

Las matemáticas de una red neuronal

Adán Noel Duarte Candia

Universidad Nacional de Asunción

Facultad de Ciencias Exactas y Naturales

Departamento de Matemática

San Lorenzo - Paraguay

Diciembre - 2022

Índice

Objetivo General	2
Objetivos Específicos	2
1. Introducción	2
2. Motivación	3
3. Biología de una red neuronal	5
4. Elementos de una neurona artificial	7
5. Estructura de una red multicapa	8
6. El aprendizaje de la red neuronal artificial	9
7. La primera aparición de las redes neuronales	11
8. Conceptos previos	13
8.1. Descenso del Gradiente	13
8.2. Derivadas Parciales	16
8.3. Regla de la Cadena	17
8.4. Función Sigmoidal y Tangente Hiperbólica	18
9. El Perceptrón Multicapa	20
9.1. Nomenclaturas Necesarias para el Perceptrón	20
9.2. Comportamiento de una Neurona	21
10. Algoritmo de Retropropagación	22
10.1. Imputación de Valores perdidos, Acotación de las variables de entrada y Escalamiento . . .	29
11. Ejemplo de Aplicación de un Perceptrón Multicapa	31
Conclusión	33
Bibliografía	37
Anexo	38

Objetivo General

- Analizar los conceptos matemáticos que se requieren para entrenar una red neuronal.

Objetivos Específicos

- Describir cronológicamente el inicio y los distintos usos que se ha hecho de las redes neuronales artificiales
- Describir los distintos usos que se le da al modelo matemático de una red neuronal.
- Definir una red neuronal como modelo matemático.
- Describir las metodologías matemáticas utilizadas para obtener la representación del proceso de entrenamiento de una red neuronal.
- Estudiar la arquitectura del perceptrón multicapa.
- Describir el funcionamiento del método de propagación hacia atrás.
- Comparar el funcionamiento de un modelo obtenido con una red neuronal contra los obtenidos con una regresión logística.

1. Introducción

Hoy en día es muy habitual encontrarse con información relacionada con redes neuronales artificiales o simplemente redes neuronales. Abundan bibliografías sobre Inteligencia Artificial que si bien aparecen con distintos enfoques, todas tratan de explicar la similitud que tiene el cerebro humano con estos conceptos. De hecho la idea principal de la cual parte cada uno de los enfoques está inspirada en el aparato de comunicación neuronal de los humanos, es decir, está inspirado en el funcionamiento del cerebro humano. Nuestro cerebro es capaz de procesar información de manera mucha más eficiente que cualquier computadora convencional y por tanto tenemos la capacidad de reflexionar y aprender. Se han diseñado sistemas y estructuras que simulan el funcionamiento del cerebro humano con el objetivo de dotar a estos sistemas y estructuras de la capacidad de pensar por si mismos. Estos sistemas y estructuras pueden aparecer en diferentes campos tales como: Clasificación de Imágenes Satelitales: Determinar si la imagen corresponde a Tierra Roja, Cosecha de Algodón, Tierra Gris, Tierra Gris Húmeda o Tierra con vegetación. Procesamiento de lenguaje natural: se utiliza en la mayoría de los teléfonos inteligentes para predecir la próxima palabra en la escritura de un mensaje. Análisis de Riesgo Crediticio: para determinar cual es la probabilidad de que un crédito sea cancelado. Marketing: para determinar la siguiente mejor oferta. Economía: se utiliza para predecir la producción de la cosecha o el volumen de las ventas. Salud: se aplica en la detección precoz de enfermedades graves. Psicología: se aplica en terapias para parejas para calcular la probabilidad de divorcio. Problemas de Combinatoria: en este tipo de problemas la solución mediante cálculo tradicional requiere un tiempo de proceso (CPU) que es exponencial con el número de entradas. Un ejemplo es el problema del vendedor; el objetivo es elegir el camino más corto posible que debe realizar el vendedor para cubrir un número limitado

de ciudades en una área geográfica específica. Podríamos seguir mencionando muchas más aplicaciones pero dada la importancia que tiene las matemáticas que están detrás de las redes neuronales artificiales, en esta monografía se establecen las principales arquitecturas de una red neuronal y en particular se muestra con todos los detalles el proceso completo que se debe implementar cuando se decide entrenar una red neuronal con estructura de perceptrón multicapa.

2. Motivación

Supongamos que nos visita un amigo Psicólogo y nos cuenta que tiene un estudio hecho y que sirve para detectar si una pareja de matrimonios se va a divorciar o no próximamente, y nos muestra una especie de máquina con unas entradas, unas salidas y además con una serie de ruedas que parecen potenciómetros, similar a como se observa en la Figura (1). Nos enseña que esta máquina tal cual está, con las ruedas que

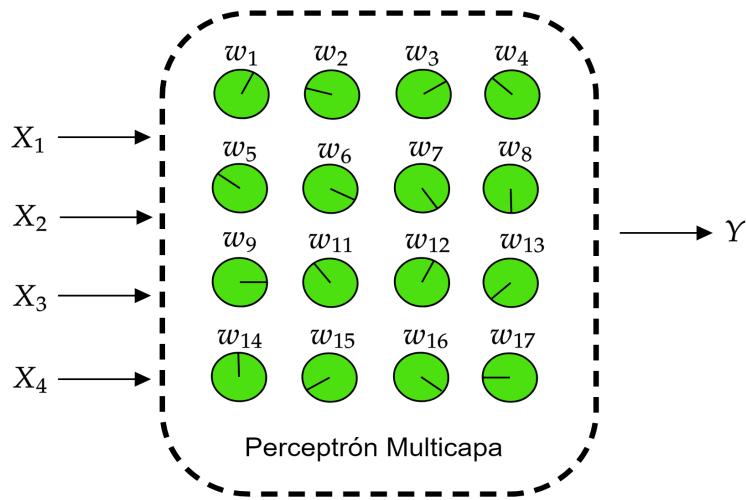


Figura 1: Perceptrón Multicapa

representan potencímetros, puestas en las diferentes posiciones, nos dice nuestro amigo Psicólogo con toda seguridad, que ha aprendido a detectar si una pareja de matrimonios se va a divorciar o no próximamente. Después de esta noticia, nosotros nos quedamos bastante sorprendidos y también con un montón de dudas sobre el funcionamiento de su máquina. Entonces él nos dice que ha utilizado un *Perceptrón Multicapa*, es decir, una especie de inteligencia artificial, que está metida en esa caja negra. Nuestro amigo continúa y nos cuenta que después de muchos casos de parejas de matrimonio que él estuvo estudiando, le dio la impresión de que los parámetros más importantes para que el divorcio ocurra es por ejemplo *El tipo de vivienda en el que vive la pareja*, es decir, si la vivienda es propia, alquilada o es de uno de los padres de la pareja, esto podría representar X_1 . Otras informaciones relevantes son: *el número de hijos que tiene la pareja* (X_2), *la edad del esposo* (X_3) y *la edad de la esposa* (X_4). Un estudio real sobre la detección temprana de un divorcio se puede leer en [Castrillón, 2021]. Después de esto le preguntamos a nuestro amigo, cómo es que él sabe que estas variables son relevantes para detectar si una pareja de matrimonio se va a divorciar, a lo que él nos

responde: Cuando queremos hacer un sistema de inteligencia artificial no podemos meter diez mil variables, en realidad hay que intentar meter una síntesis de las informaciones más relevantes para poder predecir, y es aquí donde está el arte, pero si no tenemos claro cuales son las variables mas relevantes lo que se hace es ir probando; por ahora, nos dice nuestro amigo, *creedme que estos parámetros de entrada funciona muy bien y que dejemos para otro momento la manera en que se elige estos parámetros*. Luego nos dice, este Perceptrón que he construido tiene solo una salida, ya que solo me interesa saber si la pareja se va divorciar o no por lo cual la variable $Y = 1$ puede representar *Divorcio* y la variable $Y = 0$ *No divorcio*. Después de esto nosotros seguimos perplejos porque no terminamos de entender que es lo que está pasando dentro de esa caja negra que nos está mostrando nuestro amigo Psicólogo y como es nuestro amigo le creemos por ahora que todo lo que nos dice es verdad. Por nuestra cara de preocupación, nuestro amigo, que es un gran Psicólogo nos dice: *No te preocupes, que cuando entremos en detalles podrás comprender con claridad lo que está pasando, por ahora solo debes creer que si en este sistema metemos $X_1 = \text{vivienda de los padres del novio}$, $X_2 = \text{dos hijos}$, $X_3 = 30$ y $X_4 = 25$ y resulta que $Y = 0.95$, esta pareja tiende claramente a divorciarse. Por el contrario, si resulta que $Y = 0.001$, entonces es muy probable que sean una pareja feliz y sigan en el matrimonio.* Todo esto nos genera bastante ansiedad y curiosidad por saber como funciona esto. Nuestro amigo nos dice que cuando le trajeron el Perceptrón, las manecillas de los potenciómetros estaban todos en la misma posición y lo que él hizo fue ir metiendo los parámetros de entrada y fue ajustando las manecillas hasta que llegó a comprobar que las posiciones actuales de las mismas son las que mejor predicen el divorcio.

Para ser mas preciso, nuestro amigo Psicólogo nos dice que él recopiló información en una tabla sobre cosas que ya sabe que acontecieron, tal como se observa en el Cuadro (1). En dicho cuadro tenemos varias filas que representan situaciones de parejas de matrimonio que ya pasaron, es por eso que aparece la columna S que representa la salida en la que terminó cada pareja.

X_1	X_2	X_3	X_4	S
Propia	6	45	50	0
Alquilada	2	30	29	1
Propia	0	40	38	0
:	:	:	:	:
:	:	:	:	:

Cuadro 1: Datos de Divorcios.

Nuestro amigo nos cuenta que tomaba cada fila del Cuadro (1) y los introducía en su caja negra, esta le arrojaba un resultado que comparaba contra lo que figuraba en la columna S para la fila en cuestión. Si la diferencia entre el valor observado y el valor que arroja el Perceptrón es grande, entonces ajustaba un poco los potenciómetros para que los valores que arroje la próxima vez sea igual o muy parecido al valor de la salida que encontramos en la columna S . Esto hacia con cada dato que disponía y el proceso era como una especie de entrenamiento o de aprendizaje de la caja negra. Entonces, nosotros pare aseguar que entendimos le preguntamos a nuestro amigo *Lo que me estas diciendo es: cada vez que introduces la información de una fila que aparece en el Cuadro (1), calculas la salida, en principio aleatoria, del Perceptrón; miras cuanto error se ha cometido y en base a eso modificas todos los potenciómetros para intentar minimizar ese error?*, a lo que él nos responde *eso mismo es lo que hago con todos los casos del Cuadro (1)*. Nuestro amigo nos

sigue contando que una vez que haya introducido todos los datos del Cuadro (1) al Perceptrón, obtiene una configuración de los potenciadores parecidas a las que se observa en la Figura (1). Entonces lo que se hace es volver a introducir los casos del Cuadro (1), unas cuantas veces más en el Perceptrón y esto es lo que los señores que se dedican a Inteligencia Artificial lo llaman Épocas.

Nuestro amigo Psicólogo nos dice que con unas pocas Épocas ya se llega a un resultado, a lo que nosotros preguntamos *¿Qué significa llegar a un resultado?* y él nos responde: *Significa que el Perceptrón a aprendido, es decir, que cuando volvamos a introducir los datos del Cuadro (1) el Perceptrón ya nos da como resultado una salida igual o muy parecida a la salida del Cuadro (1) y por tanto el error es prácticamente cero y eso implica que ya no necesitamos mover los potenciómetros.* Entonces nosotros preguntamos, y si viene un nuevo matrimonio, cuyos datos no estaban en el Cuadro (1) y que por ejemplo viven con los padres de la novia, no han tenido hijos, ella tiene 22 y el tiene 23, *¿Qué pasará?.* *¿Se van a divorciar o van a permanecer casados?.* Si los valores no estaban en el Cuadro (1), el Perceptrón va ser capaz de predecir correctamente?. A todo esto nuestro amigo nos responde que si los datos del Cuadro (1) son representativos y si no se ha equivocado en la elección de los parámetros de entrada X_1 , X_2 , X_3 y X_4 , efectivamente el Perceptrón será capaz de predecir correctamente lo que va pasar con este nuevo matrimonio. Nosotros estamos escuchando y muy asombrado, por lo que le preguntamos *¿Como has podido comprobado tal afirmación?* a lo que él nos dice *efectivamente lo he comprobado* y nos explica que lo que se hace aquí es un primer paso que es el *Aprendizaje*, que es todo lo que hasta ahora nos explicó, pero de todos los datos del Cuadro (1), él se ha dejado una parte y no lo usó en el paso de Aprendizaje del Perceptrón. Entonces es aquí donde se pasa a la etapa del proceso llamado validación. En esta etapa de validación se introduce en la caja negra los datos no utilizados para el aprendizaje y se compara la salida del Perceptrón con la salida esperada y efectivamente si el Perceptrón a aprendido el error será prácticamente cero en muchos de los casos, es decir, si por ejemplo introducimos 100 casos y de estos encontramos que en 80 casos el Perceptrón predice correctamente, entonces diremos que la caja negra tiene una certeza del 80 %.

A estas alturas ya no sabemos ni que pensar por lo que le preguntamos *¿Cuál es la regla o la formula que está dentro de esa caja negra para que tenga esta capacidad de predicción?* y a esto nuestro amigo Psicólogo nos responde: *Eso es lo malo del Perceptrón, no tenemos idea de cual es la ley o cual es la fórmula que está en la caja negra y que se usa para la predicción, pero las matemáticas nos pueden ayudar, y darnos un poco de luz en todo este tema.*

3. Biología de una red neuronal

Según [León and Viñuela, 2004] el aparato de comunicación de los animales y del hombre está formado por el sistema nervioso y hormonal que a su vez está conectado con los órganos de los sentidos y los órganos efectores (estos serían los músculos y las glándulas) y las mismas tienen la misión de recoger informaciones, transmitirlas y procesarlas, en partes también almacenarlas y enviarlas de nuevo pero en forma procesada.

Dicho sistema de comunicación se compone de tres partes:

- ❶ Los receptores que recogen la información en forma de estímulos ya sea del ambiente o del interior del organismo.
- ❷ El sistema nervioso, que recibe las informaciones, las procesa y luego una parte se guarda y otra parte se envía en forma procesada a los órganos efectores.

- ③ Los órganos efectores (músculos y glándulas), que reciben la información y tienen la capacidad de interpretarlas como acciones motoras, hormonales, etc.

