Lección 12:
Redes Neuronales Artificiales

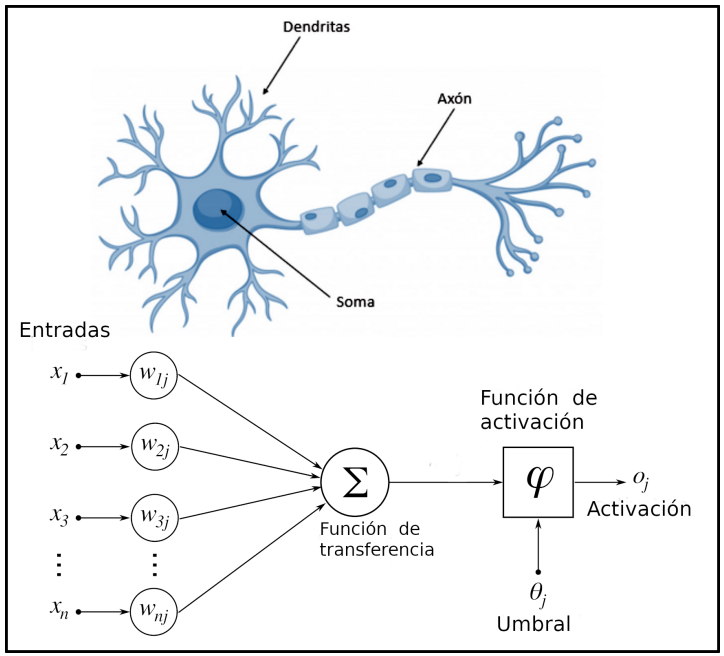
Jesús Savage

21 de abril de 2023

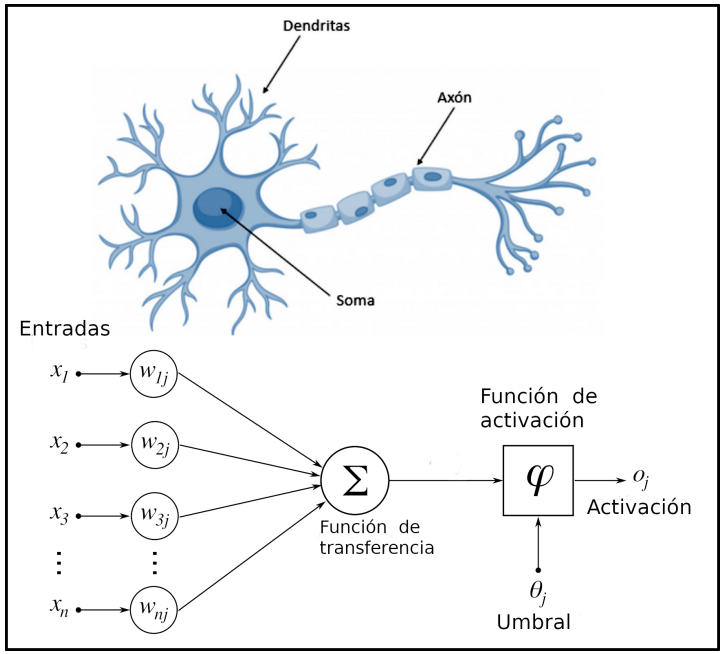
Índice

Modelos de Redes Neuronales Artificiales

El modelo matemático del comportamiento de una neurona fue propuesto en 1943 por McCullock. En este modelo la neurona cuenta con varias entradas las cuales están multiplicadas por ciertos pesos, éstas se suman y si la suma es más grande que un umbral, θ, la neurona se activa, en caso contrario se encuentra desactivada.



Modelos de Redes Neuronales Artificiales



Las entradas de la neurona son x_i, las cuales pueden provenir de las entradas al sistema o de las salidas de otras neuronas.
Los pesos w_i multiplican cada una de las entradas.

$$u = \sum_{i=1}^{n} w_i \, x_i - \theta$$

Redes Neuronales Artificiales

El umbral de activación o sesgo, θ, se puede incorporar como una entrada con $x_0 = 1$, multiplicada por una constante $w_0 = -\theta$

$$u = \sum_{i=0}^{n} w_i \, x_i$$

La salida de la neurona es generada por una función de activación no lineal $y = f(u)$. Generalmente $f(u)$ es una función sigmoide, como se muestra a continuación:

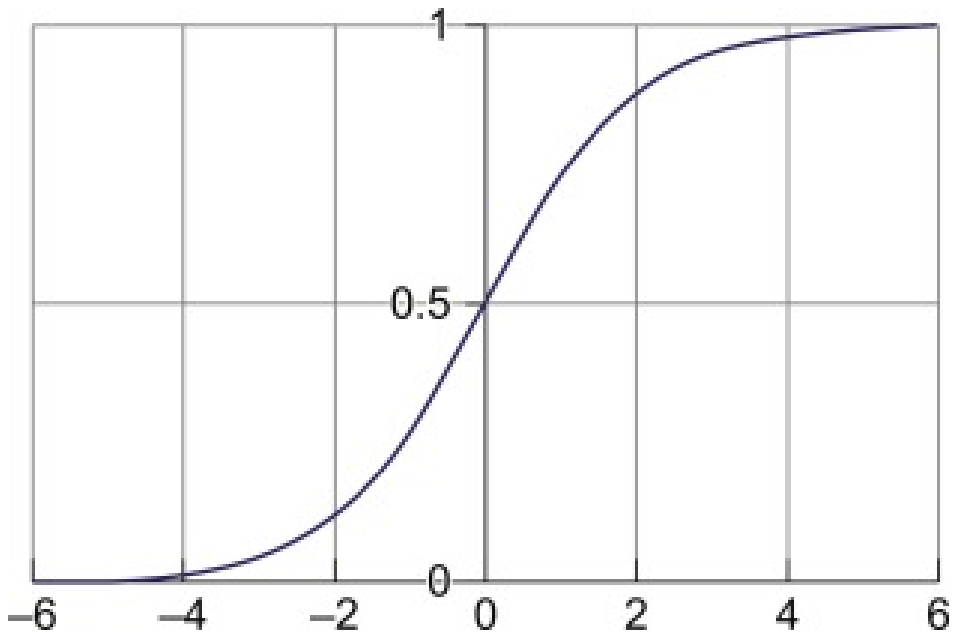
$$f(u) = \frac{1}{1 + e^{\frac{-u}{T}}}$$

Donde T es una constante.

Función Sigmoide

El uso de una función no lineal produce límites no lineales, por lo tanto, la función sigmoide se puede usar en redes neuronales para aprender funciones de decisión complejas.

$$f(u) = \frac{1}{1 + e^{\frac{-u}{T}}}$$



Función de Activación tipo Sigmoide

Una característica importante de $f(u)$ es que su derivada esta en función de ella misma.

$$\frac{df(u)}{du} = \frac{d}{du}\left(\frac{1}{1+e^{\frac{-u}{T}}}\right) = \frac{d}{du}(1+e^{\frac{-u}{T}})^{-1}$$

$$= -(1+e^{\frac{-u}{T}})^{-2} \cdot \frac{-1}{T}(e^{\frac{-u}{T}})$$

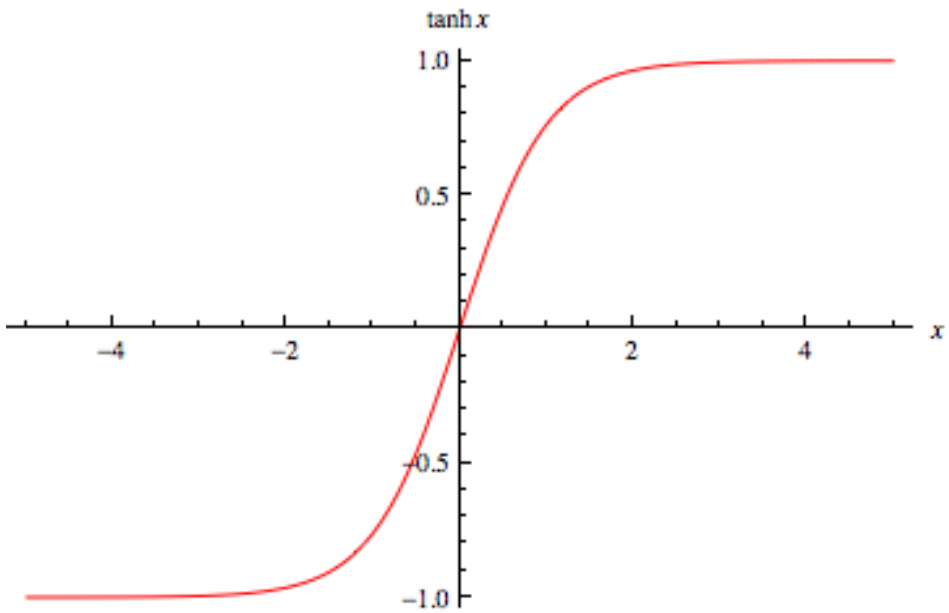
$$= \frac{1}{T} \cdot \frac{1}{1+e^{\frac{-u}{T}}} \cdot \frac{e^{\frac{-u}{T}}}{1+e^{\frac{-u}{T}}} = \frac{1}{T} \cdot \frac{1}{1+e^{\frac{-u}{T}}} \cdot \frac{1+e^{\frac{-u}{T}}-1}{1+e^{\frac{-u}{T}}}$$

$$= \frac{1}{T} \cdot \frac{1}{1+e^{\frac{-u}{T}}} \cdot \left(\frac{1+e^{\frac{-u}{T}}}{1+e^{\frac{-u}{T}}} - \frac{1}{1+e^{\frac{-u}{T}}}\right) = \frac{1}{T} f(u)(1-f(u))$$

Función de Activación de Tangente Hiperbólica

Otra función de activación muy usada también en las redes neuronales es la tangente hiperbólica.

$$f(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$$



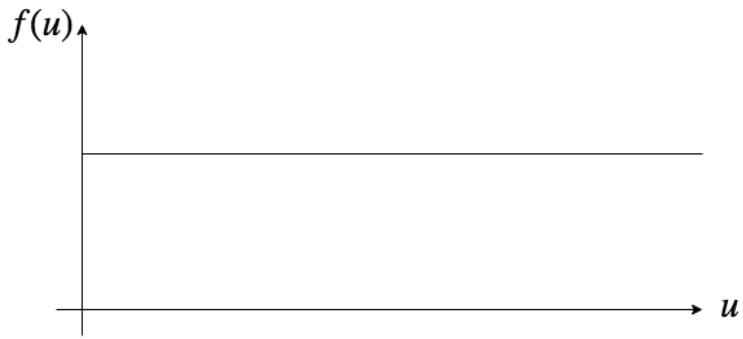
Modelo del Perceptrón

El modelo del Perceptrón fue propuesto por Rosenblatt y puede ser utilizado para *detección y clasificación*.

$$u = \sum_{i=1}^{n} w_i x_i - \theta$$

En este modelo la función de activación $f(u)$ es un escalón.

