INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

"POR MI PATRIA Y POR MI BIEN"

TESIS

ALGORITMO DE DESCUBRIMIENTO DE CONOCIMIENTO BASADO EN LÓGICA DIFUSA ARQUIMEDIANA COMPENSATORIA Y REDES NEURONALES ARTIFICIALES.

Que para obtener el Grado de
Maestro en Ciencias de la Computación

Presenta
Ing. Edgar Ramiro Pedraza Hernández
G12071424

Director de Tesis
Dra. Laura Cruz Reyes

Codirector de Tesis
Dr. Rafael Alejandro Espín Andrade

Cd. Madero, Tamaulipas Mayo, 2021

Instituto Tecnológico de Ciudad Madero
Subdirección Académica
División de Estudios de Posgrado e Investigación

Cd. Madero, Tam. 18 de mayo de 2021

OFICIO No. : U.021/21
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS

C. EDGAR RAMIRO PEDRAZA HERNÁNDEZ
No. DE CONTROL G12071424
P R E S E N T E

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su Examen de Grado de Maestría en Ciencias de la Computación, se acordó autorizar la impresión de su tesis titulada:

"ALGORITMO DE DESCUBRIMIENTO DE CONOCIMIENTO BASADO EN LÓGICA DIFUSA ARQUIMEDIANA COMPENSATORIA Y REDES NEURONALES ARTIFICIALES"

El Jurado está integrado por los siguientes catedráticos:

PRESIDENTE:	DRA. CLAUDIA GUADALUPE GÓMEZ SANTILLÁN
SECRETARIO:	DR. NELSON RANGEL VALDEZ
VOCAL:	DRA. LAURA CRUZ REYES
SUPLENTE:	DRA. MARÍA LUCILA MORALES RODRÍGUEZ
DIRECTOR DE TESIS:	DRA. LAURA CRUZ REYES
CO-DIRECTOR DE TESIS:	DR. RAFAEL ESPÍN ANDRADE

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con usted el logro de esta meta. Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

A T E N T A M E N T E
Excelencia en Educación Tecnológica.
"Por mi patria y por mi bien".

EDUCACIÓN
INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

MARCO ANTONIO CORONEL GARCÍA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

c.c.p.- Archivo
MACG 'mdcoa'

Declaración de originalidad

Yo, Edgar Ramiro Pedraza Hernández, en mi calidad de autor manifiesto que este documento de tesis es producto original de mi trabajo y que no infringe derechos de terceros, tales como derechos de publicación, derechos de autor, patentes y similares. Por lo tanto, la obra es de mi exclusiva autoría y soy titular de los derechos que surgen de la misma.

Asimismo, declaro que en las citas textuales que he incluido 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 un tercero en cuanto a los derechos de autor sobre la obra en cuestión, asumiré toda la responsabilidad y relevo de ésta a mi director de tesis, así como al Tecnológico Nacional de México, Instituto Tecnológico de Ciudad Madero y a sus respectivas autoridades.

Cd. Madero, Tamaulipas. Febrero 2021.

ISC. Edgar Ramiro Pedraza Hernández

Resumen

El uso de datos dentro de la analítica de negocios es fundamental, sin embargo, es necesario procesarlos y modelarlos para que los tomadores de decisiones puedan prevenir situaciones que pongan en riesgo la empresa, planear estrategias para proyectar grandes ganancias, optimizar procesos, entra otras situaciones.

Existen diversas técnicas de analítica de datos, que permiten explotar este recurso para apoyar a la toma de decisiones. Un problema complejo es la generación de conocimiento debido a la dificultad de modelar expresiones tal como lo hace el pensamiento humano.

El presente trabajo de tesis tiene como objetivo generar conocimiento a partir de datos usando técnicas de inteligencia artificial con soporte de la Lógica Difusa Arquimediana Compensatoria que pertenece a la rama de la Computación blanda.

La propuesta se centra en la búsqueda de predicados lógicos, cuya estructura se representa con una red neuronal difusa. Este método denominado Genetic Algorithm Neural Network and Archimedean Compensatory Fuzzy Logic (GA-NN-ACFL); tiene la ventaja de la exactitud, que es una característica de las redes neuronales, y de interpretabilidad, que distingue a los métodos basados en lógica difusa.

La lógica difusa arquimediana compensatoria ha sido poco utilizada en la generación de conocimiento expresado en predicados lógicos. Hasta nuestro saber, ningún trabajo previo ha utilizado esta lógica con redes neuronales artificiales. una aportación complementaria es un método de clasificación de datos incorporado a GA-NN-ACFL.

Los resultados experimentales confirman que GA-NN-ACFL supera estadísticamente al trabajo previo basado en algoritmos evolutivos. En la evaluación se considero la calidad de los predicados y la exactitud de clasificación.

Abstract

The use of data within business analytics is essential, however, it is necessary to process and model it so that decision makers can prevent situations that put the company at risk, plan strategies to project large profits, optimize processes, among other situations. .

There are various data analytics techniques that allow this resource to be exploited to support decision-making. A complex problem is the generation of knowledge due to the difficulty of modeling expressions such as human thought does.

The objective of this thesis work is to generate knowledge from data using artificial intelligence techniques with the support of Compensatory Archimedean Fuzzy Logic that belongs to the branch of Soft Computing.

The proposal focuses on the search for logical predicates, whose structure is represented by a fuzzy neural network. This method called Genetic Algorithm Neural Network and Archimedean Compensatory Fuzzy Logic (GA-ANN-ACFL); It has the advantage of accuracy, which is a characteristic of neural networks, and of interpretability, which distinguishes methods based on fuzzy logic.

Compensatory Archimedean fuzzy logic has been little used in the generation of knowledge expressed in logical predicates. To our knowledge, no previous work has used this logic with artificial neural networks. a complementary contribution is a data classification method incorporated into GA-NN-ACFL.

The experimental results confirm that GA-NN-ACFL statistically outperforms previous work based on evolutionary algorithms. The evaluation considered the quality of the predicates and the accuracy of the classification.

Índice general

Índice de figuras

Índice de tablas

1. Introducción **1**
1.1. Antecedentes . . . 2
1.2. Problema de investigación . . . 2
1.3. Objetivos . . . 3
1.3.1. Objetivo general . . . 3
1.3.2. Objetivos específicos . . . 3
1.4. Justificación . . . 4
1.5. Alcances y limitaciones . . . 4
1.6. Organización del documento . . . 4

2. Marco Teórico **6**
2.1. Descubrimiento de conocimiento . . . 6
2.1.1. Fundamentos . . . 6
2.1.2. Papel de los métodos metaheurísticos . . . 8
2.1.3. IA Explicable: un balance entre interpretabilidad y precisión . . . 9
2.1.4. Medición de interpretabilidad y precisión . . . 10
2.2. Representación del conocimiento . . . 11
2.2.1. Fundamentos . . . 11
2.2.2. Tipos de razonamiento . . . 12
2.2.3. Tipos de representaciones . . . 13
2.3. Representación basada en reglas de lógica difusa . . . 14
2.3.1. Introducción a la Lógica Difusa . . . 14
2.3.2. Definición, estructura y propiedades de reglas difusas . . . 14
2.4. Representación basada en predicados de lógica difusa compensatoria . . . 17
2.4.1. Introducción a la CFL y su extensión Arquimediana ACFL . . . 17
2.4.2. Definición y estructura de predicado . . . 19
2.4.3. Variables lingüísticas . . . 20
2.4.4. Funciones de pertenencia de CFL y ACFL . . . 21
2.4.5. Operadores Lógicos de CFL y ACFL . . . 23
2.5. Representación basada en redes neuronales artificiales . . . 24
2.5.1. Descripción de una red neuronal artificial . . . 24
2.5.2. Aprendizaje en las redes neuronales artificiales . . . 25
2.5.3. Retro-propagación . . . 26
2.5.4. Redes Neuronales Difusas . . . 29
2.5.5. Incorporación y extracción de reglas . . . 30
2.6. Descubrimiento de conocimiento basado en predicados . . . 30

2.6.1. Evaluación de predicados . . . 30
2.6.2. Descubrimiento de predicados . . . 31
2.6.3. Inferencia para clasificación y pronostico . . . 31

3. Estado del arte **32**
3.1. Descubrimiento de conocimiento para la toma de decisiones usando lógica difusa compensatoria . . . 32
3.1.1. Algoritmos de optimización . . . 32
3.1.2. Sistemas de apoyo a la decisión . . . 33
3.2. Redes neuro-difusas . . . 34
3.3. Análisis y comentarios finales . . . 35

4. Metodología de solución **36**
4.1. Propuesta de solución . . . 36
4.2. Arquitectura de GA-NN-ACFL . . . 37
4.2.1. Descripción de las capas . . . 39
4.2.2. Representación de predicados . . . 42
4.3. Método de ajuste de Parámetros . . . 43
4.3.1. Algoritmo genético inicializador de parámetros . . . 44
4.3.2. Ajuste de parámetros de las funciones de pertenencia . . . 45
4.3.3. Ajuste de pesos . . . 47
4.4. Aprendizaje de clasificación con GA-NN-ACFL . . . 48
4.4.1. Conversion entre series temporales e instancias de clasificación . . . 49

5. Experimentación y resultados **51**
5.1. Diseño experimental . . . 51
5.1.1. Instancias de experimentación . . . 51
5.1.2. Configuración de GA-NN-ACFL. . . . 52
5.2. Comparación del algoritmo de referencia y GA-NN-ACFL en términos de valor de verdad y medida de exactitud . . . 52
5.3. Inferencias con Series de tiempo . . . 57
5.4. Experimento 3: Inferencia con In stancia de Clasificación. . . . 63
5.4.1. Conjunto de datos IRIS . . . 63
5.4.2. Conjunto de datos Cáncer de mama . . . 66

6. Conclusiones **69**
6.1. Conclusiones . . . 69
6.2. Contribuciones . . . 69
6.3. Trabajo futuro . . . 69

Referencias **71**

APPENDICES **75**

A. Experimentación con series temporales de Covid-19 **76**

B. ANEXO B **83**

Índice de figuras

1.1. Precisión vs Interpretabilidad. Diagrama de Venn intersección de ACFL y Redes neuronales. 2

2.1. Descripción del proceso KDD. Adaptado de (Timarán, 2009) 7
2.2. Precisión vs Interpretabilidad. Diagrama de Venn para varios modelos de Aprendizaje Automático. Inspirada en (Gunning, 2017). 10
2.3. 17
2.4. Estructura jerárquica de una variable lingüística. 20
2.5. Conexión sináptica entre neuronas. Imagen inspirada de (of Health et al., 2014). 24
2.6. Estructura de la neurona artificial. 25
2.7. Representación de capas en una red neuronal artificial. 25
2.8. Representación de la *L-esima* capa de una red neuronal. 27
2.9. Representación de neurona difusa. 29

4.1. Diagrama de flujo de GA-NN-ACFL propuesto para construir modelos de clasificación. 37
4.2. Arquitectura del núcleo de Eureka Universe 38
4.3. Arquitectura Genetic Algorithm Neural Network and Archimedean Compensatory Fuzzy Logic. 38
4.4. Neurona Difusa Conjunción ACFL 40
4.5. Representación de un predicado difuso en un árbol. Donde i es el elemento en el índice de 4.11 43

5.1. Personas Susceptibles al virus Covid-19 60
5.2. Personas Expuestas al virus Covid-19 60
5.3. Personas con padecimientos Moderados por virus Covid-19 61
5.4. Personas con afecciones severas al virus Covid-19 61
5.5. Personas en estado crítico con el virus Covid-19 61
5.6. Personas que se han recuperado del virus Covid-19 62
5.7. Muertes causadas por el virus Covid-19 62
5.8. Muertes Naturales, sin implicación del virus Covid-19 62
5.9. Tanto la Figura 5.9a y 5.9b mejoran la clasificación durante el entrenamiento, sin embargo 5.9b tiene un comportamiento oscilatorio en su entrenamiento . . . 63
5.10. Clasificación de Conjunto Iris, las lineas azules representan las clases del conjunto, mientras que las rojas la clasificación resultado del entrenamiento de EU-NN-ACFL 64
5.11. Representación en árbol del predicado difuso 65
5.12. Tanto la Figura 5.12a y 5.12b se muestra el descenso de error en los entrenamientos de la red. 67

5.13. Tanto la Figura 5.13a y 5.13b representa las perdiciones del ANEXO **??**. El eje Y representa la clase *Diagnosis* y el eje X es la entrada de registro del conjunto de prueba 67
5.14. Representación en árbol del predicado difuso 67

A.1. Personas confirmadas de tener el virus Covid-19 79
A.2. Personas negativas de tener el virus Covid-19 80
A.3. Personas con sospecha de tener el virus Covid-19 80
A.4. Personas muertas tras el virus Covid-19 80
A.5. Personas recuperadas de tener el virus Covid-19 81
A.6. Personas con el virus Covid-19 activo 81

Índice de tablas

2.1. Eventos científicos con propósito Interpretable 11
2.2. Funciones generadoras 23

3.1. Contraste del trabajo propuesto con el estado del arte 35

4.1. Representación de serie histórica 49
4.2. Representación de datos dependientes. 49
4.3. Extracto de serie de tiempo Fechas-Muertos por virus Covid-19 TMA 49
4.4. Reestructuración de Conjunto de datos serie de tiempo sobre muertos por el virus COVID-19 50

5.1. Descripción de las instancias 51
5.2. Parámetros de configuración 52
5.3. Valor de verdad en el algoritmo de referencia 53
5.4. Valor de verdad en algoritmo EU-NN-ACFL utilizando GCLV 54
5.5. Valor de verdad en algoritmo EU-NN-ACFL utilizando FPG 55
5.6. Comparativas de promedios de valores de verdad 55
5.7. Prueba de Friedman (nivel de significación de 0.05) 56
5.8. Ranking de los conjuntos de datos 56
5.9. Nivel de Friedman (nivel de significación 0.05) 56
5.10. Error de clasificación de Serie Histórica TMA-Covid-19 57
5.11. Error de clasificación de Serie Histórica TMA-Covid-19 58
5.12. Comparativa de errores de clasificación 58
5.13. Valor de verdad de Series Históricas TMA-Covid-19 59
5.14. Valor de verdad de Series Históricas TMA-Covid-19 59
5.15. Prueba Wilcoxon GCLV vs FPG (nivel de significancia 0.05) 63
5.16. Predicados del entrenamiento con GCLV 64
5.17. Predicados del entrenamiento con FPG 64
5.18. Parámetros de estados lingüísticos Specie de 5.16 64
5.19. Parámetros de estados lingüísticos Specie de 5.17 65
5.20. Valores de verdad y errores de entrenamientos de EU-NN-ACLF para el conjunto de datos de cáncer de mama. 66
5.21. Predicados del entrenamiento de conjunto Cáncer de mama con GCLV 68
5.22. Predicados del entrenamiento de conjunto Cáncer de mama con FPG 68
5.23. Valores de atributos de los predicados de la Tabla 5.22 68

A.1. Valor de verdad de Series Históricas 2 TMA-Covid-19 76
A.2. Valor de verdad de Series Históricas 2 TMA-Covid-19 77
A.3. Error de clasificación de Serie Histórica 2 TMA-Covid-19 78
A.4. Error de clasificación de Serie Histórica 2 TMA-Covid-19 79
A.5. Prueba de Wilcoxon(nivel de significancia de 0.05) 81

A.6. Muestra de Conjunto de datos de Series Histórica 2 de personas afectadas por Covid-19 82

Capítulo 1

Introducción

La analítica de negocios permite comprender el estado actual de una organización e identificar sus necesidades. Se enfoca al modelado de las representaciones basadas en la búsqueda o consultas; La analítica puede ser descriptiva cuando explora el pasado (mostrando a detalle lo ocurrido), de diagnóstico (mostrando las razones por las cuales ocurrieron dichas circunstancias), predictiva (lo que se avecina en adelante, posibles consecuencias), o prescriptiva (maneras de incurrir en el problema para que siga una trayectoria deseada).

Actualmente existen herramientas enfocadas en tratar de resolver algunas de las etapas del análisis de negocio que ayudan en la toma de decisiones a través del descubrimiento de conocimiento (Howson et al., 2018). Estas herramientas están soportadas principalmente por métodos de aprendizaje desarrollados por la comunidad de inteligencia artificial (IA), entre ellos: aprendizaje automático (Machine Learning y Deep Learning), manejo de grandes volúmenes de datos (Big Data), sistemas difusos (Sistemas de reglas difusas, Arboles de decisión difusos, etc.).

En particular, los métodos de redes neuronales de aprendizaje profundo han mejorado notablemente el estado del arte en muchos dominios de aplicación (LeCun, Bengio, y Hinton, 2015). A pesar del alto nivel de exactitud alcanzado, los resultados que se producen no permiten proporcionar justificaciones entendibles por un decisor, en otras palabras, no son suficientemente interpretables (Chakraborty et al., 2017).

Adicionalmente el grupo de trabajo, donde se desarrollará el tema de investigación propuesto, del cual la asesora de tesis es líder, cuenta con una herramienta denominada Eureka Universe, una plataforma digital de analítica de negocios; construida bajo el concepto de Lógica Difusa Compensatoria (CFL, por sus siglas en ingles de Compensatory Fuzzy Logic) por la comunidad de Eurekas (Andrade, Pérez, Ortega, Gómez, y Valdés, 2014; Meshino, 2008). Esta herramienta está basada en lógica de predicados y permite: a) construir y descubrir predicados, b) evaluar la veracidad de predicados; y aplicarlos en alguna tarea de descubrimiento de conocimiento. Los predicados son difusos y compensatorios, lo cual permite una mejor representación de condiciones realistas, así como una mejor interpretación de resultados.

Sin embargo, el contraste entre exactitud e interpretabilidad ha sido poco considerado en la conceptualización de Eureka Universe. En esta tesis se propone abordar el reto del desarrollo de un nuevo algoritmo para el descubrimiento de conocimiento que permita dar soluciones más precisas e interpretables. Principalmente se busca integrar la Lógica Difusa Arquimediana-Compensatoria (ACFL, por sus siglas en ingles) a un modelo de aprendizaje automático basado en redes neuronales dando lugar a un nuevo terreno como se muestra en la Figura 1.1. Se busca sentar las bases para la incorporación futura del aprendizaje profundo.

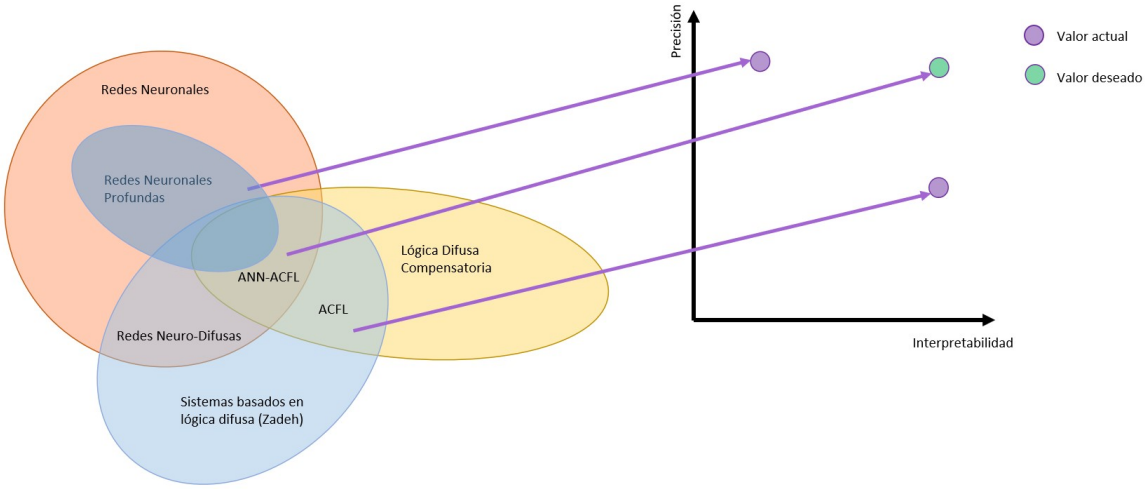


Figura 1.1: *Precisión vs Interpretabilidad. Diagrama de Venn intersección de ACFL y Redes neuronales.*

1.1. Antecedentes

Se espera que este trabajo tenga un impacto positivo en el desarrollo de Eureka Universe, puesto que contribuye al modelo actual que se planteó, ayudará en la tarea de descubrimiento de conocimiento de manera predictiva, trabajo el cual realiza actualmente de manera descriptiva y diagnostica. Es decir, este trabajo amplía un proyecto mayor, realizado por investigadores de tres universidades: la Universidad Autónoma de Nuevo León (UANL), la Universidad Autónoma de Chihuahua-Unidad Torres (UAdeC) y el Instituto Tecnológico Nacional de México, Tecnológico de Ciudad Madero (ITCM). Actualmente, Eureka Universe cuenta con un núcleo del cual se pretender formar parte este producto, así mismo, a la fecha, los trabajos de investigación que han sido realizados por investigadores del ITCM e incluyen:

- Uso de Lógica Difusa Compensatoria para el descubrimiento de conocimiento aplicado al problema Warehouse Order Picking para ordenamiento por lotes en tiempo real (Cruz-Reyes et al., 2019).
- Sistema inteligente para descubrimiento de conocimiento basado en lógica difusa compensatoria (Padrón, 2017).
- Algoritmo de virtual savant basado en lógica difusa compensatoria para problemas de empacado de objetos (Tesis de Master no publicada) (Padrón, 2020)
- Algoritmo evolutivo para descubrir conocimiento de asociación usando lógica difusa compensatoria (Llorente Peralta et al., 2019).

1.2. Problema de investigación

Para el desarrollo de este proyecto se busca generar conocimiento con base en predicados léxicos, que permiten encontrar altos grados de pertenencia bajo el concepto analizado. La formulación dada es la siguiente:

Dado:

Un conjunto de datos X formado por i objetos con j atributos, se busca generar conocimiento en un conjunto de predicados P, tal que cada predicado p es una variable de decisión que debe de satisfacer un mínimo valor de verdad σ.

Se busca:

$$maximizar\ C_{X_i \epsilon X}\ f(p, x_i),\ p > \ \sigma,\ p\ \epsilon\ \Omega \tag{1.1}$$

Donde:
p = es la variable de decisión correspondiente al predicado a evaluar.
$f(p_i, x_i)$ = valor de verdad del predicado p obtenido al evaluar el objeto i.
$g(x_{i,j})$ = Función de pertenencia asociada al atributo j del objeto i.
Ω es la región factible delimitada por un conjunto de reglas de construcción de un predicado p.

La región factible Ω está delimitada por el siguiente conjunto de reglas de construcción de un predicado p, expresadas mediante la notación Backus-Naur Form (BNF).

Predicado:=<Operación>|(<Operación>)

Operación:= <Operación_unaria>|(<Operación_unaria>)|
<Operación_binaria>|(Operación_binaria>)|<Operación_enearia>|(Operación_enearia>)

Operación_unaria:= **'NOT'** <Operando>| (**'NOT'** <Operando>)

Operación_binaria:=<Operador_binario><Operando><Operando>|
(<Operador_binario><Operando><Operando>)

Operador_binario:= **'IMP'|'EQV'**

Operador_enerario:= **'AND'|'OR'**

Operando:= $'g(x_{i,j})'$|<Predicado>

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar un Algoritmo de descubrimiento de conocimiento basado en Lógica Difusa Arquimediana Compensatoria y en la representación de predicados conjuntivos con redes neuronales.

1.3.2. Objetivos específicos

1. Desarrollar un algoritmo de descubrimiento de conocimiento, considerando el manejo de precisión con redes neuronales y la capacidad de interpretabilidad con predicados basados en Lógica Difusa Arquimediana-Compensatoria.

2. Contrastar el algoritmo propuesto con la versión previa basado en algoritmos genéticos, en términos de precisión e interpretabilidad.

3. Desarrollar un modulo de descubrimiento de conocimiento adecuado para casos de la salud.

1.4. Justificación

Con este trabajo se busca mostrar que la generación de conocimiento basado únicamente en la Lógica Difusa Compensatoria (CFL) no es suficiente para soportar los procesos de decisión solo por ser altamente interpretativa; también se requieren resultados de mayor precisión. Por ello, se propone incrementar la precisión de los métodos basados en la Lógica Difusa Arquimediana-Compensatoria mediante el uso de redes neuronales artificiales (RNA). Aunque el modelo de RNA es contrario a la estrategia tomada inicialmente para la herramienta Eureka Universe, se tiene la hipótesis que puede ser complementario.

Por otro lado, el cuestionamiento de la eficacia de la CFL en la toma de decisiones, como estrategia única e individual, puede resultar útil para desentrañar las distintas teorías de la interpretación que se manejan con asertividad mayor.

La implementación de un sistema generador de conocimientos basado en ACFL y RNA ayudará a la toma de decisiones al proporcionar un soporte más concreto en los resultados de descubrimiento de predicados.

1.5. Alcances y limitaciones

Alcances:

1. Encontrar un modelo de red neuronal entrenado con alta precisión de clasificación de datos.

2. Encontrar predicados de Lógica Difusa Arquimediana-Compensatoria con altos valores de verdad de la proposición universal para un conjunto de datos dado.

3. Se utiliza como punto de partida un algoritmo de referencia para la evaluación del propuesto(Llorente Peralta et al., 2019).

Limitaciones:

1. Las dimensiones de los conjuntos de datos analizadas no son de gran escala.

1.6. Organización del documento

A continuación, se describen la estructura en la que esta organizado esta tesis, así como una breve explicación de cada uno de los capítulos que lo componen.

El capítulo 2 contiene el marco teórico, dentro del cual se mencionan los principales conceptos que serán necesarios para entender las secciones posteriores y que son de suma importancia, los fundamentos del descubrimiento de conocimiento y como se representa, el papel de métodos heurísticos, la importancia entre el balance de interpretabilidad y precisión, representación de reglas y de predicados en la lógica difusa y lógica difusa compensatoria y su extensión arquimediana respectivamente, representación basada en redes neuronales artificiales, entre otros.

Por otro lado en el capítulo 3 se analiza el estado del arte, con el fin de buscar aportaciones realizadas al tema y hacer un enfoque a las diferencias existentes.

En el capitulo 4 se describen los métodos usados para el desarrollo de algoritmos que buscan satisfacer los objetivos planteados en la tesis. El capítulo 5 describe la implementación del algoritmo diseñado, y todo lo relacionado al proceso de evaluación realizado para el mismo mostrando finalmente los resultados obtenidos.

Finalmente, en el capítulo 6 se establecen las conclusiones de este trabajo, así como las aportaciones obtenidas y la mención a trabajos futuros que se pueden abordar mediante el desarrollo de la tesis.

Capítulo 2

Marco Teórico

Previamente a entrar en el trabajo expuesto en este proyecto, se necesita comprender varios conceptos, tales como descubrimiento de conocimiento, inteligencia artificial explicable, lógica difusa compensatoria, redes neuronales artificiales, predicados, entre otras. Esto con el objetivo de proveer al lector el entendimiento necesario de las acciones realizadas durante el proceso de diseño e implementación del algoritmo, ya que esta construido bajo estos conceptos.

2.1. Descubrimiento de conocimiento

De acuerdo con (Mining, 1996), y lo encontrado en la literatura, la *minería de datos* es la tarea principal en el *descubrimiento de conocimiento en base de datos* (KDD por sus siglas en ingles). El cual es un proceso automatizado donde se combinan las operaciones de descubrimiento y análisis. Consiste además de aplicar varias técnicas computacionales para extraer patrones de gran utilidad o conocimiento que se toma de los datos, típicamente se expresa en forma de un modelo predictivo o descriptivo.

2.1.1. Fundamentos

El proceso de extraer conocimiento en grandes volúmenes de datos es un tópico de investigación el cual es clave en los sistemas de base de datos y en la industria, presentando ser un área importante y que da la oportunidad de generar mayores ganancias (Llorente Peralta et al., 2019). Inicialmente el KDD fue definido como:

"KDD es *la extracción no trivial de implícitos, previamente desconocidos e información potencialmente útil de los datos.*"

por (Shapiro y Frawley, 1991). En 1996, la definición de KDD fue revisada de la siguiente manera por (Fayyad, Piatetsky-Shapiro, y Smyth, 1996):

"*El proceso no trivial de identificación de patrones válidos, novedosos, potencialmente útiles y fundamentalmente entendibles al usuario a partir de los datos.*"

A pesar de que la minería de datos es considerada la tarea principal, es indiscutible que la mayoría de las veces los datos necesiten un preprocesa-miento. De acuerdo con (Timarán, 2009) el proceso de KDD es interactivo e iterativo en involucrar al usuario en la toma de decisiones a través de un gran número de pasos, los cuales comprenden:

1. **Integración y recopilación de datos:** En esta parte se determinan las bases de datos a emplear en el proceso y de donde se obtendrán.

2. **Selección, limpieza, y transformación:** Se detectan los errores, información faltan te o perdida, construir nuevos atributos y enumerar y/o discretizar los atributos.

3. ***Minería de datos:*** En el proceso de minería de datos (DM por sus siglas en ingles), se produce nuevo conocimiento, para lograrlo se construye un modelo que describe los patrones y relaciones entre los datos que pueden usarse para hacer predicciones , para entender mejor los datos y explicar las situaciones pasadas, Esta sección incluye:
   - Elegir el tipo de modelo.
   - Determinar el tipo de tarea de minería mas apropiado.
   - Seleccionar el algoritmo de minería que resuelva la tarea y obtenga el modelo que se está buscando.
4. **Evaluación e interpretación:** Se presentan diferentes medidas de evaluación de los modelos, los cuales deben ser precisos, comprensibles e interesantes.
5. **Difusión:** Una vez que el modelo ha sido construido y validado, puede ser utilizado en una variedad de finalidades y deben ser evaluados los resultados obtenidos, de esta manera observamos si debe ser revaluado, rentrenado o reconstruido completamente.

La Figura 2.1 se describe el proceso de KDD, que es parte fundamentos del descubrimiento de conocimiento.

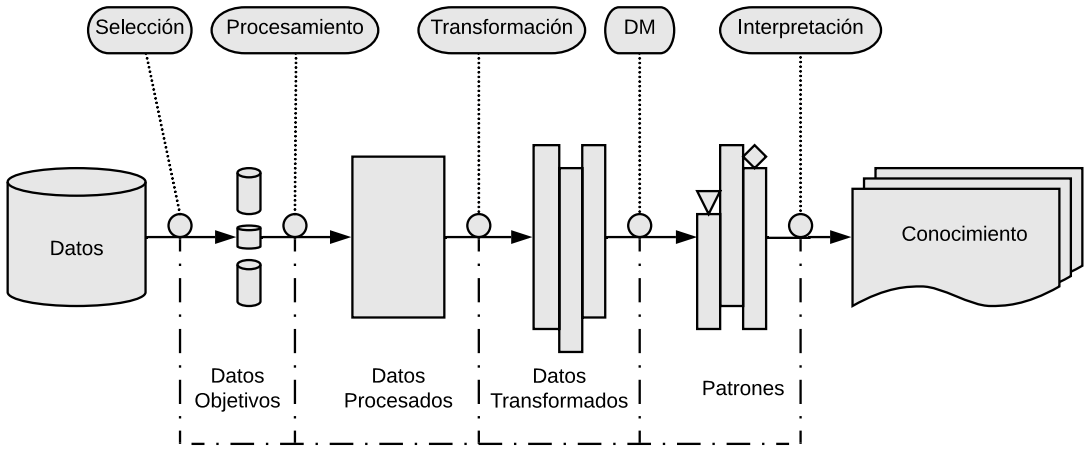


Figura 2.1: *Descripción del proceso KDD. Adaptado de (Timarán, 2009)*

La minería de datos es una parte muy importante del descubrimiento de conocimiento, y en gran parte de la literatura son tratados como si fueran lo mismo. El termino de minería de datos es utilizado frecuentemente por estadísticos, analistas de datos, y administradores de sistemas informáticos, así como en todo el proceso de descubrimiento de conocimiento. Mientras que el termino KDD es adoptado principalmente por especialistas en inteligencia artificial (Hand y Adams, 2014).

En (Frawley, Piatetsky-Shapiro, y Matheus, 1992) la definición de descubrimiento de conocimiento abarca una serie de criterios, connotaciones de términos que sugieren relevancia para el problema de KDD. Entre ellos la eficiencia sobre el descubrimiento, es de interés que el proceso de descubrimiento que se pueda implementar sea de una manera eficiente en una computadora. Un algoritmo es considerado eficiente si el tiempo de ejecución y el espacio utilizado son una función polinomial de bajo grado de la longitud de entrada. Por lo que el problema de descubrimiento de conocimiento interesante que satisface dados hechos es intrínsecamente difícil. De acuerdo con teorías de aprendizaje computacional (Valiant, 1984; Haussler, 1988) y estudios mas recientes (Gatica, Reyes, Contreras-Bolton, Linfati, y Escobar, 2016) han demostrado que no es posible aprender eficientemente un concepto booleano arbitrario; que el problema de descubrimiento de conocimiento (KD por sus siglas en ingles de Knowledge Discovery) es un problema *NP-Duro*.

2.1.2. Papel de los métodos metaheurísticos

(Treviño, 2007) menciona que para la resolución de problemas de optimización es posible emplear o hacer uso de algoritmos exactos o algoritmos aproximados. El inconveniente de hacer uso de algoritmos exactos es que, al resolver un problema con grandes instancias, puede resultar impráctico evaluar todas las soluciones generadas, ya que implica un gran consumo de recursos, ya sea la memoria utilizada o el tiempo necesario de procesamiento.

Por otra parte, los algoritmos aproximados son una mejor opción cuando lo que sea desea resolver son casos de instancias grandes de un problema que es considerado NP-Duro. debido a que estos algoritmos generan soluciones aproximadas sin el costo excesivo de recursos como el tiempo o memoria, pero sin la garantía de obtener la solución óptima.

Se ha señalado en (Cantú Cuéllar, Chávez Guzmán, y Cantú Mata, 2013) otras situaciones además de la mencionada donde se puedan aplicar algoritmos aproximados:

- No existe un algoritmo exacto para resolver el problema.
- No se requiere una solución óptima.
- Los datos con los que se cuenta no son fiables.
- Se requiere una rápida respuesta, a costa de la precisión.
- Las soluciones generadas son *punto de partida de algún otro algoritmo*.

Se han descrito que los algoritmos heurísticos de propósito especifico o deterministas tienen la característica de obtener la misma solución en diferentes ejecuciones (Treviño, 2007). A pesar de las situaciones anteriores o ventajas que proporciona el uso de algoritmos heurísticos, destaca el hecho de que el principal problema que presentan estos modelos, es su capacidad para escapar de los óptimos locales.

