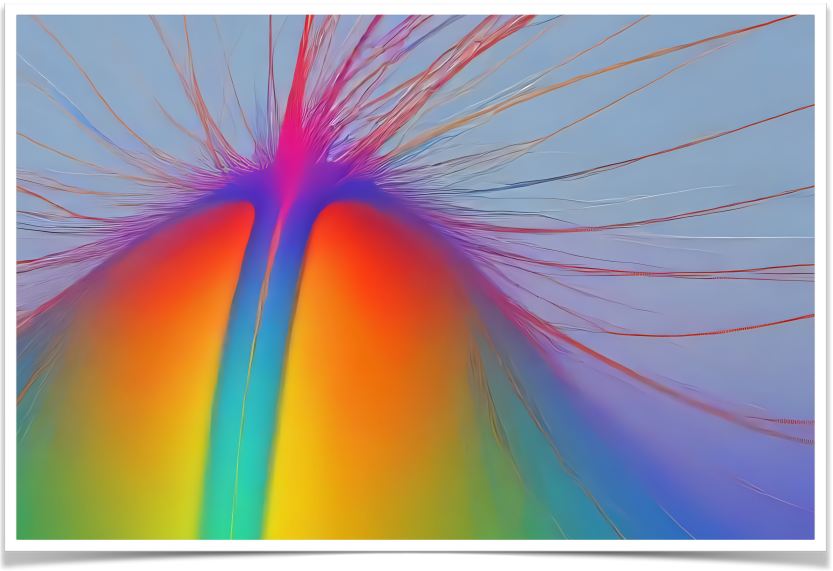

El perceptrón

La neurona artificial que funda las bases de la inteligencia artificial.

La historia del perceptrón se remonta a los años 50 y 60 cuando el psicólogo y científico informático Frank Rosenblatt, definió las bases del perceptrón como una aproximación matemática del funcionamiento de una neurona del cerebro humano.

Hitos históricos

McCulloch-Pitts Neuron: En 1943, Warren McCulloch y Walter Pitts publicaron un artículo que describía un modelo simplificado de neurona llamado la "neurona de McCulloch-Pitts". Esta neurona artificial tenía múltiples entradas binarias, cada una con un peso asociado, y producía una salida binaria en función de una función de activación simple.

Perceptrón de Rosenblatt: En 1957, Frank Rosenblatt introdujo el perceptrón como una mejora del modelo de la neurona de McCulloch-Pitts. El perceptrón era capaz de aprender mediante la adaptación de los pesos de las conexiones entre las entradas y la salida. Rosenblatt demostró que un perceptrón de una sola capa podía aprender a clasificar patrones linealmente separables.

Libro de Rosenblatt: En 1962, Rosenblatt publicó su libro "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms", que detallaba su investigación sobre el perceptrón y su teoría sobre cómo podría emular el aprendizaje y la percepción en el cerebro humano.

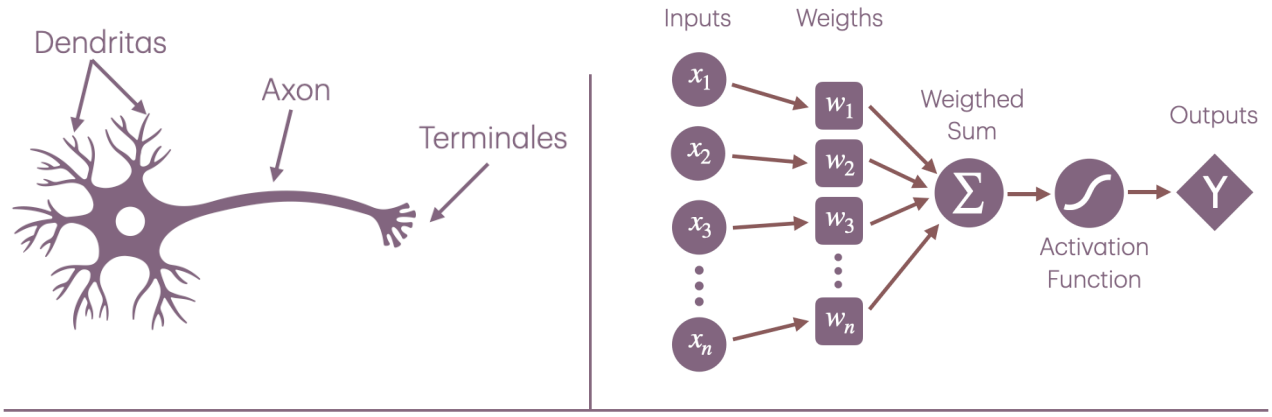

Figura 1: Esquema básico de una neurona (izquierda) y el perceptrón (derecha)

Críticas de Minsky y Papert: En 1969, Marvin Minsky y Seymour Papert publicaron el libro "Perceptrons", en el que presentaron críticas fundamentales sobre las limitaciones del perceptrón de una sola capa. Mostraron que los perceptrones simples no podían aprender a clasificar patrones que no fueran linealmente separables y argumentaron que las redes neuronales necesitaban más capas para abordar problemas más complejos.

