
# Técnicas de Inteligencia artificial explicable basadas en una integración de lógica simbólica y no-simbólica


| | |
|---|---|
| Tesista: | Mg. Ing. Pablo Ariel Negro |
| Director: | Dra. Claudia Pons |
| Codirector: | Dr. Carlos Neil |


Trabajo presentado para obtener el título de
Doctor en Informática

## Doctorado en Tecnología Informática

### Facultad de Tecnología Informática

Julio de 2024

# Resumen

El Aprendizaje Profundo (*Deep Learning* - *DL*) aprovecha el potencial de las redes neuronales profundas para influir en el proceso genérico del aprendizaje mediante la prueba y el error. Su efectividad se ha demostrado de manera sólida en diversas áreas. No obstante, los sistemas de DL heredan deficiencias de las técnicas actuales de aprendizaje profundo, como la necesidad de conjuntos de datos extensos para su óptimo funcionamiento. Carecen de la capacidad de razonamiento abstracto, lo que dificulta la ejecución de funciones cognitivas avanzadas como el razonamiento analógico o basado en hipótesis. Además, su funcionamiento es mayormente opaco para los humanos, lo que los hace inadecuados en ámbitos donde la verificabilidad es crucial. La explicabilidad se refiere a la capacidad de explicar las decisiones tomadas por un modelo en términos comprensibles para las personas. Su propósito es comprender por qué el modelo adoptó una decisión particular y reconocer las condiciones en las que tiene éxito o fracasa. En este sentido, la necesidad de integración entre lo neuronal y lo simbólico se vuelve evidente al enfrentar problemas de mayor complejidad. Los métodos de búsqueda para extraer reglas en redes neuronales profundas entrenadas realizan un análisis de los pesos y sesgos generados por la red. Al calcular la correlación entre los vectores de pesos con las salidas en cada capa, es factible reducir el espacio de búsqueda e identificar el camino crítico que conecta las entradas con los resultados obtenidos por la red neuronal. Basándonos en esta observación, este trabajo tiene como objetivo presentar un método para extraer el patrón de reglas aprendido por una red neuronal entrenada de tipo *feedforward*, analizar sus propiedades y explicar estos patrones mediante el uso de lógica de primer orden (FOL).

**Palabras claves:** Aprendizaje Profundo, Inteligencia Artificial Explicable, Lógica, Redes Neuronales Artificiales, Similitud Coseno.

# Dedicatoria

Este trabajo está dedicado a mi familia, y en especial a mi esposa Romina y a mi amada hija Agustina, quienes me acompañaron a lo largo del camino.

Agradecimientos a Dios, a la familia y a las buenas intenciones.

# Reconocimientos

A todos aquellos que hicieron posible cumplir este sueño y que pueda haber alcanzado la estrella que más brilla a lo largo de mi carrera académica.

A la UAI por haber creado el espacio y la carrera.
Dr. Carlos Neil
Dra. Claudia Pons
Dr. Gustavo Rossi
Dra. Maria Isabel Iñigo Petralanda
Lic. Gonzalo Zabala
Ing. Nestor Ballich

# Índice General

*Tema* *Página*

# Índice de Tablas

# Índice de Figuras

# CAPITULO 1 Análisis de Contexto

## 1.1. Introducción

En la actualidad, la inteligencia artificial (IA) es un campo próspero con numerosas aplicaciones prácticas y áreas de investigación activas. En sus inicios, la IA se centró en abordar y resolver rápidamente problemas que resultaban intelectualmente complicados para los seres humanos, pero relativamente simples para las computadoras. Estos problemas podían describirse mediante una lista de reglas matemáticas formales. Sin embargo, el verdadero desafío para la inteligencia artificial surgió al intentar resolver tareas que son fáciles de llevar a cabo para las personas, pero difíciles de describir formalmente: problemas que resolvemos de manera intuitiva y que parecen automáticos, como el reconocimiento del habla o de rostros en imágenes. (Goodfellow et al., 2016). Esta dualidad genera dos enfoques predominantes, pero sumamente divergentes: la Inteligencia Artificial simbólica, que se inspira en la lógica matemática y se fundamenta en la manipulación de representaciones lingüísticas abstractas y la Inteligencia Artificial no simbólica, que se enfoca en la creación de modelos matemáticos predictivos a partir de extensos conjuntos de datos de muestra

Por otro lado, el aprendizaje profundo o *Deep Learning* (DL) es una forma específica de Inteligencia Artificial no simbólica. Este enfoque plantea una inspiración biológica al emplear las capacidades de las redes neuronales profundas para influir en el proceso general del aprendizaje mediante el método de prueba y error. Su eficacia ha sido convincentemente demostrada en una amplia gama de campos (Garnelo et al., 2016). A pesar de su éxito innegable, varias investigaciones han dado evidencia de importantes deficiencias en el aprendizaje profundo contemporáneo (Garnelo et al., 2016; Lake et al., 2017; Marcus, 2018; Battaglia et al., 2018; Yann LeCun, Yoshua Bengio, 2015). En particular, podemos destacar :

*a)* Ineficiencia en los datos (alta complejidad de la muestra), ya que necesitan conjuntos de datos muy extensos para operar de manera eficiente, lo que también implica que su proceso de aprendizaje es lento. (Schmidhuber, 2015; Tsividis et al., 2017).

*b)* Limitada capacidad de generalización, ya que las actuales redes neuronales tienden a fallar de manera significativa al enfrentarse a datos que difieren de la distribución en la que fueron entrenadas. Por ejemplo, la simple modificación del color o tamaño de un objeto en un videojuego podría forzar a un agente entrenado mediante aprendizaje por refuerzo a tener que reaprender el juego desde sus fundamentos. (Tsividis et al., 2017).

*c)* Carecen de la capacidad de razonar a un nivel abstracto, lo cual supone un obstáculo para la implementación de funciones cognitivas avanzadas como el

aprendizaje de transferencia, el razonamiento analógico y la inferencia basada en hipótesis. Se han llevado a cabo diversos intentos recientes para emular procesos lógicos en redes neuronales. (Rocktäschel & Riedel, 2017; Evans et al., 2018).

*d)* Por último, debido a su falta de interpretabilidad, el funcionamiento de las redes neuronales es en gran medida opaco para los seres humanos, lo que las vuelve inadecuadas para dominios en los que la verificabilidad es crucial. (Santoro et al., 2017; Garnelo et al., 2016; Lake et al., 2017; Marcus, 2018; Battaglia et al., 2018).

Además, las soluciones basadas en la lógica podrían abrir nuevas oportunidades para desarrollar sistemas inteligentes que sean explicativos y posean habilidades de razonamiento lógico. La creación de un modelo que integre ambos enfoques podría proporcionarnos lo mejor de ambos mundos. Sus representaciones se fundamentarían en lógica y se aprenderían a partir de datos con pocos precedentes. Una arquitectura que combine todas estas características podría respaldar las propiedades deseadas de eficiencia en los datos, capacidad de generalización e interpretación comprensible para los seres humanos (Domingos, 2018).

En este sentido, dada la situación actual de las tecnologías de inteligencia artificial, las cuales han tenido principalmente éxito en contextos de aplicaciones específicas, un tema crucial para la ingeniería de sistemas inteligentes es la integración de diversas técnicas de IA. Esto implica, desde una perspectiva de ingeniería de software, no solo la combinación de diferentes tecnologías, sino también la preservación de la integridad conceptual al fusionar enfoques altamente heterogéneos que generan múltiples abstracciones de distintas naturalezas (Battaglia et al., 2016).

Por otro lado, en la actualidad, se está evidenciando una clara división en el campo de la Inteligencia Artificial. A pesar del éxito abrumador de técnicas no simbólicas como el Aprendizaje Profundo (DL), que ha disipado en gran medida las objeciones contra la IA, la preocupación pública sobre el papel que desempeñarán los sistemas inteligentes en la sociedad ha generado nuevas interrogantes. Es imprescindible explicar el comportamiento de estos sistemas y lograr que sean comprensibles para los seres humanos, especialmente cuando la IA desempeña un papel crucial dentro de las organizaciones (Calegari et al., 2020).

En tal sentido, el enfoque general sobre la capacidad de la IA de explicarse requerirá la adopción extensiva de técnicas simbólicas (posiblemente integradas con no-simbólicas) para alcanzar sus objetivos principales, como la observabilidad, interpretabilidad, explicabilidad y responsabilidad (Barredo Arrieta et al., 2020). Como resultado, en un momento en el que el enfoque predominante en la IA se centra en las técnicas no simbólicas, las cuales han posibilitado el funcionamiento de la IA en aplicaciones del mundo real, los enfoques simbólicos están siendo objeto de un examen cuidadoso. Esto se debe a que podría permitir que la IA se ajuste también al entendimiento (Domingos, 2018).

Por otra parte, las técnicas fundamentadas en la lógica no solo son las más antiguas, sino que posiblemente representan el camino más directo hacia la comprensión humana. Si

no es por otra razón, el enfoque de la lógica como el estudio del razonamiento apropiado se asemeja a los procesos cognitivos humanos (McCarthy, 1987). Además, los enfoques basados en lógica son desde hace mucho tiempo el núcleo de muchos modelos y tecnologías exitosas basadas en agentes. En términos generales, los agentes cognitivos inteligente explotan directamente modelos y tecnologías basados en lógica para el proceso racional, la representación del conocimiento, la comunicación expresiva y la coordinación efectiva (Russell & Norvig, 2010).

En el pasado, otros paradigmas han competido con las redes neuronales, incluyendo la inteligencia artificial simbólica (o clásica), que posiblemente fue el enfoque predominante hasta finales de la década de 1980. Un sistema de IA simbólica opera realizando una secuencia de pasos de razonamiento lógico sobre representaciones lingüísticas. Estas representaciones suelen ser de naturaleza proposicional y afirman la existencia de ciertas relaciones entre objetos específicos, mientras que cada paso de razonamiento calcula un conjunto adicional de relaciones que se derivan de las ya establecidas, siguiendo un conjunto formalmente especificado de reglas de inferencia.

Significativamente, las deficiencias del aprendizaje profundo se alinean con las fortalezas de la IA simbólica, lo que sugiere que una integración entre ellos sería beneficiosa (Marcus, 2001; Garnelo et al., 2016; Garnelo y Shanahan, 2019; Doumas, Puebla, y Martin, 2019). Lo cual implica que:

- Por su naturaleza declarativa, las representaciones simbólicas son susceptibles de ser reutilizadas en múltiples dominios, lo que fomenta la eficiencia en el manejo de los datos.

- Debido a que estas representaciones suelen ser de alto nivel y abstractas, facilita la capacidad de generalización.

- Por su carácter proposicional, similar al lenguaje, las representaciones simbólicas son comprensibles para los seres humanos. Como resultado, los investigadores han comenzado a explorar la manera de integrar ideas relevantes de la IA simbólica en un contexto de aprendizaje profundo.

En este sentido, el objetivo de este trabajo es desarrollar un algoritmo que permita extraer reglas comprensibles de redes neuronales *feedforward* entrenadas mediante el análisis de las matrices de pesos. Estas reglas podrán validarse y explicarse mediante la aplicación de técnicas de lógica proposicional y de primer orden. Además, se validarán las reglas extraídas utilizando estadísticas obtenidas de los atributos de los conjuntos de datos, ya que el conocimiento está distribuido en toda la red y no está claro qué información de los datos de entrada influye realmente en sus decisiones.

El método presentado en este trabajo se encuentra en el contexto de la IA neuro simbólica, que es un tipo de inteligencia artificial que integra arquitecturas de IA neuronal y simbólica para abordar las debilidades de cada una, proporcionando una IA robusta capaz de razonar, aprender y modelar cognitivamente.

Es importante destacar que este trabajo solo considera redes neuronales entrenadas con datos tabulares (tablas que contienen datos numéricos y texto) y no incluye aquellas entrenadas para el análisis de imágenes.

## 1.2. Explicabilidad: Legislación y Aplicación

Con la irrupción de las redes neuronales profundas, el desafío de explicar los resultados de estas redes está siendo cada vez más reconocido. Aunque existen numerosos métodos para explicar las decisiones de las redes neuronales profundas, actualmente no hay consenso sobre cómo evaluarlas.
A pesar del enorme progreso, su aceptación en áreas de aplicación críticas se ve obstaculizada por dos limitaciones significativas:

- Existe una inherente incapacidad para explicar las decisiones de manera comprensible para los seres humanos.

- Las DNN (redes neuronales profundas) son vulnerables a ataques adversarios, es decir, alteraciones maliciosas e imperceptibles en la entrada, que pueden engañar a las redes entrenadas y alterar drásticamente sus decisiones.

Se ha planteado que las limitaciones en términos de explicabilidad en DNN son complejas debido a que el proceso de toma de decisiones de estas es difícil de entender, el conocimiento en la red neuronal se almacena como parámetros de valor real (pesos y sesgos), el conocimiento se codifica de manera distribuida y el mapeo aprendido por la red podría ser no lineal y no monótono. Entonces, ¿Por qué se deben usar DNN cuando la comprensibilidad es un tema importante?
La razón es que la precisión predictiva también es muy importante y las redes neuronales tienen un sesgo inductivo apropiado para muchos dominios de *Machine Learning*.

En este sentido, la explicabilidad aborda el problema crítico de que los humanos no pueden comprender directamente el comportamiento complejo de las DNN ni explicar su proceso subyacente de toma de decisiones. La explicabilidad de los modelos de *Machine Learning* es un requisito fundamental para generar confianza con los usuarios y es clave para su implementación segura, justa y exitosa en aplicaciones del mundo real.

Desde el punto de vista legal, la cuestión de la explicabilidad va más allá del ámbito del interés científico. La adopción del Reglamento General de Protección de Datos (GDPR) por parte de la Unión Europea en mayo de 2018 (https://gdpr-info.eu/), concede a cualquier ciudadano el derecho a recibir una explicación sobre una decisión algorítmica tomada en relación con él. El GDPR establece que las personas tienen derecho a no ser objeto de una decisión basada únicamente en el procesamiento automatizado. Por lo tanto, en este contexto, la explicabilidad es tanto un derecho legal como una

responsabilidad que tiene amplias implicaciones sociales (Samek, Wiegand, y Müller 2017).

Desde la perspectiva empresarial, y tomando como ejemplo el ámbito del mercado de capitales, siempre ha sido crucial predecir el futuro en términos de precios, ejecuciones, estrategias óptimas y algoritmos de trading que mejoren el desempeño en la implementación de dichas estrategias. Las carencias en la industria debido a la falta de herramientas predictivas se traducen en pérdidas de oportunidades y en una gestión limitada del riesgo. En el contexto del panorama actual conocido como *Big Data*, y considerando que las empresas disponen de grandes volúmenes de datos, es evidente que frecuentemente no saben cómo aprovecharlos o identificar nuevas oportunidades de mejora o crecimiento. A pesar de que las técnicas de *Business Intelligence* han sido fundamentales en el desarrollo de sistemas de apoyo a la toma de decisiones, el análisis de los resultados debe ser llevado a cabo por un experto en el sector específico.

Por otro lado, en Argentina, contamos con un mercado de capitales de tamaño reducido y el crecimiento y la incorporación de nuevos participantes no ha sido tan significativo como en otros países de la región. Aunque esto se debe a diversos factores coyunturales, la falta de herramientas que respalden la toma de decisiones e incluso que sean capaces de prever posibles movimientos futuros del mercado podría contribuir a democratizar y acercar el mercado al público en general.

En este sentido, el mercado de capitales demanda soluciones que operen a alta velocidad y con una disponibilidad elevada. Desde la perspectiva del mercado, como centro de transacciones, la latencia ha sido una preocupación constante. Cuando un sistema no puede optimizarse más desde el punto de vista del software, se busca que el mismo se expanda para funcionar en una mayor cantidad de servidores, incrementando así la potencia desde el hardware. Si pudiéramos implementar un modelo híbrido como el propuesto en este trabajo, deberíamos ser capaces, basados en la experiencia, de predecir el próximo movimiento y la dirección de la siguiente ejecución. Esta predicción podría ser explicada, permitiéndonos anticiparnos y ajustar pasos de validación y secuencias de emparejamiento, por ejemplo, en un sistema de trading electrónico, o calcular riesgos asociados a una cartera específica, minimizando así la exposición al riesgo, entre otros beneficios. Este sería un paso más hacia una optimización inteligente desde el punto de vista del software.

## 1.3. Problemas y soluciones propuestas

En esta sección se plantean los problemas principales que se atacan a través del presente trabajo. Entre otros, las redes neuronales profundas presentan los siguientes inconvenientes:

Problema:

- **Carecen de capacidad de razonar en un nivel abstracto**, lo que dificulta la implementación de funciones cognitivas de alto nivel, como el aprendizaje por

transferencia, el razonamiento analógico y el razonamiento basado en hipótesis. Se han realizado varios intentos recientes para imitar procesos lógicos en redes neuronales (Evans et al., 2018). Este problema también es tratado en (Manhaeve et al., 2018; Cai et al., 2017; Han et al., 2021)

Solución:

Se propone construir una solución que permita:

- Incorporar lógica en la arquitectura del método de extracción de reglas, que permita realizar operaciones lógicas básicas y razonar en un nivel más abstracto.
- Desarrollar técnicas que permitan el razonamiento analógico y basado en hipótesis mediante la manipulación de la representación del conocimiento aprendido por la red neuronal.
- Confiar en las decisiones del sistema a partir de poder verificación su corrección.
- Habilitar la mejora de los sistemas al contar con una mejor comprensión de su funcionamiento y sus posibles puntos débiles.

Problema:

- **Falta de interpretabilidad**. Las redes neuronales son típicamente *cajas negras*. Los cálculos realizados por capas sucesivas rara vez corresponden a pasos de razonamiento humanamente comprensibles, y los vectores intermedios de activaciones que se generan carecen de una semántica humanamente comprensible (Garnelo et al., 2016). Este problema también es tratado en (AmirHosseini y Hosseini, 2019); Mahdavifar y Ghorbani, 2020; Cocarascu, Cyras, y Toni, 2018; Csiszár, Csiszár, y Dombi, 2020; Dombi & Csiszár, 2021; Schmid y Finzel, 2020).

Solución:

Se propone desarrollar un método que permita:

- Extraer las reglas aprendidas por la red neuronal a partir de su matriz de pesos.
- Aprender del sistema mediante la extracción del conocimiento aprendido y codificado.
- Mapear las reglas extraídas con los datos de entrada.

Dado que no existe una solución satisfactoria al problema de la explicabilidad, de las redes neuronales artificiales, este trabajo se propone refinar formalismos y métodos que aporten a los criterios de calidad de la explicación provista.

A su vez, los métodos simbólicos presentan la siguiente deficiencia:

**Problema:**

- **Symbol Grounding.** Una limitación importante de la IA simbólica se relaciona con el llamado problema de *Symbol Grounding* (Harnad, 1990), y se refiere a la medida en que sus elementos simbólicos están hechos a mano en lugar de aprender de los datos. Por el contrario, uno de los puntos fuertes del aprendizaje profundo es su capacidad para descubrir características en datos de gran dimensión con poca o ninguna intervención humana.

Solución:

Se propone desarrollar un método que permita:

- Utilizar el conocimiento codificado en la red neuronal, como entrada al motor de inferencia.
- Validar las reglas extraídas a través de FOL.
- Proveer de un conjunto de reglas lógicas que faciliten el entendimiento de los usuarios, respecto de las decisiones que toma la DNN.

A continuación, se provee una descripción de alto nivel de la solución propuesta. Para el dominio de la solución, consideraremos la siguiente notación:

$$(x, y) = x \in \mathbb{R}^{n_x},\ y \in (Class0, Class1, ClassN) \qquad \text{eq 1.1}$$

Donde =

- $x \in \mathbb{R}^{m \cdot n_x}$ Siendo x la matriz con valores de entrada.
- $y \in \mathbb{R}^{1 \cdot m}$ Siendo y el vector de salida.

*m* representa los ejemplos de entrenamiento, y la similitud coseno viene dada por la siguiente ecuación.

$$similarity = \cos(\emptyset) = \frac{A.B}{||A||.||B||} = \frac{\sqrt{\sum_{i-1}^{n} A_i B_i}}{\sqrt{\sum_{i-1}^{n} A_i^2}\sqrt{\sum_{i-1}^{n} B_i^2}} \qquad \text{eq 1.2}$$

En este contexto, el algoritmo comenzara con un proceso de extracción de pesos de la red neuronal entrenada, y se calculara la similitud coseno entre los vectores que forma cada variable de entrada para cada neurona entre sí, y a también en relación con el vector de *bias*, y el vector formado por los valores de activación en cada capa. Además, se realiza el mismo procedimiento, para los vectores de pesos de cada neurona, lo cual resulta en una comparación interneurona de vectores de pesos.

El objetivo es analizar cómo se correlación las variables de entrada, y como estas explican los resultados en la salida. De este modo podremos identificar caminos críticos neuronales que van desde las variables de entrada hasta la salida de la red. Adicionalmente, se calculan estadísticas sobre los datos de entrada, y se utilizan para el

método de extracción de reglas, aquellas que mejor explican los resultados arrojados por las redes (máxima precisión). En tal sentido, los rangos de valores en entre ambas métricas serán nuestro *baseline* que nos ayudarán a validar y verificar las reglas extraídas.

## 1.4. Arquitectura de la solución propuesta

*Todo el conocimiento, pasado, presente y futuro, debería poder ser derivado desde los datos por un modelo universal simple* (Domingos, 2018). Desde esta perspectiva, la solución propuesta para abordar el problema de la explicabilidad en las DNN es diseñar un modelo combinado entre la inteligencia artificial simbólica y no simbólica. Este modelo busca maximizar las fortalezas de cada enfoque particular mientras complementa las debilidades de ambos de manera integrada. La creación, incluso a nivel conceptual, de un cerebro universal construido a partir de la combinación de métodos de aprendizaje, proporcionaría un mecanismo de aprendizaje general con capacidad de razonamiento multipropósito.

En este sentido, el objetivo principal de este trabajo es desarrollar un algoritmo que permita extraer reglas comprensibles de redes neuronales entrenadas mediante el análisis de las matrices de pesos, y que estas reglas puedan validarse y explicarse utilizando técnicas de lógica de primer orden. En esta línea, las tareas de extracción de reglas pueden ser consideradas como una tarea de búsqueda o como una tarea de aprendizaje. Como se mencionó anteriormente, en una red neuronal entrenada, el conocimiento adquirido durante la fase de entrenamiento está codificado en la arquitectura de la red, la función de activación utilizada y los pesos y sesgos de las neuronas (Goodfellow et al., 2016). Por lo tanto, la tarea de la extracción de reglas es usar una o más de estas piezas de información y extraer un conjunto de reglas de las neuronas. En la Fig. 1 se muestra la arquitectura del modelo híbrido ampliado propuesto en esta tesis.

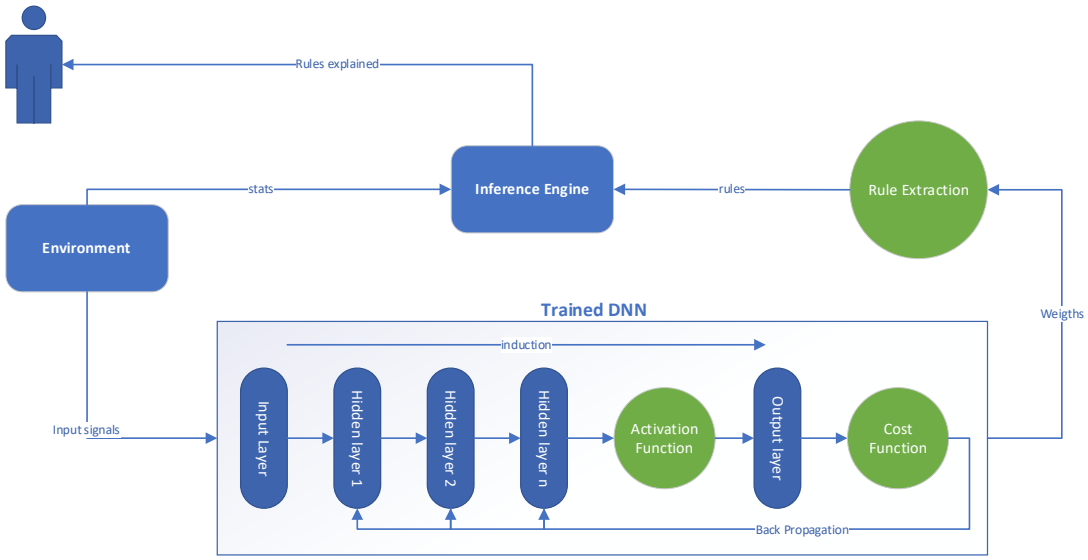


*Figura 1. Esquema arquitectónico del modelo hibrido.*

Si bien el modelo propuesto ofrece la capacidad de explicabilidad a una red neuronal, este solo integra dos modelos de IA, la IA simbólica y la no simbólica, lo que plantea

ciertas limitaciones. Si el modelo se prueba e integra con éxito, deberíamos avanzar en la incorporación de otros modelos provenientes de las diversas corrientes dentro de la IA, que han demostrado ser efectivos en la resolución de problemas específicos. Cada uno de estos enfoques podría aportar capacidades que los modelos aquí propuestos carecen. La investigación propuesta en este trabajo es compleja, pero se enriquece al considerar las distintas corrientes de pensamiento en competencia dentro del Aprendizaje Automático.

## 1.5. Objetivo del trabajo

**Objetivo general:**

- Desarrollar un algoritmo simbólico-neuronal híbrido que permita extraer reglas comprensibles en redes neuronales *feedforward* profundas entrenadas, a través del análisis de las matrices de pesos, y la validación y explicación de reglas por medio de la aplicación de técnicas de lógica proposicional y de primer orden.

**Objetivos específicos:**

- Diseñar un modelo formal capaz de incorporar ideas relevantes de la lógica simbólica en un marco de aprendizaje profundo, con el objetivo de extraer reglas y proporcionar una explicación para modelos de aprendizaje profundo.

- Diseñar un mecanismo de aprendizaje relacional y racional, con el objetivo de entender las decisiones que tomo una red neuronal profunda, y poder explicarlos con lógica simbólica.

- Desarrollar un algoritmo de extracción de reglas unificando técnicas de inducción y deducción, con la finalidad de mapear las reglas extraídas con los datos y poder aportar una explicación completa.

- Construir un prototipo funcional que implemente las funciones definidas en el modelo formal, que automatice el proceso de extracción y validación de las reglas y, de esto modo, hacer el método susceptible de ser aplicable a escala.

## 1.6. Transferencia de los resultados obtenidos

Para el presente trabajo, la solución esta propuesta con el objetivo de entregar al usuario de la solución un conjunto de reglas comprensibles, a través del uso de lógica de primer orden, que faciliten la tarea de comprender que es lo que aprendió la DNN, y de que variables de entrada se valió para tal caso. Además, estas reglas se puedan mapean con los datos de entrada y de este modo conformar las explicaciones basadas en los datos.
En el presente trabajo se pone en el centro de la discusión al individuo humano, como destinatario de las decisiones algorítmicas entregadas por una DNN. Hecha esta observación, cualquier industria que utilice soluciones de IA y donde los individuos se

vean afectados por tales modelos, pueden hacer uso de estas técnicas de explicabilidad en general, y del método propuesto en esta tesis en particular. Como desarrolladores de IA, la adopción de prácticas sobre IA responsable (RAI, por sus siglas en inglés) nos interpela a todos, donde la explicabilidad es un habilitador tecnológico fundamental para generar confianza en los usuarios.
Si bien inicialmente el método está pensado para dar soluciones al mercado de capitales, idealmente, el método explicable, también asistirá equipos de desarrollo de modelos de *Deep Learning* a través de la comprensión de los datos, en la mejora del proceso de *tuning* de hiperparámetros, como así también en el diseño de la topología de la DNN optima. En tal sentido el entrenamiento de la DNN sería más asertivo, y se debería poder llevar delante con más criterio.
Otro ámbito de gran relevancia donde el método de extracción de reglas puede aportar, son los sistemas de salud, donde los profesionales médicos podrían valerse de la solución al momento de tomar alguna decisión. Los diagnósticos por imágenes tienen asociados a cada imagen (radiografía) una gran cantidad de información provista por los profesionales. Esta información tabular asociada a la imagen puede ser *input* de entrenamiento de una DNN, donde se aplique a posteriori el método de extracción presentado en este trabajo para comprender las decisiones que tomo la DNN, y en que *features* de entrada se basó.
Sin perjuicio del dominio inicial planteado, la solución propuesta puede ser utilizada en cualquier solución que requiera un método de explicabilidad sobre una DNN, entrenada a partir de datos tabulares.

## 1.7. Contribuciones principales

El objetivo principal de este trabajo es desarrollar un algoritmo simbólico-neuronal híbrido que permita extraer reglas comprensibles en redes neuronales *feedforward* profundas entrenadas, a través del análisis de las matrices de pesos, y la validación y explicación de reglas por medio de la aplicación de técnicas de lógica de primer orden. Por último, mejorar la performance del método y proceso de aprendizaje de una red neuronal, al hacerlo más eficiente.

Como objetivo a colación y de más largo plazo, continuaría en línea con la incorporación de otros modelos proporcionados por las diferentes escuelas de pensamiento dentro de *Machine Learning*.

*A través de la combinación de ambos enfoques,* una síntesis que satisfaga y verifique la integración de lógica simbólica con aprendizaje profundo nos daría lo mejor de ambas escuelas de pensamiento. Sus representaciones estarían fundamentadas, y derivadas del conocimiento propio generado por la DNN. Soportaría secuencias arbitrariamente largas de pasos de inferencia utilizando todos esos elementos, como la lógica formal. Pero no estaría limitado por las reglas de la lógica formal, y sería capaz de aprender formas de inferencia que trasciendan las restricciones que la lógica formal implica. *Una arquitectura que combina todas estas características podrá dar soporte a las propiedades deseadas de eficiencia de datos, poder de generalización e interpretabilidad humana.*

La integración de estos enfoques propende a exacerbar las bondades de cada modelo en particular, mientras que se complementan las falencias de cada uno de manera acoplada. En consecuencia, el conocimiento podría derivarse y validarse desde los datos, y explicarse a través de un modelo hibrido universal simple.

En el presente trabajo se presenta un método para extraer el patrón de reglas aprendido de una red neuronal de *feedforward* entrenada y analizar sus propiedades. El método integra lógica simbólica y no simbólica para explicar el proceso de toma de decisiones del modelo.

En este trabajo se propone utilizar el análisis de correlación entre los vectores de pesos y las salidas de cada capa para identificar los caminos críticos que conectan las entradas y los resultados en la red neuronal.

Las reglas extraídas se escriben en términos de lógica proposicional y lógica de primer orden (FOL) para explicar la relación entrada-salida.

El enfoque tiene como objetivo mejorar la explicabilidad de las redes neuronales profundas y comprender las condiciones en las que el modelo tiene éxito o fracasa.
En general, el enfoque de este trabajo sobre explicabilidad en redes neuronales *feedforward* profundas tiene implicaciones prácticas para mejorar la transparencia, la interpretabilidad y la confianza en el proceso de toma de decisiones de estos modelos.

## 1.8. Trabajos presentados vinculados con la tesis

En esta sección se presentan y se hace una breve descripción de los trabajos relacionados motivados por el presente trabajo de tesis. Los principales resultados de esta tesis están respaldados en las siguientes publicaciones científicas con referato.

En (Negro y Pons, 2022) se proporciona una visión integral de las soluciones disponibles en la literatura dentro del campo de la Inteligencia Artificial (IA) aplicada, utilizando tecnologías basadas en técnicas de IA que Integran la Inteligencia Artificial Simbólica y No-Simbólica (en particular redes neuronales artificiales), haciéndolas objeto de una revisión sistemática de la literatura (SLR). Las tecnologías resultantes se discuten y evalúan desde ambas perspectivas: la IA simbólica y la no simbólica.
En este trabajo, utilizamos el método PICOC más Límites para definir las preguntas de investigación y analizar los resultados.

El desarrollo de métodos de búsqueda para la extracción de reglas de redes neuronales ha permitido abordar problemas complejos que van más allá de la clasificación. Estos métodos funcionan al enviar combinaciones de datos de entrada que activan un conjunto de neuronas. La adecuada modificación de los pesos de entrada de una neurona permite acotar el espacio de búsqueda. Por lo tanto, el presente trabajo tiene como objetivo proponer un método para extraer el patrón de reglas aprendido por una red neuronal *feedforward* entrenada, analizar sus propiedades y explicar estos patrones mediante el uso de lógica de primer orden (LPO) (Negro y Pons, 2023).

En el contexto de explicar las decisiones tomadas por las DNN, los vectores de peso de la red neuronal se pueden comparar usando similitud de coseno, para identificar las entradas que mejor explican las salidas en cada capa neuronal y, de esta manera, trazar el camino crítico. que sigue a la red para tomar decisiones. Entonces, si podemos identificar dicha ruta neuronal crítica para cada capa de la red, deberíamos poder explicar esta función usando lógica de primer orden y extraer reglas que expliquen y mapeen los datos en las entradas. Al definir un conjunto de reglas lógicas que capturan el comportamiento de la red, se podría razonar sobre el comportamiento de la red mediante inferencia lógica. Por ejemplo, dado un conjunto de valores de entrada, es posible utilizar la inferencia lógica para determinar la salida de la red (Negro y Pons 2024).

## 1.9. Estructura General de la tesis

El documento está organizado de la siguiente manera. En el capítulo 1, se plantea el análisis de contexto, la problemática a abordar y los objetivos del presente trabajo de tesis. En el capítulo 2 se plantea un cambio de enforque hacia modelos dirigidos por datos, donde se deja sobre relieve la importancia que tienen los mismos, y como la calidad de los datos impactan en los modelos que toman decisiones basadas en datos.
En el capítulo 3 se presenta un estudio evolutivo de las redes neuronales y donde se presentan y exponen los diferentes componentes de una red neuronal profunda, arquitecturas y métodos de aprendizaje. En el capítulo 4 se presenta el estado de arte, el cual brinda una revisión sistemática de la literatura exhaustiva sobre el tema. En el capítulo 5 trata sobre la necesidad de sistemas de IA interpretables, y el uso responsable de estas tecnologías basadas en los individuos. En el capítulo 6 se presenta y explica el algoritmo COLOSSUS como método de extracción y explicación de reglas basado en datos y en lógica. En el capítulo 7 se presentan casos de corroboración empírica sobre diferentes conjuntos de datos, y se explica empíricamente la función de distancia que complementa al algoritmo. En el capítulo 8 se exponen los trabajos relacionados y se provee una indicación del grado de originalidad del presente trabajo de tesis. Además, se realiza una comparación de diferentes modelos que atacan el problema de la explicabilidad en DNN, desde diferentes ángulos y que utilizan diferentes técnicas y métodos. En el capítulo 9 se presentan las conclusiones y trabajo futuro. Finalmente, todas las siglas utilizadas en el presente trabajo pueden ser consultadas en el Anexo I – Acrónimos y siglas.

# CAPITULO 2 Modelado Dirigido por Datos

## 2.1 Introducción

El proceso de aprendizaje humano es un viaje complejo a través de la observación y la experiencia del mundo, viaje del cual recopilamos datos de propiedades, a veces fácilmente cuantificables en su aprehensión, otras, de tinte más subjetivo (lo propiamente cualitativo). El adjetivo "*humano"* que califica al acto de aprender*,* requiere una precisión *anterior* que ronda en lo *subjetivo,* es decir en ese *quien* conoce. La persona que conoce el mundo interviene modificando algo, propio, de terceros, la materia, la cultura y la natura.

Hasta ahora, nada de lo dicho escapa a la moralidad o ética de los actos humanos, que buscan verdad científica y bondad, es decir que *no dañe y que aporte un valor diferencial virtuoso.* Conocer, investigar tendrán implicancias morales, porque todo lo que de la voluntad humana dependa es moral.

A partir de la observación, como humanos que conocemos el mundo, relacionamos eventos con los datos de esas propiedades. Es a partir de experiencias repetitivas a partir de las cuales podemos determinar algunos patrones que relacionan eventos con datos y eventos con eventos mismos. En el caso del descubrimiento científico, estos patrones y relaciones se formalizan en leyes y ecuaciones, los datos se formalizan en propiedades y variables, y las observaciones se formalizan en mediciones de eventos, que pueden ser acciones o propiedades en sí mismas. Leyes y ecuaciones, propias de la ciencia, nos permiten realizar predicciones y facilitan la transmisión del procedimiento de aprendizaje de una forma muy compacta, con la mínima cantidad de información. Sin embargo, el proceso clásico de aprendizaje en ciencias es un proceso lento que requiere mucha experiencia observacional, para descubrir las principales variables involucradas y su influencia en los eventos para una cantidad probablemente enorme de combinaciones posibles, perdiendo con frecuencia variables relevantes no previstas. Además, el enfoque científico clásico se basa en hipótesis y, por lo tanto, está sesgado por ellas.

El método científico se estableció para arbitrar la subjetividad intrínseca del conocer y aprender humano, de lo propio de la inclinación humana por buscar explicaciones *metafísicas* que no se sostienen en la evidencia material, cuantificable que ocurre en la observación. Sin embargo, como toda pretensión humana de asignación de verdad, el método científico clásico todavía está sesgado por el pensamiento deductivo de la mente humana.

El modelo como resultado y los procedimientos que acerquen al humano a conocer la verdad a través de datos, materia que ocupa esta tesis, busca en lo posible, y de manera deseable, un enfoque implícito e imparcial de nuestra experiencia humana de

aprendizaje basado en ellos (los datos) sin procesar observaciones reales (por tanto, subjetivas). *Estos procedimientos tienen la ventaja adicional de probar correlaciones entre diferentes variables y observaciones, aprender patrones no previstos en la naturaleza y permitirnos descubrir nuevas leyes científicas o incluso más, realizar predicciones sin la disponibilidad de dichas leyes*.

Como humanos, estamos viviendo la era de la *ciencia de datos*, que impacta en todos los aspectos de la vida personal (y en ella implicado lo económico, social, educativo, científico etc). Los datos adquieren sentido porque la persona que conoce vive dos pasos propios de su función humana: la de informarse y la de decidir sobre los actos de manera responsable y libre.
Los datos en el mundo organizado son datos que atraviesan a las personas y sus decisiones en un mundo desarrollado. De hecho 10 de los 17 Objetivos del Desarrollo Sostenible (Agenda 2030) que lideran la economía global son *objetivos digitales.*
*Fuente:* *https://www.un.org/sustainabledevelopment/es/2015/09/la-asamblea-general-adopta-la-agenda-2030-para-el-desarrollo-sostenible/*

Los procedimientos basados en datos están dando lugar a una nueva economía *digital*. La ciencia basada en datos también cambiará nuestras vidas y la forma en que hacemos ciencia y desarrollamos soluciones y decidimos conductas. La recopilación de datos, la extracción de datos y la visualización de datos también serán de suma importancia en el descubrimiento científico (Montáns et al., 2019).
Luego de estas precisiones, de corte general y necesario, entendidas con los determinantes y limitaciones esbozados, el propósito *de este capítulo es el de poner en relieve la importancia que tiene los datos para el diseño de un modelo y obtención de modelos de DL de buena calidad (por tanto, moral). Además, como se dijo anteriormente, es deseable para la voluntad humana informada, alcanzar la explicabilidad del propósito de tal modelo.*
*Para obtener un sistema explicable, nos apoyaremos sobre la idea de que, si los modelos de DL pueden extraer patrones y aprender de los datos, los datos también deben poder explicarse.*

## 2.2. Modelado dirigido por datos

En el pasado, los datos en los que se basa la ciencia y la ingeniería eran escasos y frecuentemente se obtenían mediante experimentos propuestos para verificar una determinada hipótesis. Cada experimento solo podía producir datos muy limitados. Hoy en día, los datos son abundantes y se recopilan abundantemente en cada experimento individual a un costo muy pequeño. El modelado dirigido por datos y el descubrimiento científico es un cambio de paradigma sobre cómo se abordan muchos problemas, tanto en ciencia como en ingeniería. Algunos campos científicos llevan tiempo utilizando inteligencia artificial debido a la dificultad inherente de obtener leyes y ecuaciones para describir algunos fenómenos. Sin embargo, hoy en día los enfoques basados en datos también están inundando campos como la mecánica y la ciencia de los materiales, donde el enfoque tradicional parecía ser muy satisfactorio. (Montáns et al. 2019).

En esta sección nos enfocaremos en analizar como la adopción de un enfoque centrado en los datos puede mejorar el rendimiento de un algoritmo de aprendizaje. Para ello se identifican dos visiones sobre el desarrollo de modelos de IA, el desarrollo centrado en modelos versus el desarrollo centrado en datos.
Con la mirada centrada en el proceso dirigido por el modelo, los desarrolladores de IA tomarían los datos que tienen disponibles y luego trabajarían arduamente para desarrollar un modelo que funcione lo mejor posible con los datos disponibles. Debido a que gran parte de la investigación académica sobre IA fue impulsada por investigadores que obtuvieron un conjunto de datos de referencia e intentaron desarrollar el mejor modelo posible con ese punto de referencia, la mayoría de las investigaciones académicas sobre IA se centran en el modelo, debido a que el conjunto de datos de referencia es una cantidad fija y una calidad dada. Desde esta perspectiva, *el desarrollo centrado en el modelo mantendría los datos fijos y mejoraría iterativamente el modelo*. *Entonces, desde este punto de vista dirigida por el modelo, se mantendrían los datos fijos y se mejoraría iterativamente el código o el modelo.*
Existe un gran reto al tratar de mejorar los modelos existentes, y esto implica una perspectiva distinta de los métodos tradicionales de inteligencia artificial, la cual es más beneficiosa para muchas aplicaciones. *Se trata de pasar de un enfoque basado en el modelo a uno centrado en los datos.*
Desde esta perspectiva, consideramos que la calidad de los datos es primordial y se pueden utilizar herramientas como el análisis de errores o el *aumento de datos (data augmentation)* para mejorar sistemáticamente la cantidad y calidad de los datos. Para muchas aplicaciones, si los datos son lo suficientemente buenos, hay múltiples modelos que funcionarán bien. Bajo esta premisa, *podemos mantener fijo el código y mejorar iterativamente los datos*.

Existe un espacio para el desarrollo centrado en modelos y hay un espacio para el desarrollo centrado en datos. Si pasamos de la experiencia de modelado centrado en el *modelo de IA* con aprendizaje automático, a considerar la adopción de una visión centrada en los datos, donde se intenta mejorar el rendimiento de los resultados de aprendizaje, nuestro interrogante orbitará sobre el ¿cómo hacer para que el conjunto de datos resulte aún mejor? Uno de los métodos en el *aumento de datos*.

Las tres entradas clave que intervienen en el entrenamiento de un modelo de aprendizaje automático son el *código que es el algoritmo o la arquitectura del modelo* de red neuronal que se vaya a definir. También debemos definir el *set de hiperparámetros* y luego están *los datos*. Entonces al ejecutar el código con el conjunto de *hiperparámetros* sobre el conjunto de datos, obtendremos el modelo de aprendizaje automático que buscamos; Un modelo de aprendizaje automático que aprenderá, por ejemplo, de clips de audio y entregará transcripciones de texto.
Muchos trabajos de investigación o trabajos académicos tienden a mantener los datos fijos y varían el código, y también los hiperparámetros para tratar de obtener un buen rendimiento. Por el contrario, para muchos equipos de producto, si su objetivo principal es simplemente construir e implementar un valioso sistema de aprendizaje automático que funcione, puede ser aún más efectivo mantener el código fijo y, en cambio, *centrarse en optimizar los datos y tal vez los hiperparámetros*. Para obtener un modelo de alto rendimiento, un sistema de aprendizaje automático incluye tanto código como

datos y también hiperparámetros que pueden ser un poco más fáciles de optimizar que el código o los datos. Entonces, en lugar de tener una visión centrada en el modelo y tratar de optimizar el código para un conjunto de datos fijos para muchos problemas, puede implementarse el código abierto y, en su lugar, solo concéntrese en optimizar los datos. De esta forma, durante el modelado, se debe seleccionar y entrenar alguna arquitectura o modelo. *Tal vez algún análisis de errores de la arquitectura de la red neuronal pueda decirnos dónde la performance del modelo aún no es la adecuada. Al utilizar este análisis de errores para entender cómo mejorar sistemáticamente el set de datos, también se estará mejorando el código*. *A menudo, si el análisis de errores puede decirnos cómo mejorar sistemáticamente los datos, esta puede ser una forma muy eficiente de obtener un modelo de alta precisión.*

Parte del desafío de la tecnología involucrada, radica en evitar obtención de datos que sin un sentido específico procurado por aquel que se plantea su incorporación en el diseño determinado, resulta fútil y costoso en términos económicos y morales, todo ello adecuado a los principios de AI Responsable. La proporción entre finalidad, adecuación y resultado procurado. La recopilación virtuosa de datos ayuda a la seguridad, eficacia, eficiencia y sostenibilidad del modelo. Finalmente, cuando se haya entrenado el modelo y cuando el análisis de errores parezca sugerir que funciona lo suficientemente bien, entonces estaremos listos para comenzar la implementación.

## 2.3. Hacia datos de buena calidad

El aumento cada vez mayor del poder computacional en las últimas décadas ha llevado a avances significativos en las técnicas estadísticas y de aprendizaje automático. La colección de algoritmos y métodos de minería de datos desarrollados como resultado han formado la arquitectura matemática central de los agentes (sin limitarse a softwares) de inteligencia artificial, y aunque la IA tiene una larga historia en el descubrimiento científico, los enfoques dirigidos por datos en las computadoras modernas pueden ahora ingerir y procesar algoritmos a escala. El poder de cómputo si bien es una condición en potencia necesaria, no se agota en suficiencia. Esto ha sido posible en gran medida no solo por la potencia computacional sino también por las tecnologías de almacenamiento de datos. De hecho, cantidades tan grandes de datos nos brindan nuevas oportunidades para el descubrimiento basado en datos (Montáns et al., 2019).

La idea sobre el modelo de pensamiento sobre el cambio de *big data* a *good data*, radica en que una gran cantidad de IA moderna o aquella que implica el *deep learning*, creció *ad intra* de grandes empresas de Internet de consumo con quizás miles de millones de usuarios, y empresas de esta tipología, tienen una gran cantidad de datos sobre la *población con acceso a conectividad y dispositivos* que transforma en masivo el número de adherencia coincidente con el perfil de sus usuarios. Poseer grandes volúmenes de datos como estos, contribuye al rendimiento potencialmente eficiente de los algoritmos de ML. Pero, una situación que es común para compañías de software de consumo masivo en relación con la disponibilidad de enormes cantidades de datos, para otras industrias para las cuales los son de igual importancia, simplemente no hay tal cantidad

de datos disponibles. Puede ser aún más importante que esas aplicaciones se centren no solo en *big data* sino también en *good data*. En este punto es importante destacar que, si es posible garantizar datos de alta calidad constantes en todas las fases del ciclo de vida del proyecto de aprendizaje automático, se podrá garantizar la existencia de una implementación de aprendizaje automático confiable y de alto rendimiento.
Por datos de buena calidad diremos que *son aquellos que cubren los casos más importantes*, para lo que se debe tener una buena cobertura para diferentes entradas de *x*. En caso de observar que no disponemos de suficientes datos de buena calidad, el aumento de datos puede ayudar a obtener más datos, obtener entradas más diversas para *x*, y poder así brindar esa cobertura.
Entonces, los datos de buena calidad también se definen de manera consistente con la definición de etiquetas y eso es inequívoco. Los datos de buena calidad también tienen retroalimentación oportuna de los datos de producción. Finalmente, para resumir, *diremos que es necesario un conjunto de datos de tamaño razonable, durante el ciclo de vida del proyecto de aprendizaje automático, dado que los modelos de DL no aprenden más allá de un umbral conocido en relación con la cantidad de datos*.

## 2.4. Data Augmentation

La mejor manera de hacer que un modelo de aprendizaje automático generalice mejor es entrenarlo con más datos. Por supuesto, en la práctica, la cantidad de datos que tenemos es limitada y normalmente ruidosa. Una forma de solucionar este problema es crear datos artificiales y agregarlos al conjunto de entrenamiento. Para algunas tareas de aprendizaje automático, es razonablemente sencillo crear nuevos datos falsos.
Este enfoque es el más fácil para problemas de clasificación. Un clasificador necesita tomar una entrada **x** compleja y de alta dimensionalidad y resumirla con una única identidad de categoría **y**. Esto significa que la principal tarea a la que se enfrenta un clasificador es ser invariante frente a una amplia variedad de transformaciones. Podemos generar nuevos pares (x; y) fácilmente simplemente transformando las entradas **x** en nuestro conjunto de entrenamiento.

La inyección de ruido en la entrada de una red neuronal (Sietsma y Dow, 1991) también puede verse como una forma de aumento de datos. Para muchas tareas de clasificación e incluso algunas de regresión, la tarea aún debería ser posible de resolver incluso si se agrega un pequeño ruido aleatorio a la entrada. Sin embargo, las redes neuronales demuestran no ser muy resistentes al ruido (Tang y Eliasmith, 2010). Una forma de mejorar la solidez de las redes neuronales es simplemente entrenarlas con ruido aleatorio aplicado a sus entradas. La inyección de ruido de entrada es parte de algunos algoritmos de aprendizaje no supervisados, como el codificador automático de eliminación de ruido (Vincent et al., 2008). La inyección de ruido también funciona cuando el ruido se aplica a las unidades ocultas, lo que puede verse como un aumento del conjunto de datos en múltiples niveles de abstracción. Poole et al., (2014) mostró recientemente que este enfoque puede ser muy eficaz siempre que se ajuste cuidadosamente la magnitud del ruido. La deserción, una poderosa estrategia de regularización, puede verse como un proceso de construcción de nuevas entradas multiplicando por ruido.

Al comparar los resultados de las pruebas comparativas de aprendizaje automático, es importante tener en cuenta el efecto del aumento del conjunto de datos. A menudo, los esquemas de aumento de conjuntos de datos diseñados a mano pueden reducir drásticamente el error de generalización de una técnica de aprendizaje automático. Para comparar el rendimiento de un algoritmo de aprendizaje automático con otro, es necesario realizar experimentos controlados.
Cuando se compara el algoritmo de aprendizaje automático A y el algoritmo de aprendizaje automático B, debemos asegurarnos de que ambos algoritmos se evalúen con los mismos esquemas de aumento de conjuntos de datos diseñados a mano. Suponga que el algoritmo A funciona mal sin aumento del conjunto de datos, y el algoritmo B funciona bien cuando se combina con numerosas transformaciones sintéticas de la entrada. En tal caso, las transformaciones sintéticas probablemente causaron y expliquen la mejora del rendimiento en el uso del algoritmo B de aprendizaje automático. A veces, decidir si un experimento se ha controlado correctamente requiere un juicio subjetivo. Por ejemplo, los algoritmos de aprendizaje automático que inyectan ruido en la entrada están realizando una forma de aumento del conjunto de datos.
Por lo general, las operaciones que son de aplicación general (como agregar ruido gaussiano a la entrada) se consideran parte del algoritmo de aprendizaje automático, mientras que las operaciones que son específicas de un dominio de aplicación (como recortar una imagen al azar) se consideran pasos de preprocesamiento independientes.