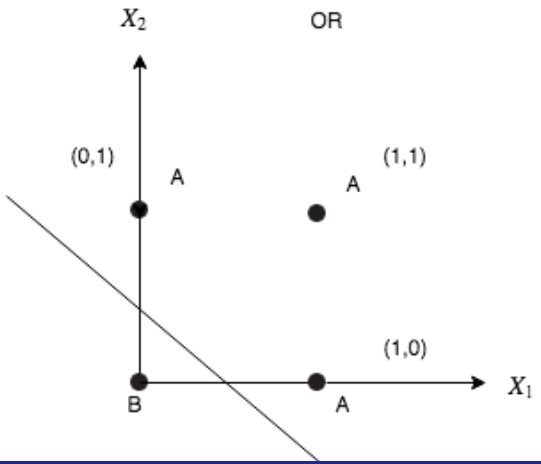
$$y = \begin{cases} 1, & u \geq 0 \\ 0, & u < 0 \end{cases}$$



Modelo del Perceptrón

Desde que fue propuesto el Perceptrón se probó tratando de resolver problemas básicos desde el punto de vista de un classificador, como el resolver las ecuaciones booleanas OR, AND y XOR.
Para una compuerta OR:

X_1, X_2	OR	Clase
0 0	0	B
0 1	1	A
1 0	1	A
1 1	1	A



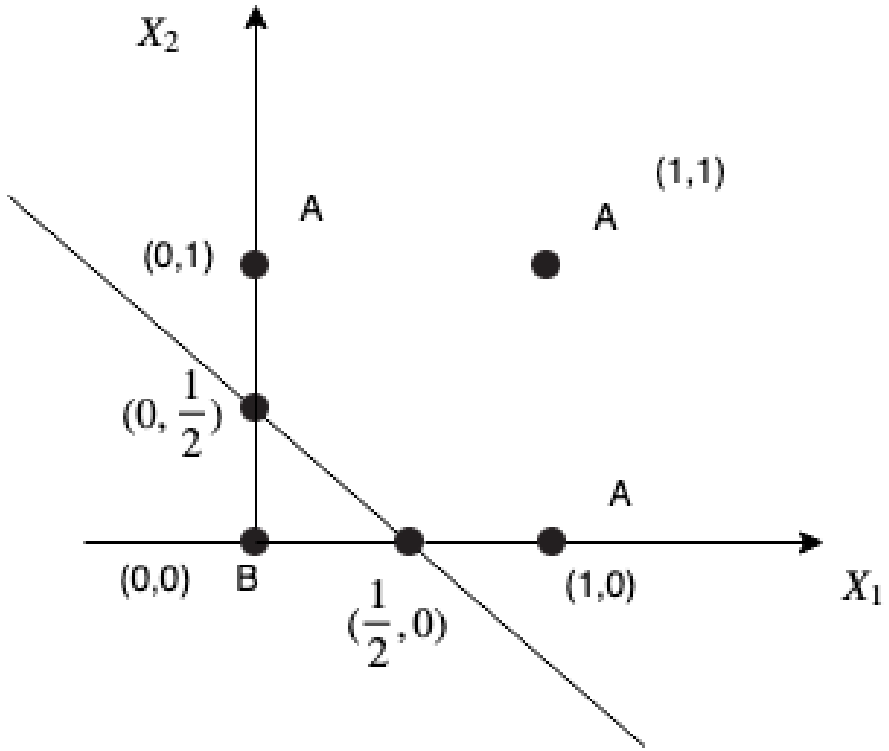
Compuerta Lógica OR

La recta que
separa las dos clases es:

$$y = m\,x\ +\ b =$$

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

$$b = \frac{y_1\ x_2 - y_2\ x_1}{x_2 - x_1}$$



Con:
$(x_1, y_1) = (\frac{1}{2}, 0)$,
$(x_2, y_2) = (0, \frac{1}{2})$

Compuerta Lógica OR

$$m = \frac{\frac{1}{2}}{-\frac{1}{2}} = -1, b = \frac{-\frac{1}{4}}{-\frac{1}{2}} = \frac{1}{2}$$

Por lo tanto la ecuación de la recta que separa las dos clases es:

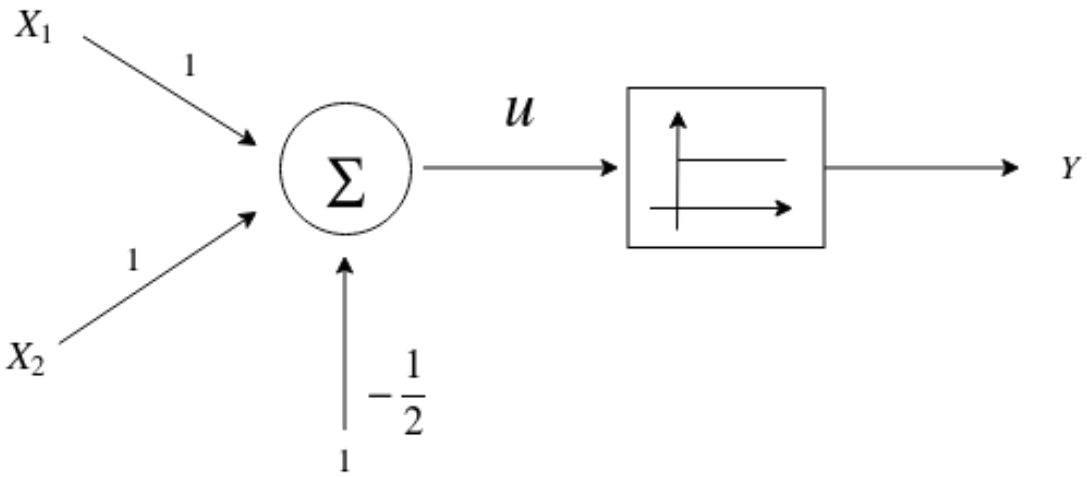
$$y = -x + \frac{1}{2}$$

$$y + x - \frac{1}{2} = 0$$

Para una neurona que haga la separación del espacio

$$f(w_2\, X_2 + w_1\, X_1 - \; w_0)$$

Compuerta Lógica OR Usando un Perceptrón



Con

$$w_2 = 1, \quad w_1 = 1, \quad w_0 = -\frac{1}{2}$$

$$f(X_2 + X_1 - \frac{1}{2})$$

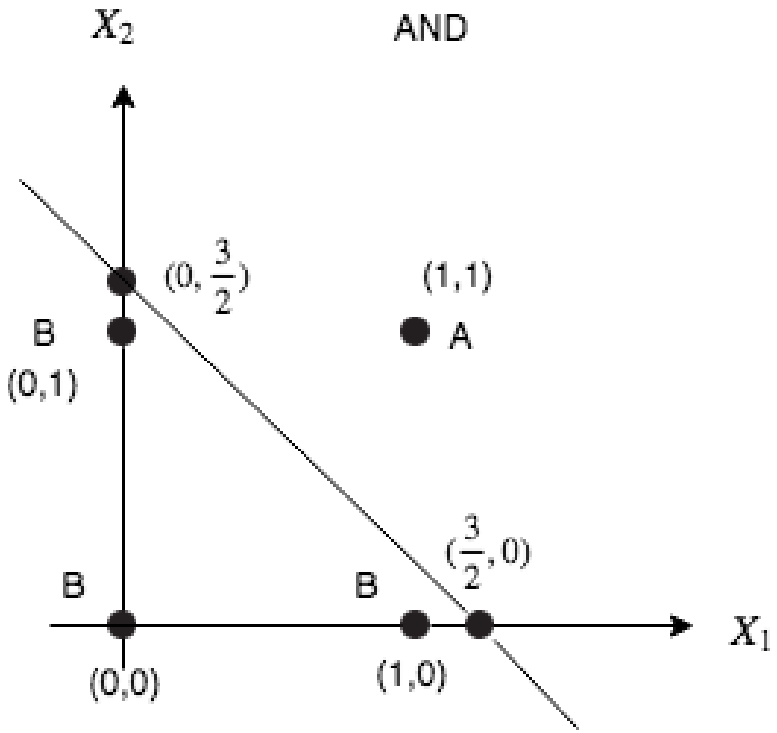
Compuerta Lógica OR Usando un Perceptrón

$$
\begin{array}{lll}
f(X_2 + X_1 - \frac{1}{2}) & & \\
0+0-\frac{1}{2} \leq & 0 & \Rightarrow B \\
0+1-\frac{1}{2} \geq & 0 & \Rightarrow A \\
1+0-\frac{1}{2} \geq & 0 & \Rightarrow A \\
1+1-\frac{1}{2} \geq & 0 & \Rightarrow A
\end{array}
$$

X_1, X_2	u	$NN - OR$
0 0	$-\frac{1}{2}$	0
0 1	$+\frac{1}{2}$	1
1 0	$+\frac{1}{2}$	1
1 1	$+\frac{3}{2}$	1

Compuerta Lógica AND

X_1, X_2	AND	Clase
0 0	0	B
0 1	0	B
1 0	0	B
1 1	1	A

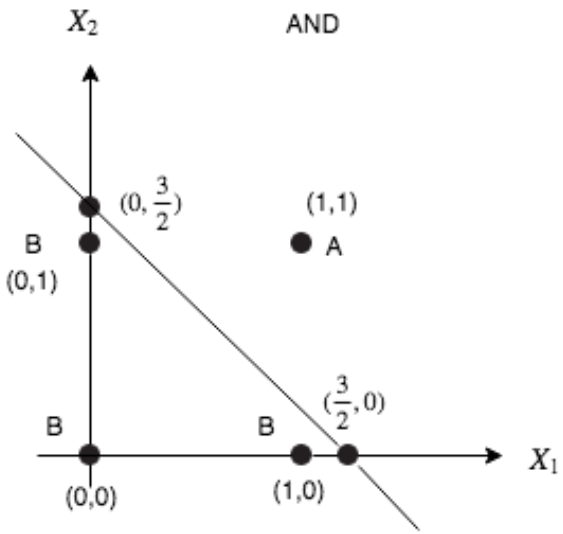


Compuerta Lógica AND

Con
$(x_1, y_1) = (\frac{3}{2}, 0)$
$(x_2, y_2) = (0, \frac{3}{2})$

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} = \frac{\frac{3}{2}}{-\frac{3}{2}} = -1$$

$$b = \frac{y_1\ x_2 - y_2\ x_1}{x_2 - x_1} = \frac{-\frac{9}{4}}{-\frac{3}{2}} = \frac{3}{2}$$



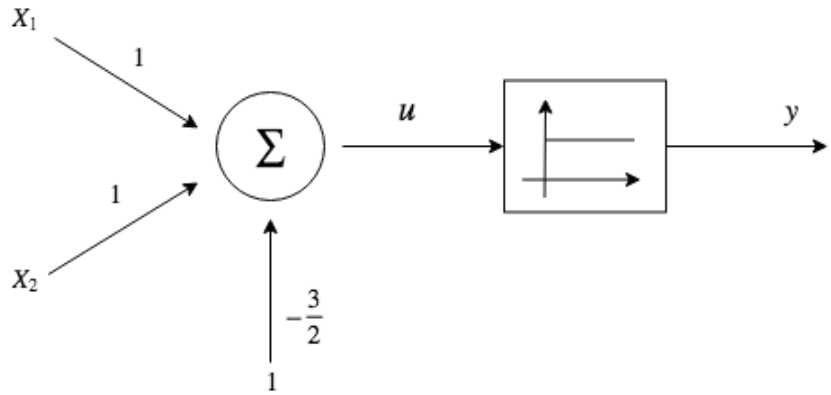
Compuerta Lógica AND Usando un Perceptrón

$$y = mx + b = -x + \frac{3}{2}$$

$$y + x - \frac{3}{2} = 0$$

$$f(w_2\, X_2 + w_1\, X_1 - \; w_0)$$

$$= f(X_2 + X_1 - \frac{3}{2})$$



Compuerta Lógica AND Usando un Perceptrón

$$X_2 \quad + \quad X_1 \quad - \quad \frac{3}{2}$$

$$0 + 0 - \frac{3}{2} \leq 0 \quad \Rightarrow B$$

$$0 + 1 - \frac{3}{2} \leq 0 \quad \Rightarrow B$$

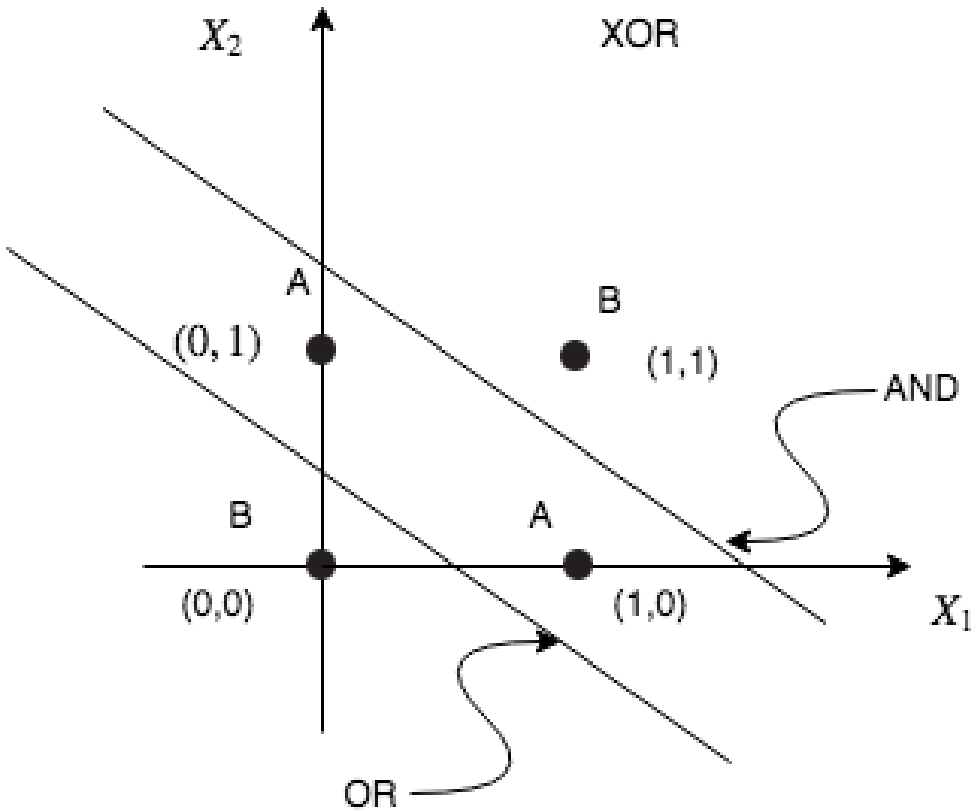
$$1 + 0 - \frac{3}{2} \leq 0 \quad \Rightarrow B$$

$$1 + 1 - \frac{3}{2} > 0 \quad \Rightarrow A$$

X_1, X_2	u	$NN - AND$
0 0	$-\frac{3}{2}$	0
0 1	$-\frac{1}{2}$	0
1 0	$-\frac{1}{2}$	0
1 1	$\frac{1}{2}$	1

Compuerta Lógica XOR

No se puede construir una compuerta XOR con una sola neurona, ya que el espacio se tiene que separar en tres regiones diferentes, por lo que se necesitan dos neuronas para lograr ésto.



