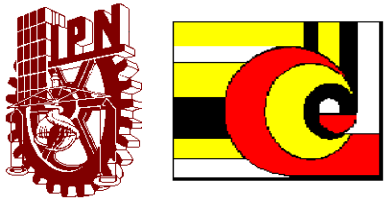
Introducción a las Redes Neuronales Artificiales y RNAs Avanzadas



Juan Humberto SOSSA AZUELA

E-mail: hsossa@cic.ipn.mx and humbertososa@gmail.com

http://sites.google.com/site/cicvision/

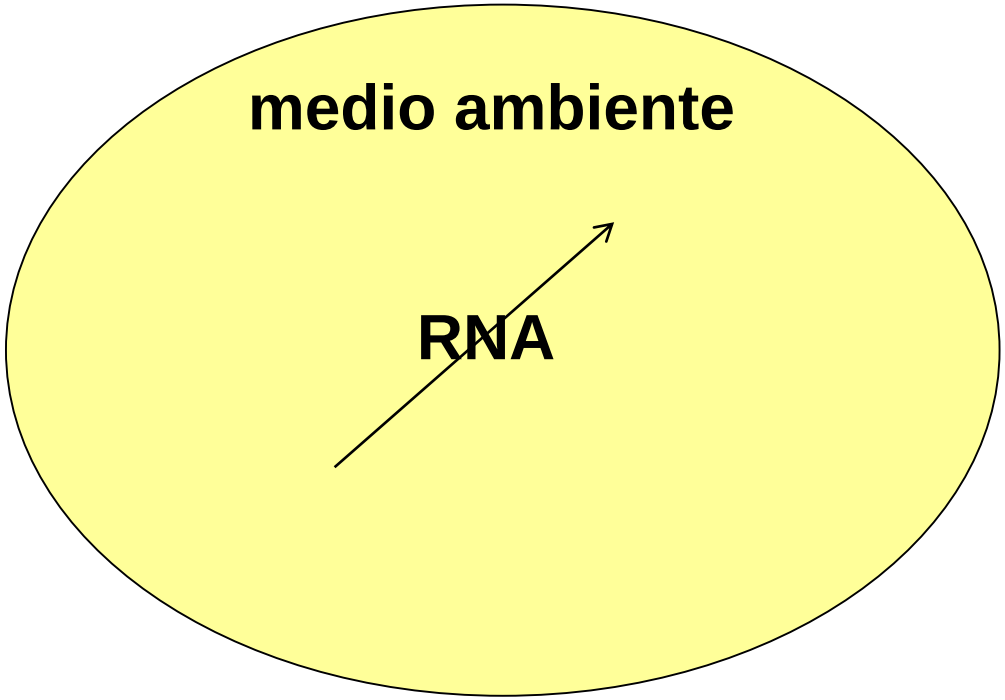
Entrenamiento de RNAs:

Una propiedad muy importante de una red neuronal artificial (RNA) es su capacidad de **aprender** del medio ambiente.

La RNA aprende de su medio ambiente a través de un proceso iterativo de ajuste de sus pesos y biases.

Este proceso de aprendizaje, en el contexto de las RNA, puede ser descrito como sigue:

Aprendizaje es el proceso mediante el cual los parámetros libres (los pesos, arquitectura,…) de una RNA son adaptados a través una estimulación del medio ambiente donde se encuentra la RNA.



NOTA: Este proceso de ajuste no tiene paralelo en las conexiones sinápticas de las neuronas reales.

El tipo de aprendizaje está determinado por la manera en que los parámetros son ajustados. Esta definición implica los siguientes tres eventos:

1 La RNA es estimulada por un medio ambiente.

El tipo de aprendizaje está determinado por la manera en que los parámetros son ajustados. Esta definición implica los siguientes tres eventos:

1 La RNA es estimulada por un medio ambiente.

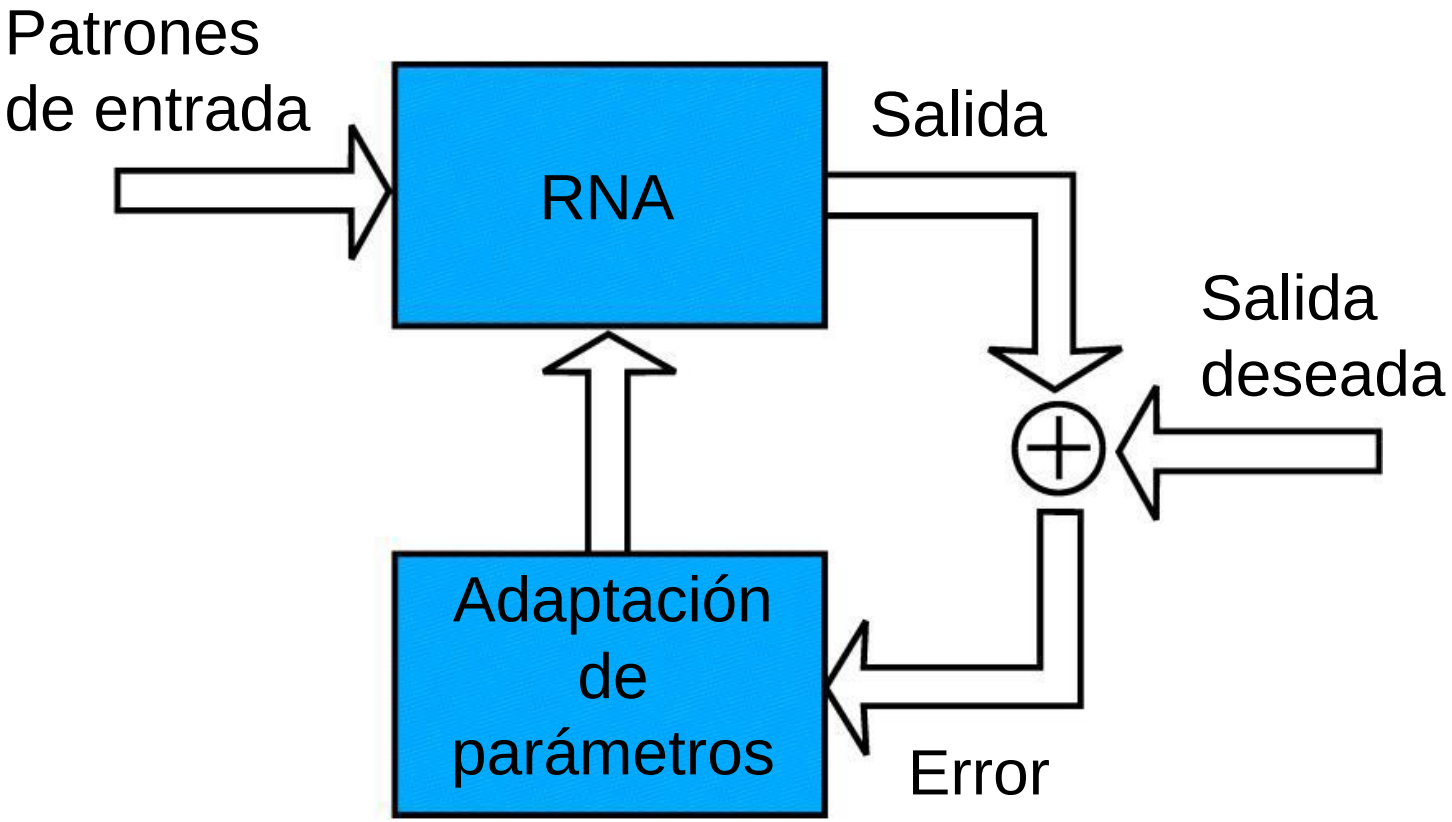
2 Los parámetros libres de la RNA experimentan cambios por esta estimulación, y

El tipo de aprendizaje está determinado por la manera en que los parámetros son ajustados. Esta definición implica los siguientes tres eventos:

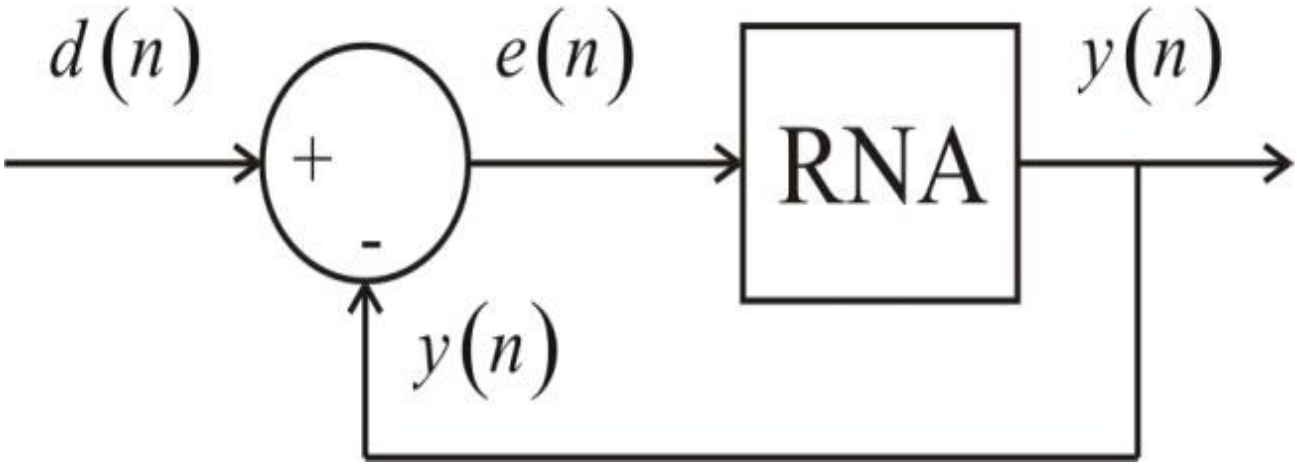
1 La RNA es estimulada por un medio ambiente.

2 Los parámetros libres de la RNA experimentan cambios por esta estimulación, y

3 La RNA responde de una nueva manera al ambiente por los cambios en su estructura interna.


Patrones
de entrada
RNA
Salida
Salida
deseada
Adaptación
de
parámetros
Error

Sin pérdida de generalidad consideremos el caso de una neurona aislada. La neurona es alimentada con un vector $x(n)$, donde n denota un tiempo discreto, su salida es $y(n)$. Esta salida $y(n)$ es comparada con la respuesta deseada de salida $d(n)$. El error $e(n) = d(n) - y(n)$ es la señal de control que es usada para el ajuste iterativo de los parámetros libres de la RNA.



Se utilizará este principio para derivar varias reglas para el ajuste de pesos.