## 2.5. Calidad de los datos y procesamiento estocástico

En cualquier modelo o proceso dirigido por datos, la calidad de los datos es muy importante, ya que los datos erróneos o sesgados pueden producir modelos y decisiones erróneos. Es evidente que la redundancia de datos y los análisis de series temporales de datos pueden, en la mayoría de las ocasiones, mejorar la calidad de los datos para generar mejores modelos y predicciones de modelos. Según la norma ISO 9001:2015, la definición y evaluación de la calidad de los datos depende del contexto y el uso de estos. En el contexto del aprendizaje de modelos en ciencia e ingeniería, la calidad de los datos cuantitativos, tal como se obtienen de los experimentos, parece fundamental para la calidad de los modelos y las propias predicciones. En este sentido, algunas características (dimensiones de calidad) son muy importantes, como la precisión, la exhaustividad, la coherencia y la credibilidad.
La calidad en la descripción de un proceso a menudo está relacionada con la comprensión de su naturaleza estocástica o de las variables y observaciones involucradas, por lo que la aplicación de métodos basados en datos a fenómenos y procesos estocásticos también es un área de investigación actual.
El siglo XXI es considerado el siglo del Big Data. El cambio de siglo ha supuesto un cambio histórico en nuestra sociedad. Los ordenadores, Internet y los nuevos dispositivos digitales están produciendo una gran cantidad de datos. El poder computacional actual y la computación en la nube también permiten un número imprevisto de simulaciones. Sin embargo, en lugar de sentirnos abrumados por tal cantidad de información sin procesar, estamos aprendiendo cómo aprovechar el nuevo paradigma. Las empresas de software nos han enseñado que se pueden obtener muchos beneficios e información clave del análisis de datos (Montáns et al., 2019).

## 2.6. El poder de los algoritmos: inducir conocimiento a partir de datos

En un famoso ensayo de 1959, el físico y premio Nobel Eugene Wigner se maravilló ante lo que llamó La efectividad irrazonable de matemáticas en las ciencias naturales: ¿Por qué milagro resultan las leyes inducidas a partir de observaciones escasas que se aplican mucho más allá de ellas? ¿Cómo pueden las leyes ser muchos órdenes de magnitud más precisos que los datos en los que se basan? Sobre todo, ¿por qué el lenguaje simple y abstracto de las matemáticas puede capturar con precisión tanto de nuestro mundo infinitamente complejo? Wigner consideró esto un misterio profundo, de partes iguales, afortunados e insondables. Sin embargo, tal y como se plantea, el algoritmo maestro es una extensión lógica de la misma.

Si el mundo fuera solo una confusión floreciente y vibrante, habría razón para dudar de la existencia de un *learner* universal. Pero si todo lo que experimentamos es producto de unas pocas leyes simples, entonces tiene sentido que un solo algoritmo pueda inducir todo lo que se puede inducir. Todo lo que tiene que hacer el algoritmo maestro es proporcionar un atajo a las consecuencias de las leyes, reemplazando derivaciones matemáticas imposiblemente largas por otras muchas más cortas basadas en observaciones reales. (Domingos, 2018).

Los algoritmos de aprendizaje son capaces de predecir con precisión eventos raros y nunca vistos; incluso se podría decir que de eso se trata el aprendizaje automático. Una objeción relacionada y que se escucha con frecuencia es que *los datos no pueden reemplazar a la intuición*. De hecho, es al revés: la intuición humana no puede reemplazar datos. La intuición es lo que se usa cuando no se conocen los hechos, y como a menudo no es así, la intuición es preciosa. Pero cuando la evidencia está ante nosotros, ¿por qué la negaríamos? Debido a la afluencia de datos, el límite entre la evidencia y la intuición está cambiando rápidamente y, como ocurre con cualquier revolución, deben superarse las formas arraigadas.

*Deep Learning* induce teorías desde los datos. Una teoría científica es un conjunto de restricciones sobre lo que el mundo podría ser, y no una descripción completa de este. Para obtener esto último, tenemos que combinar las teorías con los datos. En tal sentido, el poder de una teoría reside en que magnitud simplifica nuestra descripción del mundo. Para describir nuestro dominio los más preciso posible, necesitamos un conjunto de datos nuevo a intervalos regulares. ***En este sentido, no importa cuán bueno sea nuestro learner, dado que será tan buen como los datos que se le provean***. El punto entonces es, inferir conocimiento desde los datos, debido a que es más eficiente en costos; Por lo tanto, cuanto más generalice nuestro algoritmo, mejor.

## 2.7. Conclusiones

Los procedimientos dirigidos por datos se enfocan en los datos e intentan extraer variables y relaciones directamente de los datos sin procesar, brindando con frecuencia respuestas más precisas sin el uso de leyes y ecuaciones analíticas clásicas.

En este capítulo se destaca la creciente importancia de los enfoques centrados en datos en el campo de la inteligencia artificial y el aprendizaje automático. Esto se debe al hecho de que tener más datos de alta calidad conduce a un mejor rendimiento y

generalización de los modelos. Con la disponibilidad de grandes cantidades de datos y los avances en la potencia computacional y el almacenamiento de datos, se ha producido un cambio de paradigma hacia el descubrimiento y el modelado basados en datos, no solo en la IA sino también en otros campos científicos y de ingeniería.

Un aspecto clave de los enfoques centrados en datos es el aumento en la cantidad de estos, lo que puede implicar crear datos artificiales y agregarlos al conjunto de entrenamiento para mejorar la calidad y diversidad del conjunto de datos. Esto es particularmente útil en problemas de clasificación, donde los modelos deben ser invariantes ante diversas transformaciones.

Es fascinante ver cómo los enfoques centrados en datos están revolucionando los métodos tradicionales en campos como la mecánica y la ciencia de los materiales. Esto resalta el potencial del descubrimiento basado en datos para complementar y mejorar los métodos tradicionales, lo que lleva a soluciones nuevas y mejoradas para problemas desafiantes.

En general, el floreciente campo de los enfoques centrados en datos y su impacto no sólo en la inteligencia artificial, sino también en diversas disciplinas científicas y de ingeniería, seguirán desempeñando un papel crucial en el avance de la tecnología y el conocimiento.

# CAPITULO 3 Redes Neuronales

## 3.1. Fundamentos Redes Neuronales Profundas

Las redes neuronales se han convertido en la familia de algoritmos de *machine learning* más populares de esta última oleada que estamos viviendo. En realidad, como modelo computacional, existen desde mediados del siglo pasado, pero no ha sido hasta hace unos años con la mejora de la técnica y la tecnología que hemos empezado a utilizarlas asiduamente en, por ejemplo, reconocimiento de caracteres, de imágenes, de voz, predicción bursátil, generación de texto, traducción de idiomas, prevención de fraude, conducción autónoma, análisis genético, pronóstico de enfermedades entre otros.

El futuro ha llegado, y como vemos, a través de una familia de algoritmos muy potentes podemos modelar comportamientos inteligentes. Pero ¿cómo funcionan estos algoritmos? Como suele ocurrir con la mayoría de los comportamientos y estructuras avanzadas, la complejidad de estos sistemas emerge de la interacción de muchas partes más simples trabajando conjuntamente. Este enfoque de aprendizaje profundo de IA permite que las computadoras aprendan de la experiencia y comprendan el mundo en términos de una jerarquía de conceptos, con cada concepto definido a través de su relación con conceptos más simples. Al recopilar el conocimiento de la experiencia, este enfoque evita la necesidad de que los operadores humanos especifiquen formalmente todo el conocimiento que necesita la computadora. Entonces la jerarquía de conceptos permite que la computadora aprenda conceptos complicados construyéndolos a partir de otros más simples. Si se dibuja un gráfico que muestre cómo se construyen estos conceptos uno encima del otro, el gráfico es profundo, con muchas capas. Goodfellow et al., (2016),

En el caso de una red neuronal, a cada una de estas partes se le denomina neurona, la cual podemos utilizar para codificar información dentro de ellas. Estas neuronas se agrupan para formar una red neuronal y así conformar sistemas que son capaces de aprender. Una neurona es la unidad básica de procesamiento que vamos a encontrar dentro de una red neuronal, similar a una neurona biológica estas neuronas tienen conexiones de entrada a través de los que reciben estímulos externos. Con estos valores de entrada, la neurona realizará un cálculo interno y generará un valor de salida. Vemos entonces que realmente una neurona no deja de ser un nombre tomado de la biología para referirnos a una función matemática. Internamente la neurona utiliza todos los valores de entrada para realizar una suma ponderada de ellos.

El aprendizaje profundo moderno proporciona un marco poderoso para el aprendizaje supervisado. Al agregar más capas y más unidades dentro de una capa, una red profunda puede representar funciones de complejidad creciente. La mayoría de las tareas que consisten en un vector de entrada a un vector de salida, y que son fáciles de realizar rápidamente para una persona, se pueden lograr a

través del aprendizaje profundo, dados modelos suficientemente grandes y conjuntos de datos de mapeo suficientemente grandes de ejemplos de entrenamiento etiquetados. Otras tareas, que no pueden describirse como la asociación de un vector con otro, o que son lo suficientemente difíciles como para que una persona requiera tiempo para pensar y reflexionar para realizar la tarea, quedan fuera del alcance del aprendizaje profundo por ahora (Goodfellow et al., 2016).

## 3.2. El modelo biológico

Una neurona es una célula viva, y como tal, los mismos elementos que forman parte de todas las células biológicas. Además, contienen elementos característicos que las diferencian, (Britos, 2005).
Una neurona consta de un cuerpo celular más o menos esférico del que se desprende una rama principal o axón y varias ramas más cortas llamadas dendritas. A su vez el axón presenta ramas en torno a su punto de arranque, y con frecuencia se ramifica extensamente cerca de su extremo.
Un de las características que diferencian a las neuronas del resto de las células vivas, es la capacidad que tienen éstas de comunicarse. En términos generales, las dendritas y el cuerpo celular reciben señales de entrada, el cuerpo celular las combina e integra y emite señales de salida. El axón transporta estas señales a los terminales axónicos, que se encargan de distribuir información de miles de otras neuronas y envía información a miles de otras neuronas. Se calcula que en el cerebro humano existe del orden de 1015 conexiones.

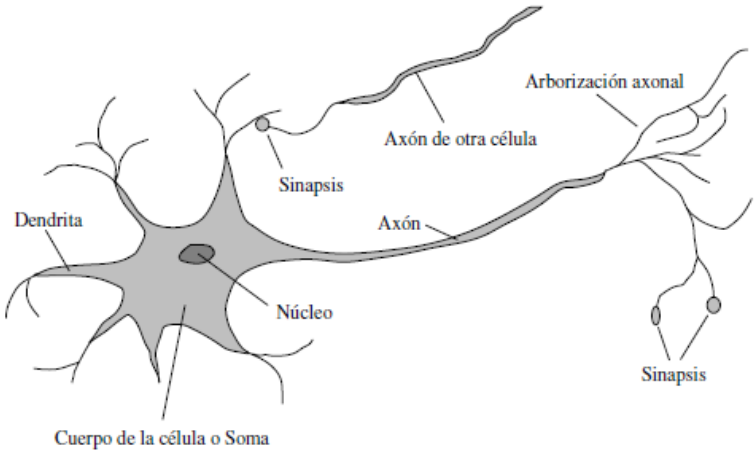


*Figura 2. Estructura de una neurona biológica (Russell y Norvig 2010)*

## 3.3. Elementos de una red neuronal artificial

Las redes neuronales son modelos que intentan reproducir el comportamiento del cerebro. Tal modelo realiza una simplificación, averiguando cuales son los elementos relevantes del sistema. Una elección adecuada de sus características,

más una topología conveniente, es el procedimiento convencional utilizado para construir redes capaces de realizar una determinada tarea.

Cualquier modelo de red neuronal consta de dispositivos elementales de proceso: ***las neuronas***. A partir de ellas, se pueden generar representaciones específicas de forma tal que un estado conjunto de ellas puede significar una letra, un numero o cualquier otro objeto (Britos, 2005). En general, se pueden encontrar tres tipos de neuronas:

Aquellas que reciben estímulos externos, relacionadas con el aparato sensorial, que tomarían la información de entrada proveniente del exterior a la red.

Dicha información se transmite a ciertos elementos internos que se ocupan de su procesado. Es en las sinapsis y neuronas correspondientes a este segundo nivel donde se genera cualquier tipo de representación interna de la información. La ponderación de cada una de las entradas viene dada por el peso que se le asigna a cada una de las conexiones de entrada, es decir que cada conexión que llega a nuestra neurona tendrá asociado un valor que servirá para definir con qué intensidad cada variable de entrada afecta a la neurona. Estos pesos son los parámetros de nuestro modelo, y serán los valores que podremos ajustar para que nuestra red neuronal pueda aprender. Observamos también que los pesos de la red actúan como parámetros de la *función*; escribiendo **W** para los parámetros, la red calcula $h_w(X)$. Ajustando los pesos, cambiamos la función que representa la red. Esta es la manera en que se produce el aprendizaje en las redes neuronales (Russell y Norvig 2010). Dado que no tienen relación directa con la información de entrada ni con las salidas, estos elementos se denominan **unidades ocultas**.

Una vez finalizado el periodo de procesado, la información llega a las unidades de salida, cuya misión es dar la respuesta del sistema.

La neurona artificial como se muestra en la Fig. 3, pretende mimetizar las características más importantes de las neuronas biológicas. Cada neurona *i-ésima* está caracterizada en cualquier instante por un valor numérico denominado **valor** o **estado de activación**, una función de activación que transforma el estado actual de activación en una señal de salida. Una conexión de la unidad *j* a la unidad *i* sirve para propagar la activación $a_j$ de *j* a *i*. Además, cada conexión tiene un peso numérico $W_{ji}$ asociado, que determina la fuerza y el signo de la conexión. Cada unidad *i* primero calcula una suma ponderada de sus entradas (Russell y Norvig, 2010):

$$in_j = \sum_{j=0}^{n} W_{j,i} a_j$$

EQ.3.1

La señal de salida es enviada a través de canales de comunicación unidireccionales a otras unidades de red; en estos canales, la señal se modifica de acuerdo con la sinapsis (el peso $w_{ij}$) asociada a cada uno de ellos según una determinada regla.

Las señales moduladas que han llegado a la unidad *j-ésima* se combinan entre ellas, generando de este modo la entrada total, (Britos, 2005).

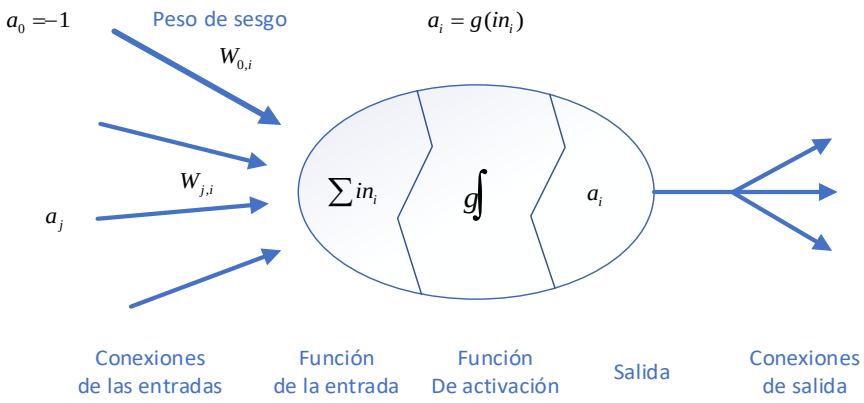


*Figura 3.* Un modelo matemático sencillo para una neurona. (Russell y Norvig 2010)

### 3.3.1. La neurona artificial

Una neurona es la unidad básica de procesamiento que se encuentra dentro de una red neuronal. Similar a una neurona biológica, estas neuronas tienen conexiones de entrada a través de los que reciben estímulos externos. Con estos valores de entrada, la neurona realizará un cálculo interno y generará un valor de salida.
Si se tienen N unidades (neuronas), podemos ordenarlas arbitrariamente y designar a la *j-ésima* unidad como $U_j$, La responsabilidad de la unidad consiste en recibir las entradas de las neuronas vecinas y calcular un valor de salida el cual es enviado a todas las neuronas conectadas con $U_j$.
Como se mencionó anteriormente, en cualquier red neuronal profunda que se esté modelando, podemos identificar tres tipos de unidades: entradas, ocultas y salidas. Las unidades de entrada reciben señales desde el exterior de la red, es decir, desde el entorno. Las neuronas en la salida de la red son las responsables de entregar las respuestas (predicciones o clasificaciones), como respuesta a un estímulo externo. Estas salidas pueden a su vez, controlar otros sistemas. Por último, las unidades ocultas son aquellas cuyas entradas y salidas se encuentran dentro del sistema y no mantienen contacto con el exterior.
Se conoce como **capa** o **nivel** a un conjunto de neuronas cuyas entradas provienen de la misma fuente (que puede ser otra capa de neuronas).
La ponderación de cada una de las entradas viene dada por el peso que se le asigna a cada una de las conexiones de entrada, es decir que cada conexión que llega a nuestra neurona tendrá asociado un valor que servirá para definir con qué intensidad cada variable de entrada afecta a la neurona. Intuitivamente esto se suele representar como potenciómetros que podemos subir o bajar para modificar positiva o negativamente el valor de nuestra suma (Russell y Norvig, 2010).

### 3.3.2. Estado de Activación

Adicionalmente al conjunto de unidades, la representación necesita los estados del sistema en un tiempo *t*. Esto se especifica por un vector de números reales *A(t)*, que representa el estado de activación del conjunto de unidades de procesamiento (Britos,

2005). Cada elemento del vector representa la activación de una unidad en el tiempo *t*. La activación de una unidad $U_i$ en el tiempo *t* se designa por $a_i(t)$, es decir:

$$A(t) = (a_i(t), a_2(t), \dots, a_N(t))$$

EQ.3.2

Entonces, el procesamiento que realiza la red se ve como la evolución de un patrón de activación en el conjunto de unidades que lo componen a través del tiempo.
Todas las neuronas que componen la red se hallan en un estado determinado y los valores que puede tomar un estado pueden ser continuos o discretos. Además, pueden ser limitados o ilimitados. Si son discretos, suelen tomar un conjunto pequeño de valores o bien valores binarios.
Finalmente, es necesario saber qué criterios o reglas siguen las neuronas para alcanzar tales estados de activación. En principio esto va a depender de dos factores: Por un lado, es necesario tener idea del mecanismo de interacción entre las neuronas. El estado de activación estará fuertemente influenciado por tales interacciones, dado que el efecto que producirá una neurona sobre otra será proporcional a la fuerza, peso o magnitud de la conexión entre ambas (Goodfellow et al., 2016). Por otro lado, la señal que envía cada neurona a sus vecinas dependerá de su propio estado de activación.

### 3.3.3. Conexiones entre neuronas

Las conexiones que unen a las neuronas que forman una RNA tienen asociado un peso, que es el que hace que la red aprenda.
Consideramos ***a*** como el valor de salida de una neurona ***i*** en un determinado instante. Una neurona recibe un conjunto de señales que le dan información del estado de activación de todas las neuronas con las que se encuentra conectada. Cada conexión (sinapsis) entra la neurona *i* y la *j* esta ponderada por un peso $w_{ij}$. Normalmente, como simplificación, se considera que el efecto de cada señal es aditivo, de tal forma que la entrada neta que recibe una neurona (potencial postsináptico) $Net_j$ es la suma del producto de cada señal individual, por el valor de la sinapsis que conecta ambas neuronas:

$$\mathrm{Net}_j = \sum_{i=1}^{N} w_{ij}\ a_i$$

EQ.3.3

Esta regla muestra el procedimiento a seguir para combinar los valores de entrada a una unidad con los pesos de las conexiones que llegan a esa unidad y es conocida como *regla de propagación.*
Suele utilizarse una matriz W con todos los pesos $w_{ij}$ que reflejan la influencia que tiene la neurona *j* sobre la neurona *i.* W es un conjunto de elementos positivos, negativos o nulos. Si $w_{ij}$ es positivo, indica que la interacción entre las neuronas *i* y *j* es excitadora; es decir que, siempre que la neurona *i* este activada, la neurona *j* recibirá una señal de *i* que tendera a activarla. Si $w_{ij}$ es negativo la sinapsis será inhibidora. En este caso si *i* esta activada, enviara una señal a *j* que tendera a desactivarla. Finalmente, si $w_{ij}$ es 0 se supone que no hay conexión entre ambas (Britos, 2005).

### 3.3.4. Función de activación

Entre las unidades o neuronas que forman una red neuronal artificial existe un conjunto de conexiones que unen unas con otras. Cada unidad $U_i$ transmite señales a aquellas que están conectadas con su salida. Cada neurona dentro de la red, tiene asociada una funciona de activación o transferencia $f(Net_i)$ que transforma la entrada neta de la neurona en una señal de salida $Y_i(t) = f(Net_i\ (t))$. El vector que contiene las salidas de todas las neuronas en un instante *t* es Y(y) = ($f_1$ ($Net_1$(t), $f_2$ ($Net_2$(t),…, $f_N$ ($Net_N$(t)).
La función de activación *g* se diseña con dos objetivos. Primero, queremos que la unidad esté ***activa*** (cercana a +1) cuando se proporcionen las entradas ***correctas***, e ***inactiva*** (cercana a 0) cuando se den las entradas ***erróneas***. Segundo, la activación tiene que ser no *lineal*, en caso contrario la red neuronal en su totalidad colapsaría con una función lineal sencilla. En la Fig. 4 se muestran dos posibles funciones *g*: la función **umbral** y la **función sigmoide** (también conocida como **función logística**). La función sigmoide tiene la ventaja de ser diferenciable, lo cual, es importante para el algoritmo del aprendizaje de los pesos (Russell y Norvig, 2010).

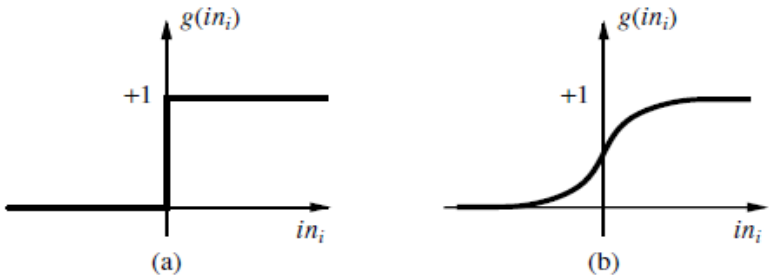


*Figura 4. (*a) La función de activación umbral, con salida 1 cuando la entrada es positiva y 0 en otro caso. (Algunas veces se usa en lugar de esta la función signo, con salida + 1 dependiendo de la señal de la entrada.) (b) La función sigmoide 1/(1 + *e*-*x*).(Russell y Norvig 2010)

Los pesos de sesgo $W_{0,i}$ constituyen el umbral *real* de la unidad, en el sentido de que la unidad se activa cuando la suma de los pesos de las entradas ***reales*** $\sum_{j=1}^{n} w_{j,i}\ a_j$ incluyendo la entrada del sesgo, excede $W_{0,i}$.
Normalmente, la función de activación no está centrada en el origen del eje de abscisas, sino que existe cierto desplazamiento debido a las características internas de la neurona y que no es igual en todas ellas. Este valor se denota como $\theta$ y representa el umbral de activación de la neurona *i*.

$$y_i(t+1) = f(Net - \theta_i) = f(\sum_{i=1}^{N} w_{ij}\ y_j(t) - \theta_i)$$

EQ.3.4

Típicamente, existen cuatro tipos de funciones de activación que determinan distintos tipos de neuronas (Britos, 2005):

- Función umbral

- Función lineal y mixta
- Sigmoidal
- Función Gaussiana

La **función umbral**, solo se utiliza cuando las salidas de la red son binarias. La salida de una neurona se activa solo cuando la entrada neta, es mayor o igual a cierto valor de umbral. La forma más simple de definir la activación de una neurona es considerar que ésta es binaria. La función de activación umbral se asocia a neuronas binarias en las cuales cuando la suma de las entradas es mayor o igual que el umbral de la neurona, la activación es 1; si es menor, la activación es 0 o -1. Las redes formadas por este tipo de neuronas son fáciles de implementar, pero a menudo sus capacidades están limitadas.
La **función lineal o identidad** responde a la expresión ***f(x) = x***. En las neuronas con función mixta, si la suma de las señales de entrada es menor que un límite inferior dado, la activación se define como 0 o -1. Si dicha suma es mayor que un límite superior, entonces la activación es 1. Si la suma de entradas está entre ambos limites, entonces la activación se define como una función lineal de la suma de las entradas.
Cualquier función definida simplemente en un intervalo de posibles valores de entrada, con un incremento monotónico, y que tenga ambos limites superiores e inferiores (función sigmoidal o arco tangente), podrá realizar la función de activación de forma satisfactoria.
Con la **función sigmoidal**, para la mayoría de los valores del estímulo de entrada (variables independientes), el valor dado por la función es cercano a uno de los valores asintóticos. Esto hace que, en la mayoría de los casos, el valor de salida este comprendido en la zona alta o baja del sigmoide. De hecho, cuando la pendiente es elevada, esta función tiende a la *función umbral*. Sin embargo, la importancia de la *función sigmoidal* es que su derivada es siempre positiva y cercana a cero para los valores grandes positivos o negativos; además, toma su valor máximo cuando X = 0. Esto hace que se puedan utilizar las reglas de aprendizaje definidas para la función umbral, con la ventaja de que la derivada está definida en todo el intervalo. La función umbral no tiene derivada definida en el punto de transición y esto no ayuda a los métodos de que utilizan derivadas como método de aprendizaje (Russell y Norvig, 2010).
Los centros y anchura de las **funciones neuronales de transferencia gaussiana** pueden adaptarse lo cual las hace más adaptativas que las funciones sigmoidales (Britos, 2005). Mapeos que suelen requerir dos niveles ocultos utilizando neuronas con funciones de transferencia sigmoidales, con frecuencia se pueden realizar con un solo nivel en redes con funciones gaussianas.
Para simplificar la expresión de la salida de una neurona *i*, es habitual considerar la existencia de una neurona ficticia con valor de salida asociada a la entrada de cada neurona *i* mediante una conexión con peso de valor. De esta forma la expresión de salida quedaría de la siguiente manera:

$$y_i(t+1) = f(Net - \theta_i) = f(\sum_{i=1}^{N} w_{ij}\ y_j(t) - \theta_i) = f(\sum_{i=0}^{N} w_{ij}\ y_j(t))$$

EQ.3.5

#### 3.3.4.1. Rectified Linear Units (ReLu)

Las unidades lineales rectificadas utilizan la función de activación $g(z) = \max\{0, z\}$. Estas unidades son fáciles de optimizar porque son muy similares a las unidades lineales. La única diferencia entre una unidad lineal y una unidad lineal rectificada es que una unidad lineal rectificada genera cero en la mitad de su dominio. Esto hace que las derivadas a través de una unidad lineal rectificada permanezcan grandes siempre que la unidad esté activa. Los gradientes no solo son grandes sino también consistentes.
Las unidades lineales rectificadas se utilizan normalmente sobre una transformación afín:

$$Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]} \quad \text{EQ.3.6}$$

$$A^{[l]} = g^{[l]}(Z^{[l]}) \quad \text{EQ.3.7}$$

Al inicializar los parámetros de la transformación afín, puede ser una buena práctica establecer todos los elementos de ***b*** en un valor positivo pequeño, como 0,1. De este modo es muy probable que las unidades lineales rectificadas estén inicialmente activas para la mayoría de las entradas en el conjunto de entrenamiento y permitan el paso de las derivadas.
Un inconveniente de las unidades lineales rectificadas es que no pueden aprender a través de métodos basados en gradientes en ejemplos para los que su activación es cero. Varias generalizaciones de unidades lineales rectificadas garantizan que reciben gradiente en todas partes.
Las unidades lineales rectificadas y todas las generalizaciones de la misma se basan en el principio de que los modelos son más fáciles de optimizar si su comportamiento es más cercano al lineal (Goodfellow et al., 2016).

#### 3.3.4.2. Tangente Logística Sigmoide e Hiperbólica

Antes de la introducción de las unidades lineales rectificadas, la mayoría de las redes neuronales utilizaban la función logística de activación sigmoide.

$$g(z) = \sigma(z)$$

o la función de activación de la tangente hiperbólica

$$g(z) = \tanh(z)$$

Estas funciones de activación están estrechamente relacionadas dado que $\tanh(z) = 2\sigma(2z) - 1$.
A diferencia de las unidades lineales por tramos, las unidades sigmoidales se saturan en la mayor parte de su dominio: se saturan a un valor alto cuando ***z*** es muy positivo, se saturan a un valor bajo cuando ***z*** es muy negativo y solo son muy sensibles a su entrada cuando ***z*** está cerca de 0. La saturación generalizada de unidades sigmoides puede dificultar mucho el aprendizaje basado en gradientes. Por esta razón, se desaconseja el uso en unidades ocultas en las redes ***feedforward***. Su uso como unidades de salida es

compatible con el uso del aprendizaje basado en gradientes cuando una función de costo adecuada puede deshacer la saturación del sigmoide en la capa de salida.
Cuando se debe usar una función de activación sigmoide, la función de activación de tangente hiperbólica generalmente se desempeña mejor que la sigmoidea logística. Se parece más a la función identidad, en el sentido de que $\tanh(0) = 0$, mientras $\sigma(0) = \frac{1}{2}$.
Debido a que *tanh* es similar a la función identidad cerca de 0, entrenando una red neuronal profunda $\hat{y} = w^T \tanh\,(U^T \tanh(V^T\, x))$, se asemeja a la formación de un modelo lineal $\hat{y} = w^T\, U^T\, V^T x$, siempre y cuando las activaciones de la red puedan mantenerse pequeñas. Este facilita el entrenamiento de la red *tanh* (Goodfellow et al. 2016).
Las funciones de activación sigmoidal son más comunes en entornos distintos de las redes ***feedforward***. Las redes recurrentes, muchos modelos probabilísticos y algunos codificadores automáticos tienen requisitos adicionales que descartan el uso de funciones de activación lineal por partes y hacen que las unidades sigmoideas sean más atractivas a pesar de los inconvenientes de la saturación.

## 3.4. Regla de aprendizaje

Existen muchas definiciones de concepto general de aprendizaje, y una de ellas podría ser: *La modificación del comportamiento inducido por la interacción con el entorno y cómo resultado de experiencias conducente al establecimiento de nuevos modelos de respuesta a estímulos externos*.
Esta definición fue enunciada muchos años antes de que surgieran las redes neuronales, sin embargo, puede ser aplicada también a los procesos de aprendizaje de estos sistemas. Biológicamente, se suele aceptar que la información memorizada en el cerebro está más relacionada con los valores sinápticos de las conexiones entre las neuronas que con las neuronas mismas: es decir, el conocimiento se encuentra en la sinapsis. En el caso de las redes neuronales artificiales, se puede considerar que el conocimiento se encuentra representado en los pesos y sesgos de las conexiones entre neuronas. Todo proceso de aprendizaje implica cierto número de cambios en estas conexiones. En realidad, puede decirse que se aprende modificando los valores de los pesos y sesgos de la red.
Al igual que el funcionamiento de una red depende del número de neuronas de las que disponga y de cómo estén conectadas entre sí, cada modelo dispone de su o sus propias técnicas de aprendizaje.

## 3.5. Representación vectorial

En ciertos modelos de redes neuronales, se utiliza la forma vectorial como herramienta de representación de algunas magnitudes. Si consideramos una red formadas por varias capas de neuronas idénticas, podemos considerar las salidas de cierta capa de n unidades como un vector n-dimensional $Y= (y_{1j}, y_{2j}, \ldots, y_{nj})$ .La entrada neta de la *j-ésima* unidad se puede escribir en forma de producto escalar del vector de entradas por el vector de pesos. Cuando los vectores tienen igual dimensión, este producto se define como la suma de los productos de los componentes de ambos vectores. En las

ecuaciones 3.6 y 3.7, se exponen la representación vectorial de propagación de las neuronas en las redes neuronales *feedforward* en su forma canónica.

$$net_j = \sum_{i=1}^{n} w_{ij}\, y_i$$

EQ.3.8

Para la ecuación 3.8, *n* representa el número de conexiones de la *j-ésima* unidad. La ventaja de la notación vectorial es que la ecuación anterior se puede escribir de la forma: $net_j = W.\ Y$

## 3.6. Estructura de una RNA

Existen dos categorías principales de estructuras de redes neuronales: acíclicas o **redes con alimentación-hacia-delante** y cíclicas o **redes recurrentes**.
Una red con *alimentación-hacia-adelante* representa una función de sus entradas actuales; de este modo, no tiene otro estado interno que no sea el de sus propios pesos. Por otro lado, una red recurrente permite que sus salidas alimenten sus propias entradas.
Esto significa que los niveles de activación de la red forman un sistema dinámico que debe alcanzar un estado estable, exhibir oscilaciones o incluso un comportamiento caótico. Además, la respuesta de la red dadas unas entradas depende de su estado inicial, que dependerá de entradas previas. Por lo tanto, las redes recurrentes (a diferencia de las redes con alimentación hacia delante) pueden tener memoria a corto plazo. Esto las hace más interesantes como modelos del cerebro, pero también más difíciles de entender (Russell y Norvig, 2010).

### 3.6.1. Formas de conexión entre neuronas

Se dice que una red es totalmente conectada si todas las salidas desde un nivel llegan a todos y cada uno de los nodos del nivel siguiente. La conectividad entre los nodos de una red neuronal está relacionada con la forma en que las salidas de las neuronas están canalizadas para convertirse en entradas de otras neuronas Fig. 5. La señal de salida de un nodo puede ser una entrada de otro elemento de proceso, o incluso ser una entrada de sí mismo (conexión auto recurrente).

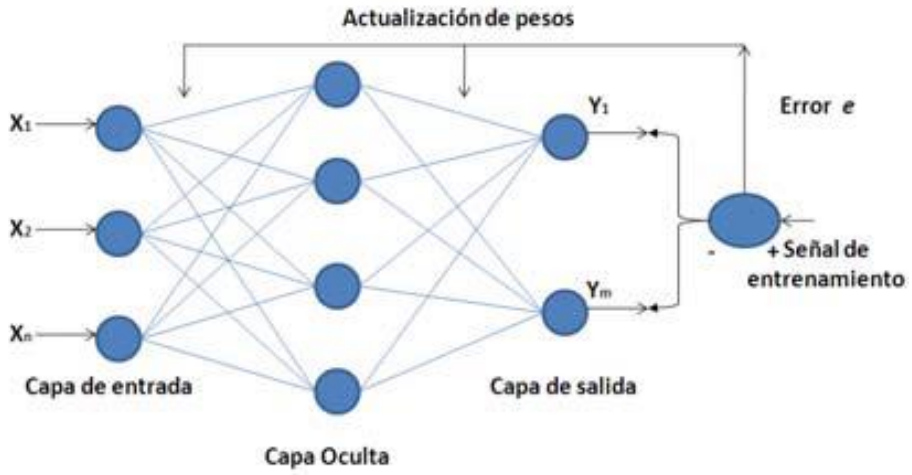


*Figura 5. Red neuronal con propagación hacia atrás*

Cuando ninguna salida de las neuronas es entrada de neuronas del mismo nivel o de niveles precedentes, la red se escribe como de propagación hacia adelante. Cuando las salidas pueden ser conectadas como entradas de neuronas de niveles previos o del mismo nivel, incluyéndose ellas mismas, la red es de propagación hacia atrás. Las redes de propagación hacia atrás que tienen lazos cerrados son sistemas recurrentes.

### 3.6.2. Características de las redes neuronales

Existen 4 aspectos que caracterizan una red neuronal: su topología, el mecanismo de aprendizaje, el tipo de asociación realizada entre la información de entrada y de salida y la forma de representación de estas informaciones.
La topología o arquitectura de las redes neuronales consiste en la organización y disposición de las neuronas en la red formando capas o agrupaciones de neuronas más o menos alejadas de la entrada y salida de la red. En tal sentido, los parámetros fundamentales de la red son: el número de capas, el número de neuronas por capa, el grado de conectividad y el tipo de conexiones entre neuronas (Goodfellow et al., 2016).

## 3.7. Redes monocapa

Una red con todas las entradas conectadas directamente a las salidas se denomina **red neuronal de una sola capa**, o red **perceptrón**. Dado que cada unidad de salida es independiente de las otras (cada peso afecta sólo a una de las salidas) podemos limitar nuestro estudio a los perceptrones con una única unidad de salida, como el que se muestra en la Fig. 6 (Russell y Norvig, 2010).

En las redes monocapa, como la red de *Hopfield* y la red *State In-a-box*, se establecen conexiones laterales entre las neuronas que pertenecen a la única capa que constituye la red. También pueden existir conexiones auto recurrentes (salida de una neurona conectada a la entrada).

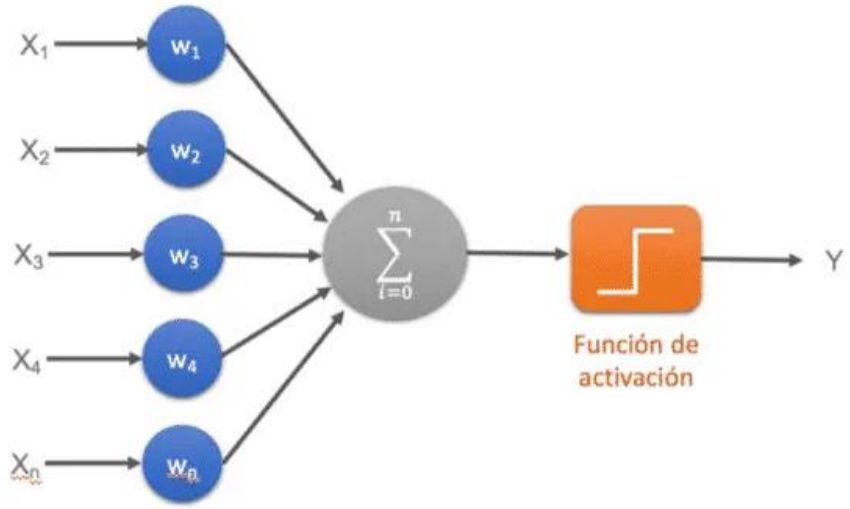


*Figura 6. Red perceptrón monocapa.*

Si comenzamos a examinar el espacio de hipótesis que un perceptrón puede representar con una función de activación umbral, el perceptrón puede representar una función booleana. Además de las funciones booleanas elementales **AND, OR, y NOT**, un perceptrón puede representar algunas funciones booleanas un poco más *complejas* de forma compacta. Por ejemplo, la **función mayoría**, cuya salida es 1 sólo si más de la mitad de sus *n* entradas están en 1, puede representarse con un perceptrón con peso $W_j = 1$ y umbral $W_0 = \frac{n}{2}$. Un árbol de decisión necesitaría $O(2^n)$ nodos para representar esta función.

Desafortunadamente, existen muchas funciones booleanas que el perceptrón umbral no puede representar. Dada la ecuación:

$$\sum_{j=0}^{n} w_j x_j > 0 \quad o \quad W.X > 0$$

EQ.3.9

Observamos que el perceptrón umbral devuelve 1 si y sólo si la suma ponderada de sus entradas (incluyendo los sesgos) es positiva (Russell y Norvig, 2010).

Dado que la ecuación **W.X** = 0 define un *hiperplano* en el espacio de entrada, el perceptrón devuelve 1 *si y sólo si* la entrada está en un lado de ese hiperplano. Por esta razón, el perceptrón umbral se denomina **separador lineal**. La Fig. 7 muestra el hiperplano (una recta, en dos dimensiones) para la representación mediante un perceptrón de las funciones **AND y OR** de dos entradas. Los puntos verdes indican un punto del espacio de entrada donde el valor de la función es 1, y los puntos rojos indican un punto donde el valor es 0. El perceptrón puede representar estas funciones porque existe una recta que separa todos los puntos blancos de todos los puntos negros. A estas funciones se las denomina **linealmente separables**. La Fig. 8 muestra un ejemplo de una función que *no* es linealmente separable. Claramente para la función **XOR** no hay manera de que el perceptrón umbral aprenda esta función. En general, *los perceptrones umbral pueden representar sólo funciones linealmente separables*. Dado que el modelo lineal no es capaz de representar la función **XOR**, una forma de resolver este problema es usar un modelo que aprenda un espacio de características diferente en el que un modelo lineal pueda representar la solución (Goodfellow et al., 2016).

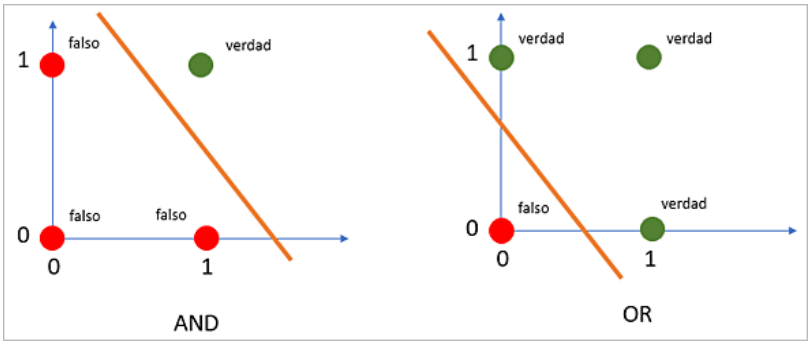


*Figura 7.* Funciones linealmente separables

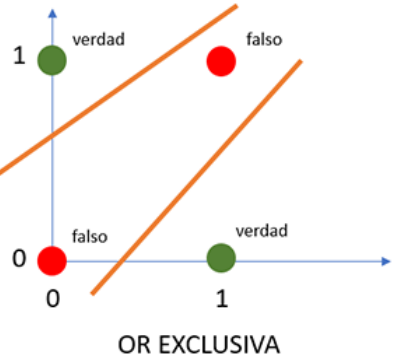


*Figura 8.* Función XOR linealmente NO separable

La idea del algoritmo, y en realidad de muchos algoritmos para aprendizaje de redes neuronales, es ajustar los pesos de la red para minimizar alguna medida del error que se produce con el conjunto de entrenamiento. Así, el aprendizaje se formula como una búsqueda optimizada en el espacio de pesos y sesgos. La medida *clásica* del error es la suma de los errores cuadrados, que se utiliza para regresión lineal.
El error cuadrado para un único ejemplo de entrenamiento con entrada **x** y valor verdadero de la salida ***y*** es

$$E = \frac{1}{2}\, Err^2 \equiv \frac{1}{2}\,(y - h_w(x))^2$$

EQ.3.10

donde $\boldsymbol{h}_{\mathbf{w}}(\mathbf{x})$ es la salida del perceptrón para el ejemplo e ***y*** es el valor real de la salida. Podemos usar el método del *descenso del gradiente* para reducir el error cuadrado calculando la derivada parcial de *E* con respecto a cada peso. Tenemos

$$\begin{aligned}\frac{\partial E}{\partial w_j} &= Err \,.\frac{\partial Err}{\partial w_j} \\ &= Err \,.\frac{\partial}{\partial w_j}\, g(y - \textstyle\sum_{j=0}^{n} w_j x_{j)} \\ &= -Err \,.\, g'(in) \,.\, x_j\end{aligned}$$

EQ.3.11

donde *g*’ es la derivada de la función de activación. En el algoritmo del descenso del

gradiente, para *reducir E*, actualizamos los pesos de la siguiente manera:

$$w_j \leftarrow \mathrm{w}_j + \alpha \,.\, Err \,.\, g'(in) \,.\, x_j$$

EQ.3.12

donde α es la **tasa de aprendizaje**. Intuitivamente, esto tiene mucho sentido. Si el error ***Err* = *y* – $h_{\mathbf{w}}$(x)** es positivo, la salida de la red es demasiado pequeña y por ello los pesos se *incrementan* para las entradas positivas y se *decrementan* para las entradas negativas. Cuando el error es negativo, ocurre lo contrario (Russell y Norvig, 2010).
El algoritmo introduce en la red los ejemplos de entrenamiento uno a uno, ajustando los pesos un poco después de cada ejemplo para reducir el error. Cada ciclo de cálculo incluyendo todos los ejemplos se denomina **época**, es decir, una pasada completa por todos los datos de entrenamiento. Las épocas se repiten hasta que se alcanza algún criterio de parada (típicamente, que los cambios de los pesos sean muy pequeños). Otros métodos calculan el gradiente para el conjunto total de entrenamiento añadiendo todas las contribuciones del gradiente en la ecuación 3.12 antes de actualizar los pesos. El método del **gradiente estocástico** selecciona ejemplos aleatoriamente del conjunto de entrenamiento en vez de hacer ciclos con ellos.

Hasta ahora se han tratado a los perceptrones como funciones determinísticas con salidas erróneas posibles, pero también es posible interpretar la salida de un perceptrón sigmoide como una probabilidad, específicamente, la probabilidad de que la salida verdadera sea 1, en función de las entradas. Con esta interpretación, se puede usar la sigmoide como una representación canónica para distribuciones condicionadas en redes bayesianas. También se puede derivar una regla de aprendizaje usando el método estándar de maximización del logaritmo de la verosimilitud de los datos (Russell y Norvig, 2010). Para esto se considera un único ejemplo de entrenamiento con valor de salida verdadera **T**, y sea ***P*** la probabilidad que retorna el perceptrón para este ejemplo. Si **T = 1**, la probabilidad condicional del dato es ***P***, y si **T = 0**, la probabilidad condicional del dato es (1 - *p*). Ahora se puede escribir el logaritmo de la verosimilitud de una forma que sea diferenciable. El truco es que una variable 0/1 en el exponente de una expresión actúa como una variable indicadora: $P^{\mathrm{T}}$ es *P* si T = 1, y 1 en otro caso. De forma análoga $(1 - \mathrm{P})^{(1-\mathrm{T})}$ es (1 – P) si T=0 y 1 en caso contrario. Por ello, se puede escribir el logaritmo de la verosimilitud del dato como:

$$L = \log p^{T} (1- p)^{(1-T)} = T \log p + (1- T) \log(1- p)$$

EQ 3.13

Gracias a las propiedades de la función sigmoide, el gradiente se reduce a una fórmula muy sencilla:

$$\frac{\partial L}{\partial W_j} = Err \,.\, \alpha_j$$

EQ 3.14

Nótese que *el vector de actualización de los pesos para aprendizaje por máxima verosimilitud en los perceptrones sigmoides es esencialmente idéntico al vector de*

*actualización para minimización del error cuadrado*. Por ello, podemos decir que los perceptrones tienen una interpretación probabilística incluso cuando la regla de aprendizaje se obtiene desde un punto de vista determinístico (Goodfellow et al., 2016).

## 3.8. Redes multicapa

Las redes multicapa son aquellas que disponen de conjuntos de neuronas agrupadas en varios (2, 3, …, N) niveles o capas. La ventaja de añadir capas ocultas es que se amplía el espacio de hipótesis que puede representar la red (Russell y Norvig, 2010).
En estos casos, un mecanismo para distinguir la capa a la que pertenece cada neurona consistiría en observar el origen de las señales que recibe a la entrada y el destino de la señal de salida. Normalmente, todas las neuronas de una capa reciben señales de entrada de otra capa anterior más cercana a las entradas de la red, y envían las señales de salida a una capa posterior, más cercana a las salidas de la red. A estas conexiones se las denomina conexiones hacia delante o *feedforward* (Britos, 2005).
Sin embargo, en un gran número de estas redes existe también la posibilidad de conectar las salidas de las neuronas de capas posteriores a las entradas de las capas anteriores, a estas conexiones se las denomina conexiones hacia atrás o *feedback*.
Estas dos posibilidades permiten distinguir entre dos tipos de redes con múltiples capas: las redes con conexiones hacia adelante o redes *feedforward* y las redes que disponen de conexiones tanto hacia adelante como hacia atrás o redes *feedforward/feedback*.
Los algoritmos de aprendizaje para redes multicapa son similares al algoritmo de aprendizaje del perceptrón simple. Una pequeña diferencia es que podemos tener varias salidas, con lo cual se obtiene un vector de salida $\mathbf{h_W(x)}$ en vez de un único valor, y cada ejemplo tiene un vector de salida ***Y***. La principal diferencia es que, mientras que el error $\mathbf{y - h_W}$ en la capa de salida es claro, el error en las capas ocultas no se conoce, porque los datos de entrenamiento no dicen cuál es el valor que han de tomar los nodos ocultos. Resulta que es posible propagar hacia atrás el error desde la capa de salida a las capas ocultas. El proceso de ***propagación hacia atrás*** proviene directamente a partir del gradiente del error total. En la capa de salida, la regla de actualización de pesos es idéntica a la Ecuación (3.12). Se obtendrán múltiples unidades de salida, así será $Err_i$ la *i-ésima* componente del vector de error $\mathbf{y - h_W}$.(Russell y Norvig, 2010)

## 3.9. Redes con conexión hacia adelante

Las redes *feedforward*, también llamadas redes neuronales de avance, o perceptrones multicapa (MLP), son los modelos de aprendizaje profundo por excelencia.
El objetivo de una red *feedforward* es aproximar alguna función $f^*$. Por ejemplo, para un clasificador, $y = f(x)$ asigna una entrada ***x*** a una categoría ***y***. Una red *feedforward* define un mapeo $y = f(x; \theta)$ y aprende el valor de los parámetros $\theta$ que dan como resultado la mejor aproximación de la función.
Estos modelos se denominan *feedforward* porque la información fluye a través de la función que se evalúa desde **x**, a través de los cálculos intermedios utilizados para definir **f**, y finalmente a la salida **y**. No hay conexiones de retroalimentación en las que las salidas del modelo se retroalimenten a sí mismo. Cuando las redes neuronales *feedforward* se amplían para incluir conexiones de retroalimentación, se denominan *redes neuronales recurrentes* (Goodfellow et al., 2016).

En las redes *feedforward* todas las señales neuronales se propagan hacia delante a través de las capas de la red. No existen conexiones hacia atrás (ninguna salida de neuronas de la capa i se aplica a la entrada de neuronas de capas i-1, i-2,…), y normalmente tampoco auto-recurrentes (salida de una neurona conectada a su propia entrada), ni laterales (salida de una neurona conectada a entradas de neuronas de la misma capa), excepto en el caso de los modelos de red propuestos por Kohonen denominadas *Learning vector quantizer (LVQ)* y *Topology Preserving Map (TPM)* , en las que existen unas conexiones implícitas muy particulares entre las neuronas de la capa de salida.
Si analizamos más en detalle la afirmación de que una red con *alimentación-hacia-adelante* representa una función de sus entradas, y consideramos la red sencilla mostrada en la Fig. 9, que tiene dos unidades de entrada, dos **unidades ocultas** y una unidad de salida. (Para mantener simplicidad, se han omitido las unidades de sesgo en el ejemplo.) Dado un vector de entrada **x** = (*x*1, *x*2), las activaciones de las unidades de entrada se ponen a (*a*1, *a*2) = (*x*1, *x*2) y la red calcula:

$$a_5 = g(W_{3,5}a_3 + W_{4,5}a_4) = g(W_{3,5}\, g(W_{1,3}a_1 + W_{2,3}a_2) + W_{4,5}\, g(W_{1,4}a_1 + W_{2,4}a_2))$$

Expresando la salida de cada unidad oculta como una función de sus entradas, se demuestra la salida de la red como una suma, $a_5$, en función de las entradas de la red.

