

3^a
Emisión

DATA SCIENCE

Módulo 8

Introducción a las redes neuronales

Edgar Morales Palafox



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Dirección General de Cómputo y de Tecnologías de Información y Comunicación

Dirección de Docencia en TIC



**Educación
Continua
1971 - 2021**

Objetivo

- El participante identificará los elementos básicos de las redes neuronales artificiales.

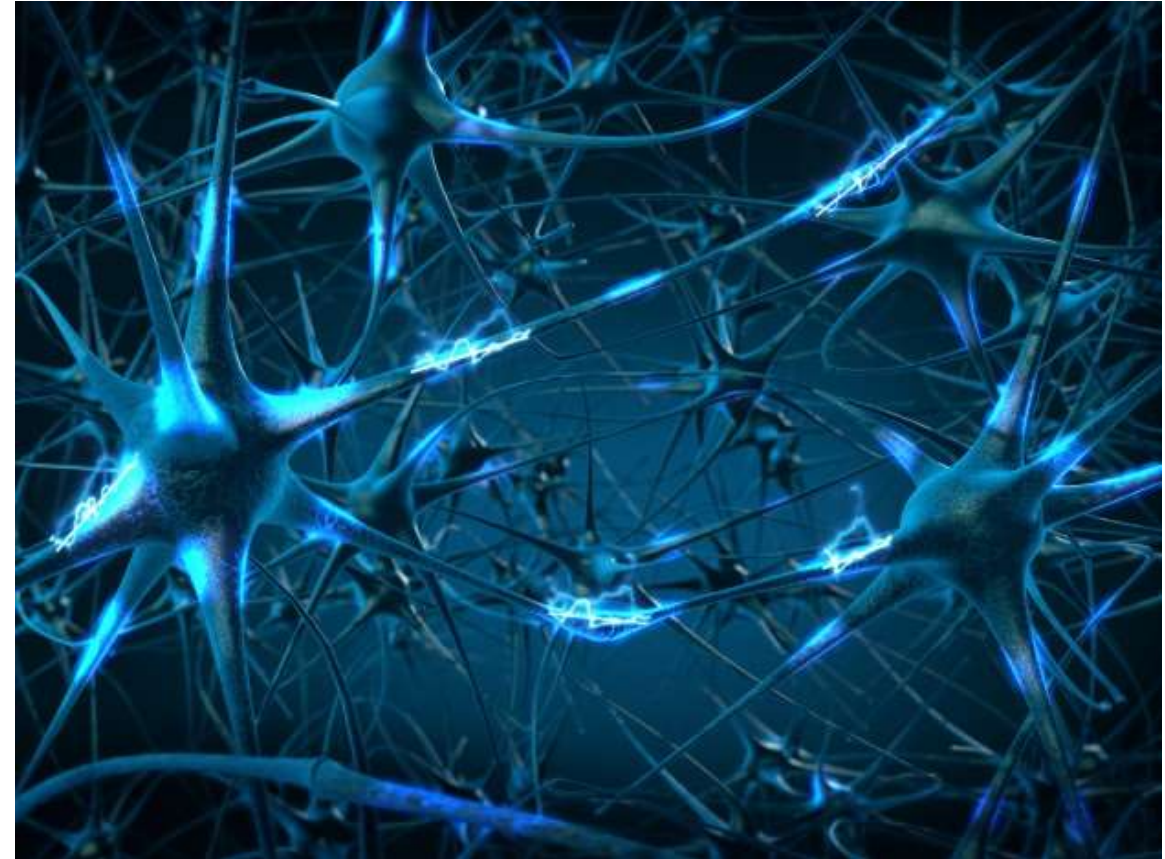
Contenido

- La neurona natural
- La neurona artificial
- Aproximación de compuertas lógicas
- Funciones de activación
- Red neuronal densa
- Funciones de pérdida