En particular se verá el caso de las reglas:

La reglas **perceptrón**, y

La **Regla Delta**.

Un algoritmo de aprendizaje es un **método adaptativo** que permite encontrar en **forma automática** los pesos de los elementos de cómputo de una RNA.

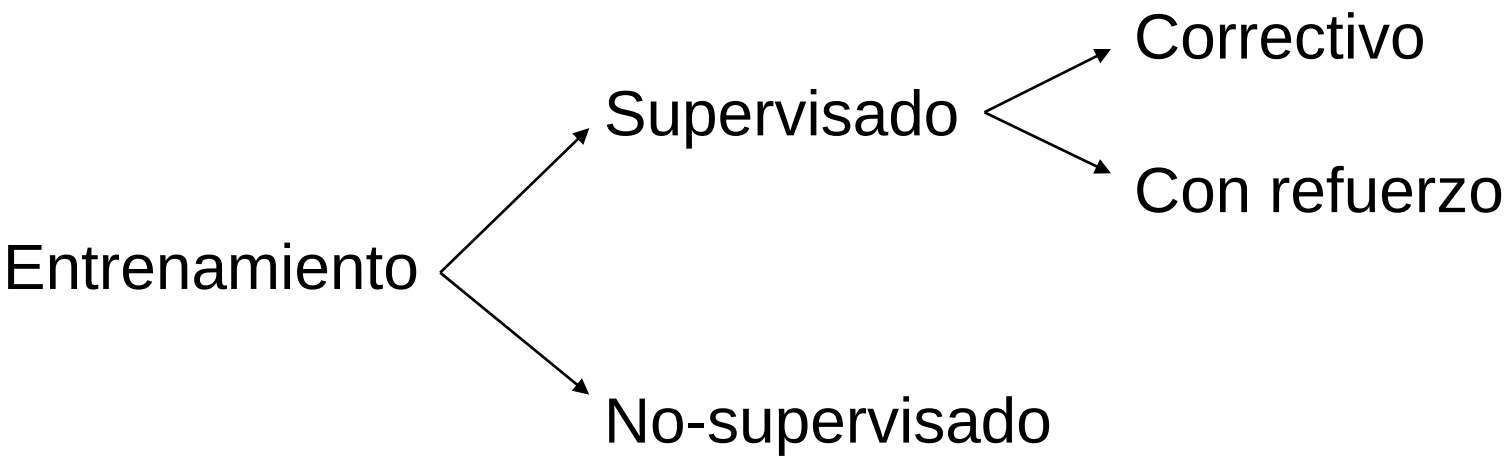
Este método debe ser auto-organizado, en el sentido de que al emplear patrones de entrada con sus salidas deseadas, debe poder aplicar un **paso de corrección** de los parámetros libres de la RNA.

Este paso es ejecutado en **forma iterada** hasta que la RNA aprende a producir la salida deseada. Los parámetros son **adaptados** sobre la base de la experiencia previa hasta converger a una solución (si esta existe).

El algoritmo de aprendizaje opera, como ya vimos, en lazo cerrado.

Tipos de algoritmos de entrenamiento:

Se pueden dividir en **supervisados** y **no-supervisados**:



El algoritmo de entrenamiento para el perceptrón es un ejemplo de un algoritmo supervisado con refuerzo.

Algoritmos de entrenamiento supervisados (continuación):

Aprendizaje por refuerzo: En este caso solo se conoce si la RNA produce el resultado deseado o no. Dependiendo de esta información, solo el vector de entrada se aplica a la corrección del peso.

Aprendizaje correctivo: En este caso **la magnitud del error junto con el vector de entrada** son usados para corregir los pesos.

Regla formal de ajuste: (si la salida actual ≠ salida deseada):

$\theta_{nuevo} = \theta_{viejo} + \Delta\theta$ Actualizar umbrales

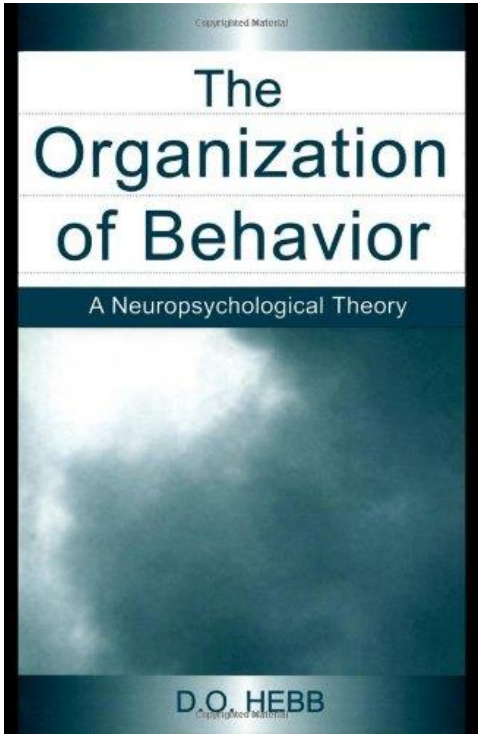
$w_{nuevo,i} = w_{viejo,i} + \Delta w_i$, con $i \in \{1, \ldots, N\}$ y pesos.

Donald Hebb y el aprendizaje artificial:

Hebb y la biopsicología:

La biopsicología tiene que ver con el estudio de la biología del comportamiento.

La obra: La Organización del Comportamiento en 1949 desempeñó un papel clave en su aparición.



Hebb y la biopsicología:

En este libro Hebb desarrolló la primera teoría comprensible sobre el modo en que los fenómenos psicológicos como las percepciones, las emociones, los pensamientos y la memoria, pueden ser producidos por la actividad cerebral.

Hebb basó su teoría en experimentos, tanto con seres humanos como con animales de laboratorio, en estudios clínicos y en argumentos lógicos desarrollados a partir de sus propias observaciones de la vida.

Principio básico:

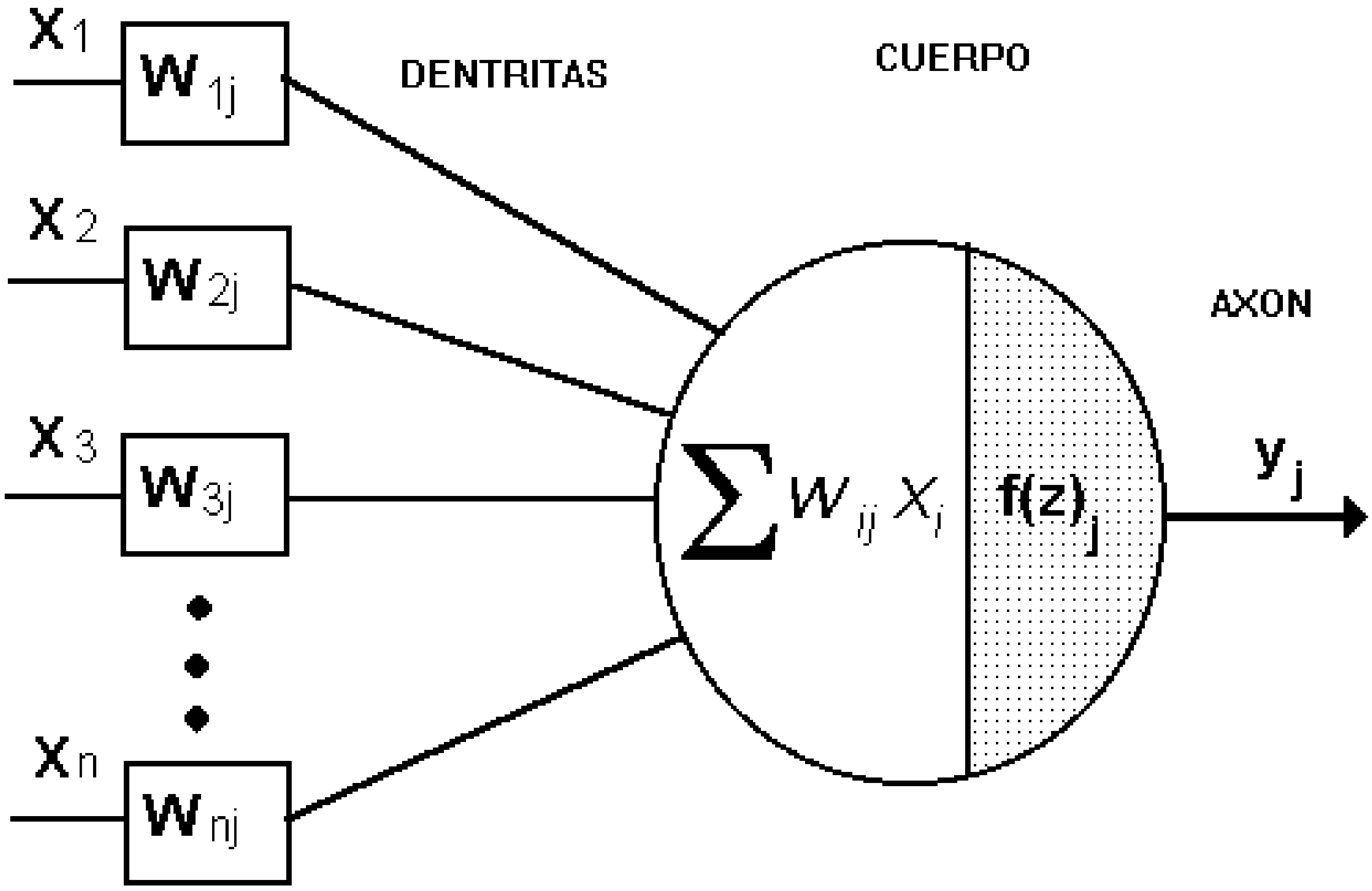
Su principal aporte se relaciona con la formación de **ensambles neuronales** regidos por el siguiente principio:

"Cuando un axón de una célula A **está lo suficientemente cerca** de una célula B, como para excitarla, y participa repetida o persistentemente en su disparo, ocurre algún proceso de crecimiento o cambio metabólico, en una o en ambas células, de modo tal que la eficacia de A, como una de las células que hacen disparar a B, aumenta".

Este principio (o Ley de Hebb), en ciencia cognitiva, se denomina la "**Regla de Hebb**" y provee el algoritmo básico de aprendizaje mediante redes neuronales artificiales.

Sobre la base de la regla de Hebb se presentarán algunas técnicas para el entrenamiento de redes neuronales.