A pesar de las críticas de Minsky y Papert, el perceptrón y los conceptos relacionados con las redes neuronales experimentaron un renacimiento en las décadas siguientes. Esto se debió en parte a los avances en algoritmos de aprendizaje como la retropropagación, que permitieron el entrenamiento eficiente de redes neuronales profundas con múltiples capas.

Aunque el perceptrón original de una sola capa resultó limitado en su capacidad para resolver problemas complejos, sentó las bases para el desarrollo posterior de redes neuronales más avanzadas, que hoy en día son fundamentales en numerosas aplicaciones de inteligencia artificial y aprendizaje automático.

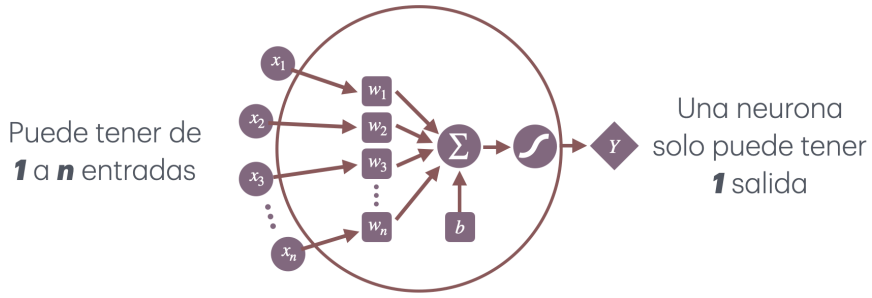

Figura 2: Esquema básico de una neurona artificial o perceptrón simple

El perceptrón, ver figura 1 y 2, es un modelo básico de red neuronal, inspirado en el funcionamiento del cerebro humano. Consiste en una estructura simple que toma entradas, las pondera y luego las pasa a través de una función de activación para producir una salida.

Conceptualmente, el funcionamiento del perceptrón implica los siguientes pasos:

- Entradas: El perceptrón recibe entradas $[x_1, x_2, \ldots x_n]$, las cuales son valores numéricos que representan características de un objeto o evento. Por ejemplo, en un perceptrón que clasifica imágenes de dígitos escritos a mano, las entradas podrían ser los valores de píxeles de la imagen.
- Pesos: Cada entrada x_n se multiplica por su peso correspondiente $[w_1, w_2, \ldots w_n]$. Los pesos son parámetros ajustables que determinan la importancia relativa de cada entrada para la salida del perceptrón. Inicialmente, estos pesos suelen asignarse aleatoriamente y se ajustan mediante un proceso de aprendizaje.
- Sesgo: Cada neurona agrega un valor constante denominado sesgo o *bias* el cual es representado por b en la figura 2.
- Suma ponderada: Después de multiplicar cada entrada por su peso correspondiente, se suman todos estos productos ponderados. Esta suma representa la combinación lineal de las entradas ponderadas por sus pesos.
- Función de activación: La suma ponderada se pasa a través de una función de activación, que determina la salida del perceptrón. La función de activación introduce no linealidades en el modelo y permite al perceptrón capturar relaciones más complejas entre las entradas y la salida. Existen varias funciones de activación tales como sigmoide, ReLU, entre otras. La función de activación que se utiliza se debe elegir en base al: tipo de problema que se desea resolver y mediante la experimentación, es decir, cual función da mejores resultados, para cierta tarea es algo que se debe observar y obtener durante el proceso de entrenamiento.
- Salida: La salida del perceptrón se determina por el resultado de la función de activación. En el caso de la función escalón, la salida puede ser una clase binaria que indica si la entrada pertenece a una categoría específica o no.

El perceptrón, o la red en su conjunto, se entrena ajustando los pesos y los sesgos de manera que la o las salidas del modelo se acerquen lo más posible a los valores deseados, para un conjunto de *datos de entrenamiento* dados. Esto se logra mediante algoritmos de aprendizaje, siendo el más típico el algoritmo de *descenso de gradiente*, que permiten ir ajustando gradualmente los pesos a partir de la definición de una *función de costo o función de pérdida* la cual permite medir y minimizar la discrepancia entre la/las salida/s que entrega la red neuronal y los valores que se están usando como ejemplos para el entrenamiento. Este proceso se repite iterativamente hasta que la red aprende a generar correctamente los datos de salida, con un margen de error considerado como aceptable. En la figura 3, se puede ver un ejemplo específico de una red simple, ya que no cuenta con capas ocultas y la descripción de esta red se resume en la tabla 1.