Compuerta Lógica XOR

Se puede observar que la primera recta cumple con la ecuación de una OR y la segunda con la de una AND

OR

$$w_2 = 1, \quad w_1 = 1, \quad w_0 = -\frac{1}{2}$$

$$f(X_2 + X_1 - \frac{1}{2}) = Y1$$

AND

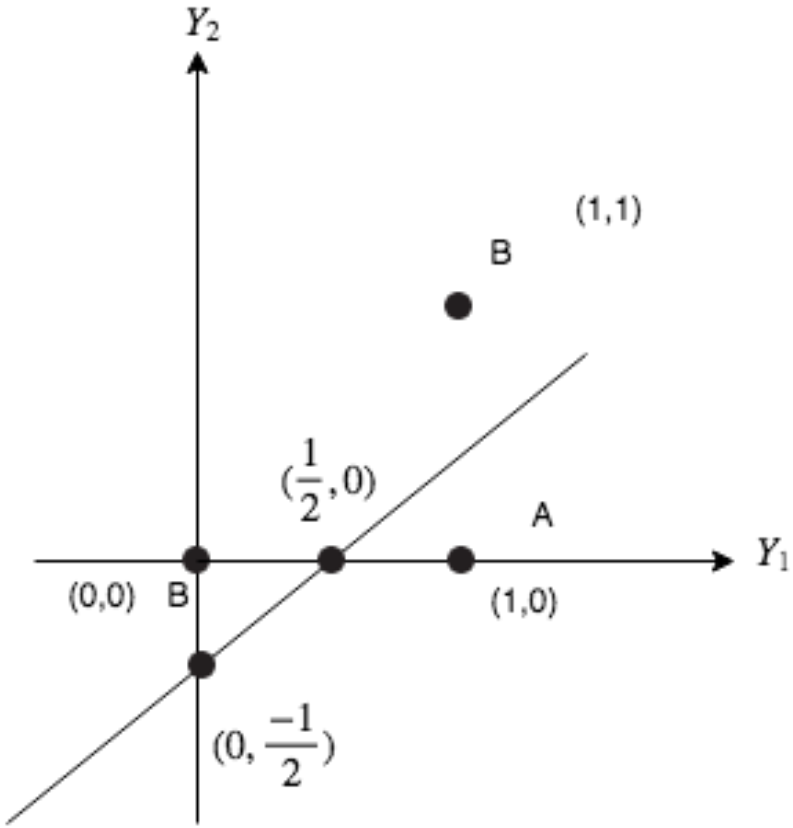
$$w_2 = 1, \quad w_1 = 1, \quad w_0 = -\frac{3}{2}$$

$$f(X_2 + X_1 - \frac{3}{2}) = Y2$$

	NN OR , AND	N XOR
X_1, X_2	Y_1, Y_2	Y
0 0	0 0	B 0
0 1	1 0	A 1
1 0	1 0	A 1
1 1	1 1	B 0

Transformación XOR

Con esta transformación se puede observar como las dos clases quedan separadas por una recta



Compuerta Lógica XOR Usando Perceptrones

$$(x_1, y_1) = (\frac{1}{2}, 0)$$

$$(x_2, y_2) = (0, -\frac{1}{2})$$

$$m = \frac{(-\frac{1}{2})}{-\frac{1}{2}} = +1$$

$$b = \frac{+\frac{1}{4}}{-\frac{1}{2}} = -\frac{1}{2}$$

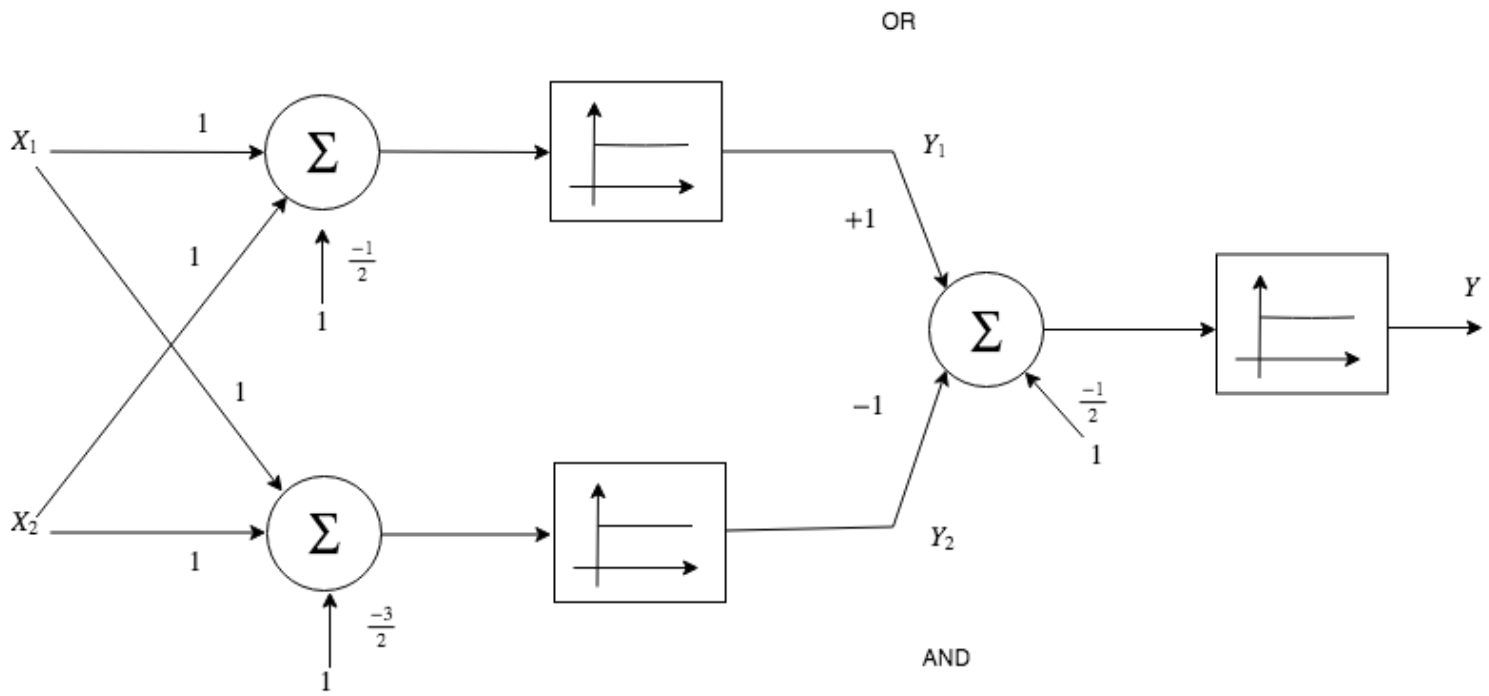
$$y - x + \frac{1}{2} = 0$$

$$f(w_2 Y_2 - w_1 Y_1 + w_0)$$

Compuerta Lógica XOR Usando Perceptrones

Ecuación de la recta que cumple con la desigualdad, pero no para ponerla en la NN.

$$Y_2 - Y_1 + \frac{1}{2} = 0 \Rightarrow -Y_2 + Y_1 - \frac{1}{2} = 0$$



Compuerta Lógica XOR Usando Perceptrones

$$-Y_2 \quad + \quad Y_1 \quad - \quad \frac{1}{2}$$

$$0 + 0 - \frac{1}{2} \leq 0 \quad \Rightarrow B$$

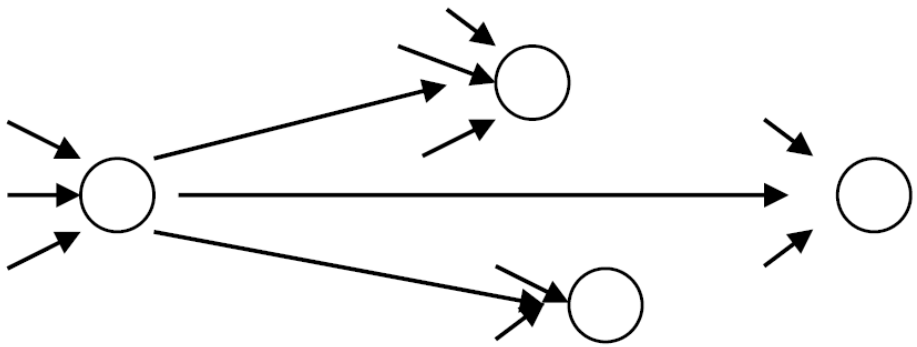
$$0 + 1 - \frac{1}{2} > 0 \quad \Rightarrow A$$

$$-1 + 1 - \frac{1}{2} \leq 0 \quad \Rightarrow B$$

	NN		
	OR , AND		
X_1, X_2	Y_1, Y_2	$u \to f(u)$	NN-XOR
0 0	$\frac{-1}{2} \to 0$, $\frac{-3}{2} \to 0$	$\frac{-1}{2} \to 0$	0
0 1	$\frac{1}{2} \to 1$, $\frac{-1}{2} \to 0$	$\frac{1}{2} \to 1$	1
1 0	$\frac{1}{2} \to 1$, $\frac{-1}{2} \to 0$	$\frac{1}{2} \to 1$	1
1 1	$\frac{3}{2} \to 1$, $\frac{1}{2} \to 1$	$\frac{-1}{2} \to 0$	0

Redes Neuronales Artificiales

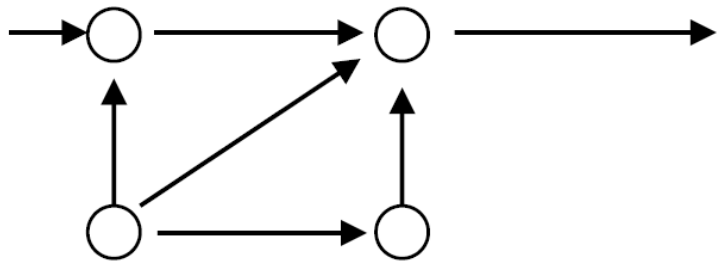
Si las salidas de un neurona se conectan a las entradas de otras se forma un red neuronal artificial.



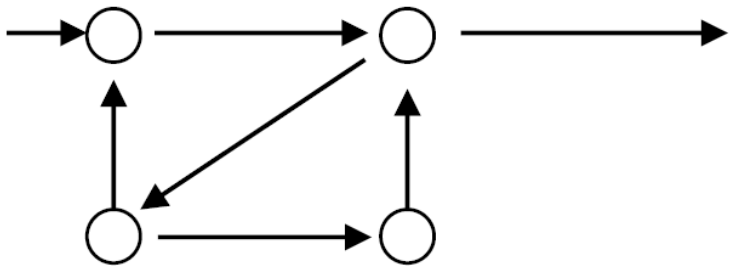
En la robótica de servicio se les pueda dar varios usos a las redes neuronales, como son el reconocimiento de patrones o para que los robots tengan comportamientos nuevos. El objetivo es encontrar los pesos W de la red que permitan tener estos comportamientos.

Topología de Redes Neuronales

En las topología acíclica no hay retroalimentación de la neurona de salida hacia otra interna.

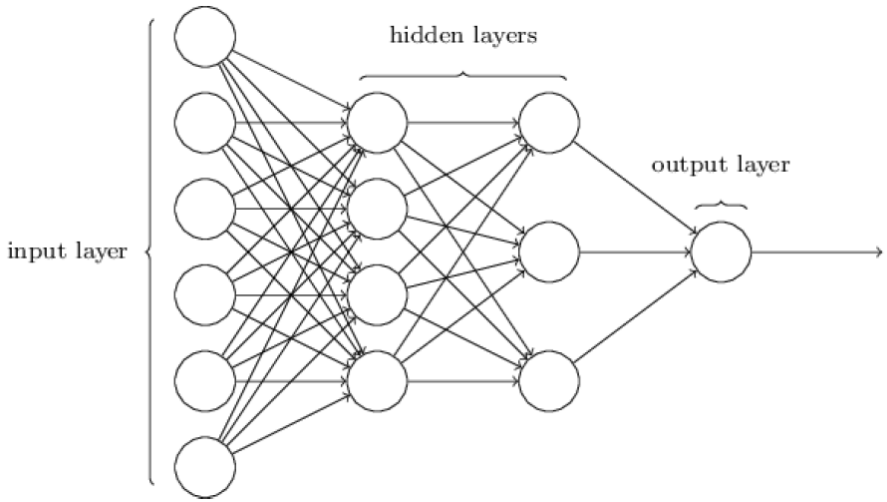


Topología cíclica hay retroalimentación de la neurona de salida con otras internas.