Entrenamiento de TLU´s:



Axones Sinápsis

El producto punto (se vale del operador suma que toma como argumentos productos de dos cantidades, digamos: w_i y x_i, $w_i \times x_i$,**):**

$$a = \sum_{i=1}^{n} w_i \times x_i \,.$$

Los w_i y los x_i son, respectivamente, los componentes de los vectores $\mathbf{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}$ y $\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$.

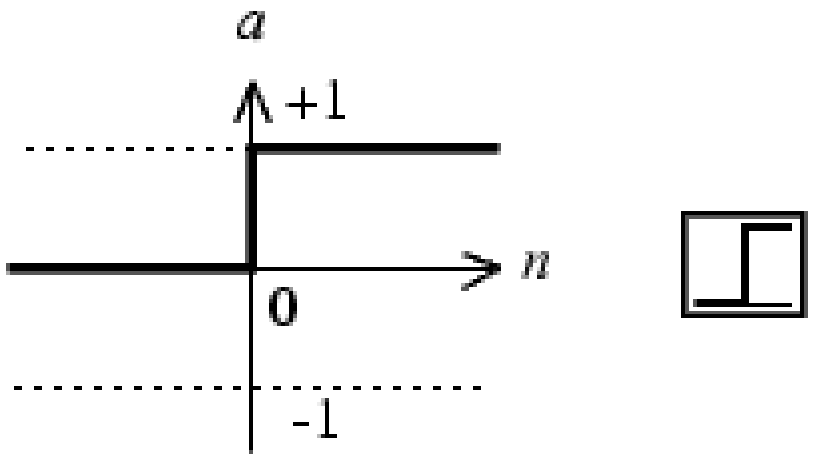
$a = 0$
for $i = 1$ to n
    $a = a + w_i \times x_i$
next i

Función de activación: Es la operación que lleva a cabo el mapeo del vector al conjunto (RNAs de primera y segunda generación):

Objetivo: Introducir una no-linealidad a la RNA.

NOTA: Sin esta no-linealidad la RNA **no podrá aprender funciones no lineales**.

Caso de la TLU:



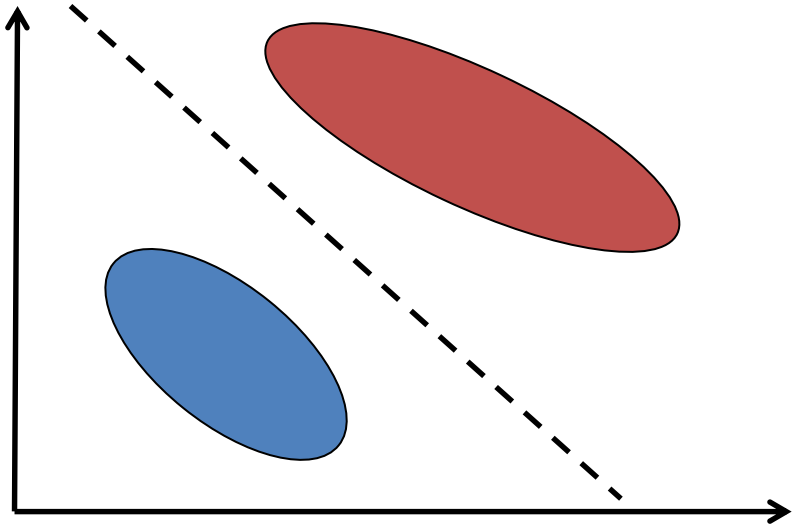
Hard-Limit Transfer Function

Entrenamiento de redes con TLUS:

Se introduce el concepto de entrenamiento de una red para desarrollar una tarea dada.

Se verá para el caso de TLUs.

Una TLU requiere de una superficie de separación.



Como ésta depende del vector de pesos y del umbral, es **necesario ajustarlos** para encontrar dicha superficie.

Este proceso de ajuste se hace mediante un proceso iterativo de entrenamiento a partir de un conjunto de patrones.

Ajuste (entrenamiento) del umbral como un peso:

A fin de poder entrenar el umbral como un peso se hace un pequeño truco matemático.

Ya vimos que: $\mathbf{w} \cdot \mathbf{x} \geq \theta$

Para la condición de un “1”.

Al restar θ a ambos lados y al hacer el signo menos explícito:

$$\mathbf{w} \cdot \mathbf{x} + (-1)\theta \geq 0$$

Se puede pensar del umbral como un **peso extra *colgado*** a un valor de -1.

A este umbral negativo se le llama *desplazamiento* o *bias*.

Al conjunto de patrones que se usan para entrenar la red se le llama de **conjunto de entrenamiento**.

Los ajustes en los pesos y el umbral se dan mediante una **regla de aprendizaje**.

Sea el conjunto de entrenamiento:

$$(\{x_i\},\{t_i\}), i = 1,\cdots,n \quad \{t_i\} = \{0,1\}$$

Ejemplo: (0,0)->0 (0,1)->0 (1,0)->0
(1,1)->1.

Se verá el modo de entrenamiento supervisado.

El vector de pesos aumentado queda como:

$$\{w_1, w_2, \cdots, w_n, w_{n+1} = \theta\}$$

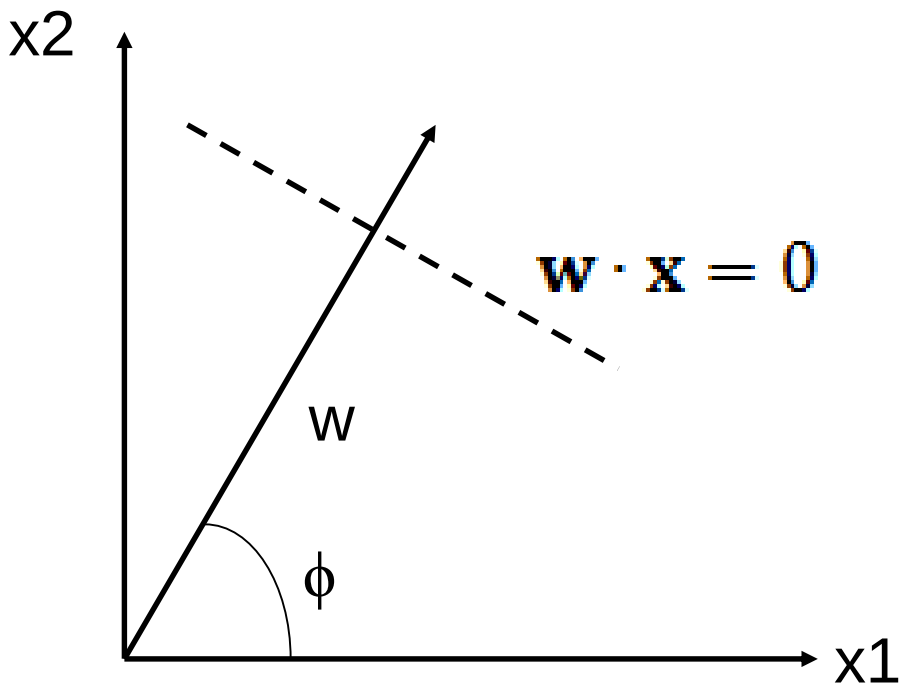
La función del nodo quedaría como:

$$\mathbf{w} \cdot \mathbf{x} \geq 0 \rightarrow y = 1 \qquad \mathbf{w} \cdot \mathbf{x} < 0 \rightarrow y = 0$$

Ahora bien:

$$\mathbf{w} \cdot \mathbf{x} = 0$$

Define el **híper-plano** de separación que es perpendicular al vector de pesos, que pasa por el origen.

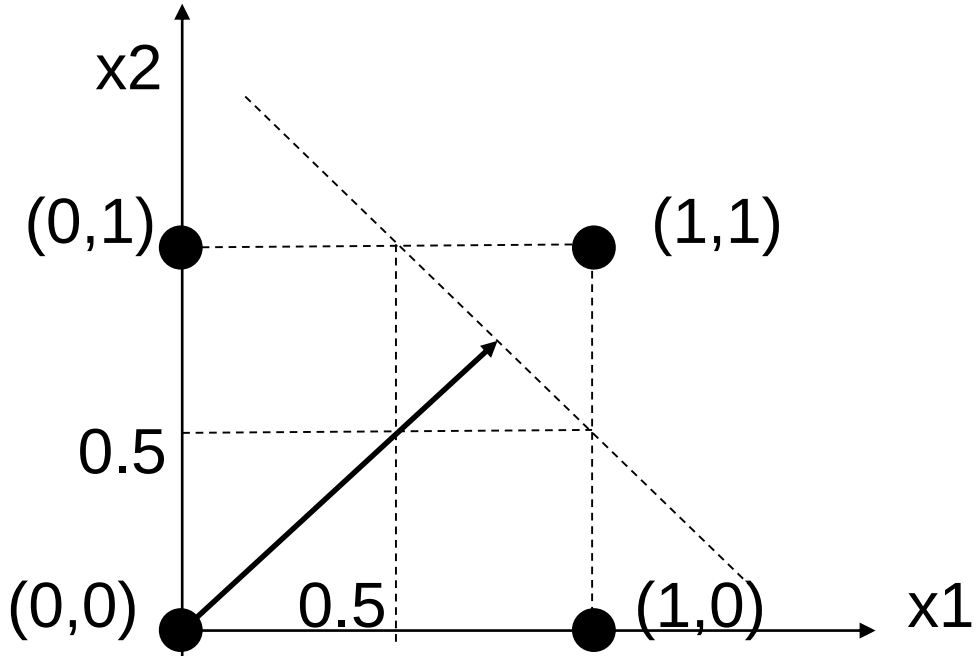


NOTA: El híper-plano de separación siempre es perpendicular al vector de pesos.

