

Redes Neuronales Artificiales

Semestre Otoño 2023

Modulo 1

Cerebro Humano

- Es el órgano más complejo del mundo animal.
- En los humanos pesa cerca de 1,3 kg.
- Contiene aproximadamente 1 000 000 000 000 (1^{12}) neuronas
- Cada neurona se conecta aproximadamente, en promedio, con otras 10.000 neuronas (1^4) . Por lo tanto tiene aproximadamente 1^{16} conexiones sinápticas.
- Consume muy poca energía.



- Almacena a través de asociaciones (memoria se direcciona por contenido).
- Trabaja en forma masivamente paralelo. Posee millones de redes neuronales.
- Es flexible y plástico, lo cual le permite adaptarse y adaptar a los organismos que lo poseen.
- El cerebro es un sistema auto-organizado y no programado.
- Se ocupa de controlar todo el organismo, en sus múltiples funciones.
- Es robusto y tolerante a fallas.



Cerebro Humano

- Emula características propias de los humanos: memorizar y asociar hechos.
- Se aprende de la **experiencia**.
- El cerebro humano es el ejemplo más perfecto de sistema capaz de adquirir conocimiento.
- Se modela artificialmente ese sistema.

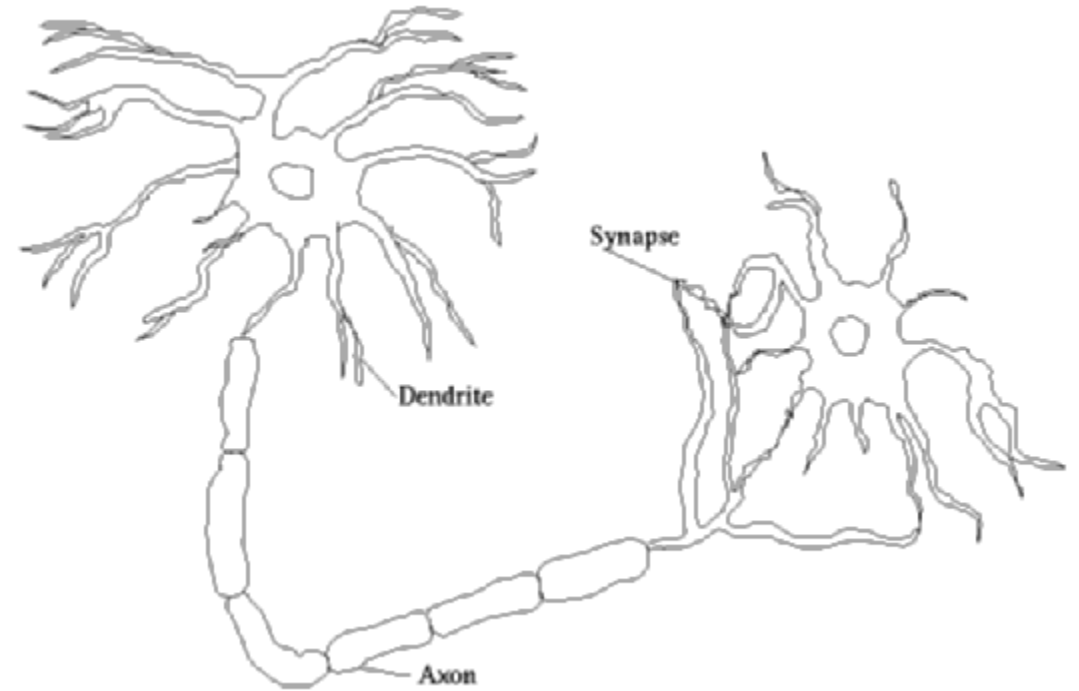


Redes Neuronales

- Una red neuronal es "un nuevo sistema para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: **la neurona**".
- Las neuronas son un componente relativamente simple pero conectadas de a miles forman un poderoso sistema.

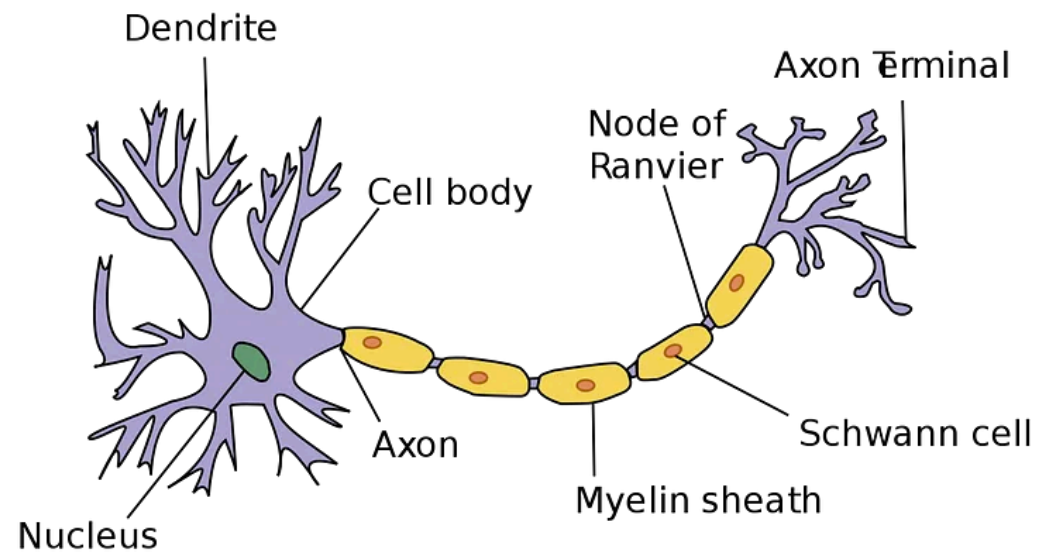
Fundamento Biológico

- El cerebro humano contiene más de cien mil millones de neuronas.
- La clave para el procesamiento de la información son las conexiones entre ellas llamadas **sinápsis**.

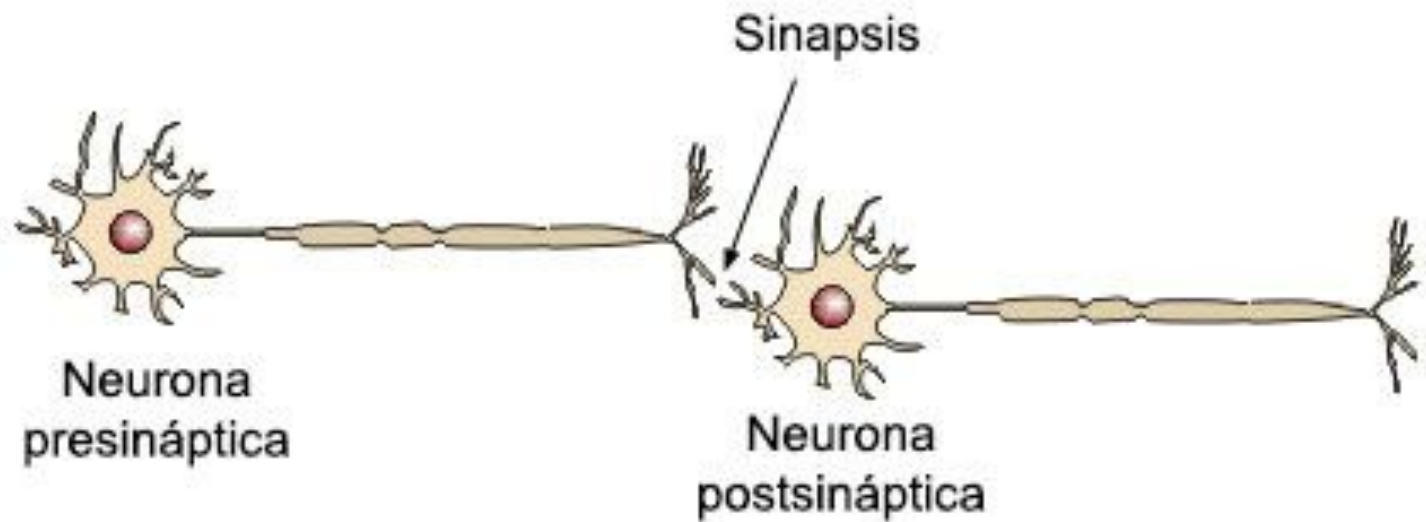


Estructura Biologica

- Las **dendritas** son la vía de entrada de las señales que se combinan en el cuerpo de la neurona.
- El **axón** es el camino de salida de la señal generada por la neurona.
- En las terminaciones de las **sinápsis** se encuentran unas vesículas que contienen unas sustancias químicas llamadas neurotransmisores, que propagan señales electroquímicas de una neurona a otra.
- La neurona es estimulada por sus entradas y cuando alcanza cierto umbral, se dispara o activa pasando una señal hacia el axón.