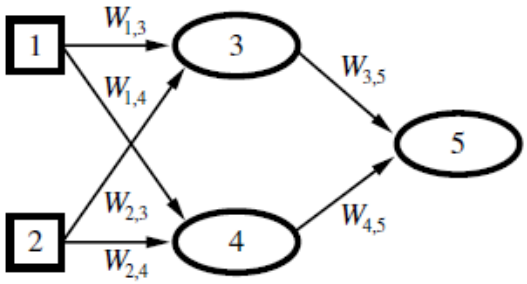


*Figura 9.* Una red neuronal muy simple con dos entradas, una capa oculta de dos neuronas y una salida.(Russell y Norvig 2010)

En lugar de pensar en cada capa de la red como si representara una sola función de *vector-a-vector*, también podemos pensar que una capa dada consta de muchas unidades que actúan en paralelo, cada una de las cuales representa una función de *vector-a-escalar*. Cada unidad se asemeja a una neurona en el sentido de que recibe información de muchas otras unidades y calcula su propio valor de activación. La idea de usar muchas capas de representaciones con valores vectoriales se extrae de la neurociencia. La elección de las funciones $f^i(x)$ usadas para computar estas representaciones también está vagamente guiada por observaciones neurocientíficas sobre las funciones que computan las neuronas biológicas. Sin embargo, la investigación moderna de redes neuronales está guiada por muchas disciplinas matemáticas y de ingeniería, y el objetivo de las redes neuronales no es modelar perfectamente el cerebro humano. En tal sentido, es preferible pensar en las redes *feedforward* como máquinas de aproximación de funciones que *están diseñadas para lograr la generalización estadística*, extrayendo ocasionalmente algunas ideas de lo que sabemos sobre el cerebro, en lugar de modelos de la función cerebral.

Las redes *feedforward* son un trampolín conceptual en el camino hacia las redes recurrentes, que impulsan muchas aplicaciones de procesamiento del lenguaje natural (NLP). Este tipo de redes se denominan *redes* porque normalmente se representan componiendo muchas funciones diferentes. El modelo está asociado con un gráfico acíclico dirigido que describe cómo se componen las funciones juntas (Goodfellow et al., 2016).
Durante el entrenamiento de la red neuronal, manejamos $f(x)$ para que coincida con $f^*(x)$. Los datos de entrenamiento nos proporcionan ejemplos aproximados y ruidosos de $f^*(x)$ evaluados en diferentes puntos del entrenamiento. Cada ejemplo $x$ va acompañado de una etiqueta $y \approx f^*(x)$. Los ejemplos de entrenamiento especifican directamente lo que debe hacer la capa de salida en cada punto $x$; debe producir un valor cercano a $y$. Los datos de entrenamiento no especifican directamente el comportamiento de las otras capas. El algoritmo de aprendizaje debe decidir cómo usar esas capas para producir el resultado deseado, pero los datos de entrenamiento no dicen qué debe hacer cada capa individual. En cambio, el algoritmo de aprendizaje debe decidir cómo usar estas capas para implementar mejor una aproximación de $f^*$. Debido a que los datos de entrenamiento no muestran el resultado deseado para cada una de estas capas, se denominan **capas ocultas** (Goodfellow et al., 2016).
Cada capa oculta de la red suele tener un valor vectorial y la dimensionalidad de estas capas ocultas determina el *ancho* del modelo. Puede interpretarse que cada elemento del vector desempeña un papel análogo al de una neurona (Goodfellow et al., 2016).

## 3.10. Redes con conexión hacia adelante y hacia atrás

En este tipo de redes circula información tanto hacia delante (*forward*) como hacia atrás (*backward*) durante el entrenamiento de la red. Para que esto sea posible existen conexiones ***feedforward*** y conexiones ***feedback*** entre las neuronas.
En general, estas redes suelen ser bicapa existiendo por lo tanto dos conjuntos de pesos: los correspondientes a las conexiones *feedforward* de la primera capa (capa de entrada) hacia la segunda (capa de salida) y los de las conexiones *feedback* de la segunda a la primera. Los valores de los pesos de estos tipos de conexiones no tienen por qué coincidir, siendo diferentes en la mayor parte de los casos.
Este tipo de estructura (bicapa) es particularmente adecuada para realizar una asociación de una información o patrón de entrada (en la primera capa) con otra información o patrón de salida en la segunda capa (hetero-asociación), aunque también pueden ser utilizadas para la clasificación de patrones.
Algunas redes de este tipo tienen un funcionamiento basado en lo que se conoce como *resonancia*, de tal forma que las informaciones en la primera y segunda capas interactúan entre sí hasta que alcanzan un estado estable. Este funcionamiento permite un mejor acceso a las informaciones almacenadas en la red.
Algunas de las configuraciones en este tipo de redes presentan conexiones laterales entre neuronas de la misma capa. Estas conexiones se diseñan como conexiones excitadoras (con peso positivo), permitiendo la cooperación entre neuronas, o como inhibidoras (peso negativo), estableciéndose una competición entre las neuronas correspondientes.
Finalmente, algunas topologías dentro de este tipo de redes *feedforward/feedback* multicapa, las neuronas se disponen en planos superpuestos (capas bidimensionales), lo

cual permite que puedan eliminarse las variaciones geométricas (tamaños, giros, desplazamientos) o distorsiones que presenten las informaciones o patrones de entrada a la red (Britos, 2005).

Cuando usamos una red neuronal *feedforward* que acepta una entrada ***x*** y produce una salida $\bar{\mathbf{Y}}$, la información fluye hacia adelante a través de la red. La entrada ***x*** proporciona la información inicial que luego se propaga hasta las unidades ocultas en cada capa y finalmente produce $\bar{\mathbf{Y}}$. Esto se llama propagación directa. Durante el entrenamiento, la propagación directa puede continuar hasta que produzca un costo escalar *J(Ø).*

### 3.10.1. Back-propagation

El algoritmo de ***back-propagation*** (Hinton y Sejnowski, 1986) , a menudo llamado simplemente ***backprop***, permite que la información de costo fluya hacia atrás a través de la red para calcular el gradiente. Calcular una expresión analítica para el gradiente es sencillo, pero evaluar numéricamente dicha expresión puede ser computacionalmente costoso. El algoritmo de *back-propagation* lo hace utilizando un procedimiento simple y económico.

El término *back-propagation* a menudo se malinterpreta como el algoritmo de aprendizaje completo para redes neuronales multicapa. En realidad, la propagación hacia atrás se refiere solo al *método para calcular el gradiente*, mientras que otro algoritmo, como el descenso de gradiente estocástico, se usa para realizar el aprendizaje usando este gradiente.
También, *back-propagation* a menudo se malinterpreta como una característica específica de las redes neuronales multicapa, pero en realidad puede ser aplicado para calcular derivadas de cualquier función (para algunas funciones, la respuesta correcta es informar que la derivada de la función no está definida).

De forma simplificada, el funcionamiento de una red que implementa *back-propagation* consiste en el aprendizaje de un conjunto predefinido de pares *entrada-salida* dados como ejemplo, empleando un ciclo de *propagación-adaptacion* de dos fases: primero se aplica un patrón de entrada en la capa de entrada de la red, el cual se propaga hacia las capas superiores hasta generar una salida, se compara el resultado obtenido en cada neurona de salida con el valor deseado para esa neurona y se obtiene un error para dicha unidad. A continuación, estos errores se transmiten hacia atrás hacia todas las neuronas de las capas intermedias que contribuyan directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia en la salida original. Este proceso se repite capa por capa hasta llegar a la entrada y hasta que cada neurona haya recibido un error que describa su aporte al error total. Según el valor del error recibido, se reajustan los pesos y sesgos de las conexiones entre cada par de neuronas en la red, de modo que el error total cometido para ese patrón disminuya.

Muchas tareas de aprendizaje automático implican calcular otras derivadas, ya sea como parte del proceso de aprendizaje o para analizar el modelo aprendido. El algoritmo de *back-propagation* también se puede aplicar a estas tareas y no se limita a calcular el gradiente de la función de costo con respecto a los parámetros (Goodfellow et al., 2016).

La idea de calcular derivadas propagando información a través de una red es muy general y se puede usar para calcular valores como el jacobiano de una función ***f*** con múltiples salidas.
Dentro de los algoritmos de aprendizaje, el gradiente que más a menudo se utiliza es el gradiente de la función de costo con respecto a los parámetros,$\nabla_0 J(\emptyset)$, donde el cálculo del gradiente $\nabla_x f(x, y)$ para una función arbitraria ***f***, donde ***x*** es un conjunto de variables cuyas derivadas se desean conocer, e ***y*** es un conjunto adicional de variables que son entradas de la función, pero cuyas derivadas no son requeridas.

### 3.10.2. Cálculo de la regla de la cadena

La regla de la cadena en cálculo (que no debe confundirse con la regla de la cadena de la probabilidad) se utiliza para calcular las derivadas de funciones formadas al componer otras funciones cuyas derivadas se conocen. *Back-propagation* es un algoritmo que calcula la regla de la cadena, con un orden específico de operaciones que es altamente eficiente.
Sea ***x*** un número real, y sean ***f*** y ***g*** funciones que mapean de un número real a un número real, y suponiendo que $y = g(x)$ y $\mathrm{z} = \mathrm{f}(\mathrm{g}(\mathrm{x})) = \mathrm{f}(\mathrm{y})$, entonces la regla de la cadena establece que:

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

EQ 3.15

Podemos generalizar esto más allá del caso escalar. Suponiendo que $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$, $g$ mapea de $\mathbb{R}^m$ a $\mathbb{R}^n$, y $f$ mapea de $\mathbb{R}^n$ a $\mathbb{R}$. Si $y = g(x)$ y $\mathrm{z} = \mathrm{f}(\mathrm{y})$, entonces:

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}$$

EQ 3.16

Una equivalencia en notación vectorial seria como sigue:

$$\nabla_{x^z} = \left(\frac{\partial y}{\partial x}\right)^T \nabla_{y^z}$$

EQ 3.17

Donde $\left(\frac{\partial y}{\partial x}\right)$ es la matriz jacobiana n * m de $g$.

De esto vemos que el gradiente de una variable ***x*** se puede obtener multiplicando una matriz jacobiana $\left(\frac{\partial y}{\partial x}\right)$ por un gradiente $\nabla_{y^z}$.
El algoritmo de *back-propagation* consiste en realizar dicho producto de gradiente jacobiano para cada operación en el gráfico. (Goodfellow et al., 2016).
Por lo general, se aplica el algoritmo de *back-propagation* a tensores de dimensionalidad arbitraria, no simplemente a vectores. Conceptualmente, esto es exactamente lo mismo que *back-propagation* con vectores. La única diferencia es cómo se organizan los

números en una cuadrícula para formar un tensor. Se podría imaginar aplanar cada tensor en un vector antes de ejecutar la propagación *back-propagation*, calcular un gradiente de valor vectorial y luego remodelar el gradiente nuevamente en un tensor. En esta vista reorganizada, *back-propagation* sigue multiplicando jacobianos por gradientes (Goodfellow et al., 2016).
Para denotar el gradiente de un valor ***z*** con respecto a un tensor **X**, se escribe $\nabla_{X^z}$, como si **X** fuera un vector. Los índices en **X** ahora tienen varias coordenadas; por ejemplo, un tensor 3D está indexado por tres coordenadas. Se podría abstraer esto usando una sola variable ***i*** para representar la tupla completa de índices. Para todas las tuplas de índice posibles ***i***, $(\nabla_{X^z})i$ da $\frac{\partial y}{\partial X_i}$. Esto es exactamente lo mismo que para todos los índices enteros posibles ***i*** en un vector, $(\nabla_{X^z})i$ resulta en $\frac{\partial y}{\partial x_i}$. Usando esta notación, podemos escribir la regla de la cadena tal como se aplica a los tensores. Si $Y = g(x)$ y $z = f(Y)$ entonces:

$$\nabla_{X^z} = \sum_j \left(\nabla_X Y_j\right) \frac{\partial z}{\partial Y_j}$$

EQ 3.18

### 3.10.3. Aplicación recursiva de la regla de la cadena para obtener *backprop*

Usando la regla de la cadena, es sencillo escribir una expresión algebraica para el gradiente de un escalar con respecto a cualquier nodo en el gráfico computacional que produjo ese escalar. Sin embargo, evaluar realmente esa expresión en una computadora introduce algunas consideraciones adicionales.
Específicamente, muchas subexpresiones pueden repetirse varias veces dentro de la expresión general del gradiente. Cualquier procedimiento que calcule el gradiente deberá elegir si almacenar estas subexpresiones o volver a calcularlas varias veces.
Primero consideremos un gráfico computacional que describa cómo calcular un solo escalar $u^{(n)}$ (la pérdida en un ejemplo de entrenamiento). Este escalar representa la cantidad cuyo gradiente que se desea obtener, con respecto a los $n_i$ nodos de entrada $u^{(1)}$ a $u^{(n_i)}$. En otras palabras, se quiere calcular $\frac{\partial u^{(n)}}{\partial u^{(i)}}$ para todo $i \in \{1,2,\dots,n_i\}$. En la aplicación de *back-propagation* para calcular gradientes para descenso de gradiente sobre parámetros, $u^{(n)}$ será el costo asociado con un ejemplo o un minilote, mientras que $u^{(1)}$ a $u^{(n_i)}$ corresponden a los parámetros del modelo.

Este algoritmo especifica el cálculo de la propagación hacia adelante, que se podría poner en un grafo ***G***. Para realizar la propagación hacia atrás, podemos construir un grafo computacional que dependa de ***G*** y le agregue un conjunto adicional de nodos. Estos forman un subgrafo ***B*** con un nodo por nodo de ***G***. El cómputo en ***B*** procede exactamente en el orden inverso del cómputo en ***G***, y cada nodo de ***B*** calcula la derivada $\frac{\partial u^{(n)}}{\partial u^{(i)}}$ asociado con el nodo de grafo directo $u^{(i)}$. Esto se hace usando la regla de la cadena con respecto a la salida escalar $u^{(n)}$:

$$\frac{\partial u^{(n)}}{\partial u^{(j)}} = \sum_{i:j \in Pa(u^{(i)})} \frac{\partial u^{(n)}}{\partial u^{(i)}} \frac{\partial u^{(i)}}{\partial u^{(j)}}$$

EQ 3.19

El algoritmo de *back-propagation* está diseñado para reducir el número de *sub-expresiones* comunes sin tener en cuenta la memoria. Específicamente, funciona en el orden de un producto jacobiano por nodo en el grafo.

Esto puede verse en el hecho de que *backprop* visita cada borde desde el nodo $u^{(j)}$ hasta el nodo $u^{(i)}$ del gráfico exactamente una vez para obtener la derivada parcial asociada $\frac{\partial u^{(i)}}{\partial u^{(j)}}$ . *Back-propagation* evita así la explosión exponencial en *sub-expresiones* repetidas (Goodfellow et al., 2016).

## 3.11. Mecanismo de aprendizaje

El aprendizaje es el proceso por el cual una red neuronal modifica sus pesos y sesgos en respuesta a una información de entrada. Los cambios que se producen durante el proceso de aprendizaje se reducen a la destrucción, modificación y creación de conexiones entre las neuronas. En los sistemas biológicos existe una continua creación y destrucción de conexiones. En los modelos de redes neuronales artificiales la creación de una nueva conexión implica que el peso de esta pasa a tener un valor distinto de 0. De la misma forma, una conexión se destruye cuando su peso pasa a ser 0 (Britos, 2005). Diseñar y entrenar una red neuronal no es muy diferente de entrenar cualquier otro modelo de aprendizaje automático con descenso de gradiente. La mayor diferencia entre los modelos lineales que se han presentado hasta ahora y las redes neuronales, es que la no linealidad de una red neuronal hace que las funciones de pérdida más interesantes se vuelvan no convexas. Esto significa que las redes neuronales generalmente se entrenan mediante el uso de optimizadores iterativos basados en gradientes que simplemente llevan la función de costo a un valor muy bajo, a diferencia de los solucionadores de ecuaciones lineales, que se usan para entrenar modelos lineales de regresión o algoritmos de optimización convexos con garantías de convergencia global utilizados para entrenar regresión logística o SVMs (Goodfellow et al., 2016).
La optimización convexa converge a partir de cualquier parámetro inicial (en teoría, en la práctica es robusta, pero puede encontrar problemas numéricos). El descenso de gradiente estocástico aplicado a funciones de pérdida no convexas no tiene tal garantía de convergencia y es sensible a los valores de los parámetros iniciales.
Para redes neuronales *feedforward*, es importante inicializar todos los pesos a valores aleatorios pequeños. Los sesgos pueden inicializarse a cero o a pequeños valores positivos. Durante el proceso de aprendizaje, los pesos de las conexiones de la red sufren modificaciones, por lo tanto, se puede afirmar que este proceso ha terminado (la red ha aprendido) *cuando los valores de los pesos permanecen estables* $\frac{dwi}{dt} = 0$.
Un aspecto importante respecto al aprendizaje en las redes neuronales es el de conocer cómo se modifican los valores de los pesos, es decir, cuáles son los criterios que se

siguen para cambiar los valores asignados a las conexiones cuando se pretende que la red aprenda una nueva información.
Estos criterios determinan lo que se conoce como *la regla de aprendizaje de la red*. De forma general se suelen considerar dos tipos de reglas: *las que responden al aprendizaje supervisado, y las correspondientes al aprendizaje no supervisado*.
La diferencia fundamental entre ambos tipos radica en la existencia o no de un agente externo (*supervisor*) que controle el proceso de aprendizaje de la red.
Otro criterio que se puede utilizar para diferenciar las reglas de aprendizaje se basa en considerar si la red puede aprender durante su funcionamiento habitual o si el aprendizaje supone la desconexión de la red; es decir, su des habilitación hasta que el proceso finalice. En el primer caso se trataría de un aprendizaje off line.
Cuando el aprendizaje es off line, se distingue entre una fase de aprendizaje o entrenamiento y una fase de operación o funcionamiento, existiendo un conjunto de datos de entrenamiento y otro de prueba que serán utilizados en la fase correspondiente. En las redes con aprendizaje off line, los pesos de las conexiones permanecen fijos después de que termina la etapa de entrenamiento de la red. Debido precisamente a su carácter estático, estos sistemas no presentan problemas de estabilidad en su funcionamiento.
En las redes con aprendizaje *on-line* no se distingue entre fase de entrenamiento y de operación, de tal forma que los pesos varían dinámicamente siempre que se presente una nueva información al sistema. En este tipo de redes, debido al carácter dinámico de las mismas, el estudio de la estabilidad suele ser un aspecto fundamental (Britos, 2005).
Por supuesto, también podemos entrenar modelos como regresión lineal y máquinas de vectores de soporte con *descenso de gradiente* y, de hecho, esto es común cuando el conjunto de entrenamiento es extremadamente grande. Desde este punto de vista, entrenar una red neuronal no es muy diferente de entrenar cualquier otro modelo. Calcular el gradiente es un poco más complicado para una red neuronal, pero aún se puede hacer de manera eficiente y exacta (Goodfellow et al., 2016).

### 3.11.1. Redes con aprendizaje supervisado

El aprendizaje supervisado se caracteriza por un entrenamiento controlado por un agente externo (*supervisor*) que determina la respuesta que debería generar la red a partir de una entrada determinada. El supervisor comprueba la salida de la red y en el caso de que esta no coincida con la deseada, se procederá a modificar los pesos de las conexiones, con el fin de conseguir que la salida obtenida se aproxime a la deseada.
En este tipo de aprendizaje se suelen considerar, a su vez, tres formas de llevarlo a cabo que dan lugar a los siguientes aprendizajes supervisados:

**Aprendizaje por corrección de error**: Consiste en ajustar los pesos de la red en función de la diferencia entre los valores deseados y los obtenidos en la salida de la red; es decir, en función del error cometido en la salida.

- $\Delta w_{ij}$: variación en el peso de la conexión entre las neuronas i y j ($Dw_{ij} = w^{ijactual-} w^{ijanterior}$).
- $Y_j$= Valor de salida de la neurona i
- $d_j$= Valor de salida deseado para la neurona j

- $Y_j$= Valor de salida obtenido en la neurona i
- α= Factor de aprendizaje (*learning rate*) (0<α<1), el cual regula la velocidad de aprendizaje.

Una deficiencia que presenta este método de aprendizaje es que no considera la magnitud del error global cometido durante el proceso completo de aprendizaje de la red, considerando únicamente los errores individuales (locales) correspondientes al aprendizaje de cada información por separado (Britos, 2005).

**Aprendizaje por refuerzo**: Este algoritmo de aprendizaje supervisado, se basa en la idea de no disponer de un ejemplo completo del comportamiento deseado; es decir, durante el entrenamiento no se indica exactamente cual salida es la esperada que la red proporcione ante un determinado input.

La función de supervisión se reduce a indicar mediante una señal de refuerzo si la salida obtenida se ajusta a la deseada (éxito=+1 o fracaso=-1), entonces en función de ello se ajustan los pesos, basándose en un mecanismo de probabilidades.

**Aprendizaje estocástico.** Este tipo de aprendizaje consiste básicamente en realizar cambios aleatorios en los valores de los pesos de las conexiones de la red, y evaluar su efecto a partir del objetivo deseado y de distribuciones de probabilidad.
Según lo anterior, si luego de introducir un cambio aleatorio de los valores de los pesos, el comportamiento de la red se asemeja al deseado, entonces se acepta el cambio. De lo contrario, el cambio se acepta de acuerdo con una distribución de probabilidades.

### 3.11.2. Redes con aprendizaje no supervisado

Las redes con aprendizaje no supervisado no requieren influencia externa para ajustar los pesos de las conexiones entre sus neuronas. La red no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta. Debido a esto suele decirse que estas redes son capaces de autoorganizarse (Britos, 2005).
Este tipo de redes deben encontrar las características, regularidades, correlaciones o categorías que se puedan establecer entre los datos que se presentan en su entrada. Dado que no hay un *supervisor* que indique a la red la respuesta que debe generar ante una entrada concreta, cabría preguntarse qué es lo que la red genera en estos casos. Existen varias posibilidades en cuanto a la interpretación de la salida de estas redes, que dependen de su estructura y del algoritmo de aprendizaje utilizado.
En algunos casos, la salida representa el grado de familiaridad entre la formación que se le está presentando a la entrada y las informaciones que se le han mostrado hasta entonces. En otro caso, podría realizar una *clusterizacion*, indicando a la salida de la red, a que categoría pertenece la información presentada a la entrada, siendo la propia red la que debe encontrar las categorías apropiadas a partir de correlaciones entre las informaciones presentadas. Una variación de esta categorización es el prototipado. En este caso la red obtiene ejemplares o

prototipos representantes de las clases a las que pertenecen las informaciones de entrada.

En cuanto a los algoritmos de aprendizaje no supervisado, en general se suelen considerar dos tipos que dan lugar a los siguientes tipos de aprendizaje:

**Aprendizaje hebbiano**: Este tipo de aprendizaje, formulado por Donald Hebb en 1949, consiste básicamente en el ajuste de los pesos de las conexiones de acuerdo con la correlación (multiplicación en el caso de valores binarios +1 y -1), de los valores de activación (salidas) de las dos neuronas conectadas:

$$\Delta w_{ij} = y_i y_j$$

EQ 1.19

Esta expresión responde a la idea de Hebb, dado que si las dos unidades se activan (positivas), se produce un refuerzo de la conexión. Se trata de una regla de aprendizaje no supervisado, pues la modificación de los pesos se realiza en función de los estados (salidas) de las neuronas, obtenidos tras la presentación de cierto estimulo (entrada), sin tener en cuenta si se deseaba o no esos estados de activación.

**Aprendizaje competitivo y cooperativo:** En este tipo de aprendizaje, las neuronas compite (y cooperan) unas con otras con el objetivo de llevar a cabo una tarea dada. Con este tipo de aprendizaje, se pretende que cuando se presente a la red cierta información de entrada, solo una de las neuronas de salida de la red, o una para determinado grupo de neuronas, se active (alcanza su valor de respuesta máximo). Por lo tanto, las neuronas compiten por activarse quedando solamente una, o una por grupo, como neurona vencedora (*winner take all unit*). De este modo el resto de las neuronas quedan anuladas y son forzadas al valor de respuesta mínimo.
La competencia entre neuronas se realiza en todas las capas de la red, existiendo en estas conexiones recurrentes de autoexcitación y conexiones de inhibición (signo negativo) por parte de neuronas vecinas. Si el aprendizaje es cooperativo, estas conexiones con las vecinas serán de excitación (signo positivo).
El objetivo de este aprendizaje es categorizar (*clusterizar*) los datos de entrada a la red. De esta forma, las informaciones similares son clasificadas formando parte de la misma categoría, y por lo tanto deben activar la misma neurona de salida. Las clases o categorías deben ser creadas por la propia red, puesto que se trata de un aprendizaje no supervisado a través de las correlaciones de los datos de entrada (Britos, 2005).

## 3.12. Aprendizaje de la estructura de las redes neuronales

Hasta aquí, hemos considerado el problema del aprendizaje de los pesos y sesgos, dada una estructura fija de la red. Al igual que con las redes bayesianas, es también necesario comprender cómo encontrar la mejor estructura o topología de la red. Si se diseña una red que es demasiado grande, esta será capaz de memorizar todos los ejemplos formando una gran tabla de búsqueda, pero no generalizará necesariamente bien para

entradas que no se han visto anteriormente. En otras palabras, como todos los modelos estocásticos, las redes neuronales son sujeto de **sobreajuste** cuando hay demasiados parámetros en el modelo. Los modelos con muchos parámetros se ajustan en todos los datos, pero no generalizarían tan bien como los modelos con pocos parámetros.
Si nos centramos en redes totalmente conectadas, las únicas elecciones por las que nos podemos preocupar son el número de capas ocultas y su tamaño. El enfoque más usual es intentar varias y quedarnos con la mejor. Se necesitan las técnicas de **validación cruzada** si se desea evitar el ***peeking*** del mejor conjunto. Es decir, se elige la arquitectura de la red que proporciona la mayor precisión de predicción en los conjuntos de validación.
Si se desea considerar redes que no están conectadas en su totalidad, necesitamos encontrar algún método de búsqueda efectivo a través del gran espacio de topologías de posibles conexiones. El algoritmo del **daño cerebral óptimo** (*optimal brain damage*) comienza con una red totalmente conectada y va eliminando conexiones. Después de que la red está entrenada por primera vez, un enfoque teórico de información identifica una selección óptima de las conexiones que pueden ser eliminadas. La red vuelve a entrenarse, y si su rendimiento no se ve decrementado se repite el proceso. Además de la eliminación de conexiones, también es posible eliminar unidades que no contribuyen demasiado al resultado (Russell y Norvig, 2010).

# CAPITULO 4 Integración de la Inteligencia Artificial Simbolica y No Simbolica

## 4.1. Introducción

La Inteligencia Artificial se aborda desde dos enfoques predominantes pero muy diferentes entre sí: la Inteligencia Artificial simbólica, que está inspirada en la lógica matemática y se basa en la manipulación de representaciones lingüísticas abstractas, y la Inteligencia Artificial no simbólica, que se centra en la construcción de modelos matemáticos predictivos a partir de grandes conjuntos de datos de muestra.
Significativamente, las deficiencias de cada uno de estos enfoques se alinean con las fortalezas del otro, lo que sugiere que una integración entre ellos sería beneficiosa. Una síntesis satisfactoria de inteligencia artificial simbólica y no-simbólica nos daría las ventajas de ambos mundos.
Este trabajo tiene como objetivo identificar y clasificar soluciones y arquitecturas que utilizan técnicas de Inteligencia Artificial aplicada, basadas en la integración *de lógica simbólica y no-simbólica (en particular aprendizaje automático con redes neuronales artificiales),* para proporcionar una visión integral, exhaustiva y organizada de las soluciones disponibles en la literatura, haciéndolas objeto de una SLR: Systematic Literatura Review (Revisión Sistemática de la Literatura) cuidadosamente diseñada e implementada sobre el tema. Las tecnologías resultantes se discuten y evalúan desde ambas perspectivas: la Inteligencia Artificial simbólica y la no simbólica.
Se ha utilizado el método PICOC *(Población, Intervención, Comparación, Salidas, Contexto)* más Límites, que determinan el alcance de la búsqueda, para definir las preguntas de investigación y analizar los resultados. De un total de 65 estudios candidatos encontrados, se seleccionaron 24 artículos (37%) relevantes para este estudio. Cada estudio, además, se centra en diferentes dominios de aplicación tales como agentes inteligentes, clasificación de imágenes, comprobadores de teoremas, *cyber*-seguridad, interpretación de imágenes, matemática, medicina, robótica y de aplicación general.
A través del análisis de los trabajos seleccionados, se logró clasificar, organizar y explicar las diferentes formas en que las deficiencias de la Inteligencia Artificial no simbólica son abordadas por propuesta basadas en lógica simbólica. El estudio también determinó en qué etapas del proceso de desarrollo dichas propuestas son aplicadas. Complementariamente, el estudio permitió determinar cuáles son las herramientas de la lógica que se aplican preferentemente, para cada área y cada dominio. Si bien no se ha encontrado un *patrón arquitectónico* claro, los esfuerzos por encontrar un *modelo de propósito general* que combine ambos mundos dirigen las tendencias y esfuerzos de investigación.
En la presente revisión, se pretende entender y representar el estado actual de las tecnologías de *DL potenciadas con sistemas basados en lógica* que son realmente utilizables en la ingeniería de sistemas inteligentes. Por lo tanto, el objetivo de este trabajo es identificar y clasificar soluciones y arquitecturas que utilizan técnicas de Inteligencia Artificial aplicada, basadas en la integración *de la inteligencia artificial simbólica y no-simbólica (en particular aprendizaje automático con redes neuronales*

*artificiales),* para proporcionar una visión integral, exhaustiva y organizada de las soluciones disponibles en la literatura, haciéndolas objeto de una SLR cuidadosamente diseñada e implementada sobre el tema.

## 4.2. Método de Investigación

En esta sección se presenta la metodología de investigación utilizada, la cual sigue la estrategia propuesta por (Kitchenham et al., 2009). Siguiendo la estrategia propuesta, en subsecuentes subsecciones del protocolo de revisión, se discuten los criterios de inclusión y exclusión, el proceso de estrategia de búsqueda, el proceso de selección y los procesos de extracción y síntesis de datos.

En el resto del estudio se ha elegido incluir sólo tecnologías Deep Learning potenciadas con lógica formal de primer orden que sean científicamente identificables y técnicamente disponibles. Esto significa, en primer lugar, que debe existir al menos una publicación científica que defina y describa claramente la tecnología y, por lo tanto, cubra sus aspectos fundamentales; luego, que la tecnología en sí debe mencionarse claramente en la publicación referida, así como técnicamente accesible en el momento de la revisión.
Más precisamente, en este trabajo consideramos como tecnología Deep Learning basada en lógica a cualquier arquitectura, framework o lenguaje de software orientado a agentes que:

(i) Implique algún modelo lógico claramente definido.
(ii) Provea alguna reificación[1] tecnológica real.

De hecho, se argumenta que los aspectos tecnológicos están lejos de ser marginales, ya que la tecnología es la aplicación del conocimiento científico con fines prácticos, y ahora más que nunca la ingeniería de sistemas inteligentes precisa de manera urgente tecnologías confiables.
Claramente, las tecnologías pueden involucrar lógicas en varios niveles diferentes y de formas dispares, de la misma manera que pueden abordar diversas topologías de una red neuronal. Dentro del alcance de la revisión, en particular, estamos interesados principalmente en tecnologías que combinan redes neuronales profundas con sistemas basados en lógica de primer orden.
En La Fig. 10 se muestra el proceso general. Para la fase de planificación, se elaboró una descripción detallada del protocolo de investigación. Las siguientes subsecciones presentan la especificación del objetivo y las preguntas de investigación y detallan los pasos de la fase de ejecución. Durante la etapa de ejecución se implementa el proceso definido durante la fase de planificación. También cabe destacar que el proceso es iterativo y puede requerir revisiones (Kitchenham et al., 2009). Durante la etapa de reporte, se clasifican y describen los resultados obtenidos, se discuten los resultados y se presentan conclusiones.

[1] Técnica de programación orientada a objetos que consiste en tener un tipo de datos para una abstracción.

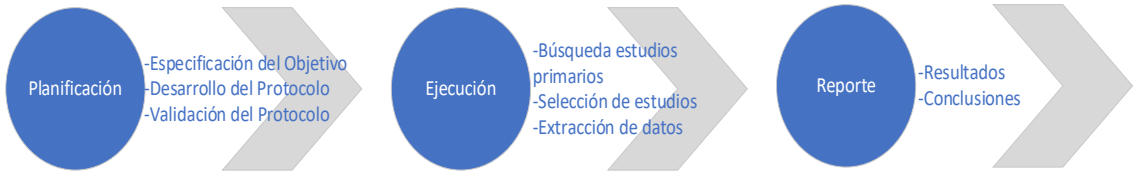


*Figura 10. Diagrama de fases y pasos de la investigación en este estudio*

### 4.2.2. Especificación del Objetivo

Para la presente revisión, el objetivo se estableció de la siguiente manera:
Identificar y clasificar soluciones y arquitecturas que utilizan tecnologías basadas en *Técnicas de IA que integran lógica simbólica y no-simbólica (en particular redes neuronales artificiales).*
A partir de este objetivo, se derivan un conjunto de preguntas de investigación (PI) y palabras clave (PC).

### 4.2.3. Preguntas de Investigación

Todas las SRL se basan en preguntas de investigación fundamentales. Una revisión sistemática intenta identificar, evaluar y sintetizar toda la evidencia empírica que cumple con los criterios de elegibilidad preespecificados para responder una pregunta de investigación determinada. Los investigadores que realizan revisiones sistemáticas utilizan métodos explícitos destinados a minimizar el sesgo, a fin de producir hallazgos más confiables que se puedan utilizar para informar la toma de decisiones.
En particular, la presente revisión sistemática tiene como objetivo responder las siguientes preguntas:

***PI 1. ¿Cuál de las deficiencias del DL es abordada/mejorada por la propuesta descripta en el artículo?***

***PI 2. ¿En qué parte del proceso de DL se aplica Lógica?***

***PI 3 ¿Que herramientas de la lógica se aplican? ¿Qué tipos de tareas se abordan? ¿Sobre cuáles dominios?***

Las preguntas anteriores dirigen el proceso de búsqueda, al articular la forma en que los artículos y las tecnologías se buscan, examinan y condicionalmente seleccionan para la revisión. Luego, se utilizó el método PICOC propuesto por Petticrew & Roberts, 2008 para definir el alcance de la revisión:

**Población (P)**: “Deep Learning”, “First Order Logic”, “Deep Neural Networks”, “Deductive reasoning”, “Relational thinking”, “Inference Engines”, “Meta-interpretive learning”, “Symbolic logic”.

**Intervención (I):** realizar una revisión sistemática de la literatura sobre arquitecturas de *software*, modelos simbólico-neuronal híbrido basado en DL y Lógica simbólica.
**Comparación (C):** sin comparación

**Salidas (Outputs - O):** precisión, interpretabilidad, tecnología, implementación, arquitectura, modelo.

**Contexto(C):** Contextos relacionados con tecnología aplicada e IA (industria, empresa, negocio, academia)

**Límites:** Artículo de Revista académica, Artículo en conferencia, Página de libro.

### 4.2.4. Fuentes

Las fuentes electrónicas utilizadas en el proceso de búsqueda son de dos tipos:

(i) Bases de datos bibliográficas generales
(ii) Fuentes específicas, como conferencias, simposios y talleres, sobre los temas centrales.

Los primeros se buscan de forma intensiva a través de consultas específicas, mientras que los segundos se exploran extensamente, artículo por artículo. Esto se debe a que a menudo faltan talleres pequeños y especializados en las principales bases de datos bibliográficas utilizadas para nuestra búsqueda basada en consultas; en cambio, normalmente se puede contar con la indexación completa de las principales conferencias y revistas internacionales.
En consecuencia, el escrutinio exhaustivo de todos los trabajos publicados en los talleres especializados sobre temas cercanos al *Deep Learning* y *lógica de primer orden* es mayoritariamente destinado a complementar la búsqueda intensiva realizada a través de los motores bibliográficos, de modo que resulte en la búsqueda más completa posible en general.

En particular, consideramos ampliamente los artículos publicados en simposios tales como:

- International Conference on Learning Representations
- International Joint Conference on Artificial Intelligence (IJCAI)
- International Conference on Machine Learning (ICML)
- AAAI Conference on Artificial Intelligence

En detalle, las fuentes utilizadas para la búsqueda intensiva (a través de consultas basadas en palabras clave) son las siguientes:

- Google Scholar http://scholar.google.com
- IEEE Xplore http://ieeexplore.ieee.org
- ScienceDirect http://www.sciencedirect.com
- SpringerLink https://link.springer.com
- ACM Digital Library http://dl.acm.org

Cada una de estas fuentes se consulta con las siguientes combinaciones de palabras clave:

**PC1. Todas las formas de decir Deep learning**
("Deep Learning" OR "Neural Networks")

**PC2: Todas las formas de decir lógica**
("First Order Logic" OR "Symbolic logic" OR "deductive reasoning" OR "Relational thinking" OR "inference engines" OR "Meta-interpretive learning" OR "Logic programming")

**PC3: Todas las Intervenciones**
("hybrid model" OR "explicabilidad")

**PC4: todas las salidas (definidos en la O del PICOC)**
("Accuracy" OR "interpretability" OR "technology" OR "implementation" OR "architecture" OR "model")

La consulta resultante vendría dada de la siguiente manera: PC1 AND PC2 AND PC3 AND PC4

Los primeros cincuenta resultados devueltos se consideran candidatos potenciales. Primero se verifica la ocurrencia de un enfoque basado en la *lógica de primer orden en DL* para cada artículo; Luego, en caso de que haya concordancia de términos, se busca y verifica la existencia de la tecnología en todo el documento.

### 4.2.5. Clasificación de artículos

Los artículos resultantes de la actividad de búsqueda anterior se clasifican de acuerdo con tres clases principales:

**Artículos primarios**: Aquellos artículos que presentan y describen una tecnología de interés.
**Artículos auxiliares**: Aquellos artículos que mencionan una tecnología de interés, lo que significa que la tecnología mencionada se ha empleado de alguna manera en el desarrollo que contribuye al artículo.
**Artículos secundarios:** Aquellos artículos que analizan tecnologías de interés.
Dada su relevancia obvia para una encuesta sistemática, los artículos secundarios no se utilizan sólo como fuente directa de tecnologías de interés; en cambio, se busca en sus referencias bibliográficas extensas y organizadas de forma recursiva en busca de posibles elementos de interés.
Este documento tiene como objetivo comprender y representar el estado actual de las tecnologías basadas en *lógica de primer orden disponibles para DL*; por lo tanto, esta revisión se centra en aquellas tecnologías que realmente se pueden utilizar en la ingeniería de sistemas inteligentes.

### 4.2.6 Selección de trabajos y criterios de exclusión

Primero eliminamos los duplicados en el conjunto de artículos recuperados. Esta tarea preliminar garantiza que no se analizará el mismo documento más de una vez. Después

de eso, todos los artículos se recopilaron en una sola hoja de cálculo de cada estrategia de búsqueda y se llevó a cabo un proceso de dos pasos:

**Paso 1**: los resultados se analizaron únicamente por título, palabras clave y lugar (revista o conferencia), para excluir artículos que cumplan algunos de los siguientes requisitos:

- No estén en Ingles.
- Publicaciones no revisadas por pares.
- No aparecen términos de búsqueda importantes en palabras clave.
- Libros, editoriales, tutoriales, paneles, prefacios, opiniones, cartas, diapositivas y cualquier obra que pueda considerarse literatura gris.
- Año de publicación anterior al año 2015 inclusive.
- Artículos duplicados.
- Artículos que no tienen acceso completo

*Fuera del alcance*: estos son trabajos irrelevantes que fueron recuperados debido a una mala ejecución de la búsqueda o debido a errores humanos en las búsquedas manuales. Consideramos que este criterio de exclusión es el más eficaz.
*Límites*: artículos que no pertenecen al dominio de IT / CS / SE / IS (Tecnología de la información / Ciencias de la computación / Ingeniería de *software* / Sistemas de información)
*Otros*: También se excluyeron las tesis de doctorado o maestría, bajo el supuesto de que publicaciones relevantes, resultantes de la investigación cubierta por las tesis, estaban disponibles e incluidos en el conjunto de artículos recuperados.

**Paso 2**: De los artículos del Paso 1 se evaluaron sus títulos, resúmenes y palabras clave donde se aplicaron los criterios de inclusión y exclusión.
Finalmente, cada artículo candidato fue leído en su totalidad para decidir si cumple un conjunto de características y criterios de calidad. Luego de la lectura completa de los trabajos, se identificaron otros trabajos relevantes. Concretamente, dos artículos, que también se leyeron en profundidad para responder las preguntas descritas anteriormente.

### 4.2.7 Extracción de datos

El proceso de extracción de datos es un proceso iterativo e incremental que se ha dividido en varias etapas en las que se han llevado a cabo diferentes actividades. Para describir el proceso, se utiliza un flujo, según se describe en (Moher et al., 2010)
Se ha desarrollado un Formulario de Extracción de Datos (FED); una hoja de cálculo para recopilar todos los datos extraídos de los trabajos primarios seleccionados. Se utilizó una plantilla para organizar los elementos de interés.
En primer lugar, se han identificado los resultados obtenidos tras aplicar las cadenas de búsqueda. Para esta tarea se utilizó Zotero® como herramienta de recolección y organización de trabajos. En segundo lugar, se analizó el título, el resumen y las palabras clave de cada trabajo, y se han aplicado los criterios de inclusión y exclusión. Los trabajos se organizaron en otra hoja dentro del mismo documento y cada trabajo se ha marcado como candidato o no, según los criterios de inclusión y exclusión. Finalmente, cada

trabajo candidato se leyó en detalle para decidir si son relevantes respecto de las preguntas de investigación.

## 4.3. Resultados de la Revisión Sistemática

En esta sección se presentan los datos obtenidos tras el proceso de extracción y el análisis de los trabajos seleccionados. Los resultados del análisis preliminar en la presente revisión revelaron 65 estudios para mayor consideración.
Los resultados obtenidos después de llevar a cabo este proceso se describen en la Fig. 11, y se detallan a continuación:

- Después de aplicar las cadenas de búsqueda en cada fuente, se han obtenido 103 artículos, de los cuales 2 son de ACM, 52 de GoogleScholar, 11 de IEEE, 19 de ScienceDirect, y 19 de SpringerLink.
- Después de eliminar los duplicados, quedaron 83 artículos.
- Una vez que se han aplicado los criterios de inclusión y exclusión al título, el resumen y las palabras clave, quedan 75 artículos.
- Finalmente, se han analizado en profundidad un total de 34 artículos que son los más relevantes para el presente estudio (33% de los trabajos únicos recuperados).

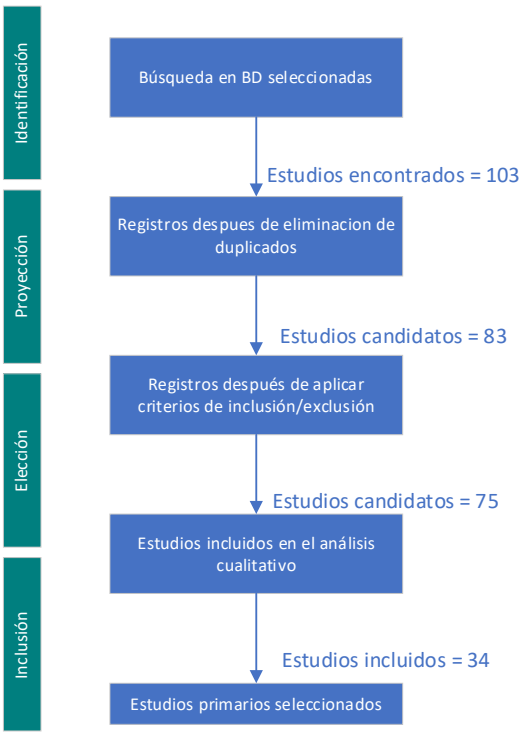


*Figura 11. Procedimiento de búsqueda de estudios primarios*

### 4.3.1 PI1. ¿Cuál de las deficiencias del DL es abordada/mejorada por la propuesta descrita en el artículo?

Esta pregunta de investigación se subclasifica y aborda basada en las diferentes deficiencias del DL. En tal sentido, se definen las siguientes las categorías:

- Ineficiencia de los datos
- Falta de generalización
- Falta de razonamiento
- Falta de explicabilidad o comprensión humana

#### *Ineficiencia de Datos*

Los modelos profundos a menudo requieren grandes cantidades de datos de entrenamiento, lo que puede limitar su aplicación en situaciones en las que los datos no están tan fácilmente disponibles. Basados en esta idea, se dan los primeros pasos hacia un enfoque práctico para entrenar modelos secuenciales profundos utilizando una combinación de datos y conocimiento temporal previo (Xie et al., 2021).
Para abordar este problema dentro de los sistema de DL, Melin et al., 2018 proponen un modelo que utiliza lógica difusa con el objetivo de crear aproximaciones matemáticas para resolver ciertos y determinados problemas. Los autores sostienen que este tipo de lógica produce resultados precisos a partir de datos imprecisos.

En la misma dirección Pal & Kar, 2019 han utilizado lógica difusa de tipo 2 general para obtener los pesos en la red neuronal y tratar de lidiar con la incertidumbre en los datos reales. En este trabajo se plantea que los sistemas de lógica difusa de orden superior son capaces de lidiar con una gran cantidad de incertidumbres en aplicaciones del mundo real.
Los autores plantean un modelo basado en *zSlices*, donde se aplican pesos difusos entre la capa de salida oculta y el error difuso en la capa de salida.
El concepto de *zSlices* basados en intervalos se ha utilizado para obtener los pesos difusos generales de tipo 2 para la arquitectura de una red neuronal.
Al cortar un conjunto difuso de tipo 2 general en la tercera dimensión, denotado como z en algún nivel, digamos $z_i$, obtenemos *zSlices*. Un *zSlice* $\check{Z}_i$ es un conjunto difuso de tipo 2 de intervalo con grado de membresía $z_i$ dentro del rango $0 <= z_i <= 1$.
Aquí, los conjuntos difusos de tipo 2 generalizados son conjuntos difusos de tipo 2 de intervalo basados en *zSlices*, los cuales son capaces de capturar la incertidumbre intra-intermedia que se encuentra dentro de los pesos de la arquitectura de la red neuronal de tres capas con aprendizaje de retro propagación.
En (de Penning et al., 2014) se plantea la utilización de conocimiento del dominio, y se sostiene que el comportamiento aprendido se puede extraer para actualizar el dicho conocimiento existente para su validación, informes, mantenimiento, evolución y retroalimentación. Además, el enfoque permite codificar el conocimiento del dominio en el modelo y se ocupa de la incertidumbre en los datos del mundo real.
En (Li & Srikumar, 2019) también se sugiere que el entrenamiento *end-to-end* exige de una gran cantidad de ejemplos de entrenamiento, donde entrenar una red típica para la traducción automática puede requerir millones de pares de oraciones. En este trabajo se razona sobre las dificultades y los costos de curar grandes cantidades de datos

anotados y, en consecuencia, plantean la problemática de que no se disponga de conjuntos de datos masivos para nuevas tareas, dominios o idiomas.
Como solución a este problema de datos insuficientes o incompletos en Borges et al., 2006 se plantea la inserción de conocimientos previos donde advierten que, en todos los casos, las redes lograron un desempeño efectivo al aprender operadores temporales.
Por su parte, Diligenti et al., 2017 sugiere que el aprendizaje profundo permite desarrollar representaciones de características y entrenar modelos de clasificación de una manera totalmente integrada. Sin embargo, aprender redes profundas es bastante difícil, y mejora en arquitecturas superficiales sólo si una gran cantidad de datos de entrenamiento están disponibles. Inyectar conocimientos previos al *learner* es una forma basada en principios de reducir la cantidad de datos de formación necesarios, ya que este no necesita inducir el conocimiento a partir de los datos en sí.
El modelo presentado en (Riegel et al., 2020) es diferenciable *end-to-end*, y el aprendizaje minimiza una función de pérdida novedosa que captura la contradicción lógica, lo que proporciona resistencia al conocimiento inconsistente. También permite la suposición de mundo abierto al mantener límites en los valores de verdad que pueden tener semántica probabilística, lo que produce resistencia al conocimiento incompleto.
En el artículo presentado en (Marra et al., 2019), presentan LYRICS, que es una herramienta para IA implementada en TensorFlow (TF). LYRICS proporciona un lenguaje de entrada que permite definir un conocimiento arbitrario previo como lógica de primer orden (LPO). En este trabajo se muestra la generalidad del enfoque presentando algunos casos de uso del lenguaje, incluida la verificación de modelos, el aprendizaje supervisado y la clasificación colectiva. La introducción de conocimientos previos en el proceso de aprendizaje es un paso fundamental para superar estas limitaciones, dado que no requiere que el proceso de entrenamiento induzca las reglas del conjunto de entrenamiento, por lo que se reduce la cantidad de datos de entrenamiento requeridos.
Además, el uso de conocimientos previos se puede utilizar para expresar el comportamiento deseado del *learner* en cualquier entrada, proporcionando mejores garantías de comportamiento en un entorno adverso o no controlado.

### *Falta de generalización*

(Sinha et al. 2020) estudian la tarea de la generalización lógica, en el contexto del razonamiento relacional utilizando *Graph Neural Network*s (GNN). En particular, estudian cómo las GNN pueden inducir reglas lógicas y generalizar combinando estas reglas de formas novedosas después del entrenamiento.
Para Ramirez et al., 2019 la hibridación de métodos puede *aumentar el rendimiento* de un sistema y aprovechar los beneficios que ofrecen estas técnicas para resolver problemas complejos.
Del mismo modo, Wang & Pan, 2020 demuestran la efectividad y generalización del modelo propuesto en múltiples tareas de extracción de información. Para mejorar el rendimiento de extracción en el dominio del procesamiento natural del lenguaje (PNL), proponen incorporar conocimiento del dominio como reglas lógicas que se integran en el sistema de aprendizaje de representación a través de un marco unificado.
La solución propuesta por (Li et al., 2021), exhibe una fuerte generalización sistemática con una precisión general del 72%, superando a los métodos neuronales de *end-to.end* en casi un 33%.

