFL en el proceso de entrenamiento del conjunto de datos Pima, que permite diagnosticar si una persona padece diabetes o no, muestran los siguientes resultados para las métricas de evaluación implementadas. Para el MSE y el MAE se obtuvieron 0.16166 y 0.3420 respectivamente, lo cual indica que las predicciones del modelo son precisas en general. Sin embargo, el valor de R^2 es de 0.33302, lo que significa que solo explica el 33.30% de la variabilidad, indicando que el modelo no se ajusta muy bien a los datos. Lo cual sugiere la posibilidad de mejoras para optimizar los resultados. Por otro lado, para este modelo solo se usan los intervalos de confianza de 0.25 y 0.5, los cuales son de

40.909 y 77.402 respectivamente. Indicando que el modelo no puede predecir con gran confianza si una persona padece diabetes (ver Tabla 4.33).

En la Figura 4.35, se presenta una representación gráfica que permite observar de manera visual la diferencia entre las predicciones realizadas por el modelo y la verdadera condición de enfermedad. Se ha optado por un modelo de gráfica que facilita la interpretación de los datos, ofreciendo una visualización clara y comprensible de la información.

Tabla 4.33: Resultados de la inferencia con el conjunto de datos Pima.

Registro	Predichos	Reales	Diferencia	Registro	Predichos	Reales	Diferencia
1	0.770	1	0.230	**11**	0.070	0	0.070
2	0.296	0	0.296	**12**	0.436	0	0.436
3	0.189	0	0.189	**13**	0.170	0	0.170
4	0.288	1	0.712	**14**	0.238	1	0.762
5	0.263	0	0.263	**15**	0.334	0	0.334
6	0.655	1	0.345	**16**	0.373	1	0.627
7	0.137	0	0.137	**17**	0.241	0	0.241
8	0.529	1	0.471	**18**	0.164	0	0.164
9	0.145	0	0.145	**19**	0.289	1	0.711
10	0.641	0	0.641	**20**	0.157	0	0.157

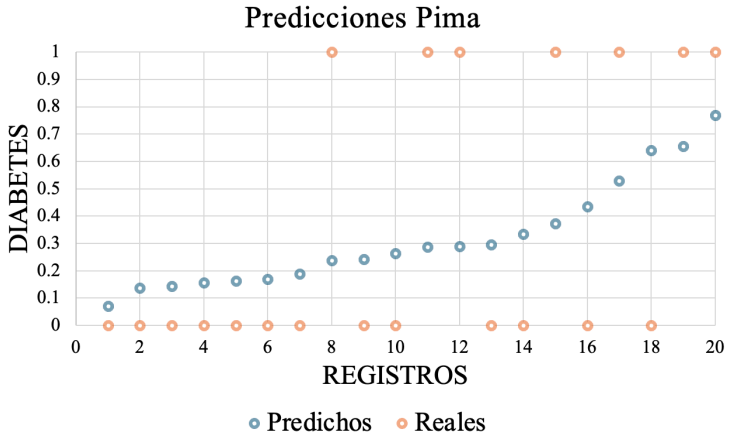


Figura 4.35: Representación gráfica de valores predichos contra valores reales para Pima.

Para una mejor comprensión de las métricas de evaluación, se puede consultar el Anexo A, donde se encuentran detalladas. De la misma forma, los conjuntos de datos usados en los presentes experimentos se encuentran detallados en el Anexo B.

Capítulo 4: Métodos propuestos de representación, interpretación, búsqueda e inferencia

La NN-ACFL utiliza los parámetros w, α, γ y m para definir su arquitectura, los cuales son descubiertos por el algoritmo durante el proceso de entrenamiento. Los parámetros definidos, permiten que el modelo pueda ser interpretado y explicado en lenguaje natural, lo que brinda una comprensión sencilla de su funcionamiento. Para lograr esta expresión, se emplea el algoritmo de cálculo de preimágenes descrito en la sección 4.3.2.

De manera clásica la expresión de los predicados lógicos en lenguaje natural se hace a partir de etiquetas y se describe en el siguiente ejemplo:

> Si los atributos $x = BMI$, $y = Peso$ y $z = Individuo$ y las etiquetas $A = Medio$, $B = Bajo$ y $C = Saludable$. La interpretación en lenguaje natural para un predicado del tipo $If\ x\ is\ A\ and\ y\ is\ B\ Then\ z\ is\ C$ se expresa:
>
> $Si\ BMI\ es\ Medio\ y\ Peso\ es\ Bajo\ Entonces\ Individuo\ is\ Saludable$

En el caso de la expresión de predicados lógicos en lenguaje natural en función de las preimágenes de 0.5 y 1 de una función de membresía definida por una variable lingüística continua generalizada, pueden expresarse como se observa en el siguiente ejemplo:

> Si preimagen de $0.5_1 = 18.5$, preimagen de $0.5_2 = 24.9$ y preimagen de $1.0 = 21.7$ para $GCLV(BMI)$ y preimagen de $0.5_1 = 67$, preimagen de $0.5_2 = 81$ y preimagen de $1.0 = 74$ para $GCLV(Peso)$ se expresaria como:
>
> $Si\ BMI\ esta\ entre\ 18.5\ y\ 24.9\ y\ es\ aproximadamente\ igual\ a\ 21.7\ y\ Peso\ esta\ entre\ 67\ y\ 81\ y\ es\ aproximadamente\ igual\ a\ 74\ Entonces\ Individuo\ es\ Saludable$

Una vez observada la expresión en función de las preimágenes a través del ejemplo anterior, la interpretación del modelo se realizará usando el modelo de clasificación desarrollado para el conjunto de datos Iris. El cual consta de cinco atributos, donde cuatro de ellos se utilizan para predecir la clase de flor iris registrada en la quinta columna. Estos atributos son: longitud del sépalo, ancho del sépalo, longitud del pétalo y ancho del pétalo. Cada uno de los atributos se normaliza en el intervalo [0,1], luego se modela mediante una variable lingüística continua generalizada. Se observa un ejemplo de este proceso en la Figura 4.36.

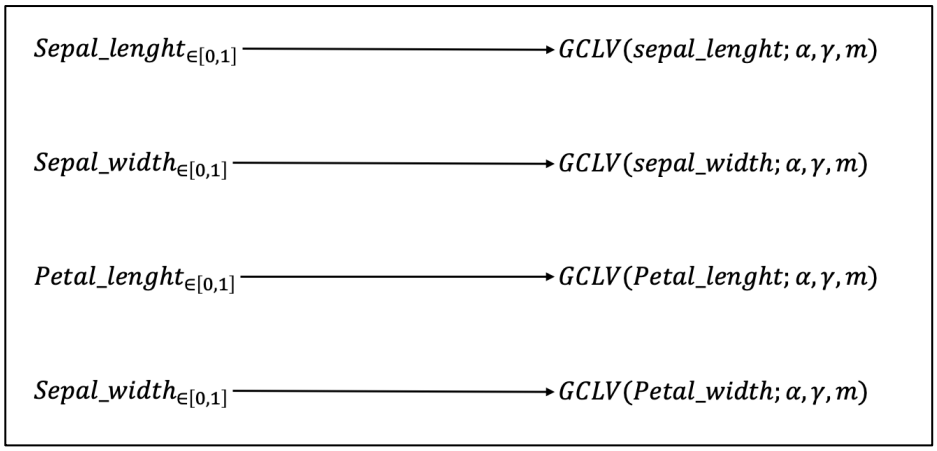


Figura 4.36: Modelado de los atributos, a partir de sus GCLV correspondientes.

Los resultados obtenidos para esta primera capa de GCLV o modelado de datos, se observan en la Tabla 4.34.

Tabla 4.34: Parámetros α, γ y m descubiertas para cada GCLV en la capa de modelado.

GCLV	α	γ	m
Sepal_lenght	8.2354	0.3145	0.9968
Sepal_width	9.7205	0.3673	0.9961
Petal_lenght	10.913	0.3716	0.9488
Petal_width	4.9518	0.5049	0.2411

Entonces, estas GCLV pueden ser interpretadas en lenguaje natural en función de las preimágenes de los valores difusos 0.5, 1 y 1-δ:

$GCLV(sepal\_lenght;\ \alpha = 8.2354, \gamma = 0.3145, m = 0.9968)$: Sepal_lenght esta entre 5.4262 y 7.9 centímetros y debe ser aproximadamente igual a 6.325 centímetros.

$GCLV(sepal\_width;\ \alpha = 9.7205, \gamma = 0.3673, m = 0.9961)$: Sepal_width esta entre 2.8774 y 4.4 centímetros y debe ser aproximadamente igual a 3.35 centímetros.

$GCLV(Petal\_length;\ \alpha = 10.913, \gamma = 0.3716, m = 0.9488)$: Petal_length esta entre 3.1246 y 6.9 centímetros y debe ser aproximadamente igual a 3.95 centímetros.

$GCLV(Petal\_width;\ \alpha = 4.9518, \gamma = 0.5049, m = 0.2411)$: Petal_width esta entre 0.5368 y 2.5 centímetros y debe ser aproximadamente igual a 1.15 centímetros.

Las GCLV definen las funciones de membresía que modelan inicialmente la entrada a la capa de predicados conjuntivos. En esta capa, cada nodo representa un predicado que utiliza las

GCLV iniciales afectadas por un modificador. Esta configuración permite la generación y exploración de múltiples predicados que capturan diferentes aspectos de los datos. La Figura 4.37 proporciona un ejemplo visual de este proceso.

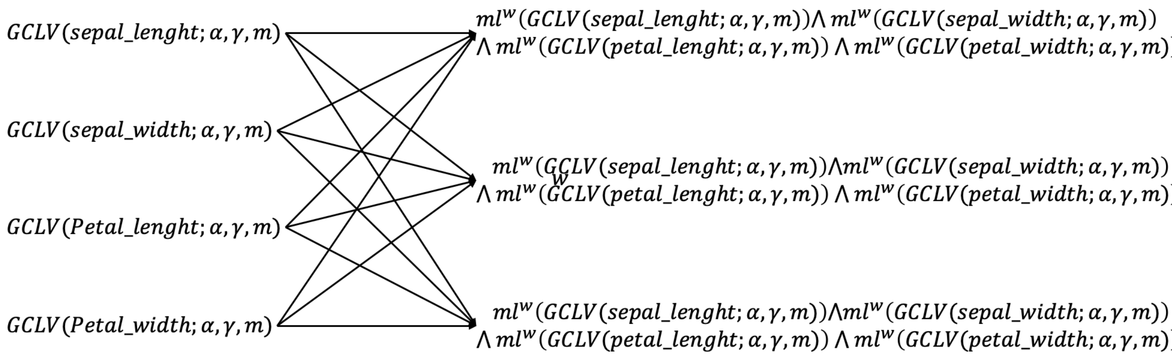


Figura 4.37: Predicados conjuntivos formados a partir de las funciones de membresía de las capas de GCLV.

En la Tabla 4.35 se observan los parámetros *w* descubiertos para cada función de membresía en cada predicado de la capa conjuntiva.

Tabla 4.35: Parámetros *w* descubiertos para la capa conjuntiva.

Atributo	**Predicado 1**	**Predicado 2**	**Predicado 3**	**Predicado 4**	**Predicado 5**
	w	*w*	*w*	*w*	*w*
Sepal_lenght	-3.35e-04	3.18e-05	4.33e-05	6.42e-01	-3.97e-04
Sepal_width	1.14e-04	1.96e-05	-9.40e-05	2.48e-01	2.29e-04
Petal_lenght	-1.22e+00	1.31e+00	1.62e+00	9.00e-01	-7.21e-01
Petal_width	-5.48e-01	2.27e-01	8.20e-01	6.76e-01	-4.54e-01

Atributo	**Predicado 6**	**Predicado 7**	**Predicado 8**
	w	*w*	*w*
Sepal_lenght	1.17e-04	-4.27e-05	-3.16e-04
Sepal_width	9.16e-05	5.64e-05	4.73e-04
Petal_lenght	1.53e+00	1.72e+00	-1.20e+00
Petal_width	4.54e-01	9.29e-01	-5.58e-01

Dada la información de la Tabla 4.35, se puede expresar cada predicado de la siguiente forma:

Para el Predicado 1 correspondiente al Nodo 1 de la capa conjuntiva $N_{1,c}$ se podria expresar de manera aritmética de la siguiente manera:

$$p1 = ml^{-3.35E^{-4}}\left(GCLV\left(sepal_{length}\right)\right) \wedge ml^{1.14E^{-4}}\left(GCLV(sepal_{width})\right)$$

$\wedge ml^{-1.22E^{00}}\left(GCLV(Petal\_length)\right)\wedge ml^{-5.48E^{-1}}\left(GCLV(sepal\_length)\right)$

Para el predicado 2:

$p1 = ml^{3.18E^{-5}}\left(GCLV\left(sepal_{length}\right)\right)\wedge ml^{1.96E^{-5}}\left(GCLV(sepal_{width})\right)$
$\wedge ml^{1.31E^{00}}\left(GCLV(Petal\_length)\right)\wedge ml^{2.27E^{-1}}\left(GCLV(sepal\_length)\right)$

Y así para cada predicado conjuntivo hasta llegar al predicado 8 del $N_{8,c}$.

$p8 = ml^{-3.16E^{-4}}\left(GCLV\left(sepal_{length}\right)\right)\wedge ml^{1.96E^{-5}}\left(GCLV(sepal_{width})\right)$
$\wedge ml^{-1.20E^{00}}\left(GCLV(Petal\_length)\right)\wedge ml^{-5.58E^{-1}}\left(GCLV(sepal\_length)\right)$

los cuales a través del cálculo de sus preimágenes pueden ser expresados de la siguiente manera:

P1: Si Sepal Length está entre 4.3 y 6.16 centímetros O 6.26 y 7.9 centímetros **Y** Sepal Width es menor que 4.15 y se aproxima a 3.2 centímetros **Y** Petal Length está entre 1.0 y 3.08 centímetros O 5.98 y 6.9 centímetros **Y** Petal Width está entre 0.1 y 0.77 centímetros O 1.398 y 2.5 centímetros.

P2: Si Sepal Length es menor que 7.69 y se aproxima a 6.1centimetros **Y** Sepal Width es menor que 4.15 y se aproxima a 3.2 centímetros **Y** Petal Length está entre 3.17 y 5.58 y se aproxima a 3.65 centímetros **Y** Petal Width es menor que 2.025 y se aproxima a 1.15 centímetros.

P3: Si Sepal Length es menor que 7.69 y se aproxima a 6.1centimetros **Y** Sepal Width está entre 2.0 y 4.4 y es aproximadamente igual a 3.29 centímetros **Y** Petal Length está entre 3.21 y 5.26 y se aproxima a 3.65 centímetros **Y** Petal Width está entre 0.44 y 1.55 y se aproxima a 1.15 centímetros.

