Variante enfocada en fidelidad de un algoritmo de extracción de reglas en redes neuronales artificiales

Milagros Jacinto[1], Martín Moschettoni[1], Claudia Pons[1,2,3], and Gabriela Pérez[1,4]

[1] Laboratorio de Investigación y Formación en Informática Avanzada (LIFIA), Facultad de Informática - UNLP, Buenos Aires - Argentina
https://lifia.info.unlp.edu.ar
{mjacinto, mmoschettoni, cpons, gperez}@lifia.info.unlp.edu.ar
[2] Centro de Altos Estudios en Tecnología Informática (CAETI), Facultad de Tecnología Informática - UAI, Buenos Aires - Argentina
https://caeti.uai.edu.ar/
[3] Comisión de Investigaciones Científicas (CIC), Gobierno de la Provincia de Buenos Aires - Argentina
https://www.cic.gba.gob.ar
[4] Instituto de Ingeniería y Agronomía, Universidad Nacional Arturo Jauretche (UNAJ), Buenos Aires - Argentina
https://www.unaj.edu.ar

Resumen Las redes neuronales tienen la capacidad de alcanzar altos niveles de precisión en tareas de clasificación, pero su falta de explicabilidad es un claro inconveniente y lleva a denominarlas "cajas negras". En este artículo, se presenta una modificación del algoritmo RxREN que se centra en la explicabilidad de las redes neuronales generando reglas precisas y fácilmente interpretables. El objetivo de esta modificación es comprender el proceso de decisión de la red neuronal, y para lograrlo, se analiza la relación entre el nivel de abstracción y su fidelidad. Se implementó un algoritmo con tres configuraciones en dos problemas distintos (Iris, WBC) Se analizó cómo el nivel de abstracción de las reglas afecta su fidelidad, buscando reglas precisas y evaluando el impacto del nivel de abstracción. En conclusión, este estudio tiene por objetivo mejorar significativamente la fidelidad de las reglas generadas por el algoritmo, permitiendo a los usuarios entender mejor el proceso de clasificación y además destacar la importancia de considerar el nivel de abstracción al extraer reglas interpretables y fieles.

Keywords: Fidelidad · Redes neuronales artificiales · Inteligencia artificial explicable · XAI

1. Introducción

La inteligencia artificial [1], y en particular las técnicas de aprendizaje profundo [2] usando redes neuronales artificiales, han progresado en las últimas décadas.

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 marcos de software como Caffe o TensorFlow fueron factores cruciales para este progreso. 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 resultados 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. En su lugar, cada decisión trascendental debería ser accesible para una validación adecuada por parte de un humano experto. También en los coches autónomos, donde una sola predicción incorrecta puede ser muy costosa, debe garantizarse la confianza del modelo en las características correctas. 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.
Este artículo propone una modificación del algoritmo de extracción de reglas denominado RxREN [13], cambiando su enfoque y modificando su proceso. La estructura del artículo es la siguiente: En la sección 2 se discute la relevancia de poder explicar las redes neuronales artificiales y se revisan investigaciones previas relacionadas con el trabajo propuesto. En la sección 3 se hace una breve introducción de la extracción de reglas y las métricas a utilizar. En la sección 4 se realiza un resumen del funcionamiento del algoritmo RxREN. En la sección 5 se explican los cambios realizados sobre el algoritmo. En la sección 6 se realiza una extracción de reglas de clasificación utilizando el algoritmo modificado para conjuntos de datos experimentales: Iris [4] y WBC (Winsconsin Breast Cancer) [5] y en la sección 7 se observan y analizan los resultados obtenidos y se realiza una conclusión del tema.

2. Importancia del problema y estado del arte

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 previo 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 [3] 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.

- 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 [6,7,8] 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. 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 [9] 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.

Existen varias propuestas para producir explicaciones para redes neuronales profundas. Uno de los enfoques más aceptados consiste en extraer reglas declarativas de la forma IF-THEN, u otro tipo de expresiones similares a enunciados de primer orden. [10,11,12,13,14]

3. Extracción de reglas en redes neuronales

La extracción de reglas en redes neuronales artificiales se puede definir de la siguiente manera [15]: 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.
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 decomposicionales entonces se lo llama un **acercamiento ecléctico**.