Para resolver este problema se introducen otros algoritmos de búsqueda mas inteligentes que eviten en la medida de lo posible quedar atrapado en óptimos locales. Estos algoritmos son denominados metaheurísticos. Por otro lado los algoritmos metaheurísticos son algoritmos aproximados de optimización y búsqueda de propósito general. Juegan un rol importante debido a que son procedimientos iterativos que guían una heurística subordinada combinando de forma inteligente distintos conceptos para explotar y explotar adecuadamente el espacio de búsqueda (Herrera, 2006).

Diversos autores en la literatura concuerdan que definir un marco general en el que definir una metaheurística resulta un poco complicado hoy en día, aunque se esta estudiando la manera de englobarlas a todas (Herrera, 2017; Rodríguez Ortiz et al., 2010). Por el momento se pueden clasificar de la siguiente manera: Atendiendo la inspiración, atendiendo al numero de soluciones, atendiendo la función objetivo, atendiendo la vecindad, atendiendo el uso de memoria.

Cualquiera de las alternativas por si solas, no son suficientemente finas como para permitir una separación clara entre todas las metaheurísticas. Generalmente, esas características se suelen combinar para permitir una clasificación mas elaborada.

Sin embargo conforme al teorema de *NFL* (*No Free Lunch Theorem*), que demuestra que al mismo tiempo que una metaheurística es muy eficiente para una colección de problemas,

es muy ineficiente para otra colección), los métodos generales de búsqueda, entre los que se encuentran las metaheurísticas, se comportan exactamente igual cuando se promedian sobre todas las funciones objetivo posibles, de tal forma que si un algoritmo A es más eficiente que un algoritmo B en un conjunto de problemas, debe existir otro conjunto de problemas de igual tamaño para los que el algoritmo B sea más eficiente que el A. Esta aseveración establece que, en media, ninguna metaheurística (algoritmos genéticos, búsqueda dispersa, búsqueda tabú, etc.) es mejor que la búsqueda completamente aleatoria. Una segunda característica que presentan las metaheurísticas es que existen pocas pruebas sobre su convergencia hacia un óptimo global; es decir, que a *priori* no se puede asegurar ni que la metaheurística converja, ni la calidad de la solución obtenida. Por último, las metaheurísticas más optimizadas son demasiado dependientes del problema o al menos necesitan tener un elevado conocimiento heurístico del problema. Esto hace que, en general, se pierda la generalidad original con la que fueron concebidas. (Adam, Alexandropoulos, Pardalos, y Vrahatis, 2019).

2.1.3. IA Explicable: un balance entre interpretabilidad y precisión

Existen herramientas que están soportadas principalmente por métodos de aprendizaje de conocimiento, desarrollados por la comunidad de inteligencia artificial (IA), entre los cuales el aprendizaje automático se ha vuelto frecuentes en nuestros tiempo. Estas herramientas son útiles en la toma de decisiones a través del descubrimiento de conocimiento que estas generan (Howson et al., 2018). La demanda de IA predecible y responsable crece a medida que las tareas con mayor sensibilidad e impacto social se confían mas comúnmente a los servicios de IA. Por lo que, la claridad en como se maneja la información en el algoritmo es un factor clave para mantener organizaciones responsables, es decir, que adquieran la necesidad de ser responsables con sus productos , servicios y comunicación de información.

Los sistemas de Inteligencia Artificial Explicables (XAI por sus siglas en ingles) son una solución para la IA, lo que hace posible al explicar los procesos y la lógica de la toma de decisiones de IA para los usuarios finales (Gunning, 2017). Específicamente, estos algoritmos explicables tienen la posibilidad de permitir el control y la supresión en caso de situaciones adversas o no deseadas, situaciones como la toma de decisiones sesgada o la discriminación social. Un sistema XAI se puede definir como un sistema inteligente auto explicativo que describe el razonamiento detrás de sus decisiones y predicciones.

Se ha enfatizado que hay dos criterios principales en el descubrimiento de conocimiento que cualquier profesional debe considerar cuando determina qué tipo de técnicas de aprendizaje automático (ML, por sus siglas en inglés) se debe utilizar para al abordar regresión, clasificación, o problemas de decisión: precisión e interpretabilidad. La Figura 2.2 muestra este hecho con una compensación por algunas técnicas de ML bien conocidas, y el desafío actual de aumentar la interpretabilidad sin reducir la precisión (Fernandez, Herrera, Cordon, del Jesus, y Marcelloni, 2019). Para los modelos más interpretables, el desafío es aumentar ambos ejes.

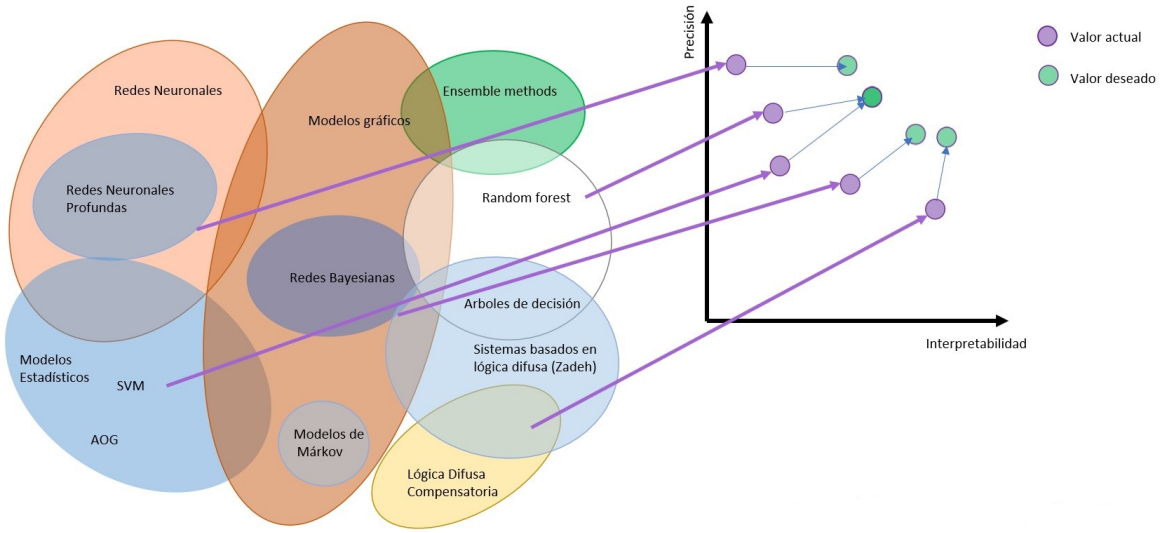


Figura 2.2: *Precisión vs Interpretabilidad. Diagrama de Venn para varios modelos de Aprendizaje Automático. Inspirada en (Gunning, 2017).*

En particular, los métodos de aprendizaje profundo han mejorado notablemente el estado del arte en muchos dominios de aplicación (LeCun et al., 2015). A pesar del alto nivel de exactitud alcanzado, los resultados que se producen no permiten proporcionar justificaciones entendibles por un decisor, en otras palabras, no son suficientemente interpretables (Chakraborty et al., 2017). En las circunstancias antes mencionadas, solo se requiere conocer lo *que* se predijo . Sin embargo, en la mayoría de los casos , también es muy importante saber *¿porqué?* se hizo una determinada predicción, por que una correcta predicción solo resuelve parcialmente el problema original (Carvalho, Pereira, y Cardoso, 2019).

Por otro lado la precisión de los resultados en los modelos de IA son altos y definidos como "*Precisión se refiere a la capacidad del modelo para expresar fielmente el sistema real. Sería mejor si el modelo entrenado es mas similar al sistema real*"según (He, Ma, y Wang, 2020).

2.1.4. Medición de interpretabilidad y precisión

(Carvalho et al., 2019) plantea que contemplando la interpretabilidad en el ML como dos caminos de comunicación entre dos entidades, los humanos como usuarios y la computadora como el interprete de los modelos de ML, uno puede considerar que este es un campo de investigación que al menos pertenece a tres áreas generales de investigación en ciencia: la ciencia de datos, ciencia humana, interacción humano-computadora. Esto no solo significa que la interpretabilidad del ML puede tener contribuciones de diferentes área, si no además que una de las áreas puede mejorar la investigación de otras. El problema de interpretabilidad solo sera posible mediante una investigación verdaderamente interdisciplinaria, uniendo la ciencia de datos, con las ciencias humanas, incluida la filosofía y psicología cognitiva.

Además del rápido crecimiento del volumen de investigación sobre interpretabilidad (Doshi-Velez y Kim, 2017), el creciente interés en ML, la interpretabilidad también se ha reflejado en numerosos eventos científicos, Algunos ejemplo de sesiones y talleres sobre el tema en las principales conferencias se presentan en la Tabla 2.1.

Tabla 2.1: *Eventos científicos con propósito Interpretable*

Nombre	Año
Fairness, Accountability, and Transparency in Machine Learning (NIPS, ICML, DTL, KDD) (FAT/ML, 2014)	2014-2018
ICML Workshop on Human Interpretability in Machine Learning (WHI) (B. Kim, Malioutov, Varshney, y Weller, 2017)	20016-2018
NIPS Workshop on Interpretable Machine Learning for Complex Systems (Wilson, Kim, y Herlands, 2016)	2016
NIPS Symposium on Interpretable Machine Learning (Wilson, Yosinski, Simard, Caruana, y Herlands, 2017)	2017
XCI: Explainable Computational Intelligence Workshop (Pereira-Fariña y Reed, 2017)	2017
IJCNN Explainability of Learning Machines (IJCNN, 2017)	2017
IJCAI Workshop on Explainable Artificial Intelligence (XAI) (IJCAI, 2017, 2018)	2017-2018
Ünderstanding and Interpreting Machine Learning in Medical Image Computing Applications"(MLCN, DFL, and iMIMIC) workshops (Stoyanov et al., 2018)	2018
IPMU 2018—Advances on Explainable Artificial Intelligence (IPMU, 2018)	2018
CD-MAKE Workshop on explainable Artificial Intelligence (Holzinger, Kieseberg, Tjoa, y Weippl, 2019; CD-MAKE, 2018)	2018-2019
Workshop on Explainable Smart Systems (ExSS) (Lim, Smith, y Stumpf, 2018; Lim, Sarkar, Smith-Renner, y Stumpf, 2019)	2018-2019
ICAPS—Workshop on Explainable AI Planning (XAIP) (ICAPS, 2018; Hoffmann y Magazzeni, 2019)	2018-2019
AAAI-19 Workshop on Network Interpretability for Deep Learning (Zhang Q, 2019)	2019
CVPR—Workshop on Explainable AI (CVPR, 2019)	2019

La precisión se mide en diversos algoritmos de distintas maneras, en general va de acuerdo con la definición de (He et al., 2020) donde se refiere a la capacidad del modelo para ajustarse fielmente al sistema real. Algunos estudios comparativos sobre distintos modelos clasificadores, y según (Nikam, 2015) donde se compararon la precisión de 74 técnicas de clasificación, no existe clasificador que supere a los demás en todas las situaciones.

2.2. Representación del conocimiento

En el proceso de la toma de decisiones el cual es un problema que se soluciona en base a la información que se conoce y la representación del conocimiento, es un concepto fundamental que se revisa en diversos campos de la inteligencia artificial, ya que el conocimiento que adquieren los diversos modelos debe ser representado por los programas que razonan.

Por lo tanto para poder operar con el conocimiento es necesario representarlo y poder hacer un modelo del mundo sobre el cual se quieren hacer inferencias. De esta manera surgen así distintos métodos de representación del conocimiento en forma simbólica, como: *La lógica de primer orden, marcos o plantillas, las redes semánticas, interpretaciones probabilísticas*, entre algunas otras.

2.2.1. Fundamentos

Y como se ha indicado al comienzo de esta sección mas certeramente en la Sección 2.1.3, surge la necesidad de tener una excelente representación del conocimiento, es decir, la necesidad de la interpretabilidad surge de un escenario incompleto en la formalización de problemas. Además, esta falta de exhaustividad en la especificación de los problemas se puede mostrar

en diferentes escenarios , algunos de los cuales se incluyen los siguientes (Doshi-Velez y Kim, 2017):

- Seguridad: porque el sistema nunca se puede probar por completo, ya que no se puede crear una lista completa de escenarios en los que el sistema puede fallar.
- Ética: porque la noción humana de, por ejemplo, justicia puede ser demasiado abstracta para codificarse por completo en el sistema.
- Objetivos no cocientes: porque el algoritmo puede estar optimizando un objetivo incompleto, es decir, una definición indirecta del objetivo final real.
- Compensación de objetivos múltiples: dos desideratas bien definidos en los sistemas de aprendizaje automático pueden competir uno contra otro, como la privacidad y la calidad de predicciones (Rüping et al., 2006).

Una de las razones detrás de este problema de la representación del conocimiento no resuelto, es que la interpretabilidad es un concepto muy subjetivo, por lo tanto, difícil de formalizar (Rüping et al., 2006). Además, la interpretabilidad es una noción de dominio especifico (Rüping et al., 2006; Freitas, 2014), por lo que no puede haber una definición para todo uso(Rudin, 2018). De acuerdo con (Carvalho et al., 2019) no existe una definición matemática de interpretabilidad. Una definición (no matemática) dada por Miller es "*La interpretabilidad es el grado en que un ser humano puede comprender la causa de una decisión*"(Miller, 2019). En el contexto de los sistemas de ML, (B. Kim, Koyejo, Khanna, et al., 2016) describe la interpretabilidad como "*el grado de que un humano puede predecir consistentemente el resultado del modelo*". Esto significa que la interpretabilidad de un modelo es mas alto, si es mas fácil para una persona razonar y rastrear por qué la predicción fue hecha por el modelo. Comparativamente, un modelo es mas interpretable que otro si sus primeras decisiones son mas fáciles de entender que las sus ultimas decisiones (Miller, 2019).

Mas recientemente, (Doshi-Velez y Kim, 2017) definen la interpretabilidad como la "*capacidad de explicar o presentar en términos comprensibles para un ser humano*". (Molnar, 2020) señala que "*el ML interpretable se refiere a los métodos y modelos que hacen que el comportamiento y las predicciones de los sistemas de ML comprensibles para los humanos*". En consecuencia, la representación de conocimiento esta evidentemente relacionada con la capacidad de que tan bien lo humanos captan cierta información al mirarla y razonarla sobre ella.

2.2.2. Tipos de razonamiento

Los motores de inferencia de las distintas técnicas de IA pueden encadenar las reglas de diversas maneras: es lo que se denomina el **método de razonamiento**. Los dos principales métodos de razonamiento son el razonamiento deductivo y el razonamiento inductivo, aunque existen motores que poseen un método de razonamiento mixto.

Razonamiento deductivo

Un motor de razonamiento deductivo también se denomina motor de inferencia dirigido por los datos. De acuerdo con (Mathivet, 2018) en este tipo de motor se parte de los datos disponibles en la base de hechos , y se comprueba para cada regla si se puede aplicar o no. Si se puede, se aplica y se agrega a la conclusión de la base de hechos.

El motor explora, entonces, todas las posibilidades, hasta encontrar el hecho buscado o hasta que no pueda aplicar nuevas reglas.

Razonamiento inductivo

Un motor de inferencia de razonamiento inductivo también se denomina dirigido por objetivos.

En este caso, se parte de los hechos que se desea obtener y se busca alguna regla que permita obtener dicho hecho. Se agregan todas las premisas de esta regla en los nuevos objetivos a alcanzar.

(Mathivet, 2018) menciona: vuelve a iterar, hasta que los nuevos objetivos a alcanzar estén presentes en la base de hechos, Si un buen hecho esta ausente en la base de hechos o se ha probado como falso, entonces se sabe que las reglas no se pueden aplicar, Estos motores poseen, por tanto, un mecanismo (backtracking) que les permite pasar a una nueva regla, que sera un nuevo medio de probar el hecho.

De acuerdo con (Mora-Flórez, Echaverri, y Castañeda, 2005) El proceso de razonar el conocimiento y convertirlo en un formato en particular es denominado *representación de conocimiento"*. Una vez el conocimiento ha sido representado correctamente puede utilizarse en un sistema inteligente que con el empleo de herramientas de análisis, tratamiento y manipulación automática tiene la capacidad de inducir o deducir nuevo conocimientos.

2.2.3. Tipos de representaciones

Los tipos de representaciones del conocimiento están dados por las estructuras generadas por las distintas técnicas de la IA y un ejemplo son las mostradas en la Figura 1.1, las estructuras que representan este conocimiento es denominado formalización de conocimiento (Pérez, 1994).

Representación del conocimiento con métodos simbólicos

Los primeros sistemas de IA resolvían problemas definidos con una representación simbólica, la cual permite que un algoritmo sea capaz de operar sobre ella y a la vez generar una representación simbólica como solución. Los métodos simbólicos en IA siguen este principio; se fundamentan en técnicas de representación simbólica de conocimiento asociadas a mecanismos con capacidad de inferir soluciones y nuevos conocimientos a partir del conocimiento representado (Osório, 1998). Entre los principales métodos simbólicos están los que se enumeran (Mora-Flórez et al., 2005) a continuación:

1. Árboles de decisión.
2. Sistemas expertos.
3. Lógica Difusa.
4. Sistema basado en casos
5. Agentes inteligentes
6. Métodos basados en los primeros principios o basados en el modelo (MBM)

Representación del conocimiento con métodos conexionistas

Este tipo de métodos puede caracterizar la representación y la adquisición del conocimiento y técnicas de aprendizaje a partir del conocimiento empírico (Alur, Dang, y Ivančić, 2006).

1. Redes Neuronales Artificiales
2. Algoritmo de aprendizaje para el análisis de datos multivariables –LAMDA
3. Maquinas de soporte vectorial (SVM)

2.3. Representación basada en reglas de lógica difusa

La lógica es una de las disciplinas más antiguas de la historia humana, ha habido grandes personajes que han hecho estudios sobre la misma. Entre ellos Leibniz, Boole, Russell, Turing, y muchos otros; a la fecha aún es una disciplina sobre la cual se siguen realizando investigaciones.

De acuerdo con (Llorente Peralta et al., 2019) En la actualidad se realiza más investigación usando la lógica como base de los procesos computacionales, algunos de los objetivos de estos estudios han sido probar teoremas matemáticos, validar diseños de ingeniería, diagnosticar fallas, codificar y analizar leyes, regulaciones y reglas comerciales. El uso de la metodología lógica por parte de la ingeniería de software es llamada programación lógica.

2.3.1. Introducción a la Lógica Difusa

La Lógica Difusa (LD) 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*"(Siler y Buckley, 2005). En la lógica difusa se hace uso de los conceptos de la lógica y de los conjuntos de *Lukasiewicz* (gobierna los principios de implicación y equivalencia en una lógica trivaluada) mediante la definición de grados de pertenencia (Llorente Peralta et al., 2019).

La LD parte de la idea de que la forma en que se construye el pensamiento humano no es mediante números, si no sobre *etiquetas lingüísticas*. De esta manera se representa el conocimiento común, que es mayoritariamente del tipo lingüístico cualitativo y no necesariamente cuantitativo, mediante un lenguaje matemático a través de conjuntos difusos y funciones características asociadas a ellos. Los términos lingüísticos son inherentemente menos precisos que los datos numéricos, pero expresan el conocimiento en términos más asequibles para la comprensión humana (Siler y Buckley, 2005).

2.3.2. Definición, estructura y propiedades de reglas difusas

Las proposiciones de lógica difusa son afirmaciones medidas por un valor de verdad que se relaciona con el uso de límites definidos bajo el intervalo [0,1], estás expresan ideas bajo la percepción de cada individuo (Ross, 2010) y son representadas por declaraciones lingüísticas que se acercan al lenguaje natural por lo que tiende a ser vago e impreciso.

Las proposiciones difusas son asignadas a conjuntos difusos. Dado λ

$$T(\lambda) = \mu U(x);\ Donde\ 0 \leq \mu U \leq 1 \tag{2.1}$$

En la Ecuación 2.1 se indica el grado de verdad de la proposición $\lambda : x\ \epsilon\ U$ que es equivalente al grado de pertenencia de x dentro de un conjunto difuso μ. Para poder tener a μ es necesario pasar los valores x por las funciones de pertenencia, dando lugar a las variables lingüísticas Sección 2.4.3. T es una lógica Arquimediana.

Una Lógica Arquimediana es un trío (c, d, n) de operadores , donde c es una norma triangular (t-norma) Arquimediana continua , d es su correspondiente conorma triangular (t-conorma) Arquimediana continua y n es su operador de negación. Cada una de ellas, T_c y T_d presentan las siguientes propiedades.

Propiedades de t-norma
Una t-norma es una función $T : [0, 1]^2 \to [0, 1]$, que satisface a los siguientes axiomas:

$T(a, b) = T(b, a)(T1)$	**Conmutatividad**
$T(a, b) \leq T(c, d)$	**Monotonicidad**
Si	
$a \leq c$	
$b \leq d(T2)$	
$T(a, T(b, c)) = T(T(a, , b), c)(T3)$	**Asociatividad**
$T(a, 1) = a(T4)$	**Uno como identidad**

Propiedades de t-conorma
Una t-conorma es una función $S : [0, 1]^2 \to [0, 1]$, que satisface a los siguientes axiomas:

$S(a, b) = S(b, a)ST1)$	**Conmutatividad**
$S(a, b) \leq S(c, d)$	**Monotonicidad**
Si	
$a \leq c$	
$b \leq d(S2)$	
$S(a, S(b, c)) = S(S(a, , b), c)(S3)$	**Asociatividad**
$S(a, 1) = a(S4)$	**Uno como identidad**

Hay una propiedad clásica que coincide con la t-norma y t-conorma, se refiere a las leyes de Morgan.

$\neg T(a, b) = S(\neg a, \neg b) y \neg S(a, b) = T(\neg a, \neg b)$ y el operador de negación usual es $\neg a = 1 - a$

Una t-norma $T : [0, 1]^2 \to [0, 1]$ es una t-norma continua si y unicamente si para cada par de secuencias convergentes $\{x_n\}_{n=1}^{\infty}$ y $\{x_n\}_{n=1}^{\infty}$

Propiedades de la lógica

1. Cada t-norma $T : [0, 1]^2 \to [0, 1]$, satisface $T(x, y) \leq min\{x, y\}$
2. Cada t-conorma $S : [0, 1]^2 \to [0, 1]$, satisface $S(x, y) \geq max\{x, y\}$
3. $min(x, y)$ es la unica t-norma continua idempotente y la máxima función de la familia de operadores t-norma.

4. $max(x, y)$ es la única t-conorma continua idempotente y la mínima función de la familia de operadores t-conorma.

5. Una consecuencia de las propiedades precedentes es que cada t-norma (t-conorma) pertenece a solo uno de dos subconjuntos, el singleton de la t-norma mínima (la t-conorma máxima) o el conjunto de normas.

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

$$T(x_1, ..., x_n) = f^{(-1)}(\sum_{i=1}^{n} f(x_i)) \tag{2.2}$$

   Donde:

$$f^{(-1)}(z) = \begin{cases} f^{-1}(z) & \text{si } z \in [0,\text{f}(0)] \\ 0 & \text{si } (f(0), +\infty) \end{cases} \tag{2.3}$$

7. Existe una propiedad equivalente para las t-conormas.

Sea n un operador de negación de $[0, 1]$ a $[0, 1]$ o un operador estrictamente decreciente que cumple $n(n(x) = x, n(0) = 1$ y $n(1) = 0$.

Reglas difusas

Ejemplo de proposiciones difusas compuestas:

$$X \text{ es } A \; o \; X \text{ no es } B$$
$$X \text{ es } A \; y \; X \text{ es } B$$
$$X \text{ no es } A \; y \; X \text{ no es } B$$
$$(X \text{ es } A \; y \; X \text{ no es } B) \; o \; X \text{ es } C$$
$$X \text{ es } A \text{ y } Y \text{ es } D$$

- En una proposición difusa compuesta pueden estar implicadas variables distintas.
- Las proposiciones difusas compuestas pueden considerar relaciones difusas.
- ¿Cómo determinamos la interpretación de estas relaciones difusas?¿Cómo determinamos la función de pertenencia?
  - Para las conectivas "y"se utilizan intersecciones difusas:

$$X \text{ es } A \; y \; Y \text{ es } B$$
$$\mu_{A \cap B}(x, y) = T_c[\mu_A(x), \mu_B(y)]$$

  - Para los conectivas "o"se deben utilizar uniones difusas:

$$X \text{ es } A \; o \; Y \text{ es } B$$
$$\mu_{A \cup B}(x, y) = T_d[\mu_A(x), \mu_B(y)]$$

  - Para las conectivas "no"se deben utilizar complementos difusos.

Para obtener μ se determina pasando un valor de entrada por una función de pertenencia. Una función de pertenencia generalmente se nombran como $\mu(X)$ ó $M(X)$. Hay ciertas funciones típicas que siempre suelen usar, tanto por la facilidad de computación que su uso conlleva, como por la estructura lógica para definir un valor lingüístico asociado. Dichas funciones de

pertenencia pueden tener asociado diferentes estructuras: rectas, triangulares, sigmoideas, en Z, gaussianas, entre otras (Llorente Peralta et al., 2019).

Lo que nos indica la función de pertenencia en un conjunto, es el grado en que cada elemento de un universo dado, pertenece a dicho conjunto como se menciona en la Ecuación 2.1. En otras palabras las funciones de pertenencia son una forma de representar gráficamente un conjunto difuso sobre un universo como se ilustra en la Figura 2.3.

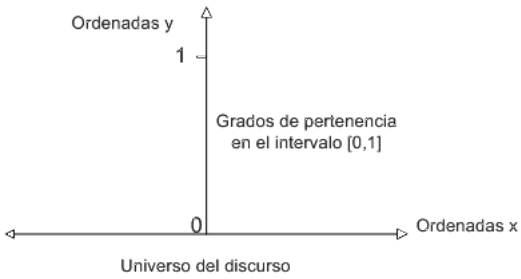


Figura 2.3

2.4. Representación basada en predicados de lógica difusa compensatoria

La lógica difusa compensatoria (CFL) surgió con un enfoque axiomático transdisciplinario, diferente del enfoque de Norma y Conorma para mejorar la interpretabilidad mediante el lenguaje natural. La CFL propone la combinación de lógicas multivalentes soportado por un conjunto de axiomas descritos en el la Sección 2.4.1, basado en preferencias expresadas a través de frases complejas que incluyen mas de una etiqueta lingüística (Andrade, González, y Caballero, 2014). La CFL esta conformado en un sistema por un cuarteto de operadores: un operador de conjunción, un operador de disyunción, un operador de negación, y un operador de orden estricto, que satisfacen axiomas pertenecientes a la lógica y la teoría de toma de decisión (Andrade, Pérez, et al., 2014).

Se introdujo la Lógica Difusa Arquimediana-Compensatoria (ACFL, por sus siglas en ingles de Archimedean-Compensatory Fuzzy Logic) armonizando conjuntos de la CFL y del enfoque Noma y Conorma de una Lógica Arquimediana, de conectivos por propiedades y diferentes interpretaciones de los valores de verdad.

2.4.1. Introducción a la CFL y su extensión Arquimediana ACFL

En la CFL se renuncia al cumplimiento de las propiedades clásicas de la conjunción y la disyunción, contraponiendo a éstas la idea de que el aumento o disminución del valor de verdad de conjunción o disyunción provocadas por el cambio del valor de verdad de una de sus componentes, pueda ser "*compensado*" con la correspondiente disminución o aumento de la otra (Llorente Peralta et al., 2019). Esta noción hace que la CFL sea una lógica sensible. Existen caso en los que la compensación no es posible. Esto ocurre cuando son sobrepasados ciertos umbrales y existe un veto que impide la compensación (Llorente Peralta et al., 2019).

La CFL esta formada por un cuarteto de operadores continuos antes mencionados conjunción (c),disyunción (d), orden estricto difuso (o) y negación (n), que satisfacen el siguiente grupo de axiomas:

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

2. **Axioma de conmutatividad o simetría:**
   $c(x_1, x_2, ..., x_n) = c(x_1, x_2, ..., x_n)$

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

4. **Axioma del veto:**
   Si $x_i = 0$ para algún i entonces $c(x) = 0$

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

6. **Axioma de transitividad difusa:**
   Si $o(x, y) \leq 0.5$ y Si $o(x, y) \leq 0.5$ entonces $o(x, y) \leq max(o(x, y), o(y, x)$

7. **Leyes de Morgan:**
   $n(c(x_1, x_2, \ldots, x_n)) = d(n(x_1), n(x_2), ..., n(x_n))$
   $n(d(x_1, x_2, \ldots, x_n)) = c(n(x_1), n(x_2), \ldots, n(x_n))$

Donde:
x: es una variable lingüística existente en un conjunto.
i: es la *i-esima* variable.
j: es la *j-esima* variable.
n: es la última variable del conjunto.

La Lógica Difusa Arquimediana-Compensatoria (ACFL) consiste en una Norma triangular Arquimediana generada por una función, que sirven como generadores de operadores compensatorias. Además, incluyen el operador de negación mas habitual y una orden difusa.

En (Espin-Andrade, González Caballero, Pedrycz, y Fernández González, 2015) demuestra cómo ambos enfoques pueden ser teóricamente compatibles. Es decir, explican que el cuarteto formado por un predicado universal y un predicado existencial basado en una t-norma continua de Arquimediana y sus dos predicados existenciales universales y uno basado en un sistema compensatorio, aumenta el poder para modelar la toma de decisiones. En esta lógica, se busca la compatibilidad de dos sistemas de lógica difusa y demostrando que para cada lógica continua de Arquimediana basada en una t-norma y t-conorma y su lógica difusa compensatoria compatible, existe el mismo orden para el valor de verdad de los predicados.

El cuantificador universal Arquimediano y su cuantificador existencial correspondientes representan la tendencia a no refutar una toma de decisiones representada por un predicado, respectivamente. De manera equivalente para la Lógica Difusa Compensatoria, el par formado por un cuantificador universal existencial CFL representa la afirmación.

Por lo que entonces la ACFL es una combinación teórica de dos lógicas diferentes, L^1 para la lógica Arquimediana y L^2 para la CFL. De esta manera se vinculan las definiciones clásicas de T-norma y T-conorma con la lógica difusa compensatoria.

En (Espin-Andrade et al., 2015) se tiene que una lógica $L = (L_c^1, L_d^1, L_c^2, L_d^2)$ es una ACFL si cuenta con los siguientes operadores lógicos:

1. $L_c^1 : [0,1]^2 \to [0,1]$ es una t-norma arquimediana generada por una función f que satisface la Ecuación 2.2
2. $L_d^1 : [0,1]^2 \to [0,1]$ satisface $L_d^1(x_1, x_2) = 1 - L_c^1(1-x_1, 1-x_2)$ para toda $x = (x_1, x_2)\, \epsilon [0,1]^2$
3. $L_c^2 : [0,1]^n \to [0,1]$ que satisface $M_f(x_1, ..., x_n) = f^{(-1)}(\sum_{i=1}^{n} f(x_i))$
4. $L_d^2 : [0,1]^n \to [0,1]$ donde para un vector $x = (x_1, x_2, ..., x_n)\ \epsilon\ [0,1]^2$, $L_d^2(x_1, x_2, ..., x_n) = 1 - L_c^2(1 - x_1, 1 - x_2, ..., 1 - x_n)$ y $L_c^2(x_1, x_2, ..., x_n) = 1 - L_d^2(1 - x_1, 1 - x_2, ..., 1 - x_n)$
5. $o : [0,1]^2 \to [0,1]$ es el orden difuso.
6. $n : [0,1] \to [0,1]$ es el operador de negación con la ecuación $n(x) = 1 - x$.

Teniendo a L como una ACFL el cuantificador universal se define como la Conjunción L_C^1, es decir, que para $x = \{x_1, ..., x_n\} \subset [0,1], \forall_{L_1} x_i\ \epsilon\ X = L_c^1(x_1, ..., x_n)$ y equivale a $\forall_{L_1} x_i\ \epsilon\ X = L_c^2(x_1, ..., x_n)$.

2.4.2. Definición y estructura de predicado

La definición de predicado esta asociado a la construcción de reglas o proposiciones que se evalúan de acuerdo con las lógicas existentes. Los predicados están conformados mediante etiquetas lingüísticas y se evalúan por medio de operaciones lógicas, sus resultados se obtienen aplicando operadores lógicos y existen diferentes tipos de lógicas para realizar dichos cálculos. Los elementos de un predicado son los estados lingüísticos, funciones de pertenencia, y operadores lógicos (Rodríguez-Fdez, Canosa, Mucientes, y Bugarín, 2015).

La estructura de un predica en CFL esta basada a partir de la construcción de una expresión verbal conformada por variables lingüísticas que se traducen al lenguaje de calculo de predicados y estos se dividen en predicados simples y predicados compuestos.

Un ejemplo para la construcción de predicados compuestos sobre predicados simples se muestra a continuación:

"*Un estado gestiona eficientemente las vacunas para el virus letal en un país, si cumple el siguiente requisito: Su gestión de inventarios es competitiva Si un estado tiene una gestión de inventario competitiva si posee una rotación normal, tiene un buen ciclo de crecimiento de rotación y una cobertura de inventario apropiada.* "

Predicados simples

- *R(X)*: "x es un estado con rotación normal de inventario".
- *C(X)*: "x es un estado con buen ritmo de crecimiento de rotación de inventarios".
- *CI(X)*: "x es un estado con una apropiada cobertura de inventario".

Predicado compuesto

GI(X): "x es un estado con una gestión de inventarios competitiva"·

Con los valores de verdad de los predicados simples, se obtiene el calculo de los valores de verdad de los predicados compuestos, para obtener el grado de verdad de un predicado complejo.

2.4.3. Variables lingüísticas

En la CFL la modelización de la vaguedad se lleva a cabo a través de *variables lingüísticas*, para representar en términos difusos que se encuentran en el lenguaje natural. Estas variables permiten el cálculo de veracidad en una expresión de manera cuantitativa, y además hace posible modelar el conocimiento basado en expertos haciendo uso de de estas variables lingüísticas asociando la información cualitativa a cuantitativa.