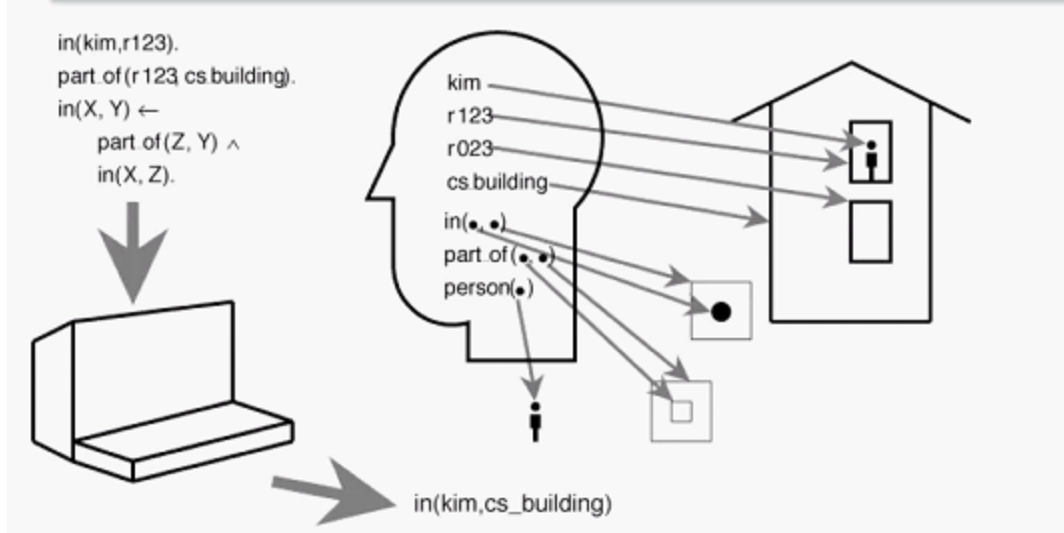


Dirección del impulso eléctrico

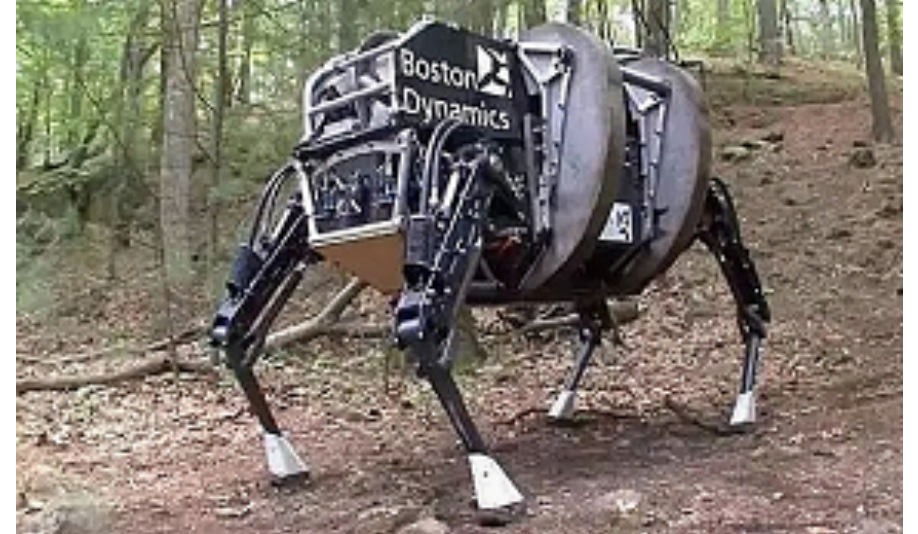


Paradigmas en Inteligencia Artificial

En la actualidad, hay **tres enfoques** principales de **inteligencia artificial**: **simbolismo**, **conexionismo** y **conductismo**.



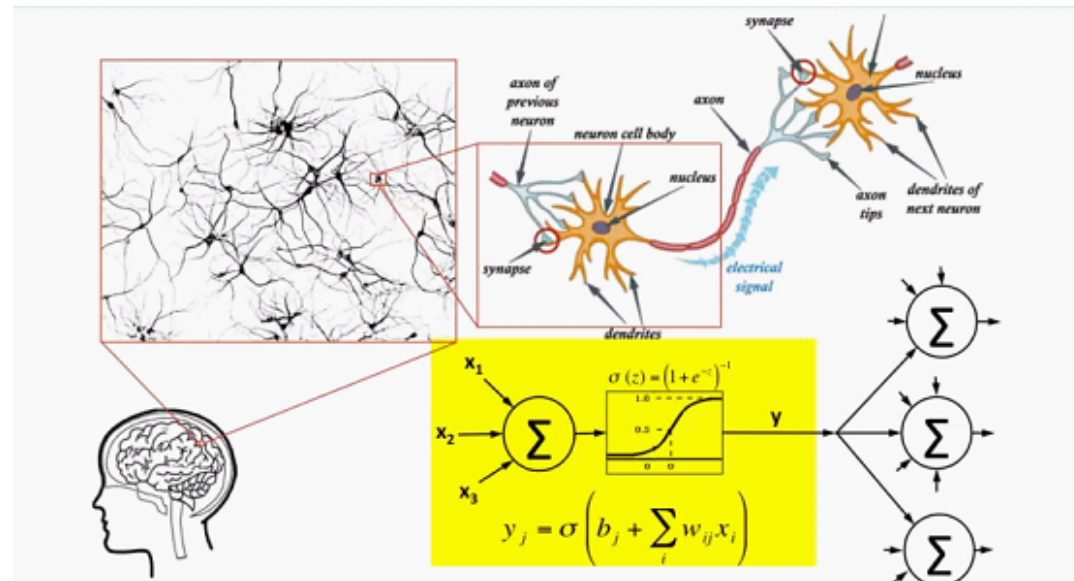
Simbolismo

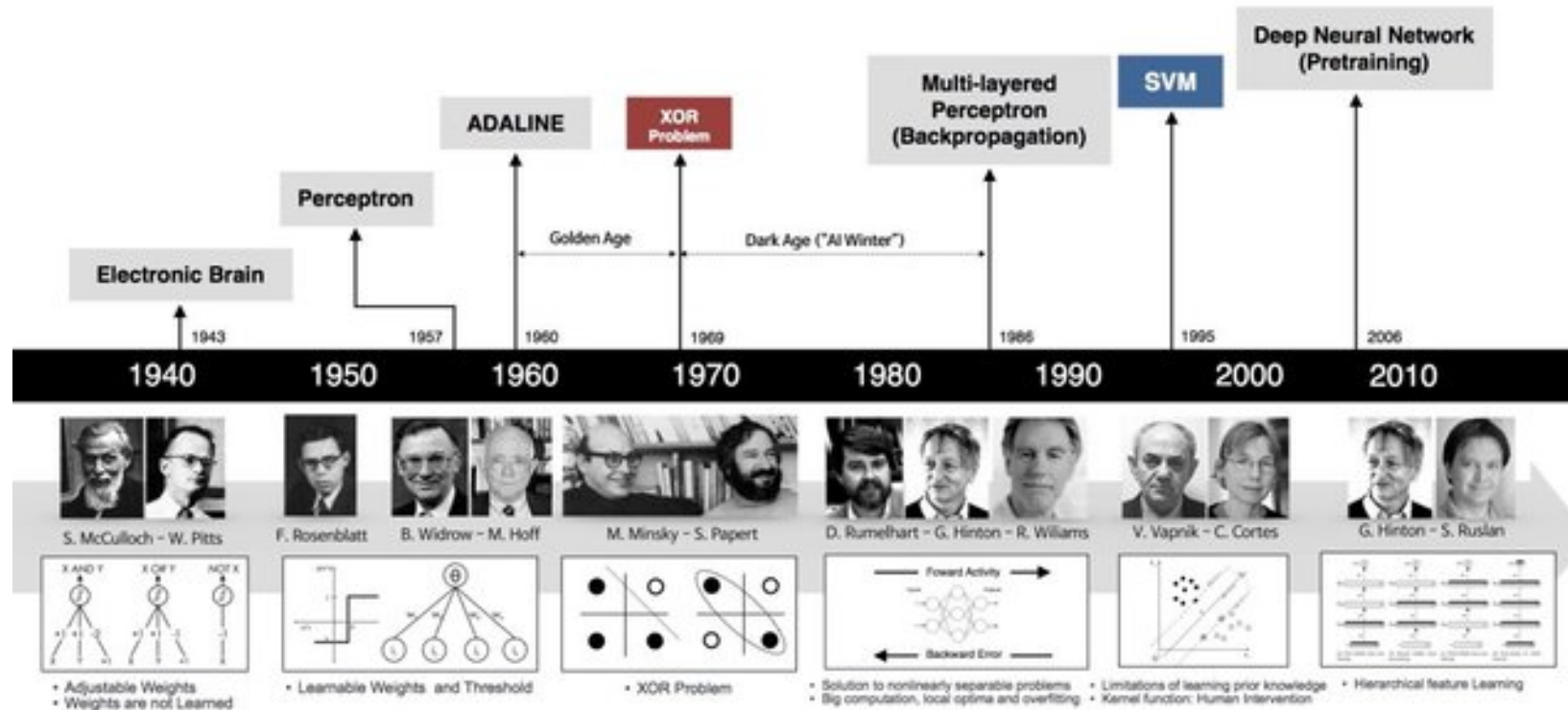


Conductivismo

Paradigma Conexionista

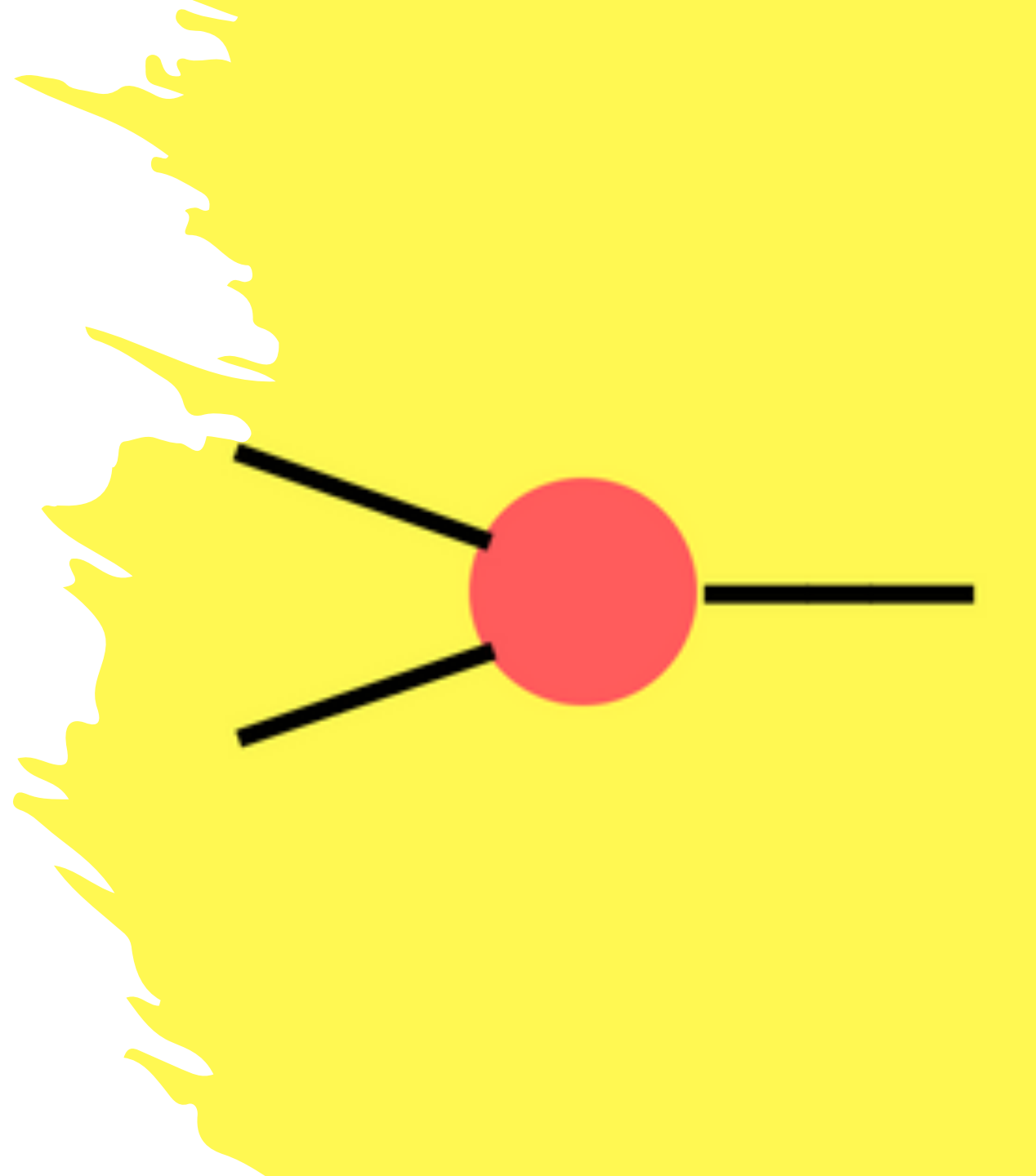
- En **conexionismo** asume que el procesamiento y el almacenamiento de la información recae en la compleja arquitectura de conexiones que forman las neuronas entre sí y con otras células, y que las neuronas son simples unidades de procesamiento y transmisión de señales.