Redes Neuronales Artificiales

En una red neuronal multicapas se tienen por lo regular tres capas: una de entradas, una intermedia y una de salida.



La capa de entrada tiene M neuronas. La capa intermedia se conoce como *capa oculta* y tiene H neuronas. La capa de salida tiene N neuronas.
El objetivo es encontrar los pesos que minimicen el error cuadrático medio de acuerdo a un patrón que se desea aprender.

¿Cómo encontrar los pesos?

Para una neurona con M entradas se desea que la neurona aprenda a generar una salida de acuerdo a una muestras de entrenamiento:

$$\{(\underline{x}_k, d_k); 1 \leq k \leq K\}$$

donde las $\underline{x}_k^T = [1 \quad x_{1k} \quad x_{2k} ... \quad x_{Mk}]^T$ son las entradas a la neurona en el tiempo k y las d_k son las salidas deseadas para esas entradas y K es el número total de muestras.

Muestras de entrenamiento

Por ejemplo se tienen las siguientes muestras de entrenamiento, en este caso la neurona se entrena con los valores d_k para que se active con un valor cercano a uno, 0.99, y que no este activada con valores cercanos a cero, 0.01.

Cuadro: Muestras de entrenamiento

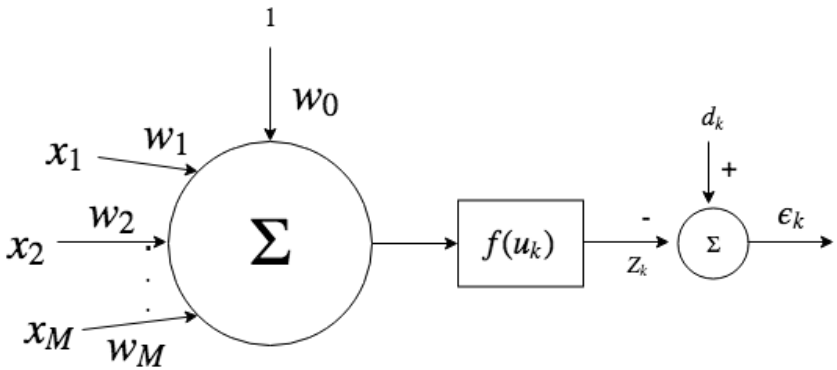
k	x_0	x_1	x_2	x_3	...	x_M	d
1	1	.14	.21	.21	...	.01	0.99
2	1	.15	.15	.07	...	.04	0.99
....	...	...	...	...	...	...	...
K-1	1	.35	.31	.41	...	.80	.01
K	1	.33	.36	.40	...	.84	.01

Calculo de los pesos

Si las $\underline{W} = [w_0 \quad w_1 \quad w_2 ... \quad w_M]$ son los pesos de la neurona, entonces:

$$u_k = \sum_{i=0}^{n} w_i \, x_{ik}$$

La salida de la neurona es: $z_k = f(u_k) = f(\underline{W} \cdot \underline{x}_k)$
El error generado por la neurona para la muestra kesima k es: $\varepsilon_k = d_k - z_k$



Generalmente los valores de $d_k \in (0,1)$.

Metodo Iterativo para Encontrar los Pesos

El error total es:

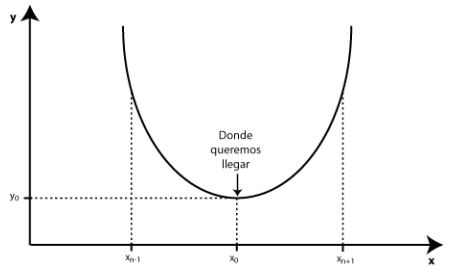
$$E = \sum_{k=1}^{K} \varepsilon_k^2 = \sum_{k=1}^{K} (d_k - z_k)^2$$

El objetivo final es minimizar la función del error total E con respecto a $\underline{W}$. Esta minimización lleva a un problema de optimización no lineal de mínimos cuadrados, ya que $f()$ es una función no lineal, por lo que generalmente se realiza la minimización con metodos iterativos: Empezando con un punto w_{n-1} cualquiera, el objetivo es llegar al punto mínimo de la ecuación w_0 utilizando una función recurrente:

$$w_n = h(w_{n-1})$$

Descendiendo por la pendiente más pronunciada

Por ejemplo, para una parábola, como se muestra en la siguiente figura:



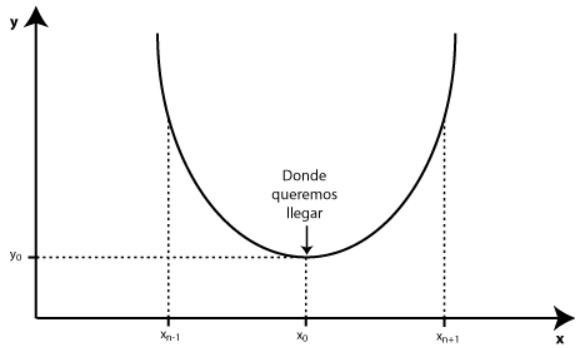
Para encontrar el mínimo de la función, se observa que empezando en un punto w_{n-1} a la izquierda del punto w_0, en donde se encuentra el mínimo de la función, se tendría que sumar a w_{n-1} una cierta cantidad para acercarse a w_0. Por otra parte si w_{n-1} esta a la izquierda de w_0, se le tendría que restar también una cierta cantidad para acercarse a w_0. Se puede observar que la cantidad a sumar o restar esta inversamente relacionada con la derivada de la función, por lo tanto una función que acerca al punto w_{n-1} a w_0 es:

$$w_n = h(w_{n-1}) = w_{n-1} - \delta \frac{dy}{dw}$$

En donde δ es el tamaño del paso.

Descendiendo por la pendiente más pronunciada

La ecuación de la parabola es:
$y = y_0 + (w - w_0)^2$



$$\frac{dy}{dw} = 2(w - w_0)$$

$$w_n = h(w_{n-1}) = w_{n-1} - \delta \frac{dy}{dw}$$

entonces, si $\delta = \frac{1}{2} \Rightarrow w_n = w_{n-1} - \frac{1}{2}(2(w_{n-1} - w_0)) = w_0,$
Se llega al mínimo en un paso.
Esta técnica se conoce como descendiendo por la pendiente más pronunciada o steepest descent.

Metodo Iterativo para Encontrar los Pesos

Por lo tanto para encontrar el vector de las $\underline{W}$ óptimas se utilizara el método de descendiendo por la pendiente más pronunciada (*steepest descent):*

$$\underline{W}_{t+1} = \underline{W}_t + \Delta \underline{W}_t$$

$$\Delta \underline{W}_t = -\eta \frac{dE}{d\underline{W}}$$

Recordar que el error total es:

$$E = \sum_{k=1}^{K} \varepsilon_k^2 = \sum_{k=1}^{K} (d_k - z_k)^2$$

$$\frac{dE}{dw_i} = -2 \sum_{k=1}^{K} (d_k - z_k)(\frac{dz_k}{dw_i})$$

Metodo Iterativo para Encontrar los Pesos

Obsérvese que

$$\frac{dz_k}{dw_i} = \frac{df(u_k)}{dw_i}$$

Recordando que

$$u_k = \sum_{i=0}^{M} w_i \, x_{ik}$$

y utilizando el método de la cadena:

$$\frac{df(u_k)}{du_k}\frac{du_k}{dw_i} = f'(u)\frac{du_k}{dw_i} = f'(u_k)x_{ik}$$

Por lo que

$$\frac{dE}{dw_i} = -2\sum_{k=1}^{K}(d_k - z_k)f'(u_k)x_{ik}$$

Metodo Iterativo para Encontrar los Pesos

Haciendo

$$\delta_k = 2[d_k - z_k]f'(u_k)$$

que es el error $e_k = d_k - z_k$ modulado por la derivada de la función de activación $f'(u_k)$.
Entonces

$$\frac{dE}{dw_i} = -\sum_{k=1}^{K} \delta_k x_{ik}$$

Por lo tanto la actualización esta dada por la siguiente expresión:

$$w_i(t+1) = w_i(t) + \eta \sum_{k=1}^{K} \delta_k x_{ik}$$

Donde η es una constante que da la velocidad de convergencia y se encuentra en forma empiricamente. t es el tiempo de iteración y el proceso se debe repetir hasta que el error total sea: $E<\epsilon$.

Metodo Iterativo para Encontrar los Pesos

En particular si $f(u)$ es una función sigmoide

$$f(u) = \frac{1}{1 + e^{\frac{-u}{T}}}$$

Donde T es una constante.

$$\frac{df(u)}{du} = \frac{1}{T} f(u)(1 - f(u))$$

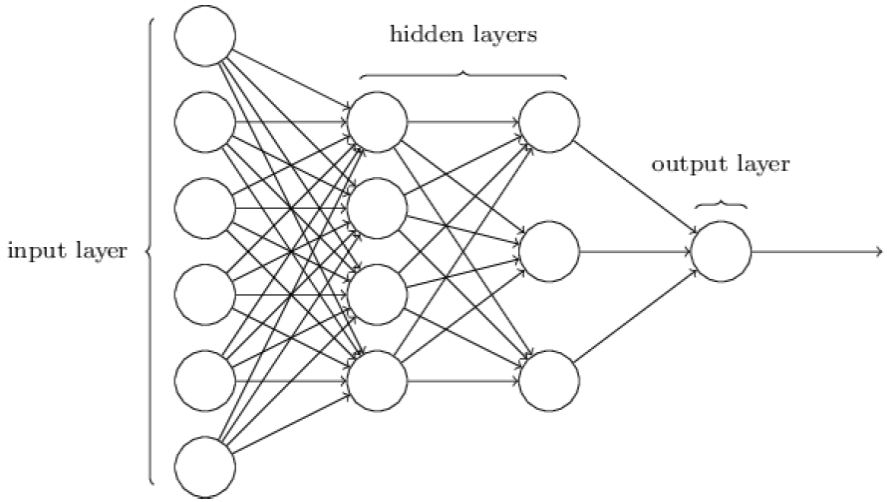
entonces:

$$\delta_k = \frac{2}{T}[d_k - z_k]\, f(u_k)\, (1 - f(u_k))$$

$$= \frac{2}{T}[d_k - z_k]\, z_k\, (1 - z_k)$$

Red Neuronal Multicapa

Para encontrar los pesos en una red neuronal multicapa, como la mostrada en la siguiente figura, se procede de la siguiente manera.

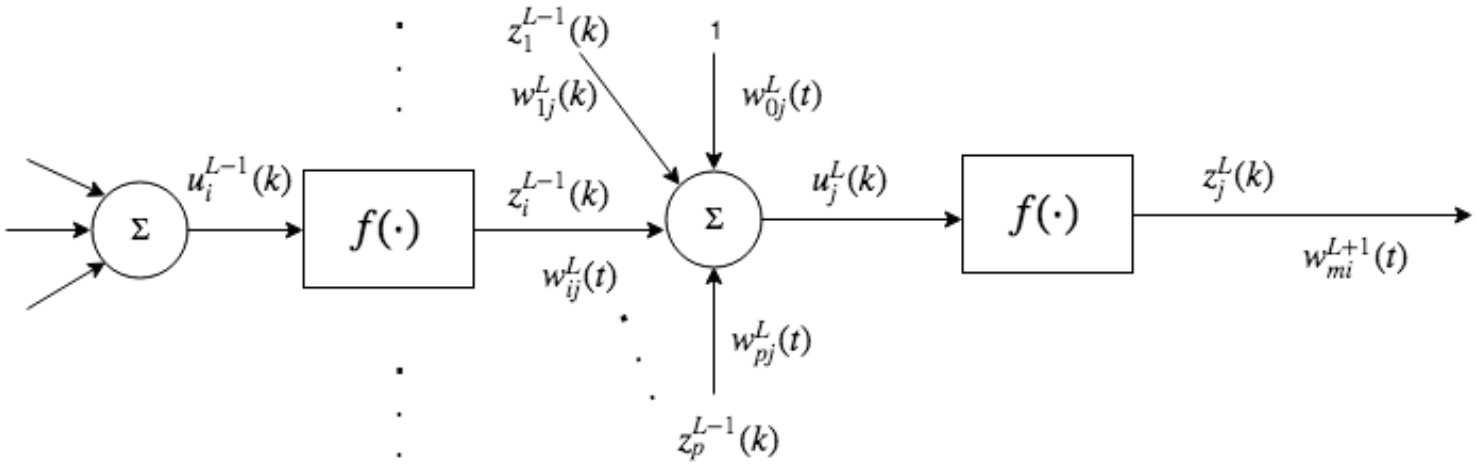


Red Neuronal Multicapa

La neurona j, en la capa L, para la muestra de entrenamiento k genera el valor z_{jk}^{L}.

$$z_{jk}^{L} = f(u_{jk}) = f\Big(\sum_{i=0}^{p} w_{ij}^{L}\, z_{ik}^{L-1}\Big)$$

Para la siguiente figura, w_{ij}^{L} es el peso que multiplica la salida de la neurona i, en la capa $L-1$ a la neurona j en la capa L, en la iteración t.