(Alzaeemi et al., 2019) basan su trabajo en la performance de los modelos híbridos, haciendo foco en la introducción de lógica para el cálculo de parámetros y para decrementar la cantidad de neuronas en las capas ocultas.
El trabajo presentado en (Barceló et al., 2020) tiene como objetivo responder a la pregunta de cuáles son los clasificadores de nodos que pueden ser capturados por arquitecturas GNN (*Graph Neural Network*) como AC-GNN (*Aggregate-Combine, Graph Neural Network*). Para empezar a responder a esta pregunta, proponen centrarse en clasificadores lógicos, es decir, en fórmulas unarias expresables en lógica de predicados de primer orden (FO): dicha fórmula clasifica cada nodo *v* dentro del grafo, según a si la fórmula es válida para tal nodo *v* o no, y validan experimentalmente sus hallazgos mostrando que la expresividad teórica de los ACR-GNN (*Aggregate-Combine-Readout, Graph Neural Network*), así como las diferencias con los AC-GNN, se pueden observar cuando la red aprende de los ejemplos. Para los AC-GNN, un punto de partida significativo para medir su poder expresivo es la lógica $FOC_2$, el fragmento de lógica FO que solo permite fórmulas con dos variables, pero a su vez permite utilizar cuantificadores de conteo de la forma $\exists^{\geq N}\varphi$, lo cual establece que existen al menos *N* nodos que satisfacen la formula $\varphi$.
En particular, muestran que en los datos de gráficos sintéticos que se ajustan a las fórmulas $FOC_2$, los AC-GNN luchan por ajustarse a los datos de entrenamiento, mientras que los ACR-GNN pueden generalizar incluso a grafos de tamaños que no son vistos durante el entrenamiento.

### *Falta de razonamiento*

Para soportar esta falencia (Manhaeve et al., 2018), presentan un *framework* donde las redes neuronales y la programación lógica probabilística se integran de una manera que explota la expresividad y las fortalezas de ambos mundos y se puede llevar adelante un entrenamiento *end-to-end* basándose en ejemplos.
En el trabajo propuesto por Han et al., 2021, los autores utilizan enfoques de razonamiento lógico simbólico, que incluyen el aprendizaje meta-interpretativo y programación lógica de primer orden para aportar conocimientos básicos y remediar el lago de información de supervisión.
En el artículo presentado por (Cai et al., 2017) proponen un método de razonamiento basado en *Deep Neural Network* (DNN). El concepto básico de este método es entrenar redes neuronales con retroalimentación profunda y hacer que sean capaces de manipular símbolos. En particular, este método permite a las redes neuronales realizar el razonamiento con variables en la lógica de predicados. Además, amplían la aplicación de este método al descubrimiento de axiomas y permitiendo que las redes neuronales aprendan el *sentido común* de los datos.

### *Falta de explicabilidad*

Para hacer frente a esta deficiencia AmirHosseini & Hosseini, 2019 proponen mejorar las compensaciones entre la precisión y la interpretabilidad del sistema de inferencia difuso (FIS), ajustando los parámetros del modelo mediante un algoritmo evolutivo que explora el espacio de búsqueda de manera más inteligente que otros algoritmos propuestos.

Mahdavifar & Ghorbani, 2020 proponen un sistema experto de redes neuronales integradas en profundidad (DeNNeS), que extrae reglas refinadas de una arquitectura de red neuronal profunda (DNN) entrenada para sustituir la base de conocimientos de un sistema experto. Posteriormente, la base de conocimientos se utiliza para clasificación de incidentes *invisibles* de seguridad, y para informar al usuario final de la regla correspondiente que hizo esa inferencia.

Por su parte Cocarascu et al., 2018, durante el entrenamiento, utilizan un codificador automático para clasificar las características de los ejemplos de entrada, como más o menos representativos. Estos ejemplos de formación están etiquetados como pertenecientes a una de dos clases dadas (resultados). Luego se usa esa clasificación para seleccionar un subconjunto de características de mayor rango, de modo que la restricción de los ejemplos de entrenamiento de estas características sea coherente (es decir, sin dos restricciones que tengan las mismas características, pero diferentes resultados). La restricción resultante de los ejemplos de entrenamiento se asigna luego a un *framework* de argumentación abstracto que, a su vez, se asigna a un conjunto de reglas lógicas, de la variedad de la programación lógica.

En el trabajo desarrollado en (Csiszár et al., 2020), los autores proponen una sugerencia sobre cómo aún se puede crear un marco teórico sintetizando los mundos de la lógica continua y los sistemas de toma de decisión de múltiple criterio (TDMC), y examinan las principales propiedades del operador de preferencia en sistemas nilpotentes[2].

Este marco coherente admite una posible aplicación de los resultados en el campo de la inteligencia artificial, como un paso importante hacia la interpretabilidad de los modelos neuronales.

El objetivo del proyecto publicado en (Schmid y Finzel, 2020) es combinar enfoques de aprendizaje profundo de caja negra, con aprendizaje automático interpretable para la clasificación de diferentes tipos de imágenes médicas, para combinar la precisión predictiva del aprendizaje profundo y la transparencia y comprensibilidad de modelos interpretables.

En este trabajo, los autores plantean como objetivo desarrollar un enfoque para una asociación humana-IA equilibrada, al hacer que las decisiones basadas en el aprendizaje automático en la medicina sean transparentes, comprensibles y corregibles. El principal resultado del proyecto será un *framework* para una interfaz de explicación que se basa en explicaciones mutuas.

En el estudio presentado en (Dombi y Csiszár, 2021), los autores demuestran que los sistemas lógicos nilpotentes ofrecen un marco matemático apropiado para una hibridación de modelos neuronales y lógicos nilpotentes continuos, lo que ayuda a mejorar la interpretabilidad y la seguridad del aprendizaje automático.

Uno de los mayores desafíos es la creciente necesidad de abordar el problema de interpretabilidad para mejorar la transparencia del modelo, rendimiento y seguridad.

La combinación de redes neuronales profundas con reglas lógicas estructuradas y herramientas de decisión de múltiple criterio, donde los operadores lógicos se aplican a los clústeres creados en la primera capa, contribuye a la reducción de la naturaleza de caja negra de los modelos neuronales.

Por su parte Aghaeipoor et al., 2023, proponen sistemas explicativos difusos basados en reglas para redes neuronales profundas, que pueden utilizarse con fines de

[2] En matemática, un elemento x de un anillo R se dice que es *nilpotente* si existe algún entero positivo n tal que $x^n = 0$. El término se utiliza para describir elementos que desaparecen cuando se elevan a una potencia.

explicabilidad local y global. El algoritmo aprende un conjunto compacto pero preciso de reglas difusas basadas en la importancia de las características extraídas de las redes entrenadas. El método puede ayudar a aclarar el proceso de decisión de las DNN en los problemas de clasificación de datos tabulares y se puede utilizar en varios campos donde la interpretabilidad de las DNN es crucial para garantizar la auditabilidad y la confiabilidad. Los resultados de la evaluación de diferentes aplicaciones revelaron que las explicaciones difusas mantenían la fidelidad y la precisión de las redes neuronales profundas originales, al tiempo que implicaban una menor complejidad y una mejor comprensibilidad.

En el estudio presentado en (Ciravegna et al., 2023), los autores presentan una metodología para implementar redes lógicas explicadas (LEN - Logic Explained Network) que permiten explicar el razonamiento detrás de las decisiones tomadas por las redes neuronales. Los LEN pueden utilizarse como clasificadores explicables o como redes adicionales para hacer que un clasificador de caja negra pueda explicarse mediante fórmulas de lógica de primer orden. Los LEN son lo suficientemente generales como para cubrir una gran cantidad de escenarios y proporcionan explicaciones más compactas y significativas que otros modelos de caja blanca. La explicabilidad es crucial en situaciones en las que se espera que la máquina respalde la decisión de los expertos humanos.

Por su parte Burkhardt et al., 2021, proponen un método para extraer reglas lógicas de redes neuronales binarias utilizando reglas convolucionales. El método implica entrenar una red neuronal binaria, extraer los pesos de la red, generar un conjunto de árboles de decisión binarios y aplicar la búsqueda local estocástica para extraer reglas convolucionales de primer orden. Las reglas extraídas se pueden utilizar para la validación del modelo y son fáciles de visualizar. El método se puede utilizar para datos de entrada de alta dimensión, como imágenes, y combina las ventajas de las redes neuronales y el aprendizaje de reglas. El artículo presenta un enfoque prometedor para extraer reglas interpretables de las redes neuronales binarias, con implicaciones prácticas en varios dominios.

En el trabajo desarrollado por Barbiero et al., 2022, los autores proponen un enfoque novedoso para extraer explicaciones lógicas de las redes neuronales utilizando la lógica de primer orden y un criterio basado en la entropía para identificar los conceptos más relevantes. Este enfoque se puede aplicar en ámbitos críticos para la seguridad, desde los datos clínicos hasta la visión por computadora. El documento define seis métricas cuantitativas para comparar el enfoque propuesto con los métodos más avanzados y presenta una función de decodificación conceptual para funciones de entrada no similares a los conceptos. El enfoque propuesto supera a los modelos de caja blanca de última generación en términos de precisión de clasificación y coincide con los rendimientos de las cajas negras. El artículo demuestra la eficacia del enfoque propuesto a través de cuatro estudios de casos diferentes. En resumen, este enfoque proporciona una manera de mejorar la interpretabilidad y el rendimiento de las redes neuronales en dominios críticos para la seguridad.

En el trabajo desarrollado en (Borges et al., 2006), los autores analizan los métodos y principios de la computación simbólica neuronal para el aprendizaje automático y el razonamiento integrados. Destaca la importancia de integrar el aprendizaje neuronal con métodos de razonamiento sólidos basados en el simbolismo para desarrollar sistemas y herramientas explicables y responsables basados en la IA y el aprendizaje

automático. Proporciona una metodología basada en principios para integrar el aprendizaje automático y el razonamiento, que puede conducir al desarrollo de sistemas de IA más efectivos y eficientes con implicaciones prácticas para varios dominios. En general, el documento proporciona una valiosa contribución al campo de la IA y el aprendizaje automático.

En el trabajo propuesto por (Garcez et al., 2019), los autores proponen un método para integrar el conocimiento simbólico en redes neuronales no supervisadas mediante la codificación del conocimiento en forma proposicional y de primer orden en la RBM. El enfoque de codificación se evalúa mediante experimentos en dos dominios y se presenta el conjunto final de reglas para codificar el conocimiento simbólico en redes neuronales no supervisadas. Las implicaciones prácticas incluyen mejorar el rendimiento de las redes neuronales no supervisadas en tareas relacionadas con la representación del conocimiento y el razonamiento, proporcionar interpretabilidad e inspirar nuevas investigaciones en el campo. Las contribuciones del artículo están relacionadas con el desarrollo de un método para integrar el conocimiento simbólico en redes neuronales no supervisadas, la codificación del conocimiento simbólico en RBM y la validación del enfoque mediante experimentos en dos dominios.

En el trabajo desarrollado en (Dombi y Csiszár, 2021), se propone un enfoque híbrido que combina redes neuronales con herramientas de toma de decisiones de lógica continua y multicriterio para mejorar la interpretabilidad y la seguridad del aprendizaje automático. El enfoque utiliza lógica nilpotente continua y operadores lógicos continuos para modelar desigualdades blandas y reduce el número de parámetros a aprender. Las implicaciones prácticas incluyen la mejora de la interpretabilidad y la seguridad del aprendizaje automático en diversos dominios, como la salud y las finanzas. El artículo presenta un modelo neuronal nilpotente, un operador basado en umbrales y una selección sencilla de funciones de activación. En resumen, el artículo propone un enfoque novedoso que puede mejorar la interpretabilidad y la seguridad del aprendizaje automático.

En el estudio presentado (Wang y Pan, 2020), los autores proponen un marco que integra el conocimiento lógico en forma de lógica de primer orden en un sistema de aprendizaje profundo para la extracción de información. El marco propuesto se compone de una red neuronal profunda, un banco lógico y una unidad de discrepancia. El marco se entrena conjuntamente de principio a fin, lo que permite integrar el conocimiento lógico con las capacidades de aprendizaje de las redes neuronales profundas. La efectividad y la generalización del modelo propuesto se demuestran en múltiples tareas de extracción de información. En general, el marco propuesto tiene el potencial de mejorar la precisión y la generalización de las tareas de extracción de información, lo que puede tener implicaciones prácticas en varios dominios, como el procesamiento del lenguaje natural, la recuperación de información y la minería de datos.

En el estudio presentado (Giunchiglia, Stoian, y Lukasiewicz, 2022), *Aprendizaje profundo con restricciones lógicas* se analizan varios métodos que utilizan conocimientos básicos especificados lógicamente en los modelos de aprendizaje profundo. Estos métodos se clasifican según el lenguaje lógico utilizado para expresar los conocimientos básicos y los objetivos logrados. Los métodos pueden mejorar el rendimiento, aprender de menos datos, garantizar el cumplimiento de los conocimientos básicos y mejorar la interpretabilidad. El documento destaca la

importancia de utilizar restricciones lógicas en aplicaciones críticas para la seguridad y proporciona un recurso valioso para los investigadores y profesionales interesados en utilizar restricciones lógicas en los modelos de aprendizaje profundo.
Por su parte Dai & Muggleton, 2021, proponen un enfoque novedoso llamado aprendizaje meta-interpretativo abductivo (Meta Abd) que une la abducción y la inducción para aprender redes neuronales e inducir teorías lógicas de forma conjunta a partir de datos sin procesar. Este enfoque puede tener importantes implicaciones prácticas en varios dominios donde se requiere razonar con datos sin procesar. Además, se demostró que Meta Abd supera a los modelos de aprendizaje profundo de extremo a extremo y a los métodos neuro-simbólicos de última generación en términos de precisión predictiva y eficiencia de datos en dos tareas de aprendizaje complejas. Los programas lógicos inductores que se aprenden y las redes neuronales se pueden reutilizar como conocimientos básicos en tareas de aprendizaje posteriores. El enfoque también se puede utilizar para inducir programas lógicos interpretables por humanos directamente a partir de datos sin procesar y se puede extender a otros dominios en los que puede resultar útil combinar redes neuronales con la programación lógica inductiva (ILP).

A modo de resumen, el grafico de barras de la Fig. 12 presenta la distribución de trabajos según la deficiencia abordada.

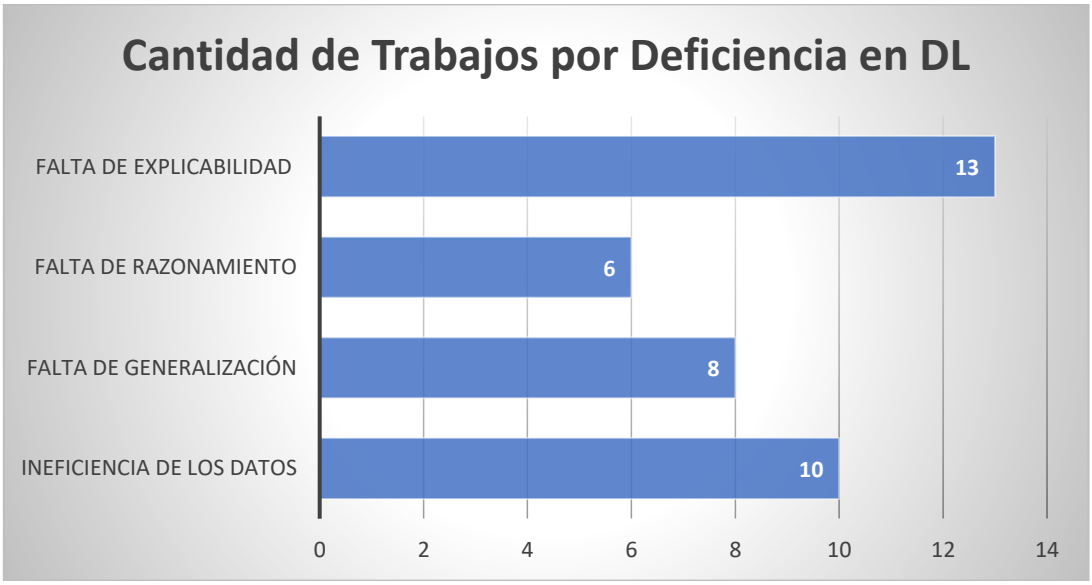


*Figura 12. Cantidad de artículos por cada deficiencia de DL abordada*

### 4.3.2 PI2. ¿En qué parte del proceso de DL se aplica Lógica?

Con esta pregunta de investigación, se intenta comprender cómo se aplican las herramientas basadas en lógica en el proceso de desarrollo de los modelos basados en IA no-simbólica. Concretamente, se busca entender si las distintas variantes de la lógica se utilizan para, por ejemplo, encontrar los híper-parámetros óptimos, para encontrar las características, para mejorar los datos de entrenamiento, o para acelerar el aprendizaje.

### Encontrar los híper-parámetros óptimos

En el trabajo desarrollado en (Pal y Kar, 2019), los autores actualizan los pesos de la red neuronal con la ayuda de conjuntos difusos de tipo 2 generales, para hacer la red más robusta en la captura de incertidumbres debido a la asignación de algunos grados de pertenencia a valores nítidos. Los conjuntos difusos tipo 2 se usan para modelar incertidumbre e imprecisión, y son esencialmente conjuntos difusos en los que los grados de pertenencia son conjuntos difusos de tipo 1. La utilización de sistemas difusos tipo 2 se justifica cuando existe un alto nivel de incertidumbre, en casos donde se busca encontrar mejores resultados que la lógica difusa tradicional (Díaz, Sotelo, y Bravo 2009).

En el trabajo presentado en (AmirHosseini y Hosseini, 2019), los autores desarrollan un modelo hibrido que busca optimizar los parámetros utilizando un enfoque de evolución difusa-diferencial (*fuzzy-DE*). Utilizan también, para el proceso de diagnóstico de metástasis hepática, un motor de inferencia difuso, donde aplican reglas difusas como base de conocimientos, incluidos los conjuntos y la membresía difusos del sistema de inferencia, de acuerdo con las variables difusas de entrada.

Por otro lado, Manhaeve et al., 2018 presentan un enfoque para entrenar conjuntamente los parámetros de hechos probabilísticos y redes neuronales en programas DeepProbLog. El aprendizaje se realiza mediante el uso de ProbLog para calcular el gradiente de la pérdida que luego se usa en métodos estándar basados en gradiente descendente, para optimizar los parámetros tanto en el programa de lógica probabilística como en las redes neuronales.

En el trabajo presentado en (Mahdavifar y Ghorbani, 2020), los autores utilizan programación lógica dentro de un sistema experto para la optimización de parámetros de la DNN. El sistema experto de DNN se integra para detectar ataques cibernéticos, que aprende patrones subyacentes en el conjunto de datos e ignora *outliers*. Debido a la naturaleza estadística de las redes neuronales, éstas podrían ser una buena sustitución a las reglas de la base de conocimientos elaboradas manualmente. Sus algoritmos de aprendizaje podrían ser instrumentos para construir reglas ***if-then*** y alimentarlas a los sistemas expertos. A la inversa, la salida de los sistemas expertos también podría ser una entrada a la red neuronal.

En el trabajo desarrollado por (Csiszár et al., 2020), los autores utilizan lógica difusa, como una herramienta de lógica continua, para poder razonar sobre los valores de los pesos y parámetros de la red. En este artículo, los autores proponen un *framework* para modelar el pensamiento humano mediante el uso de herramientas de ambos campos: operadores lógicos difusos y operadores de agregación y preferencia. En este *framework*, la agregación, la preferencia y los operadores lógicos son descritos por la misma función generadora unaria. De manera similar a la implicación definida como una composición de la disyunción y el operador de negación, los operadores de preferencia se introdujeron como una composición del operador de agregación y el operador de negación. En teoría clásica, la preferencia es una relación binaria estrechamente relacionada a las implicaciones:

$$xRy \Leftrightarrow \text{“y no es peor que x”}$$

Las preferencias entre alternativas también pueden describirse mediante un valor relación de preferencia *p*, tal que el valor *p (x, y)* se normaliza, y se introduce como el grado en el que el enunciado *"y no es peor que x"* es verdadero.

$$p(x, y) = \text{verdad de } (y \geq x).$$

Aquí, p es una función continua que es estrictamente decreciente en la primera variable, y estrictamente creciente en la segunda variable, mientras que p (x, y) = n (p (y, x)) también debe ser válido, siendo n(x) el operador de negación.
En el trabajo desarrollado por (Dombi y Csiszár, 2021), los autores presentan un modelo neuronal nilpotente, donde operadores lógicos nilpotentes[3] combinados con herramientas de decisión multicriterio, se implementan en las capas ocultas de las redes neuronales. En este modelo solo se deben aprender los pesos de la primera capa (parámetros de los hiperplanos que separan el espacio de decisión).
En el modelo neuronal nilpotente, las funciones de activación (*Squashing Function*[4]) en la primera capa son funciones de pertenencia que representan valores de verdad de las desigualdades, normalizando así las entradas. Al mismo tiempo, las funciones de activación en las capas ocultas modelan la función de corte (para evitar el problema del gradiente de desaparición) en los operadores lógicos nilpotentes. La idea básica del uso de lógica continua es la sustitución del espacio de valores de verdad (T, F) por un intervalo compacto como [0; 1]. Esto significa que las entradas y salidas de las puertas lógicas extendidas son números reales del intervalo unitario, que representan valores de verdad de desigualdades. Los cuantificadores $\forall x$ y $\exists x$ se reemplazan por Supx e Infx, y las conectivas lógicas son funciones continuas. Basados en esta idea, el pensamiento humano y el lenguaje natural se puede modelar de una manera sofisticada.
Los sistemas lógicos nilpotentes ofrecen un nuevo enfoque para diseñar redes neuronales utilizando lógica continua, ya que las funciones de pertenencia (que representan el valor de verdad de una desigualdad), y también los operadores lógicos nilpotentes son modelados por perceptrones.
(Schmid y Finzel, 2020) introducen la programación lógica inductiva (ILP) como un enfoque poderoso de aprendizaje automático interpretable que, naturalmente, permite combinar el razonamiento y el aprendizaje. La programación lógica inductiva permite aprender modelos compuestos de tales reglas lógicas, mediante la combinación del razonamiento y el aprendizaje de forma natural.
Las teorías de fondo pueden explotarse durante el aprendizaje, y las reglas aprendidas se pueden combinar con reglas de inferencia predefinidas opcionalmente. Se ha demostrado que las reglas aprendidas con ILP pueden apoyar la toma de decisiones humanas en dominios complejos.

### Encontrar las características

En (Melin et al., 2018) se diseñó un modelo híbrido que utiliza redes neuronales modulares y lógica difusa para proporcionar el diagnóstico de riesgo de hipertensión de una persona. Con el objetivo de realizar una integración de los resultados de los

[3] Un sistema conectivo es nilpotente, si la conjunción c es una t-norma nilpotente, y la disyunción d es una t-conorma nilpotente.
[4] Aproximación del rectificador ReLU. Es una función que dirige la entrada hacia uno de los extremos de un pequeño intervalo. En las redes neuronales, estos se pueden usar en los nodos de una capa oculta para aplastar la entrada. Esto introduce la no linealidad a la NN y permite que la NN sea efectiva.

módulos, las redes neuronales modulares utilizan integradores de respuesta como la lógica difusa. En esta investigación se utilizan sistemas difusos como integradores de respuesta, porque de esta forma se puede manejar la incertidumbre en las decisiones.
En el modelo presentado en (Sinha et al., 2020), los autores usan lógica para predecir relaciones entre nodos de una GNN (*Graph Neural Network*). En lugar de razonar en un solo grafo de conocimiento estático durante el entrenamiento y las pruebas, consideran el entorno en el que el modelo debe aprender a generalizar en grafos no vistos durante la evaluación.
Definen un gran conjunto de reglas que se basan en lógica proposicional. Luego dividen el conjunto de reglas en subconjuntos superpuestos, que se usan para definir los subconjuntos únicos. Finalmente, dentro de cada subconjunto generan varios grafos de conocimiento que son gobernados por el conjunto de reglas de inicial.
El modelo que proponen en (Cocarascu et al., 2018) genera programas lógicos, que capturan el razonamiento desde un conjunto de ejemplos *S*. Estas reglas permiten hacer exactamente las mismas predicciones para nuevos ejemplos que un predictor basado en un *autoencoder* + ANN (Artificial Neural Net). Al mismo tiempo, estas reglas resumen la estructura del conjunto de ejemplos *S*. En particular, describen de manera concisa en términos lógicos cuál debería ser la predicción y cuáles características deberían influir en ella. Las reglas son genéricas para S, pero cuando se hace una predicción para un nuevo ejemplo, sus características se agregan como hechos a las reglas.
En (Ramirez et al., 2019) se utiliza una unidad de módulo básico para realizar la clasificación de señales dirigida. En la arquitectura resultante, se define la integración de varios módulos, donde cada unidad de módulo básico se compone de tres clasificadores diferentes basados en los siguientes modelos: algoritmo KNN (*K-Nearest Neighbor*) difuso, perceptrón multicapa con descenso de gradiente y momento (MLP-GDM) y perceptrón multicapa con retropropagación de gradiente conjugada escalada (MLP-SCG). Las salidas de los clasificadores se combinan mediante un sistema difuso para la integración de los resultados. Diseñaron dos sistemas de lógica difusa, el sistema de inferencia difuso Mamdani tipo 1 (FIS tipo 1) y un sistema difuso de intervalo tipo 2 (IT2FIS). El objetivo es realizar una comparación entre FIS tipo 1 e IT2FIS en el modelo híbrido. Los conjuntos difusos tipo 2 se usan para modelar incertidumbre e imprecisión, y son esencialmente conjuntos difusos en los que los grados de pertenencia son conjuntos difusos de tipo 1. La utilización de sistemas difusos tipo 2 se justifica cuando existe un alto nivel de incertidumbre, en casos donde se busca encontrar mejores resultados que la lógica difusa tradicional.
La diferencia fundamental entre la lógica difusa tipo1 y tipo 2 está en su definición: Un conjunto difuso A en U puede ser representado como un conjunto de pares ordenados de elementos, y su valor de pertenencia está dado por:

$$A = \{x, \mu_A(x) \mid x \in U\}$$

Donde U es el universo del discurso continuo (por ejemplo, todos los números reales de U = R).
Un conjunto difuso tipo 2 es denotado por Ā y es caracterizado por una función de pertenencia $\mu_{\bar{A}}(x,u)$ donde $x \in X$ y $u \in \int_x^u \subseteq [0,1]$ y $0 \le \mu_{\bar{A}}(x,u) \le 1$, es definido mediante la ecuación:

$$\bar{A} = \{(x, u, \mu_{\bar{A}}(x,u)) / x \in X, \forall u \in \int_x \subseteq [0,1]\}$$

Entonces se puede afirmar que existe incertidumbre en la lógica difusa T2 tanto en la variable primaria como en la secundaria, mientras que en la lógica T1 no existe incertidumbre en la variable primaria (Díaz et al., 2009).

En (Wang y Pan, 2020), los autores utilizan reglas lógicas para hacer cumplir correlaciones complejas en el espacio de salida, e integrar estas reglas en el sistema de representación distribuida de aprendizaje con un mecanismo de aprendizaje para lograr inferencia conjunta. El módulo lógico se compone de un conjunto de reglas lógicas representadas por LPO. Estas reglas especifican explícitamente las relaciones complejas en el espacio de etiquetas de salida, que no se pueden manejar con restricciones simples. Estas dependencias pueden considerarse restricciones complejas que pueden expresarse de manera eficiente como reglas lógicas.
En (Riegel et al., 2020), los autores proponen un *framework* novedoso que proporciona propiedades clave tanto de las redes neuronales (aprendizaje) como de la lógica simbólica (conocimiento y razonamiento). Cada neurona tiene un significado como componente de una fórmula en una lógica ponderada de valor real, produciendo una representación desenredada altamente interpretable. La inferencia es más bien omnidireccional, que se centra en variables objetivo predefinidas, y corresponde al razonamiento lógico, incluido el teorema de lógica clásica de primer orden que se demuestra como un caso especial.
En (Barceló et al. 2020) se plantea la utilización de Lógica $FOC_2$ (Lógica de predicados de primer orden con cuantificadores de conteo), un fragmento bien estudiado de lógica de primer orden, el cual está estrechamente relacionado con la prueba Weisfeiler-Lehman (WL) y, por lo tanto, con los GNN.
La idea principal detrás de las GNN (*Graph Neural Network)* es que las conexiones entre neuronas no son arbitrarias, sino que reflejan la estructura de los datos de entrada. Este enfoque está motivado por redes neuronales convolucionales y recurrentes y generaliza ambas.
En este artículo los autores dan un paso hacia la comprensión del poder expresivo, al establecer conexiones entre GNN y formalismos lógicos conocidos. Se sugiere además que, estas conexiones son conceptualmente importantes, ya que permiten comprender los aspectos de comportamiento inherentemente procedimentales de algunos fragmentos de GNN en términos del aspecto más declarativo de los lenguajes lógicos.
La arquitectura AC-GNN se amplía de una manera muy simple al permitir lecturas globales, donde en cada capa también se calcula un vector de características para todo el grafo, y lo combinan con agregaciones locales; A estos los definen como *GNN de lectura combinada agregada* (ACR-GNN: *Aggregate-Combine-Readout, Graph Neural Network*).
En este trabajo demuestran que cada fórmula de FOC-2 puede ser capturada por un ACR-GNN.
Este estudio relaciona el poder de los GNN con el de los clasificadores expresados en la lógica de predicados de primer orden (FO), sobre grafos (no dirigidos) donde cada vértice tiene características únicas.
En (Li et al. 2021), los autores adoptan programas funcionales para entender el significado semántico de los conceptos, por lo que ven a la semántica del aprendizaje

como una inducción del programa. La semántica de un concepto es tratada como una función.
En (Dombi y Csiszár, 2021) los autores proponen un enfoque híbrido que combina redes neuronales con herramientas de toma de decisiones de lógica continua y multicriterio para mejorar la interpretabilidad y la seguridad del aprendizaje automático. El enfoque propuesto utiliza lógica nilpotente continua y operadores lógicos continuos para modelar desigualdades blandas, es decir, funciones de pertenencia. La arquitectura de red se diseña antes del entrenamiento mediante operadores lógicos continuos y herramientas de decisión multicriterio con pesos determinados que funcionan en las capas ocultas. La estructura está diseñada adecuadamente para conducir a una reducción drástica en el número de parámetros que deben aprenderse. El documento también presenta un operador basado en umbrales que puede expresar un operador de conjunción, disyunción y agregación. La base teórica ofrece una selección sencilla de funciones de activación, como la función de corte o su aproximación diferenciable, la función de aplastamiento, y también sugiere una explicación del gran éxito de la unidad lineal rectificada (*ReLU*). El documento se centra en la arquitectura de un modelo híbrido y presenta los componentes básicos para futuras aplicaciones en redes neuronales profundas.
En (Wang y Pan, 2020), los autores proponen un marco que integra el conocimiento lógico en forma de lógica de primer orden en un sistema de aprendizaje profundo para la extracción de información. El marco propuesto consta de tres componentes:

- Componente de red neuronal profunda (DNN): este componente toma una secuencia de palabras como entrada y produce un vector de predicción para cada palabra (y, posiblemente, las relaciones candidatas) como salida.
- Banco lógico: este componente se alimenta con conocimientos generales de dominio que son fáciles de obtener y formaliza el conocimiento como un conjunto de reglas lógicas de primer orden. A cada regla lógica se le asigna un peso no negativo para indicar su nivel de confianza, que se actualiza de acuerdo con el corpus de entrenamiento.
- Unidad de discrepancia: este componente se utiliza para medir la discrepancia entre las salidas neuronales y las reglas lógicas. Se utiliza para mejorar las salidas neuronales mediante la regularización del conocimiento mediante reglas lógicas y actualizar los pesos de las reglas lógicas para cumplir con las características de los datos de entrenamiento.

El marco propuesto se entrena conjuntamente de principio a fin, lo que permite integrar el conocimiento lógico con las capacidades de aprendizaje de las redes neuronales profundas. La efectividad y la generalización del modelo propuesto se demuestran en múltiples tareas de extracción de información.

En (Giunchiglia et al., 2022) El documento analiza varios métodos que utilizan conocimientos básicos especificados lógicamente en los modelos de aprendizaje profundo. Estos métodos se clasifican según el lenguaje lógico utilizado para expresar los conocimientos básicos y los objetivos logrados. El documento analiza los siguientes métodos:

- Métodos basados en pérdidas: estos métodos utilizan funciones de pérdida para entrenar redes neuronales con restricciones. No pueden garantizar la satisfacción de las restricciones, pero pueden generalizar a partir de menos datos y/o aprovechar los datos no etiquetados.
- Métodos basados en reglas: estos métodos utilizan reglas lógicas para expresar las restricciones e incorporarlas al proceso de aprendizaje. Pueden garantizar la satisfacción de las restricciones, pero pueden requerir más datos para lograr un buen rendimiento.
- Métodos híbridos: estos métodos combinan métodos basados en pérdidas y en reglas para lograr un mejor rendimiento y satisfacer las restricciones.
- Métodos de programación lógica inductiva (ILP): estos métodos utilizan técnicas de ILP para aprender reglas lógicas a partir de ejemplos y conocimientos básicos. Pueden aprender reglas complejas y garantizar la satisfacción de las restricciones, pero pueden requerir más recursos computacionales.

Por lo tanto, el trabajo analiza varios métodos que utilizan restricciones y razonamientos lógicos en los modelos de aprendizaje profundo para lograr un mejor rendimiento, aprender de menos datos y garantizar el cumplimiento de los conocimientos básicos para las aplicaciones críticas para la seguridad.

### Mejorar los datos de entrenamiento

En el trabajo realizado por (Alzaeemi et al., 2019), los autores buscan encontrar los valores de verdad de los valores de entrada de una red neuronal de función de base radial (RBFNN - *Radial Basis Function Neural Network*) para las cláusulas, a través de programación lógica con 2SAT (*2-satisfiability*)[5].
En (de Penning et al., 2014) logran aprovechar la integración neuronal-simbólica, utilizando las redes para realizar un aprendizaje y adaptación robusta, y la extracción de conocimiento simbólico para representar las *relaciones temporales* explícitamente y por razonamiento cualitativo.
La solución propuesta en (Li y Srikumar, 2019) compila sistemáticamente declaraciones lógicas en grafos de cálculo que aumentan una red neuronal sin parámetros extra que aprender, o rediseño manual. Los autores sostienen que pueden *combatir la sed de datos de las redes neuronales* aprovechando el conocimiento del dominio expresado como lógica de primer orden. En tal sentido, las redes neuronales son provistas del conocimiento del dominio, el cual queda codificado y expresado como reglas lógicas de primer orden (LPO).
En (Diligenti et al., 2017), los autores sostienen que la regularización basada en semántica (SBR - *Semantic Based Regularization)* se utiliza como marco subyacente para representar el conocimiento previo, expresado como una colección de cláusulas LPO, y donde cada tarea a aprender corresponde a un predicado en la base de conocimiento. Dicha base correlaciona las tareas que se deben aprender y se traduce en un conjunto de restricciones que se integran en el proceso de aprendizaje a través de *backpropagation*.

[5] Problema computacional que asigna valores a variables, cada una de las cuales tiene dos valores posibles, para satisfacer un sistema de restricciones sobre pares de variables. Puede involucrar restricciones en más de dos variables, y de problemas de satisfacción de restricciones, que pueden permitir más de dos opciones para el valor de cada variable.

La regularización semántica es una estadística relacional de aprendizaje, que integra la capacidad de aprender de ejemplos y reglas lógicas. El conocimiento previo en SBR se expresa a través de un conjunto de cláusulas LPO.
El aprendizaje relacional estadístico es particularmente adecuado para inyectar conocimiento lógico en el aprendizaje, porque transforma el conocimiento en un conjunto de restricciones continuas que se pueden integrar en las funciones de costos que generalmente se consideran en el aprendizaje automático.
En el trabajo desarrollado por (Han et al., 2021), los autores llevan a cabo razonamiento lógico simbólico para hacer un análisis *causa-efecto* no supervisado, de entidades detectadas con anomalías a través del aprendizaje meta-interpretativo. Utilizan LPO para embeber conocimiento previo, y sistemas de razonamiento dentro de la red de grafos.
Finalmente, el enfoque que presenta el trabajo en (Xie, Zhou, y Soh, 2021), se trata de embeber conocimiento simbólico en una DNN, expresado como lógica temporal lineal (LTL), y usar esta información para guiar el entrenamiento del modelo. Específicamente, construyen incrustaciones semánticas de autómatas generados a partir de la fórmula LTL a través de una red neuronal gráfica.

### Acelerar el aprendizaje

En el trabajo presentado por Borges et al., 2006, los autores proponen una arquitectura que combina redes neuronales recurrentes de entrada-salida y un sistema neural-simbólico para el aprendizaje de conocimiento *simbólico temporal*. Mediante un enfoque experimental, han demostrado que la integración del conocimiento simbólico puede mejorar las características de una red neuronal, en particular la tolerancia al ruido, rendimiento y convergencia de un algoritmo de aprendizaje.
En (Marra et al., 2019), los autores sugieren que los predicados y funciones del conocimiento de LPO se pueden vincular a cualquier grafo computacional de *TensorFlow* (TF), y las fórmulas se convierten en un conjunto de restricciones de valor real, que participan en el problema de optimización general. Esto permite conocer el peso de los *learners*, bajo las limitaciones impuestas por el conocimiento previo. El marco es extremadamente general, ya que no impone restricciones en cuanto a qué modelos o conocimientos se pueden integrar.
Este documento presenta LYRICS, un entorno de TF basado en un lenguaje declarativo para la integración de conocimientos previos en el aprendizaje automático, que permite la plena expresividad de LPO para definir conocimiento.
LYRICS tiene su raíz en marcos como la Regularización basada en semántica (SBR) construida sobre Kernel Machines y Logic Tensor Networks (LTN) que se pueden aplicar a redes neuronales. Estos marcos transforman las cláusulas LPO en un conjunto de restricciones que se optimizan conjuntamente durante el aprendizaje.
En particular, cualquier teoría lógica de primer orden con muchas clasificaciones se puede expresar en el *framework*, lo cual permite declarar dominios de diferente tipo, con constantes, predicados y funciones, proporcionando una integración muy estrecha de aprendizaje y lógica, ya que cualquier grafo computacional puede vincularse a un predicado LPO. Esto permite limitar al *learner* tanto durante el entrenamiento como durante la inferencia.

Dado que el *framework* es agnóstico para los *learners* que están vinculados a los predicados, se puede utilizar en una amplia gama de aplicaciones, incluida la clasificación, el aprendizaje automático generativo o adversario, el aprendizaje secuencia a secuencia, la clasificación colectiva, etc. Las funciones LPO permiten el mapeo entre individuos de los dominios de entrada a un individuo del dominio de salida.
En el trabajo desarrollado en (Cai et al., 2017), se propone un método de razonamiento simbólico basado en DNN, y este método se aplica al descubrimiento de axiomas, el cual hace uso del concepto de manipulación simbólica. Específicamente, se basa en la capacidad de aprendizaje de las DNN y la capacidad de razonamiento de un sistema lógico: el sistema lógico genera ejemplos de entrenamiento, que indican cómo manipular símbolos a partir de una serie dada de datos, y luego las DNN intentan aprender tales ejemplos, puntuarlos y realizar abstracciones de posibles axiomas. En particular, este método permite a las redes neuronales profundas realizar un razonamiento simple con variables en la lógica de predicados.
Las redes neuronales de alimentación profunda pueden colaborar con el sistema lógico y aprender a manipular símbolos. En particular, son capaces de terminar algunas tareas de razonamiento con variables. Esta capacidad se puede utilizar para aprender reglas lógicas y se puede utilizar más para descubrir axiomas.

### Extracción de reglas

En el trabajo desarrollado en (Aghaeipoor et al., 2023), se proponen sistemas explicativos difusos basados en reglas para redes neuronales profundas. El algoritmo aprende un conjunto compacto pero preciso de reglas difusas basadas en la importancia de las características (es decir, los valores de atribución) extraídas de las redes entrenadas. Estos sistemas se pueden utilizar con fines de explicabilidad local y global.
Los resultados de la evaluación de diferentes aplicaciones revelaron que las explicaciones difusas mantenían la fidelidad y la precisión de las redes neuronales profundas originales, al tiempo que implicaban una menor complejidad y una mejor comprensibilidad.

En (Ciravegna et al., 2023) los autores presentan un enfoque general de XAI para arquitecturas neuronales, conocido como redes lógicas explicadas (LENs). Estas redes solo requieren entradas comprensibles para los humanos y proporcionan explicaciones en términos de fórmulas simples de lógica de primer orden (FOL). Los resultados experimentales muestran que los LEN pueden generar mejores clasificaciones que los modelos de caja blanca establecidos, como los árboles de decisión y las listas de reglas bayesianas, al tiempo que proporcionan explicaciones más compactas y significativas. El documento destaca la importancia de proporcionar explicaciones comprensibles para los humanos en varios ámbitos de aplicación.

En el trabajo de Garcez et al., (2019), se analizan los métodos y principios de la computación simbólica neuronal para el aprendizaje automático y el razonamiento integrados. Se destacan los enfoques metodológicos clave, como la integración basada en principios del aprendizaje neuronal con la representación y el razonamiento simbólicos del conocimiento, el uso de diferentes formalismos de representación del conocimiento, la preposicionalización para aprender lógica de primer orden, el uso de

sistemas de incrustación relacional para razonar sobre las relaciones entre entidades y el uso de redes neuronales profundas para el razonamiento visual. El documento destaca la importancia de integrar el aprendizaje neuronal con métodos sólidos de razonamiento basados en el simbolismo para desarrollar sistemas y herramientas explicables y responsables basados en la IA y el aprendizaje automático.

En (Burkhardt et al., 2021), los autores proponen un algoritmo para la extracción de reglas de descomposición, llamado *Deep Convolutional DNF Learner* (DCDL), que puede extraer reglas características para datos de entrada de alta dimensión, como imágenes. Además, se propone el uso de redes neuronales binarias (BNN) para la extracción de reglas, así como un método para la validación de modelos mediante reglas lógicas rigurosas. Finalmente, se presenta la visualización de las reglas extraídas para ayudar a comprender la funcionalidad de la red neuronal.

En Barbiero et al. (2022) proponen un enfoque diferenciable de extremo a extremo para extraer explicaciones lógicas de redes neuronales utilizando el formalismo de la lógica de primer orden. El método se basa en un criterio basado en la entropía para identificar automáticamente los conceptos más relevantes. Seis métricas cuantitativas se definen para comparar el enfoque propuesto con los métodos más avanzados, y una función de decodificación conceptual se presenta para derivar la fórmula FOL en el caso de funciones de entrada no similares a las de un concepto. Los resultados de los cuatro estudios de casos diferentes demuestran que el enfoque propuesto es eficaz para extraer explicaciones lógicas concisas en ámbitos críticos para la seguridad, como los datos clínicos y la visión por computadora.

En el trabajo desarrollado en Tran (2017) los autores proponen un método para integrar el conocimiento simbólico en redes neuronales no supervisadas, utilizando la codificación de la información en forma proposicional y de primer orden en la RBM (Restricted Boltzmann Machine). El enfoque se evaluó mediante experimentos en dos dominios: la predicción de los promotores del ADN y la comprensión de las relaciones familiares. El artículo también analiza la regla de confianza, una forma de conocimiento para representar fórmulas simbólicas en los RBM, y presenta el conjunto final de reglas para codificar el conocimiento simbólico en redes neuronales no supervisadas. Los resultados de los experimentos demuestran la validez del enfoque propuesto.

En (Giunchiglia et al., 2022) se analizan varios métodos de aprendizaje profundo que utilizan conocimientos básicos especificados lógicamente para lograr un mejor rendimiento. Estos métodos se clasifican según el lenguaje lógico utilizado para expresar los conocimientos básicos y los objetivos logrados. Estos incluyen métodos basados en pérdidas, métodos basados en reglas, métodos híbridos y métodos de programación lógica inductiva. Tales métodos pueden ayudar a generalizar a partir de menos datos, aprovechar los datos no etiquetados y garantizar la satisfacción de las restricciones. Los métodos en cuestión son útiles para aplicaciones críticas para la seguridad.
En conclusión, el trabajo analiza varios métodos que utilizan restricciones y razonamientos lógicos en los modelos de aprendizaje profundo para lograr un mejor rendimiento, aprender de menos datos y garantizar el cumplimiento de los conocimientos básicos para las aplicaciones críticas para la seguridad.

En (Dai & Muggleton, 2021), los autores proponen un enfoque novedoso llamado aprendizaje meta-interpretativo abductivo (Meta Abd) que combina la abducción y la inducción para aprender redes neuronales e inducir teorías lógicas de forma conjunta a partir de datos sin procesar. Esta técnica demuestra una mejora significativa en la precisión predictiva y la eficiencia de datos en comparación con los modelos de aprendizaje profundo de extremo a extremo y los métodos neuro-simbólicos de última generación. El sistema también permite la integración de redes neuronales con la programación lógica inductiva (ILP) para admitir la inducción de la teoría lógica de primer orden a partir de datos sin procesar, lo que permite la reutilización de programas lógicos inductores como conocimientos básicos en tareas de aprendizaje posteriores.

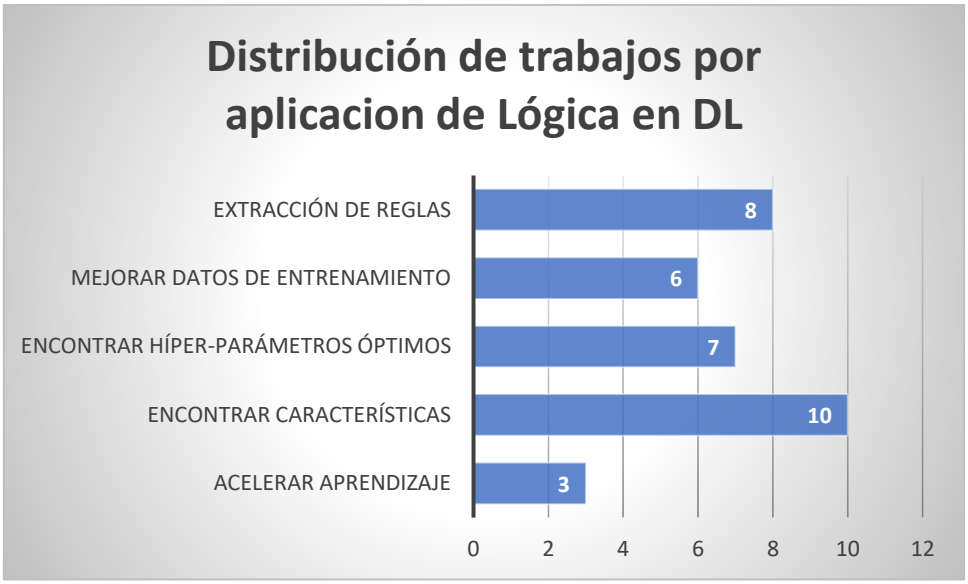


*Figura 13. Aplicación de Lógica en el proceso de DL*

### 4.3.3 PI3. ¿Qué herramientas de la lógica se aplican? ¿Qué tipos de tareas se abordan? ¿Sobre cuáles dominios?

Esta pregunta de investigación fue abordada en base a los diferentes tipos de lógica utilizada en cada trabajo.

#### Programación Lógica

En el trabajo presentado en (Alzaeemi et al., 2019), los autores presentan una solución de aplicación general, a través del uso de programación lógica tipo 2SAT, donde se busca la optimización de la red. (Parámetros y neuronas en la capa oculta).
En (Manhaeve et al., 2018), los autores utilizan programación lógica para resolver tareas de interpretación y clasificación de imágenes.
Por su parte, en el trabajo realizado por Cocarascu et al., 2018, los autores utilizan programación lógica, para tareas de clasificación, en una solución de aplicación general.
También se utiliza programación lógica en el trabajo realizado en (Mahdavifar y Ghorbani, 2020), aplicada dentro de un ***sistema experto***, para tareas de clasificación en la detección de *cyber*-ataques.

## Lógica Difusa

En (Melin et al., 2018), los autores utilizan lógica difusa (*Mamdani Fuzzy Inference*), y para optimizar el sistema de lógica difusa utilizaron un algoritmo genético. En este trabajo aplicado a Medicina, se busca hacer mediciones de presión sanguínea para hacer predicciones de riesgos de Hipertensión.
En (Pal y Kar, 2019), los autores plantean una solución de aplicación general para realizar predicciones en el cual utilizan lógica difusa de tipo 2.
El trabajo realizado por (AmirHosseini y Hosseini, 2019), los autores utilizan lógica difusa aumentado con Algoritmos evolutivos, para tareas de clasificación. En este trabajo de aplicación médica, proveen una solución para clasificación de tumores en imágenes de tomografía computarizada del hígado.
En (Ramirez et al., 2019) también utilizan lógica difusa del tipo 1 y 2. Esta solución de aplicación médica busca hacer clasificaciones de arritmias cardiacas.

En (Azizan y Sathasivam, 2023) los autores proponen un nuevo modelo que integra la lógica difusa aleatoria de corte alfa con el 3SAT en la red Hopfield para resolver problemas de optimización combinatoria. El artículo proporciona información sobre cómo se puede utilizar la lógica difusa para minimizar la carga de procesamiento y permitir que la estrategia sugerida dé cabida a neuronas de mayor precisión. El modelo propuesto se puede utilizar en diversas aplicaciones, como el procesamiento de imágenes, el reconocimiento de patrones y los problemas de optimización. Los hallazgos del artículo pueden ayudar a crear enfoques novedosos para la recopilación de datos para la próxima exploración de la programación lógica.
En (Aghaeipoor et al., 2023), los autores proponen sistemas explicativos difusos basados en reglas para redes neuronales profundas, que pueden utilizarse con fines de explicabilidad local y global.
El algoritmo aprende un conjunto compacto pero preciso de reglas difusas basadas en la importancia de las funciones (es decir, los valores de atribución) extraídas de las redes entrenadas.
El método propuesto puede ayudar a aclarar el proceso de decisión de las DNN en los problemas de clasificación de datos tabulares.
El uso de la representación lingüística difusa en el método propuesto modela el conocimiento semántico del espacio de entrada de una manera similar a la cognición humana, lo que refuerza la comprensibilidad de los modelos sustitutos basados en reglas.

## Lógica Temporal

En (de Penning et al., 2014), los autores utilizan lógica temporal aumentada con Redes bayesianas. La arquitectura descrita utiliza la lógica temporal como teoría T y una máquina de Boltzmann restringida (RBM - *Restricted Boltzmann Machines*) como red neuronal N. mediante este trabajo buscan predecir la reducción de emisiones de CO2 utilizando Agentes en Sistemas de transporte inteligentes.