Ítem	Número de parámetros	Tipo de parámetro
Número de entradas	2	Features (características)
Número de salidas	1	Ground Truth, Label (valor real, etiqueta)
Capa de entrada con 3 neuronas	6 weigths + 3 bias	weigths, bias (pesos, sesgos)
Capa de salida con 1 neuronas	3 weigths + 1 bias	weigths, bias (pesos, sesgos)

Tabla 1: Descripción de parámetros de la red de la figura 3

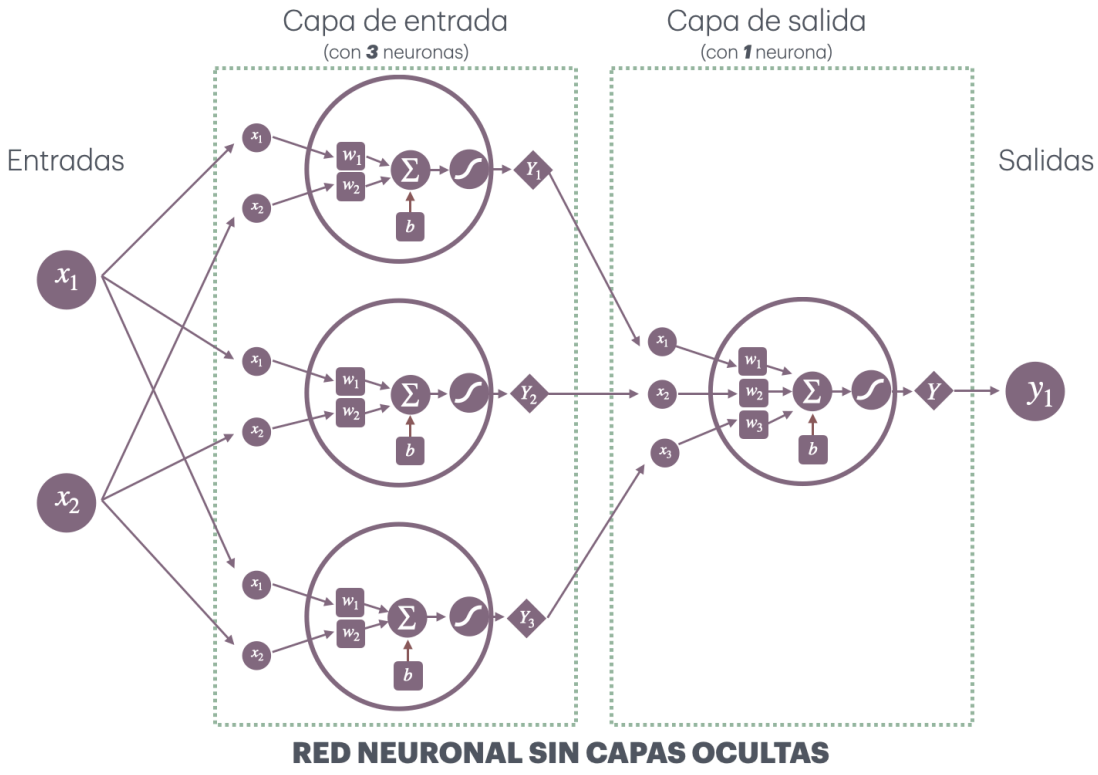

Figura 3: Red neuronal artificial simple (sin capas ocultas)

En la figura 4, se puede ver un ejemplo específico de una red profunda, ya que cuenta con una capa oculta, la descripción de esta red se resume en la tabla 2. Debe quedar claro que una red neuronal como mínimo necesita 2 capas, una para las entradas y otra para las salidas, sin embrago, en la medida que se van aumentando el número de capas ocultas o intermedias también aumentará la cantidad de variables a almacenar en memoria, aumentará el tiempo de cómputo para entrenar la red y en definitiva se incrementarán todos los recursos que son necesarios para lograr que la red neuronal trabaje eficientemente en la tarea definida.

Ítem	Número de parámetros	Tipo de parámetro
Número de entradas	2	Features (características)
Número de salidas	1	Ground Truth, Label (valor real o etiqueta)
Capa de entrada con 3 neuronas	6 weigths + 3 bias	weigths, bias (pesos, sesgos)
Capa oculta con 2 neuronas	6 weigths + 2 bias	weigths, bias (pesos, sesgos)
Capa de salida con 1 neurona	2 weigths + 1 bias	weigths, bias (pesos, sesgos)