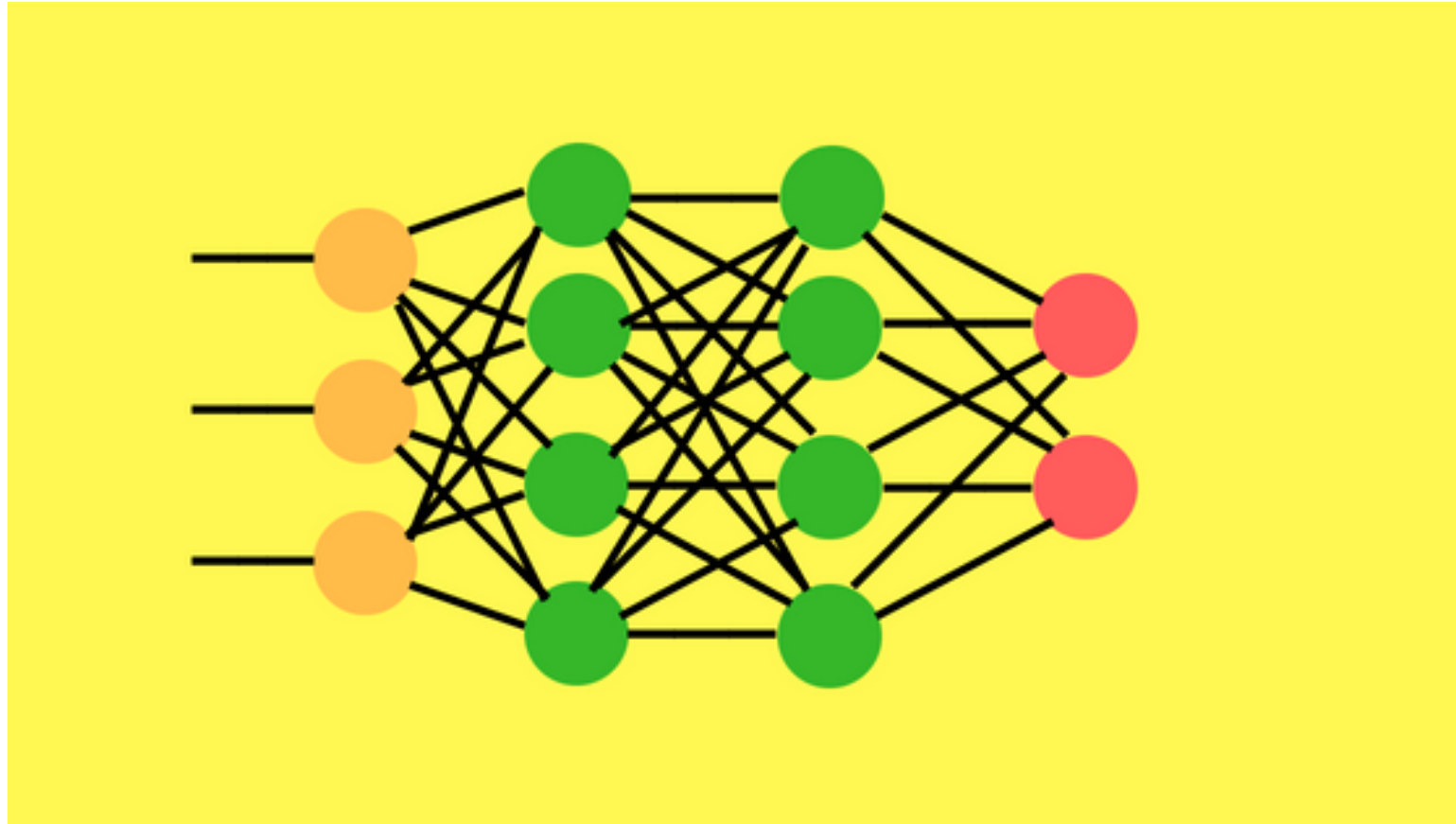


1958 – Perceptrón

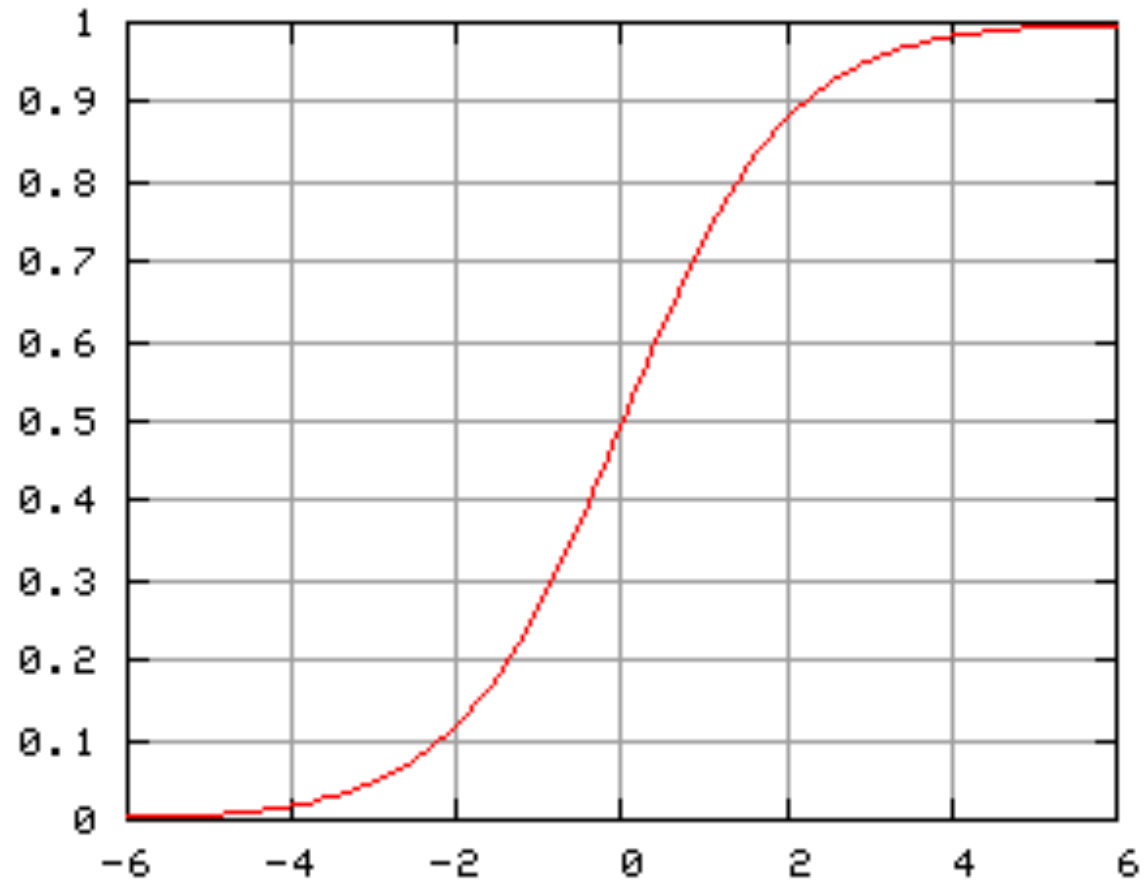
- Entre las décadas de 1950 y 1960 el científico [Frank Rosenblatt](#), inspirado en el trabajo de Warren McCulloch y Walter Pitts creó el Perceptron, la unidad desde donde nacería y se potenciarían las redes neuronales artificiales.



1965 – Multilayer Perceptrón



1980 - Neuronas Sigmoides



1980 - Redes Feedforward

- Se les llama así a las *redes en que las salidas de una capa son utilizadas como entradas en la próxima capa*. Esto quiere decir que no hay loops “hacia atrás”. Siempre se “alimenta” de valores hacia adelante. Hay redes que veremos más adelante en las que sí que existen esos loops (Recurrent Neural Networks).
- Además existe el concepto de “fully connected Feedforward Networks” y se refiere a que todas las neuronas de entrada, están conectadas con todas las neuronas de la siguiente capa.