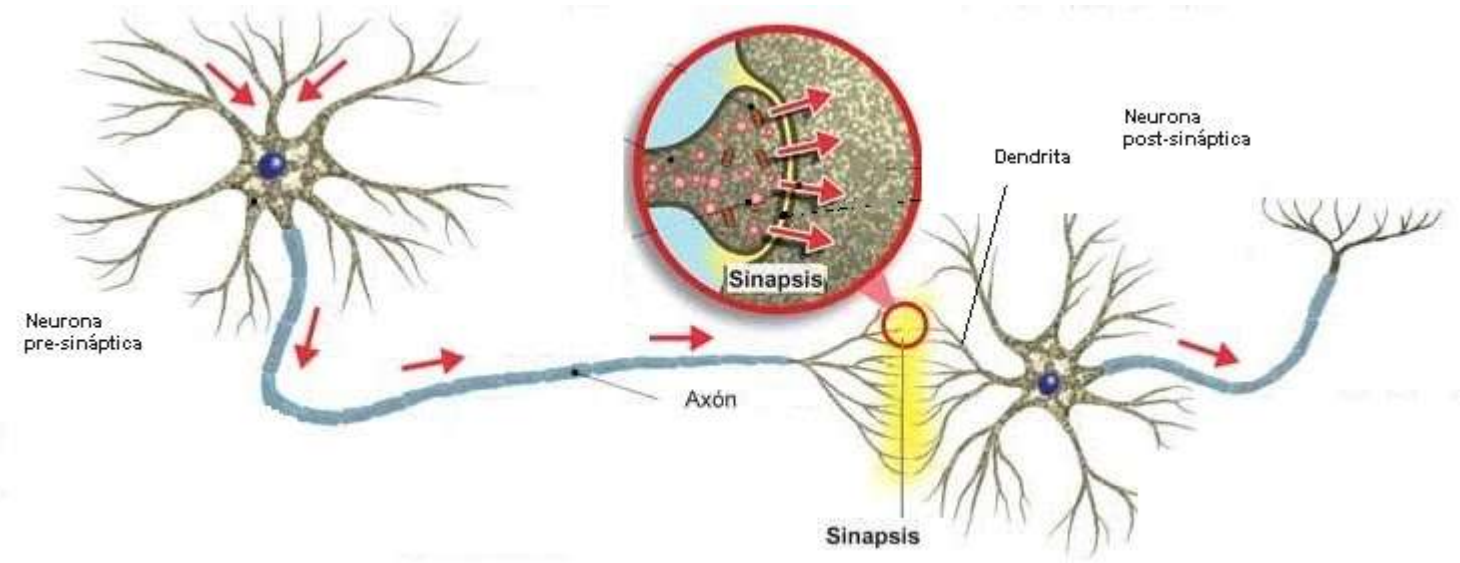
La neurona natural

Fue Ramón y Cajal (1888) quién descubrió la estructura celular (neurona) del sistema nervioso. Defendió la teoría de que las neuronas se interconectaban entre sí de forma paralela, y no formando un circuito cerrado como el sistema sanguíneo.

Una neurona consta de un cuerpo celular (soma) de entre 10 y 80 μm , del que surge un denso árbol de ramificaciones (dendritas) y una fibra tubular (axón) de entre 100 μm y un metro.

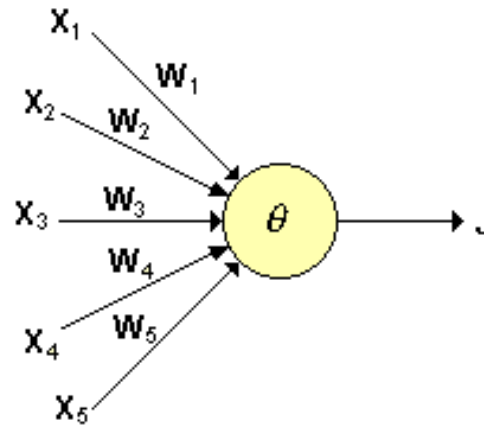
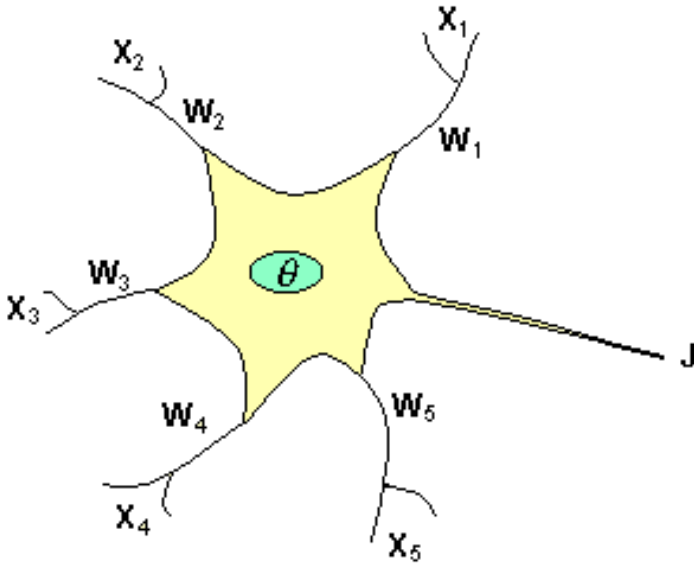


La neurona natural



La neurona artificial

Las RNA se basan en la analogía que existe en el comportamiento y función del cerebro humano, en particular del sistema nervioso, el cual está compuesto por redes de neuronas biológicas que poseen bajas capacidades de procesamiento, sin embargo toda su capacidad cognitiva se sustenta en la conectividad de éstas.