Red Neuronal Multicapa

El objetivo es encontrar los pesos w_{ij}^L de cada capa L que minimicen el error cuadrático medio total. El error entre los valores generados por la neurona j, en la capa L, para la muestra de entrenamiento k, es decir, z_{jk}^L, y los valores deseados d_{jk}^L:

$$E_j^L = \sum_{k=1}^{K} (d_{jk}^L - z_{jk}^L)^2$$

Este error se minimiza encontrando los pesos w_{ij}^L óptimos.
Encontrando la parcial del error con respecto a w_{ij}^L:

$$\frac{\partial E_j^L}{\partial w_{ij}^L} = -2 \sum_{k=1}^{K} (d_{jk}^L - z_{jk}^L) \frac{\partial z_{jk}^L}{\partial w_{ij}^L}$$

Red Neuronal Multicapa

Usando la regla de la cadena:

$$\frac{\partial E_j^L}{\partial w_{ij}^L} = -2\sum_{k=1}^{K}(d_{jk}^L - z_{jk}^L)\frac{\partial z_{jk}^L}{\partial u_{jk}^L}\frac{\partial u_{jk}^L}{\partial w_{ij}^L} = -2\sum_{k=1}^{K}(d_{jk}^L - z_{jk}^L)f'(u_{jk}^L)\frac{\partial u_{jk}^L}{\partial w_{ij}^L}$$

Usando la definición anterior de δ_{jk}:

$$\delta_{jk}^L = 2[d_{jk}^L - z_{jk}^L]f'(u_{jk}^L)$$

$$\frac{\partial E_j^L}{\partial w_{ij}^L} = -\sum_{k=1}^{K}\delta_{jk}^L\frac{\partial u_{jk}^L}{\partial w_{ij}^L}$$

$$= -\sum_{k=1}^{K}\delta_{jk}^L\frac{\partial}{\partial w_{ij}^L}\left(\sum_m w_{mj}^L z_{mk}^{L-1}\right)$$

Red Neuronal Multicapa

Entonces

$$\frac{\partial E_j^L}{\partial w_{ij}^L} = -\sum_{k=1}^{K} \delta_{jk}^L z_{ik}^{L-1}$$

Se observa que el error modulado por la deriva, δ_{jk}^L, no esta disponible ya que d_{jk}^L se desconoce para la capa L, las δ_{jk}^L solo se conocen para la capa de salida. Entonces los errores se necesitan ir acarreando de la capa de salida a las capas interiores, esto se conoce como retropropagación (*backpropagation*).

Red Neuronal Multicapa

La δ_{jk}^{L} puede ser obtenida también de la siguiente forma:
Definiendo el error del valor generado por la neurona j en la capa L con el valor deseado d_{jk}^{L}:

$$E_k = (d_{jk}^{L} - z_{jk}^{L})^2$$

$$\frac{\partial E_k}{\partial u_{jk}^{L}} = \frac{\partial}{\partial u_{jk}^{L}}(d_{jk}^{L} - z_{jk}^{L})^2$$

$$= -2(d_{jk}^{L} - z_{jk}^{L})\frac{\partial z_{jk}^{L}}{\partial u_{jk}^{L}}$$

$$= -2(d_{jk}^{L} - z_{jk}^{L})f'(u_{jk}^{L})$$

Usando la definición anterior de δ_{jk}:

$$\delta_{jk}^{L} = 2[d_{jk}^{L} - z_{jk}^{L}]f'(u_{jk}^{L})$$

Red Neuronal Multicapa

Entonces: $\dfrac{\partial E_k}{\partial u_{jk}^L} = -\delta_{jk}^L$

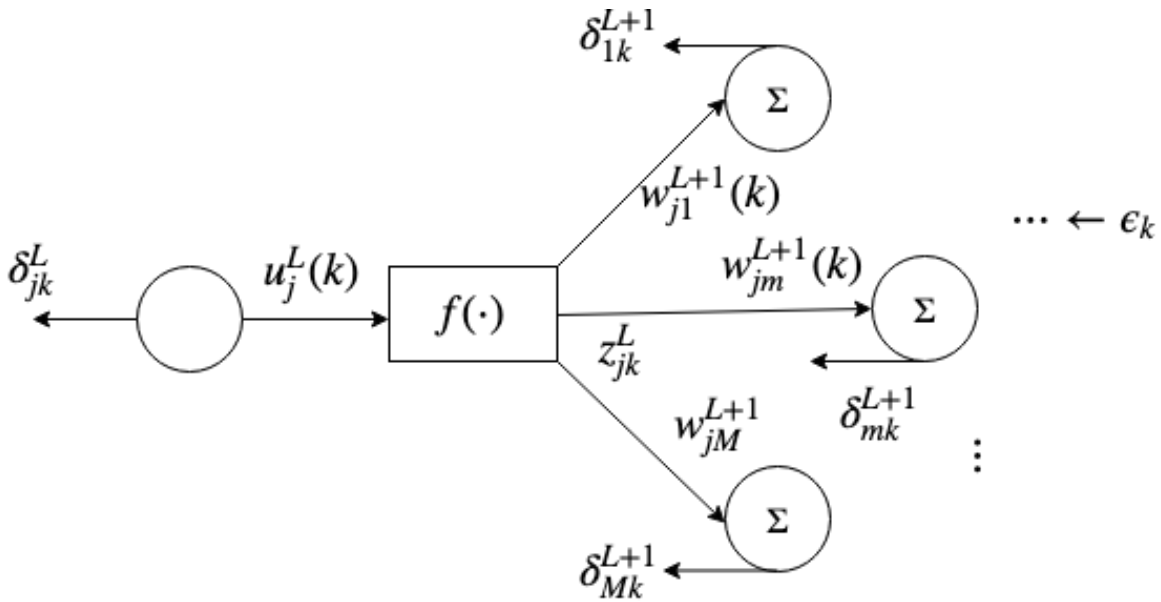
$$
\begin{aligned}
-\delta_{jk}^L = \frac{\partial E_k}{\partial u_{jk}^L} &= \frac{\partial E_k}{\partial u_{jk}^L} \cdot 1 \\
&= \frac{\partial E_k}{\partial u_{jk}^L} \left(\frac{1}{M} \sum_{m=1}^{M} \frac{\partial u_{mk}^{L+1}}{\partial u_{mk}^{L+1}} \right) \\
&= \frac{1}{M} \sum_{m=1}^{M} \frac{\partial E_k}{\partial u_{mk}^{L+1}} \frac{\partial u_{mk}^{L+1}}{\partial u_{jk}^L} \\
&= \frac{1}{M} \sum_{m=1}^{M} -\delta_{mk}^{L+1} \left[\frac{\partial}{\partial u_{jk}^L} \sum_{n=1}^{N} w_{nm}^{L+1} f(u_{nk}^L) \right] \\
&= \frac{1}{M} \sum_{m=1}^{M} -\delta_{mk}^{L+1} w_{jm}^{L+1} f'(u_{jk}^L)
\end{aligned}
$$

Red Neuronal Multicapa

Por lo tanto:

$$\delta_{jk}^{L} = f'(u_{jk}^{L}) \frac{1}{M} \sum_{m=1}^{M} \delta_{mk}^{L+1} w_{jm}^{L+1}$$

Las δ_{jk}^{L} dependen de las δ_{mk}^{L+1}, es decir que se necesitan estos valores que son calculados en la siguiente etapa.



Red Neuronal Multicapa

La δ_{jk}^{L} en la capa de salida se calcula por la diferencia del valor deseado d_{jk} de entrenamiento con el valor obtenido por la neurona de salida z_{jk} modulada por la derivada de la función de activación.

$$\delta_{jk}^{L} = 2 \cdot \left[d_{jk} - z_{jk}^{L} \right] z_{jk}^{L} \left[1 - z_{jk}^{L} \right]$$

Donde L es el valor de la última capa de salida.
Recordando que para una neurona la actualización de los pesos esta dado por:

$$w_i(t+1) = w_i(t) + \eta \sum_{i=1}^{K} \delta_k x_{ik}$$

Extendiendo esta expresión para una red neuronal multicapa, la fórmula de actualización es entonces:

$$w_{ij}^{L}(t+1) = w_{ij}^{L}(t) + \eta \sum_{k=1}^{K} \delta_{jk}^{L} z_{ik}^{L-1}$$

Red Neuronal Multicapa

A esta última ecuación se le agrega un termino para hacer que el sistema converja más rápido, comparando los valores anteriores de los pesos. Se el agrega tambien ruido aleatorio pequeño para sacar a los pesos de mínimos locales.

$$w_{ij}^{L}(t+1) = w_{ij}^{L}(t) + \eta \sum_{k=1}^{K} \delta_{jk}^{L} z_{ik}^{L-1} + \mu[w_{ij}^{L}(t) - w_{ij}^{L}(t-1)] + \epsilon_{ij}^{L}(t)$$

La η da la velocidad de convergencia, si es muy grande, se puede dar la situación que se oscile demasiadas veces antes de llegar al mínimo. Por lo tanto η asume valores pequeños, por ejemplo, $0 \leq \eta \leq 0.3$. μ tiene valores más grandes $0.6 \leq \mu \leq 0.9$.

Red Neuronal Multicapa

Se observa que las salidas z_{ik}^{L-1} en la capa $L-1$ de la neurona i para la muestra de entrenamiento k son las entradas para la neurona j en la capa L. Por lo tanto para la primera capa se utilizaran las entradas x_{ik} de la red neuronal:

$$w_{ij}^{1}(t+1) = w_{ij}^{1}(t) + \eta \sum_{k=1}^{K} \delta_{jk}^{1} x_{ik} + \mu[w_{ij}^{1}(t) - w_{ij}^{1}(t-1)] + \epsilon_{ij}^{1}(t)$$

Implementación del algoritmo para encontrar los pesos

1. Se configura la red indicando el número de entradas N, el número de capas L, el número de neuronas por capa $N_i, 1 \leq i \leq L$ y el número de salidas N_s.
2. Se especifican las entradas y las salidas correspondientes:

$$\{(\underline{x}_k, \underline{d}_k);\ 1 \leq k \leq K\}$$

donde las $\underline{x}_k = [1 \quad x_{1k} \quad x_{2k} \ldots \quad x_{Nk}]$ son las entradas a cada una de las neuronas en la capa 1 y las $\underline{d}_k = [d_{1k} \quad d_{2k} \ldots \quad x_{N_s k}]$ son las salidas deseadas para esas entradas para la k muestra de entrenamiento. K es el número total de muestras de entrenamiento.