El elemento más importante en el sistema de comunicación neuronal es la *NEURONA* y la misma cumple generalmente cinco funciones:

- ① Las neuronas recogen la información que les llega en forma de impulsos procedentes de otras neuronas o de receptores.
- ② La integran en un código de activación propio de la célula.
- ③ La transmite codificada en forma de impulso a través de su *axón*.
- ④ Mediante sus ramificaciones el axón distribuye los diferentes mensajes.
- ⑤ En sus terminales transmite los impulsos a las neuronas subsiguientes o a los órganos efectores

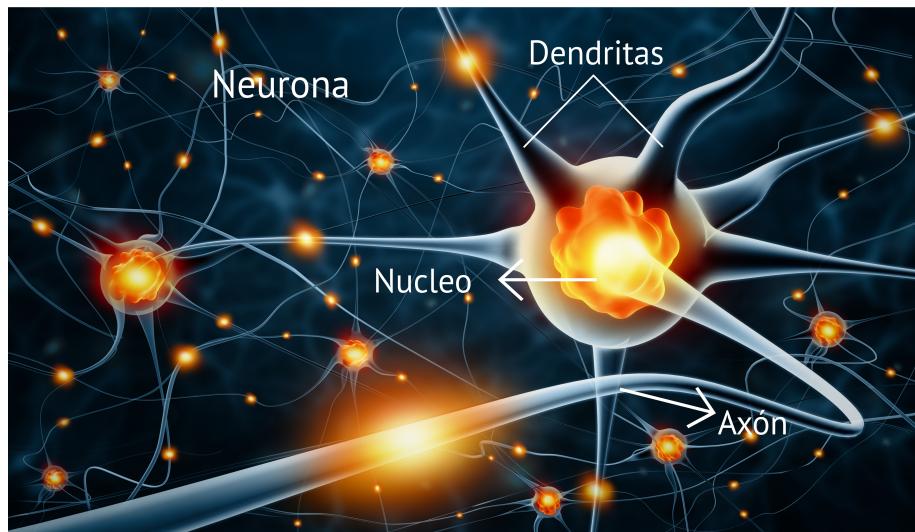


Figura 2: Nuerona típica

Un diagrama de una célula nerviosa típica se muestra en el Figura (2) y en dicho esquema se puede apreciar que la neurona consta de un cuerpo celular y de un núcleo, como todas las células del organismo, pero también aparecen algunos elementos específicos. El primero de ellos es el *axón*, que es una ramificación de salida de la neurona a través de la cual se propagan una serie de impulsos electro-químicos. Por otra parte, la neurona cuenta con un gran número de ramificaciones de entrada denominadas *Dendritas* cuya misión es propagar las señales al interior de la neurona.

El proceso general funciona de la siguiente manera: Las sinapsis recogen información electro-química que provienen de las células vecinas a la que la célula en cuestión está conectada; esta información llega al núcleo donde se procesa para generar una respuesta que es propagada por el axón. Luego, la señal propagada por el axón se ramifica y llega a dendritas de otras células a través de lo que se denomina sinapsis. Esta

sinapsis es el elemento de unión entre el axón y las dendritas. Es un espacio líquido que contiene ciertas concentraciones de elementos ionizados que la dotan con propiedades de conductividad y que la convierte en la responsable de activar o impedir el paso del impulso eléctrico.

Algunas de las células nerviosas reciben información directamente del exterior, estas células se denominan receptores, y la información recibida, estímulo. Los estímulos son el mecanismo de contacto de un organismo con el mundo exterior. Son estos estímulos los que básicamente ponen en funcionamiento la red neuronal del sistema nervioso, la cual propaga todas las señales hasta que llega a los órganos, glándulas y músculos, que son capaces de transformar la información recibida en acciones motoras, hormonales, etc. Es así como cualquier organismo recibe, procesa y actúa ante la información que llega del exterior.

Nuestro comportamiento (inteligente o no) sigue un esquema de este tipo, y son estos comportamientos lo que tratan de incorporar los modelos artificiales que han aparecido desde los inicios de la historia de las Redes Neuronales Artificiales.

4. Elementos de una neurona artificial

Pensemos que una neurona es un elemento con un estado interno, llamado nivel de activación, que recibe señales que le permiten cambiar de estado [León and Viñuela, 2004]. Supongamos que \mathcal{S} es el conjunto de estados posibles; el primer ejemplo podría ser el conjunto $\mathcal{S} = \{0, 1\}$ donde 1 representa el estado activo y 0 es estado inactivo, pero no existe ninguna restricción para imaginarnos que el conjunto \mathcal{S} puede estar dado por $\mathcal{S} = \{0, 1, 2, \dots, n\}$, donde los $n + 1$ valores pueden representar por ejemplo la escala de grises de una imagen, incluso el conjunto \mathcal{S} podría ser un intervalo tal como $\mathcal{S} = [0, 1]$ que puede representar cualquier variable continua.

Las neuronas tienen una función interna que les permite cambiar de estado, esta función interna se la conoce como función de activación. Para calcular el estado de activación de una neurona se ha de calcular en primer lugar la entrada total z_j a la célula, esta entrada total es la suma total de todas las entradas ponderadas por ciertos valores.

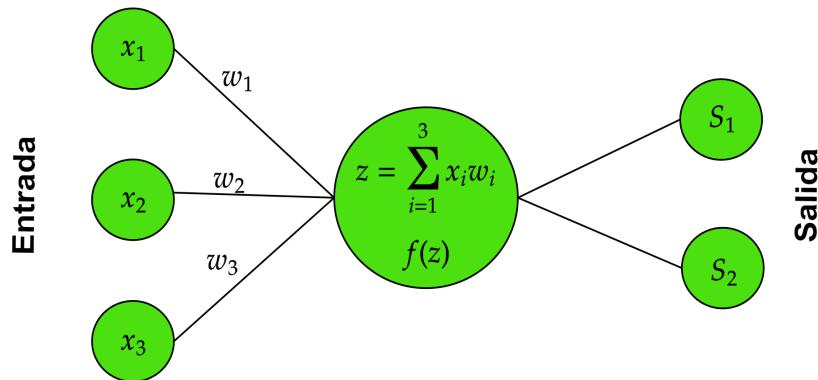


Figura 3: Esquema de una unidad de proceso

En la Figura (3) se muestra un esquema que representa la idea, en la misma tenemos tres elementos de

entrada y dos elementos de salida. Las estradas corresponden a las señales de la sinapsis de una neurona biológica. Cada señal se multiplica por un peso w asociado a cada entrada antes de ser aplicado al sumatorio. Cada peso representa la fuerza de la conexión sináptica, es decir, el nivel de concentración iónica de la sinapsis.

El sumatorio, corresponde al cuerpo de la neurona, suma todas las entradas ponderadas algebraicamente, produciendo una señal de salida z . Esta señal z es procesadas a su vez por una función de activación o función de salida f . En secciones siguientes se detallan los distintos tipos de funciones de activación y su importancia en una Red Neuronal Artificial.

5. Estructura de una red multicapa

En la Figura (3) se muestra un ejemplo de una unidad típica de proceso de una red neuronal artificial. A la izquierda tenemos una serie de tres entradas a la neurona, cada una de ellas puede llegar de una salida de otra neurona de la red. Una vez que se calcula la salida de una neurona, esta se propaga, vía conexiones de salida, a las células de destino.

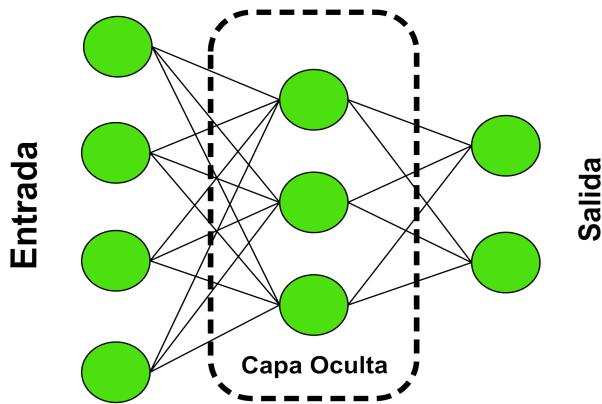


Figura 4: Esquema de una red multicapa

A la forma en que se conecta cada neurona entre si, se conoce como patrón de conectividad o arquitectura de la red. En la Figura (4) se muestra la estructura básica de interconexión entre células de una red multicapa.

El primer nivel representa las células de entrada; estas unidades reciben los valores de entrada a la red. A continuación se observa una serie de capas intermedias (las capas ocultas), cuyas unidades responden a rasgos particulares que pueden aparecer en los patrones de entrada. Puede haber más de un nivel en la capa oculta. El último nivel es la salida que sirve como salida de toda la red.

Cada interconexión entre unidades de proceso actúa como una vía de comunicación a través de las cuales viajan valores numéricos de una célula a otra.

Así pues, una Red Neuronal Artificial puede pensarse como un grafo cuyos nodos están constituidos por idénticas unidades de proceso que propagan información a través de los arcos. En dicho grafo se puede distinguir tres tipos de nodos: Nodos de entrada, Nodos de salida y Nodos de la capa oculta.

6. El aprendizaje de la red neuronal artificial

En una red neuronal artificial, el aprendizaje es la parte más importante, ya que asociado al esquema de este aprendizaje está el tipo de problemas que se puede resolver. Las redes neuronales artificiales son sistemas de aprendizaje que se basan en ejemplos conocidos. La capacidad de una red para resolver un problema es más o menos preciso según, el ejemplo que se disponga, sea mas o menos representativo del problema que se desea resolver. Considerando la importancia de los ejemplos en el proceso de aprendizaje de las redes neuronales artificiales, es conveniente que el conjunto de aprendizaje tenga las siguientes características:

- ❶ *Ser Significativos:* Esto implica tener un número suficiente de casos en el conjunto de datos de entrenamiento. Si dicho conjunto es reducido, la red neuronal artificial no será capaz de modificar adecuadamente sus pesos para representar la situación general del problema
- ❷ *Ser Representativos:* Los componentes del conjunto de datos de entrenamiento deben ser diversos. Por ejemplo, si estamos interesados en predecir el fraude a partir de transacciones con tarjetas de crédito es muy común que menos del 1 % de los casos de la muestra corresponda a fraude y en este caso el conjunto de aprendizaje tiene muchos más ejemplo de un caso que del otro y por tanto la red se especializará en la detección de casos legítimos y no en el fraude, es decir, nuestro modelo de red neuronal artificial no será de aplicación general. Es importante que todas las regiones significativas del espacio de estados estén suficientemente representadas en el conjunto de datos de entrenamiento. Una estrategia que se podría aplicar para el caso del fraude es redefinir el conjunto de entrenamiento incluyendo todos los casos de fraude y seleccionando aleatoriamente algunos casos de transacciones legítimas para tener un nuevo conjunto de datos de entrenamiento con igual cantidad de transacciones fraudulentas y transacciones legítimas.

El proceso de aprendizaje de una red neuronal artificial consiste en la determinación adecuada de los pesos w para cada una de las conexiones predefinidas y que capacite a la red en la solución eficiente del problema en cuestión. El proceso general de aprendizaje consiste en ir metiendo poco a poco en la red cada uno de los casos de nuestro conjunto de entrenamiento, y modificar los pesos de las conexiones siguiendo algún criterio. Cada vez que todo el conjunto de entrenamiento pasa por la red se denomina *Época*. Después de cada época se comprueba si se ha cumplido cierto criterio de convergencia; si no se cumple se repite todo el proceso introduciendo nuevamente poco a poco cada uno de los casos de nuestro conjunto de entrenamiento. El criterio de convergencia está asociado al tipo de red utilizada y al tipo de problema a resolver por lo que el periodo de aprendizaje puede terminar cuando:

- ❶ Se ejecutó un número de ciclos. Antes del proceso se decide cuantas épocas se van a ejecutar y una vez superado este número se detiene el proceso.
- ❷ Cuando el error sea menor que un valor predeterminado. Para que esto sea aplicable habrá que definir primeramente una función de error. Para este caso puede que la red nunca genere un valor de error que esté por debajo del valor predefinido por lo que suele combinarse con el caso anterior para que el proceso termine si no se consigue un error menor al valor definido después de un número finito de épocas. Si el proceso termina sin que se haya obtenido un error menor al predefinido se dice que la red diverge, es decir, la red no fue capaz de llegar a una solución y por tanto habrá que cambiar el valor del error exigido o el número de épocas a ejecutar.

- ③ Cuando la modificación de los pesos ya no afecte a la red. En algunos modelos se utiliza un esquema de aprendizaje que hace que las conexiones se vayan modificando cada vez menor intensidad. Si el proceso de aprendizaje continúa, llegará un momento en el que ya no habrá cambios en los valores de los pesos de las conexiones; en ese momento se detiene el proceso y se dice que la red ha convergido.

Según sea el problema a resolver habrá que elegir el esquema de aprendizaje de la red neuronal artificial. Hay tres tipos de esquemas:

- ① *Esquema supervisado*: En este tipo de aprendizaje, los datos del conjunto de entrenamiento tienen dos tipos de atributos: las variables independientes y la variable de respuesta o dependiente. Por ejemplo si se trata de definir un clasificador para un conjunto de datos, un sistema capaz de distinguir entre especies de una flor, los ejemplos contendrán datos de los rasgos morfológicos de la flor, e información de la solución, es decir, de qué especie se trata. El esquema supervisado utilizará esta información para cambiar los pesos de las conexiones. La forma habitual de corregir los pesos se puede ver en la Figura (5). Cada vez que un ejemplo se introduce en la red y se procesa para obtener una salida, esta última se compara con la salida deseada. Si la diferencia es grande, será grande la modificación de los pesos, mientras que si la salida se parece a la salida esperada, la modificación de los pesos será mucho menor.
- ② *Esquema no supervisado*: En este esquema los datos del conjunto de entrenamiento sólo tiene información de los ejemplos, y no hay nada que permita guiar en el proceso. La red neuronal modifica los valores de los pesos a partir de información interna. Cuando se utiliza un esquema de aprendizaje no supervisado la red trata de determinar rasgos significativos, regularidades o redundancias en los datos del conjunto de entrenamiento. A este tipo de modelos también se lo llama modelos autoorganizados, ya que la red se va ajustando dependiendo únicamente de los valores de entrada.
- ③ *Esquema mixto o por refuerzo*: Este esquema es una versión modificada del esquema supervisado en el que no se dispone de información concreta sobre el error que comete la red para cada ejemplo de entrada, si no que simplemente se determina si la red produce o no una salida adecuada.

Una característica particular que tiene el esquema mixto y que es importante mencionar es que el conjunto de datos de entrenamiento contiene las variables independientes como así también la variable respuesta y aunque en el proceso se van ajustando los pesos de la red para obtener la salida deseada, lo que realmente importa es que la red sea capaz de predecir, a partir de nuevos datos que se presenten en el futuro, valores de salida que se desconocen.