Tabla 2: Descripción de parámetros de la red de la figura 4

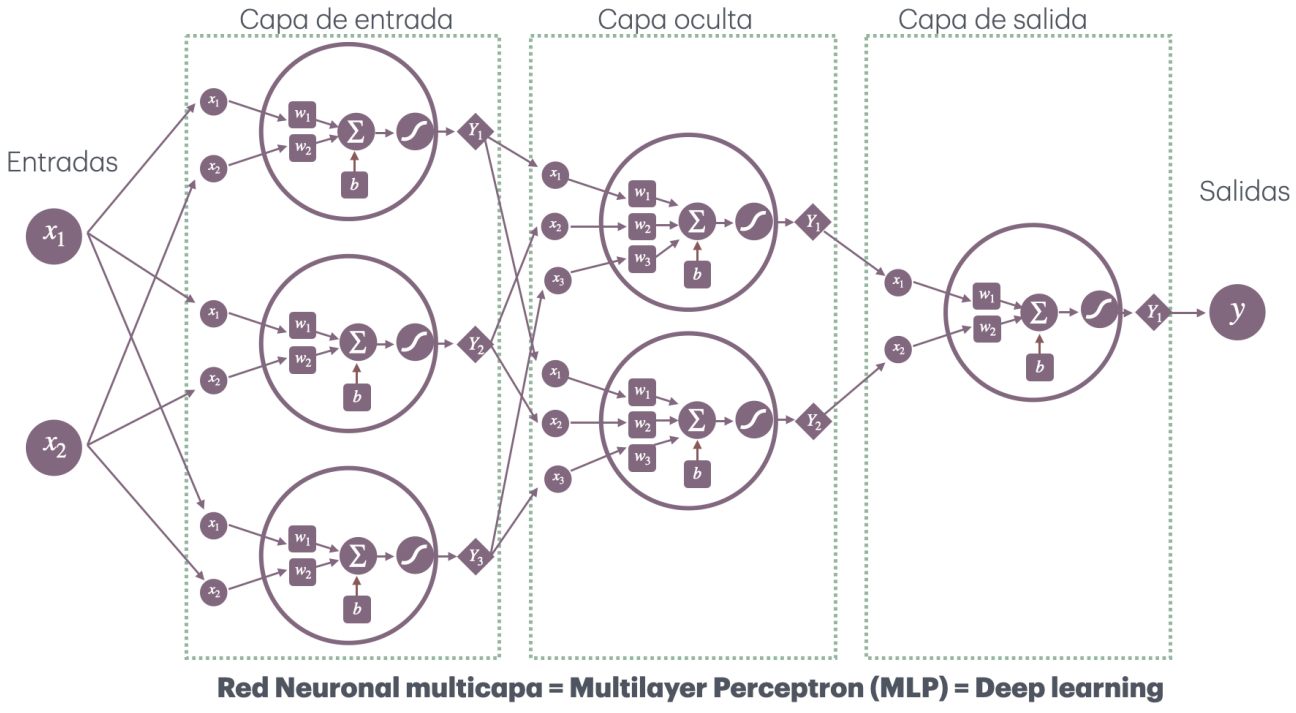

Figura 4: Red neuronal artificial multicapa

ANN y Feedforward

Una red de neuronas artificiales o *ANN* (*Artificial Neural Network*) suele utilizar una red *feedforward*, este tipo de red en una de las más comúnmente utilizadas en el campo del aprendizaje profundo y las redes neuronales artificiales. Su nombre *feedforward* proviene del hecho de que la información fluye en una sola dirección, desde la capa de entrada a través de una o más capas ocultas (si las hay) hasta la capa de salida, sin retroalimentación o conexiones cíclicas. Como se puede ver en la figura 5, en una red *feedforward* típica, las neuronas están organizadas en capas, donde cada neurona en una capa está conectada a todas las neuronas de la capa siguiente, formando así un patrón de "alimentación hacia adelante". La primera capa se conoce como la capa de entrada, la última como la capa de salida y cualquier capa en medio se denomina capa oculta. Cada conexión entre neuronas está asociada con un peso que modula la influencia de la neurona de entrada en la neurona de salida.

El proceso de propagación hacia adelante implica que las entradas se pasen a través de la red capa por capa, donde cada una realiza operaciones matemáticas con las entradas y las transforma en representaciones progresivamente más abstractas y complejas. Finalmente, la salida de la última capa se produce como la respuesta de la red a la entrada dada. Una vez que se calcula la salida de la red, se compara con la salida deseada (etiqueta o valor objetivo) y se utiliza una función de pérdida para medir la discrepancia entre la salida real y la deseada. Luego, mediante técnicas de optimización como el descenso de gradiente, se ajustan los pesos de las conexiones en la red para minimizar esta función de pérdida y mejorar la capacidad de la red para producir salidas más precisas.

Las redes feedforward se utilizan en una variedad de tareas de aprendizaje automático, como clasificación, regresión, reconocimiento de patrones y más. Aunque son relativamente simples en comparación con otros tipos de redes neuronales, como las redes neuronales recurrentes (*RNN*) o las redes neuronales convolucionales (*CNN*), siguen siendo fundamentales y pueden ser muy efectivas en una amplia gama de aplicaciones.