3.1. Métricas utilizadas

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 = $\{D_1, ...D_n\}$ un conjunto de ejemplos. Para cada ejemplo, D_i, donde

$D_i \subseteq D$, la fidelidad es "1" si para dicho ejemplo el conjunto de reglas y la red neuronal tienen como resultado la misma clasificación,

$$Fidelidad_i = \begin{cases} 1 & \text{If(reglas}(D_i) = \text{red neuronal}(D_i)) \\ 0 & \text{en caso contrario} \end{cases} \quad (1)$$

Para N ejemplos, la fidelidad está dada por:

$$Fidelidad = \frac{\sum_{i=1}^{N} Fidelidad_i}{N} \quad (2)$$

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

$$precisión = \frac{Cantidad\ de\ ejemplos\ correctamente\ clasificados}{Cantidad\ total\ de\ ejemplos}$$

- La comprensibilidad de las reglas se relaciona con la facilidad con que las personas puedan interpretarlas [13]. 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 comprensibilidad disminuye a medida que aumenta la cantidad de reglas y/o condiciones dentro de cada regla.

4. Algoritmo RxREN

RxREN es un algoritmo que utiliza un acercamiento pedagógico al proceso de extracción de reglas. Esto se lleva a cabo a partir del análisis de las entradas y las salidas de una red neuronal. Para hacerlo, se identifica las entradas de la red que no influyen en el resultado, analizando cada neurona de la capa de entrada y desactivando aquellas que no afectan la salida.

4.1. 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 la 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 posicion/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: $minimo \leq atributo \leq maximo \rightarrow Clase$. Se construye una regla if then compuesta por las condiciones obtenidas de la matriz, conectadas mediante conjunciones, formando la condición 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 podrá 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 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.

5. Modificación propuesta

Se ha propuesto 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.

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.

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.

5.1. Algoritmo RxREN modificado

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 version 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.
- C_k k-ésima clase objetivo.
- E_i Conjunto de ejemplos incorrectamente clasificados de la RNA sin I_i.
- err_i Cardinalidad de E_i
- q_{ik} Cantidad de ejemplos de E_i pertenecientes a la clase C_k
- RF Fidelidad de las reglas respecto a la red original.

Fase uno

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 $\theta = min(err_i), \mathbf{i} = 1...\mathbf{m}$
Paso 4: Formar el conjunto $B = \{I_i / err_i = \theta\}$, 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 C_k y encontrar el número de ejemplos q_{ik}, seleccionando solo las clases de E_i que satisfacen la condición $q_{ik} > \alpha \cdot err_i$, donde $\alpha \in [0{,}1; 0{,}5]$ y encontrar su valor mínimo, llamado MIN_{ik}, y su valor máximo, llamado MAX_{ik}, para construir las reglas.

Fase dos

Contrucción de la regla

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-énesima neurona de entrada I_i fuera seleccionada en el paso 8 para construir la regla, entonces $condicion_j = (data(I_i) \geq MIN_{ik} \wedge data(I_i) \leq MAX_{ik})$.
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: R_k = (If *condicion* then class = C_k)

Poda de reglas

Paso 15: Se calcula RF

Paso 16: Para cada regla construida R_k, 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 $data(I_i) \geq MIN_{ik}$) . Si $PF \geq RF$ entonces la regla quedará podada a izquierda y RF = PF.
3. En otro caso, se calcula PF con la $condicion_j$ podada a derecha (sin $data(I_i) \leq MAX_{ik}$) . 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 MIN_{ik} y MAX_{ik} de las condiciones correspondientes de las reglas.

Paso 20: Se modifican los límites inferior y superior de cada $condición_j$ 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.

5.2. 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 mas ajustados posibles que representan la red. Esto garantiza que las reglas sean lo más especificas 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.

6. Experimentos realizados

Se realizaron dos experimentos utilizando los siguientes conjuntos de datos para el algoritmo propuesto: iris y wbc. Las características de ambos pueden verse en el Cuadro 1.

6.1. Descripción de los datos

1. Conjunto de datos Iris: Este conjunto de datos consta de 150 ejemplos, cada uno con cuatro atributos continuos y tres clases objetivo. 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 consta de 699 ejemplos, cada uno con nueve atributos continuos y dos clases objetivo.