## Lógica de Primer Orden (LPO)

En (Li y Srikumar, 2019) se plantea una solución de Aplicación general para realizar predicciones, utilizando Lógica de primer orden.
En (Sinha et al., 2020) para tareas de predicción, se utiliza lógica de primer orden para evaluar la generalización lógica en una GNN.
En el trabajo presentado por (Diligenti et al., 2017) para tareas de clasificación, se utiliza lógica de primer orden, para la clasificación de Imágenes.
En (Riegel et al., 2020) plantean una solución con la capacidad general de resolución de problemas de un demostrador de teoremas completo, utilizando lógica de primer orden para tareas de predicción.
Para realizar extracción de información de textos, Wang & Pan, 2020 utilizan una combinación de lógica de primer orden con lógica probabilística para clasificación.
En (Han et al., 2021), los autores proponen una solución de aplicación médica, para la identificación de discos en imagen de columnas. Para este trabajo de clasificación utilizan una combinación entre programación lógica de primer orden con aprendizaje meta-interpretativo.
En (Marra et al., 2019) se plantea una solución de Aplicación general para realizar clasificaciones, utilizando Lógica de primer orden aumentado con Redes Bayesianas.
En (Barceló et al. 2020) para tareas de clasificación, utilizan *LPO – FOC-2 (Lógica de primer orden con 2 variables) & Counting*, para aportar una solución de aplicación general. Si bien el trabajo está orientado a distinguir colores en nodos de un grafo, se han hecho pruebas también sobre una base de datos para identificación de proteínas, lo cual plantea una solución de aplicación real muy interesante.
En (Ciravegna et al., 2023) se presentan métodos fundamentales utilizados para implementar las redes explicadas por lógica (LEN - Logic Explained Networks).
Los autores describen el procedimiento utilizado para extraer reglas lógicas de los LEN para una observación individual o un grupo de muestras. Este procedimiento es común a todos los objetivos y casos de uso. Además, se proporciona la definición formal de los objetivos de aprendizaje que limitan a los LeN a proporcionar los tipos requeridos de explicaciones de la lógica de primer orden (FOL). Por último, el artículo analiza cómo restringir los LENs para que produzcan fórmulas lógicas concisas.
En (Burkhardt et al., 2021) los autores proponen un método para extraer reglas lógicas de redes neuronales binarias utilizando reglas convolucionales. El método se basa en la extracción de reglas de redes neuronales binarias con búsqueda local estocástica. El método propuesto se puede utilizar para extraer reglas interpretables de redes neuronales binarias, que a menudo se consideran menos interpretables que los árboles de decisión.
Las reglas extraídas se pueden utilizar para la validación del modelo, que es una parte importante de los sistemas que tienen como objetivo garantizar el correcto funcionamiento de un modelo determinado.
En (Barbiero et al., 2022) los autores proponen un novedoso enfoque diferenciable de extremo a extremo que permite extraer explicaciones lógicas de las redes neuronales utilizando el formalismo de la lógica de primer orden.
El método se basa en un criterio basado en la entropía que identifica automáticamente los conceptos más relevantes, lo que permite extraer explicaciones lógicas concisas en ámbitos críticos para la seguridad, desde los datos clínicos hasta la visión artificial.

(Wang y Pan, 2020) proponen un marco que integra el conocimiento lógico en forma de lógica de primer orden en un sistema de aprendizaje profundo para la extracción de información. En este trabajo, se demuestra la eficacia y la generalización del modelo propuesto en múltiples tareas de extracción de información, incluido el reconocimiento de entidades nombradas y la extracción de relaciones.
En (Giunchiglia et al., 2022) los autores analizan varios métodos que utilizan conocimientos básicos especificados lógicamente en los modelos de aprendizaje profundo. Estos métodos se clasifican según el lenguaje lógico utilizado para expresar los conocimientos básicos y los objetivos logrados. El trabajo proporciona un estudio exhaustivo del uso de restricciones lógicas en los modelos de aprendizaje profundo y destaca los beneficios potenciales de este enfoque para diversas aplicaciones prácticas.
(Dai y Muggleton, 2021) proponen un método llamado aprendizaje meta-interpretativo abductivo (Meta Abd) que une la abducción y la inducción para aprender redes neuronales e inducir teorías lógicas de forma conjunta a partir de datos sin procesar. El trabajo compara el enfoque propuesto con modelos de aprendizaje profundo de extremo a extremo y métodos neuro-simbólicos de última generación en dos tareas de aprendizaje complejas y demuestra que Meta Abd supera significativamente a los modelos comparados en términos de precisión predictiva y eficiencia de datos.

### Lógica continúa nilpotente

También para la solución de aplicación general, pero para tareas de predicción, en (Csiszár et al., 2020) utilizan lógica difusa nilpotente aumentada con lógica continua a través de la toma decisiones multicriterio.
En (Dombi y Csiszár, 2021) utilizan lógica continua nilpotente combinada con herramientas de decisión multicriterio, para la solución de aplicación general.
En (Dombi y Csiszár, 2021) los autores propone un enfoque híbrido que combina redes neuronales con herramientas de toma de decisiones de lógica continua y multicriterio para mejorar la interpretabilidad y la seguridad del aprendizaje automático. El enfoque propuesto utiliza lógica nilpotente continua y operadores lógicos continuos para modelar desigualdades blandas, es decir, funciones de pertenencia tanto para tareas de clasificación como de predicción.

### Programación Lógica Inductiva

Para realizar predicción de expresiones matemáticas, (Li et al., 2021) utilizan programación lógica inductiva, aumentada con redes bayesianas.
Otra solución de aplicación medica propuesta por (Schmid y Finzel, 2020), para tareas de clasificación, utilizan programación lógica inductiva aumentada con conocimiento del dominio. El objetivo de este trabajo es la detección de tumores a través de un modelo de aprendizaje e Inferencia, partiendo de una base de datos de imágenes.
En (Garcez et al., 2019) los autores plantean la integración basada en principios del aprendizaje neuronal con la representación y el razonamiento simbólicos del conocimiento. Además, se analizan los métodos y principios de la computación simbólica neuronal para el aprendizaje automático y el razonamiento integrados. El uso de diferentes formalismos de representación del conocimiento como conocimiento

básico para un aprendizaje potencial a gran escala y un razonamiento eficiente. El uso de la proposicionalización para aprender la lógica de primer orden en redes bayesianas. El uso de sistemas de incrustación relacional para razonar sobre las relaciones entre entidades, y el uso de redes neuronales profundas para el razonamiento visual, donde aprenden y deducen relaciones y características de múltiples objetos en imágenes.

### Lógica Temporal Lineal

Para aprender actividades de reconocimiento e imitación de secuencias y series temporales aplicadas a robótica, (Xie et al., 2021) aplican lógica temporal lineal (LTL), para tareas de Predicción.

### Lógica Proposicional

(Borges et al., 2006) presentan un *framework* de aplicación general para realizar predicciones, utilizando lógica proposicional.
En (Tran, 2017) los autores propone un método para integrar el conocimiento simbólico en redes neuronales no supervisadas. El método se basa en el hallazgo teórico de que cualquier fórmula proposicional puede representarse en máquinas restringidas de Boltzmann (RBM). El artículo propone un método para integrar el conocimiento simbólico en redes neuronales no supervisadas, que puede ofrecer un mejor aprendizaje y razonamiento, al tiempo que proporciona un medio de interpretabilidad mediante la representación del conocimiento simbólico.

### Lógica de predicados

En (Cai et al., 2017) para realizar predicciones de expresiones matemáticas lógica de predicados, en particular para realizar el descubrimiento de axiomas.

A modo de resumen para la pregunta 3, se presenta en la Fig. 14 la distribución de trabajos encuadrados según el tipo de lógica utilizada. En la Fig. 15 se presenta la distribución de trabajos según las tareas de predicción o clasificación. Por último, en la Fig. 16 se presenta del mismo modo la distribución de trabajos por dominios de aplicación.

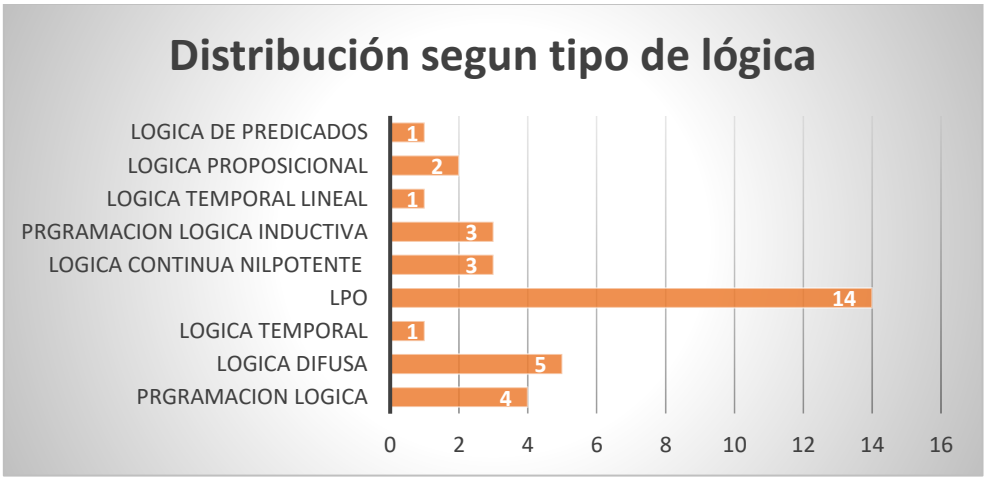


*Figura 14. Distribución de trabajos por tipo de lógica utilizada.*

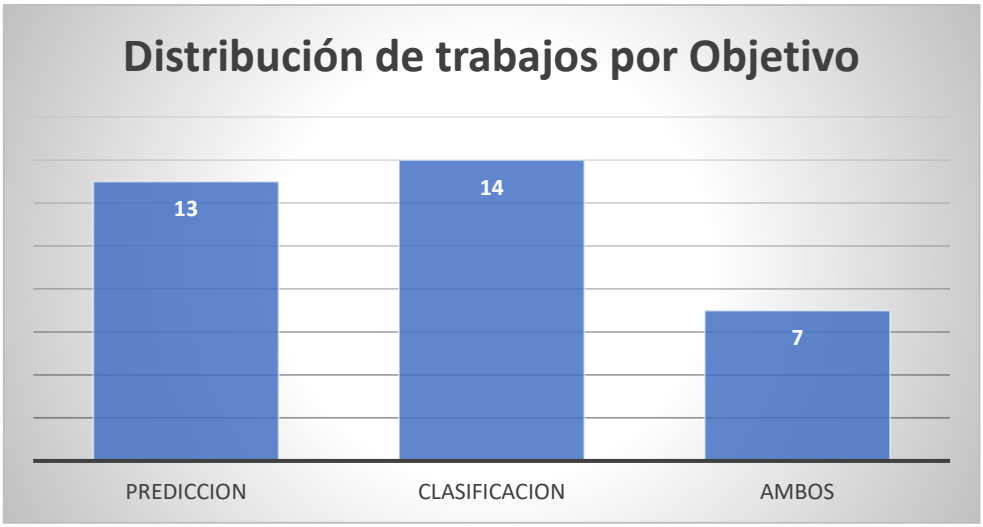


*Figura 15. Distribución de trabajos por objetivo.*

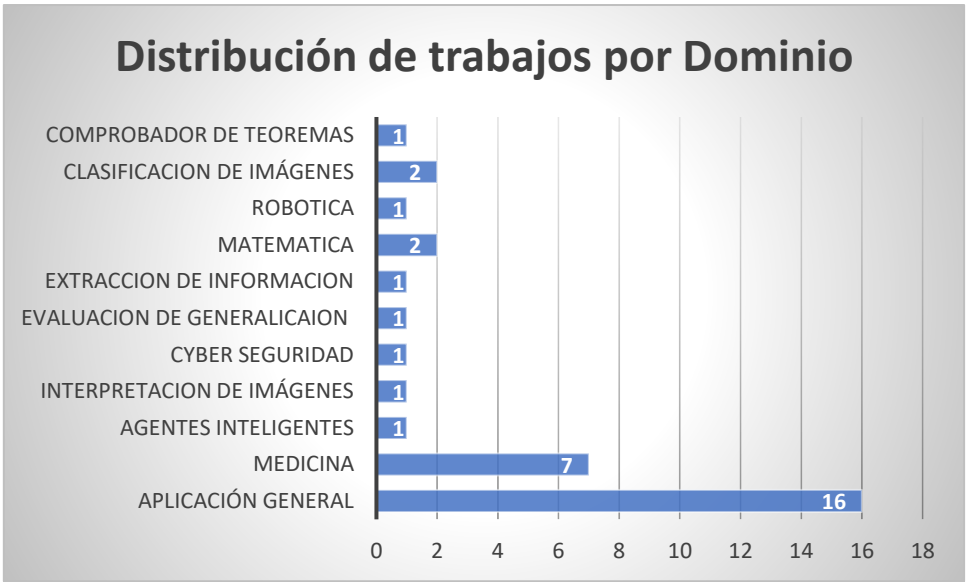


*Figura 16. Distribución de trabajos por dominio de aplicación*.

## 4.4. Limitaciones del estudio

En la siguiente sección se presentan las limitaciones del estudio, y se introduce una indicación sobre el trabajo a futuro.

Al aplicar las cadenas de búsqueda, los motores de búsqueda han listado artículos con contenidos relevantes para la presente revisión, pero estos no estaban disponibles para una lectura y análisis en profundidad, los cuales debieron ser excluidos.
Dada su naturaleza de no ser difundida de forma habitual mediante la publicación, sino a través de canales limitados y de difícil acceso a ellas, los artículos encuadrados como bibliografía gris pueden haber quedado fuera del alcance de la revisión al no ser encontrados por los motores de búsqueda.
Presencia de publicación segmentada, la cual es una forma distinta de publicación redundante que generalmente se caracteriza por la similitud de hipótesis, metodología o resultados, pero no por la similitud del texto. Estos aspectos de las publicaciones no

son detectados objetivamente por las aplicaciones de software y, por lo tanto, representan una seria amenaza para el análisis de la revisión.
Como idea de trabajo a futuro, se ha comenzado a desarrollar una herramienta de búsqueda y análisis semántico de artículos, a través de la cual, con una clasificación inicial mínima entre artículos relevantes e irrelevantes sobre una temática dada, y el uso de un clasificador del tipo *few shots learning*, la herramienta sea capaz de identificar artículos relevantes para el análisis dentro de un resultado de búsqueda mucho mayor.

## 4.5. Discusión

Se han analizado 34 artículos seleccionados que abordan temas de IA no-simbólica aumentada con IA simbólica sobre ecosistemas de *software*, en diferentes dominios de aplicación.
Es muy interesante observar como las soluciones resultantes se vuelven una herramienta de asistencia que, en algunos casos, se alimenta e interactúa con especialistas del dominio logrando mejorar a partir de esta interacción. Es importante destacar que los artículos analizados en la presente revisión presentan combinaciones de técnicas que propenden a mejorar los rendimientos de los sistemas resultantes.
Haciendo un análisis de las tendencias de elecciones tecnológicas, observado desde el punto de vista de las falencias de la IA no-simbólica abordadas en la pregunta PI1, se desprende el siguiente análisis:
Al abordar el problema de ***Falta de explicabilidad*** (38,2% de los casos), la preferencia de los investigadores es por la utilización de *Logica de primer orden y Logica continua nilpotente* (23,1% para cada caso), normalmente aumentada con sistemas expertos o utilización de conocimiento del dominio. La tendencia sigue con la utilización *Lógica difusa* (15,4% de los casos), aumentada con *Algoritmos Evolutivos*. Con la misma participación que la lógica difusa, aparecen en el ranking de preferencia la *Programación Lógica,* aumentada con sistemas expertos *y Programación Lógica Inductiva* aumentada con la utilización de conocimiento del dominio, ambas con un 15,4%. Por último, aparece el uso de *Lógica Proposicional* solo en el 7,7% de los casos.
La tendencia de los trabajos respecto a los objetivos es hacia tareas de *clasificación* (46,1%), mientras que solo el 7,7% hace foco en tareas de *predicción*. Por utlimo, y con una participación igual del 46,1% aparecen trabajos que abordan ambas categorías.
Finalmente, para esta deficiencia, el análisis muestra que el 38,5% de los estudios, busca *encontrar los híper-parámetros óptimos*, el 15,4% se enfoca en *encontrar las características,* mientras que el 46,2% se enfoca en la *extracción de reglas.* Todos los trabajos que utilizan *Programación Lógica Inductiva y Lógica Difusa* se enfocan en tareas de Clasificación, mientras que solo un caso en que se utilizó *lógica difusa* se utilizó para ambas categorías. Para los trabajos que implementan *Lógica continua nilpotente*, la tendencia es 33% predicción, 33% clasificación y 33% para ambas categorías. Para el caso de la *lógica de primer orden*, la tendencia es hacia el abordaje ambas categorías (67% de los casos), mientras que el 33% restante solo abordo tareas de clasificación. En los casos donde se prefiere *lógica proposicional*, el abordaje es en ambas categorías en su totalidad.
Al abordar el problema de ***Falta de generalización*** *(23,5% de los casos),* el 62,5% de los casos aplica *LPO*, el 12,5% utiliza *Programación Lógica Inductiva,* el 12,5% utilizo *Lógica Difusa* aumentada con *Lógica difusa tipo 2,* y el 12,5% restante utilizo *programación*

*lógica*. La tendencia respecto a los objetivos presenta una distribución más o menos homogénea, dado que el 50% de los trabajos dirigió sus esfuerzos hacia tareas de *clasificación*, mientras que el 37,5% lo hizo orientado a taras de *predicción y el 12,5%* restante se enfocó en ambas categorías. De los trabajos que utilizaron LPO como herramienta base, el 37,5% enfoco su análisis en tareas de *clasificación*, mientras que el 12,5% restante lo enfoco hacia *predicción*, y el 12,5% restante se enfocó en ambas categorías.

Se observo una alta correlación entre los trabajos que utilizan LPO los cuales se centran en *encontrar las características del modelo* en un 75%, mientras que el 12,5% de los trabajos se centraron en *extracción de reglas y el 12,5%* restantes se centraron en *mejorar los datos de entrenamiento.* Los trabajos que aplican *Programación Lógica inductiva* se concentran en *encontrar las características* del modelo, mientras que los trabajaos que utilizaron *Programación Lógica* y *Lógica Difusa* se enfocaron en *mejorar los datos de entrenamiento*. El 80% de los trabajos que utilizo *LPO*, baso su trabajo en *encontrar las características* del modelo, mientras que el restante 20% se focalizo en *extracción de reglas*.

Para los trabajos que se utilizó *Lógica Difusa*, los autores se enfocaron en tareas de *clasificación* con la finalidad de encontrar las características del modelo.

Los trabajos que utilizaron *programación lógica Inductiva* centraron sus esfuerzos en encontrar las características del modelo, sobre tareas de predicción.

Los trabajos que utilizaron *programación lógica* centraron sus esfuerzos en mejorar los datos de entrenamiento, sobre tareas de predicción.

Finalmente, los trabajos que utilizaron *LPO* el 60%, orientaron el análisis hacia tareas de clasificación tanto para *encontrar las características* del modelo como para *extracción de reglas*. El 15% de los trabajos se basó en tareas de *predicción* para *encontrar las características,* mientras que el restante 15% se basó en *ambas* tareas también para *encontrar las características del modelo.*

Al abordar el problema de ***Falta de razonamiento*** *(17,6% de los casos)***,** la preferencia es hacia el uso de *lógica de primer orden* (66,6% de los casos), En el trabajo que utilizo LPO como herramienta base, se aumentó el modelo con *Aprendizaje Meta Interpretativo*, para resolver tareas de *clasificación* donde proponen *mejorar los datos de entrenamiento*. El trabajo que implemento *Programación Lógica* (16,7% de los casos), se enfocó más en encontrar los *híper-parámetros óptimos del modelo* también para tareas de *clasificación,* mientras que el trabajo que utilizo *Lógica de Predicados* (16,7% de los casos), como herramienta de razonamiento, se enfocó en la *aceleración del aprendizaje* en tareas de *predicción*. Esta es quizás el área donde menos trabajos se han realizado.

Al abordar el problema de ***Ineficiencia de los datos*** *(29,4% de los casos)***,** se encontró que el 50% de los trabajos utilizaron LPO como herramienta base de razonamiento, y solo un caso aumento el modelo combinando *LPO* con *Redes Bayesianas.* La distribución respecto al objetivo es más o menos homogénea, dado que el 40% enfoco sus esfuerzos en tareas de *clasificación*, mientras que el otro 60% lo dirigió a tareas de *predicción*.

En relación con la aplicación de LPO, el 40% de los trabajos se enfocó en mejorar los datos de entrenamiento, el 20% en *encontrar las características*, otro 20% de los estudios se centró en *acelerar el aprendizaje,* y el restante 20% se enfocó en la *extracción de reglas.*

El 50% de los trabajos analizados utilizaron *lógica de primer orden,* aumentadas con *redes bayesianas* en un solo caso. El 40% de los estudios se enfocó en el *mejoramiento*

*de los datos de entrenamiento*, mientras que el 60% restante se enfocó en *acelerar el aprendizaje, encontrar las características y extracción de reglas*, dividido en partes iguales.
El 20% de los autores utilizaron *lógica difusa* combinada con *algoritmos genéticos* y lógica difusa de tipo 2. El 50% de los trabajos se enfocó en *encontrar las características del modelo*, mientras que el otro 50% se centró en *encontrar los híper-parámetros óptimos.* Aquí hay una fuerte correlación con tareas de *predicción*, dado que todos los trabajos se concentraron en ello.
También con un 20% de adopción, la tendencia se mantiene con el uso de *Lógica Temporal Lineal*. Solo un trabajo aumento el modelo lógico utilizando además *Redes Bayesianas*. Algo a destacar en este segmento es que la correlación hacia el objetivo y la aplicación son del 100% para tareas de *predicción* y *mejora de los datos de entrenamiento.*
Un 10% de los trabajos, utilizaron *Lógica Proposicional*, para tareas de *predicción* aplicado a *acelerar el aprendizaje.*

*Es importante destacar que hay trabajos que se enfocan en más de una categoría, con lo cual la suma de las distribuciones de los porcentajes para cada una no resulte en el 100%. En este estudio se trató de ofrecer una mirada los más completa y exhaustiva posible.*
Este capítulo proporciona una base para identificar brechas y oportunidades de investigación en el área de la IA, más precisamente en el mundo de DL. Proporciona un estado global del arte a través de estudios validados.
Finalmente, el análisis de los diferentes estudios sienta las bases para realizar una revisión sistemática de la literatura centrada en soluciones en el campo de la IA aplicada a los ecosistemas de *software*, con especial énfasis en los ecosistemas tecnológicos.

## 4.6. Conclusiones

Dada la popularidad reciente que ha obtenido la IA y el papel potencial que podrían desempeñar las tecnologías basadas en lógica en la ingeniería de sistemas inteligentes complejos, se desprende que la combinación de técnicas, dentro de IA, basadas en diferentes tipos de lógicas que se combinen y potencien las capacidades del DL, serian el camino a seguir.
Las tecnologías resultantes de esta revisión sistemática se analizan y evalúan desde dos perspectivas diferentes – a saber, *Deep Learning* y Lógica.
A través del análisis de los trabajos presentados a lo largo de este estudio, se han visto combinaciones de lo más variadas entre sistemas lógicos, con alguna forma de red neuronal (DNN, GNN, CNN).
Esta investigación logró clasificar, organizar y explicar las diferentes formas en que las deficiencias del DL son abordadas por propuesta basadas en lógica simbólica. El estudio también determinó en qué etapas del proceso de DL dichas propuestas son aplicadas. Y complementariamente, el estudio permitió determinar cuáles son las herramientas de la lógica que se aplican preferentemente, para cada área y cada dominio.
Si bien no se observa un patrón arquitectónico claro, los esfuerzos por encontrar un *modelo de propósito general* que combine ambos mundos dirigen las tendencias y esfuerzos de investigación.

En cuanto a los ecosistemas tecnológicos, si bien este concepto se enmarca en el campo de los ecosistemas de la ingeniería de *software*, presenta diferentes matices que necesitan soluciones enfocadas a la evolución del dominio, y la inclusión del factor humano al mismo nivel que el componente de *software*.
Finalmente, debido al enfoque científico, bien fundamentado y reproducible de esta revisión, se deduce que representa una herramienta confiable para evaluar el estado actual del arte, con la esperanza que pueda usarse para comprender las direcciones futuras en esta área.

# CAPITULO 5 Explicabilidad en Redes Neuronales Profundas

## 5.1. Introducción

La explicabilidad aborda el problema crítico de que los humanos no pueden comprender directamente el comportamiento complejo de las DNN o explicar su proceso de toma de decisiones subyacente. La explicabilidad de los modelos de DL es el requisito fundamental para *generar confianza* con los usuarios y es la clave para su implementación segura, *justa* y exitosa en aplicaciones del mundo real.

El surgimiento de sistemas de decisión opacos y ubicuos, que son sistemas de caja negra y que utilizan modelos de aprendizaje automático para predecir información sensible, ha generado preocupación por la falta de explicación y comprensión de tales sistemas.
El Parlamento Europeo adoptó recientemente el Reglamento General de Protección de Datos (GDPR), que se convirtió en ley en mayo de 2018. Un aspecto innovador del GDPR son las cláusulas sobre datos automatizados, toma de decisiones, incluida la elaboración de perfiles, que introducen por primera vez, hasta cierto punto, un derecho de explicación para que todos los individuos obtengan *explicaciones significativas de la lógica involucrada* cuando se lleva a cabo la toma de decisiones automatizada. A pesar de las opiniones divergentes entre los juristas sobre el alcance real de estas cláusulas, existe un acuerdo general sobre la necesidad de que la implementación de tal principio sea urgente y que represente hoy un enorme desafío científico abierto. Sin una tecnología habilitadora capaz de explicar la lógica de las cajas negras, el derecho a una explicación seguirá siendo una *dead letter* (Guidotti et al., 2019). Al confiar en sofisticados modelos de clasificación de aprendizaje automático entrenados en conjuntos de datos masivos gracias a infraestructuras escalables y de alto rendimiento, se corre el riesgo de crear y utilizar sistemas de decisión que realmente no entendemos. En tal sentido, GDPR ha introducido el derecho a una explicación para las personas afectadas por la toma de decisiones automatizada, destacando la necesidad de tecnologías que puedan explicar la lógica de los sistemas de caja negra. La falta de comprensión y validación de los componentes del aprendizaje automático puede conducir a decisiones equivocadas, violaciones éticas y riesgos de seguridad en diversas industrias, incluidos los vehículos autónomos y la medicina personalizada.

La disponibilidad de tecnologías transparentes de aprendizaje automático puede mejorar la confianza, la conciencia y la responsabilidad en los procesos de toma de decisiones. En esa dirección, la explicación es crucial para una ciencia de datos abierta y responsable y para la investigación científica en diversos ámbitos.
Del mismo modo, el uso de modelos de aprendizaje automático en la investigación científica, por ejemplo, en medicina, biología y ciencias socioeconómicas, requiere una explicación no sólo para la confianza y la aceptación de los resultados, sino también para la apertura del descubrimiento científico y los avances de la investigación. Como

consecuencia, la explicación está en el centro de una ciencia de datos abierta y responsable, en múltiples sectores industriales y disciplinas científicas.
La literatura existente carece de una organización y clasificación sistemática de metodologías para la interpretación de sistemas de caja negra, lo que motiva la necesidad de una clasificación clara considerando diferentes aspectos simultáneamente (Guidotti et al., 2019).

## 5.2. Necesidad de Modelos de IA explicables

Modelos interpretables son necesarios para abordar los peligros y las preocupaciones éticas asociadas con los modelos de caja negra. Las cajas negras pueden provocar discriminación, problemas de confianza y la perpetuación de prejuicios, ya que pueden tomar decisiones basadas en reglas y prejuicios ocultos.
La falta de interpretabilidad en los sistemas de toma de decisiones puede dar lugar a decisiones equivocadas, violaciones éticas y riesgos de seguridad en diversos ámbitos, como los vehículos autónomos y la medicina personalizada.
Los modelos interpretables pueden mejorar la confianza, la conciencia y la responsabilidad en los procesos de toma de decisiones, ya que permiten a los usuarios comprender y validar la lógica detrás de las decisiones tomadas por los modelos.
Las tecnologías transparentes de aprendizaje automático también pueden facilitar la ciencia de datos abiertos y la investigación científica responsable en diversos ámbitos (Guidotti et al., 2019).

La cuestión de la explicabilidad trasciende el ámbito del interés científico. La capacidad de explicar la razón de ser de las decisiones propias a otras personas es un aspecto importante de la inteligencia humana. No solo es importante en las interacciones sociales, sino que también es crucial en el contexto educativo, donde los estudiantes intentan comprender el razonamiento de sus maestros. Además, la explicación de las propias decisiones suele ser una *necesidad previa para establecer una relación de confianza* entre las personas, por ejemplo, cuando un médico explica la decisión terapéutica a su paciente. Aunque estos aspectos sociales pueden ser de menor importancia para los sistemas técnicos de IA, hay muchos argumentos a favor de *explicabilidad* en inteligencia artificial. A continuación, revisamos los más importantes:

**Verificación del sistema:** Como se mencionó anteriormente, en muchas aplicaciones no se debe confiar en un sistema de caja negra por defecto. Por ejemplo, en el cuidado de la salud, el uso de modelos que puedan ser interpretados y verificados por expertos médicos es una necesidad absoluta. Los autores en (Caruana et al., 2015) presentan un ejemplo de este dominio, donde un sistema de IA que fue entrenado para predecir el riesgo de neumonía de una persona llega a conclusiones totalmente erróneas.
La aplicación de este modelo en forma de caja negra no reduciría, sino que aumentaría el número de muertes relacionadas con la neumonía. En definitiva, el modelo aprende que los pacientes asmáticos con problemas cardíacos tienen un riesgo mucho menor de morir de neumonía que las personas sanas. Un médico reconocería de inmediato que esto no puede ser cierto, ya que el asma y los problemas cardíacos son factores que afectan negativamente el pronóstico de recuperación. Sin embargo, el modelo de IA no sabe nada sobre el asma o la neumonía, *solo infiere de los datos*. En este ejemplo, los

datos estaban sistemáticamente sesgados porque, a diferencia de las personas sanas, la mayoría de los pacientes con asma y del corazón estaban bajo estricta supervisión médica. Debido a esa supervisión y la mayor sensibilidad de estos pacientes, este grupo tiene un riesgo significativamente menor de morir de neumonía.
Sin embargo, esta correlación no tiene carácter causal y, por lo tanto, no debe tomarse como base para la decisión sobre el tratamiento de la neumonía.

**Mejora del sistema:** El primer paso para mejorar un sistema de IA es comprender sus debilidades.
Obviamente, es más difícil realizar dicho análisis de debilidad en modelos de caja negra que en modelos que son interpretables. Además, detectar sesgos en el modelo o el conjunto de datos es más fácil si se comprende qué está haciendo el modelo y por qué llega a sus predicciones. La interpretabilidad del modelo puede ser útil al comparar diferentes modelos o arquitecturas. Por ejemplo, se ha observado que los modelos pueden tener el mismo rendimiento de clasificación, pero difieren en gran medida en cuanto a las características que utilizan como base para sus decisiones. Entonces, la identificación del modelo más *apropiado* requiere explicabilidad. **Incluso se puede afirmar que cuanto mejor entendamos lo que están haciendo nuestros modelos (y por qué a veces fallan), más fácil será mejorarlos.**

**Aprender del sistema**: Debido a que los sistemas de IA actuales están entrenados con millones de datos de ejemplo, pueden observar patrones en los datos que no son evidentes para los humanos, quienes solo son capaces de aprender con un número limitado de ejemplos. *Cuando usamos sistemas de IA explicables, podemos intentar extraer este conocimiento destilado del sistema de IA para adquirir nuevos conocimientos*.
Los sistemas de IA identifican nuevas estrategias para jugar *Go*, que seguramente ahora también han sido adaptadas por jugadores humanos profesionales. Otro dominio donde la extracción de información del modelo puede ser crucial son las ciencias. En pocas palabras, los físicos, químicos y biólogos están más interesados en identificar las leyes ocultas de la naturaleza que en predecir alguna cantidad con modelos de caja negra. Por lo tanto, solo los modelos explicables son útiles en este dominio.

**Cumplimiento de la legislación**: Los sistemas de IA están afectando cada vez más áreas de nuestra vida cotidiana, y con eso también los aspectos legales. Por ejemplo, la asignación de responsabilidad cuando los sistemas toman una decisión equivocada ha recibido recientemente una mayor atención. Dado que puede ser imposible encontrar respuestas satisfactorias para estas preguntas legales cuando se confía en modelos de caja negra, los futuros sistemas de IA necesariamente tendrán que ser más explicables. Otro ejemplo en el que las regulaciones pueden convertirse en una fuerza impulsora para una mayor explicabilidad en inteligencia artificial, son los derechos individuales. Las personas inmediatamente afectadas por las decisiones de un sistema de IA (por ejemplo, personas rechazadas por el banco para un préstamo) pueden exigir saber por qué el sistema ha decidido de esta manera. Solo los sistemas de IA explicables proporcionarán esta información. Estas preocupaciones llevaron a la Unión Europea a adaptar nuevas regulaciones que implementan un *derecho a la explicación* por el cual

un usuario puede solicitar una explicación de una decisión algorítmica que se tomó en cada caso (Goodman y Flaxman, 2017).
La adopción del Reglamento General de Protección de Datos (GDPR) por parte de la Unión Europea otorga a cualquier ciudadano el *derecho a la explicación* de una decisión algorítmica tomada sobre él. El RGPD establece que las personas *tienen derecho a no ser objeto de una decisión basada únicamente en el tratamiento automatizado*.
En tal sentido, la explicabilidad es tanto un derecho legal como una responsabilidad que tiene implicaciones sociales amplias.

### 5.2.1. Responsables por Diseño

El uso y escalamiento de la IA conlleva riesgos asociados para las personas, la sociedad. La IA responsable (RAI) por diseño y por defecto ayuda a establecer los principios, la gobernanza, las políticas y el control, la tecnología y la cultura apropiados para mitigar esos riesgos. Al hacerlo, se evita el daño a la reputación y el riesgo de cumplimiento, pero también creamos valor adicional (confianza) al diferenciar nuestro enfoque como uno altamente responsable y ético.

Ser responsables por diseño es de suma importancia dado que se pone a la persona humana en el centro de la discusión y, de este modo, se construye una reputación positiva entre los distintos participantes involucrados en el proceso. Para ser *responsables por diseño*, las organizaciones deben pasar de una **estrategia de cumplimiento reactivo al desarrollo proactivo** de capacidades maduras de IA responsable. Con las bases establecidas para respaldar el uso responsable de la IA en toda la empresa, se vuelve más fácil adaptarse a medida que surgen nuevas regulaciones.

El objetivo de crear e implementar soluciones de IA que tengan en cuenta el valor para todos los humanos, significa que se seguirán estos principios al vivir nuestro propósito y cumplir la promesa de la tecnología y el ingenio humano.

El objetivo entonces es crear e implementar soluciones de IA que sean:

**Centradas en los humanos y justos**: Incluyendo conjuntos de datos representativos y equipos inclusivos para mitigar el potencial de sesgo injusto y otras consecuencias negativas no deseadas, junto con la promoción de nuestros objetivos de diversidad, derechos humanos y sostenibilidad. La práctica de producir resultados razonables y justos para las personas que están sujetas al algoritmo de IA.

**Confiables y Transparentes:** Asegurando la visibilidad de la gestión de datos, *maximizando la capacidad de explicación*, la solidez y la precisión y permitiendo una comprensión clara de los resultados impulsados por la IA y la toma de decisiones informada.
La capacidad de un sistema de IA para funcionar de manera confiable y precisa en condiciones adversas, para generar confianza entre todas las partes interesadas. Desarrollar IA explicable que sea transparente en todos los procesos y funciones

agregando contexto a los algoritmos, construidos con datos relevantes y de alta calidad, que conduce a una IA confiable.

**Seguros y Protegidos:** Definir la seguridad, la privacidad y la protección desde el diseño para proteger tanto los datos como a los humanos, y capturar, seleccionar y procesar los datos de manera responsable, además de comprender la idoneidad contextual y defender los derechos de uso.
Fomentar un enfoque que priorice la privacidad y la seguridad para garantizar que los datos personales y/o confidenciales se procesen de manera adecuada y legal.

**Abierta y Responsable:** Adoptar prácticas de datos e AI responsables para defender la rendición de cuentas, la trazabilidad y el monitoreo continuo, mediante la promoción del aprendizaje y la mejora continua, una gobernanza más sólida.
Establece estructuras de gobierno transparentes y entre dominios, identificando roles, expectativas y responsabilidad para generar confianza interna y confianza en las tecnologías de IA.

En conclusión, una IA responsable es crucial en el panorama tecnológico que avanza rápidamente. Con el uso y escalamiento cada vez mayores de la IA, es imperativo que las organizaciones prioricen e implementen prácticas responsables. Esto incluye establecer una gobernanza, políticas y controles claros, desarrollar tecnologías y herramientas para respaldar la equidad, la explicabilidad y la privacidad, y crear una cultura de capacitación y educación en torno a la IA responsable. Al ser responsables por diseño, las organizaciones no sólo mitigan los riesgos, sino que también generan confianza y reputación entre todas las partes interesadas. Al colocar a los humanos en el centro del desarrollo de la IA y considerar la justicia, la transparencia, la seguridad y la responsabilidad, podremos crear soluciones de IA que aporten valor a todas las personas y promuevan la diversidad, los derechos humanos y la sostenibilidad. Es esencial que las organizaciones prioricen la IA responsable para adelantarse a las regulaciones y ganarse la confianza de la sociedad.

## 5.3. Líneas de Investigación

En los trabajos presentados en (Nielsen et al., 2022), (Guidotti et al., 2019) se destacan 4 líneas de investigación principales derivadas de los trabajos de investigación presentes en la literatura, donde se aborda desde diferentes enfoques el problema de explicabilidad. Estas líneas se presentan y describe a continuación:

- Metaheurísticas
- Enfoque dirigido por datos
- Explicaciones Locales
- Explicaciones Globales

### 5.3.1. Metaheuristicas

Un predictor de caja negra es un modelo de aprendizaje automático que se caracteriza por su complejidad y falta de transparencia. Esto significa que sus componentes internos son desconocidos para el observador o conocidos, pero no comprensibles para los humanos. En otras palabras, es difícil entender cómo este modelo toma decisiones basándose en su funcionamiento interno. Para facilitar la comprensión, una explicación sirve como una *interfaz* entre el modelo y los humanos. Por ejemplo, en el campo de la salud, un modelo de caja negra podría ser utilizado por un médico para tomar decisiones sobre un paciente, y a su vez interpretaría la explicación proporcionada por el modelo para comunicarla al paciente de una manera comprensible.
Otro aspecto relevante es la *razón* por la cual es necesaria una explicación: se puede requerir un modelo interpretable ya sea para revelar hallazgos en datos que expliquen la decisión, o para explicar cómo funciona la propia caja negra.
Los modelos interpretables se refieren a modelos que los humanos pueden entender y explicar fácilmente. Estos modelos tienen un funcionamiento interno y procesos de toma de decisiones transparentes, lo que permite a los usuarios obtener información sobre cómo el modelo llega a sus predicciones (Guidotti et al., 2019).
La interpretabilidad se define como la capacidad de asignar un significado comprensible para los humanos sobre las predicciones de un modelo. La interpretabilidad es una característica *pasiva* de un modelo que puede referirse al nivel en el que un modelo dado tiene sentido para un ser humano.
La idea detrás del desarrollo de metaheurísticas es entrenar un modelo construido para ser inherentemente interpretable en primer lugar, por ejemplo, modelos lineales o árboles de decisión, con el objetivo de lograr la misma capacidad predictiva que la DNN o modelo de caja negra, y luego extraer la lógica derivada del modelo interpretable.
Como contrapartida, estas técnicas podrían ser impracticables para DNN grandes dada la complejidad del árbol de decisiones. Además, al no interactuar con el modelo de caja negra (DNN), no tienen una relación directa.

### 5.3.2. Enfoque dirigido por datos

Desde la perspectiva de análisis para esta línea de estudio y como se analizó en el capítulo 2, la iniciativa es la de *cambiar de una vista centrada en el modelo a una vista más centrada en los datos*.
Desde esta perspectiva, se considera que la calidad de los datos es primordial y se pueden utilizar herramientas como el análisis de errores o el *aumento de datos (data augmentation)* para mejorar sistemáticamente la calidad de los datos.
Para muchas aplicaciones, si los datos son lo suficientemente buenos, hay múltiples modelos que funcionaran correctamente. Bajo esta premisa, *podemos mantener fijo el código y mejorar iterativamente los datos*.
Dado que siempre será posible obtener más datos, parte del desafío radica en analizar qué datos recopilar. Esto puede ayudar a ser mucho más eficientes en la construcción de un modelo más preciso.
Claramente, en línea de investigación la calidad de los datos es el factor clave, donde se mejora la calidad de los datos por sobre la calidad del modelo. La premise central en

esta línea de estudio sostiene que, si los modelos pueden aprender y generalizar desde los datos, entonces los datos deben ser capaces de explicarse.
En conclusión, se pueden inferir reglas a partir del cálculo de estadísticas sobre los datos, y del análisis de perturbación en los datos de entrada al modelo y posterior análisis de como estas perturbaciones impactan en las respuestas del modelo (Nielsen et al., 2022), (Guidotti et al., 2019).

### 5.3.3. Explicaciones Locales

El problema de interpretabilidad local se puede formular como la estimación de un número para cada característica de entrada, que captura el *efecto* del cambio en el valor de la característica en la salida de la red. En el caso de análisis de imágenes, los números estimados se presentan como mapas de calor y tienen la misma dimensión que las entidades de entrada.

Los métodos de interpretabilidad local intentan explicar decisiones específicas, es decir, qué características de la entrada (p. ej., píxeles de una imagen) pueden haber contribuido (positiva o negativamente) a la salida del modelo (Nielsen et al., 2022).
Se indica entonces *interpretabilidad local* a la situación en la que es posible comprender sólo las razones de una decisión específica: sólo la predicción/decisión única es interpretable (Guidotti et al., 2019).

Dada una red neuronal entrenada, las características de entrada se perturban y se monitorea su efecto en la salida de la red. Una señal de la salida de la red se retro propaga a la entrada, en tal sentido la información resultante de las perturbaciones o la propagación del gradiente proporciona una estimación de la contribución de las características de entrada a la salida y puede presentarse como mapas de calor.

Existen dos grandes categorías:

**Perturbación de *Features*:**

- Este método perturba las características de entrada (o un conjunto de características) enmascarando o alterando sus valores y registra el efecto de estos cambios en el rendimiento de la red.
- Esto *requiere varias pasadas* a través de la red para determinar la importancia[6] de cada *feature* de entrada, lo que hace que la atribución basada en perturbaciones sea **muy intensiva desde el punto de vista computacional**.

Dentro de las Técnicas de transparencia y explicabilidad en ML las permutaciones de *features* son importantes para oobtener **información sobre qué características tienen el mayor impacto** en el modelo. Además, **mide el valor predictivo de una característica** para cualquier estimador, clasificador o regresor de caja negra. Esto se logra **evaluando**

[6] En la literatura, los términos *atribución, relevancia, importancia, contribución, sensibilidad* y *puntajes de prominencia* se usan como sinónimos.

**cómo aumenta el error de predicción cuando un *feature* no está disponible**. Cuanto mayor sea la diferencia entre los errores de predicción, más importante será la característica. Se puede utilizar cualquier métrica de puntuación para medir el error de predicción.

**Información del Gradiente**

- En estos métodos, los gradientes de la salida (*logits* o probabilidades *soft-max*), con respecto a las características extraídas o la entrada, se calculan mediante retro propagación y se utilizan para estimar las puntuaciones de atribución.
- Generalmente, los gradientes son ruidosos, lo que lleva a mapas de atribución que pueden mostrar contribuciones de características irrelevantes.
- Los enfoques basados en gradientes *no miden directamente* el efecto de perturbar las características de entrada.

Una de las técnicas más populares respecto a la perturbación de *features* es el método *Layer Wise Relevance Propagation,* el cual se basa en el algoritmo de expansión de primer orden de Taylor (Montavon et al., 2017).
En este caso, dada una imagen a la entrada de la DNN la señal de salida se retropropaga hacia la entrada utilizando una función de relevancia hasta formar un mapa de calor, el cual indica cuales pixeles fueron los más relevantes y en los cuales se basó la red para entregar una estimación dada.

### 5.3.4. Explicaciones Globales

Los métodos de interpretabilidad global intentan explicar el proceso general de toma de decisiones del modelo, es decir, cómo las entradas se transforman en decisiones de salida a nivel del modelo (Nielsen et al., 2022). Un modelo puede ser completamente interpretable, es decir, podemos comprender toda la lógica de un modelo y seguir todo el razonamiento que conduce a los diferentes resultados posibles. En este caso, estamos hablando de interpretabilidad global (Guidotti et al., 2019).
Estos pueden ser más útiles para los investigadores e ingenieros que intentan comprender sus modelos. Explicar un modelo en un solo punto y luego generalizar a todo el conjunto de datos es una pregunta abierta para la comunidad investigadora.

El problema de explicación para modelos de caja negra consiste en proporcionar una explicación clara y comprensible del modelo. La tarea de extracción de reglas implica usar la información codificada en la arquitectura de la red, la función de activación y los pesos y sesgos de las neuronas para extraer un conjunto de reglas. Esto se puede lograr mediante métodos de búsqueda que buscan combinaciones de valores de entrada que produzcan una activación cercana a 1 (para una regla de confirmación) o una activación cercana a 0 (para una regla de rechazo). El objetivo es proporcionar una explicación global del modelo a través de un método interpretable y transparente, que permita extraer reglas de aprendizaje de la DNN y explicarlas en un lenguaje comprensible para los humanos.

## 5.4. Explicabilidad en DNN mediante extracción de reglas

Las tareas de extracción de reglas pueden verse como una tarea de búsqueda o como una tarea de aprendizaje. En una DNN entrenada, el conocimiento adquirido en la fase de entrenamiento está codificado en la arquitectura de la red, la función de activación utilizada y los pesos y sesgos (*bias*) de las neuronas. La tarea de la extracción de reglas es usar una o más de estas piezas de información para extraer un conjunto de reglas de las neuronas.
Los métodos de búsqueda para extraer reglas intentan encontrar combinaciones de los valores de entrada a una neurona que dan como resultado que tenga una activación cercana a 1 (para una regla de confirmación) o una activación cercana a 0 (para una regla de rechazo). En tal sentido, el problema de explicación para modelos de caja negra consiste en proporcionar una explicación global del modelo a través de un método interpretable y transparente. Los resultados de este método deberían poder extraer reglas de aprendizaje de la DNN, y también deberían poder explicar estas reglas en un lenguaje comprensible para los humanos.

En el presente trabajo de tesis se propone un método de análisis basado en la extracción de reglas *post-hoc* sobre redes neuronales profundas *feedforward* entrenadas, con el objetivo de obtener explicaciones globales. Esta técnica se centra en la identificación de patrones y relaciones entre los atributos de entrada y la salida de la red neuronal, permitiendo una comprensión más profunda de los procesos subyacentes.

CAPITULO 6

# El Algoritmo COLOSSUS

## 6.1. Introducción

*Todo el conocimiento, pasado, presente y futuro, puede derivarse de datos mediante un algoritmo de aprendizaje único y universal* (Domingos, 2018).

En el capítulo 4 se han analizado numerosas propuestas en pos de integrar la inteligencia artificial simbólica y no simbólica. Posteriormente, en el capítulo 5 se ha puesto el foco sobre el problema de la explicabilidad de la inteligencia artificial no-simbólica, especialmente, las redes neuronales artificiales.
En base a este análisis, en el presente capítulo se presenta un método para extraer el patrón de reglas aprendido de una red neuronal de *feedforward* entrenada y analizar sus propiedades. El método aplica la lógica simbólica para explicar el proceso de toma de decisiones del modelo no-simbólico.
La estrategia propuesta se basa en utilizar el análisis de correlación entre los vectores de pesos y las salidas de cada capa, aplicando la métrica de similitud coseno, para identificar los caminos críticos que conectan las entradas y los resultados en la red neuronal. Las reglas extraídas se escriben en términos de lógica proposicional y lógica de primer orden para explicar la relación entrada-salida.

El uso de la lógica de primer orden para explicar las DNN implica representar la red neuronal como un conjunto de proposiciones y reglas lógicas. Este enfoque puede ayudar a identificar la estructura y las relaciones subyacentes dentro de la red, así como a explicar cómo la red toma decisiones en función de sus entradas.
La similitud del coseno para comparar los vectores de pesos en una DNN es un enfoque válido y es una métrica de uso común para medir la similitud entre dos vectores en el aprendizaje automático y el procesamiento del lenguaje natural. Esta técnica mide el coseno del ángulo entre dos vectores, que va de -1 a 1, donde 1 indica que los vectores son idénticos y 0 indica que son ortogonales.
Un posible enfoque para utilizar lógica de primer orden para explicar las DNN es representar los pesos de la red como proposiciones lógicas y usar la similitud coseno para comparar la similitud entre los vectores de pesos de diferentes modelos o en diferentes puntos en el tiempo durante el entrenamiento. Esto puede ayudar a identificar patrones y tendencias en los pesos y monitorear el progreso del entrenamiento. Estas técnicas pueden ayudar a identificar las reglas y patrones que utiliza la red para tomar decisiones y proporcionar una indicación sobre cómo explicar tales decisiones basadas en datos.

En el contexto de explicar las decisiones tomadas por las DNN, los vectores de pesos de la red neuronal se pueden comparar utilizando similitud coseno, para identificar las entradas que mejor explican las salidas en cada capa neuronal y, de este modo, trazar el camino critico que sigue la red para tomar decisiones. Entonces, si podemos identificar tal camino critico neuronal para cada capa de la red, deberíamos ser capaces de explicar esta función

a través del uso de lógica de primer orden, y extraer reglas que expliquen los datos en las entradas. Al definir un conjunto de reglas lógicas que capturan el comportamiento de la red, se podría razonar sobre el comportamiento de la red mediante inferencia lógica. Por ejemplo, dado un conjunto de valores de entrada, es posible utilizar la inferencia lógica para determinar la salida de la red.

Es importante tener en cuenta que el uso de reglas lógicas para representar el comportamiento de las DNN puede ser un desafío, ya que el comportamiento de estas redes suele ser muy complejo y difícil de capturar usando reglas lógicas. Además, el uso de la inferencia lógica para razonar sobre el comportamiento de las DNN puede resultar costoso desde el punto de vista computacional, especialmente para redes grandes.

Investigaciones recientes sobre la comprensión del funcionamiento de una red neuronal entrenada se ha centrado en la extracción de reglas simbólicas (AmirHosseini y Hosseini, 2019); Csiszár et al., 2020; Mahdavifar y Ghorbani, 2020).

## 6.2. Seudocódigo para el algoritmo COLOSSUS

En esta sección se presenta el seudocódigo del algoritmo propuesto. En las secciones siguientes, cada paso se analiza y explica en detalle.