Los valores que pueden tener una variable lingüística se caracterizan por valores sintácticos o etiquetas que están conformadas por elementos infinitos. Un ejemplo para la variable "*edad*":

$T(edad) = joven + muy\ joven + no\ tan\ joven + mediana\ edad + adulto + no\ tan\ adulto + viejo + extremadamente\ viejo$

Representada en un gráfico es mas fácil de comprender, donde los limites de cada termino dado en el conjunto de datos como se muestra en la Figura 2.4.

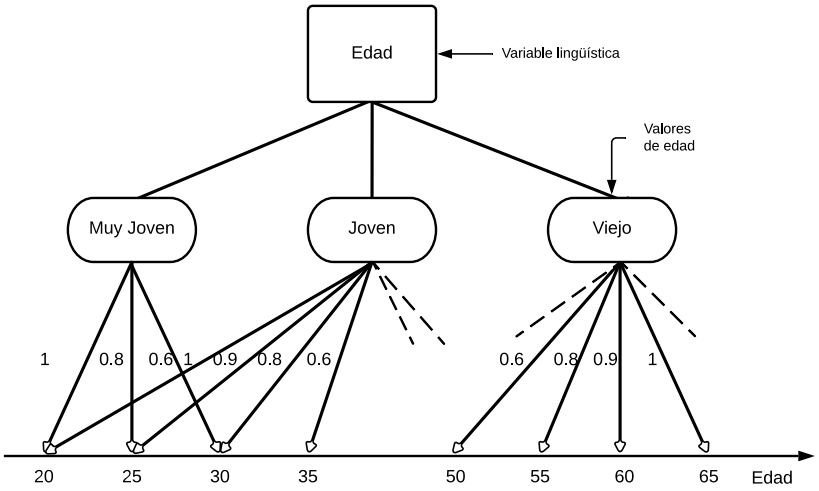


Figura 2.4: *Estructura jerárquica de una variable lingüística.*

Los estados lingüísticos son aquellas variables dentro del lógica difusa cuyos valores son palabras o etiquetas en lugar de números; estas palabras describen elementos que son difíciles de expresar numéricamente (Cruz-Reyes et al., 2019) definición formal es dada por el quinteto *(X, T(X), U, G, M)* donde:

- ***X*:** Variable lingüística.
- ***T(X)*:** el conjunto de etiquetas que X puede tomar.
- ***U*:** El dominio subyacente, los valores que están representados por las etiquetas lingüísticas.
- ***G*:** Gramática para generar etiquetas lingüísticas.
- ***M*:** Reglas semánticas que asocian cada etiqueta de *T(X)*

Los estados lingüísticos ayudan en:

- Dar granulación, que comprime la información ya que una etiqueta incluye muchos valores posibles.

- Proveer caracterización de situaciones poco definidas y/ó complejas.
- Traducción de variables lingüísticas a valores numéricos automáticamente.
- Ampliación de herramientas, donde las herramientas existen se pueden utilizar para manejar otras variables lingüísticas.

2.4.4. Funciones de pertenencia de CFL y ACFL

Las funciones de pertenencia representan el grado de veracidad de un miembro en un conjunto de datos definido. Son curvas que definen cómo cada punto de la entrada el espacio se asigna a un grado de pertenencia en el intervalo [0,1] como se vio anteriormente en la Figura 2.3. Supongamos que se desea saber que tan bien sabe un alimento. Se debe catalogar al individuo entonces de las siguientes maneras:

- Rico
- Normal
- Desagradable

En un conjunto clásico, esto se puede representar de la siguiente manera:

$X = \{'Rico','Normal','Desagradable'\}$

Pero es posible que no desee calificar el estado de un alimento solo en estos tres formas. Se necesita diferentes formas para poder expresen una gran variedad de grados.Por lo tanto, también puede agregar estas calificaciones:

- Asombroso
- Rico
- Normal
- Deplorable
- Insoportable

Se define una función en la que cada calificación tenga un valor específico. Esta función le permitirá ir más allá de las calificaciones. Esta función tiene un límite superior y un límite inferior. Eso significa que todas las categorías de calificación tendrán un valor que caerá en un punto de esa curva. Donde sus parámetros describirán el comportamiento que tienen los datos.

Función de Pertenencia Generalizada

Existe en si un gran número de funciones de pertenencia, sin embargo en el desarrollo de las herramientas difusas, no se considera una función en especifico mas adecuada para una aplicación dada (Llorente Peralta et al., 2019).

Drakopolus y su teorema de burbuja sigmoidal forma la base para aproximar cada función de pertenencia a una función sigmoidea (Llorente Peralta et al., 2019). Se realizan estudios en

(Andrade, Pérez, et al., 2014) de la generación de una Función de Pertenencia Generalizada (FPG). y se crea a partir del calculo siguiente.

La forma de calcular La función sigmoidal positiva acepta dos parámetros llamados gamma (γ) y beta (β); Su valor está dado por:

$$S(x;\gamma,\beta) = \frac{1}{1+e^{-\alpha(x-y))}} \tag{2.4}$$

$$\alpha = \frac{ln(0.99) - ln(0.01)}{\beta - \gamma} \tag{2.5}$$

La función sigmoidal negativa acepta dos parámetros llamados gamma (γ) y beta (β); Su valor está dado por 1 - Ecuación(2.4).

Y una FPG es definida como sigue:

$$FPG(x;\gamma,\beta,m) = \frac{Sg(x;\gamma,\beta)^m * (1 - Sg(x;\gamma,\beta)^{1-m})}{M} \tag{2.6}$$

- gamma (γ) es el valor que corresponde con el centro (0.5).
- beta (β) es el valor que corresponde con un limite inferior a (0.01), debe cumplir que $\gamma > \beta$.
- m determina la distribución de la función de pertenencia, y existe en el rango de [0,1]. La función se inclina hacia una sigmoidal negativa en m = 0, en positiva en m = 1 o una convexa en m = 0.5.
- $M = m^m * (1-m)^{1-m}$

Variable Lingüística Continua Generalizada

Para poder explicar la forma en que se genera una GCLV, es necesario explicar las funciones pertenecientes a la misma.

De acuerdo con una lógica difusa compensatoria L y su generador de funciones f, se define un modificador lingüístico generalizado $m(x, L)$ mediante la siguiente ecuación (Espin-Andrade et al., 2015):

$x_L^a = f^{-1}(af(x))$ donde $a \epsilon \mathbb{R}^+$

Sustituyendo $f(x)$ por $-g(ln(x))$ como el modificador lingüístico generalizado, puede llamarse a g función generadora secundaria de L. Además puede notar se que si $a = 1$ y $f(x) = 0$ y g es una función impar entonces $x_L^a = f^{(-1)}(0) = e^{-g^{-1(0)}}$.

Las funciones que se usan como estos modificadores los observamos en la Tabla 1 donde $n \epsilon \mathbb{N}$: También hay que observar que, x_L^a generaliza la ecuación:

$$C_T = \underbrace{(x, x, .., x)}_{\text{n veces}} = f^{(-1)}\underbrace{(f(x) + f(x) + ... + f(x))}_{\text{n veces}} = f^{(-1)}(nf(x)), donde\ n \epsilon \mathbb{N}$$

Tabla 2.2: *Funciones generadoras*

f	f^{-1}
$f(x) = -ln(x)$	$f^{-1}(x) = e^{-x}$
$f(x) = -ln^n(x)$	$f^{-1}(x) = e^{-\sqrt[n]{x}}$
$f(x) = -arcsinh(ln(x))$	$f^{-1}(x) = e^{-sinh(x)}$
$f(x) = -log_b(x)^n$	$f^{-1}(x) = b^{-\lvert x\rvert^{\frac{1}{n}}}$
$f(x) = -log_n(x)$	$f^{-1}(x) = e^{-ln(n)x}$

Además, si L es una ACLF y g es su función generadora secundaria, se obtiene una función sigmoidal generalizada.

$$S_g(x) = \frac{1}{1 + e^{-g(-1)(x)}} \tag{2.7}$$

Y de acuerdo con lo anterior se define una función sigmoidal generalizada parametrizada donde $\gamma \epsilon \mathbb{R}, \alpha > 0$ son los parámetros.

$$S_g(x; \alpha, \gamma) = \frac{1}{1 + e^{-g^{(-1)}(\alpha(x-\gamma))}} \tag{2.8}$$

Una vez visto lo anterior, una GCLV con los parámetros $\gamma \epsilon \mathbb{R}, \alpha > 0$ y $m \epsilon$ [0,1] generados por la función generadora secundaria g de una lógica L Arquimediana-Compensatoria, se define de la siguiente manera (Espin-Andrade et al., 2015):

$$GCLV_L(x; \alpha, \gamma, m) = \frac{C_T(S_g(x; \alpha, \gamma)^m_L, (1 - S_g(x; \alpha, \gamma))^{1-m}_L)}{M} \tag{2.9}$$

Donde C_T es una t-norma arquimediana continua en L y M es el máximo de $C_T(S_g(x; \alpha, \gamma)^m_L, (1 - S_g(x; \alpha, \gamma))^{1-m}_L)$ en $\mathbb{R}$. $S_g(x; \alpha, \gamma)^m_L$ y $(1 - S_g(x; \alpha, \gamma))^{1-m}_L$ son funciones lingüísticas generalizadas sobre $S_g(x; \alpha, \gamma)$ y $1 - S_g(x; \alpha, \gamma)$.

2.4.5. Operadores Lógicos de CFL y ACFL

Al inicio de esta Sección se hablaron de los operadores que conlleva cada lógica, además de de las propiedades axiomáticas que se deben respetar para cada una respectivamente. Recapitulando los cuatro principales operadores de la ACFL (Conjunción y Disyunción) tanto de la CFL y de la Lógica Arquimediana. Tenemos que si L es una ACFL y g su función generadora secundaria,, entonces la t-norma generada por g es la siguiente ecuación:

$$L^1_c(x_1, x_2) = e^{-g^{(-1)}(-g(ln(1-x_1)) - g(ln(1-x_2)))} \tag{2.10}$$

De acuerdo con esta ecuación y a una lógica L podemos obtener lo siguiente:

$$L^1_d(x_1, x_2) = 1 - e^{-g^{(-1)}(-g(ln(1-x_1)) - g(ln(1-x_2)))} \tag{2.11}$$

$$L^2_c(x_1, x_2, ..., x_n) = e^{-g^{(-1)}\left(\frac{-\sum_{i=1}^n g(ln(x_i))}{n}\right)} \tag{2.12}$$

$$L^2_d(x_1, x_2, ..., x_n) = 1 - e^{-g^{(-1)}\left(\frac{-\sum_{i=1}^n g(ln(1-x_i))}{n}\right)} \tag{2.13}$$

Otra de las ecuaciones que es importante mencionar es el cuantificador de universalidad que para un predicado p de una lógica L se calcula de la siguiente forma:

$$\forall = f^{-1}\left(\frac{\sum_{i=1}^n f(p_i)}{n}\right) \tag{2.14}$$

2.5. Representación basada en redes neuronales artificiales

La representación del conocimiento en las redes neuronales artificiales es un estudio que se ha vuelto relevante en el medio científico, tal como se ha descrito en la Sección 2.1.4.

En la actualidad el termino de caja negra se ha acuñado para los algoritmo de aprendizaje automático, mas específicamente para el aprendizaje profundo. Esto debido a la complejidad que conlleva las estructuras del aprendizaje automático, (Bengio, Goodfellow, y Courville, 2017) menciona "*Al mismo tiempo que aumenta la precisión y la escala de las redes profundas, también lo ha hecho la complejidad de las tareas que puede resolver*". Diversos estudios utilizan los beneficios del enfoque en la Inteligencia Artificial Explicable (XAI) para convertir estos modelos de caja negra en modelos de caja de cristal (Rai, 2020).

2.5.1. Descripción de una red neuronal artificial

Las redes neuronales artificiales actualmente son populares por las técnicas de aprendizaje automático que simulan el mecanismo de aprendizaje en los organismos biológicos. En otras palabras las redes neuronales artificiales (RNA) son algoritmos bioinspirados.

El sistema nervioso humano contienen células, las cuales son referidas como *neuronas*. Las neuronas están conectadas unas con otras con el uso de *axones* y *dendritas*, y la conexión entre axones y dendritas en el proceso de intercambio de información que ocurre entre cada neurona con estímulos electroquímicos se le conoce como *sinapsis*. Esta conexión se ilustra en la Figura 2.5. La fuerza de las conexiones sinápticas frecuentemente cambian en respuesta de un estimulo externo, Este cambio es como el aprendizaje toma lugar en los organismos vivos.

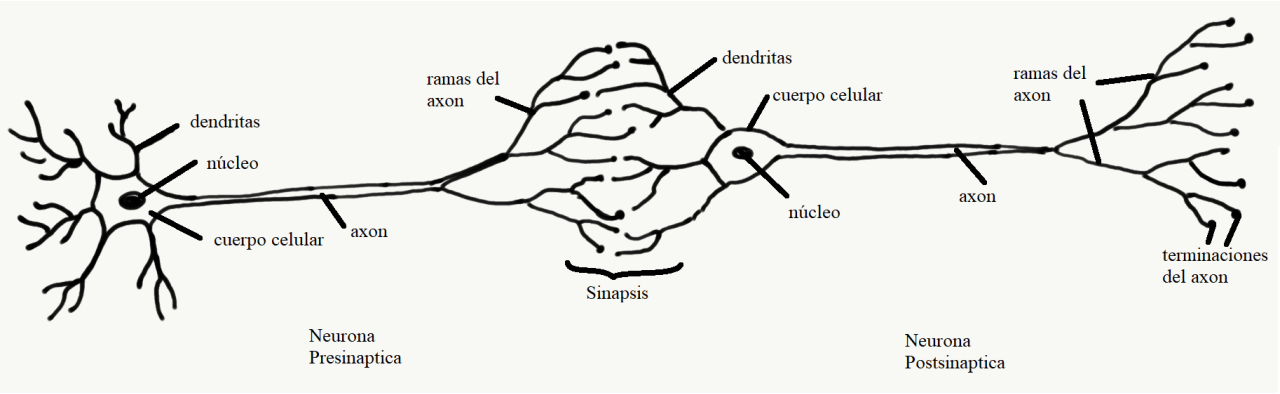
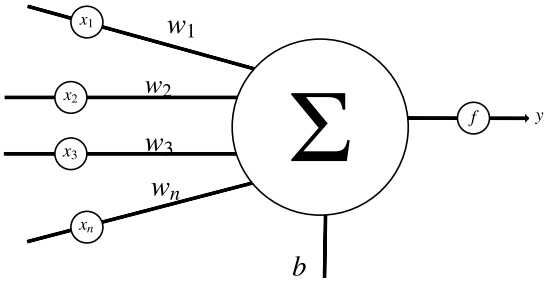


Figura 2.5: *Conexión sináptica entre neuronas. Imagen inspirada de (of Health et al., 2014).*

Al igual que en el sistema biológico, en las redes neuronales artificiales la unidad básica es la neurona. La neurona es un nombre para referirse a un modelo matemático que consta de varios elementos, $x_i \epsilon X$ que representan la información de entrada y $w_i \epsilon W$, Σ es una suma ponderada de las entradas y un termino independiente llamado sesgo b, para posteriormente pasar por una función de pertenencia f (Charu, 2018) que se muestra en la Figura 2.6.

Una red neuronal artificial consiste en cientos o miles de simples neuronas de procesamiento que están interconectados en distintas capas Figura 2.7, modelos mas fieles en (Skansi, 2018; Charu, 2018) las capas se describen a continuación:

- La capa donde las neuronas reciben los parámetros de un conjunto de datos de entrada se denomina Capa de entrada.



$$y = f(\Sigma_{i=1}^{n} x_i w_i \ + b)$$

Figura 2.6: *Estructura de la neurona artificial.*

- La ultima capa de la red es la Capa de salida.
- A las capas intermedias se les denomina Capas ocultas.

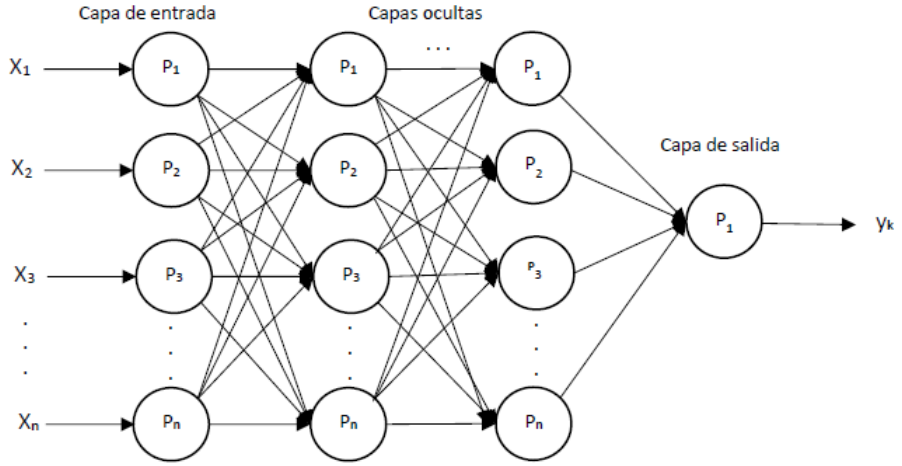


Figura 2.7: *Representación de capas en una red neuronal artificial.*

El objetivo de este modelo es aprender modificándose automáticamente a si mismo de forma que pueda llegar a realizar tareas complejas que no podrían ser realizadas mediante la clásica programación.

2.5.2. Aprendizaje en las redes neuronales artificiales

El aprendizaje de las redes neuronales no siempre fue de la misma manera en como actualmente se hace, (Minsky y Papert, 2017) tuvo su primera publicación en 1969 donde se demostraba matemáticamente las limitaciones en las redes neuronales en la época, la cual tuvo una repercusión negativa, un periodo de mas de 15 años conocido como *Invierno de la IA*. El florecimiento nuevo de la investigación en inteligencia artificial con la publicación (Rumelhart, Hinton, y Williams, 1986) donde se demuestra experimentalmente como usando un nuevo algoritmo de aprendizaje en redes neuronales auto ajustara sus parámetros para así aprender una representación interna de la información que estaba procesando. Una contribución importante del paradigma de las redes neuronales es llevar el pensamiento modular en el aprendizaje automático. En otras palabras, la modularidad en el diseño de las redes neuronales se traduce a la modularidad en el aprendizaje de sus parámetros (Charu, 2018); el nombre especifico para

esta modularidad es "*retro-propagación*".

En una sola capa de una red neuronal, el proceso de entrenamiento es relativamente sencillo porque el error (o perdida de la función) puede ser computada en función directa a los pesos, lo cual permite un fácil calculo del gradiente. En el caso de una red multi-capa como lo son en aprendizaje profundo, el problema es que la perdida es una composición de funciones complicadas de los pesos en las capas iniciales de la red. El gradiente de la composición de funciones es computada utilizando el algoritmo de retro-propagación.

2.5.3. Retro-propagación

El algoritmo de retro-propagación aprovecha la regla de la cadena del cálculo diferencial, que calcula el error gradientes en términos de sumas de productos de gradiente local sobre las diversas rutas de un nodo a la salida. Aunque esta suma tiene un número exponencial de componentes (rutas), se puede calcular de manera eficiente utilizando *programación dinámica*. El algoritmo de retro-propagación es una aplicación directa de la programación dinámica. Contiene dos fases principales, referidas como las fases *hacia adelante* y *hacia atrás*, respectivamente. Se requiere la fase de avance para calcular los valores de salida y las derivadas locales en varios nodos, y fase hacia atrás requiere acumular los productos de estos valores locales en todos los caminos desde el nodo a la salida (Charu, 2018):

1. *Fase hacia adelante*: En esta fase, las entradas para una instancia de entrenamiento son alimentadas dentro de la red neuronal. Esto da como resultado una cascada de cálculos a través de las capas, utilizando el conjunto actual de pesos. La salida final prevista se puede comparar con la del instancia de entrenamiento y la derivada de la función de pérdida con respecto a la salida es calculado. La derivada de esta pérdida ahora debe calcularse con respecto a la pesos en todas las capas en la fase "hacia atrás".

2. *Fase hacia atrás*: El objetivo principal de la fase hacia atrás es aprender el gradiente de la función de pérdida con respecto a los diferentes pesos mediante el uso de la regla de la cadena del cálculo diferencial. Estos gradientes se utilizan para actualizar los pesos. Dado que estos gradientes se aprenden en la dirección hacia atrás, comenzando desde el nodo de salida, este proceso de aprendizaje se denomina fase hacia atrás.

Como ya se menciono anteriormente se necesita el Gradiente, para poder resolver el sistema de ecuaciones y para ello se necesita preguntar (Gunning, 2017), ¿Como varia (∂) el costo del error (C) con respecto a la derivada de uno de los parámetros (Pesos, w) y (Sesgo, b)? $\frac{\partial C}{\partial w}$, $\frac{\partial C}{\partial b}$.

Para el calculo de estas derivadas es importante analizar cual es el camino que conecta el valor del parámetro el coste final (Bau et al., 2020). Tomando en cuenta una red que tiene L capas, L indicara la capa en la que este calculando el gradiente Figura 2.8. El proceso es dividida en varios pasos:

- $Z^L = W^L X + b^L \Leftarrow$ Resultado de la suma ponderada.
- $a(Z^L) \Leftarrow$ Función de activación.
- $C(a(Z^L)) \Leftarrow$ Costo de Error.

A estos pasos se le conocen como Composición de funciones. Donde L es el numero de capa en el que nos encontramos tomando a L como la última de ellas, y nos va a ayudar a calcular

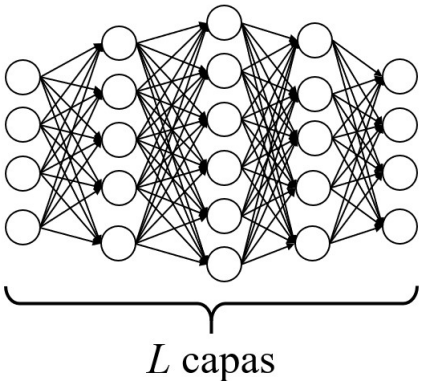


Figura 2.8: *Representación de la L-esima capa de una red neuronal.*

múltiples derivadas; Se hacen uso de las siguientes ecuaciones para el calculo del error de costo:

$$\frac{\partial C}{\partial w^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial w^L} \tag{2.15}$$

$$\frac{\partial C}{\partial b^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial b^L} \tag{2.16}$$

Realmente las dos primeras derivadas de la Ecuación 2.15 se comparten con las derivada parcial de la Ecuación 2.16, de manera que tendremos que resolver 4 derivadas.

La derivada de la activación (a) con respecto al costo (C), nos habla la manera en que varia el costo de la red cuando varia un poco la salida de activación de las neuronas en la ultima capa (L). Claro que en la ultima capa las activaciones de las neuronas es la salida de la red, con lo cual lo que se esta pidiendo es que se calcule la derivada de la función de costo con respecto a la salida de la red neuronal.

Por ejemplo, si la función de costo se calcula a partir del error medio cuadrado (MSE):

$$C(a_i^L) = \frac{1}{n} \sum_{i}^{n} \left(y_i - a_i^L\right)^2 \tag{2.17}$$

La derivada parcial correspondiente Ecuación 2.17:

$$\frac{\partial C}{\partial a_i^L} = a_i^L - y_i \tag{2.18}$$

La variación de la salida de la neurona cuando varia la suma ponderada de la neurona, es decir, la derivada de la activación (a) con respecto a Z. Por ahora lo unico que separa a Z con la activación de la neurona es la función de activación, con lo cual, lo que realmente se necesita es calcular la derivada de la función de activación.

Por ejemplo la función de activación de la neurona es una función sigmoidal Ecuación 2.19:

$$a_i^L(Z^L) = \frac{1}{1 + e^{-Z^L}} \tag{2.19}$$

Su derivada correspondiente sería:

$$\frac{\partial a^L}{\partial Z^L} = a^L(Z^L) \cdot (1 - a^L(Z^L)) \tag{2.20}$$

Y finalmente se tienen estas dos derivadas:

$$\frac{\partial C}{\partial w} \; y \; \frac{\partial C}{\partial b} \tag{2.21}$$

Lo que expresan las derivadas de la Ecuación 2.21 es la variación de la suma ponderada (Z^L) con respecto a una variación de los parámetros pesos (w) y sesgo (b).

Tendremos la derivada de la suma ponderada:

$$Z^L = \sum_i^n a_i^{L-1} \; w_i^L \; + \; b^L \tag{2.22}$$

Con respecto a w^L:

$$\frac{\partial Z^L}{\partial w^L} = a_i^{L-1} \tag{2.23}$$

Con respecto a b^L y debido a que el sesgo es un termino independiente la derivada parcial será:

$$\frac{\partial Z^L}{\partial b^L} = 1 \tag{2.24}$$

Notar que para la Ecuación 2.23 los valores de entrada de las neuronas en la capa L se corresponden con la salida de la capa anterior $(L-1)$.

Por lo tanto finalmente la solución que se esta buscando para los parámetros de la ultima capa Ecuación 2.15 + 2.16 (denominada regla de cadena multivariada (Charu, 2018))se calcula multiplicando cada uno con otras de los resultados.

Utilizando los mismos principios, se tiene que el bloque de la siguiente Ecuación 2.25, representa cuanto varia el error en función de la suma ponderada calculada dentro de la neurona.

$$\frac{\partial C}{\partial w^L} = \underbrace{\frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L}}_{\frac{\partial C}{\partial Z^L}} \cdot \frac{\partial Z^L}{\partial b^L} \tag{2.25}$$

Lo que representa la derivada parcial de $\frac{\partial C}{\partial Z^L}$ es en que grado se modifica el error cuando se produce un cambio en la suma de la neurona, si el resultado aplica un cambio en el valor final de la neurona, entonces se vera reflejado en el resultado final. Por el contrario si el valor de la derivada es pequeña, tendrá poca importancia que tanto varíe el valor de la suma, ya que este no afectara al error de la red. Visto de otro modo, lo que nos permite saber esta derivada que responsabilidad tiene cada una de las neuronas en el resultado final, y por lo tanto en el error. A este calculo se le conoce como *Error imputado a la neurona* (Bengio et al., 2017). Y se representa con δ^L, de manera que podremos reestructurar las Ecuaciones 2.15 y 2.16 de la siguiente manera:

$$\frac{\partial C}{\partial w^L} = \delta^L \cdot a_i^{L-1} \tag{2.26}$$

$$\frac{\partial C}{\partial b^L} = \delta^L \tag{2.27}$$

Ahora falta calcular el error imputado de las capas anteriores al error final de la red. Y aquí es donde entra es recursividad propia de la retro-propagación es decir que para el error de la capa $L-1$ se tiene:

$$\frac{\partial C}{\partial w^{L-1}} = \delta^L \cdot \frac{\partial Z^L}{\partial a^{L-1}} \cdot Ec.2.20 \cdot a^{L-2} \tag{2.28}$$

$$\frac{\partial C}{\partial b^{L-1}} = \delta^L \cdot \frac{\partial Z^L}{\partial a^{L-1}} \cdot Ec.2.20 \cdot 1 \tag{2.29}$$

De las Ecuaciones 2.28 y 2.29 solo faltaría calcular $\frac{\partial Z^L}{\partial a^{L-1}}$, derivada parcial de la matriz de parámetros que representa los pesos W^L que conecta la capa $L-1$ y L. Por lo que las primeras tres derivadas de dichas ecuaciones se convierten en:

$$\frac{\partial C}{\partial Z^{L-1}} = \delta^{L-1} \tag{2.30}$$

Con esto podemos aplicarlo al resto de capas de manera recursiva. Una diferente expresión del mismo modelo matemático se puede encontrar en (Bengio et al., 2017).

2.5.4. Redes Neuronales Difusas

Las redes neuronales difusas se utilizan para encontrar los parámetros relacionados con un Sistema Difuso aprendiéndolos a través de los datos dados, con la ayuda de redes neuronales. Estos parámetros pueden ser Conjuntos Difusos, Reglas Difusas, Funciones de Pertenencia Difusas, etc. Las redes neuronales difusas simples tienen las siguientes propiedades (Singh y Lone, 2020):

- Una red neuronal difusa se basa en un sistema de datos enfoque utilizando la metodología de redes neuronales.
- Las redes neuronales difusas se pueden crear con o sin el conocimiento previo de las reglas difusas, ya que pueden ser aprendido de los datos en paralelo utilizando redes neuronales.
- Las propiedades del sistema difuso subyacente son mantenido en todo momento, a pesar de que los parámetros se aprenden en el camino.
- n sistema difuso se puede representar como si tuviera múltiples nodos y su representación es similar a 2.8

Dentro de la arquitectura de una red neuronal difusa destaca el elemento único denominado neurona difusa, similar a 2.6 una neurona difusa se representa Figura 2.9 de la siguiente manera:

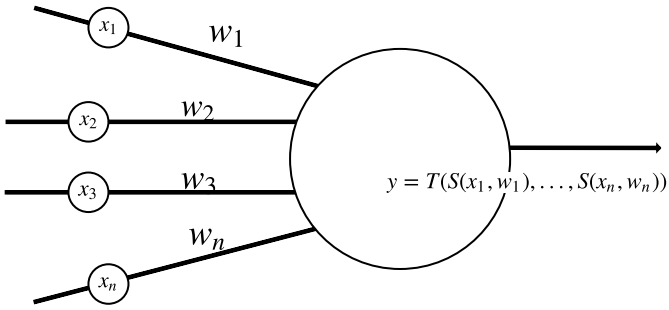


Figura 2.9: *Representación de neurona difusa.*

Esta Figura es una representación de una neurona difusa que hace uso de un operador de T-norma, sin embargo existen varias mas en la literatura (Singh y Lone, 2020):

- Implication-OR Neuron
- Kwan and Cai's Neuron
- K&C's Max Neuron
- K&C's Min Neuron
- etc.

2.5.5. Incorporación y extracción de reglas

Como se ha dicho la deficiencia que tienen las redes neuronales, hace que sean altamente incomprensibles, lo cual es inaceptable en algunas aplicaciones prácticas. Sin embargo perseguir una interpretación en estos modelos la mayoría de las veces empeorara el rendimiento del modelo (He et al., 2020).

Aunque la extracción de reglas en una RNA conlleva un costo adicional en términos de recursos y esfuerzo, cada vez más investigaciones se centran en la intención de resolver algunas preguntas comunes en RNA. El valor de incluir técnicas de extracción de reglas como complemento de las Las técnicas de RNA incluyen las siguientes (Andrews, Diederich, y Tickle, 1995): proporcionar capacidad de explicación a los usuarios, aplicar sistemas RNA a dominios de problemas críticos para la seguridad, mejorar la generalización y el rendimiento, adquirir conocimientos.

La manera en que se extraen reglas de un sistema de red neuronal es a partir de la generación de reglas de tipo *IF-THEN* y mediante un árbol de decisión, el cual es más común que use el clasificador de estructura de árbol en aprendizaje automático y minería de datos. Se puede utilizar en clasificaciones y problemas de regresión. El árbol de decisiones utiliza un sistema de caja blanca naturalmente tiene interpretabilidad que es fácil de explicar.

En (He et al., 2020) se visualiza como los pesos de una red recurrente son utilizados como apoyo a la decisión basado en reglas de arboles.

2.6. Descubrimiento de conocimiento basado en predicados

2.6.1. Evaluación de predicados

Una función de evaluación de predicados es aquella por el medio del cual se obtiene el valor de pertenencia global de cada predicado descubierto, esto significa hacer uso de la constante de universalidad expresado en las operaciones de conjuntos y por medio de esta determinar que tanto la proposición cumple con las reglas para todos los objetos pertenecientes a la misma instancia de datos.

En (Llorente Peralta et al., 2019) se menciona que dependiendo del tipo de operación difusa utilizada, varia la formula de obtención de grado de universalidad de las reglas encontradas, sin embargo, tal como se ha mencionado, esta formulación se basa en la operación de conjunción de los valores de pertenencia de cada uno de los elementos del registro, es por tal que la manera de representarla es la siguiente:

Se busca evaluar:

$$\lambda_{x\epsilon U} p_L(x) > \sigma(p_L);\ p_L \epsilon P_L \tag{2.31}$$

Donde:
P = Universo de predicados que pueden ser generados mediante el uso de operadores y variables.
p = Predicado seleccionado del universo de predicados.
U = Conjunto de objetos que conforman la instancia.
x = Objeto que forma parte del conjunto de objetos.
λ = operador de una lógica dada, de cada registro que evalúa el predicado.
σ = un mínimo de valor de verdad aceptado.
L = Metodología de una Lógica que evalúa el predicado.

2.6.2. Descubrimiento de predicados

La tarea de descubrimiento busca relaciones entre los estados lingüísticos de un conjunto de datos (predicados difusos) que cumplan con las especificaciones del usuario. La búsqueda se realiza mediante algoritmos genéticos para la búsqueda de predicados difusos y para el ajuste de los parámetros de las funciones de pertenencia definidas en los estados lingüísticos.

2.6.3. Inferencia para clasificación y pronostico

Actualmente para llevar a cabo experimentos y tareas de inferencia para clasificación y pronostico de datos, es necesario hacer uso de las tareas de descubrimiento de predicados, existen métodos como (Cruz-Reyes et al., 2019; Llorente Peralta et al., 2019) donde se hace uso de algoritmos genéticos que permiten navegar entre posibles valores de los parámetros para una lógica dada.

La inferencia es un proceso que se da utilizando los cálculos de la distinta variedad de operadores disponibles dentro de la CFL o ACFL, generalmente lo que se busca es evaluar un predicado de forma que se utilice en un operador de equivalencia en ambos sentidos, es decir, dado un predicado complejo $P \leftrightarrow class$ a través del cuantificador de universalidad, el valor obtenido se denominara valor *for all*. Y sirve para evaluar cual es el grado de pertenencia de cada registro para clase existente.

Capítulo 3

Estado del arte

El contenido de este capitulo, se muestran distintos trabajado de investigación que se desarrollaron por el grupo Eureka, varios de los experimentos de estos trabajos hacen uso de una variedad de lógicas difusas compensatorias (CFL, por sus siglas en ingles de Compensatory Fuzzy Logic) con la finalidad de evaluar una serie de tareas las cuales son evaluación de predicados difusos, descubrimiento de predicados lógicos difusos compensatorios, así como la implementación de la función de pertenencia generalizada, la cual nos permite crear funciones de pertenencia a partir de los datos contenidos en los almacenes de datos usados sin usar el conocimiento de un experto.

También se describen algunas herramientas que hacen uso de las teorías de CFL, así como de construcción de predicados mediante las tareas de descubrimiento de conocimiento, su respectiva evaluación, y la tarea de inferencia. Además de ello se describen algunos trabajos relevantes que hacen uso de redes neuronales y lógica difusa para la inferencia de datos, donde se mide la interpretabilidad de los datos y sus medidas de precisión.