1986 – Backpropagation

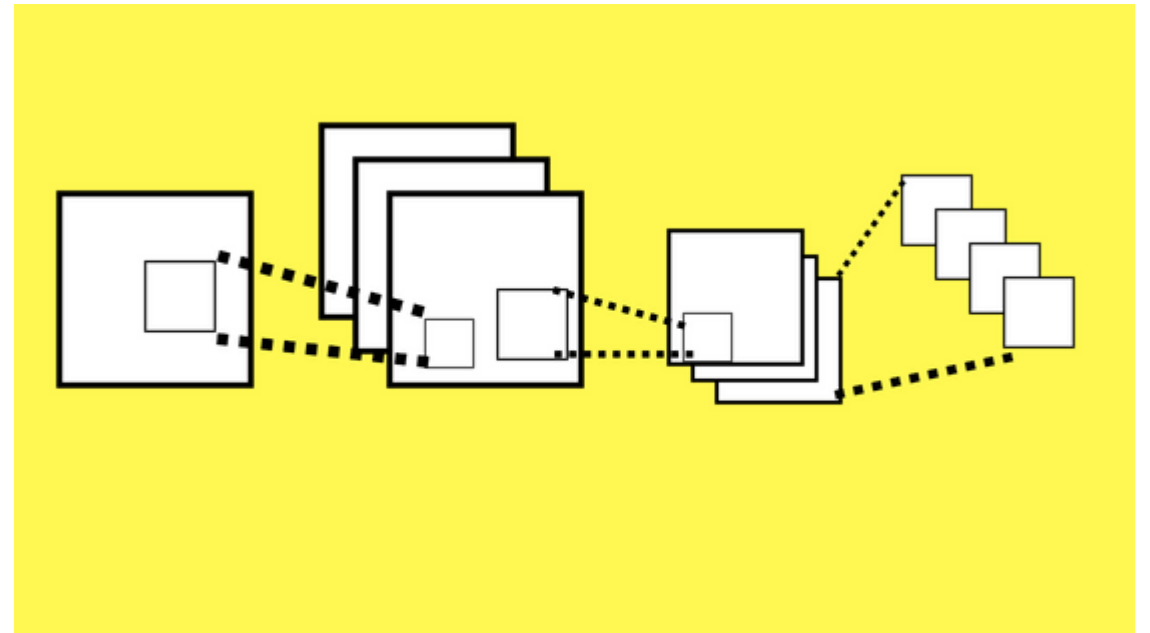
- Gracias al algoritmo de [backpropagation](#) se hizo posible entrenar redes neuronales de multiples capas de manera supervisada.
- Al calcular el error obtenido en la salida e ir propagando hacia las capas anteriores se van haciendo ajustes pequeños (minimizando costo) en cada iteración para lograr que la red aprenda consiguiendo que la red pueda -por ejemplo- clasificar las entradas correctamente.

1989 - Convolutional Neural Network

Las [Convolutional Neural Networks](#) son redes multilayered que toman su inspiración del cortex visual de los animales.

Esta arquitectura es útil en varias aplicaciones, principalmente **procesamiento de imágenes**.

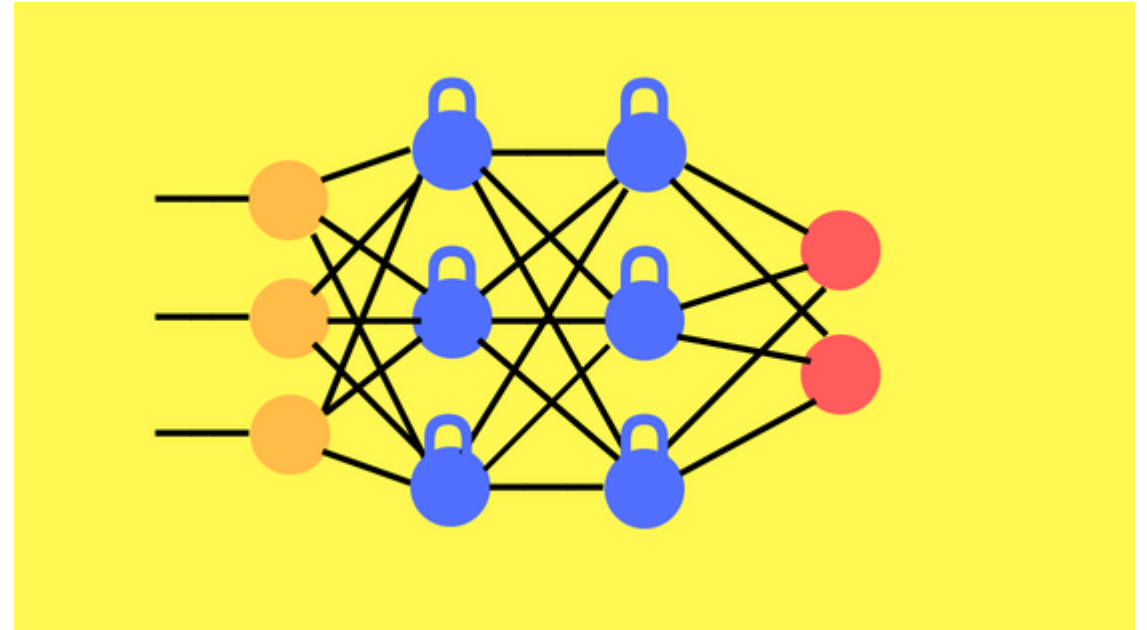
La primera CNN fue creada por [Yann LeCun](#) y estaba enfocada en el reconocimiento de letras manuscritas.



1997 - Long Short Term Memory / Recurrent Neural Network

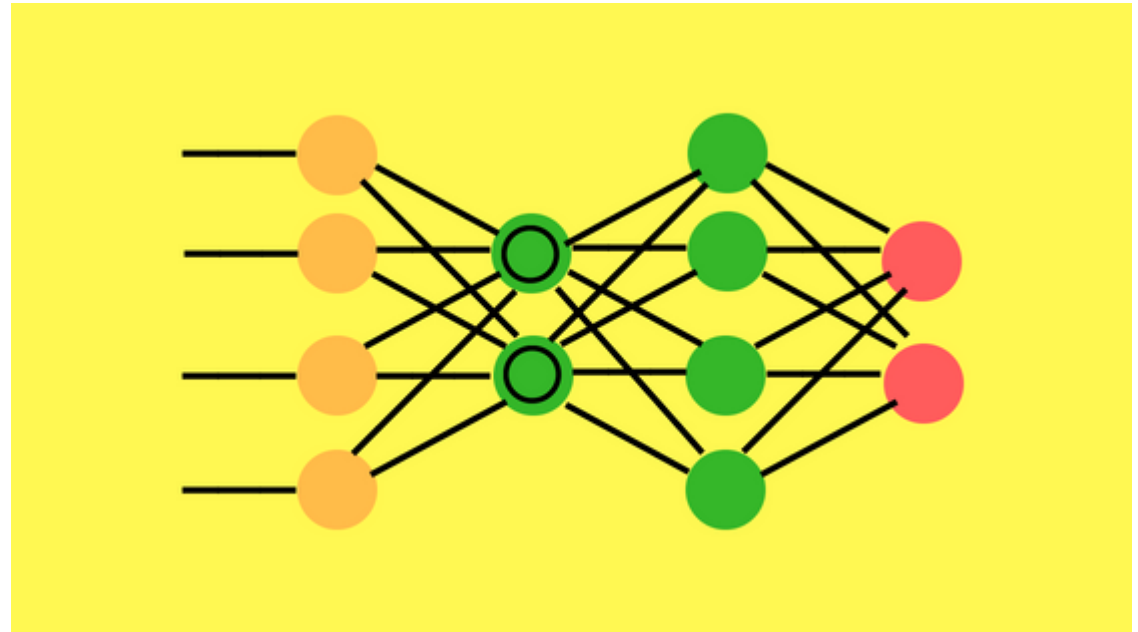
Aquí vemos que la red LSTM tiene neuronas ocultas con loops hacia atrás (en azul).

Esto permite que almacene información en celdas de memoria.



2006 – Deep Belief Networks (DBN)

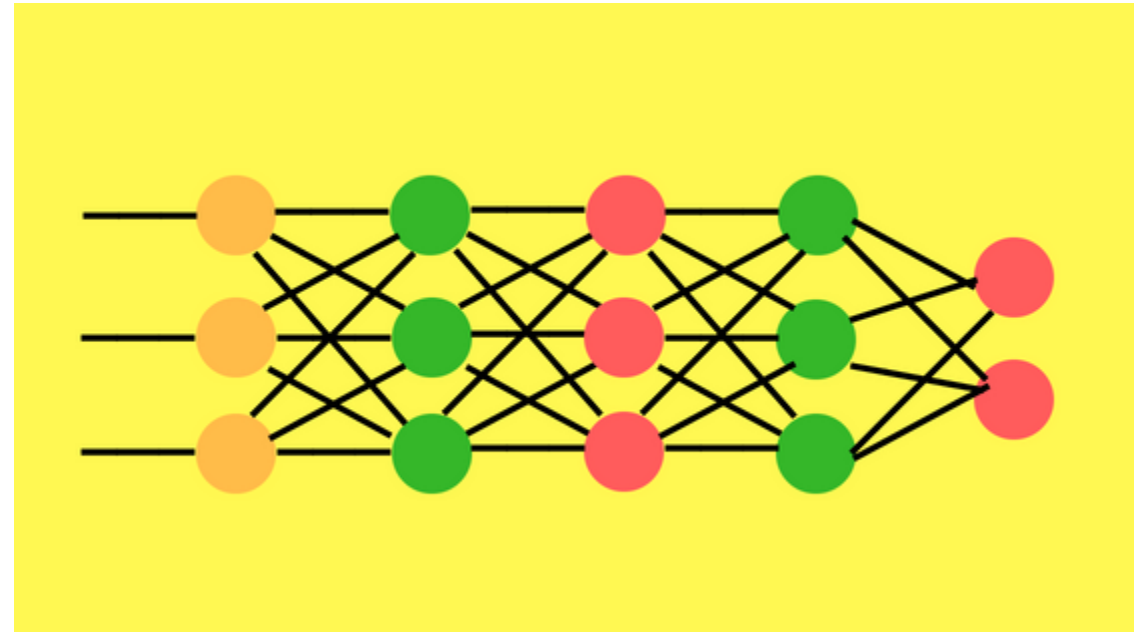
La Deep Belief Network utiliza un Autoencoder con Restricted Boltzmann Machines para preentrenar a las neuronas de la red y obtener un mejor resultado final.



2014 – Generative Adversarial Networks

Las GAN, entrenan dos redes neuronales en simultáneo.