**Input**
DS = Un DataSet
DNN = Red Neuronal Entrenada
Umbral

**Output**
reglas = conjunto de reglas en forma de fbf, en formato de lógica proposicional.
explicación = conjunto de fbf en formato de lógica de predicados.

**Seudocódigo**
**# Paso 1: Calcular estadísticas para DS**
1.1 Para el conjunto de datos de entrada calcular estadísticas Max, Min, Media, Perc25, Perc50, Perc75.
1.2 Identificar las estadísticas que maximizan la precisión predictiva de la red por clase.
1.3 Determinar cotas para cada *feature* de entrada crítico con las estadísticas del punto

**# Paso 2: Calcular la similitud del coseno**
2.1 Para cada capa L en DNN
2.2 Para cada Neurona en la capa L
2.3 if cosine_similarity($Z^{[L]}_n$, $T^{[L]}_n$) > umbral
2.4 critical_neurons_set.append($L^{[L]}_n$)
2.5 Para cada *nc* en critical_neurons_set:
2.6 Para cada *ncn* en non_critical_neurons_set
2.7 If cosine_similarity(weights_$nc^{[L]}_n$, weights_$ncn^{[L]}_n$) > umbral
2.8 seccondary_neurons_set.append($ncn^{[L]}_n$)

**# Paso 3: Calcular relaciones conjuntivas**
3.1 Para cada capa L en DNN
*#Obtener una única tabla de verdad por capa.*

3.2 $TV^{[L]}$ = calcular_conjunciones(critical_neurons_set)

**# Paso 4: Calcular las relaciones de implicancia**

4.1 Taut = Calcular_implicancias (TVDnn) = ($TV^{[l-1]} \rightarrow TV^{[l]} \rightarrow TV^{[l+1]}$ = TVdnn)

#Escribir las reglas obtenidas en términos de fórmulas bien formadas (fbf) de #lógica proposicional para explicar la función aprendida por la DNN. Relación #entrada-salida.

**# Paso 5: Calcular el factor MNN**

5.1 MNN = get_MNN (critical_neurons_set ($A^{[0]}$ ) )
5.2 nAllowed = MNN - len (critical_neurons_set ($A^{[0]}$ )
5.3 do while nAllowed > 0
5.4 etpL = []
5.5 If seccondary_neurons_set != ø
5.6 etpL = get_entropy (seccondary_neurons_set ($A^{[0]}$ )
5.7 else
5.8 etpL = get_entropy (non_critical_neurons ($A^{[0]}$ )
5.9 critical_neurons_set.append( get_neurons (etpL, nAllowed))
5.10 nAllowed = MNN - len (critical_neurons_set ($A^{[0]}$ ) )

**# Paso 6: Mapear neuronas críticas**

*Mapear las neuronas identificadas como críticas en critical_neurons_set para $A^{[0]}$ con las cotas de las estadísticas obtenidos en el primer paso para determinar la precisión de estas reglas.*

6.1 Calcular Max, Min Para todo *feature* F en critical_neurons_set
6.2 Para cada *item* dentro del dataset:
6.3 $\forall$ F[item] in critical_neurons_set($A^{[0]}$ )
6.4 If $F[item]_n \subset$ [classN_max[F[item]] .. classN_min[F[item]]
6.5 ClassN.append( classN )

*# Resolver overlapping entre clases utilizando la función de distancia.*
6.6 If len(ClassN) > 1
*#calcular distancia entre feature en comun (fcc), con el de mayor entropia (fme).*
6.7 Class0 = (fme – MinVal_Class0(fcc)) + (MaxVal_Class0(fcc) - fme)
6.8 Class1 = (fme – MinVal_Class1(fcc)) + (MaxVal_Class1(fcc) - fme)
6.9 Class = (1 if dst1 < dst0 else 0)
6.10 Validar Precision: Ground Truth Vs. Class

**# Paso 7: Obtener fórmulas lógicas de primer orden**

7.1 Obtener fórmulas lógicas de primer orden que expliquen los datos de entrada.

## 6.3. COLOSSUS: Extracción de reglas con lógica de primer orden para redes neuronales profundas *feedforward* entrenadas.

COLOSSUS significa *Extracción de reglas basada en lógica cósmica para comprender estructuras estadísticamente*, por sus siglas en inglés, lo que refleja el uso de similitud de coseno y métodos estadísticos en este algoritmo.
El algoritmo COLOSSUS es un método de extracción de reglas diseñado con el objetivo de extraer reglas lógicas de redes neuronales profundas *feedforward* entrenadas (DNN), combinando lógica de primer orden, similitud de cosenos y estadística. Las DNN son modelos potentes y complejos que han demostrado un gran éxito en diversos campos, pero su falta de interpretabilidad y transparencia sigue siendo un desafío importante. COLOSSUS aborda este problema proporcionando un enfoque sistemático para extraer reglas comprensibles e interpretables de las DNN.
La lógica de primer orden es la columna vertebral de este algoritmo, ya que permite la traducción de las representaciones aprendidas del DNN en declaraciones lógicas. Al asignar los pesos y activaciones de las neuronas en las capas ocultas a variables y predicados lógicos, el algoritmo crea una representación basada en reglas del proceso de toma de decisiones de la DNN.
La similitud del coseno se utiliza para identificar y agrupar neuronas similares, en función de sus representaciones aprendidas. Esta agrupación permite que el algoritmo extraiga reglas más generales, en lugar de reglas individualizadas para cada neurona. Al considerar la similitud entre neuronas, el algoritmo puede capturar las características importantes de los datos de entrada que contribuyen a la decisión final de la DNN.
Por último, el algoritmo utiliza estadísticas para refinar las reglas extraídas. Evalúa la frecuencia y significado de cada regla para determinar su relevancia e importancia en el proceso de toma de decisiones de la DNN. Este paso garantiza que solo se extraigan las reglas más relevantes y significativas, lo que lleva a un conjunto de reglas más conciso e interpretable.
En general, el algoritmo COLOSSUS combina las potentes capacidades de representación de la lógica de primer orden, la capacidad de capturar patrones generales mediante similitud de cosenos y el refinamiento de reglas mediante estadísticas para extraer reglas interpretables de DNN. Esto permite una mejor comprensión del proceso de toma de decisiones de las DNN y puede ayudar a mejorar su rendimiento y confiabilidad.

## 6.4. Extracción de reglas en redes neuronales

La tarea de extracción de reglas puede verse como una tarea de búsqueda o como una tarea de aprendizaje (Montavon et al., 2017), donde para el enfoque de búsqueda, las reglas se extraen a nivel de las neuronas individuales (ocultas y de salida) en la red, observando sus pesos y sesgos (Krishnan et al., 1999). En tal sentido, uno de los principales problemas con el enfoque de extracción de reglas es cómo restringir el espacio de búsqueda para las posibles combinaciones de pesos y sesgos.
En una red neuronal entrenada, el conocimiento adquirido en la fase de entrenamiento está codificado en la arquitectura de la red, las funciones de activación utilizadas y los pesos y sesgos de las neuronas.
En el presente trabajo, la tarea de extracción de reglas consiste en analizar los vectores de pesos generados en cada neurona, junto con el vector de sesgos para cada capa de

la red, y extraer un conjunto de reglas de las neuronas que puedan ser explicadas con lógica de primer orden. Consideramos los casos donde las entradas a la red *feedforward* son tabulares y las salidas son categóricas (es decir, problemas de clasificación).

El método de propagación de las neuronas en las redes neuronales *feedforward* viene definido por la siguiente forma canónica:

$$Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]} \quad \text{eq.6.1}$$

$$A^{[l]} = g^{[l]}(Z^{[l]}) \quad \text{eq.6.2}$$

Donde:

$W^{[l]}$ representa el matriz de pesos para las neuronas de la capa l.
$A^{[l-1]}$ representa los valores de entrada de la capa l y donde X = $A^{[0]}$.
$b^{[l]}$ es el vector de sesgos para la capa l.
$g^{[l]}$ es la función de activación en la capa l.
$A^{[l]}$ contiene los valores de salida de las neuronas de la capa l.

La función de activación para las capas ocultas tiene activaciones definidas por:

$$ReLU = A^{[l]} = \max(0, x) \quad \text{eq.6.3}$$

y para la capa de salida definidas por:

$$SoftMax \text{ function} = f(z)_j = \frac{e^{zj}}{\sum_{n=1}^{m} e^{zn}} \quad \text{eq.6.4.1}$$

o

$$Sigmoid \text{ function} = f(z) = \frac{1}{1+e^{-z}} \forall\, z\ real \quad \text{eq.6.4.2}$$

## 6.5. Alcance

Este trabajo se centra en redes neuronales profundas *feedforward* entrenadas clásicas, dado que el foco está puesto en la extracción de reglas a partir de las matrices de pesos de estas. Quedan fuera de este estudio variaciones de este tipo de redes tales como las redes neuronales recurrentes, y los *Transformers* que, si bien hacen uso de las redes neuronales *feedforward,* presentan arquitecturas diferentes dado que utilizan capas de atención, *embeedings* y transformaciones creando arquitecturas hibridas más complejas. Entonces, al referir a tecnologías de *Deep Learning*, referimos a los conceptos vertidos por Goodfellow et al., (2016) y Russell & Norvig, (2010), o a una definición equivalente presentadas en Domingos, (2018) donde las redes neuronales profundas son presentadas como capas de neuronas densas, las cuales se combinan con capas de entrada y salida, y alguna función de activación en sus neuronas.

## 6.6. El método propuesto

En esta sección se explica como la combinación de las técnicas de similitud coseno y lógica de primer orden funcionan para extraer reglas de una DNN *feedforward*

entrenada. Los pesos y sesgos se tratan con la misma precisión y signos a como son generados por la red, es decir, no se aplica ninguna transformación sobre los mismos. Para ejemplificar el presente algoritmo de extracción de reglas, se han utilizado tres redes *feedforward* de tres capas entrenadas usando la regla de *backpropagation* (Goodfellow et al., 2016).

A una neurona de una capa dada se la considera *critica* cuando el cálculo de la métrica de similitud coseno respecto de los datos de entrada con los pesos de la neurona, superan un valor de umbral dado. Se explicará en detalle el procedimiento para extraer una regla de confirmación.

Como se mencionó en la Sección 1, una de las cuestiones más cruciales en el desarrollo de un algoritmo de extracción de reglas es cómo restringir el tamaño del espacio de solución buscado. Sea R una DNN, cada neurona dentro de la red entrena W cantidad de pesos siendo, W la cantidad de entradas para cada neurona. Se podría pensar en W como la cantidad de neuronas en la capa $R^{L-1}$. Entonces en una pasada *feedforward* tendremos para la capa $R^{L}$, W entradas en cada neurona y M salidas, siendo M la cantidad de neuronas en la capa $R^{L}$. Cada valor en las salidas de las neuronas de la capa $R^{L-1}$ son presentados en cada neurona de la capa $R^{L}$ donde estos son tratados con multiplicaciones y sumas y expuestos luego a alguna función de activación. Esto se expresa en la eq.1.

Entonces se calculará la similitud coseno entre el vector $Z^{[l]}$ de la eq.1 con el vector de pesos formado por el peso que cada neurona le asigno al *feature* $R^{L}_{(F)}$. Llamaremos a este vector $T^{L}_{(F)}$.

Es importante destacar que, en la nomenclatura utilizada, el superíndice indica la capa dentro de la DNN, mientras que el subíndice identifica la neurona dentro de esa capa.

Esto nos permitirá entender cuales *features* de entrada mejor explican los resultados. De este modo, si después de aplicar el cálculo de similitud coseno entre ambos vectores, la métrica supera el valor de umbral, diremos que la neurona asociada al *feature* $R^{L}_{(F)}$ es crítica, y se la incluirá en el grupo de neuronas de la capa $R^{L}$ que explican la función que aprendió la red para una clase dada. En la Fig. 17, se identifica cada componente que interviene en el análisis.

En la siguiente sección se explica en detalle como la comparación vectorial ayuda en la determinación del camino crítico y extracción de la función aprendida por la DNN.

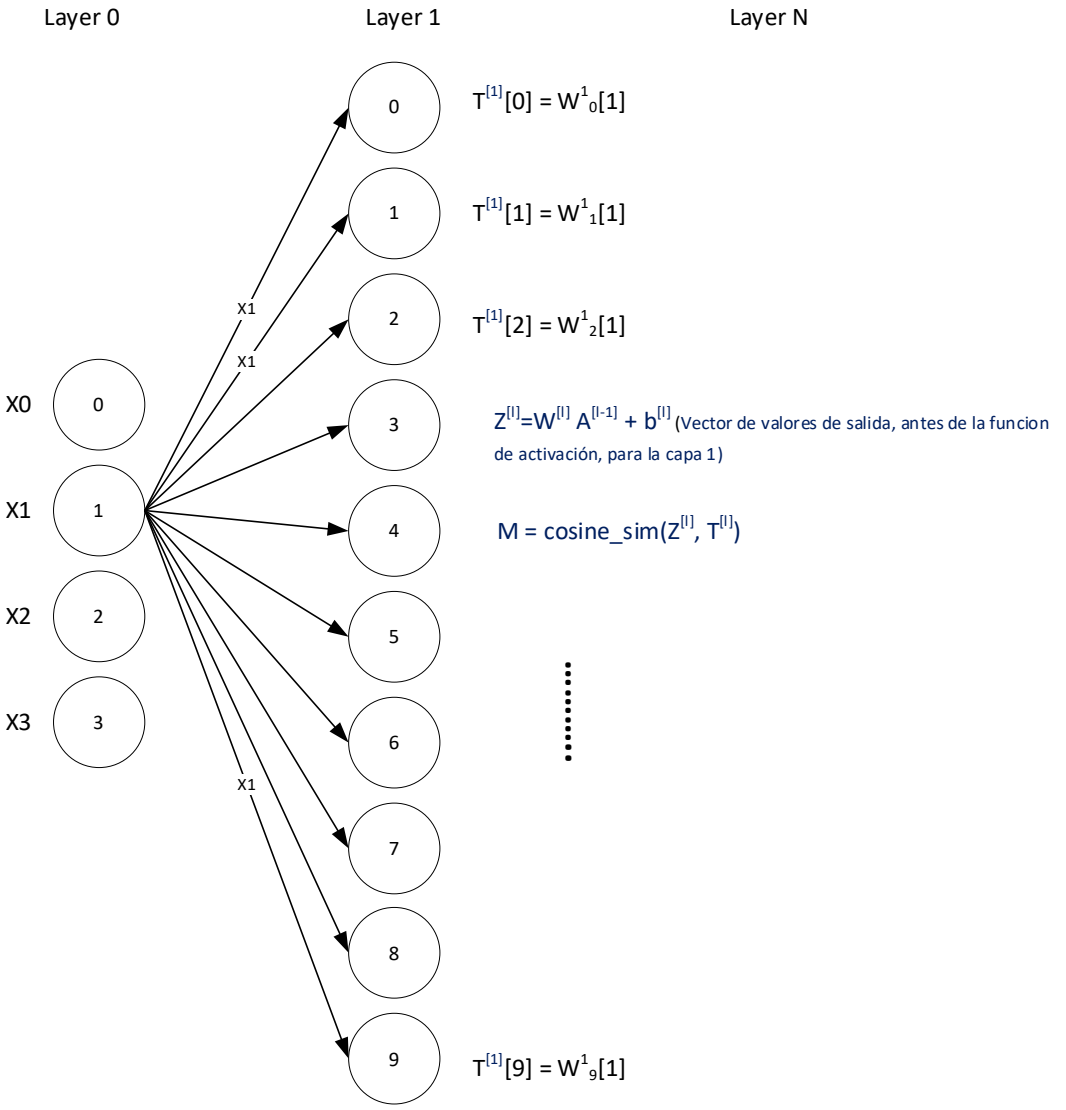


*Figura 17. Cálculo de similitud coseno entre vectores de pesos*

## 6.7. Similitud Coseno

La similitud coseno es una medida de la similitud existente entre dos vectores en un espacio que posee un producto interior con el que se evalúa el valor del coseno del ángulo comprendido entre ellos. Esta función trigonométrica proporciona un valor igual a 1 si el ángulo comprendido es cero, es decir si ambos vectores apuntan a un mismo lugar. Cualquier ángulo existente entre los vectores, el coseno arrojaría un valor inferior a uno. Si los vectores fuesen ortogonales el coseno se anularía, y si apuntasen en sentido contrario su valor sería -1. De esta forma, el valor de esta métrica se encuentra entre -1 y 1, es decir en el intervalo cerrado [-1,1]. Dados dos vectores *a* y *b* de dimensión *N*, la similitud coseno se calcula como:

$$similarity = \cos(\emptyset) = \frac{A.B}{||A||.||B||} = \frac{\sqrt{\sum_{i-1}^{n} A_i B_i}}{\sqrt{\sum_{i-1}^{n} A_i^2}\sqrt{\sum_{i-1}^{n} B_i^2}} \qquad \text{eq. 6.5}$$

## 6.8. Número mínimo de neuronas.

Como se explica en la siguiente sección, durante la segunda etapa del algoritmo COLOSSUS, se identifican las neuronas críticas en toda la red, para poder así identificar un camino neuronal critico por clase (en caso de tareas de clasificación), y de este modo ser capaces de extraer las funciones que la red aprendió y explicarlas con lógica proposicional (Pons et al., 2017). Para la capa cero, donde las neuronas de esta capa se mapean con los datos de entrada, existe un conjunto mínimo de neuronas necesarias (MNN), para que el algoritmo que explica las reglas alcance una precisión alta (acc >= 90%). Este conjunto MNN puede calcularse de dos maneras:

1. (NFDS/2) + 1 ➔ Donde NFDS= Cantidad total de *features* en el conjunto de datos.
   Para el caso en que el número de neuronas de la capa de entrada es menor a 10.

2. NFDS / QOL ➔ Donde QOL= Cantidad de neuronas en la capa de salida.

Entonces, llegado el caso donde el método de extracción de reglas identifique menos neuronas que el conjunto MNN, es posible seleccionar neuronas adicionales del siguiente modo:

- Calcular la entropía para los *features* que mapean con las neuronas del *conjunto secundario* identificado en la etapa 2 del algoritmo.
- Si el *conjunto de neuronas secundario* es un conjunto vacío, entonces se calcula la entropía sobre todos los *features* no críticos.
- Seleccionar la cantidad de neuronas faltantes tomando como criterio de selección los *features* que mayor entropía tengan.
- Las neuronas adicionales se agregan al conjunto de neuronas críticas por clase para $A^{[0]}$, y se tienen en cuenta en el cálculo de conjunciones explicado en la tercera etapa del algoritmo en la siguiente sección.

Dado que cada *feature* mapea con una neurona en la capa de entrada, se marcarán estas neuronas como críticas y se las tomara en cuenta en el algoritmo de validación de las reglas. Tener en cuenta que esta adición de neuronas es opcional.

## 6.9. El algoritmo COLOSSUS para extracción de reglas

El algoritmo de extracción de reglas COLOSSUS se basa en la estrategia de comparación vectorial de pesos discutida anteriormente y funciona en cuatro etapas.
**En la primera etapa**, se calculan estadísticas para los datos de entrada para las cuales la precisión predictiva de la red se maximiza. Dado que COLOSSUS busca aquellos valores que maximicen la precisión de la red, se buscaran las neuronas criticas dentro de la red con dichos valores estadísticos. Además, las estadísticas ayudaran en la verificación de las reglas que expliquen a los datos. Estas determinaran las cotas para cada *feature* critico de entrada.

**En la segunda etapa** se calcula la similitud coseno entre el vector $Z^{[l]}$ de la eq.1 con el vector $T^{[l]}$ de pesos formado por el peso que cada neurona le asigno al *feature* $R^{L}_{(F).}$ con el objetivo de entender cuales *features* de entrada mejor explican los resultados. De este modo, si después de aplicar el cálculo de similitud coseno entre ambos vectores la métrica supera el valor de umbral, diremos que la neurona asociada al *feature* $R^{L}_{(F)}$ es *crítica*, y se la incluirá en el grupo de neuronas de la capa $R^{[L]}$ que permitirán explicar la función que aprendió la red para una clase dada. Este proceso se repite para cada neurona de cada capa.
Utilizando la misma técnica, se calcula la similitud coseno entre los vectores de pesos de las diferentes neuronas en cada capa, con el objetivo de entender cuales neuronas guardan una relación entre sus sistemas de pesos. Entonces si una neurona critica guarda una relación vectorial con una o más neuronas, se dirá que estás neuronas trabajan en conjunto, y se hará referencia a estas como el *conjunto de neuronas secundarias.*
Se analiza el factor MNN, según lo expuesto en la sección 6.8, para completar el conjunto de neuronas críticas.
**En la tercera etapa**, se calculan las relaciones de conjunción entre las neuronas *críticas* de cada capa *para todas las clases*, con el objetivo de obtener una única tabla de verdad por capa.
Para calcular las relaciones de conjunción se utilizan los vectores de peso de las neuronas críticas. Si el valor del peso es positivo el valor será V o 1, y si es negativo, será F o 0. Así, si, por ejemplo, para una determinada capa, se identifican dos neuronas como críticas, la conjunción será calcularse como $R^{[L]}_{n1} \wedge R^{[L]}_{n2} = TT^{[N]}$.
Una vez obtenidas las tablas de verdad por capa, se calcula la relación de implicancia entre las tablas de verdad de cada capa ($TT^{[L-1]} \rightarrow TT^{[L]} \rightarrow TT^{[L+1]}$ = TTdnn). El objetivo de calcular las implicancias es comprobar que los resultados sean tautologías, y asegurarnos de que las neuronas marcadas como críticas son válidas y así garantizar que el *camino neuronal critico* encontrado tiene un fundamento basado en lógica valido. Las reglas obtenidas se escriben en términos de fórmulas bien formadas (fbf) de la lógica proposicional (Pons et al., 2017) para obtener reglas que expliquen la relación entrada-salida. Con estas fbf se explican las funciones que aprendió la red para cada clase.
Es importante notar que, además de las fbf extraídas, también podemos identificar una subred regularizada que explica la clase que se esté analizando.
**En la etapa final**, las neuronas identificadas como críticas en la capa de entrada $A^{[0]}$ se mapean con los datos estadísticos obtenidos en la primera etapa, para determinar la precisión de estas de las reglas, y se calcula una función de distancia para resolver posibles solapamientos entre clases. Como resultado de esta etapa se obtienen fórmulas de lógica de primer orden (Pons et al., 2017), que explican los datos de entrada. El algoritmo de validación se explica en detalle en la siguiente sección.

## 6.10. Algoritmo de validación de las reglas

Como se mencionó anteriormente, en la primera etapa del algoritmo de extracción de reglas se calculan e identifican las estadísticas que maximicen la precisión de la red, para cada clase. Con lo cual, el algoritmo de validación de las reglas utiliza tales estadísticas por clase y por *feature* para utilizarlas como valores de cotas.

Es importante destacar que, el algoritmo de validación de reglas opera sobre la ***unión*** de los *features* identificados como críticos para la capa $A^{[0]}$ para cada clase. Es decir, si un conjunto de datos tiene 3 clases posibles, y el algoritmo de extracción identifico para la capa $A^{[0]}$ las siguientes neuronas por clase:

1. Clase 1 = $L^0_0$
2. Clase 2 = $L^0_0 \wedge L^0_2$
3. Clase 3 = $L^0_1$

El algoritmo de validación operara sobre la unión de los *features* para todas las clases: if($L^0_0 \wedge L^0_1 \wedge L^0_2$) ➔ ClassN

Como siguiente paso, se lee e itera por todo el conjunto de datos y se verifica que cada *feature* relevante se encuentre dentro del rango de valores acotados por las estadísticas obtenidas en la primera fase. Si la verificación es verdadera, marcamos a dicha instancia (registro dentro del conjunto de datos), como perteneciente a una clase determinada.
En caso de que los datos en el conjunto de datos estén muy solapados entre sí, pude darse el caso que el algoritmo, para una misma instancia de los datos de entrada, nos indiqué que tal instancia pertenece a más de una clase. En tal caso, se determinará la clase correcta calculando una función que calcula la distancia que tiene el *feature* de mayor entropía respecto al centroide del *feature* que las clases tienen en común para las clases solapadas, de la siguiente manera:

- fme = Es la variable que mayor entropía tiene entre el conjunto de *features* seleccionado.

- fcc = Es un *feature* que ambas clases tienen en común. (Del conjunto de *features* seleccionado. De no haber *features* en común o si hay más de uno, elegir el que mayor entropía tenga.)

Para este ejemplo se asume que las clases que se solapan son las clases 0 y 1.

- dst0 = (fme – MinVal_Class0(fcc)) + (MaxVal_Class0(fcc) - fme) eq. 6.6
- dst1 = (fme – MinVal_Class1(fcc)) + (MaxVal_Class1(fcc) - fme) eq. 6.7

Finalmente se determina la clase correcta del siguiente modo:

Class = (1 if dst1 < dst0 else 0) eq. 6.8 ➔ La desigualdad puede variar dependiendo de la función de activación utilizada en la capa de salida.

*Es importante destacar que las estadísticas que maximizan la precisión de la red se utilizan para realizar la identificación de las neuronas y caminos neuronales críticos para cada clase. Para el algoritmo de validación de las reglas y para un mayor grado de generalización, las estadísticas para cada feature se toman de sus valores máximos y mínimos.*

Finalmente, para medir la precisión del algoritmo de extracción de reglas se compara la clase determinada *Class* con el *ground truth* provisto en el conjunto de datos.

## 6.11. Caso de estudio sobre el conjunto de datos *Iris*

En la presente sección se presentan los resultados del testeo del algoritmo COLOSSUS sobre el conjunto de datos de Iris.
El *dataset* **Iris** es un conjunto de datos de clasificación multiclase clásico. El conjunto de datos de Iris contiene cuatro características (longitud y anchura de sépalos y pétalos) de 50 muestras de tres especies de flores Iris (Iris setosa, Iris virginica e Iris versicolor). Incluye tres especies de iris con 50 muestras cada una, así como algunas propiedades de cada flor. Una especie de flor es linealmente separable de las otras dos, pero las otras dos no son linealmente separables entre sí. Los *features* de entrada a la red para este conjunto de datos son los siguientes: *sepal length, sepal width, petal length, petal width.*
Para este caso de ejemplo se configuro un DNN con 4 neuronas en la capa de entrada, dos capas ocultas con 10 neuronas por cada capa. Como función de activación para las capas ocultas se utilizó la función *ReLU*. Finalmente, para la capa de salida se configuro con 3 neuronas y se utilizó la función *SoftMax* como función de activación. El umbral utilizado para determinar si una neurona es crítica o no fue de 0.6.
La red neuronal entrenada con este conjunto de datos alcanzo un *accuracy* del %97.
Luego de aplicar los pasos identificados en las etapas 2 y 3 del algoritmo, se obtuvieron las siguientes fbf para cada clase, antes de aplicar el factor MNN:

Clase 0 = $L^0_{(0)}$ -> $L^1_{(1)}$ ^ $L^1_{(2)}$ -> $L^2_{(2)}$ ^ $L^2_{(4)}$
Clase 1 = $L^0_{(0)}$ ^ $L^0_{(2)}$ -> $L^1_{(2)}$ ^ $L^1_{(5)}$ -> $L^2_{(3)}$ ^ $L^2_{(6)}$ ^ $L^2_{(7)}$ ^ $L^2_{(9)}$
Clase 2 = $L^0_{(2)}$ -> $L^1_{(5)}$ -> $L^2_{(1)}$ ^ $L^2_{(9)}$

En las fig.18, 19 y 20 se presentan las redes regularizadas para cada clase.

*Tabla 1. Estadísticas para dataset Iris, para el problema de clasificación*

| | $X_0$ | $X_1$ | $X_2$ | $X_3$ |
|---|---|---|---|---|
| | sepal length | sepal width | petal length | petal width |
| **Class-0** | | | | |
| Min | 4,3 | 2,3 | 1,0 | 0,1 |
| Perc50 | 5 | 3,4 | 1,5 | 0,2 |
| Max | 5,8 | 4,4 | 1,9 | 0,6 |
| Cotas | [4.3..5,8] | [2.3..4,4] | [1..1,9] | [0,1..0,6] |
| **Class-1** | | | | |
| Perc25 | 5,6 | 2,5 | 4 | 1,2 |
| Min | 4,9 | 2 | 3 | 1 |
| Max | 7 | 3,4 | 5,1 | 1,8 |
| Cotas | [4,9..7] | [2..3.4] | [3..5,1] | [1..1,8] |
| **Class-2** | | | | |
| Min | 4,9 | 2,2 | 4,5 | 1,4 |
| Perc75 | 6,9 | 3,2 | 5,9 | 2,3 |
| Max | 7,9 | 3,8 | 6,9 | 2,5 |
| Cotas | [4,9..7,9] | [2,2..3,8] | [4.5..6,9] | [1.4..2,5] |

En la tabla 1, se presentan las estadísticas extraídas del *dataset* Iris, con relación a lo expuesto en la etapa 1. Notar que para las clases 0 y 2 se han adicionado los valores para el mínimo, y para la clase 1 se realizó lo propio para el máximo. Esto es así como método de comprobación donde al extender los límites de las cotas entre los valores mínimos y máximos de cada *feature* ayude a resolver el solapamiento entre clases.
Una vez extraídas las funciones que la red aprendió, y calculadas además las estadísticas para cada clase, se deben validar las reglas con los datos del *dataset*. Entonces, como se indica en la sección anterior se debe iterar por los datos del *dataset* y evaluar cada instancia en que clase entra, verificando si cada *feature* relevante para cada clase se encuentra dentro de los valores de las cotas entregados por las estadísticas.
Por lo expuesto en el punto 6.8, es posible aumentar la cantidad de *features* a considerar como críticos, con el objetivo de incrementar la precisión de la explicación de los datos. Entonces, teniendo en cuenta que la red tiene menos de 10 *features*, se aplica el método MNN 1 donde MNN = (NFDS/2) + 1, se obtiene MNN = (4/2) + 1 = 3. Con lo cual podemos adicionar 1 *feature* más como crítico.
Dado que para este caso de uso el conjunto de neuronas secundarias es un conjunto vacío, se calcula la entropía para todos los *features* no críticos. El análisis identifica a *petal width* ($X_3$) como crítico y susceptible de ser agregado al conjunto de neuronas críticas.
Para definir los parámetros de la función de distancia, y dado que para esta DNN las clases no presentan un *feature* en común, para resolver el solapamiento se calcula la entropía entre las neuronas criticas quedando los *features sepal length* y *petal length* como los parámetros para la función de distancia. Además, observando las fbf, ambas características son los identificadas para la clase 1, y esta clase tiene al menos 1 *feature* en común con las demás.
Entonces, suponiendo que hay una entrada N dentro del conjunto de datos, cuyos valores hacen que la regla determine las clases 0 y 1, el cálculo de distancia quedaría del siguiente modo:

$dst_0$ = (petal_length$_{(N)}$ – MinValue_Class0('sepal length')) + (MaxValue_Class0('sepal length') - petal_length$_{(N)}$) por eq. 6.6

$dst_1$ = (petal_length$_{(N)}$ – MinValue_Class1('sepal length')) + (MaxValue_Class1('sepal length') - petal_length$_{(N)}$) por eq. 6.7

Luego de aplicar el cálculo en eq. 6.8, obtenemos Class = (1 if dst1 < dst0 else 0)
Finalmente, el valor obtenido en Class, se compara con el *ground truth* para la instancia N. Entonces la regla obtenida para las distintas clases del conjunto de datos Iris quedaría como sigue:

Class0 = ($\forall X_0$[4.3..5,8] ∧ $\forall X_2$[1..1,9] ∧ $\forall X_3$[0,1..0,6])
^ fnc_dst_class(petal_length$_{(N)}$, sepal length, 0, N) == 0

Class1 = ($\forall X_0$[4,9..7] ∧ $\forall X_2$[3..5,1] ∧ $\forall X_3$[1,2..1,8])
^ fnc_dst_class(petal_length$_{(N)}$, sepal length, 1, N) == 1

Class2 = ($\forall X_0$[4,9..7,9] ∧ $\forall X_2$[4,5..6,9] ∧ $\forall X_2$[1,4..2,5])
^ fnc_dst_class(petal_length$_{(N)}$, sepal length, 2, N) == 2

fnc_dst_class(fme, fcc, ClassX, ClassY) = $\exists X_n$ ($dst_x < dst_y$)

Para este conjunto de datos, el algoritmo de explicación de reglas logra una precisión del 95%.

### 6.11.1 Redes regularizadas por clase – Iris

En la presente sección se presentan de manera grafica las redes neuronales extraídas según las reglas extraídas por COLOSSUS por cada clase para el dataset *Iris*. Cada subred, o red regularizada, presenta el camino neuronal mínimo necesario para explicar cada clase. Cada subred corresponde a la fbf extraída en el tercer paso del algoritmo.

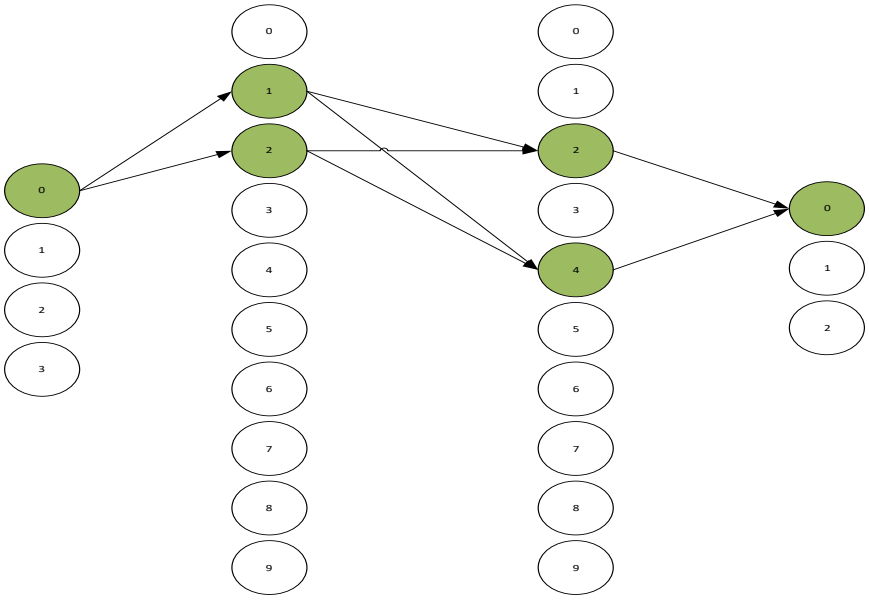


*Figura 18. Camino neuronal crítico para Clase 0 = $L^0_{(0)}$ -> $L^1_{(1)}$ ^ $L^1_{(2)}$ -> $L^2_{(2)}$ ^ $L^2_{(4)}$*

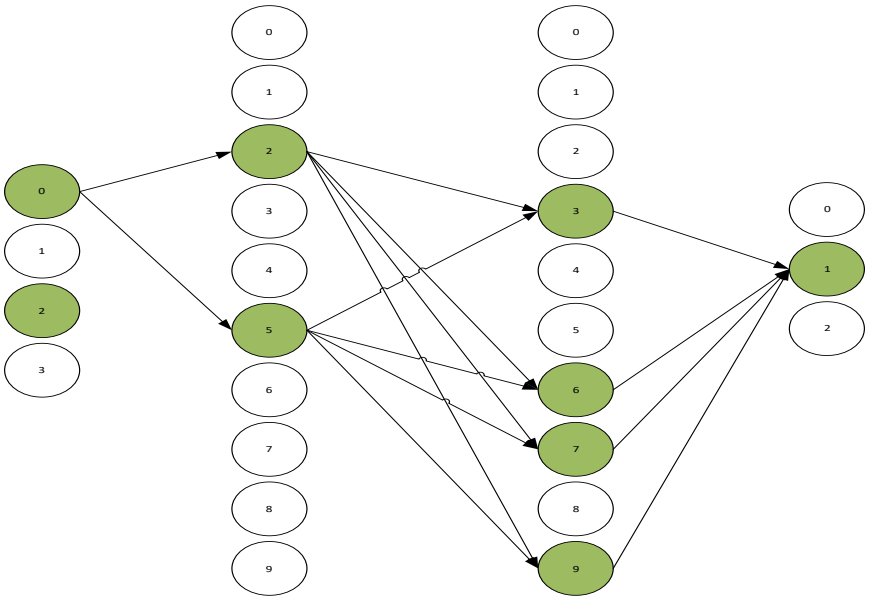


*Figura 19. Camino neuronal crítico para Clase 1 = $L^0_{(0)}$ ^ $L^0_{(2)}$ -> $L^1_{(2)}$ ^ $L^1_{(5)}$ -> $L^2_{(3)}$ ^ $L^2_{(6)}$ ^ $L^2_{(7)}$ ^ $L^2_{(9)}$*

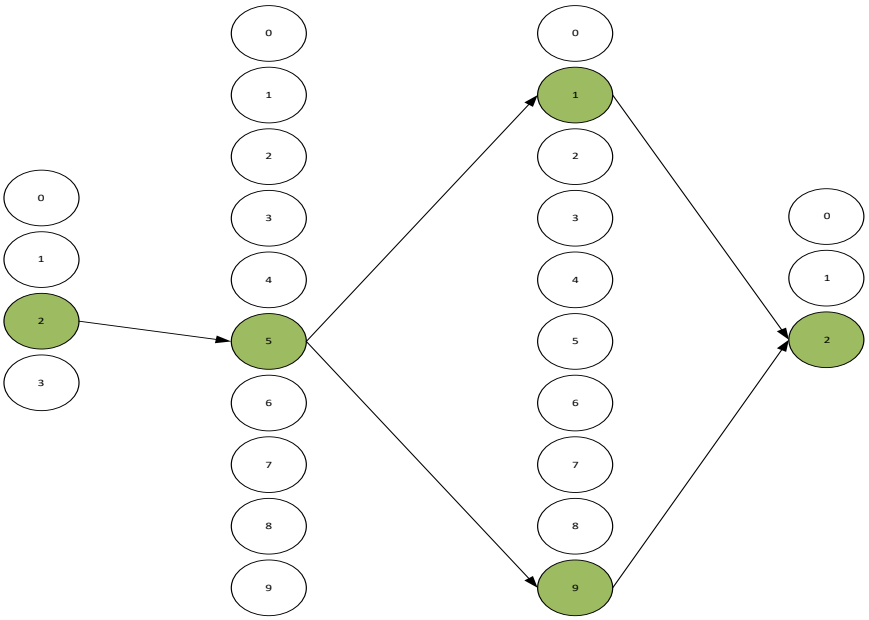


*Figura 20. Camino neuronal crítico para Clase 2 = $L^0_{(2)}$ -> $L^1_{(5)}$ -> $L^2_{(1)}$ ^ $L^2_{(9)}$*

## 6.12. Complejidad del algoritmo

La complejidad del algoritmo se puede calcular analizando el número de operaciones y el tiempo que lleva ejecutarlas. En este caso, el algoritmo consta de varios pasos, cada uno con su propio conjunto de operaciones. A continuación, se analiza la complejidad de cada paso:

***Paso 1: Calcular estadísticas para el conjunto de datos de entrada***
Este paso implica calcular estadísticas para el conjunto de datos de entrada, como máximo, mínimo, media y percentiles. La complejidad de este paso depende del tamaño del conjunto de datos, ya que el algoritmo necesita pasar por cada punto de datos para calcular las estadísticas. Por lo tanto, la complejidad de este paso es O (n), donde n es el número de puntos de datos en el conjunto de datos.

***Paso 2: Calcular la similitud del coseno***
Este paso implica calcular la similitud coseno (SC) entre neuronas en cada capa adyacente de la DNN, entonces la complejidad de este paso depende de la cantidad de capas y neuronas en la DNN. Aquí es importante destacar que la comparativa completa entre las neuronas de capas adyacentes, solo sucede entre la capa $A^{[0]}$, la cual mapea con los datos, y la primer capa oculta. De aquí en más el algoritmo solo opera, para las capas subsecuentes, con las neuronas seleccionadas como críticas. Entonces, digamos que la DNN tiene L capas y N neuronas en cada capa, podemos definir un vector T que contendrá las neuronas críticas de una capa dada $N^{[l]}_{[n]}$, cuya SC exceda el valor de umbral.
Entonces, la complejidad de este paso sería O(L*T^2), ya que, para cada capa, el algoritmo necesita calcular la similitud del coseno para cada par de neuronas.

***Paso 3: Calcular relaciones conjuntivas***
Este paso implica calcular relaciones conjuntivas para cada capa de la DNN. La complejidad de este paso depende de la cantidad de capas y de la cantidad de neuronas críticas en cada capa. Digamos que el DNN tiene L capas y C neuronas críticas en cada capa. Entonces, la complejidad de este paso sería O(L*C), ya que el algoritmo necesita pasar por cada neurona crítica en cada capa para calcular las relaciones disyuntivas.

***Paso 4: Calcular relaciones de implicancia***
Este paso implica calcular relaciones de implicancia entre las relaciones disyuntivas obtenidas en el paso anterior. La complejidad de este paso depende del número de relaciones disyuntivas, que es igual al número de capas. Por tanto, la complejidad de este paso es O(L), donde L es el número de capas en la DNN.

***Paso 5: Calcular el factor MNN***
Este paso implica calcular el factor MNN, que es el número máximo de neuronas permitidas en el conjunto de neuronas críticas. La complejidad de este paso depende de la cantidad de neuronas críticas y de la cantidad de neuronas no críticas. Digamos que hay M neuronas críticas y N neuronas no críticas. Entonces, la complejidad de este paso sería O (M+N), ya que el algoritmo necesita pasar por cada neurona crítica y no crítica para calcular el factor MNN.

***Paso 6: Mapeo de neuronas críticas***
Este paso implica mapear las neuronas críticas identificadas en los pasos anteriores al conjunto de datos de entrada. La complejidad de este paso depende de la cantidad de neuronas críticas y del tamaño del conjunto de datos. Digamos que hay M neuronas críticas y N puntos de datos en el conjunto de datos. Dado el vector M mapea directamente con los *features* del conjunto de datos (vector N), la complejidad de este paso sería O (1), dado que M puede verse como un subconjunto de N.

***Paso 7: Obtener fórmulas lógicas de primer orden***
Este paso implica obtener fórmulas lógicas de primer orden que expliquen los datos de entrada. La complejidad de este paso depende de la cantidad de neuronas críticas y la cantidad de clases en el conjunto de datos. Digamos que hay M neuronas críticas y K clases en el conjunto de datos. Entonces, la complejidad de este paso sería O(M*K), ya que el algoritmo necesita pasar por cada neurona crítica y cada clase para obtener las fórmulas lógicas.

En general, la complejidad del algoritmo se puede expresar como O (n + L*T^2 + L*C + L + M+N + 1 + M*K), que se puede simplificar a O(n + L *T^2 + L*C + 1 + M*K). Por lo tanto, la complejidad del algoritmo es polinómica y puede considerarse eficiente para la mayoría de los conjuntos de datos. Sin embargo, el tiempo de ejecución puede aumentar para conjuntos de datos más grandes con una mayor cantidad de capas y neuronas.

## 6.13. Conclusiones

En este capítulo se describe el algoritmo COLOSSUS para la extracción y explicación de reglas, el cual comenzará con un proceso de cálculo de estadísticas sobre los datos, y posterior extracción de pesos de la red neuronal entrenada, donde se calculará la similitud coseno entre los vectores que forma cada variable de entrada para cada neurona entre sí, y también en relación con el vector de sesgos, y el vector formado por los valores de activación en cada capa. Además, se realiza el mismo procedimiento, para los vectores de pesos de cada neurona, lo cual resulta en una comparación interneurona de vectores de pesos, con el objetivo de entender la colaboración entre las neuronas de una capa dada.
Las reglas extraídas, se componen y explican en formato de fórmulas bien formadas de la lógica proposicional.
En tal sentido el objetivo es analizar cómo se correlación las variables de entrada, y como estas explican los resultados en la salida de la red. De este modo se demuestra
cómo es posible identificar caminos críticos neuronales basados en los datos, que va desde las variables de entrada hasta la salida de la red.
Las estadísticas obtenidas sobre los datos de entrada se utilizan para el método de extracción de reglas, sobre aquellas que mejor explican los resultados arrojados por las redes (máxima precisión). De este modo, los rangos de valores en entre ambas métricas sientan las bases que asisten en la validación y verificación de las reglas extraídas.
En el siguiente capitulo, se presentan ejemplos sobre otros dos conjuntos de datos con el objetivo de validar y verificar el método. Además, se presenta el análisis de casos puntuales con el objetivo de explicar de un modo empírico el cálculo de la función de distancia a través de datos reales.

CAPITULO 7

# Corroboración Empírica

## 7.1. Corroboración Empírica

En el presente capitulo, se presentan ejemplos adicionales con un análisis más exhaustivo, con el objetivo de explicar de un modo empírico el cálculo de la función de distancia a través de casos reales.

## 7.2. Caso de estudio sobre el conjunto de datos *Wine*

El dataset **Wine** incluye información sobre tres clases de vinos producidos en Italia y 13 características predictivas. Este conjunto de datos es el resultado de un análisis químico de vinos cultivados en la misma región de Italia, pero derivados de tres cultivares diferentes. El análisis determinó las cantidades de 13 componentes encontrados en cada uno de los tres tipos de vinos.
Los *features* de entrada a la red para este conjunto de datos son los siguientes:
*Alcohol, Malic acid, Ash, Alkalinity of Ash, Magnesium, Total Phenols, Flavonoids, Nonflavanoids Phenols, Proanthocyanins, Color Intensity, Hue, Dilution, Proline.*

Con el objetivo de identificar las neuronas críticas, a la DNN se le pasan los valores obtenidos en las estadísticas obtenidas en el primer paso del algoritmo. Dado que COLOSSUS busca aquellos valores que maximicen la precisión de la red, se buscaran las neuronas criticas dentro de la red con dichos valores estadísticos. Entonces utilizando tales estadísticas, la salida esperada será la mayor precisión en la predicción del resultado para las tres clases de vino (tinto, rosado y blanco).

Para este caso de ejemplo se configuro una DNN con 13 neuronas en la capa de entrada, dos capas ocultas con 12 neuronas por cada capa. Como función de activación para las capas ocultas se utilizó la función *ReLU*. Finalmente, para la capa de salida se configuro con 3 neuronas y se utilizó la función *SoftMax* como función de activación. El umbral utilizado para determinar si una neurona es crítica o no fue de 0.6.
La red neuronal entrenada con este conjunto de datos alcanzo un *accuracy* del %92.
Luego de aplicar los pasos identificados en las etapas 2 y 3 del algoritmo, se obtuvieron las siguientes fbf para cada clase, antes de aplicar el factor MNN:

Clase 0 = $L^0_{(12)}$ -> $L^1_{(10)}$ -> $L^2_{(2)}$ ^ $L^2_{(5)}$ ^ $L^2_{(8)}$ ^ $L^2_{(9)}$
Clase 1 = $L^0_{(12)}$ -> $L^1_{(1)}$ ^ $L^1_{(10)}$ -> $L^2_{(5)}$ ^ $L^2_{(8)}$ ^ $L^2_{(9)}$
Clase 2 = $L^0_{(12)}$ -> $L^1_{(10)}$ -> $L^2_{(5)}$ ^ $L^2_{(8)}$ ^ $L^2_{(9)}$

En las fig. 21, 22 y 23 se presentan las redes regularizadas para cada clase.
En la tabla 2, se presentan las estadísticas extraídas del *dataset* Wine, con relación a lo expuesto en la etapa 1. Notar que para las clases 0 y 2 se ha adicionado los valores para

el mínimo. Una vez extraídas las funciones que la red aprendió, se validan las reglas con los datos del *dataset*.
El *feature* que identifico el algoritmo como crítico es *proline* ($X_{12}$), y por lo expuesto en el punto 6.8, es posible aumentar la cantidad de *features* a considerar como críticos, con el objetivo de incrementar la precisión de la explicación de los datos. Entonces, por MNN 2, si dividimos la cantidad de *features* (13) por la cantidad de neuronas de la capa de salida (3), obtenemos un MNN = 4. Con lo cual podemos adicionar 3 *features* más al conjunto de críticos. Dado que para este caso de uso el conjunto de neuronas secundarias es un conjunto vacío, se calcula la entropía para todos los *features* no críticos. El análisis identifica a *flavanoids* ($X_6$), *color_intensity* ($X_9$) *malic_acid* ($X_1$) como crítico y susceptible de ser agregado al conjunto de neuronas críticas.

*Tabla 2.* Estadísticas para dataset Wine, para el problema de clasificación

| | $X_1$ | $X_6$ | $X_9$ | $X_{12}$ |
|---|---|---|---|---|
| | malic_acid | flavanoids | color intensity | proline |
| **Clase 0** | | | | |
| Min | 1.35 | 2,19 | 3.52 | 680 |
| Perc50 | 1.77 | 2,98 | 5.4 | 1095 |
| Max | 4.04 | 3,93 | 8.9 | 1680 |
| Cotas | [1,35..4,04] | [2,19..3,93] | [3,52..8,9] | [680..1680] |
| **Clase 1** | | | | |
| Min | 0.74 | 0,57 | 1.28 | 278 |
| Max | 5.8 | 5,08 | 6 | 985 |
| Cotas | [0,74..5,8] | [0,57..5,08] | [1,26..6] | [278..985] |
| **Clase 2** | | | | |
| Min | 1.24 | 0,34 | 3.85 | 415 |
| Perc50 | 3.27 | 0,685 | 7.55 | 627.5 |
| Max | 5.65 | 1,57 | 13 | 880 |
| Cotas | [1,24..5,65] | [0,34..1,57] | [3,85..13] | [415..880] |

Para el cálculo de distancia se utilizaron los *features proline* ($X_{12}$) y *flavanoids* ($X_6$) dado que el primero es el *feature* que identifico el algoritmo de extracción y es común a todas las clases, y el segundo el que mayor entropía presenta, por lo tanto, las métricas para el cálculo de distancia quedan expresadas como sigue:

$dst_X$ = (*proline*$_{(N)}$ – MinValue_ClassX(*'flavanoids'*)) + (MaxValue_ClassX(*'flavanoids'*) - *proline*$_{(N)}$) por eq. 6.6

$dst_Y$ = (*proline*$_{(N)}$ – MinValue_ClassY(*'flavanoids'*)) + (MaxValue_ClassY(*flavanoids'*) - *proline*$_{(N)}$) por eq. 6.7

Luego de aplicar el cálculo en eq. 6.8, obtenemos Class = (X if $dst_X < dst_Y$ else Y).

