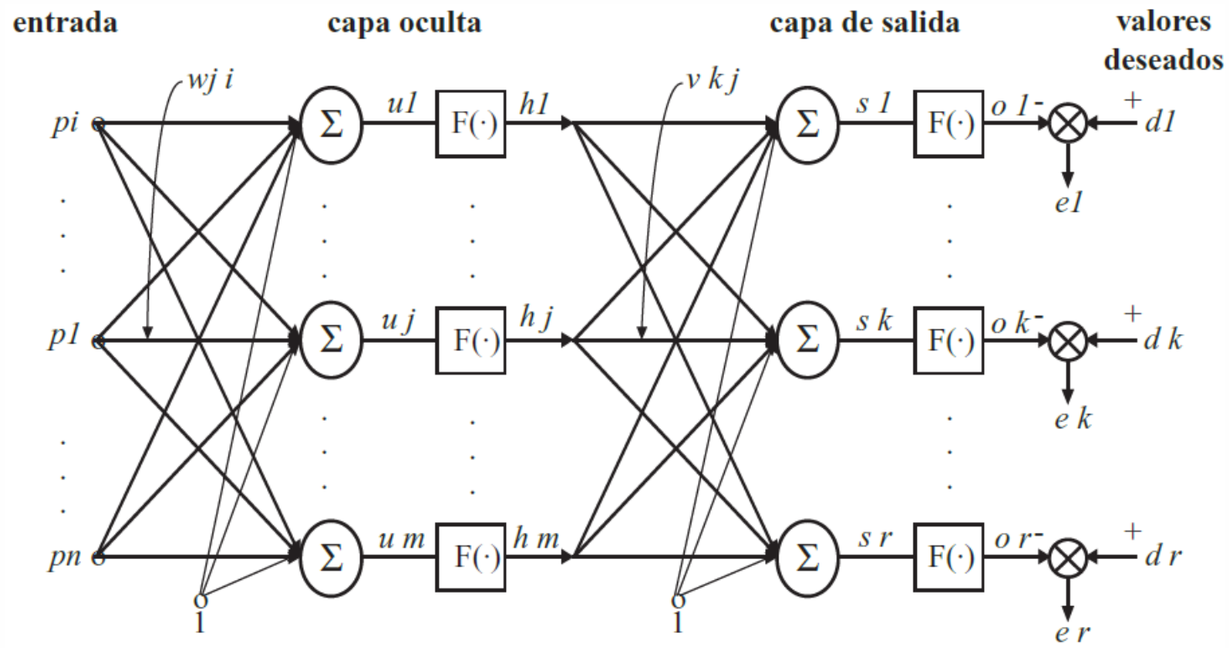
Redes Neuronales Multicapa

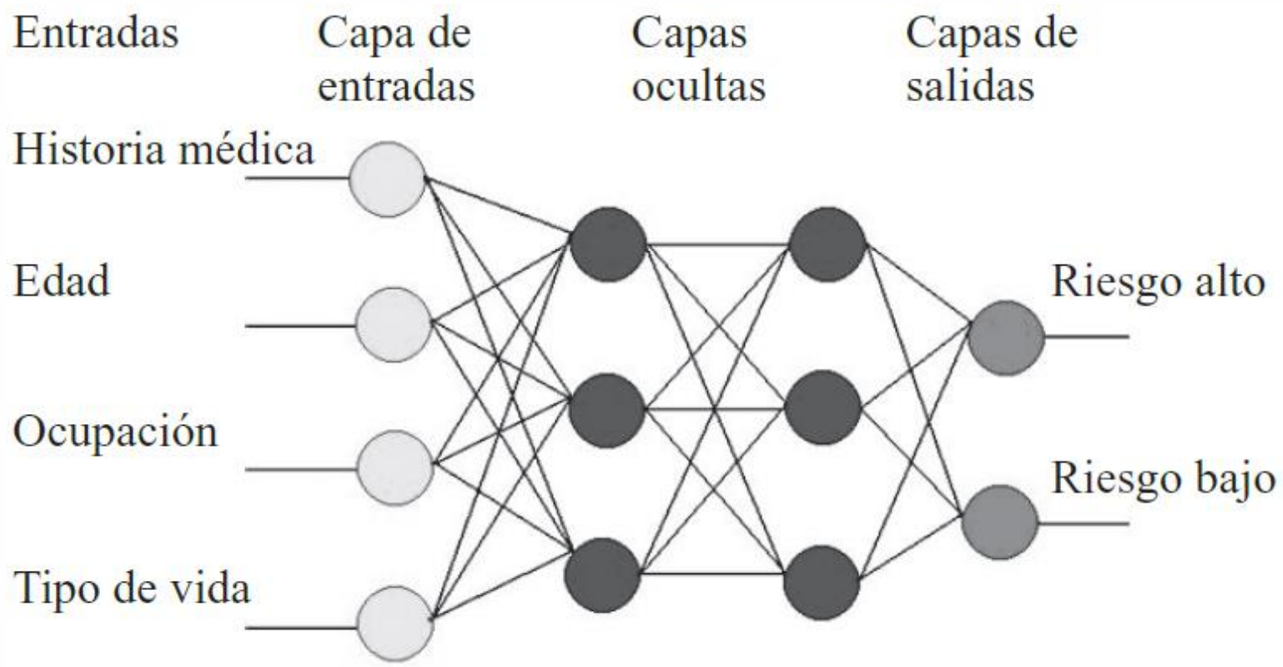
Red Neuronal Multicapa (MLP - Multilayer Perceptron): Una red neuronal multicapa es un tipo de red neuronal artificial que consiste en múltiples capas de nodos (neuronas), incluyendo una capa de entrada, una o más capas ocultas y una capa de salida.