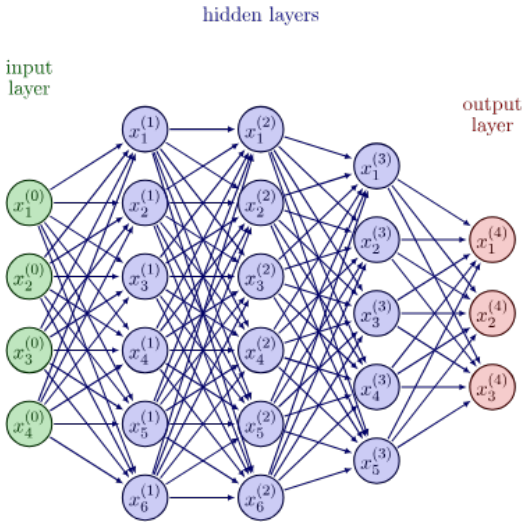

Figura 5: Red tipo *feedforward* (imagen de la página de Deep-mind)

Teorema Universal de Aproximación

El Teorema Universal de Aproximación, también conocido como Teorema de Aproximación de Weierstrass, es un resultado fundamental en análisis matemático que establece que cualquier función continua en un intervalo cerrado y acotado puede aproximarse arbitrariamente bien por medio de polinomios. Fue demostrado por el matemático alemán Karl Weierstrass en el siglo XIX, los aportes de este trabajo fueron incorporados en los trabajos de Kurt Hornik [1] y Cybenko [2].

Formalmente, el teorema establece lo siguiente:

Dada una función $f(x)$ continua en el intervalo cerrado y acotado $[a, b]$, existe una secuencia de polinomios $\{P_n(x)\}$ tal que:

1. Para cada n, $P_n(x)$ es un polinomio de grado n.
2. Para cada $\varepsilon > 0$ (cualquier número positivo tan pequeño como se desee), existe un polinomio $P_N(x)$ en la secuencia tal que para todo x en el intervalo $[a, b]$, la diferencia $|f(x) - P_N(x)| < \varepsilon$.

En otras palabras, esto significa que dado cualquier margen de error positivo (ε), podemos encontrar un polinomio en la secuencia que se aproxime a la función $f(x)$ tan cerca como queramos dentro de ese margen de error.

Este teorema es de gran importancia tanto teórica como práctica en matemáticas y sus aplicaciones. Por un lado, muestra que los polinomios son "densos" en el espacio de funciones continuas en un intervalo dado, lo que significa que podemos aproximar cualquier función continua con polinomios. Además, proporciona una herramienta poderosa para la aproximación de funciones en la práctica, lo que es útil en áreas como la física, la ingeniería y la computación, donde a menudo se necesita aproximar funciones complicadas con polinomios simples para facilitar el cálculo y el análisis.

Para una explicación formal más completa ver el material *Universal Approximation Theorem* de Deep-Mind en y los documentos dispuestos en la bibliografía

Cálculos con una neurona

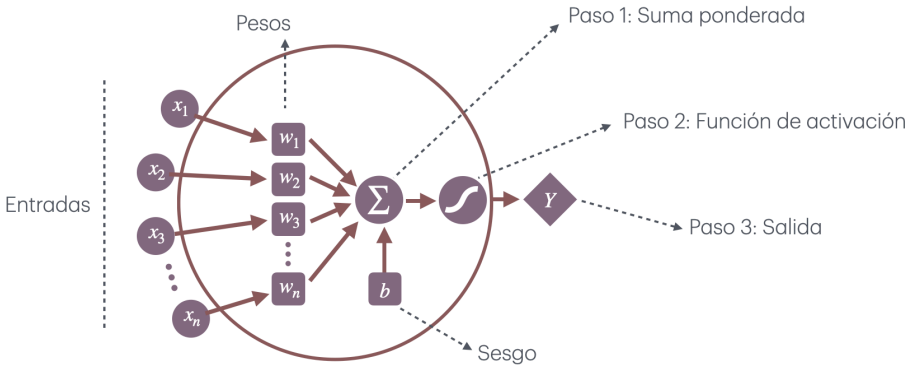

Figura 6: Esquema de una neurona artificial.

Para obtener el valor de salida Y de la neurona de la figura 6, se realizan las operaciones mostradas en la ecuaciones (1), (2), (3):

$$\sum = (x_1 \,.\, w_1 + x_2 \,.\, w_2 + \ldots x_n \,.\, w_n) + b \qquad (1)$$

$$\sigma = ReLU(x) = max(0,x) \qquad (2)$$

$$Y = \sigma\left(\sum_{1}^{n} x_n \,.\, w_n\right) \qquad (3)$$

Con $[x_1, x_2, \ldots x_n]$ los *features* o variables de entrada a la red, $[w_1, w_2, \ldots w_n]$ son los pesos o ponderaciones para cada entrada, b es el sesgo de la neurona y σ es la función de activación seleccionada, que n este ejemplo corresponde a la función ReLU.