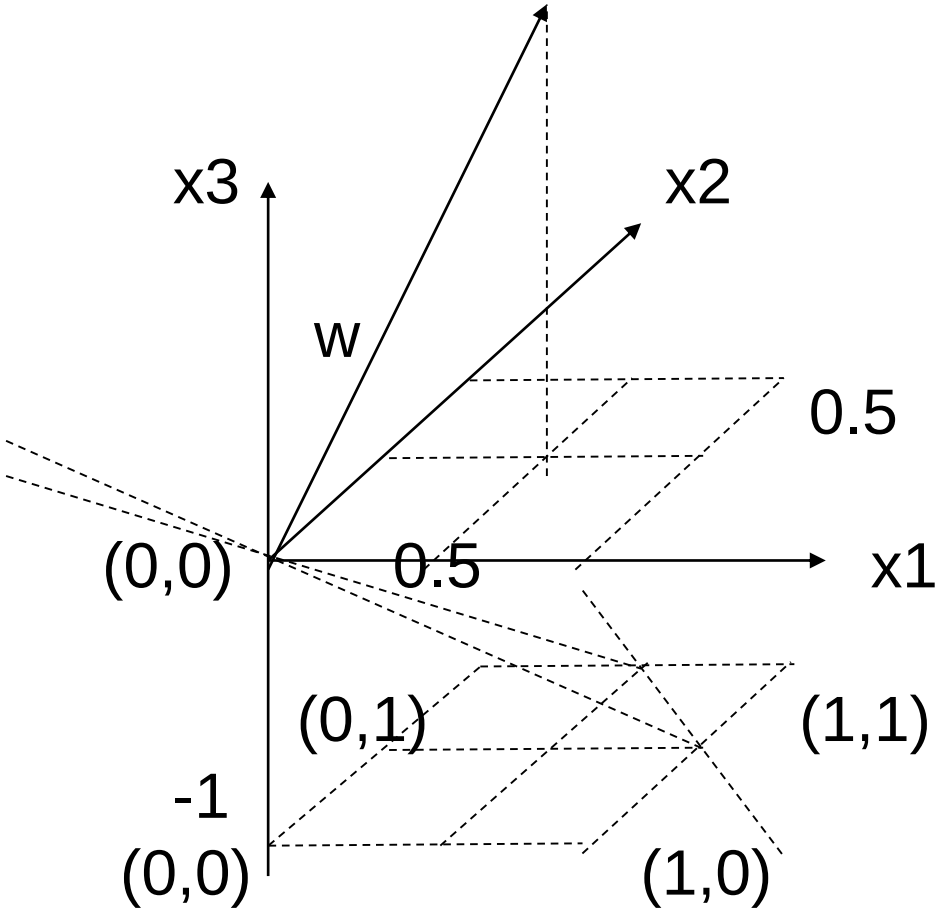
Ejemplo. Sea una TLU que emite un “1” para la entrada (1,1) y “0” para las otra tres entradas, mediante la selección:

Vector de pesos = (1/2, 1/2) y umbral = ¾.

Esta línea pasa por los puntos: x1=(0.5,1) y x2=(1,0.5).



Para el patrón aumentado:



Como se puede ver todos los patrones tienen la forma: (x1, x2, -1), ya que la tercera componente está atada al valor constante -1.

Note que vector aumentado tiene una tercera componente = al umbral y es perpendicular al plano que pasa por el origen.

La regla del perceptrón para la TLU:

Frank Rosenblatt (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386–408.

Frank Rosenblatt (1962), Principles of Neurodynamics. Washington, DC:Spartan Books.

Ajuste del vector de pesos (Regla del perceptrón):

Suponemos ahora que queremos entrenar una TLU mediante un conjunto de entrenamiento:

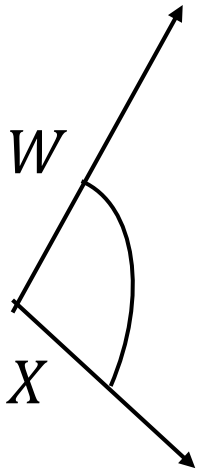
$$(X_k, t_k), k = 1, \dots, N, X_k = (x_1, x_2, \dots, x_{n+1})^T$$

Caso 1. Se presenta a la entrada de la TLU un vector X con respuesta deseada $t = 1$, y que el vector actual de pesos da $y = 0$.

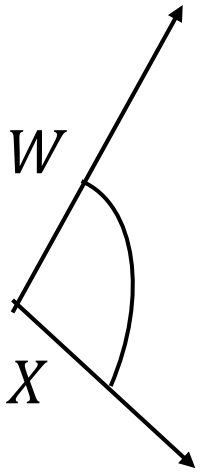
Debido a que $y \neq t$, se dice que la TLU ha mal clasificado.

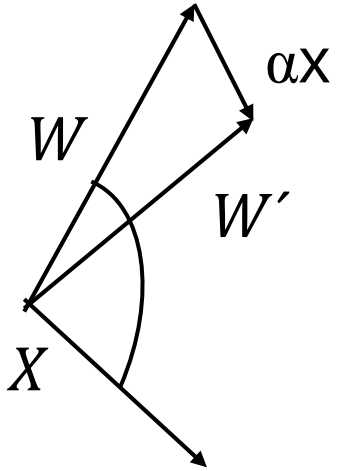
Los pesos deben ser ajustados.

Para producir un $y = 0$, la activación debió haber sido negativa. Luego $\mathbf{w} \cdot \mathbf{x}$ fue negativo, $\mathbf{w}$ y $\mathbf{x}$ apuntan en sentidos opuestos.



Para producir un $y = 0$, la activación debió haber sido negativa. Luego $\mathbf{w} \cdot \mathbf{x}$ fue negativo, $\mathbf{w}$ y $\mathbf{x}$ apuntan en sentidos opuestos.





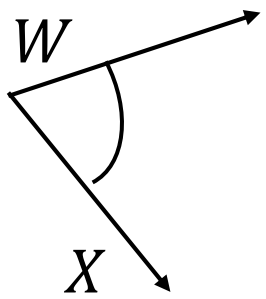
Esto provoca una rotación también del plano de decisión.

A fin de corregir esto es necesario rotar W un poco hacia X. Esto se logra al sumar una fracción de X a W como:

$$W' = W + \alpha X$$

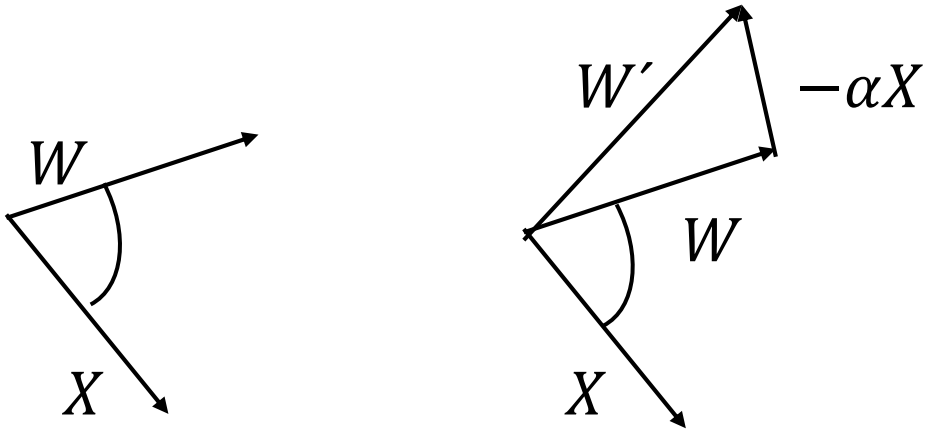
Con $0 < \alpha < 1$.

Caso 2. deseada $t = 0$, y que $y = 1$.



Esto significa que la activación fue positiva cuando debió haber sido negativa.

Caso 2. deseada $t = 0$, y que $y = 1$.



Esto significa que la activación fue positiva cuando debió haber sido negativa.

Se requiere ahora rotar W lejos de X, lo que se puede hacer al sustraer una fracción de X de W:

$$W' = W - \alpha X$$

Esto provoca una rotación también del plano de decisión.

Ambas expresiones:

$$W' = W + \alpha X \; W' = W - \alpha X$$

se pueden combinar en una sola como:

$$W' = W + \alpha(t - y)X$$

Ambas expresiones:

$$W' = W + \alpha X \; W' = W - \alpha X$$

se pueden combinar en una sola como:

$$W' = W + \alpha(t - y)X$$

El cambio en cada componente viene dado como:

$$\Delta w_i = \alpha(t - y)x_i, i = 1, \ldots, n + 1$$

Con: $w_{n+1} = \theta$ $x_{n+1} = -1$

α es la tasa de entrenamiento.

Regla de entrenamiento (regla de entrenamiento del perceptrón):

Poner los valores del vector de pesos W en valores aleatorios, por ejemplo, en el rango $\{-1, +1\}$

Inicializar α en algún valor entre $0 < \alpha < 1$

Repetir

Para cada vector de entrenamiento (X, t)

Evaluar la salida y cuando X es presentada a la TLU

Si $y \neq t$ entonces

Actualizar el vector de pesos de acuerdo a:

$$W' = W + \alpha(t - y)X$$

Sino

No hacer nada

Fin del si

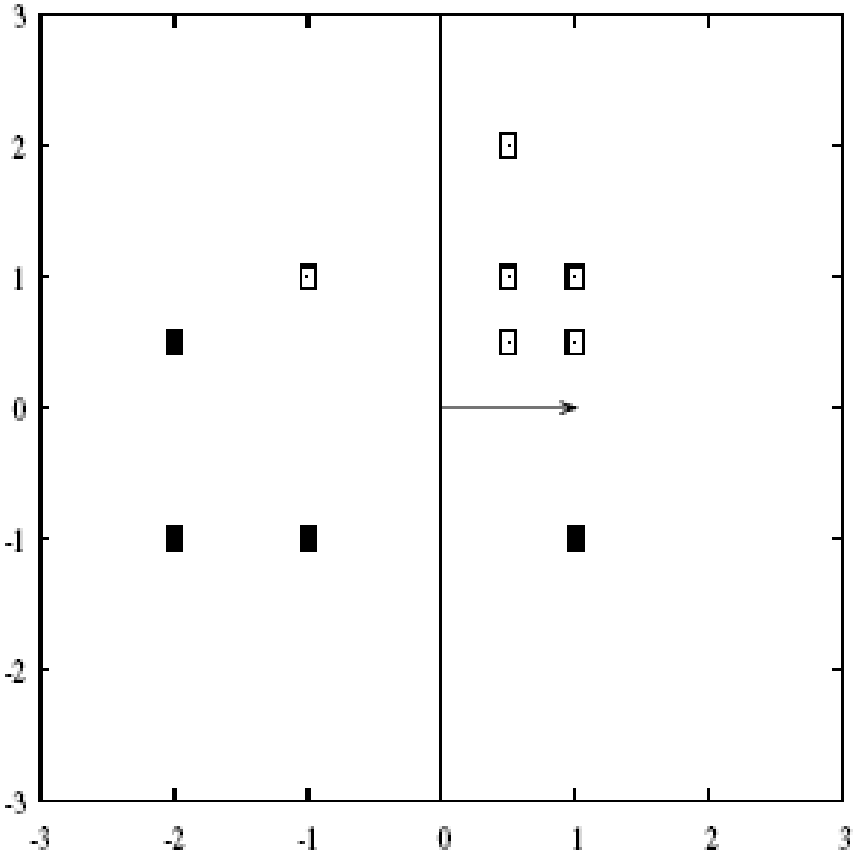
Fin del para

Hasta que $y = t$ para todo par de entrenamiento (X, t).