3.1. Descubrimiento de conocimiento para la toma de decisiones usando lógica difusa compensatoria

Se separa en dos partes importantes aquellos trabajos que hacen uso de la Lógica Difusa Compensatoria, las investigaciones que se centran en la optimización del descubrimiento así como del uso de los predicados descubiertos. Y otra sección donde se detallan las herramientas en el mercado que ayudan a la toma de decisiones mediante el KD.

3.1.1. Algoritmos de optimización

Algoritmo de virtual savant basado en lógica difusa compensatoria para problemas de empacado de objetos.

En (Cruz-Reyes et al., 2019) se toma el problema de empacado de objetos (BPP). Se propone un algoritmo basado en el paradigma de Virtual Savant cuyo objetivo es inferir el comportamiento de un algoritmo mediante el aprendizaje automático, para reproducirlo en arquitecturas paralelas. La clasificación de datos que maneja esta basada en CFL y para las nuevas instancias no resueltas de BPP se producen probabilidades de asignación de clases.

Algortimo EK-CFL (Evolutionary Knowledge based on Compensatory Fuzzy Logic

El algoritmo de descubrimiento de conocimiento expresado en predicados de CFL nombrado EK-CFL (Evolutionary Knowledge based on Compensatory Fuzzy Logic), a través de un algoritmo genético que hace uso de las metodologías de programación datos usando varias metodologías de CFL, el funcionamiento general de genética, se lleva a cabo la evolución de los

predicados y descubre reglas en instancias (Llorente Peralta et al., 2019).

Algoritmo AG-GSF (Evolutionary Optimization of Generalized Sigmoidal Function)

En (Llorente Peralta et al., 2019) se presenta el algoritmo genético de optimización de parámetro de la función de pertenencia generalizada nombrado AG-GSF (Evolutionary Optimization of Generalized Sigmoidal Function) en, la cual está basada en la función sigmoidal generalizada (GSF) que es mediante la cual descubrimos las mejores configuraciones para los parámetros.

3.1.2. Sistemas de apoyo a la decisión

Eureka Universe

(Universe, 2021) Eureka Universe (EU) es una plataforma de Analítica de Negocios que utiliza una representación universal basada en predicados de lógica difusa y técnicas relacionadas. Su principal característica distintiva de otras herramientas analíticas es que la UE no requiere que los usuarios tengan un conocimiento significativo de áreas distintas de su propio campo, que se pueden incorporar en forma de lenguaje natural. EU trabaja con una arquitectura cliente-servidor escrita en código libre que se centra en el conjunto de datos del usuario. Eureka Universe permite a sus usuarios explotar este conjunto de datos para I) evaluar el conocimiento del usuario, o II) descubrir nuevos conocimientos.

Software Fuzzy Tree Studio (FTS)

(Studio, 2018) Este software posee un módulo cuyo objetivo es el de ayudar al usuario formalizar y calcular el valor de verdad de predicados implementando Lógica Difusa para cuantificar el valor de verdad de predicados parciales y operar adecuadamente con ellos, generalizando los conceptos de la Lógica de Predicados tradicional.

El propósito final del software el de apoyar a los decisores en el análisis de datos, la generación de inteligencia y la evaluación y comparación de alternativa.

Funciones:

- El sistema permite a los usuarios diseñar diagramas que representen árboles de predicados de Lógica Difusa, ofreciendo herramientas gráficas para su confección.
- Una vez finalizada esta etapa, la generación de datos de entrada desde diversas fuentes, verificando su coherencia respecto al modelo propuesto.
- Para concluir el proceso, el sistema es capaz de evaluar el modelo y brindar información gráfica y completa sobre los resultados obtenidos.
- Esta solución de software es escalable.

Características del usuario:

- Conocimientos relacionados a la Lógica Difusa y el diseño de predicados.
- Respecto a la interacción con el sistema son suficientes nociones básicas sobre manejo de aplicaciones Windows.

Software Icpro

(Meshino, 2008) Este software fue presentado por primera vez en el 2008 por profesores investigadores de la Universidad de Mar del Plata, Argentina; dirigidos por el Ing. Gustavo Meschino. Se conceptualiza como un framework de análisis de datos con técnicas de Inteligencia Computacional.

El sistema permite crear un nuevo proyecto que puede ser de los siguientes tipos: de lógica de predicados, de mapas auto organizados o de agrupamiento Kmedias. Asociado a esta investigación el proyecto que más se utilizó fue el de lógica de predicados; con el objetivo de insertar los predicados, tanto simples como compuestos, asociados al modelo basado en CFL. No obstante, a partir de su frecuente utilización se han encontrado algunas deficiencias en su funcionamiento desde el punto de vista informático.

3.2. Redes neuro-difusas

A Mamdani-Takagi-Sugeno based Linguistic Neural-Fuzzy Inference System for Improved Interpretability-Accuracy Representation

(Tung y Quek, 2009) propone un trabajo donde se mezclan dos enfoques conocidos las redes neuronales difusas, y los sistemas difusos. Se toma la representación altamente interpretable de las estructura de reglas que genera el sistema de Mandani, y la precisión de los sistemas de modelos difusos que propone Takagi-Sugeno. Esto debido a que los modelos difusos de Mamdani generalmente implican el uso de una base de reglas más grande con mayor complejidad y menor capacidad de interpretación.

El algoritmo diseñado bajo el nombre MTS sistema lingüístico de inferencia neuro-difuso (MTS-LiNFIS, por sus siglas en ingles). A pesar de que surge una unión entre los dos sistemas, la interpretabilidad en este sistema esta basado en el entendimiento de las reglas generadas a partir de una secuencia de gráficas dadas por la evaluación de los datos mediante funciones de pertenencia.

Este algoritmo se compara con diversos métodos de la literatura que general reglas y modelos de clasificación mediante ellas. Superado solo por el algoritmo HyFis (J. Kim y Kasabov, 1999) en generación de reglas (15) y precisión ($0.42\text{x}10^{-3}$), obteniendo MTS-LiNFIS 15 reglas y tasa de error $0.43\text{x}10^{-3}$.

Data-Driven Interval Type-2 Neural Fuzzy System With High Learning Accuracy and Improved Model Interpretability

(Juang y Chen, 2012) Propone un algoritmo basado en el modelo de Takagi-Sugeno-Kang, el cual se enfoca en la interpretabilidad que porta la generación de las reglas difusas del sistema, el método denominado DIT2NFS-IP, modelo del tipo Sistema Neuro-Difuso de tipo 2, se hace una comparación de de distintos sistemas de inferencia difusa, con 5 problemas de predicción.

La precisión del trabajo esta medido en base a la tasa de error en los datos mediante RMSE, mientras que la interpretabilidad esta medida de acuerdo a la cantidad de reglas que se generan y un numero pequeño de conjuntos difusos que se generan, menores a otros modelos contra los que se compara su trabajo.

El algoritmo se compara con los siguientes sistemas te tipo inferencia difusa: DIT2-LFR (Juang y Chen, 2012), DIT2NFS-AC (Juang y Chen, 2012). Y estos modelos tipo neuro-difusos: ANFIS (Jang, 1993), S-FSM (Ishibuchi y Nojima, 2007), T2FLS (Mendel, 2004).

3.3. Análisis y comentarios finales

En la tabla 3.1 se presenta una comparación de características entre los trabajos revisados y la propuesta de esta tesis. Se puede observar que el presente trabajo abarca características no contempladas de manera integral en los trabajos previos, destacando el uso de redes neuronales y predicados lógicos de ACFL.

Tabla 3.1: *Contraste del trabajo propuesto con el estado del arte*

Trabajo	Descubrimiento de predicados				Clasificación basadas en predicados	Núcleo de un Framework de apoyo a la decision
	Algoritmo de optimización 1. Propio 2. Abierto 3. Híbrido	Estructura de predicados 1. Árbol 2. Red Neuro Difusa	Lógica Difusa 1. CFL 2. ACFL	Función de pertenencia adaptiva		
IcPRO (Meshino, 2008)	0	0	1	No	No	Si
Fuzzy Tree Studio	0	0	1	No	No	Si
Eureka Universe(Universe, 2021)	1,2	1	1	Si	Si	Si
(Padron,2020)	1,2	1	1	Si	Si	No
(Llorente Peralta et al., 2019)	1,2	1	1	Si	No	No
(Tung y Quek, 2009)	1,3	2	0	No	No	Si
(Juang y Chen, 2012)	1	2	0	No	No	Si
Este trabajo	1,2,3	2	1,2	Si	Si	Si

Capítulo 4

Metodología de solución

En este capitulo, se describen la metodología seguida y conceptos que originan al presente proyecto de tesis, y como forma parte de un sistema de descubrimiento de conocimiento denominado Eureka-Universe-Core. Posteriormente, se detalla la estructura que la conforma dividida en dos grandes partes, el algoritmo que prepara los datos de entrada (un algoritmo genético), y el algoritmo que los procesa en función del objetivo planteado (red neuronal difusa), ambos con el uso de CFL y su extensión Arquimediana.

Como se describe en el Capitulo 2.4, la ACFL es una metodología lógica multivalente la cual nos permite obtener conocimiento expresado en predicados, y permite modelar el conocimiento de expertos de la misma manera, de tal forma que todo lo expresado en lenguaje natural puede ser transformado a un predicado de ACFL.

En el mismo capitulo, se definen las funciones de pertenencia propios de las lógicas y funciones generalizadas a partir de una función de pertenencia Sigmoidal, las cuales participan como mediadores difusos en los predicados y normalizadores difusos en la propuesta de solución.

4.1. Propuesta de solución

La propuesta de solución Genetic Algorithm and Artificial Neural Networks and Archimedean Compensatory Fuzzy Logic (GA-NN-ACFL). El diseño busca predicados interpretables basado en ACFL.

Se describen los procesos para ofrecer una contextualización de lo que se propone. El funcionamiento de GA-NN-ACFL es a través de dos algoritmos, el primero es un algoritmo genético que busca una serie de parámetros para ser inicializados en la red neuronal, y el segundo algoritmo es una red neuronal que hace uso de las funciones de pertenencia, así como de los operadores de la Lógica Difusa Arquimediana-Compensatoria, lo que convierte al sistema en una red neuro-difusa, y al comparar con otro algoritmo, al menos en la literatura se le denomina como sistema de inferencia difusa cooperativa. La Figura 4.1 representa el flujo del proceso que siguen los algoritmos para generar predicados, un modelo de clasificación entrenado y una serie de predicados conjuntivos evaluados.

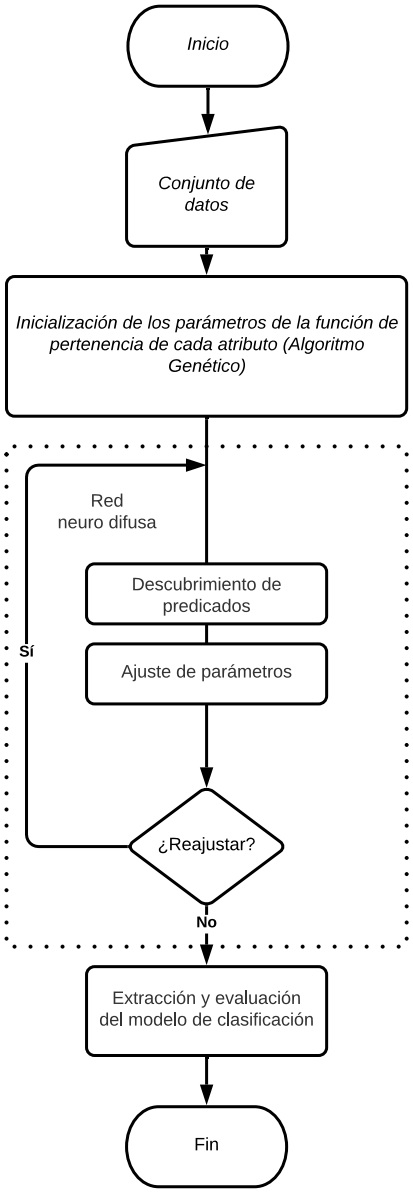


Figura 4.1: *Diagrama de flujo de GA-NN-ACFL propuesto para construir modelos de clasificación.*

4.2. Arquitectura de GA-NN-ACFL

Este proceso esta construido en una arquitectura de varias capas 4.2 denomidado núcleo de Eureka Universe, donde el mismo framework permite la interacción de distintos componentes establecidos dentro de ella, es decir, los algoritmos genéticos y la red neuronal están en una misma capa, mientras que las lógicas o las funciones de pertenencia que hace uno de ellas, se encuentran en una distinta. Esta arquitectura es donde se soporta el algoritmo de GA-NN-ACFL, que cuenta con su propia arquitectura de solución detallada en la Figura 4.3.

La primera capa de arriba hacia abajo, se compone por la entrada de los datos que se utilizaran para los cálculos, así mismo las Funciones de membresía se expondrán en el mismo nivel para poder ser construidas mas adelante por el constructor de predicados. La segunda capa se refiere al proceso y estructuración de los datos, es decir, en este nivel se define la estructura (árbol, grafo, matriz, etc.) que se empleará. La capa de descubrimiento engloba el resto de la arquitectura, esta hace uso de las lógicas y sus operadores, que son empleados por los algoritmos de las capas inferiores (los Genéticos y/o redes neuronales). A su misma vez los algoritmos de

la capa Algoritmos, tienen comunicación entre ellos, enviando y devolviendo datos procesados en forma de arboles de predicados. Por ultimo la capa de evaluación es una capa abstracta que hace uso de la solución generada para poder evaluar los predicados y obtener los resultados.

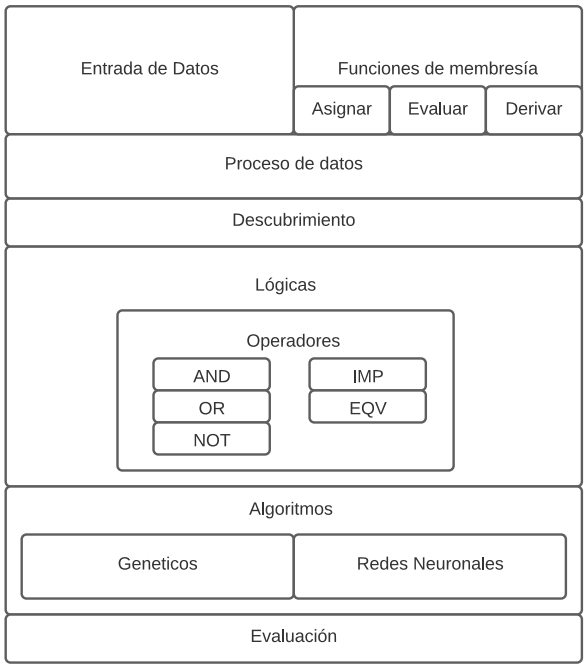


Figura 4.2: *Arquitectura del núcleo de Eureka Universe*

En la Figura 4.3 se muestra la arquitectura de solución que se propone en este trabajo de investigación, dicha arquitectura esta compuesta por un algoritmo de optimización del que se hablara mas adelante en secciones posteriores, el primer algoritmo provee los valores de parámetros iniciales a un segundo algoritmo, el segundo algoritmo es una red neuronal denominada bajo el nombre de Eureka Universe Neural Network and Archimidean-Compensatory Fuzzy Logic (EU-NN-ACFL). La red esta conformada por una serie de capas donde emplean las funciones de pertenencia, así como de los operadores declarados dentro de la ACFL. La propuesta de trabajo permite entrenar un modelo de aprendizaje supervisado, y extraer una serie de predicados para poder evaluarlos.

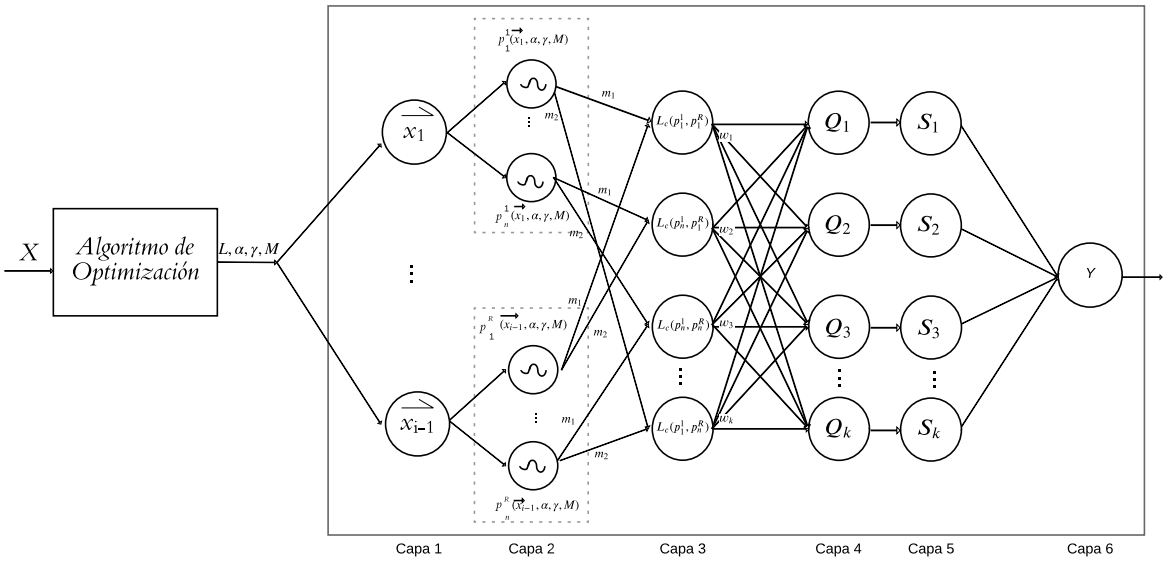


Figura 4.3: *Arquitectura Genetic Algorithm Neural Network and Archimedean Compensatory Fuzzy Logic.*

Este esquema de trabajo nos permite mediante el algoritmo GA-ANN-ACFL obtener predicados de primer orden.

4.2.1. Descripción de las capas

La estructura de GA-NN-ACFL esta dividida en varias capas que se describen de la siguiente manera:

Capa 1: capa de entrada de datos

Tal como se muestra en la Figura 4.3 La primera capa lo que hace es recibir un conjunto de datos, pero además cada dato de entrada $x_i \epsilon\ X$ tiene asociada una función de pertenencia cuyos parámetros (α, γ, M) cuyos valores son descubiertos, así mismo a la red entra una base y exponente que serán requeridas en su lógica (L) para los cálculos en la ACFL.

Estos juegos de parámetros son los que se utilizaran para caracterizar cada uno de los atributos del conjunto de datos y obtener variables lingüísticas, con ello permitirá obtener diversos predicados. Como se observa en la Figura 4.2, dichos parámetros formarían parte de las Funciones de pertenencia que se representan en el primer nivel de la Arquitectura.

Capa 2: capa de normalización difusa y de granulación

La capa de normalización difusa es la combinación de las funciones de pertenencia para cada una de las entradas. Observemos que para las variables x_1 y x_{i-1} son definidas por una función de pertenencia como predicados atómicos o predicados simples (Llorente Peralta et al., 2019; Andrade, Pérez, et al., 2014). Lo que representaría una variable lingüística, es decir, que $p_{1,\dots,n}$ representan a x_1 en distintas medidas (alto, medio, bajo,etc.) y así para cada una de ellas.

$$\begin{aligned} p_1^1 &= \mu(x_1) \\ &\dots \\ p_n^1 &= \mu(x_1) \\ &\vdots \\ p_1^R &= \mu(x_{i-1}) \\ &\dots \\ p_n^R &= \mu(x_{i-1}) \end{aligned} \tag{4.1}$$

Donde μ son las funciones de pertenencia GCLV (2.9) o FPG (2.6), $n = 1, 2, 3, \dots$ y $R = 1, 2, \dots, i-1$.

Capa 3: Capa de antecedente u Operador

En esta capa, se definen los antecedentes para los predicados mas complejos. Para este trabajo se utiliza predicados conjuntivos de ACFL (L_c), por lo que todas las señales que llegan a esta capa generan una salida que se encuentra en la operación de conjunción para cada una de ellas. Estas pueden ser representadas de la siguiente manera:

$$\begin{aligned} w_1 &= L_c(p_1^1, p_1^R) \\ w_2 &= L_c(p_n^1, p_1^R) \\ &\vdots \\ w_k &= L_c(p_n^1, p_n^R) \end{aligned} \tag{4.2}$$

Donde k = 1,2,...,R^n (Permutación con repetición).

Se crea una nueva neurona difusa, donde se utiliza los valores de verdad de los predicados p y las m que funcionan como modificadores en la capa, es decir la neurona se representa en la Figura 4.4. En esta figura $p_{1,...,n}$ representan los predicados simples y m juega el papel de los exponentes en el calculo de conjunción de acuerdo con los operadores de la Sección 2.4.5.

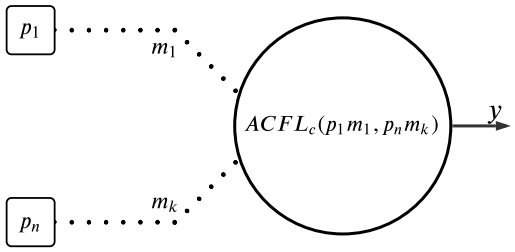


Figura 4.4: *Neurona Difusa Conjunción ACFL*

Estas neuronas son las que se emplean en la capa 3.

Capa 4: Capa de normalización de pesos

Cada nodo en esta capa es declarada con $Q_{1,..,k}$, cada k-esimo neurona calcula la relación entre los pesos(valor de verdad del k-esimo predicado , y la suma de veracidad de todos ellos, por conveniencia las salidas en esta capa son denominadas fuerza de salida normalizada (pesos).

$$Q_{1,..,k} = \frac{w_{1,...,k}}{\sum_{i=1}^{k} w_i} \tag{4.3}$$

Capa 5: cálculo de consecuente

A partir de esta capa el valor generado por las capas de antecedente, es decir, de la capa 1-3 es expresado como una regresión lineal. Aunque se puede aplicar el método de gradiente para identificar el valor del parámetro consecuente en la red, el método generalmente es lento y probablemente quede atrapado en los mínimos locales. Aquí se propone una regla de aprendizaje que combina un método de gradiente y la estimación de mínimos cuadrados (LSE) para identificar el valor del consecuente.

De acuerdo con la literatura por simplicidad (Dong, Huang, Wu, y Zeng, 2020), se asume que una red con neuronas difusas, trabaja bajo consideración teniendo una salida.

$$S = F(\vec{I}, X_i) \tag{4.4}$$

Donde $\vec{I}$ es el conjunto de antecedentes y X_i es el conjunto de clases. Se toman los valores de X_i, para conectar datos de entrenamiento P en 4.4 y obtener una ecuación matricial:

$$AX = B \tag{4.5}$$

Donde X es un vector conocido cuyos elementos son parámetros de X_i. Sea $|X_i| = M$, entonces las dimensiones de A, X y B son $P \cdot M$, $M \cdot 1$ y $P \cdot 1$, respectivamente. Desde P (numero de pares de datos de entrenamiento) suele ser mayor que M (numero de parámetros lineales), este es un problema sobre determinado y generalmente no existe una solución exacta para 4.5. En cambio, al menos la estimación de cuadrados (LSE) de X, X^*, se busca para minimizar la el error al cuadrado $||AX - B||^2$. Este es un problema estándar que forma la base

para la regresión lineal, filtrado adaptativo y procesamiento de señal. la formula mas conocida para usos de X^* usa el pseudo-inverso de X, es decir:

$$X^* = (A^T A)^{-1} A^T B \tag{4.6}$$

donde A^T es la transpuesta de A, y $(A^T A)^{-1} A^T$ es la pseudo-inversa de A si $A^T A$ es no singular. Mientras 4.6 es concisa en notación, es caro en calculo cuando tratar con la matriz inversa y, además, se vuelve mal definido si $A^T A$ es singular. como resultado, se emplea formulas secuenciales para calcular el LSE de X. Este método secuencias de LSE es el mas eficiente (Especialmente cuando M es pequeño) y se puede modificar a una versión para sistemas con características cambiantes. Específicamente, dejando el i-ésimo vector de la fila de la matriz A definida en 4.5 sea a_i^T y el i-ésimo elemento de B sea b_i^T, entonces X puede ser calculada iterativamente usando las formulas secuenciales ampliamente adoptadas en la literatura (Astrom y Wittenmark, 1984; Goodwin y Sin, 1984; Strobach, 2012) y mas recientemente en (Zhu et al., 2020; Pizarro, Alberto, et al., 2020; Dong et al., 2020; Boyd y Vandenberghe, 2018):

$$\left.\begin{aligned} X_{i+1} &= X_i + \lambda_{i+1} a_{i+1} (b_{i+1}^T - a_{i+1}^T X_i) \\ \lambda_{i+1} &= \lambda_i - \frac{\lambda_{i a_{i+1}} a_{i+1}^T \lambda_i}{1 + a_{i+1}^T \lambda_{i a_{i+1}}}, i = 0, 1, ..., P-1 \end{aligned}\right\} \tag{4.7}$$

Donde λ_i es frecuentemente llamada *matriz de covarianza* y la estimación de mínimos cuadrados X^* es igual a X_P Las condiciones iniciales para la ecuación 4.7 son $X_0 = 0$ y $S_0 = \gamma I$, donde γ es un numero grande positivo e I es la matriz identidad de dimensiones $M \cdot M$. Cuando se trata de una red adaptativa con múltiples salidas (salida en 4.4 es un vector columna), 4.7 se aplica excepto que b^T es la i-ésima fila de la matriz B.

Capa 6: Salida

La única neurona de la ultima capa es la que calcula la salida general como la suma de todas las señales entrantes, es decir:

$$y_i = \sum_{i=1}^{n} Q_i S_i \tag{4.8}$$

Cálculo de error

El propósito del calculo de error es ir midiendo la diferencia que existe entre el valor real del atributo a inferir y la salida de un elemento sobre el conjunto de datos. Para este calculo se utiliza la ecuación ya antes vista de MSE (2.17), expresada en 4.9.

Para ello se considera que el conjunto de datos de entrada tiene P entradas, se define la medida del error para cada p-ésimo ($10 \leq p \leq P$) entrada del conjunto de datos de entrenamiento como la suma de los errores cuadrados medios:

$$E_p = \sum_{m=1}^{\#(L)} (T_{m,p} - Y_{m,p}^L)^2 \tag{4.9}$$

donde $\#(L)$ es la representación de la ultima capa, x$T_{m,p}$ es el m-ésimo componente de p-ésimo vector de salida objetivo, es decir de la columna a inferir, y $Y_{m,p}^L$ es el m-ésimo componente del vector actual de salida producido por la presentación del p-ésimo vector de entrada, Por eso la medida del error general es $E = \sum_{m=1}^{P} E_P$.

El siguiente proceso después de esto es reajustar ciertos parámetros de la red que permitirán ir entrenando la red, para poder obtener distintos resultados, que bien en la mayoría de las ocasiones, estos mejoran. Se verá en las secciones mas adelante.

4.2.2. Representación de predicados

La arquitectura permite la representación de predicados conjuntivos (Andrade, González, y Caballero, 2014), esta representación esta dado por la unión de la capa 2 y la capa 3, cada una de ellas juega un papel importante debido a que como se menciona en 4.1 se da lugar a una serie de predicados simples.

Generalmente una variables lingüística se genera a partir de una serie de etiquetas que describen al atributo con diferentes granulaciones, es decir, x_1 puede ser bajo, medio, alto, etc. Sin embargo debido a que los parámetros de las funciones de pertenencia están dados por un algoritmo antecesor y no por un experto decisor, es incorrecto decir que se pueden caracterizar estas variables al inicio del entrenamiento, ya que se desconoce la forma de la función que se genera a partir de dichas entradas, no es hasta que finaliza el entrenamiento cuando podemos caracterizar a cada uno de los datos. Sin embargo 4.1 se refieren a distintas granulaciones de la variable $x_{1,\dots,n}$.

Ahora además la capa 3 lo que permite es permutar cada una de las variables lingüísticas en predicados mas complejos, donde interviene el operador de conjunción ($\wedge$) de la ACFL, es decir que ahora lo que tendríamos seria una serie de predicados permutados, por ejemplo:

Supongamos que tenemos dos variables A y B, y una clase C ahora a cada variables le asociamos 2 estados lingüísticos, si bien A puede ser A_{bajo}, también podría ser alguna otra etiqueta que caracterice ese atributo como $A_{medio\ bajo,medio,medio\ alto,alto,etc}$ y de igual manera para B. Entonces por ahora solo lo representaremos como A_1,A_2 y B_1,B_2.

En las representaciones basadas en reglas difusas de la Lógica Difusa, lo que se hace es que cada una de estas variables se combinan de la siguiente manera:

$$\text{Si } A \text{ es } A_1 \text{ y } B \text{ es } B_1, \text{ entonces } C.$$

Sin embargo para la ACFL A_1,A_2 y B_1,B_2 ya son variables lingüísticas que se pueden utilizar en predicados mas complejos, es decir que se pueden generar hasta 8 (r^{n+1}) predicados complejos. Donde r = numero de atributos o variables en el conjunto de datos (A,B) y n la granulación de estos atributos (2) para este ejemplo. Por lo que los predicados conjuntivos (T) que podemos crear a partir de ello en la capa 3 serian los siguientes:

$$\begin{array}{cccc}
 & & A_1 & \leftrightarrow C \\
 & & A_2 & \leftrightarrow C \\
B_1 & & & \leftrightarrow C \\
B_1 & \wedge & A_1 & \leftrightarrow C \\
B_1 & \wedge & A_2 & \leftrightarrow C \\
B_2 & & & \leftrightarrow C \\
B_2 & \wedge & A_1 & \leftrightarrow C \\
\underbrace{B_2 \quad \wedge \quad A_2}_{T} & & & \underbrace{\leftrightarrow C}_{Y}
\end{array} \tag{4.10}$$

Donde T son los predicados en forma normal conjuntiva antecedentes e Y el consecuente. Notar que la Ausencia de A o B en cualquiera de esos predicados T pueden ser representados como A_0 o B_0, de manera que 4.10 puede ser representado como el conteo de números en base (n+1), es decir que tendremos un total de 8 predicados expresados de la siguiente manera: 01, 02, 10, 11, 12, 20, 21, 22. Si bien 00 no esta expresado dentro del conjunto, aunque pertenezca, no es de interés debido a que 0 representa la nula participación de una variable, sin embargo se pueden proponer que 0 represente el operador de negación de la lógica en algunos otros trabajos futuros. Por lo que tendremos una matriz de indexación que representen a cada uno de los antecedentes del predicado 4.11.

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 2 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 2 & 0 \\ 2 & 1 \\ 2 & 2 \end{bmatrix} \tag{4.11}$$

Mientras que para el consecuente (Y) no es mas que la evaluación de la capa 4 y 5 como ya se ha descrito en las secciones anteriores. Por lo que con esta matriz de índices y 4.10 se construye un árbol representativo de predicados Figura 4.5.

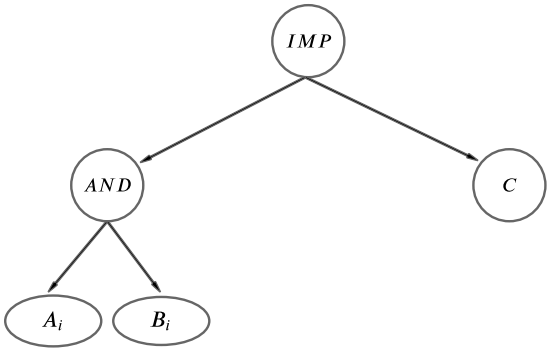


Figura 4.5: *Representación de un predicado difuso en un árbol. Donde i es el elemento en el índice de 4.11*

4.3. Método de ajuste de Parámetros

La red propuesta tiene una estructura secuencial con dos fases, la propagación hacia adelante y la retro-propagación (o propagación hacia atrás). Por lo que el ajuste de parámetros a lo largo de la red es mediante el descenso de gradiente en la segunda fase.

Esta segunda fase reajusta cada valor de los parámetros iniciales proporcionados por un algoritmo genético inicial, el cual provee de un conjunto de juegos de parámetros para una función de pertenencia que se utilizaran mas tarde en la capa 2 de la red propuesta.

Además también hace el calculo del error (E) imputado de cada neurona de cada una de las capas del antecedente y consecuente.

