Redes Neuronales

Este capítulo explora las redes neuronales con conexiones hacia adelante, incluyendo el Perceptrón, ADALINE, MADALINE y la red Back-Propagation. Aprenderemos sobre su arquitectura, funcionamiento y reglas de aprendizaje.

Introducción a las Redes Neuronales con Conexiones Hacia Adelante

Perceptrón

El primer modelo de red neuronal artificial, desarrollado por Frank Rosenblatt en 1958. Capaz de aprender a reconocer patrones sencillos.

Redes ADALINE

Una mejora del Perceptrón, capaz de resolver problemas de regresión lineal. Utiliza un algoritmo de aprendizaje adaptativo.

Redes MADALINE

Una red de múltiples ADALINE, capaz de resolver problemas más complejos. Se utiliza para la clasificación y la predicción.

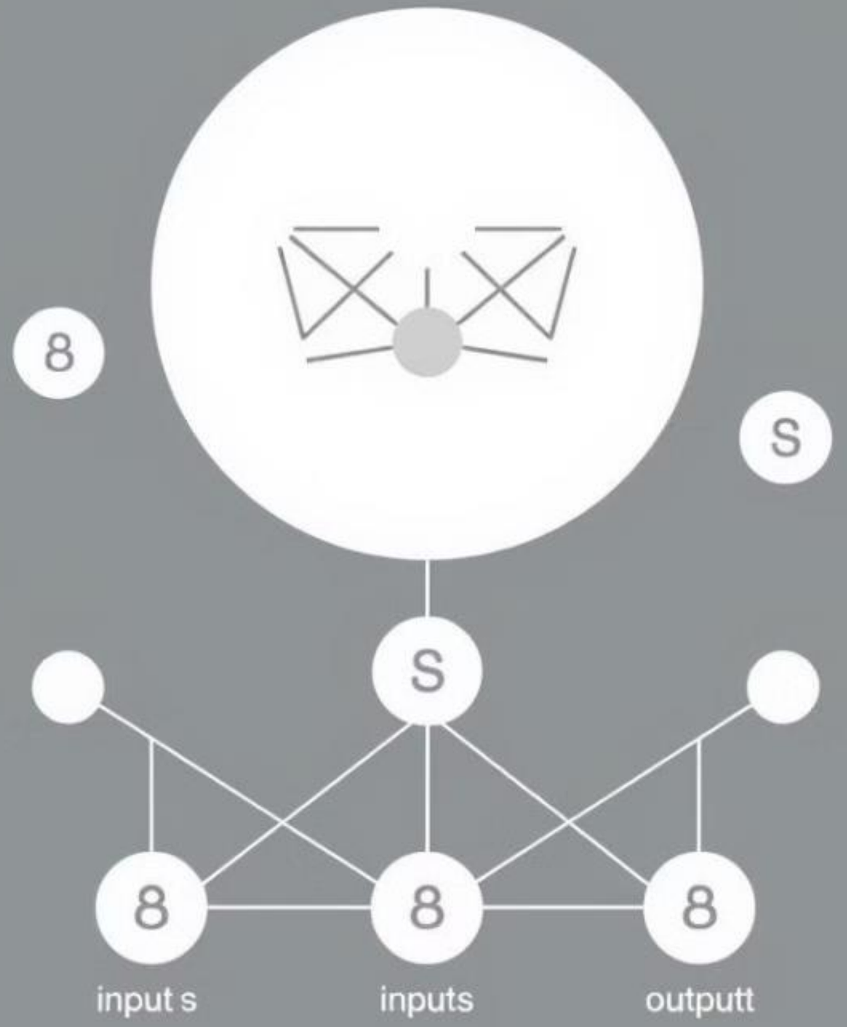
Red Back-Propagation

Un algoritmo de aprendizaje que se utiliza para entrenar redes neuronales multicapa. Permite la adaptación de los pesos en todas las capas.

El Perceptrón: Un Clasificador Lineal

El Perceptrón es un clasificador binario lineal que mapea una o más entradas en una salida deseada. Su algoritmo de aprendizaje es supervisado, donde los pesos se ajustan para minimizar el error entre la salida predicha y la salida deseada.

