UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Inteligencia Artificial Explicable: Técnicas de extracción de reglas en redes neuronales artificiales

AUTORES: Milagros Aylén Jacinto, Martín Moschettoni

DIRECTOR/A: Dra. Claudia Pons

CODIRECTOR/A: Lic. Gabriela Pérez

ASESOR/A PROFESIONAL:

Resumen

Las redes neuronales artificiales se destacan en tareas complejas, pero la imposibilidad de comprender y validar el proceso de decisión de un sistema de IA es un claro inconveniente. En el caso de las tareas de clasificación, la extracción de reglas se presenta como una herramienta para mejorar la comprensión de la red y sus resultados.
En esta tesina se estudiaron diferentes técnicas de extracción de reglas para explicar las redes neuronales artificiales. Se diseñó y desarrolló un algoritmo de extracción de reglas llamado FORxREN el cual genera reglas fieles y fácilmente interpretables que pueden esclarecer en parte el razonamiento de las redes neuronales.

Palabras Clave

- *Fidelidad*
- *Redes Neuronales Artificiales*
- *Inteligencia Artificial Explicable*
- *XAI*

Conclusiones

Los resultados de las ejecuciones han demostrado que FORxREN genera reglas con un alto porcentaje de fidelidad. Además, se ha observado que, fue posible aumentar la generalidad de las reglas (y, por lo tanto, su comprensibilidad) sin perder la fidelidad.
Ha quedado demostrado que dicha orientación a la fidelidad por parte de FORxREN puede ser una estrategia efectiva para mejorar la explicabilidad de la red y generar reglas más fiables en determinadas aplicaciones. Además de que la explicabilidad en redes neuronales artificiales es de suma importancia en la actualidad.

Trabajos Realizados

Se analizaron distintas técnicas de extracción de reglas, incluyendo ECLAIRE, FERNN y DeepRED, cada una con un enfoque particular. Posteriormente, re-implementando el método RxREN y ajustando su enfoque original para dar prioridad a la fidelidad de las reglas sobre la precisión, se desarrolló el algoritmo FORxREN. Este algoritmo fue evaluado utilizando los conjuntos de datos Iris, WBC y Wine, mediante la construcción de múltiples redes neuronales artificiales . Además se evaluaron las reglas obtenidas utilizando las métricas de comprensibilidad y fidelidad.

Trabajos Futuros

- *Investigar la aplicación de FORxREN en redes neuronales mas complejas y de mayor escala para evaluar su escalabilidad y rendimiento.*
- *Realizar estudios empíricos para evaluar la eficiencia de FORxREN en diferentes dominios y tareas, como datos no numéricos.*
- *Analizar el impacto de diferentes arquitecturas de red en la explicabilidad de las reglas.*
- *Investigar el potencial del método propuesto en combinación con otras técnicas de explicabilidad.*
- *Evaluar la comprensibilidad de manera formal con un enfoque mas social y centrado en el humano*

Fecha de la presentación: Febrero, 2024

Tesina de grado

Inteligencia Artificial Explicable: Técnicas de extracción de reglas en redes neuronales artificiales

Autores
Jacinto, Milagros Aylén
Moschettoni, Martín

Director
Dra. Claudia Pons

Codirector
Lic. Gabriela Perez

Titulación
Licenciatura en Informática

Lugar y fecha de presentación
La Plata - Febrero de 2024

Facultad de Informática
Universidad Nacional de La Plata (UNLP)

Resumen

Las redes neuronales artificiales se destacan en tareas complejas, pero su falta de explicabilidad es un claro inconveniente y lleva a denominarlas "cajas negras". En muchas aplicaciones, la imposibilidad de comprender y validar el proceso de decisión de un sistema de IA es un claro inconveniente. Por ejemplo, en el diagnóstico médico sería irresponsable y poco prudente depender exclusivamente de las predicciones de un sistema de caja negra. En su lugar, cada decisión debería estar sujeta a una validación adecuada por parte de un profesional experto. Del mismo modo, en numerosos otros contextos, resulta fundamental contar con la capacidad de validar y comprender el funcionamiento de la red neuronal.

El uso de modelos de IA explicables e interpretables por humanos es un requisito para ofrecer tal garantía. Como era de esperar, el desarrollo de técnicas para "abrir" modelos de caja negra ha recibido recientemente mucha atención en la comunidad. Esto incluye el desarrollo de métodos que ayudan a comprender mejor lo que el modelo ha aprendido (es decir, su representación), así como técnicas para explicar predicciones individuales.

En el caso de las tareas de clasificación, la extracción de reglas se presenta como una herramienta para mejorar la comprensión de la red y sus resultados. En esta tesina se estudiaron diferentes técnicas de extracción de reglas y sus características en búsqueda de entender como se pueden explicar las redes neuronales artificiales. Se seleccionó una de estas técnicas (RxREN) y se analizó en profundidad su funcionamiento. A partir de RxREN que está orientado a la minería de datos y explicación de los datos originales con los que se entrenó la red neuronal artificial, se diseñó y desarrolló un algoritmo de extracción de reglas llamado FORxREN. FORxREN tiene como objetivo mejorar la explicabilidad de las redes neuronales artificiales debido a que está orientado hacia la extracción de reglas fieles y fácilmente interpretables que puedan esclarecer en parte el razonamiento subyacente de las mismas.

FORxREN fue desarrollado en Python, usando bibliotecas populares como Keras, Scikit-Learn y Tensorflow. Se llevaron a cabo pruebas utilizando conjuntos de datos conocidos como Iris, WBC y Wine. En este proceso, se entrenaron redes neuronales artificiales, sobre las cuales se aplicó FORxREN para extraer reglas que priorizan la fidelidad y la comprensibilidad. El resultado de esta aplicación consistió en la generación de reglas fieles a la red y fácilmente comprensibles.

Palabras claves

Fidelidad, Redes Neuronales Artificiales, Inteligencia Artificial Explicable, XAI

Agradecimientos

En el final de esta etapa, queremos expresar nuestro profundo agradecimiento a aquellos que han sido pilares fundamentales en nuestro recorrido no solo de la carrera, sino también de la vida. Este logro no hubiera sido posible sin el constante apoyo de nuestras familias y amigos, además de la guía y acompañamiento de nuestras directoras, el apoyo y confianza brindado por el LIFIA, y el entorno perfecto para aprender y crecer bridado por la facultad.

En primer lugar, a nuestras familias, queremos agradecerles por su apoyo absoluto, sus constantes palabras de aliento y comprensión, que fueron un impulso enorme para no bajar los brazos y siempre estar listos para darlo todo.

A nuestros amigos Felipe, Julián y Leo (están ordenados alfabéticamente, no se enojen) queremos agradecerles, ya que formaron junto a nosotros el mejor grupo de estudio del universo conocido y también del desconocido. Junto a ellos compartimos risas, escribimos canciones, hicimos memes, nos enfrentamos a desafíos enormes a lo largo de la carrera y estudiamos hasta cualquier hora para poder seguir trabajando en equipo. Sin ellos esto hubiera sido imposible.

A nuestras directoras Claudia y Gabriela, que nos mostraron el camino a recorrer aun cuando no sabíamos lo que buscábamos. Agradecemos sus incontables consejos, las oportunidades que nos dieron, y su acompañamiento constante que nos dio confianza para animarnos a nuevas experiencias que nos permitieron desarrollarnos profesionalmente.

Al LIFIA le debemos un enorme agradecimiento, es donde tuvimos la oportunidad de ampliar nuestros conocimientos. Además, la experiencia que adquirimos nos ha demostrado de lo que somos capaces, y nos hace sentir preparados y listos para lo que venga. Agradecemos la colaboración, el intercambio de ideas y el ambiente familiar que caracterizó nuestro tiempo en el laboratorio.

A la facultad y universidad que nos abrieron las puertas del conocimiento, les estamos agradecidos por proporcionarnos una educación de calidad y por ayudarnos a desarrollar nuestro pensamiento crítico, además de demostrarnos durante la carrera, que siempre hay más para aprender, y que no hay límites en lo que podemos hacer.

Por último, y no menos importante, queremos agradecernos mutuamente por el amor incondicional y apoyo constante a lo largo de toda la carrera, fue un combustible vital para seguir adelante. Aunque al principio nos costó entendernos y trabajar en equipo, con el tiempo nuestra compatibilidad tanto en la vida personal como académica y profesional se fue fortaleciendo día a día. Esto nos demostró que este equipo es para toda la vida.

En resumen, este largo camino y su conclusión, es el resultado de un esfuerzo enorme de muchas de las personas que forman parte de nuestras vidas y su ayuda fue fundamental en cada paso que dimos. Este logro no solo es nuestro, sino de todos aquellos que creyeron en nosotros y nos acompañaron a lo largo de este emocionante camino. ¡Gracias!

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La inteligencia artificial (Russell y Norvig, 2021), y en particular las técnicas de aprendizaje profundo (Goodfellow, Bengio, y cols., 2016) usando redes neuronales artificiales, han progresado en las últimas décadas. Los motores de este desarrollo han sido, la disponibilidad de grandes bases de datos como ImageNet o Sports1M, las ganancias de velocidad obtenidas con potentes tarjetas GPU y la gran flexibilidad de frameworks como Caffe, TensorFlow o PyTorch fueron factores cruciales para el éxito. En la actualidad, los sistemas de IA basados en el aprendizaje automático destacan en una serie de tareas complejas, que van desde la detección de objetos en imágenes y la comprensión de lenguajes naturales hasta el procesamiento de señales del habla.

Estos inmensos éxitos de los sistemas de IA, especialmente de los modelos de aprendizaje profundo, muestran el carácter revolucionario de esta tecnología, que tendrá un gran impacto más allá del mundo académico y también dará lugar a cambios disruptivos en industrias y sociedades. Sin embargo, aunque estos modelos alcanzan precisiones de predicción impresionantes, su estructura no lineal anidada los hace muy poco transparentes, es decir, no está claro qué información de los datos de entrada les hace llegar realmente a sus decisiones. Por ello, estos modelos suelen considerarse cajas negras.

En muchas aplicaciones, la imposibilidad de comprender y validar el proceso de decisión de un sistema de IA es un claro inconveniente. Por ejemplo, en el diagnóstico médico, sería irresponsable confiar por defecto en las predicciones de un sistema de caja negra. También en los coches autónomos, donde una sola predicción incorrecta puede ser muy costosa, debe garantizarse la confianza del modelo en las características correctas. En su lugar, cada decisión trascendental debería ser accesible para una validación adecuada por parte de un humano experto. El uso de modelos de IA explicables e interpretables por humanos es un requisito previo para ofrecer tal garantía.

Como era de esperar, el desarrollo de técnicas para abrir modelos de caja negra ha recibido recientemente mucha atención en la comunidad. Esto incluye el desarrollo de métodos que ayudan a comprender mejor lo que el modelo ha aprendido (es decir, su representación), así como técnicas para explicar predicciones individuales.

Por lo tanto, la investigación realizada en esta tesis pone foco en una nueva técnica para generar explicaciones de las clasificaciones de una red neuronal artificial, el proceso a través del cual fue construida y como la misma aporta al campo de la explicabilidad haciendo hincapié en la importancia de generar explicaciones fieles que ayuden a los usuarios a comprender mejor los resultados esperados de una red neuronal artificial.

1.1. Motivación e importancia del campo

La capacidad de explicar a los demás los fundamentos de las propias decisiones es un aspecto importante de la inteligencia humana. Además, la explicación de las propias decisiones es a menudo un requisito 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 tener menos importancia para los sistemas técnicos de IA, hay muchos argumentos a favor de la explicabilidad en la inteligencia artificial. He aquí los más importantes:

- Verificación del sistema: Como ya se ha mencionado, en muchas aplicaciones no se puede confiar por defecto en un sistema de caja negra. Por ejemplo, en sanidad, el uso de modelos que puedan ser interpretados y verificados por expertos médicos es una necesidad absoluta. Los autores de (Caruana y cols., 2015) muestran un ejemplo de este ámbito, en el que un sistema de IA entrenado para predecir el riesgo de neumonía de una persona llega a conclusiones totalmente erróneas. La aplicación de este modelo a modo de caja negra no reduciría, sino que aumentaría, el número de muertes relacionadas con la neumonía.

- Mejora del sistema: El primer paso para mejorar un sistema de IA es conocer sus puntos débiles. Obviamente, es más difícil realizar este análisis de debilidades en modelos de caja negra que en modelos interpretables. También es más fácil detectar sesgos en el modelo o en el conjunto de datos (como en el ejemplo de la neumonía) si se entiende lo que hace el modelo y por qué llega a sus predicciones. Además, la interpretabilidad del modelo puede ser útil a la hora de comparar diferentes modelos o arquitecturas. Por ejemplo, los autores de (Arras, Horn, Montavon, Müller, y Samek, 2016), (Arras, Montavon, Müller, y Samek, 2017) y (Lapuschkin, Binder, Montavon, Muller, y Samek, 2016) observaron que los modelos pueden tener el mismo rendimiento de clasificación, pero diferir en gran medida en cuanto a las características que utilizan como base para sus decisiones. Estos trabajos demuestran que la identificación del modelo más "apropiado" requiere explicabilidad. Incluso se puede afirmar que cuanto mejor entendamos lo que hacen nuestros modelos (y por qué a veces fallan), más fácil será mejorarlos.

- Aprender del sistema: Como los sistemas de IA actuales se entrenan con millones de ejemplos, pueden observar patrones en los datos que no son accesibles a los humanos, que solo son capaces de aprender con un número limitado de ejemplos. Al utilizar sistemas de IA explicables, podemos intentar extraer este conocimiento destilado del sistema de IA para adquirir nuevos conocimientos. Un ejemplo de este tipo de transferencia de conocimientos del sistema de IA al ser humano lo menciona Fan Hui, un experto en Go en 2016 “No es un movimiento humano. Nunca vi a un humano jugar este movimiento” refiriéndose al trigésimo séptimo movimiento del juego histórico de Go entre Lee Sedol, un gran jugador de Go, y AlphaGo, una inteligencia artificial construida por DeepMind. El sistema de IA identifica nuevas estrategias para jugar al Go, que sin duda ahora también han sido adaptadas por jugadores humanos profesionales. Otro ámbito en el que la extracción de información del modelo puede ser crucial son las ciencias. En pocas palabras, a los físicos, químicos y biólogos les interesa identificar las leyes ocultas de la naturaleza en lugar de limitarse a predecir una cantidad con modelos de caja negra. Por tanto, solo los modelos explicables son útiles en este ámbito.