Cuadro 1. Propiedades de los conjuntos de datos.

	Iris	**WBC**
Ejemplos totales	150	699
Ejemplos de entrenamiento	75	349
Ejemplos de prueba	75	350
N° total de atributos	4	9
N° de atributos continuos	4	9
N° de atributos discretos	0	0
N° de clases	3	2

6.2. Red neuronal

Se crearon dos redes neuronales para cada uno de los conjuntos de datos, utilizando diferentes arquitecturas para cada uno. El primer conjunto de datos, iris, se utilizó una red neuronal feedforward con una arquitectura 5-3-3. Para el segundo conjunto de datos, el de cáncer de mama de Wisconsin, se utilizó una red neuronal feedforward con una arquitectura 10-3-2. En ambos casos el número de neuronas de entrada es igual al número de atributos de un conjunto de datos correspondiente y el número de neuronas de salida es igual al número de clases objetivo. Ambas redes se entrenaron utilizando el 80 % de tamaño total del conjunto de datos correspondiente y el 20 % restante para las pruebas. El entrenamiento se realizó utilizando un tamaño de lote de 16 y se completaron 200 épocas.

6.3. 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 2.

Cuadro 2. Elementos mal clasificados para cada neurona

Neurona	I_0	I_1	I_2	I_3
Cantidad de elementos	61	48	100	77

El umbral que determina que neuronas se deben "apagar" es el valor mínimo de elementos mal clasificados (en este caso tiene el valor $\theta = 48$), por lo tanto se agrega la neurona I_1 a B (el conjunto de neuronas insignificantes). La red sin la neurona I_1 alcanza una fidelidad de 63.33 % por lo que I_1 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 = 11 - clase 1 = 50 - clase 2 = 0
- I_1: Cantidad de elementos de la clase 0 = 0 - clase 1 = 48 - clase 2 = 0
- I_2: Cantidad de elementos de la clase 0 = 50 - clase 1 = 50 - clase 2 = 0
- I_3: Cantidad de elementos de la clase 0 = 0 - clase 1 = 27 - clase 2 = 50

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 = 15.25 (0.25 * 61)
- I_1 el umbral = 12.0 (0.25 * 48)
- I_2 el umbral = 25.0 (0.25 * 100)
- I_3 el umbral = 19.25 (0.25 * 77)

Superan el umbral las siguientes entradas:

- Todas las entradas para la clase 1
- I_2 e I_3 para la clase 2

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:

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

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. Las reglas obtenidas de la matriz son:

- if((data(I1) $\geq$ 4.9 $\wedge$ data(I1) $\leq$ 7.2) $\wedge$ (data(I2) $\geq$ 2.0 $\wedge$ data(I2) $\leq$ 3.4) $\wedge$ (data(I3) $\geq$ 3.0 $\wedge$ data(I3) $\leq$ 5.8) $\wedge$ (data(I4) $\geq$ 1.0 $\wedge$ data(I4) $\leq$ 1.5)) then Class = 1
- if((data(I3) $\geq$ 4.5 $\wedge$ data(I3) $\leq$ 6.9) $\wedge$ (data(I4) $\geq$ 1.4 $\wedge$ data(I4) $\leq$ 2.5)) then Class = 2
- Else Class = 0

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 96 %. 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 obtiene un 96 % de fidelidad. La condición se quita.
- La posición [1,1] = (2.0, 3.4) se poda y se obtiene un 96 % de fidelidad. La condición se quita.
- La posición [2,1] = (3.0 , 4.8) se poda y se obtiene un 88 % de fidelidad. La condición no se quita y se poda el valor 3.0 de la condición, obteniendo 96 % de fidelidad, por lo que se poda la condición a izquierda
- La posición [3,1] = (1.0 , 1.5) se poda y se obtiene 54.67 % de fidelidad. La condición no se quita y se prueba realizando la poda al valor 1.0, el cual da 62.67 % de fidelidad, por lo que no se quita. Luego se prueba realizando la poda al valor 1.5 el cual da 75.33 % de fidelidad por lo que se tampoco se quita.
- La posición [2,2] = (4.5 , 6.9) se poda y se obtiene un 96 % de fidelidad. La condición se quita.
- La posición [3,2] = (1.4 , 2.5) se poda y se obtiene un 64 % de fidelidad. La condición no se quita y se prueba realizando la poda al valor 1.4 el cual da

62.67 % de fidelidad, por lo que no se quita. Luego se prueba realizando la poda al valor 2.5 el cual da 96 % de fidelidad por lo que se quita.
- Las posiciones [0,0], [0,2], [1,0], [1,2], [2,0], [3,0] = Vacía, por lo que no se analizan.