4.3.1. Algoritmo genético inicializador de parámetros

La parte inicial de la propuesta comprende a un algoritmo genético, capaz de proveer a la red de una serie de valores para los parámetros de las funciones de pertenencia que se utilizaran mas adelante en el proceso de creación de variables lingüísticos, para así poder crear predicados. El algoritmo inicial es tomado de (Llorente Peralta et al., 2019) el cual se separa en el Algoritmo 1, el cual genera un grupo de predicados con la siguiente característica:

Algorithm 1 Algoritmo genético contructor de parámetros

Entrada: Conjunto de datos, conjunto de funciones de pertenencia
Salida: Conjunto de funciones de pertenencias asociados a cada atributo

1: Generar la población inicial
   $poblacion \leftarrow random\_poblacion()$
2: Evalúa el fitness de cada individuo de la población
   $fitness\_individuo \leftarrow calcula_fitness(individual)$
3: $evaluar(poblacion)$
4: **while** $mejoresNindividuos ==$ **False** $||$ $generacion < max\_generaciones$ **do**
5: $\quad torneo\_seleccion(poblacion)$ ▷ Selección individuos a modificar
6: $\quad nuevo\_hijo \leftarrow cruza(padre_1, padre_2)$ ▷ Cruza de individuos seleccionados
7: $\quad nuevo\_hijo \leftarrow mutar(padre_1)$ ▷ Muta a los individuos seleccionados
8: $\quad$ **if** $FPG ==$ **True** $||$ $GCLV ==$ **True** **then**
9: $\quad\quad EOF - GSF(individuo)$ ▷ Optimizador de parámetros
10: $\quad\quad evaluar(poblacion)$
11: **return** $mejores\ N$ $||$ $mejor\_individuo$ ▷ N es conjunto de indiv. que superaron un valor de verdad mínimo

Sea X un conjunto de datos con L atributos, de los cuales x_L representa el atributo a inferir, y $x_{L-n}, ..., x_{L-1}$ representan los atributos para construir el antecedente del predicado. el objetivo es crear predicados donde la premisa se presenta en forma normal conjuntiva (FNC) y una lógica difusa arquimediana-compensatoria. Por lo que la forma del predicado seria el siguiente:

$$x_{L-n} \wedge x_{L-(n+1)} \wedge x_{L-(n+2)} \wedge ... \wedge x_{L-3} \wedge x_{L-2} \wedge x_{L-1} \leftrightarrow x_L \tag{4.12}$$

En el Algoritmo 1 tiene como primer propósito generar una población inicial, el cual pasara por los operadores del algoritmo, es decir, cada individuo tendrá un proceso de selección, cruza y muta; manteniendo la estructura con la que fueron construidos, debido a que mediante esta estructura se lleva a cabo la optimización de los parámetros que conforman cada uno de los *alelos* de un individuo o variable lingüística. En algoritmos genéticos un alelo es la estructura del cromosoma que contiene la información (gen) de un individuo. Debido a que la información de cada alelo esta estrechamente ligada con los parámetros de la función de pertenencia su estructura se representa de la siguiente manera:

Cromosoma del individuo:

$$\left[x_{L-n}^{[\beta,\gamma,m]},\quad x_{L-(n-1)}^{[\beta,\gamma,m]},\quad x_{L-(n-2)}^{[\beta,\gamma,m]},\quad ...,\quad x_{L-2}^{[\beta,\gamma,m]},\quad x_{L-1}^{[\beta,\gamma,m]}\right] \tag{4.13}$$

Estos parámetros de la GCLV o de la función de pertenencia empleada se inicializan para cada individuo que los contiene mediante un método pseudo aleatorio uniforme, así también se establecen limites inferiores y superiores para cada uno de los valores de cada parámetros

$(\beta, \gamma,\ m)$ de cada individuo. A demás de ello se establecen una lógica con la que se trabaja en esta tesis, actualmente se propone trabajar con la cuarta lógica de la Tabla 2.2, ($f(x) = -log_n(x)$ y $f^{-1}(x) = e^{-ln(n)x}$). Posteriormente se utiliza el Algoritmo 2 para mejorar los valores de los alelos, de una forma no muy intensiva. Este regresa un individuo evaluado con un mejor valor de verdad o igual de no haber encontrado mejoria.

Algorithm 2 Algoritmo Evolutionary Optimization of Generalized Sigmoidal Function

Entrada: Predicado de ACFL

Salida: Predicado de ACFL con igual o mayor valor de verdad

1: N es el número de individuos por generación
   $x_{best} : TipoSolucion \leftarrow EA(N :\ integer;\ f :\ tipoFuncionObjetivo)$
2: var
3: $xg, x * g :\ array[1, ..., N]\ ofTipoSolucion$ ▷ Población
4: **begin**
   Inicializar el número de generación
   $g \leftarrow 1$
5: Inicializar la población
   $x \leftarrow inicializar(N)$
6: **while** $g \leq max\_generaciones$ **do**
7: $x \leftarrow seleccionar(x, f)$ ▷ Selecciona los mejores individuos
8: $x :=\ alterar(x)$ ▷ Altera los individuos seleccionados
9: $g \leftarrow g + 1$
10: $x_{best} := seleccionar(x, f)$ ▷ Selecciona la mejor solución encontrada

Por ultimo del algoritmo 1 se rescata el mismo número de predicados aleatoriamente, como de granulación se requiera en la red neuronal, separando el juego de parámetros de cada individuo que haya superado el valor de verdad mínimo establecido.

4.3.2. Ajuste de parámetros de las funciones de pertenencia

El aprendizaje de la red neuronal es un proceso que se espera de este algoritmo por lo que una parte importante de ella es ajustar los valores de los parámetros de las funciones que se emplean, actualmente en la arquitectura de EU 4.2 cada función de pertenencia tiene el calculo de su derivada de acuerdo con cada parámetro, este trabajo desarrolla la formulación pertinente para las funciones de pertenencia GCLV y FPG.

Tal como se ha descrito antes, se necesitan calcular las ecuaciones de las derivadas parciales de cada uno de los parámetros que actúan en la función de pertenencia. Sin embargo para la nueva función, a demás de los parámetros ya conocidos como γ, β y m, también son utilizados dos parámetros denominados *base* y *exponente*.

Sea x, a, $b\ \epsilon\ \mathbb{R}$, $0 < x \leq 1$ y n(exponente) número natural impar.

Si $g(x; b) = \frac{x^n}{(ln(b))^n}$ y $g^{-1}(x; b) = ln(b)\sqrt[n]{x}$, entonces:

$f(x; b) = -(log_b(x))^n$, $f^{-1}(x; b) = b^{-\sqrt[n]{x}}$,

$$f^{-1}(z;b) = \begin{cases} b^{-\sqrt[n]{z}} & \text{si } z\epsilon[0,+\infty) \\ 0 & \text{si } z = +\infty \end{cases} \tag{4.14}$$

$$g^{-1}(z;b) = \begin{cases} ln(b)\sqrt[n]{z} & \text{si } z\epsilon(-\infty,0] \\ -ln(b)\sqrt[n]{-z} & \text{si } z\epsilon(0,\infty) \\ -\infty & \text{si } z = -\infty \\ +\infty & \text{si } z = +\infty \end{cases} \tag{4.15}$$

Si $a > 0$, aplicando la definición 1, $x_L^a = f^{-1}(a \cdot f(x;b);b) = x^{\sqrt[n]{z}}$ (no depende de b).

$$S_g(x;a,\gamma,b) = \begin{cases} \frac{1}{1+e^{-ln(b)\sqrt[n]{a(x-\gamma)}}} & \text{si } x \leq \gamma \\ \frac{1}{1+e^{ln(b)\sqrt[n]{-a(x-\gamma)}}} & \text{si } x > \gamma \\ 0 & \text{si } x = -\infty \\ 1 & \text{si } x = +\infty \end{cases} \tag{4.16}$$

$S_g(x = \gamma; a, \gamma, b = 0.5$, la gráfica $S_g(x;a,\gamma,b$ es cóncava hacia abajo $(\gamma, +\infty)$, cóncava hacia arriba $(-\infty, \gamma)$, y simétrica respecto a $(\gamma, 0.5)$.

- Considerando $f(x;b) = -(log_b(x))^n = -(\frac{ln(x)}{ln(b}))^n$

$$\frac{\partial f}{\partial b}(x;b) = \begin{cases} -\frac{1}{x \cdot ln(b)} & \text{si } n = 1 \\ -\frac{n \cdot (ln(x))^{n-1}}{x \cdot (ln(b))^n} & \text{si } n > 1 \end{cases} \tag{4.17}$$

- Considerando $g^{-1}(x;b) = ln(b)\sqrt[n]{x}$

$$\frac{\partial g^{-1}}{\partial b}(x;b) = \frac{\sqrt[n]{x}}{b} \tag{4.18}$$

- Sea $G^{-1}(x;a,\gamma,b) = g^{-1}(a(x-\gamma);b) = ln(b) \cdot \sqrt[n]{a(x-\gamma)}$, para $x \leq \gamma$.

$$\frac{\partial G^{-1}}{\partial a}(x;a,\gamma,b) = \begin{cases} ln(b) \cdot (x-\gamma) & \text{si } x < \gamma \\ \frac{ln(b) \cdot \sqrt[n]{x-\gamma}}{n \cdot \sqrt[n]{a^{n-1}}} & \text{si } n > 1 \text{ y } x < \gamma \end{cases}, con\ a > 0. \tag{4.19}$$

$$\frac{\partial G^{-1}}{\partial \gamma}(x;a,\gamma,b) = \begin{cases} -a \cdot ln(b) & \text{si } n = 1 \text{ y } x < \gamma \\ \frac{-a \cdot ln(b)}{n \cdot \sqrt[n]{(a \cdot (x-\gamma))^{n-1}}} & \text{si } n > 1 \text{ y } x < \gamma \end{cases}, con\ a > 0. \tag{4.20}$$

$$\frac{\partial G^{-1}}{\partial b}(x;a,\gamma,b) = \frac{\sqrt[n]{a \cdot (x-\gamma)}}{b}, para\ b > 1 \text{ y } x < \gamma \tag{4.21}$$

A partir de ellos podemos construir el vector gradiente de G^{-1} y de S_g cuyas componentes son las derivadas parciales de interés de estudio.

Para hallar el vector gradiente de $S_g(x;a,\gamma,b)$ tenemos en cuenta:

$$\frac{\partial S_g(x;a;\gamma,b)}{\partial a} = S_g(x;a;\gamma,b) \cdot (1 - S_g(x;a;\gamma,b)) \cdot \frac{\partial G^{-1}}{\partial a}(x;a,\gamma,b) \tag{4.22}$$

$$\frac{\partial S_g(x;a;\gamma,b)}{\partial \gamma} = S_g(x;a;\gamma,b)\cdot(1-S_g(x;a;\gamma,b))\cdot\frac{\partial G^{-1}}{\partial \gamma}(x;a,\gamma,b) \tag{4.23}$$

$$\frac{\partial S_g(x;a;\gamma,b)}{\partial b} = S_g(x;a;\gamma,b)\cdot(1-S_g(x;a;\gamma,b))\cdot\frac{\partial G^{-1}}{\partial b}(x;a,\gamma,b) \tag{4.24}$$

Esto con el fin de saber cual es el impacto que tiene cada uno de ellos en el resultado de coste de error en la ultima capa. Por ejemplo para la Ecuación Sigmoidal (4.25), deriva sus parámetros (γ,β) en las Ecuaciones 4.26 y 4.27 respectivamente.

$$S(x;\gamma,\beta) = \frac{1}{1+e^{-\gamma(x-y))}} \tag{4.25}$$

$$\frac{\partial S(x;\gamma,\beta)}{\partial \gamma} = -\frac{(\beta-x)*e^{\gamma(x-\beta)})}{(e^{\gamma(x-\beta)}+1)^2} \tag{4.26}$$

$$\frac{\partial S(x;\gamma,\beta)}{\partial \beta} = -\frac{\gamma e^{\gamma(\beta+x)}}{(e^{\beta\gamma}+e^{\gamma x})^2} \tag{4.27}$$

De la misma manera obteniendo las derivadas de la FPG Ecuacion(4.28) tendremos las derivada con respecto a β,γ y m, en las Ecuaciones 4.29, 4.30 y 4.31 respectivamente.

$$\frac{\partial FPG(x;\gamma,\beta,m)}{\partial \beta} : \frac{\partial\left(\frac{S^m*(1-S)^{1-m}}{m^m*(1-m)^{1-m}}\right)}{\partial \beta} \tag{4.28}$$

$$\frac{\partial FPG(x;\gamma,\beta,m)}{\partial \beta} = -\frac{\gamma e^{\gamma(x-\beta)}*(1-m)^{m-1}*m^{-m}*\left(\frac{1}{1+e^{\gamma(\beta-x)}}-1\right)^{m-1}}{\left(1+e^{\gamma(x-\beta)}\right)^2} \tag{4.29}$$

$$\frac{\partial FPG(x;\gamma,\beta,m)}{\partial \gamma} = -\frac{\left(e^{\gamma(x-\beta)}*(1-m)^{m-1}*m^{-m}*\left(\left(\frac{1}{1+e^{\gamma(\beta-x)}}\right)^{m-1}-1\right)\right)(\beta-x)}{\left(1+e^{\gamma(x-\beta)}\right)^2} \tag{4.30}$$

$$\frac{\partial FPG(x;\gamma,\beta,m)}{\partial m} \Rightarrow \frac{\partial\left(\frac{S^m*(1-S)^{1-m}}{m^m*(1-m)^{1-m}}\right)}{\partial m} = \tag{4.31}$$

$$(1-m)^{1+m}*m^{-m}*(1-S)^{1-m}*S^m*(ln(1-m)-ln(m)-ln(1-S)+ln(S))$$

Estos reajustes además de ayudar en el proceso de clasificación, también tienen como objetivo generar predicados con mas altos valores de verdad al evaluarse. Estos valores de parámetros servirán en el proceso de extracción y evaluación de predicados descubiertos dentro de la red neuronal.

4.3.3. Ajuste de pesos

Para el desarrollo de un modelo que aprenda mediante el descenso de gradiente a través del Error (ver Fórmula 4.9) sobre un espacio de parámetros, primero se tiene que calcular la tasa del error, manera muy similar señalado en el marco teórico, la tasa de error $\frac{\partial E_p}{\partial Y}$ para p-ésimo valor de entrenamiento y para cada neurona de salida Y. La tasa de error de la neurona de salida en (L,i) re puede calcular fácilmente a partir de 4.32:

$$\frac{\partial E_p}{\partial Y_{i,p}^L} = -2(T_{i,p} - Y_{i,p}^L) \tag{4.32}$$

Para la neurona interna en (k, i), la tasa de error se puede derivar siguiendo la regla de la cadena:

$$\frac{\partial E_p}{\partial Y_{i,p}^L} = \sum_{m=1}^{\#(k+1)} \frac{\partial E_p}{\partial Y_{m,p}^{k+1}} \frac{\partial Y_{m,p}^{k+1}}{\partial Y_{i,p}^k} \tag{4.33}$$

Donde $1 \leq k \leq L-1$. Es decir, la tasa de error de una neurona interna puede expresarse como una combinación lineal de las tasas de error de las neuronas en la siguiente capa. Por lo tanto para todos $1 \leq k \leq L$ y $1 \leq i \leq \#(k)$, podemos encontrar $\frac{\partial E_p}{\partial Y_{i,p}^k}$ gracias a 4.32 y 4.33.

Ahora si Q_k es un parámetro de la red (Capa 4), se tiene:

$$\frac{\partial E_p}{\partial Q_k} = \sum_{Y^* \epsilon S} \frac{\partial E_p}{\partial Y^*} \frac{\partial Y^*}{\partial Q_k} \tag{4.34}$$

Donde S es el conjunto de neuronas de la capa 5, quienes dependen de Q_k. Entonces la derivada de la medida de error general E con respecto a Q_k es:

$$\frac{\partial E}{\partial Q_k} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial Q_k} \tag{4.35}$$

En consecuencia, si se considera la adición de capas para predicados de orden mayor, la formula de actualización para un parámetro genérico es:

$$\triangle Q_k = -\eta \frac{\partial E}{\partial Q_k} \tag{4.36}$$

En el que η es una tasa de aprendizaje que puede expresarse además como:

$$\eta = -\frac{\vartheta}{\sqrt{\sum_{Q_k} \left(\frac{\delta E}{\delta Q_k}\right)^2}} \tag{4.37}$$

donde ϑ es el paso (capa), la longitud de cada transición de gradiente en el espacio de parámetros. Por lo general, podemos cambiar el valor de ϑ para variar la velocidad de convergencia.

4.4. Aprendizaje de clasificación con GA-NN-ACFL

La red neuronal tiene como propósito trabajar principalmente con el proceso de clasificación, como se ha descrito anteriormente este proceso se da gracias al entrenamiento de los diferentes elementos del trabajo. Sin embargo una de las características que se desea abordar es hacer predicciones de tiempo, sobre diversos registros históricos de un conjunto de datos dado, para ellos se necesita hacer un preprocesamiento de datos que permita a la red poder trabajar con la entradas.

4.4.1. Conversion entre series temporales e instancias de clasificación

Los métodos de aprendizaje automático, se pueden utilizar para el pronostico de series de tiempo. Para ellos, primero estos datos (registros históricos) deben enmarcar se como problemas de aprendizaje supervisado. De una secuencia de pares a pares de secuencias de entrada y salida.

Para ellos se debe comprender mejor la forma de las series de tiempo y los datos de aprendizaje supervisado. Como se ha descrito la serie de tiempo es una secuencia de números ordenados por un índice de tiempo como en la Tabla 4.1.

Tabla 4.1: *Representación de serie histórica*

índice	valor
1	0
2	1
3	2
...	...
10	9

Mientras que en un problema de aprendizaje supervisado los datos se componen de patrones de entrada (x) y patrones de salida (y), de modo que un algoritmo puede aprender a predecir los patrones de salida a partir de los patrones de entrada, Tabla 4.2.

Tabla 4.2: *Representación de datos dependientes.*

X	Y
1	2
2	3
3	4
...	...
8	9

Para realizar la transformación se tiene: Dada una secuencia de números para un conjunto de datos de series de tiempo. Podemos reestructurar los datos para que parezcan un problema de aprendizaje supervisado. Utilizando los pasos de tiempo anteriores como variables de entrada y usar el siguiente paso de tiempo como variables de salida. Se toma los datos de la Tabla 4.3. Teniendo como datos las fechas y los valores de mortalidad en dicha fecha.

Tabla 4.3: *Extracto de serie de tiempo Fechas-Muertos por virus Covid-19 TMA*

Fecha	Número Muertos
12/06/2020	19
14/06/2020	20
15/06/2020	22
16/06/2020	27

Reestructurar los datos de este conjunto de series históricas como un problema de aprendizaje utilizando el valor en el paso de tiempo anterior para predecir el valor en el siguiente paso de tiempo. Al reorganizar el conjunto de datos de series temporales de esta manera, los datos se verían como se muestra en la Tabla 4.4

Tabla 4.4: *Reestructuración de Conjunto de datos serie de tiempo sobre muertos por el virus COVID-19*

Fecha	Número Muertos
Desconocido	19
19	20
20	22
22	27
27	*Desconocido*

De este proceso se pueden observar lo siguiente:

- El paso de tiempo anterior es la entrada (x) y el siguiente paso de tiempo de salida (y) en el problema de aprendizaje supervisado.
- El orden entre las observaciones se conserva y se debe continuar conservándose cuando se usa este conjunto de datos para entrenar un modelo supervisado.
- No existe ningún valor previo que podamos usar para predecir el primer valor de la secuencia. Eliminaremos esta fila ya que no es conveniente utilizarla.
- No existe un próximo valor conocido para predecir el ultimo valor de la secuencia. Es posible que se desee eliminar este valor mientras también entrenemos el modelo supervisado.

El uso de pasos de tiempo anteriores para predecir el siguiente paso de tiempo se denomina método de ventana deslizante (Brownlee, 2016). Que se muestra en el algoritmo 3.

Algorithm 3 Algoritmo de Ventana deslizante para instancias de series de tiempo

Entrada: (S) Serie de Tiempo, n
Salida: (M) matriz de conjunto de datos para aprendizaje supervisado

1: Inicializar M
   $M \leftarrow array[1, ..., N]$
2: **for** `i to len(S)- n` **do**
3: $m \leftarrow array[1, ..., n]$
4: **for** j to n **do**
5: $m \leftarrow m + S[i + j]$
6: $M \leftarrow M + m$
7: **return** M

Capítulo 5

Experimentación y resultados

En este capítulo se presenta un conjunto de experimentos realizados para evaluar la calidad del algoritmo EU-NN-ACFL propuesto para el descubrimiento de predicados, optimización de parámetros, y clasificación de datos.

5.1. Diseño experimental

El equipo usado en cada uno de los experimentos es una maquina con un procesador Ryzen 7 2700x con 4Ghz *8, una memoria Ram de 16 GB, y un sistema operativo Windows 10. El lenguaje utilizado para la ejecución del programa es Python 3.8.

5.1.1. Instancias de experimentación

A continuación, en la Tabla 5.1 se muestran las instancias (NSF, 1987) utilizadas en la experimentación. Donde se indica el nombre, su descripción y la cantidad de atributos y objetos de cada una.

Tabla 5.1: *Descripción de las instancias*

Nombre de la instancia	Descripción	No. atributos	No. objetos
iris	Incluye tres especies de iris. Una especie de flor es linealmente separable de las otras dos, pero las otras dos no son linealmente separables entre sí.	6	150
tinto	Describe características relevantes sobre el vino tinto y el vino blanco.	12	1599
cancer	Las características se calculan a partir de una imagen digitalizada de un aspirado con aguja fina (FNA) de una masa mamaria. Describen las características de los núcleos celulares presentes	32	569
diabetes	Los conjuntos de datos consisten en varias variables predictoras médicas y una variable objetivo. Las variables predictoras incluyen el número de embarazos que ha tenido el paciente, su IMC, nivel de insulina, edad, etc.	9	768
Serie histórica TMA-Covid19	Se muestran los registros diarios de afecciones por el virus Covid-19 en la zona metropolitana Tampico, Madero y Altamira. Los registros corresponden a enfermos, recuperados, activos, muertos, etc.	9	532

Propósito de los experimentos

Comparar el rendimiento de la red utilizando Lógica Difusa Arquimediana-Compensatoria para medir la calidad en base a los resultados a través del error obtenido en la clasificación de los datos, mediante el uso de varias instancias utilizadas en la Tabla 5.1. Para la configuración del algoritmo genético inicial se describe en la Tabla 5.2.

5.1.2. Configuración de GA-NN-ACFL.

Tabla 5.2: *Parámetros de configuración*

Configuración del AG	
Tamaño de la población	50
Porcentaje de cruza	95 %
Porcentaje de mutación	5 %
Iteraciones	30
Mínimo Valor de verdad	0.9
Configuración del Red Neuro Difusa	
Numero de épocas	25
Granulación en func. de pertenencia	3

5.2. Comparación del algoritmo de referencia y GA-NN-ACFL en términos de valor de verdad y medida de exactitud

Se realiza la comparación entre los mas altos valores de verdad obtenidos por el algoritmo de referencia Tabla 5.2 y los valores de verdad obtenido por GA-NN-ACFL utilizando la GCLV Tabla 5.4 y la FPG Tabla 5.5 se muestran las 30 iteraciones realizadas de cada uno de los algoritmos, el ultimo renglón de cada tabla muestra el promedio de las instancias evaluadas.

Tabla 5.3: *Valor de verdad en el algoritmo de referencia*

No. de corrida	Iris	Tinto	Cancer	Diabetes
1	0.665038	0.704736	0.523840	0.467525
2	0.743698	0.790549	0.649902	0.546669
3	0.824974	0.859086	0.769418	0.605588
4	0.855437	0.897511	0.860768	0.645336
5	0.883727	0.939838	0.899727	0.687443
6	0.895359	0.946431	0.918521	0.708386
7	0.908029	0.974716	0.928784	0.722338
8	0.916063	0.969845	0.951204	0.729295
9	0.924801	0.983328	0.95605	0.739488
10	0.928933	0.997351	0.95529	0.752313
11	0.939652	0.98945	0.977382	0.746184
12	0.92161	0.987647	0.977285	0.751701
13	0.920178	0.981813	0.980171	0.764505
14	0.923245	0.974865	0.975711	0.766916
15	0.935195	0.984227	0.975619	0.749097
16	0.938426	0.982561	0.968929	0.766781
17	0.938114	0.988857	0.964333	0.773939
18	0.927106	0.994874	0.966566	0.761773
19	0.922437	0.985674	0.961404	0.764278
20	0.935442	0.990768	0.967947	0.766032
21	0.942345	0.986654	0.95943	0.775688
22	0.938645	0.984233	0.972308	0.77003
23	0.93623	0.984765	0.976712	0.769298
24	0.940492	0.988927	0.961218	0.775865
25	0.926967	0.984044	0.982665	0.770277
26	0.907192	0.974648	0.988431	0.767936
27	0.928009	0.976363	0.975524	0.7695
28	0.927233	0.985959	0.988713	0.766625
29	0.936861	0.989953	0.987499	0.771277
30	0.939033	0.972419	0.988067	0.769887
Promedio	**0.905682**	**0.958403**	**0.930314**	**0.730732**

Tabla 5.4: *Valor de verdad en algoritmo EU-NN-ACFL utilizando GCLV*

	Instancias			
No. de corrida	Iris	Tinto	Cancer	Diabetes
1	0.999996	0.999907	0.999463	1.000000
2	0.999957	0.996649	0.999889	1.000000
3	0.999491	0.999961	0.999977	1.000000
4	0.999999	0.999812	0.999906	0.999993
5	0.999840	0.999863	0.999997	0.999988
6	0.999500	0.999247	0.985466	1.000000
7	0.999580	0.999999	0.999473	0.999994
8	0.998815	0.998856	0.999692	0.999957
9	0.999939	0.998988	0.999860	0.999975
10	0.999583	0.999849	0.999919	0.999870
11	0.999998	0.999750	1.000000	1.000000
12	1.000000	0.999928	0.999917	1.000000
13	0.999979	0.999703	0.999480	1.000000
14	0.999996	0.999991	0.999503	0.999912
15	0.999956	0.998425	0.999400	0.999995
16	0.999989	0.999971	0.999672	1.000000
17	0.999960	0.999741	0.999863	1.000000
18	0.999672	0.999992	0.999951	1.000000
19	0.996377	0.999170	0.998018	0.999996
20	0.998322	0.999895	0.999959	0.999961
21	0.999491	0.999840	0.999645	0.999689
22	1.000000	0.999601	0.999993	0.999784
23	0.999505	1.000000	0.999974	0.999970
24	0.999997	0.999897	1.000000	0.999884
25	1.000000	0.999877	0.999665	0.999788
26	0.999876	0.998409	0.999999	1.000000
27	0.999998	0.999999	0.999972	0.999920
28	1.000000	0.998560	0.999718	0.999996
29	0.999336	0.999761	1.000000	1.000000
30	0.997784	0.999627	0.999994	0.999959
Promedio	**0.999565**	**0.999509**	**0.999279**	**0.999954**

Tabla 5.5: *Valor de verdad en algoritmo EU-NN-ACFL utilizando FPG*

	Instancias			
No. de corrida	Iris	Tinto	Cancer	Diabetes
1	1.000000	1.000000	0.999959	0.999965
2	0.999555	0.999561	0.999986	0.999875
3	0.999986	0.999570	0.999432	0.999979
4	0.999534	0.999577	0.999999	0.999999
5	0.999199	0.999965	0.999835	0.999999
6	0.998914	0.999704	0.999912	0.999936
7	0.998806	0.999317	0.999698	1.000000
8	1.000000	0.997132	0.999793	1.000000
9	0.999885	0.999510	0.999993	1.000000
10	0.999852	0.998979	0.999994	0.999971
11	0.999982	0.999998	0.999089	1.000000
12	0.999778	0.999828	0.999819	0.999999
13	0.999761	1.000000	0.999765	1.000000
14	0.999574	0.996438	0.999980	1.000000
15	1.000000	0.999955	0.999762	1.000000
16	1.000000	0.999963	1.000000	0.999276
17	1.000000	0.999786	0.999631	1.000000
18	1.000000	0.998157	0.999483	0.999969
19	0.999592	0.998096	1.000000	0.999741
20	0.998889	0.996555	0.999998	0.999997
21	1.000000	0.999998	0.999983	1.000000
22	0.999997	0.999880	0.999931	0.999951
23	1.000000	0.999941	0.999898	1.000000
24	0.999999	0.995065	1.000000	0.999814
25	0.996237	1.000000	0.999999	0.999997
26	1.000000	0.998986	0.999869	0.999773
27	0.999999	0.999742	0.999785	0.999926
28	0.997590	0.998741	0.999759	0.999989
29	0.999999	0.999836	0.998850	0.999954
30	0.999988	0.997064	0.999288	0.999651
Promedio	**0.999571**	**0.999045**	**0.999783**	**0.999925**

En la Tabla 5.6 se muestra una comparación de manera resumida de los promedios obtenidos de cada algoritmo.

Tabla 5.6: *Comparativas de promedios de valores de verdad*

Instancia	Algoritmo de referencia	EU-NN-ACFL GCLV	EU-NN-ACFL FPG
Iris	0.905682407	0.999565	**0.999571**
Vino tinto	0.958403019	**0.999509**	0.999045
Cáncer mama	0.930313919	0.999279	**0.999783**
Diabetes	0.730732325	**0.999954**	0.999925

Se puede observar que en las instancias los mejores resultados marcados en negrita son obtenidos a través de la EU-NN-ACFL mediante el uso de dos distintas funciones de pertenencia de la ACFL. Para ambos casos estos son cercanos a un valor de verdad de 1.

Para ello se hace un análisis con soporte estadístico, donde se tiene Rankeo Friedman con una Hipótesis nula (*H0*): La media de los resultados de 2 o mas algoritmos son lo mismo. Mostrando en la Tabla 5.7 y 5.8.

Tabla 5.7: *Prueba de Friedman (nivel de significación de 0.05)*

Conjunto de datos	**Estadística**	**p-value**	**Resultado**
Iris	89.09955	0	H0 es rechazada
Tinto	87	0	H0 es rechazada
Cancer mama	90.31429	0	H0 es rechazada
Diabetes	87.12903	0	H0 es rechazada

Tabla 5.8: *Ranking de los conjuntos de datos*

Algoritmo	Iris Rango	Tinto Rango	Cancer mama Rango	Diabetes Rango
Referencia	1	1	1	1
EU-NN-ACFL (GCLV)	2.43333	2.5	2.41667	2.48333
EU-NN-ACFL (FPG)	2.56667	2.5	2.58333	2.51667

En la Tabla 5.9 se muestran para cada conjunto de datos la comparacion entre los algoritmos EU-NN-ACFL (F) utilizando la FPG como función de pertenencia, EU-NN-ACFL (G) utilizando la GCLV como función de pertenencia, y el algoritmo de referencia (R). Teniendo como Hipótesis nula (H0): La media de los resultados de cada par de grupos es igual.

Tabla 5.9: *Nivel de Friedman (nivel de significación 0.05)*

Conjunto de datos	**Comparación**	**Estadística**	**p-value ajustado**	**Resultados**
Iris	F vs R	6.06767	0	H0 es rechazada
	G vs R	5.55128	0	H0 es rechazada
	G vs F	0.5164	0.60558	**H0 es aceptada**
Tinto	F vs R	5.80948	0	H0 es rechazada
	G vs R	5.80948	0	H0 es rechazada
	G vs F	0	1	**H0 es aceptada**
Cancer mama	F vs R	6.13222	0	H0 es rechazada
	G vs R	5.48673	0	H0 es rechazada
	G vs F	0.6455	0.51861	**H0 es aceptada**
Diabetes	F vs R	5.87402	0	H0 es rechazada
	G vs R	5.74493	0	H0 es rechazada
	G vs F	0.1291	0.89728	**H0 es aceptada**

De esta manera podemos observar que *H0* para las dos configuraciones de la red EU-NN-ACLF utilizando dos funciones de pertenencia distintas son rechazadas con respecto al algoritmo de referencia, y entre ellas es aceptada. Por lo que entonces se crea un experimento donde el objetivo es comparar los resultados mediante la precisión obtenida en función de la tasa del error de entrenamiento entre una configuración y la otra.

5.3. Inferencias con Series de tiempo

Este experimento tiene la finalidad de medir el error en la predicción de series de registros históricos a partir de la Lógica Difusa Arquimediana Compensatoria.

Se utiliza la misma configuración experimental de la Tabla 5.2 y además se hace un preprocesamiento de datos mediante el método de (Ver sección **??**). Tomando $T-3$ días para inferir el día T.

La Tabla 5.10 muestra el error obtenido para cada uno de los siguientes atributos del conjunto de datos Serie histórica TMA-Covid19: Personas Susceptibles, Expuestos, Asintomáticos, Moderados, Severos, Críticos, Recuperados, Muertos, Muertes Naturales por el virus covid-19 Utilizando la ACFL y la función de pertenencia de GCLV. De la misma forma la Tabla 5.3 muestra los datos obtenidos al utilizar la función de pertenencia FPG.