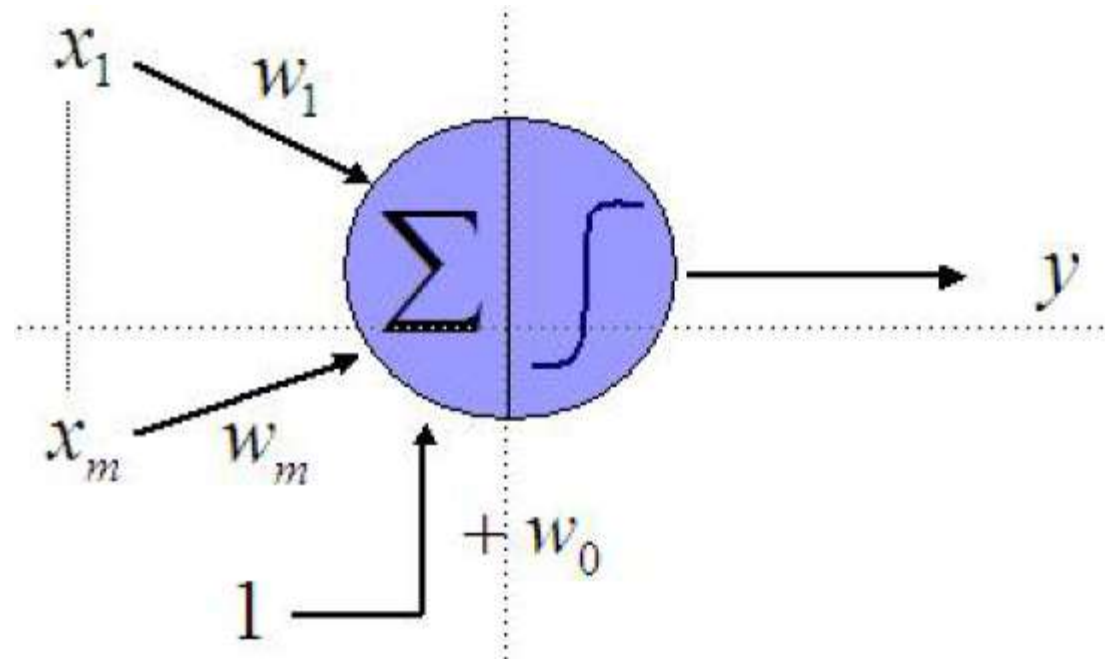
Las entradas x_i representan las señales que provienen de otras neuronas.

Los pesos w_i son la intensidad con que están conectadas dos neuronas; tanto x_i como w_i son valores reales.