Reglas obtenidas luego de la poda:

- If (data(I3) $\leq 5.8 \wedge$ data(I4) $\geq 1.0 \wedge$ data(I4) ≤ 1.5) Then Clase = 1
- If (data(I4) ≥ 1.4) Then Clase = 2
- Else Clase = 0

Por último se realiza el proceso de actualización donde se busca tener reglas más especificas 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 [2,1] el valor 5.8 por el nuevo valor 5.0, dando una fidelidad de 96 % por lo que no se actualiza, ya que se perdería generalidad innecesariamente.
- Se reemplaza en la posición [3,1] al valor 1.5 con el nuevo valor 1.6, dando una fidelidad de 66.67 % por lo que no se actualiza.
- Se reemplaza en la posición [3,2] el valor 1.4 con el nuevo valor 1.5, dando una fidelidad de 96 % por lo que no se actualiza.

Luego de realizar la actualización de las reglas nos queda:

- If (data(I3) $\leq 5.8 \wedge$ data(I4) $\geq 1.0 \wedge$ data(I4) ≤ 1.5) Then Clase = 1
- If (data(I4) ≥ 1.4) Then Clase = 2
- Else Clase = 0

Al final de la ejecución se obtuvieron reglas con una precisión de 94.67 % y una fidelidad de 96 %.

6.4. Resultados de ejecución

A continuación se encuentran los resultados de las ejecuciones con ambos conjuntos de datos: Iris en el Cuadro 3 y WBC en el Cuadro 4.

Cuadro 3. Resultados de Iris

Procesos	**Modo**	**Reglas**	**Fidelidad**	**Precisión**

Construcción	1,2,3	If((data(I1) $\geq$ 4.9 $\wedge$ data(I1) $\leq$ 7.2) $\wedge$
		(data(I2) $\geq$ 2.0 $\wedge$ data(I2) $\leq$ 3.4) $\wedge$
		(data(I3) $\geq$ 3.0 $\wedge$ data(I3) $\leq$ 5.8) $\wedge$
		(data(I4) $\geq$ 1.0 $\wedge$ data(I4) $\leq$ 1.5))
		then Class = 1
		If((data(I3) $\geq$ 4.5 $\wedge$ data(I3) $\leq$ 6.9) $\wedge$
		(data(I4) $\geq$ 1.4 $\wedge$ data(I4) $\leq$ 2.5))
		then Class = 2
		Else Class = 0
			96 %	94.67 %

Poda	1,2	If (data(I3) $\leq$ 5.8 $\wedge$ data(I4) $\geq$ 1.0
		$\wedge$ data(I4) $\leq$ 1.5)
		Then Clase = 1
		If(data(I4) $\geq$ 1.4) then Class = 2
		Else Class = 0
			96 %	94.67 %

Actualización	1	If (data(I3) $\leq$ 5.8 $\wedge$ data(I4) $\geq$ 1.0
		$\wedge$ data(I4) $\leq$ 1.5)
		Then Clase = 1
		If(data(I4) $\geq$ 1.4) then Class = 2
		Else Class = 0
			96 %	94.67 %