Cada neurona en una capa está conectada a todas las neuronas de la siguiente capa, y estas conexiones tienen pesos asociados que se ajustan durante el entrenamiento..

Redes Neuronales Multicapa



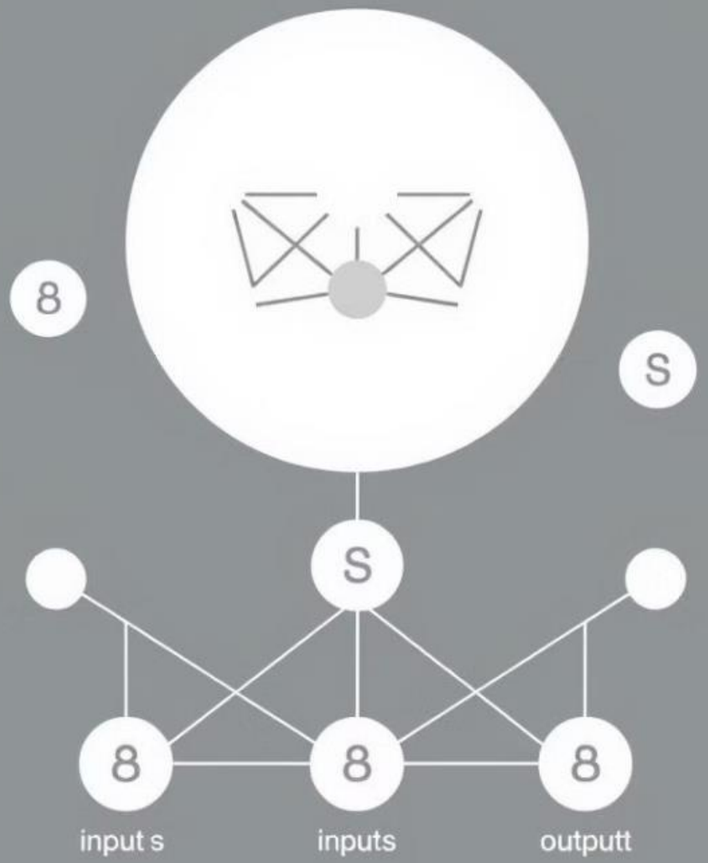
Redes Neuronales Multicapa



Funciones de Activación:

Las funciones de activación introducen no linealidades en la red, permitiendo que la red aprenda relaciones complejas entre las entradas y las salidas.