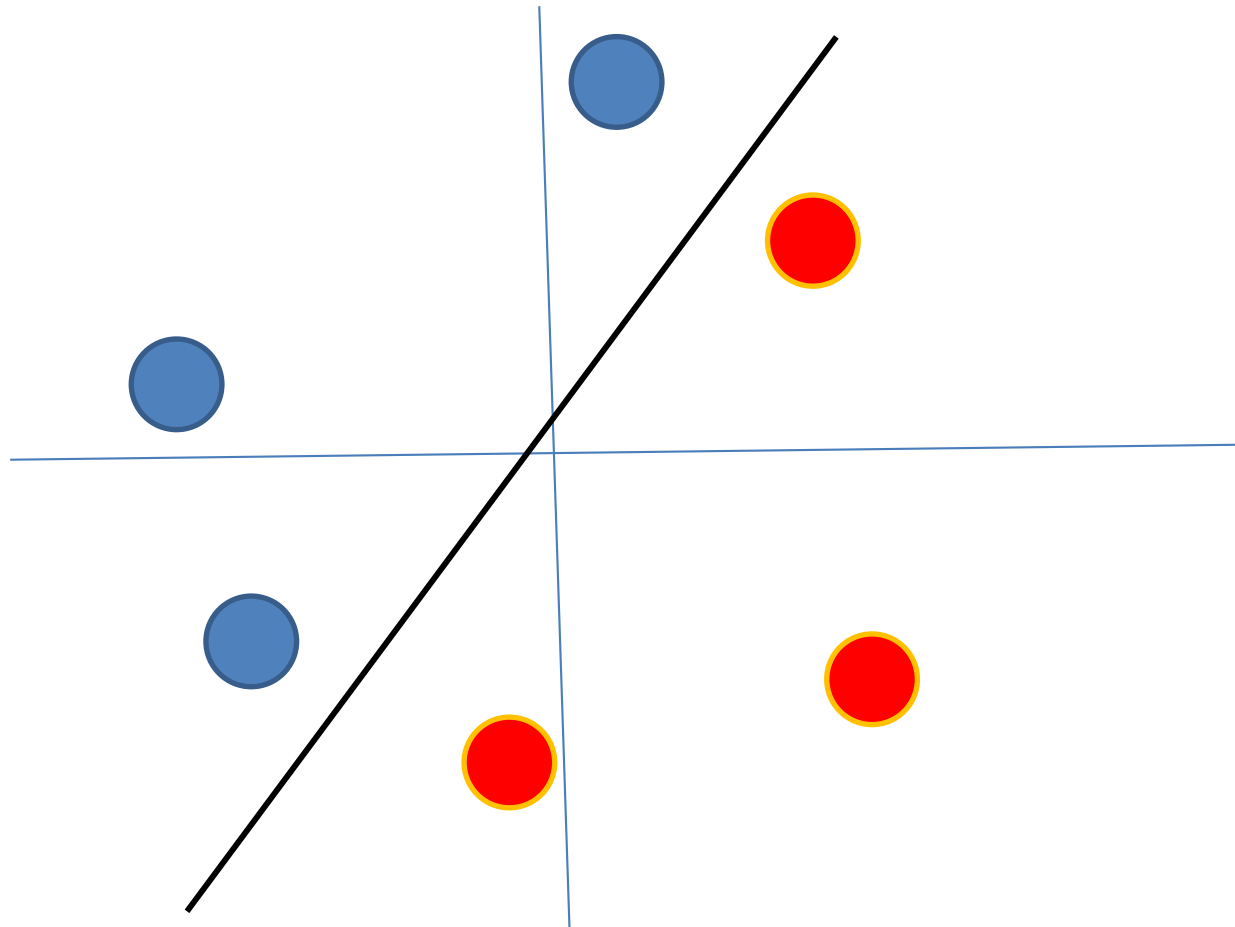
$$neta = \sum x_i w_i$$

La neurona artificial

McCulloch and Pitts en 1943 concibieron un modelo abstracto y simple de una neurona artificial, este es el elemento básico de procesamiento en una red neuronal artificial.



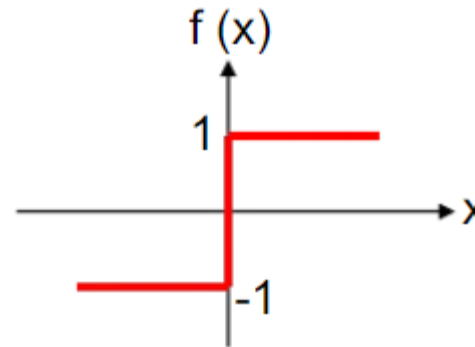
Agrupación



- ❖ La función de salida de la red es llamada función umbral o función de transferencia (tipo *hardlim*):

$$f(\text{neta}+\text{bias}) = \begin{cases} 1 & \text{si } (\text{neta}+\text{bias}) \geq 0 \\ 0 & \text{si } (\text{neta}+\text{bias}) < 0 \end{cases}$$

- ❖ También puede utilizarse una función de transferencia tipo hardlims (salidas 1 ó -1)



Algoritmo general de aprendizaje

1. Se inicializa la matriz de pesos y el valor del bias, por lo general se asignan valores aleatorios a cada uno de ellos.
2. Se presenta el primer patrón a la red, junto con la salida esperada en forma de pares entrada/salida Se calcula la salida de la red por medio de

$$\mathbf{a} = f(\sum \mathbf{X_i W_i} + \mathbf{bias}) \text{ donde } f \text{ puede ser la función } \textit{hardlim} \text{ o } \textit{hardlims}$$

3. Cuando la red no retorna la salida correcta, es necesario alterar el valor de los pesos, tratando de llevarlo hasta p y así aumentar las posibilidades de que la clasificación sea correcta, una posibilidad es adicionar p a w haciendo que el vector w apunte en la dirección de p , y de esta forma después de repetidas presentaciones de p a la red, w se aproximará a p ; este es el procedimiento adoptado para la regla de aprendizaje del Perceptrón.