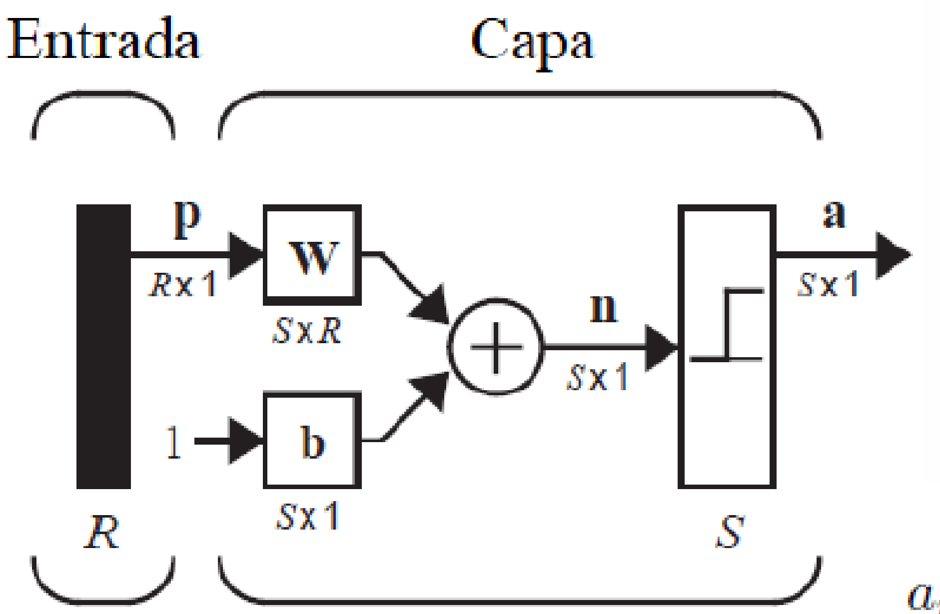
El Perceptrón sigue siendo una red rápida y confiable para problemas que puede resolver, y su comprensión es fundamental para comprender redes más complejas.



Arquitectura del Perceptrón

La red del Perceptrón se compone de neuronas interconectadas, cada una con un conjunto de pesos que determinan su influencia en la salida. La salida de la red es una función de activación aplicada a la suma ponderada de las entradas.

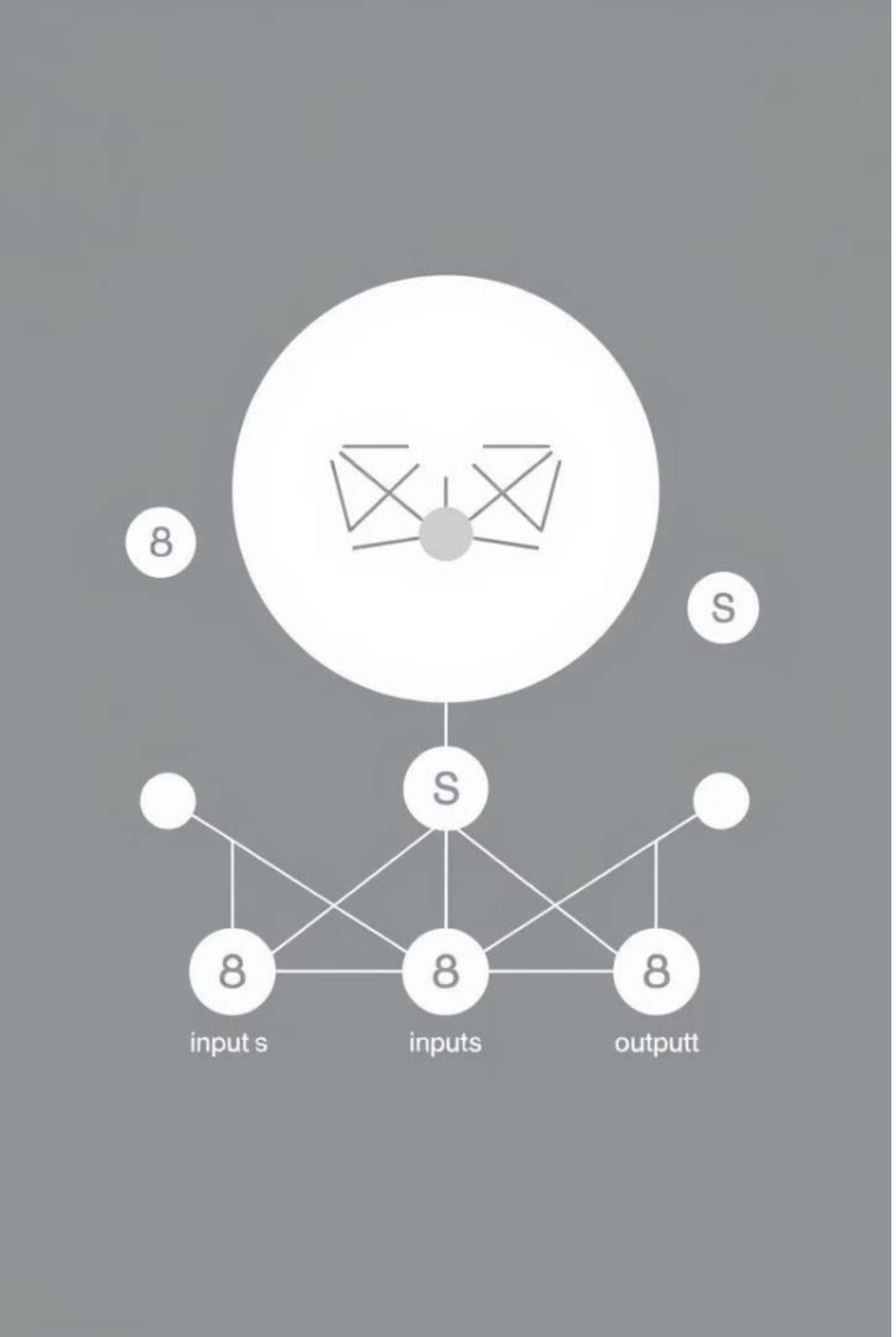
La matriz de pesos W representa las conexiones entre las entradas y las neuronas, mientras que el vector de sesgo b desplaza el límite de decisión.