Este algoritmo generará un vector de pesos válido para el problema en cuestión, si dicho vector existe.

.

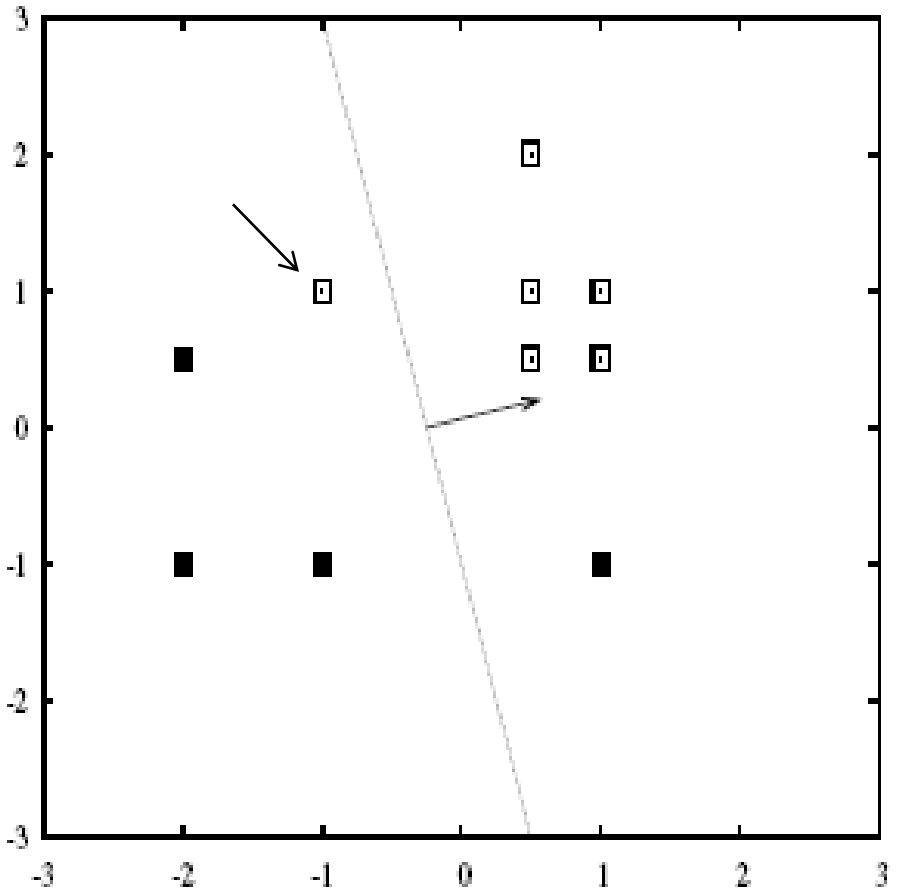
Ejemplo: Dados los patrones mostrados y $w = (1\ 0);\ b = 0$
Situación inicial:



$$w_i' = w_i + \alpha(y - t)x_i$$

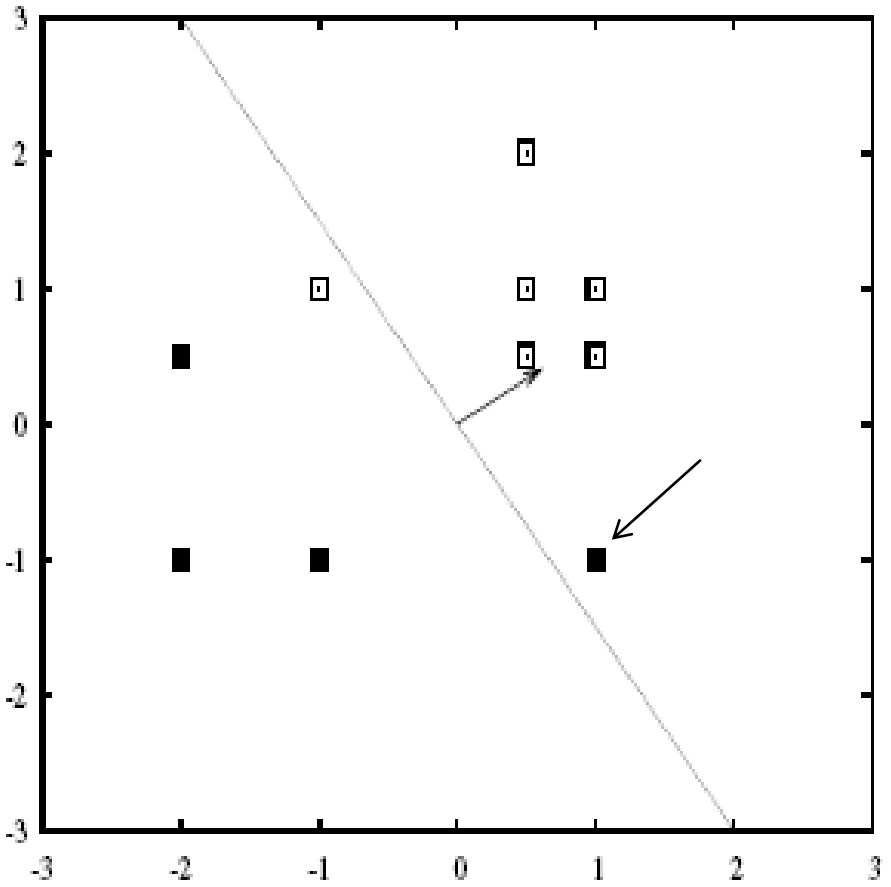
Ajustar w y b para: $\alpha = 0.2$ para $x = (-1\ 1)$:

$$\mathbf{w}_1 = \mathbf{w}_1 - 0{,}2 * (0-1) * \mathbf{x}_1 = 1 - 0{,}2 * (0-1) * (-1) = 0{,}8$$
$$\mathbf{w}_2 = \mathbf{w}_2 - 0{,}2 * (0-1) * \mathbf{x}_2 = 0 - 0{,}2 * (0-1) * (1) = 0{,}2$$
$$b = b - 0{,}2 * (0-1) = 0 - 0{,}1 * (0-1) = 0{,}2$$



Modificar w y b para: $\alpha = 0.2$ para $x = (1\ -1)$:

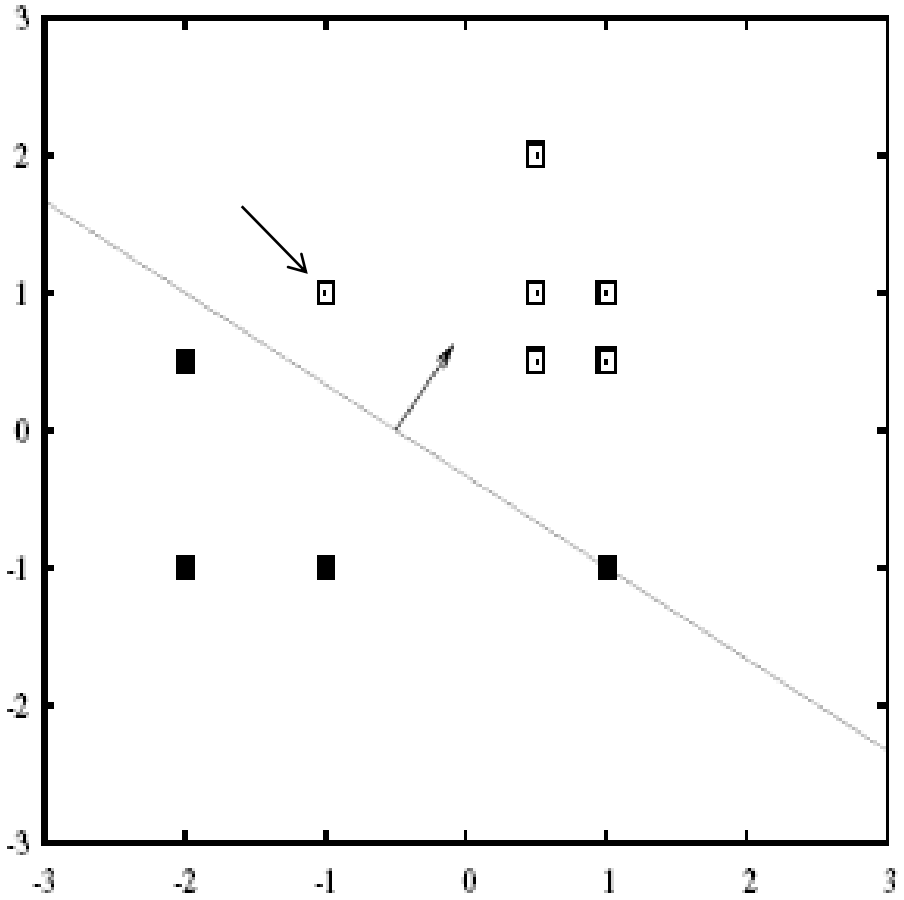
$$\mathbf{w}_1 = \mathbf{w}_1 - 0{,}1 * (1-0) * \mathbf{x}_1 = 0{,}8 - 0{,}2 * (1-0) * (1{,}0) = 0{,}6$$
$$\mathbf{w}_2 = \mathbf{w}_2 - 0{,}1 * (1-0) * \mathbf{x}_2 = 0{,}2 - 0{,}2 * (1-0) * (-1) = 0{,}4$$
$$b = b - 0{,}2(1-0) = 0{,}2 - 0{,}2 * (1-0) = 0{,}0$$



Modificar w y b para: $\alpha = 0.1$ de nuevo para $x = (-1\ 1)$:

$$\mathbf{w}_1 = \mathbf{w}_1 - 0{,}2 * (0-1) * \mathbf{x}_1 = 0{,}6 - 0{,}2 * (0-1) * (-1) = 0{,}4$$
$$\mathbf{w}_2 = \mathbf{w}_2 - 0{,}2 * (0-1) * \mathbf{x}_2 = 0{,}4 - 0{,}2 * (0-1) * (1) = 0{,}6$$
$$b = b - 0{,}2(0-1) = 0{,}0 - 0{,}2 * (0-1) = 0{,}2$$

Se puede verificar que todos los vectores quedan correctamente clasificados.
Fin del algoritmo.



NOTAS:

El algoritmo del perceptrón generará un vector de pesos válido para el problema en cuestión, si dicho vector existe.

El algoritmo convergerá a una solución **en un número finito** de iteraciones, si el problema es linealmente separable.

Es útil para resolver **problemas linealmente separables**.

Es útil para el entrenamiento de **arreglos de perceptrones en una capa**.