Tabla 5.10: *Error de clasificación de Serie Histórica TMA-Covid-19*

	Tasa de Error (GCLV)								
No. de corrida	Suceptibles	Expuestos	Asintomáticos	Moderados	Severos	Críticos	Recuperados	Muertos	Muertes Naturales
1	2.52E-03	0.00760135	0.00459759	0.00531838	0.00485698	0.0065422	0.00166376	0.0003861	0.00022915
2	0.00246998	0.00499626	0.00796895	0.00858662	0.00890343	0.00913172	0.00045444	0.0002241	0.00032707
3	0.0049186	0.00676462	0.00708394	0.00663709	0.00849967	0.00643248	0.00033988	0.0003445	0.00032536
4	0.00305202	0.00826931	0.00819939	0.00706564	0.00904235	0.00521672	0.00036976	0.00017927	0.00022287
5	0.00400986	0.00747394	0.00450646	0.00829243	0.00810681	0.00754377	0.00027894	6.24E-04	0.00035664
6	0.00308255	0.00851417	0.00507369	0.00817592	0.00835818	0.00831789	5.44E-04	0.00016064	0.00026001
7	0.00213095	0.00712529	0.00789533	0.00554396	0.00899726	0.00888838	0.00067733	9.81E-05	0.00050757
8	0.00366164	0.00728377	0.0077994	0.0050267	0.00575066	0.00520664	0.00017824	0.00042089	0.00016664
9	0.00330087	0.00581103	0.00758544	0.0041493	0.00861612	0.00940001	0.00028879	0.00030617	0.00028808
10	0.00580941	0.00589326	0.00430339	0.00765057	0.00757035	0.00784073	0.00079901	0.00022535	0.00032258
11	0.00412479	0.00596482	0.00677332	0.00478815	0.00691353	0.00848595	0.00026535	0.00047282	0.00024564
12	0.00349284	0.00786403	0.00666755	0.00439597	0.00602895	0.00763033	0.00024519	0.00127079	0.0003525
13	0.00245403	0.00799185	0.00477351	0.00283688	0.00807222	0.00895011	0.00053734	2.55E-04	0.00020226
14	0.00435774	0.00836278	0.00567977	0.00611269	0.00829247	0.00527957	0.00029983	0.00029951	0.00032371
15	0.00223671	0.00741294	0.00634933	0.00735288	0.0066662	0.00905645	0.00019528	9.90E-05	0.00043186
16	0.00446055	0.00691049	0.00545365	0.00709018	0.00911055	0.0065185	0.00025407	0.0001786	0.00023725
17	0.00466812	0.00875344	0.00537689	0.00709486	0.00933899	0.00708061	0.00013229	0.000147	0.00025134
18	0.00266975	0.00530956	0.00623852	0.00759178	0.00790316	0.00776765	0.00028863	2.42E-04	0.00027595
19	0.00381755	0.00702054	0.00663777	0.0087472	0.00901758	0.00874605	0.00021825	0.00073846	0.00017976
20	0.00164776	0.00514449	0.00637958	0.00629493	0.0084574	0.00622199	0.00017399	1.29E-04	0.00021032
21	0.00111472	0.00893285	0.0056038	0.00712758	0.00886692	0.00752302	0.00018726	1.03E-04	0.00024762
22	0.00094993	0.00979282	0.00635758	0.007612	0.00269408	0.00900137	0.00027838	0.00043315	0.00023077
23	0.00182581	0.00650526	0.00546613	0.0078656	0.00866552	0.00753326	0.00025556	0.00029301	0.00020531
24	0.00133807	0.00752693	0.00748443	0.0054031	0.00650824	0.00639047	0.00049312	0.00055121	0.00022953
25	0.00271107	0.00709803	0.00454975	0.00787459	0.00704069	0.0083508	0.00014506	0.00010244	0.00031121
26	0.00115797	0.00728568	0.00589299	0.00810908	0.00508153	0.00957879	0.00029471	0.00026784	0.00036778
27	0.00277323	0.0076709	0.00608128	0.00730651	0.00937023	0.00906572	0.00043048	0.00011761	0.00044572
28	0.00739524	0.00800589	0.00928675	0.0079168	0.00592676	0.00917197	0.00029109	0.00026179	0.00025992
29	0.00048892	0.00421769	0.00579155	0.00578415	0.00543286	0.00343919	0.0002432	0.00011807	0.00023777
30	0.00040289	0.0056841	0.00318965	0.00358016	0.00573218	0.00727463	0.0004931	0.00019238	0.00030611
Promedio	**2.97E-03**	**7.11E-03**	**6.17E-03**	**6.58E-03**	**7.46E-03**	**7.59E-03**	**3.77E-04**	**3.08E-04**	**2.85E-04**

Tabla 5.11: *Error de clasificación de Serie Histórica TMA-Covid-19*

No. de corrida	Tasa de Error (FPG)								
	Suceptibles	Expuestos	Asintomáticos	Moderados	Severos	Críticos	Recuperados	Muertos	Muertes Naturales
1	5.89E-05	0.01010697	0.0034695	0.00369853	0.00491455	0.00906069	0.00020312	0.00011858	0.00021672
2	0.000396053	0.00367035	0.00514723	0.00351524	0.00364428	0.00519284	0.00020803	0.0001605	0.00028535
3	0.000415277	0.00294757	0.0042757	0.00303977	0.00377703	0.00408033	0.00033277	0.00011769	0.00016934
4	0.000823486	0.00484817	0.00346788	0.00465676	0.00677451	0.00220048	0.00042401	0.00019167	0.00019381
5	0.000638788	0.00326321	0.00378816	0.00275414	0.00527944	0.00569234	0.0001314	9.36E-05	0.00018675
6	0.000369896	0.00441733	0.0038993	0.00101008	0.00543959	0.00435223	8.84E-05	0.00010721	0.00028846
7	0.000486034	0.00366	0.00264766	0.00655432	0.00042551	0.00692778	0.00042378	0.00019922	0.00026945
8	0.000602037	0.00365306	0.00395548	0.00748533	0.00357665	0.00550057	0.00063919	0.00012127	0.00037193
9	0.001407917	0.00687278	0.00236182	0.00504677	0.00369001	0.00445028	0.00059061	0.00010812	0.00027506
10	0.001222223	0.00330026	0.0044023	0.00477163	0.00273157	0.00448695	0.00053213	0.00046024	0.00017426
11	0.001797715	0.00579252	0.00420844	0.0057734	0.00370343	0.00505235	0.00014441	0.00044905	0.00026323
12	0.000626933	0.00495307	0.00650011	0.00451575	0.0033636	0.0063958	0.00056862	0.00014183	0.00031148
13	0.000524913	0.00275175	0.00643325	0.0052992	0.00642612	0.00236323	0.000134	6.26E-05	0.00030306
14	0.000557274	0.00338846	0.00341925	0.00410757	0.00787645	0.00365377	0.00019857	0.00018657	0.00018103
15	0.000322759	0.00656179	0.0042973	0.00835983	0.00357828	0.00502628	0.00034038	3.24E-05	0.00020576
16	0.000460483	0.00537186	0.00356695	0.0046134	0.00420489	0.00567672	0.00023975	0.00010625	0.00030842
17	0.000664261	0.00589396	0.00322592	0.00559486	0.00544461	0.00840991	0.00012366	0.00020276	0.000256
18	0.000209279	0.00317016	0.00342156	0.00425226	0.00378234	0.00546217	0.00021192	5.41E-05	0.0002082
19	0.001197235	0.0030472	0.00505578	0.00323413	0.00771072	0.0049444	0.00017371	0.00055448	0.00018491
20	0.001914449	0.00427538	0.00575999	0.00608487	0.00539886	0.00342778	0.00045901	6.54E-05	0.00031798
21	0.000147776	0.00266518	0.00631283	0.00617478	0.00673443	0.0053662	0.00016717	7.19E-05	0.00026832
22	0.000471068	0.00300414	0.00718074	0.00550912	0.00485313	0.00457134	0.00042389	0.00014266	0.00032946
23	0.000546362	0.00270891	0.00204814	0.00411904	0.00421909	0.00437159	0.00011415	0.00011939	0.00020215
24	0.002047148	0.00543635	0.00570165	0.00570165	0.00318244	0.00462812	0.00024688	0.00034456	0.00020765
25	0.000887045	0.00524681	0.00507386	0.00434809	0.00278679	0.00395493	0.00076682	0.00012523	0.00030434
26	0.000547644	0.00483247	0.00296464	0.00310569	0.00419114	0.00490502	0.00012401	0.00010094	0.00039697
27	0.000800351	0.00484615	0.00271683	0.00491296	0.00413152	0.00536286	0.00036862	0.0002649	0.00031034
28	0.00031408	0.00457742	0.00472733	0.00496088	0.00677954	0.00524412	0.00034259	0.00012712	0.00016777
29	0.000488916	0.00421769	0.00579155	0.00578415	0.00543286	0.00343919	0.0002432	0.00011807	0.00023777
30	0.000402887	0.0056841	0.00318965	0.00358016	0.00573218	0.00727463	0.0004931	0.00019238	0.00030611
Promedio	**7.12E-04**	**4.51E-03**	**4.30E-03**	**4.75E-03**	**4.66E-03**	**5.05E-03**	**3.15E-04**	**1.71E-04**	**2.57E-04**

Para poder observar mejor, en la Tabla 5.12 se resumen los promedios de cada serie histórica del conjunto de datos.

Tabla 5.12: *Comparativa de errores de clasificación*

	Suceptibles	Expuestos	Asintomáticos	Moderados	Severos	Críticos	Recuperados	Muertos	Muertes Naturales
FPG	2.97E-03	7.11E-03	6.17E-03	6.58E-03	7.46E-03	7.59E-03	3.77E-04	3.08E-04	2.85E-04
GCLV	**7.12E-04**	**4.51E-03**	**4.30E-03**	**4.75E-03**	**4.66E-03**	**5.05E-03**	**3.15E-04**	**1.71E-04**	**2.57E-04**

Como se puede observar, el uso de la función de pertenencia GCLV tiene una tasa de error mas bajo que usar FPG en todos los casos. Pero también observar cuales son los valores de verdad que arrojan cada uno de estos entrenamientos. En la Tabla 5.13 se muestran los valores de verdad máximos alcanzados para cada una de las series temporales del conjunto de datos Series Históricas TMA- Covid-19 utilizando la GCLV. Mientras que en la Tabla 5.14 se muestran cuando se utiliza la FPG.

Tabla 5.13: *Valor de verdad de Series Históricas TMA-Covid-19*

Valor de verdad GCLV

No. de corrida	Suceptibles	Expuestos	Asintomáticos	Moderados	Severos	Críticos	Recuperados	Muertos	Muertes Naturales
1	1.00E+00	0.99997277	1	0.99591453	0.99572606	0.9997215	0.999986724	0.99950501	0.99990714
2	0.99949102	0.99932753	0.99991524	1	0.99992226	0.99997649	0.999993553	0.99949553	0.9999996
3	0.99990432	0.99999742	0.99996979	0.99570199	0.99982063	0.99902567	0.99910768	0.99961778	0.9999998
4	0.99999928	0.99979732	0.998642	0.9999826	0.99710243	0.99922973	0.999999999	0.99999999	0.99997196
5	0.99997377	0.99525755	0.99945488	0.99297384	0.9891087	0.99977565	0.99742959	9.99E-01	0.99999998
6	0.9990626	0.99978055	0.99867819	0.99871785	0.99998546	0.99995331	1.00E+00	0.9963305	0.99999943
7	0.99999681	0.99912001	0.9999965	0.99455939	0.9999304	0.99999436	0.998819301	0.99997137	0.99995365
8	0.99999944	0.99999744	0.99957464	0.98889036	0.99930432	0.98591813	0.99999974	0.99996311	1
9	0.99942419	0.99719638	0.99999962	0.99968359	0.99997158	0.99503829	0.999944464	0.99811825	0.99998434
10	0.99724558	0.99981151	0.99994853	0.99992378	0.99737997	0.99999873	0.999998628	0.99224655	0.99988981
11	0.99999953	0.99999984	0.99959506	0.99999605	0.99973727	0.99999893	0.999991247	0.99903037	0.99967077
12	0.99987963	0.99928255	0.99902284	0.99999767	0.99999837	0.99968315	0.999832971	0.99999999	0.9978099
13	0.99994855	0.99995802	0.9997002	0.99237773	0.98846605	0.99809594	0.99862842	9.96E-01	0.99999947
14	0.99954986	0.99946203	0.9999873	0.99999341	0.99995811	0.99997813	0.999988079	0.99999999	0.99988848
15	0.99848074	0.99999972	0.99999921	0.99999215	0.99999998	0.99993952	0.995310755	1.00E+00	1
16	0.99999952	0.99976845	1	0.99920656	0.99924612	0.9997847	0.999999992	0.99999999	0.99998697
17	0.99999481	0.99999917	0.99995103	0.99694739	0.99790206	0.99891852	0.999999978	0.9999946	0.99691505
18	0.99959164	0.99872528	0.99999257	0.98578941	0.99687082	0.99397396	0.998191911	9.98E-01	0.99682994
19	0.99999628	0.99996182	0.99845108	0.9968222	0.99730584	1	0.999626618	0.99999997	0.99983102
20	0.99999999	0.99993497	0.99985944	0.9999193	0.99984714	0.99999703	1	1.00E+00	0.9963783
21	0.99999684	0.99999956	0.99923795	0.99922662	0.99999977	1	0.996273245	1.00E+00	0.99999999
22	0.99710288	0.99997247	0.99998032	0.99763709	0.9998805	0.99843415	0.999986237	1	0.99999176
23	0.99993321	0.99971394	0.99998674	0.99675049	0.99994603	0.99831751	0.999248083	0.98822249	0.99975216
24	0.99956328	1	0.9999221	0.99996572	0.99560938	0.99195043	0.999236029	0.9990616	0.99995309
25	1	0.9999996	0.99996424	0.99913164	0.9999921	0.99868737	0.991276669	1	0.99999543
26	0.99999984	0.99998354	0.9994339	0.99989946	1	0.99966393	0.999999998	1	0.99999989
27	0.9992849	0.99696351	0.99994605	0.99977177	0.99548185	0.99980133	0.999996619	0.99991492	0.99853931
28	1	0.99528457	0.99266712	0.99863155	0.99999998	0.9999999	0.996509117	0.99848016	0.99904827
29	0.99934571	0.99981853	0.99799707	0.99999947	0.99992395	0.99964053	0.995893335	0.99999999	0.99992245
30	0.99927683	0.99999477	0.99774284	0.99854244	0.99972524	0.99999998	0.999835289	0.99325569	0.99999998
Promedio	**1.00E+00**	**9.99E-01**	**9.99E-01**	**9.98E-01**	**9.98E-01**	**9.99E-01**	**9.99E-01**	**9.99E-01**	**9.99E-01**

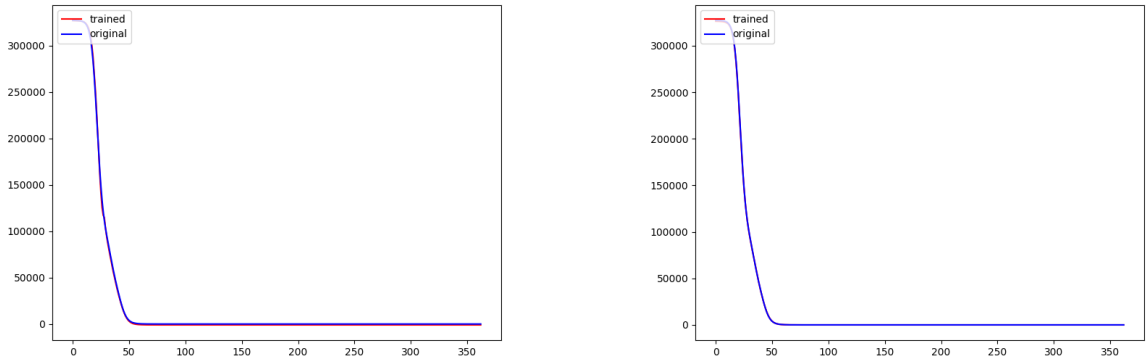
Tabla 5.14: *Valor de verdad de Series Históricas TMA-Covid-19*

Valor de verdad FPG

No. de corrida	Suceptibles	Expuestos	Asintomáticos	Moderados	Severos	Críticos	Recuperados	Muertos	Muertes Naturales
1	1.00E+00	0.99877926	0.99999763	0.98984691	0.99983937	0.99997048	0.99958964	0.99999592	0.99935775
2	0.998310672	0.99999824	0.994381896	0.99092256	0.99737396	0.9998199	0.99804583	1	0.99999949
3	0.999987371	0.99864902	0.999932185	0.99949492	0.98211923	0.99406296	0.99991327	0.99999953	0.99822911
4	0.999980682	1	0.99986468	0.99999989	0.99383927	0.99999998	0.99992975	0.9999937	0.999945
5	0.999988707	0.99997366	0.999965644	0.99991614	0.99608903	0.99996181	0.99748216	9.98E-01	0.99757898
6	0.999404545	0.99999999	0.999999991	0.99999997	0.99999891	0.99913196	1.00E+00	0.99958471	0.99969449
7	0.99808724	0.99979723	0.99998582	0.99998243	0.99788903	0.99998709	0.99999605	0.99292952	0.99999998
8	0.999682494	0.99999841	0.999003962	0.99997077	0.99914367	0.99999201	0.99999999	0.99645502	0.99994694
9	0.999997861	0.99955819	0.999993804	0.99154395	0.99999998	1	0.99997762	0.99999997	0.9968651
10	0.999942728	0.99995456	0.999375347	0.99977022	0.99996587	0.99995819	1	0.99998548	0.99999999
11	0.998919137	0.99927891	1	0.99999813	0.99934516	0.99997375	0.99997151	0.99984508	0.99967047
12	0.999996785	0.99898047	0.999862919	0.99993213	0.99751381	0.99875965	0.99998843	1	0.99994581
13	0.998452855	0.99998532	0.999999994	0.99990895	0.99926601	0.99999994	0.99998682	1.00E+00	0.99992533
14	0.999999888	0.99981318	0.999988946	0.98744889	0.99967806	0.9836023	0.99832237	0.99990226	0.99988633
15	0.997664261	0.99964242	0.997491626	0.9953946	0.9999886	0.99999761	0.99974817	1.00E+00	0.99999953
16	0.999999964	0.99999903	0.988418547	0.99826806	0.99999569	0.99999987	0.99999116	0.9994829	0.99999973
17	0.999999999	0.99998268	0.999743931	0.99999991	0.99985055	0.99993039	0.9999981	0.99999987	0.99981749
18	0.996693916	0.99996096	0.999992134	0.99842512	0.99999941	0.99999706	0.99983069	1.00E+00	0.999975
19	0.999780933	0.99999982	0.999765493	0.99999995	0.99962369	0.99999552	0.99930672	0.99998174	0.99861301
20	0.999994493	0.9967856	0.997660027	0.99999878	0.97223155	0.99996666	0.99999917	1.00E+00	0.98026195
21	0.999996202	0.99915957	0.999777166	0.99976838	0.99997113	0.99204528	0.99999993	9.98E-01	0.99404062
22	0.999989817	0.99925586	0.9988084	0.99945929	0.99983265	0.99973376	0.99998719	0.99987213	0.9983096
23	0.999988174	0.99999654	0.999988702	0.99983909	0.99739613	0.99996256	0.99711353	0.99999987	0.99906722
24	0.999847579	0.99994257	0.994810854	0.99591255	0.98770906	0.99981456	0.99782276	0.98635466	0.99999986
25	0.999046467	0.99977078	0.999967936	0.99999988	0.99828794	0.97297875	0.9974748	0.99999998	0.99290692
26	0.999999999	0.99996443	0.999507029	0.99999939	0.99999755	0.9950494	0.99987513	1	0.99908443
27	0.99962883	0.99718079	0.999632875	0.99999701	0.99985172	0.9804615	1	0.99983657	0.99078211
28	0.999999995	0.99898506	0.998111562	0.99856166	0.99977887	0.9999903	0.99937044	0.99866407	1
29	0.999446319	0.99999737	0.999999953	0.99997561	0.99944254	0.99986543	0.99988933	0.99999985	0.99977289
30	0.999984699	0.99874813	0.999999947	0.99999912	0.98775037	0.99999931	0.99980661	0.9998424	0.99999997
Promedio	**9.99E-01**	**9.99E-01**	**9.99E-01**	**9.98E-01**	**9.97E-01**	**9.97E-01**	**9.99E-01**	**9.99E-01**	**9.98E-01**

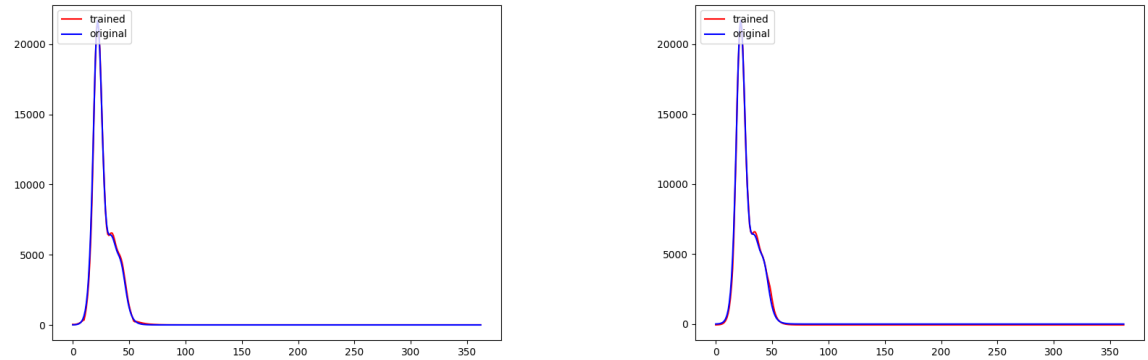
En las siguientes Gráficas se muestra el resultado final de EU-NN-ACFL infiriendo cada una de las series temporales, la linea de color azul representan los datos reales del conjunto de datos, y la linea color rojo representa los datos inferidos por la red, resultado de su entrenamiento. Cabe destacar que debido a que el error es bajo, esta lineas se solapan unas con otras.

El eje Y de cada gráfica representa el numero de personas alcanzadas, mientras que el eje X representa los registros por día del conjunto de datos.



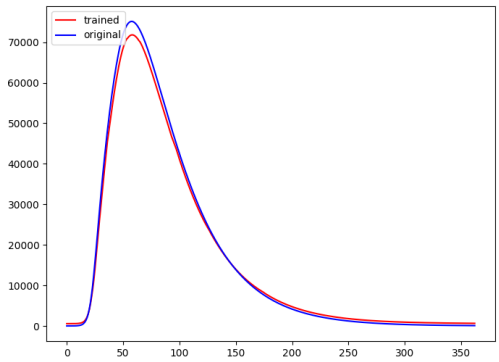
(a) *Resultados utilizando GCLV* **(b)** *Resultados utilizando FPG*

Figura 5.1: *Personas Susceptibles al virus Covid-19*

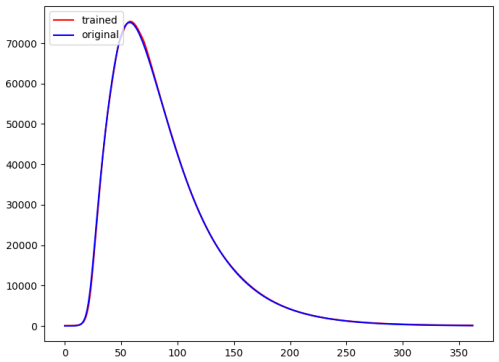


(a) *Resultados utilizando GCLV* **(b)** *Resultados utilizando FPG*

Figura 5.2: *Personas Expuestas al virus Covid-19*

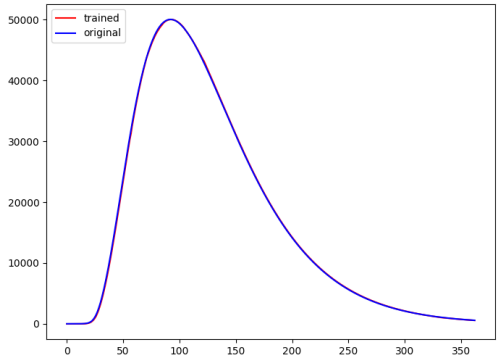


(a) *Resultados utilizando GCLV*

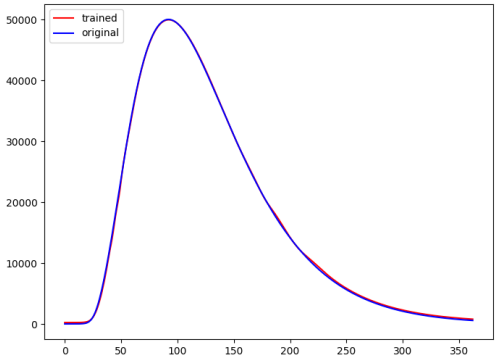


(b) *Resultados utilizando FPG*

Figura 5.3: *Personas con padecimientos Moderados por virus Covid-19*

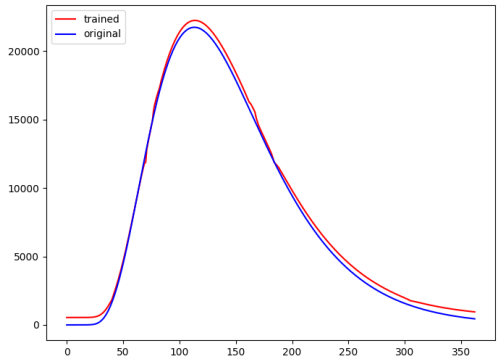


(a) *Resultados utilizando GCLV*

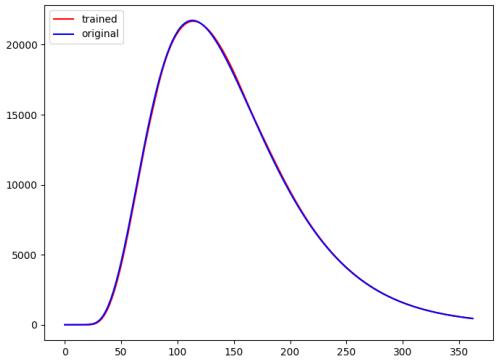


(b) *Resultados utilizando FPG*

Figura 5.4: *Personas con afecciones severas al virus Covid-19*

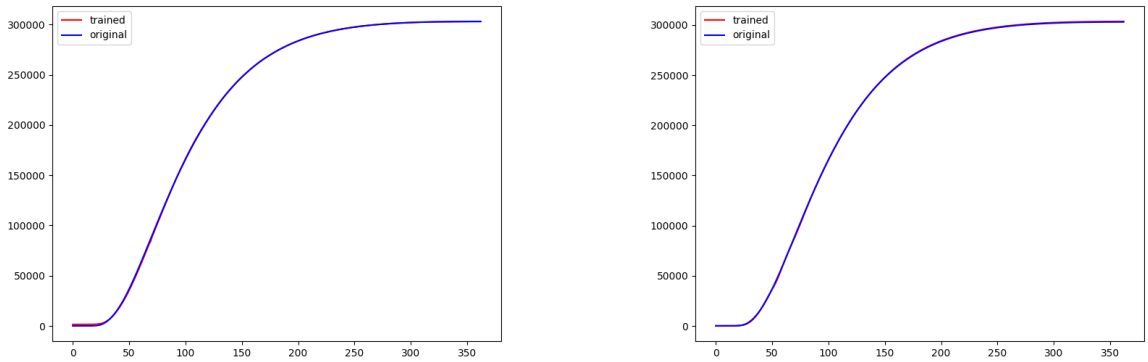


(a) *Resultados utilizando GCLV*



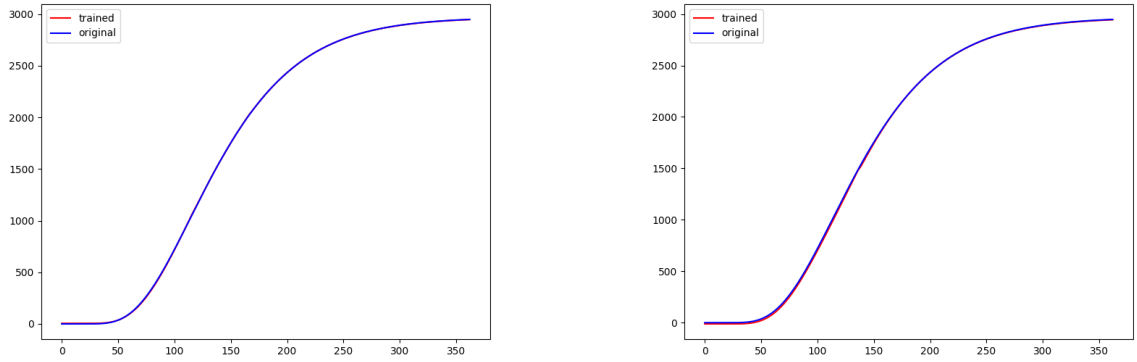
(b) *Resultados utilizando FPG*

Figura 5.5: *Personas en estado crítico con el virus Covid-19*



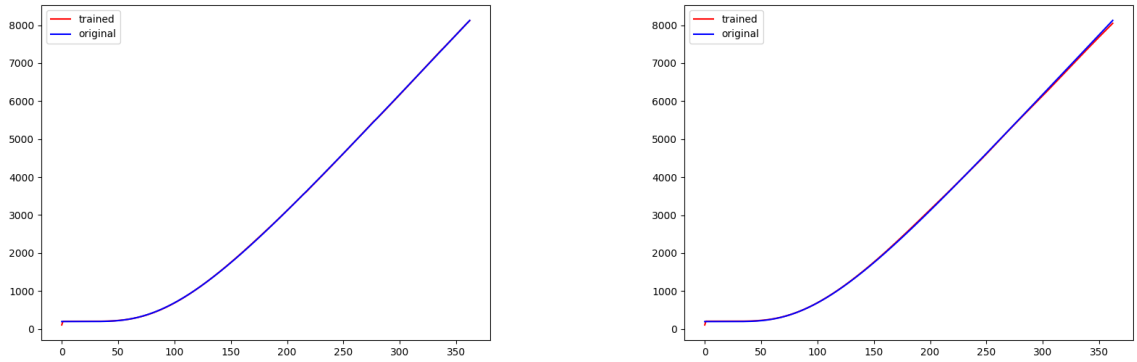
(a) *Resultados utilizando GCLV* **(b)** *Resultados utilizando FPG*

Figura 5.6: *Personas que se han recuperado del virus Covid-19*



(a) *Resultados utilizando GCLV* **(b)** *Resultados utilizando FPG*

Figura 5.7: *Muertes causadas por el virus Covid-19*



(a) *Resultados utilizando GCLV* **(b)** *Resultados utilizando FPG*

Figura 5.8: *Muertes Naturales, sin implicación del virus Covid-19*

De igual manera se hace la comparación entre estas dos configuraciones haciendo un análisis de los resultados con soporte estadístico empleando la prueba de Wilcoxon con un nivel de sigfnificancia de 0.05, con una Hipótesis nula *H0*: La media de las diferencias entre los dos algoritmos son iguales. La Tabla 5.15 muestra los resultados obtenidos de la prueba.

Tabla 5.15: *Prueba Wilcoxon GCLV vs FPG (nivel de significancia 0.05)*

Atributo	Statistic	p-value	Result
Suceptibles	5	6.521E-06	H0 es rechazada
Expuestos	15	1.86E-05	H0 es rechazada
Asintomáticos (I0)	27	6.129E-05	H0 es rechazada
Moderados (I1)	60	0.0011287	H0 es rechazada
Severos (I2)	12	1.365E-05	H0 es rechazada
Críticos (I3)	17	2.281E-05	H0 es rechazada
Recuperados (R)	159	0.3163713	H0 es aceptada
Muertos (D)	57	0.0008854	H0 es rechazada
Muertes Naturales (NR)	141	0.1580005	H0 es aceptada

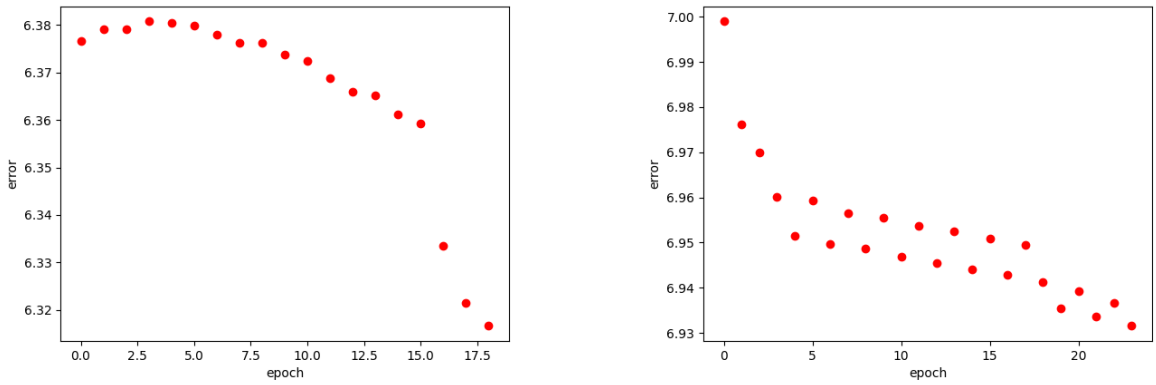
Dónde se observa que solo en los atributos de Recuperados y muertes naturales, los resultados obtenidos no se diferencian entra la predicción de una red y la otra.

5.4. Experimento 3: Inferencia con In stancia de Clasificación.

5.4.1. Conjunto de datos IRIS

En este experimento se mide la precisión en función al error de clasificación del conjunto de datos IRIS. Además poder evaluar las funciones de pertenencia generadas en la red para obtener un valor de verdad alto.

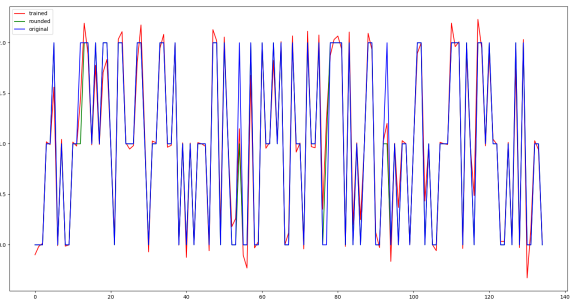
En la Figura 5.9 se muestran los errores por época de la red para cada función de pertenencia.



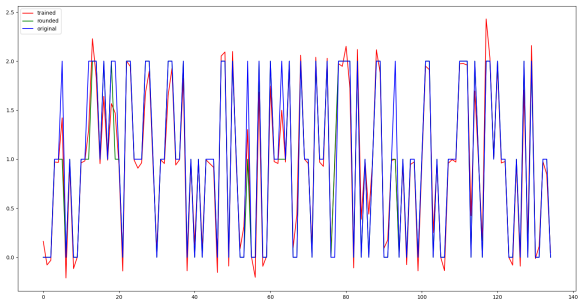
(a) *Error de entrenamiento en IRIS con GCLV* **(b)** *Error de entrenamiento en IRIS con FPG*

Figura 5.9: *Tanto la Figura 5.9a y 5.9b mejoran la clasificación durante el entrenamiento, sin embargo 5.9b tiene un comportamiento oscilatorio en su entrenamiento*

En las siguientes figuras se muestran las salidas de cada entrenamiento, el color azul muestra las clases del conjunto de datos Iris, y las rojas las clases inferidas por la red. El eje Y representan las clases del conjunto {0: Setosa; 1: Virginica; 2: Versicolor}, mientras que el eje X representan la entrada del conjunto de datos.



(a) *Resultados del clasificador utilizando GCLV*



(b) *Resultados del clasificador utilizando FPG*

Figura 5.10: *Clasificación de Conjunto Iris, las lineas azules representan las clases del conjunto, mientras que las rojas la clasificación resultado del entrenamiento de EU-NN-ACFL*

Para cada una de las redes se descubre un juego de parámetros donde cada una de éstos podría describir los predicados para la clase de Iris. Se obtiene por lo tanto en la Tabla 5.16 tres predicados con uso de GCLV y en la Tabla 5.17 tres predicados utilizando FPG.

Tabla 5.16: *Predicados del entrenamiento con GCLV*

Valor de verdad	Predicado
1.000000	*(IMP(AND "SepalLengthCm" "SepalWithCm" "PetalLengthCm" "PetalWithCm") "Species")*
0.999555	*(IMP(AND "SepalLengthCm" "SepalWithCm" "PetalLengthCm" "PetalWithCm") "Species")*
0.999986	*(IMP(AND "SepalLengthCm" "SepalWithCm" "PetalLengthCm" "PetalWithCm") "Species")*

Tabla 5.17: *Predicados del entrenamiento con FPG*

Valor de verdad	Predicado
0.999672	*(IMP(AND "SepalLengthCm" "SepalWithCm" "PetalLengthCm" "PetalWithCm") "Species")*
0.996377	*(IMP(AND "SepalLengthCm" "SepalWithCm" "PetalLengthCm" "PetalWithCm") "Species")*
0.998322	*(IMP(AND "SepalLengthCm" "SepalWithCm" "PetalLengthCm" "PetalWithCm") "Species")*

Los parámetros de los estados lingüísticos para los predicados anteriores se destacan en las Tablas 5.18 y 5.19

Tabla 5.18: *Parámetros de estados lingüísticos Specie de 5.16*

Estado	Beta	Gamma	m
Predicado 1			
SepalLengthCm	4.405688414	5.54797031	0.428259603
SepalWithCm	2.895564426	3.28024012	0.96619476
PetalLengthCm	1.616160873	1.77859912	0.456129155
PetalWithCm	1.901906495	2.18630102	0.591407603
Predicado 2			
SepalLengthCm	4.983062205	5.04657584	0.157055947
SepalWithCm	2.338167662	2.63002621	0.490659224
PetalLengthCm	4.963078386	6.45174601	0.997553198
PetalWithCm	0.368989471	1.27688584	0.03618371
Predicado 3			
SepalLengthCm	5.380837277	5.46913075	0.435971911
SepalWithCm	2.612205034	2.69804923	0.621505704
PetalLengthCm	1.839734481	3.32647532	0.141643885
PetalWithCm	0.316245131	0.57239133	0.528387832

Tabla 5.19: *Parámetros de estados lingüísticos Specie de 5.17*

Estado	Beta	Gamma	m
Predicado 1			
SepalLengthCm	4.89265857	5.0016396	0.25073592
SepalWithCm	2.30416957	2.68551908	0.33379021
PetalLengthCm	2.62892266	4.1138248	0.92966133
PetalWithCm	1.92010683	1.99935945	0.1102614
Predicado 2			
SepalLengthCm	4.46835307	5.29500757	0.33914719
SepalWithCm	2.16979935	2.39628045	0.59143659
PetalLengthCm	1.6528125	1.7270591	0.90062368
PetalWithCm	0.93741883	1.53469552	0.30409976
Predicado 3			
SepalLengthCm	5.55385939	5.76789861	0.23208468
SepalWithCm	2.96991969	3.02809387	0.43344572
PetalLengthCm	1.25272768	1.52767898	0.92904457
PetalWithCm	0.92926808	2.25240025	0.47843204

De los predicados descubiertos en la etapa de aprendizaje, se toma el segundo como el modelo de clasificación para la etapa de prueba, tomando en cuenta la precisión obtenida, el árbol de este predicado difuso se muestra en la Figura 5.11.