Cuadro 4. Resultados de WBC

Procesos	**Modo**	**Reglas**	**Fidelidad**	**Precisión**
Construcción	1,2,3	If((data(I1) $\geq$ 9.029 $\wedge$ data(I1) $\leq$ 16.3) $\wedge$ (data(I3) $\geq$ 59.82 $\wedge$ data(I3) $\leq$ 106.3) $\wedge$ (data(I4) $\geq$ 143.5 $\wedge$ data(I4) $\leq$ 819.8) $\wedge$ (data(I23) $\geq$ 54.49 $\wedge$ data(I23) $\leq$ 125.4) $\wedge$ (data(I24) $\geq$ 223.6 $\wedge$ data(I24) $\leq$ 947.9)) Then Class = 1 Else Class = 0	88.58 %	86.99 %
Poda	1,2	if(data(I23) $\leq$ 125.4 $\wedge$ data(I24) $\leq$ 947.9) Then Class = 1 Else Class = 0	92.27 %	90.69 %
Actualización	1	if(data(I23) $\leq$ 125.4 $\wedge$ data(I24) $\leq$ 947.9) Then Class = 1 Else Class = 0	92.27 %	90.69 %

7. Conclusión

Tras haber llevado a cabo esta investigación, se ha modificado el enfoque original de RxREN, el cual se centraba en la precisión de las reglas obtenidas para lograr una clasificación correcta de los ejemplos. En este estudio, 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. Esto se ha hecho con el fin de lograr una mayor explicabilidad de la red, ya que las reglas fieles no solo explican los aciertos de la red, sino también sus desaciertos. Los resultados obtenidos han demostrado que este nuevo enfoque ha permitido generar 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, este estudio muestra que la orientación del algoritmo 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.

Referencias

1. Russell, S., & Norvig, P. (2021). Artificial Intelligence A Modern Approach (4th Edition). In Pearson Series.
2. Goodfellow I., B. Y. (2016). Courville A-Deep learning-MIT (2016). Nature, 26(7553).
3. Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., & Elhadad, N. (2015). Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015 August.https://doi.org/10.1145/2783258.2788613
4. Andrews, D.F., Herzberg, A.M. (1985). Iris Data. In: Data. Springer Series in Statistics. Springer, New York, NY. https://doi.org/10.1007/978-1-4612-5098-2_2
5. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
6. Arras, L., Horn, F., Montavon, G., Müller, K.-R., & Samek, W. (2016). Explaining Predictions of Non-Linear Classifiers in NLP. https://doi.org/10.18653/v1/w16-1601
7. Arras, L., Montavon, G., Müller, K. R., & Samek, W. (2017). Explaining recurrent neural network predictions in sentiment analysis. EMNLP 2017 - 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA 2017 - Proceedings of the Workshop. https://doi.org/10.18653/v1/w17-5221
8. Lapuschkin, S., Binder, A., Montavon, G., Muller, K. R., & Samek, W. (2016). Analyzing Classifiers: Fisher Vectors and Deep Neural Networks. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-December. https://doi.org/10.1109/CVPR.2016.318
9. Goodman, B., & Flaxman, S. (2017). European union regulations on algorithmic decision making and a "right to explanation." AI Magazine, 38(3). https://doi.org/10.1609/aimag.v38i3.2741

10. Averkin, A. N., & Yarushev, S. A. (2021). Review of Research in the Field of Developing Methods to Extract Rules From Artificial Neural Networks. In Journal of Computer and Systems Sciences International (Vol. 60, Issue 6). https://doi.org/10.1134/S1064230721060046
11. Markowska-Kaczmar, U., & Wnuk-Lipiński, P. (2004). Rule extraction from neural network by genetic algorithm with Pareto optimization. Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science), 3070. https://doi.org/10.1007/978-3-540-24844-6_66
12. ÖzbakIr, L., Baykasoğlu, A., & Kulluk, S. (2008). Rule extraction from neural networks via ant colony algorithm for data mining applications. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5313 LNCS. https://doi.org/10.1007/978-3-540-92695-5_14
13. Gethsiyal Augasta, M., & Kathirvalavakumar, T. (2012). Reverse engineering the neural networks for rule extraction in classification problems. Neural Processing Letters, 35(2).https://doi.org/10.1007/s11063-011-9207-8
14. Santos, R. T., Nievola, J. C., & Freitas, A. A. (2000). Extracting comprehensible rules from neural networks via genetic algorithms. Proceedings of the 1st IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks. https://doi.org/10.1109/ECNN.2000.886228
15. Averkin, A.N., Yarushev, S.A. (2021). Review of Research in the Field of Developing Methods to Extract Rules From Artificial Neural Networks. J. Comput. Syst. Sci. Int. 60, 966–980. https://doi.org/10.1134/S1064230721060046