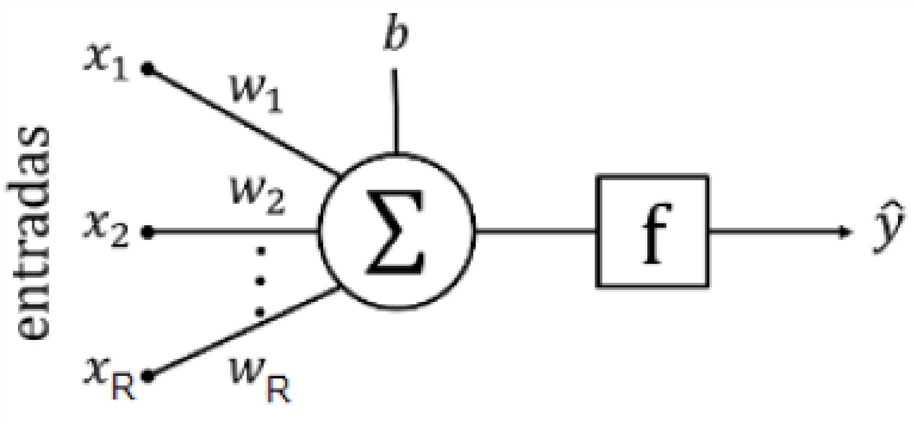
$$a = H(n) = H(Wp + b)$$

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

$$a_i = H({}_iw^T p + b_i) = f\left(\sum_i^R x_i w_i + b\right)$$



Arquitectura del Perceptrón



$$\sum_{i}^{R} x_i w_i + b$$

$$\hat{y}_i = f\left(\sum_{i}^{R} x_i w_i + b\right)$$

La estructura básica de un Perceptrón consiste en:

Entradas ($x1$,$x2$,...,xR): Son los valores de entrada que recibe la red.

Pesos (w): Son valores que multiplican a cada entrada y determinan la importancia de cada una.

Bias (b): Es un término adicional que permite ajustar la salida.

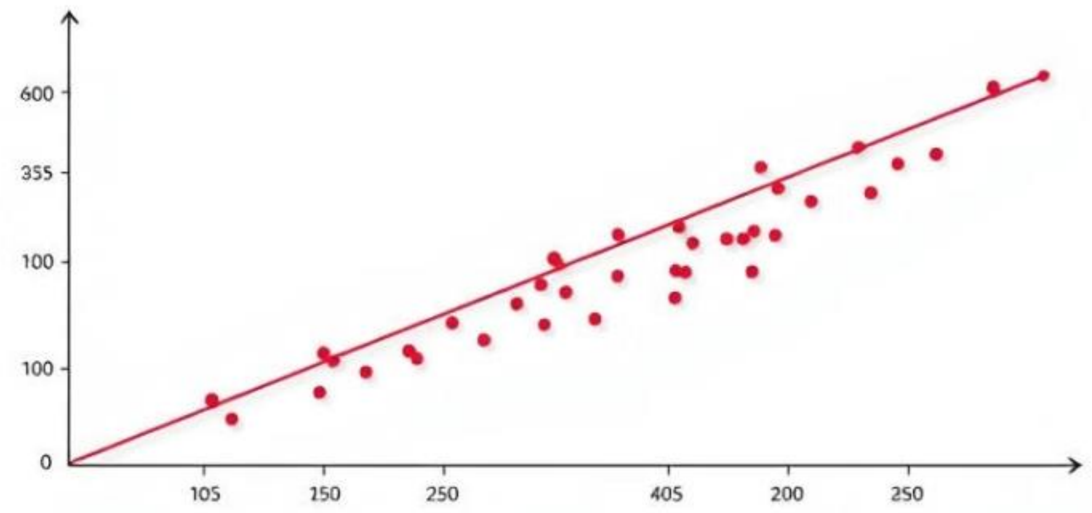
Función de activación $H(n)$: Generalmente es una función escalón, que produce una salida discreta (0 o 1).