P4: Si Sepal Length está entre 5.33 y 7.69 y se aproxima a 6.1centimetros **Y** Sepal Width está entre 2.66 y 4.15 y se aproxima a 3.35 centímetros **Y** Petal Length está entre 3.101 y 5.98 y se aproxima a 3.67 centímetros **Y** Petal Width está entre 0.32 y 1.59 y se aproxima a 1.15 centímetros

P5: Si Sepal length está entre 4.3 y 7.9 centímetros y es aproximadamente igual a 5.2 centímetros **Y** si Sepal width está entre 2 y 4.4 centímetros y es aproximadamente igual a 3.2 **Y** si Petal length está entre 1 y 5.438 centímetros y es aproximadamente igual a 1.7375 centímetros **Y** si Petal length esta mayor a 1.3633 y es aproximadamente igual a 2.5 centímetros.

P6: Si Sepal length está entre 4.3 y 7.9 centímetros y es aproximadamente igual a 6.1 **Y** si sepal width está entre 2 y 4.4 centímetros y es aproximadamente igual a 3.2 centímetros **Y** si

Petal length está entre 3.2067 y 6.9 centímetros y es aproximadamente igual a 3.95 **Y** si Petal width está entre 0.1 y 2.5 centímetros y es aproximadamente igual a 1.15.

P7: Si Sepal length está entre 4.3 y 7.9 centímetros y es aproximadamente igual a 6.1 centímetros **Y** si sepal width está entre 2.0 y 4.4 centímetros y es aproximadamente igual a 3.2 centímetros **Y** si Petal length está entre 3.2276 y 6.9 centímetros y es aproximadamente igual a 3.95 centímetros **Y** si petal width está entre 0.504 y 2.5 centímetros y es aproximadamente igual a 1.15 centímetros.

P8: Si Sepal Length está entre 4.3 y 6.16 O 6.26 y 7.9 centímetros **Y** Sepal Width es menor que 4.15 y se aproxima a 3.2 centímetros **Y** Petal Length está entre 1.0 centímetros y 3.08 O 5.98 y 6.9 centímetros **Y** Petal Width está entre 0.1 y 0.77 O 1.40 y 2.5 centímetros.

Los predicados presentados no son del todo correctos, ya que, debido a la morfología definida a partir de las configuraciones para cada función, el algoritmo no siempre alcanza las preimágenes de 0.5 y 1.0 de manera exacta, sin embargo, son aproximaciones muy cercanas. Es importante tenerlo en cuenta para asegurar una representación precisa y adecuada de los predicados en el modelo

Después de este proceso se pasa a una última capa, la cual es una capa disyuntiva, en esta capa todos los predicados generados en la capa conjuntiva son tratados a partir de un operador de disyunción un ejemplo se observa en la Figura 4.38.

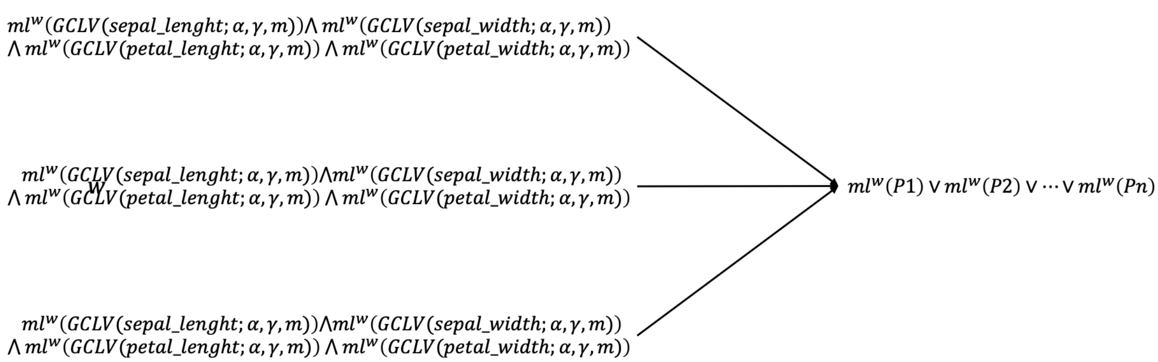


Figura 4.38: Predicado disyuntivo formado a partir de los predicados de la capa conjuntiva.

En la Tabla 4.36 se presentan los parámetros *w* descubiertos para cada predicado definido en la capa conjuntiva.

Tabla 4.36: Parámetros *w* descubiertos para el predicado disyuntivo.

	Predicado 1	**Predicado 2**	**Predicado 3**	**Predicado 4**	**Predicado 5**
w	-1.7704	1.5369	1.2044	0.9771	-1.4072
	Predicado 6	**Predicado 7**	**Predicado 8**		
w	1.6617	1.1849	-1.3176		

Este predicado disyuntivo puede ser expresado de forma aritmética de la siguiente manera:

$PD = ml^{-1.7704}(P1) \vee ml^{1.5369}(P2) \vee ml^{1.2044}(P3) \vee ml^{0.9771}(P4) \vee ml^{-1.4072}(P5) \vee ml^{1.6617}(P6) \vee ml^{1.1849}(P7) \vee ml^{-1.3176}(P8)$

El cual si puede expresar de la siguiente forma: Si P1 intensificado a la -1.7704 o P2 intensificado a la 1.5369 o P3 intensificado a la 1.2044 o P4 intensificado a la 0,9771 o P5 intensificado a la -1.4072 o P6 intensificado a la 1.6617 o P7 intensificado a la 1.1849 o P8 intensificado a la -1.3176.

El conocimiento adquirido a través del modelo de una NN-ACFL permite abordar problemas de clasificación y estimación al descubrir una serie de parámetros modelados a partir de una estructura que representa un predicado en forma normal disyuntiva, como se muestra en la Figura 4.39. Además, mediante la evaluación de las funciones de membresía generadas y la definición de sus preimágenes, es posible expresar este conocimiento de manera interpretable en lenguaje natural. Este enfoque no solo proporciona predicciones precisas, sino que también permite comprender y explicar los resultados del modelo de una manera comprensible y significativa.

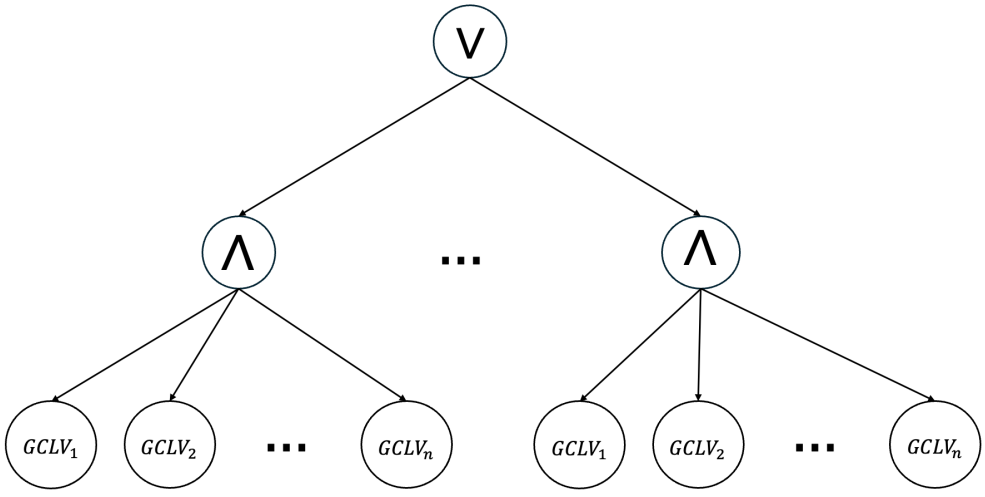


Figura 4.39: Predicado en forma normal disyuntiva (FND) representado en la estructura de una NN-ACFL.

El modelo que representa la red neuronal de lógica difusa Arquimediana compensatoria NN-ACFL, al integrar los predicados conjuntivos en el predicado disyuntivo, puede expresarse a través del siguiente predicado en FND:

Si Sepal Length está entre 4.3 y 6.16 centímetros O 6.26 y 7.9 centímetros **Y** Sepal Width es menor que 4.15 y se aproxima a 3.2 centímetros **Y** Petal Length está entre 1.0 y 3.08 centímetros O 5.98 y 6.9 centímetros **Y** Petal Width está entre 0.1 y 0.77 centímetros O 1.398 y 2.5 centímetros. intensificado a la -1.7704.

O Sepal Length es menor que 7.69 y se aproxima a 6.1centimetros **Y** Sepal Width es menor que 4.15 y se aproxima a 3.2 centímetros **Y** Petal Length está entre 3.17 y 5.58 y se aproxima a 3.65 centímetros **Y** Petal Width es menor que 2.025 y se aproxima a 1.15 centímetros. intensificado a la 1.5369

O Sepal Length es menor que 7.69 y se aproxima a 6.1centimetros **Y** Sepal Width está entre 2.0 y 4.4 y es aproximadamente igual a 3.29 centímetros **Y** Petal Length está entre 3.21 y 5.26 y se aproxima a 3.65 centímetros **Y** Petal Width está entre 0.44 y 1.55 y se aproxima a 1.15 centímetros. intensificado a la 1.2044.

O Sepal Length está entre 5.33 y 7.69 y se aproxima a 6.1centimetros **Y** Sepal Width está entre 2.66 y 4.15 y se aproxima a 3.35 centímetros **Y** Petal Length está entre 3.101 y 5.98 y se aproxima a 3.67 centímetros **Y** Petal Width está entre 0.32 y 1.59 y se aproxima a 1.15 centímetros. intensificado a la 0,977.

O Sepal length está entre 4.3 y 7.9 centímetros y es aproximadamente igual a 5.2 centímetros **Y** si Sepal width está entre 2 y 4.4 centímetros y es aproximadamente igual a 3.2 **Y** si Petal length está entre 1 y 5.438 centímetros y es aproximadamente igual a 1.7375 centímetros **Y** si Petal length esta mayor a 1.3633 y es aproximadamente igual a 2.5 centímetros. intensificado a la -1.4072.

O Sepal length está entre 4.3 y 7.9 centímetros y es aproximadamente igual a 6.1 **Y** si sepal width está entre 2 y 4.4 centímetros y es aproximadamente igual a 3.2 centímetros **Y** si Petal length está entre 3.2067 y 6.9 centímetros y es aproximadamente igual a 3.95 **Y** si Petal width está entre 0.1 y 2.5 centímetros y es aproximadamente igual a 1.15. intensificado a la 1.6617

O Sepal length está entre 4.3 y 7.9 centímetros y es aproximadamente igual a 6.1 centímetros **Y** si sepal width está entre 2.0 y 4.4 centímetros y es aproximadamente igual a 3.2 centímetros **Y** si Petal length está entre 3.2276 y 6.9 centímetros y es aproximadamente igual a 3.95 centímetros **Y** si petal width está entre 0.504 y 2.5 centímetros y es aproximadamente igual a 1.15 centímetros. intensificado a la 1.1849

O Sepal Length está entre 4.3 y 6.16 O 6.26 y 7.9 centímetros **Y** Sepal Width es menor que 4.15 y se aproxima a 3.2 centímetros **Y** Petal Length está entre 1.0 centímetros y 3.08 O 5.98 y 6.9 centímetros **Y** Petal Width está entre 0.1 y 0.77 O 1.40 y 2.5 centímetros. intensificado a la -1.3176.

Entonces, este predicado nos permite modelar una función que busca igualar a una función de clases y, a través de esta, inferir la clase o estimar el valor de la función con los valores reales. Este enfoque es fundamental para la capacidad predictiva y clasificatoria del modelo, ya que permite relacionar las características de entrada con las clases objetivo o los valores de la función que se desea predecir. En la Figura 4.40 se observa un ejemplo de este proceso.

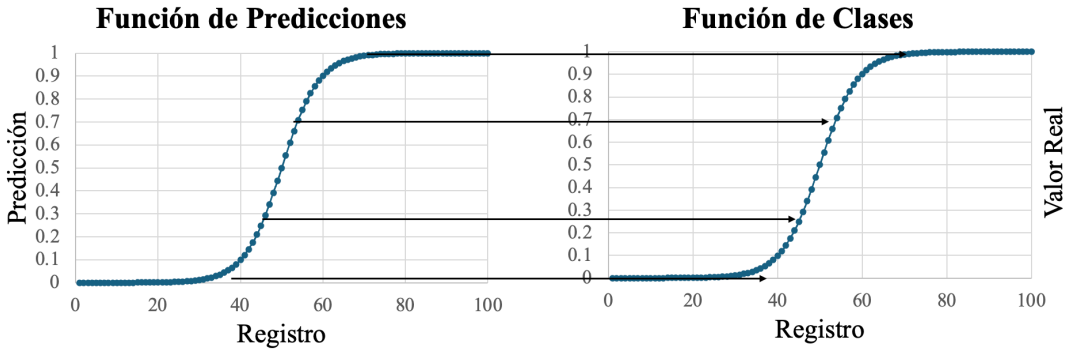


Figura 4.40: Función de predicción que asemeja la función de clases.

Los resultados de los parámetros descubiertos en los procesos de entrenamiento y evaluación de una NN_ACFL aplicada a los conjuntos de datos Daily Order, Dow Jones y Pima, se muestran en el Anexo C.

En la Tabla 4.37, se comparan los resultados de la NN-ACFL con los resultados encontrados en la literatura en función de su precisión.

Tabla 4.37: Comparación de los resultados de una NN-ACFL con los resultados publicados en la literatura.

Conjunto de datos	NN-ACFL	Literatura	
Daily Orders	66.0	-	
Dow Jones	82.25	84.2	(Livieris et al., 2018)
Iris	96.33	97	(Yang & Zhao, 2023)
Pima	77.4	74.54	(Yang & Zhao, 2023)

Como se puede observar, en el caso de Daily Orders, no fue posible encontrar un artículo con el cual comparar los resultados. Debido a que los documentos encontrados miden su asertividad en función de métricas y mediciones diferentes a las usadas en este documento. Además, al comparar los resultados de la NN-ACFL con los encontrados en Livieris et al. (2018) y Yang & Zhao (2023). Se puede apreciar que los resultados obtenidos están cerca de los reportados en estas revistas de renombre, lo cual indica resultados competitivos. Sin embargo, es necesario tener en cuenta que, aunque se evalúe en función de la precisión reportada, no necesariamente coinciden en los métodos de evaluación.

Por otro lado, en cuanto a la interpretabilidad, se presenta un nuevo modelo de interpretación de los resultados, el cual ofrece una forma innovadora no solo de entender los resultados de una ACFL, que como característica de las lógicas difusas es inherentemente interpretable, sino también de expresar el conocimiento en lenguaje natural. Este proceso es altamente

valorado en el campo de la analítica de datos, ya que no solo presenta los resultados, sino que parte de una justificación basada en datos y es expresable a partir de los mismos.