Implementación del algoritmo para encontrar los pesos

3. Mientras no se haya convergido para obtener los pesos óptimos:

3.1 Se inicia una nueva iteracción $t = t + 1$

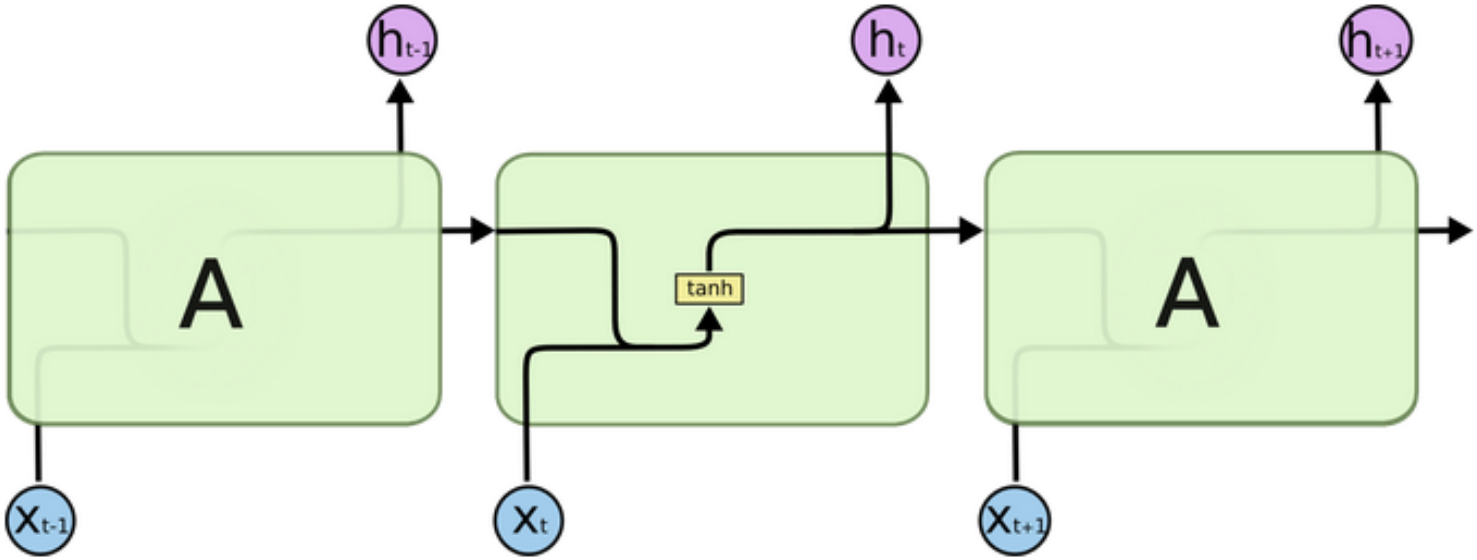
3.2 Se colocan las entradas $\underline{x}_k$ y se evalua la red comparando las salidas de ésta, $\underline{z}_k^L$, con las salidas deseadas $\underline{d}_k$ para $1 \leq k \leq K$

3.3 Se actualizan los pesos de cada capa con:

$$w_{ij}^L(t+1) = w_{ij}^L(t) + \eta \sum_{k=1}^{K} \delta_{jk}^L z_{ik}^{L-1} + \mu[w_{ij}^L(t) - w_{ij}^L(t-1)] + \epsilon_{ij}^L(t)$$

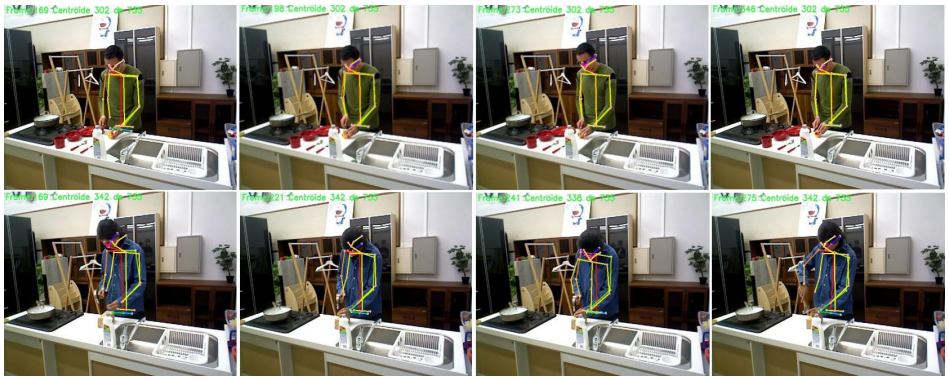
Redes Neuronales Artificiales Retroalimentadas

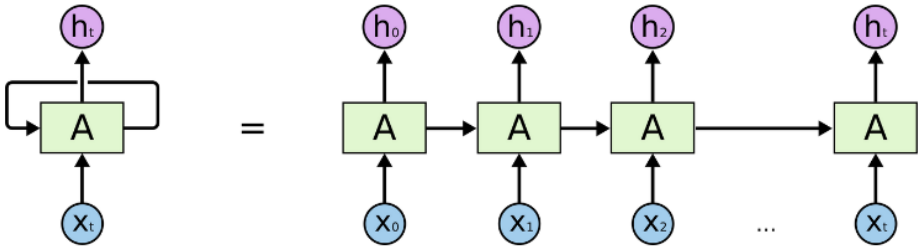
Para poder usar las redes neuronales con datos que van cambiando con el tiempo, se necesita tener redes neuronales las cuales contengan alambrada una memoria, que representa el estado h_t en el cual está el sistema, y ésto se logra retroalimentando las salidas de la red, parte de ellas o todas, como entradas. Así, se tiene una red neuronal artificial retroalimentada (RNN por sus siglas en inglés, Recurrent Neural Networks):



Redes Neuronales Retroalimentadas

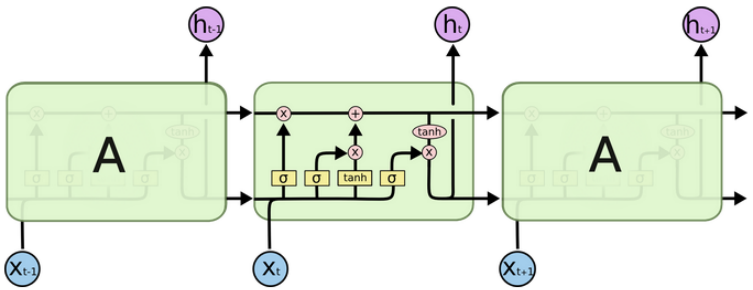
Las RNN son utilizadas para analizar secuencias de datos temporales en un tiempo dado, en áreas como por ejemplo el procesamiento del lenguaje natural, procesamiento de textos, y también particularmente se puede implementar para analizar secuencia de imgenes o video para reconocer acciones.





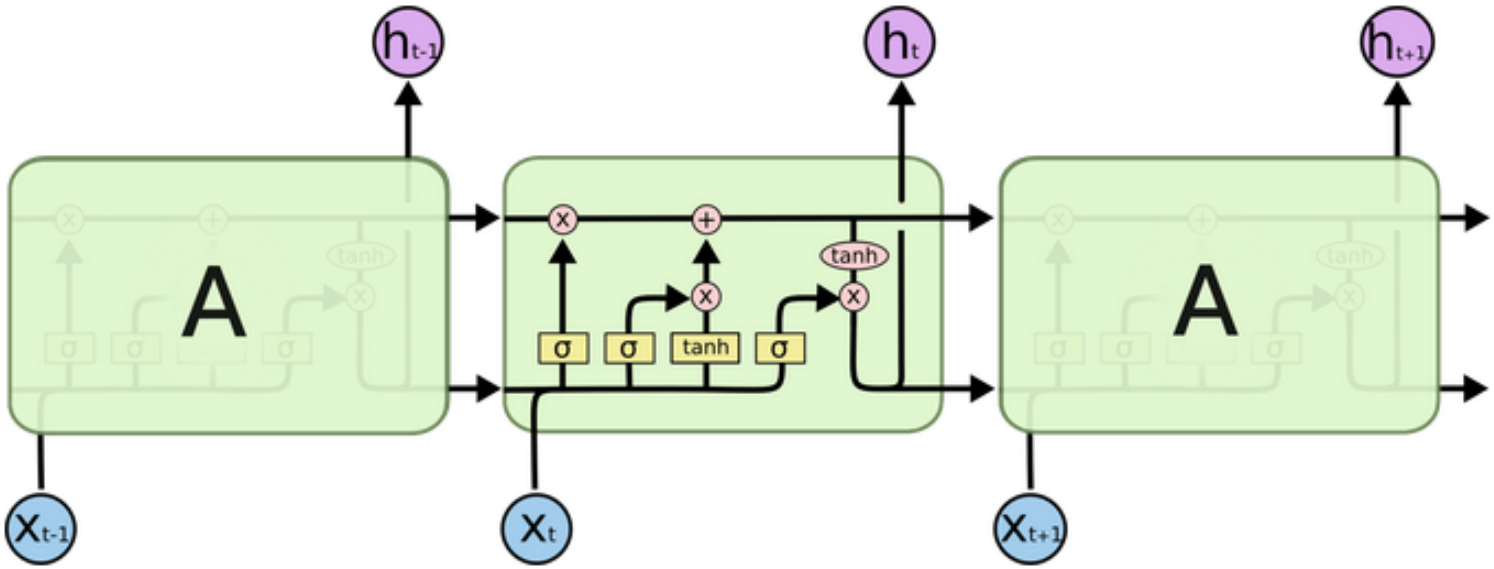
Redes Neuronales Retroalimentadas de Memoria de Corto y Largo Plazo

Una extensión de estas redes RNN son las redes LSTM (Long-Short Term Memory), introducidas en 1997 por Hochreiter y Schmidhuber, con el objetivo de solucionar algunas deficiencias de las redes RNN. Especficamente con la memoria a corto plazo, cuando las secuencias de datos son suficientemente largas, estas redes no lograban recordar información de iteraciones anteriores lejanas, o dicho de otro modo, las redes no lograban mantener información después de un largo periodo de tiempo. En parte ésto sucede porque cuando se ejecuta el algoritmo de retropropagación, en el cual el gradiente que sirve para actualizar los pesos se hace cercano a cero por errores de representación númerica.



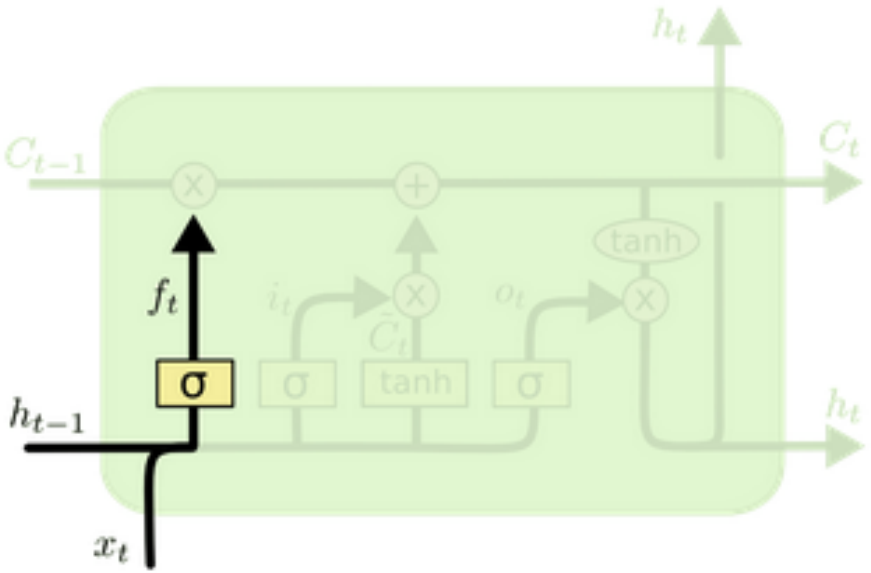
Redes Neuronales LSTM

Para poder tener una memoria de largo plazo se agregan las salidas nuevas c_t a la red neuronal, las cuales representan la memoria a largo plazo, junto con las salidas h_t, las cuales representan la memoria a corto plazo. La estructura interna consta en general de cuatro capas de redes neuronales (recuadros amarillos en la figura), teniendo dos funciones de activación, una sigmoide y una función tanh (tangente hiperblica).



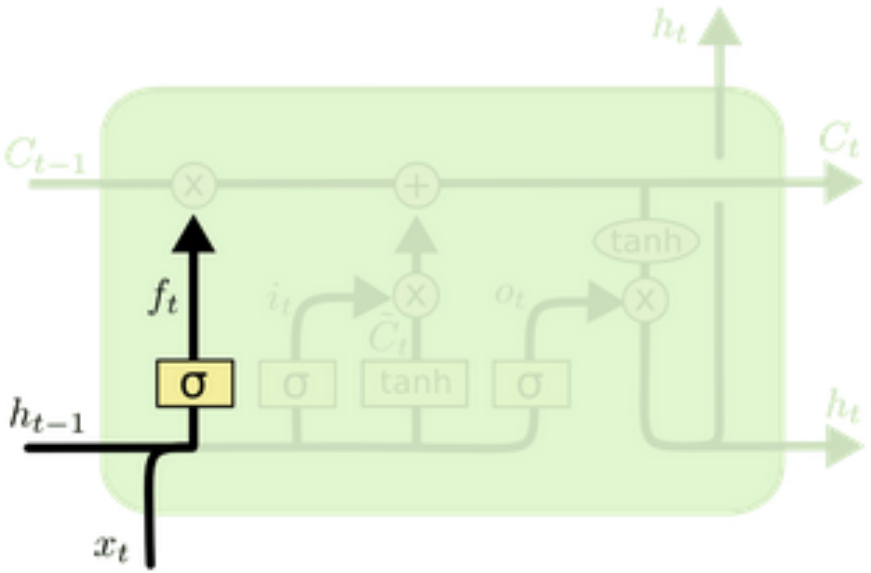
Redes Neuronales LSTM

El funcionamiento interno se puede ver en distintas etapas de la red, una primera es utilizando una capa neuronal σ llamada "compuerta del olvido", el cual tiene el propósito de procesar la información que se va a desechar o se va a guardar con la información de la iteración anterior h_{t-1} y los datos de entrada actual x_t.



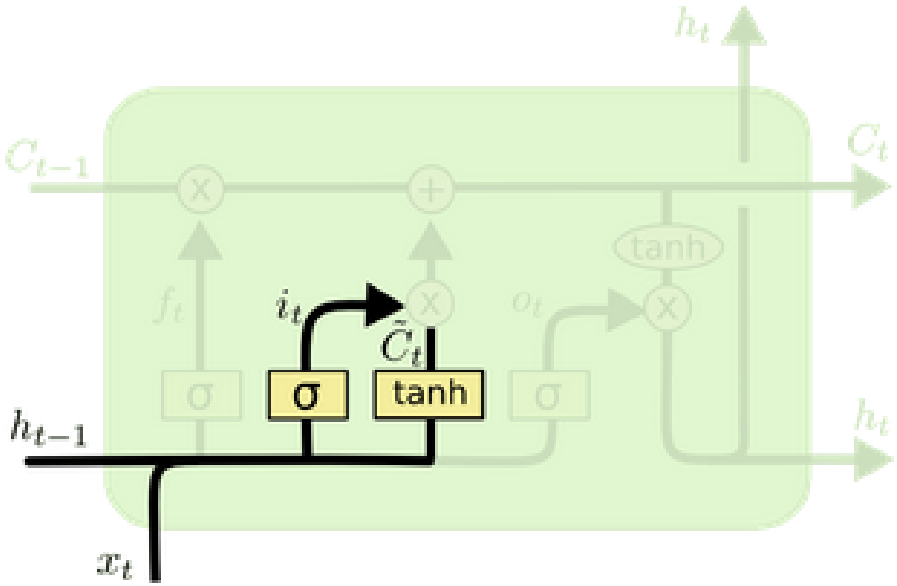
Redes Neuronales LSTM

Dado que la función sigmoide oscila entre 0 y 1, así se establece qué valores en el estado de la celda c_{t-1} deben descartarse (multiplicarse por 0), recordarse (multiplicarse por 1) o recordarse parcialmente (multiplicarse por algún valor entre 0 y 1).



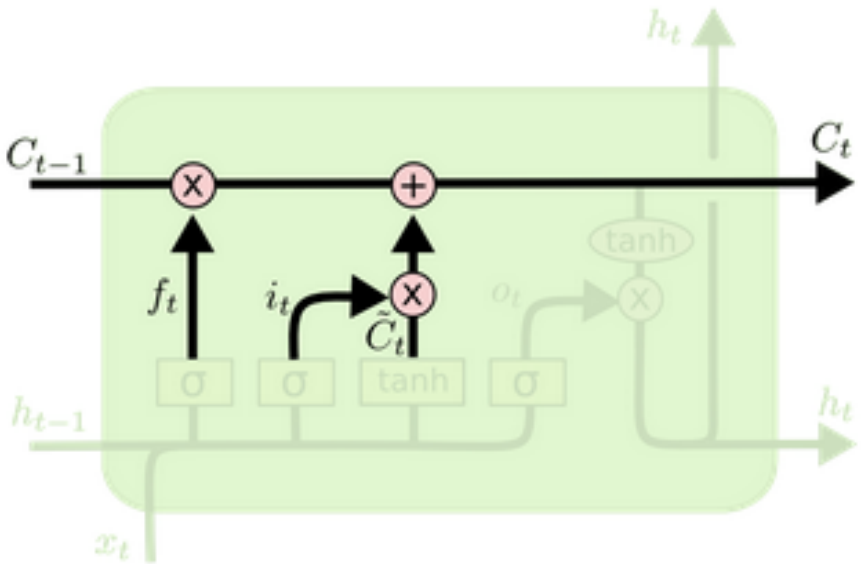
Redes Neuronales LSTM

La "compuerta de entrada" ayuda a identificar elementos importantes que deben agregarse al estado de la celda, tiene el proposito de decidir qué información nueva se guarda. Los resultados de la compuerta de entrada σ se multiplican por el estado de celda candidato $\hat{c}_t$, generada con una función de activación *tanh*, y solo la información que la compuerta de entrada considera importante se agrega al estado de celda.



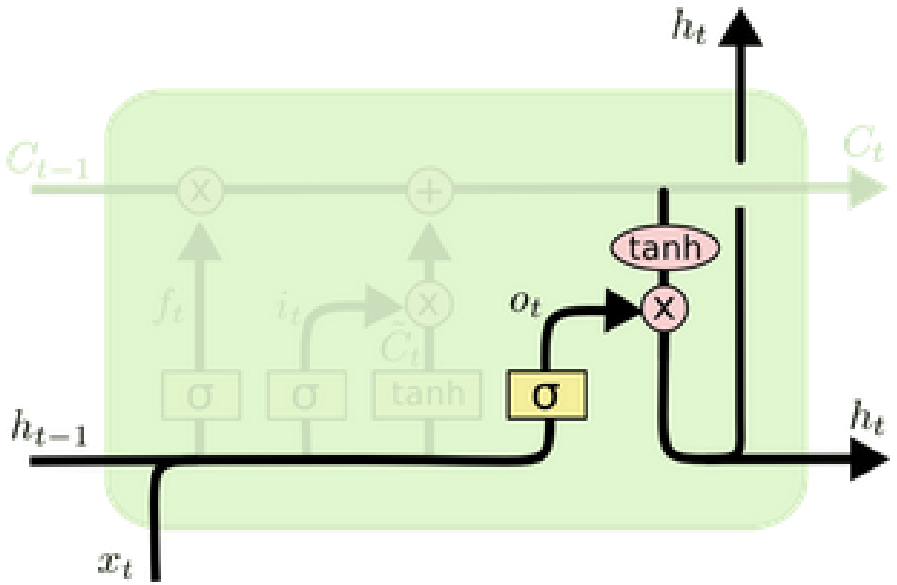
Redes Neuronales LSTM

Estos resultados son multiplicados y sumados a la información que resulta de la etapa del olvido multiplicada por la información c_{t-1} de la iteración anterior.

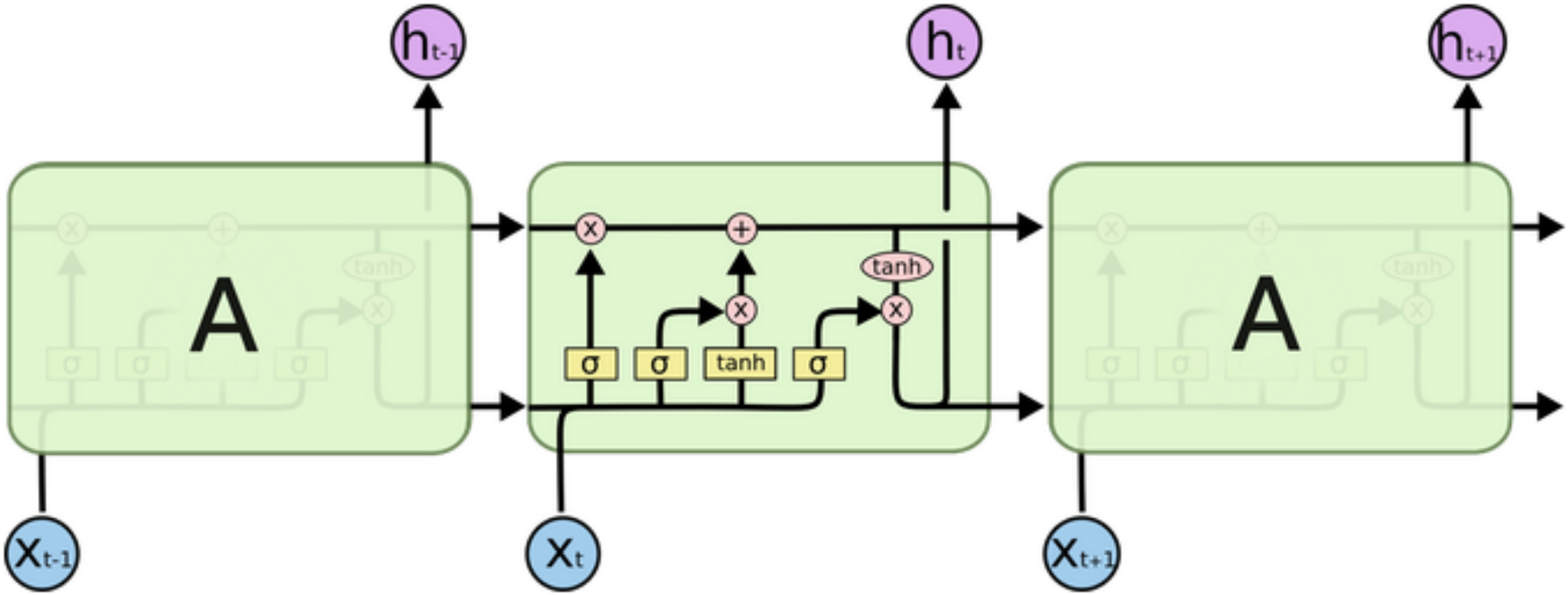


Redes Neuronales LSTM

Por último una etapa que actualiza el estado oculto h_t que decide que información saldrá mediante una "compuerta de salida" σ, que decide que datos de la neurona sera producida a la salida. Este resultado será multiplicado con la nueva información c_t a través de una función de activación tanh que pone la información en el rango de $(-1, 1)$. El resultado será la salida h_t que también será procesada en la siguiente iteración.

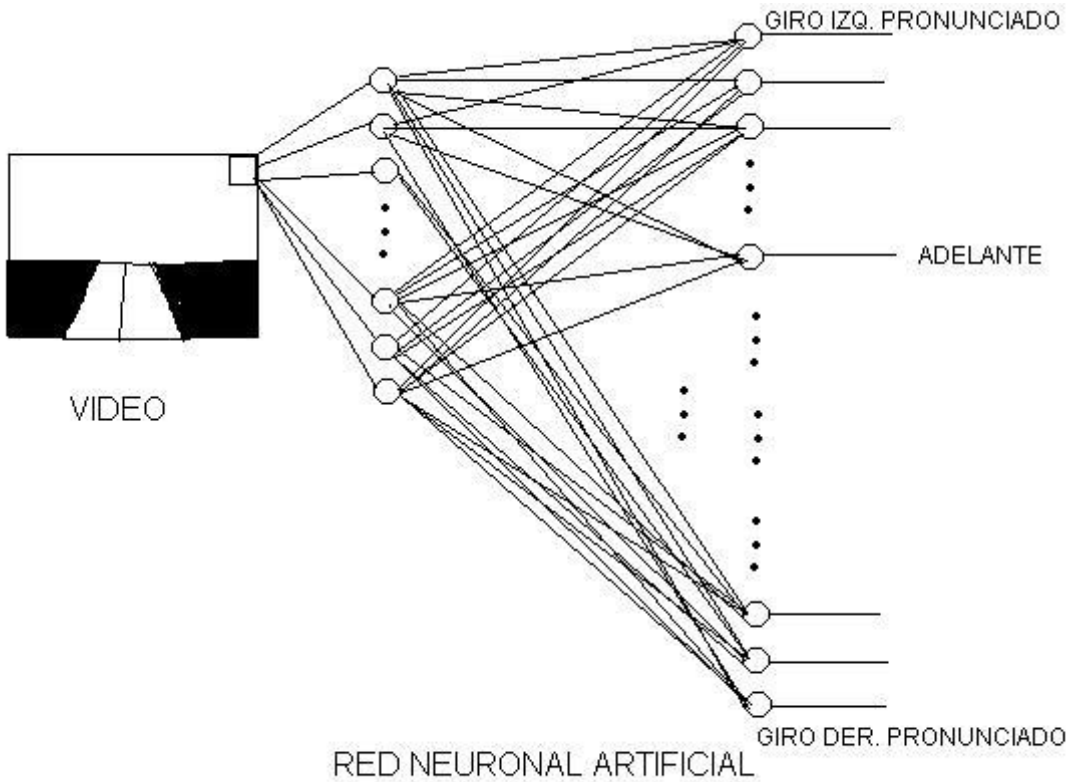


Redes Neuronales LSTM



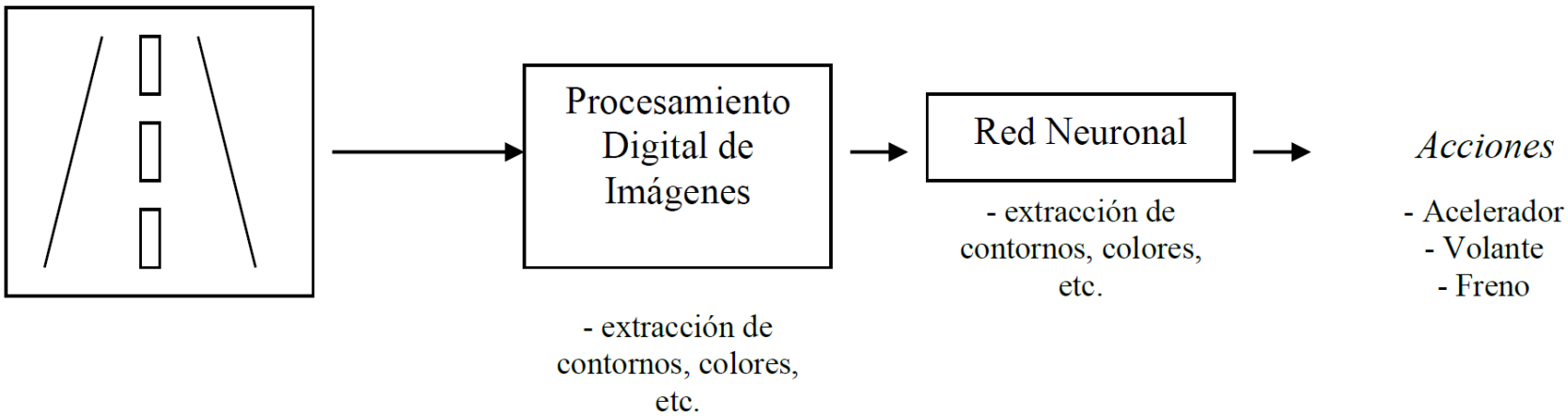
Comportamientos de Robots sin memoria

Se pueden usar las redes neuronales para tener comportamientos en los robots móviles. En Carnegie-Mellon University, se realizó un experimento en el cual un robot pudo manejar 90 % autónomamente en una autopista de Estados Unidos.