La función ReLU (*Rectified Lineal Unit*) también es conocida como función rampa y si bien hay varias funciones de activación, la ReLU es ampliamente utilizada para variados tipos de problemas, su curva se puede ver en la figura 7.

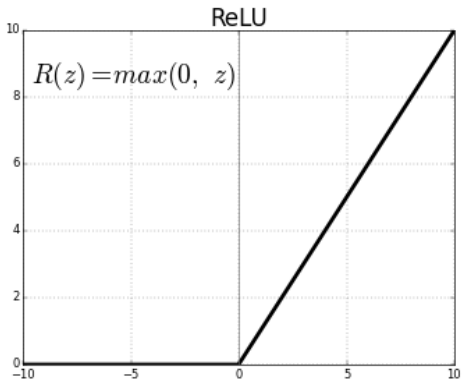

Figura 7: Función ReLU.

Cálculos en red *feedforward*

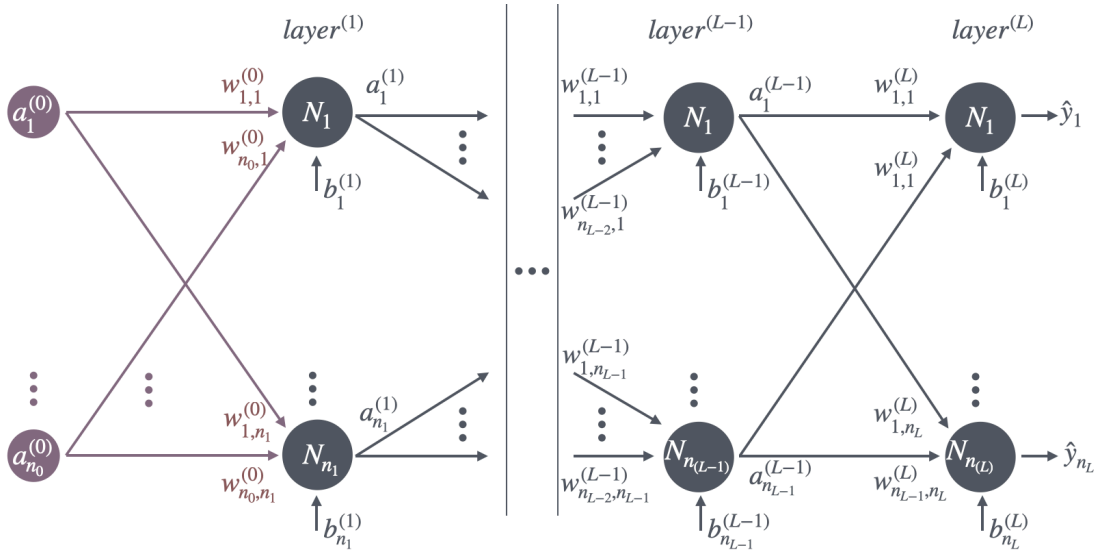

Figura 8: Esquema de una red *feedforward*.

En la figura 8 se muestra el esquema de una red con múltiples capas o *layers* y cada una con múltiples neuronas, si se toma el valor de L como el número de capas, n_i es el número de neuronas de la capa i, y n_0 es el número de características o features de entrada.

Para cada neurona de la capa i (donde $i = 1,2,3,...,L$), la salida $a_j^{(i)}$ (donde $j = 1,2...,n_i$) se calcula utilizando las siguientes ecuaciones:

- Para la i-ésima capa oculta, se tiene:

$$z_j^{(i)} = \sum_{k=1}^{n_i-1} w_{jk}^{(i)} \cdot a_k^{(i-1)} + b_j^{(i)} \qquad (4)$$

$$a_j^{(i)} = \sigma(z_j^{(i)}) \qquad (5)$$

Donde:

$z_j^{(i)}$: es el valor de entrada ponderado a la neurona j de la capa i
$w_{jk}^{(i)}$: es el peso asociado a la conexión entre la neurona k de la capa $i-1$ y la neurona j de la capa i
$a_k^{(i-1)}$: es la salida de la neurona k de la capa $i-1$
$b_j^{(i)}$: es el sesgo de la neurona j de la capa i
σ : es la función de activación

- Para la capa de salida L, se tiene:

$$z_j^{(L)} = \sum_{k=1}^{n_{L-1}} w_{jk}^{(L)} \cdot a_k^{(L-1)} + b_j^{(L)} \qquad (6)$$

$$\hat{y} = \sigma(z_j^{(L)}) \qquad (7)$$

Donde:

$z_j^{(L)}$: es el valor de entrada ponderado a la neurona j de la capa de salida.
$\hat{y}_j$: es la salida de la neurona j de la capa de salida.