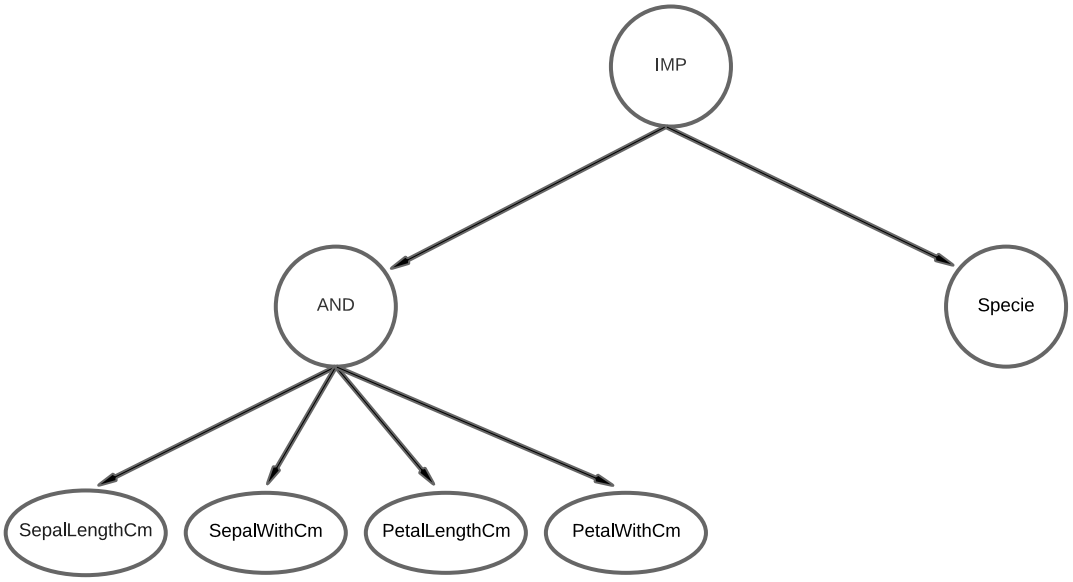


Figura 5.11: *Representación en árbol del predicado difuso*

Se compara el antecedente de cada predicado con respecto a su valor de verdad. Estos pueden ser evaluados y categorizados mediante un umbral de aceptación referente a la clase. Sin embargo el uso de la red facilita el proceso obteniendo la suma ponderada de estos valores de verdad para así poder obtener el consecuente. Estos experimentos tuvieron como resultado una tasa de error de clasificación de 5.365628 y 6.316644, para EU-NN-ACFL con GCLV y FPG respectivamente.

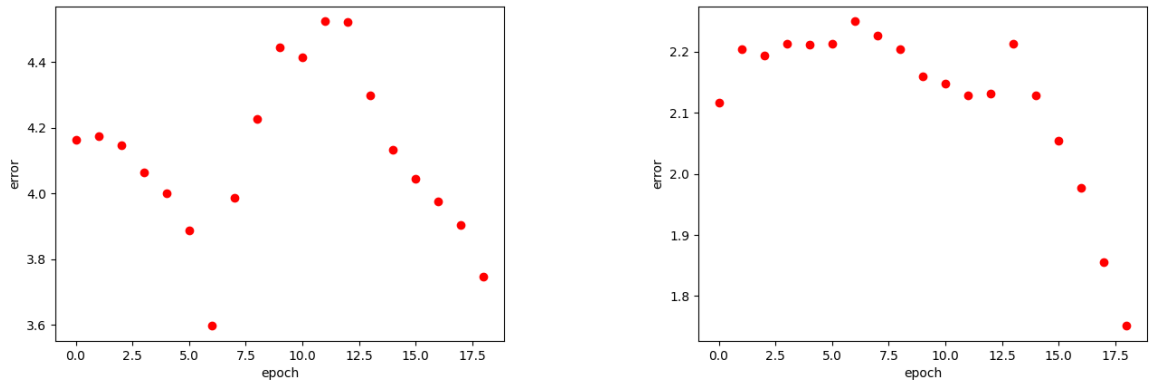
5.4.2. Conjunto de datos Cáncer de mama

En este experimento se mide la precisión en función al error de clasificación del conjunto de datos Cáncer de mama. Además poder evaluar las funciones de pertenencia generadas en la red para obtener un valor de verdad alto. En la Tabla 5.20 se muestras los valores de verdad de los entrenamientos y los errores de EU-NN-ACFL, con FPG y GCLV.

Tabla 5.20: *Valores de verdad y errores de entrenamientos de EU-NN-ACLF para el conjunto de datos de cáncer de mama.*

	Función de pertenencia GCLV		Función de pertenencia FPG	
No. de corrida	Valor de verdad	Errores	Valor de verdad	Errores
1	0.99999955	5.494300649	0.99989128	2.53094965
2	0.99869027	5.654246797	0.99670688	3.64262281
3	0.999092	4.975029495	0.99887059	3.41831957
4	0.99995495	4.879534957	0.99809779	2.5953405
5	0.99715978	5.323651553	0.99999709	3.4639003
6	0.99685987	4.371807965	0.99966282	2.87182189
7	0.99834449	4.540782979	0.99999961	3.19058761
8	0.99982065	4.407426203	0.99998683	2.76308494
9	0.99936467	5.019013315	0.99961251	2.81391473
10	1	5.003730716	0.99995599	2.53984961
11	0.99996195	4.938795829	0.9999994	3.15118962
12	0.99980204	4.862117901	0.99833321	2.7090546
13	0.99526395	4.908749139	0.99980246	2.77434593
14	0.99958986	5.227288268	0.99434281	2.22473839
15	0.99994824	5.115773583	0.99887446	2.94519792
16	0.99972615	4.091420589	0.99999937	1.75112045
17	0.99992182	4.250882076	0.99999891	3.19237526
18	0.99934854	4.95124684	0.99370296	3.00116115
19	1	4.269539267	1	2.92333601
20	0.99997999	5.211716267	0.99999997	2.91041031
21	0.99953864	5.459238681	0.99985453	1.97844469
22	0.99997894	4.858379962	0.99993242	4.01176242
23	0.99999979	4.722477425	0.99999643	2.11384777
24	0.99766273	3.746673208	0.99999994	3.34428565
25	0.99889082	4.531730758	0.99990509	2.56929697
26	0.99916931	4.934653794	0.9999742	2.20976671
27	0.99371361	5.266443436	0.99999556	2.88362451
28	0.99991497	5.00428308	1	2.9037295
29	0.99958696	5.12810596	0.99996769	2.59297878
30	0.9999998	5.641137017	0.99999997	3.70808326
Promedio	**0.99904281**	**4.893005924**	**0.99924869**	**2.85763805**

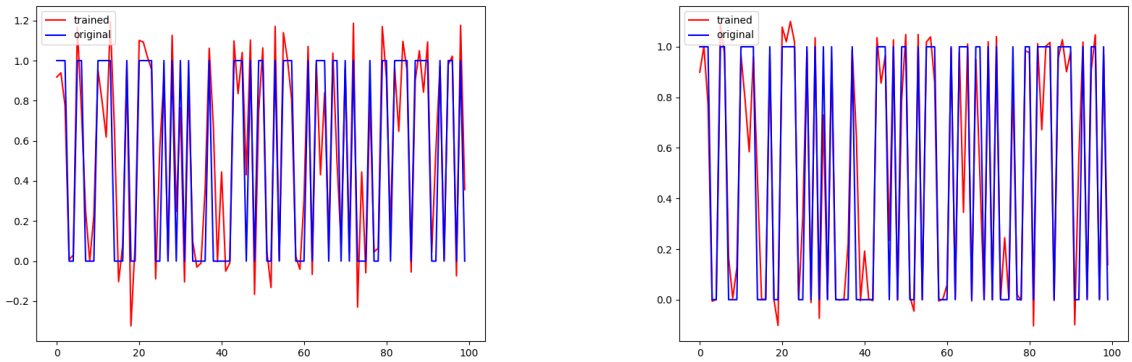
En la Figura 5.12a y 5.12b se muestra el error de entrenamiento con el uso de la GCLV y FPG respectivamente. De acuerdo con los datos de la Tabla 5.20 el error en la red es mayor en la primera configuración 5.12a que en 5.12b.



(a) *Error de entrenamiento en Cáncer de mama con GCLV* **(b)** *Error de entrenamiento en Cáncer de mama con FPG*

Figura 5.12: *Tanto la Figura 5.12a y 5.12b se muestra el descenso de error en los entrenamientos de la red.*

En el ANEXO **??** se muestran los resultados de la clasificación de una instancia de prueba de Cáncer de mama, representados en las gráficas de las Figuras 5.13a y 5.13b.



(a) *Clasificación Cáncer de mama con GCLV* **(b)** *Clasificación Cáncer de mama con FPG*

Figura 5.13: *Tanto la Figura 5.13a y 5.13b representa las perdiciones del ANEXO* **??**. *El eje Y representa la clase Diagnosis y el eje X es la entrada de registro del conjunto de prueba*

La representación del árbol del predicado difuso encontrada para estos experimentos se muestra en la Figura 5.14.

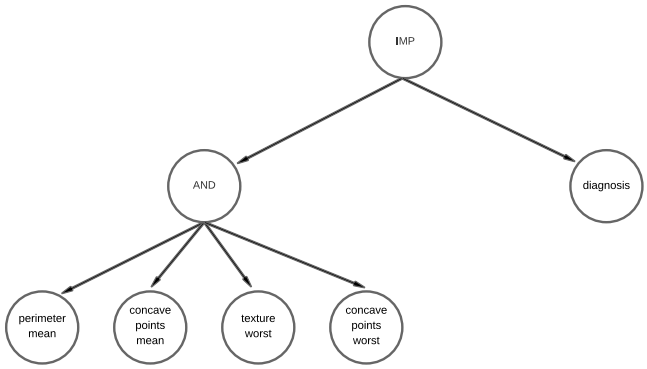


Figura 5.14: *Representación en árbol del predicado difuso*

En la Tabla 5.21 y 5.22 se muestran los predicados descubiertos y evaluados durante la etapa de entrenamiento.

Tabla 5.21: *Predicados del entrenamiento de conjunto Cáncer de mama con GCLV*

Valor de verdad	Predicado
0.998097	*(IMP(AND "PerimeterMean" ÇoncavePointsMean" "TextureWorst" ÇoncavePointsWorst") "Diagnosis")*
0.993702	*(IMP(AND "PerimeterMean" ÇoncavePointsMean" "TextureWorst" ÇoncavePointsWorst") "Diagnosis")*
0.994342	*(IMP(AND "PerimeterMean" ÇoncavePointsMean" "TextureWorst" ÇoncavePointsWorst") "Diagnosis")*

Tabla 5.22: *Predicados del entrenamiento de conjunto Cáncer de mama con FPG*

Valor de verdad	Predicado
0.999999	*(IMP(AND "PerimeterMean" ÇoncavePointsMean" "TextureWorst" ÇoncavePointsWorst") "Diagnosis")*
0.999932	*(IMP(AND "PerimeterMean" ÇoncavePointsMean" "TextureWorst" ÇoncavePointsWorst") "Diagnosis")*
0.998333	*(IMP(AND "PerimeterMean" ÇoncavePointsMean" "TextureWorst" ÇoncavePointsWorst") "Diagnosis")*

Para EU-NN-ACFL con el uso de FPG se descubre un juego de parámetros donde cada una de éstos podría describir los predicados para la clase de Cáncer de mama. Los valores de estos se muestran a continuación.

Tabla 5.23: *Valores de atributos de los predicados de la Tabla 5.22*

Estado	Beta	Gamma	m
Predicado 1			
perimeterMean	0.70100306	0.73169844	0.66018533
concavePointsMean	-0.04292968	: 0.22799494243914	0.13739839
textureWorst	0.23696513	0.51747212	0.38208624
concavePointsWorst	0.20030844	0.93887717	0.5355966
Predicado 2			
perimeterMean	0.0678739	0.44157701	0.02265232
concavePointsMean	0.61506095	0.77428296	0.58374226
textureWorst	0.01039618	0.04042832	0.46855519
concavePointsWorst	0.08360526	0.39070766	0.33003777
Predicado 3			
perimeterMean	0.6413921	0.8422007	0.87473191
concavePointsMean	-0.13421618	0.16643943	0.20530956
textureWorst	-0.01829336	0.30878264	0.90657489
concavePointsWorst	0.09940024	0.30255693	0.26180185

Capítulo 6

Conclusiones

6.1. Conclusiones

En esta tesis se cumple el objetivo de desarrollar un algoritmo de descubrimiento de conocimiento Genetic Algorithm Neural Network and Archimedean Compensatory Fuzzy Logic, que utiliza la Lógica Difusa Arquimediana-Compensatoria como su nombre lo indica, así como el uso de predicados de la forma normal conjuntiva, y funciones de pertenencia asociados a la lógica (GCLV y FPG). Utilizando técnicas de aprendizaje automático para el descubrimiento de predicados y optimización de parámetros de las funciones de pertenencia. Siendo este trabajo original por que es el primer algoritmo que integra el uso de una Lógica Difusa Arquimediana Compensatoria con el uso de Redes Neuronales para la generación de predicados, que apoyan a la inferencia de datos.

6.2. Contribuciones

En particular se desarrollo una nueva arquitectura para el núcleo Eureka Universe que brinda un soporte a la inclusión de trabajos basados en redes neuronales y predicados de ACFL. Además:

1. Este documento de tesis describe la integración de los algoritmos genéticos EK-CFL y AG-GSF para contar con un algoritmo de descubrimiento de conocimiento de referencia (ver sección 4.3.1).

2. Incrementar la precisión en términos tasa de error en clasificación.

3. Mantener la interpretabilidad en los predicados que otorga de manera inherente el uso de la Lógica Difusa Compensatoria Arquimediana.

4. De acuerdo con los resultados obtenidos de las experimentaciones:

   Los resultados de la experimentación muestran que el algoritmo GA-NN-ACFL con la incorporación del uso de una Variable lingüística Continua Generalizada mejora al algoritmo de referencia, ya que presenta una mejora en términos de calidad de valor de verdad con respecto a todas las instancias desde 9 % al 30 % de ellas (ver sección 5).

5. Métodos adicionales: nueva Neurona Difusa a los actuales del Estado del arte, implementación de preprocesamiento de datos para series temporales, desarrollo de capas de operadores de primer orden.

6.3. Trabajo futuro

1. Incrementar la complejidad de orden de predicados, utilizando disyunción de conjunciones.

2. Precisar mejor el rol del algoritmo de búsqueda de las funciones de pertenencia, los parámetros de la configuración, y explorar otras formas de búsqueda y/o optimización para este paso de la búsqueda de los parámetros de las funciones de pertenencia. Así como de los parámetros de Base y exponentes propias de la ACFL.

3. Explorar otro tipo de redes neuronales como las de aprendizaje profundo.

Referencias

Adam, S. P., Alexandropoulos, S.-A. N., Pardalos, P. M., y Vrahatis, M. N. (2019). No free lunch theorem: A review. *Approximation and optimization*, 57–82.

Alur, R., Dang, T., y Ivančić, F. (2006). Predicate abstraction for reachability analysis of hybrid systems. *ACM transactions on embedded computing systems (TECS)*, *5*(1), 152–199.

Andrade, R. A. E., González, E. F., y Caballero, E. G. (2014). Un sistema lógico para el razonamiento y la toma de decisiones: la lógica difusa compensatoria basada en la media geométrica. *Investigación operacional*, *32*(3), 230–245.

Andrade, R. A. E., Pérez, R. B., Ortega, A. C., Gómez, J. M., y Valdés, A. R. (2014). *Soft computing for business intelligence*. Springer.

Andrews, R., Diederich, J., y Tickle, A. B. (1995). Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based systems*, *8*(6), 373–389.

Astrom, K. J., y Wittenmark, B. (1984). *Computer controlled systems: theory and design*. Prentice Hall Professional Technical Reference.

Bau, D., Zhu, J.-Y., Strobelt, H., Lapedriza, A., Zhou, B., y Torralba, A. (2020). Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, *117*(48), 30071–30078.

Bengio, Y., Goodfellow, I., y Courville, A. (2017). *Deep learning* (Vol. 1). MIT press Massachusetts, USA:.

Boyd, S., y Vandenberghe, L. (2018). *Introduction to applied linear algebra: vectors, matrices, and least squares*. Cambridge university press.

Brownlee, J. (2016). Machine learning mastery with python. *Machine Learning Mastery Pty Ltd*, *527*, 100–120.

Cantú Cuéllar, R., Chávez Guzmán, L., y Cantú Mata, J. L. (2013). Método numérico heurístico para el cálculo de raíces de polinomios.

Carvalho, D. V., Pereira, E. M., y Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics*, *8*(8), 832.

CD-MAKE. (2018). *CD-MAKE cd-make 2019 workshop on explainable artificial intelligence*. Descargado Accessed: 2020-12-05, de `https://cd-make.net/special-sessions/make-explainable-ai`

Chakraborty, S., Tomsett, R., Raghavendra, R., Harborne, D., Alzantot, M., Cerutti, F., ... others (2017). Interpretability of deep learning models: a survey of results. En *2017 ieee smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, internet of people and smart city innovation (smartworld/scalcom/uic/atc/cbdcom/iop/sci)* (pp. 1–6).

Charu, C. A. (2018). *Neural networks and deep learning*.

Cruz-Reyes, L., Espin-Andrade, R. A., Irrarragorri, F. L., Medina-Trejo, C., Tristán, J. F. P., Martinez-Vega, D. A., y Peralta, C. E. L. (2019). Use of compensatory fuzzy logic for knowledge discovery applied to the warehouse order picking problem for real-time order batching. En *Handbook of research on metaheuristics for order picking optimization in warehouses to smart cities* (pp. 62–88). IGI Global.

CVPR. (2019). *CVPR cvpr 19—workshop on explainable ai.* Descargado Accessed: 2020-03-12, de https://explainai.net/

Dong, M., Huang, L., Wu, X., y Zeng, Q. (2020). Application of least-squares method to time series analysis for 4dpm matrix. En *Iop conference series: Earth and environmental science* (Vol. 455, p. 012200).

Doshi-Velez, F., y Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Espin-Andrade, R. A., González Caballero, E., Pedrycz, W., y Fernández González, E. R. (2015). Archimedean-compensatory fuzzy logic systems. *International Journal of Computational Intelligence Systems*, *8*(sup2), 54–62.

FAT/ML. (2014). *Fairness, accountability, and transparency in machine learning.* (http://www.fatml.org/)

Fayyad, U., Piatetsky-Shapiro, G., y Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, *17*(3), 37–37.

Fernandez, A., Herrera, F., Cordon, O., del Jesus, M. J., y Marcelloni, F. (2019). Evolutionary fuzzy systems for explainable artificial intelligence: why, when, what for, and where to? *IEEE Computational Intelligence Magazine*, *14*(1), 69–81.

Frawley, W. J., Piatetsky-Shapiro, G., y Matheus, C. J. (1992). Knowledge discovery in databases: An overview. *AI magazine*, *13*(3), 57–57.

Freitas, A. A. (2014). Comprehensible classification models: a position paper. *ACM SIGKDD explorations newsletter*, *15*(1), 1–10.

Gatica, G., Reyes, P., Contreras-Bolton, C., Linfati, R., y Escobar, J. W. (2016). An algorithm for the strip packing problem obtained by extracting expert skills using data mining. *Ingeniería, investigación y tecnología*, *17*(2), 179–190.

Goodwin, G., y Sin, K. (1984). Adaptive filtering prediction and control". prentice hall inc., englewood cliffs, new jersey, usa.

Gunning, D. (2017). Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, *2*(2).

Hand, D. J., y Adams, N. M. (2014). Data mining. *Wiley StatsRef: Statistics Reference Online*, 1–7.

Haussler, D. (1988). Quantifying inductive bias: Ai learning algorithms and valiant's learning framework. *Artificial intelligence*, *36*(2), 177–221.

He, C., Ma, M., y Wang, P. (2020). Extract interpretability-accuracy balanced rules from artificial neural networks: A review. *Neurocomputing*, *387*, 346–358.

Herrera, F. (2006). Introducción a los algoritmos metaheurísticos grupo de investigación soft computing and intelligent information systems. *Dpto. Ciencias de la Computación e IA Universidad de Granada, ESPAÑA*.

Herrera, F. (2017). Introducción a los algoritmos metaheurísticos.

Hoffmann, J., y Magazzeni, D. (2019). Explainable ai planning (xaip): overview and the case of contrastive explanation. *Reasoning Web. Explainable Artificial Intelligence*, 277–282.

Holzinger, A., Kieseberg, P., Tjoa, A. M., y Weippl, E. (2019). *Machine learning and knowledge extraction: Third ifip tc 5, tc 12, wg 8.4, wg 8.9, wg 12.9 international cross-domain conference, cd-make 2019, canterbury, uk, august 26–29, 2019, proceedings* (Vol. 11713). Springer Nature.

Howson, C., Sallam, R. L., Richardson, J. L., Tapadinhas, J., Idoine, C. J., y Woodward, A. (2018). Magic quadrant for analytics and business intelligence platforms. *Retrieved Aug*, *16*, 2018.

ICAPS. (2018). *ICAPS icaps 2018—workshop on explainable ai planning (xaip).* Descargado Accessed: 2020-10-21, de http://icaps18.icapsconference.org/xaip/

IJCAI. (2017). *IJCAI ijcai 2017—workshop on explainable artificial intelligence (xai).*

Descargado Accessed: 2020-06-01, de http://home.earthlink.net/~dwaha/research/meetings/ijcai17-xai/

IJCAI. (2018). *IJCAI 2018—workshop on explainable artificial intelligence (xai).* Descargado Accessed: 2020-07-03, de http://home.earthlink.net/~dwaha/research/meetings/faim18-xai/

IJCNN. (2017). *IJCNN ijcnn 2017 explainability of learning machines.* Descargado Accessed: 2020-06-01, de http://gesture.chalearn.org/ijcnn17_explainability_of_learning_machines

IPMU. (2018). *IPMU ipmu 2018—advances on explainable artificial intelligence.* Descargado Accessed: 2020-12-05, de http://ipmu2018.uca.es/submission/cfspecial-sessions/special-sessions/#explainabl/

Ishibuchi, H., y Nojima, Y. (2007). Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. *International Journal of Approximate Reasoning*, *44*(1), 4–31.

Jang, J.-S. (1993). Anfis: adaptive-network-based fuzzy inference system. *IEEE transactions on systems, man, and cybernetics*, *23*(3), 665–685.

Juang, C.-F., y Chen, C.-Y. (2012). Data-driven interval type-2 neural fuzzy system with high learning accuracy and improved model interpretability. *IEEE transactions on cybernetics*, *43*(6), 1781–1795.

Kim, B., Koyejo, O., Khanna, R., y cols. (2016). Examples are not enough, learn to criticize! criticism for interpretability. En *Nips* (pp. 2280–2288).

Kim, B., Malioutov, D. M., Varshney, K. R., y Weller, A. (2017). Proceedings of the 2017 icml workshop on human interpretability in machine learning (whi 2017). *ArXiv e-prints*, arXiv–1708.

Kim, J., y Kasabov, N. (1999). Hyfis: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems. *Neural networks*, *12*(9), 1301–1319.

LeCun, Y., Bengio, Y., y Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

Lim, B., Sarkar, A., Smith-Renner, A., y Stumpf, S. (2019). Exss: Explainable smart systems 2019. En *Proceedings of the 24th international conference on intelligent user interfaces: Companion* (pp. 125–126).

Lim, B., Smith, A., y Stumpf, S. (2018). Exss 2018: Workshop on explainable smart systems. En *Ceur workshop proceedings* (Vol. 2068).

Llorente Peralta, C. E., y cols. (2019). Algoritmo evolutivo para descubrir conocimiento de asociación usando lógica difusa compensatoria.

Mathivet, V. (2018). *Inteligencia artificial para desarrolladores: conceptos e implementación en c.* Ediciones ENI.

Mendel, J. M. (2004). Computing derivatives in interval type-2 fuzzy logic systems. *IEEE Transactions on Fuzzy Systems*, *12*(1), 84–98.

Meshino, G. (2008). Icpro framework de análisis de datos con técnicas de inteligencia computacional (version 1.0). *Mar del Plata*.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, *267*, 1–38.

Mining, K. D. T. D. (1996). What is knowledge discovery. *Tandem Computers Inc*, *253*.

Minsky, M., y Papert, S. A. (2017). *Perceptrons: An introduction to computational geometry.* MIT press.

Molnar, C. (2020). *Interpretable machine learning.* Lulu. com.

Mora-Flórez, J. J., Echaverri, M. G., y Castañeda, L. E. M. (2005). Los métodos de representación del conocimiento en inteligencia artificial y su integración en sistemas híbridos de localización de fallas. *Tecnura*, *9*(17), 98–109.

Nikam, S. S. (2015). A comparative study of classification techniques in data mining algorithms. *Oriental journal of computer science & technology*, *8*(1), 13–19.

NSF. (1987). *Uc irvine machine learning repository, center for machine learning and intelligentsystems. uc irvine machine learning repository.* (`https://archive.ics.uci.edu/ml/index.php`)

of Health, N. I., y cols. (2014). *Neurons, brain chemistry, and neurotransmission.* Retrieved March.

Osório, F. S. (1998). *Inss: un système hybride neuro-symbolique pour l'apprentissage automatique constructif* (Tesis Doctoral no publicada). Institut National Polytechnique de Grenoble-INPG.

Padrón, T. (2017). Sistema inteligente para descubrimiento de conocimiento basado en lógica difusa compensatoria.

Padrón, T. (2020). Algoritmo de virtual savant basado en lógica difusa compensatoria para problemas de empacado de objetos (tesis de máster no publicada).

Pereira-Fariña, M., y Reed, C. (2017). Proceedings of the 1st workshop on explainable computational intelligence (xci 2017). En *Proceedings of the 1st workshop on explainable computational intelligence (xci 2017).*

Pérez, A. G. (1994). *Modelo de diseño orientado a marcos para sistemas basados en el conocimiento* (Tesis Doctoral no publicada). Universidad Politécnica de Madrid.

Pizarro, P., Alberto, C., y cols. (2020). Regresión lineal con errores localmente estacionarios lsma.

Rai, A. (2020). Explainable ai: From black box to glass box. *Journal of the Academy of Marketing Science*, *48*(1), 137–141.

Rodríguez-Fdez, I., Canosa, A., Mucientes, M., y Bugarín, A. (2015). Stac: a web platform for the comparison of algorithms using statistical tests. En *2015 ieee international conference on fuzzy systems (fuzz-ieee)* (pp. 1–8).

Rodríguez Ortiz, C., y cols. (2010). Algoritmos heurísticos y metaheurísticos para el problema de localización de regeneradores.

Ross, T. J. (2010). Properties of membership functions, fuzzification, and defuzzification. *Fuzzy logic with engineering applications*, 89–116.

Rudin, C. (2018). Please stop explaining black box models for high stakes decisions. *arXiv preprint arXiv:1811.10154*, *1*.

Rumelhart, D. E., Hinton, G. E., y Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Rüping, S., y cols. (2006). Learning interpretable models.

Shapiro, G. P., y Frawley, W. (1991). Knowledge discovery in databases. *AAAI/MIT Press, Cambridge, MA*.

Siler, W., y Buckley, J. J. (2005). *Fuzzy expert systems and fuzzy reasoning.* Wiley Online Library.

Singh, H., y Lone, Y. A. (2020). *Deep neuro-fuzzy systems with python.* Springer.

Skansi, S. (2018). *Introduction to deep learning: from logical calculus to artificial intelligence.* Springer.

Stoyanov, D., Taylor, Z., Kia, S. M., Oguz, I., Reyes, M., Martel, A., ... others (2018). *Understanding and interpreting machine learning in medical image computing applications: First international workshops, mlcn 2018, dlf 2018, and imimic 2018, held in conjunction with miccai 2018, granada, spain, september 16-20, 2018, proceedings* (Vol. 11038). Springer.

Strobach, P. (2012). *Linear prediction theory: a mathematical basis for adaptive systems* (Vol. 21). Springer Science & Business Media.

Studio, F. T. (2018). *FTS fuzzy three studio.* Descargado Accessed: 2020-09-24, de `https://www.researchgate.net/figure/Fuzzy-Tree-Studio-Environment_fig5_224049046`

Timarán, R. (2009). Una mirada al descubrimiento de conocimiento en bases de datos. *Ventana Informática*, *20*, 39–58.

Treviño, I. V. H. G. (2007). *Muestreo estadístico aplicado al estudio del desempeño de algoritmos heurísticos* (Tesis Doctoral no publicada). INSTITUTO TECNOLÓGICO DE CIUDAD MADERO.

Tung, W. L., y Quek, C. (2009). A mamdani-takagi-sugeno based linguistic neural-fuzzy inference system for improved interpretability-accuracy representation. En *2009 ieee international conference on fuzzy systems* (pp. 367–372).

Universe, E. (2021). *EU eureka universe.* Descargado Accessed: 2020-12-18, de `https://www.eurekascommunity.org/`

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, *27*(11), 1134–1142.

Wilson, A. G., Kim, B., y Herlands, W. (2016). Proceedings of nips 2016 workshop on interpretable machine learning for complex systems. *arXiv preprint arXiv:1611.09139*.

Wilson, A. G., Yosinski, J., Simard, P., Caruana, R., y Herlands, W. (2017). Proceedings of nips 2017 symposium on interpretable machine learning. *arXiv e-prints*, arXiv–1711.

Zhang Q, Z. B., Fan L. (2019). *network interpretability for deep learning.* Descargado Accessed: 2020-03-12, de `http://networkinterpretability.org/`

Zhu, W., Huang, K., Xiao, X., Liao, B., Yao, Y., y Wu, F.-X. (2020). Alsbmf: Predicting lncrna-disease associations by alternating least squares based on matrix factorization. *IEEE Access*, *8*, 26190–26198.

Apéndice A

Experimentación con series temporales de Covid-19

Experimento de la segunda serie temporal de registros sobre las afecciones causadas por el virus covid-19 en las ciudades Tampico, Madero y Altamira. Estos registros a diferencia de del Series Temporales TMA, no se han separado por días, si no que se utilizan cada uno de los registros de ingresos de casos a lo largo del día durante el período de Abril-Octubre de 2020, justo como se muestra en la Tabla A.6.

En las Tablas A.1 y A.2 se muestran los valores de verdad mas altos obtenidos para cada experimentación.