Comportamientos de Robots Usando Redes Neuronales Artificiales

La entrada a la red neuronal es una imagen de la cámara del robot. Se puede entregar la información en bruto o procesada usando técnicas de procesamiento digital de señales. Se entrena a la red para que siga los comportamientos que un conductor haría bajo ciertas circunstancias. La respuesta de la red es: para una cierta imagen, se giró a la izquierda violentamente; para otra imagen, se giró ligeramente; etcétera.



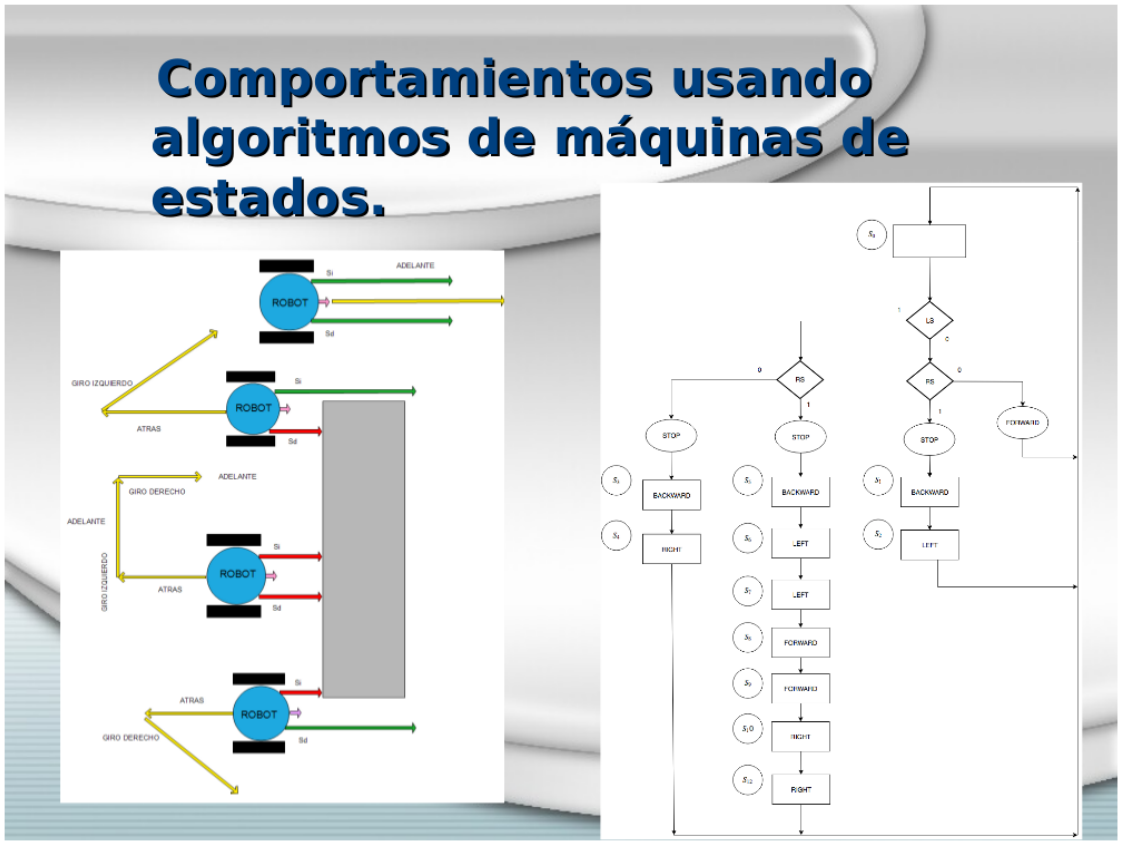
Estos comportamientos no tienen memoria.

Máquina de Estados Finitas

Una máquina de estados finita se define como:
$M = (Q, X, Y, \delta, \lambda)$ donde:
Q: es un conjunto finito de estados.
X: es un conjunto finito de símbolos de entrada.
Y: es un conjunto finito de símbolos de salida.
δ: $QxX \rightarrow Q$, es una función que genera el estado siguiente.
λ: $QxX \rightarrow Y$, es una función que genera las salidas.

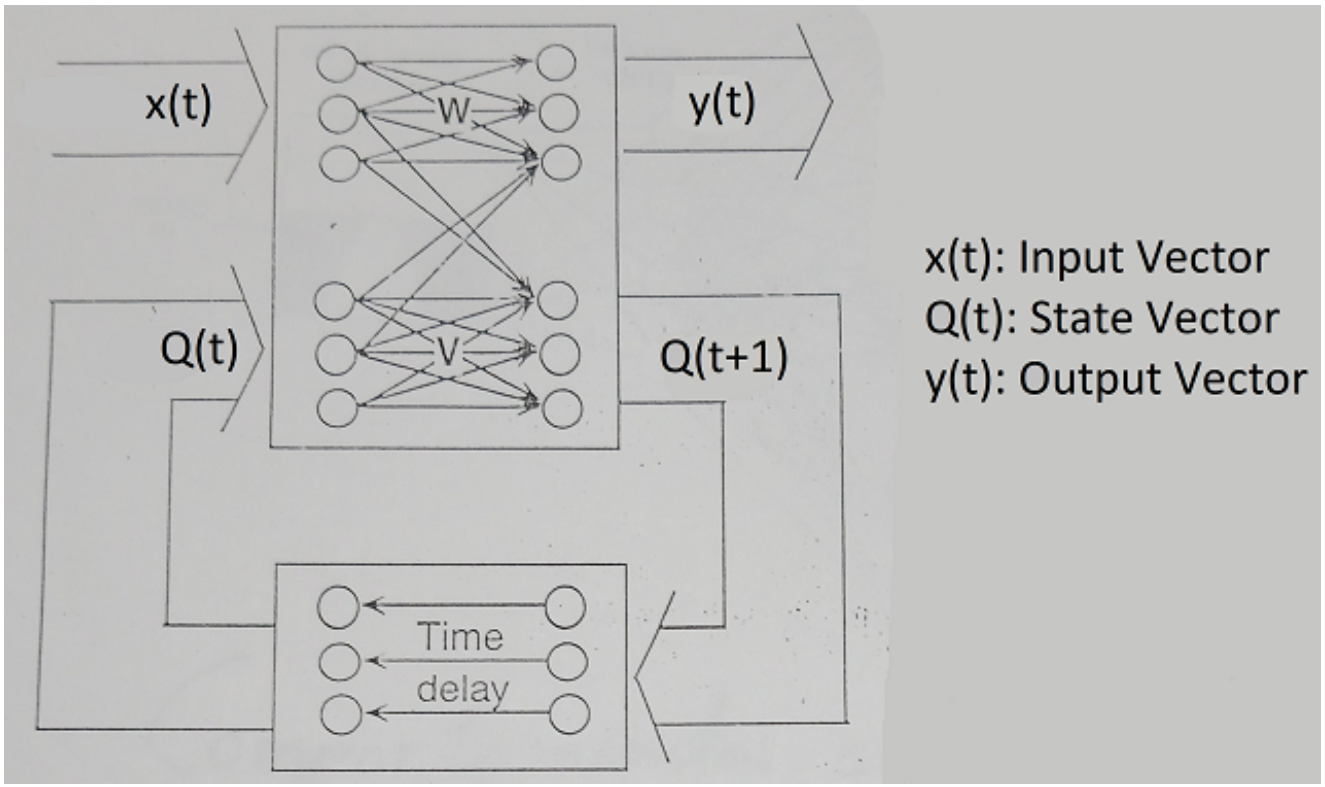
Comportamientos de Robots Usando Redes Neuronales Artificiales

Con una máquina de estados se pueden ejecutar algoritmos como el que se muestra en la siguiente figura:



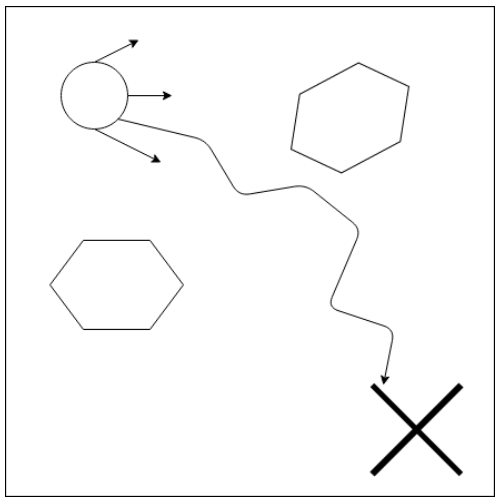
Comportamientos de Robots Usando Redes Neuronales Artificiales

Para poder tener comportamientos usando redes neuronales similar a las máquinas de estado se necesita tener memoria y ésto se logra retroalimentando algunas de las salidas de la red como entradas. Así, se tiene una red neuronal artificial retroalimentada:



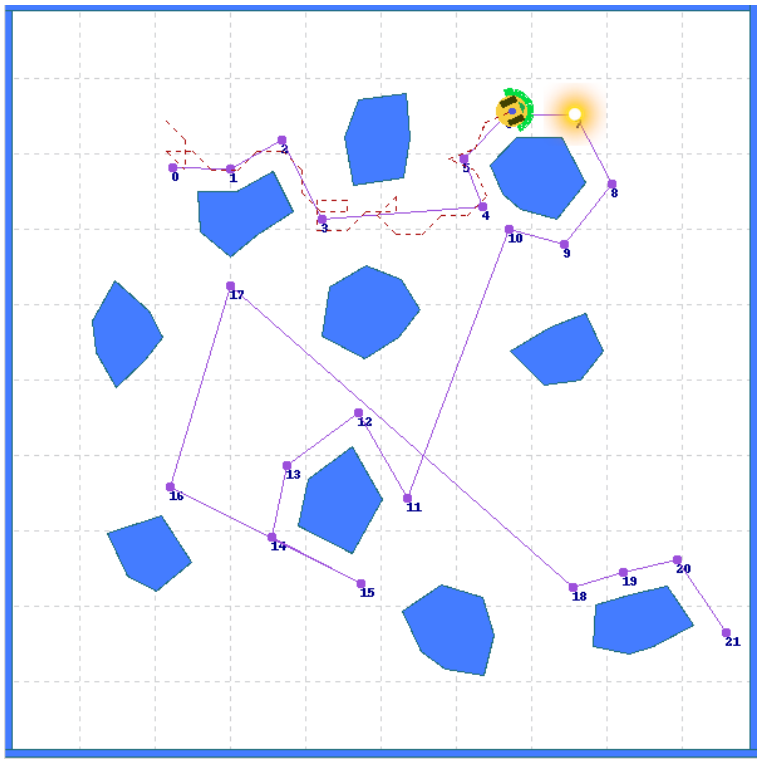
Entrenamiento de Comportamientos usando el concepto de Máquina de Estados

Para un entrenamiento supervisado se tomarían las muestras de un operador moviendo el robot de un origen a un destino.
Si se usa un Joystick se guardan las acciones del operador como salidas deseadas de la red neuronal y se obtienen las lecturas de los sensores como entradas.
Muestras de
entrenamiento $\{X_k, d_k\}$
x_k: Sensores
d_k: Joystick:
adelante,
atrás, GI, GD, alto



Entrenamiento de Comportamientos usando el concepto de Máquina de Estados

¿Cómo se indica el estado? Se podría usar como maestro a un robot que tenga una máquina de estados para evadir obstáculos y se guardan las entradas, estados y salidas.



Entrenamiento no Supervisado

Se indica la posición inicial y final del robot y se tiene que entrenar el algoritmo por si mismo. Si el algoritmo esta hecho con NN se utilizan algoritmos genéticos para encontrar los pesos w_{ij}^{L} para poder resolver el problema.