Ejemplo de cálculo en forma matricial

Las matrices son herramientas muy utilizadas en álgebra lineal, lo cual aplica perfectamente para realizar los cómputos subyacentes en las redes neuronales artificiales. Las matrices permiten ordenar los datos de una forma más intuitiva en filas y columnas como se puede ver en la figura 9 . A continuación se dan ejemplos de matrices con sus respectivos tamaños:

$$m_1 = \begin{bmatrix} 6 & 8 & 1 \\ 2 & -9 & 3 \\ 4 & 5 & -1 \end{bmatrix}, \text{tamaño } 3 \times 3$$

$$m_2 = [6 \quad 8 \quad 1], \text{tamaño } 1 \times 3$$

$$m_3 = \begin{bmatrix} 6 \\ 2 \\ 4 \end{bmatrix}, \text{tamaño } 3 \times 1$$

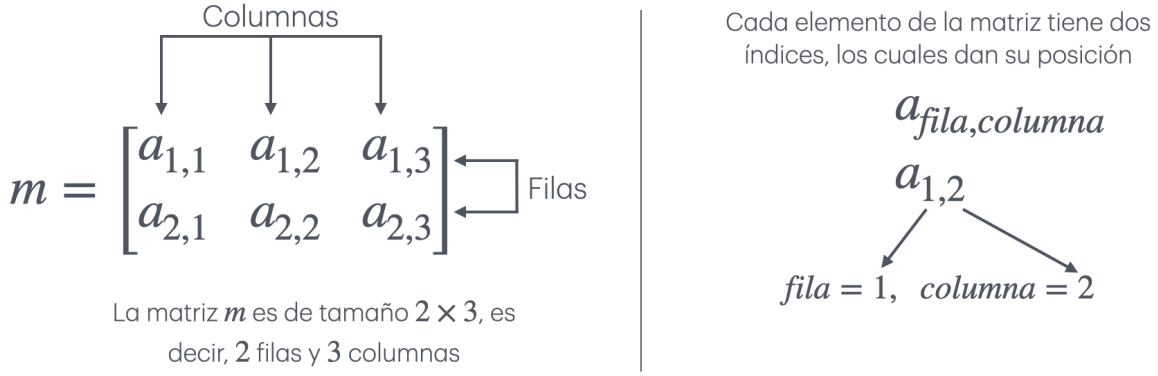

Figura 9: Simbología de matrices de dos dimensiones

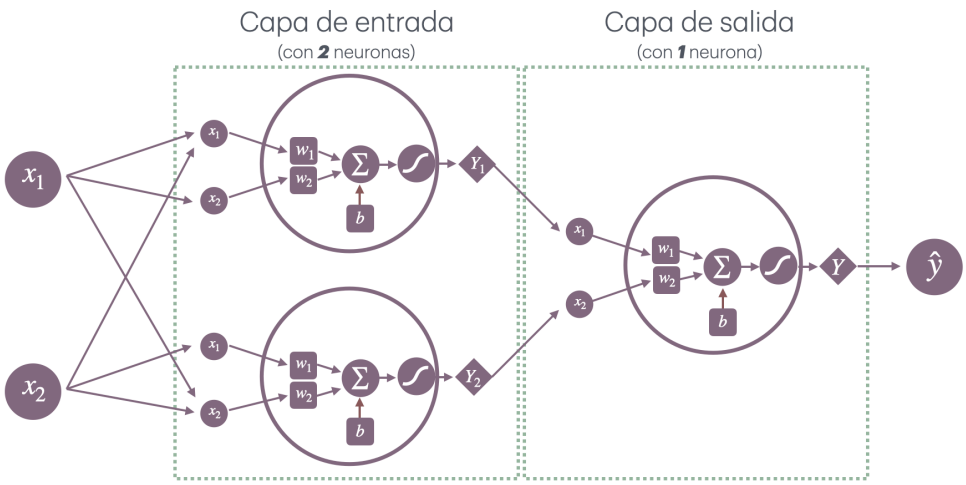

Figura 10: Esquema de una red *feedforward* con 2 entradas y 2 capas, sin capa oculta.

Si consideramos la red de la figura 10 y asumimos que los datos x_1, x_2 son dos datos de un experimento el cual se ha repetido n veces y utilizamos los datos presentados en la tabla 3 y 4 para calcular Y_1 en el experimento $Ex = 1$, se tiene:

$$X = [2 \quad 4]\,, W = [2 \quad 1], \text{ matrices de tamaño } 1 \times 2$$
$$XW = [2 \quad 4]\,[2 \quad 1] = [2 \times 2 \quad 4 \times 1] = [4 \quad 4]$$
$$Y_1 = 4 + 4 + 3 = 11$$

Luego para obtener Y_2:

$$X = [2 \quad 4], W = [3 \quad 0], \text{ matrices de tamaño } 1 \times 2$$
$$XW = [2 \quad 4]\,[3 \quad 0] = [2 \times 3 \quad 4 \times 0] = [6 \quad 0]$$
$$Y_2 = 6 + 0 + 1 = 7$$