Un ejemplo de lo expresado es que en artículos como en Hong Lan et al. (2020), Vivek et al. (2020), Nguyen Thu Hien et al. (2022). La interpretación se realiza de manera clásica, es decir, a través de una serie de etiquetas definidas para cada función de membresía. Esta es una forma bastante sencilla, simple y, a su vez, comprensible de interpretar los resultados. Sin embargo, en muchos casos, puede ser una interpretación subjetiva, ya que la comprensión puede variar según el observador y el entorno.

Por otro lado, artículos como los de De Campos Souza (2020) y Xie et al. (2021), presentan modelos interpretables basados en reglas semánticas y la aplicación de conceptos extra lógicos, que, si bien permiten obtener resultados interpretables y expresables, aún hay conceptos no completamente claros en su interpretación.
Por ejemplo, en Vivek et al. (2020), se utilizan predicados del tipo $Si\ x\ es\ A\ Y\ y\ es\ B, entonces\ z\ es\ C$. Donde A, B y C son etiquetas definidas, como, por ejemplo, A puede ser alto, medio o bajo valor del atributo A. De esta manera, la expresión del predicado podría ser la siguiente: $Si\ x\ es\ Alto\ Y\ y\ es\ Bajo, entonces\ z\ es\ Alto$. El problema con esta expresión es su subjetividad. Aunque es expresable en lenguaje natural, la expresión sigue sin ser completamente clara y, en muchos casos, solo es entendible por usuarios expertos en el tema.

Por otro lado, de Campos Souza (2020), propone una red neuronal que es capaz de generar predicados del tipo AND y OR o una mezcla entre ambos, también se hace uso de la definición de etiquetas asociadas a cada atributo de entrada. En esta red, las neuronas, al verse afectadas por un factor de peso *w*, pueden expresarse de la siguiente forma:

$$Regla: Si\ x_1\ es\ A_1\ con\ un\ impacto\ de\ w_1\ y/o\ x_2\ es\ A_2 \ldots entonces\ y\ es\ [c_1, \ldots c_n]$$

Donde:
c = clase.

Esto permite incluir la forma en que los pesos afectan a cada una de las neuronas. Sin embargo, esta expresión sigue siendo en cierta medida poco clara.

4.3.3 Resumen de experimentos con NN-GCLV y NN-ACFL

Los resultados de los experimentos realizados a través de las redes neuronales presentadas en las secciones 4.3.1 y 4.3.2 permiten evaluar el desempeño del algoritmo propuesto en esta tesis. Asimismo, facilitan una comparación objetiva de los resultados obtenidos con aquellos reportados en la literatura especializada, en función de las métricas de evaluación presentadas.

En la Tabla 4.38, se presentan los siguientes resultados.

Tabla 4.38: Resultados observados en NN-GCLV, NN-ACFL y la literatura especializada.

Conjunto de datos	Modelo	Indicador	Resultado
Clasificación			
Iris	PMNN	Exactitud	98.2
	NN-GCLV	Exactitud	1.0
	NN-ACFL	Exactitud	96.33
Pima	PMNN	Exactitud	74.54
	NN-GCLV	Exactitud	95.83
	NN-ACFL	Exactitud	77.4
Estimación			
Daily orders	MLPNN	R cuadrada	99.2
	NN-GCLV	R cuadrada	99.48
	NN-ACFL	R cuadrada	86.02
Dow jones	LMT	Exactitud	84.2
	NN-GCLV	Exactitud	87.06
	NN-ACFL	Exactitud	82.25

Los resultados observados comparan el algoritmo PMNN reportado por Yang y Zhao (2023). En ambos casos, la red neuronal NN-GCLV, mencionada en la sección 4.3.1, supera los resultados presentados por PMNN en el proceso de entrenamiento con los conjuntos de datos Iris y Pima. Por otro lado, la NN-ACFL solo supera a PMNN en el conjunto de datos Pima, por un pequeño margen de diferencia. Por lo tanto, se puede concluir que ambas redes presentan resultados competitivos al ser comparadas con resultados presentados en artículos del estado del arte en procesos de clasificación.

También se pueden comparar los resultados de los algoritmos MLPNN y LMT, reportados por Gündogdu (2022) y Livieris et al. (2018), respectivamente. En los artículos mencionados, se busca solucionar el problema de lograr estimaciones correctas para los conjuntos de datos Daily Orders y Dow Jones, cuyo valor objetivo pertenece al orden de los números reales. En este caso, se usa R cuadrada como métrica de evaluación comparativa. Se puede observar que NN-GCLV supera el valor reportado, mientras que NN-ACFL queda significativamente por debajo del valor reportado por el algoritmo MLPNN. En el caso del conjunto de datos Dow Jones, se reporta la exactitud como la diferencia entre un 20% del valor real y el valor predicho. De la misma forma, se reportan los valores de exactitud de NN-GCLV y NN-ACFL, donde NN-GCLV supera el valor reportado y NN-ACFL presenta una diferencia poco significativa. Cabe aclarar que, en los casos de estimación, se tuvieron que realizar experimentos que se asemejaran lo más posible a aquellos reportados en los artículos de referencia.

Si bien es cierto que en estos casos se observan mejores resultados en los algoritmos presentados en la literatura y en la NN-GCLV, la cual incluso supera los resultados reportados, la NN-ACFL no solo ofrece resultados competitivos, sino que además presenta la capacidad de ser interpretada y expresada en lenguaje natural, una capacidad que ninguno de los otros algoritmos posee.

Capítulo 5: Conclusiones y trabajos futuros

En esta tesis se presentó una serie de enfoques propuestos para llevar a cabo procesos de analítica de datos basados en métodos de representación, interpretación y búsqueda de predicados de lógica difusa Arquimediana compensatoria.

Estos enfoques permitieron definir un modelo de red neuronal híbrido que incorpora una arquitectura de red de propagación hacia adelante junto con conceptos de lógica difusa Arquimediana compensatoria. Esta combinación posibilitó la realización de procesos de clasificación e inferencia de datos con alta precisión en las predicciones, además de proporcionar un enfoque interpretable y explicable en lenguaje natural.

El modelo de red neuronal híbrido resultante permitió el aprendizaje basado en datos, un enfoque fundamental en el campo de la analítica de datos. Además, en la sección 5.2 de este capítulo, se presentan diferentes trabajos futuros que podrían explorarse, dado que el presente modelo ofrece una amplia gama de posibilidades y aplicaciones.

5.1 Conclusiones

La analítica de datos es actualmente un enfoque altamente valorado y aplicado en la resolución de problemas en diversas áreas del desarrollo humano. Se han desarrollado numerosos modelos de analítica de datos destinados a resolver problemas en medicina, ingeniería, negocios, entre otras áreas.

La principal razón de su implementación radica en que, a través del análisis de datos, es posible obtener la capacidad para extraer conocimientos, identificar patrones y tomar decisiones basadas en datos. Esta capacidad permite, entre otras cosas, optimizar procesos, es decir, identificar áreas de mejora dentro de los procesos, lo que resulta en una mejora de la eficiencia operativa. También facilita la toma de decisiones informadas y estratégicas, ya que se basa en el análisis riguroso de datos concretos. A través de la información encontrada, se pueden identificar tendencias y oportunidades que de otro modo pasarían desapercibidas. Además, permite anticiparse a problemas futuros, lo cual posibilita prepararse y tomar medidas preventivas.

Específicamente para este trabajo, se han utilizado redes neuronales que implementan enfoques de lógica difusa. El uso de este modelo de redes neuronales difusas aporta a los procesos de analítica de datos la capacidad de modelar y manejar la incertidumbre de manera más efectiva que las redes neuronales tradicionales. Además, permite encontrar relaciones entre las variables de entrada y salida de manera flexible y expresiva, así como la integración

del conocimiento de expertos en el proceso de modelado. Al igual que las redes neuronales clásicas, presenta una gran capacidad de adaptación a los cambios en los datos de entrada.

Sin embargo, las redes neuronales difusas presentan una serie de dificultades. La combinación de los conceptos de redes neuronales y lógica difusa resulta en un modelo con una gran dificultad conceptual, lo que dificulta entender cómo funcionan y se ajustan sus parámetros. Además, al igual que con las redes clásicas, la capacidad de interpretación del modelo neuro difuso se vuelve complicada debido a su naturaleza no lineal y la presencia de funciones de membresía difusas. Los modelos neuro difusos suelen presentar en su arquitectura una gran cantidad de parámetros que necesitan ser ajustados durante el proceso de entrenamiento del modelo. Descubrir la mejor combinación suele requerir técnicas avanzadas de optimización y un dominio del problema. Por último, su implementación suele requerir conocimientos especializados para su implementación y entrenamiento eficaz.

En esta tesis se ha desarrollado un modelo híbrido de red neuronal basado en lógica difusa Arquimediana compensatoria, denominado NN-ACFL, que incorpora estrategias de optimización para la representación, búsqueda y uso de predicados en modelos de inferencia. Este modelo ha demostrado resultados competitivos en términos de exhaustividad y diversidad en la búsqueda de predicados, así como en exactitud e interpretabilidad en los procesos de inferencia.

Entre los principales modelos de redes neuronales difusas encontrados en la literatura especializada se encuentran las redes neuronales difusas adaptativas (ANFIS), las cuales modelan sistemas no lineales para tareas de clasificación o predicción. Las redes neuronales basadas en reglas difusas (RBFN) combinan funciones de base radial y reglas difusas para modelar relaciones entre variables de entrada y salida. Por otro lado, las redes neuronales difusas evolutivas (EFuNN) implementan modelos evolutivos para optimizar la arquitectura y los parámetros de la red.

Cada uno de estos modelos presenta una arquitectura compleja y enfrenta dificultades en el proceso de ajuste de parámetros, lo que dificulta su interpretación debido a la falta de transparencia en su estructura interna. Las ANFIS requieren una gran cantidad de datos para evitar el sobreajuste, y la complejidad de sus reglas difusas dificulta su interpretación. En las RBFN, la determinación del número y la ubicación de las funciones de base radial, así como la selección de los parámetros asociados, presenta dificultades. Las EFuNN dependen en gran medida de la configuración de los parámetros de los algoritmos evolutivos, lo que requiere ajustes manuales.

Dadas las características observadas en los modelos descritos dentro de la literatura especializada, se decidió implementar un nuevo modelo de red neuronal híbrida de lógica difusa Arquimediana compensatoria (ACFL), denominada NN-ACFL. A través de este modelo, se busca abordar problemas de analítica de datos, como clasificación e inferencia, utilizando una serie de métodos que permiten representar, interpretar y descubrir predicados de lógica difusa para resolver dichos problemas.

Con este fin, se plantearon una serie de enfoques que condujeron a la definición del modelo propuesto. Este modelo permite representar predicados a partir de la definición de la

arquitectura de la red, evitando el uso de recursos extra lógicos, lo cual lo convierte en un modelo interpretable en función de los conceptos de una ACFL, haciéndolo transparente y explicable.

En la sección 4.1 se propuso una teoría lógica denominada lógica difusa Arquimediana compensatoria (ACFL), cuyos conceptos son generalizables a partir de una función generadora. Principalmente introduce una variable lingüística continua generalizada, la cual permite la generación de una familia de funciones de membresía cuya morfología es definida en función de sus parámetros, y un modificador lingüístico, el cual, en vez de servir como un potenciador, se define como un recurso lógico que permite intensificar la función de membresía, pero aun así es interpretable y explicable.

Para probar las características de esta ACFL, se implementó un algoritmo genético que permite generar y descubrir predicados de ACFL exhaustiva y diversamente que permitieran solucionar problemas de clasificación. Se logró comprobar que la ACFL es una teoría lógica pluralista contextual que permite descubrir predicados variando la lógica usada y también a partir de las GCLV, adaptándose a la mejor configuración de los datos, modificando de esta manera la clásica definición de funciones de membresía a partir de etiquetas e introduciendo una interpretación en función de los parámetros de una GCLV. Asimismo, los predicados descubiertos permiten solucionar problemas relacionados con la analítica de datos, entre ellos la analítica descriptiva y el diagnóstico de datos.

Por otro lado, en la sección 4.2, se introduce un modelo de interpretación de predicados a partir de la caracterización de una variable lingüística continua generalizada (GCLV). En este modelo, se sustituye la clásica interpretación de las funciones de membresía a través de etiquetas lingüísticas, las cuales suelen ser subjetivas, por una interpretación basada en las preimágenes de los valores difusos 0.5, 1.0 y 1-δ.

Como resultado, se obtiene un modelo que puede interpretar las diversas representaciones de un predicado de ACFL en función de valores objetivos, como las preimágenes de una GCLV, permitiendo expresar el modelo en un lenguaje natural comprensible para el usuario. De esta manera, no solo se obtienen altos niveles de certidumbre en procesos de analítica de datos, sino que también es posible justificar los resultados a partir de las interpretaciones semánticas expresadas en lenguaje natural.

A partir de la integración de los conceptos mencionados, se propone una red neuronal híbrida que introduce un modelo de red neuronal que utiliza una arquitectura de propagación hacia adelante con las características de una ACFL. En esta propuesta, se introduce el proceso de descubrimiento de conocimiento a través de un predicado en forma normal disyuntiva, representado a través de la arquitectura de una red neuronal de ACFL. Este modelo sigue un enfoque puramente lógico, evitando el uso de recursos ajenos a la lógica en su construcción.

A través de esta estructura, se permite generar predicados de forma exhaustiva y diversa, los cuales pueden ser aplicados a procesos de analítica de datos, obteniendo muy buenos resultados en términos de precisión e interpretación.

Capítulo 5: Conclusiones y trabajos futuros

De acuerdo con los resultados presentados, se puede observar que, en términos de precisión, el modelo alcanza niveles competitivos al compararlos con otros modelos de clasificación e inferencia reportados en revistas científicas de renombre. Además, los valores de parámetros descubiertos permiten aplicar el proceso de caracterización de una GCLV, lo cual permite expresar de manera clara, concisa y objetiva los resultados en lenguaje natural. Esto aporta una capacidad de análisis de las relaciones entre las características que componen un conjunto de datos no vista en otras publicaciones observadas.