Finalmente, el valor obtenido en Class, se compara con el *ground truth* para la instancia N. Entonces las reglas obtenidas para las distintas clases del conjunto de datos Wine quedarían como sigue:

Class0 = ($\forall X_1$[1,35..4,04] ∧ $\forall X_6$[2,19..3,93] ∧ $\forall X_9$ [3,52..8,9] ∧ $\forall X_{12}$[680..1680])
^ fnc_dst_class(*proline*$_{(N)}$, *flavanoids*, 0, N) == 0

Class1 = ($\forall X_1$[0,74..5,8] ∧ $\forall X_6$[0,57..5,08] ∧ $\forall X_9$ [1,26..6] ∧ $\forall X_{12}$[278..985])
^ fnc_dst_class(*proline*$_{(N)}$, *flavanoids*, 1, N) == 1

Class2 = ($\forall X_1$[1,24..5,65] ∧ $\forall X_6$[0,34..1,57] ∧ $\forall X_9$ [3,85..13] ∧ $\forall X_{12}$[415..880])
^ fnc_dst_class(*proline*$_{(N)}$, *flavanoids*, 2, N) == 2

fnc_dst_class(fme, fcc, ClassX, ClassY) = $\exists X_n$ ($dst_x < dst_y$)

Para este conjunto de datos, el algoritmo de explicación de reglas logra una precisión del 97%.

### 7.2.1 Redes regularizadas por clase – Wine

En la presente sección se presentan de manera grafica las redes neuronales extraídas según las reglas extraídas por COLOSSUS por cada clase para el dataset *Wine*. Cada subred, o red regularizada, presenta el camino neuronal mínimo necesario para explicar cada clase. Cada subred corresponde a la fbf extraída en el tercer paso del algoritmo.

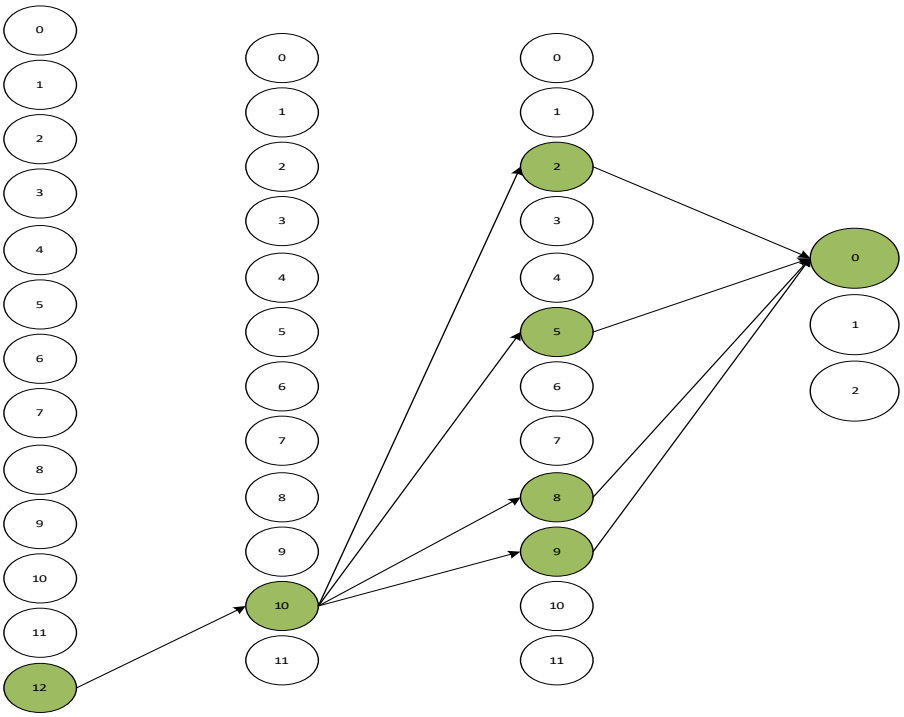


*Figura 21. Camino neuronal crítico para Clase 0 = $L^0_{(12)}$ -> $L^1_{(10)}$ -> $L^2_{(2)}$ ^ $L^2_{(5)}$ ^ $L^2_{(8)}$ ^ $L^2_{(9)}$*

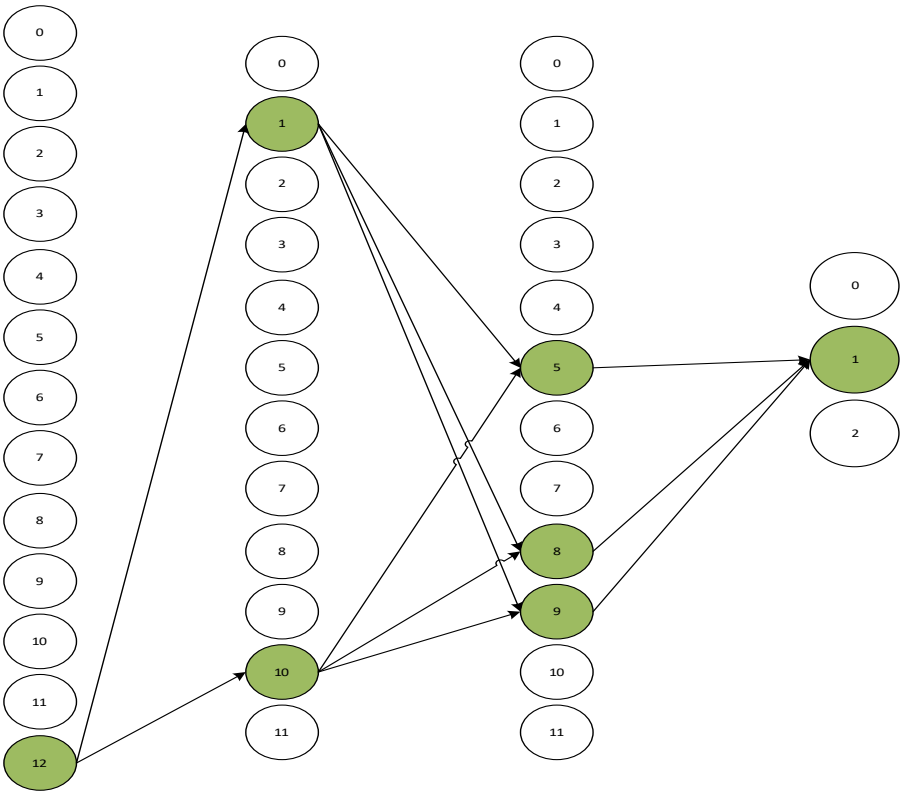


*Figura 22. Camino neuronal crítico para Clase 1 = $L^0_{(12)}$ -> $L^1_{(1)}$ ^ $L^1_{(10)}$ -> $L^2_{(5)}$ ^ $L^2_{(8)}$ ^ $L^2_{(9)}$*

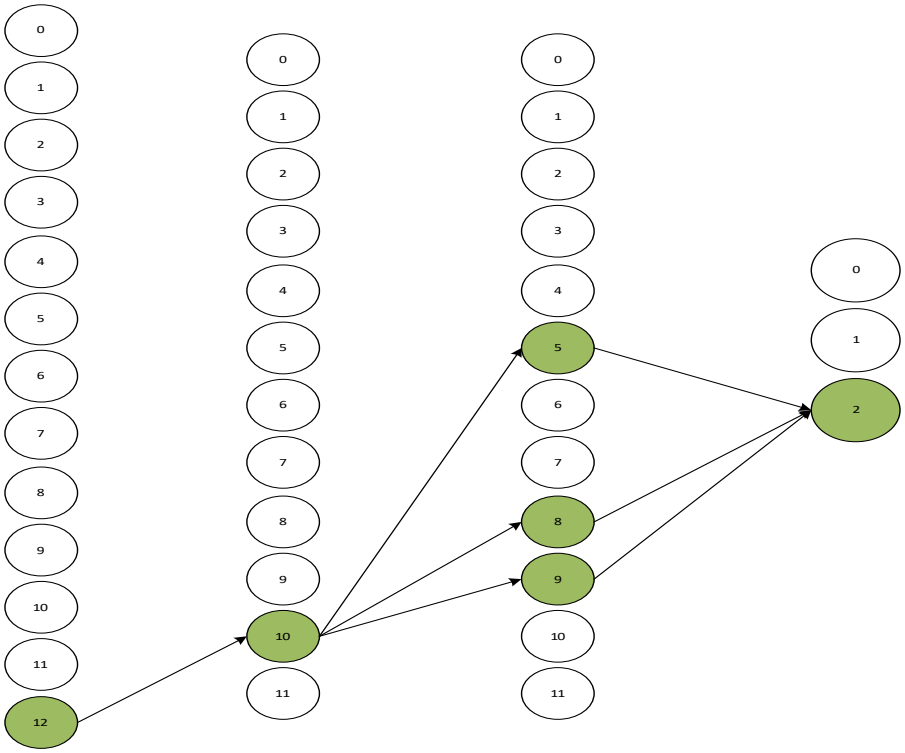


*Figura 23. Camino neuronal crítico para Clase 2 = $L^0_{(12)}$ -> $L^1_{(10)}$ -> $L^2_{(5)}$ ^ $L^2_{(8)}$ ^ $L^2_{(9)}$*

## 7.3. Caso de estudio sobre el conjunto de datos *BreastCancer*

El dataset **BreastCancer** incluye información sobre predicciones acerca de si un tumor es benigno o maligno. Las características se calculan a partir de una imagen digitalizada de una aspiración con aguja fina (PAAF) de una masa mamaria. Describen características de los núcleos celulares presentes en la imagen.
Los *features* de entrada a la red para este conjunto de datos son los siguientes:
*radius_mean, texture_mean, perimeter_mean, area_mean, smoothness_mean, compactness_mean, concavity_mean, concave points_mean, symmetry_mean, fractal_dimension_mean, radius_se, texture_se, perimeter_se, area_se, smoothness_se, compactness_se, concavity_se, concave points_se, symmetry_se, fractal_dimension_se, radius_worst, texture_worst, perimeter_worst, area_worst, smoothness_worst, compactness_worst, concavity_worst, concave points_worst, symmetry_worst, fractal_dimension_worst.*

Con el objetivo de identificar las neuronas críticas, a la DNN se le pasan los valores obtenidos en las estadísticas obtenidas en el primer paso del algoritmo. Dado que COLOSSUS busca aquellos valores que maximicen la precisión de la red, se buscaran las neuronas criticas dentro de la red con dichos valores estadísticos. Entonces utilizando tales estadísticas, la salida esperada será la mayor precisión en la predicción del resultado para las dos clases de tumores (benigno, maligno).

Para este caso de ejemplo se configuro un DNN con 30 neuronas en la capa de entrada, dos capas ocultas con 16 neuronas por cada capa. Como función de activación para las capas ocultas se utilizó la función *ReLU*. Finalmente, para la capa de salida se configuro con 1 neuronas y se utilizó la función *Sigmoide* como función de activación. El umbral utilizado para determinar si una neurona es crítica o no fue de 0.95.
La red neuronal entrenada con este conjunto de datos alcanzo un *accuracy* del %99.
Luego de aplicar los pasos identificados en las etapas 2 y 3 del algoritmo, se obtuvieron las siguientes fbf.

$$\begin{aligned}\text{Clase 0} = {} & L^0_{(0)} \wedge L^0_{(1)} \wedge L^0_{(3)} \wedge L^0_{(6)} \wedge L^0_{(7)} \wedge L^0_{(10)} \wedge L^0_{(12)} \wedge L^0_{(13)} \wedge L^0_{(20)} \wedge L^0_{(21)} \wedge \\ & L^0_{(22)} \wedge L^0_{(23)} \wedge L^0_{(24)} \wedge L^0_{(26)} \wedge L^0_{(27)} \wedge L^0_{(28)} \\ & \rightarrow L^1_{(3)} \wedge L^1_{(5)} \wedge L^1_{(6)} \wedge L^1_{(9)} \wedge L^1_{(11)} \\ & \rightarrow L^2_{(0)} \wedge L^2_{(1)} \wedge L^2_{(2)} \wedge L^2_{(3)} \wedge L^2_{(9)} \wedge L^2_{(11)} \wedge L^2_{(13)} \wedge L^2_{(15)}\end{aligned}$$

$$\begin{aligned}\text{Clase 1} = {} & L^0_{(0)} \wedge L^0_{(1)} \wedge L^0_{(3)} \wedge L^0_{(6)} \wedge L^0_{(7)} \wedge L^0_{(10)} \wedge L^0_{(12)} \wedge L^0_{(13)} \wedge L^0_{(20)} \wedge L^0_{(21)} \wedge \\ & L^0_{(22)} \wedge L^0_{(23)} \wedge L^0_{(24)} \wedge L^0_{(26)} \wedge L^0_{(27)} \wedge L^0_{(28)} \\ & \rightarrow L^1_{(3)} \wedge L^1_{(5)} \wedge L^1_{(6)} \wedge L^1_{(9)} \wedge L^1_{(11)} \\ & \rightarrow L^2_{(0)} \wedge L^2_{(1)} \wedge L^2_{(2)} \wedge L^2_{(3)} \wedge L^2_{(9)} \wedge L^2_{(11)} \wedge L^2_{(13)} \wedge L^2_{(15)}\end{aligned}$$

Como se puede apreciar, ambas fbf son idénticas. Esto es así, dado que el enfoque de tratamiento para este conjunto de datos puede verse como un problema de clasificación o de predicción. En tal sentido, y para los problemas de predicción, el algoritmo entregara una red normalizada y conjunto de reglas único para ambas clases.

*Tabla 3.* Estadísticas para el *dataset* BreastCancer, para el problema de clasificación.

| Stat | $X_0$ | $X_1$ | $X_2$ | $X_3$ | $X_6$ | $X_7$ | $X_{10}$ | $X_{12}$ | $X_{13}$ | $X_{20}$ | $X_{21}$ | $X_{22}$ | $X_{23}$ | $X_{24}$ | $X_{26}$ | $X_{27}$ | $X_{28}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Clase 0** | | | | | | | | | | | | | | | | | |
| Min | 6.98 | 9.71 | 43.79 | 143.5 | 0 | 0 | 0.115 | 0.758 | 6.802 | 7.93 | 12.02 | 50.41 | 185.2 | 0.07117 | 0 | 0 | 0.1566 |
| P75 | 13.37 | 19.76 | 86.1 | 551.1 | 0.05999 | 0.03251 | 0.3416 | 2.388 | 25.03 | 14.8 | 26.51 | 96.59 | 670 | 0.1376 | 0.2216 | 0.0974 | 0.2983 |
| Max | 17.85 | 33.81 | 114.2 | 992.1 | 0.4108 | 0.08534 | 0.811 | 5.118 | 77.11 | 19.82 | 41.78 | 127.1 | 1210 | 0.2006 | 1.252 | 0.175 | 0.4228 |
| **Clase 1** | | | | | | | | | | | | | | | | | |
| Min | 10.95 | 10.38 | 71.9 | 361.6 | 0.02398 | 0.0231 | 0.1938 | 1.334 | 13.99 | 12.84 | 16.67 | 85.1 | 508.1 | 0.08822 | 0.0239 | 0.0289 | 0.1565 |
| P50 | 17.32 | 21.46 | 114.2 | 932 | 0.1513 | 0.08628 | 0.5472 | 3.6795 | 54.455 | 20.59 | 28.945 | 138 | 1303 | 0.14345 | 0.4049 | 0.182 | 0.3103 |
| Max | 28.11 | 39.28 | 188.5 | 2501 | 0.4268 | 0.2012 | 2.873 | 21.98 | 542.2 | 36.04 | 49.54 | 251.2 | 4254 | 0.2226 | 1.17 | 0.291 | 0.6638 |

*Tabla 4. Referencias y cotas por feature para el dataset BreastCancer, para el problema de clasificación.*

| # | Features | Cotas Clase 0 | Cotas Clase 1 |
|---|---|---|---|
| X0 | radius mean | [6.98..17.85] | [10.95..28.11] |
| X1 | texture mean | [9.71..33,81] | [10,38..39.28] |
| X2 | perimeter mean | [43.79..114,2] | [71,9..188.5] |
| X3 | area mean | [143.5..992,1] | [361,6..2501] |
| X6 | concavity mean | [0..0.4108] | [0.02398..0.4268] |
| X7 | Concave points mean | [0..0.08534] | [0,0231..0.2012] |
| X10 | radius se | [0.115..0.811] | [0.1938..2.873] |
| X12 | perimeter se | [0.758..5,118] | [1,334..5,118] |
| X13 | area se | [6.802..77,11] | [13,99..542.2] |
| X20 | radius worst | [7.93..19,82] | [12,84..36.04] |
| X21 | texture worst | [12.02..41,78] | [16,67..49.54] |
| X22 | perimeter worst | [50.41.. 127,1] | [85,1..251.2] |
| X23 | area worst | [185.2..1210] | [508,1..4254] |
| X24 | smoothness worst | [0.07117.. 0.2006] | [0,08822.. 0.2226] |
| X26 | concavity worst | [0.. 1,252] | [0,0239..1.17] |
| X27 | concave points worst | [0.. 0.0175] | [0,0289..0.291] |
| X28 | symmetry worst | [0.1566.. 0.4228] | [0,1565.. 0.6638] |

En la tabla 3, se presentan las estadísticas extraídas del *dataset* BreastCancer, con relación a lo expuesto en la etapa 3. Notar que para las clases 0 se han adicionado los valores para el máximo, mientras que para la clase 1 se han adicionado los valores para el mínimo. Esto es así como paso previo al cálculo de distancia de los *features* en común, como método adicional de comprobación que si al extender los límites de las cotas entre los valores mínimos y máximos de cada *feature* ayude a resolver el solapamiento entre clases.

Una vez extraídas las funciones que la red aprendió, y calculadas además las estadísticas para cada clase, se validan las reglas con los datos del *dataset* en su totalidad. Para este caso en particular, los *features* relevantes para las 2 clases son las mismas y estas se presentan en las tablas 3 y 4.

Para este caso, los *features* que identifico el algoritmo como críticos son suficientes, por lo tanto, no es necesario aumentar la cantidad de *features* a considerar como críticos utilizando neuronas del conjunto secundario.
Para el cálculo de distancia se utilizaron los *features área worst* ($X_{23}$) y *radius mean* ($X_0$) dado que estos son los *features* que identifico el algoritmo de extracción, y además ambos son *features* en común entre las clases. De ambos, el que mayor entropía presenta es *área worst* ($X_{23}$) por lo tanto, las métricas para el cálculo de distancia quedan expresadas como sigue:

$dst_X$ = (*area_worst*$_{(N)}$ – MinValue_ClassX('*radius_mean'*)) +
(MaxValue_ClassX('*radius_mean'*) – *area_worst* $_{(N)}$) por eq. 6.6

$dst_Y$ = (*area_worst*$_{(N)}$ – MinValue_ClassY('*radius_mean'*)) +
(MaxValue_ClassY('*radius_mean'*) – *area_worst* $_{(N)}$) por eq. 6.7

Luego de aplicar el cálculo en eq. 6.8, obtenemos Class = (X if $dst_X$ < $dst_Y$ else Y).

Finalmente, el valor obtenido en Class, se compara con el *ground truth* para la instancia N. Entonces la regla obtenida para las distintas clases del conjunto de datos Iris quedaría como sigue:

Class0 = ($\forall X_0$[6,98..13.37] ∧ $\forall X_1$[9,71..19.76] ∧ $\forall X_2$ [43,79..86,1] ∧ $\forall X_3$[143,5..992,1]
∧ $\forall X_6$[0..0,4108] ∧ $\forall X_7$[0..0,08534] ∧ $\forall X_{10}$[0,115..0,811] ∧ $\forall X_{12}$[0,758..5,118]
∧ $\forall X_{13}$[6,802..77,11] ∧ $\forall X_{20}$[7,93..19,82] ∧ $\forall X_{21}$[12,02..41,78] ∧ $\forall X_{22}$[50,41.. 127,1]
∧ $\forall X_{23}$[185,2..1210] ∧ $\forall X_{24}$[0,07117.. 0.2006] ∧ $\forall X_{26}$[0.. 1,252] ∧ $\forall X_{27}$[0.. 0,0175]
∧ $\forall X_{28}$[0,1566.. 0,4228])
^ fnc_dst_class(*area_worst*$_{(N)}$, *radius_mean*, 0, N) == 0

Class1 = ($\forall X_0$[10,95..28,11] ∧ $\forall X_1$[10,38..39,28] ∧ $\forall X_2$[71,9..188,5] ∧ $\forall X_3$[361,6..2501]
∧ $\forall X_6$[0,02398..0,4268] ∧ $\forall X_7$[0,0231..0,2012] ∧ $\forall X_{10}$[0,1938..2,873]
∧ $\forall X_{12}$[1,334..5,118] ∧ $\forall X_{13}$[13,99..542,2] ∧ $\forall X_{20}$[12,84..36,04]
∧ $\forall X_{21}$ [16,67..49,54] ∧ $\forall X_{22}$ [85,1..251,2] ∧ $\forall X_{23}$[508,1..4254] ∧ $\forall X_{24}$ [0,08822..
0,2226] ∧ $\forall X_{26}$[0,0239..1.17] ∧ $\forall X_{27}$ [0.0289..0.291] ∧ $\forall X_{28}$ [0,1565.. 0.6638])
^ fnc_dst_class(*area_worst*$_{(N)}$, *radius_mean*, 1, N) == 1

fnc_dst_class(fme, fcc, ClassX, ClassY) = $\exists X_n$ ($dst_x$ < $dst_y$)
Para este conjunto de datos, el algoritmo de explicación de reglas logra una precisión del 95%.

### 7.3.1 Redes regularizadas por clase – BreastCancer

Por cuestiones de espacio y comprensibilidad de la imagen, no se presenta la DNN regularizada para este caso. De todos modos, cabe señalar que, para este caso en particular donde la tarea puede ser de un problema de predicción o de clasificación, la red regularizada es presenta la misma topología para ambas clases. Esto se puede comprobar revisando las fbf.

## 7.4. Revisión empírica del algoritmo de distancia

En esta sección se explicará el funcionamiento de la función de distancia, con datos provenientes del conjunto de datos Wine, para un caso donde los valores de un $X_n$ presentan solapamiento entre clases.
El caso que se analiza es para el Id = 19 dentro del conjunto de datos, donde los valores de las X referidos a las neuronas críticas se solapan para las clases 0 y 1. Los valores para esta instancia son los siguientes:

- $X_1$ malic_acid = 3,1
- $X_6$ flavanoids = 3,03
- $X_9$ color_intensity = 5,1
- $X_{12}$ proline = 845

Estos valores hacen que el algoritmo identifique dos clases (0 y 1) diferentes, puesto que estos valores caen dentro de las cotas para cada variable y cada clase donde:

Class0 = ($\forall X_1$[1,35..4,04] $\wedge$ $\forall X_6$[2,19..3,93] $\wedge$ $\forall X_9$ [3,52..8,9] $\wedge$ $\forall X_{12}$[680..1680])
Class1 = ($\forall X_1$[0,74..5,8] $\wedge$ $\forall X_6$[0,57..5,08] $\wedge$ $\forall X_9$ [1,26..6] $\wedge$ $\forall X_{12}$[278..985])

Entonces, para resolver el solapamiento entre clases, aplicamos la función de distancia con el objetivo de verificar del centroide de que clase el *feature* de mayor entropía se encuentra más próximo. Los centroides vienen dados por los valores de las cotas del *feature* en común entre clases.
De este modo, para el caso propuesto se obtiene que:

- flavanoids min_class0 = 2,19
- flavanoids max_class0 = 3,93
- flavanoids min_class1 = 0,57
- flavanoids max_class1 = 5,08

dst0 = (*proline*$_{(N)}$ – MinValue_Class0('flavanoids')) + (MaxValue_Class0('flavanoids') - *proline* $_{(N)}$)

dst1 = (*proline*$_{(N)}$ – MinValue_Class1('flavanoids')) + (MaxValue_Class1('flavanoids') - *proline* $_{(N)}$)

dst0 = (845 – 2,19) + (3,93 – 845) = 1,74 ➔ Distancia al centro clase 0
dst1 = (845 – 0,57) + (5,08 – 845) = 4,51 ➔ Distancia al centro clase 1

Finalmente, para determinar la clase correcta se aplica:

Class = (0 if $dst_0$ < $dst_1$ else 1)
Class = (0 if 1,74 < 4,51 else 1)

Con lo cual la clase resultante es 0. Esta salida además es coincidente con el *ground truth* provisto en el origen de datos.

## 7.5. Reproducibilidad

Para finalizar, se destaca que todo el material utilizado para el desarrollo de este trabajo se encuentra disponible en el siguiente repositorio: https://gitlab.com/ia-pnegro/ia-tesis.git.
Los artefactos allí disponibles son:

- Conjuntos de datos.
- Diagramas.
- Código de configuración y entrenamiento de las DNNs.
- Código para extracción de reglas.
- Código para validación de reglas.

## 7.6. Conclusiones

En este capítulo se presentan ejemplos prácticos de aplicación del algoritmo COLOSSUS para extraer y validar exhaustivamente los conjuntos de reglas extraídos. Estos ejemplos se realizan sobre los conjuntos de datos Wine y BreastCancer, analizando las topologías de las redes neuronales profundas (DNN) con configuraciones específicas de neuronas y capas, así como las funciones de activación utilizadas. Los resultados presentados incluyen la precisión lograda por la red neuronal entrenada y la presentación de los conjuntos de reglas extraídas para cada clase, lo que demuestra los caminos neuronales mínimos necesarios para explicar cada clase. Esto permite identificar características críticas, y considerar características adicionales como críticas para mejorar la precisión de las explicaciones de las reglas y proporcionar información relevante sobre el conjunto de datos. En general, los hallazgos y metodologías presentados se pueden aplicar en escenarios prácticos, donde la explicabilidad es un factor clave.
En conclusión, el algoritmo COLOSSUS puede ser una herramienta útil para extraer conjuntos de reglas que expliquen las predicciones realizadas por la red neuronal. Esto puede ayudar a comprender mejor los resultados de la red neuronal y a mejorar la precisión de las predicciones.

CAPITULO 8

# Trabajos Relacionados

## 8.1. Trabajos Relacionados

En este capítulo se presentan los trabajos que guardan relación con el tema central de este trabajo de tesis.

### 8.1.1. Problema central

Hoy en día, la IA no-simbólica es el área más atractiva de IA y está relacionada con el aprendizaje automático (Alpaydin, 2020; Alpaydin, 2016; Bishop, 2006; Flach, 2012) más conocido como *Machine Learning* (ML). El ML es el estudio científico de algoritmos y modelos estadísticos que los sistemas informáticos utilizan para realizar de manera efectiva una tarea específica sin utilizar instrucciones explícitas, los cuales se basan en patrones e inferencias. Los algoritmos de ML construyen un modelo matemático a partir de datos de muestra (conocido como *datos de entrenamiento*), para hacer predicciones o tomar decisiones sin estar explícitamente programado para realizar la tarea.
El aprendizaje profundo es un enfoque del ML biológicamente inspirado que implica la formación de Redes Neuronales Artificiales (ANNs) con muchas capas que se entrenan iterativamente usando grandes conjuntos de datos (Yann LeCun, Yoshua Bengio 2015; Schmidhuber 2015). En los últimos diez años, se ha establecido como una de las áreas de investigación más impactantes dentro de la IA. Sus éxitos notables incluyen aplicaciones comercialmente importantes, como subtitulado de imágenes, por ejemplo, (Karpathy & Fei-Fei, 2015; Xu et al., 2015) y traducción automática, por ejemplo, (Vaswani et al., 2017), y se ha combinado fructíferamente con el aprendizaje de refuerzo en el contexto de la robótica (Levine et al., 2016), los videojuegos (Mnih et al., 2015), y juegos de mesa (Silver et al., 2016).

Para hacer frente a la falta de ***explicabilidad*** (AmirHosseini y Hosseini, 2019) proponen mejorar las compensaciones entre la precisión y la interpretabilidad del *fuzzy inference system* (FIS) ajustando los parámetros del modelo utilizando un algoritmo evolutivo que explora el espacio de búsqueda de manera más inteligente que otros algoritmos propuestos.
El trabajo propone un modelo híbrido de evolución diferencial difusa para la clasificación de tumores con metástasis hepáticas en imágenes de tomografía computarizada del hígado. Se utilizan sistemas de inferencia difusa, algoritmo de evolución diferencial, vector de distancia y operadores de mutación, cruce y selección para optimizar el sistema. Se evaluó el modelo con un conjunto de datos reales y se comparó con otros modelos existentes. El modelo propuesto proporciona alta precisión e interpretabilidad en el diagnóstico de tumores con metástasis hepáticas y puede ayudar a los médicos a diagnosticar precozmente el cáncer de hígado. Además, puede reducir el número de variables de entrada necesarias para el diagnóstico y mejorar la precisión del

diagnóstico. En general, el modelo propuesto puede tener importantes implicaciones prácticas en el campo de la imagen y el diagnóstico médicos.
En (Mahdavifar y Ghorbani, 2020) los autores proponen un sistema experto en redes neuronales profundas (DenNES) que extrae reglas refinadas de una arquitectura de redes neuronales profundas (DNN) entrenada para sustituir la base de conocimientos de un sistema experto. Las reglas extraídas se pueden utilizar para clasificar un incidente de seguridad invisible e informar al usuario final de la regla correspondiente que hizo esa inferencia. Los experimentos realizados muestran que DenNes supera a otros algoritmos en ambos conjuntos de datos de ciberseguridad y logra una precisión sobresaliente. Los profesionales de ciberseguridad pueden utilizar el marco propuesto para mejorar la precisión de sus sistemas de detección de amenazas y comprender mejor las causas de las ciberamenazas.

Por su parte (Cocarascu et al., 2018), presentan una metodología llamada ANNA (Artificial Neural Networks & Abstract Augmentation) que combina redes neuronales artificiales y argumentación abstracta para la predicción. ANNA proporciona predicciones basadas en argumentos y reglas lógicas que ofrecen predicciones equivalentes. Las predicciones hechas por ANNA se pueden explicar tanto dialéctica como lógicamente, y las explicaciones forman modelos de datos basados en argumentos y reglas. ANNA se puede utilizar en varios ámbitos en los que se requiere una IA explicable, como los sistemas sanitario, financiero y legal. ANNA puede ayudar a tomar mejores decisiones al proporcionar predicciones transparentes e interpretables. ANNA se puede utilizar para generar reglas lógicas que se pueden utilizar para comprender el proceso de toma de decisiones del modelo. ANNA se puede utilizar para identificar las características más importantes de los datos de entrada, lo que puede ayudar a seleccionar las funciones y al preprocesamiento de los datos. ANNA se puede utilizar para mejorar el rendimiento de los modelos existentes mediante la combinación de ANN y AA.
Durante el entrenamiento, utilizan un codificador automático para clasificar las características de los ejemplos de entrada, como más o menos representativos. Estos ejemplos de formación están etiquetados como pertenecientes a una de dos clases dadas (resultados). Luego se usa esa clasificación para seleccionar un subconjunto de características de mayor rango, de modo que la restricción de los ejemplos de entrenamiento de estas características sea coherente (es decir, sin dos restricciones que tengan las mismas características, pero diferentes resultados). La restricción resultante de los ejemplos de entrenamiento se asigna luego a un *framework* de argumentación abstracto que, a su vez, se asigna a un conjunto de reglas lógicas, de la variedad de la programación lógica.

En el trabajo desarrollado en (Csiszár et al., 2020), los autores proponen un marco coherente para modelar el pensamiento humano mediante el uso de herramientas de toma de decisiones multicriterio (MCDM) y lógica difusa. El objetivo es implementar la preferencia en redes neuronales e ilustrar los resultados en aplicaciones prácticas. El documento presenta los operadores de preferencia como una composición del operador agregativo y el operador de negación, y examina las principales propiedades del operador de preferencia en sistemas nilpotentes. Este marco coherente admite una posible aplicación en el campo de la inteligencia artificial, como un paso importante

hacia la interpretabilidad de los modelos neuronales. El trabajo propone una sugerencia sobre cómo aún se puede crear un marco teórico sintetizando los mundos de la lógica continua y MCDM, y examinan las principales propiedades del operador de preferencia en sistemas nilpotentes.
Este marco coherente admite una posible aplicación de los resultados en el campo de la inteligencia artificial, como un paso importante hacia la interpretabilidad de los modelos neuronales.

El objetivo del proyecto en (Schmid y Finzel, 2020), es combinar enfoques de aprendizaje profundo de caja negra, con aprendizaje automático interpretable para la clasificación de diferentes tipos de imágenes médicas, para combinar la precisión predictiva del aprendizaje profundo y la transparencia y comprensibilidad de modelos interpretables.
El trabajo presenta un proyecto de investigación interdisciplinario sobre el aprendizaje automático transparente para el apoyo a la toma de decisiones médicas. El objetivo del proyecto es combinar los enfoques del aprendizaje profundo con el aprendizaje automático interpretable para clasificar diferentes tipos de imágenes médicas a fin de combinar la precisión predictiva del aprendizaje profundo y la transparencia y la comprensibilidad de los modelos interpretables. El artículo propone un marco para utilizar las explicaciones mutuas para la toma de decisiones conjunta en medicina, que puede utilizarse para mejorar la precisión y la transparencia de los sistemas de apoyo a la toma de decisiones médicas. En general, el documento proporciona un marco práctico para mejorar la precisión y la transparencia de los sistemas de apoyo a la toma de decisiones médicas mediante la incorporación de conocimientos especializados y el suministro de explicaciones mutuas para la toma de decisiones conjunta.

En el estudio presentado en (Dombi y Csiszár, 2021) demuestran que los sistemas lógicos nilpotentes ofrecen un marco matemático apropiado para una hibridación de modelos neuronales y lógicos nilpotentes continuos, lo que ayuda a mejorar la interpretabilidad y la seguridad del aprendizaje automático.
Uno de los mayores desafíos es la creciente necesidad de abordar el problema de interpretabilidad para mejorar la transparencia del modelo, rendimiento y seguridad.
La combinación de redes neuronales profundas con reglas lógicas estructuradas y herramientas de decisión multicriterio, donde los operadores lógicos se aplican a los clústeres creados en la primera capa, contribuye a la reducción de la naturaleza de caja negra de los modelos neuronales.
Los autores proponen un método para reducir la naturaleza de caja negra de los modelos neuronales mediante la combinación de redes neuronales con herramientas de toma de decisiones de lógica continua y multicriterio. El objetivo es mejorar la interpretabilidad y la seguridad del aprendizaje automático. El modelo propuesto utiliza sistemas lógicos nilpotentes y ofrece una selección sencilla de funciones de activación. El método se puede utilizar en futuras aplicaciones en redes neuronales profundas y puede tener implicaciones prácticas en varios campos en los que se utiliza el aprendizaje automático.

En (Aghaeipoor et al., 2023), los autores proponen sistemas explicativos difusos basados en reglas para redes neuronales profundas, que pueden utilizarse con fines de

explicabilidad local y global. El algoritmo aprende un conjunto compacto pero preciso de reglas difusas basadas en la importancia de las características (es decir, los valores de atribución) extraídas de las redes entrenadas. El método propuesto puede ayudar a aclarar el proceso de decisión de las DNN en los problemas de clasificación de datos tabulares. Los resultados de la evaluación de diferentes aplicaciones revelaron que las explicaciones difusas mantenían la fidelidad y la precisión de las redes neuronales profundas originales, al tiempo que implicaban una menor complejidad y una mejor comprensibilidad. El método propuesto se puede utilizar en varios campos, como la sanidad, las finanzas y la conducción autónoma, donde la interpretabilidad de las DNN es crucial para garantizar la auditabilidad y la confiabilidad.
Estos sistemas se pueden utilizar con fines de explicabilidad local y global.

- El trabajo propone sistemas explicativos difusos basados en reglas para redes neuronales profundas, que pueden utilizarse con fines de explicabilidad local y global.
- El algoritmo aprende un conjunto compacto pero preciso de reglas difusas basadas en la importancia de las funciones (es decir, los valores de atribución) extraídas de las redes entrenadas.
- El método propuesto puede ayudar a aclarar el proceso de decisión de las DNN en los problemas de clasificación de datos tabulares.
- El uso de la representación lingüística difusa en el método propuesto modela el conocimiento semántico del espacio de entrada de una manera similar a la cognición humana, lo que refuerza la comprensibilidad de los modelos sustitutos basados en reglas.
- Los resultados de la evaluación de diferentes aplicaciones revelaron que las explicaciones difusas mantenían la fidelidad y la precisión de las redes neuronales profundas originales, al tiempo que implicaban una menor complejidad y una mejor comprensibilidad.

En (Ciravegna et al., 2023) se presenta los métodos fundamentales utilizados para implementar las redes explicadas por lógica (LEN - Logic Explained Networks).
Los autores describen el procedimiento utilizado para extraer reglas lógicas de los LEN para una observación individual o un grupo de muestras. Este procedimiento es común a todos los objetivos y casos de uso. Además, se proporciona la definición formal de los objetivos de aprendizaje que limitan a los LeN a proporcionar los tipos requeridos de explicaciones de la lógica de primer orden (FOL). Por último, el artículo analiza cómo restringir los LENs para que produzcan fórmulas lógicas concisas.
El artículo propone un enfoque general de la inteligencia artificial explicable (XAI) en el caso de las arquitecturas neuronales, y muestra cómo un diseño consciente de las redes conduce a una familia de modelos de aprendizaje profundo interpretables denominados redes lógicas explicadas (LENs).
Los LEN propuestos solo requieren que sus entradas sean predicados comprensibles para los humanos y proporcionan explicaciones en términos de fórmulas simples de lógica de primer orden (FOL) que incluyan dichos predicados.
El documento muestra que los LEN son lo suficientemente generales como para cubrir una gran cantidad de escenarios, incluidos los problemas de aprendizaje supervisados y no supervisados.

Los resultados experimentales de varios conjuntos de datos y tareas muestran que los LEN pueden generar mejores clasificaciones que los modelos de caja blanca establecidos, como los árboles de decisión y las listas de reglas bayesianas, al tiempo que proporcionan explicaciones más compactas y significativas.
El documento destaca la importancia de proporcionar explicaciones comprensibles en situaciones en las que se espera que la máquina respalde la decisión de los expertos humanos. El trabajo contribuye a la investigación en curso sobre el XAI, que ha pasado a ser estratégica y se ha fomentado enormemente debido a la necesidad de explicaciones comprensibles para los humanos en varios ámbitos de aplicación.

En (Garcez et al. 2019), se analiza la computación simbólica neuronal para el aprendizaje automático y el razonamiento integrados. Destaca la importancia de integrar el aprendizaje neuronal con métodos de razonamiento sólidos basados en el simbolismo para desarrollar sistemas y herramientas explicables y responsables basados en la IA y el aprendizaje automático. El documento proporciona una metodología basada en principios para integrar el aprendizaje automático y el razonamiento, que puede conducir al desarrollo de sistemas de IA más efectivos y eficientes con implicaciones prácticas para varios dominios. En general, el artículo proporciona una valiosa contribución al campo de la IA y el aprendizaje automático.
Algunos de los enfoques metodológicos clave que se destacan en el trabajo incluyen:

- La integración basada en principios del aprendizaje neuronal con la representación y el razonamiento simbólicos del conocimiento.
- El uso del mapeo de sonidos entre reglas simbólicas y redes neuronales proporcionado por métodos de computación neural-simbólica.
- El uso de diferentes formalismos de representación del conocimiento como conocimiento básico para un aprendizaje potencial a gran escala y un razonamiento eficiente.
- El uso de la proposicionalización para aprender la lógica de primer orden en redes bayesianas.
- El uso de sistemas de incrustación relacional para razonar sobre las relaciones entre entidades.
- El uso de redes neuronales profundas para el razonamiento visual, donde aprenden y deducen relaciones y características de múltiples objetos en imágenes.

El trabajo proporciona un estudio sobre los logros recientes de la computación simbólica neuronal como metodología basada en principios para integrar el aprendizaje automático y el razonamiento. Además, se destacan las características clave de la computación simbólica neuronal, incluida la integración basada en principios del aprendizaje neuronal con la representación simbólica del conocimiento y el razonamiento, lo que permite construir sistemas de IA explicables.
El documento enfatiza la importancia de integrar el aprendizaje neuronal con métodos sólidos de razonamiento basados en el simbolismo para desarrollar sistemas y herramientas explicables y responsables basados en la IA y el aprendizaje automático.
El documento proporciona información sobre la necesidad cada vez más importante de sistemas de IA interpretables y responsables.

El artículo concilia las ventajas del aprendizaje robusto en redes neuronales y el razonamiento y la interpretabilidad de la representación simbólica.
El documento proporciona un marco para unir el procesamiento de la información de nivel inferior (para la percepción y el reconocimiento de patrones) y el conocimiento abstracto de nivel superior (para el razonamiento y la explicación) en los sistemas de IA.

En (Burkhardt et al. 2021), se propone un método para extraer reglas lógicas de redes neuronales binarias utilizando reglas convolucionales de primer orden. El método se basa en la extracción de reglas de redes neuronales binarias con búsqueda local estocástica, y se puede utilizar para extraer reglas interpretables de redes neuronales binarias, validar modelos y datos de entrada de alta dimensión, como imágenes. El artículo presenta un enfoque prometedor para extraer reglas interpretables de las redes neuronales binarias, lo que puede tener implicaciones prácticas en varios dominios. Para ello, se desarrolló un algoritmo para la extracción de reglas de descomposición, denominado Deep Convolutional DNF Learner (DCDL), que puede extraer reglas características para datos de entrada de alta dimensión, como imágenes. Además, se demostró el potencial de los enfoques basados en reglas para las imágenes, que permite combinar las ventajas de las redes neuronales y el aprendizaje de reglas. El uso de redes neuronales binarias (BNN) para la extracción de reglas también se propuso, lo que puede conducir a un mejor rendimiento e interpretabilidad del modelo. Por último, se propuso un método para la validación de modelos mediante reglas lógicas rigurosas, y se visualizaron las reglas extraídas, lo que puede ayudar a comprender la funcionalidad de la red neuronal.

En (Barbiero et al., 2022), se propone un novedoso enfoque diferenciable de extremo a extremo que permite extraer explicaciones lógicas de redes neuronales utilizando el formalismo de la lógica de primer orden. El método se basa en un criterio basado en la entropía que identifica automáticamente los conceptos más relevantes. El artículo considera cuatro estudios de casos diferentes para demostrar que este criterio basado en la entropía permite extraer explicaciones lógicas concisas en ámbitos críticos para la seguridad, desde los datos clínicos hasta la visión por computadora. El documento define seis métricas cuantitativas para comparar el enfoque propuesto con los métodos más avanzados. El artículo también presenta una función decodificadora conceptual g para derivar la fórmula FOL en el caso de características de entrada no similares a las de un concepto. Por lo tanto, los métodos utilizados en este artículo incluyen un criterio basado en la entropía, la lógica de primer orden y una función de decodificación de conceptos.
El método se basa en un criterio basado en la entropía que identifica automáticamente los conceptos más relevantes, lo que permite extraer explicaciones lógicas concisas en ámbitos críticos para la seguridad, desde los datos clínicos hasta la visión artificial.
En este trabajo definen seis métricas cuantitativas para comparar el enfoque propuesto con los métodos más avanzados. El enfoque propuesto supera a los modelos de caja blanca de última generación en términos de precisión de clasificación y coincide con los rendimientos de las cajas negras.
El estudio presenta una función de decodificación conceptual g para derivar la fórmula FOL en el caso de funciones de entrada no similares a las de un concepto.

El documento demuestra la eficacia del enfoque propuesto a través de cuatro estudios de casos diferentes, lo que demuestra que se puede aplicar a una variedad de dominios, incluidos los datos clínicos y la visión por computadora.
En el trabajo desarrollado en (Tran, 2017), se propone un método para integrar el conocimiento simbólico en redes neuronales no supervisadas. El método se basa en el hallazgo teórico de que cualquier fórmula proposicional puede representarse en máquinas restringidas de Boltzmann (RBM). Los autores muestran cómo codificar el conocimiento simbólico en forma proposicional y de primer orden en la RBM ampliando la teoría de un trabajo anterior. El enfoque de codificación se evalúa mediante experimentos en dos dominios: la predicción de los promotores del ADN y la comprensión de las relaciones familiares. El artículo también analiza la idea de la regla de confianza, un formulario de conocimiento para representar fórmulas simbólicas en los RBM, y muestra cómo codificar el conocimiento en los RBM. El conjunto final de reglas se presenta en el documento.
El estudio propone un método para integrar el conocimiento simbólico en redes neuronales no supervisadas, que puede ofrecer un mejor aprendizaje y razonamiento, al tiempo que proporciona un medio de interpretabilidad mediante la representación del conocimiento simbólico.
El estudio muestra cómo codificar el conocimiento simbólico en forma proposicional y de primer orden en la RBM (Restricted Boltzmann Machine) ampliando la teoría de un trabajo anterior. El enfoque de codificación presentado en el documento se puede utilizar para codificar el conocimiento en RBM, que se pueden aplicar a varios dominios. El estudio analiza la idea de la regla de confianza, una forma de conocimiento para representar fórmulas simbólicas en los RBM, y muestra cómo codificar el conocimiento en los RBM. El artículo presenta el conjunto final de reglas para codificar el conocimiento simbólico en redes neuronales no supervisadas.
Los experimentos realizados en el artículo muestran la validez del enfoque en dos dominios: la predicción de los promotores del ADN y la comprensión de las relaciones familiares.

En (Dombi y Csiszár, 2021), los autores proponen un enfoque híbrido que combina redes neuronales con herramientas de toma de decisiones de lógica continua y multicriterio para mejorar la interpretabilidad y la seguridad del aprendizaje automático. El enfoque utiliza lógica nilpotente continua y operadores lógicos continuos para modelar desigualdades blandas y reduce el número de parámetros que deben aprenderse. Las implicaciones prácticas incluyen la mejora de la interpretabilidad y la seguridad del aprendizaje automático en diversos dominios, como la salud, las finanzas y la ingeniería. El artículo presenta un modelo neuronal nilpotente, un operador basado en umbrales y una selección sencilla de funciones de activación. En resumen, el artículo propone un enfoque novedoso que puede mejorar la interpretabilidad y la seguridad del aprendizaje automático y ofrece varias contribuciones importantes.

En el trabajo presentado en (Wang y Pan, 2020), los autores proponen un marco que integra el conocimiento lógico en forma de lógica de primer orden en un sistema de aprendizaje profundo para la extracción de información. El marco consta de tres componentes: una red neuronal profunda, un banco lógico y una unidad de discrepancia. El marco se entrena conjuntamente de principio a fin, lo que permite

integrar el conocimiento lógico con las capacidades de aprendizaje de las redes neuronales profundas. La efectividad y la generalización del modelo propuesto se demuestran en múltiples tareas de extracción de información. El marco propuesto tiene implicaciones prácticas en términos de rendimiento mejorado, generalización, destilación del conocimiento y adaptabilidad. En general, el marco propuesto tiene el potencial de mejorar la precisión y la generalización de las tareas de extracción de información, lo que puede tener implicaciones prácticas en varios dominios.

Giunchiglia et al., 2022, analiza varios métodos que utilizan restricciones y razonamientos lógicos en los modelos de aprendizaje profundo para lograr un mejor rendimiento, aprender de menos datos y garantizar el cumplimiento de los conocimientos básicos para las aplicaciones críticas para la seguridad. Los métodos se clasifican según el lenguaje lógico utilizado para expresar los conocimientos básicos y los objetivos logrados. El uso de restricciones lógicas puede mejorar el rendimiento, aprender de menos datos, garantizar el cumplimiento de los conocimientos básicos y mejorar la interpretabilidad de los modelos de aprendizaje profundo. El documento proporciona un recurso valioso para los investigadores y profesionales interesados en utilizar restricciones lógicas en los modelos de aprendizaje profundo y destaca los beneficios potenciales de este enfoque para diversas aplicaciones prácticas.
El documento analiza los siguientes métodos:

- *Métodos basados en pérdidas:* estos métodos utilizan funciones de pérdida para entrenar redes neuronales con restricciones. No pueden garantizar la satisfacción de las restricciones, pero pueden generalizar a partir de menos datos y/o aprovechar los datos no etiquetados.
- *Métodos basados en reglas:* estos métodos utilizan reglas lógicas para expresar las restricciones e incorporarlas al proceso de aprendizaje. Pueden garantizar la satisfacción de las restricciones, pero pueden requerir más datos para lograr un buen rendimiento.
- *Métodos híbridos:* estos métodos combinan métodos basados en pérdidas y en reglas para lograr un mejor rendimiento y satisfacer las restricciones.
- *Métodos de programación lógica inductiva (ILP*): estos métodos utilizan técnicas de ILP para aprender reglas lógicas a partir de ejemplos y conocimientos básicos. Pueden aprender reglas complejas y garantizar la satisfacción de las restricciones, pero pueden requerir más recursos computacionales.

En conclusión, el trabajo analiza varios métodos que utilizan restricciones y razonamientos lógicos en los modelos de aprendizaje profundo para lograr un mejor rendimiento, aprender de menos datos y garantizar el cumplimiento de los conocimientos básicos para las aplicaciones críticas para la seguridad.

En (Dai y Muggleton, 2021), los autores proponen un enfoque innovador llamado aprendizaje metainterpretativo abductivo (Meta Abd) que combina la abducción y la inducción para aprender redes neuronales e inducir teorías lógicas de primer orden de forma conjunta a partir de datos sin procesar. El enfoque supera a los modelos de aprendizaje profundo de extremo a extremo y a los métodos neuro-simbólicos de última generación en términos de precisión predictiva y eficiencia de datos. El sistema también

permite la integración de redes neuronales con la programación lógica inductiva (ILP) para admitir la inducción de la teoría lógica de primer orden a partir de datos sin procesar, permitiendo la reutilización de los programas lógicos y las redes neuronales en tareas posteriores. Esto tiene implicaciones prácticas en diversos dominios y puede ser útil para comprender el razonamiento detrás de los modelos aprendidos.

En (De et al. 2020) los autores introducen un enfoque híbrido que combina agrupamiento y árboles de decisión para explicar las predicciones realizadas por redes neuronales profundas. Este método permite la visualización del flujo de información en la red, proporcionando explicaciones interpretables por humanos para predicciones individuales. La eficacia de este enfoque se demuestra en el contexto de la predicción de impagos de tarjetas de crédito, mostrando su capacidad para generar códigos de motivo precisos y localizados.

En (Angelov y Soares 2019) los autores proponen una nueva arquitectura de aprendizaje profundo, llamada xDNN, que supera las limitaciones de los métodos tradicionales al ofrecer una arquitectura interna transparente. Utiliza prototipos derivados de datos de entrenamiento para combinar eficientemente el razonamiento y el aprendizaje. El modelo no es iterativo ni paramétrico, lo que resulta en una mayor eficiencia en términos de tiempo y recursos computacionales. Su transparencia, con reglas IF...THEN basadas en prototipos, lo hace adecuado para aplicaciones de alto riesgo en campos como los vehículos autónomos y la atención médica.

(Samek et al. 2021) ofrece una descripción general completa de las técnicas de IA explicables para redes neuronales profundas, con un enfoque en explicaciones post hoc. Se analizan varios métodos y aplicaciones para interpretar las predicciones realizadas por modelos de aprendizaje profundo, destacando la importancia de las explicaciones en este contexto. El texto cubre enfoques populares para explicar las redes neuronales profundas, ayudando a investigadores y profesionales a elegir la técnica más adecuada para su caso de uso específico. Estas explicaciones pueden mejorar la comprensión del comportamiento del modelo, identificar sesgos y proporcionar información sobre los procesos de toma de decisiones, lo que en última instancia facilitará el desarrollo y la implementación de modelos de aprendizaje profundo más transparentes y confiables. Esto es particularmente relevante en ámbitos donde la interpretabilidad es crucial, como la salud, las finanzas y los sistemas autónomos.

(Amarasinghe, Kenney, y Manic 2018) propone una nueva arquitectura de red neuronal profunda para la detección de anomalías que combina redes neuronales convolucionales y recurrentes. También presenta un nuevo conjunto de datos para evaluar métodos de detección de anomalías en el contexto de la ciberseguridad y proporciona información sobre la interpretabilidad de los modelos de aprendizaje profundo para esta tarea. El modelo propuesto ofrece un nuevo enfoque que combina técnicas de aprendizaje profundo con interpretabilidad para mejorar el rendimiento y la comprensión de los sistemas de detección de anomalías. Las implicaciones prácticas de esta investigación incluyen un rendimiento mejorado de la detección de anomalías, interpretabilidad en la detección de anomalías, medidas de ciberseguridad mejoradas y

orientación para futuras investigaciones sobre el desarrollo de sistemas de detección de anomalías transparentes y confiables.

### 8.1.2. Problemas relacionados

John McCarthy (1956) definió la IA como: *La ciencia e ingeniería de hacer máquinas inteligentes*. Más recientemente (Russell y Norvig, 2010) dan una definición más moderna: *El estudio y diseño de agentes inteligentes, donde un agente inteligente es un sistema que percibe su entorno y toma acciones que maximizan su rendimiento*.
A pesar de su éxito innegable, varias investigaciones evidencian una serie de deficiencias en el aprendizaje profundo contemporáneo (Garnelo et al., 2016); Lake et al., 2017; G. Marcus, 2018; Battaglia et al., 2018).
El trabajo presentado en (Tsividis et al., 2017), trata sobre el problema de la **Ineficiencia de datos** (alta complejidad de la muestra). Las redes neuronales actuales requieren grandes volúmenes de datos de entrenamiento para ser efectivas. Por ejemplo, un sistema típico de aprendizaje profundo por refuerzo (*deep reinforcement learning*, DRL) que logra puntajes sobrehumanos en un videojuego ve muchos millones de fotogramas, mientras que un humano (basándose en años de otra experiencia) requiere solo unos pocos cientos de fotogramas para comprender la idea de tal juego (el resto es en gran medida una cuestión de perfeccionar una habilidad motora).
En (Garnelo et al., 2016) se sugiere que las redes neuronales ofrecen **poca generalización.** Las redes neuronales actuales son propensas a fallar desastrosamente cuando se exponen a datos fuera de la distribución en la que fueron entrenadas. Por ejemplo, cambiar solo el color o el tamaño de un sprite en un videojuego podría obligar a un agente de DRL entrenado a volver a aprender el juego desde cero. Un sello distintivo de la inteligencia humana, por el contrario, es la capacidad de reutilizar la experiencia y los conocimientos adquiridos previamente, para transferirla a desafíos radicalmente diferentes.
En (Evans et al., 2018) las redes neuronales **carecen de la capacidad de razonar en un nivel abstracto**, lo que dificulta la implementación de funciones cognitivas de alto nivel, como el aprendizaje de transferencia, el razonamiento analógico y el razonamiento basado en hipótesis. Se han realizado varios intentos recientes para imitar procesos lógicos en redes neuronales
La **falta de interpretabilidad** de las redes neuronales hace que estas sean típicamente *cajas negras*. Los cálculos realizados por capas sucesivas rara vez corresponden a pasos de razonamiento humanamente comprensibles, y los vectores intermedios de activaciones que se generan carecen de una semántica humanamente comprensible (Garnelo et al., 2016).
El problema planteado en (Harnad, 1990) referido a **Symbol Grounding,** expone una limitación importante de la IA simbólica, y se refiere a la medida en que sus elementos simbólicos están hechos a mano en lugar de aprender de los datos. Por el contrario, uno de los puntos fuertes del aprendizaje profundo es su capacidad para descubrir características en datos de gran dimensión con poca o ninguna intervención humana.

### 8.1.3. Identificaron previamente el problema

Se han propuesto soluciones en (Santoro et al., 2017), a través de la implementación de una red neuronal con capacidad para realizar razonamiento relacional. En este trabajo se postula que, *el razonamiento relacional es un componente central del comportamiento generalmente inteligente, pero ha demostrado ser difícil de aprender para las redes neuronales*. Se describe cómo una Red Relacional (RN) como un módulo simple para resolver problemas que fundamentalmente dependen del razonamiento relacional. Estas redes aumentadas fueron testeadas en tres ámbitos: respuesta visual a preguntas utilizando un conjunto de datos desafiante; Respuesta a preguntas basadas en texto, utilizando el conjunto de tareas; y razonamiento complejo sobre sistemas físicos dinámicos. En este trabajo se demuestra que, usando un conjunto de datos *curado*, las poderosas redes convolucionales no tienen una capacidad para resolver cuestiones relacionales, pero pueden adquirir estas capacidades cuando se aumentan con RN. Por lo tanto, aumentando convoluciones, LSTM y MLP con RN, se puede eliminar la carga computacional de los componentes de red que no son adecuados para manejar el razonamiento relacional, reducir la complejidad general de la red, y adquirir una capacidad general para razonar sobre las relaciones entre las entidades y sus propiedades.

El trabajo realizado por (Garnelo et al., 2016), desarrolla un mecanismo sobre *Aprendizaje profundo por refuerzo simbólico.* Los autores desarrollan la idea basada en que el aprendizaje por refuerzo profundo (DRL) aporta el poder de las redes neuronales profundas para soportar la tarea genérica del aprendizaje por prueba y error, y su eficacia ha sido demostrada de manera convincente en tareas como los videojuegos. Sin embargo, los sistemas DRL contemporáneos heredan una serie de deficiencias de la generación actual de técnicas de aprendizaje profundo. Por ejemplo, requieren de grandes conjuntos de datos para que funcionen de manera eficaz, lo que implica que son lentos para aprender incluso cuando tales conjuntos de datos están disponibles y normalizados. Además, *carecen de la capacidad de razonar sobre una base abstracta*, lo que dificulta la implementación de funciones cognitivas de alto nivel como transferencia de aprendizaje, razonamiento analógico y razonamiento basado en hipótesis. Finalmente, su funcionamiento es en gran parte *opaco para los humanos*, lo que los hace inadecuados para dominios en el que la verificabilidad es importante. El estudio, propone una arquitectura de aprendizaje por refuerzo que comprende un back-end neuronal y un front-end simbólico con el potencial de superar cada una de estas deficiencias. Como prueba de concepto, presentan una implementación preliminar de la arquitectura y la aplicaron en variantes de un videojuego simple. El sistema resultante, aunque solo sea un prototipo, aprende de manera efectiva, al adquirir un conjunto de reglas que son fácilmente comprensibles para los humanos, y supera dramáticamente a un sistema convencional, completamente neuronal en una variante estocástica del juego.

A modo de contrastación, y a diferencia de los trabajos presentados por (Santoro et al., 2017) y (Garnelo et al., 2016), en el presente trabajo de tesis se introduce un modelo topológico distinto, dado que la red tendría un modelo donde el motor de inferencia trabajaría con la topología y los pesos de la red neuronal entrenada, extrayendo las

reglas a partir de allí. Luego las reglas se explican con LPO, a través de la conformación de fórmulas bien formadas, las cuales resulten comprensibles para el usuario final, y puedan justificarse con los datos de entrada.

Otros tipos de modelos se han propuesto que soportan computación basada en relaciones tales como, Graph Neural Networks, Gated Graph Sequence Neural Netoworks, e Interaction Networks (Li et al., 2016; Battaglia et al., 2016; Scarselli et al., 2009).

### 8.1.4. Misma metodología

La metodología respecto a la recolección de datos y prueba en el presente estudio es similar al utilizado en (Garnelo et al., 2016; Santoro et al., 2017). A diferencia de estos modelos, en el mecanismo para la recolección de datos, respecto al dataset privado, es que se haría a través de un algoritmo de composición de la información. En los trabajos mencionados anteriormente, si bien exponen resultados de las pruebas, en ningún momento indicaron que métricas utilizaron, ni como fue el proceso de medición.
En el presente trabajo de tesis, estas mediciones y comparaciones, se harían utilizando las siguientes métricas: Matriz de confusión (Presition, Recall, Accuracy), cross-validation, error cuadrático medio, Convergencia.

### 8.1.5. Cuál es el grado de originalidad del trabajo planteado

El objetivo de este modelo híbrido es resolver la incapacidad de las redes neuronales para explicar su proceso de aprendizaje. A través del logro de extraer y explicar las reglas aprendidas por la red neuronal en un lenguaje natural para un usuario final, se podría también estudiar y mejorar el proceso de aprendizaje. Este modelo puede ser la semilla para la creación de modelos mejorados.
La originalidad de este trabajo radica en su capacidad para extraer reglas de redes neuronales profundas y explicarlas con técnicas de lógica de primer orden. Esto hace que los sistemas de IA sean más transparentes y comprensibles para los usuarios finales que han sido objeto de alguna decisión algorítmica tomada por una red neuronal sobre ellos.
Además, este trabajo busca maximizar el alcance de las técnicas de aprendizaje al combinar la potencia de las redes neuronales con la expresividad de la lógica proposicional de primer orden, lo que combina dos enfoques conceptualmente muy diferentes.

## 8.2. Comparación de Métodos

En la presente sección se realiza el análisis de otros trabajos relacionados a métodos de explicabilidad sobre DNN, con el objetivo de hacer una comparación de modelos y poner en relieve los beneficios que aporta el algoritmo COLOSSUS.

### 8.2.1. Extracting Comprehensible Rules from Neural Networks via Genetic Algorithms

El trabajo desarrollado en (Santos et al., 2000), se propone un método para extraer reglas precisas y comprensibles de redes neuronales utilizando un algoritmo genético para encontrar una buena topología de red. Las reglas extraídas se evalúan y utilizan para generar nuevas topologías candidatas. El enfoque se evalúa en tres conjuntos de datos y muestra resultados prometedores. Ofrece una alternativa potencial a los algoritmos tradicionales de extracción de reglas y reconoce la necesidad de realizar más investigaciones y experimentaciones.
El trabajo plantea utilizar un algoritmo genético para encontrar una topología de red neuronal *feedforward* entrenada por el algoritmo RPROP. Luego, esta topología se pasa a un algoritmo de extracción de reglas para extraer un conjunto de reglas de clasificación. La calidad de estas reglas se evalúa en función de la precisión predictiva y la comprensibilidad, que se utiliza como valor de idoneidad para el algoritmo genético. El algoritmo genético se implementa utilizando la herramienta ENZO, que desarrolla una red neuronal *feedforward* totalmente conectada con una única capa oculta. La herramienta ENZO se amplía con un nuevo módulo para extraer reglas de la topología de la red neuronal entrenada. El módulo de extracción de reglas es una adaptación del algoritmo de extracción de reglas RX, elegido por su simplicidad y buenos resultados reportados en la literatura. Sin embargo, otros algoritmos de extracción de reglas podrían aplicarse también.
El trabajo evalúa un método para extraer reglas de redes neuronales utilizando tres conjuntos de datos. El enfoque se compara con un algoritmo de árbol de decisión y una red neuronal desarrollada sin un algoritmo de extracción de reglas. Los experimentos utilizan validación cruzada y las reglas finales se generan fusionando todas las particiones. El método se aplica utilizando el 70% de ejemplos para capacitación y el 30% para validación.

### 8.2.2. An Explicitly Relational Neural Network Architecture

El trabajo desarrollado en (Shanahan et al., 2020) los autores presentan una novedosa arquitectura de red neuronal de extremo a extremo, llamada PrediNet, que aprende a formar representaciones proposicionales con una estructura explícitamente relacional a partir de datos de píxeles sin procesar. La arquitectura está previamente entrenada en un plan de estudios de tareas simples de razonamiento visual relacional, lo que le permite generar representaciones reutilizables que facilitan el aprendizaje posterior en tareas nunca vistas anteriormente. Esto conduce a una mejora en la eficiencia, generalización y las propiedades de transferencia de los datos. Los antecedentes estructurales de la arquitectura permiten representaciones compatibles con el cálculo de predicados, lo que las hace adecuadas para procesos posteriores de tipo lógico. Los aportes del trabajo abren nuevas posibilidades para el uso de redes neuronales en tareas que requieren razonamiento relacional.
La arquitectura comprende tres etapas: atención, vinculación y evaluación. La etapa de atención selecciona pares de objetos de interés, la etapa de vinculación instancia los dos primeros argumentos de predicados de tres lugares con pares de objetos seleccionados, y la etapa de evaluación calcula valores para el argumento restante de cada predicado para generar una proposición verdadera. El módulo PrediNet consta de varios cabezales,

cada uno de los cuales calcula el mismo conjunto de relaciones, pero selecciona un par diferente de objetos utilizando la atención del producto escalar basándose en la coincidencia de consultas clave. El espacio clave se comparte, lo que garantiza la coherencia entre los cabezales. La entrada a PrediNet es una matriz de vectores de características calculada por una red neuronal convolucional (CNN) a partir de una imagen de entrada. Cada cabeza calcula las relaciones entre pares de objetos utilizando pesos compartidos.
La arquitectura PrediNet logró una alta precisión en diversas tareas de razonamiento relacional, superando a las arquitecturas básicas. Logró más del 90 % de precisión en todas las tareas con ambos conjuntos de objetos retenidos después de 100 000 lotes. Específicamente, en la tarea 'xoccurs', PrediNet superó las líneas base en más de un 10%, y en la tarea 'forma de color', superó todas las líneas base excepto MHA en un 25% o más. Se demostró la capacidad de la arquitectura para generar representaciones reutilizables, lo cual facilitó el aprendizaje posterior en tareas nunca vistas, lo que condujo a una mayor eficiencia y generalización de los datos.

Ideas principales

1. La arquitectura PrediNet logró una alta precisión en tareas de razonamiento relacional.
2. Logró más del 90 % de precisión en todas las tareas con conjuntos de objetos expuestos.
3. Superó las líneas de base en más del 10 % en la tarea 'xoccurs'
4. Capacidad para generar representaciones reutilizables.
7. Facilita el aprendizaje posterior sobre tareas invisibles.
8. Mejora de la eficiencia y generalización de los datos.
9. Se visualizó el entrenamiento exitoso del modelo.

### 8.2.3. Efficient Decompositional Rule Extraction for Deep Neural Networks

En trabajo presentado por (Zarlenga, Shams, y Jamnik, 2021) se introduce ECLAIRE, un algoritmo de extracción de reglas de tiempo polinomial para DNN que considera el espacio latente. Para la evaluación, los autores utilizan conjuntos de reglas inducidas a partir de árboles C5.0 como extractor de reglas intermedio para ECLAIRE, DeepRED y REM-D. Los autores también incluyen líneas de base no descomposicionales, como el uso de C5.0 como método de inducción de reglas de un extremo a otro. Está disponible la biblioteca REMIX de código abierto, que incluye ECLAIRE y una interfaz de visualización de conjuntos de reglas.
Es eficiente y práctico para aplicaciones del mundo real y ha sido evaluado en diversas tareas. La biblioteca REMIX de código abierto, que incluye ECLAIRE y una interfaz de visualización de conjuntos de reglas, permite una fácil implementación y utilización. ECLAIRE mejora la interpretabilidad de las DNN y tiene implicaciones prácticas en ámbitos como la atención sanitaria y la detección de partículas.
El algoritmo ECLAIRE propuesto proporciona una solución práctica para aumentar la interpretabilidad y depuración de redes neuronales profundas (DNN) mediante la extracción de modelos basados en reglas que se aproximan al límite de decisión de las

DNN. ECLAIRE es capaz de escalar a grandes arquitecturas DNN y grandes conjuntos de datos de entrenamiento, lo que lo hace adecuado para aplicaciones del mundo real.
El algoritmo extrae consistentemente conjuntos de reglas más precisas y comprensibles en comparación con los métodos más avanzados, lo que mejora la interpretabilidad de las DNN.
ECLAIRE utiliza órdenes de magnitud menos recursos computacionales, lo que lo hace más eficiente y rentable. Las implicaciones prácticas de ECLAIRE incluyen una mejor comprensión y confianza en el proceso de toma de decisiones de las DNN, lo que permite una mejor toma de decisiones en diversos ámbitos, como la atención sanitaria (pronóstico del cáncer de mama) y la detección de partículas.

### 8.2.4. Rule Extraction from Neural Network by Genetic Algorithm with Pareto Optimization

El método presentado en (Markowska-Kaczmar y Wnuk-Lipiński, 2004), los autores presentan un nuevo método llamado *GenPar* para la extracción de reglas de una red neuronal utilizando un enfoque genético con optimización de Pareto. El método es independiente de la arquitectura de la red neuronal y se puede aplicar a cualquier tipo de red neuronal. No requiere ninguna regla de aprendizaje especial durante la extracción de reglas, lo que facilita su uso. El método funciona de manera eficiente tanto para atributos continuos como enumerados. La optimización de Pareto utilizada en el método permite al usuario indicar qué criterio es más importante para él. El método desarrollado trata la red neuronal como una caja negra, lo que la hace aplicable incluso sin acceso al funcionamiento interno de la red. En general, el artículo aporta un método novedoso para la extracción de reglas de redes neuronales, proporcionando un enfoque flexible y eficiente que se puede aplicar a varios tipos de redes y tipos de atributos. Las implicaciones prácticas de este artículo incluyen un método flexible y eficiente para la extracción de reglas de redes neuronales, aplicable a diversas arquitecturas de red y tipos de atributos, sin la necesidad de reglas de aprendizaje especializadas. La versatilidad del método y su capacidad para manejar diferentes tipos de atributos lo hacen adecuado para aplicaciones del mundo real.
Este método es flexible, aplicable a diversas arquitecturas de red y tipos de atributos, e implica el uso de funciones de aptitud, operadores genéticos y una estructura cromosómica. El uso de la optimización de Pareto permite priorizar criterios específicos durante la extracción de reglas.

Ideas principales

1. GenPar es un método de extracción de reglas que se probó en conjuntos de datos de referencia.
2. Logró una fidelidad del 96%, comparable a otros métodos como FullRe.
3. GenPar fue eficaz a la hora de encontrar reglas para los problemas Monk, incluido el desafiante problema de Monk-2.
4. También es escalable y logró una fidelidad del 90 % en el conjunto de datos de Mamíferos.
5. En el problema de los LED, GenPar superó a la extracción directa de reglas debido a su capacidad de eliminación de ruido.

### 8.2.5. Reverse Engineering the Neural Networks for Rule Extraction in Classification Problems

El método expuesto en (Augasta y Kathirvalavakumar, 2012), propone un nuevo algoritmo de extracción de reglas llamado RxREN que puede extraer reglas de redes neuronales entrenadas para conjuntos de datos con atributos de modo mixto. El algoritmo se basa en técnicas de ingeniería inversa para podar neuronas de entrada insignificantes y descubrir los principios tecnológicos de cada neurona de entrada significativa en la red neuronal. Es capaz de extraer reglas que involucran atributos tanto discretos como continuos, lo que lo hace adecuado para una amplia gama de conjuntos de datos. El algoritmo es más eficiente a la hora de extraer el conjunto más pequeño de reglas con una alta precisión de clasificación en comparación con otros métodos. Las reglas extraídas pueden proporcionar explicaciones para el proceso de toma de decisiones de la red neuronal, lo que tiene implicaciones prácticas en diversos ámbitos, como el diagnóstico médico, la detección de fraude y la calificación crediticia.
El artículo propone un nuevo algoritmo de extracción de reglas llamado RxREN, que se basa en técnicas de ingeniería inversa para extraer reglas de redes neuronales entrenadas. El algoritmo poda neuronas de entrada insignificantes y descubre los principios tecnológicos de cada neurona de entrada significativa en la red neuronal. El algoritmo propuesto se compara con el algoritmo de extracción de reglas HYPINV, que también es un método de aproximación pedagógica. La comparación se realiza en tres conjuntos de datos: cáncer de mama de Wisconsin, ionosfera y diabetes india pima.

Ideas Principales

1. Propuesta de un nuevo algoritmo de extracción de reglas denominado RxREN.
2. Dependencia de técnicas de ingeniería inversa para extraer reglas de redes neuronales.
3. Poda de neuronas de entrada insignificantes.
4. Descubrimiento de principios tecnológicos de neuronas de entrada significativas.
5. Comparación con el algoritmo de extracción de reglas HYPINV.
6. Uso de tres conjuntos de datos para comparación: cáncer de mama de Wisconsin, ionosfera y diabetes india Pima.

## 8.3. El algoritmo COLOSSUS

En el presente trabajo se presenta un método para extraer el patrón de reglas aprendido de una red neuronal de *feedforward* entrenadas y analizar sus propiedades. El método integra la lógica simbólica y no simbólica para explicar el proceso de toma de decisiones del modelo.
En este trabajo se propone utilizar el análisis de correlación entre los vectores de peso y las salidas de cada capa para identificar los caminos críticos que conectan las entradas y los resultados en la red neuronal. Las reglas extraídas se escriben en términos de lógica proposicional y lógica de primer orden (FOL) para explicar la relación entrada-salida.
El enfoque tiene como objetivo mejorar la explicabilidad de las redes neuronales profundas y comprender las condiciones en las que el modelo tiene éxito o fracasa.

Al integrar la lógica simbólica y no simbólica, el método ofrece una forma de explicar el razonamiento que subyace a las decisiones del modelo en términos que los humanos puedan entender.
Las reglas extraídas se pueden interpretar fácilmente mediante la lógica de primer orden (FOL), lo que las hace útiles para una variedad de aplicaciones, particularmente en el campo del diagnóstico.
El algoritmo utilizado en este documento no requiere reglas de aprendizaje previas, por lo que es relativamente sencillo y eficiente trabajar con diferentes tipos de atributos.
La identificación de las neuronas relevantes y las rutas neuronales críticas puede proporcionar información valiosa sobre el funcionamiento interno de la red neuronal, lo que ayuda a interpretar y mejorar los modelos.
En general, el enfoque de este trabajo sobre explicabilidad en las redes neuronales *feedforward* profundas tiene implicaciones prácticas para mejorar la transparencia, la interpretabilidad y la confianza en el proceso de toma de decisiones de estos modelos.

### 8.3.1. Principales Contribuciones

Las principales contribuciones hechos en este trabajo son:

- Proponer un nuevo algoritmo para extraer el patrón de reglas aprendido de una red neuronal *feedforward* entrenada y analizar sus propiedades.
- Uso de lógica proposicional y lógica de primer orden (FOL) para explicar los patrones aprendidos por la red neuronal.
- Presentar un método de caja blanca que analiza las matrices de pesos de las DNN para extraer reglas, con la métrica de similitud coseno.
- Se implementa un cálculo de distancia para resolver solapamiento entre clases.
- El método no requiere ninguna regla de aprendizaje especial durante la extracción de reglas, lo que facilita su uso.
- El método funciona de manera eficiente tanto para atributos continuos como enumerados.
- El algoritmo es muy eficiente en la identificación de neuronas relevantes de la DNN y no requiere reentrenamiento después de la selección de neuronas críticas.
- Las reglas extraídas se pueden usar para explicar el proceso de toma de decisiones de la red neuronal, a través de fórmulas de la lógica de primer orden, lo que puede ser útil en diversas aplicaciones, como el diagnóstico médico, la detección de fraudes y la calificación crediticia.
- Las reglas extraídas se mapean directamente contra el conjunto de datos, y las decisiones de la red puede ser explicadas con datos.
- Indirectamente el algoritmo de extracción de reglas plantea un método de regularización de la DNN, al identificar los caminos críticos neuronales componiendo así redes más pequeñas.

# CAPITULO 9 Conclusiones y Trabajo Futuro

## 9.1. Conclusiones

El aprendizaje profundo es una forma específica de Inteligencia Artificial no simbólica. Este enfoque plantea una inspiración biológica al emplear las capacidades de las redes neuronales profundas para influir en el proceso general del aprendizaje mediante el método de prueba y error. Su eficacia ha sido convincentemente demostrada en una amplia gama de campos (Garnelo et al., 2016). A pesar de su éxito innegable, varias investigaciones han dado evidencia de importantes deficiencias en el aprendizaje profundo contemporáneo tales como:

- Carecen de capacidad de razonar en un nivel abstracto, lo que dificulta la implementación de funciones cognitivas de alto nivel, como el aprendizaje por transferencia, el razonamiento analógico y el razonamiento basado en hipótesis. Se han realizado varios intentos recientes para imitar procesos lógicos en redes neuronales (Evans et al., 2018). Este problema también es tratado en (Manhaeve et al., 2018; Cai et al., 2017; Han et al., 2021).

- Falta de interpretabilidad. Las redes neuronales son típicamente cajas negras. Los cálculos realizados por capas sucesivas rara vez corresponden a pasos de razonamiento humanamente comprensibles, y los vectores intermedios de activaciones que se generan carecen de una semántica humanamente comprensible (Garnelo et al., 2016). Este problema también es tratado en (AmirHosseini & Hosseini, 2019; Mahdavifar & Ghorbani, 2020; Cocarascu et al., 2018; Csiszár et al., 2020; Dombi & Csiszár, 2021; Schmid & Finzel, 2020).

A su vez, los métodos simbólicos presentan la siguiente deficiencia:

- Symbol Grounding. Una limitación importante de la IA simbólica se relaciona con el llamado problema de Symbol Grounding (Harnad, 1990), y se refiere a la medida en que sus elementos simbólicos están hechos a mano en lugar de aprender de los datos. Por el contrario, uno de los puntos fuertes del aprendizaje profundo es su capacidad para descubrir características en datos de gran dimensión con poca o ninguna intervención humana.

En este contexto, se ha propuesta un algoritmo de extracción y explicación de reglas, el cual comenzará con un proceso de cálculo de estadísticas sobre los datos, y posterior extracción de pesos de la red neuronal entrenada, donde se calculará la similitud coseno entre los vectores que forma cada variable de entrada para cada neurona entre sí, y también en relación con el vector de sesgos, y el vector formado por los valores de activación en cada capa. Además, se realiza el mismo procedimiento, para los vectores

de pesos de cada neurona, lo cual resulta en una comparación interneurona de vectores de pesos, con el objetivo de entender la colaboración entre las neuronas de una capa dada.
Las reglas extraídas, se componen y explican en formato de fórmulas bien formadas de la lógica proposicional.
En tal sentido el objetivo es analizar cómo se correlación las variables de entrada, y como estas explican los resultados en la salida de la red. De este modo se demuestra
cómo es posible identificar caminos críticos neuronales basados en los datos, que va desde las variables de entrada hasta la salida de la red.
Las estadísticas obtenidas sobre los datos de entrada se utilizan para el método de extracción de reglas, sobre aquellas que mejor que mejor explican los resultados arrojados por las redes (máxima precisión). De este modo, los rangos de valores en entre ambas métricas sientan las bases que asisten en la validación y verificación de las reglas extraídas.

Como objetivo principal del trabajo se planteó *Desarrollar un algoritmo simbólico-neuronal híbrido que permita extraer reglas comprensibles en redes neuronales feedforward profundas entrenadas, a través del análisis de las matrices de pesos, y la validación y explicación de reglas por medio de la aplicación de técnicas de lógica de primer orden.*

Como consecuencia de lo planteado, las principales contribuciones hechas en este trabajo son:

- Proponer un nuevo algoritmo para extraer el patrón de reglas aprendido de una red neuronal *feedforward* entrenada y analizar sus propiedades.
- Uso de lógica proposicional y lógica de primer orden (FOL) para explicar los patrones aprendidos por la red neuronal.
- Presentar un método de caja blanca que analiza las matrices de pesos de las DNN para extraer reglas, con la métrica de similitud coseno.
- Se implementa un cálculo de distancia para resolver solapamiento entre clases.
- El método no requiere ninguna regla de aprendizaje especial durante la extracción de reglas, lo que facilita su uso.
- El método funciona de manera eficiente tanto para atributos continuos como enumerados.
- El algoritmo es muy eficiente en la identificación de neuronas relevantes de la DNN y no requiere reentrenamiento después de la selección de neuronas críticas.
- Las reglas extraídas se pueden usar para explicar el proceso de toma de decisiones de la red neuronal, a través de fórmulas de la lógica de primer orden, lo que puede ser útil en diversas aplicaciones, como el diagnóstico médico, la detección de fraudes y la calificación crediticia.
- Las reglas extraídas se mapean directamente contra el conjunto de datos, y las decisiones de la red puede ser explicadas con datos.
- Indirectamente el algoritmo de extracción de reglas plantea un método de regularización de la DNN, al identificar los caminos críticos neuronales componiendo así redes más pequeñas.

- Construir un prototipo funcional que implemente las funciones definidas en el modelo formal.

## 9.2. Trabajo Futuro

Las líneas de trabajo a futuro se resumen en los siguientes puntos expuestos a continuación:

- Investigar la aplicación del método propuesto en redes neuronales más complejas y de mayor escala para evaluar su escalabilidad y rendimiento.

- Explorar la integración de marcos lógicos adicionales, como la lógica difusa o la lógica probabilística, para mejorar la interpretabilidad y explicabilidad de los patrones de reglas extraídos.

- Realizar estudios empíricos para evaluar la eficacia del método en diferentes dominios y tareas, como la clasificación de imágenes o el procesamiento del lenguaje natural, para evaluar su generalización y solidez.

- Examinar el impacto de diferentes arquitecturas de red y técnicas de capacitación en la interpretabilidad de los patrones de reglas extraídos, para obtener información sobre la relación entre la estructura de la red y la explicabilidad.

- Investigar el potencial del método propuesto en combinación con otras técnicas de explicabilidad, como mecanismos de atención o mapas de prominencia, para proporcionar una comprensión más completa y detallada del proceso de toma de decisiones en redes neuronales.

# Anexo 1 - Acrónimos o siglas

IA = Inteligencia Artificial
MI = Motor de Inferencia
ML = Machine Learning
LPO = Lógica de primer orden
PI = Pregunta de Investigación
PC = Palabra Clave
TF = TensorFlow (Librería para desarrollo de redes neuronales)
BD = Base de Datos
PNL = Procesamiento Natural del Lenguaje
LPPO = Lógica de predicados de primer orden
FOC2 = Lógica de predicados de primer orden con cuantificadores de conteo
TDMC = Toma de decisión de múltiple criterio
ILP = Inductive Logic Programming (Programación lógica inductiva)
FIS = Fuzzy Inference System (Sistema de inferencia difuso)
IT2FIS = Sistema de inferencia difuso de intervalo de tipo 2
LTL = Lógica Temporal Lineal
DL = Deep Learning
ANN = Artificial Neural Net
NN = Neural Network
DNN = Deep Neural Network
GNN = Graph Neural Network
RN = Relational Network
RBM = Restricted Boltzmann Machines
FIS = Fuzzy Inference System
AC-GNN = Aggregate-Combine, Graph Neural Network (Agregar-Combinar, Red Neuronal de Grafos)
ACR-GNN = Aggregate-Combine-Readout, Graph Neural Network (Agregar-Combinar-Lectura, Red Neuronal de Grafos)
DRL = Deep Reinforcement Learning (Aprendizaje por refuerzo profundo)
RN = Relational Network (Red Relacional)
SBR = Semantic Based Regularization (Regularización basada en semántica)
LEN = Logic Explained Networks
RBM = Restricted Boltzmann Machines
FBF = Formulas Bien Formadas
SC = Similitud Coseno

# Referencias

Aghaeipoor, Fatemeh, Mohammad Sabokrou, y Alberto Fernández. 2023. «Fuzzy Rule-Based Explainer Systems for Deep Neural Networks: From Local Explainability to Global Understanding». *IEEE Transactions on Fuzzy Systems* 1-12. doi: 10.1109/TFUZZ.2023.3243935.

Alpaydin, Ethem. 2016. *Machine Learning: The New AI*. Cambridge, MA, USA: MIT Press.

Alpaydin, Ethem. 2020. *Introduction to Machine Learning*. MIT Press.

Alzaeemi, Shehab Abdulhabib, Mohd Asyraf Mansor, Mohd Shareduwan Mohd Kasihmuddin, y Saratha Sathasivam. 2019. «2 satisfiability logic programming in radial basis function neural networks». P. 060045 en *AIP Conference Proceedings*. Vol. 2184. AIP Publishing LLC.

Amarasinghe, Kasun, Kevin Kenney, y Milos Manic. 2018. «Toward Explainable Deep Neural Network Based Anomaly Detection». Pp. 311-17 en *2018 11th International Conference on Human System Interaction (HSI)*. Gdansk, Poland: IEEE.

AmirHosseini, Banafsheh, y Rahil Hosseini. 2019. «An Improved Fuzzy-Differential Evolution Approach Applied to Classification of Tumors in Liver CT Scan Images». *Medical & Biological Engineering & Computing* 57(10):2277-87. doi: 10.1007/s11517-019-02009-7.

Angelov, Plamen, y Eduardo Soares. 2019. «Towards Explainable Deep Neural Networks (xDNN)».

Augasta, M. Gethsiyal, y T. Kathirvalavakumar. 2012. «Reverse Engineering the Neural Networks for Rule Extraction in Classification Problems». *Neural Processing Letters* 35(2):131-50. doi: 10.1007/s11063-011-9207-8.

Azizan, Farah Liyana, y Saratha Sathasivam. 2023. «Randomised Alpha-Cut Fuzzy Logic Hybrid Model in Solving 3-Satisfiability Hopfield Neural Network». *Malaysian Journal of Fundamental and Applied Sciences* 19(1):43-55. doi: 10.11113/mjfas.v19n1.2697.

Barbiero, Pietro, Gabriele Ciravegna, Francesco Giannini, Pietro Lió, Marco Gori, y Stefano Melacci. 2022. «Entropy-Based Logic Explanations of Neural Networks». *Proceedings of the AAAI Conference on Artificial Intelligence* 36(6):6046-54. doi: 10.1609/aaai.v36i6.20551.

Barceló, Pablo, Egor V. Kostylev, Mikael Monet, Jorge Pérez, Juan Reutter, y Juan Pablo Silva. 2020. «The Logical Expressiveness of Graph Neural Networks». P. hal-03356968f en *THE LOGICAL EXPRESSIVENESS OF GRAPH NEURAL NETWORKS*. Ethiopia.

Barredo Arrieta, Alejandro, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, y Francisco Herrera. 2020. «Explainable Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI». *Information Fusion* 58:82-115. doi: 10.1016/j.inffus.2019.12.012.

Battaglia, Peter, Razvan Pascanu, Matthew Lai, Danilo Rezende, y Koray Kavukcuoglu. 2016. «Interaction networks for learning about objects, relations and physics». en *Advances in Neural Information Processing Systems*.

Battaglia, Peter W., Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, y Razvan Pascanu. 2018. «Relational inductive biases, deep learning, and graph networks». *arXiv:1806.01261 [cs, stat]*.

Bishop, Christopher. 2006. *Pattern Recognition and Machine Learning*. New York: Springer-Verlag.

Borges, Rafael V., Luis C. Lamb, y Artur S. D'avila Garcez. 2006. «Combining Architectures for Temporal Learning in Neural-Symbolic Systems». Pp. 46-46 en *2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06)*.

Britos, Paola Verónica. 2005. *Minería de datos basada en sistemas inteligentes*. 1a ed. Buenos Aires: Nueva Libreria.

Burkhardt, Sophie, Jannis Brugger, Nicolas Wagner, Zahra Ahmadi, Kristian Kersting, y Stefan Kramer. 2021. «Rule Extraction From Binary Neural Networks With Convolutional Rules for Model Validation». *Frontiers in Artificial Intelligence* 4:642263. doi: 10.3389/frai.2021.642263.

Cai, Cheng-Hao, Dengfeng Ke, Yanyan Xu, y Kaile Su. 2017. «Symbolic manipulation based on deep neural networks and its application to axiom discovery». Pp. 2136-43 en *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE.

Calegari, Roberta, Giovanni Ciatto, Viviana Mascardi, y Andrea Omicini. 2020. «Logic-Based Technologies for Multi-Agent Systems: A Systematic Literature Review». *Autonomous Agents and Multi-Agent Systems* 35(1):1. doi: 10.1007/s10458-020-09478-3.

Caruana, Rich, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, y Noemie Elhadad. 2015. «Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission». Pp. 1721-30 en *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*.

Ciravegna, Gabriele, Pietro Barbiero, Francesco Giannini, Marco Gori, Pietro Lió, Marco Maggini, y Stefano Melacci. 2023. «Logic Explained Networks». *Artificial Intelligence* 314:103822. doi: 10.1016/j.artint.2022.103822.

Cocarascu, Oana, Kristijonas Cyras, y Francesca Toni. 2018. «Explanatory predictions with artificial neural networks and argumentation».

Csiszár, Orsolya, Gábor Csiszár, y József Dombi. 2020. «How to Implement MCDM Tools and Continuous Logic into Neural Computation?: Towards Better Interpretability of Neural Networks». *Knowledge-Based Systems* 210:106530. doi: 10.1016/j.knosys.2020.106530.

Dai, Wang-Zhou, y Stephen H. Muggleton. 2021. «Abductive Knowledge Induction From Raw Data».

De, Tanusree, Prasenjit Giri, Ahmeduvesh Mevawala, Ramyasri Nemani, y Arati Deo. 2020. «Explainable AI: A Hybrid Approach to Generate Human-Interpretable Explanation for Deep Learning Prediction». *Procedia Computer Science* 168:40-48. doi: 10.1016/j.procs.2020.02.255.

Díaz, Gustavo Adolfo, Jesús Alfonso López Sotelo, y Eduardo Caicedo Bravo. 2009. «Aplicación de la lógica difusa tipo dos en una planta didáctica en control de procesos industriales, respecto de las variables nivel y flujo». 7(13):21.

Diligenti, Michelangelo, Soumali Roychowdhury, y Marco Gori. 2017. «Integrating prior knowledge into deep learning». Pp. 920-23 en *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE.

Dombi, József, y Orsolya Csiszár. 2021. «Interpretable Neural Networks Based on Continuous-Valued Logic and Multi-Criteria Decision Operators». Pp. 147-69 en *Explainable Neural Networks Based on Fuzzy Logic and Multi-criteria Decision Tools*, *Studies in Fuzziness and Soft Computing*, editado por J. Dombi y O. Csiszár. Cham: Springer International Publishing.

Domingos, P. 2018. *How the Quest for the Ultimate Learning Machine will remake our World*.

Doumas, Leonidas A. A., Guillermo Puebla, y Andrea E. Martin. 2019. «Human-like generalization in a machine through predicate learning». *arXiv:1806.01709 [cs]*.

Evans, Richard, David Saxton, David Amos, Pushmeet Kohli, y Edward Grefenstette. 2018. *Can neural networks understand logical entailment?*

Flach, Peter. 2012. *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. Cambridge University Press.

Garcez, Artur d'Avila, Marco Gori, Luis C. Lamb, Luciano Serafini, Michael Spranger, y Son N. Tran. 2019. «Neural-Symbolic Computing: An Effective Methodology for Principled Integration of Machine Learning and Reasoning».

Garnelo, Marta, Kai Arulkumaran, y Murray Shanahan. 2016. «Towards Deep Symbolic Reinforcement Learning». *arXiv:1609.05518 [cs]*.

Garnelo, Marta, y Murray Shanahan. 2019. *Reconciling deep learning with symbolic artificial intelligence: representing objects and relations*.

Giunchiglia, Eleonora, Mihaela Catalina Stoian, y Thomas Lukasiewicz. 2022. «Deep Learning with Logical Constraints».

Goodfellow, Ian, Yoshua Bengio, y Aaron Courville. 2016. *Deep Learning*. Cambridge, Massachusetts: The MIT Press.

Goodman, Bryce, y Seth Flaxman. 2017. «European Union regulations on algorithmic decision-making and a "right to explanation"». *AI magazine* 38(3):50-57.

Guidotti, Riccardo, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, y Dino Pedreschi. 2019. «A Survey of Methods for Explaining Black Box Models». *ACM Computing Surveys* 51(5):1-42. doi: 10.1145/3236009.

Han, Zhongyi, Benzheng Wei, Xiaoming Xi, Bo Chen, Yilong Yin, y Shuo Li. 2021. «Unifying neural learning and symbolic reasoning for spinal medical report generation». *Medical Image Analysis* 67:101872.

Harnad, Stevan. 1990. «The symbol grounding problem». *Physica D: Nonlinear Phenomena*. doi: 10.1016/0167-2789(90)90087-6.

Hinton, Geoffrey E., y Terrence J. Sejnowski. 1986. «Learning and relearning in Boltzmann machines». *Parallel distributed processing: Explorations in the microstructure of cognition* 1(282-317):2.

Karpathy, Andrej, y Li Fei-Fei. 2015. «Deep Visual-Semantic Alignments for Generating Image Descriptions». Pp. 3128-37 en.

Kitchenham, Barbara, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, y Stephen Linkman. 2009. «Systematic Literature Reviews in Software Engineering – A Systematic Literature Review». *Information and Software Technology* 51(1):7-15. doi: 10.1016/j.infsof.2008.09.009.

Krishnan, R., G. Sivakumar, y P. Bhattacharya. 1999. «Extracting Decision Trees from Trained Neural Networks». 12.

Lake, Brenden M., Tomer D. Ullman, Joshua B. Tenenbaum, y Samuel J. Gershman. 2017. «Building machines that learn and think like people». *Behavioral and Brain Sciences*. doi: 10.1017/S0140525X16001837.

Levine, Sergey, Chelsea Finn, Trevor Darrell, y Pieter Abbeel. 2016. *End-to-end training of deep visuomotor policies*.

Li, Qing, Siyuan Huang, Yining Hong, Yixin Zhu, Ying Nian Wu, y Song-Chun Zhu. 2021. «A HINT from Arithmetic: On Systematic Generalization of Perception, Syntax, and Semantics». *arXiv preprint arXiv:2103.01403*.

Li, Tao, y Vivek Srikumar. 2019. «Augmenting neural networks with first-order logic». *arXiv preprint arXiv:1906.06298*.

Li, Yujia, Richard Zemel, Marc Brockschmidt, y Daniel Tarlow. 2016. «Gated graph sequence neural networks». en *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*.

Mahdavifar, Samaneh, y Ali A. Ghorbani. 2020. «DeNNeS: Deep Embedded Neural Network Expert System for Detecting Cyber Attacks». *Neural Computing and Applications* 32(18):14753-80. doi: 10.1007/s00521-020-04830-w.

Manhaeve, Robin, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, y Luc De Raedt. 2018. «Deepproblog: Neural probabilistic logic programming». *Advances in Neural Information Processing Systems* 31:3749-59.

Marcus, Gary. 2018. «Deep Learning: A Critical Appraisal». *arXiv:1801.00631 [cs, stat]*.

Marcus, Gary F. 2001. *The algebraic mind: Integrating connectionism and cognitive science*.

Markowska-Kaczmar, Urszula, y Paweł Wnuk-Lipiński. 2004. «Rule Extraction from Neural Network by Genetic Algorithm with Pareto Optimization». Pp. 450-55 en *Artificial Intelligence and Soft Computing - ICAISC 2004*, *Lecture Notes in Computer Science*, editado por L. Rutkowski, J. H. Siekmann, R. Tadeusiewicz, y L. A. Zadeh. Berlin, Heidelberg: Springer.

Marra, Giuseppe, Francesco Giannini, Michelangelo Diligenti, y Marco Gori. 2019. «Lyrics: A general interface layer to integrate logic inference and deep learning». Pp. 283-98 en *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*. Springer.

McCarthy, John. 1987. «Generality in artificial intelligence». *Communications of the ACM*. doi: 10.1145/33447.33448.

Melin, Patricia, Ivette Miramontes, y German Prado-Arechiga. 2018. «A Hybrid Model Based on Modular Neural Networks and Fuzzy Systems for Classification of Blood Pressure and Hypertension Risk Diagnosis». *Expert Systems with Applications* 107:146-64. doi: 10.1016/j.eswa.2018.04.023.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, y Demis Hassabis. 2015. «Human-Level Control through Deep Reinforcement Learning». *Nature* 518(7540):529-33. doi: 10.1038/nature14236.

Moher, David, Alessandro Liberati, Jennifer Tetzlaff, y Douglas G. Altman. 2010. «Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement». *Int J Surg* 8(5):336-41.

Montáns, Francisco J., Francisco Chinesta, Rafael Gómez-Bombarelli, y J. Nathan Kutz. 2019. «Data-Driven Modeling and Learning in Science and Engineering». *Comptes Rendus Mécanique* 347(11):845-55. doi: 10.1016/j.crme.2019.11.009.

Montavon, Grégoire, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, y Klaus-Robert Müller. 2017. «Explaining Nonlinear Classification Decisions with Deep Taylor Decomposition». *Pattern Recognition* 65:211-22. doi: 10.1016/j.patcog.2016.11.008.

Negro, Pablo Ariel, y Claudia Pons. 2023. «Extracción de reglas en redes neuronales feedforward entrenadas con lógica de primer orden». *Memorias de las JAIIO* 9(2):7-24.

Negro, Pablo, y Claudia Pons. 2022. «Artificial Intelligence techniques based on the integration of symbolic logic and deep neural networks: A systematic review of the literature». *Inteligencia Artificial* 25(69):13-41. doi: 10.4114/intartif.vol25iss69pp13-41.

Negro, Pablo, y Claudia Pons. 2024. «Rule Extraction in Trained Feedforward Deep Neural Networks - Integrating Cosine Similarity and Logic for Explainability.Pdf». 18. IGI-Global - International Journal of Artificial Intelligence and Machine Learning (IJAIML) (in Press)

Nielsen, Ian E., Dimah Dera, Ghulam Rasool, Nidhal Bouaynaya, y Ravi P. Ramachandran. 2022. «Robust Explainability: A Tutorial on Gradient-Based Attribution Methods for Deep Neural Networks». *IEEE Signal Processing Magazine* 39(4):73-84. doi: 10.1109/MSP.2022.3142719.

Pal, Shanoli Samui, y Samarjit Kar. 2019. «A Hybridized Forecasting Method Based on Weight Adjustment of Neural Network Using Generalized Type-2 Fuzzy Set». *International Journal of Fuzzy Systems* 21(1):308-20. doi: 10.1007/s40815-018-0534-z.

de Penning, Leo, Artur S. d'Avila Garcez, Luis C. Lamb, Arjan Stuiver, y John-Jules Ch. Meyer. 2014. «Applying Neural-Symbolic Cognitive Agents in Intelligent Transport Systems to reduce CO2 emissions». Pp. 55-62 en *2014 International Joint Conference on Neural Networks (IJCNN)*.

Petticrew, Mark, y Helen Roberts. 2008. *Systematic Reviews in the Social Sciences: A Practical Guide*. John Wiley & Sons.

Pons, Claudia, Ricardo Rosenfeld, y Clara Patricia Smith. 2017. *Lógica para Informática*. Editorial de la Universidad Nacional de La Plata (EDULP).

Ramirez, Eduardo, Patricia Melin, y German Prado-Arechiga. 2019. «Hybrid Model Based on Neural Networks, Type-1 and Type-2 Fuzzy Systems for 2-Lead Cardiac Arrhythmia Classification». *Expert Systems with Applications* 126:295-307. doi: 10.1016/j.eswa.2019.02.035.

Riegel, Ryan, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, y Udit Sharma. 2020. «Logical neural networks». *arXiv preprint arXiv:2006.13155*.

Rocktäschel, Tim, y Sebastian Riedel. 2017. «End-to-End Differentiable Proving». *arXiv:1705.11040 [cs]*.

Russell, Stuart, y Peter Norvig. 2010. *Artificial Intelligence A Modern Approach Third Edition*.

Samek, Wojciech, Gregoire Montavon, Sebastian Lapuschkin, Christopher J. Anders, y Klaus-Robert Muller. 2021. «Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications». *Proceedings of the IEEE* 109(3):247-78. doi: 10.1109/JPROC.2021.3060483.

Samek, Wojciech, Thomas Wiegand, y Klaus-Robert Müller. 2017. «Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models».

Santoro, Adam, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, y Timothy Lillicrap. 2017. «A simple neural network module for relational reasoning». en *Advances in Neural Information Processing Systems*.

Santos, R. T., J. C. Nievola, y A. A. Freitas. 2000. «Extracting Comprehensible Rules from Neural Networks via Genetic Algorithms». Pp. 130-39 en *2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks. Proceedings of the First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks (Cat. No.00EX448)*. San Antonio, TX, USA: IEEE.

Scarselli, Franco, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, y Gabriele Monfardini. 2009. «The graph neural network model». *IEEE Transactions on Neural Networks*. doi: 10.1109/TNN.2008.2005605.

Schmid, Ute, y Bettina Finzel. 2020. «Mutual Explanations for Cooperative Decision Making in Medicine». *KI - Künstliche Intelligenz* 34(2):227-33. doi: 10.1007/s13218-020-00633-2.

Schmidhuber, Jürgen. 2015. *Deep Learning in neural networks: An overview*.

Shanahan, Murray, Kyriacos Nikiforou, Antonia Creswell, Christos Kaplanis, David Barrett, y Marta Garnelo. 2020. «An Explicitly Relational Neural Network Architecture».

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, y Demis Hassabis. 2016. «Mastering the game of Go with deep neural networks and tree search». *Nature*. doi: 10.1038/nature16961.

Sinha, Koustuv, Shagun Sodhani, Joelle Pineau, y William L. Hamilton. 2020. «Evaluating logical generalization in graph neural networks». *arXiv preprint arXiv:2003.06560*.

Tran, Son N. 2017. «Unsupervised Neural-Symbolic Integration».

Tsividis, Pedro A., Thomas Pouncy, Jacqueline L. Xu, Joshua B. Tenenbaum, y Samuel J. Gershman. 2017. «Human learning in atari». en *AAAI Spring Symposium - Technical Report*.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, \Lukasz Kaiser, y Illia Polosukhin. 2017. «Attention is all you need». en *Advances in Neural Information Processing Systems*.

Wang, Wenya, y Sinno Jialin Pan. 2020. «Integrating deep learning with logic fusion for information extraction». Pp. 9225-32 en *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34.

Xie, Yaqi, Fan Zhou, y Harold Soh. 2021. «Embedding Symbolic Temporal Knowledge into Deep Sequential Models». *arXiv preprint arXiv:2101.11981*.

Xu, Kelvin, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, y Yoshua Bengio. 2015. «Show, attend and tell: Neural image caption generation with visual attention». en *32nd International Conference on Machine Learning, ICML 2015*.

Yann LeCun, Yoshua Bengio, Geoffrey Hinton. 2015. «Deep learning (2015), Y. LeCun, Y. Bengio and G. Hinton». *Nature*.

Zarlenga, Mateo Espinosa, Zohreh Shams, y Mateja Jamnik. 2021. «Efficient Decompositional Rule Extraction for Deep Neural Networks».