Cada neurona divide el espacio de entrada en dos regiones separadas por una línea de decisión.

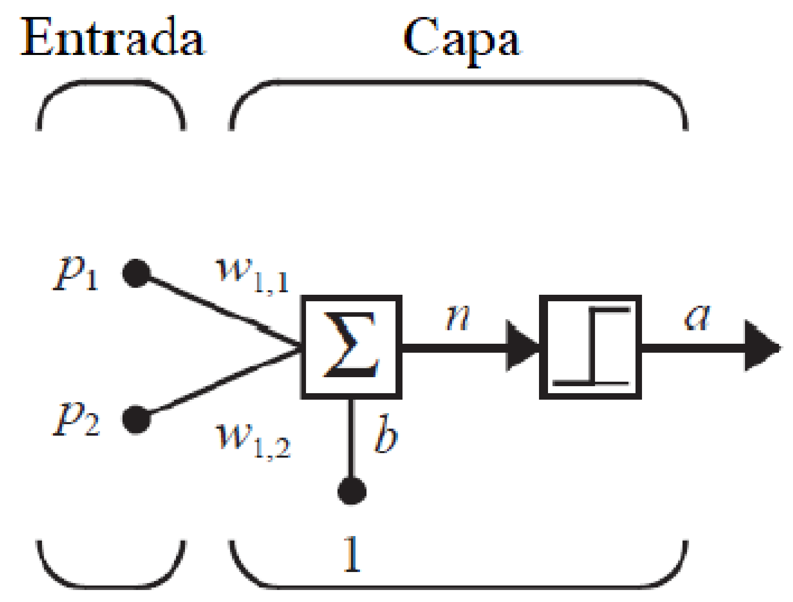
Límite de Decisión del Perceptrón

El límite de decisión del Perceptrón es una línea o hiperplano que separa el espacio de entrada en dos regiones, donde cada región corresponde a una clase de salida. La ecuación del límite de decisión se determina por los pesos y el sesgo.

El límite de decisión es ortogonal al vector de pesos, lo que significa que la dirección del vector de pesos determina la orientación del límite de decisión.



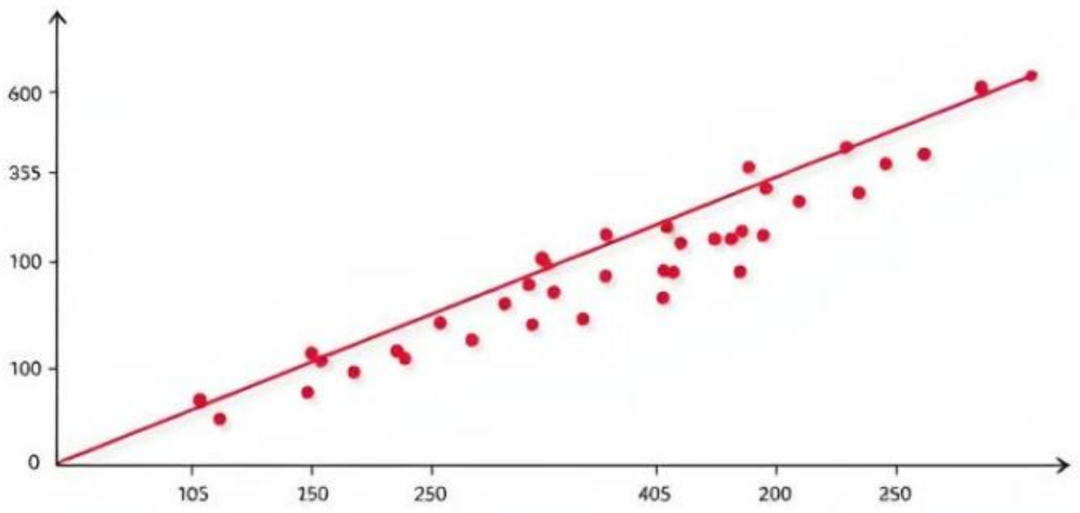
Límite de Decisión del Perceptrón



$$w_{1,1} = 1 \ , \ w_{1,2} = 1 \ , \ b = -1$$

El limite de decisión es entonces:

$$n = {}_1w^T p + b = w_{1,1}p_1 + w_{1,2}p_2 + b$$
$$= p_1 + p_2 - 1 = 0$$



La salida de esta red esta determinado por:

$$a = H(w_{1,1}p_1 + w_{1,2}p_2 + b)$$

Entonces el limite de decisión es:

$$n = {}_iw^T p + b = w_{1,1}p_1 + w_{1,2}p_2 + b = 0$$

Límite de Decisión del Perceptrón

El limite de decisión es entonces:

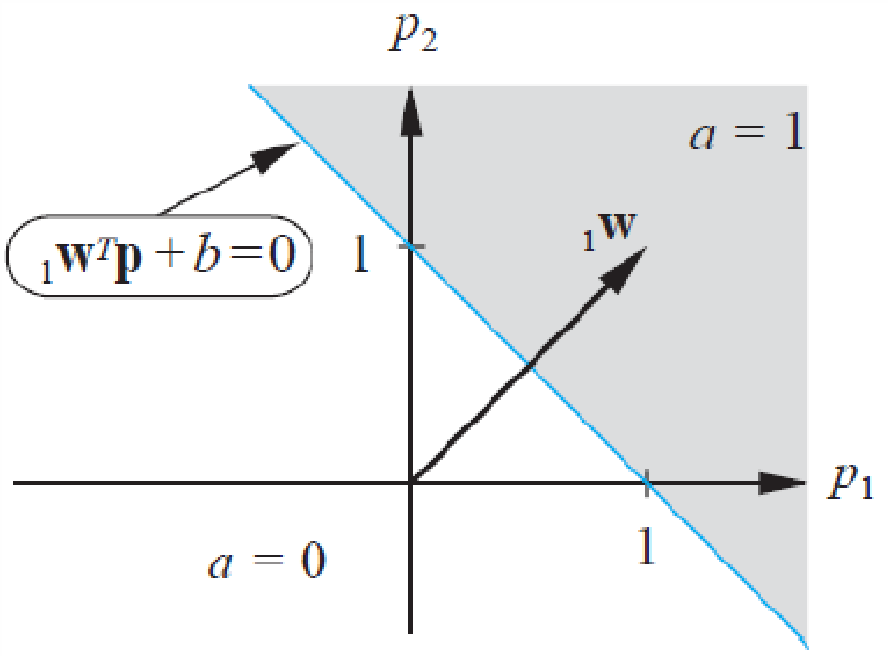
$$n = {}_1w^T p + b = w_{1,1}p_1 + w_{1,2}p_2 + b = p_1 + p_2 - 1 = 0$$

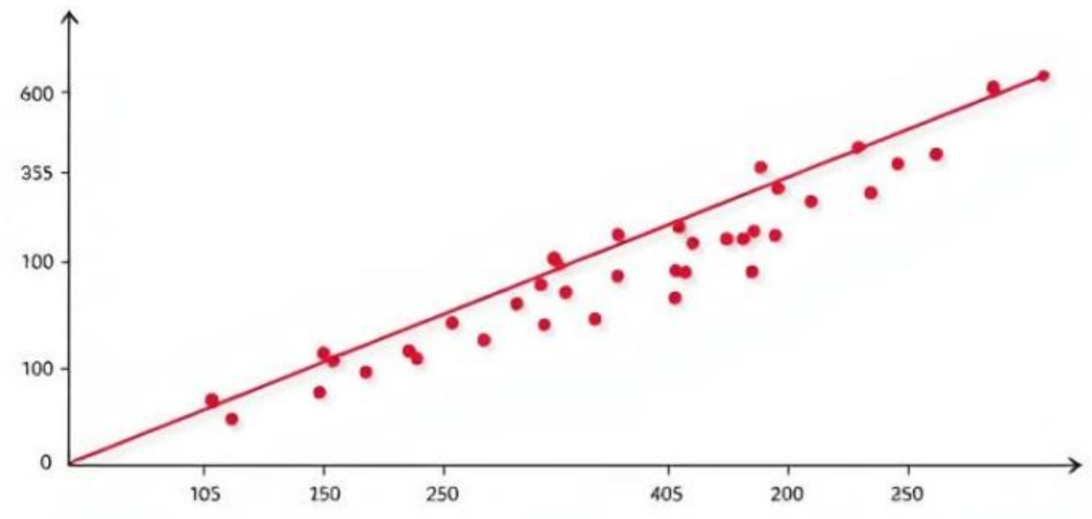
Esto define una línea en el espacio de entradas, en un lado de la línea la salida de la red será 0 y en el otro lado de la línea la salida será 1. Para dibujar la línea, se puede encontrar los puntos donde se cruza con los ejes p1 y p2.