- Cumplimiento de la legislación: Los sistemas de IA afectan cada vez a más ámbitos de nuestra vida cotidiana. Junto a esto, los aspectos legales relacionados, como la asignación de responsabilidades cuando los sistemas toman una decisión equivocada, también han recibido recientemente una mayor atención. Dado que puede ser imposible encontrar respuestas satisfactorias a estas cuestiones jurídicas cuando se confía en modelos de caja negra, los futuros sistemas de IA tendrán necesariamente que ser más explicables. Otro ejemplo en el que la normativa puede convertirse en una fuerza impulsora de una mayor explicabilidad en la inteligencia artificial son los derechos individuales. Las personas que se ven afectadas de forma inmediata por las decisiones de un sistema de IA (por ejemplo, las que son rechazadas por el banco para un préstamo) pueden querer saber por qué los sistemas han decidido de esa forma. Solo los sistemas de IA explicables proporcionarán esta información. Estas preocupaciones llevaron a la Unión Europea a adaptar una nueva normativa (Goodman y Flaxman, 2017) que implanta un "derecho a la explicación"por el que un usuario puede pedir una explicación de una decisión algorítmica que se haya tomado sobre él o ella. Estos ejemplos demuestran que la explicabilidad no solo tiene un interés académico importante y de actualidad, sino que desempeñará un papel fundamental en los futuros sistemas de IA.

1.2. Objetivos del trabajo

Objetivo General

El objetivo de esta tesina es profundizar en el estudio de las técnicas de extracción de reglas que permiten explicar las decisiones tomadas por las redes neuronales artificiales. Para realizar esto se tuvo que analizar la estructura y funcionamiento interno de las redes, en particular la influencia de cada neurona de entrada en la predicción.

Objetivos Específicos

1. Estudiar y analizar diversas técnicas de extracción de reglas utilizadas en la interpretación de redes neuronales.
2. Seleccionar una de dichas técnicas e implementarla
3. Realizar modificaciones sobre la técnica implementada para cambiar su enfoque hacia la explicabilidad de las redes.
4. Seleccionar métricas que puedan ser utilizadas para evaluar las reglas resultantes del proceso de extracción propuesto.
5. Realizar pruebas con conjuntos de datos conocidos y evaluar los resultados con las métricas seleccionadas.

1.3. Metodología de Trabajo

La investigación se llevó a cabo mediante un enfoque experimental que combinó el análisis teórico con la implementación práctica de una técnica de extracción de reglas. Se utilizaron conjuntos de datos de prueba específicos y se evaluó la efectividad de la técnica a través de métricas relevantes.

1.4. Resultados Obtenidos

Se construyó una técnica que permite representar a partir de reglas una red neuronal artificial y explicar fielmente las predicciones de dicha red. Además, se modificaron métricas existentes para abordar mejor este problema y mostrar que el enfoque utilizado tiene resultados destacables. Se realizó un análisis del funcionamiento de las técnicas de extracción de reglas que permite determinar la aplicabilidad de cada una según los requerimientos de cada tarea o las características de la red.

1.5. Estructura de la tesina

En la sección 2 se explican las redes neuronales artificiales y la relevancia de poder explicarlas junto a la forma de construir dicha explicación. En la sección 3 se enumeran las distintas técnicas existentes para la extracción de reglas. En la sección 4 se presentan en detalle las múltiples etapas del trabajo y como se construyó una nueva técnica. En la sección 5 se realiza un análisis de los resultados obtenidos por el nuevo algoritmo en las nuevas pruebas. En la sección 6 se hace un resumen del aporte realizado y se presentan las conclusiones a esta tesina. Y finalmente, en la sección 7 se plantean posibles líneas de trabajo futuro.

Capítulo 2

Marco Teórico

El presente capítulo, dedicado al marco teórico, se sumerge en los pilares esenciales que sustentan esta investigación. A lo largo de este capítulo, se explorarán conceptos fundamentales que sientan las bases para comprender a fondo las temáticas abordadas. Se explicarán conceptos claves, tales como el funcionamiento básico de una red neuronal, así como también que es la explicabilidad y por último, en que consiste la extracción de reglas. Esté capítulo proporcionará no solo una comprensión clara de estos conceptos, sino también una visión integral de cómo se aplicarán en el contexto de la investigación que se desarrolla en esta tesina.

2.1. Redes neuronales

Las redes neuronales artificiales (Russell y Norvig, 2021) están inspiradas en la capacidad de procesamiento de información del cerebro humano, la cual se piensa que proviene principalmente de redes de neuronas. Una neurona no es más que una célula del cerebro, la cual se ocupa principalmente de la recolección, procesamiento y emisión de señales eléctricas. Este principio biológico subyacente sirve como fundamento para el diseño y funcionamiento de las redes neuronales artificiales.

Al igual que la red neuronal en nuestro cerebro, que está compuesta por una red de neuronas, las redes neuronales artificiales están compuestas por nodos o unidades conectadas a través de conexiones dirigidas.

Cada nodo a su vez está compuesto por:

1. Conexiones de entradas: tienen un peso numérico asociado, que determina la fuerza y el signo de la conexión.

2. Función de entrada: suma ponderada de las entradas y está dada por el siguiente cálculo:
$$in_i = \sum_{j=0}^{n} W_{i,j} a_j \tag{2.1}$$

3. Función de activación: se diseña con dos objetivos. Primero queremos que la unidad esta «activa» (cercana a +1) cuando se proporcionen entradas «correctas», e «inactivas» (cercana a 0) cuando se den entradas «erróneas». Segundo, la activación tiene que ser no lineal:
$$g(in_i) = g(\sum_{j=0}^{n} W_{i,j} a_j) \tag{2.2}$$

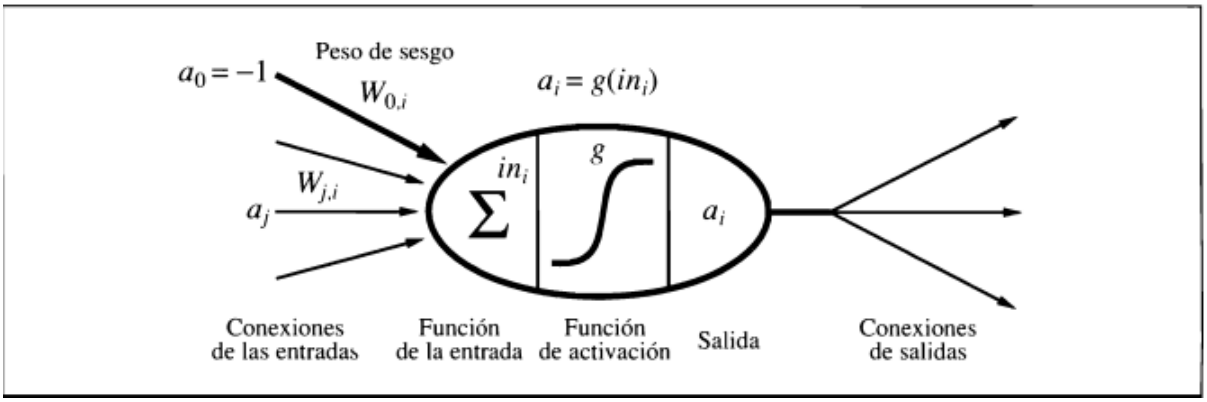

Figura 2.1: Representación de una neurona

4. Peso de sesgo: es un umbral de la unidad, que solo en caso de que se supere puede propagarse a otra neurona.

5. Salida: La salida es el resultado de aplicar la función de activación elegida a la función de entrada.

Una vez entendido que son las neuronas, vamos a explicar las diferentes estructuras de las redes, las cuales se pueden dividir en dos categorías principales:

1. Acíclicas o redes con alimentación-hacia-delante que representa una función de sus entradas actuales; de este modo, no tiene otro estado interno que no sea el de sus propios pesos.

2. Cíclicas o redes recurrentes en las cuales sus salidas pueden alimentar sus 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, las respuestas de la red dadas una entrada depende de su estado inicial, que dependerá de entradas previas. Por lo tanto, las redes recurrentes pueden tener memoria de corto plazo.

Por otra parte, las redes neuronales se pueden usar para clasificación o para regresión. Para clasificaciones booleanas con entradas continuas, es tradicional tener una única unidad de salida (con un valor por encima de 0,5 interpretados como una clase y con un valor por debajo de 0,5 de otra). Para clasificación en k-clases, se puede dividir el rango de la unidad de salida en k partes, pero es más común tener k unidades de salida separadas, donde el valor de cada una representa la verosimilitud relativa de esa clase dada una entrada actual.

A continuación se explicará en detalle las redes con alimentación hacia delante, ya que son las utilizadas en este trabajo.

2.1.1. Redes con alimentación hacia adelante

Este tipo de redes neuronales (Markowska-Kaczmar, 2008) están organizadas en capas y las neuronas en la misma capa no están conectadas entre sí, y solamente las neuronas de capas adyacentes tienen conexiones. Estas redes tienen una capa de entrada (input layer), una o más capas intermedias, llamadas capas ocultas (hidden layers) y una capa final con varias neuronas llamada capa de salida (output layer). Se habla de aprendizaje profundo

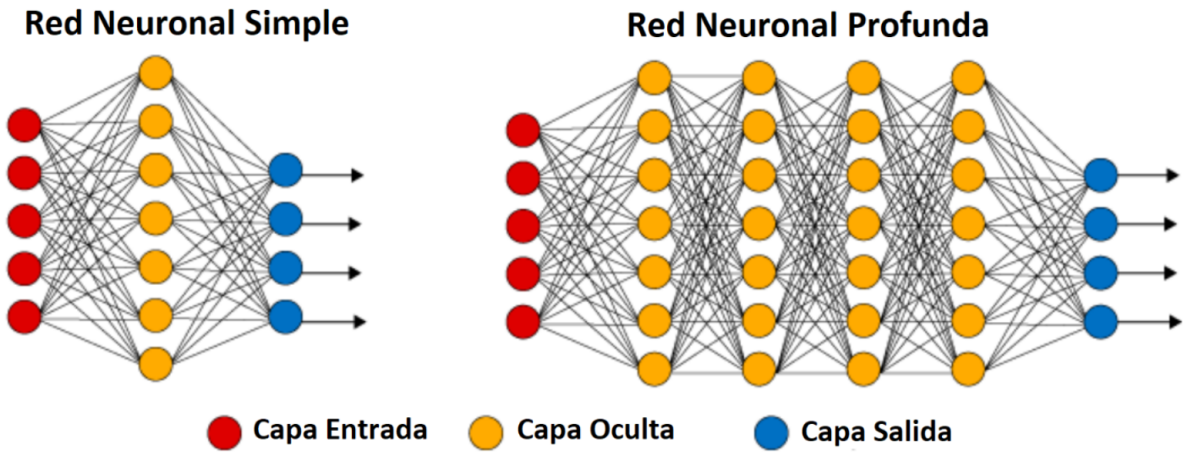

Figura 2.2: Redes neuronales simples vs. profundas

(deep learning) cuando la red neuronal artificial está compuesta por multiples capas ocultas. La diferencia se puede ver mejor en la figura (2.2).

Ahora que está clara la estructura de estas redes, la siguiente cuestión a tener en cuenta es ¿cómo aprenden las redes neuronales artificiales?.

2.1.2. ¿Cómo aprenden las redes neuronales artificiales?

Las redes neuronales artificiales son estructuras de datos. Se crean e inicializan con valores aleatorios y luego se someten a un proceso de entrenamiento. El algoritmo de entrenamiento más utilizado se denomina retro-propagación del error (En inglés, back-propagation) a grandes rasgos, consiste de los siguientes pasos:

1. Inicializar los pesos w_{ij} y los umbrales iniciales de cada neurona. Hay varias posibilidades de inicialización, siendo las más comunes las que asignan valores aleatorios pequeños.

2. Para cada par (x_i, y_i) del conjunto de los datos de entrenamiento:

   a) Obtener la predicción de la red para ese par. Esto se consigue propagando la entrada hacia adelante (feedforward).

   b) Evaluar la función de error: comprobando qué tan lejos está la predicción del valor verdadero conocido.

   c) Propagación hacia atrás con descenso de gradiente: calcular derivadas parciales de la función de error para encontrar el conjunto de pesos que minimizan la función de error.

   d) Actualizar pesos w_{ij} y umbrales.

   e) Calcular el error actual y volver al paso 2 si no es satisfactorio.

Este algoritmo se ilustra en la figura (2.3).

Generalmente, como función de costo se usa el error cuadrático medio. Es decir, que dado un par $(x_k, d_k$ correspondiente a la entrada k de los datos de entrenamiento, se calcula

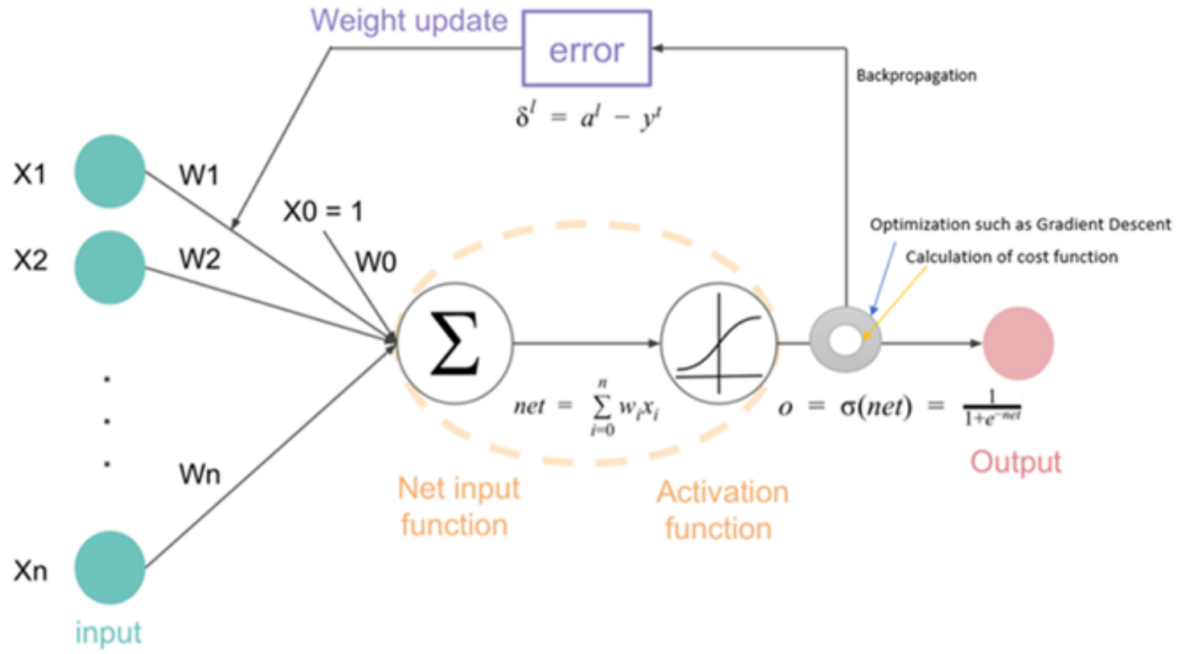

Figura 2.3: Algoritmo de aprendizaje por back-propagation

la suma de los errores parciales resultantes de la diferencia entre la salida deseada d_k y la salida que da la red, como se muestra en la ecuación (2.3).