A través de este modelo se obtiene conocimiento que permite justificar los resultados explicados a partir de información objetiva. Además, permite establecer las relaciones entre las características de entrada y la salida, permitiendo modelos precisos e interpretables.
Por lo tanto, se puede decir que las principales aportaciones y ventajas del modelo de red neuronal híbrida de lógica difusa Arquimediana compensatoria implementada con una arquitectura de propagación hacia adelante (NN-ACFL) son:

a) **Interpretabilidad:** El modelo es capaz de generar reglas interpretables a partir de los datos. La inclusión de ACFL permite expresar el conocimiento de manera natural y comprensible, lo que facilita la interpretación de los resultados y la toma de decisiones basadas en ellos, aspecto fundamental en la analítica de datos para comprender el significado detrás de los descubrimientos.

b) **Precisión en la predicción:** A diferencia de muchos modelos interpretables que pueden sacrificar precisión, la NN-ACFL mantiene altos niveles de precisión en la predicción. La arquitectura Feed-forward y el uso de técnicas de redes neuronales permiten capturar relaciones complejas entre variables y realizar predicciones precisas incluso en conjuntos de datos difíciles, lo que es crucial para obtener resultados confiables en análisis de datos de gran escala.

c) **Flexibilidad y adaptabilidad:** La NN-ACFL es altamente adaptable y puede aplicarse a una variedad de problemas de clasificación e inferencia en diferentes dominios de la analítica de datos. La combinación de ACFL y redes neuronales brinda flexibilidad para modelar relaciones no lineales y adaptarse a diferentes tipos de datos y contextos, lo que la hace una herramienta adecuada para solucionar una gran variedad de problemas de analítica de datos.

d) **Capacidad de aprendizaje:** La capacidad de aprendizaje basada en datos de la NN-ACFL le permite mejorar con el tiempo a medida que se alimenta con más información. Lo cual significa que puede adaptarse a cambios en los datos y continuar brindando predicciones precisas incluso en entornos dinámicos, lo cual es esencial para mantener la relevancia y eficacia del análisis de datos en un entorno en constante cambio.

e) **Potencial para la interpretación semántica:** Al utilizar reglas semánticas y evitar el uso de conceptos extra lógicos, la NN-ACFL puede ofrecer una interpretación semántica profunda de los resultados. Permitiendo a los usuarios a comprender las relaciones entre las variables y a obtener información valiosa sobre el dominio del

problema, lo que contribuye a una comprensión más completa y significativa de los datos analizados en el contexto de la analítica de datos.

5.2 Trabajos futuros

Los resultados obtenidos a través del modelo propuesto ciertamente ofrecen una gran ventaja y demuestran ser competitivos y novedosos. Sin embargo, aún existe la posibilidad de implementar nuevos conceptos que permitan desarrollar conocimiento a partir de la estructura ya analizada. Entre estas posibles modificaciones propuestas para su estudio, se proponen las siguientes:

a) **Análisis de funciones generadoras:** En el presente trabajo, solo se trabajó con una función logarítmica exponencial. Sería interesante observar cómo diferentes funciones generadoras de lógicas de ACFL impactan en los procesos de entrenamiento e inferencia en una NN-ACFL.
b) **Métodos de optimización:** Si bien el uso del algoritmo ADAM en el proceso de optimización permitió alcanzar resultados competitivos en función de la precisión de la predicción, se podrían estudiar un mayor número de métodos de optimización de parámetros de una NN-ACFL.
c) **Optimización de la lógica:** En el presente modelo, se definió la lógica usada en el modelo de evaluación, fijando los parámetros de esta. El análisis de la optimización de los parámetros de la lógica puede generar una ventaja significativa en el ajuste del modelo.
d) **Funciones de costo lógicas:** Para este modelo, se implementó como función de costo el error cuadrático medio, que, aunque ofrece buenos resultados, sería interesante evaluar el comportamiento de la red a través del uso de funciones de costo como la condición suficiente, la condición necesaria y la condición suficiente y necesaria.
e) **Estructuras de predicados:** Si bien la arquitectura de la red permite obtener un predicado en forma normal disyuntiva, podría generarse arquitecturas que simulen la forma normal conjuntiva y estructuras deductivas tales como modus ponens, modus tollens, entre otras.
f) **Evaluación del modelo:** Evaluar el alcance de la propuesta con un estudio experimental extenso, representativo de condiciones retadoras y validado estadísticamente con diversos indicadores de desempeño.

Bibliografía

Ahn, S., Couture, S. V., Cuzzocrea, A., Dam, K., Grasso, G. M., Leung, C. K., McCormick, K. L., & Wodi, B. H. (2019). A Fuzzy Logic Based Machine Learning Tool for Supporting Big Data Business Analytics in Complex Artificial Intelligence Environments. *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1–6. https://doi.org/10.1109/FUZZ-IEEE.2019.8858791

Andrade Tepán, E. C. (2013). *Estudio de los principales tipos de redes neuronales y las herramientas para su aplicación* [Universidad politecnica salesiana]. http://dspace.ups.edu.ec/handle/123456789/4098

Apicella, A., Donnarumma, F., Isgrò, F., & Prevete, R. (2021). A survey on modern trainable activation functions. *Neural Networks*, *138*, 14–32. https://doi.org/10.1016/j.neunet.2021.01.026

Barraza, J., Melin, P., Valdez, F., & Gonzalez, C. I. (2023). Modeling of Fuzzy Systems Based on the Competitive Neural Network. *Applied Sciences*, *13*(24), 13091. https://doi.org/10.3390/app132413091

Becerra Contreras, M. H. (2017). *Un sistema de recomendación basado en factorización de matrices y descenso en gradiente estocastico*. Instituto Tecnológico Autónomo de México.

Bělohlávek, R., Dauben, J. W., & Klir, G. J. (2017). Fuzzy logic and mathematics: A historical perspective. En *Fuzzy Logic and Mathematics: A Historical Perspective*. https://doi.org/10.1093/oso/9780190200015.001.0001

Ben Seghier, M. E. A., Carvalho, H., Keshtegar, B., Correia, J. A. F. O., & Berto, F. (2020). Novel hybridized adaptive neuro-fuzzy inference system models based particle swarm optimization and genetic algorithms for accurate prediction of stress intensity factor. *Fatigue and Fracture of Engineering Materials and Structures*. https://doi.org/10.1111/ffe.13325

Camargo-Vega, J. Jose., Camargo-Ortega, J. F., & Joyanes-Aguilar, Luis. (2015). Conociendo Big Data. *Revista de la Facultad de ingenieria*, *24*(38), 63–77. https://www.redalyc.org/articulo.oa?id=413940775006

Das, R., Sen, S., & Maulik, U. (2020). A Survey on Fuzzy Deep Neural Networks. *ACM Computing Surveys*, *53*(3). https://doi.org/10.1145/3369798

de Campos Souza, P. V. (2020). Fuzzy neural networks and neuro-fuzzy networks: A review the main techniques and applications used in the literature. *Applied Soft Computing*, *92*, 106275. https://doi.org/10.1016/j.asoc.2020.106275

de Campos Souza, P. V., & Lughofer, E. (2021). An evolving neuro-fuzzy system based on uni-nullneurons with advanced interpretability capabilities. *Neurocomputing*, *451*, 231–251. https://doi.org/10.1016/j.neucom.2021.04.065

Detyniecki, M. (2001). Fundamentals on aggregation operators. *This manuscript is based on Detyniecki's doctoral thesis*, 39. http://www.cs.berkeley.edu/~marcin/agop.pdf

Diazaraque, J. Marín. (2009). *Data mining*. http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/introduccion-DM.pdf

DOMO. (2022). *data never slepps*. https://www.domo.com/es/data-never-sleeps.

Espín Andrade, R. a., Fernándes, E., & Gonzaléz, E. (2011). Un Sistema Lógico Para El Razonamiento Y La Toma De Decisiones : La Lógica Difusa Compensatoria Basada En La Media Geométrica. *Revista Investigación Operacional*, *32*(3), 230–245.

Espín-Andrade, R. A., Cruz-Reyes, L., Llorente-Peralta, C., González-Caballero, E., Pedrycz, W., & Ruiz, S. (2021). Archimedean Compensatory Fuzzy Logic as a Pluralist Contextual Theory Useful for Knowledge Discovery. *International Journal of Fuzzy Systems*, 1–21. https://doi.org/10.1007/s40815-021-01150-6

Espin-Andrade, R. A., González, Erick., & Fernandez, Eduardo. (2012, septiembre 24). A compensatory inference system. *Congreso latino iberoamericano de investigación operativa*, 4404–4415. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1052.2991&rep=rep1&type=pdf

Espin-Andrade, R. A., Gonzalez, Erick., Pedrycz, Witold., & Fernandez, Eduardo. (2016). An Interpretable Logical Theory: The case of Compensatory Fuzzy Logic. *International Journal of Computational Intelligence Systems*, *9*(4), 612–626. https://doi.org/10.1080/18756891.2016.1204111

Espin-Andrade, R. A., Gonzalez, Erick., Pedrycz, Witold., & Fernández González, E. R. (2015). Archimedean-Compensatory Fuzzy Logic Systems. *International Journal of Computational Intelligence Systems*, *8*(2), 54–62. https://doi.org/10.1080/18756891.2015.1129591

Fan, F.-L., Xiong, J., Li, M., & Wang, G. (2021). On Interpretability of Artificial Neural Networks: A Survey. *IEEE Transactions on Radiation and Plasma Medical Sciences*, *5*(6), 741–760. https://doi.org/10.1109/TRPMS.2021.3066428

Ferreyra, C. (2005). *Redes Neuronales Difusas para modelado via agrupamiento en-línea: Aplicación a un condensador de aspiración*. http://www.ctrl.cinvestav.mx/~yuw/pdf/DoTesAFR.pdf

Gao, F. (2023). Density-based approach for fuzzy rule interpolation. *Applied Soft Computing*, *143*, 110402. https://doi.org/10.1016/j.asoc.2023.110402

Géron, A. (2023). *Hands-On Machine Learning with Scikit-Learn and TensorFlow* (3a ed.). O' Reilly Media, Inc.

Goguen, J. A. (1969). The logic of inexact concepts. *Synthese*, 325–373. https://doi.org/10.1007/BF00485654

González-Caballero, E., Espín-Andrade, R. A., Pedrycz, W., Martínez, L., & Guerrero-Ramos, L. A. (2021). Continuous Linguistic Variables and Their Applications to Data Mining and Time Series Prediction. *International Journal of Fuzzy Systems*, *23*(5), 1431–1452. https://doi.org/10.1007/s40815-020-00968-w

Gottwald, S. (2000). Generalized Solvability Behaviour for Systems of Fuzzy Equations. *Studies in fuzziness and soft computing*, *57*(April), 401–430. https://www.researchgate.net/profile/Siegfried-Gottwald/publication/261680519_Generalised_solvability_behaviour_for_systems_of_fuzzy_equations/links/0f317535048c63fbea000000/Generalised-solvability-behaviour-for-systems-of-fuzzy-equations.pdf

Gündogdu, S. (2022). Order Demand Forecast Using a Combined Approach of Stepwise Linear Regression Coefficients and Artificial Neural Network. *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, *11*(2), 564–573. https://doi.org/10.17798/bitlisfen.1059772

Hájek, P. (1998). *Metamathematics of Fuzzy Logic* (4(1)). Springer Netherlands. https://doi.org/10.1007/978-94-011-5300-3

Hakkoum, H., Abnane, I., & Idri, A. (2022). Interpretability in the medical field: A systematic mapping and review study. *Applied Soft Computing*, *117*, 108391. https://doi.org/10.1016/j.asoc.2021.108391

Hastie, T., Friedman, J., & Tibshirani, R. (2001). *The Elements of Statistical Learning* (Vol. 2). Springer New York. https://doi.org/10.1007/978-0-387-21606-5

Hernández-Julio, Y. F., Prieto-Guevara, M. J., & Nieto-Bernal, W. (2020). Fuzzy clustering and dynamic tables for knowledge discovery and decision-making: Analysis of the reproductive performance of the marine copepod Cyclopina sp. *Aquaculture*, *523*(April 2019), 735183. https://doi.org/10.1016/j.aquaculture.2020.735183

Hong Lan, L. T., Tuan, T. M., Ngan, T. T., Son, L. H., Giang, N. L., Nhu Ngoc, V. T., & Hai, P. Van. (2020). A New Complex Fuzzy Inference System With Fuzzy Knowledge Graph and Extensions in Decision Making. *IEEE Access*, *8*, 164899–164921. https://doi.org/10.1109/ACCESS.2020.3021097

Jalil, M. A., & Misas, M. (2007, junio). Evaluación de pronósticos del tipo de cambio utilizando redes neuronales y funciones de pérdida asimétricas. *Revista Colombiana de Estadística*, *30*(1), 143–161. https://www.redalyc.org/pdf/899/89930110.pdf

Ketkar, N., & Moolayil, J. (2021). *Deep Learning with Python*. Apress. https://doi.org/10.1007/978-1-4842-5364-9

Kruse, R., Held, P., & Moewes, C. (2013). *On Fuzzy Data Analysis* (pp. 343–347). https://doi.org/10.1007/978-3-642-35641-4_49

Livieris, I. E., Kanavos, A., Vonitsanos, G., Kiriakidou, N., Vikatos, A., Giotopoulos, K., & Tampakas, V. (2018). Performance Evaluation of an SSL Algorithm for Forecasting the Dow Jones Index Stocks. *2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 1–8. https://doi.org/10.1109/IISA.2018.8633692

Llorente-Peralta, C. E., Cruz-Reyes, Laura., & Espín-Andrade, R. A. (2021). Knowledge Discovery Using an Evolutionary Algorithm and Compensatory Fuzzy Logic. En Oscar. Castillo & Patricia. Melin (Eds.), *Fuzzy Logic Hybrid Extensions of Neural and Optimization Algorithms: Theory and Applications.* (Studies in, Vol. 940, pp. 363–283). Springer, Cham. https://doi.org/https://doi.org/10.1007/978-3-030-68776-2_21

Lugo Cabrera, C. Mario., & Lopez Herrera, J. (2018). Analitica de datos con aplicación en un caso práctico, mediante el uso de una herramienta libre. *UNIVERSIDAD TECNOLÓGICA DE PEREIRA FACULTAD DE INGENIERÍAS: ELÉCTRICA, ELECTRÓNICA, FÍSICA Y CIENCIAS DE LA COMPUTACIÓN*. https://hdl.handle.net/11059/9185

Matich, D. Jorge. (2001). Redes Neuronales: Conceptos Básicos y Aplicaciones. *Historia*, 1–55.