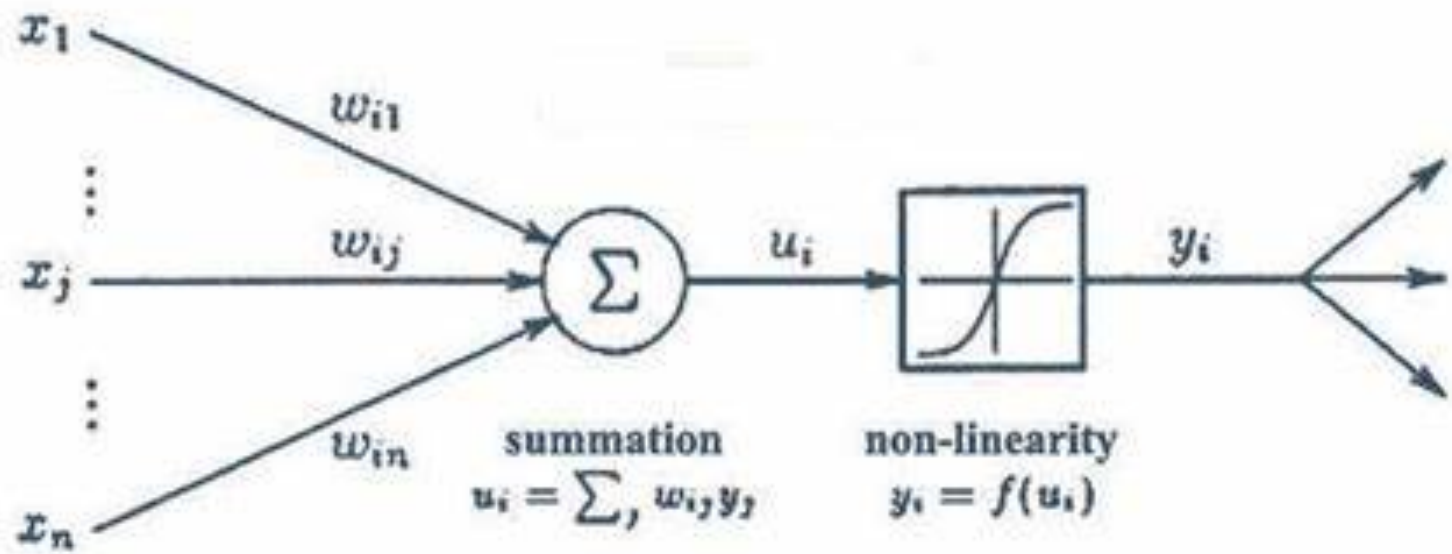
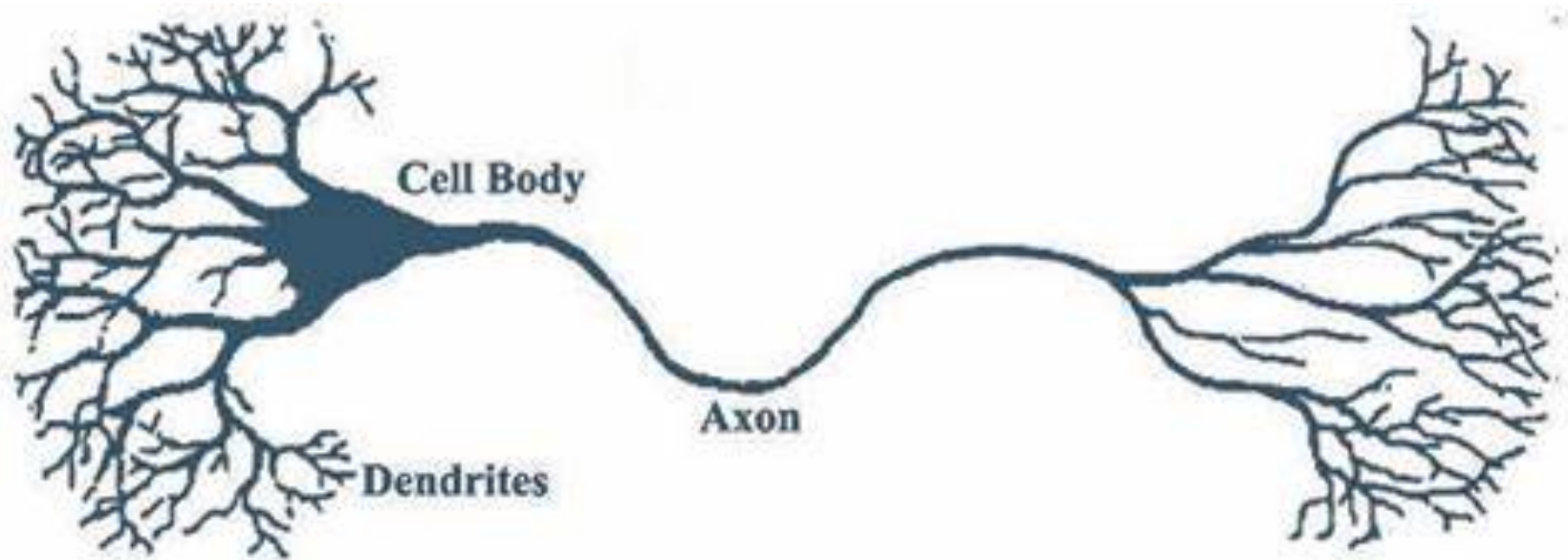
La red de Generación y la red de Discriminación. A medida que la máquina aprende, comienza a crear muestras que son indistinguibles de los datos reales.

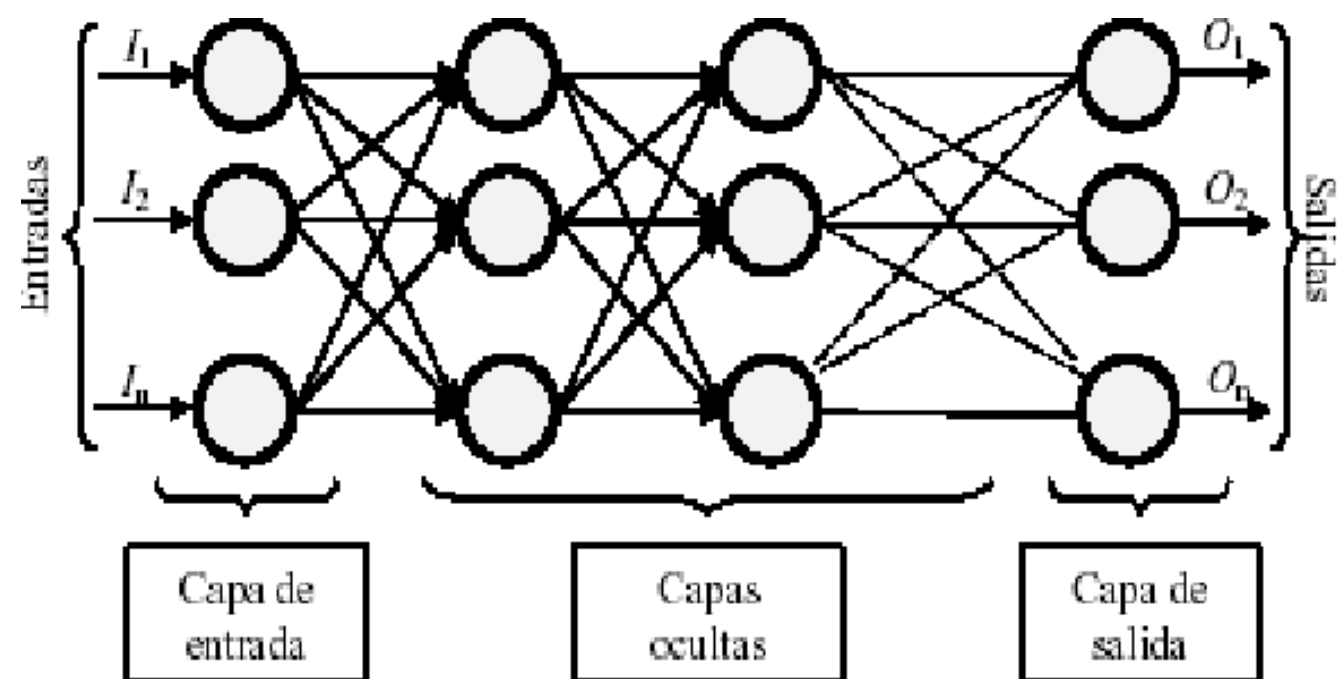


Las aplicaciones más exitosas de las RNA son:

1. Procesamiento de imágenes y de voz
2. Reconocimiento de patrones
3. Planeamiento
4. Interfaces adaptivas para sistemas Hombre/máquina
5. Predicción
6. Control y optimización
7. Filtrado de señales

Estructura Artificial





Principales características de una RNA

- **Arquitectura:** Estructura o patrón de conexiones entre las unidades de proceso
- **Dinámica de la Computación** que nos expresa el valor que toman las unidades de proceso y que se basa en unas **funciones de activación (o de transferencia)** que especifican como se transforman las señales de entrada de la unidad de proceso en la señal de salida.
- **Algoritmo de Entrenamiento o Aprendizaje:** Procedimiento para determinar los pesos de las conexiones

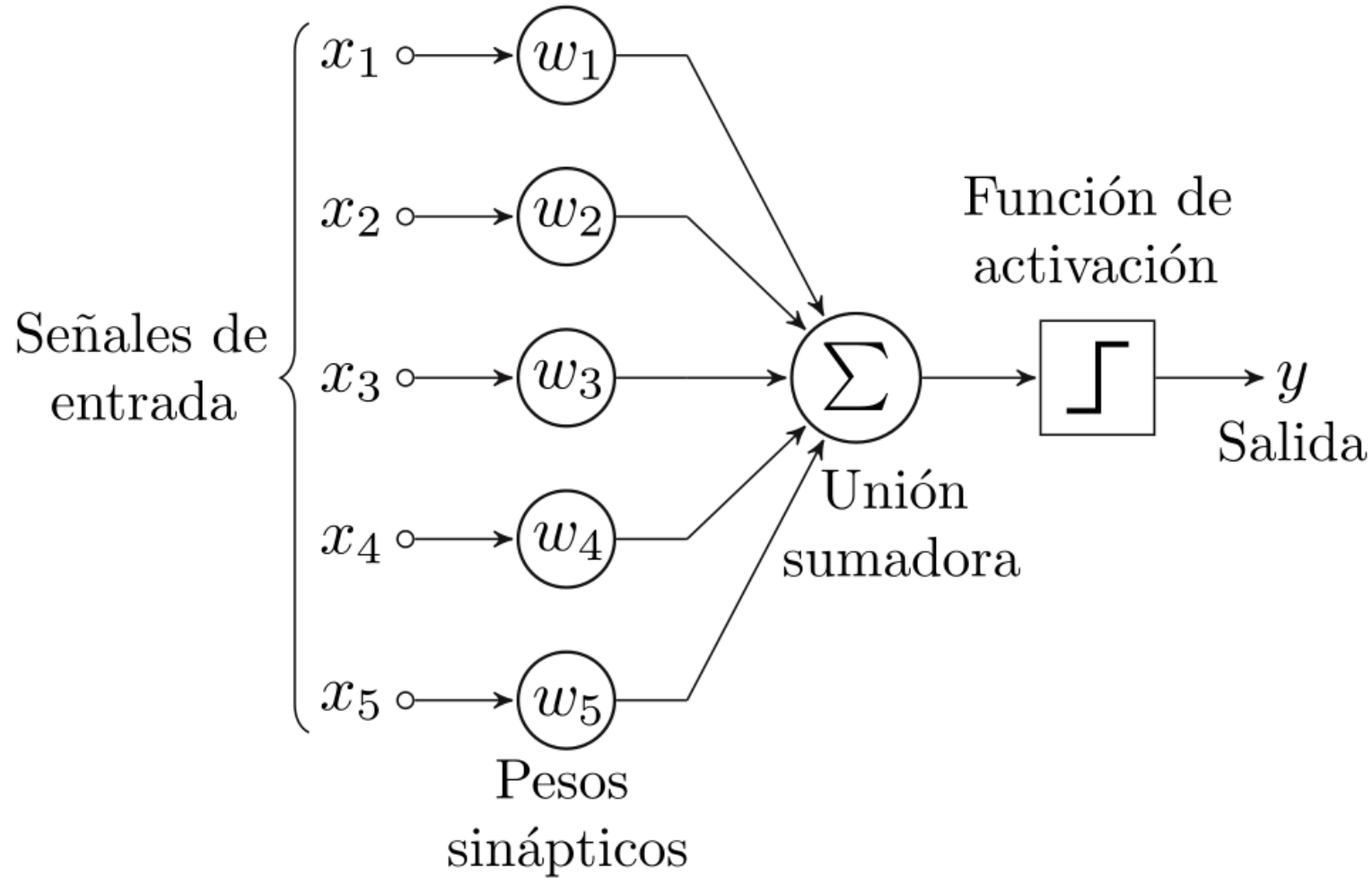
Principales características de una RNA

Una característica muy importante de estas redes es su *naturaleza adaptativa*, donde el "**aprendizaje con ejemplos**" **sustituye** a la "programación" en la resolución de problemas.