Tabla A.1: *Valor de verdad de Series Históricas 2 TMA-Covid-19*

	Valores de verdad (GCLV)					
No. de corrida	**Confirmados**	**Negativos**	**Sospechosos**	**Muertos**	**Recuperados**	**Activos**
1	0.99741276	1	0.99998306	0.99849893	0.9962698	0.98886071
2	0.99995862	0.98546514	0.9879245	0.99999772	0.98312725	0.99970646
3	0.9999494	0.99013817	0.99978394	0.99357652	0.96040301	0.99740535
4	0.98825345	0.98476625	0.99999493	0.99380152	0.9770656	0.98307776
5	0.99998193	0.99999984	0.9941357	0.99381243	0.99964352	0.99606285
6	0.98420133	0.992289	0.99840802	0.99944051	0.99999255	0.99975944
7	0.99811099	0.99999311	0.99908121	0.99988815	0.99999998	0.99493833
8	0.99972274	0.99896382	0.99981628	0.996694	0.99950408	0.99286599
9	0.99999884	0.99487721	0.99990843	0.99976445	0.99949232	0.99718932
10	0.99785618	0.99861055	1	0.99999957	0.9976379	0.9863568
11	0.99338402	0.99364519	0.99999814	0.99997996	0.98342049	0.9999045
12	0.98938121	0.99999935	0.99935855	0.99656489	0.995657	0.99902045
13	0.99719784	0.99635898	0.99998846	0.99344138	0.97117005	0.9999465
14	0.98029064	0.99939566	0.99999053	0.99958454	0.99999898	0.99732106
15	0.9996407	1	0.99999983	0.98823038	0.99442838	0.99973327
16	0.9999735	0.99819292	0.99715631	0.99999868	0.99657842	0.99998758
17	0.99253741	0.9999942	0.99999825	0.99862092	0.99312925	0.988683
18	0.99881404	0.99917258	0.9971033	0.99914128	0.98637596	0.99880192
19	0.98399575	0.98798997	0.99997065	0.99683927	0.99999998	0.98584796
20	0.99999617	0.99474841	0.99587625	0.99995939	0.99887409	0.99984756
21	0.99781446	0.99537851	1	0.99971559	0.99783151	0.99859118
22	0.99999931	0.99582165	0.99949956	0.99566896	1	0.99509939
23	0.99996529	0.99523194	0.99996769	0.99891291	0.99985963	0.99618368
24	0.99869722	1	0.99628687	0.99265154	0.99937334	0.99709715
25	0.99835293	0.98426852	0.99996317	0.98904969	0.99724258	0.99999997
26	0.99997945	0.99338297	0.99999474	0.99409445	0.99931375	0.99784106
27	0.9997724	0.99024485	0.99937303	0.9975046	0.99759338	0.99558693
28	0.99999992	0.99865291	0.99992454	0.99999998	0.99005599	0.99902881
29	0.99768913	0.99974736	0.99990602	0.99632495	0.99310403	0.99861377
30	0.99136367	0.99645693	0.99974441	0.98752418	1	0.99655604
Promedio	**0.99614304**	**0.99545953**	**0.99877121**	**0.99664271**	**0.99357143**	**0.99599716**

Tabla A.2: *Valor de verdad de Series Históricas 2 TMA-Covid-19*

	Valores de verdad (FPG)					
No. de corrida	**Confirmados**	**Negativos**	**Sospechosos**	**Muertos**	**Recuperados**	**Activos**
1	0.99515354	0.99999986	0.99192676	0.99613516	0.99992796	0.99999941
2	0.99986818	0.99867335	0.99620794	0.99682081	0.99927554	0.99883323
3	0.99823217	0.99998941	0.99981881	0.9996999	0.99531962	0.99419754
4	0.99840813	0.99997071	0.99130219	0.99948361	1	0.99995334
5	0.99760377	0.99984866	0.99983272	0.99999877	0.99987072	0.99982649
6	0.98740237	0.99896431	0.9999836	1	0.99991047	0.99924939
7	0.99971888	0.99809609	0.99167796	0.99917206	0.99993338	0.999996
8	0.99428453	0.99999972	0.99795128	0.99956422	0.99999168	0.99900508
9	0.99989852	0.99048738	0.99987844	0.99872782	0.99999846	0.99999997
10	0.99998611	0.99682656	0.99831287	0.99884077	0.99985227	0.99893866
11	0.9999562	0.99904184	0.99852153	0.99958309	0.99996526	0.99438235
12	0.99993683	0.99987196	0.9954059	0.99951405	0.99999315	0.99792404
13	0.99995673	0.99984173	0.98518936	0.99891829	0.9997385	0.99999905
14	0.99534176	0.9996715	0.99993322	0.99681831	0.99999999	0.98427088
15	0.99440163	0.99924465	0.99984959	0.99999684	0.99962401	0.99457441
16	0.99431762	0.99952059	0.99939002	0.99999064	0.99992073	0.99979029
17	0.99966831	0.99999295	0.99143036	0.99995955	0.99997952	0.99994656
18	0.99753045	0.99289541	0.99999851	0.99428857	0.99999616	0.99995752
19	0.99060012	0.99006819	0.99977453	0.99999997	0.97406302	0.99815852
20	0.99997034	0.99996326	0.99999466	0.9996179	0.99947482	0.98650187
21	0.99997241	0.99425515	0.99349306	0.99785197	0.99272555	0.99979902
22	0.99740144	0.99999954	0.99366502	0.99999548	0.99999998	0.99986756
23	0.99984661	0.99882237	0.99524714	0.99667924	0.99995799	0.99999915
24	0.99993208	0.99954662	0.9862852	0.99998816	0.9997678	0.99887728
25	0.99976252	0.99094516	0.99216561	0.99832338	0.99944484	0.99110891
26	0.99524385	0.99998642	0.99710755	0.9933359	0.99999987	0.99917845
27	1	0.99628247	0.99983524	0.9931022	0.99706812	0.99936524
28	0.99514812	0.99828174	0.99997117	0.99749938	0.99729701	0.99999964
29	0.99999639	0.99300265	0.99885331	0.99998026	0.99998155	0.99672683
30	0.99987979	0.99999183	0.99953208	0.9986097	0.99998973	0.98052997
Promedio	**0.99764731**	**0.99780274**	**0.99641785**	**0.99841653**	**0.99843559**	**0.99703189**

De la misma manera en las Tablas A.3 y A.4 se muestran la tasa de errores obtenidos para cada entrenamiento dentro de las configuraciones establecidas para cada red.

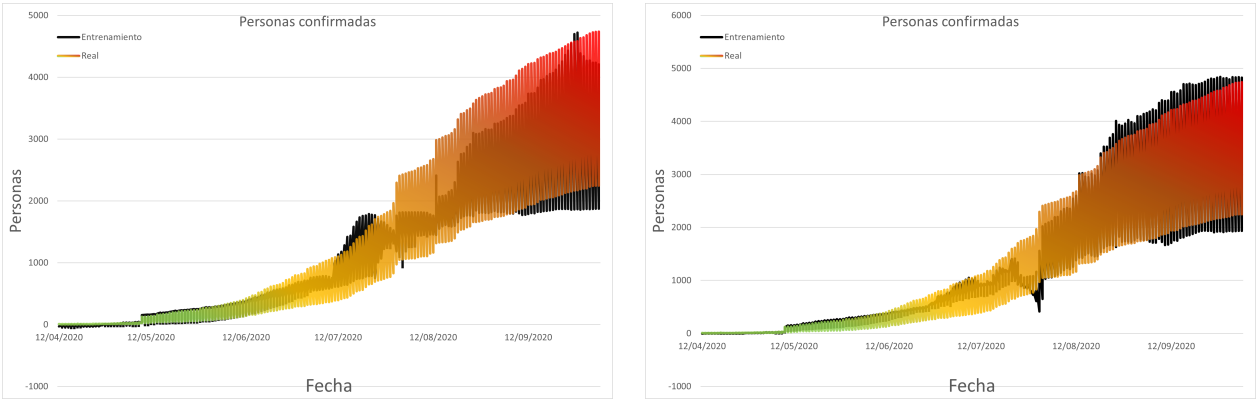
Tabla A.3: *Error de clasificación de Serie Histórica 2 TMA-Covid-19*

	Error (GCLV)					
No. de corrida	**Confirmados**	**Negativos**	**Sospechosos**	**Muertos**	**Recuperados**	**Activos**
1	8.367483149	10.8504886	4.66932944	3.4838613	4.41165245	12.4865111
2	4.050122906	11.5288191	3.54814302	4.82573505	5.90913828	12.8009412
3	6.609676372	11.2467158	7.02991808	8.70897112	4.7221882	11.8428885
4	4.957136007	10.2597761	4.1659204	4.24599384	4.90722276	12.3052205
5	8.190045232	10.5717621	6.57951374	5.65084242	6.02658945	12.5724314
6	4.83195891	10.5526137	4.35602843	9.65527736	5.30632793	11.9025555
7	3.218401114	10.595653	7.8497383	6.14427707	5.23560939	13.1447509
8	4.578878369	11.6853138	4.50552553	4.98778915	4.79452944	12.1203255
9	5.188368047	10.8603646	8.24232545	2.30526217	3.68427385	12.9465187
10	5.220975418	10.210873	5.01512767	3.25021269	5.84515513	13.8258918
11	4.764456009	11.0010322	7.39472787	1.36045552	4.46140907	14.4135272
12	8.912943183	11.1342208	3.42425094	4.78159301	4.86323723	11.7705897
13	9.156754889	11.2887083	5.36843336	9.70845129	6.26040032	12.915457
14	2.558196306	11.2043141	5.68709849	6.44729789	7.66785103	13.8956555
15	10.5068569	10.0219462	4.74518272	2.9250705	8.44924919	12.7240572
16	10.70606248	10.3828515	5.58821518	7.53970881	7.99594373	12.8579921
17	4.347044687	10.3650323	4.22095425	1.82084948	6.33072362	12.6517288
18	5.347375891	10.8482341	3.56441753	3.0760536	9.33741584	14.6129891
19	4.483140426	10.9675599	7.3683273	4.92427063	5.3369354	13.6522374
20	4.496316369	11.0728795	4.07864355	9.83608323	5.06910371	12.1631683
21	4.888547668	10.9064766	5.05239551	8.5685619	5.08583452	12.9826442
22	4.775073606	10.3344048	5.82739421	8.20520621	2.9232171	12.4381598
23	7.602013712	9.81955843	4.1722334	1.55925324	9.25349858	12.1094385
24	8.91193488	11.2541161	5.07919191	3.24298937	11.4512654	12.3011348
25	9.691769089	10.5553662	5.69465786	1.83245344	2.71360254	12.8387827
26	3.452076761	10.4092801	4.90577962	3.68596089	4.11983189	14.3092924
27	2.798727422	10.4543448	3.88201042	4.3457132	4.30821078	13.6311216
28	4.38525375	11.0453148	5.59888322	3.40789917	2.87263425	12.8973148
29	6.561660809	9.8561188	5.57111343	3.82065411	6.27061216	12.1102257
30	4.40649042	10.7057915	5.59490327	1.67886461	3.84602873	13.4438514
Promedio	**5.932191359**	**10.7329977**	**5.29267947**	**4.86752041**	**5.6486564**	**12.8889134**

Tabla A.4: *Error de clasificación de Serie Histórica 2 TMA-Covid-19*

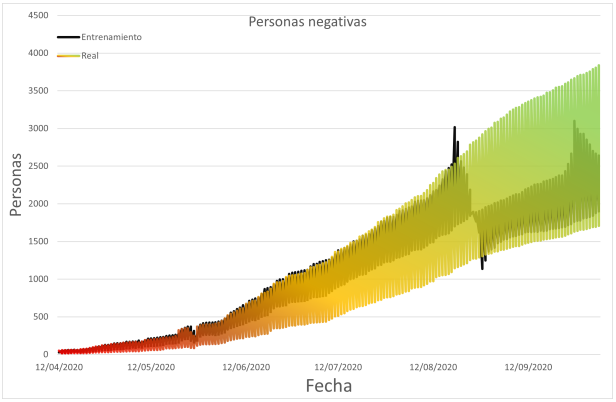
	Error (FPG)					
No. de corrida	**Confirmados**	**Negativos**	**Sospechosos**	**Muertos**	**Recuperados**	**Activos**
1	1.461969919	10.4306546	2.76039445	4.23765494	2.74349993	12.9174367
2	1.031219326	9.80893681	3.78273581	1.41607772	5.04623987	11.6921428
3	2.044106966	10.0941927	4.03263803	2.99211389	2.29201286	12.5881643
4	1.686045388	9.57480215	6.72562117	0.87285966	2.33954209	12.3302233
5	7.254808477	11.0984054	2.81001435	3.23303446	2.93420539	12.0385977
6	3.293479806	9.22156771	4.73842623	2.58990141	1.87324764	11.9384801
7	2.70140359	9.19216876	3.57106903	0.92232687	3.81590676	11.7805583
8	2.586947505	9.66695339	5.99388898	9.04991504	2.26255469	13.6230011
9	3.175077066	9.26960192	5.25149545	6.74931272	1.82447881	12.9004183
10	4.439369555	9.67046572	3.70527479	1.02574996	6.89600102	12.0244396
11	4.459346892	9.45636788	3.40366846	5.99830184	5.42033568	12.2477132
12	7.996877049	9.6556448	3.52062867	0.79734492	2.43653605	11.9197018
13	4.752693517	9.37274405	6.1992666	5.87625195	2.21278936	13.0148257
14	2.434287649	10.4978647	2.65914276	1.58684653	8.64722972	12.2813586
15	8.666531738	9.02821682	3.57713471	1.29968019	3.19431988	11.5378589
16	2.17998714	10.4499576	4.24264813	1.24788248	3.86415455	13.8680555
17	5.16977708	9.11592609	5.81921823	3.64930105	2.10319838	11.3987329
18	2.063189737	9.86638428	3.80852352	4.11342576	4.90015208	14.3654743
19	4.12156355	10.6001474	3.83654116	1.57489326	1.95098566	12.1070574
20	10.13236085	9.82506861	5.61659352	1.52022454	2.44345661	13.3160049
21	2.665424488	9.77772936	5.77237276	1.96469694	1.23652546	12.5152263
22	5.863712615	10.9444868	5.98775117	1.38414174	5.46291194	11.5875118
23	4.035544242	9.67211914	3.16554556	5.25641839	4.20072476	13.1719051
24	2.193306117	9.96326832	3.14618355	2.60482168	3.95195052	13.9203783
25	6.855035824	10.0574279	3.42285284	0.77321744	4.2905209	14.2783224
26	7.264606267	9.9236895	4.41789936	4.39267822	2.65908559	14.5601405
27	1.591512455	10.3990366	5.78104261	4.374082	2.68581752	13.291258
28	3.852102365	10.4536717	3.62809422	1.24883228	8.68426804	13.1109734
29	5.536043677	11.2779704	4.04182022	1.12541416	1.57429669	11.1247919
30	1.742859765	10.0169806	3.79754872	2.27906143	1.61899875	13.7423342
Promedio	**4.10837302**	**9.94608173**	**4.30720117**	**2.87188212**	**3.51886491**	**12.7064362**

Las siguientes figuras representan la salida obtenida por el entrenamiento de las redes y los datos reales del conjunto de datos A.6.

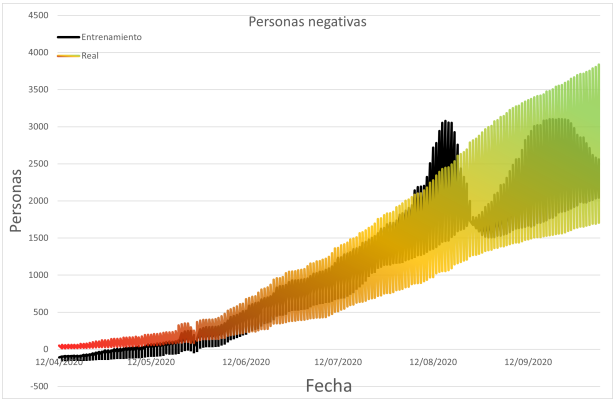


(a) *Resultados utilizando GCLV* **(b)** *Resultados utilizando FPG*

Figura A.1: *Personas confirmadas de tener el virus Covid-19*

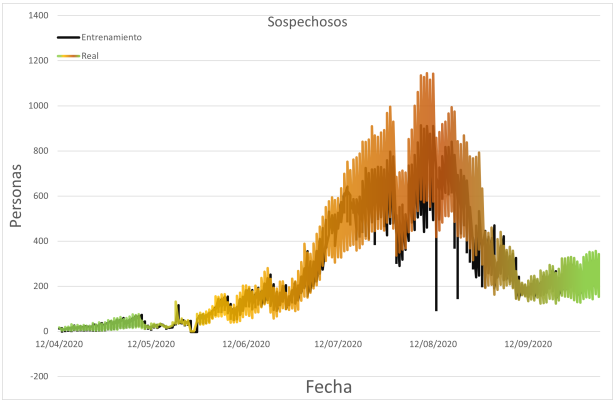


(a) *Resultados utilizando GCLV*

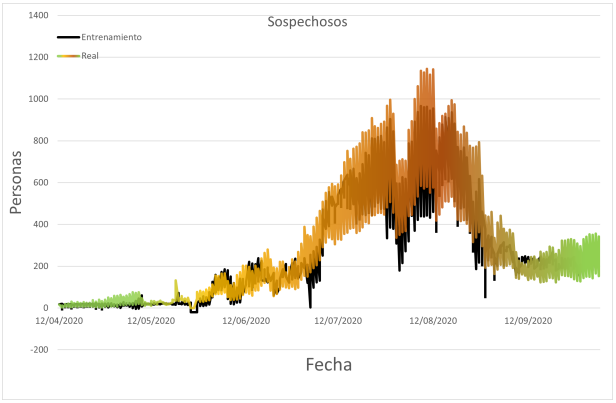


(b) *Resultados utilizando FPG*

Figura A.2: *Personas negativas de tener el virus Covid-19*

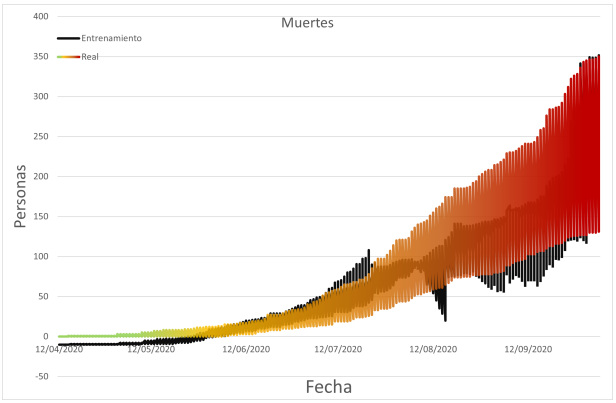


(a) *Resultados utilizando GCLV*

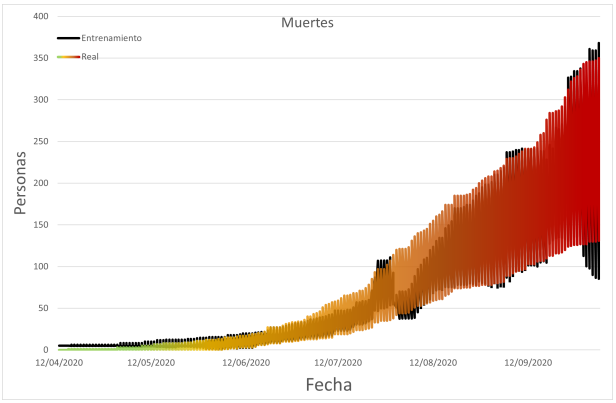


(b) *Resultados utilizando FPG*

Figura A.3: *Personas con sospecha de tener el virus Covid-19*

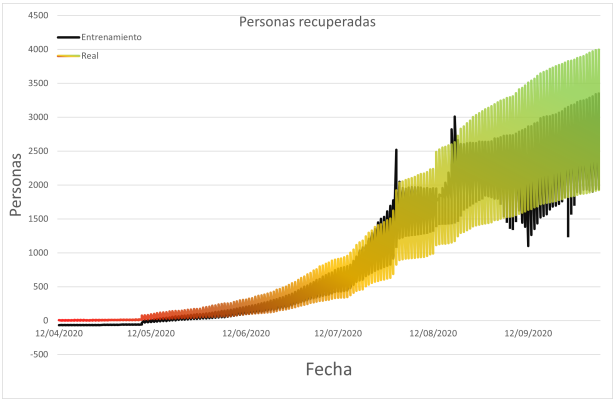


(a) *Resultados utilizando GCLV*

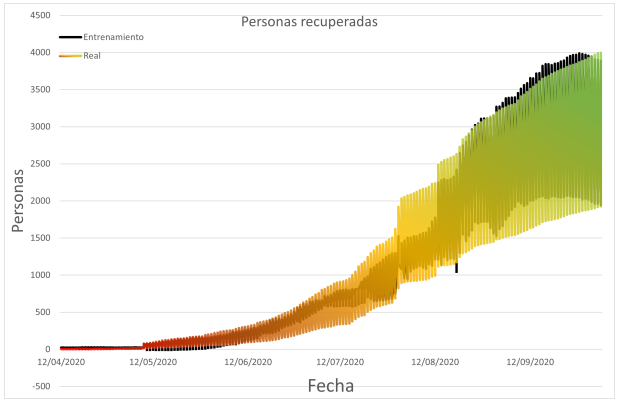


(b) *Resultados utilizando FPG*

Figura A.4: *Personas muertas tras el virus Covid-19*

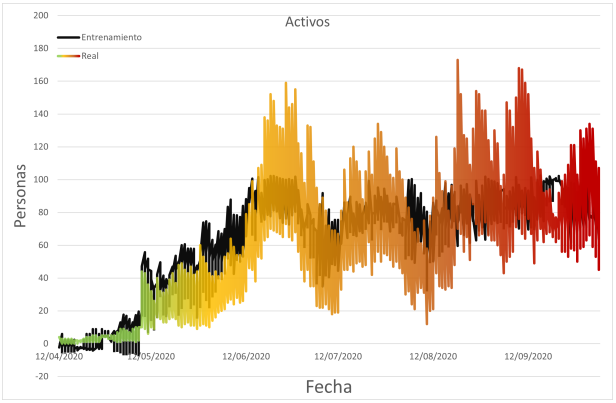


(a) *Resultados utilizando GCLV*

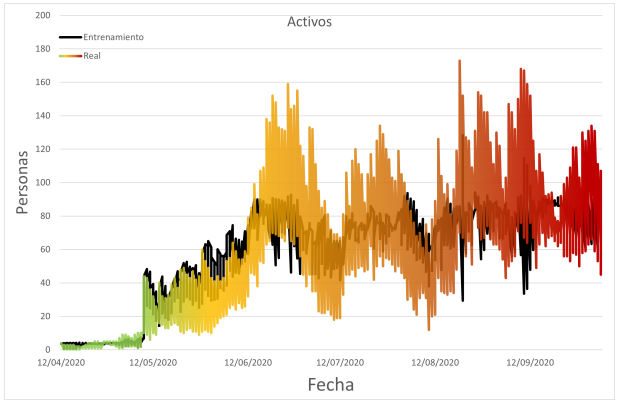


(b) *Resultados utilizando FPG*

Figura A.5: *Personas recuperadas de tener el virus Covid-19*



(a) *Resultados utilizando GCLV*



(b) *Resultados utilizando FPG*

Figura A.6: *Personas con el virus Covid-19 activo*

A partir de la experimentación se establece la siguiente Hipótesis nula (*H0*): La media de las diferencias entre los dos algoritmos son iguales. En la Tabla A.5 se observa que *H0* es rechazada en casos de personas confirmadas, negativas, sospechosas, muertes, y recuperados por el virus Covid-19 a favor de la configuración donde se utiliza la FPG. Y solamente es aceptada en los Activos de las series temporales donde no se muestra una diferencia significativa.

Tabla A.5: *Prueba de Wilcoxon(nivel de significancia de 0.05)*

Atributo	**Estadística**	**p-value**	**Resultados**
Confirmados	69	0.0007712	H0 es rechazada
Negativos	45	0.000115	H0 es rechazada
Sospechosos	115	0.0156585	H0 es rechazada
Muertos	107	0.0098421	H0 es rechazada
Recuperados	60	0.0003881	H0 es rechazada
Activos	192	0.4048347	H0 es aceptada

Tabla A.6: *Muestra de Conjunto de datos de Series Histórica 2 de personas afectadas por Covid-19*

DateUpdate	Confirmed	Negatives	Suspects	Deaths	Recovered	Active
12/04/2020	2	14	3	0	1	1
12/04/2020	5	38	17	0	4	1
12/04/2020	10	51	17	0	6	4
13/04/2020	2	14	3	0	2	0
13/04/2020	4	44	12	0	3	1
13/04/2020	11	58	12	0	8	3
14/04/2020	2	15	6	0	2	0
14/04/2020	4	47	14	0	3	1
14/04/2020	11	57	19	0	8	3
⋮	⋮	⋮	⋮	⋮	⋮	⋮
28/06/2020	295	386	90	11	201	68
28/06/2020	557	701	132	23	436	79
28/06/2020	796	1053	222	33	570	155
29/06/2020	297	397	97	11	213	56
29/06/2020	557	713	148	23	449	63
29/06/2020	796	1064	268	34	601	122
30/06/2020	298	398	136	12	224	45
30/06/2020	570	716	179	23	466	57
30/06/2020	821	1071	307	34	629	116
01/07/2020	302	399	154	12	235	38
01/07/2020	571	722	201	24	474	50
01/07/2020	826	1080	388	36	649	98
02/07/2020	320	409	163	13	246	45
02/07/2020	616	754	150	25	492	76
02/07/2020	896	1120	349	40	683	133
03/07/2020	326	421	184	13	262	35
03/07/2020	622	777	171	26	506	66
03/07/2020	921	1154	343	40	704	132
04/07/2020	335	424	191	13	271	35
04/07/2020	629	779	202	26	519	59
04/07/2020	939	1160	389	43	737	111
05/07/2020	335	428	205	13	282	23
05/07/2020	633	799	224	26	526	53
05/07/2020	946	1183	432	44	755	95
06/07/2020	344	430	229	14	292	22
⋮	⋮	⋮	⋮	⋮	⋮	⋮
02/10/2020	4705	3753	352	346	3948	134
03/10/2020	2243	1692	142	130	1905	82
03/10/2020	2778	2583	212	149	2380	59
03/10/2020	4727	3785	350	346	3969	131
04/10/2020	2246	1694	165	130	1918	71
04/10/2020	2780	2601	204	149	2387	53
04/10/2020	4732	3804	356	347	3988	111
05/10/2020	2248	1707	153	131	1926	63
05/10/2020	2785	2628	177	149	2399	45
05/10/2020	4739	3836	342	350	3997	107

ANEXO B

En la siguiente tabla se muestra un conjunto de prueba de Cáncer de mama donde se muestra la diferencia de clasificación (error) entre la clases real *diagnosis* y el resultado de la red utilizando GCLV y FPG, la desviación absoluta (MAD) de cada una de ellas.

	GCLV			FPG		
diagnosis	entrenamiento	error e	MAD $\left\|\left(Y_t - \hat{Y}_t\right)\right\|$	entrenamiento	error e	MAD $\left\|\left(Y_t - \hat{Y}_t\right)\right\|$
1	0.91799558	0.08200442	0.08200442	0.914902685	0.08509732	0.08509732
1	0.93892885	0.06107115	0.06107115	0.987626247	0.01237375	0.01237375
1	0.77624272	0.22375728	0.22375728	0.691230536	0.30876946	0.30876946
0	0.0083808	-0.0083808	0.0083808	-0.10573405	0.10573405	0.10573405
0	0.02956425	-0.02956425	0.02956425	0.052655374	-0.05265537	0.05265537
1	1.16674816	-0.16674816	0.16674816	1.149812212	-0.14981221	0.14981221
1	0.74485505	0.25514495	0.25514495	0.742365375	0.25763463	0.25763463
0	0.2501014	-0.2501014	0.2501014	0.187537808	-0.18753781	0.18753781
0	0.00034946	-0.00034946	0.00034946	-0.04333486	0.04333486	0.04333486
0	0.22635153	-0.22635153	0.22635153	0.137104925	-0.13710493	0.13710493
1	0.95353344	0.04646656	0.04646656	0.900990105	0.09900989	0.09900989
1	0.79671646	0.20328354	0.20328354	0.826943857	0.17305614	0.17305614
1	0.61862912	0.38137088	0.38137088	0.563143801	0.4368562	0.4368562
1	1.19521608	-0.19521608	0.19521608	0.993569596	0.0064304	0.0064304
0	0.65715563	-0.65715563	0.65715563	0.632393894	-0.63239389	0.63239389
0	-0.10240485	0.10240485	0.10240485	0.026067154	-0.02606715	0.02606715
0	0.07875526	-0.07875526	0.07875526	0.019736979	-0.01973698	0.01973698
1	0.96733014	0.03266986	0.03266986	0.998190057	0.00180994	0.00180994
0	-0.32310483	0.32310483	0.32310483	-0.00380157	0.00380157	0.00380157
0	0.07592735	-0.07592735	0.07592735	-0.04048751	0.04048751	0.04048751
1	1.10082516	-0.10082516	0.10082516	1.084207456	-0.08420746	0.08420746
1	1.09212061	-0.09212061	0.09212061	1.078849127	-0.07884913	0.07884913
1	1.01841835	-0.01841835	0.01841835	0.994138846	0.00586115	0.00586115
1	0.9558495	0.0441505	0.0441505	1.033401121	-0.03340112	0.03340112
0	-0.08859339	0.08859339	0.08859339	-0.02248793	0.02248793	0.02248793
0	0.56413871	-0.56413871	0.56413871	0.381700721	-0.38170072	0.38170072
1	0.91678821	0.08321179	0.08321179	0.927762357	0.07223764	0.07223764
0	0.00993801	-0.00993801	0.00993801	-0.00285702	0.00285702	0.00285702
1	1.12580902	-0.12580902	0.12580902	0.996243992	0.00375601	0.00375601
0	0.24730972	-0.24730972	0.24730972	-0.02258458	0.02258458	0.02258458
1	0.76636003	0.23363997	0.23363997	0.682215131	0.31778487	0.31778487
0	-0.10369261	0.10369261	0.10369261	0.019215047	-0.01921505	0.01921505
1	0.89038737	0.10961263	0.10961263	0.87725751	0.12274249	0.12274249
0	0.09659665	-0.09659665	0.09659665	-0.006446	0.006446	0.006446
0	-0.03005373	0.03005373	0.03005373	0.014764046	-0.01476405	0.01476405
0	-0.0096976	0.0096976	0.0096976	-0.00917168	0.00917168	0.00917168

diagnosis	entrenamiento	error	MAD	entrenamiento	error	MAD
0	0.35701143	-0.35701143	0.35701143	0.251968745	-0.25196874	0.25196874
1	1.06131665	-0.06131665	0.06131665	0.992898656	0.00710134	0.00710134
0	0.67337124	-0.67337124	0.67337124	0.58474842	-0.58474842	0.58474842
0	-0.00276508	0.00276508	0.00276508	-0.05751451	0.05751451	0.05751451
0	0.44462039	-0.44462039	0.44462039	0.225854234	-0.22585423	0.22585423
0	-0.05017964	0.05017964	0.05017964	-0.0529075	0.0529075	0.0529075
0	-0.00892803	0.00892803	0.00892803	-0.05091427	0.05091427	0.05091427
1	1.09786378	-0.09786378	0.09786378	1.08692787	-0.08692787	0.08692787
1	0.83500471	0.16499529	0.16499529	0.870654293	0.12934571	0.12934571
1	1.04038409	-0.04038409	0.04038409	1.010450479	-0.01045048	0.01045048
0	0.43022983	-0.43022983	0.43022983	0.336582154	-0.33658215	0.33658215
1	1.10248953	-0.10248953	0.10248953	1.002084885	-0.00208489	0.00208489
0	-0.16589768	0.16589768	0.16589768	0.037870735	-0.03787073	0.03787073
1	0.74278018	0.25721982	0.25721982	0.756830129	0.24316987	0.24316987
1	1.06348642	-0.06348642	0.06348642	1.100518643	-0.10051864	0.10051864
0	0.04624259	-0.04624259	0.04624259	0.09598724	-0.09598724	0.09598724
0	-0.13160283	0.13160283	0.13160283	-0.00725327	0.00725327	0.00725327
1	1.1704132	-0.1704132	0.1704132	1.063833252	-0.06383325	0.06383325
0	0.0028821	-0.0028821	0.0028821	-0.02120975	0.02120975	0.02120975
1	1.13947494	-0.13947494	0.13947494	1.011437596	-0.0114376	0.0114376
1	1.00402491	-0.00402491	0.00402491	1.027867571	-0.02786757	0.02786757
1	0.80464947	0.19535053	0.19535053	0.767732838	0.23226716	0.23226716
0	0.02434666	-0.02434666	0.02434666	-0.09290584	0.09290584	0.09290584
0	-0.0409556	0.0409556	0.0409556	0.035882234	-0.03588223	0.03588223
0	0.30585623	-0.30585623	0.30585623	0.004183139	-0.00418314	0.00418314
1	1.07060958	-0.07060958	0.07060958	1.001457771	-0.00145777	0.00145777
0	-0.06623707	0.06623707	0.06623707	-0.02184932	0.02184932	0.02184932
1	0.98624775	0.01375225	0.01375225	1.00637108	-0.00637108	0.00637108
1	0.43044524	0.56955476	0.56955476	0.311102314	0.68889769	0.68889769
1	0.84057682	0.15942318	0.15942318	0.870301248	0.12969875	0.12969875
0	0.08814023	-0.08814023	0.08814023	-0.01138006	0.01138006	0.01138006
1	1.0375766	-0.0375766	0.0375766	1.018912631	-0.01891263	0.01891263
1	0.50070494	0.49929506	0.49929506	0.476971557	0.52302844	0.52302844
0	0.08050065	-0.08050065	0.08050065	0.020895195	-0.0208952	0.0208952
1	0.96092466	0.03907534	0.03907534	1.006988275	-0.00698827	0.00698827
0	0.04555762	-0.04555762	0.04555762	-0.04308605	0.04308605	0.04308605
1	1.18627973	-0.18627973	0.18627973	1.051251508	-0.05125151	0.05125151
0	-0.22927669	0.22927669	0.22927669	0.023319158	-0.02331916	0.02331916
0	0.44465974	-0.44465974	0.44465974	0.340559016	-0.34055902	0.34055902
0	-0.05844459	0.05844459	0.05844459	-0.02480653	0.02480653	0.02480653
1	0.81681774	0.18318226	0.18318226	0.771733542	0.22826646	0.22826646
0	0.04607257	-0.04607257	0.04607257	0.152307891	-0.15230789	0.15230789
0	0.06441552	-0.06441552	0.06441552	0.072823903	-0.0728239	0.0728239
1	1.16979825	-0.16979825	0.16979825	1.002122002	-0.002122	0.002122
1	0.91218073	0.08781927	0.08781927	0.978766304	0.0212337	0.0212337
0	0.15451593	-0.15451593	0.15451593	-0.02252764	0.02252764	0.02252764
1	0.64716153	0.35283847	0.35283847	0.636985297	0.3630147	0.3630147
1	1.09570484	-0.09570484	0.09570484	1.107465576	-0.10746558	0.10746558
1	0.95366802	0.04633198	0.04633198	0.929224322	0.07077568	0.07077568
0	-0.05443245	0.05443245	0.05443245	-0.03353541	0.03353541	0.03353541
1	0.90250859	0.09749141	0.09749141	0.933797715	0.06620228	0.06620228

diagnosis	entrenamiento	error	MAD	entrenamiento	error	MAD
1	1.0490159	-0.0490159	0.0490159	1.086376648	-0.08637665	0.08637665
1	0.84234595	0.15765405	0.15765405	0.846253662	0.15374634	0.15374634
1	1.0929184	-0.0929184	0.0929184	0.982697273	0.01730273	0.01730273
0	0.04352993	-0.04352993	0.04352993	-0.02240108	0.02240108	0.02240108
0	0.55081722	-0.55081722	0.55081722	0.509180953	-0.50918095	0.50918095
1	0.97269401	0.02730599	0.02730599	0.969797988	0.03020201	0.03020201
0	0.00832517	-0.00832517	0.00832517	-0.0708774	0.0708774	0.0708774
1	0.97997888	0.02002112	0.02002112	0.953545752	0.04645425	0.04645425
1	1.02214842	-0.02214842	0.02214842	1.072567094	-0.07256709	0.07256709
0	-0.07334391	0.07334391	0.07334391	-0.01563459	0.01563459	0.01563459
1	1.17594399	-0.17594399	0.17594399	0.996288038	0.00371196	0.00371196
0	0.35641443	-0.35641443	0.35641443	0.325396426	-0.32539643	0.32539643
	PROMEDIO	**-0.02488806**	**0.14892524**		**0.00146681**	**0.11180413**