Para encontrar la intercepción p2, se considera p1 = 0.

$$p_2 = -\frac{b}{w_{1,2}} = -\frac{-1}{1} = 1$$

$$p_1 = -\frac{b}{w_{1,1}} = -\frac{-1}{1} = 1$$

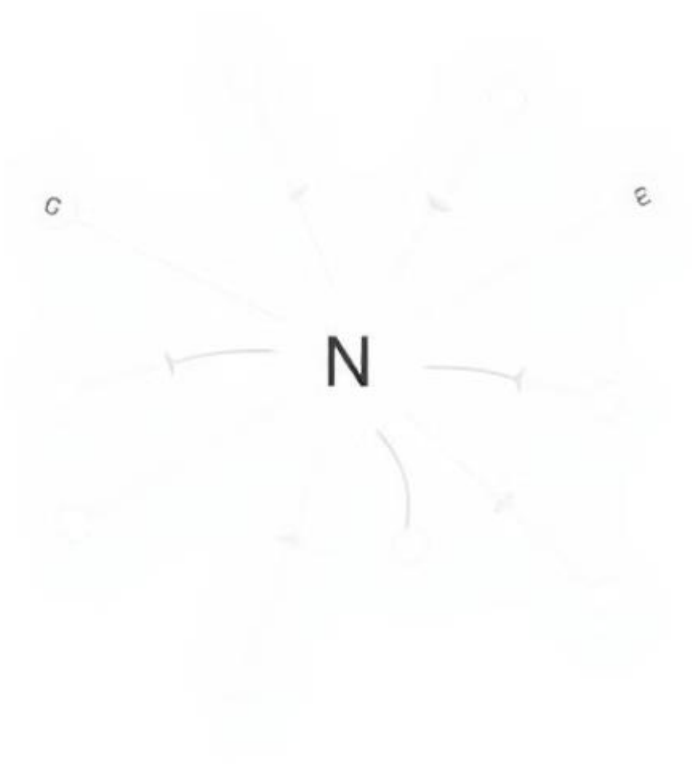




Regla de Aprendizaje del Perceptrón

La regla de aprendizaje del Perceptrón ajusta los pesos de las conexiones para minimizar el error entre la salida predicha y la salida deseada. La regla de aprendizaje se basa en la diferencia entre la salida deseada y la salida actual, y se aplica a cada peso de la red.

La taza de aprendizaje *α* controla la velocidad de aprendizaje, determinando cuánto se ajustan los pesos en cada paso de aprendizaje.



Regla de Aprendizaje del Perceptrón

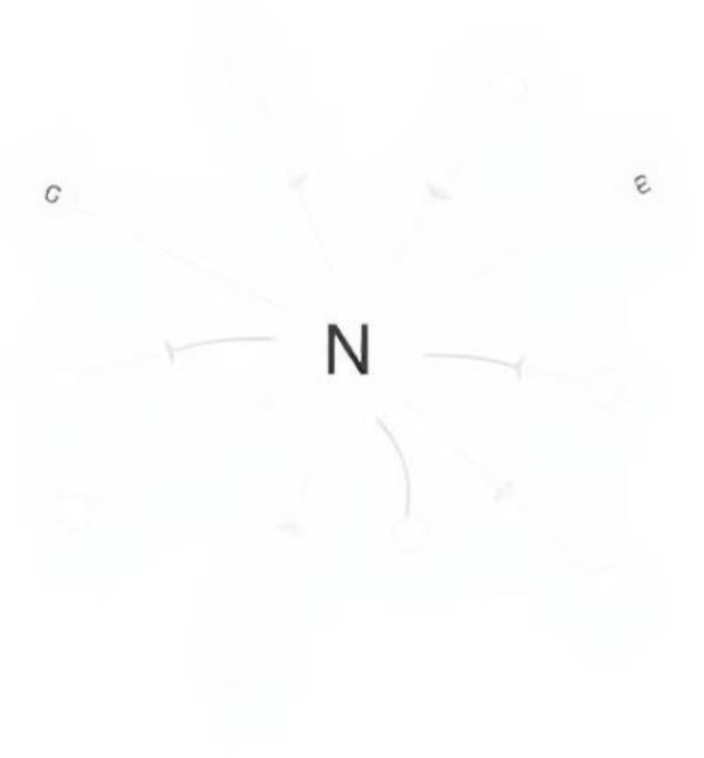
La regla de aprendizaje del perceptrón se define como:

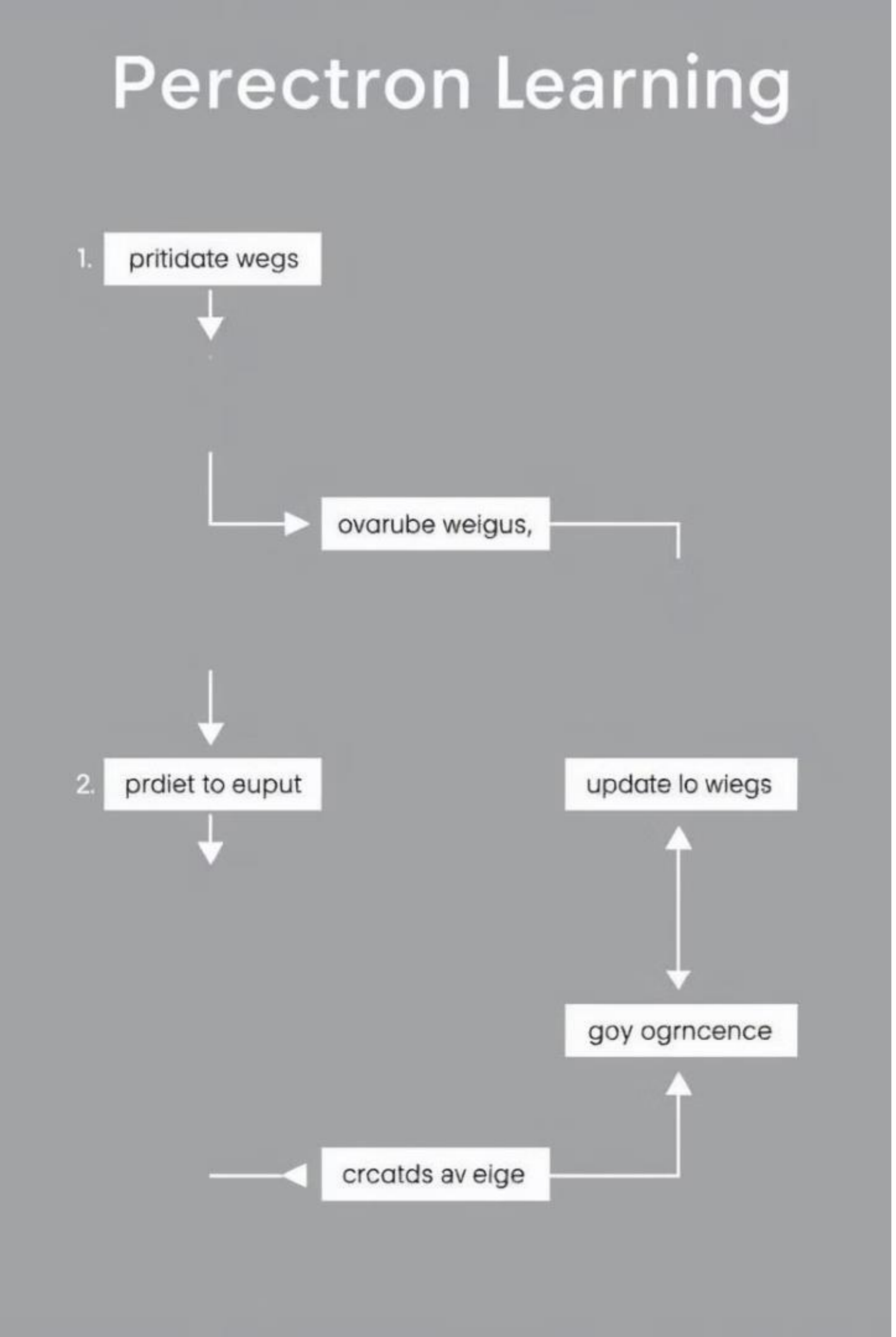
$$ {}_1w(k+1) = {}_1w(k) + \alpha e p = {}_1w(k) + \alpha(t-a)p $$

$$ b(k+1) = b(k) + \alpha e = b(k) + \alpha(t-a) $$

donde:

- $\boldsymbol{\alpha}$ es la tasa de aprendizaje ($0 < \alpha \leq 1$).
- $\boldsymbol{t}$ es la salida deseada.
- $\boldsymbol{a}$ es la salida actual.





Algoritmo de Aprendizaje del Perceptrón

1 Inicialización

Se asignan valores aleatorios a los pesos y al umbral.

$$\text{pesos } (w_i)$$

$$\text{umbral } (\theta_i)$$

2 Presentación de un Patrón

Se presenta un nuevo patrón de entrada y la salida esperada.

$$X_p = (x_1, x_2, ..., x_N)$$

$$d(t)$$

3 Cálculo de la Salida

Se calcula la salida actual utilizando la función de activación.

$$y(t) = f[\Sigma_i = w_i(t)x_i(t) + \theta]$$

4 Adaptación de los Pesos

Se ajustan los pesos utilizando la regla de aprendizaje del Perceptrón.

$$w_i(t+1) = w_i(t) + \alpha[d(t) - y(t)]x_i(t)$$

Inputs	Weights	Outputs	Weight	Outpnırs
Frat:	380.40	16.3	219	1.0
	20y, 3tx:::	17.9	160	
	20y.16	16.0	220	4.0
Errors	190.10	37.4	179	3.0
	25y, 1et:::	17.3	129	2.0
	10y.19	11.5	250	1.0
Errors:	596.10	20.9	270	1.0
Frat:	30y.10	17.5	155	1.0
	28y, 2tx:::	51.9		
Prat:	20y.19	57 9	226	1.0
Erat:	30y.10	32.5	129	1.3
	30y.10	16.9	220	