Melin, P., Mónica, J. C., Sánchez, D., & Castillo, O. (2020). Multiple Ensemble Neural Network Models with Fuzzy Response Aggregation for Predicting COVID-19 Time Series: The Case of México. *Healthcare*, *8*(2), 181. https://doi.org/10.3390/healthcare8020181

Montero, J. C., A. (2011). La Lógica Difusa Compensatoria. *Ingeniería Industrial*, *32*(2), 157–161. https://rii.cujae.edu.cu/index.php/revistaind/article/view/409/424

Montero, J. C., Andrade, R. E., & Robaina, D. A. (2012). Aplicación de la lógica difusa compensatoria en el sector empresarial. *Dyna (Spain)*, *87*(3), 271–274. https://www.researchgate.net/profile/Daniel_Alfonso2/publication/284722771_Aplica

cion_de_la_logica_difusa_compensatoria_en_el_sector_empresarial/links/5657768608ae4988a7b56cd0.pdf
Muñoz, G. Carlos. (2000). Introducción a la lógica matemática. En *Pearson Educación* (Número 1). Pearson Educación.
Neita-Sánchez, A. G., & Mora-Giraldo, J. D. (2022). *Uso de machine learning para la predicción de resultados de las pruebas saber 11 en el Instituto San Ricardo Pampurí.* Universidad Católica de Colombia.
Nguyen Thu Hien, Nguyen Phuong Nhung, & Nguyen Tuan Linh. (2022). Adaptive neuro-fuzzy inference system classifier with interpretability for cancer diagnostic. *Journal of Military Science and Technology*, *CSCE6*, 56–64. https://doi.org/10.54939/1859-1043.j.mst.CSCE6.2022.56-64
Ojha, V. K., Abraham, A., & Snášel, V. (2017). Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, *60*(January), 97–116. https://doi.org/10.1016/j.engappai.2017.01.013
Oladipupo, T. (2010). Machine Learning Overview. En *New Advances in Machine Learning*. InTech. https://doi.org/10.5772/9374
Pérez Pueyo, R. (2005). Procesado y Optimización de Espectros Raman mediante Técnicas de Lógica Difusa: Aplicación a la identificación de Materiales Pictóricos. *Universitat Politécnica de Catalunya*. http://hdl.handle.net/2117/94198
Pointer, Ian. (2019). *Programming pytorch for deep learning: Creating and deploying deep learning applications* (Melissa Potter, Ed.; 1a ed.). O'Reilly Media.
Python Software Foundation. (2024, marzo 28). *Python documentation*. https://docs.python.org/3/
Reddy, G. T., Reddy, M. P. K., Lakshmanna, K., Rajput, D. S., Kaluri, R., & Srivastava, G. (2020). Hybrid genetic algorithm and a fuzzy logic classifier for heart disease diagnosis. *Evolutionary Intelligence*, *13*(2), 185–196. https://doi.org/10.1007/s12065-019-00327-1
Ren, W., Ma, O., Ji, H., & Liu, X. (2020). Human Posture Recognition Using a Hybrid of Fuzzy Logic and Machine Learning Approaches. *IEEE Access*, *8*, 135628–135639. https://doi.org/10.1109/ACCESS.2020.3011697
Rivas-Asanza, W., Mazon-Olivo, B., & Mejía-Peñafiel, E. (2018). Capítulo 1: Generalidades de las redes neuronales artificiales. En *Redes neuronales artificiales aplicadas al reconocimiento de patrones* (1a ed.). UTMACH. http://repositorio.utmachala.edu.ec/handle/48000/14223
Ruiz, Susana., Espín-Andrade, R. Alejandro., Cruz-Reyes, Laura., & Llorente-Peralta, C. Eric. (s/f). Characterization of the Generalized Continuous Linguistic Variable defined in the Archimedean Compensatory Fuzzy Logic. *International Journal of Fuzzy Systems*.
Svozil, D., Kvasnicka, V., & Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, *39*(1), 43–62. https://doi.org/10.1016/S0169-7439(97)00061-0
Valencia Asqui, P. U. (2017). *Implementación de analítica de datos sobre datos geoespaciales en una aplicación de micro-localización que sirva para generar un software de guía dentro de la Universidad Politécnica Salesiana Campus Sur*. 90. http://dspace.ups.edu.ec/handle/123456789/14561

Vidal, E. (1994). Grammatical inference: An introductory survey. En R. Carrasco & J. Oncina (Eds.), *Lecture notes in computer science* (1a ed., Vol. 862, pp. 1–4). Springer. https://doi.org/10.1007/3-540-58473-0_131

Vivek, K., Subbarao, K. V., Routray, W., Kamini, N. R., & Dash, K. K. (2020). Application of Fuzzy Logic in Sensory Evaluation of Food Products: a Comprehensive Study. *Food and Bioprocess Technology*, *13*(1), 1–29. https://doi.org/10.1007/s11947-019-02337-4

Xie, C., Rajan, D., & Chai, Q. (2021). An interpretable Neural Fuzzy Hammerstein-Wiener network for stock price prediction. *Information Sciences*, *577*, 324–335. https://doi.org/10.1016/j.ins.2021.06.076

Yang, J., & Zhao, J. (2023). A novel parallel merge neural network with streams of spiking neural network and artificial neural network. *Information Sciences*, *642*, 119034. https://doi.org/10.1016/j.ins.2023.119034

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338–353. https://doi.org/10.1016/S0019-9958(65)90241-X

Zhu, M., Chen, D., Wang, J., & Sun, Y. (2021). Analysis of oceanaut operating performance using an integrated Bayesian network aided by the fuzzy logic theory. *International Journal of Industrial Ergonomics*, *83*, 103129. https://doi.org/10.1016/j.ergon.2021.103129

Anexo A: Fundamentos de las arquitecturas de red neuronal

Con el objetivo de evitar redundancias y facilitar la comprensión de los modelos desarrollados en los siguientes capítulos, en esta sección se presentan los conceptos fundamentales relacionados con cada una de las arquitecturas implementadas. Todas las implementaciones se llevaron a cabo utilizando el lenguaje de programación Python.

Bibliotecas:

Para el desarrollo de la presente tesis fueron implementadas y desarrolladas bibliotecas que permiten implementar las arquitecturas propuestas donde las funcionalidades que aportan estas bibliotecas son:

ACFLLogic: En esta biblioteca se implementan todos los conceptos relacionados con el manejo de una lógica de ACFL-ELF necesarios para realizar los cálculos en la red. Esta se encuentra descrita en la sección de lógica.

ActivationsFunctions: Biblioteca desarrollada para definir las funciones de activación usadas en las capas intermedias para cada arquitectura, esta es mayormente descrita en la sección de funciones de activación.

Datareader: En esta biblioteca se recibe el conjunto de datos que será tratado a través de la red neuronal, permite la normalización de los datos y obtiene información relevante para el manejo de esta. Esta biblioteca se explica abajo en la sección de tratamiento de los datos.

Resultados: Esta biblioteca toma los resultados entregados por la red neuronal y permite su evaluación a partir de las diferentes métricas de evaluación definidas, esta biblioteca se detalla explícitamente en la parte inferior en la sección de entrega de resultados

Math: Permite la implementación de funciones matemáticas y constantes matemáticas comunes, es decir, permite el uso de operaciones matemáticas como las operaciones trigonométricas, logarítmicas, exponenciales, entre otras (Python Software Foundation, 2024).

Numpy: Es una biblioteca usada para modelos de computación científica y numérica. Proporciona un potente objeto de matriz multidimensional llamado ndarray, junto con una variedad de funciones para realizar operaciones en estas matrices (Python Software Foundation, 2024).

Pytorch: Desarrollada por Facebook's AI Research Lab (FAIR), es una biblioteca de aprendizaje automático muy popular entre los desarrolladores de modelos de aprendizaje automático debido a su flexibilidad y a su arquitectura dinámica de gráficos computacionales (Pointer, 2019).

Proporciona diversas herramientas para el desarrollo de modelos de aprendizaje con redes neuronales y aprendizaje profundo, entre los cuales se tienen:

A. es: Estructuras de datos parecidas a matrices multidimensionales que permiten representar datos y operaciones en el modelo.
B. Módulos y redes neuronales: Proporciona una interfaz intuitiva para definir, entrenar y utilizar modelos de redes neuronales.
C. Autograd: Es un sistema de diferenciación automática el cual permite calcular automáticamente gradientes para la optimización de modelos.
D. Integración con otras bibliotecas: Es posible integrar Pytorch con otras bibliotecas de aprendizaje automático y ciencia de datos tales como NumPy, SciPy, scikit-learn, entre otros.

Sklearn: Biblioteca de aprendizaje automático de código abierto para el lenguaje de programación Python. Es ampliamente utilizada para realizar tareas de aprendizaje automático, incluyendo clasificación, regresión, clustering, entre otros.

Proporciona una variedad de algoritmos de aprendizaje automático pre implementados, así como herramientas para preprocesamiento de datos, selección de modelos, evaluación de rendimiento y validación cruzada (Python Software Foundation, 2024).

Sys: Proporciona funciones y variables relacionadas con el sistema y el entorno de ejecución de python (Python Software Foundation, 2024).

Biblioteca datareader

En cada una de las arquitecturas presentadas, se lleva a cabo una normalización del conjunto de datos de entrada, es decir, se normalizan las características del conjunto de datos al. intervalo [0,1]. Este proceso es importante, ya que permite evitar la influencia de las magnitudes en el entrenamiento, ayuda en el proceso de convergencia, aumenta la estabilidad del algoritmo y permite prevenir problemas de sensibilidad a los datos (Géron, 2023), además en (Llorente-Peralta et al., 2021) se menciona que normalizar las propiedades de un conjunto de datos a un determinado intervalo, permite homogeneizar la definición de los parámetros a descubrir para una lógica difusa compensatoria.

En el Algoritmo 1A, se detalla el proceso seguido para tratar los datos de entrada, los cuales serán usados en los modelos de red neuronal a través de la importación de la biblioteca datareader. Para la construcción de este algoritmo se hace uso de las librerías numpy y csv de Python.

Algoritmo 1A: Lectura_Dataset

Entrada: *Dataset* Conjunto de datos compuesto de filas y columnas
Salida: *Dataset* Conjunto de datos cuyos valores están normalizados en el intervalo $[0,1]$

1. *Dataset* ← Cargar_Dataset("Archivo.csv") // Se lleva a cabo la lectura del *Dataset* y se asigna a *Dataset*
2. **Para cada** *columna* **en** *Dataset*

```
3.      Convertir_float(Dataset,Columna) //Castea los valores del conjunto de datos a
        valores tipo float
4.  MinMax ← Buscar_MinMax(Dataset) //Busca los valores mínimos y máximos de
    cada atributo del conjunto de datos
5.  Normalizar_Dataset(Dataset, MinMax) // Transforma los datos de su magnitud
    original al intervalo [0,1]
6.  Posición ← longitud(MinMax)-1 // Determina el valor mínimo y máximo del atributo
    clase
7.  Mínimo ← MinMax[Posición][0] // Guarda el valor mínimo del atributo clase
8.  Máximo ← MinMax[Posición][1] // Guarda el valor máximo del atributo clase
9.  Devolver Dataset, Mínimo, Máximo

Función Cargar_Dataset(Archivo)
    Dataset = lista() // se define una lista de datos vacía de nombre conjunto de datos
    Data ← Abrir_archivo(Archivo) //Abre el archivo
    LeerCSV ← reader(Data) //Lee el archivo csv
    Para cada fila en LeerCSV:
        Si fila no está vacía:
            Agregar fila a Dataset //Agrega un registro de LeerCSV a un conjunto de datos
    Devolver Dataset

Función Convertir_float(Dataset, columna):
    Para cada fila en Dataset:
        fila[columna] = float(fila[columna].quitar_espacios()) //Convierte los datos de
        tipo string a float sin espacios

Función Buscar_MinMax(Dataset):
    minmax = lista() // se define una lista de datos vacía de nombre minmax
    Para cada columna en Dataset:
        min = mínimo(columna) //Busca el mínimo de la columna
        max = máximo(columna) //Busca el máximo de la columna
        Agregar [min, max] a minmax //Guarda el mínimo y el máximo en minmax
    Devolver minmax

Función Normalizar_Dataset(Dataset, minmax):
    Para cada fila en Dataset:
        Para cada columna en fila:
            Si minmax[columna][0] es igual a minmax[columna][1]: //Si el mayor es igual al
            menor, toda la columna es igual a 1.0
                fila[columna] = 1.0
            Si no: // Se normaliza en el intervalo [0,1] toda la columna
                fila[columna] = (fila[columna] - minmax[columna][0]) / (minmax[columna][1] -
minmax[columna][0])
```

En la línea 1, se abre y lee el archivo sin haber sido modificado, manteniendo así las características originales del conjunto de datos. En las líneas 2 y 3, se itera a lo largo de cada

columna del conjunto de datos y se transforman sus valores originales a tipo float para evitar conflictos en el manejo de los datos, ya que la red neuronal solo puede manejar valores de tipo real. En la línea 4, se identifican los valores mínimo y máximo de cada atributo, los cuales serán utilizados para homogeneizar los datos en el intervalo [0,1] utilizando la ecuación de normalización $f(x) = \frac{x-minimo}{maximo-minimo}$. En la línea 5, se lleva a cabo el proceso de normalización de los datos contenidos en el conjunto de datos. En las líneas 6, 7 y 8 se determinan los valores máximo y mínimo del atributo de clase, que serán útiles posteriormente para transformar las salidas de la red y compararlas con los valores reales del atributo de clase en su escala original. Finalmente, la línea 9 devuelve el conjunto de datos con los datos normalizados, así como los valores mínimo y máximo del atributo de clase.

Biblioteca ACFLLogic

Como se ha presentado anteriormente, es posible generalizar los conceptos de una lógica difusa Arquimediana compensatoria (ACFL) a través de la definición de una función generadora. En Espín et al., (2021) se enumeran algunas de estas funciones, lo que permite definir distintos tipos de lógica difusa Arquimediana compensatoria. Sin embargo, en las arquitecturas presentadas, se ha tomado la decisión de implementar una lógica difusa Arquimediana compensatoria basada en una función logarítmica exponencial (ACFL-ELF).

Para facilitar su implementación dentro de las arquitecturas de red neuronal propuestas, se ha desarrollado una biblioteca denominada ACFLLogic. Esta biblioteca está implementada en el lenguaje de programación Python y hace uso de las bibliotecas Pytorch, Math y Sys.
Una vez establecidas las bibliotecas que serán usadas dentro de la biblioteca ACFLLogic, se procede a crear la clase ACFLLOGIC, la cual hereda propiedades de nn.Module que es una clase base usada para definir modelos de redes neuronales. Este proceso se observa en el siguiente seudocódigo:

Clase ACFLLOGIC(nn.Module):
  Función __init__(*b*, *n*):
    self.b = *b*
    self.n = *n*

La clase ACFLLOGIC contiene un método constructor llamado init. Este método permite definir los valores base logarítmica *b* y exponente *n* de una ACFL-ELF. Posibilitando que los parámetros definidos para la lógica puedan ser utilizados por cada función que forme parte de la clase.

Una vez definida la clase, e inicializada, se procede a enlistar todos los métodos que forman parte de la biblioteca que contiene la lógica de ACFL-ELF:

Función generadora: Corresponde a la ecuación $f(x) = log_b^n(x)$, en esta, el valor de *x* no puede ser 0, ya que el logaritmo de 0 genera un error. En ese caso, se asigna un valor cercano a 0 para evitar problemas. La función se construye a partir del siguiente seudocódigo:

Función FFunción(self,*x*):

Si *x* **es menor o igual a** 0, **entonces**
x = 0.0000001
Fin Si
$f = -log_{self.b}^{self.n}(x)$
Devolver *f*

Función generadora inversa: Corresponde a la ecuación $f^{-1}(x) = b^{-\sqrt[n]{x}}$. En esta función se tiene en cuenta que el valor obtenido por la función FFunction puede ser negativo, por lo que es necesario considerarlo al realizar la operación. Además, es importante tener en cuenta que los valores calculados pueden ser tan pequeños que no puedan ser representados con precisión en la computadora. Por lo tanto, en este caso se utiliza el valor mínimo representable por la computadora.