Supongamos que necesitamos predecir la dirección del frente principal de fuego de un incendio forestal y para el efecto disponemos de un conjunto de datos sobre incendios ocurridos, donde para cada registro y hora se dispone de información sobre humedad del ambiente, la dirección del viento, temperatura, el tipo de terreno, la intensidad del viento, el tipo de vegetación, etc que representan las variables independientes, y además, en qué dirección se desplazó el foco principal del incendio dos horas después (la variable de respuesta a predecir). Al entrenar una red con estos datos lo que realmente importa no es que ajuste adecuadamente los datos de entrenamiento para la salida conocida, sino que sea capaz de predecir donde va estar el frente principal de un incendio futuro dos horas después, y esto es algo que se desconoce por lo cual no se puede utilizar en el proceso de aprendizaje.

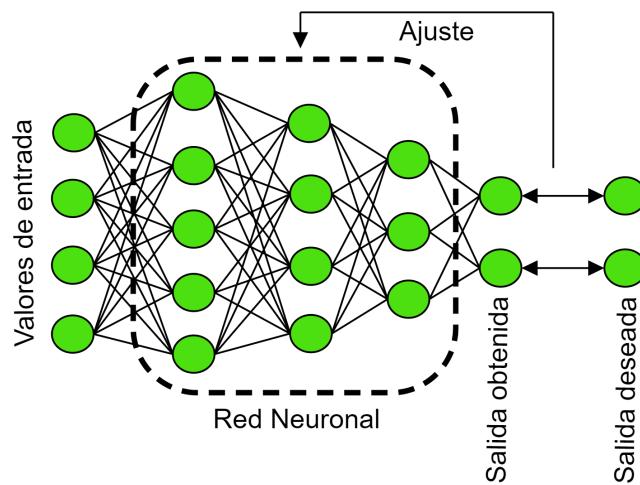


Figura 5: Esquema de aprendizaje supervisado

7. La primera aparición de las redes neuronales

La primera vez que parece una investigación sobre redes neuronales lo podemos encontrar en las obras [Freud and Strachey, 1964] correspondientes a las investigaciones de Sigmund Freud en el periodo del pre-sicoanálisis. La primera implementación de una Red Neuronal Artificial fue mediante un dispositivo hidráulico descripto por Russell [Russell et al., 2013]. Pero no fue hasta que se notara un gran avance de los hardware que cobran relevancia la redes neuronales. Obviamente hay que mencionar a los científicos que han hecho posible el avance de las redes neuronales y que se citan a continuación:

- **Warren McCulloch y Walter Pitts** [Fitch, 1944] hicieron el primer modelo matemático de una red neuronal artificial. Este modelo se basa en la idea de que las neuronas trabajan mediante impulsos binarios. Aunque el modelo generó un gran interés al brindar medidas de desempeño sofisticadas a través de cálculos sencillos, el factor clave fue la capacidad de aprendizaje.
- **Donald Hebb** mediante su libro [Shaw, 1986] describe un procedimiento matemático de aprendizaje para una red neuronal artificial conocida como *aprendizaje hebbiano*.
- **Marvin Minsky** fue el primero en obtener resultados prácticos en las redes neuronales[Minsky, 1954]. Diseñó una maquina con 40 neuronas cuyas conexiones se iban ajustando dependiendo de una serie de sucesos que ocurrían al realizar ciertas tareas; estos ajustes implementaban el aprendizaje hebbiano.
- **Albert Uttley** desarrolló una red neuronal artificial, creando una máquina teórica compuesta por informes (o elementos de proceso)[Uttley, 1956]. El informe era un separador lineal que ajustaba sus parámetros de entrada según la medida de entropía de Shannon.
- **Frank Rosenblatt** generalizó el modelo de McCulloch agregandole aprendizaje y llamó a este modelo el *PERCEPTRÓN* [Rosenblatt, 1957]. Este perceptrón era un modelo de dos niveles, que ajustaba los

pesos de las conexiones entre niveles en proporción al error entre la salida observada y la salida esperada. Muchos de los trabajos de Rosenblatt se pueden leer en [Rosenblatt, 1961].

- **Bernard Widrow** diseñó una red neuronal muy similar al perceptrón de Rosenblatt conocida como *Adaptive Linear Element* o simplemente *ADALINE* [Widrow, 1959, Widrow et al., 1960]. Esta red ajusta los pesos de las conexiones entre niveles en proporción al error entre la salida observada y la salida esperada. La diferencia con la red de Rosenblatt es muy pequeña pero los problemas a los que se aplican son muy diferentes.
- **Karl Steinbuch** es uno de los primeros investigadores en temas de desarrollo de métodos de codificación de información en Redes Neuronales. Las redes que diseñó se aplicaron específicamente a reconocimiento de escritura a mano distorsionada [Steinbuch, 1965].
- **Stephen Grossberg** es considerado el más formal e influyente investigador en Redes Neuronales Artificiales [Grossberg, 2012]. Los estudios de Grossberg incluyen rigurosos análisis matemáticos que permiten la realización de nuevos paradigmas de Redes Neuronales.
- **Shun-Ichi Amari** logró combinar la actividad de las redes neuronales biológicas con rigurosos modelos matemáticos de redes neuronales artificiales. Una de estas redes es solución de un famoso problema, que estuvo sin resolverse mucho tiempo, el de la asignación de créditos [Amari, 1971, Amari, 1972, Amari, 1974].
- **James Anderson** desarrolló un modelo de memoria asociativa lineal, siguiendo el planteamiento de Hebb. Empleó un nuevo método de corrección de error, y sustituyó la función umbral lineal por otra en rampa, creando así un nuevo modelo conocido como *Brain-state-in-a-box (BSB)* [Anderson and Murphy, 1986].
- **Kunihiko Fukushima** estudió redes neuronales artificiales con énfasis en modelos espaciales y espacio-temporales para sistemas de visión y el cerebro [Fukushima, 1969]. Su trabajo más notable ha sido la creación de un paradigma de Red Neuronal multicapa al que llamó *COGNITRON* [Fukushima, 1975].
- **A. Harry Klopff** se encargó del estudio de las relaciones entre la psicología de la mente y la biología del cerebro. Propuso que la neurona es un componente hedonístico del cerebro que se mueve por búsqueda de metas [Klopff and Gose, 1969].
- **Teuvo Kohonen** se encargó principalmente del estudio de redes neuronales artificiales con paradigmas de conexiones aleatorias [Kohonen, 1971]. Sus trabajos principales estaban centrados en memorias asociativas y matrices de correlaciones [Kohonen, 1972].
- **Terence Sejnowski** trabajó combinando modelos matemáticos y biológicos y una de sus más importantes contribuciones en este campo de investigación, junto a Geoffrey Hinton, es el descubrimiento del algoritmo de la máquina de Boltzmann [Hinton et al., 1984]. Así también, hizo contribuciones a la aplicación del algoritmo que se conoce como *Retroproyagación*.
- **McClelland y Rumelhart** son Psicólogos que se dedican a estudiar la utilización de modelos de redes neuronales artificiales para la ayuda a la comprensión de las funciones psicológicas de la

mente. David Rumelhart [Rumelhart, 1977] se interesó por las redes neuronales a través de sus trabajos *HEARSAY* de reconocimiento del lenguaje hablado. Por su lado, James McClelland comenzó sus investigaciones formulando un modelo semiparalelo de procesos mentales. Tiempo después, [McClelland and Rumelhart, 1981] ambos se unieron para elaborar un paradigma de reconocimiento de voz *Interactive Activation Model*.

- **Robert Hecht-Nielsen** se convirtió en el responsable principal del diseño de uno de los primeros computadores neuronales, dedicado al procesamiento de paradigmas de Redes Neuronales Artificiales. Este Neuro-computador, el *TRW MARK III*, está soportado por un computador VAX de *DIGITAL*, y ha estado comercialmente disponible en 1986.
- **John Hopfield** presentó en 1982 un método de análisis de estado estable en una red autoasociativa [Hopfield, 1982]. También introdujo una función de energía en sus análisis sobre Sistemas de Ecuaciones No Lineales y demostró que se puede construir una ecuación de energía que describa la actividad de una Red Neuronal de una capa, en tiempo discreto, y que dicha ecuación de energía puede disiparse llevando al sistema de ecuaciones a converger en un mínimo local. Con este avance resurge el interés por aplicar los paradigmas de Redes Neuronales Artificiales a problemas difíciles que los computadores convencionales no pueden resolver.

8. Conceptos previos

Para poder encarar la esquematización del proceso de aprendizaje de una red neuronal artificial necesitamos recordar algunos conceptos previos

8.1. Descenso del Gradiente

Sea $f : \mathbb{R}^n \rightarrow \mathbb{R}$ diferenciable. La derivada direccional de f en la dirección $d \in \mathbb{R}$ está dada por:

$$Df(x; d) = \nabla f(x)^t d$$

Para obtener una dirección de *máximo descenso* de la función f en un punto $x \in \mathbb{R}$ tal que $\nabla f(x) \neq 0$, se debe resolver el problema

$$\begin{aligned} &\min_{d \in \mathbb{R}^n} \nabla f(x)^t d \\ &\|d\|^2 = 1 \end{aligned}$$

La solución a este problema es:

$$d = -\frac{\nabla f(x)}{\|\nabla f(x)\|}$$

y por lo tanto la dirección de máximo descenso de la función es:

$$d = -\nabla f(x)$$

Supongamos por un momento que dado $f(\omega) = \omega^2 - 2\omega + 1$, debemos encontrar ω^* tal que $f(\omega^*) \leq f(\omega)$ para todo ω del dominio de $f(\omega)$. La gráfica de esta función lo podemos ver en la Figura (6)

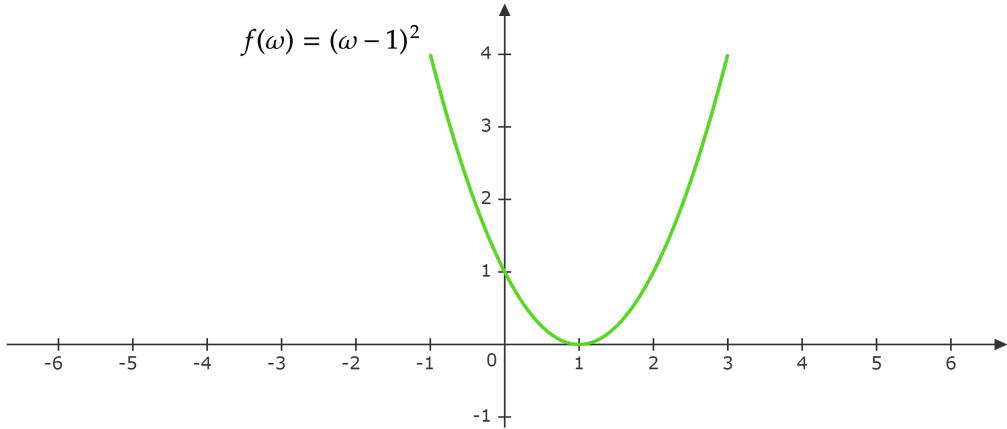


Figura 6: Función cuadrática

Sabemos que para encontrar ω^* que minimice la función $f(\omega)$ lo primero que tenemos que hacer es derivar dicha función respecto de ω y encontrar el cero de la derivada, es decir:

$$f'(\omega) = 2(\omega - 1) = 0 \iff \omega = 1$$

Y como $f''(1) = 2 > 0$ se concluye que $\omega^* = 1$ es el que minimiza la función. Ahora bien, el Descenso del Gradiente consiste en tomar un punto inicial arbitrario ω_0 y calcular sucesivamente

$$\begin{aligned} \omega_1 &= \omega_0 - f'(\omega_0) \\ \omega_2 &= \omega_1 - f'(\omega_1) \\ \omega_3 &= \omega_2 - f'(\omega_2) \\ \vdots &= \vdots \\ \omega_{n+1} &= \omega_n - f'(\omega_n) \end{aligned}$$

hasta llegar al valor que minimiza la función. Supongamos que tomamos $\omega_0 = 2$ entonces

$$\omega_1 = \omega_0 - f'(\omega_0) = \omega_0 - 2(\omega_0 - 1) = 2 - 2(2 - 1) = 0$$

y esto nos da $\omega_1 = 0$. Continuando para calcular ω_2 a partir de ω_1 tenemos que

$$\omega_2 = \omega_1 - f'(\omega_1) = \omega_1 - 2(\omega_1 - 1) = 0 - 2(0 - 1) = 2$$

y volvimos a nuestro punto inicial ya que $\omega_1 = \omega_0 = 2$. Claramente el método está oscilando entre los puntos 0 y 2 así como se muestra en la Figura (7).

Para evitar que las aproximaciones tengan este comportamiento el método del Descenso del Gradiente se generaliza incluyendo un factor α llamado *Razón de Aprendizaje*, que es en general, un número real arbitrariamente pequeño, y entonces las actualizaciones de las distintas aproximaciones nos queda como:

$$\omega_{n+1} = \omega_n - \alpha f'(\omega_n) \quad (1)$$

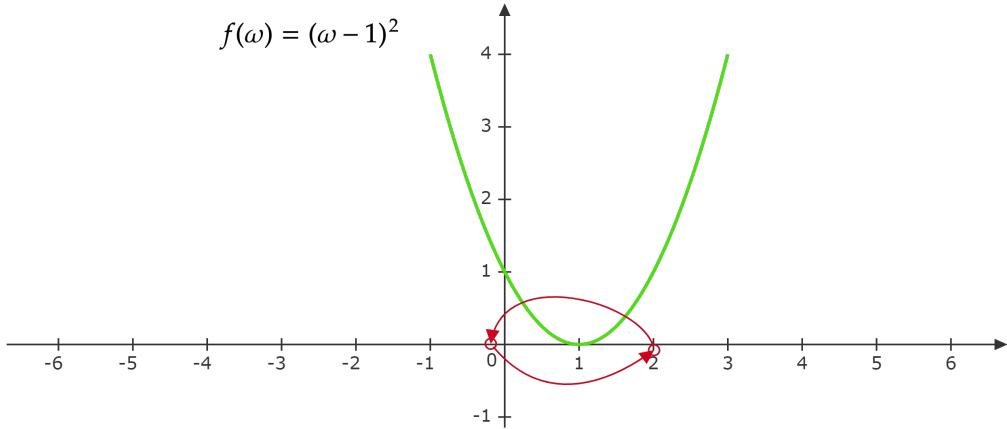


Figura 7: Oscilación de las aproximaciones

En el Cuadro (2) se puede observar el resultado de 10 iteraciones del método del Descenso del Gradiente con razón de aprendizaje $\alpha = 0.2$ cuando $f(\omega) = \omega^2 - 2\omega + 1$ y $\omega_0 = 2$.