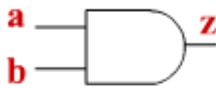

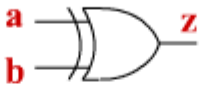
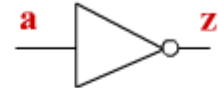
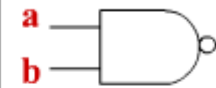

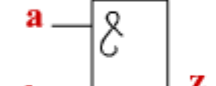
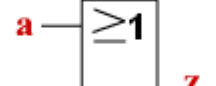
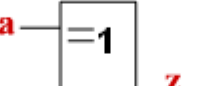
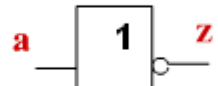
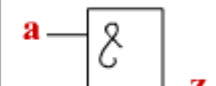
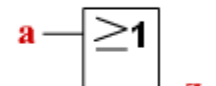
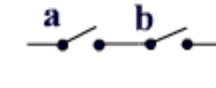
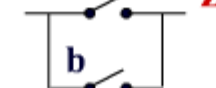
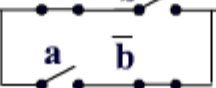
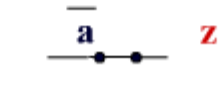
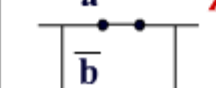
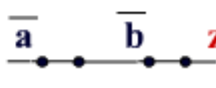
Elementos de una red neuronal artificial (RNA)

- Las neuronas están conectadas por canales unidireccionales con peso.
- El peso w_{ij} está asociado al canal que conecta la neurona j con la neurona i .
- La entrada total de la neurona j es $net_j = \sum w_{ij}y_i$.
- La salida de la neurona j es $y_j = f(net_j)$.

PERCEPTRÓN: McCulloch y Pits (1943)

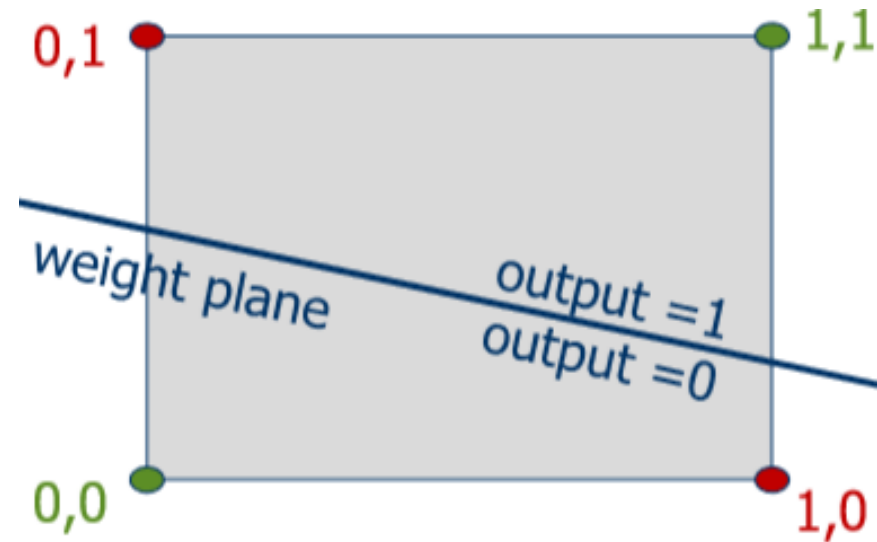
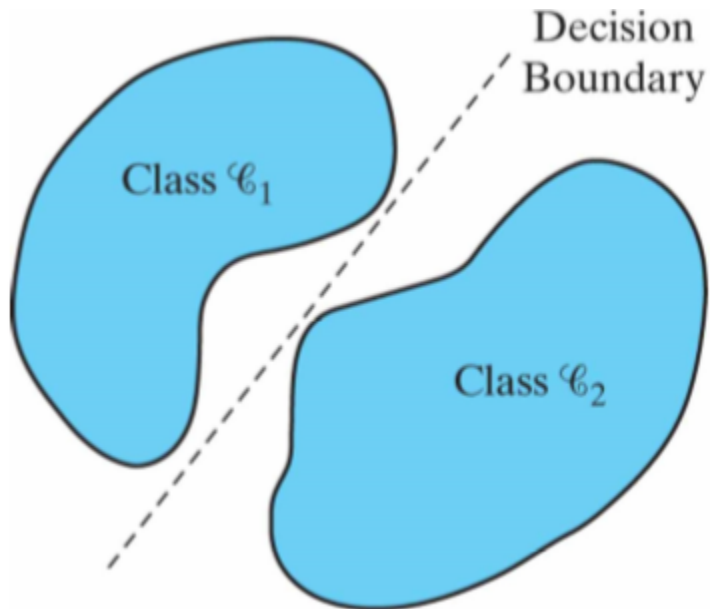


El Perceptrón Solo Resuelve Problemas Linealmente Separables

NOMRE	AND - Y	OR - O	XOR O-exclusiva	NOT Inversor	NAND	NOR																																																																																	
SÍMBOLO																																																																																							
SÍMBOLO																																																																																							
TABLA DE VERDAD	<table><tr><th>a</th><th>b</th><th>z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	a	b	z	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>a</th><th>b</th><th>z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	a	b	z	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>a</th><th>b</th><th>z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	a	b	z	0	0	0	0	1	1	1	0	1	1	1	0	<table><tr><th>a</th><th>z</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	a	z	0	1	1	0	<table><tr><th>a</th><th>b</th><th>z</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	a	b	z	0	0	1	0	1	1	1	0	1	1	1	0	<table><tr><th>a</th><th>b</th><th>z</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	a	b	z	0	0	1	0	1	0	1	0	0	1	1	0
a	b	z																																																																																					
0	0	0																																																																																					
0	1	0																																																																																					
1	0	0																																																																																					
1	1	1																																																																																					
a	b	z																																																																																					
0	0	0																																																																																					
0	1	1																																																																																					
1	0	1																																																																																					
1	1	1																																																																																					
a	b	z																																																																																					
0	0	0																																																																																					
0	1	1																																																																																					
1	0	1																																																																																					
1	1	0																																																																																					
a	z																																																																																						
0	1																																																																																						
1	0																																																																																						
a	b	z																																																																																					
0	0	1																																																																																					
0	1	1																																																																																					
1	0	1																																																																																					
1	1	0																																																																																					
a	b	z																																																																																					
0	0	1																																																																																					
0	1	0																																																																																					
1	0	0																																																																																					
1	1	0																																																																																					
EQUIVALENTE EN CONTACTOS																																																																																							

Limitaciones Del Perceptrón Problema Del O Exclusivo

- Solo resuelve problemas de clasificación linealmente separables
- No separa $(0,0)$, $(1,1)$ de $(1,0)$, $(0,1)$

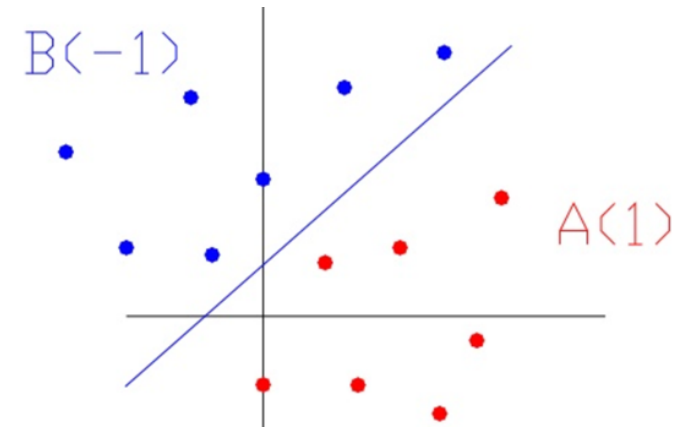


Rosenblatt (1956) Algoritmo De Entrenamiento Del Perceptrón

- Datos de entrenamiento $\{P_1, t_1\}, \dots, \{P_n, t_n\}$

s= salida de la red

- Hiperplano separador



Algoritmo de entrenamiento

for j=1:etapas

for i=1:observaciones

Si $s=t$ (salida=target), $W_{iN} = W_{iA}$ (nuevo peso=antiguo)

Si $s=0, t=1$ $W_{iN} = W_{iA} + \alpha P_i$ (aumentar todos los pesos)

Si $s=1, t=0$ $W_{iN} = W_{iA} - \alpha P_i$ (reducir todos los pesos)

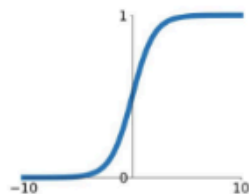
α tasa de aprendizaje

Otras Funciones De Activación

Activation Functions

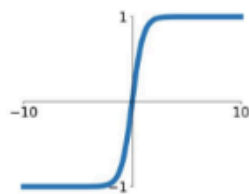
Sigmoid

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



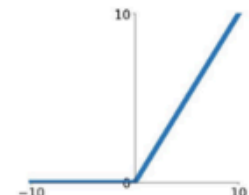
tanh

$$\tanh(x)$$



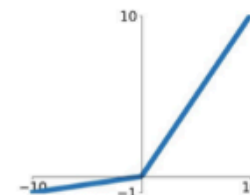
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

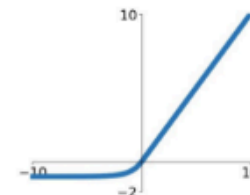


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Función de activación ReLU (Rectified Linear Unit):

- Activa la neurona si la entrada es mayor que cero, de lo contrario, la desactiva.
- Desventaja: No es diferenciable en cero, lo que puede presentar problemas en algunos algoritmos de optimización. También puede sufrir de "neuronas muertas" si la activación es cero para la mayoría de las entradas.