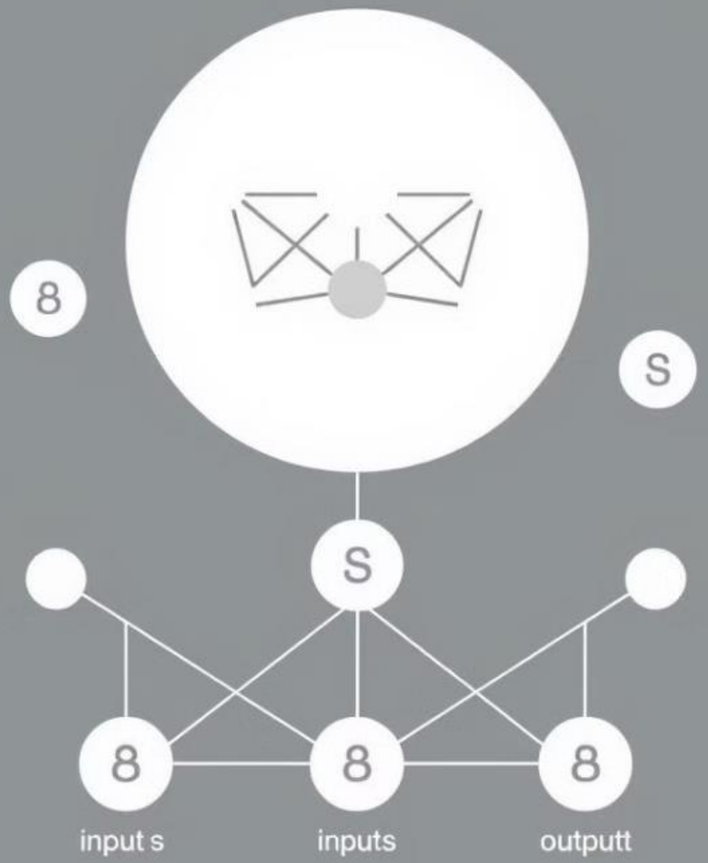
Ejemplos comunes incluyen la función sigmoide, la función ReLU (Rectified Linear Unit) y la función tangente hiperbólica.



Propagación hacia Adelante (Forward Propagation):

Durante la propagación hacia adelante, los datos de entrada se pasan a través de la red, capa por capa, hasta que se obtiene una salida.

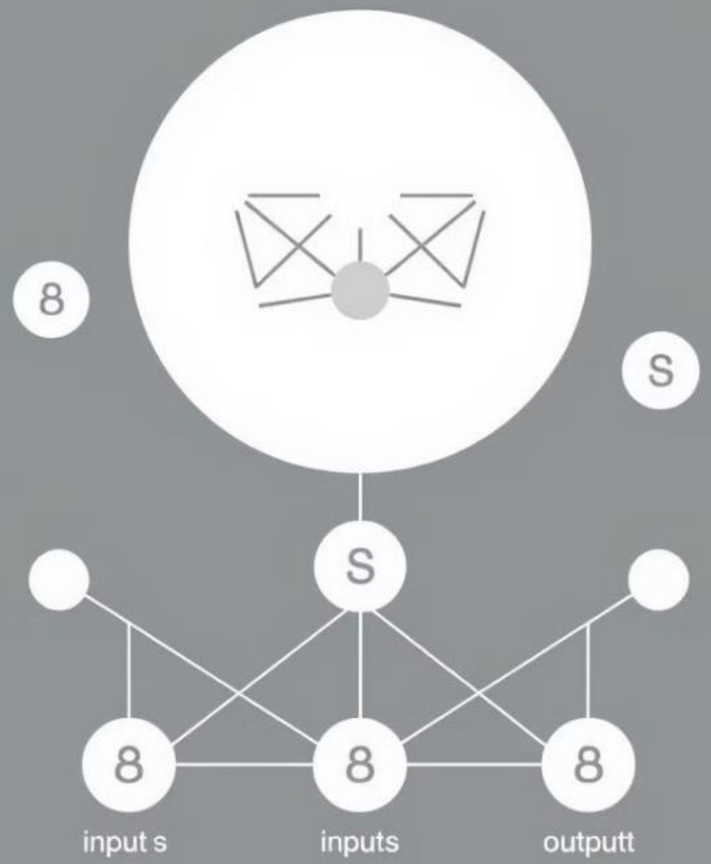
Cada neurona calcula una suma ponderada de sus entradas, aplica la función de activación y pasa el resultado a la siguiente capa.