Función FInvFunción(self, *x*)
tsign = **signo_de**(*x*)
$X = (|x|)^n$
$X = X * tsign$

Intentar
result = $self.b^{-X}$
Devolver *result*
Excepto(DesbordamientoDeDesbordamiento,ErrorDeDivisiónPorCero,ErrorDeValor)
Devolver *mínimo_valor_representable_por_la_computadora*

Función sigmoidal generalizada: Corresponde a la ecuación $S_g(x;\alpha,\gamma) = \frac{1}{1+b^{-\sqrt[n]{(\alpha(x-\gamma)}}}$, la cual transforma los valores de cada atributo del conjunto de datos al intervalo $[0,1]$, es decir lleva a cabo la normalización lógica difusa.

Función SigFunction(self, *x, α, γ*)
$X = \alpha * (x - \gamma)$
Result = $\frac{1}{1+\text{self.FInvFunction}(X)}$
Devolver *Result*

Función CT: Correspondiente a la siguiente fórmula $C_T = \sqrt[n]{b^{\log_b^n(s_g(x;\alpha,\gamma)_L^m)+\log_b^n(1-s_g(x;\alpha,\gamma)_L^{1-m})}}$, permite generar la función de membresía definida a partir de los parámetros α, γ y m para una ACFL-ELF.

Función CT_GCLVFunction(self, *x, α, γ, m*)
Sig = self.SigFunction(*x, α, γ*)
NSig = $1 - sig$
$Result = \text{self.FInvFunction}\big(\text{self.FFunction}(sig) * m + \text{self.FFunction}(NSig) * (1-m)\big)$
Devolver *Result*

Función máximo de CT: Correspondiente a la ecuación $Max_{x\in\mathbb{R}}(C_T)$, donde el valor maximo permite normalizar los valores de C_T al intervalo $[0,1]$ difuso, necesario para una ACFL-AELF.

Función Mbin_GCLV(self, α, γ, m)
x = de tamaño igual a *m* rellenado con 0.0
Para cada *reg* **en el rango del tamaño de** *m*
Si *m*[*reg*] **es igual a** 0.5
x[*reg*] = self.CT_GCLVFunction(γ [*reg*], α[*reg*], γ [*reg*], *m*[*reg*])
Si *m*[*reg*] **es igual a** 1
x[*reg*] = self.CT_GCLVFunction (1.0, α[*reg*], γ [*reg*], *m*[*reg*])
Si *m*[*reg*] **es igual a** 0
x[*reg*] = self.CT_GCLVFunction (0.5, α[*reg*], γ [*reg*], *m*[*reg*])
Si *m*[*reg*] **es mayor que** 0.5 **y** *m*[*reg*] **es menor que** 1.0
A = ([γ [*reg*]])
B = ([1.0])
C = ([(*B* - *A*) / 2) + *A*])
Mientras | self.CT_GCLVFunction(*A*, α[*reg*], γ [*reg*], *m*[*reg*]) -
self.CT_GCLVFunction(*B*, α[*reg*], γ [*reg*], *m*[*reg*])|**sea mayor que** 0.001
Si self.CT_GCLVFunction(*C*, α[*reg*], γ [*reg*], *m*[*reg*]) **es menor que**
self.CT_GCLVFunction(C + 0.1, α[*reg*], γ [*reg*], *m*[*reg*])
A = *C*
C = ((*B* - *A*) / 2) + *A*
Si no
B = *C*
C = ((*B* - *A*) / 2) + *A*
x[*reg*] = self.CT_GCLVFunction(*C*, α[*reg*], γ [*reg*], *m*[*reg*])
Si *m*[*reg*] **es menor que** 0.5 y *m*[*reg*] **es mayor que** 0
A = ([0.0])
B = ([γ [*reg*]])
C = ([(*B* - *A*) / 2) + *A*])
Mientras | self.CT_GCLVFunction(*A*, α[*reg*], γ [*reg*], *m*[*reg*]) –
self.CT_GCLVFunction(*B*, α[*reg*], γ [*reg*], *m*[*reg*])| **sea mayor que** 0.001
Si self.CT_GCLVFunction(*C*, α[*reg*], γ [*reg*], *m*[*reg*]) **es menor que**
self.CT_GCLVFunction(C + 0.1, α[*reg*], γ [*reg*], *m*[*reg*])
A = *C*
C = ((*B* - *A*) / 2) + *A*
Sino
B = *C*
C = ((*B* - *A*) / 2) + *A*
x[*reg*] = self.CT_GCLVFunction(*C*, α[*reg*], γ [*reg*], *m*[*reg*])
Devolver *x*

Función modificador lingüístico: Correspondiente a la ecuación $x_L^a = b^{-\sqrt[n]{a-log_b^n(x)}}$, pero para este caso enfocando su implementación en los cálculos necesarios dentro de una red neuronal.

Función modifier(self, *w*, *x*)
Valor = de tamaño igual a *w* rellenado con 0.0
Para *fila* **en rango de filas de** *w*
Para *columna* **en rango de columnas de** *w*
Si *w*[*fila*][*columna*] < 0 **entonces**
Valor[*fila*][*columna*] = self.FInvFunction(abs(*w*[*fila*][*columna*]) *
self.FFunction(1 - *x*[*columna*]))
Si no
Valor[*fila*][*columna*] = self.FInvFunction(*w*[*fila*][*columna*] *
self.FFunction(*x*[*columna*]))
Devolver *Valor*

Biblioteca ActivationsFunctions

En esta sección se presenta una biblioteca llamada ActivationsFunctions, la cual define las funciones de activación que se utilizarán en las arquitecturas de las redes neuronales artificiales propuestas. Esta biblioteca utiliza las bibliotecas Pytorch y Math de Python, además de una biblioteca propia denominada ACFLLogic, que contiene conceptos fundamentales necesarios para la construcción de dichas funciones.

A continuación, se enumeran las funciones de activación contenidas en ActivationsFunctions, las cuales permiten realizar los cálculos necesarios para el entrenamiento y evaluación de la red neuronal.

Clase GCLV: Correspondiente a la ecuación $GCLV_L(x;\alpha,\gamma,m) = \frac{b^{\sqrt[n]{\log_b^n(s_g(x;\alpha,\gamma)_L^m)+\log_b^n(1-s_g(x;\alpha,\gamma)_L^{1-m})}}}{\max_{x\in\mathbb{R}}\left[b^{\sqrt[n]{\log_b^n(s_g(x;\alpha,\gamma)_L^m)+\log_b^n(1-s_g(x;\alpha,\gamma)_L^{1-m})}}\right]}$, la cual permite generar una familia de funciones de membresía, este concepto es altamente importante, debido a que mitiga el problema de tener que definir la variable especifica al problema específico, debido a la adaptabilidad del perfil de esta.

Clase GCLV:
Función __init__(self, *neurons*, *b*, *n*):
Inicializar parámetros α, γ y *m* como es aleatorios de tamaño '*neurons*'
//Donde $\alpha\epsilon[4,10]$, $\gamma = 0.5$ y $m\epsilon[0,1]$.
Inicializar *b* y *n* con los valores definidos
Self.ACFL = ACFLLOGIC(self.*b*, self.*n*) //Crea una instancia de la clase ACFLLOGIC con los valores de *b* y *n*
Fin de la función __init__

Función forward(self, *x*):
Limitar los valores de $m = [0,1]$
M = Mbin_GCLV(α, γ, *m*)
result = self.ACFL.CT_GCLVFunction(*x*, α, γ, *m*)/M
Si algún valor de resultado es **mayor que** 1, se ajusta a 1
Retornar *result*

Clase Conjunción CFL: Correspondiente a la ecuación $c_c(x_1, x_{2,\dots,} x_n) = b\sqrt[n]{\frac{\sum_{i=1}^{n} \log_b^n(x_i)}{n}}$. Permite usar un operador de conjunción de una lógica difusa compensatoria, con lo cual es resultado es un predicado conjuntivo para la neurona donde se implementa esta clase como función de activación, es decir el nodo puede ser interpretado algebraicamente como $atributo_1 \wedge \dots \wedge atributo_n$.

Clase Conjunction_CFL:
Función __init__(self, *in_features*, *out_features*, *b*, *n*):
Inicializar *b* y *n* como parámetros de la clase
Crear un conjunto de parámetros *w* de tamaño (*in_features*, *out_features*)
self.reset_parameters() // Inicializar *w* usando la inicialización Kaiming_uniform
Self.ACFL = ACFLLOGIC(self.*b*, self.*n*)

Función reset_parameters(self):
n.init.kaiming_uniform_(self.*w*, *a*= sqrt(5))

Función forward(self, x):
batch_size, in_features = x.size() //Inicializa el número de filas (batch size) y columnas (in_features)
FilasPesos, *ColumnasPesos* = self.*w*.size() //Obtiene la dimensión de w
Crear un vacío *output* donde almacena la salida (*batch_size*, *ColumnasPesos*)

Para cada *fila* **en** *batch_size*:
Para cada *columna* **en** *ColumnasPesos*:
Inicializar *result* **como** 0
Para cada *columna2* **en** *in_features*:
Si *w* [*columna2*, *columna*] **es mayor o igual** a 0:
result += self.ACFLLOGIC.FinvFunction(self.ACFLLOGIC.FFunction(*x*[*fila*, *columna2*]) * self.*w*[*columna2*, *columna*]))
Si no
result += self.ACFLLOGIC.FinvFunction(self.ACFLLOGIC.FFunction(1-*x*[*fila*, *columna2*]) * abs(self.*w*[*columna2*, *columna*])))

output[*fila*, *columna*] = *result* /*in_features*

Devolver self.ACFLLOGIC.FinvFunction(*output*)

Clase Conjunction_AFL: Corresponde a la ecuación $c_T(x_1, x_2, \dots, x_n) = b^{\sqrt[n]{\sum_{i=1}^{n} log_b^n(x_i)}}$. Permite usar un operador de conjunción de una lógica difusa Arquimediana, con lo cual es resultado es un predicado conjuntivo para la neurona donde se implementa esta clase como función de activación, es decir el nodo puede ser interpretado algebraicamente como $atributo_1 \wedge \dots \wedge atributo_n$. El algoritmo usado es exactamente igual que en el anterior, a diferencia que la salida se calcula de la siguiente manera: *output*[*fila*, *columna*] = *result*.

Clase Disyunción CFL: Correspondiente a la ecuación $d_c(x_1, x_{2,\dots,} x_n) = 1 - b^{\sqrt[n]{\frac{\sum_{i=1}^{n} \log_b^n(1-x_i)}{n}}}$. Permite usar un operador de disyunción de una lógica difusa compensatoria, con lo cual es resultado es un predicado disyuntivo para la neurona donde se implementa esta clase como función de activación, es decir el nodo puede ser interpretado algebraicamente como $atributo_1 \vee \dots \vee atributo_n$.

Clase Disjunction_CFL:
Función __init__(self, *in_features*, *out_features*, *b*, *n*):
Inicializar *b* y *n* como parámetros de la clase
Crear un conjunto de parámetros *w* de tamaño (*in_features*, *out_features*)
self.reset_parameters() // Inicializar *w* usando la inicialización Kaiming_uniform
Self.ACFL = ACFLLOGIC(self.*b*, self.*n*)

Función reset_parameters(self):
n.init.kaiming_uniform_(self.*w*, *a*= sqrt(5))

Función forward(self, x):
batch_size, in_features = x.size() //Inicializa el número de filas (batch size) y columnas (in_features)
FilasPesos, *ColumnasPesos* = self.*w*.size() //Obtiene la dimensión de w
Crear un conjunto vacío *output* donde almacena la salida (*batch_size*, *ColumnasPesos*)

Para cada *fila* **en** *batch_size*:
Para cada *columna* **en** *ColumnasPesos*:
Inicializar *result* **como** 0
Para cada *columna2* **en** *in_features*:
Si *w* [*columna2*, *columna*] **es mayor o igual** a 0:
result += self.ACFLLOGIC.FinvFunction(self.ACFLLOGIC.FFunction(1-*x*[*fila,columna2*]) * self.*w*[*columna2*, *columna*]))
Si no
result += self.ACFLLOGIC.FinvFunction(self.ACFLLOGIC.FFunction(*x*[*fila*, *columna2*]) * abs(self.*w*[*columna2*, *columna*])))

output[*fila*, *columna*] = *result* /*in_features*

Devolver 1-self.ACFLLOGIC.FinvFunction(*output*)

Clase Disyunción AFL: Correspondiente a la ecuación $d_T(x_1, x_2, \dots, x_n) = 1 - b\sqrt[n]{\sum_{i=1}^{n} log_b^n(1-x_i)}$. Permite usar un operador de disyunción de una lógica difusa Arquimediana, con lo cual el resultado es un predicado disyuntivo para la neurona donde se implementa esta clase como función de activación, es decir el nodo puede ser interpretado algebraicamente como $atributo_1 \vee \dots \vee atributo_n$. El algoritmo usado es exactamente igual que en el anterior, a diferencia que la salida se calcula de la siguiente manera: *output*[*fila*, *columna*] = *result*.

Biblioteca resultados

Después de completar las pruebas para cada arquitectura, se emplea la biblioteca resultados, la cual realiza los cálculos necesarios para presentar las métricas de evaluación establecidas las cuales se detallan en la sección de forma de evaluación. Esta biblioteca utiliza las siguientes librerías: PyTorch, Scikit-learn, NumPy y Sys, todas ellas integradas dentro del lenguaje de programación Python.

Dentro de esta biblioteca existen dos funciones principales. En la primera se hace la recepción de los datos de cada una de las experimentaciones y posteriormente se evalúan a través de las métricas antes indicadas, los resultados son impresos en un archivo txt. En la segunda, se reciben los datos de los parámetros descubiertos mediante el proceso de optimización y posteriormente se imprimen usando un determinado formato.

Función imprimir_resultados(*predicted*, *labels*, *max*, *min*)
  original_stdout = sys.stdout
  output_file_name = 'Nombre_Archivo.txt'

  Abrir *output_file_name* **para agregar como** *file*
  sys.stdout = *file*
  resultados_prediccion(*predicted*, *labels*, *max*, *min*)
  Cerrar *output_file_name*

  sys.stdout = *original_stdout*
  Imprimir "Los resultados han sido guardados en ", *output_file_name*

Función imprimir_parametros(*parámetros*)
  original_stdout = sys.stdout
  output_file_name = 'Nombre.txt'

  Abrir *output_file_name* **para agregar como** *file*
  sys.stdout = *file*
  resultados_parametros(*parámetros*)
  Cerrar *output_file_name*

  sys.stdout = *original_stdout*
  Imprimir "Los resultados han sido guardados en ", *output_file_name*

Estas dos funciones hacen uso de las siguientes funciones secundarias, las cuales son las que evalúan los resultados a través de las métricas establecidas e imprimen los datos en el formato definido.