$$E(w_{ij}, \theta_j, w'_{kj}, \theta'_k) = \frac{1}{2} \sum_p \sum_k \left[d_k^p - f\left(\sum_j w'_{kj} y_j^p - \theta'_k \right) \right]^2 \tag{2.3}$$

Sobre esta función de costo global se aplica algún procedimiento de optimización para minimizarla. Generalmente, se aplica el método descenso por gradiente, calculando las derivadas parciales respecto a cada una de las variables w_{ij} de la función de costo, como se muestra a continuación.

$$\delta w'_{kj} = -\varepsilon \frac{\partial E}{\partial w'_{kj}}$$

$$\delta w'_{ji} = -\varepsilon \frac{\partial E}{\partial w_{ji}}$$

$$\delta w'_{kj} = \varepsilon \sum_p \Delta_k^{'p} y_j^p \quad \text{con} \quad \Delta_k^{'p} = [d_k^p - f(v_k^{'p})] \frac{\partial f(v_k^{'p})}{\partial v_k^{'p}}$$

$$\delta w_{ij} = \varepsilon \sum_p \Delta_j^p x_i^p \quad \text{con} \quad \Delta_j^p = (\sum_k \Delta_k^{'p} w'_{kj}) \frac{\partial f(v_j^p)}{\partial v_j^p}$$

2.2. Explicabilidad: Métodos

En la sección anterior, se exploraron los conceptos básicos para comprender una red neuronal artificial y como se pueden alcanzar niveles altos en las predicciones. Sin embargo, la necesidad de comprender y validar el proceso de una decisión de un sistema de IA nos lleva a esta sección donde abordaremos diferentes metodologías de explicabilidad.

La explicabilidad está en el centro de una ciencia de datos responsable y abierta, en múltiples sectores industriales y disciplinas científicas. Diferentes comunidades científicas han estudiado el problema de explicar los modelos de decisión de aprendizaje automático. Sin embargo, cada comunidad aborda el problema desde una perspectiva diferente y le da un significado diferente a la explicación generada. La mayoría de los trabajos en la literatura provienen de las comunidades de aprendizaje automático y minería de datos. El primero se centra principalmente en describir cómo funcionan las cajas negras, mientras que el segundo está más interesado en explicar las decisiones, incluso sin comprender los detalles sobre cómo funcionan en general los sistemas de decisión opacos.

Se requiere un modelo interpretable para proporcionar una explicación. Por lo tanto, para lograr un modelo interpretable es necesario tener en cuenta el siguiente conjunto de aspectos mencionados por algunos documentos del estado del arte (Guidotti, Monreale, Pedreschi, y Giannotti, 2021), (Andrews, Diederich, y Tickle, 1995), (Doshi-Velez y Kim, 2017), (Freitas, 2014):

- Interpretabilidad: hasta que punto el modelo y/o la predicción son comprensibles para los humanos. La discusión más abordada se relaciona con la forma en que se puede medir la interpretabilidad. En la publicación (Freitas, 2014), un componente para medir la interpretabilidad es la complejidad del modelo predictivo en términos del tamaño del modelo. Según la literatura, nos referimos a la interpretabilidad también con el nombre de comprensibilidad.

- Precisión: hasta que punto el modelo predice correctamente las instancias no vistas. La precisión de un modelo se puede medir utilizando diversas medidas de evaluación como accuracy, F1 score, etc. Producir un modelo interpretable manteniendo niveles competitivos de precisión es el objetivo más común entre los documentos de la literatura.

- Fidelidad: captura que tan bueno es un modelo interpretable en la imitación del comportamiento de una caja negra. De manera similar a la precisión, la fidelidad se mide en términos de fidelity, F1-score, etc., pero con respecto al resultado de la caja negra que se considera como un oráculo.

En el análisis de la interpretabilidad de modelos predictivos, podemos identificar un conjunto de dimensiones a tener en cuenta, y que caracterizan la interpretabilidad del modelo (Doshi-Velez y Kim, 2017).

- Interpretabilidad global y local: Un modelo puede ser completamente interpretable, es decir, somos capaces de comprender toda la lógica de un modelo y seguir el razonamiento completo que conduce a todos los diferentes resultados posibles. En este caso, hablamos de interpretabilidad global. En cambio, indicamos con interpretabilidad local la situación en la que es posible comprender solo las razones de una decisión específica: solo la predicción/decisión individual es interpretable.

- Limitación de tiempo: Un aspecto importante es el tiempo que el usuario tiene disponible o se le permite dedicar a comprender una explicación. La disponibilidad de tiempo del usuario está estrechamente relacionada con el escenario en el que se debe utilizar el modelo predictivo. Por lo tanto, en algunos contextos donde el usuario necesita tomar rápidamente la decisión (por ejemplo, un desastre es inminente), es preferible tener una explicación simple de entender. Mientras que en contextos donde el tiempo de

decisión no es una limitación (por ejemplo, durante un procedimiento para otorgar un préstamo), uno podría preferir una explicación más compleja y exhaustiva.

- Naturaleza de la experiencia del usuario: Los usuarios de un modelo predictivo pueden tener diferentes conocimientos previos y experiencia en la tarea: tomadores de decisiones, científicos, ingenieros de cumplimiento y seguridad, científicos de datos, etc. Conocer la experiencia del usuario en la tarea es un aspecto clave de la percepción de la interpretabilidad de un modelo. Los expertos en el campo pueden preferir un modelo más grande y sofisticado en lugar de uno más pequeño y, a veces, más opaco.

Un análisis preciso y revisión de la literatura conducen a la identificación de diferentes categorías de problemas. A un nivel muy alto, podemos distinguir entre ingeniería inversa y diseño de explicaciones. En el primer caso, dada la decisión de registros producidos por un tomador de decisiones de caja negra, el problema consiste en reconstruir una explicación para ello. El conjunto de datos original sobre el cual se entrena la caja negra generalmente no se conoce en la vida real. En el segundo caso, dado un conjunto de datos de registros de decisiones de entrenamiento, la tarea consiste en desarrollar un modelo predictor interpretable junto con sus explicaciones. Para la tarea de generar una explicación, existen de acuerdo con (Guidotti y cols., 2021) las siguientes formas de realizar dicha tarea:

- Árbol de decisión (*Decision Tree*) o árbol único. Se reconoce comúnmente que el árbol de decisión es uno de los modelos más interpretables y fácilmente comprensibles, principalmente para explicaciones globales, pero también locales. De hecho, una técnica muy extendida para abrir la caja negra es la llamada "aproximación de árbol único".

- Reglas de decisión (*Decision Rules*) o Explicador basado en reglas. Las reglas de decisión son una de las técnicas más comprensibles para el ser humano. Se utilizan para explicar el modelo, el resultado y también para el diseño transparente. Se destaca la existencia de técnicas para transformar un árbol en un conjunto de reglas.

- Importancia de las características (*Features Importance*). Una solución muy sencilla, pero eficaz, que actúa como explicación global o local consiste en devolver como explicación el conjunto de características utilizadas por la caja negra junto con su peso.

- Máscara saliente (*Salient Mask*). Una forma eficaz de señalar lo que provoca un determinado resultado, especialmente cuando se tratan imágenes o textos, consiste en utilizar "máscaras"que resalten visualmente los aspectos determinantes del registro analizado. Suelen utilizarse para explicar las redes neuronales profundas.

- Análisis de sensibilidad (*Sensitivity Analysis*). Consiste en evaluar la incertidumbre en el resultado de una caja negra con respecto a distintas fuentes de incertidumbre en sus entradas. Suele utilizarse para desarrollar herramientas visuales de inspección de cajas negras.

- Gráfico de dependencia parcial (*Partial Dependence Plot*). Estos gráficos ayudan a visualizar y comprender la relación entre el resultado de una caja negra y la entrada en un espacio de características reducido.

- Selección del prototipo (*Prototype Selection*). Este explicador consiste en devolver, junto con el resultado, un ejemplo muy similar al registro clasificado, para dejar claro con

qué criterio se ha devuelto la predicción. Un prototipo es un objeto representativo de un conjunto de instancias similares y que forma parte de los puntos observados, o bien es un artefacto que resume un subconjunto de ellos con características similares.

- Activación de neuronas (*Neuron Activation*). La inspección de redes neuronales y redes neuronales profundas puede realizarse también observando cuáles son las neuronas fundamentales activadas con respecto a determinados registros de entrada.

Estas técnicas además pueden ser utilizadas sobre distintos tipos de cajas negras explicables, por lo que se enumeran a continuación las consideradas por (Guidotti y cols., 2021):

- Red neuronal (*Neural Network, NN*). Inspiradas en las redes neuronales biológicas, las redes neuronales artificiales aprenden a realizar tareas teniendo en cuenta ejemplos. Una NN está formada por un conjunto de neuronas conectadas. Cada enlace entre neuronas puede transmitir una señal. La neurona receptora puede procesar la señal y luego transmitirla a las neuronas conectadas a ella. Normalmente, las neuronas se organizan en capas. Las distintas capas realizan diferentes transformaciones en sus entradas. Las señales viajan desde la capa de entrada hasta la capa de salida, pasando varias veces por la capa o capas ocultas del medio. Las neuronas y las conexiones también pueden tener un peso que varía a medida que avanza el aprendizaje, lo que puede aumentar o disminuir la fuerza de la señal que envía.

- Ensamblado de árboles (*Tree Ensemble, TE*). Los métodos de ensamblaje combinan más de un algoritmo de aprendizaje para mejorar el poder predictivo de cualquiera de los algoritmos de aprendizaje individuales que combinan. Los bosques aleatorios, los árboles reforzados y el ensamblaje de árboles son ejemplos de Ensamblado de árboles. Combinan las predicciones de diferentes árboles de decisión, cada uno de ellos entrenado en un subconjunto independiente de los datos de entrada.

- Máquina de vectores de soporte (*Support Vector Machine, SVM*). Estas máquinas utilizan un subconjunto de los datos de entrenamiento, denominados vectores de soporte, para representar el límite de decisión, además son un clasificador que busca hiperplanos con el mayor margen para el límite de decisión.

- Red neuronal profunda (*Deep Neural Network, DNN*). Una red neuronal profunda es una red neuronal que puede modelar relaciones no lineales complejas con múltiples capas ocultas. Una arquitectura DNN está formada por una composición de modelos expresados como una combinación en capas de unidades básicas. En las DNN, los datos suelen fluir de la capa de entrada a la de salida sin bucle de retorno. Las DNN más utilizadas son las redes neuronales recurrentes (*Recurrent Neural Networks, RNN*). Un componente peculiar de las RNN son los nodos de memoria a largo plazo (*Long Short-Term Memory, LSTM*), que resultan especialmente eficaces para el modelado del lenguaje. Por otro lado, en el procesamiento de imágenes se suelen utilizar redes neuronales convolucionales (*Convolutional Neural Networks, CNN*).

Teniendo en cuenta las técnicas de extracción y los tipos de cajas negras, se procede a explicar el problema a resolver en este trabajo, llamado “Explicación del modelo de caja negra”: Dado un modelo de caja negra que resuelve un problema de clasificación, el problema de explicación de la caja negra consiste en proporcionar un modelo interpretable y transparente que sea capaz de imitar el comportamiento de la caja negra y que también sea comprensible

para los humanos. En otras palabras, el modelo interpretable que se aproxima a la caja negra debe ser interpretable a nivel global. En consecuencia, definimos el problema de explicación del modelo de caja negra de la siguiente manera:

Definición 1 (Explicación del modelo de caja negra). Dado un predictor de caja negra b y un conjunto de datos $D = X, Y$, el problema de explicación del modelo de caja negra consiste en encontrar una función

$$f : (X^m \rightarrow Y) \cdot (X^{n \cdot m} \text{ Ö } Y^n) \rightarrow (X^m \rightarrow Y) \tag{2.4}$$

que tome como entrada una caja negra b y un conjunto de datos D, y devuelva un predictor global comprensible cg, es decir, $f(b, D) = cg$, de modo que cg sea capaz de imitar el comportamiento de b, y exista una función explicadora global

$$\epsilon g : (X^m \rightarrow Y) \rightarrow E \tag{2.5}$$

que pueda derivar de cg un conjunto de explicaciones $E \in E$ que modelen de manera comprensible para los humanos la lógica detrás de cg, es decir, $\epsilon g(cg) = E$.

Para resolver este problema lo que se utilizara como caja negra será una red neuronal profunda (Deep neural network, o DNN) y como técnica para explicar dicha caja negra se utilizaran dos técnicas generadoras de reglas de decisión. Dicho proceso se conoce como extracción de reglas en redes neuronales.

2.3. Extracción de reglas en redes neuronales artificiales

En las secciones previas, se abordaron los conceptos fundamentales de las redes neuronales artificiales, así como la importancia de comprender y explicar las mismas para entender adecuadamente el proceso de toma de decisiones. En esta sección, nos enfocamos en uno de los métodos específicos diseñados para abordar esta necesidad: la extracción de reglas.

De acuerdo a (Averkin y Yarushev, 2021) la extracción de reglas en redes neuronales artificiales se puede definir de la siguiente manera: Dada una red neuronal artificial y los datos con los que la misma fue entrenada, se crea una descripción del razonamiento de la red neuronal que es comprensible, pero al mismo tiempo es cercano al comportamiento demostrado por la red. Hay tres formas de clasificar las distintas técnicas para extraer reglas:

1. Si el método solamente utiliza a la red neuronal como una caja negra, ignorando la estructura de la misma, entonces se lo considera un acercamiento pedagógico. En este tipo de método se tiene en cuenta la relación que existe entre las entradas y las salidas de la red. Los enfoques pedagógicos no tienen en cuenta la estructura interna de la red neuronal. El objetivo de los enfoques pedagógicos es considerar las redes neuronales entrenadas como un objeto integral o como una caja negra. La idea principal es extraer las reglas emparejando directamente los datos de entrada con los de salida. Los enfoques pedagógicos trabajan con una red neuronal como una función. Esta función establece la salida de la red neuronal para una entrada arbitraria, pero no permite comprender la estructura interna de la red neuronal ni sus pesos. Para una red neuronal, esta clase de algoritmos intenta encontrar una relación entre las posibles variaciones de entrada y las salidas generadas por la red neuronal, y algunos utilizan datos de entrenamiento predefinidos y otros no.