Backpropagation (Retropropagación):

El backpropagation es un algoritmo utilizado para entrenar redes neuronales. Consiste en calcular el error entre la salida predicha y la salida deseada, y luego propagar este error hacia atrás a través de la red para ajustar los pesos.

El objetivo es minimizar la función de pérdida (loss function), que mide la diferencia entre la salida predicha y la salida real.



Gradiente Descendente (Gradient Descent):

El gradiente descendente es un método de optimización utilizado para minimizar la función de pérdida. Se ajustan los pesos de la red en la dirección opuesta al gradiente de la función de pérdida con respecto a los pesos.

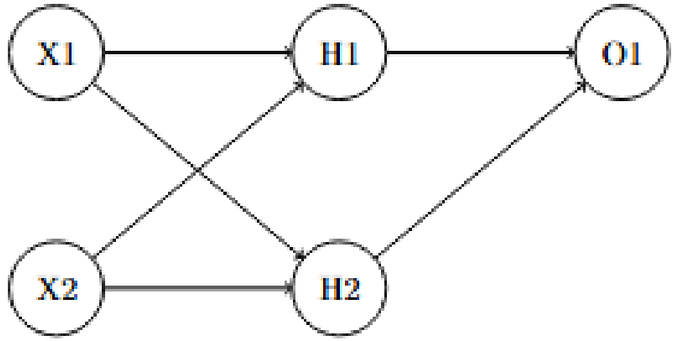
Inputs	Weights	Outputs	Weight	Outpnırs
Frat:	380.40	16.3	219	1.0
	20y, ɜtx:::	17.9	160	
	20y.16	16.0	220	4.0
Errors	190.10	37.4	179	3.0
	25y, 1et:::	17.3	129	2.0
	10y.19	11.5	250	1.0
Errors:	596.10	20.9	270	1.0
Frat:	30y.10	17.5	155	1.0
	28y, 2tx:::	51.9		
Prat:	20y.19	57 9	226	1.0
Erat:	30y.10	32.5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

Ejemplo de Aprendizaje

Vamos a considerar un ejemplo simple: una red neuronal con una capa oculta para resolver el problema de la compuerta XOR.

Paso 1: Definir la Arquitectura de la Red

- Capa de Entrada: 2 neuronas (para las dos entradas de XOR). X 1 ,X 2
- Capa Oculta: 2 neuronas con función de activación sigmoide. H 1 ,H 2
- Capa de Salida: 1 neurona con función de activación sigmoide. O 1



Inputs	Weights	Outputs	Weight	Outpnırs
Frat:	380.40	16.3	219	1.0
	20y, ɜtx:::	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

Asignamos pesos iniciales aleatorios para cada conexión:

1. Pesos entre entrada y capa oculta:

- $w11$=0.2 (de $X1$ a $H1$)
- $w12$=0.4 (de $X2$ a $H1$)
- $w21$=0.3 (de $X1$ a $H2$)
- $w22$=0.5 (de $X2$ a $H2$)