Función de activación Tangente hiperbólica (Tanh):

- Rango de salida entre -1 y 1.
- Desventaja: Sufre del problema de "desvanecimiento del gradiente" en redes neuronales profundas, similar a la función sigmoide.

Función de activación Leaky ReLU:

- Similar a ReLU, pero tiene una pendiente pequeña para entradas negativas.
- Desventaja: Puede presentar "neuronas muertas" si la pendiente para las entradas negativas no es apropiada.

Función de activación Sigmoide:

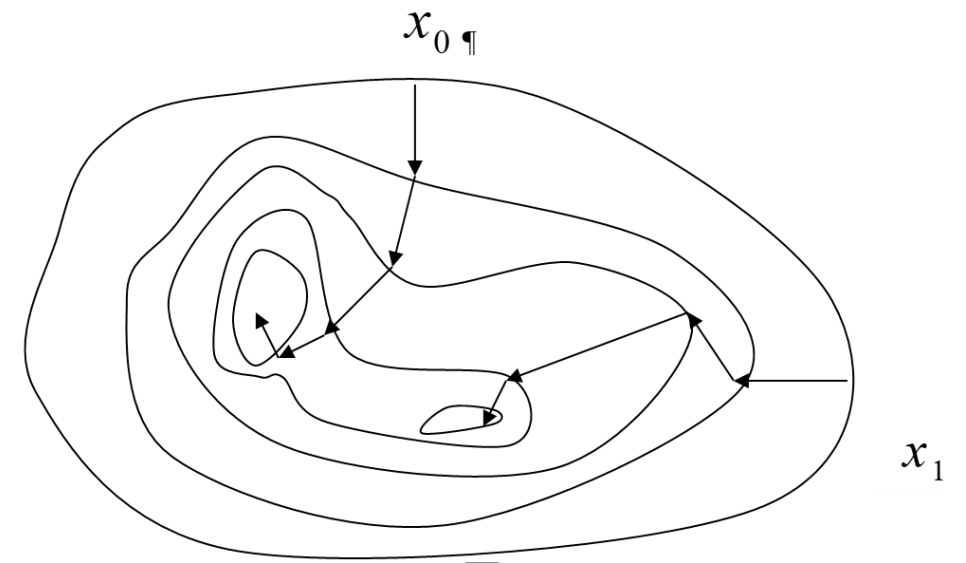
- Rango de salida entre 0 y 1.
- Desventaja: Sufre del problema de "desvanecimiento del gradiente" en redes neuronales profundas, lo que puede dificultar el aprendizaje.

****desvanecimiento del gradiente**

fenómeno que ocurre durante el entrenamiento de redes neuronales profundas, en el cual los gradientes se vuelven cada vez más pequeños a medida que se retropropagan hacia las capas anteriores de la red. Esto puede provocar que las capas anteriores aprendan muy lentamente o incluso dejen de aprender por completo.

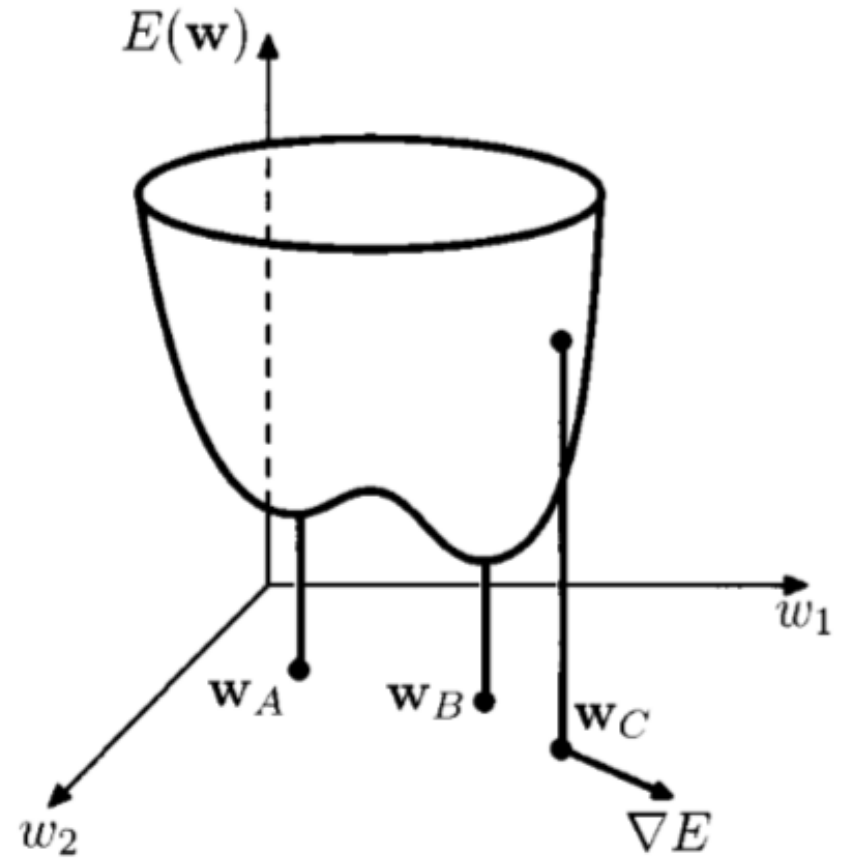
Algoritmo Del Gradiente Descendente

es un método de optimización utilizado en el aprendizaje automático para minimizar una función de pérdida en un modelo de machine learning. Este algoritmo es ampliamente utilizado para ajustar los pesos y los sesgos de un modelo de manera iterativa con el fin de encontrar los valores que minimizan la función de pérdida.



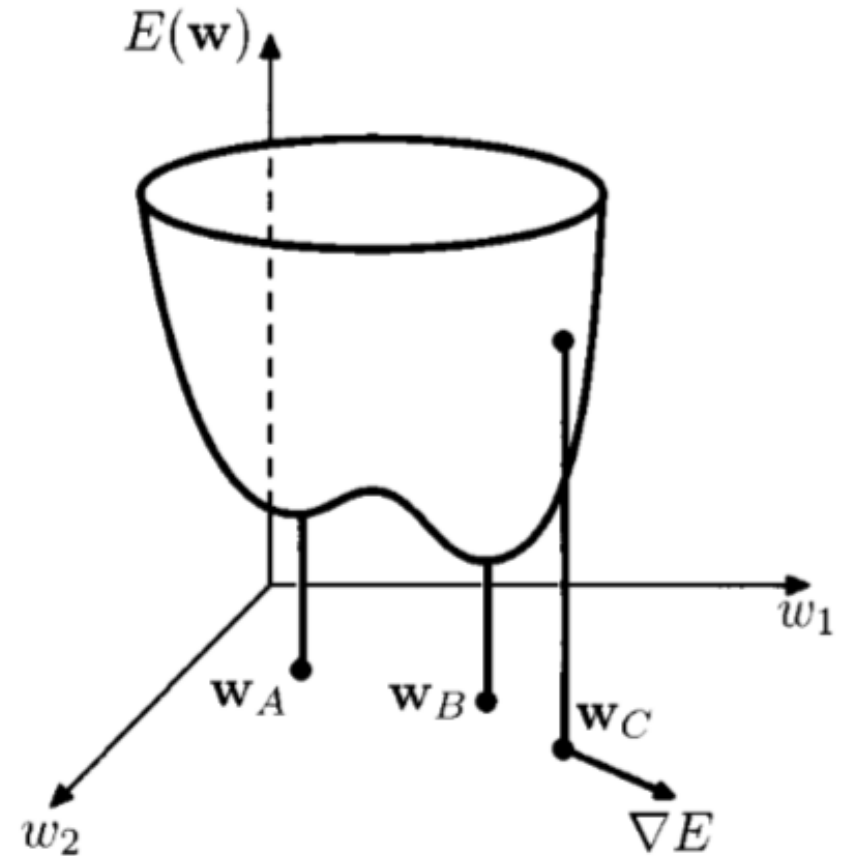
Algoritmo Del Gradiente Descendente

- **Inicialización:** Selecciona valores iniciales para los pesos y los sesgos del modelo.



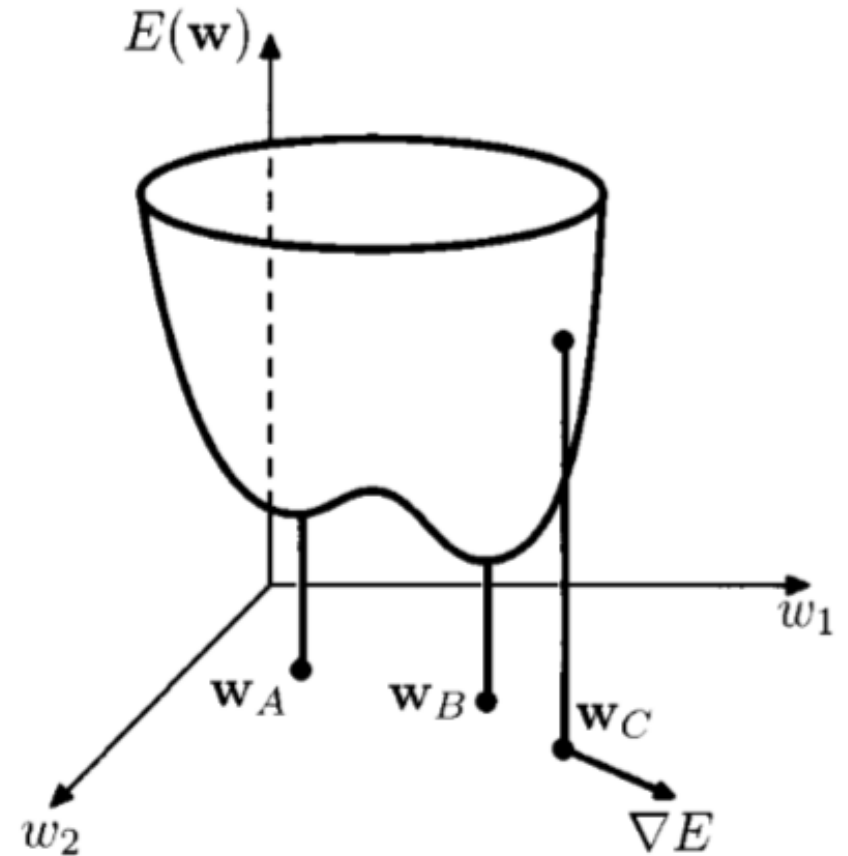
Algoritmo Del Gradiente Descendente

- **Evaluación del modelo:** Utiliza los valores actuales de los pesos y los sesgos para calcular las predicciones del modelo sobre un conjunto de datos de entrenamiento.