Función resultados_prediccion(*predicted, labels*, *max*, *min*, *time*):
predicted = *predicted* * (*max* - *min*) + *min*
labels = *labels* * (*max* - *min*) + *min*
diferencia = valor_absoluto(*predicted* - *labels*)

range1 = (d*iferencia* < 0.25).sum().convertir_a_entero()
range2 = (*diferencia* < 0.5).sum().convertir_a_entero()
range3 = (*diferencia* < 1.0).sum().convertir_a_entero()

mse = calcular_error_cuadrado_medio(*labels*.convertir_a_numpy(),
predicted.detach().convertir_a_numpy())
mae = calcular_error_absoluto_medio(*labels*.convertir_a_numpy(),
predicted.detach().convertir_a_numpy())
r2 = calcular_r_cuadrada(*labels*.convertir_a_numpy(),
predicted.detach().convertir_a_numpy())

imprimir("Mean Squared Error (MSE); ", *mse*)
imprimir("Mean Absolute Error (MAE); ", *mae*)
imprimir("R cuadrada (R^2); ", *r2*)
imprimir("Accuracy whit .25; ", (*range1* * 100) / longitud(*predicted*))
imprimir("Accuracy whit .5; ", (*range2* * 100) / longitud(*predicted*))
imprimir("Accuracy whit .1; ", (*range3* * 100) / longitud(*predicted*))

predicted = redondear(*predicted*).convertir_a_entero()
conf = calcular_matriz_de_confusion(*labels*, *predicted*)
imprimir("Matriz de confusión")
imprimir(*conf*)

Función resultados_parametros(*parámetros*)
Para cada *nombre*, *parámetro* **en** *parametros*.item
Imprimir *nombre*, " ; ", *parámetro*

Métricas de evaluación

Para llevar a cabo la evaluación de las arquitecturas propuestas, se hará uso del algoritmo de cross fold validation, el cual es ampliamente usado en modelos de aprendizaje automático y modelos estadísticos.

El algoritmo Consiste en dividir el conjunto de datos en subconjuntos de entrenamiento y prueba de manera repetida, de tal manera que cada subconjunto de prueba se utilice una vez como datos de prueba y los restantes como datos de entrenamiento.

De tal manera que el proceso básico para evaluar el modelo usando el algoritmo de validación cruzada implica los siguientes pasos:

I. Dividir el conjunto de datos en cinco subconjuntos aproximadamente iguales (llamados "folds").
II. Entrenar el modelo cinco veces, cada vez utilizando cuatro de los folds como datos de entrenamiento y el fold restante como datos de prueba.
III. Calcular las métricas de evaluación para cada iteración.
IV. Calcular la métrica promedio de todas las iteraciones para obtener una estimación confiable del rendimiento del modelo.

Este proceso ayuda a mitigar el sesgo y la variabilidad asociados con una sola división de los datos, proporcionando una evaluación robusta del rendimiento del modelo (Hastie et al., 2001).

Dentro de los datos que se usan para evaluar el funcionamiento del modelo se usan los siguientes:

Mean Squared Error (MSE):
Fórmula: MSE = $\frac{1}{n}\sum(y_i - y)^2$
Interpretación: Mide el promedio de los cuadrados de las diferencias entre las predicciones del modelo (y_i) y los valores reales ($\bar{y}$). Cuanto menor sea el MSE, mejor será el modelo (Neita-Sánchez & Mora-Giraldo, 2022).

Mean Absolute Error (MAE):
Fórmula: MAE = $\frac{1}{n}\sum|y_i - \bar{y}|$
Interpretación: Similar al MSE, pero en lugar de elevar al cuadrado las diferencias, toma el valor absoluto. MAE es menos sensible a valores atípicos que el MSE (Neita-Sánchez & Mora-Giraldo, 2022).

R cuadrada (R^2):
Fórmula: $R^2 = 1 - (\sum(y_i - \bar{y})^2)/\sum(y_i - \bar{y})^2$
Interpretación: Mide la proporción de la variabilidad en la variable dependiente que es explicada por el modelo. R^2 varía de 0 a 1; un valor cercano a 1 indica un mejor ajuste (Neita-Sánchez & Mora-Giraldo, 2022).

Accuracy with .25, .5, 1:
Fórmula: Número de predicciones correctas / Total de predicciones
Interpretación: Mide la precisión del modelo para diferentes umbrales (thresholds). Por ejemplo, "Accuracy with .25" considera correctas las predicciones que están dentro de ±0.25 del valor real (González-Caballero et al., 2021).

Anexo B: Detalles de los conjuntos de datos utilizados

Los conjuntos de datos utilizados en esta investigación se descargaron del repositorio UCI Machine Learning, donde cada uno está disponible para su descarga.

BUPA: llamada también liver desorders, es una base de datos de Medical Research Ltd. donada por Richard S. Forsyth. Algunas de las características del conjunto de datos es que los datos que presenta son multivariados, está enfocado a área de la salud y la medicina, está enfocada a tareas de regresión, consta de 354 registro y cinco características.

Las primeras 5 variables son todos análisis de sangre que se cree que son sensibles a los trastornos hepáticos que pueden surgir por el consumo excesivo de alcohol. Cada línea del conjunto de datos constituye el registro de un solo individuo masculino.

Variable	Rol	Tipo	Descripción	Valores perdidos
mcv	Característica	Continuous	mean corpuscular volume	no
alkphos	Característica	Continuous	alkaline phosphotase	no
sgpt	Característica	Continuous	alanine aminotransferase	no
sgot	Característica	Continuous	aspartate aminotransferase	no
gammagt	Característica	Continuous	gamma-glutamyl transpeptidase	no
drinks	Objetivo	Continuous	number of half-pint equivalents of alcoholic beverages drunk per day	no

Liver Disorders. (1990). UCI Machine Learning Repository. https://doi.org/10.24432/C54G67.

Iris Flowers es un conjunto de datos creado por Fisher en 1936 y es uno de los primeros conjuntos de datos conocidos utilizados para evaluar métodos de clasificación. Este conjunto de datos está enfocado en el área de la biología y se utiliza comúnmente en problemas de clasificación. Las variables en este conjunto de datos son valores numéricos reales, y consta de 150 registros y cuatro características.

Es uno de los primeros conjuntos de datos utilizados en la literatura sobre métodos de clasificación y es ampliamente utilizado en estadística y aprendizaje automático. El conjunto de datos contiene 3 clases de 50 registros cada una, donde cada clase se refiere a un tipo de planta de iris. Una clase es linealmente separable de las otras dos; sin embargo, las últimas dos clases no son linealmente separables entre sí.

Variable	Rol	Tipo	Descripción	Unidad	Valores perdidos
sepal length	Característica	Continua		cm	No
sepal width	Característica	Continua		cm	No
petal length	Característica	Continua		cm	No
petal width	Característica	Continua		cm	No
class	Objetivo	Discreta	Clase de planta iris: Iris Setosa, Versicolor, o Virginica		No

Iris (1988). UCI Machine Learning Repository. **10.24432/C56C76**

Pima Indians Diabetes: Ampliamente utilizado en tareas de aprendizaje automático y ciencia de datos. Se compone de datos médicos de mujeres de ascendencia pima india, una población nativa americana en Arizona, Estados Unidos.
El conjunto de datos contiene información sobre varios factores de riesgo para la diabetes, así como también información sobre si las mujeres desarrollaron diabetes o no en los siguientes cinco años al momento de la medición.

Se compone de un total de 768 registros y ocho características.

Características:

- Número de embarazos.
- Concentración plasmática de glucosa a 2 horas en una prueba de tolerancia a la glucosa oral.
- Presión arterial diastólica (mm Hg).
- Grosor del pliegue cutáneo del tríceps (mm).
- Insulina sérica de 2 horas (mu U/ml).
- Índice de masa corporal (IMC).
- Función del pedigrí de la diabetes.
- Edad.
- Variable de resultado: resultado binario que indica si la paciente desarrolló diabetes en los siguientes cinco años (1 para diabetes, 0 para no diabetes).

Tinto: El objetivo es modelar la calidad del vino a partir de pruebas fisicoquímicas. Entre sus características presenta valores multivariados, está orientado al área de los negocios, principalmente enfocado en tareas de clasificación y regresión, la variable objetivo es real, cuanta con un total de 1599 registros y 11 características.

Variable	Role	Tipo	Valores perdidos
fixed_acidity	Característica	Continuo	no
volatile_acidity	Característica	Continuo	no
citric_acid	Característica	Continuo	no
residual_sugar	Característica	Continuo	no
chlorides	Característica	Continuo	no

Variable	Role	Tipo	Valores perdidos
free_sulfur_dioxide	Característica	Continuo	no
total_sulfur_dioxide	Característica	Continuo	no
density	Característica	Continuo	no
pH	Característica	Continuo	no
sulphates	Característica	Continuo	no
Alcohol	Característica	Continuo	
quality	objetivo	Real	no

Cortez, Paulo, Cerdeira, A., Almeida., Matos,T., and Reis,J. (2009). Wine Quality. UCI Machine Learning Repository. https://doi.org/10.24432/C56S3T.

Daily Demand Forecasting Orders: Se trata de una base de datos real de una empresa de logística brasileña. Es una serie de tiempo que se asocia con el área de los negocios y se utiliza principalmente en tareas de regresión. La variable objetivo es continua y el conjunto de datos consta de 60 registros con 12 características.

Variable	Rol	Tipo	Valores perdidos
Week of the month	Característica	Entero	no
Day of the week	Característica	Entero	no
Non-urgent order	Característica	Continuo	no
Urgent order	Característica	Continuo	no
Order type A	Característica	Continuo	no
Order type B	Característica	Continuo	no
Order type C	Característica	Continuo	no
Fiscal sector orders	Característica	Continuo	no
Orders from the traffic controller sector	Característica	Entero	no
Banking orders (1)	Característica	Entero	no
Banking orders (2)	Característica	Entero	
Banking orders (3)	Característica	Entero	
Total orders	Objetivo	Continuo	

Ferreira,Ricardo, Martiniano,Andrea, Ferreira,Arthur, Ferreira,Aleister, and Sassi,Renato. (2017). Daily Demand Forecasting Orders. UCI Machine Learning Repository. https://doi.org/10.24432/C5BC8T.

Dow Jones Index: Este conjunto de datos contiene datos semanales del índice industrial Dow Jones. Se ha utilizado en la investigación de inversión computacional. Es una serie de tiempo que se asocia con el área de los negocios, usado en problemas de clasificación y agrupación, la función objetivo es un valor continuo y se compone de 750 registros con 15 características.

Variable	Rol	Tipo	Valores perdidos
quarter	Característica	Entero	no
stock	Característica	Discreto	no
date	Característica	Fecha	no
open	Característica	Discreto	no
high	Característica	Discreto	no
low	Característica	Discreto	no
close	Característica	Discreto	no
volume	Característica	Entero	no
percent_change_price	Característica	Continuo	no
percent_change_volume_over_last_wk	Característica	Continuo	no
previous_weeks_volume	Característica	Entero	no
next_weeks_open	Característica	Discreto	no
next_weeks_close	Característica	Discreto	no
percent_change_next_weeks_price	Objetivo	Continuo	no

Brown,Michael. (2014). Dow Jones Index. UCI Machine Learning Repository. https://doi.org/10.24432/C5788V.

Anexo C: Resultados de una NN-ACFL y su interpretación en lenguaje natural

En el siguiente apartado, se presentan los resultados obtenidos en los experimentos realizados con una NN-ACFL, y se interpretan en lenguaje natural.

Resultados del conjunto de datos Daily Orders.

En las Tablas 1, 2 y 3 se presentan los resultados de la NN- ACFL para el conjunto de datos Daily Orders.

Tabla 1: Valores de α, γ y m descubiertos para cada característica de Daily Orders

GCLV	α	γ	m
WEEK OF THE MONTH	5.4833	0.4267	5.43E-01
DAY OF THE WEEK	6.5361	0.4031	7.4537e-01
NON-URGENT ORDER	6.9213	0.4375	1.8109e-01
URGENT ORDER	6.8329	0.2845	7.5805e-01
ORDER TYPE A	6.5874	0.695	3.5574e-01
ORDER TYPE B	5.1296	0.0661	1.6157e-01
ORDER TYPE C	7.9137	0.3869	9.3713e-01

FISCAL SECTOR ORDERS	9.6491	0.4088	5.1783e-01
ORDERS FROM CONTROLLER SECTOR	7.5653	0.4577	6.1133e-01
BANKING ORDERS (1)	8.3677	0.6693	2.6809e-04
BANKING ORDERS (2)	5.4051	0.2562	4.0860e-01
BANKING ORDERS (3)	6.855	0.5652	4.40E-01

Tabla 2: Parámetros *w* descubiertos para la capa conjuntiva

ATRIBUTO	**Predicado 1**	**Predicado 2**	**Predicado 3**
	w	*w*	*w*
WEEK OF THE MONTH	-3.80E-03	-4.66E-03	-4.59E-03
DAY OF THE WEEK	-2.46E-04	1.5147e-04	-4.04E-04
NON-URGENT ORDER	-3.83E-02	-2.62E-03	-1.48E-02
URGENT ORDER	-5.32E-03	-3.61E-03	-7.11E-03
ORDER TYPE A	-9.58E-03	-9.37E-03	-7.55E-02
ORDER TYPE B	4.68E-03	2.2661e-03	3.4887e-02
ORDER TYPE C	-6.15E-03	9.5747e-05	-2.71E-02
FISCAL SECTOR ORDERS	-2.71E-03	-1.65E-02	-7.14E-03
ORDERS FROM CONTROLLER SECTOR	-1.27E-03	-1.44E-03	-4.51E-04
BANKING ORDERS (1)	1.61E-01	4.4526e-02	3.0931e-01
BANKING ORDERS (2)	1.39E-02	-5.56E-03	-4.00E-03
BANKING ORDERS (3)	-2.40E-03	-2.54E-03	-3.48E-03
ATRIBUTO	**Predicado 4**	**Predicado 5**	**Predicado 6**
	w	*w*	*w*
WEEK OF THE MONTH	-2.60E-04	-2.91E-03	-5.01E-01
DAY OF THE WEEK	-3.81E-02	-2.55E-04	-5.39E-01
NON-URGENT ORDER	-1.22E-01	-3.65E-02	5.34E-01
URGENT ORDER	2.9104e-02	-3.25E-03	-3.67E-01
ORDER TYPE A	-3.17E-02	-6.04E-02	-2.92E-01
ORDER TYPE B	-2.57E-01	1.5021e-03	8.32E-01
ORDER TYPE C	9.2718e-02	-7.80E-04	-6.50E-01
FISCAL SECTOR ORDERS	-1.78E-02	-5.36E-02	5.46E-01