2. Pesos entre capa oculta y salida:

- $w31$=0.7 (de $H1$ a $O1$)
- $w32$=0.9 (de $H2$ a $O1$)

3. Bias (sesgos):

- $b1$=0.1 (para $H1$)
- $b2$=0.2 (para $H2$)
- $b3$=0.3 (para $O1$)

Ejemplo de Aprendizaje

Paso 2: Forward Propagation (Propagación Hacia Adelante)Tomemos un ejemplo de entrada: $X1$=0, $X2$=1 → Salida esperada: Y=1

1 Cálculo de la Capa Oculta

Calculamos la activación de cada neurona de la capa oculta:

$$H_1 = \sigma(X_1 \cdot w_{11} + X_2 \cdot w_{12} + b_1)$$

$$H_2 = \sigma(X_1 \cdot w_{21} + X_2 \cdot w_{22} + b_2)$$

Sustituyendo valores:

$$H_1 = \sigma(0 \cdot 0.2 + 1 \cdot 0.4 + 0.1) = \sigma(0.5)$$

$$H_2 = \sigma(0 \cdot 0.3 + 1 \cdot 0.5 + 0.2) = \sigma(0.7)$$

Usamos la función sigmoide:

$$\sigma(0.5) = \frac{1}{1 + e^{-0.5}} \approx 0.62$$

$$\sigma(0.7) = \frac{1}{1 + e^{-0.7}} \approx 0.67$$

Entonces, $H_1 = 0.62$ y $H_2 = 0.67$.

2 Cálculo de la Salida

$$O_1 = \sigma(H_1 \cdot w_{31} + H_2 \cdot w_{32} + b_3)$$

Sustituyendo valores:

$$O_1 = \sigma(0.62 \cdot 0.7 + 0.67 \cdot 0.9 + 0.3)$$

$$O_1 = \sigma(0.434 + 0.603 + 0.3) = \sigma(1.337)$$

$$\sigma(1.337) = \frac{1}{1 + e^{-1.337}} \approx 0.79$$

El resultado de la red es 0.79, pero queremos que sea 1. Hay un error que debemos corregir con backpropagation.

Ejemplo de Aprendizaje

Paso 3: Backpropagation (Propagación Hacia Atrás)

1 Cálculo del Error en la Salida

$$Error = Y - O_1 = 1 - 0.79 = 0.21$$

Calculamos el **gradiente de la salida**:

$$\delta O_1 = Error \times \sigma'(1.337)$$

$$\sigma'(1.337) = \sigma(1.337) \times (1 - \sigma(1.337)) = 0.79 \times (1 - 0.79) = 0.166$$

$$\delta O_1 = 0.21 \times 0.166 = 0.035$$

2 Ajuste de los pesos de la capa de salida

Los nuevos pesos se actualizan con **descenso del gradiente**:

$$w_{31} = w_{31} + \eta \times \delta O_1 \times H_1$$

$$w_{32} = w_{32} + \eta \times \delta O_1 \times H_2$$

Tomemos una tasa de aprendizaje $\eta = 0.5$:

$$w_{31} = 0.7 + 0.5 \times 0.035 \times 0.62 = 0.7 + 0.011 = 0.711$$

$$w_{32} = 0.9 + 0.5 \times 0.035 \times 0.67 = 0.9 + 0.012 = 0.912$$

3 Retropropagación del Error a la Capa Oculta

Calculamos el error de cada neurona oculta:

$$\delta H_1 = \delta O_1 \times w_{31} \times \sigma'(0.5)$$

$$\sigma'(0.5) = 0.62 \times (1 - 0.62) = 0.235$$

$$\delta H_1 = 0.035 \times 0.7 \times 0.235 = 0.0058$$

$$\delta H_2 = 0.035 \times 0.9 \times \sigma'(0.7) = 0.0069$$

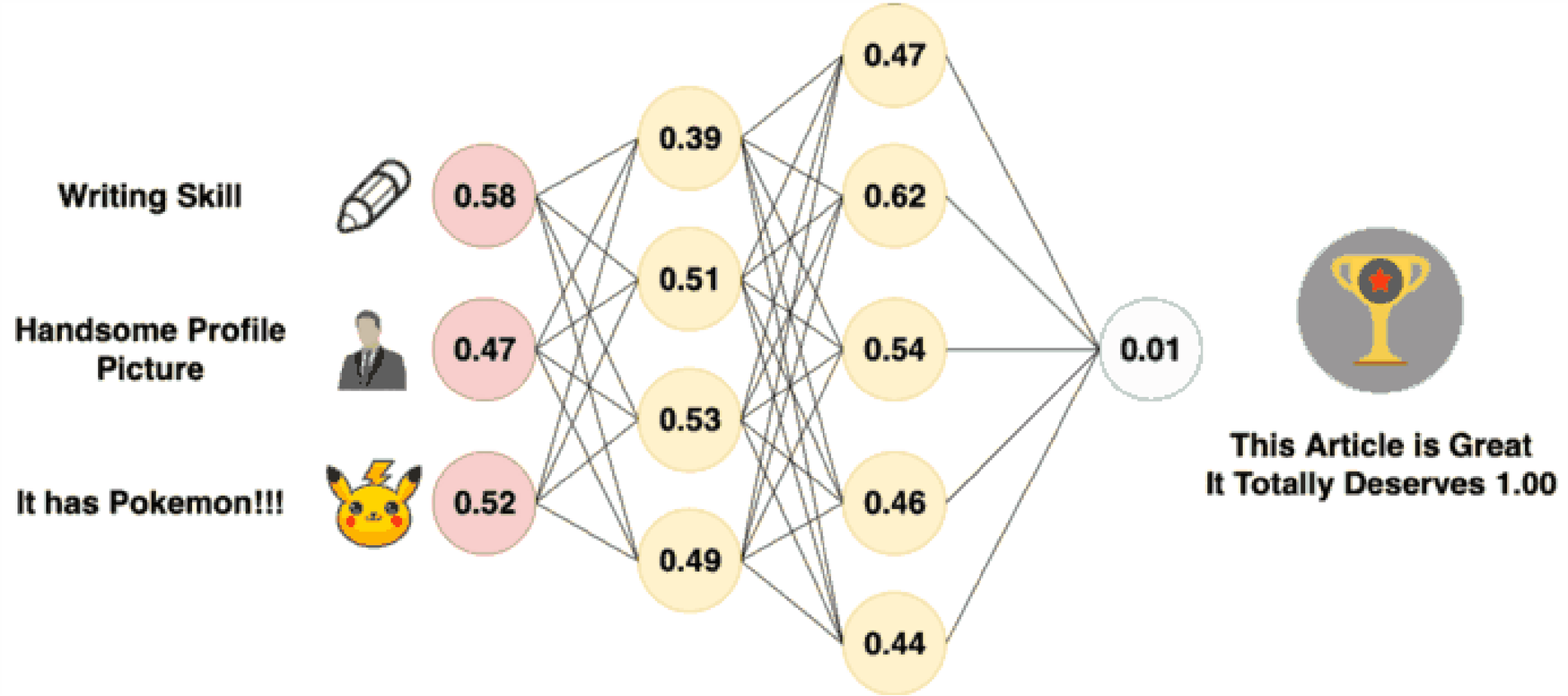
Actualizamos los pesos de la capa oculta:

$$w_{12} = w_{12} + \eta \times \delta H_1 \times X_2$$

$$w_{22} = w_{22} + \eta \times \delta H_2 \times X_2$$

Después de actualizar pesos en varias iteraciones, la red se entrenará correctamente.

Ejemplo de Aprendizaje



Ejemplo de Aprendizaje Antena Patch