$\omega_1 = 1.6$	$\omega_6 = 1.046656$
$\omega_2 = 1.36$	$\omega_7 = 1.027994$
$\omega_3 = 1.216$	$\omega_8 = 1.016796$
$\omega_4 = 1.1296$	$\omega_9 = 1.010078$
$\omega_5 = 1.07776$	$\omega_{10} = 1.006047$

Cuadro 2: Descenso del Gradiente con $\alpha = 0.2$ para $f(\omega) = \omega^2 - 2\omega + 1$ y $\omega_0 = 2$.

Nótese que al agregar la razón de aprendizaje al método, éste tiende al valor esperado, sin embargo, puede ser un gran desafío encontrar un valor razonable para α .

Consideremos ahora la función $f(\omega) = 0.1\omega^6 + 0.6\omega^5 - 0.7\omega^4 - 6\omega^3 + 2\omega^2 + 2\omega + 1$ cuya gráfica se puede ver en la Figura (8).

Visualmente se aprecian dos puntos donde la curva descende y vuelve a ascender (se han marcado con puntos en rojo), por supuesto, el segundo a la derecha es el mínimo absoluto, pero, ¿Qué pasaría si se hubiese hecho una búsqueda iniciando en $x = -6$? La respuesta es que el algoritmo se hubiese decantado por el mínimo de la izquierda. Con esto queremos resaltar que en general el método del Descenso del Gradiente converge a un mínimo local, o dicho de otro modo, converge al mínimo más cercano comparado con el ω_0 inicial. En el Cuadro (4) se puede observar como el método converge al mínimo local que está mas cerca del punto inicial ω_0 .

Es relativamente fácil darse cuenta donde está el mínimo absoluto de la función mirando la gráfica, pero el problema se acentúa cuando ya no se puede generar el gráfico o peor aún, cuando la función ya empieza a tener varias variables independientes.

$$f(\omega) = 0.1\omega^6 + 0.6\omega^5 - 0.7\omega^4 - 6\omega^3 + 2\omega^2 + 2\omega + 1$$

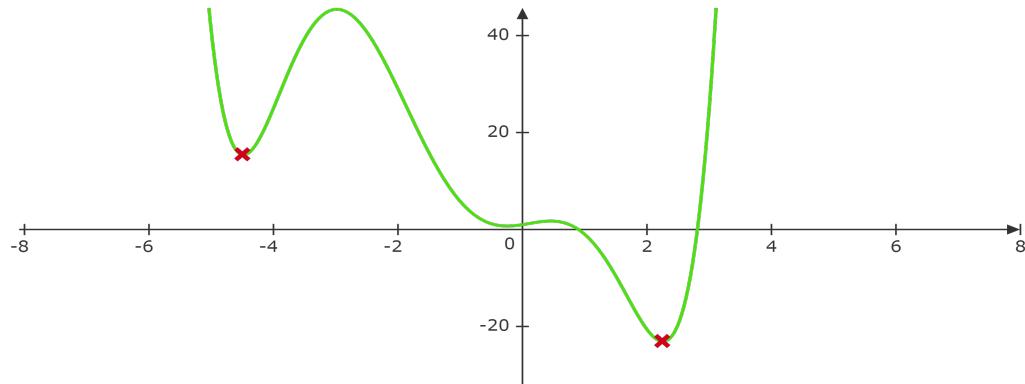


Figura 8: Función con más de un punto mínimo

$\omega_1 = -4.314400$	$\omega_6 = -4.433739$	$\omega_1 = 2.336600$	$\omega_6 = 2.317506$
$\omega_2 = -4.348565$	$\omega_7 = -4.445258$	$\omega_2 = 2.332317$	$\omega_7 = 2.314310$
$\omega_3 = -4.377047$	$\omega_8 = -4.454159$	$\omega_3 = 2.328281$	$\omega_8 = 2.311292$
$\omega_4 = -4.400312$	$\omega_9 = -4.460982$	$\omega_4 = 2.324478$	$\omega_9 = 2.308439$
$\omega_5 = -4.418982$	$\omega_{10} = -4.466179$	$\omega_5 = 2.320891$	$\omega_{10} = 2.305743$

Cuadro 3: Descenso del Gradiente con $\alpha = 0.002$ para $\omega_0 = -6$ y con $\alpha = 0.0005$ para $\omega_0 = 6$.

8.2. Derivadas Parciales

De acuerdo con [Leithold et al., 2001] la derivada parcial de una función de dos variables $f(x, y)$ con respecto a la variable y en el punto (a, b) y que se puede denotar como $\frac{\partial f}{\partial y}$, se obtiene al mantener fija la variable $x(x = a)$ y determinar la derivada ordinaria en $y = b$ de la función $G(y) = f(a, y)$. Más específicamente si f es una función de dos variables, sus derivadas parciales son las funciones definidas por:

$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \frac{f(x + h, y) - f(x, y)}{h}$$

$$\frac{\partial f}{\partial y} = \lim_{h \rightarrow 0} \frac{f(x, y + h) - f(x, y)}{h}$$

Existen muchas otras notaciones para las derivadas parciales si $z = f(x, y)$ y que se resume en el siguiente esquema:

$$f_x(x, y) = f_x = \frac{\partial f}{\partial x} = \frac{\partial}{\partial x} f(x, y) = \frac{\partial z}{\partial x} = f_1 = D_1 f = D_x f$$

$$f_y(x, y) = f_y = \frac{\partial f}{\partial y} = \frac{\partial}{\partial y} f(x, y) = \frac{\partial z}{\partial y} = f_2 = D_2 f = D_y f$$

Las derivadas parciales se pueden interpretar como *razones de cambio*, esto es, si $z = f(x, y)$, entonces $\frac{\partial z}{\partial x}$ representa la razón de cambio de z respecto a x cuando y permanece constante. Análogamente, $\frac{\partial z}{\partial y}$ representa la razón de cambio de z respecto a y cuando x permanece constante. Por ejemplo, para $f(x, y) = x^2 - 2y^2 - 4$ tenemos que la razón de cambio con respecto a x y con respecto a y están dadas por:

$$f_x(x, y) = 2x$$

$$f_y(x, y) = 2$$

En general, si u es una función de n variables $u(x_1, x_2, \dots, x_n)$, su derivada parcial con respecto a la i -ésima variable x_i es

$$\frac{\partial u}{\partial x_i} = \lim_{h \rightarrow 0} \frac{f(x_1, \dots, x_{i-1}, x_i + h, x_{i+1}, \dots, x_n) - f(x_1, \dots, x_i, \dots, x_n)}{h}$$

y también se puede escribir

$$\frac{\partial u}{\partial x_i} = \frac{\partial f}{\partial x_i} = f_{x_i} = f_i = D_i f$$

8.3. Regla de la Cadena

Supongamos que f y g son funciones derivables tales que $y = f(x)$ y $x = g(t)$, entonces podemos decir que y es una función derivable de t y su derivada es:

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

Para funciones de más de una variables se pueden enumerar tres casos

- ① Supongamos que $z = f(x, y)$ es una función derivable de x e y , donde $x = g(t)$ e $y = h(t)$ son funciones diferenciables de t . Entonces z es una función derivable de t y

$$\frac{dz}{dt} = \frac{\partial z}{\partial x} \frac{dx}{dt} + \frac{\partial z}{\partial y} \frac{dy}{dt} \quad (2)$$

- ② Supongamos que $z = f(x, y)$ es una función derivable de x e y , donde $x = g(s, t)$ e $y = h(s, t)$ son funciones derivables de s y t . Entonces

$$\frac{\partial z}{\partial s} = \frac{\partial z}{\partial x} \frac{\partial x}{\partial s} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial s} \quad \frac{\partial z}{\partial t} = \frac{\partial z}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial t} \quad (3)$$

- ③ Supongamos que u es una función derivable de n variables x_1, x_2, \dots, x_n y cada x_j es una función derivable de las m variables t_1, t_2, \dots, t_m . Entonces u es una función de t_1, t_2, \dots, t_m y

$$\frac{\partial u}{\partial t_i} = \frac{\partial u}{\partial x_1} \frac{\partial x_1}{\partial t_i} + \frac{\partial u}{\partial x_2} \frac{\partial x_2}{\partial t_i} + \dots + \frac{\partial u}{\partial x_n} \frac{\partial x_n}{\partial t_i}$$

para cada $i = 1, 2, \dots, m$.

Una manera de recordar la regla de la cadena es mediante un diagrama de árbol. En la Figura (9) se muestra el caso para la ecuaciones (3).

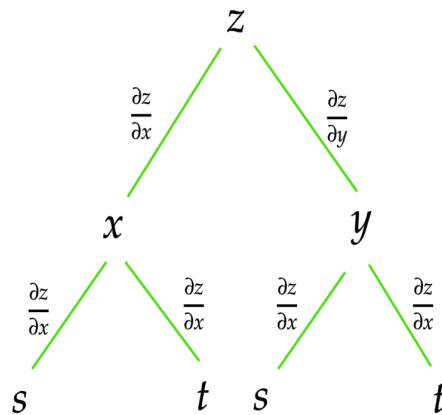


Figura 9: Regla de la cadena

Dibujamos ramas desde la variable dependiente z a las variables intermedias x e y para indicar que z es una función de x e y . Luego continuamos dibujando ramas desde x e y a las variables independientes s y t . En cada rama escribimos la derivada parcial correspondiente.

Para determinar $\frac{\partial z}{\partial s}$ calculamos el producto de las derivadas parciales en cada trayectoria desde z hasta s y luego sumamos los productos:

$$\frac{\partial z}{\partial s} = \frac{\partial z}{\partial x} \frac{\partial x}{\partial s} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial s}$$

Análogamente se determina $\frac{\partial z}{\partial t}$ mediante las trayectorias de z a t .

8.4. Función Sigmoidal y Tangente Hiperbólica

La función sigmoidal y la función tangente hiperbólica poseen como imagen un rango continuo de valores dentro de los intervalos $[0, 1]$ y $[-1, 1]$ respectivamente. Cada una de estas funciones vienen dadas por

- Función Sigmoidal

$$f(x) = \frac{1}{1 + e^{-x}}$$

■ Función Tangente Hiperbólica

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

Ambas son funciones crecientes y sus gráficas se pueden ver en la Figura (10)

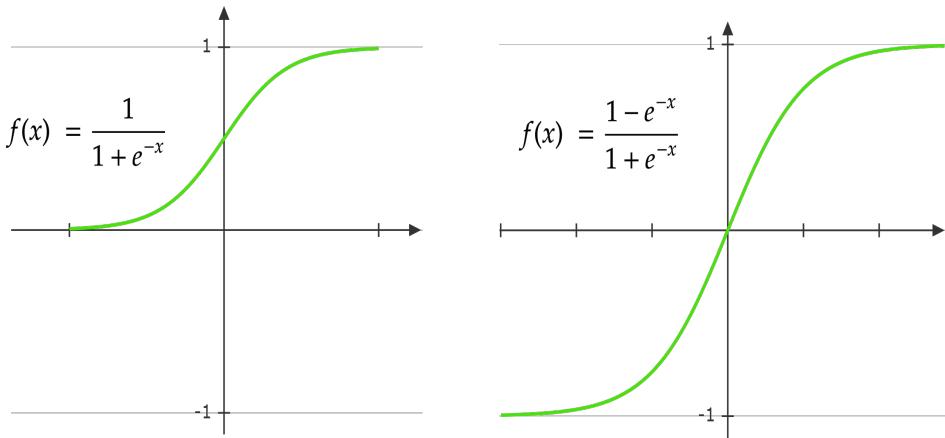


Figura 10: Sigmoidal - Tangente Hiperbólica

La primera derivada de la sigmoide está dada por

$$f'(x) = (1 + e^{-x})^{-2} e^{-x}$$

pero $f(x)^2 = (1 + e^{-x})^{-2}$ y $e^{-x} = 1/f(x) - 1$, por tanto

$$f'(x) = f(x)(1 - f(x)) \quad (4)$$

La función hiperbólica se puede reescribir como

$$g(x) = \frac{1 - e^{-x}}{1 + e^{-x}} = \frac{2 - 1 - e^{-x}}{1 + e^{-x}} = \frac{2}{1 + e^{-x}} - \frac{1 + e^{-x}}{1 + e^{-x}} = \frac{2}{1 + e^{-x}} - 1 = 2f(x) - 1$$

y por tanto la función hiperbólica se relaciona con la función sigmoide mediante la ecuación:

$$g(x) = 2f(x) - 1 \quad (5)$$

esto implica que

$$g'(x) = 2f'(x) = 2f(x)(1 - f(x)) = (g(x) + 1)(1 - g(x))/2$$

y por tanto

$$g'(x) = \frac{1 - g(x)^2}{2} \quad (6)$$

9. El Perceptrón Multicapa

En esta sección vamos a definir todas las nomenclaturas necesarias para describir matemáticamente el proceso de aprendizaje de una red neuronal artificial configurada como un perceptrón multicapa y para el efecto vamos a enfocarnos en un perceptrón con 3 entradas, dos capas ocultas con 4 neuronas y 2 salidas, tal cual se observa en la Figura (11)

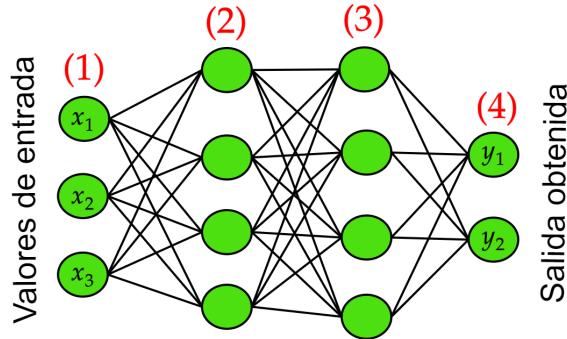


Figura 11: Perceptrón con dos capas ocultas

9.1. Nomenclaturas Necesarias para el Perceptrón

El perceptrón de la Figura (11) tiene la primera capa con tres entradas, dos capas ocultas con 4 neuronas y dos salidas. Estas cantidades lo podemos representar por $n_1 = 3$, $n_2 = 4$, $n_3 = 4$ y $n_4 = 2$, respectivamente. Los números entre paréntesis de la Figura (11) no ayudan a establecer las relaciones entre cada capa.

Comenzando con la primera capa, sabemos que estas se conectan a las neuronas de segunda capa mediante pesos que denotamos por ω . Más precisamente los pesos de la primera capa son:

$$\omega_{11}^{(1)}, \omega_{12}^{(1)}, \dots, \omega_{ij}^{(1)}, \quad \text{para } i = 1, 2, \dots, n_1 \text{ y } j = 1, 2, \dots, n_2$$

Análogamente podemos escribir el resto de los pesos que conectan con las neuronas de las otras capas y entonces tenemos:

$$\omega_{11}^{(2)}, \omega_{12}^{(2)}, \dots, \omega_{ij}^{(2)}, \quad \text{para } i = 1, 2, \dots, n_2 \text{ y } j = 1, 2, \dots, n_3$$

$$\omega_{11}^{(3)}, \omega_{12}^{(3)}, \dots, \omega_{ij}^{(3)}, \quad \text{para } i = 1, 2, \dots, n_3 \text{ y } j = 1, 2, \dots, n_4$$

En general, los pesos que conectan la capa k con la capa $k + 1$ están dados por:

$$\omega_{ij}^{(k)}, \quad \text{para } i = 1, 2, \dots, n_k \text{ y } j = 1, 2, \dots, n_{k+1} \tag{7}$$

9.2. Comportamiento de una Neurona

Si bien hemos visto de manera general como actúa una neurona esquematizada en la Figura (3), falta agregar un último condimento que completa el funcionamiento de una neurona. Este condimento es una entrada igual a 1 y que se asocia con un peso μ tal cual como se muestra en la Figura (12).

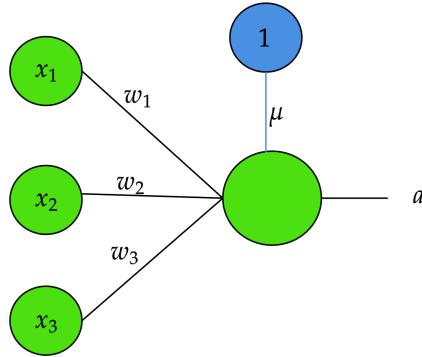


Figura 12: Funcionamiento de una neurona

El peso μ se conoce como *Umbral de activación* y nos permite completar la dinámica interna de la neurona. Otro elemento importante de dicha dinámica es la *función de activación*. Hay varias opciones para esta función pero las más comunes son la función Sigmoidal y la función Tangente Hiperbólica. Usar la una o la otra depende únicamente del rango de salida deseada. Si necesitamos una salida entre 0 y 1 pues se debe usar la función Sigmoidal.

Considerando que la neurona de la Figura (12) produce una salida a , la misma está dada por

$$a = f(\mu + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3),$$

donde f es la función de activación y la misma será la función Sigmoidal a menos que se indique explícitamente lo contrario. Supongamos que necesitamos calcular algunas derivadas parciales de a , para ello aplicamos la regla de la cadena y con eso obtenemos:

$$\begin{aligned} \frac{\partial a}{\partial \mu} &= \frac{\partial f}{\partial \mu} = f' \mu' = f(1 - f) = a(1 - a) \\ \frac{\partial a}{\partial \omega_1} &= \frac{\partial f}{\partial \omega_1} = f' \cdot (\omega_1 x_1)' = f(1 - f)x_1 = a(1 - a)x_1 \end{aligned}$$

Otra nomenclatura asociada a una red neuronal artificial como la que se ve en Figura (11) que vamos a usar es la que corresponde a las salidas de las neuronas. Si estamos en la capa de entrada, entonces $a_1^{(1)} = x_1$, $a_2^{(1)} = x_2$, $a_3^{(1)} = x_3$. En general, para las demás capas tenemos $a_i^{(k)}$ para $i = 1, 2, \dots, n_k$, pero notemos que

$$\begin{aligned} a_1^{(2)} &= f(\mu_1^{(2)} + a_1^{(1)} \omega_{11}^{(1)} + a_2^{(1)} \omega_{21}^{(1)} + a_3^{(1)} \omega_{31}^{(1)}) \\ a_i^{(2)} &= f(\mu_i^{(2)} + a_1^{(1)} \omega_{1i}^{(1)} + a_2^{(1)} \omega_{2i}^{(1)} + a_3^{(1)} \omega_{3i}^{(1)}) \quad \text{para } i = 1, 2, \dots, n_2 \end{aligned}$$

Y por tanto tenemos que

$$a_i^{(k)} = f\left(\mu_i^{(k)} + \sum_{j=1}^{n_{k-1}} a_j^{(k-1)} \omega_{ji}^{(k-1)}\right) \quad \text{para } i = 1, 2, \dots, n_k \quad (8)$$

10. Algoritmo de Retropropagación

El algoritmo de retropropagación o *Back Propagation* es uno de muchos algoritmos que se utiliza en la etapa de aprendizaje de una red neuronal artificial, es decir, se aplica el Back Propagation para estimar los pesos ω para las distintas conexiones entre neuronas de una red neuronal artificial. Consideremos la

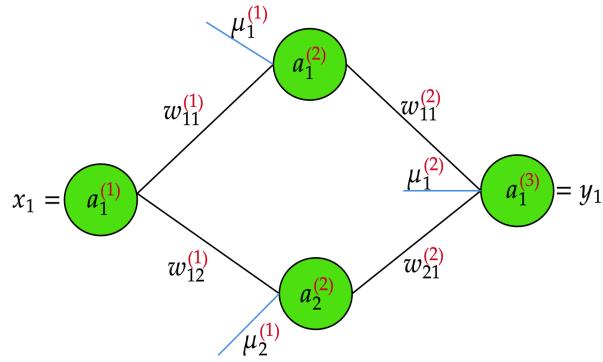


Figura 13: Back Propagation

red neuronal artificial de la Figura (13) con una capa de entrada, una capa oculta con dos neuronas y una capa de salida. Y arranquemos por la salida $a_1^{(3)}$ escribiéndolo en términos de la función de activación y las conexiones que le llegan, por tanto tenemos

$$\begin{aligned} a_1^{(3)} &= f\left(\mu_1^{(2)} + \omega_{11}^{(2)} a_1^{(2)} + \omega_{21}^{(2)} a_2^{(2)}\right) \quad \text{pero} \\ a_1^{(2)} &= f\left(\mu_1^{(1)} + \omega_{11}^{(1)} a_1^{(1)}\right) \quad \text{y} \quad a_2^{(2)} = f\left(\mu_2^{(1)} + \omega_{12}^{(1)} a_1^{(1)}\right) \end{aligned}$$

Lo anterior nos permite escribir $a_1^{(3)}$ como:

$$a_1^{(3)} = f\left(\mu_1^{(2)} + \omega_{11}^{(2)} f\left(\mu_1^{(1)} + \omega_{11}^{(1)} a_1^{(1)}\right) + \omega_{21}^{(2)} f\left(\mu_2^{(1)} + \omega_{12}^{(1)} a_1^{(1)}\right)\right)$$

Volviendo a los potenciómetros que aparecen en la Figura (1), recordemos que la idea es ajustar dichos potenciómetros para obtener la salida deseada. El Back Propagation utiliza el método del descenso del gradiente para actualizar los pesos ω que representan nuestros potenciómetros. Por tanto vamos a necesitar calcular algunas derivadas parciales. Primero vamos tomar la salida $a_1^{(3)}$ para calcular las derivadas parciales con respecto a cada uno de los pesos ω

$$\frac{\partial a_1^{(3)}}{\partial \omega_{11}^{(1)}} = a_1^{(3)} (1 - a_1^{(3)}) \omega_{11}^{(2)} a_1^{(2)} (1 - a_1^{(2)}) a_1^{(1)}$$

$$\frac{\partial a_1^{(3)}}{\partial \omega_{12}^{(1)}} = a_1^{(3)} (1 - a_1^{(3)}) \omega_{21}^{(2)} a_2^{(2)} (1 - a_2^{(2)}) a_1^{(1)}$$

Nótese que en las derivadas parciales anteriores hemos aplicado la regla de la cadena y la propiedad que se muestra en la ecuación (4) para la función Sísmoide. En resumen estas derivadas parciales nos queda:

$$\frac{\partial a_1^{(3)}}{\partial \omega_{1j}^{(1)}} = a_1^{(3)} (1 - a_1^{(3)}) \omega_{j1}^{(2)} a_j^{(2)} (1 - a_j^{(2)}) a_1^{(1)} \quad \text{para } j = 1, 2$$

Continuando con los otros pesos ω tenemos:

$$\frac{\partial a_1^{(3)}}{\partial \omega_{11}^{(2)}} = a_1^{(3)} (1 - a_1^{(3)}) a_1^{(2)}$$

$$\frac{\partial a_1^{(3)}}{\partial \omega_{21}^{(2)}} = a_1^{(3)} (1 - a_1^{(3)}) a_2^{(2)}$$

Y resumiendo nos queda

$$\frac{\partial a_1^{(3)}}{\partial \omega_{j1}^{(2)}} = a_1^{(3)} (1 - a_1^{(3)}) a_j^{(2)} \quad \text{para } j = 1, 2$$

Ahora toca el turno de las derivadas parciales con respecto a los umbrales μ :

$$\frac{\partial a_1^{(3)}}{\partial \mu_1^{(1)}} = a_1^{(3)} (1 - a_1^{(3)}) \omega_{11}^{(2)} a_1^{(2)} (1 - a_1^{(2)})$$

$$\frac{\partial a_1^{(3)}}{\partial \mu_2^{(1)}} = a_1^{(3)} (1 - a_1^{(3)}) \omega_{21}^{(2)} a_2^{(2)} (1 - a_2^{(2)})$$

$$\frac{\partial a_1^{(3)}}{\partial \mu_1^{(2)}} = a_1^{(3)} (1 - a_1^{(3)})$$

De esta manera tenemos todas las derivadas que vamos a necesitar más adelante.

Consideremos ahora la red neuronal artificial que se aprecia en la Figura (14). Usando dicha Figura podemos encontrar gráficamente las derivadas parciales con respecto a los pesos ω y los umbrales μ . Arranquemos por la derivada parcial de la salida $a_1^{(4)}$ con respecto al peso $\omega_{11}^{(1)}$; si partimos de $a_1^{(4)}$ y retrocedemos

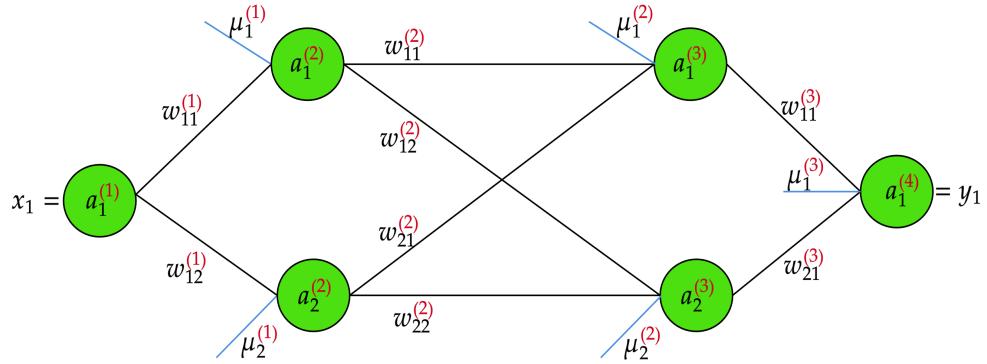


Figura 14: Back Propagation - Regla para las derivadas parciales

para pasar por $\omega_{11}^{(1)}$ (por eso el algoritmo se llama Retropropagación o Back Propagation) tenemos dos caminos posibles para hacerlo y los mismos se pintan en amarillo y lila en la Figura (15). Por cada camino vamos a tener una expresión, para el caso del camino amarillo tenemos:

$$\frac{\partial a_1^{(4)}}{\partial \omega_{11}^{(1)}} = a_1^{(4)} (1 - a_1^{(4)}) \omega_{11}^{(3)} a_1^{(3)} (1 - a_1^{(3)}) \omega_{11}^{(2)} a_1^{(2)} (1 - a_1^{(2)}) a_1^{(1)}$$

Lo que hicimos fue arrancar de $a_1^{(4)}$ y por tanto la expresión comienza con su derivada que es $a_1^{(4)} (1 - a_1^{(4)})$,

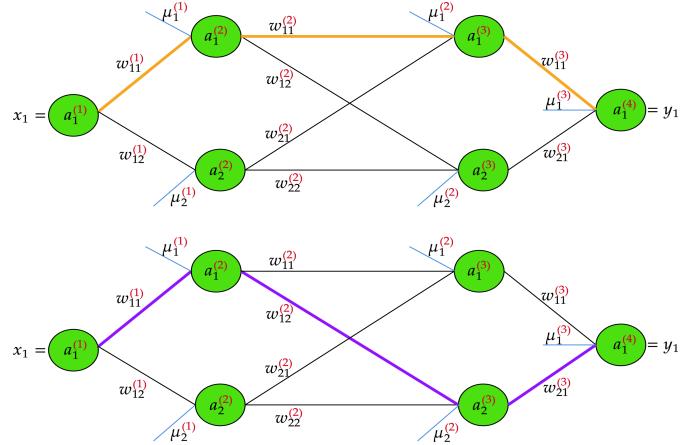


Figura 15: Back Propagation - Regla para las derivadas parciales

como estamos en el camino amarillo el siguiente que se encuentra es $\omega_{11}^{(3)}$ que multiplica la expresión anterior; ahora aparece $a_1^{(3)}$ y por tanto todo lo anterior se multiplica por su derivada $a_1^{(3)} (1 - a_1^{(3)})$; se sigue retrocediendo por el camino amarillo y nos encontramos con $\omega_{11}^{(2)}$ que van multiplicando a todo lo que ya

tenemos; ahora aparece $a_1^{(2)}$ y nuevamente usamos su derivada $a_1^{(2)}(1 - a_1^{(2)})$ para multiplicar a lo que se viene arrastrando; el siguiente que aparece es $\omega_{11}^{(1)}$ que se pasa de largo porque estamos calculando la derivada con respecto a este; finalmente se llega a $a_1^{(1)}$ que es el ultimo factor en la multiplicación. Para obtener la fórmula completa se debe recorrer también al camino en lila y sumar a la expresión encontrada por el camino amarillo. Esto nos permite escribir que:

$$\begin{aligned} \frac{\partial a_1^{(4)}}{\partial \omega_{11}^{(1)}} &= a_1^{(4)}(1 - a_1^{(4)})\omega_{11}^{(3)}a_1^{(3)}(1 - a_1^{(3)})\omega_{11}^{(2)}a_1^{(2)}(1 - a_1^{(2)})a_1^{(1)} + \\ &\quad a_1^{(4)}(1 - a_1^{(4)})\omega_{21}^{(3)}a_2^{(3)}(1 - a_2^{(3)})\omega_{12}^{(2)}a_1^{(2)}(1 - a_1^{(2)})a_1^{(1)} \end{aligned} \quad (9)$$

Análogamente podemos obtener la derivada parcial de la salida $a_1^{(4)}$ con respecto al peso $\omega_{12}^{(1)}$

$$\begin{aligned} \frac{\partial a_1^{(4)}}{\partial \omega_{12}^{(1)}} &= a_1^{(4)}(1 - a_1^{(4)})\omega_{21}^{(3)}a_2^{(3)}(1 - a_2^{(3)})\omega_{22}^{(2)}a_2^{(2)}(1 - a_2^{(2)})a_1^{(1)} + \\ &\quad a_1^{(4)}(1 - a_1^{(4)})\omega_{11}^{(3)}a_1^{(3)}(1 - a_1^{(3)})\omega_{21}^{(2)}a_2^{(2)}(1 - a_2^{(2)})a_1^{(1)} \end{aligned} \quad (10)$$

Ahora bien, para generalizar las derivadas parciales con respecto a los pesos ω vamos a reescribir las ecuaciones (9) y (10) considerando que la salida $a_1^{(4)} = y_1$, entrada $a_1^{(1)} = x_1$ y cambiando el orden de los factores de cada sumando, esto es:

$$\begin{aligned} \frac{\partial y_1}{\partial \omega_{11}^{(1)}} &= x_1 a_1^{(2)}(1 - a_1^{(2)})\omega_{11}^{(2)}a_1^{(3)}(1 - a_1^{(3)})\omega_{11}^{(3)}y_1(1 - y_1) + \\ &\quad x_1 a_1^{(2)}(1 - a_1^{(2)})\omega_{12}^{(2)}a_2^{(3)}(1 - a_2^{(3)})\omega_{21}^{(3)}y_1(1 - y_1) \end{aligned}$$

$$\begin{aligned} \frac{\partial y_1}{\partial \omega_{12}^{(1)}} &= x_1 a_2^{(2)}(1 - a_2^{(2)})\omega_{22}^{(2)}a_2^{(3)}(1 - a_2^{(3)})\omega_{21}^{(3)}y_1(1 - y_1) + \\ &\quad x_1 a_2^{(2)}(1 - a_2^{(2)})\omega_{21}^{(2)}a_1^{(3)}(1 - a_1^{(3)})\omega_{11}^{(3)}y_1(1 - y_1) \end{aligned}$$

Resumiendo un poco esa suma nos queda:

$$\frac{\partial y_1}{\partial \omega_{11}^{(1)}} = x_1 a_1^{(2)}(1 - a_1^{(2)}) \left[\sum_{j=1}^2 \omega_{1j}^{(2)}a_j^{(3)}(1 - a_j^{(3)})\omega_{j1}^{(3)} \right] y_1(1 - y_1)$$

$$\frac{\partial y_1}{\partial \omega_{12}^{(1)}} = x_1 a_2^{(2)}(1 - a_2^{(2)}) \left[\sum_{j=1}^2 \omega_{2j}^{(2)}a_j^{(3)}(1 - a_j^{(3)})\omega_{j1}^{(3)} \right] y_1(1 - y_1)$$

Así quedaron las fórmulas que permiten calcular la derivada parcial de la salida $a_1^{(4)}$ con respecto a los pesos $\omega^{(1)}$. Supongamos ahora que necesitamos calcular las derivadas parciales con respecto a los pesos $\omega^{(2)}$ y comencemos por $\omega_{11}^{(2)}$. Notemos que solo tenemos un camino que nos lleva por $\omega_{11}^{(2)}$ y por tanto

$$\begin{aligned}\frac{\partial a_1^{(4)}}{\partial \omega_{11}^{(2)}} &= a_1^{(4)} \left(1 - a_1^{(4)}\right) \omega_{11}^{(3)} a_1^{(3)} \left(1 - a_1^{(3)}\right) a_1^{(2)} \\ &= a_1^{(2)} a_1^{(3)} \left(1 - a_1^{(3)}\right) \omega_{11}^{(3)} a_1^{(4)} \left(1 - a_1^{(4)}\right) \\ &= a_1^{(2)} a_1^{(3)} \left(1 - a_1^{(3)}\right) \omega_{11}^{(3)} y_1 (1 - y_1)\end{aligned}$$

Para el resto de los pesos nos queda:

$$\begin{aligned}\frac{\partial a_1^{(4)}}{\partial \omega_{12}^{(2)}} &= a_1^{(2)} a_2^{(3)} \left(1 - a_2^{(3)}\right) \omega_{21}^{(3)} y_1 (1 - y_1) \\ \frac{\partial a_1^{(4)}}{\partial \omega_{21}^{(2)}} &= a_2^{(2)} a_1^{(3)} \left(1 - a_1^{(3)}\right) \omega_{11}^{(3)} y_1 (1 - y_1) \\ \frac{\partial a_1^{(4)}}{\partial \omega_{22}^{(2)}} &= a_2^{(2)} a_2^{(3)} \left(1 - a_2^{(3)}\right) \omega_{21}^{(3)} y_1 (1 - y_1)\end{aligned}$$

Para completar las derivadas parciales con respecto a los pesos ω notemos que los dos últimos que nos faltan son:

$$\begin{aligned}\frac{\partial a_1^{(4)}}{\partial \omega_{11}^{(3)}} &= a_1^{(3)} y_1 (1 - y_1) \\ \frac{\partial a_1^{(4)}}{\partial \omega_{21}^{(3)}} &= a_2^{(3)} y_1 (1 - y_1)\end{aligned}$$

Ahora, las derivadas que nos faltan son las que corresponden a los umbrales de cada capa; cada una de ellas se pueden calcular gráficamente como antes pero considerando que en cada umbral la entrada siempre es un 1. Entonces tenemos:

$$\begin{aligned}\frac{\partial a_1^{(4)}}{\partial \mu_1^{(1)}} &= a_1^{(2)} \left(1 - a_1^{(2)}\right) \omega_{11}^{(2)} a_1^{(3)} \left(1 - a_1^{(3)}\right) \omega_{11}^{(3)} y_1 (1 - y_1) + \\ &\quad a_1^{(2)} \left(1 - a_1^{(2)}\right) \omega_{12}^{(2)} a_2^{(3)} \left(1 - a_2^{(3)}\right) \omega_{21}^{(3)} y_1 (1 - y_1) \\ \frac{\partial a_1^{(4)}}{\partial \mu_2^{(1)}} &= a_2^{(2)} \left(1 - a_2^{(2)}\right) \omega_{21}^{(2)} a_1^{(3)} \left(1 - a_1^{(3)}\right) \omega_{11}^{(3)} y_1 (1 - y_1) + \\ &\quad a_2^{(2)} \left(1 - a_2^{(2)}\right) \omega_{22}^{(2)} a_2^{(3)} \left(1 - a_2^{(3)}\right) \omega_{21}^{(3)} y_1 (1 - y_1)\end{aligned}$$

$$\frac{\partial a_1^{(4)}}{\partial \mu_1^{(2)}} = a_1^{(3)} (1 - a_1^{(3)}) \omega_{11}^{(3)} y_1 (1 - y_1)$$

$$\frac{\partial a_1^{(4)}}{\partial \mu_2^{(2)}} = a_2^{(3)} (1 - a_2^{(3)}) \omega_{21}^{(3)} y_1 (1 - y_1)$$

$$\frac{\partial a_1^{(4)}}{\partial \mu_1^{(3)}} = y_1 (1 - y_1)$$

De esta manera hemos calculado explícitamente todas las derivadas parciales con respecto a los pesos ω y los umbrales μ para la red neuronal artificial de la Figura (14), y por tanto estamos en condiciones de escribir de manera general todas las derivadas parciales que se puedan requerir para el Perceptrón de la Figura (11). Las mismas quedan de la siguiente manera:

$$\left. \begin{aligned} \frac{\partial y_i}{\partial \omega_{ji}^{(3)}} &= a_j^{(3)} y_i (1 - y_i) \text{ para } i = 1, 2 \text{ y para } j = 1, 2, 3, 4 \\ \frac{\partial y_i}{\partial \mu_i^{(3)}} &= y_i (1 - y_i) \text{ para } i = 1, 2 \end{aligned} \right\} \text{En la capa 3} \quad (11)$$

$$\left. \begin{aligned} \frac{\partial y_i}{\partial \omega_{jk}^{(2)}} &= a_j^{(2)} a_k^{(3)} (1 - a_k^{(3)}) \omega_{ki}^{(3)} y_i (1 - y_i) \\ &\quad \text{para } i = 1, 2 \text{ para } k = 1, 2, 3, 4 \text{ para } j = 1, 2, 3, 4 \\ \frac{\partial y_i}{\partial \mu_j^{(2)}} &= a_j^{(3)} (1 - a_j^{(3)}) \omega_{ji}^{(3)} y_i (1 - y_i) \\ &\quad \text{para } i = 1, 2 \text{ y para } j = 1, 2, 3, 4 \end{aligned} \right\} \text{En la capa 2} \quad (12)$$

$$\left. \begin{aligned} \frac{\partial y_i}{\partial \omega_{jk}^{(1)}} &= x_j a_k^{(2)} (1 - a_k^{(2)}) \left[\sum_{p=1}^4 \omega_{kp}^{(2)} a_p^{(3)} (1 - a_p^{(3)}) \omega_{pi}^{(3)} \right] y_i (1 - y_i) \\ &\quad \text{para } i = 1, 2 \text{ para } j = 1, 2, 3 \text{ para } k = 1, 2, 3, 4 \\ \frac{\partial y_i}{\partial \mu_j^{(1)}} &= a_j^{(2)} (1 - a_j^{(2)}) \left[\sum_{p=1}^4 \omega_{jp}^{(2)} a_p^{(3)} (1 - a_p^{(3)}) \omega_{pi}^{(3)} \right] y_i (1 - y_i) \\ &\quad \text{para } i = 1, 2 \text{ y para } j = 1, 2, 3, 4 \end{aligned} \right\} \text{En la capa 1} \quad (13)$$

El ultimo paso que necesitamos para completar el algoritmo de Retropropagación es una medida para el error que se puede cometer en cada paso que se ajuste cada uno de los potenciómetros anteriormente

mencionado y que se esquematiza en la Figura (1). Para el efecto hay que considerar la salida deseada y tomando la red neuronal artificial de la Figura (11) podemos decir que $\{s_1, s_2\}$ son dichas salidas. El candidato para el error que funciona en prácticamente todos los casos es la distancia euclíadiana, de hecho, es la mitad del cuadrado de dicha distancia y para nuestro caso está dado por

$$E(y_1, y_2) = \frac{(s_1 - y_1)^2}{2} + \frac{(s_2 - y_2)^2}{2} \quad (14)$$

Nótese que hemos definido el error como una función que depende de la salida de la red. El objetivo desde ahora es minimizar el error y para el efecto vamos utilizar el descenso del gradiente y las derivadas parciales que se encontraron en los apartados anteriores. Si calculamos la derivada parcial con respecto a algunos de los pesos ω tendremos un esquemas del tipo

$$\begin{aligned} \frac{\partial E}{\partial \square} &= \frac{\partial E}{\partial y_1} \frac{\partial y_1}{\partial \square} + \frac{\partial E}{\partial y_2} \frac{\partial y_2}{\partial \square} \\ &= -(s_1 - y_1) \frac{\partial y_1}{\partial \square} + -(s_2 - y_2) \frac{\partial y_2}{\partial \square} \\ &= \sum_{i=1}^2 -(s_i - y_i) \frac{\partial y_i}{\partial \square} \end{aligned}$$

Dejamos el cubo vacío en la formula anterior como representante de cualquiera de los pesos ω ya que funciona para todos los casos, incluido el caso de los umbrales μ .

Ahora bien, las derivadas parciales con respecto a los pesos nos queda:

$$\frac{\partial E}{\partial \omega_{ji}^{(3)}} = a_j^{(3)} y_i (1 - y_i) [-(s_i - y_i)] \quad (15)$$

$$\frac{\partial E}{\partial \omega_{jk}^{(2)}} = a_j^{(2)} a_k^{(3)} (1 - a_k^{(3)}) \sum_{i=1}^{n_4} \omega_{ki}^{(3)} y_i (1 - y_i) [-(s_i - y_i)] \quad (16)$$

$$\frac{\partial E}{\partial \omega_{jk}^{(1)}} = x_j a_k^{(2)} (1 - a_k^{(2)}) \sum_{p=1}^{n_3} \omega_{kp}^{(2)} a_p^{(3)} (1 - a_p^{(3)}) \sum_{i=1}^{n_4} \omega_{pi}^{(3)} y_i (1 - y_i) [-(s_i - y_i)] \quad (17)$$

Para obtener las derivas parciales con respecto a los umbrales lo que se hace es omitir el primer factor en las ecuaciones (15), (16), (18) y con eso nos queda:

$$\begin{aligned} \frac{\partial E}{\partial \mu_i^{(3)}} &= y_i (1 - y_i) [-(s_i - y_i)] \\ \frac{\partial E}{\partial \mu_k^{(2)}} &= a_k^{(3)} (1 - a_k^{(3)}) \sum_{i=1}^{n_4} \omega_{ki}^{(3)} y_i (1 - y_i) [-(s_i - y_i)] \\ \frac{\partial E}{\partial \mu_k^{(1)}} &= a_k^{(2)} (1 - a_k^{(2)}) \sum_{p=1}^{n_3} \omega_{kp}^{(2)} a_p^{(3)} (1 - a_p^{(3)}) \sum_{i=1}^{n_4} \omega_{pi}^{(3)} y_i (1 - y_i) [-(s_i - y_i)] \end{aligned}$$

A partir de este momento ya estamos en condiciones para aplicar el algoritmo de Backpropagation, es decir, dado un valor inicial ω_i para los pesos ω y μ_i para los umbrales μ la actualización de los mismos en

cada paso será

$$\omega_{i+1} = \omega_i - \alpha \frac{\partial E}{\partial \omega_i}$$

$$\mu_{i+1} = \mu_i - \alpha \frac{\partial E}{\partial \mu_i}$$

10.1. Imputación de Valores perdidos, Acotación de las variables de entrada y Escalamiento

Ahora que ya tenemos todo detallado el proceso del algoritmo de Retropropagación debemos hacer mención a tres temas no menores y en general fundamentales para el éxito en la predicción mediante cualquier modelo. El primero de ellos es la imputación de valores perdidos o *imputación de missing*: esto consiste en no trabajar con variables que tengan valores faltantes, de hecho este proceso se realiza en la etapa de armado de la base de datos a utilizar en desarrollo. Existen varios métodos para imputar valores perdidos, los más comunes consisten en imputar por la media, la moda, o la mediana de la variable en cuestión, otra se imputa considerando la naturaleza de la variable. Más alternativas para imputar valores perdidos se pueden ver en [Castro and Ávila, 2006], en [Munoz Rosas et al., 2009] y también en [Rosati, 2021].

El siguiente en la lista es la acotación o capeo de variables, esto consiste en elegir una cota superior en todos los casos y una cota inferior en muchos casos para las variables de entrada de la red neuronal artificial. Para la cota superior (o máximo valor permitido) se utiliza generalmente el valor del percentil noventa y tantos, puede ser el percentil 95, o el percentil 97 o el percentil 99. Para la cota inferior (o mínimo valor permitido) se usa también el percentil que puede estar entre el percentil 1 o percentil 2 o incluso el percentil 3. Debemos tener presente que hay variables de entrada que por su naturaleza ya están inferiormente acotadas o capeadas, por ejemplo, cualquier variable de cantidad ya se sabe que el mínimo que puede tomar es cero y por tanto no hace mucha falta aplicar el capeo inferiormente y en esta situación solo se aplica el capeo superior. Al aplicar los capeos a las variables de entrada antes de entrenar el modelo estamos agregando un control extra sobre la red neuronal artificial porque estamos evitando los efectos que puede provocar un valor atípico a la salida de nuestra red. Supongamos que en la base que se usó para el entrenamiento de la red neuronal artificial teníamos la variable de entrada "número de hijos en la pareja"uyo valor máximo era 2 y cuando queremos hacer una predicción sobre nuevos datos nos encontramos que la variable "número de hijos en la pareja"toma el valor 7, valor que no estaba presente en la base de entrenamiento y por tanto la salida puede dar un valor mucho más alejado del valor real. Si hubiéramos aplicado el capeo previo al entrenamiento de nuestra red, entonces antes de la predicción también debemos aplicarla, es decir, imputamos el valor de esta nueva entrada por el valor del percentil que se utilizó y luego metemos en la red para obtener la predicción.

El siguiente punto destacado que se debe mencionar es el escalamiento de las variables de entrada y salida. Hasta ahora el capeo y la imputación de missing solo se aplica a las variables de entrada. El escalamiento se debe aplicar a todas las variables, tanto de entrada como de salida. La necesidad del escalamiento proviene de la función de activación que se utiliza en la red neuronal artificial. Recordemos que el rango de valores posible de la Sigmoide está en el intervalo (0, 1) mientras que la tangente hiperbólica tiene rango en (-1, 1). Entonces antes del entrenamiento y después de imputar missing y capear las variables debemos hacer un cambio de escala para todas las variables que no se encuentren en el intervalo [0, 1]. Si usamos la

función Sigmoide como función de activación el cambio de escala se obtiene con la siguiente formula:

$$\hat{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (18)$$

donde x_{\max} y x_{\min} es el máximo y el mínimo de la variable x . Así \hat{x} ya está en el intervalo $[0, 1]$. Por ejemplo, si los datos del Cuadro (4) representan los datos originales que queremos usar para entrenar una red neuronal artificial con función de activación Sigmoidal, primeramente habrá que hacer el cambio de escala. Aplicando

X_1	X_2	X_3	S_1	S_2
5	6	45	50	3
2	2	30	29	5
-1	0	40	38	7
4	7	10	14	2
-2	5	20	11	8

Cuadro 4: Datos Originales.

el cambio de escala mediante la formula (18) nos queda los datos que se muestran en el Cuadro (5)

X_1	X_2	X_3	S_1	S_2
1.00	0.86	1.00	1.00	0.17
0.57	0.29	0.57	0.46	0.50
0.14	0.00	0.86	0.69	0.83
0.86	1.00	0.00	0.08	0.00
0.00	0.71	0.29	0.00	1.00

Cuadro 5: Datos Escalados.

Una vez que tengamos la red entrenada y estemos en posición de hacer las predicciones con nuevos datos, habrá que despejar la x de la ecuación (18) para tener la formula que nos permita encontrar el valor de la salida que nos va dar como predicción nuestra red en la misma escala que el valor original, dicha formula es

$$x = (x_{\max} - x_{\min})\hat{x} + x_{\min}$$

donde x_{\max} y x_{\min} son los valores máximos y mínimos que se habían encontraron en la base de desarrollo mientras se entrenaba la red neuronal artificial. Como lo que nos interesa es la salida de la red escribimos la formula anterior en términos de S y nos queda

$$S = (S_{\max} - S_{\min})\hat{S} + S_{\min}$$

Nótese que la imputación, el capeo y el escalamiento solo funciona para variables cuantitativas, por tanto, en presencia de variables categóricas necesitamos algún método para convertirlas a variables numéricas. Existen estrategias que nos permiten llevar a cabo esto y las mismas dependen de la cardinalidad de la variables categóricas, por ejemplo, la variable *países del mundo* tiene más de 150 categorías, pero la variable

sexo puede como mucho tener 3 categorías¹ por lo que cada caso se trata de manera diferente, incluso los valores perdidos podrían representar una categoría más de dicha variables. Mas detalles de este tema están en [Seger, 2018], en [Dahouda and Joe, 2021] y también en [Cerda and Varoquaux, 2020].

11. Ejemplo de Aplicación de un Perceptrón Multicapa

Para mostrar este ejemplo tomamos de referencia la base de datos y el análisis que se hizo en [Duarte, 2017]. En efecto, tomamos la misma base de datos simulada que recrea la información de 1000 clientes que tiene un banco² clasificados en *bueno* = 1 o *malo* = 0 y guardada en la variable *Credit*. Para predecir la probabilidad de que un cliente sea *malo* en un futuro se dispone de informaciones como *El saldo actual de la cuenta corriente*, *Pagos de créditos anteriores*, *Porpósito del crédito*, *Cantidad de créditos en el banco*, *Valor de la caja de ahorro*, *Antigüedad laboral*, *Cuota en % de los ingresos disponibles*, *Estado civil* y *sexo* entre otras variables. Con esta base [Duarte, 2017] aplica una regresión logística y obtiene un modelo cuyo comportamiento se puede ver en el Cuadro (6)

Tramos	Score medio	%Total	%AcTotal	%Bueno	%AcBueno	%DesBueno	%Malo	%AcMalo	%DesMalo	Tasa Malo	TasaAcMalo	KS
[975, 1000]	986.88	9.7	9.7	13.43	13.43	100.00	1.00	1.00	100.00	3.09	3.09	12.43
[947, 975)	960.56	10.1	19.8	13.43	26.86	86.57	2.33	3.33	99.00	6.93	5.05	23.52
[906, 947)	925.85	10.1	29.9	13.57	40.43	73.14	2.00	5.33	96.67	5.94	5.35	35.10
[845, 906)	875.16	10.1	40.0	11.29	51.71	59.57	7.33	12.67	94.67	21.78	9.50	39.05
[778, 845)	810.96	10.0	50.0	11.00	62.71	48.29	7.67	20.33	87.33	23.00	12.20	42.38
[684, 778)	730.01	9.9	59.9	10.71	73.43	37.29	8.00	28.33	79.67	24.24	14.19	45.10
[549, 684)	620.48	10.0	69.9	9.00	82.43	26.57	12.33	40.67	71.67	37.00	17.45	41.76
[429, 549)	495.68	10.1	80.0	8.00	90.43	17.57	15.00	55.67	59.33	44.55	20.88	34.76
[302, 429)	361.50	10.0	90.0	6.14	96.57	9.57	19.00	74.67	44.33	57.00	24.89	21.90
[0, 302)	184.81	10.0	100.0	3.43	100.00	3.43	25.33	100.00	25.33	76.00	30.00	0.00
Total	94.76	100.0	NA	100.00	NA	NA	100.00	NA	NA	NA	NA	45.10

Cuadro 6: Tabla de Performance Score de Riesgo - Regresión Logística

En el cuadro (6) se agrupa los mil clientes y los ordena según el score en 10 tramos, donde en cada uno se tiene aproximadamente 10 % de los casos. A demás se observa como se distribuyen los buenos y los malos; en este caso se tiene una tasa de malo del 30 % (esto es, la penúltima fila de la columna “TasaAcMalo”). La columna “Tasa Malo” es el porcentaje de malos que cae en cada tramo de score. Notemos que el mejor tramo [975, 1000] tiene una tasa de malo del 3.09 %, mientras que el peor tramo [0, 302] tiene una tasa del 76.00 % (25 veces más que el mejor tramo) y que la tasa de malo va creciendo a medida que disminuye el score salvo el tramo [906, 947] que tiene un pequeño salto ya que la tasa de malo va de 6.93 bajando a 5.94 y luego subiendo a 21.78. Otra la columna a mencionar es la columna *KS* que se corresponde con la prueba estadística de *Kolmogórov - Smirnov* y que determina la bondad de ajuste de dos distribuciones, para este caso son las distribuciones de buenos y malos. El *KS* varía de 0 a 100 donde 100 significa que hay una separación total de las distribuciones y 0 significa que las dos distribuciones son idénticas. Cuanto más alto es el *KS* mejor es el modelo ya que esto implica que el modelo logra separar en gran medida a los buenos de los malos. La experiencia sugiere que en poblaciones con poca información crediticia el *KS* varía entre 15 y 25, mientras que en poblaciones con mucha información de crédito varía entre 30 y 70.

Para modelos de respuesta binaria el *KS* de cada tramo de score es el valor absoluto de la diferencia

¹Cuando se estudia abortos espontáneos en salud humana o animal puede darse que el feto se haya desarrollado muy poco y por tanto el sexo es indeterminado

²tomados de http://www.learnanalytics.in/datasets/Credit_Scoring.zip

entre el porcentaje acumulado de buenos y el porcentaje acumulado de malos, luego el *KS* final del modelo es el máximo entre todos estos *KS*'s. El modelo de este caso tiene un *KS* = 45.10 %.

Con los mismo datos entrenamos una red neuronal artificial con 20 entradas, una capa oculta de siete nodos, y dos salidas. Dicha configuración se puede ver en la Figura (16) Con la red neuronal artificial de

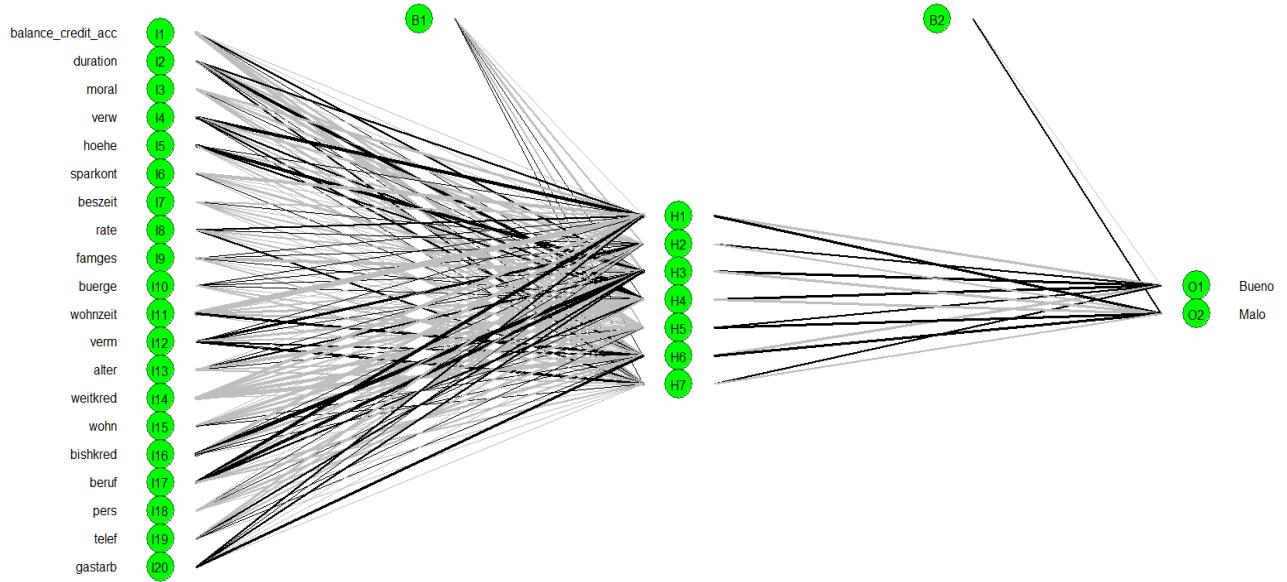


Figura 16: Back Propagation - Score de Riesgo Crediticio

la Figura (16) mejoramos el desempeño del modelo comparado con aquel que se entrenó con una regresión logística. Los resultados se pueden ver en el Cuadro (7)

Tramos	Score medio	%Total	%AcTotal	%Bueno	%AcBueno	%DesBueno	%Malo	%AcMalo	%DesMalo	Tasa Malo	TasaAcMalo	KS
[979, 1000]	992.31	9.7	9.7	13.00	13.00	100.00	2.00	2.00	100.00	6.19	6.19	11.00
[947, 979]	962.47	10.2	19.9	13.43	26.43	87.00	2.67	4.67	98.00	7.84	7.04	21.76
[906, 947]	925.75	10.0	29.9	13.14	39.57	73.57	2.67	7.33	95.33	8.00	7.36	32.24
[864, 906]	884.92	9.9	39.8	12.00	51.57	60.43	5.00	12.33	92.67	15.15	9.30	39.24
[805, 864)	835.89	9.9	49.7	11.86	63.43	48.43	5.33	17.67	87.67	16.16	10.66	45.76
[717, 805)	764.15	10.3	60.0	12.14	75.57	36.57	6.00	23.67	82.33	17.48	11.83	51.90
[615, 717)	667.94	10.0	70.0	10.43	86.00	24.43	9.00	32.67	76.33	27.00	14.00	53.33
[442, 615)	531.30	10.0	80.0	7.57	93.57	14.00	15.67	48.33	67.33	47.00	18.12	45.22
[258, 442)	349.75	10.0	90.0	4.71	98.29	6.43	22.33	70.67	51.67	67.00	23.56	27.62
[0, 258)	95.04	10.0	100.0	1.71	100.00	1.71	29.33	100.00	29.33	88.00	30.00	0.00
Total	700.47	100.0	NA	100.00	NA	NA	100.00	NA	NA	NA	NA	53.33

Cuadro 7: Tabla de Performance Score de Riesgo - Red Neuronal Artificial

Comparando los resultados del Cuadro (6) y el Cuadro (7) tenemos que la red neuronal le saca 8 puntos de *KS* a la regresión logística, 53.33 % contra 45.10 %, además elimina el pequeño salto que se da en la tasa de malo. Nótese que en todos los tramos de score del Cuadro (7) la tasa de malos va creciendo sin dar saltos. Otro punto de comparación es el 50 % mejor puntuado que en el Cuadro (6) vemos para los casos con score mayor o igual a 778, mientras que para el Cuadro (7) se corresponde con los casos de score mayor o igual a 805; para cada uno de estos score vemos que la tasa acumulada de malos (TasaAcMalo) es 12.20 %

y 10.66 %, respectivamente, es decir, el modelo de red neuronal artificial está dejando pasar menos casos malos hacia los rangos de score más alto que el modelo de regresión logística. Otra manera de ver una ganancia como esta es considerar por ejemplo el 30 % con peor score; para el Cuadro (6) le corresponde los casos con score menor a 549 y para el Cuadro (7) le corresponde los casos con score menor a 615; mirando la distribución desacumulada de malos (%DesMalo) para estos tramos de score tenemos un 59.33 % para la regresión logística y 67.33 % para la red neuronal artificial, es decir, la red está captando más casos malos en los peores tramos de score comparado con lo que capta la regresión logística. Por lo tanto, con la red neuronal artificial las predicciones serán mejores que con la regresión logística. Todo el código usado en *r-project* para entrenar la red se encuentra en el anexo.

Conclusión

La primera vez que una investigación se ocupa de una red neuronal lo encontramos en los trabajos de Sigmund Freud en 1964 y desde ese momento varios científicos fueron marcando etapas en el avance de este tema, entre ellos Marvin Minsky con su maquina de 40 neuronas, Karl Steinbuch aplicando redes en reconocimiento de escritura a mano distorsionada, Frank Rosenblatt dando nacimiento al perceptron, Stephen Grossberg cambiando paradigmas mediante el análisis matemático, Shun-Ichi Amari resolviendo mediante la red el problema de asignación de créditos, Terence Sejnowski con participación clave en el algoritmo de retropropagación y John Hopfield que logra despertar nuevamente el interés de otros científicos en las redes neuronales artificiales con su red neuronal de una capa, todos ellos con fundamentales aportes.

Las aplicaciones de las redes neuronales son bastante diversas, encontramos aplicaciones en psicología donde se han desarrollado modelos de redes neuronales artificiales para predecir el divorcio. También se aplicaron a reconocimientos de escritura a mano distorsionada; se aplicaron para resolver el problema de asignación de créditos como así también al reconocimiento del lenguaje hablado. Hoy en día con el avance tecnológico aparecen aplicaciones en Marketing, Medicina, Finanzas, Deportes, incluso en temas de leyes jurídicas.

Una red neuronal artificial es un modelo matemático inspirado en el aparato de comunicación neuronal de los humanos, es decir, está basada en el funcionamiento del cerebro humano. Nuestro cerebro es capaz de procesar información de manera mucha más eficiente que cualquier computadora convencional y por tanto tenemos la capacidad de reflexionar y aprender. Se han diseñado sistemas y estructuras que simulan el funcionamiento del cerebro humano con el objetivo de dotar a estos sistemas y estructuras de la capacidad de pensar por si mismos.

Existen varias metodologías disponibles que se pueden utilizar para el entrenamiento de una red neuronal artificial. Aquí hemos descripto paso a paso el algoritmo de retropropagación o Backpropagation, sin embargo también están disponibles algoritmos como el forward propagation, el ADALINE, el aprendizaje Hebbiano, el Brain-state-in-a-box entre otros.

La arquitectura del perceptrón multicapa está formado por unas entradas, una, dos o varias capas ocultas compuesta por una, dos o varias neuronas interconectadas a las entradas, y una capa de salida que consta de una, dos o varias neuronas. Cada neurona de la capa oculta representa un elemento con un estado interno que recibe señales que le permiten cambiar a otro estado, estas señales son procesadas mediante la función de activación de la neurona.

El algoritmo de retropropagación utiliza el método del descenso del gradiente para actualizar los pesos para las distintas conexiones entre neuronas y umbrales de una red neuronal artificial. Estos pesos se van

modificando con la intención de minimizar el error que se comete en el proceso de aprendizaje durante el entrenamiento de una red. Este error a minimizar está definido en términos del valor real y el valor que se obtiene en la salida de la red.

Finalmente vimos en el ejemplo con datos simulados que el modelo obtenido mediante una red neuronal artificial brinda mejores predicciones que el modelo obtenido con una regresión logística, ya que en términos de KS, o en términos de malos captados en los peores tramos, o en términos de malos acumulados en los mejores tramos de score, funcionaba mejor los resultados de la red neuronal.

Bibliografía

- [Amari, 1971] Amari, S.-I. (1971). Characteristics of randomly connected threshold-element networks and network systems. *Proceedings of the IEEE*, 59(1):35–47.
- [Amari, 1972] Amari, S.-I. (1972). Characteristics of random nets of analog neuron-like elements. *IEEE Transactions on systems, man, and cybernetics*, (5):643–657.
- [Amari, 1974] Amari, S.-i. (1974). A method of statistical neurodynamics. *Kybernetik*, 14(4):201–215.
- [Anderson and Murphy, 1986] Anderson, J. A. and Murphy, G. L. (1986). Psychological concepts in a parallel system. Technical report, BROWN UNIV PROVIDENCE RI CENTER FOR NEURAL SCIENCE.
- [Castrillón, 2021] Castrillón, O. D. (2021). Predicción del divorcio por medio de técnicas inteligentes. *Información tecnológica*, 32(5):111–120.
- [Castro and Ávila, 2006] Castro, L. M. U. and Ávila, D. M. M. (2006). Una introducción a la imputación de valores perdidos. *Terra. Nueva Etapa*, 22(31):127–151.
- [Cerda and Varoquaux, 2020] Cerda, P. and Varoquaux, G. (2020). Encoding high-cardinality string categorical variables. *IEEE Transactions on Knowledge and Data Engineering*.
- [Dahouda and Joe, 2021] Dahouda, M. K. and Joe, I. (2021). A deep-learned embedding technique for categorical features encoding. *IEEE Access*, 9:114381–114391.
- [Duarte, 2017] Duarte, A. (2017). <https://www.academia.edu/s/581a36d837?source=link>, 23 de Diciembre de 2022.
- [Fitch, 1944] Fitch, F. B. (1944). Warren s. mcculloch and walter pitts. a logical calculus of the ideas immanent in nervous activity. bulletin of mathematical biophysics, vol. 5 (1943), pp. 115–133. *The Journal of Symbolic Logic*, 9(2):49–50.
- [Freud and Strachey, 1964] Freud, S. and Strachey, J. E. (1964). The standard edition of the complete psychological works of sigmund freud.
- [Fukushima, 1969] Fukushima, K. (1969). Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333.
- [Fukushima, 1975] Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3):121–136.
- [Grossberg, 2012] Grossberg, S. T. (2012). *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*, volume 70. Springer Science & Business Media.
- [Hinton et al., 1984] Hinton, G. E., Sejnowski, T. J., and Ackley, D. H. (1984). *Boltzmann machines: Constraint satisfaction networks that learn*. Carnegie-Mellon University, Department of Computer Science Pittsburgh, PA.
- [Hopfield, 1982] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

- [Klop and Gose, 1969] Klopf, A. H. and Gose, E. (1969). An evolutionary pattern recognition network. *IEEE Transactions on Systems Science and Cybernetics*, 5(3):247–250.
- [Kohonen, 1971] Kohonen, T. (1971). A class of randomly organized associative memories. *Acta Polytechnica Scandinavica*, 25.
- [Kohonen, 1972] Kohonen, T. (1972). Correlation matrix memories. *IEEE transactions on computers*, 100(4):353–359.
- [Leithold et al., 2001] Leithold, L. et al. (2001). Cálculo.
- [León and Viñuela, 2004] León, I. G. and Viñuela, P. I. (2004). Redes neuronales artificiales, un enfoque práctico.
- [McClelland and Rumelhart, 1981] McClelland, J. L. and Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception: I. an account of basic findings. *Psychological review*, 88(5):375.
- [Minsky, 1954] Minsky, M. (1954). Neural nets and the brain-model problem. *Unpublished doctoral dissertation, Princeton University, NJ*.
- [Munoz Rosas et al., 2009] Munoz Rosas, J. F., Alvarez Verdejo, E., et al. (2009). Métodos de imputación para el tratamiento de datos faltantes: aplicación mediante r/splus.
- [Rosati, 2021] Rosati, G. (2021). Métodos de machine learning como alternativa para la imputación de datos perdidos. un ejercicio en base a la encuesta permanente de hogares. *Estudios del trabajo*, (61):1–23.
- [Rosenblatt, 1957] Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.
- [Rosenblatt, 1961] Rosenblatt, F. (1961). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY.
- [Rumelhart, 1977] Rumelhart, D. E. (1977). Toward an interactive model of reading. In *Attention and performance VI*, pages 573–603. Routledge.
- [Russell et al., 2013] Russell, B., Blackwell, K., and Eames, E. R. (2013). *Theory of knowledge: The 1913 manuscript*. Routledge.
- [Seger, 2018] Seger, C. (2018). An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing.
- [Shaw, 1986] Shaw, G. (1986). Donald hebb: The organization of behavior. In *Brain Theory*, pages 231–233. Springer.
- [Steinbuch, 1965] Steinbuch, K. (1965). Adaptive networks using learning matrices. *Kybernetik*, 2(4):148–152.

[Uttley, 1956] Uttley, A. M. (1956). Probability machine. *Automata Studies: Annals of Mathematics Studies. Number 34*, (34):277.

[Widrow, 1959] Widrow, B. (1959). Adaptive sample data systems-a statistical theory of adaptation. 1959 wescon convention record. *Part, 4*:74–85.

[Widrow et al., 1960] Widrow, B. et al. (1960). Adaptive sampled-data systems. In *Proceedings of the First International Congress of the International Federation of Automatic Control*, pages 406–411.

Anexo

Código usado en *r – project* para la obtención del modelo aplicando el algoritmo de Backpropagation

```
# Cargamos las librerias que vamos a necesitar
library(caTools)
library(neuralnet)
library(NeuralNetTools)
# Leemos la base de datos que están en credit.csv
datos <- read.csv("DATA/credit.csv",sep = ",", dec = ".", header = TRUE)
# Escalamos los datos antes de arrancar. Por tratarse de datos
# simulados no tienen valores perdidos
maxs <- apply(datos[,-c(1)], 2, max)
mins <- apply(datos[,-c(1)], 2, min)
datos <- as.data.frame(cbind(datos$Credit,scale(datos[,-c(1)],
center = mins, scale = maxs - mins)))
datos$Credit =as.factor(datos$V1)
datos$V1 <- NULL
# Armamos una variables aleatoria para elegir la base de entrenamiento
set.seed(1234)
datos$rand <- runif(dim(datos)[1])
# Configuramos nuestro perceptrón multicapa
modelo <- neuralnet(
  formula = Credit ~ .,
  data = datos[datos$rand <= 0.7,-c(22)],
  hidden = c(7),
  linear.output = T,
  act.fct = "logistic",
  algorithm = "backprop",
  threshold = 0.3,
  learningrate = 0.002

)
# Tomamos los resultados de la red
dato_test <- neuralnet::compute(modelo,datos)$net.result
# Graficamos la red
plotnet(modelo,y_names = c("Bueno","Malo")
        ,circle_col = "green",bord_col = "darkgreen")
# Convertimos el resultado en un score multiplicando por 1000
# la probabilidad arrojada por la red
datos$score <- round(1000*dato_test[,2])
datos$score[datos$score>1000] <- 1000
datos$score[datos$score<0] <- 0
datos$target <- ifelse(datos$Credit==1,0,1)
```

```

# Hacemos un resumen del score para ver que sea
# un numero entre 0 y 1000
summary(datos$score)

#Armamos una función que calcula la tabla de comportamiento
tabl_comp<-function(x,y,g=10){
  #Para instalar el paquete Hmisc ejecutamos install.package('Hmisc')
  library(Hmisc)

  rango<-cut2(x,g=g)
  ordena<-seq(1:length(unique(rango)))
  tab_base<-cbind(ordena,table(factor(rango,exclude=NULL),y))
  tab_base<-tab_base[order(-tab_base[,1]),2:3]

  total<-margin.table(tab_base,1)
  total_ac<-cumsum(total)
  total_des<-cbind(ordena,cumsum(total[order(-total_ac)]))
  total_des<-total_des[order(-total_des[,1]),2]

  porc_total<-prop.table(total)*100
  porc_ac_total<-cumsum(porc_total)
  porc_des_total<-cbind(ordena,cumsum(porc_total[order(-porc_ac_total)]))
  porc_des_total<-porc_des_total[order(-porc_des_total[,1]),2]

  buenos<-tab_base[,1]
  porc_buenos<-prop.table(buenos)*100
  porc_ac_buenos<-cumsum(porc_buenos)
  porc_des_buenos<-cbind(ordena,cumsum(porc_buenos[order(-porc_ac_buenos)]))
  porc_des_buenos<-porc_des_buenos[order(-porc_des_buenos[,1]),2]

  malos<-tab_base[,2]
  malos_ac<-cumsum(malos)
  malos_des<-cbind(ordena,cumsum(malos[order(-malos_ac)]))
  malos_des<-malos_des[order(-malos_des[,1]),2]

  porc_malos<-prop.table(malos)*100
  porc_ac_malos<-cumsum(porc_malos)
  porc_des_malos<-cbind(ordena,cumsum(porc_malos[order(-porc_ac_malos)]))
  porc_des_malos<-porc_des_malos[order(-porc_des_malos[,1]),2]

  tasa_malos<-tab_base[,2]/total*100
  tasa_ac_malos<-malos_ac/total_ac*100
  tasa_des_malos<-malos_des/total_des*100

```

```
ks<-abs(porc_ac_buenos-porc_ac_malos)

val_medio<-cbind(c(1:length(unique(rango))),  
tapply(x,factor(rango,exclude=NULL),mean))
med_score<-mean(x)
val_medio<-val_medio[order(-val_medio[,1]),2]
tab_final<-cbind(val_medio,total,porc_total,porc_ac_total,porc_des_total,  
buenos,porc_buenos,porc_ac_buenos,porc_des_buenos,malos,porc_malos,  
porc_ac_malos,porc_des_malos,tasa_malos,tasa_ac_malos,ks)
colnames(tab_final)<-c('Score medio','Total','%Total','%AcTotal',  
'%DesTotal','Bueno','%Bueno','%AcBueno','%DesBueno','Malo',  
'%Malo','%AcMalo','%DesMalo','Tasa_Malo','TasaAcMalo','KS')

total_tab <- cbind(med_score,sum(tab_final[,2]),sum(tab_final[,3]),NA,NA,  
sum(tab_final[,6]),sum(tab_final[,7]),NA,NA,sum(tab_final[,10]),  
sum(tab_final[,11]),NA,NA,NA,NA,max(tab_final[,16]))

tab_final <- round(rbind(tab_final,total_tab),2)
tab_final
}

# Calculamos la tabla de performance
tabl_comp(datos$score,datos$target)
```