Finalmente para obtener $\hat{y}$:

$$Y = \left[Y_1 \quad Y_2\right] = [11 \quad 7], W = [-1 \quad -2], \text{ matrices de tamaño } 1 \times 2$$
$$YW = [11 \quad 7]\,[-1 \quad -2] = [11 \times -1 \quad 7 \times -2] = [-11 \quad -14]$$
$$suma = -11 - 14 + 0 = -25$$
$$\hat{y} = \sigma(-25) = 0, \text{ si la función de activación es la ReLU}$$
$$\hat{y} = 0$$

Ex.	x_1	x_2	$\hat{y}$
1	2	4	?
2	1	3	?
3	0	2	?
…			?
n	1	3	?

Tabla 3, valores de entrada

	w_1	w_2	b
Neurona 1 capa 1	2	1	3
Neurona 1 capa 2	3	0	1
Neurona 2 capa 1	-1	-2	0

Tabla 4, valores de w y b

Aprendizaje

Existen varias teorías con respecto a como las personas logran aprender algo en particular, algunas de estas teorías vienen del campo de la psicología del aprendizaje y otras emanan desde la neurociencia. Por ejemplo Piaget en 1970 en *Genetic epistemology* postula que el aprendizaje es un proceso activo en el que los individuos construyen su propio entendimiento del mundo a través de la interacción con la información disponible y su estructura cognitiva previa. La asimilación y la acomodación son procesos clave en esta teoría siendo la interacción con la información, que proviene de nuestro entorno, un elemento común en todas las teorías de aprendizaje. Si bien esta interacción en los seres humanos es multi-sensorial. Los primeros algoritmos de aprendizaje de máquina (*Machine Learning*) solamente utilizaban datos obtenidos mediante un solo modo, por ejemplo, datos numéricos generados por sensores y transductores (sistemas de conversión de energía), datos estadísticos extraídos mediante encuestas y observación, etc. Actualmente las redes neuronales profundas, dada su alta capacidad de proceso y cómputo, son capaces de admitir en su entrada datos extraídos de modos diferentes y provenientes de diversas fuentes de información. Este desarrollo a dado paso a los de sistemas de inteligencia artificial *Multi Modales,* por ejemplo, si un modelo o redes neuronales es capaz de dar una respuesta en base a un mensaje de texto en su entrada (el clásico chatGPT usado desde algún navegador de internet) se dice que el sistema no es *multi modal* ya que solo acepta texto en su *input,* por otro lado, si el modelo puede generar una respuesta en base a texto e imagen, ambos tipos de datos ingresados simultáneamente, entonces el sistema es *multi modal.* La figura 5 es un ejemplo de uso de un modelo mono modal, esta imagen se ha generado con el "tres momentos de tiempo diferentes de un niño aprendiendo a montar una bicicleta en un parque, primero se ve que se a caído, segundo se mantiene en la bicicleta de forma inestable y con cara de preocupación, tercero cuando logra aprender se ve confiado y contento sobre la bicicleta. Las imágenes tienen el mismo fondo con colores alegres."

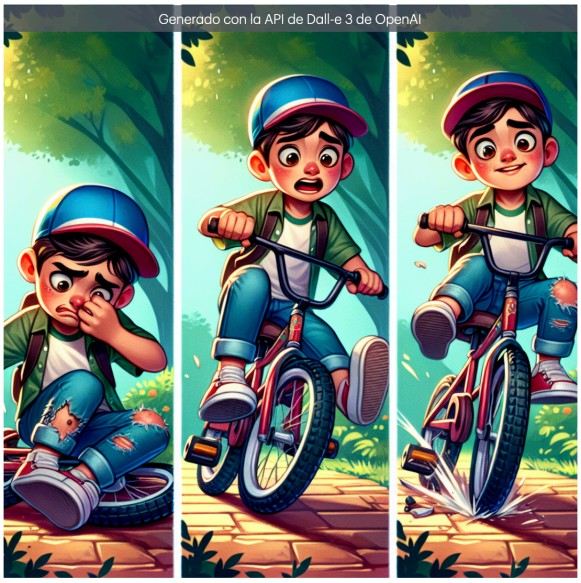

Figura 5: Aprendizaje humano, generado con Dall-e 3

Bibliografía

[1] Hornik, K., Stinchcombe, M. and White, H. (1989) 'Multilayer feedforward networks are universal approximators', Neural Networks, 2(5), pp. 359–366. Available at: https://doi.org/10.1016/0893-6080(89)90020-8.

[2] Cybenko, G. (1989) 'Approximation by superpositions of a sigmoidal function', Mathematics of Control, Signals, and Systems, 2(4), pp. 303–314. Available at: https://doi.org/10.1007/BF02551274.