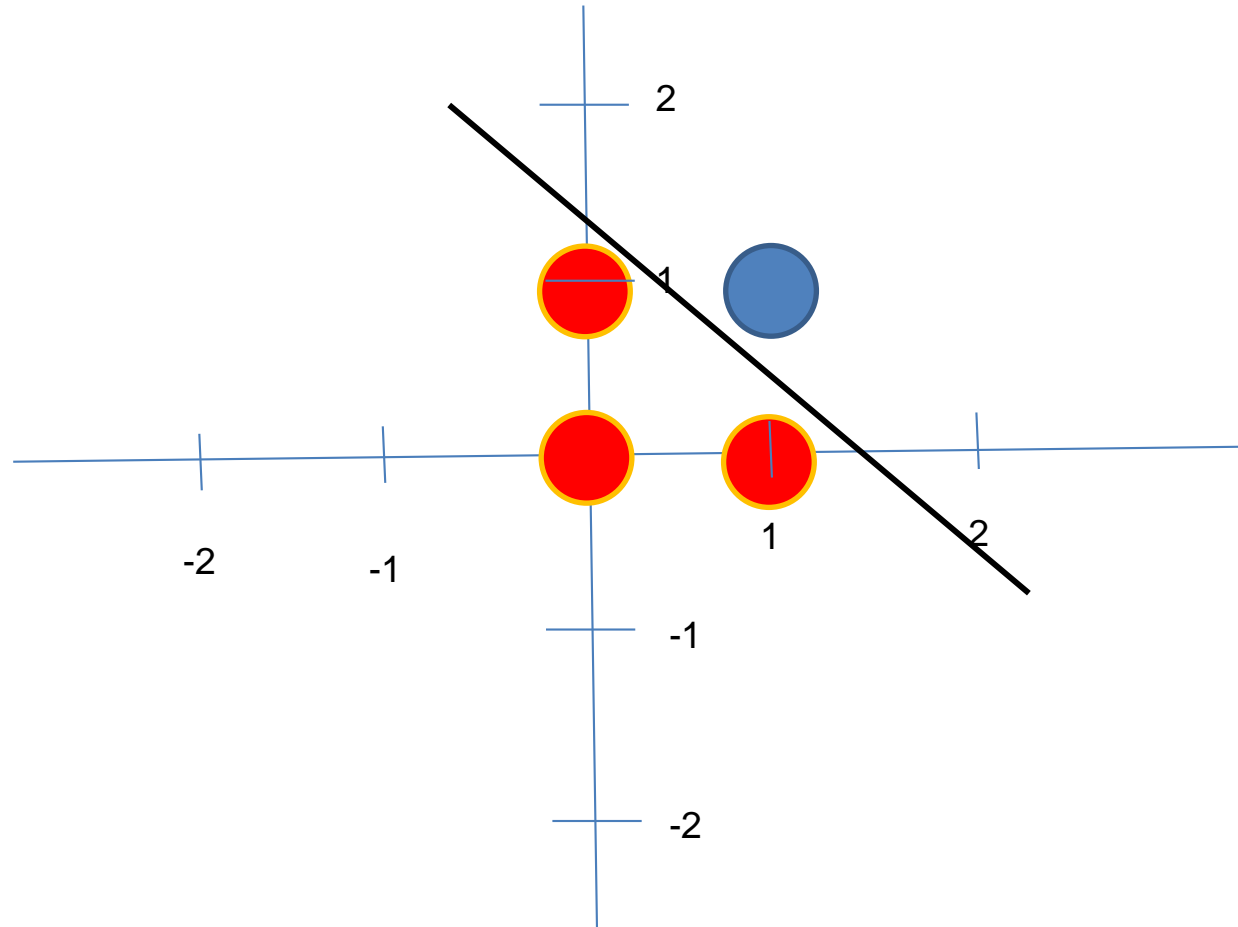
Aproximación de compuertas lógicas

Or

X1	X1	T
0	0	0
0	1	1
1	0	1
1	1	1

$W(0.5, 1.5)$ bias= 1.5

Verificar si es linealmente separable



Paso 1. Entrada $P1 = (0,0)$ $T1 = 0$

$$W = (0.5, 1.5) \text{ bias} = 1.5$$

Aplicamos la regla de aprendizaje para el patrón 1

$$\text{neta} + \text{bias} = (0 * 0.5) + (0 * 1.5) + 1.5 = 1.5$$

$$a \Rightarrow \text{hardlim}(1.5) = 1$$

$$T1 = 0 \neq a = 1$$

Por tanto es necesario ajustar los pesos:

$$e = T1 - a \Rightarrow 0 - (1) = -1$$

$$W_N = W + e * P1 = (0.5, 1.5) + -1 * (0, 0)$$

$$W_N = (0.5, 1.5)$$

$$\text{bias}_N = \text{bias} + e = 1.5 + (-1) = 0.5$$

Paso 2. Patrón 2 = (0,1) T2 = 1

$$W = (0.5, 1.5) \text{ bias} = 0.5$$

Aplicamos la regla de aprendizaje para P2

$$\text{neta} + \text{bias} = (0 * 0.5) + (1 * 1.5) + 0.5 = 2$$

$$a \Rightarrow \text{hardlim}(2) = 1$$

$$T2 = 1 = a = 1$$

Por tanto NO es necesario ajustar los pesos

Paso 3. Patrón 3 = (1,0) T3 = 1
 $W = (0.5, 1.5)$ bias = 0.5

Aplicamos la Red para P3

$\text{neta} + \text{bias} = (1 * 0.5) + (0 * 1.5) + 0.5 = 1$
 $a \Rightarrow \text{hardlim}(1) = 1$

$$T3 = 1 = a = 1$$

Por tanto NO es necesario ajustar los pesos

Paso 4. Patrón 4 = (1,1) $T4 = 1$
 $W = (0.5, 1.5)$ bias = 0.5

Aplicamos la Red para P4

$neta + bias = (1 * 0.5) + (1 * 1.5) + 0.5 = 2.5$
 $a \Rightarrow \text{hardlim}(2.5) = 1$

$(T4 = 1) = (a = 1)$

Fase de verificación

Patrón 1 (0,0) $T=0$

$$(0*0.5) + (0*1.5) + 0.5 = 0.5$$

$$a \Rightarrow \text{hardlim}(0.5) = 1$$

$$T1 = 0 \neq a = 1$$

No ha aprendido bien, por lo tanto necesita reentrenar.

Segunda iteración (Epoca)

Por tanto es necesario ajustar los pesos:

$$e = T1 - a \Rightarrow 0 - (1) = -1$$

$$WN = W + e * P1 = (0.5, 1.5) + -1 * (0, 0)$$

$$WN = (0.5, 1.5)$$

$$\text{biasN} = \text{bias} + e = 0.5 + (-1) = -0.5$$

Paso 2. Patrón 2 = (0,1) T2 = 1

$$W = (0.5, 1.5) \text{ bias} = -0.5$$

Aplicamos la regla de aprendizaje para P2

$$\text{neta} + \text{bias} = (0 \cdot 0.5) + (1 \cdot 1.5) + (-0.5) = 1$$

$$a \Rightarrow \text{hardlim}(1) = 1$$

$$T2 = 1 = a = 1$$

Por tanto NO es necesario ajustar los pesos

Paso 3. Patrón 3 = (1,0) T3 = 1
W = (0.5, 1.5) bias = -0.5

Aplicamos la Red para P3

neta+bias = $(1*0.5) + (0*1.5) + (-0.5) = 0$
a => hardlim(0) = 1

T3 = 1 = a = 1

Por tanto NO es necesario ajustar los pesos

Paso 4. Patrón 4 = (1,1) T4 = 1
W = (0.5, 1.5) bias = -0.5

Aplicamos la Red para P4

neta+bias = $(1*0.5) + (1*1.5) + (-0.5) = 1.5$
a => hardlim(1.5) = 1

(T4 = 1) = (a = 1)

Fase de verificación

Patrón 1 (0,0) $T=0$

$$(0 \cdot 0.5) + (0 \cdot 1.5) + (-0.5) = -0.5$$
$$a \Rightarrow \text{hardlim}(-0.5) = 0$$

$$T1 = 0 = a = 0$$

Paso 2. Patrón 2 = (0,1) T2 = 1

$$W = (0.5, 1.5) \text{ bias} = -0.5$$

Aplicamos la regla de aprendizaje para P2

$$\text{neta} + \text{bias} = (0 \cdot 0.5) + (1 \cdot 1.5) + (-0.5) = 1$$

$$a \Rightarrow \text{hardlim}(1) = 1$$

$$T2 = 1 = a = 1$$

Por tanto NO es necesario ajustar los pesos

Paso 3. Patrón 3 = (1,0) T3 = 1
 $W = (0.5, 1.5)$ bias = -0.5

Aplicamos la Red para P3

$\text{neta} + \text{bias} = (1 * 0.5) + (0 * 1.5) + (-0.5) = 0$
 $a \Rightarrow \text{hardlim}(0) = 1$

$T3 = 1 = a = 1$

Por tanto NO es necesario ajustar los pesos

Paso 4. Patrón 4 = (1,1) T4 = 1
W = (0.5, 1.5) bias = -0.5

Aplicamos la Red para P4

neta+bias = $(1*0.5) + (1*1.5) + (-0.5) = 1.5$
a => hardlim(1.5) = 1

(T4 = 1) = (a = 1)

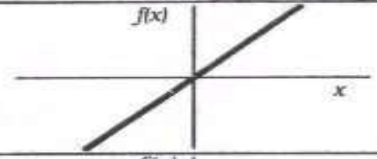
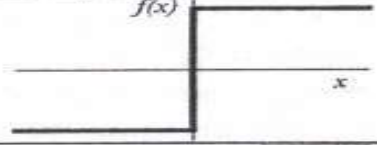
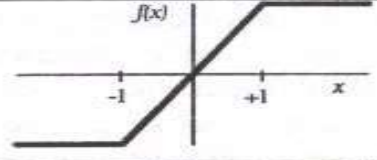
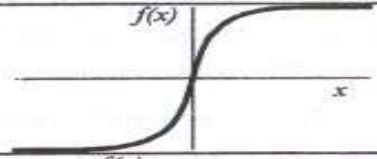
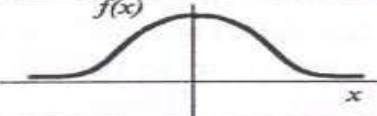
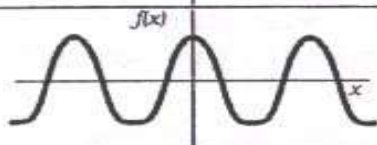
Por tanto NO es necesario ajustar los pesos

Funciones de activación

En el caso más simple, a la salida de la neurona se pueden tener dos estados: nivel alto o nivel bajo.

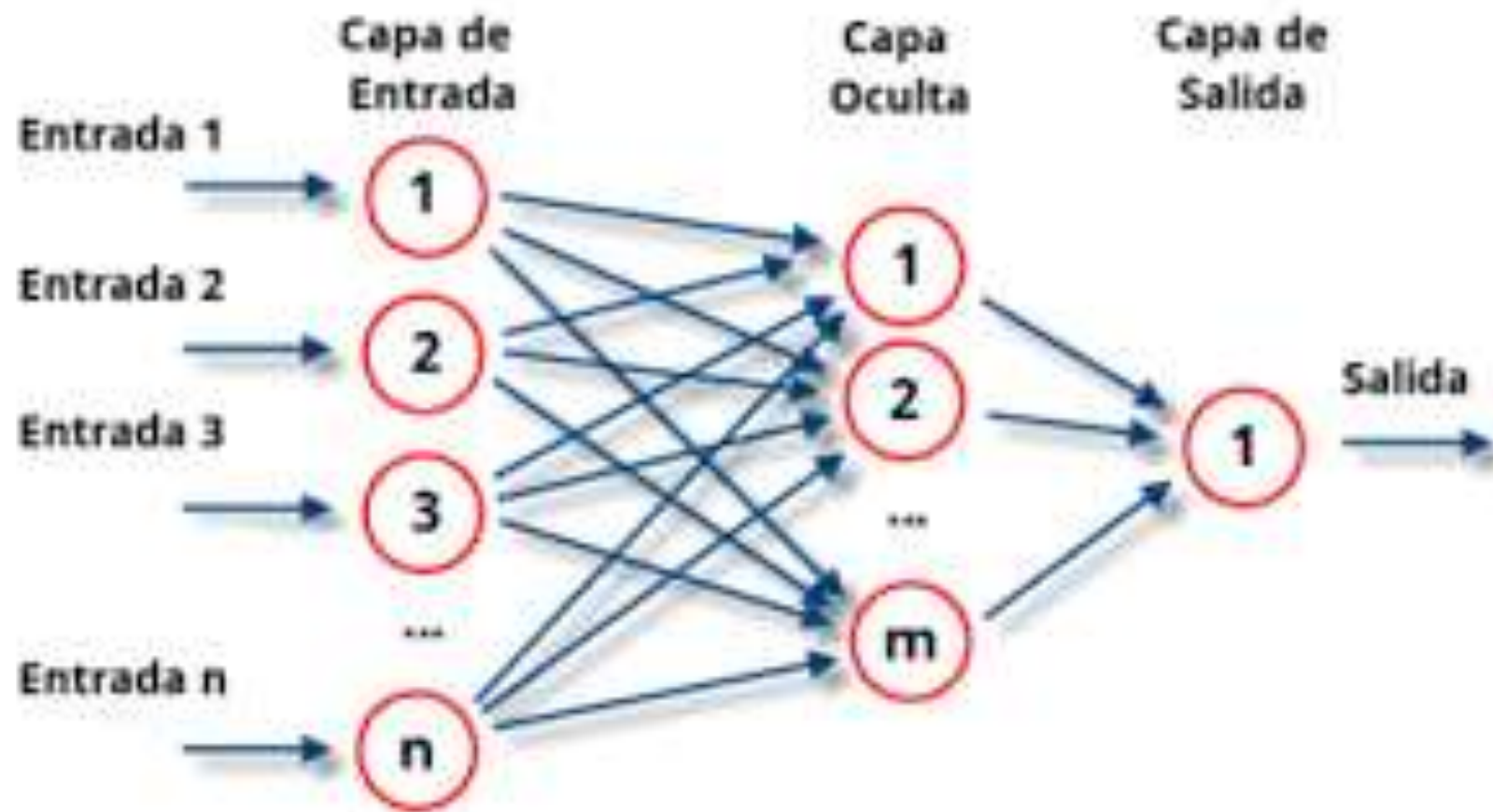
La función que implementa este tipo de discriminación se denomina función de activación dura (Escalón). El problema de esta función es que al presentar una discontinuidad no es derivable. Para evitar este inconveniente se asigna una función de activación derivable (sigmoideal o tangente) La cual se denomina función de activación blanda y supone una transición gradual de nivel bajo a alto.

La función de salida de la red es llamada función umbral o función de transferencia.

	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = \text{sign}(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & \text{si } x < -l \\ x, & \text{si } -l \leq x \leq +l \\ +1, & \text{si } x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1 + e^{-x}}$ $y = \text{tgh}(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0, +1]$	
Sinusoidal	$y = A \text{sen}(\omega x + \varphi)$	$[-1, +1]$	

Red neuronal densa

Son aquellas redes en las que todas las neuronas de una capa están conectadas a todas las neuronas de la siguiente. Estas redes, obviamente son más eficaces en cuanto al tratamiento de información porque cada neurona dispone de más datos para tratar, pero a su vez, una Red Neuronal Densa, es más lenta a la hora de procesar los datos, por la misma razón, la gran cantidad de información que mueven de una capa a otra



Funciones de pérdida

Una función de pérdida, o Loss function, es una función que evalúa la desviación entre las predicciones realizadas por la red neuronal y los valores reales de las observaciones utilizadas durante el aprendizaje. Cuanto menor es el resultado de esta función, más eficiente es la red neuronal. Su minimización, es decir, reducir al mínimo la desviación entre el valor predicho y el valor real para una observación dada, se hace ajustando los distintos pesos de la red neuronal.

Error lineal o error local

Como acabamos de indicar, el error de aprendizaje, también llamado error local, se calcula obteniendo la diferencia entre el valor real que hay que predecir y el valor predicho por la neurona artificial.

$$\text{Error} = \text{Prediccion_real} - \text{Prediccion_realizada}$$

Este es el error que buscaremos minimizar durante los aprendizajes. Este error se puede considerar como error local porque se centra en una observación dada comparando el valor real y el valor predicho

Error media cuadrática MSE o error global

El error cuadrático medio es una función de error global del aprendizaje. Es decir, que esta función nos permitirá conocer de manera global el porcentaje de error cometido por nuestra neurona.

Descenso de gradiente

El objetivo del descenso de gradiente es minimizar la función de error ajustando poco a poco los parámetros de aprendizaje representados por los distintos pesos.

El descenso del gradiente corresponde a esta metáfora y se lleva a cabo mediante el ajuste de los distintos pesos de la red neuronal hasta obtener una convergencia, es decir, un mínimo de errores. Este ajuste se hace mediante pasos cortos, usando un hiperparámetro denominado tasa de aprendizaje (*learning rate*)

Tarea

And

X1	X1	T
0	0	0
0	1	0
1	0	0
1	1	1

$W(0.5, 1.5)$ bias= 1.5

Realizar en documento Word y subir a la plataforma

Referencias

- Dayhoff, J. Neural Networks Architectures: An Introduction Van Nostrand Reinhold, New York, 1990
- Diederich, J. ed. Artificial Neural Networks: Concept Learning Computer Society Press, Los Alamitos, CA, 1990
- Fogelman-Soulie, F. ed. Automata Networks in Computer Science Princeton Univ. Press, Princeton, NJ, 1987

Ejercicio de Percepción simple

- https://colab.research.google.com/drive/1iskq_EW--kl70eEiSR7JZy8UTb0AcuGE#scrollTo=CPNIbQvIkzQo

Contacto

Edgar Morales Palafox

Doctor en Ciencias de la Computación

Edgar_morales_p@yahoo.com.mx

Tels: (55 3104 1600)