ORDERS FROM CONTROLLER SECTOR	-1.45E-02	-1.61E-03	2.33E-01
BANKING ORDERS (1)	6.1699e-02	1.9900e-01	-3.11E-01
BANKING ORDERS (2)	-5.29E-02	-5.12E-03	2.57E-01
BANKING ORDERS (3)	-1.50E-03	-4.87E-02	3.43E-01

Tabla 3: Parámetros w descubiertos para la capa disyuntiva

	Predicado 1	**Predicado 2**	**Predicado 3**
W	-0.4926	-0.6843	-0.6938
	Predicado 4	**Predicado 5**	**Predicado 6**
W	0.1971	-0.2364	0.0079

A partir de los datos encontrados y por medio del Algoritmo 4.3 y 4.4 se expresan los resultados en lenguaje natural:

Si

Week of the month está entre 5.0 y 1.0 y es aproximadamente igual a 2 **Y** Day of the week está entre 6.0 y 2.0 y es aproximadamente igual a 4 **Y** Non-urgent order está entre 435.304 y 43.651 y es aproximadamente igual a 141.5642 **Y** Urgent order está entre 223.27 y 77.371 y es aproximadamente igual a 205.0326 **Y** Order type A esta entre 118.178 y 21.825 y es aproximadamente igual a 94.09 **Y** Order type B esta entre 267.342 y 25.125 y es aproximadamente igual a 252.2034 **Y** Order type C esta entre 302.448 y 74.372 y es aproximadamente igual a 245.4290 **Y** Fiscal sector orders está entre 865.0 y 0.0 y es aproximadamente igual a 216.2500 **Y** Orders from the traffic controller sector está entre 71772.0 y 11992.0 y es aproximadamente igual a 56827 **Y** Banking orders (1) es menor que 210508.0 y es aproximadamente igual a 158744 **Y** Banking orders (2) está entre 188411.0 y 16411.0 y es aproximadamente igual a 145411 **Y** Banking orders (3) está entre 73839.0 y 7679.0 y es aproximadamente igual a 57299 intensificado a la -0.4926

O

Week of the month está entre 5.0 y 1.0 y es aproximadamente igual a 2 **Y** Day of the week está entre 6.0 y 2.0 y es aproximadamente igual a 4 **Y** Non-urgent order está entre 435.304 y 43.651 y es aproximadamente igual a 141.5642 **Y** Urgent order está entre 223.27 y 77.371 y es aproximadamente igual a 205.0326 **Y** Order type A esta entre 118.178 y 21.825 y es aproximadamente igual a 94.09 **Y** Order type B esta entre 267.342 y 25.125 y es aproximadamente igual a 252.2034 **Y** Order type C esta entre 302.448 y 74.37 y es aproximadamente igual a 188.4100 **Y** Fiscal sector orders está entre 865.0 y 0.0 y es aproximadamente igual a 216.2500 **Y** Orders from the traffic controller sector está entre

71772.0 y 11992.0 y es aproximadamente igual a 56827 **Y** Banking orders (1) está entre 210508.0 y 3452.0 y es aproximadamente igual a 106980 **Y** Banking orders (2) está entre 188411.0 y 16411.0 y es aproximadamente igual a 102411 **Y** Banking orders (3) está entre 73839.0 y 7679.0 y es aproximadamente igual a 57299 intensificado a la -0.6843

O

Week of the month está entre 5.0 y 1.0 y es aproximadamente igual a 2 **Y** Day of the week está entre 6.0 y 2.0 y es aproximadamente igual a 5 **Y** Non-urgent order está entre 435.304 y 43.651 y es aproximadamente igual a 141.5642 **Y** Urgent order está entre 223.27 y 77.371 y es aproximadamente igual a 205.0326 **Y** Order type A esta entre 118.178 y 21.8259 y es aproximadamente igual a 115.1670 Y Order type B esta entre 267.342 y 25.125 y es aproximadamente igual a 265.449 **Y** Order type C esta entre 302.448 y 74.37 y es aproximadamente igual a 273.9385 **Y** Fiscal sector orders está entre 865.0 y 0.0 y es aproximadamente igual a 216.25 **Y** Orders from the traffic controller sector está entre 71772.0 y 11992.0 y es aproximadamente igual a 56827 **Y** Banking orders (1) es menor que 210508.0 y es aproximadamente igual a 158744 **Y** Banking orders (2) está entre 188411.0 y 16411.0 y es aproximadamente igual a 102411**Y** Banking orders (3) está entre 73839.0 y 7679.0 y es aproximadamente igual a 57299 intensificado a la -0.6938

O

Week of the month está entre 5.0 y 1.0 y es aproximadamente igual a 3 **Y** Day of the week está entre 6.0 y 2.0 y es aproximadamente igual a 5.5 **Y** Non-urgent order está entre 435.304 y 43.651 y es aproximadamente igual a 141.5642 **Y** Urgent order está entre 223.27 y 77.371 y es aproximadamente igual a 150.3204 **Y** Order type A esta entre 118.178 y 21.825 y es aproximadamente igual a 106.1340 **Y** Order type B esta entre 259.979 y 25.125 y es aproximadamente igual a 85.6793 **Y** Order type C esta entre 287.7478 y 74.372 y es aproximadamente igual a 188.41 **Y** Fiscal sector orders está entre 865.0 y 0.0 y es aproximadamente igual a 216.25 **Y** Orders from the traffic controller sector está entre 71772.0 y 11992.0 y es aproximadamente igual a 64299.5 **Y** Banking orders (1) está entre 210508.0 y 3452.0 y es aproximadamente igual a 106980 **Y** Banking orders (2) está entre 188411.0 y 16411.0 y es aproximadamente igual a 59411 **Y** Banking orders (3) está entre 73839.0 y 7679.0 y es aproximadamente igual a 57299 intensificado a la 0,1971

O

Week of the month está entre 5.0 y 1.0 y es aproximadamente igual a 2 **Y** Day of the week está entre 6.0 y 2.0 y es aproximadamente igual a 4 **Y** Non-urgent order está entre 435.304 y 43.651 y es aproximadamente igual a 141.5642 **Y** Urgent order está entre 223.27 y 77.371 y es aproximadamente igual a 205.0326 **Y** Order type A esta entre 118.178 y 21.8259 y es aproximadamente igual a 112.1560 **Y** Order type B esta entre 267.342 y 25.125 y es aproximadamente igual a 252.2034 **Y** Order type C esta entre 302.448 y 74.372 y es aproximadamente igual a 245.429 **Y** Fiscal sector orders está entre 865.0 y 0.0 y es aproximadamente igual a 216.25 **Y** Orders from the traffic controller sector está entre 71772.0 y 11992.0 y es aproximadamente igual a 56827 **Y** Banking orders (1) es menor que 210508.0 y es aproximadamente igual a 158744 **Y** Banking orders (2) está entre 188411.0 y

16411.0 y es aproximadamente igual a 102411 **Y** Banking orders (3) está entre 73839.0 y 7679.0 y es aproximadamente igual a 69704 intensificado a la -0.2364

O

Week of the month es menor que 5.0 y es aproximadamente igual a 1 **Y** Day of the week es mayor que 2.0 y es aproximadamente igual a 6 **Y** Non-urgent order está entre 435.304 y 43.6511 y es aproximadamente igual a 288.4341 **Y** Urgent order es mayor a 77.37 y es aproximadamente igual a 223.2699 **Y** Order type A es mayor a 21.825999999999993 y es aproximadamente igual a 118.1780 **Y** Order type B es mayor que 25.1251 y es aproximadamente igual a 267.1055 **Y** Order type C esta entre 302.448 y 74.372 y es aproximadamente igual a 288.1932 **Y** Fiscal sector orders está entre 615.6143 y 0.0002] y es aproximadamente igual a 506.8359 **Y** Orders from the traffic controller sector es menor que 59045.3867 y es aproximadamente igual a 41881.9688 **Y** Banking orders (1) está entre 88048.7344 y 3452.0 y es aproximadamente igual a 16393 **Y** Banking orders (2) está entre 188411.0 y 16411.0625 y es aproximadamente igual a 148098.5 **Y** Banking orders (3) es menor que 54940.7578 y es aproximadamente igual a 3662 intensificado a la 0.0079.

Resultados del conjunto de datos Pima Indian Diabetes.

En las Tablas 4, 5 y 6 se presentan los resultados de la NN- ACFL para el conjunto de datos Pima.

Tabla 5: Valores de α, γ y m descubiertos para cada característica de Pima

GCLV	α	γ	m
Pregnancies	8.9572	0.4325	0.8029
Glucose	8.3436	0.5853	0.0172
Bloodpressure	9.7763	0.4806	0.8107
Skinthickness	4.7575	0.5239	0.8512
Insulin	6.646	0.1659	0.8846
Bmi	8.8779	0.4537	0.1168
Diabetespedigreefunction	9.4287	0.5304	0.4706
Age	8.3158	0.1826	0.0081

Tabla 5: Parámetros *w* descubiertos para la capa conjuntiva

Atributo	Predicado 1	Predicado 2	Predicado 3
	w	*w*	*w*
PREGNANCIES	5.72E-01	-3.87E-02	-5.87E-01
GLUCOSE	-1.86E-01	9.13E-01	1.58E+00
BLOODPRESSURE	2.57E-05	1.76E-03	4.98E-02
SKINTHICKNESS	2.91E-01	-2.18E-03	-2.48E-03
INSULIN	5.89E-02	-1.18E-05	-6.73E-04

BMI	-7.58E-01	3.18E-01	5.88E-01
DIABETESPEDIGREEFUNCTION	8.67E-01	-1.61E+00	-1.49E+00
AGE	-5.19E-01	7.32E-04	7.76E-06

Atributo	**Predicado 4**	**Predicado 5**
	w	*w*
PREGNANCIES	-3.10E-02	-2.65E-04
GLUCOSE	9.25E-01	-6.14E-01
BLOODPRESSURE	2.81E-01	1.68E-03
SKINTHICKNESS	-3.16E-03	6.77E-03
INSULIN	-4.07E-05	3.96E-02
BMI	2.55E-01	-1.07E+00
DIABETESPEDIGREEFUNCTION	-1.66E-01	-1.11E-01
AGE	3.26E-04	-1.54E+00

Tabla 6: Parámetros w descubiertos para la capa disyuntiva

	Predicado 1	**Predicado 2**	**Predicado 3**	**Predicado 4**	**Predicado 5**
w	0.5225	-1.46	-1.3504	-0.6472	1.0888

A partir de los datos encontrados y por medio del Algoritmo 4.3 y 4.4 se expresan los resultados en lenguaje natural:

Si

Pregnacies está entre 10.8076 y 3.8147e-06 y es aproximadamente igual a 8.5000 **Y** Glucose está entre 199.0 y 0.0 y es aproximadamente igual a 24.875 **Y** Blood pressure está entre 122.0 y 0.0 y es aproximadamente igual a 61 **Y** Skin Thickness está entre 68.0625 y 0.0 y es aproximadamente igual a 37.125 **Y** Insuline está entre 846.0 y 0.0 y es aproximadamente igual a 422.9996 **Y** BMI está entre 67.1 y 0.0 y es aproximadamente igual a 16.7750 **Y** Diabetes Pedigree está entre 1.4154 y 0.0780 y es aproximadamente igual a 1.2490 **Y** Age esta entre 72.0352 y 21.0 y es aproximadamente igual a 51 intensificado a la 0.5225

O

Pregnacies está entre 17.0 y 0.0 y es aproximadamente igual a 12.7500 **Y** Glucose está entre 199.0 y 4.5776e-05 y es aproximadamente igual a 111.9375 Y Blood pressure está entre 122.0 y 0.0 y es aproximadamente igual a 61 **Y** Skin Thickness está entre 99.0 y 0.0 y es aproximadamente igual a 74.25 **Y** Insuline está entre 846.0 y 0.0 y es aproximadamente igual a 423 **Y** BMI está entre 67.1 y 2.2888e-05 y es aproximadamente igual a 41.9375
Y Diabetes Pedigree es menor que 2.42 y aproximadamente igual a 0.0786 **Y** Age esta entre 81.0 y 21.0 y es aproximadamente igual a 73.5 intensificado a -1.46

O

Pregnacies está entre 17.0 y 0.0 y es aproximadamente igual a 14.875 **Y** Glucose está entre 199.0 y 4.5776e-05 y es aproximadamente igual a 111.9375 **Y** Blood pressure está entre 122.0 y 0.0 y es aproximadamente igual a 57.1875 **Y** Skin Thickness está entre 99.0 y 0.0 y es aproximadamente igual a 74.25 **Y** Insuline está entre 846.0 y 0.0 y es aproximadamente igual a 423 **Y** BMI está entre 67.1 y 2.2888e-05 y es aproximadamente igual a 41.9375 **Y** Diabetes Pedigree es menor a 2.42 y aproximadamente igual a 0.0786 **Y** Age esta entre 81.0 y 21.0 y es aproximadamente igual a 51 intensificado a -1.3504

O

Pregnacies está entre 17.0 y 0.0 y es aproximadamente igual a 12.75 **Y** Glucose está entre 199.0 y 4.5776e-05 y es aproximadamente igual a 111.937 **Y** Blood pressure es menor que 77.3781 y es aproximadamente igual a 57.1875 **Y** Skin Thickness está entre 99.0 y 0.0 y es aproximadamente igual a 74.2500 **Y** Insuline está entre 846.0 y 0.0 y es aproximadamente igual a 423 **Y** BMI está entre 67.1 y 2.2888e-05 y es aproximadamente igual a 41.9375 **Y** Diabetes Pedigree está entre 2.42 y 0.0779 y es aproximadamente igual a 0.2244 **Y** Age esta entre 81.0 y 21.0 y es aproximadamente igual a 66. Intensificado a -0.6472

O

Pregnacies está entre 17.0 y 0.0 y es aproximadamente igual a 12.75 **Y** Glucose está entre 86.3459 y 0.0 y es aproximadamente igual a 24.875 **Y** Blood pressure está entre 122.0 y 0.0 y es aproximadamente igual a 61 **Y** Skin Thickness está entre 99.0 y 0.0 y es aproximadamente igual a 24.75 **Y** Insuline está entre 846.0 y 0.0 y es aproximadamente igual a 422.999 Y BMI está entre 67.1 y 0.0 y es aproximadamente igual a 16.7750 **Y** Diabetes Pedigree está entre 2.42 y 0.0779 y es aproximadamente igual a 0.3707 **Y** Age esta entre 68.6074 y 21.0 y es aproximadamente igual a 51 intensificado a 1.0888