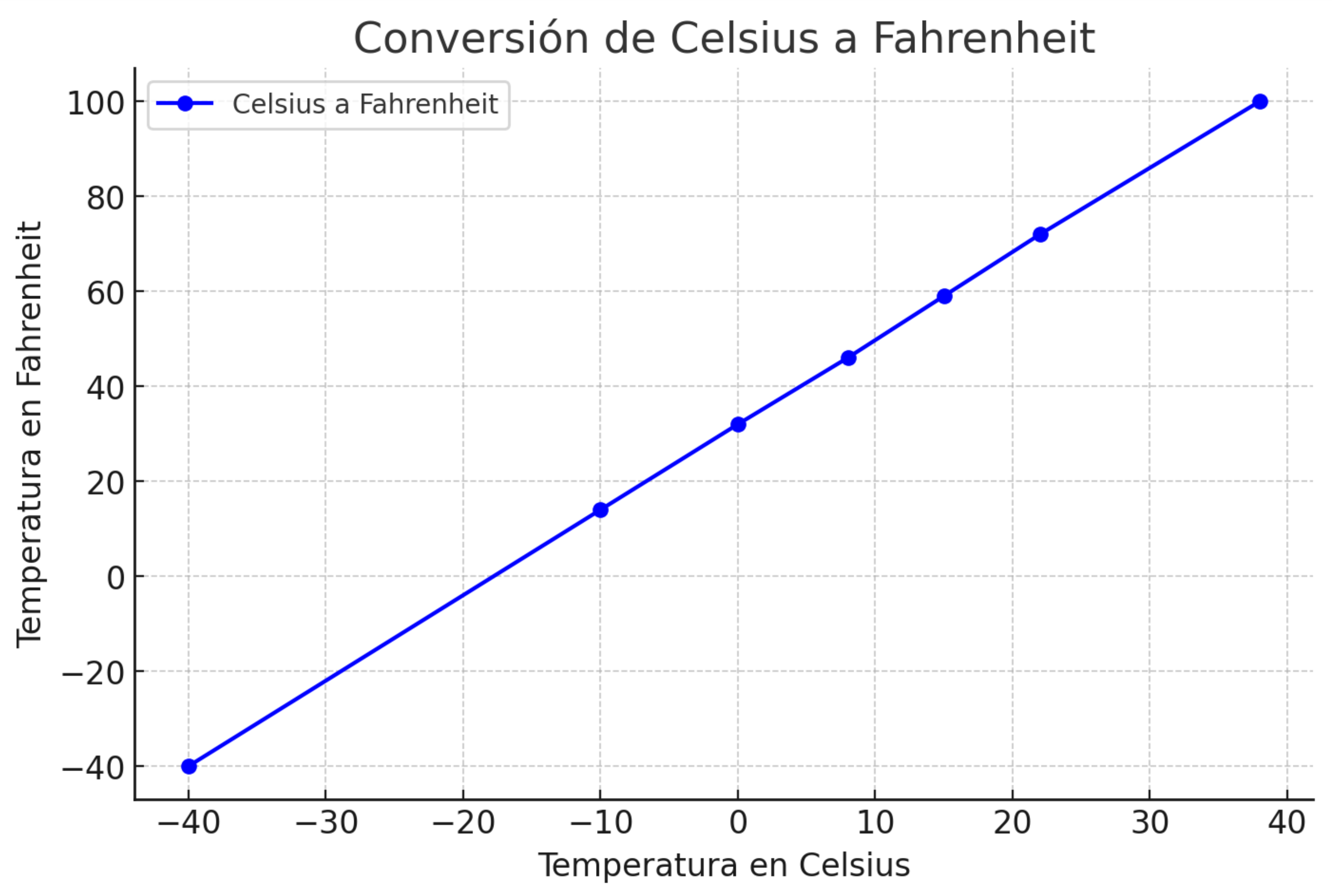
Ejemplo de Aprendizaje del Perceptrón

Este ejemplo ilustra el proceso de aprendizaje del Perceptrón para la función OR. Se muestra como se ajustan los pesos en cada paso para minimizar el error y lograr la clasificación correcta.

El algoritmo se repite hasta que el error es cero o menor que un valor preestablecido, lo que indica que la red ha aprendido la función OR.

Ejemplo. Se tiene una neurona con dos entradas, umbral y una salida, la cual debe realizar la función OR. Donde, sean inicialmente los valores elegidos aleatoriamente: w1 = 0.5, w2 = 1.5, b = 1.5 y α =1.

Ejemplo de Aprendizaje de temperatura



$$°F = (°C \times 1.8) + 32$$