Teorema de convergencia del perceptrón:

Teorema de convergencia del perceptrón: Si dos clases de vectores A y B son linealmente separables, la aplicación del **algoritmo de entrenamiento del perceptrón** (AEP) eventualmente resultará en un vector de pesos W_0 tal que W_0 define una TLU cuya superficie de decisión separa a A de B .

Una vez que W_0 ha sido encontrado permanece estable, no más cambios ocurren en el vector de pesos.

Pruebas:

Rosenblatt, F. (1962). Principles of Neurodynamics. Spartan Books, New York.

M. Minsky and S. Papert (1969). Perceptron. MIT Press.

Haykin, S. (1994) Neural Networks. A Comprehensive Foundation. Macmillan, New York, NY.

Implicación del Teorema de convergencia del perceptrón:

Si las dos clases NO son linealmente separables, entonces el **Algoritmo de Entrenamiento del Perceptrón** (AEP) NO convergerá a una solución.

Ejemplo de entrenamiento:

Ejemplo. Entrenar una UUL de dos entradas para que realice la función lógica del "Y".

Pesos iniciales: $p_1 = 0.25, p_2 = 0.25$ y $\theta = 1.0$. Además sea $\alpha = 0.25$.

Solución: Para recta de separación entre clases, se recorren pares de entrenamiento como sigue: $\mathbf{X}_1 = (0\ 0)$, $t_1 = 0$, luego $\mathbf{X}_2 = (0\ 1)$, $t_2 = 0$, luego $\mathbf{X}_3 = (1\ 0)$, $t_3 = 0$, finalmente $\mathbf{X}_4 = (1\ 1)$, $t_4 = 1$.

Para cada par se aplica algoritmo en forma iterada hasta obtener la solución deseada.

Vectores aumentados:

$\mathbf{X}_1 = (0\ 0 - 1)$, $\mathbf{X}_2 = (0\ 1 - 1)$, $\mathbf{X}_3 = (1\ 0 - 1)$, y $\mathbf{X}_4 = (1\ 1 - 1)$.

Debido a que el problema es linealmente separable, el teorema de convergencia del percetrón garantiza que se podrá encontrar la UUL buscada.

Para facilitar la explicación se acomodarán los resultados iteración a iteración en una tabla como sigue:

Primera época ($p_1 = 0.25, p_2 = 0.25, p_3 = \theta = 1.00, r = p_1 x_1 + p_2 x_2 - \theta, \Delta p_i = \alpha(t - \mathrm{y}) x_i$):

x_1	x_2	p_1	p_2	θ	r	t	y	$\alpha(t-\mathrm{y})$	Δp_1	Δp_2	$\Delta p_3 = \Delta\theta$
0	0	0.25	0.25	1.00	-1.00	0	0	0.00	0.00	0.00	0.00
0	1	0.25	0.25	1.00	-0.75	0	0	0.00	0.00	0.00	0.00
1	0	0.25	0.25	1.00	-0.75	0	0	0.00	0.00	0.00	0.00
1	1	0.25	0.25	1.00	-0.50	1	0	0.25	0.25	0.25	-0.25

Primera época ($p_1 = 0.25, p_2 = 0.25, p_3 = \theta = 1.00, r = p_1 x_1 + p_2 x_2 - \theta, \Delta p_i = \alpha(t - y) x_i$):

x_1	x_2	p_1	p_2	θ	r	t	y	$\alpha(t - y)$	Δp_1	Δp_2	$\Delta p_3 = \Delta\theta$
0	0	0.25	0.25	1.00	-1.00	0	0	0.00	0.00	0.00	0.00
0	1	0.25	0.25	1.00	-0.75	0	0	0.00	0.00	0.00	0.00
1	0	0.25	0.25	1.00	-0.75	0	0	0.00	0.00	0.00	0.00
1	1	0.25	0.25	1.00	-0.50	1	0	0.25	0.25	0.25	-0.25

Segunda época ($p_1 = 0.50, p_2 = 0.50, p_3 = 0.75$):

x_1	x_2	p_1	p_2	θ	r	t	y	$\alpha(t - y)$	Δp_1	Δp_2	$\Delta p_3 = \Delta\theta$
0	0	0.50	0.50	0.75	-0.75	0	0	0.00	0.00	0.00	0.00
0	1	0.50	0.50	0.75	-0.25	0	0	0.00	0.00	0.00	0.00
1	0	0.50	0.50	0.75	-0.25	0	0	0.00	0.00	0.00	0.00
1	1	0.50	0.50	0.75	0.25	1	1	0.00	0.00	0.00	0.00

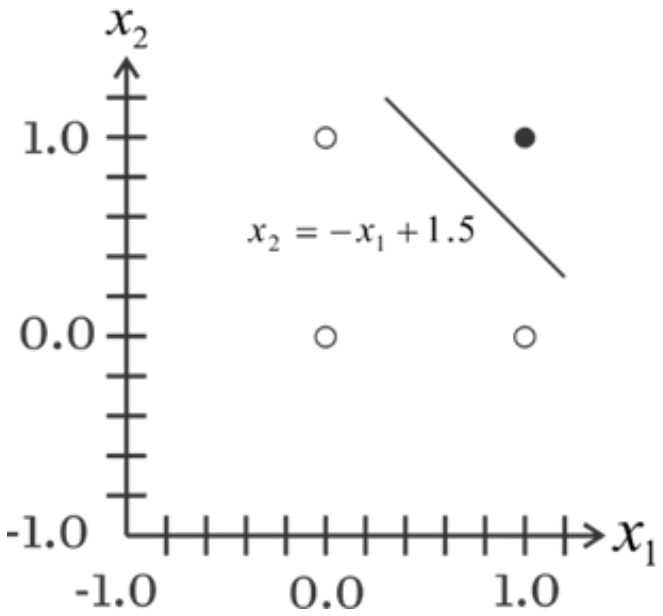
De esta última tabla, se puede ver que se ha logrado la convergencia. Luego entonces, los parámetros de la UUL que resuelven el problema quedan como sigue:

$$p_1 = 0.50, p_2 = 0.50 \text{ y } \theta = 0.75.$$

De acuerdo a lo visto, la ecuación de la frontera de separación entre clases viene dada como:

$$x_2 = -\left(\frac{p_1}{p_2}\right)x_1 + \left(\frac{\theta}{p_2}\right) = -\left(\frac{0.5}{0.5}\right)x_1 + \left(\frac{0.75}{0.5}\right) = -x_1 + 1.5.$$

En la figura se muestra esta línea de decisión, la cual, como se puede ver, logra separar las dos clases de puntos y, por tanto, implementar la función “Y” lógica.



Ejemplo. Entrenar una UUL de dos entradas que se quiere realice la función lógica del “O”.

Pesos iniciales se seleccionan los siguientes:

$p_1 = 0.5, p_2 = 0.5$ y $\theta = -0.5$.

Además sea $\alpha = 0.25$.

Solución: De nuevo, para encontrar los pesos de la recta de separación entre las dos posibles clases, se irán recorriendo los pares de entrenamiento como sigue:

$\mathbf{X}_1 = (0\ 0)$, $t_1 = 0$, luego $\mathbf{X}_2 = (0\ 1)$, $t_2 = 1$, luego $\mathbf{X}_3 = (1\ 0)$, $t_3 = 1$, y luego $\mathbf{X}_4 = (1\ 1)$, $t_4 = 1$.

Para cada uno de estos pares aplicar algoritmo en forma iterada hasta obtener la solución deseada.

Vectores aumentados son los siguientes:

$\mathbf{X}_1 = (0\ 0\ -1)$, $\mathbf{X}_2 = (0\ 1\ -1)$, $\mathbf{X}_3 = (1\ 0\ -1)$, y $\mathbf{X}_4 = (1\ 1\ -1)$.

De nuevo, se puede ver qué el problema es linealmente separable. De acuerdo a lo visto, el teorema de convergencia del percetrón garantiza que se podrá encontrar la UUL buscada.

Para facilitar la explicación, una vez más, se acomodarán los resultados iteración a iteración en una tabla como sigue:

Primera época ($p_1 = 0.50, p_2 = 0.50, p_3 = \theta = -0.50, r = p_1x_1 + p_2x_2 - \theta, \Delta p_i = \alpha(t - y)x_i$):

x_1	x_2	p_1	p_2	θ	r	t	y	$\alpha(t - y)$	Δp_1	Δp_2	$\Delta p_3 = \Delta\theta$
0	0	0.50	0.50	-0.50	0.50	0	1	-0.25	0.00	0.00	0.25
0	1	0.50	0.50	-0.25	0.75	1	1	0.00	0.00	0.00	0.00
1	0	0.50	0.50	-0.25	0.75	1	1	0.00	0.00	0.00	0.00
1	1	0.50	0.50	-0.25	1.25	1	1	0.00	0.00	0.00	0.00

Segunda época ($p_1 = 0.50, p_2 = 0.50, p_3 = \theta = -0.25$:

x_1	x_2	p_1	p_2	θ	r	t	y	$\alpha(t - y)$	Δp_1	Δp_2	$\Delta p_3 = \Delta\theta$
0	0	0.50	0.50	-0.25	0.25	0	1	-0.25	0.00	0.00	0.25
0	1	0.50	0.50	0.00	0.50	1	1	0.00	0.00	0.00	0.00
1	0	0.50	0.50	0.00	0.50	1	1	0.00	0.00	0.00	0.00
1	1	0.50	0.50	0.00	1.00	1	1	0.00	0.00	0.00	0.00

Tercera época ($p_1 = 0.50, p_2 = 0.50, p_3 = \theta = 0.00$:

x_1	x_2	p_1	p_2	θ	r	t	y	$\alpha(t - y)$	Δp_1	Δp_2	$\Delta p_3 = \Delta\theta$
0	0	0.50	0.50	0.00	0.00	0	1	-0.25	0.00	0.00	0.25
0	1	0.50	0.50	0.25	0.25	1	1	0.00	0.00	0.00	0.00
1	0	0.50	0.50	0.25	0.25	1	1	0.00	0.00	0.00	0.00
1	1	0.50	0.50	0.25	0.75	1	1	0.00	0.00	0.00	0.00

Cuarta época ($p_1 = 0.50, p_2 = 0.50, p_3 = \theta = 0.25$:

x_1	x_2	p_1	p_2	θ	r	t	y	$\alpha(t - y)$	Δp_1	Δp_2	$\Delta p_3 = \Delta\theta$
0	0	0.50	0.50	0.25	-0.25	0	0	0.00	0.00	0.00	0.00
0	1	0.50	0.50	0.25	0.25	1	1	0.00	0.00	0.00	0.00
1	0	0.50	0.50	0.25	0.25	1	1	0.00	0.00	0.00	0.00
1	1	0.50	0.50	0.25	0.75	1	1	0.00	0.00	0.00	0.00