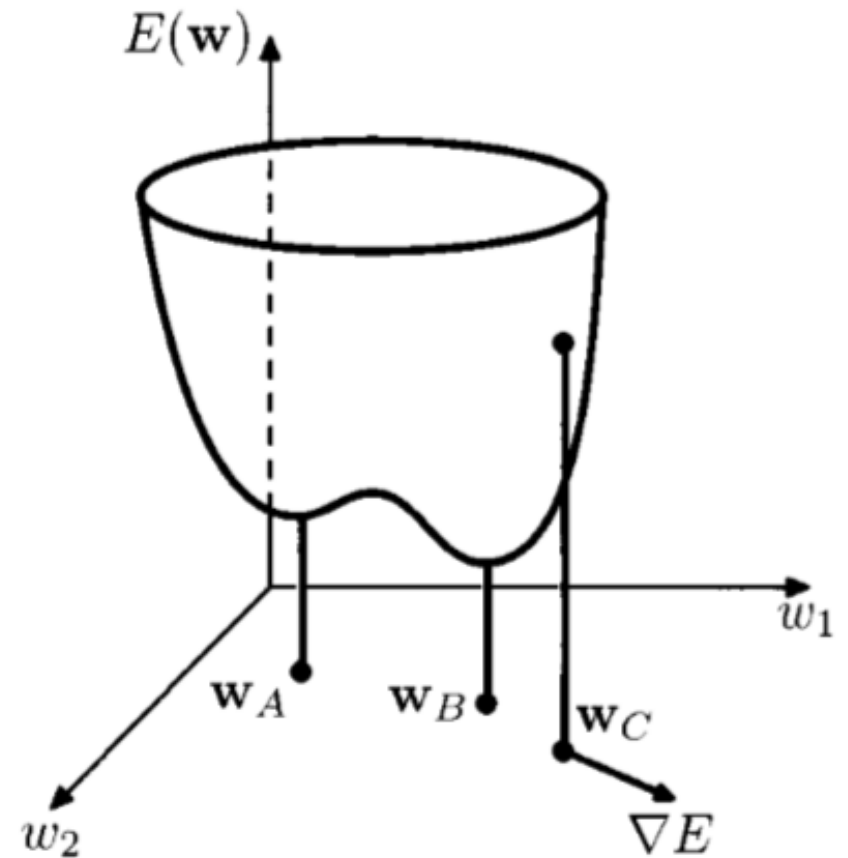
Algoritmo Del Gradiente Descendente

- **Cálculo del gradiente:** Calcula el gradiente de la función de pérdida con respecto a los pesos y los sesgos.
- El gradiente indica la dirección y la magnitud del cambio más pronunciado en la función de pérdida.



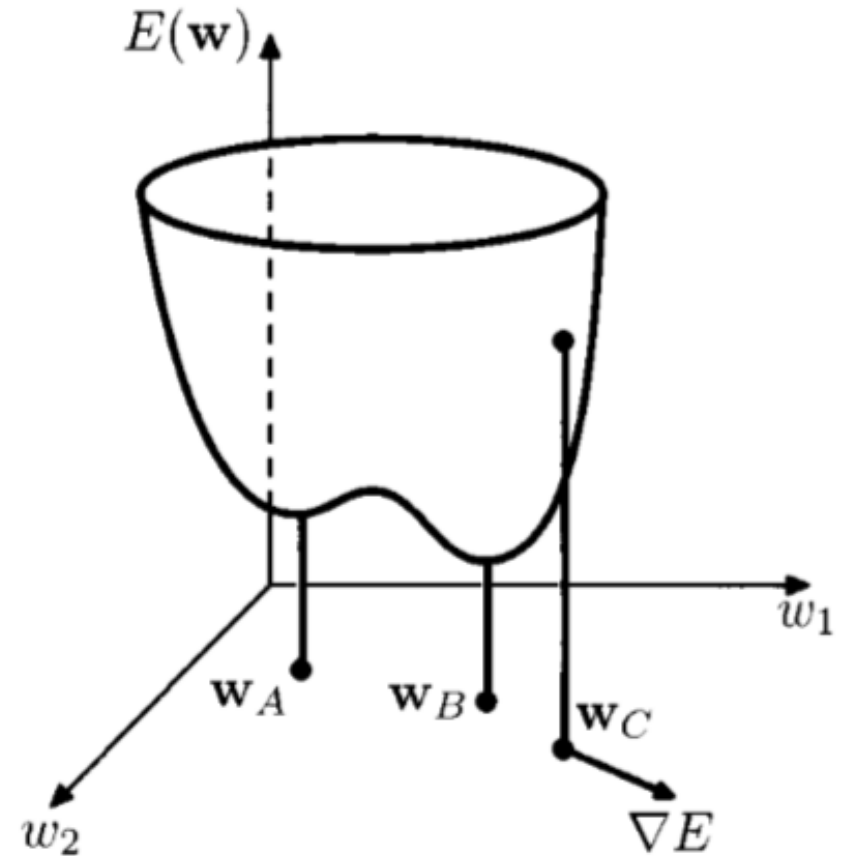
Algoritmo Del Gradiente Descendente

- **Actualización de los pesos y sesgos:** Ajusta los valores de los pesos y los sesgos utilizando el gradiente calculado. El objetivo es moverse en la dirección opuesta al gradiente para minimizar la función de pérdida.



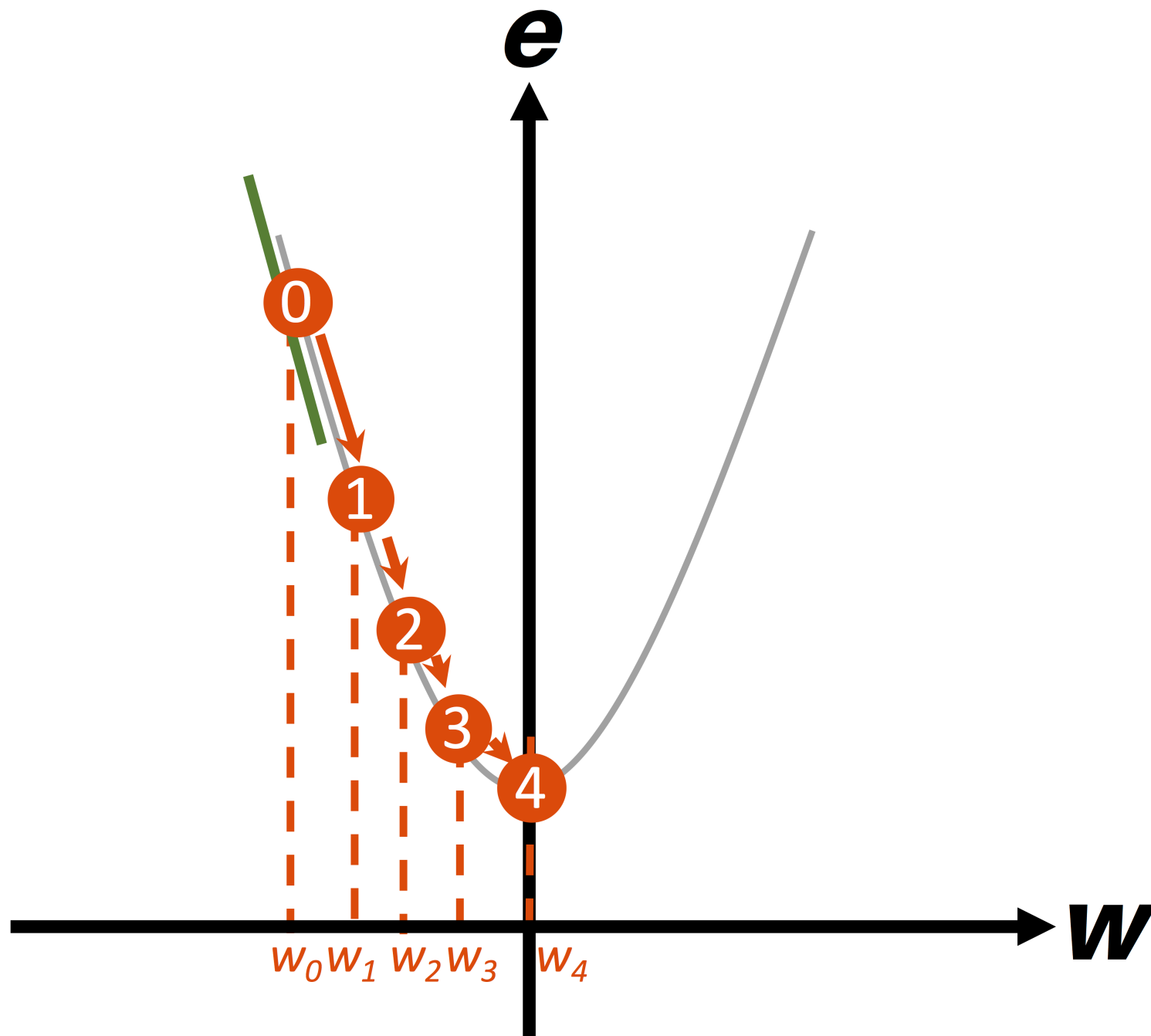
Algoritmo Del Gradiente Descendente

- **Repetición del proceso:** Repite los pasos 2 a 5 hasta que se cumpla un criterio de parada, como un número máximo de iteraciones o una convergencia satisfactoria de la función de pérdida.



El algoritmo del Gradiente Descendente se basa en la idea de buscar la dirección de mayor descenso en la función de pérdida para encontrar un mínimo local o global.

Siguiendo el gradiente negativo, el algoritmo ajusta los pesos y los sesgos en cada iteración, lo que conduce a una mejora gradual en el rendimiento del modelo.



¿Como se trabaja con las redes neuronales ?