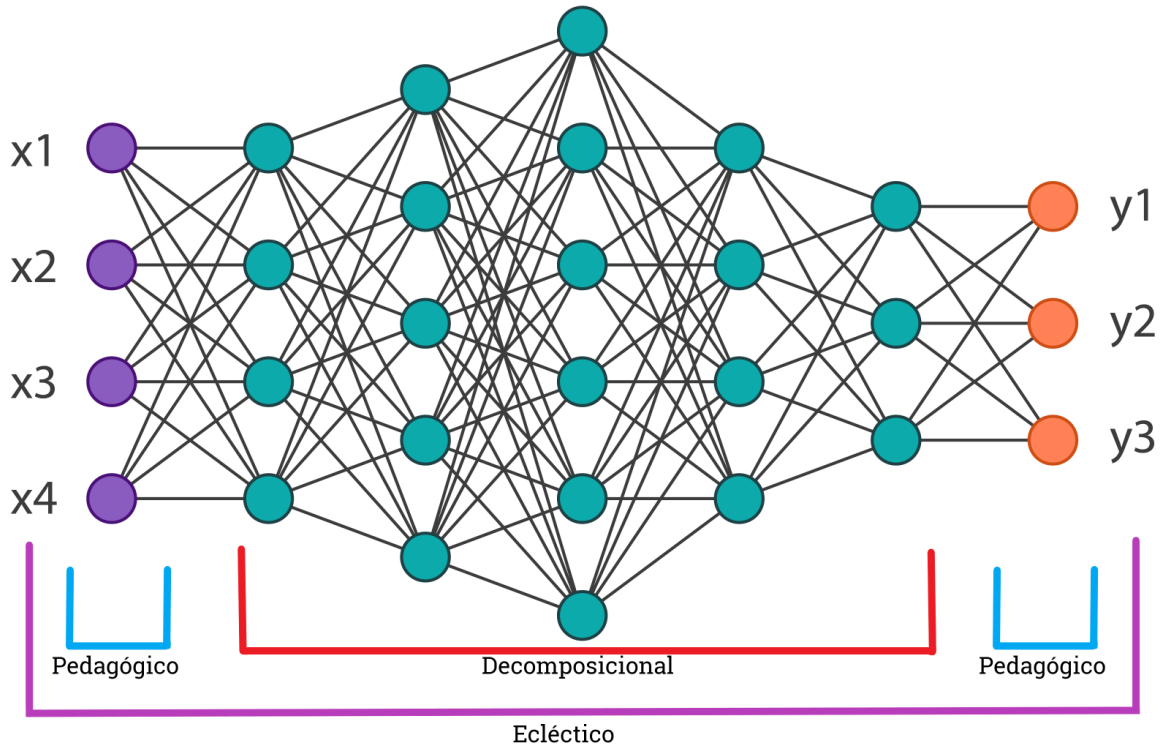

Figura 2.4: Secciones de una red neuronal que las distintas técnicas de extracción analizan para la construcción de las reglas

2. Cuando se tiene en cuenta la estructura interna de la red neuronal, se lo considera un acercamiento decomposicional. En este tipo de método se utilizan partes de la estructura de la red para la extracción.

3. En el caso de que el método tenga características de los acercamientos pedagógicos y de composicionales entonces se lo llama un acercamiento ecléctico.

2.3.1. Tipos de reglas

Cabe señalar que las reglas extraídas pueden diferir significativamente en su naturaleza y, por lo tanto, en su legibilidad para los expertos en la materia en función del método de extracción utilizado. Cada tipo de regla describe de forma diferente el significado y las correlaciones internas de los parámetros de entrada. A menudo, existe un equilibrio entre la legibilidad y un alto índice de clasificación. A continuación describiremos brevemente cada tipo de regla lógica de acuerdo a (Bondarenko, Zmanovska, y Borisov, 2010)

- **Reglas proposicionales If-Then / If-Then-Else:** Las reglas de este tipo constan de una cláusula condicional If y una cláusula de clasificación Then. Opcionalmente, la regla puede contener una cláusula de clasificación "Si". La parte de condición If de una regla proposicional es una combinación booleana de condiciones lógicas sobre las variables de entrada. La parte de la condición puede contener conjunciones, disyunciones y negaciones. Un ejemplo de regla de este tipo es "Si X = x e Y = y entonces Clase = A", con X, Y variables de entrada y x, y posibles valores de estas variables. Para variables de

entrada continuas, las condiciones suelen especificarse en forma de restricciones de rango sobre los valores, por ejemplo "$X \in [c1, c2]$ e $Y > c3$ con $c1, c2, c3 \in R$".

- **Reglas M-de-N:** Este tipo de reglas está estrechamente relacionado con las reglas proposicionales. Son expresiones de la forma "Si (al menos/exactamente/máximo) M de las N condiciones C1, C2, . . . , CN se cumplen, entonces Clase = A". Por lo general, estas reglas pueden transformarse fácilmente en reglas proposicionales. Por ejemplo, la regla M-de-N "si exactamente 2- de-X = a1, Y = a2, Z = a3 entonces Clase = A es lógicamente equivalente a "Si $((X = a1$ e $Y = a2)$ o $(X = a1$ y $Z = a3)$ o $(Y = a2$ y $Z = a3))$ $entonces$ $Clase = A$". Obsérvese que para $M = 1$ todas las reglas pueden escribirse como colección de disyunciones, mientras que en caso de M = N la expresión será conjunción de condiciones.

- **Reglas oblicuas:** Si bien los tipos de reglas mencionados anteriormente denotan regiones del hiperespacio mediante la definición de hiperplanos paralelos a los ejes, a veces puede ser necesario un gran número de condiciones para definir la clasificación con una calidad satisfactoria. Por otro lado, las reglas oblicuas son capaces de definir hiperplanos no paralelos a los ejes del espacio de entrada, por lo que una menor cantidad de reglas puede dividir satisfactoriamente el espacio en subespacios. Estas reglas tienen la forma Si $(a1X1 + a2X2 + a3 > a4)$ y $(a4X1 + a5X3 > c6)$ $entonces$ $Clase = A$.

- **Reglas de ecuación:** Este tipo de reglas es muy similar a las reglas oblicuas, pero define los límites utilizando ecuaciones de hiperplanos ligeramente más complejas, como elipsoides : $Si(a1X21+a2X22+a3X1X2+a4X1+a5X2+a6) \leq a7$ $entonces$ $Clase = B$. El inconveniente de estas reglas es que son más difíciles de entender.

- **Reglas difusas:** Las reglas de este tipo son muy similares a las reglas proposicionales Si-Entonces-Si, la única diferencia es que en lugar de lógica booleana, las reglas difusas son multivaluadas. En las reglas difusas, el universo de discurso de las variables es un conjunto difuso. Un ejemplo de regla de clasificación difusa es "Si X es alto e Y es medio, entonces Clase = A". Aquí alto, medio y bajo son miembros del conjunto difuso. En los conjuntos difusos, cada elemento se asocia con el grado de pertenencia correspondiente. Al igual que las reglas proposicionales, las reglas difusas son generalmente comprensibles, ya que operan con conceptos lingüísticos.

También hay que tener en cuenta que el conocimiento puede extraerse de la red de distintas formas, como la representación visual o las máquinas de estados. El principal factor de evaluación de las reglas extraídas es su calidad. Más concretamente, la calidad equivale a la comprensibilidad, que a su vez es una medida más o menos subjetiva en comparación con la precisión, ya que a menudo maximizar una de ellas conlleva cierta degradación de la otra.

En esta tesina se trabajó con reglas If-Then, que son las reglas resultantes del algoritmo implementado. Este algoritmo es evaluado con distintas métricas que se explicarán a continuación.

2.3.2. Métricas de evaluación

Para lograr el objetivo de este trabajo, las reglas extraídas deben replicar el comportamiento de la red neuronal artificial entrenada de la forma más fiel posible. En este escenario el razonamiento interno de la red neuronal artificial se considera la parte más importante del proceso. Por lo tanto, es muy claro que la fidelidad va a tener un rol fundamental para

determinar la calidad de las reglas generadas. Y aún más, la precisión de las reglas se vuelve irrelevante porque no importa si las reglas generalizan bien o no, lo importante es que reproduzcan fielmente la red neuronal artificial. Reglas orientadas a una alta precisión no explican como funciona la red. Para evaluar la efectividad del algoritmo de extracción de reglas es necesario poder medir distintos aspectos de los resultados a lo largo del proceso. Para este fin se definen las siguientes métricas:

- La **fidelidad** define la capacidad de las reglas extraídas para imitar el comportamiento de la red de la que se extrajeron. Sea D = D1, ...Dn un conjunto de ejemplos. Para cada ejemplo, Di , donde Di D, la fidelidad es "1" si para dicho ejemplo el conjunto de reglas y la red neuronal tienen como resultado la misma clasificación,

$$\text{Fidelidad}_i = \begin{cases} 1 \text{ If(reglas}(D_i) = \text{red neuronal}(D_i)) \\ 0 \text{ en caso contrario} \end{cases} \tag{2.6}$$

  Para N ejemplos, la fidelidad está dada por:

$$\text{Fidelidad} = \frac{\sum_{i=1}^{N} \text{Fidelidad}_i}{N} \tag{2.7}$$

  Es la métrica más importante porque tiene como objetivo reproducir la clasificación de la red neuronal con las reglas resultantes y que las mismas reflejen como funciona por dentro la red.

- La **precisión** de la regla se define como el porcentaje de datos clasificados correctamente por las reglas extraídas.

$$\text{Precisión} = \frac{\text{Cantidad de ejemplos correctamente clasificados}}{\text{Cantidad total de ejemplos}} \tag{2.8}$$

- La **comprensibilidad** de las reglas se relaciona con la facilidad con que las personas puedan interpretarlas . De acuerdo a (Augasta y Kathirvalavakumar, 2012) tiene en cuenta dos aspectos:

  - Comprensibilidad global: Se relaciona con la cantidad reglas.
  - Comprensibilidad individual: Se relaciona con la cantidad de condiciones en cada regla.

  La compresibilidad disminuye a medida que aumenta la cantidad de reglas y/o condiciones dentro de cada regla.

  En esta tesina se propone analizar la comprensibilidad de otra forma, en donde el cálculo de comprensibilidad esté conformado por:

  - Comprensibilidad global: La comprensibilidad del conjunto de reglas
  - Comprensibilidad individial: La comprensibilidad de una regla específica

Se elaboró una fórmula para el cálculo de la comprensibilidad en un conjunto de reglas con el objetivo de obtener una métrica numérica que ayude a determinar si la mejora de la fidelidad en las reglas resultantes también puede ser acompañada por una disminución del tamaño de las mismas y a su vez las haga mas comprensibles.

La comprensibilidad de una regla es la siguiente:

$$\text{Comprensibilidad de una regla} = (w1 * (1/T)) + (w2 * (1/K)) \tag{2.9}$$

Donde T es la cantidad de términos en dicha regla, K es la cantidad de atributos presentes en la regla, w1 es un peso para determinar la importancia de los términos en la comprensibilidad y w2 es un peso para determinar la importancia de la cantidad de atributos en el cálculo de comprensibilidad.

En esta tesina se utilizarán los pesos de 0.7 y 0.3 para w1 y w2 respectivamente, dando mayor importancia a la cantidad de términos en una regla por sobre la cantidad de atributos presentes en dicha regla.

La comprensibilidad del conjunto de reglas es el siguiente:

$$\text{Comprensibilidad total} = \frac{\sum_{i=1}^{N} \text{Comprensibilidad de regla i}}{N} \tag{2.10}$$

Donde N es la cantidad de clases del conjunto de datos o a su vez, la cantidad de reglas generadas.

Resumen

En este capítulo se explicaron en detalle los conceptos importantes para entender qué son las redes neuronales y cómo aprenden. Además, cómo se pueden interpretar las mismas, qué significa extraer reglas de dichas redes, qué tipos de reglas existen y qué métricas pueden utilizarse para evaluar un algoritmo que extrae reglas.

En el próximo capítulo se presentarán los trabajos relacionados que sirvieron para guiar la investigación realizada y tener una visión amplia de los distintos tipos de técnicas existentes.

Capítulo 3

Trabajos relacionados

A la hora de abordar la tarea de generar explicaciones a partir de redes neuronales artificiales, nos encontramos con una variedad de técnicas y enfoques, cada uno con sus propios procedimientos y representaciones para lograr una explicación comprensible y útil. Estas técnicas permiten desentrañar el funcionamiento interno de las redes neuronales y traducirlo en información interpretable y valiosa para comprender cómo toman decisiones y cómo procesan la información.

A continuación se mencionarán distintos algoritmos que dirigieron la investigación y permitieron explorar una gran variedad de técnicas

3.1. Métodos Pedagógicos

- BIO-RE es el acrónimo de Binarized Input-Output Rule extraction (Extracción de reglas binarias de entrada-salida) propuesto por (Taha y Ghosh, 1999). Es un algoritmo de caja negra que extrae reglas binarias de cualquier red neuronal generando primero la salida de la red para cada posible patrón de entrada y luego una tabla de verdad concatenando cada patrón de entrada con su correspondiente salida de red. Luego, a partir de los valores de la tabla de verdad, genera una función booleana a partir de la tabla de verdad. Finalmente, dicha función booleana funciona como una regla de la red.

- En (Kahramanli y Allahverdi, 2009) se propone un algoritmo de extracción de reglas basado en la búsqueda ortogonal (OSRE). Se trata de un método basado en un enfoque pedagógico. Se aplica tanto a máquinas de vectores soporte (SVM) como a redes neuronales artificiales (ANN). Convierte la entrada dada en la forma 1 de N y luego realiza la extracción de reglas basándose en las respuestas de activación. Y los resultados muestran que OSRE puede utilizarse para extraer reglas consistentes y precisas tanto de SVM como de ANN.

3.2. Métodos Decomposicionales

- En (Setiono, Baesens, y Mues, 2008) se propuso un algoritmo de extracción de reglas recursivas (Re-RX) que genera reglas de clasificación a partir de conjuntos de datos que tienen atributos tanto discretos como continuos. El algoritmo es recursivo por

naturaleza y genera reglas jerárquicas. Las condiciones de las reglas que tienen atributos discretos son disjuntas de las que tienen atributos continuos.

- DeepRED (Deep neural network Rule Extraction via Decision tree induction) publicado por (Zilke, Loza Mencía, y Janssen, 2016).

  DeepRED es un algoritmo de extracción de reglas diseñado específicamente para abordar redes neuronales profundas (DNNs), llenando una brecha existente en la consideración de DNNs en la mayoría de los algoritmos de vanguardia. Se basa en la extensión del algoritmo CRED. Utiliza C4.5 para crear reglas que describen las neuronas basándose en las neuronas de la capa anterior. El proceso de extracción en DeepRED descompone la DNN en reglas intermedias para cada capa, y luego realiza una fusión final para producir un conjunto de reglas que imitan el comportamiento global de la DNN en función de sus entradas. En esencia, DeepRED es capaz de extraer reglas precisas y comprensibles que representan el comportamiento de una DNN, permitiendo una mejor comprensión de su funcionamiento y comportamiento a partir de las reglas generadas.

- En la publicación de (Hayashi, 2013) se propuso el algoritmo EnsembleRecursive-Rule eXtraction (E-Re-RX) que utiliza una red neuronal ensemble que consigue altas tasas de reconocimiento. Se trata de un algoritmo basado en la aplicación del algoritmo Re-Rx a una red neuronal ensemble. Para la red neuronal ensemble se utiliza un número mínimo de redes neuronales, es decir, dos. Este algoritmo puede tratar tanto atributos discretos como continuos. En la primera parte del algoritmo se generan reglas primarias y en la siguiente parte se generan reglas secundarias. A continuación, ambos conjuntos de reglas se integran y se genera un conjunto de reglas final.

- El algoritmo Full Rule Extraction (full-RE) propuesto por (Taha y Ghosh, 1999) es capaz de extraer reglas precisas sin necesidad de binarizar o normalizar los atributos continuos de los datos antes del entrenamiento de la red. Para cada nodo oculto de la red, full-RE genera una regla intermedia que predice su salida en función de las combinaciones lineales de los atributos de entrada. Para obtener el conjunto final de reglas de clasificación que no incluyan pesos de red en sus condiciones de regla, full-RE discretiza los atributos de entrada y, a continuación, resuelve un problema de programación lineal para seleccionar el límite de discretización pertinente.

- En (Etchells y Lisboa, 2006) se propone un método que utiliza una red neuronal con una función de activación optimizada. Se añade un término de penalización en la función de activación para obtener una mejor aproximación. Este método se aplica a dos conjuntos de datos. Los resultados se comparan con los de otros métodos y muestran que las reglas extraídas tienen mayor precisión.

- ECLAIRE (Zarlenga, Shams, y Jamnik, 2021) es un algoritmo de extracción para Redes Neuronales Profundas, su nombre proviene de "**E**fficient **CLA**use-w**I**se **R**ule **E**xtraction". ECLAIRE evita la intratabilidad común a otros métodos descomposicionales haciendo uso de un proceso en dos pasos que utiliza eficientemente el espacio latente de una DNN para guiar su extracción de reglas. Construye un conjunto de reglas iterando sobre cada capa oculta e induce reglas con un algoritmo para formar un conjunto de reglas que luego procesa sustituyendo las cláusulas de cada capa con las de la capa anterior para finalmente armar un conjunto de reglas que asigne valores de entrada y los asocie a clasificaciones concretas.

3.3. Métodos Eclécticos

- El algoritmo ERENNR (Eclectic Rule Extraction from Neural Network Recursively) de (Chakraborty, Biswas, y Purkayastha, 2019) es una herramienta diseñada para extraer reglas de clasificación simbólicas a partir de una red neuronal de avance de una sola capa. Su novedad radica en su enfoque analítico de los nodos en la red neuronal. El proceso de extracción de reglas comienza con el análisis de un nodo oculto, el cual se basa en los rangos de datos de los atributos de entrada en relación con su salida. Luego, se procede a analizar un nodo de salida utilizando combinaciones lógicas de las salidas de los nodos ocultos en función de la clase de salida. Finalmente, el algoritmo genera un conjunto de reglas al moverse en dirección inversa, comenzando desde la capa de salida. Para cada regla en el conjunto, el algoritmo repite el proceso de extracción si la regla cumple ciertos criterios específicos. Esto permite refinar y mejorar las reglas generadas, garantizando que sean precisas y útiles para describir el comportamiento y la toma de decisiones de la red neuronal en la tarea de clasificación.

- En (Hruschka y Ebecken, 2006) han propuesto un algoritmo genético de agrupamiento (clustering genetic algorithm, CGA) para extraer reglas de perceptrones multicapa. Se trata de un enfoque basado en la agrupación que emplea CGA para encontrar grupos de valores de activación de unidades ocultas y genera reglas lógicas a partir de estos grupos.

- En (Setiono y Leow, 2000) se propuso un algoritmo para la "Extracción rápida de reglas de redes neuronales" (FERNN), que extrae las reglas sin poda de pesos, ya que el entrenamiento y la poda repetitivos de la red es un proceso que lleva mucho tiempo. El algoritmo identifica primero las conexiones relevantes desde las unidades de entrada a las unidades ocultas útiles basándose en las magnitudes de sus pesos. Y después se aplica el algoritmo C4.5 para identificar las unidades ocultas útiles basándose en la información contenida en estas unidades. Por tanto, puede decirse que FERNN es un algoritmo mixto: realiza el análisis de los pesos de entrada a oculto, pero utiliza el enfoque de caja negra (empleando C4.5) para los pesos de oculto a salida.

Resumen

Las distintas publicaciones fueron analizadas para entender los distintos acercamientos a la extracción de reglas. A continuación se explicará y presentará el camino que siguió la investigación realizada y como las técnicas encontradas influyeron en las decisiones tomadas a lo largo de la construcción de esta tesina.

Capítulo 4

Trabajo realizado

Inicialmente, el objetivo de esta tesina era el de comparar múltiples técnicas de extracción de reglas existentes para hacer un análisis detallado sobre las ventajas y desventajas de los distintos métodos (pedagógico, decomposicional y ecléctico).

El obstáculo que se encontró fue que en gran parte de las publicaciones donde se presentaban los distintos algoritmos no se podía obtener de ninguna forma el código fuente para probar o analizar y en las que sí, el código estaba desactualizado o no se pudieron replicar los resultados mencionados por sus autores. Este obstáculo nos condujo por el camino de realizar una selección de algoritmos para reimplementar partiendo desde las descripciones y pseudocódigos disponibles en las publicaciones pertinentes.

Luego de analizar varios algoritmos, se seleccionó el algoritmo de (Augasta y Kathirvalavakumar, 2012), llamado RxREN (Rule Extraction by Reverse Engineering from Neuronal Networks), este método al tener un acercamiento pedagógico y que el análisis que realiza sobre la red sea solamente sobre la relación de las entradas y las salidas, lo convirtió en el candidato perfecto para ser reimplementado.

Pero antes de explicar reimplementación realizada, y la posterior modificación al algoritmo, hay qué determinar lo que caracteriza a RxREN.

4.1. Características de RxREN

Una de las características principales de RxREN es el tratamiento de la red como una "caja negra", y debido a su acercamiento pedagógico, solamente tiene en cuenta las entradas y salidas que tiene la red y no profundiza en la arquitectura interna.

Es en esa relación entre ambas partes de la red que RxREN realiza una ingeniería inversa analizando las entradas y de esa manera determina cuáles entradas de la red neuronal artificial son aquellas que influyen verdaderamente la salida de la red y cuáles son innecesarias a la hora de clasificar elementos.

Al descubrir las neuronas innecesarias, RxREN procede a apagarlas. Esto significa que los pesos de salida de la neurona se vuelven cero, y, por lo tanto, las neuronas apagadas dejan de influir en los cálculos de la red neuronal artificial.

Finalmente, usando las neuronas que se mantuvieron encendidas, el algoritmo construye reglas que expliquen lo mejor posible los datos con los que fue entrenada la red neuronal.

A continuación se describe en mayor detalle el proceso de extracción de reglas llevado a cabo por el algoritmo RxREN:

4.1.1. Fases del algoritmo

RxREN se divide en dos fases. En la primera construye reglas iniciales y en la segunda, se modifican dichas reglas para mejorar los resultados.

Fase uno

Esta fase incluye tres tareas: encontrar para cada entrada los elementos clasificados incorrectamente, eliminar aquellas neuronas que son insignificantes, y construir la matriz para representar las reglas. A continuación se explican cada una de estas tareas con más detalle.

1. Consiste en "apagar" una neurona de entrada. Esto significa actualizar con peso 0 a todas las conexiones de salida de esa neurona. De esta manera se logra que esa neurona no influya en el resultado de la red. Luego, se realiza una evaluación de la red clasificando todos los elementos del conjunto de datos y se registran los elementos mal clasificados debido a la ausencia de esa neurona específica. Este proceso se realiza para cada una de las neuronas de entrada.

2. En esta tarea se compara la precisión de la red original con la obtenida apagando una neurona. Si la diferencia es menor o igual a 1 %, entonces el aporte de esa neurona a la red se considera insignificante, y se mantendrá apagada. Esto se realizará por cada una de las neuronas de entrada.

3. Se construye la matriz para representar las reglas. Cada fila representa un atributo de los datos de entrada (vinculado con una neurona de entrada). Cada columna representa una clase de las posibles salidas de la red. En los casos en que la neurona influye lo suficiente para clasificar elementos, se guarda la información en la matriz en la/las posición/es correspondientes. Esto afectará a la fila correspondiente a esa neurona. Se analiza para cada columna si la neurona influye lo suficiente para clasificar en esa clase y en caso de hacerlo, se guardan dos números: el valor mínimo y máximo que tienen los elementos del conjunto de datos en ese atributo específico para ser clasificados como la clase correspondiente. En caso de que la neurona no influya lo suficiente, se guardará un valor nulo.

Fase dos

Se realizan tres tareas importantes, como son la construcción de reglas, la poda de reglas y la actualización de reglas.

1. Construcción de las reglas: Utilizando la matriz anteriormente creada, a partir de los valores existentes, se empiezan a extraer las reglas. Por cada clase se construye una condición con el formato siguiente:

$mínimo \leq atributo \leq máximo \rightarrow Clase$. Se construye una regla if then compuesta por las condiciones obtenidas de la matriz, conectadas mediante conjunciones, formando la condiciòn completa de la regla. Finalmente, se ordenan las reglas if then por la cantidad de condiciones, de mayor a menor, con el objetivo de tener las reglas más pequeñas posibles y que no haya redundancia de condiciones. Teniendo en cuenta que las reglas se evalúan de forma secuencial, es decir, la primera regla que satisface las condiciones es la que clasificará finalmente al dato, la última regla podría escribirse como ELSE.

2. Poda de las reglas: Para ayudar a reducir el número de condiciones de una regla y mejorar su comprensibilidad y generalización, a cada condición existente de la matriz de condiciones se la anula y se calcula nuevamente la precisión de la regla. Si la precisión es igual o mayor, entonces esa condición permanecerá anulada, ya que esa condición no aporta a la clasificación de la red. Si permanece, agregaría especificidad innecesaria.

3. Actualización de las reglas: Una vez podadas las condiciones innecesarias, se actualizan los rangos de las condiciones restantes. Por cada posición de la matriz, se analizan los elementos del conjunto de datos utilizados en la construcción de ese rango buscando reducirlo. Para ello, se calcula el segundo mínimo y máximo y se determina si con esos elementos se mejora la precisión de la regla. Si ese es el caso, se actualizan los valores del rango. Este proceso se repetirá con los siguientes valores mínimos y máximos, con el objetivo de mejorar la precisión tanto como sea posible.

Una vez implementado RxREN, se realizaron pruebas para analizar las reglas resultantes. Al calcular la métrica de fidelidad se observaron resultados bajos, y se determinó que esto ocurre debido a que el algoritmo no busca explicar la red neuronal artificial, sino que busca crear reglas de decisión para explicar los datos que se utilizaron para entrenar la red neuronal, o en otras palabras, es una técnica utilizada para minería de datos.

Dado que el objetivo de esta tesina es generar reglas fieles, se propuso una modificación del algoritmo que se centre en representar correctamente (es decir, fielmente) la red con las reglas generadas, y es lo que se presentará a continuación.

4.2. Algoritmo propuesto: FORxREN (Fidelity Oriented Rule extraction by Reverse Engineering of Neural networks)

Se propuso una modificación en el enfoque del algoritmo RxREN, que implica un cambio en el objetivo. En lugar de buscar obtener una precisión similar a la red mediante las reglas, se busca tener una alta fidelidad respecto a la red, ya que la fidelidad es un indicador directo del grado de representación de la red que tiene un conjunto de reglas y, por lo tanto, se lo considera más representativo de la red que el nivel de precisión. Para poder lograr este objetivo, se modificaron las siguientes secciones del algoritmo:

Preprocesamiento: El algoritmo original utiliza solo los datos correctamente clasificados por la red. En la modificación propuesta, se utilizan todos los datos del conjunto, reemplazando las etiquetas de clase originales con aquellas con las que efectivamente predice la red

4.2.1. Cambios en la Fase uno:

- Segunda tarea: Se "apaga" una neurona de entrada, y utilizando el preprocesamiento explicado anteriormente, se calcula el nivel de fidelidad con dicha neurona apagada. Si el nivel de fidelidad disminuye un porcentaje menor o igual a 5 % entonces la neurona se dejará apagada. Este proceso se realiza para cada una de las neuronas de entrada.

4.2.2. Cambios en la Fase dos:

- Poda de las reglas: Para ayudar a reducir el número de condiciones en las reglas y mejorar su comprensibilidad y generalización, a cada condición existente de la matriz se la anula y se calcula la fidelidad. Si la fidelidad disminuye entonces se restaura la condición.

- Actualización de las reglas: Por cada posición de la matriz [atributo, clase] se busca tomar los elementos del conjunto de datos que se usaron para construir ese rango y buscar el segundo mínimo y máximo, luego calcular la fidelidad y si la fidelidad mejora entonces se actualizan los valores del rango. Esta acción se sigue realizando hasta mejorar la fidelidad y acortar el rango de la clasificación lo máximo posible

4.3. Algoritmo FORxREN paso a paso

Entrada del algoritmo: Una red neuronal entrenada con **L** neuronas de entrada, **h** neuronas ocultas, **n** neuronas de salida y un conjunto de datos con **np** ejemplos. **Notaciones:**

- **T** Conjunto de datos etiquetados por la red.

- **NF** Fidelidad de la red neuronal con alguna/s neurona/s apagada/s respecto a su versión anterior.

- **B** Conjunto de neuronas de entrada insignificantes.

- **PF** Fidelidad de la RNA podada respecto a la red original.

- **m** Número de neuronas de entrada activas en la RNA podada.

- I_i i-ésima neurona en la capa de entrada I.

- **Ck** k-ésima clase objetivo.

- E_i Conjunto de ejemplos incorrectamente clasificados de la RNA sin I_i.

- err_i Cardinalidad de E_i

- $q_{i,k}$ Cantidad de ejemplos de E_i pertenecientes a la clase C_k

- **RF** Fidelidad de las reglas respecto a la red original.

4.3.1. Fase uno paso a paso

Paso 1: Para cada neurona de entrada I_i de la RNA entrenada, encontrar los ejemplos clasificados incorrectamente E_i de la RNA sin I_i .
Paso 2: Inicializar el valor de m como i.
Paso 3: Calcular el umbral $\theta = \min(\text{err}_i),\ i = 1...m$
Paso 4: Formar el conjunto $B = I_i / \text{err}_i = \theta$, el conjunto de neuronas de entradas insignificantes.
Paso 5: Forma la red podada temporal eliminando todas las neuronas de entrada insignificantes de B de la RNA entrenada.
Paso 6: Calcular PF temporal en el conjunto de datos comparando con la salida de la red inicial.
Paso 7: Si ($PF \geq (NF - 5\%)$) entonces se acepta esta red podada, $NF = PF$ y regresa al paso 3.
Paso 8: Para cada neurona de entrada I_i en la red podada, agrupar los ejemplos pertenecientes a E_i con respecto a cada clase objetivo Ck y encontrar el número de ejemplos $q_{i,k}$, seleccionando solo las clases de E_i que satisfacen la condición $q_{i,k}\alpha \cdot \text{err}_i$, donde $\alpha \in [0,1;0,5]$ y encontrar su valor mínimo, llamado MINik, y su valor máximo, llamado MAXik, para construir las reglas.

4.3.2. Fase dos paso a paso

Construcción de las reglas

Paso 9: Para k = 1...n hacer los pasos del 10 al 14.
Paso 10: j = 1 que es el contador de condiciones
Paso 11: Para i = 1..m
Paso 12: Para la clase k, si la i-ésima neurona de entrada I_i fuera seleccionada en el paso 8 para construir la regla, entonces $\text{condicion}_j = (\text{data}(I_i) \geq \text{MIN}ik \wedge \text{data}(I_i) \leq \text{MAX}ik)$.
Paso 13: si (j = 1) entonces condicion = condicion_j sino condicion = condicion $\wedge$ condicion_j ; incrementar j en 1
Paso 14: Se escribe la regla para la clase k usando el formato de la regla "if then". Por ejemplo: Rk = (If condicion then class = Ck)

Poda de reglas

Paso 15: Se calcula RF **Paso 16:** Para cada regla construida Rk, se analizan tres situaciones:

1. Se calcula PF sin la condicion_j . Si $PF \geq RF$ entonces la condicion_j debe ser eliminada de la regla y RF = PF

2. En caso contrario, se calcula PF con la condicion_j podada a izquierda (sin $\text{data}(I_i) \geq \text{MIN}ik$) . Si $PF \geq RF$ entonces la regla quedara podada a izquierda y RF = PF.

3. En otro caso, se calcula PF con la condicion_j podada a derecha (sin $\text{data}(I_i) \leq \text{MAX}ik$) . Si $PF \geq RF$ entonces la regla quedará podada a derecha y RF = PF.

Actualización de reglas

Paso 17: Se clasifican los ejemplos utilizando las reglas podadas.
Paso 18: Se calculan el valor mínimo y máximo de los ejemplos clasificados incorrectamente correspondientes a cada clase de cada atributo de la red podada.
Paso 19: Si la fidelidad aumenta después de considerar el nuevo valor mínimo y máximo seleccionado para los ejemplos de datos clasificados incorrectamente de las clases, entonces se deben actualizar los valores existentes de MINik y MAXik de las condiciones correspondientes de las reglas.
Paso 20: Se modifican los límites inferior y superior de cada condicion$_j$ mediante los valores consecutivos mínimos y máximos de los ejemplos de datos clasificados incorrectamente, respectivamente, hasta que exista alguna mejora en su fidelidad

Salida: Un conjunto de n reglas para un conjunto de datos dado.

4.3.3. Configuraciones

El sistema permite tres configuraciones para las reglas resultantes del algoritmo.

- **Modo 1:** Se realiza una ejecución completa del algoritmo explicado previamente (filtro de neuronas, podado y actualización). Como resultado se obtiene un conjunto de reglas con la mínima cantidad de condiciones y rangos lo más ajustados posible que representan la red. Esto garantiza que las reglas sean lo más específicas posible, sin perder fidelidad.
- **Modo 2:** Se realiza el filtrado de neuronas insignificantes y el podado de las reglas, dejando de lado la actualización de intervalos. El resultado es similar al modo 1, pero los rangos son más amplios, por lo tanto, las reglas son más generales manteniendo la fidelidad.
- **Modo 3:** Solo se efectúa el filtrado de neuronas. Al no hacer pruning las reglas son más largas y se pierde generalidad.

4.4. Resultados parciales

Lo trabajado hasta este punto fue escrito en una publicación para las JAIIO 52 (Jornadas Argentinas de Informática) (Moschettoni, Jacinto, Pérez, y Pons, 2023) y allí se presentó FORx-REN y sus resultados satisfactorios a la hora de extraer reglas fieles.

Continuando con el trabajo, se decidió trabajar sobre otra técnica que se pueda utilizar para comparar con FORxREN y de esa manera construir el eje de la tesina. Es así como la técnica ECLAIRE (mencionada en trabajos relacionados) fue seleccionada para ser comparada.

Dicha técnica composicional lleva en su publicación un hipervínculo al código fuente y el mismo pudo ser puesto a prueba para corroborar los resultados, lamentablemente no se pudieron replicar los resultados, y tampoco se pudo adecuar la técnica para realizar una comparación que deje en evidencia las ventajas de desventajas de los distintos acercamientos a la extracción de reglas.

Se cambió el enfoque de esta tesina en el aporte que significa la técnica de extracción de reglas FORxREN, y se continuo trabajando en mejorar su implementación. Las modificaciones realizadas en la técnica para facilitar su uso y optimizar su funcionamiento serán explicadas en mayor detalle a continuación.

4.4.1. Modificaciones realizadas

Se realizaron modificaciones con el proposito de simplificar la incorporación de nuevos conjuntos de datos numéricos. Anteriormente, en las primeras versiones, era necesario generar una subclase de la técnica y ajustarla para trabajar con conjuntos de datos específicos, esto. Además, se definió un método específico para la extracción de reglas, junto a parámetros que permiten un fácil acceso a la configuración de las ejecuciones (*softcoding*). Por último se trabajo en general en la legibilidad del código, su documentación y la adición de nuevas configuraciones para tener un detalle durante la ejecución. Todo esto es para permitir que pueda ser fácilmente extendido en el futuro

4.4.2. Resumen y conclusión

En este capítulo se presentó el algoritmo FORxREN junto a las modificaciones realizadas al algoritmo RxREN. Además, se presentaron las pruebas realizadas y sus resultados, donde se obtuvo un algoritmo que genera reglas con una fidelidad destacable sobre la red neuronal artificial base. A continuación se procede a presentar las nuevas pruebas realizadas y sus resultados.

Capítulo 5

Evaluación empírica del algoritmo y sus resultados

Las pruebas realizadas son extracciones de reglas utilizando tres datasets puntuales: Iris, WBC y Wine. A continuación se realiza una descripción detallada de cada uno de ellos.

5.1. Descripción de los datos

1. Conjunto de datos Iris: Este conjunto de datos consta ejemplos de distintas flores con sus características físicas. Las clases objetivo se codifican como 0 para la clase Iris setosa, 1 para la clase Iris versicolor y 2 para la clase Iris virginica.

2. Conjunto de datos de cáncer de mama de Wisconsin (WBC): Este conjunto de datos contiene información para determinar la presencia de cáncer de mama a partir de datos de estudios realizados sobre glándulas mamarias. Las clases objetivo se codifican como 0 para benigno y 1 para maligno.

3. Conjunto de datos de vinos (Wine): Este conjunto de datos contiene información de componentes químicos sobre vinos cultivados. Las clases objetivo son los 3 posibles productores que cultivaron dicho vino y, en el conjunto de datos, no se especifica quien es quien, simplemente se las llama "$class\_x$" donde "x.^{es} el número de productor, y hay "$class\_0$", "$class\_1$" y "$class\_2$".

En el Cuadro 5.1 se presenta un resumen de los conjuntos de datos.

Dataset	Iris	WBC	Wine
Clases	3	2	3
Muestras	[50,50,50]	[212,357]	[59,71,48]
Muestras en total	150	569	178
Atributos de entrada	4	30	13

Cuadro 5.1: Resumen de los datasets

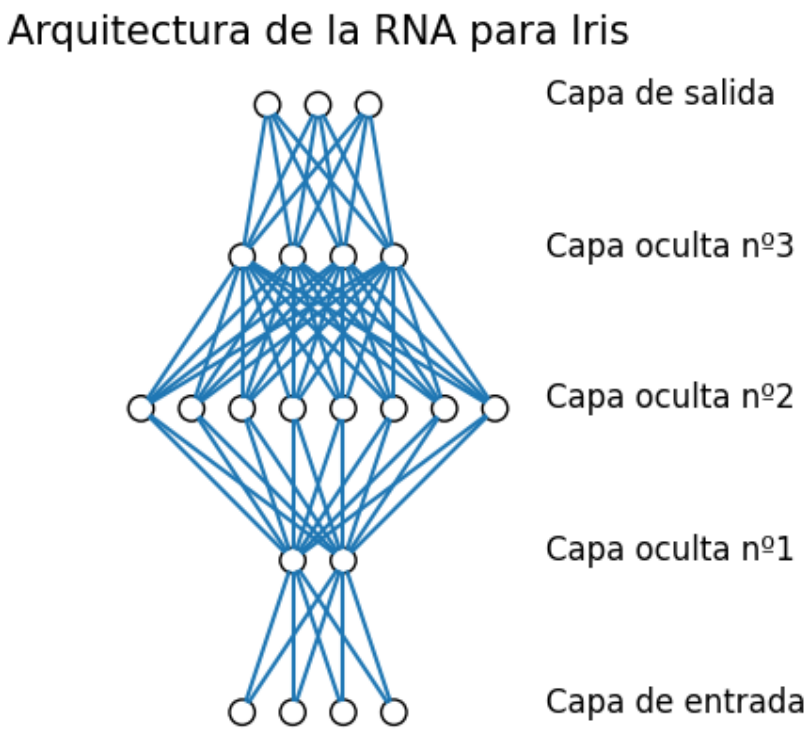

Figura 5.1: Arquitectura de la red neuronal utilizada para el dataset Iris

5.2. Redes neuronales entrenadas

Las redes que fueron entrenadas tenían la siguiente estructura: Las redes tienen 3 capas ocultas, la primera capa oculta tiene la mitad de neuronas que la capa de entrada, la segunda capa oculta tiene el doble de neuronas que la capa de entrada, y la última capa oculta tiene la misma cantidad. Con esto generamos redes diferentes para los distintos datasets, pero de una forma genérica. En las Figuras 5.1, 5.2 y 5.3 se pueden observar las arquitecturas de las redes neuronales utilizadas para los datasets Iris, WBC y Wine respectivamente.

Las arquitecturas son de la siguiente forma: Iris [2,8,4] WBC [15,60,30] y Wine [6,26,13]

A continuación se analizará una ejecución de la extracción con el conjunto de datos Iris para presentar de forma detallada el funcionamiento de FORxREN.

5.3. Ejecución detallada - Configuración 1 - Iris

El algoritmo propuesto comienza realizando el preprocesamiento de datos. Analiza la cantidad de elementos mal clasificados luego de apagar temporalmente cada neurona de entrada. Esto se puede ver en el Cuadro 5.2. El umbral que determina que neuronas se deben "apagar" es el valor mínimo de elementos mal clasificados (en este caso tiene el valor $\theta = 52$), por lo tanto, se agrega la neurona I_0 a B (el conjunto de neuronas insignificantes). La red sin

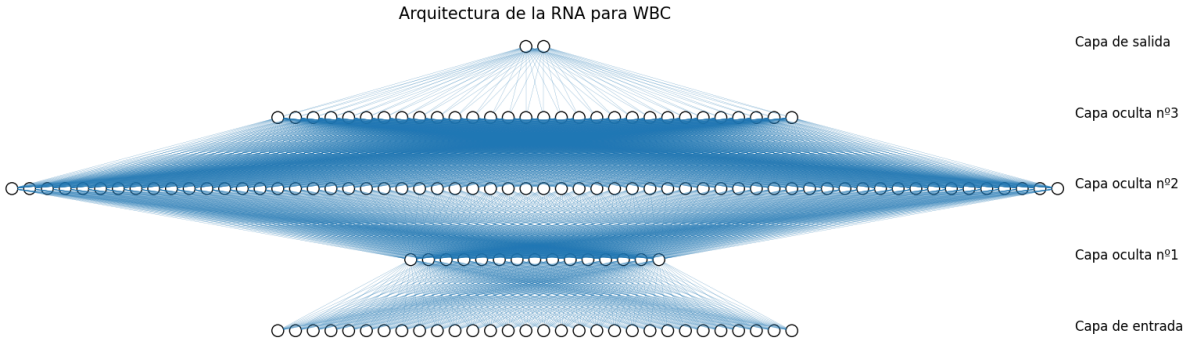

Figura 5.2: Arquitectura de la red neuronal utilizada para el dataset WBC

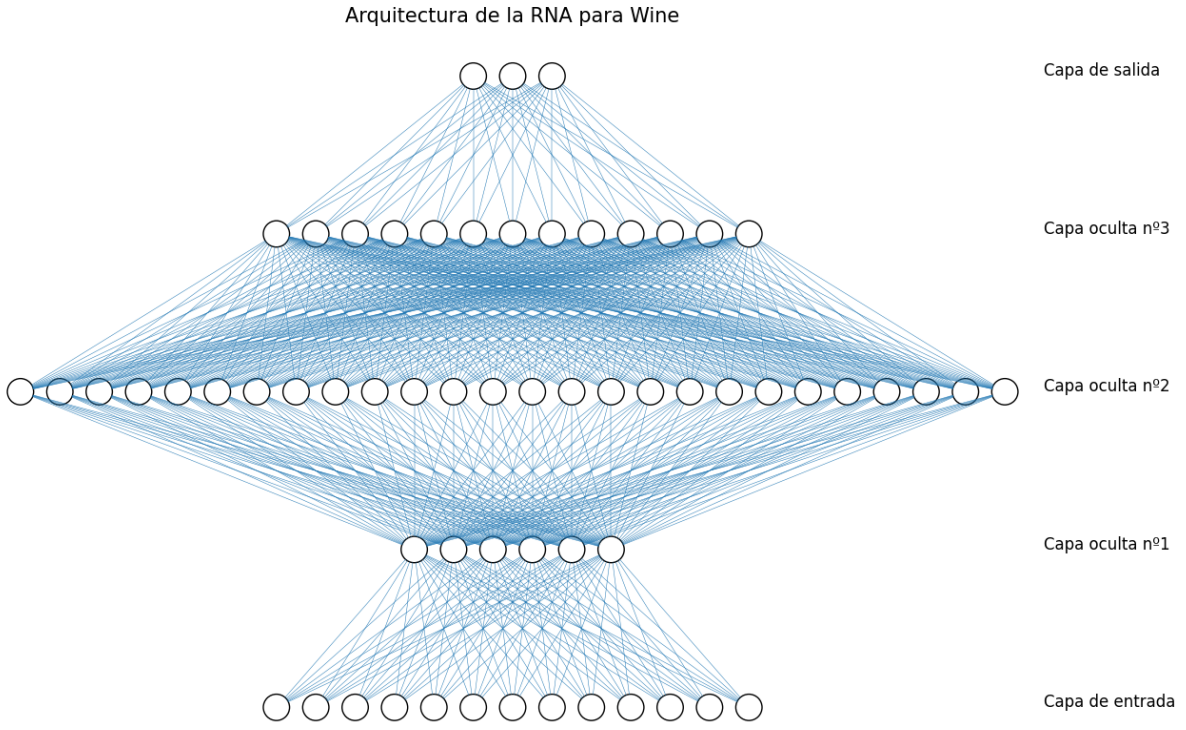

Figura 5.3: Arquitectura de la red neuronal utilizada para el dataset Wine

Cuadro 5.2: Elementos mal clasificados para cada neurona

Neurona	I_0	I_1	I_2	I_3
Cantidad de elementos	52	97	100	56

la neurona I_0 alcanza una fidelidad de 60 % por lo que I_0 no se apaga, ya que la pérdida de fidelidad supera el 5 % permitido.
El resto de neuronas no se intentará apagar, ya que claramente influyen mucho más al tener una mayor cantidad de elementos mal clasificados.
De la cantidad de elementos que son responsabilidad de la neurona, hay que determinar si hay suficientes ejemplos para cada clase como para considerar que la neurona influye en la clasificación. En el caso de que no tenga suficientes elementos, entonces no tiene la influencia necesaria para aportar a la clasificación de esa clase particular.
El siguiente paso es agrupar por clase los elementos mal clasificados por cada neurona de entrada I. Esto da como resultado:

- I_0: Cantidad de elementos de la clase 0 = 5 - clase 1 = 47 - clase 2 = 1
- I_1: Cantidad de elementos de la clase 0 = 55 - clase 1 = 94 - clase 2 = 0
- I_2: Cantidad de elementos de la clase 0 = 55 - clase 1 = 144 - clase 2 = 50
- I_3: Cantidad de elementos de la clase 0 = 55 - clase 1 = 150 - clase 2 = 100

Con esta información disponible y con $\alpha = 0{,}25$, por cada neurona de entrada I verificamos si la cantidad de elementos es mayor al umbral = $\alpha * err_i$ (cantidad de elementos mal clasificados con la neurona i apagada). Los umbrales obtenidos son los siguientes:

- I_0 el umbral = 13 (0.25 * 52)
- I_1 el umbral = 24,25 (0.25 * 97)
- I_2 el umbral = 25.0 (0.25 * 100)
- I_3 el umbral = 14 (0.25 * 56)

Superan el umbral las siguientes entradas:

- I_0 para la clase 1
- I_1 para las clases 0 y 1
- I_2 e I_3 para todas las clases

Al haber superado el umbral, se calculará para cada par entrada/clase el valor mínimo y máximo de los ejemplos del conjunto de datos en el atributo particular para guardar en la matriz. La matriz obtenida es la siguiente:

$$\begin{array}{ccc} None, & (4{,}9\ ;\ 7{,}0), & None \\ (2{,}3\ :\ 4{,}4), & (2{,}0\ :\ 3{,}4), & None \\ (1{,}0\ :\ 1{,}9), & (3{,}0\ ;\ 5{,}1), & (4{,}5\ ;\ 6{,}9) \\ (0{,}1\ :\ 0{,}6), & (1{,}0\ ;\ 1{,}8), & (1{,}4\ ;\ 2{,}5) \end{array}$$

Con la matriz se construyen las reglas if-then con los rangos obtenidos, ordenando las reglas por cantidad de condiciones, de forma decreciente. La última regla se escribe como else. Además, se nombran los inputs de acuerdo al dato que representan. Por ejemplo I_0 representa "$SepalLength$", I_1 "$SepalWidth$", I_2 "$PetalLength$" e I_3 "$PetalWidth$" Las reglas obtenidas de la matriz son:

- if((sepal length (cm) ≥ 4.9 ∧ sepal length (cm) ≤ 7.0) ∧ (sepal width (cm) 2.0 ∧ (sepal width (cm) ≤ 3.4) ∧ (petal length (cm) ≥ 3.0 ∧ petal length (cm) ≤ 5.1) ∧ (petal width (cm) ≥ 1.0 ∧ petal width (cm) ≤ 1.8))
  then Class = versicolor

- if((sepal width (cm) ≥ 2.3 ∧ sepal width (cm) ≤ 4.4) ∧ (petal length (cm) ≥ 1.0 ∧ petal length (cm) ≤ 1.9) ∧ (petal width (cm) ≥ 0.1 ∧ petal width (cm) ≤ 0.6) then Class = setosa

- Else Class = virginica

El siguiente paso es realizar la poda de las reglas. Para ello primero se calcula la fidelidad de las reglas hasta este punto que tiene un valor de 92.6 %. Luego, por cada regla construida se analizan sus condiciones y se determina cuáles deben ser podadas por su falta de influencia en el resultado. Por cada condición se la poda y se calcula nuevamente la fidelidad. Si se mantiene o aumenta, la condición se quita de la matriz. Si no, se intentará podar dicha condición, esta vez analizando sus dos partes, primero la parte izquierda, y luego la derecha, actualizando la matriz en el caso de que la fidelidad se mantenga o aumente.
Con la matriz previamente construida se explicará el proceso de poda:

- La posición [0,1] = (4.9 , 7.2) se poda y se mantiene la fidelidad. La condición se quita.

- La posición [1,1] = (2.0, 3.4) se poda y se mantiene la fidelidad. La condición se quita.

- La posición [2,1] = (3.0, 5.1) se poda y se pierde fidelidad, por lo que se intenta podar primero el lado izquierdo de la condición, la fidelidad se mantiene, por lo que se quita la condición, al intentar lo mismo con la condición de la derecha, se pierde fidelidad nuevamente, por lo que la condición se deja.

- La posición [3,1] = (1.0, 1.8) se poda y se pierde fidelidad, y al intentar podar en ambos lados por separado, también se pierde fidelidad, por lo que la condición entera permanece.

- La posición [1,0] = (2.3 , 4.4) se poda y se mantiene la fidelidad. La condición se quita.

- La posición [2,0] = (1.0, 1.9) se poda y se mantiene la fidelidad. La condición se quita.

- La posición [3,0] = (0.1, 0.6) se poda y se pierde fidelidad, por lo que se intenta podar solamente una parte, se comienza por la izquierda con el elemento "0.1" y se logra mantener la fidelidad, por lo que 0.1 se elimina de la condición, luego se intenta con el elemento derecho, pero la fidelidad baja a 64.67 % , por lo que no se quita esa condición.

- La posición [2,2] = (4.5 , 6.9) se poda y se mantiene la fidelidad. La condición se quita.

- La posición [3,2] = (1.4, 2.5) no se poda porque es la última condición presente para la clase 2.

- Las posiciones [0,0], [0,2], [1,2] = Vacía, por lo que no se analizan.

Reglas obtenidas luego de la poda:

- If (petal length (cm) ≤ 5.1 ∧ (petal width (cm) ≥ 1.0 ∧ petal width (cm) ≤ 1.8) Then Clase = versicolor

- If (petal width (cm) $\leq$ 0.6) Then Clase = setosa
- Else Clase = virginica

Por último se realiza el proceso de actualización donde se busca tener reglas más específicas sin perder fidelidad. Se busca, por cada posición de la matriz, nuevos valores mínimos y máximos. Si la fidelidad aumenta al reemplazar por estos valores, entonces la matriz se actualiza.
Los siguientes pasos son:

- Se reemplaza en la posición [3,2] el valor 2.5 por el nuevo valor 1.8, dando una fidelidad de 92.67 % por lo que no se actualiza, ya que se perdería generalidad innecesariamente.
- Se reemplaza en la posición [3,2] al valor 1.4 con el nuevo valor 1.5, dando una fidelidad de 92.67 % por lo que no se actualiza para no perder generalidad.

Luego de realizar la actualización de las reglas nos queda:

- If (petal length (cm) $\leq$ 5.1 $\wedge$ (petal width (cm) $\geq$ 1.0 $\wedge$ petal width (cm) $\leq$ 1.8) Then Clase = versicolor
- If (petal width (cm) $\leq$ 0.6) Then Clase = setosa
- Else Clase = virginica

Al final de la ejecución se obtuvieron reglas con una precisión de 94.67 % y una fidelidad de 92.67 %.

A continuación se muestran los resultados de las ejecuciones con todos los conjuntos de datos. En cada una de estas pruebas se registró la fidelidad y comprensibilidad de las reglas generadas, y la precisión original de la red neuronal artificial.

5.4. Resultados de las ejecuciones

Los resultados de las ejecuciones están en el Cuadro 5.3 para Iris donde la red neuronal tuvo una precisión del 93 %, en el Cuadro 5.4 para WBC con una red neuronal con 94.73 % de precisión y en los Cuadros 5.5 y 5.6 para Wine con la red neuronal que tuvo 91.01 % de precisión. En dichos cuadros se pueden apreciar las reglas resultantes de cada ejecución en sus distintas configuraciones y además la fidelidad de las reglas y su comprensibilidad.

Cuadro 5.3: Resultados de Iris

Paso	**Reglas**	**Fidelidad**	**Comprensibilidad**

Inicial	REGLA 1:
	If((sepal length (cm) ≥ 4.9 ∧ sepal length (cm) ≤ 7.0) ∧
	(sepal width (cm) 2.0 ∧ (sepal width (cm) ≤ 3.4) ∧
	(petal length (cm) ≥ 3.0 ∧ petal length (cm) ≤ 5.1) ∧
	(petal width (cm) ≥ 1.0 ∧ petal width (cm) ≤ 1.8))
	Then Class = versicolor
	
	REGLA 2:
	If((sepal width (cm) ≥ 2.3 ∧ sepal width (cm) ≤ 4.4) ∧
	(petal length (cm) ≥ 1.0 ∧ petal length (cm) ≤ 1.9) ∧
	(petal width (cm) ≥ 0.1 ∧ petal width (cm) ≤ 0.6))
	Then Class = setosa
	
	REGLA 3:
	Else Class = virginica
		92.67 %	45.97 %

Poda	REGLA 1:
	If (petal length (cm) ≤ 5.1 ∧
	(petal width (cm) ≥ 1.0 ∧ petal width (cm) ≤ 1.8)
	Then Class = versicolor
	
	REGLA 2:
	If(petal width (cm) ≤ 0.6) Then Class = setosa
	
	REGLA 3: Else Class = virginica
		92.67 %	79.44 %

Mejora	REGLA 1:
	If (petal length (cm) ≤ 5.1 ∧
	(petal width (cm) ≥ 1.0 ∧ petal width (cm) ≤ 1.8)
	Then Class = versicolor
	
	REGLA 2:
	If(petal width (cm) ≤ 0.6) Then Class = setosa
	
	REGLA 3:
	Else Class = virginica
		92.67 %	79.44 %

Cuadro 5.4: Resultados de WBC

Paso	**Reglas**	**Fidelidad**	**Comprensibilidad**
Inicial	REGLA 1: If((mean radius ≥10.95 ∧ mean radius ≤ 16.69) ∧ (mean perimeter ≥71.9 ∧ mean perimeter ≤ 109.8) ∧ (mean area ≥ 361.6 ∧ mean area ≤ 2499) ∧ (worst texture ≥ 18.47∧ worst texture ≤ 49.54) ∧ (worst perimeter ≥ 85.1 ∧ worst perimeter ≤ 188.5) ∧ (worst area ≥ 508.1 ∧ worst area ≤ 2499) ∧ Then Class = Malignant REGLA 2: Else Class = Benign	45.87 %	55.41 %
Poda	REGLA 1: If(worst texture ≥ 18.47∧ worst texture ≤ 85.1) Then Class = Malignant REGLA 2: Else Class = Benign	66.08 %	82.25 %
Mejora	REGLA 1: If (worst texture ≥ 18.47∧ worst texture ≤ 125) Then Class = Malignant REGLA 2: Else Class = Benign	90.69 %	82.25 %

Cuadro 5.5: Resultados de Wine - Reglas iniciales

Paso	**Reglas**	**Fidelidad**	**Comprensibilidad**

Inicial	REGLA 1:
	If((alcohol ≥ 11.61 ∧ alcohol ≤ 13.34) ∧
	(malic acid ≥ 0.94 ∧ malic acid ≤ 3.86) ∧
	(ash ≥ 1.7 ∧ ash ≤ 2.7) ∧
	(alcalinity of ash ≥ 15.0 ∧
	alcalinity of ash ≤ 30.0) ∧
	(magnesium ≥ 78.0 ∧ magnesium ≤ 162.0) ∧
	(total phenols ≥ 1.38 ∧ total phenols ≤ 3.52) ∧
	(flavanoids ≥ 0.57 ∧ flavanoids ≤ 5.08) ∧
	(color intensity ≥ 1.28 ∧ color intensity ≤ 6.0) ∧
	(od280/od315 of diluted wines ≥ 1.59 ∧
	od280/od315 of diluted wines ≤ 3.69) ∧
	(proline ≥ 278.0 ∧ proline ≤ 985.0))
	Then Class = 1
	
	REGLA 2:
	If((alcohol ≥ 12.88 ∧ alcohol ≤ 13.36) ∧
	(malic acid ≥ 1.24 ∧ malic acid ≤ 5.51) ∧
	(ash ≥ 2.15 ∧ ash ≤ 2.72) ∧
	(alcalinity of ash ≥ 17.5 ∧
	alcalinity of ash ≤ 27.0) ∧
	(magnesium ≥ 80.0 ∧ magnesium ≤ 123.0) ∧
	(total phenols ≥ 0.98 ∧ total phenols ≤ 2.32) ∧
	(flavanoids ≥ 0.34 ∧ flavanoids ≤ 1.57) ∧
	(color intensity ≥ 3.85 ∧ color intensity 13.0) ∧
	(od280od315 of diluted wines ≥ 1.27 ∧
	od280od315 of diluted wines ≤ 2.47) ∧
	(proline ≥ 415.0 ∧ proline ≤ 880.0))
	Then Class = 2
	
	REGLA 3:
	Else Class = 0
		84.83 %	37.66 %

Cuadro 5.6: Resultados de Wine - Poda y Mejora

Paso	**Reglas**	**Fidelidad**	**Comprensibilidad**

Poda	REGLA 1:
	If(magnesium ≥ 78.0 ∧ total phenols ≥ 1.38 ∧
	flavanoids ≥ 0.57 ∧
	color intensity ≤ 6.0 ∧ proline ≤ 985.0)
	Then Clase = 1
	
	REGLA 2:
	If(color intensity ≥ 3.85 ∧ proline ≤ 880.0)
	Then Class = 2
	
	REGLA 3:
	Else Class = 0
		90.45 %	56.66 %

Mejora	REGLA 1:
	If(magnesium ≥ 78.0 ∧ total phenols ≥ 1.38 ∧
	flavanoids ≥ 1.22 ∧
	color intensity ≤ 6.0 ∧ proline ≤ 795.0)
	Then Clase = 1
	
	REGLA 2:
	If(color intensity ≥ 3.85 ∧ proline ≤ 880.0)
	Then Class = 2
	
	REGLA 3:
	Else Class = 0
		90.45 %	56.66 %

Capítulo 6

Conclusiones

La capacidad de explicar a los demás los fundamentos de las propias decisiones es un aspecto importante de la inteligencia humana. Además, la explicación de las propias decisiones es a menudo un requisito para establecer una relación de confianza entre las personas. Aunque estos aspectos sociales pueden tener menos importancia para los sistemas técnicos de IA, hay muchos argumentos a favor de la explicabilidad en la inteligencia artificial. He aquí los más importantes:

- Como ya se ha mencionado, en muchas aplicaciones no se puede confiar ciegamente en un sistema de caja negra. Por ejemplo, en sanidad, el uso de modelos que puedan ser interpretados y verificados por expertos médicos es una necesidad absoluta.
- El primer paso para mejorar un sistema de IA es conocer sus puntos débiles. Obviamente, es más difícil realizar este análisis de debilidades en modelos de caja negra que en modelos interpretables. También es más fácil detectar sesgos en el modelo o en el conjunto de datos si se entiende lo que hace el modelo y por qué llega a sus predicciones. Incluso se puede afirmar que cuanto mejor entendamos lo que hacen nuestros modelos (y por qué a veces fallan), más fácil será mejorarlos.
- Como los sistemas de IA actuales se entrenan con millones de ejemplos, pueden observar patrones en los datos que no son accesibles a los humanos, que solo son capaces de aprender con un número limitado de ejemplos. Al utilizar sistemas de IA explicables, podemos intentar extraer este conocimiento destilado del sistema de IA para adquirir nuevos conocimientos.
- Los sistemas de IA afectan cada vez a más ámbitos de nuestra vida cotidiana. Junto a esto, los aspectos legales relacionados, como la asignación de responsabilidades cuando los sistemas toman una decisión equivocada, también han recibido recientemente una mayor atención. Dado que puede ser imposible encontrar respuestas satisfactorias a estas cuestiones jurídicas cuando se confía en modelos de caja negra, los futuros sistemas de IA tendrán necesariamente que ser más explicables. Las personas que se ven afectadas de forma inmediata por las decisiones de un sistema de IA (por ejemplo, las que son rechazadas por el banco para un préstamo) pueden querer saber por qué los sistemas han decidido de esa forma. Solo los sistemas de IA explicables proporcionarán esta información. La explicabilidad no solo tiene un interés académico importante y de actualidad, sino que desempeñará un papel fundamental en los futuros sistemas de IA.

El objetivo de esta tesina es profundizar en el estudio de las técnicas de extracción de reglas que permiten explicar las decisiones tomadas por las redes neuronales artificiales. Para realizar esto se tuvo que analizar la estructura y funcionamiento interno de las redes, en particular la influencia de cada neurona de entrada en la predicción.

Se comenzó la investigación estudiando y analizando diferentes técnicas de extracción de reglas como ECLAIRE, FERNN, DeepRED y finalmente RxREN donde se tuvo conocimiento de los distintos enfoques que se pueden tener a la hora de extraer las reglas para interpretar redes neuronales artificiales.

Tras haber llevado a cabo esta investigación, se reimplemento el método realizado por RxREN a partir de descripciones y pseudocódigo y luego se modificó el enfoque original, el cual se centraba en la precisión de las reglas obtenidas para lograr una clasificación correcta de los ejemplos, por uno donde se ha dado prioridad a la fidelidad de las reglas por encima de su precisión, incluso aunque esto implique que la clasificación no sea del todo exacta. Este nuevo algoritmo fue llamado FORxREN, que tiene como fin de lograr una mayor explicabilidad de la red neuronal, ya que las reglas fieles no solo explican los aciertos de la red, sino también sus desaciertos. Los resultados obtenidos han demostrado que FORxREN genera reglas con un alto porcentaje de fidelidad. Además, se ha observado que, fue posible aumentar la generalidad de las reglas (y, por lo tanto, su comprensibilidad) sin perder la fidelidad.
En resumen, esta tesina muestra que la orientación de FORxREN hacia la fidelidad de las reglas puede ser una estrategia efectiva para mejorar la explicabilidad de la red y generar reglas más fiables en determinadas aplicaciones.

Capítulo 7

Trabajos futuros

En base a la investigación realizada y a los resultados obtenidos se proponen las siguientes líneas de trabajo futuro:

- Investigar la aplicación del método propuesto en redes neuronales más complejas y de mayor escala para evaluar su escalabilidad y rendimiento.
- Realizar estudios empíricos para evaluar la eficacia del método en diferentes dominios y tareas, como la clasificación de datos no numéricos. Esto requiere expandir el algoritmo para que soporte este tipo de datos.
- Analizar el impacto de diferentes arquitecturas de red con el fin de investigar la relación entre la estructura de la red y la explicabilidad resultante de las técnicas de extracción.
- Investigar el potencial del método propuesto en combinación con otras técnicas de explicabilidad, como ECLAIRE, DeepRED y FERNN, para proporcionar una comprensión más completa y detallada del proceso de toma de decisiones en redes neuronales.
- Evaluar comprensibilidad de manera formal, teniendo en cuenta un enfoque mas social y centrado en el humano analizando características como claridad, longitud de la regla, cantidad de términos, comprensibilidad para un usuario no-técnico y la confianza en el sistema de caja negra luego de la explicación.

Apéndice A

Tecnologías utilizadas

A.1. Google Colab

Google Colab o también conocido como "Colaboratory" es un entorno interactivo diseñado por Google que permite programar y ejecutar Python en un navegador con las siguientes características:

- No requiere configuración
- Se puede acceder a GPUs sin coste adicional
- Permite compartir contenido fácilmente

Este entorno fue utilizado inicialmente para la codificación del trabajo realizado, desde el diseño y entrenamiento de las redes neuronales artificiales, hasta la ejecución de los algoritmos de extracción de reglas seleccionados y su comparación.

A.2. Sklearn

Scikit-learn también conocida como sklearn es una librería gratis de machine learning para Python. Su autor es David Cournapeau, y fue lanzada en junio de 2007. Contiene múltiples algoritmos de clasificación, regresión y clustering, incluyendo bosques aleatorios, k-medias, entre otros. Está diseñada para inter operar con las librerías numéricas y científicas NumPy y SciPy. Los datasets utilizados ya se encuentran en Sklearn y fueron descargados desde esta librería.

A.3. Keras

Keras es una API de aprendizaje profundo escrita en Python y capaz de ejecutarse sobre JAX, TensorFlow o PyTorch. En este trabajo fue usada con TensorFlow.

Keras es:

- Simple - pero no simplista. Keras reduce la carga cognitiva del desarrollador para que pueda centrarse en las partes del problema que realmente importan.
- Flexible - Keras adopta el principio de divulgación progresiva de la complejidad: los flujos de trabajo simples deben ser rápidos y fáciles, mientras que los flujos de trabajo

arbitrariamente avanzados deben ser posibles a través de un camino claro que se basa en lo que ya has aprendido.

- Potente - Keras proporciona rendimiento y escalabilidad a la altura de la industria: lo utilizan organizaciones como la NASA, YouTube o Waymo.

A.4. Numpy

NumPy es el paquete fundamental para la computación científica en Python. Es una biblioteca de Python que proporciona un objeto de matriz multidimensional, varios objetos derivados (como matrices y matrices enmascaradas) y una variedad de rutinas para realizar operaciones rápidas con matrices, incluidas operaciones matemáticas, lógicas, de manipulación de formas, ordenación, selección, E/S, transformadas discretas de Fourier, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más.

A.5. Pandas

Pandas es un paquete de Python que proporciona estructuras de datos rápidas, flexibles y expresivas diseñadas para que trabajar con datos relacionales.º .etiquetados"sea fácil e intuitivo. Pretende ser el bloque de construcción de alto nivel fundamental para realizar análisis de datos prácticos y reales en Python.

Pandas es adecuado para muchos tipos diferentes de datos:

- Datos tabulares con columnas de tipos heterogéneos, como en una tabla SQL u hoja de cálculo Excel.
- Datos de series temporales ordenados y desordenados (no necesariamente de frecuencia fija).
- Datos matriciales arbitrarios (de tipificación homogénea o heterogénea) con etiquetas de filas y columnas.
- Cualquier otra forma de conjunto de datos observacionales/estadísticos. No es necesario etiquetar los datos para colocarlos en una estructura de datos de pandas.

Referencias

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

Arras, L., Horn, F., Montavon, G., Müller, K.-R., y Samek, W. (2016). Explaining predictions of non-linear classifiers in nlp. *arXiv preprint arXiv:1606.07298*.

Arras, L., Montavon, G., Müller, K.-R., y Samek, W. (2017). Explaining recurrent neural network predictions in sentiment analysis. *arXiv preprint arXiv:1706.07206*.

Augasta, M. G., y Kathirvalavakumar, T. (2012). Reverse engineering the neural networks for rule extraction in classification problems. *Neural processing letters*, *35*, 131–150.

Averkin, A., y Yarushev, S. (2021). Review of research in the field of developing methods to extract rules from artificial neural networks. *Journal of Computer and Systems Sciences International*, *60*, 966–980.

Bondarenko, A., Zmanovska, T., y Borisov, A. (2010). Decompositional rules extraction methods from neural networks. En *Proc. of the 16th int. conf. on soft computing* (pp. 256–262).

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., y Elhadad, N. (2015). Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. En *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining* (pp. 1721–1730).

Chakraborty, M., Biswas, S. K., y Purkayastha, B. (2019). Rule extraction from neural network using input data ranges recursively. *New Generation Computing*, *37*, 67–96.

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

Etchells, T. A., y Lisboa, P. J. (2006). Orthogonal search-based rule extraction (osre) for trained neural networks: a practical and efficient approach. *IEEE transactions on neural networks*, *17*(2), 374–384.

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

Goodfellow, I., Bengio, Y., y cols. (2016). *a courville, a* (Vol. 1) (n.º 2). MIT press Cambridge.

Goodman, B., y Flaxman, S. (2017). European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, *38*(3), 50–57.

Guidotti, R., Monreale, A., Pedreschi, D., y Giannotti, F. (2021). Principles of explainable artificial intelligence. *Explainable AI Within the Digital Transformation and Cyber Physical Systems: XAI Methods and Applications*, 9–31.

Hayashi, Y. (2013). Neural data analysis: Ensemble neural network rule extraction approach and its theoretical and historical backgrounds. En *Artificial intelligence and soft computing: 12th international conference, icaisc 2013, zakopane, poland, june 9-13, 2013, proceedings, part i 12* (pp. 1–19).

Hruschka, E. R., y Ebecken, N. F. (2006). Extracting rules from multilayer perceptrons in classification problems: A clustering-based approach. *Neurocomputing*, *70*(1-3), 384–397.

Kahramanli, H., y Allahverdi, N. (2009). Rule extraction from trained adaptive neural networks using artificial immune systems. *Expert Systems with Applications*, *36*(2), 1513–1522.

Lapuschkin, S., Binder, A., Montavon, G., Muller, K.-R., y Samek, W. (2016). Analyzing classifiers: Fisher vectors and deep neural networks. En *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2912–2920).

Markowska-Kaczmar, U. (2008). Evolutionary approaches to rule extraction from neural networks. En *Engineering evolutionary intelligent systems* (pp. 177–209). Springer.

Moschettoni, M., Jacinto, M., Pérez, G., y Pons, C. (2023). Variante enfocada en fidelidad de un algoritmo de extracción de reglas en redes neuronales artificiales. *52 JAIIO Jornadas Argentinas de Informática. Sociedad Argentina de Informática (SADIO)*.

Russell, S., y Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson.

Setiono, R., Baesens, B., y Mues, C. (2008). Recursive neural network rule extraction for data with mixed attributes. *IEEE transactions on neural networks*, *19*(2), 299–307.

Setiono, R., y Leow, W. K. (2000). Fernn: An algorithm for fast extraction of rules from neural networks. *Applied Intelligence*, *12*, 15–25.

Taha, I. A., y Ghosh, J. (1999). Symbolic interpretation of artificial neural networks. *IEEE Transactions on knowledge and data engineering*, *11*(3), 448–463.

Zarlenga, M. E., Shams, Z., y Jamnik, M. (2021). Efficient decompositional rule extraction for deep neural networks. *arXiv preprint arXiv:2111.12628*.

Zilke, J. R., Loza Mencía, E., y Janssen, F. (2016). Deepred–rule extraction from deep neural networks. En *Discovery science: 19th international conference, ds 2016, bari, italy, october 19–21, 2016, proceedings 19* (pp. 457–473).