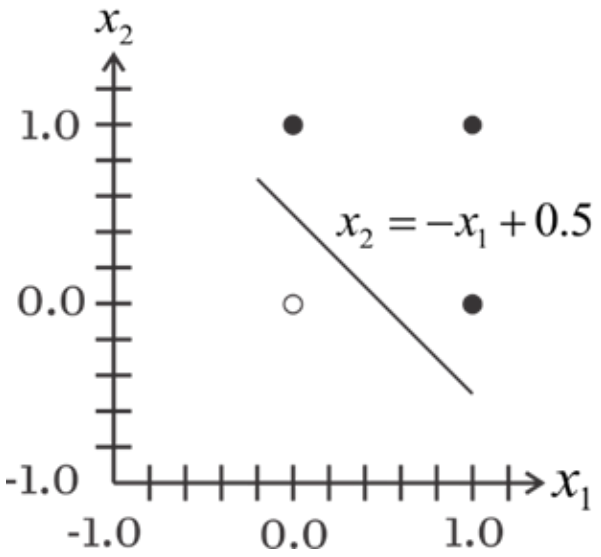
De esta última tabla, se puede ver que se ha logrado la convergencia. Luego entonces, los parámetros de la UUL que resuelven el problema quedan como sigue:

$$p_1 = 0.50, p_2 = 0.50 \text{ y } \theta = 0.25.$$

De acuerdo a lo visto, la ecuación de la frontera de separación entre clases viene dada como:

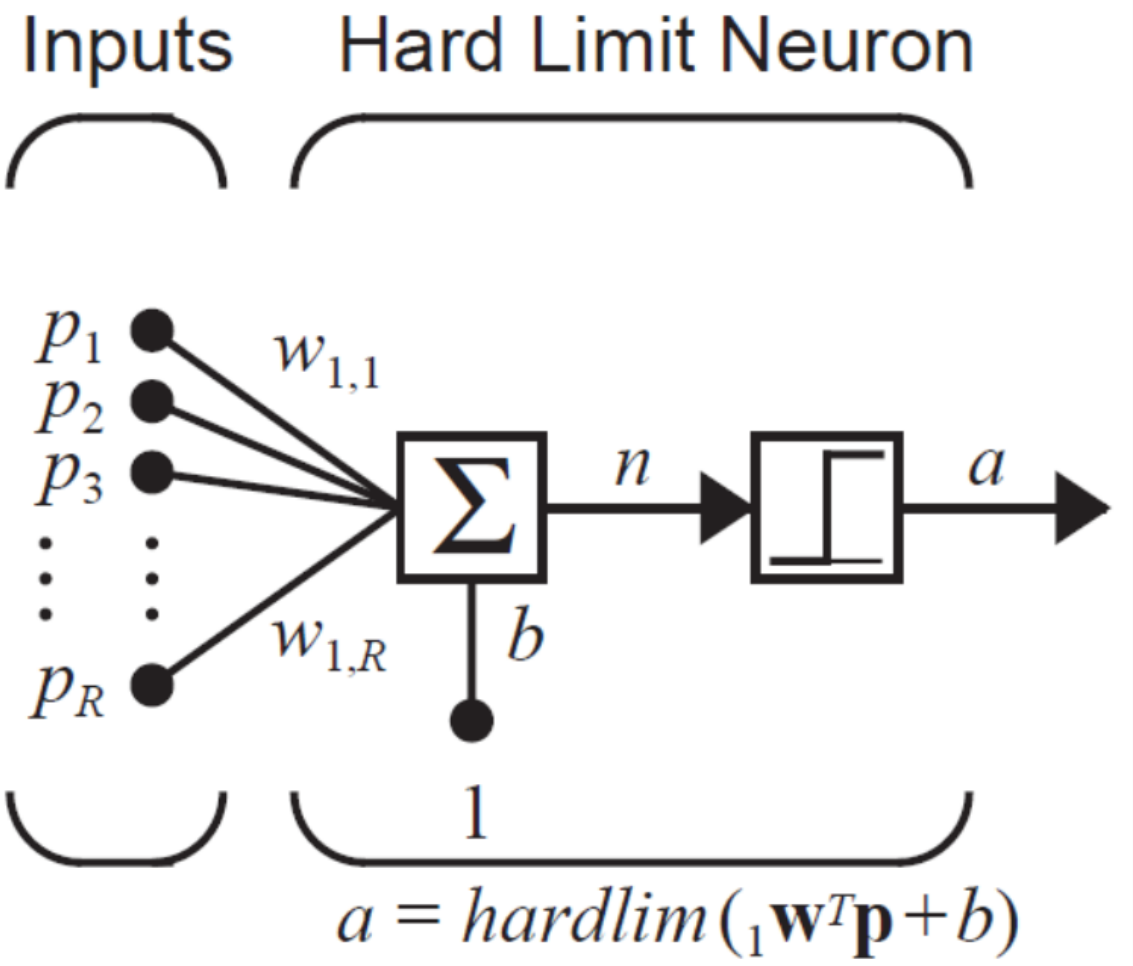
$$x_2 = -\left(\frac{p_1}{p_2}\right)x_1 + \left(\frac{\theta}{p_2}\right) = -\left(\frac{0.5}{0.5}\right)x_1 + \left(\frac{0.25}{0.5}\right) = -x_1 + 0.5.$$

En la figura se muestra esta línea de decisión, la cual, como se puede ver, logra separar las dos clases de puntos y, por tanto, implementar la función “O” lógica.

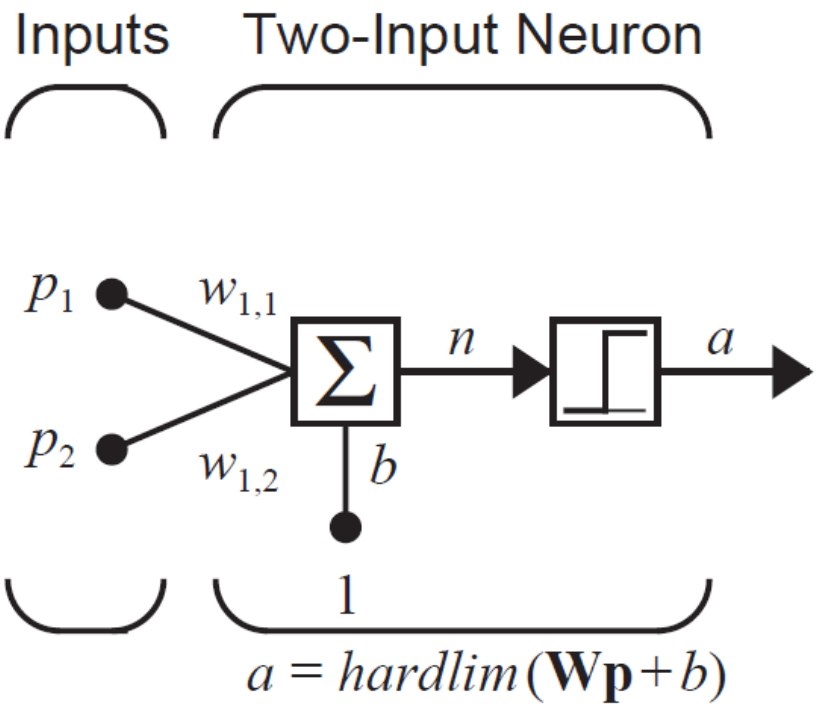


Resumen:

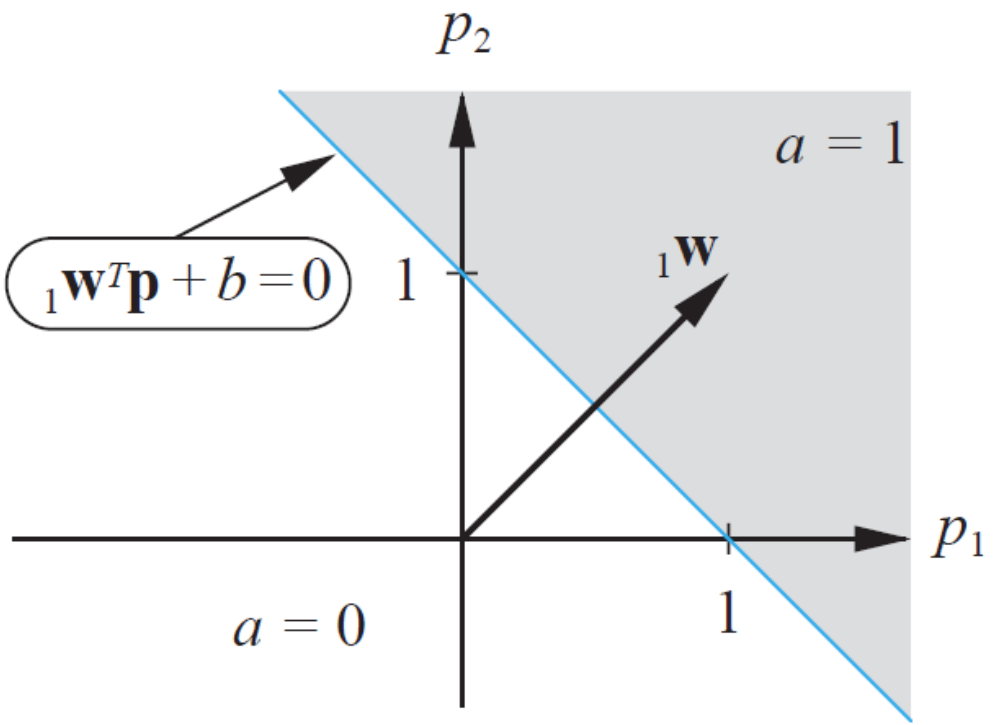
Perceptrón simple:



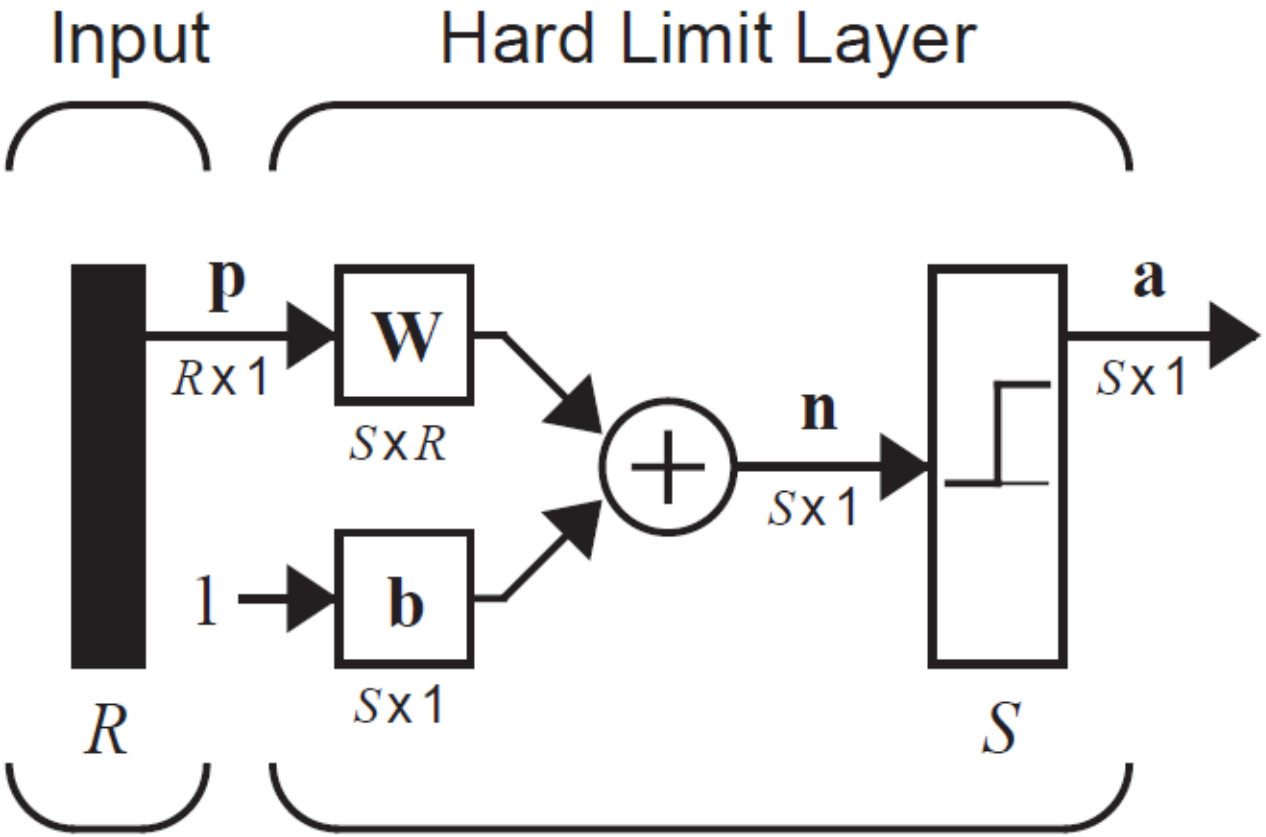
Perceptrón simple (dos entradas):



Su frontera de decisión:



Perceptrón múltiple en una capa:



$$\mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_S\mathbf{w}^T \end{bmatrix}$$

$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp} + \mathbf{b})$$

$$a_i = hardlim(n_i) = hardlim({}_i\mathbf{w}^T\mathbf{p} + b_i)$$

Su frontera de decisión:

$$_i\mathbf{w}^T\mathbf{p} + b_i = 0$$

La frontera generada es siempre perpendicular al vector de pesos.

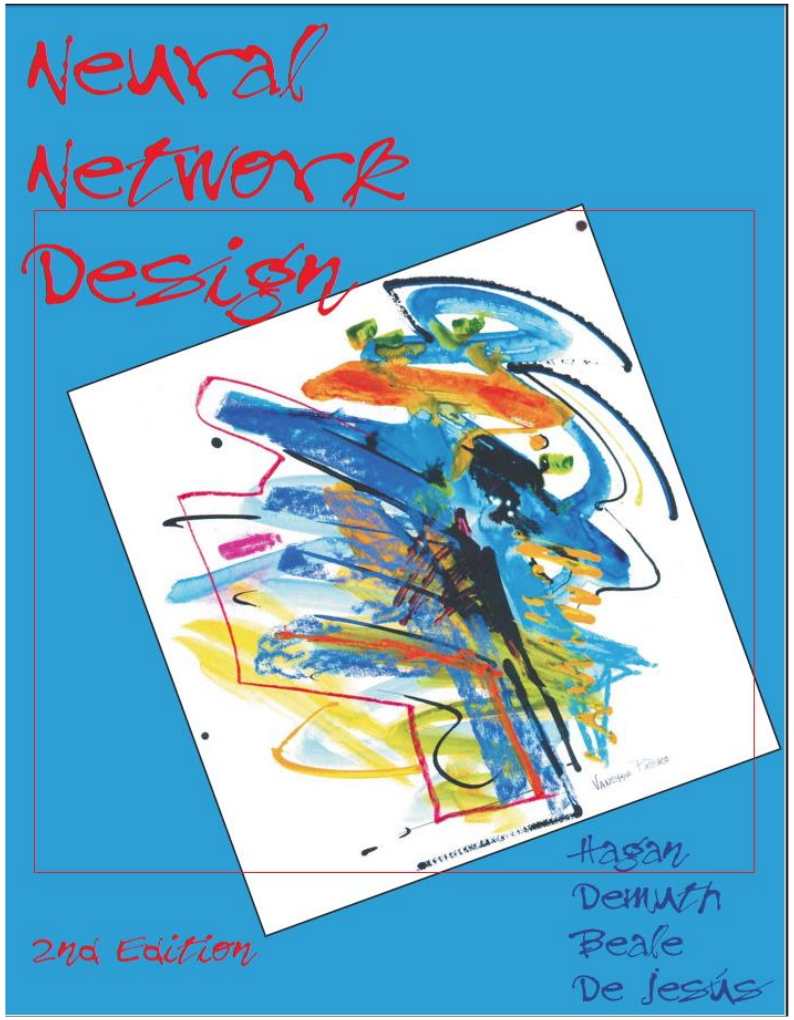
Perceptrones en una capa sólo pueden clasificar patrones linealmente separables.

Su regla de entrenamiento:

$$\mathbf{w}' = \mathbf{w} + \alpha(t - y)\mathbf{x}$$

Revisar el siguiente material:

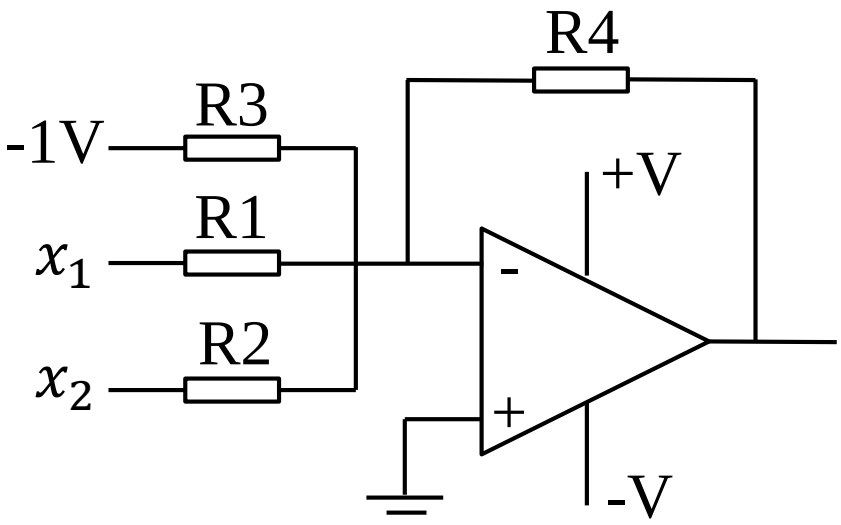
Páginas 4-21 a 4-42 y tratar de resolver los problemas E4.1 a E.4.13 del libro de M. Hagan.



Implementación en Hardware de RNAs:

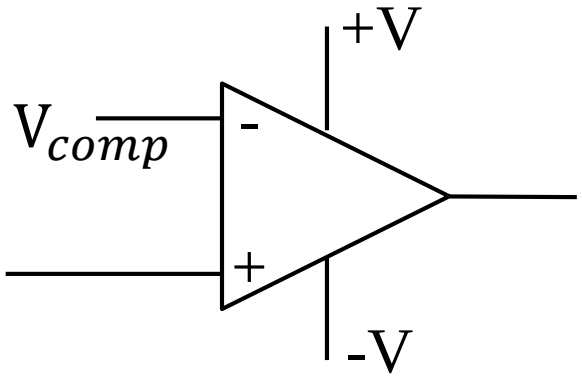
Implementación de una neurona TLU mediante amplificadores operacionales. Se requiere de:

Un sumador para realizar la operación de **producto interno** (caso de 2 entradas y desplazamiento):



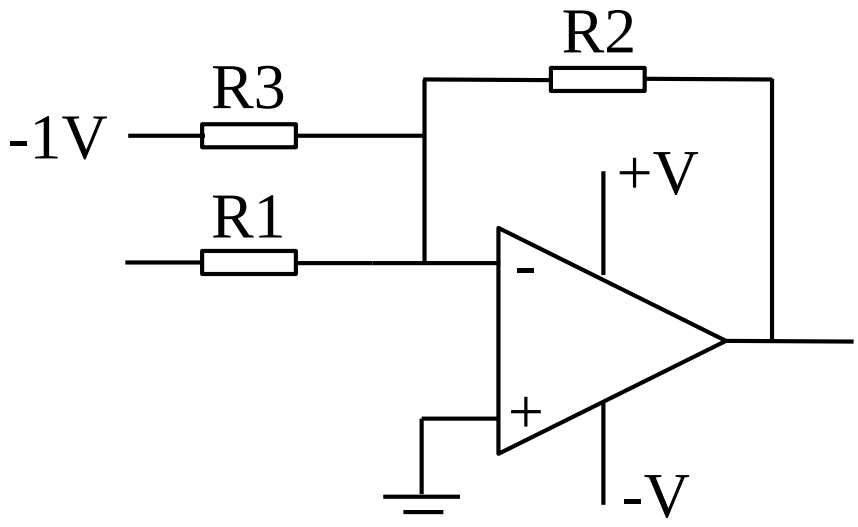
Implementación de una neurona TLU mediante amplificadores operacionales. Se requiere de:

Un comparador para **modelar la función de activación estándar**:



Implementación de una neurona TLU mediante amplificadores operacionales. Se requiere de:

Un acondicionador para **acondicionar** la salida del comparador a valores de voltaje en {0,1}:



La neurona TLU quedaría como: