

# Introducción a las RNAs y Modelos de RNAs Avanzadas



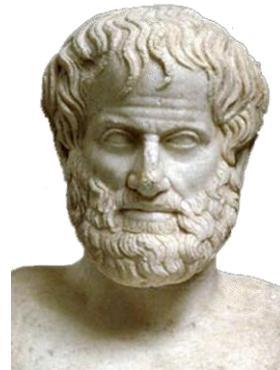
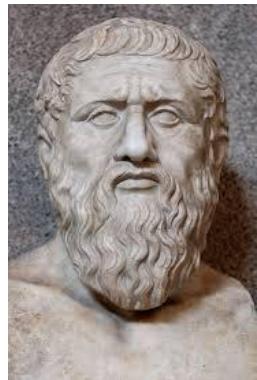
Juan Humberto SOSSA AZUELA

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

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

# **Panorama histórico sintético sobre las RNAs:**

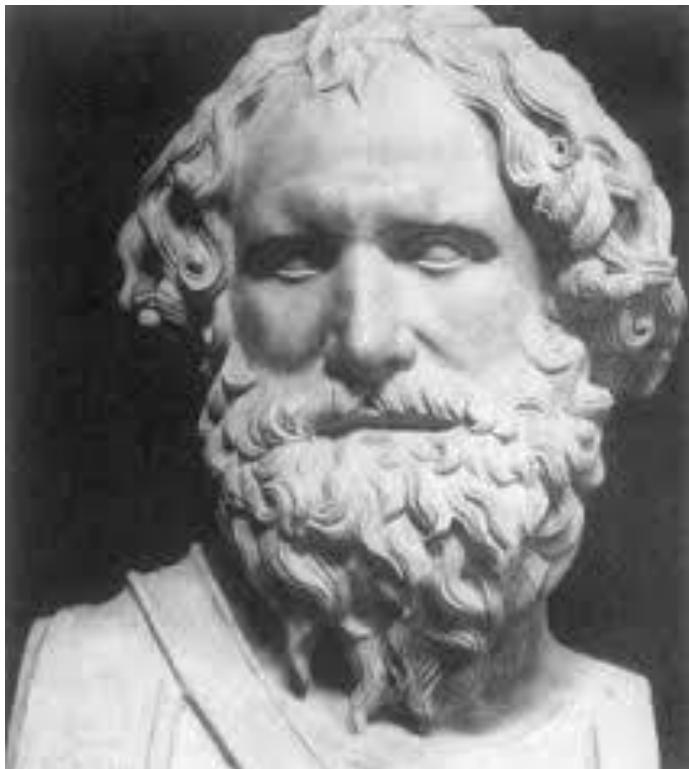
Las primeras explicaciones teóricas sobre el cerebro y el pensamiento fueron dadas ya por Platón (427-347 a.C.) y Aristóteles (348-322 a.C.).



Las mismas ideas también las mantuvo Descartes (1569-1650) y los filósofos empiristas del siglo XVIII.



La clase de las llamadas **máquinas cibernéticas**, a la cual la computación neuronal pertenece, tiene más historia de la que se cree: Herón (100 a.C) construyó un autómata hidráulico.



1936 Alan Turing. Fue el primero en estudiar el cerebro como una forma de ver el mundo de la computación.



Turing estableció los principios bajo los que operan las computadoras modernas.

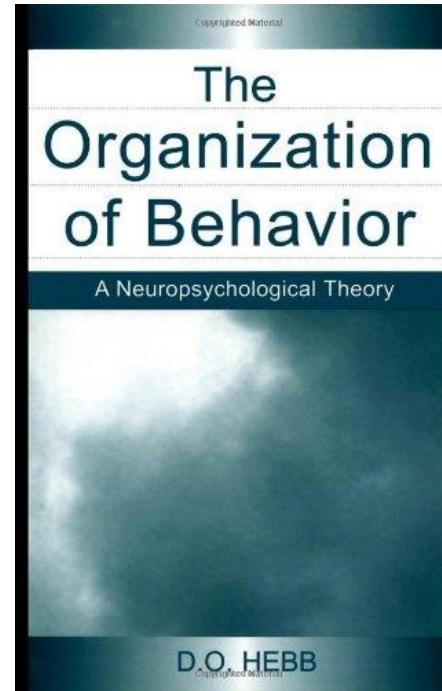
1943 Al parecer los primeros teóricos que **concibieron los fundamentos de la computación neuronal** fueron Warren McCulloch (neurofisiólogo) y Walter Pitts (matemático).



En 1943, lanzaron una teoría acerca de la **forma de trabajar de las neuronas** (Un Cálculo Lógico de la Inminente Idea de la Actividad Nerviosa - Boletín de Matemática Biofísica 5: 115-133).

Modelaron una **red neuronal simple mediante circuitos eléctricos**.

1949 En este año, Donald Hebb escribió un importante libro: **La organización del comportamiento**, en el que se establece una conexión entre la psicología y la fisiología.



Fue el primero en explicar los procesos del aprendizaje (que es el elemento básico de la inteligencia humana) desde un punto de vista psicológico.

Hebb desarrolló **una regla de como el aprendizaje ocurría:**

Esta regla establece que:

*Cuando el axón de una célula A está lo suficientemente cerca como para excitar a una célula B y repetidamente toma parte en la activación, ocurren procesos de crecimiento o cambios metabólicos en una o ambas células de manera que tanto la eficiencia de la célula A, como la capacidad de excitación de la célula B son aumentadas.*

Esta teoría se resume como:

**“las células que se disparan juntas, permanecerán conectadas”**

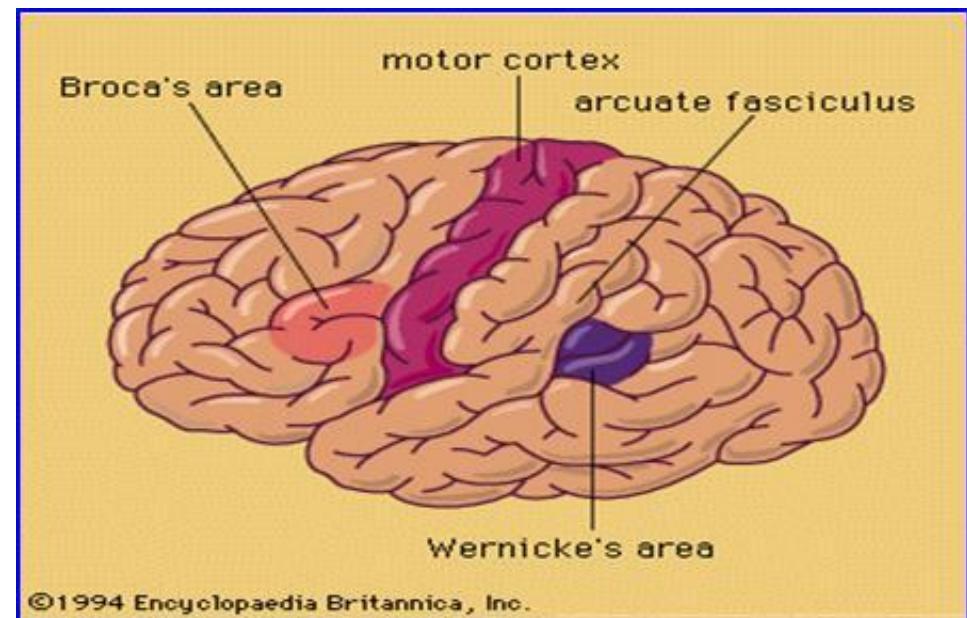
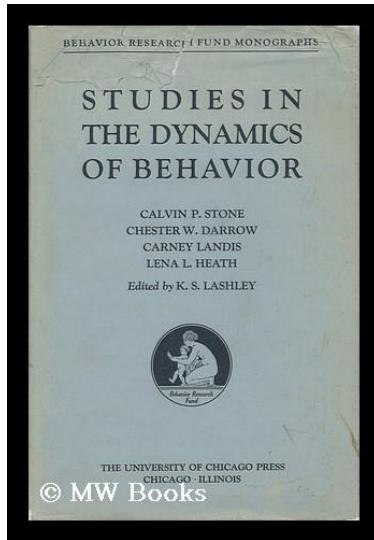
Aun hoy, este es el fundamento de la mayoría de las funciones de aprendizaje que pueden hallarse en una red neuronal.

Su idea es que **el aprendizaje ocurre** cuando ciertos cambios en una neurona son activados.

También intentó encontrar semejanzas entre el aprendizaje y la actividad nerviosa.

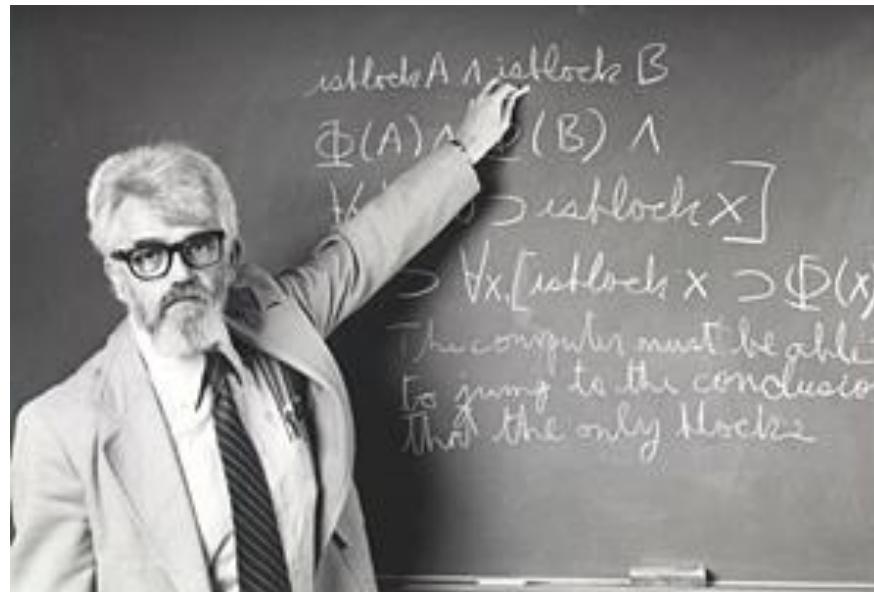
Los trabajos de Hebb forman las bases de la Teoría de las Redes Neuronales.

1950 En este año Karl Lashley encuentra en ensayos, que la información no se almacena en forma centralizada en el cerebro sino que está distribuida encima de él.



1956 El Congreso de Dartmouth se menciona frecuentemente para indicar el nacimiento de la inteligencia artificial.

Fue organizada por John McCarthy, uno de los padres de la IA.



Los temas discutidos versaron alrededor de las computadoras, procesamiento de lenguaje natural, **redes neuronales**, teoría de la computación, abstracción, creatividad, entre otras.

Algunos de los asistentes al Congreso de Dartmouth 50 años después:



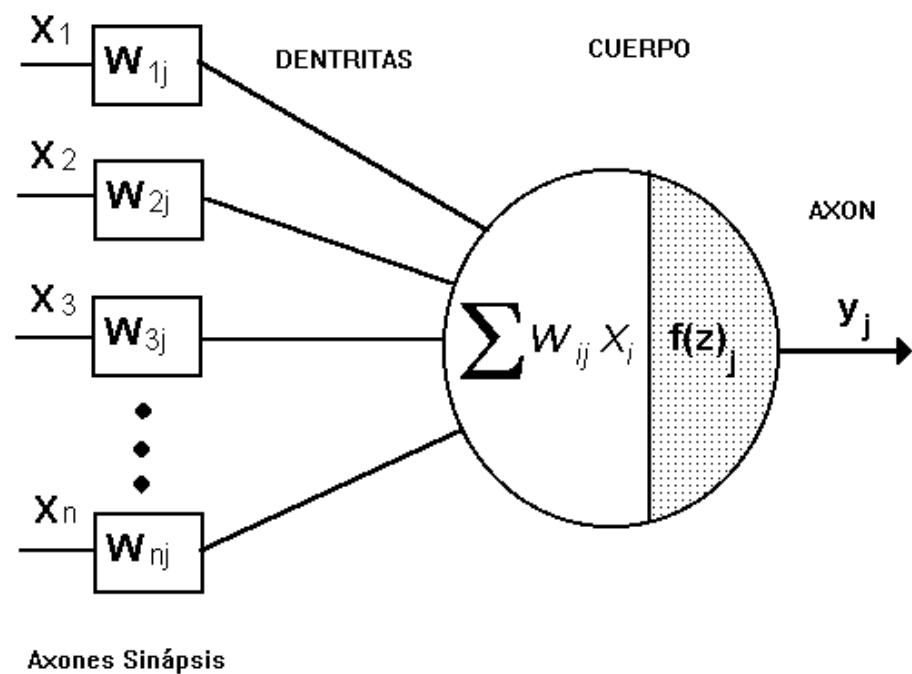
# Trenchard More

# John McCarthy

# Marvin Minsky

Oliver Ray  
Selfridge Solomonoff

1957 Frank Rosenblatt. Comenzó el desarrollo del Perceptrón. Se le considera como **la red neuronal artificial más antigua**; utilizándose como reconocedor de patrones.

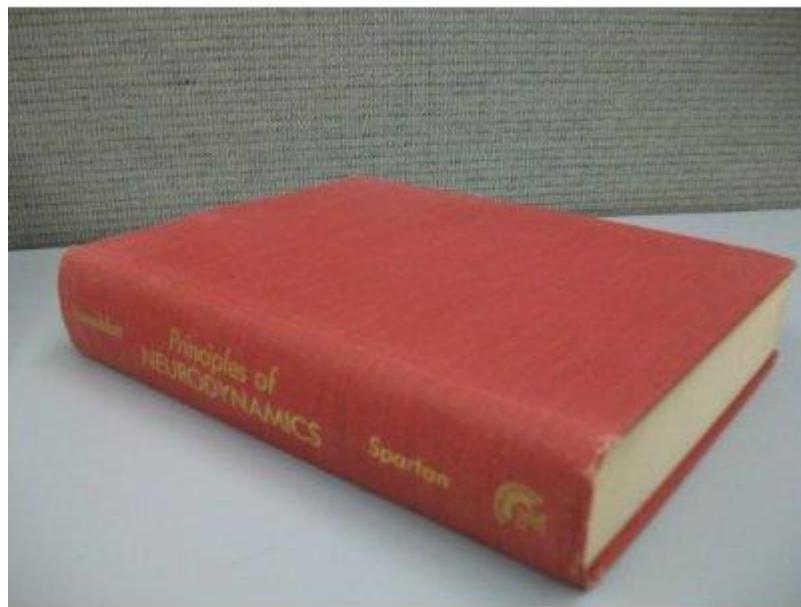


Este modelo **era capaz de generalizar**, es decir, después de haber aprendido una serie de patrones poder reconocer otros similares, aunque no se le hubiesen presentado anteriormente.

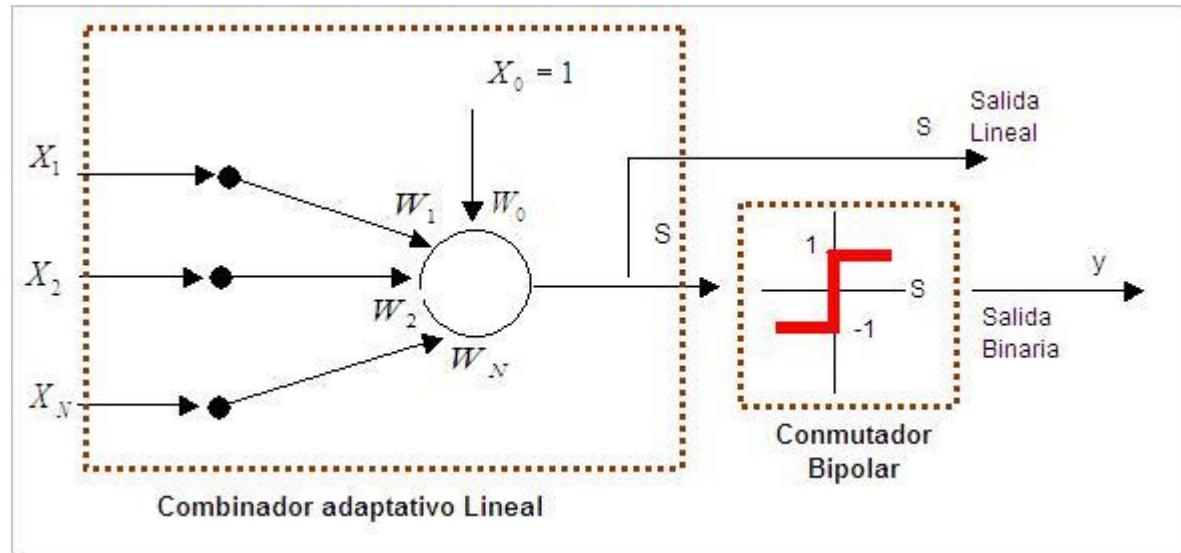
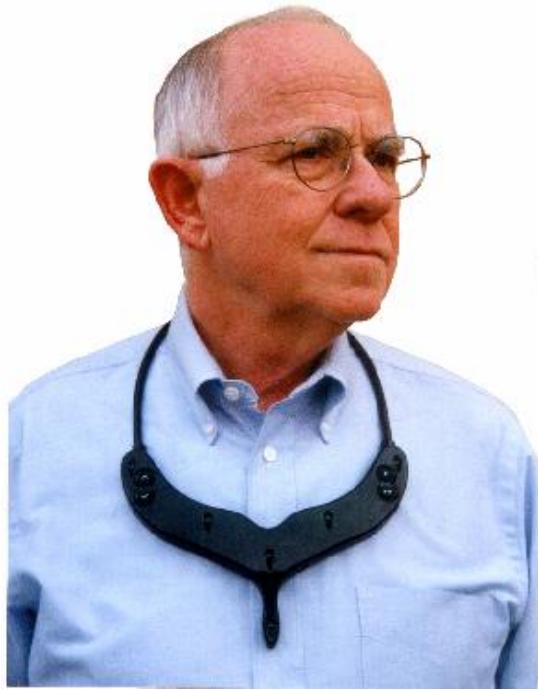
Sin embargo, tenía una serie de limitaciones, por ejemplo, su **incapacidad** para resolver el problema de la función OR-exclusiva y, en general, era incapaz de clasificar clases no separables linealmente.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

1959 Frank Rosenblatt escribió el libro Principios de Neurodinámica, en el que confirmó que, bajo ciertas condiciones, el aprendizaje del Perceptrón convergía hacia un estado finito (Teorema de Convergencia del Perceptrón).



1960 Bernard Widrow y Marcial Hoff desarrollaron el modelo Adaline (ADaptive LINear Elements).

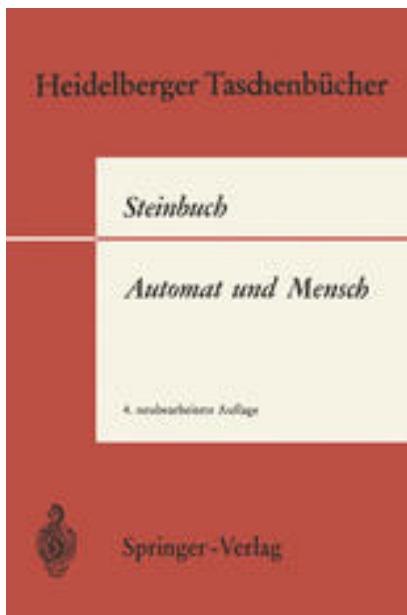


Esta fue la primera red neuronal aplicada a un problema real: filtro adaptativo para eliminar ecos en las líneas telefónicas.

Ha sido utilizado comercialmente durante varias décadas.

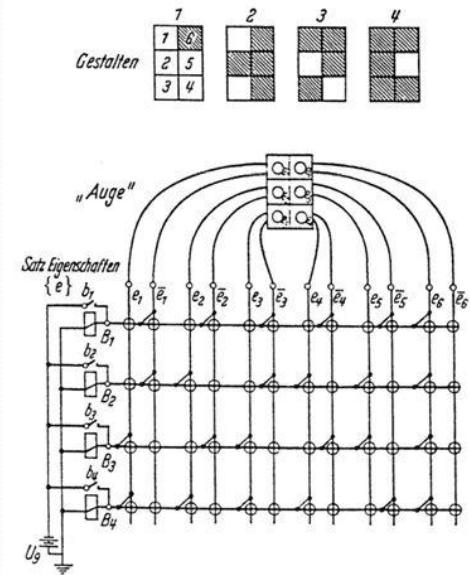
# 1961 Karl Steinbeck propone la conocida Die Lernmatrix.

Es una red neuronal que opera como una memoria asociativa.



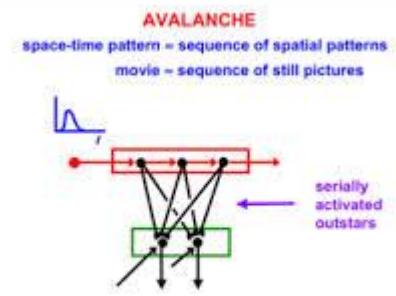
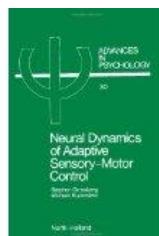
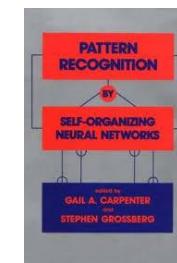
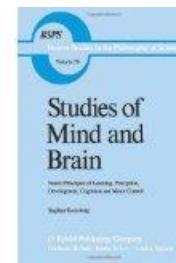
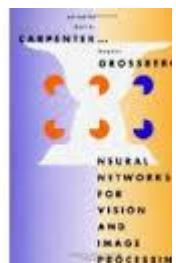
## 3. Die Lernmatrix

- originale Darstellung der Lernmatrix nach Steinbuch
- Kreisförmige Elemente an Kreuzungspunkten
- elektronisches „Auge“ wandelt Gestalten in el. Signale um (Eigenschaften)
- Ausgangssignale werden nach links abgegeben (Bedeutungen)
- Satz von Eigenschaften führt zu Satz von Bedeutungen



Prinzip der Lernmatrix

1967 Stephen Grossberg. A partir de sus conocimientos fisiológicos, ha escrito numerosos libros y desarrollado modelo de redes neuronales.

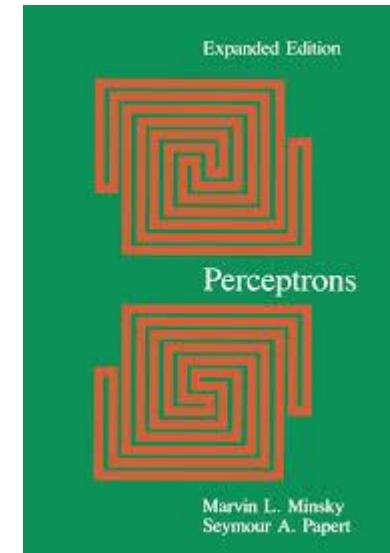


Realizó la **red Avalanche**, que consiste en elementos discretos con actividad que varía en el tiempo que satisface ecuaciones diferenciales continuas, para resolver actividades como reconocimiento continuo de habla y aprendizaje de los brazos de un robot.

1969 Marvin Minsky y Seymour Papert (MIT). En este año surgieron críticas que frenaron, hasta 1982, el crecimiento que estaban experimentando las investigaciones sobre redes neuronales.



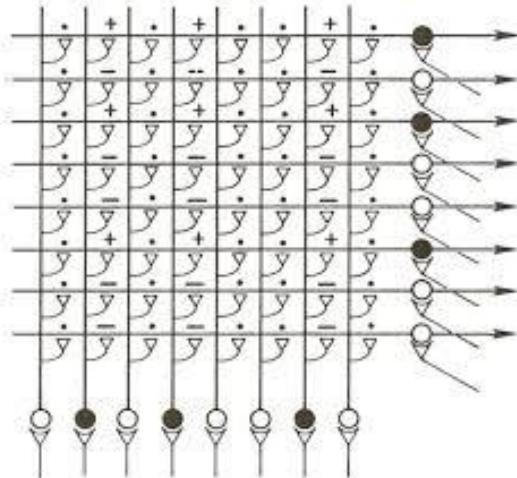
Minsky y Papert publican un libro **Perceptrons**.



Probaron (matemáticamente) que el Perceptrón **no era capaz de resolver problemas relativamente fáciles**, tales como el aprendizaje de una función no-lineal. Esto demostró que el Perceptrón era muy débil, dado que las funciones no-lineales son extensamente empleadas en computación y en los problemas del mundo real.

A pesar del libro, algunos investigadores continuaron su trabajo.

Tal fue el caso de James Anderson, que desarrolló un modelo lineal, llamado **Asociador Lineal**, que consistía en unos elementos integradores lineales (neuronas) que sumaban sus entradas. Este modelo se basa en el principio de que las conexiones entre neuronas son reforzadas cada vez que son activadas.



En 1977 Anderson diseñó una potente extensión del Asociador Lineal, llamada Brain State in a Box (BSB).

1974 Paul Werbos desarrolló **la idea básica del algoritmo de aprendizaje de propagación hacia atrás** (backpropagation); cuyo significado quedó definitivamente aclarado en 1985.

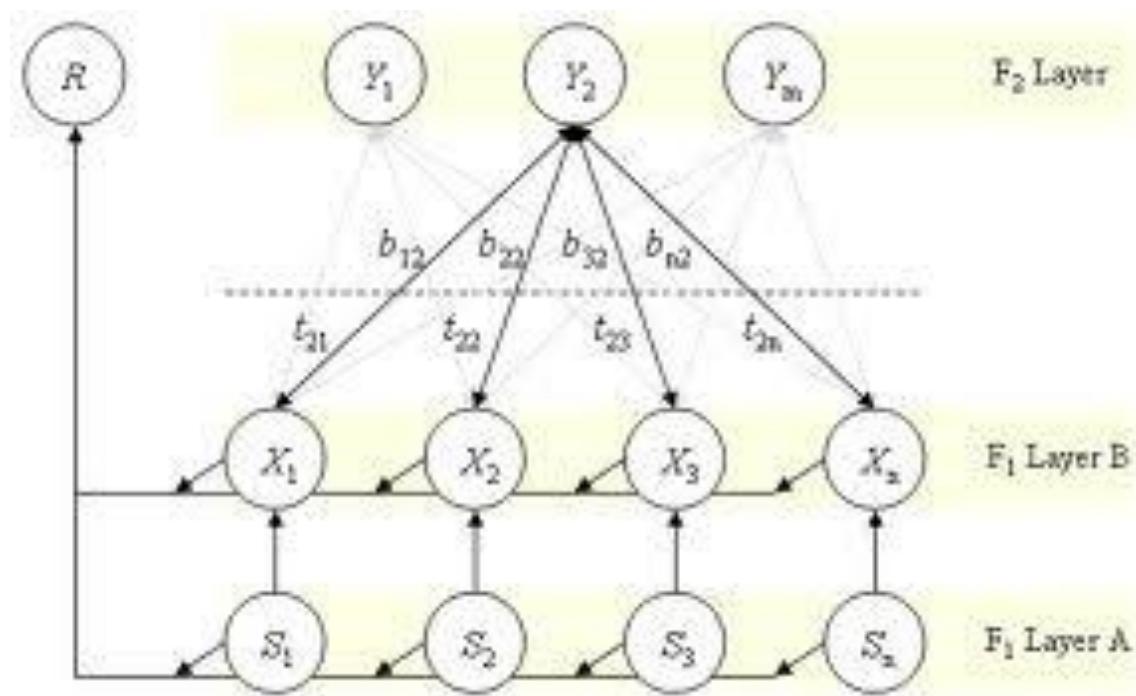


Paul J. Werbos. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD thesis, Harvard University, 1974.

Fue Werbos realmente quien propuso el algoritmo de entrenamiento de propagación hacia atrás para las redes compuestas por perceptrones.

1977 Stephen Grossberg propone su Teoría de Resonancia Adaptada (TRA).

La Teoría de Resonancia Adaptada es una arquitectura de red que se diferencia de todas las demás previamente inventadas. La misma simula otras habilidades del cerebro: memoria a largo y corto plazo.



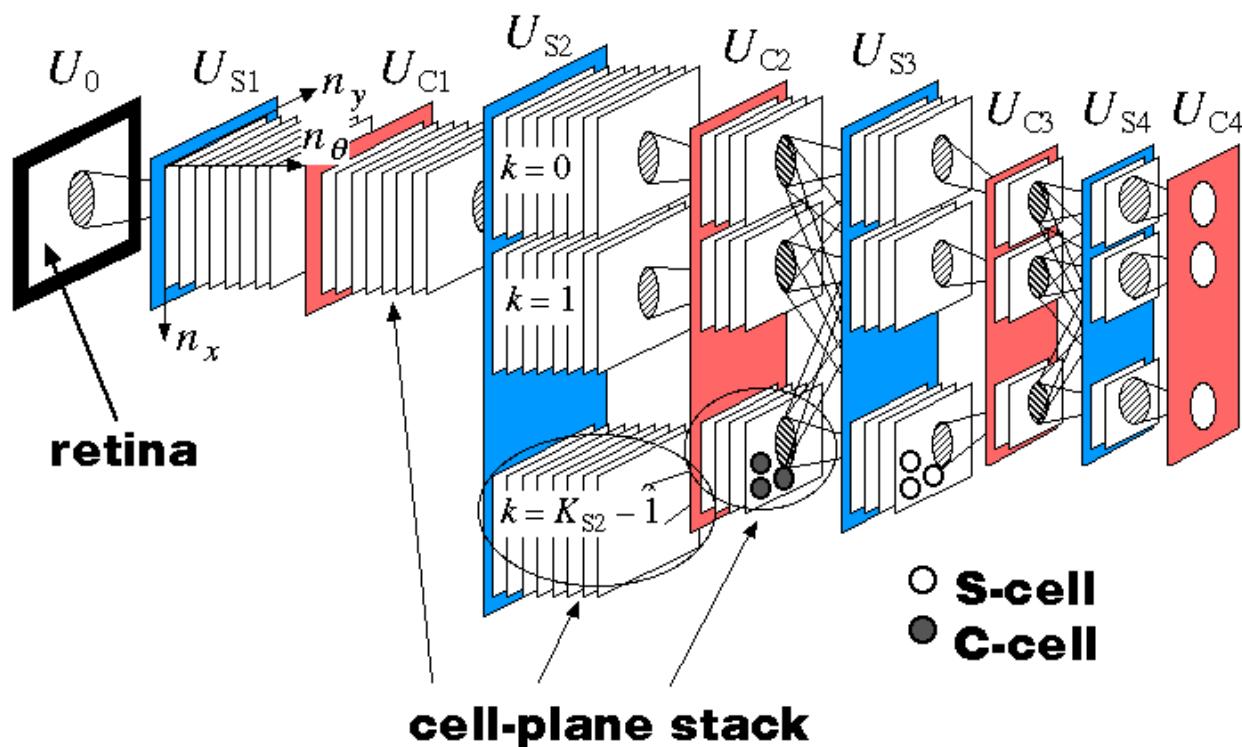
1977 Teuvo Kohonen. Ingeniero electrónico de la Universidad de Helsinki, desarrolló un modelo similar al de Anderson, pero independientemente.



1980 Kunihiko Fukushima. Desarrolló un modelo neuronal para el reconocimiento de patrones visuales (Neocognitron).



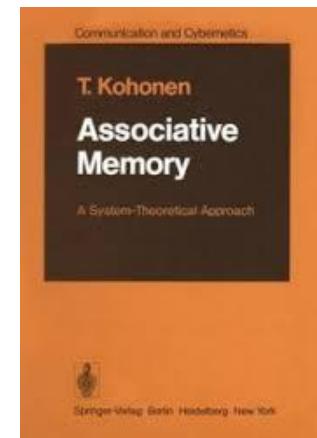
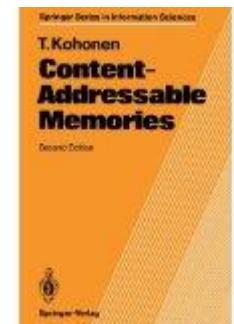
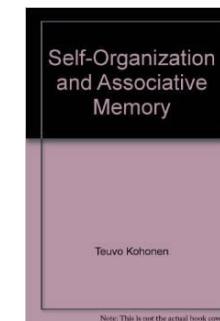
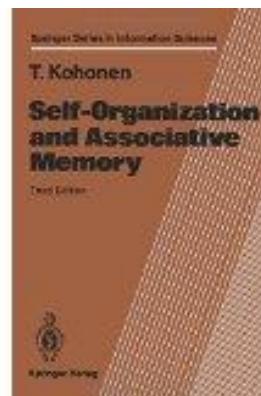
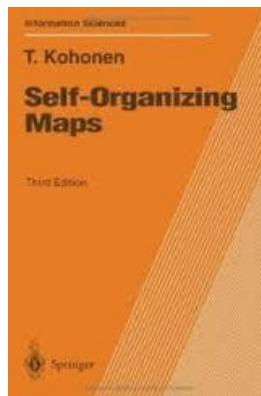
La teoría básica de las Redes Neuronales profundas fue desarrollada por Fukushima



# 1977 Teuvo Kohonen describe su famoso SOM:

Kohonen, Teuvo (1982). Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics* 43(1):59–69.

Sobre el tema, escribe varios libros:



También publica libros sobre memorias asociativas:

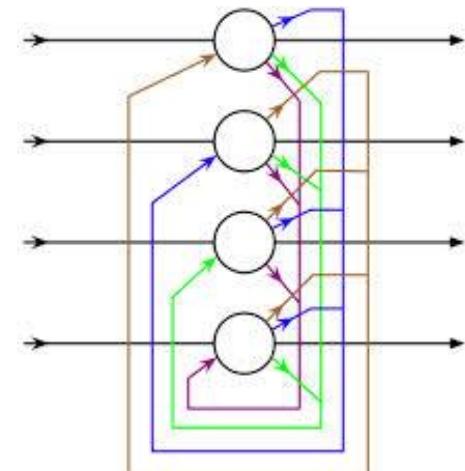
1985 John Hopfield. Provocó el renacimiento de las redes neuronales en sus dos trabajos:

Neural networks and physical systems with emergent collective computational abilities, Proc. NatL Acad. Sci. 79:2554-2558, April 1982.

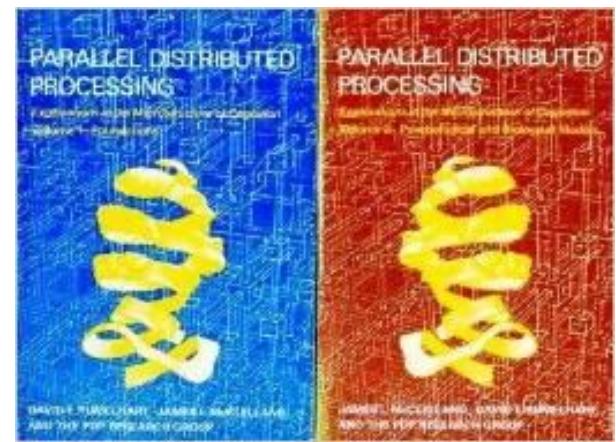
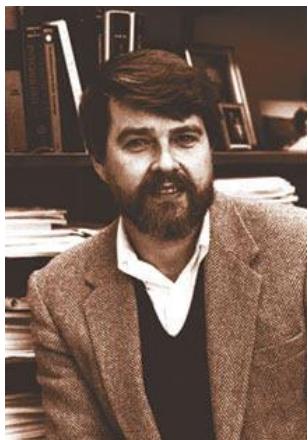
"Neural" Computation of Decisions in Optimization Problems  
Biol. Cybern 52:141-152, 1985.



Inventó la famosa red de Hopfield:



1986 David Rumelhart y G. Hinton. Redescubrieron el algoritmo de aprendizaje de propagación hacia atrás (backpropagation).



Son de particular interés los libros: Parallel Distributed Processing - 2 Vol. Set: Explorations in the Microstructure of Cognition by Rumelhart, David E.; McClelland, James L.; Group, the PDP Re published by The MIT Press Paperback

A partir de 1986, el panorama ha sido muy alentador con respecto a las investigaciones y el desarrollo de las redes neuronales.

En la actualidad, son numerosos los trabajos que se realizan y publican cada año, las aplicaciones nuevas que surgen:

(sobre todo en el área de control).

y las empresas que lanzan al mercado productos nuevos, tanto hardware como software

(sobre todo para simulación).

# **Conceptos básicos y clasificación de las RNAs:**

**Vamos a revisar, entre otros, los siguientes términos:**

Neurona.

Cuerpo neuronal.

Dendrita.

Sinapsis.

Axón.

Red neuronal.

Patrón.

Clase o categoría.

Asociación.

**Neurona:** Elemento básico de procesamiento de información de una red neuronal.

**Neurona:** Elemento básico de procesamiento de información de una red neuronal.

**Cuerpo neuronal:** Parte de la neurona que recibe la información procesada del conjunto de dendritas conectadas a él.

**Neurona:** Elemento básico de procesamiento de información de una red neuronal.

**Cuerpo neuronal:** Parte de la neurona que recibe la información procesada del conjunto de dendritas conectadas a él.

**Dendrita:** Rama de entrada a una neurona que recibe información de otras neuronas, que la procesa y envía su resultado al cuerpo de la neurona.

**Neurona:** Elemento básico de procesamiento de información de una red neuronal.

**Cuerpo neuronal:** Parte de la neurona que recibe la información procesada del conjunto de dendritas conectadas a él.

**Dendrita:** Rama de entrada a una neurona que recibe información de otras neuronas, que la procesa y envía su resultado al cuerpo de la neurona.

**Sinapsis:** Espacio de conexión entre dos neuronas.

**Neurona:** Elemento básico de procesamiento de información de una red neuronal.

**Cuerpo neuronal:** Parte de la neurona que recibe la información procesada del conjunto de dendritas conectadas a él.

**Dendrita:** Rama de entrada a una neurona que recibe información de otras neuronas, que la procesa y envía su resultado al cuerpo de la neurona.

**Sinapsis:** Espacio de conexión entre dos neuronas.

**Axón:** Elemento de la neurona que sirve para trasmitir hacia otra neuronas un resultado.

## **Red neuronal:**

**Definición 1.** Una red neuronal (RN) es un **conjunto interconectado de elementos simples de procesamiento**, unidades o nodos, cuya funcionalidad se basa de manera muy vaga con la neurona animal. La habilidad de procesamiento de la red se encuentra almacenada en las **conexiones** entre unidades (pesos). Estos son obtenidos mediante un proceso de **adaptación** o de **aprendizaje** a partir de un conjunto de entrenamiento.

**Definición 2.** Una red neuronal es un procesador distribuido masivamente paralelo construido de unidades de procesamiento simples que, de manera natural, es propenso a almacenar conocimiento experimental poniéndolo a disposición para su uso.

**Patrón:** Arreglo de elementos perteneciente a una clase o categoría útil para entrenar y probar el desempeño de una neurona o red neuronal.

Ejemplo de un patrón con  $n$  elementos:  $X^j = [x_1, x_2, \dots, x_n]^T, j = 1, 2, \dots, M$

En este caso  $M$  es el número de patrones del conjunto total  $C_T = \{X^1, X^2, \dots, X^M\}$ .

**Clase o categoría:** Familia a la cual pertenece un subconjunto  $C_k \in C_T$ . En este caso  $k = 1, 2, \dots, p$ , con  $p$  el número de clases o categorías.

Si  $t^j$  es la etiqueta (un número) que designa la clase a la cual pertenece el patrón  $X^j$ , entonces la notación:  $X^j, t^j$  o  $X^j \rightarrow t^j$  indica la pertenencia del patrón  $X^j$  a la clase  $C_{t^j}$ .

**Asociación:** Es la liga entre dos patrones. Si  $X$  y  $Y$  son dos patrones, la notación  $X, Y$  indica que estos dos patrones están asociados o ligados.

La notación  $(X^k, Y^k)^k$ , por otro lado, indica que se trata de la asociación  $k$ .

Puede suceder que las dimensiones de los patrones asociados sean diferentes. Si esto sucede, se supondrá lo siguiente:

$$X = [x_1, x_2, \dots, x_n]^T$$

$$Y = [y_1, y_2, \dots, y_m]^T$$

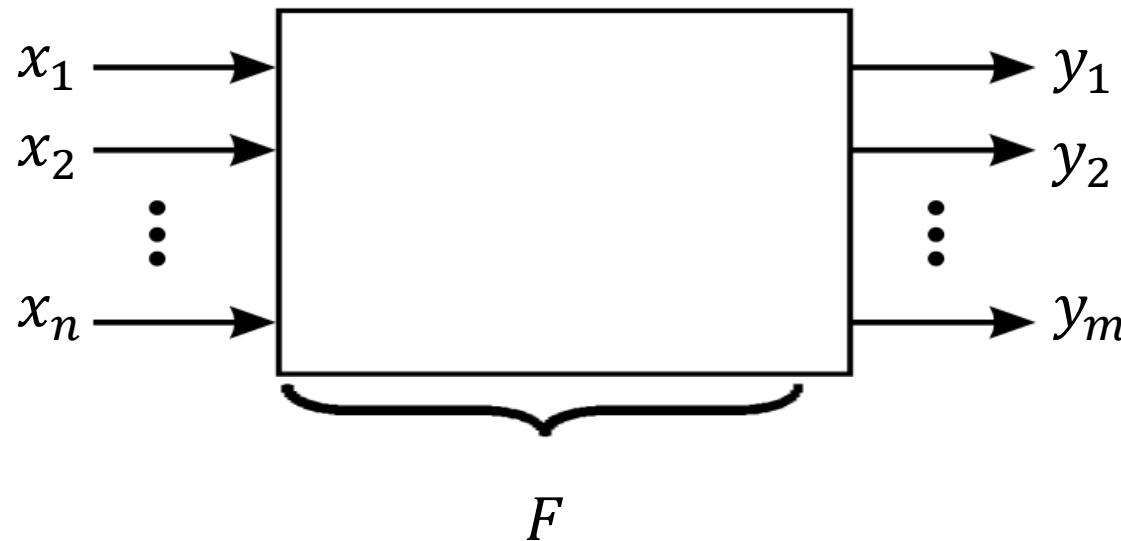
**Memoria asociativa (MA):** Dispositivo que permite ligar una o mas asociaciones de patrones a través de la codificación de la información de sus elementos.

Una memoria asociativa permite recuperar un patrón  $Y$  a través de un patrón  $X$  cuanto éste es presentado como entrada a la memoria. Si dicha memoria se designa, por ejemplo, como  $M$ , entonces la notación:

$$X \rightarrow M \rightarrow Y$$

indicará la recuperación del patrón  $Y$  mediante el patrón  $X$  a través de la memoria  $M$ .

En lo general se debe ver a una RNA como un mapeo entre un vector  $X = [x_1, x_2, \dots, x_n]^T$  a un vector  $Y = [y_1, y_2, \dots, y_m]^T$ .



Una entrada dada deberá producir una entrada deseada.

Una RNA se comporta pues como un mecanismo de mapeo capaz de modelar la función:  $F = \mathbb{R}^n \rightarrow \mathbb{R}^m$

NOTA: Una MA es un caso especial de RNA que realiza el mapeo de  $X$  a  $Y$  (en lo general) en un solo paso.

**Conjuntos  
fundamental y de  
prueba:**

Para la operación de una RNA se requiere de dos conjuntos:

Uno de **entrenamiento** o de **ajuste de parámetros** que lleva ese nombre (CE). Sea este conjunto:  $X^k, t^k, k = 1, 2, \dots, M$ .

Uno de **prueba** (CP) a través del cual se quiere atestiguar si la RNA es capaz de **generalizar**, esto es verificar si la RNA puede clasificar a otros patrones en sus respectivas clases. Sea este conjunto:  $X^j, t^j, j = 1, 2, \dots, N$ . En este caso los  $N$  patrones son diferentes a los usados para entrenar a la RNA.

Para la operación de una MA se requiere de dos conjuntos.

Uno de **construcción**, llamado **conjunto fundamental** (CF) de asociaciones. Sea este conjunto:  $(X^k, Y^k), k = 1, 2, \dots, M$ , con  $M$  el número de asociaciones.

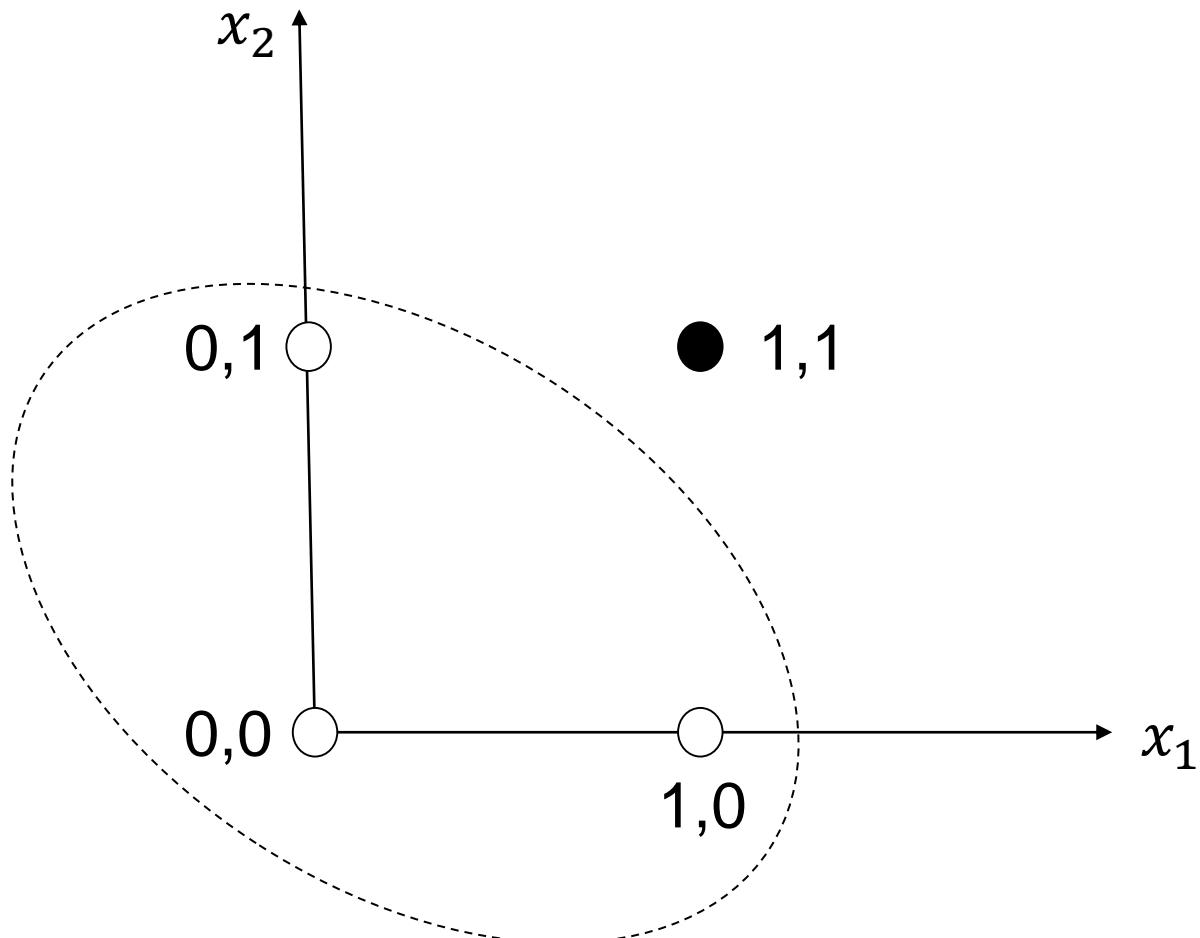
El CF y un **conjunto de prueba** (CP); a través de estos dos conjuntos se quiere probar si la MA construida es capaz, primero recuperar el CF y en qué porcentaje, y segundo recuperar, también y en qué porcentaje el conjunto de prueba.

El CF viene expresado como  $(X^k, Y^k), k = 1, 2, \dots, P$ , con  $P$  el número de asociaciones. El CP viene expresado como:  $(\tilde{X}^k, Y^k), k = 1, 2, \dots, Q$ , con  $Q$  el número de asociaciones de prueba.

**Ejemplo** de un conjunto de entrenamiento para el caso de una red neuronal operando **como clasificador**:

Ejemplo:  $p = 2, n = 2, M = 4$ :  $(0,0), 0; (0,1), 0; (1,0), 0; (1,1), 1$  es un conjunto de entrenamiento para el problema de AND.

$x_1$	$x_2$	$t^k$
0	0	0
0	1	0
1	0	0
1	1	1

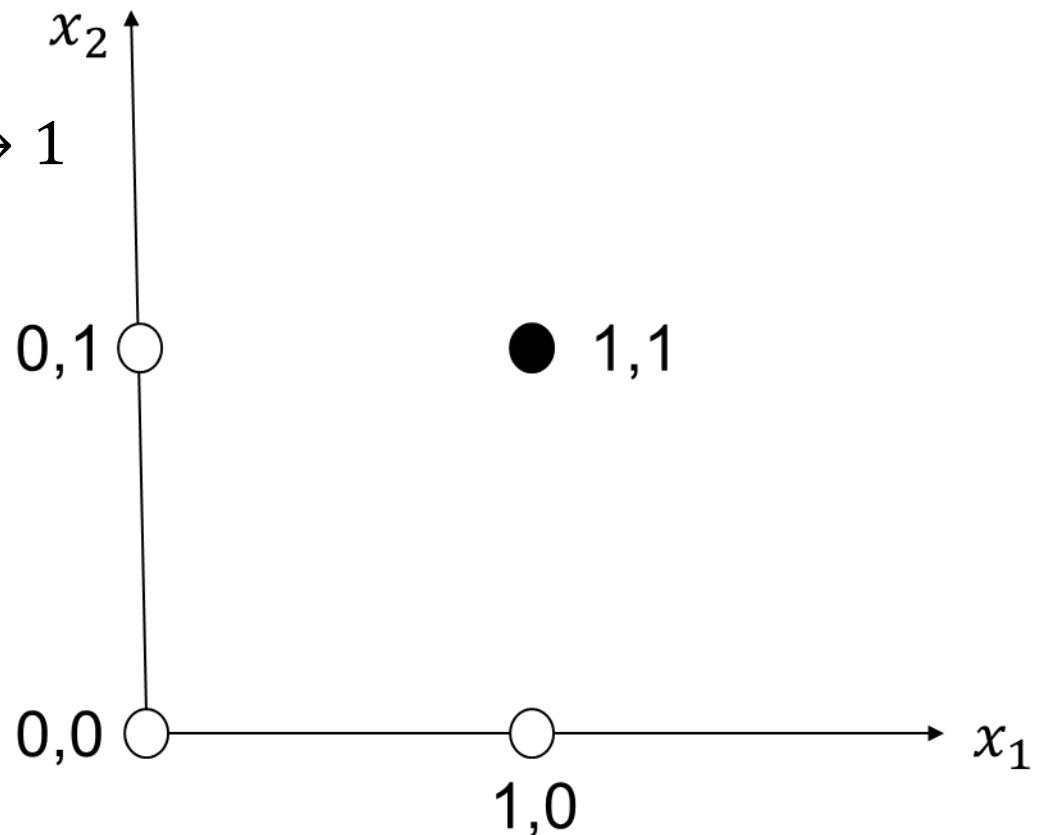


**Ejemplo** de un conjunto de prueba para el caso de una red neuronal operando **como clasificador** (ejemplo anterior):

Ejemplo:

$$(0.1,0.2), (0.1,0.9), (1.1,0.2) \rightarrow 0$$

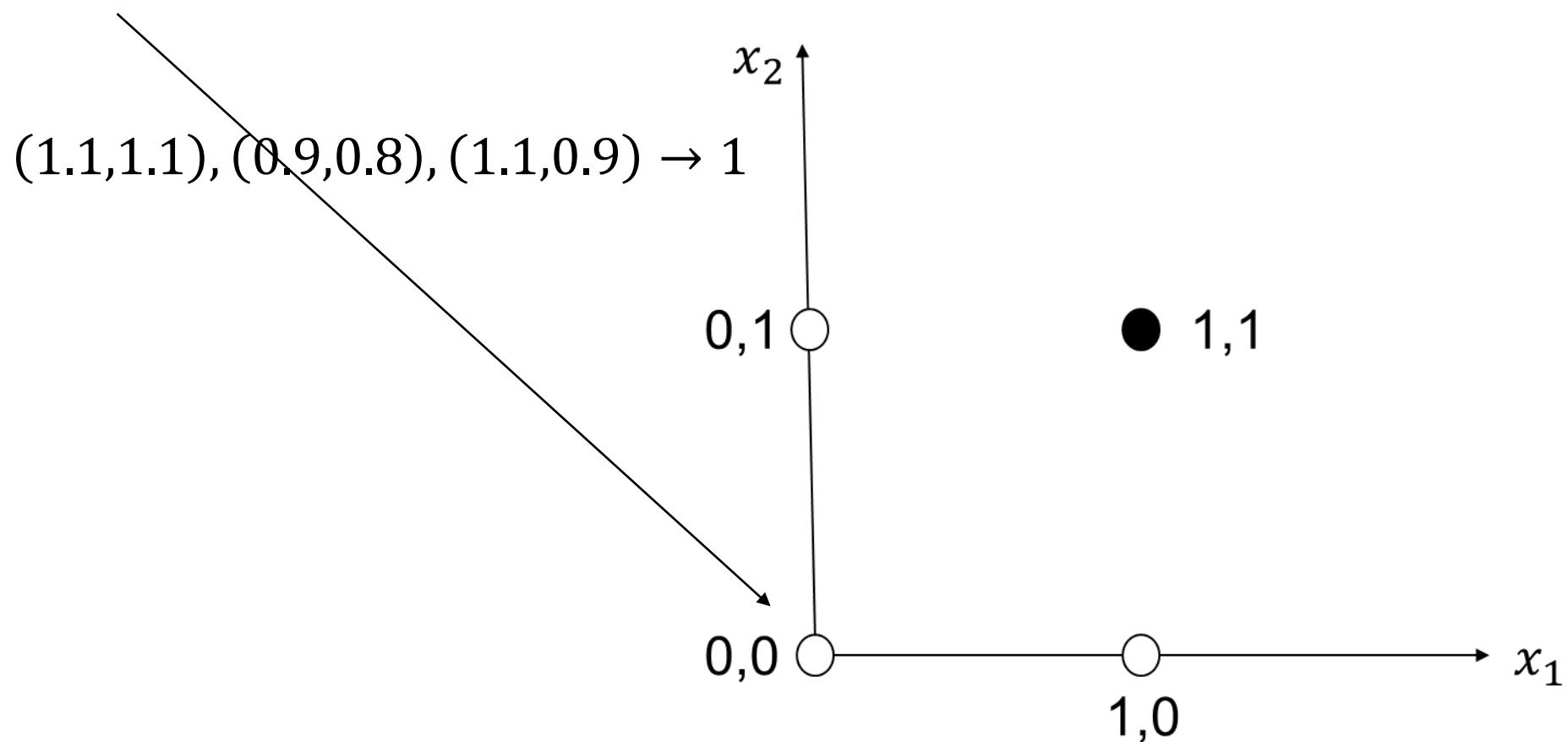
$$(1.1,1.1), (0.9,0.8), (1.1,0.9) \rightarrow 1$$



**Ejemplo** de un conjunto de prueba para el caso de una red neuronal operando **como clasificador** (ejemplo anterior):

Ejemplo:

$$(0.1,0.2), (0.1,0.9), (1.1,0.2) \rightarrow 0$$

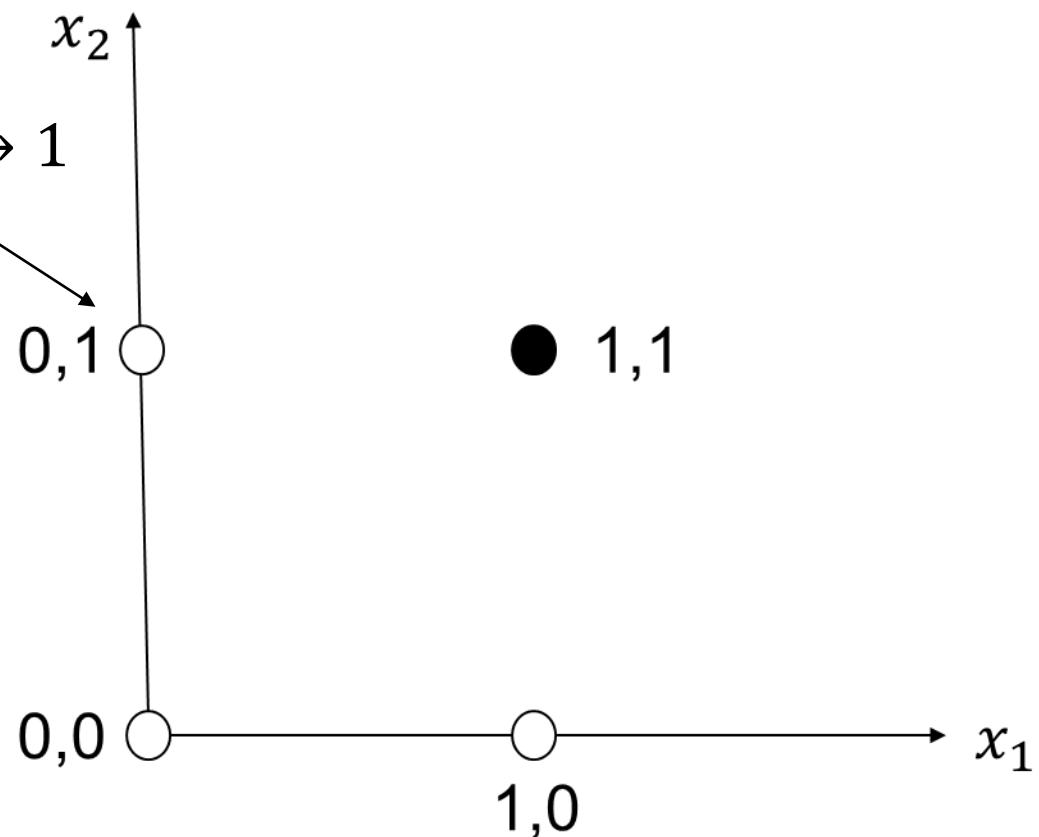


**Ejemplo** de un conjunto de prueba para el caso de una red neuronal operando **como clasificador** (ejemplo anterior):

Ejemplo:

$$(0.1,0.2), (0.1,0.9), (1.1,0.2) \rightarrow 0$$

$$(1.1,1.1), (0.9,0.8), (1.1,0.9) \rightarrow 1$$

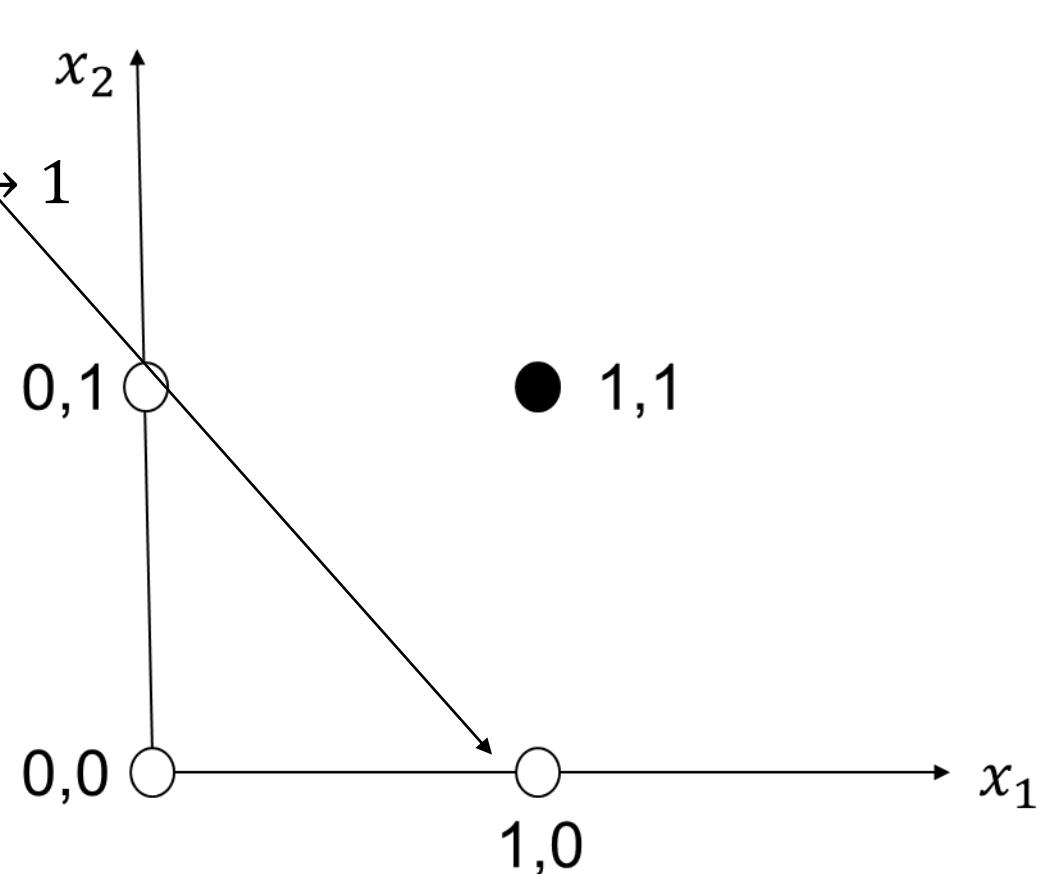


**Ejemplo** de un conjunto de prueba para el caso de una red neuronal operando **como clasificador** (ejemplo anterior):

Ejemplo:

$$(0.1,0.2), (0.1,0.9), (1.1,0.2) \rightarrow 0$$

$$(1.1,1.1), (0.9,0.8), (1.1,0.9) \rightarrow 1$$

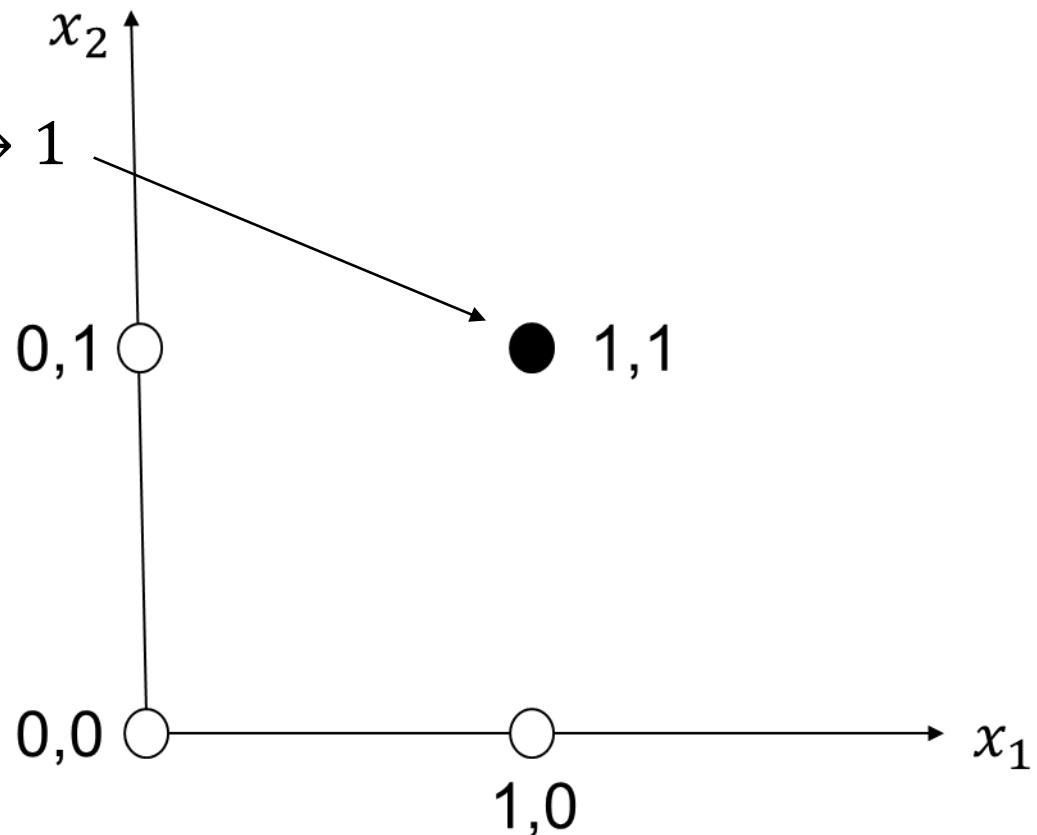


**Ejemplo** de un conjunto de prueba para el caso de una red neuronal operando **como clasificador** (ejemplo anterior):

Ejemplo:

$$(0.1,0.2), (0.1,0.9), (1.1,0.2) \rightarrow 0$$

$$(1.1,1.1), (0.9,0.8), (1.1,0.9) \rightarrow 1$$



## Ejemplos de conjuntos fundamentales para el caso de una memoria asociativa.

Ejemplo 1:  $p = 4, n = 2, m = 3$ : Los elementos de todos los patrones toman valores 0 o 1.

$$(0,0), (0,0,1); \quad (1,0), (0,1,1); \quad (1,0), (1,1,0); \quad (1,0), (1,1,1)$$

Ejemplo 2:  $p = 4, n = 2, m = 4$ : Los elementos de los patrones  $X^k = [x_1, x_2]^T$  toman valores en los reales mientras que los elementos de los patrones  $Y^k = [y_1, y_2, y_3, y_4]^T$  toman valores en el conjunto  $\{0,1\}$ :

$$(0.1, -0.5), (0,1,1,0); \quad (1.6, 10.5), (1,0,0,1);$$

$$(-0.5, -5), (0,0,0,0); \quad (10.7, 2.6), (1,1,1,1)$$

# **Fases de aprendizaje y prueba de una RNA y una MA:**

# **Caso de una RNA:**

## Fases de aprendizaje de una RNA:

Para que una RNA sea útil, ésta debe ser entrenada, en otras palabras sus **parámetros libres** deben ser ajustados.

Los parámetros libres (PL) de una RNA son los valores de red que el usuario puede ajustar mediante una regla de aprendizaje.

Una **regla de aprendizaje** (RA), en la forma de un algoritmo, es un **procedimiento** a través del cual el usuario puede ajustar los PL de la RNA que desea usar, más adelante, para resolver un problema dado.

**Época:** Recorrido completo del conjunto de entrenamiento para el entrenamiento de una RNA.

**Básicamente, las fases de aprendizaje o entrenamiento de una RNA son las siguientes:**

Dada una RNA y un conjunto de entrenamiento:

**Básicamente, las fases de aprendizaje o entrenamiento de una RNA son las siguientes:**

Dada una RNA y un conjunto de entrenamiento:

1 Poner los valores de los PL de la RNA en valores iniciales y poner a la variable auxiliar  $i = 1$ ;

**Básicamente, las fases de aprendizaje o entrenamiento de una RNA son las siguientes:**

Dada una RNA y un conjunto de entrenamiento:

- 1 Poner los valores de los PL de la RNA en valores iniciales y poner a la variable auxiliar  $i = 1$ ;
- 2 Presentar a la entrada de la RNA el par  $X^i, t^i$ ;

**Básicamente, las fases de aprendizaje o entrenamiento de una RNA son las siguientes:**

Dada una RNA y un conjunto de entrenamiento:

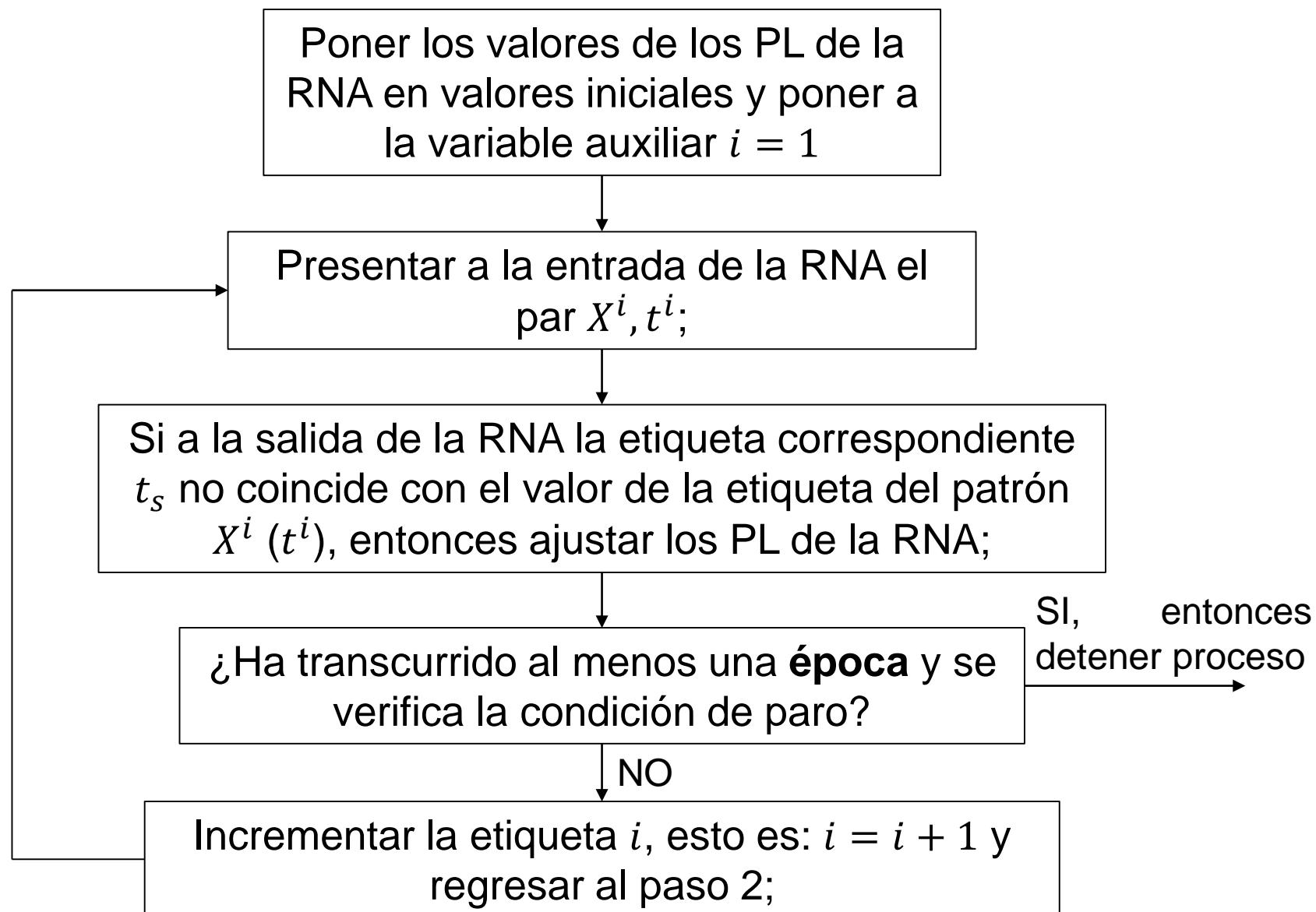
- 1 Poner los valores de los PL de la RNA en valores iniciales y poner a la variable auxiliar  $i = 1$ ;
- 2 Presentar a la entrada de la RNA el par  $X^i, t^i$ ;
- 3 Si a la salida de la RNA la etiqueta correspondiente  $t_s$  no coincide con el valor de la etiqueta del patrón  $X^i$  ( $t^i$ ), entonces ajustar los PL de la RNA;

**Básicamente, las fases de aprendizaje o entrenamiento de una RNA son las siguientes:**

Dada una RNA y un conjunto de entrenamiento:

- 1 Poner los valores de los PL de la RNA en valores iniciales y poner a la variable auxiliar  $i = 1$ ;
- 2 Presentar a la entrada de la RNA el par  $X^i, t^i$ ;
- 3 Si a la salida de la RNA la etiqueta correspondiente  $t_s$  no coincide con el valor de la etiqueta del patrón  $X^i$  ( $t^i$ ), entonces ajustar los PL de la RNA;
- 4 Al menos en una **época**, verificar la condición de paro. Si ésta se satisface detener el proceso de ajuste, sino Incrementar la etiqueta  $i$ , esto es:  $i = i + 1$  y regresar al paso 2;

# Diagrama de las fases de aprendizaje o entrenamiento de una RNA:



## Fases de prueba de una RNA:

Durante esta fase, se prueba el desempeño de la RNA entrenada. Para esto se aplican los siguientes pasos:

Poner la variable auxiliar  $j = 1$ ;

Poner la variable de eficiencia en  $E = 0$ ;

1 Presentar el patrón  $X^j, j = 1, 2 \dots, N$  a la entrada de la RNA;

2 Permitir que la RNA procese dicho patrón;

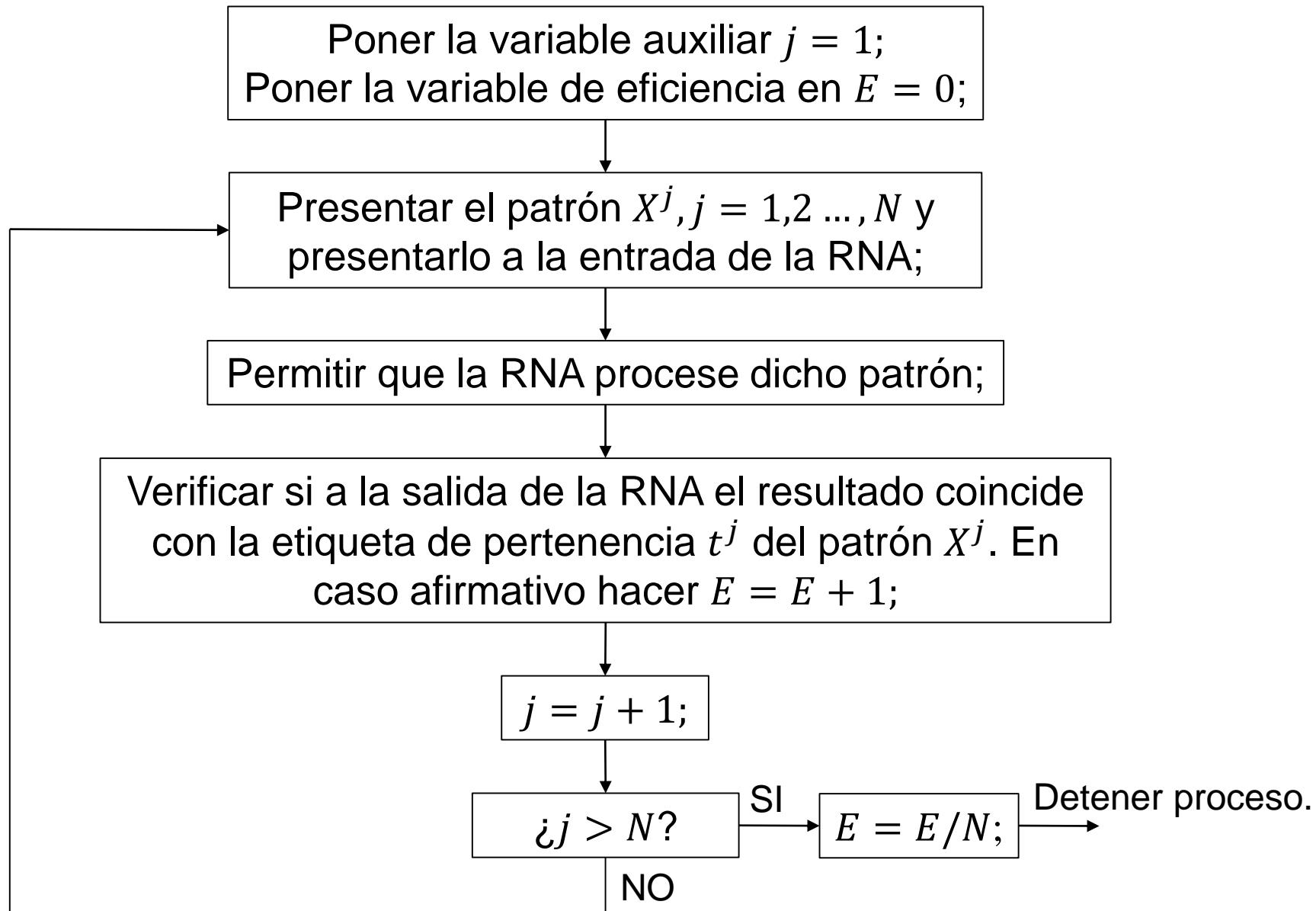
3 Verificar si a la salida de la RNA el resultado coincide con la etiqueta de pertenencia  $t^j$  del patrón  $X^j$ . En caso afirmativo hacer  $E = E + 1$ ;

4  $j = j + 1$ ;

5 Si  $j > N$ ,  $E = E/N$ , terminar la fase de prueba;

6 Si no, regresar al paso 1;

# Diagrama de las fases de prueba de una RNA:



# **Caso de una MA:**

## Fases de construcción de una MA:

Para que una MA sea útil, ésta debe ser construida. En este caso, dado un CF de asociaciones, se arma una MA (usualmente esta tiene la forma de una matriz).

Después los elementos de esta matriz son calculados sobre la base de los valores de los elementos de las diferentes asociaciones.

NOTA: El cálculo o ajuste de estos valores se realiza en una sola época de recorrido de los  $p$  elementos del CF.

**Básicamente, las fases de construcción de una MA son las siguientes:**

Dada un CF de asociaciones  $(X^k, Y^k), k = 1, 2, \dots, p$ :

**Básicamente, las fases de construcción de una MA son las siguientes:**

Dada un CF de asociaciones  $(X^k, Y^k), k = 1, 2, \dots, p$ :

1 Sobre la base de la dimensión de cualquiera de las  $k$  asociaciones, definir una matriz  $M$ ; poner a la variable auxiliar  $k = 1$ ; Sabiendo que las dimensiones para  $Y$  y  $X$  son,

respectivamente  $m$  y  $n$ , entonces:  $M = \begin{bmatrix} m_{11} & \cdots & m_{m1} \\ \vdots & \ddots & \vdots \\ m_{1n} & \cdots & m_{mn} \end{bmatrix}$ ;

**Básicamente, las fases de construcción de una MA son las siguientes:**

Dada un CF de asociaciones  $(X^k, Y^k), k = 1, 2, \dots, p$ :

1 Sobre la base de la dimensión de cualquiera de las  $k$  asociaciones, definir una matriz  $M$ ; poner a la variable auxiliar  $k = 1$ ; Sabiendo que las dimensiones para  $Y$  y  $X$  son,

respectivamente  $m$  y  $n$ , entonces:  $M = \begin{bmatrix} m_{11} & \cdots & m_{m1} \\ \vdots & \ddots & \vdots \\ m_{1n} & \cdots & m_{mn} \end{bmatrix}$ ;

2 Usar la asociación  $(X^k, Y^k)$  y realizar el ajuste de los pesos de  $M$  mediante la regla de ajuste correspondiente al modelo de MA seleccionado;

**Básicamente, las fases de construcción de una MA son las siguientes:**

Dada un CF de asociaciones  $(X^k, Y^k), k = 1, 2, \dots, p$ :

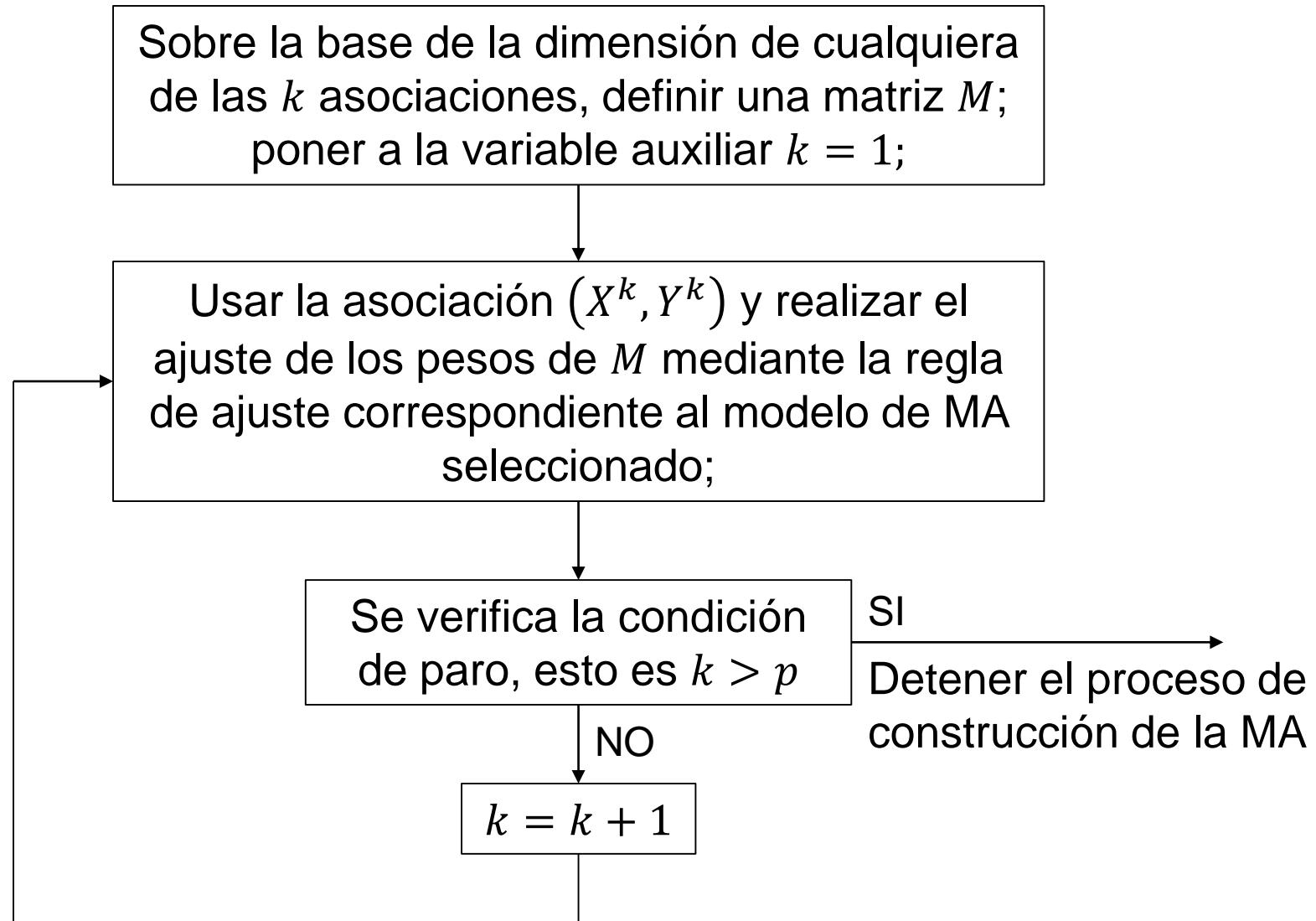
1 Sobre la base de la dimensión de cualquiera de las  $k$  asociaciones, definir una matriz  $M$ ; poner a la variable auxiliar  $k = 1$ ; Sabiendo que las dimensiones para  $Y$  y  $X$  son,

respectivamente  $m$  y  $n$ , entonces:  $M = \begin{bmatrix} m_{11} & \cdots & m_{m1} \\ \vdots & \ddots & \vdots \\ m_{1n} & \cdots & m_{mn} \end{bmatrix}$ ;

2 Usar la asociación  $(X^k, Y^k)$  y realizar el ajuste de los pesos de  $M$  mediante la regla de ajuste correspondiente al modelo de MA seleccionado;

3 Si  $k > p$  entonces detener el proceso de construcción de la MA, sino incrementar la etiqueta  $i$ , esto es:  $k = k + 1$  y regresar al paso 2;

# Diagrama de las fases de construcción de una MA:



## Fases de prueba de una MA:

Durante esta fase, se prueba el desempeño de la MA construida. Para esto se aplican los siguientes pasos:

Dadas  $q$  asociaciones de prueba  $(\tilde{X}^k, Y^k), k = 1, 2, \dots, q$ . Los  $\tilde{X}^k$  son versiones distorsionadas de los patrones  $X^k$ .

Poner las variable auxiliar  $k = 1$  y de eficiencia en  $E = 0$ ;

1 Presentar el patrón  $\tilde{X}^k, k = 1, 2, \dots, q$  como llave de entrada a la MA;

2 Permitir que la MA procese dicho patrón;

3 Verificar si a el patrón a salida de la MA,  $\hat{Y}$ , coincide con el deseado  $Y^k$ . En caso afirmativo hacer  $E = E + 1$ ;

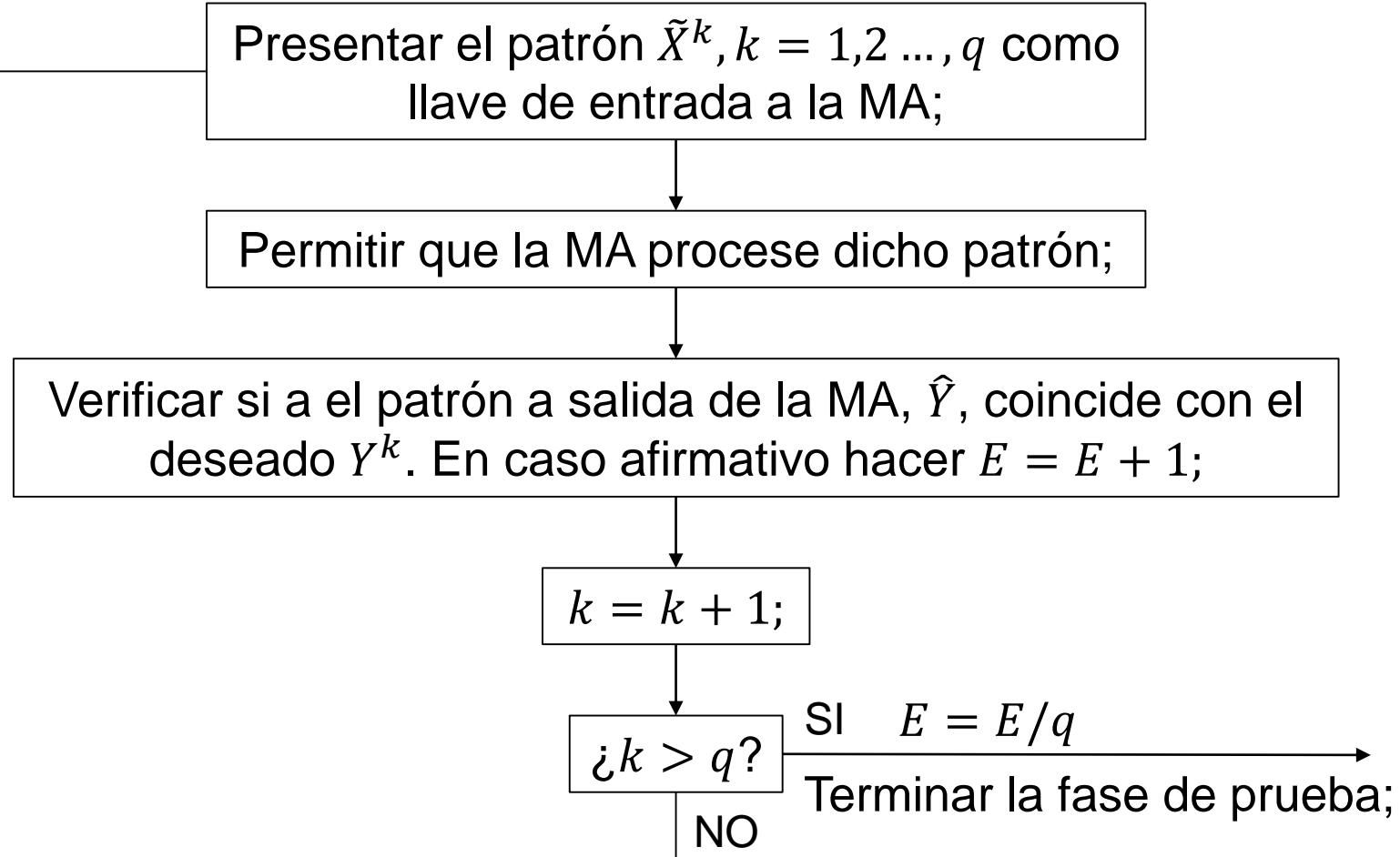
4  $k = k + 1$ ;

5 Si  $k > q$ ,  $E = E/q$ , terminar la fase de prueba;

6 Si no, regresar al paso 1;

# Diagrama de las fases de prueba de una MA:

Dadas  $q$  asociaciones de prueba  $(\tilde{X}^k, Y^k), k = 1, 2, \dots, q$ . Los  $\tilde{X}^k$  son versiones distorsionadas de los patrones  $X^k$ . Poner las variable auxiliar  $k = 1$  y de eficiencia en  $E = 0$ ;



**Otros aspectos  
básicos:**

En lo global, una RNA se asemeja al cerebro en dos aspectos:

1. El conocimiento se adquiere del ambiente a través de un **proceso de aprendizaje**.
2. Las fuerzas de interconexión (pesos sinápticos) se usan para **almacenar** el conocimiento adquirido.

Se supone que **la habilidad de procesamiento** de cada neurona **reside** en las excitaciones, inhibiciones y las fuerzas de conexión sinápticas con otras neuronas.

Debido a la importancia de la interconexión neuronal, a este sistema se le llama **conexionista**.

Al estudio de este enfoque general se le llama **conexionismo**.

**Ejemplo de un  
equivalente  
artificial de la  
neurona biológica:**

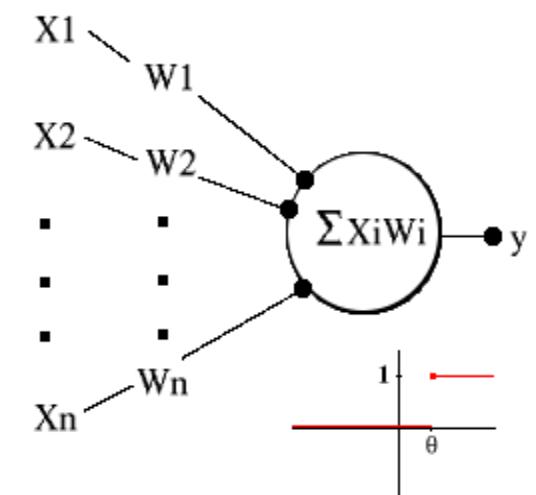
Un equivalente artificial de la neurona biológica es la unidad lógica de umbralado (del inglés TLU).

Las sinapsis se modelan como números (pesos). Cada peso multiplica a una entrada antes de ser enviada al equivalente del **cuerpo celular** de la neurona real.

En este lugar todos los productos son **sumados aritméticamente** para dar como resultado la **activación** del nodo.

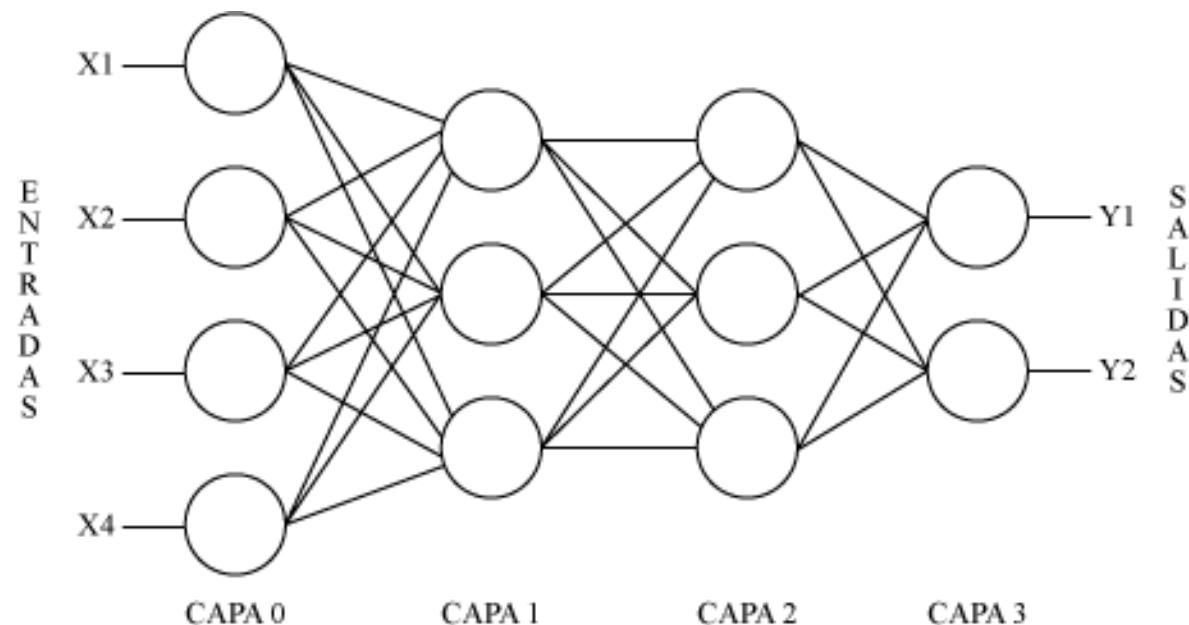
La activación es comparada con un umbral. Si la activación excede este umbral, la TLU produce un 1, sino produce un 0.

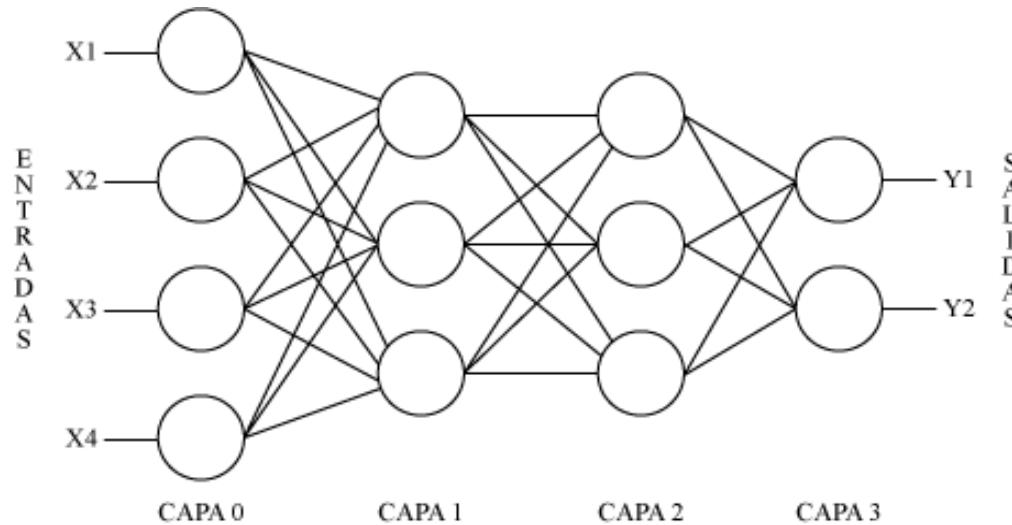
Este modelo fue propuesto por McCulloch-Pitts en 1943.



El término “red” será usado para designar a cualquier sistema de neuronas artificiales, desde un simple nodo hasta todo un arreglo de nodos interconectados de alguna manera entre ellos.

Una arquitectura tipo es el arreglo en capas, donde los nodos de una capa conectan hacia delante con las neuronas de la siguiente capa.





Este tipo de arreglo es, por supuesto, una de tantas maneras en que los nodos de una red se pueden interconectar.

En el caso de neuronas reales sus fuerzas sinápticas pueden ser modificadas de acuerdo a los estímulos de entrada según sea necesario.

En el caso de una neurona artificial esto se manifiesta en la modificación de los valores de los pesos.

En términos de procesamiento de información, nótese que no hay programas computacionales, se parte del hecho que **el “conocimiento” se almacena en los pesos**, que pueden ir cambiando a través de un proceso de adaptación sobre la base de estimulaciones obtenidas a partir de un conjunto de patrones.

Un mecanismo de adaptación de los pesos es a través de que se conoce como **aprendizaje supervisado**.

En este caso un patrón de entrada es presentado a la entrada del arreglo neuronal.

La respuesta de la red es comparada con la salida objetivo.

La diferencia entre la salida de la red y la salida objetivo determina cómo los pesos de la red son modificados.

Cada **receta** de modificación de los pesos constituye lo que se conoce en la literatura como una **regla de aprendizaje**.

Cuando los ajusten a los pesos han sido hechos, se presenta a la red otro nuevo patrón de entrada y los nuevos cambios a los pesos son hechos.

Esta secuencia se repite de manera iterada hasta que el comportamiento de la red **converge**.

A todo este proceso de ajuste de pesos se le conoce como **algoritmo de aprendizaje**.

¿Qué pasa si después de esto, a la red se le presenta una versión diferente de alguno de los patrones de entrada?

Si la red ha aprendido la **estructura del problema** en cuestión, la red debería clasificar de manera correcta el nuevo patrón de entrada.

A esta característica se le conoce como **capacidad de generalización de la red**.

Si la red no logra esto, no es más que una tabla tipo “lookup” para el conjunto de patrones de entrenamiento.

Una buena **generalización** es una de las propiedades clave de una red neuronal.

# ¿Porqué estudiar redes neuronales?

Las redes neuronales son frecuentemente usadas en el análisis estadístico y modelado de datos, como una alternativa a otros métodos existentes (regresión no lineal o técnicas de formación de cúmulos).

Ejemplos:

Análisis de imágenes.

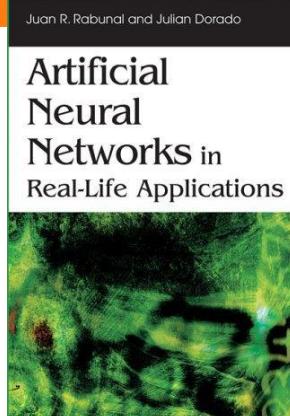
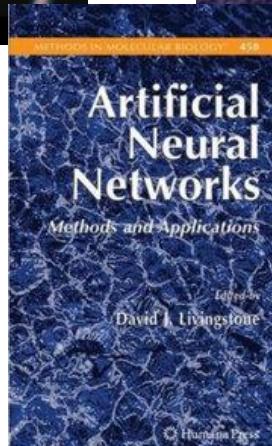
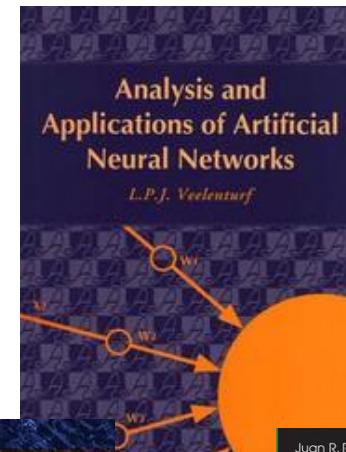
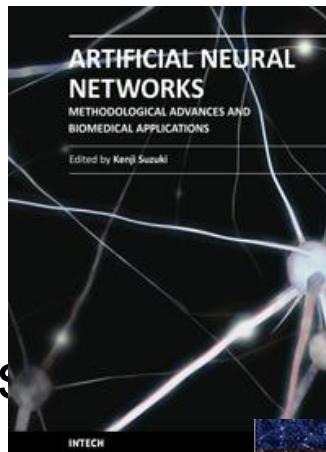
Reconocimiento de voz.

Reconocimiento de caracteres.

Diagnóstico médico.

Predicción de mercados.

Etcétera.



**Los estudiosos de  
las redes neuronales  
se pueden clasificar  
en un de cuatro  
grupos:**

**Los ingenieros y científicos computacionales.** Quieren construir máquinas inteligentes. Buscan implementar redes en hardware.

**Los ingenieros y científicos computacionales.** Quieren construir máquinas inteligentes. Buscan implementar redes en hardware.

**Los neuro-científicos y los sicólogos.** Se interesan en la redes como modelos computacionales de los cerebros animales desarrollados con el objeto de abstraer lo que se cree son las propiedades de los tejido nerviosos reales esenciales para el procesamiento de la información. Quieren saber cómo los cerebros animales trabajan.

**Los físicos y los matemáticos.** Se interesan en el estudio de las redes neuronales por el interés en los llamados sistemas dinámicos no lineales, la mecánica estadística y la teoría de autómatas. El rol de los matemáticos es descubrir y formalizar las propiedades de los modelos de redes. Estos quieren entender las propiedades fundamentales de las redes como sistemas complejos.

**Los físicos y los matemáticos.** Se interesan en el estudio de las redes neuronales por el interés en los llamados sistemas dinámicos no lineales, la mecánica estadística y la teoría de autómatas. El rol de los matemáticos es descubrir y formalizar las propiedades de los modelos de redes. Estos quieren entender las propiedades fundamentales de las redes como sistemas complejos.

**Los usuarios en general.** Se encuentran en las industrias. Usan las redes neuronales para modelar y analizar grandes bases de datos.

**Todos tienen en  
común el estudio de  
las redes neuronales.**

**Esto lleva a la  
cooperación.**

# **Generaciones de redes neuronales:**

## RNAs de primera generación:

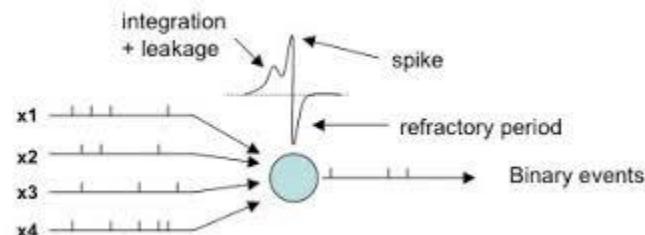
$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \rightarrow \{0,1\}$ . Realizan mapeos de vectores al conjunto  $\{0,1\}$ .

## RNAs de segunda generación:

$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \rightarrow \mathbb{R}$ . Realizan mapeos de vectores a subconjunto de  $\mathbb{R}$ .

## RNAs de tercera generación:

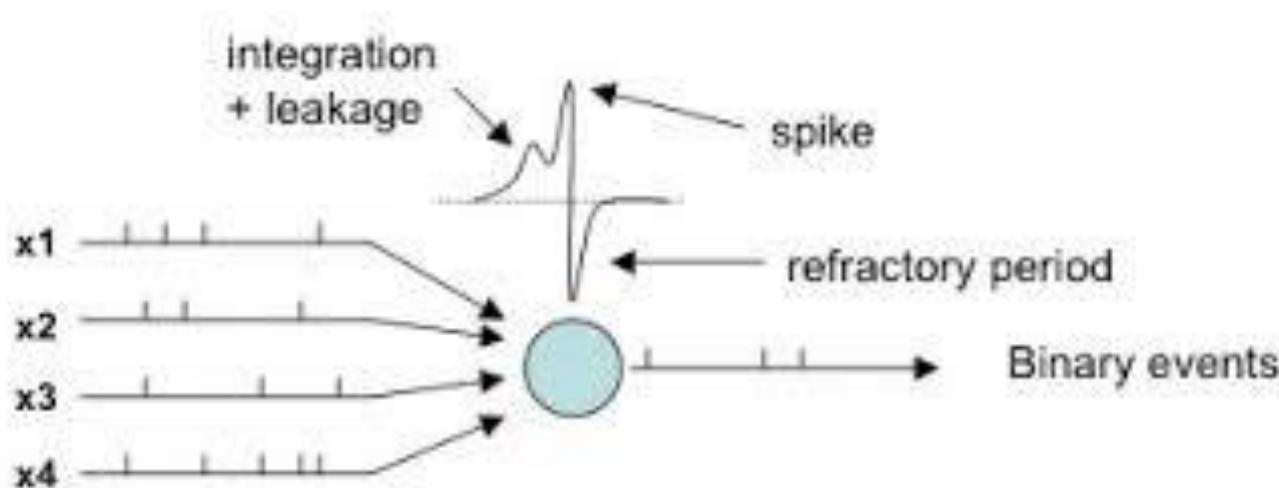
Aplican múltiples operaciones a conjuntos de patrones espacio-temporales.



**Las RNAs de primera generación:** Combinan dos operaciones: Un producto punto y un mapeo (función de activación).

**Las RNAs de segunda generación:** Combinan dos operaciones. Un producto punto y un mapeo (función de activación).

**Las RNAs de tercera generación:** Combinan múltiples operaciones.



**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  $i$

# **Resumen de la operación de una RN biológica:**

La función de las neuronas reales es muy compleja. Sin embargo, las propiedades esenciales de procesamiento pueden ser resumidas como sigue:

La función de las neuronas reales es muy compleja. Sin embargo, las propiedades esenciales de procesamiento pueden ser resumidas como sigue:

Una neurona recibe señales de entrada de muchas neuronas (**aferentes**).

La función de las neuronas reales es muy compleja. Sin embargo, las propiedades esenciales de procesamiento pueden ser resumidas como sigue:

Una neurona recibe señales de entrada de muchas neuronas (**aferentes**).

Cada una de estas señales de entrada es modulada por el mecanismo sináptico dando como resultado un PPS.

La función de las neuronas reales es muy compleja. Sin embargo, las propiedades esenciales de procesamiento pueden ser resumidas como sigue:

Una neurona recibe señales de entrada de muchas neuronas (**aferentes**).

Cada una de estas señales de entrada es modulada por el mecanismo sináptico dando como resultado un PPS.

Los PPS son integrados por los árboles dendríticos tanto en el espacio como en el tiempo (los PPS no decaen a cero de forma instantánea).

Los PPS son de excitación o de inhibición y su resultado integrado se manifiesta en un cambio en el potencial de la membrana en la colina axonal.

Los PPS son de excitación o de inhibición y su resultado integrado se manifiesta en un cambio en el potencial de la membrana en la colina axonal.

Este potencial de membrana puede activar o inhibir la neurona.

Los PPS son de excitación o de inhibición y su resultado integrado se manifiesta en un cambio en el potencial de la membrana en la colina axonal.

Este potencial de membrana puede activar o inhibir la neurona.

La dinámica de la membrana bajo estos cambios es muy compleja, pero puede ser descrita al suponer un **umbral de potencial de membrana** más allá del cual un potencial de acción (PA) es generado, y abajo no tal PA ocurre.

Los PPS son de excitación o de inhibición y su resultado integrado se manifiesta en un cambio en el potencial de la membrana en la colina axonal.

Este potencial de membrana puede activar o inhibir la neurona.

La dinámica de la membrana bajo estos cambios es muy compleja, pero puede ser descrita al suponer un **umbral de potencial de membrana** más allá del cual un potencial de acción (PA) es generado, y abajo no tal PA ocurre.

El tren de PAs constituye la salida de la neurona.

Los PPS son de excitación o de inhibición y su resultado integrado se manifiesta en un cambio en el potencial de la membrana en la colina axonal.

Este potencial de membrana puede activar o inhibir la neurona.

La dinámica de la membrana bajo estos cambios es muy compleja, pero puede ser descrita al suponer un **umbral de potencial de membrana** más allá del cual un potencial de acción (PA) es generado, y abajo no tal PA ocurre.

El tren de PAs constituye la salida de la neurona.

Estos viajan a lo largo del axón hasta otras sinapsis.

La información es codificada de muchas maneras en las neuronas, pero un método comúnmente adoptado es la frecuencia o tasa de producción de los PAs.

La información es codificada de muchas maneras en las neuronas, pero un método comúnmente adoptado es la frecuencia o tasa de producción de los PAs.

La integración de señales en el espacio puede ser modelada mediante una suma ponderada de las entradas.

La información es codificada de muchas maneras en las neuronas, pero un método comúnmente adoptado es la frecuencia o tasa de producción de los PAs.

La integración de señales en el espacio puede ser modelada mediante una suma ponderada de las entradas.

La acción sináptica se supone equivalente a la multiplicación por un peso.

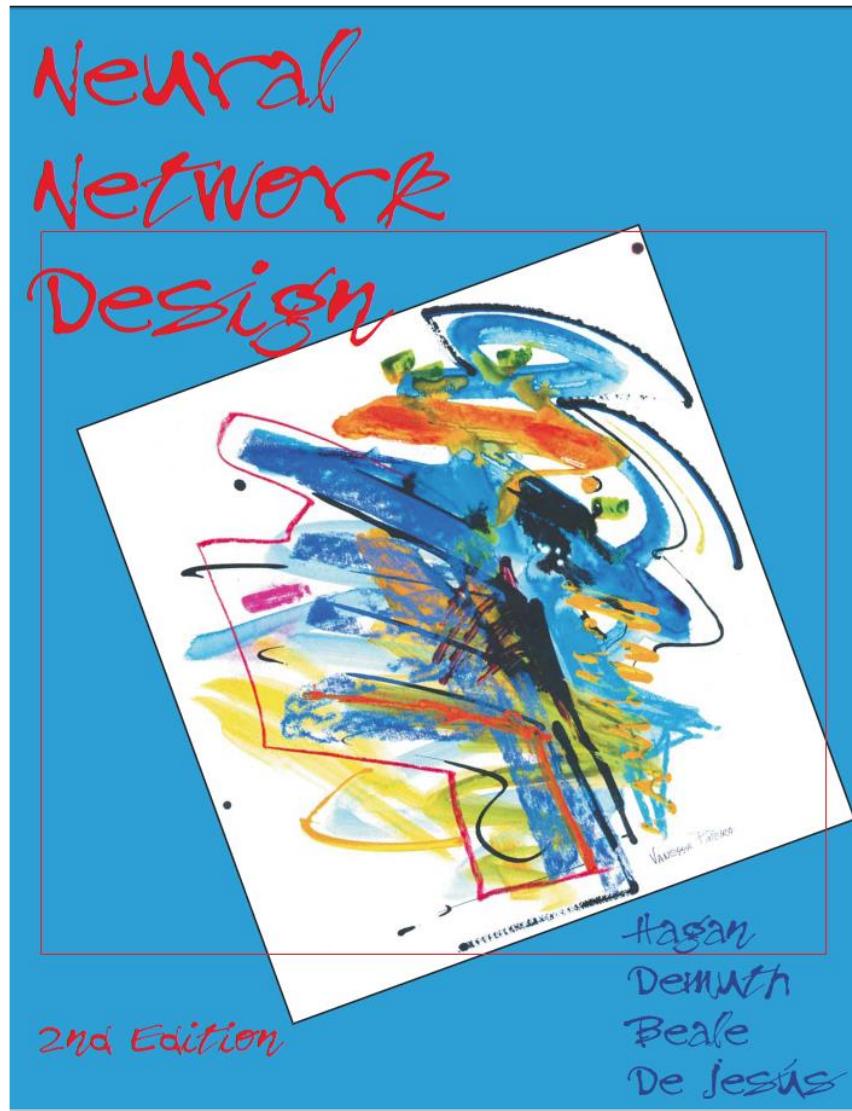
La información es codificada de muchas maneras en las neuronas, pero un método comúnmente adoptado es la frecuencia o tasa de producción de los PAs.

La integración de señales en el espacio puede ser modelada mediante una suma ponderada de las entradas.

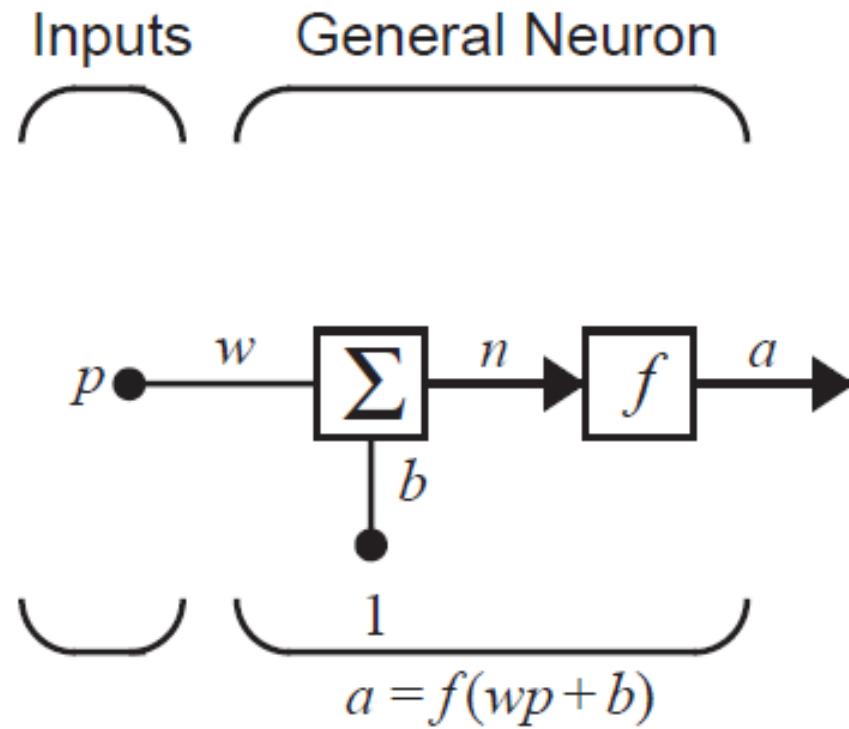
La acción sináptica se supone equivalente a la multiplicación por un peso.

La tasa de disparo puede ser representado por una función semilineal que permite una salida continua.

# Modelo sencillo de neurona:



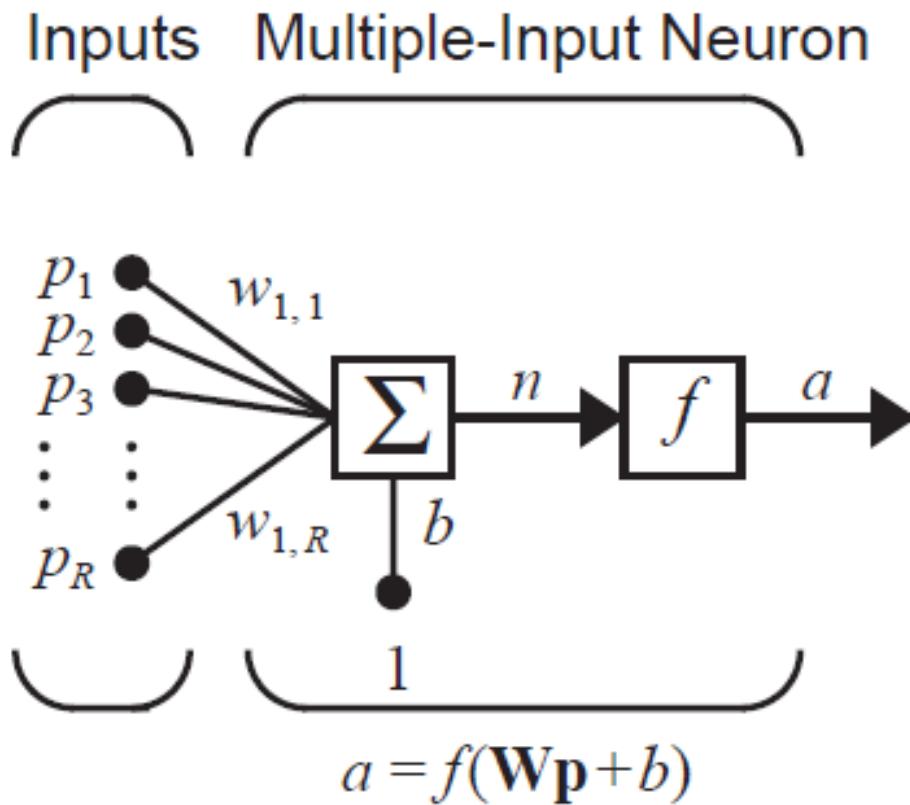
## Neurona simple:



Ejemplo:  $w = 3, p = 2$  y  $b = -1$ .

$$a = f(3(2) - 1.5) = f(4.5).$$

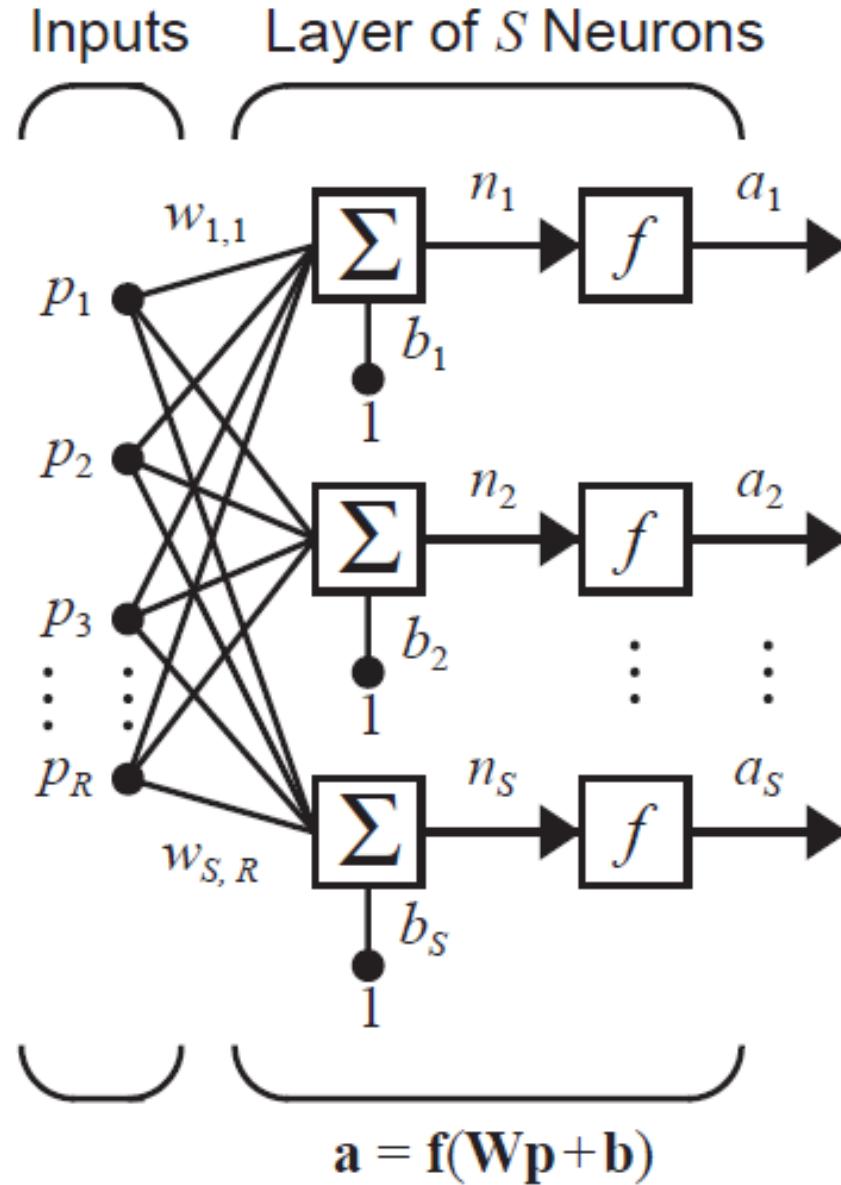
## Neurona con varias entradas:



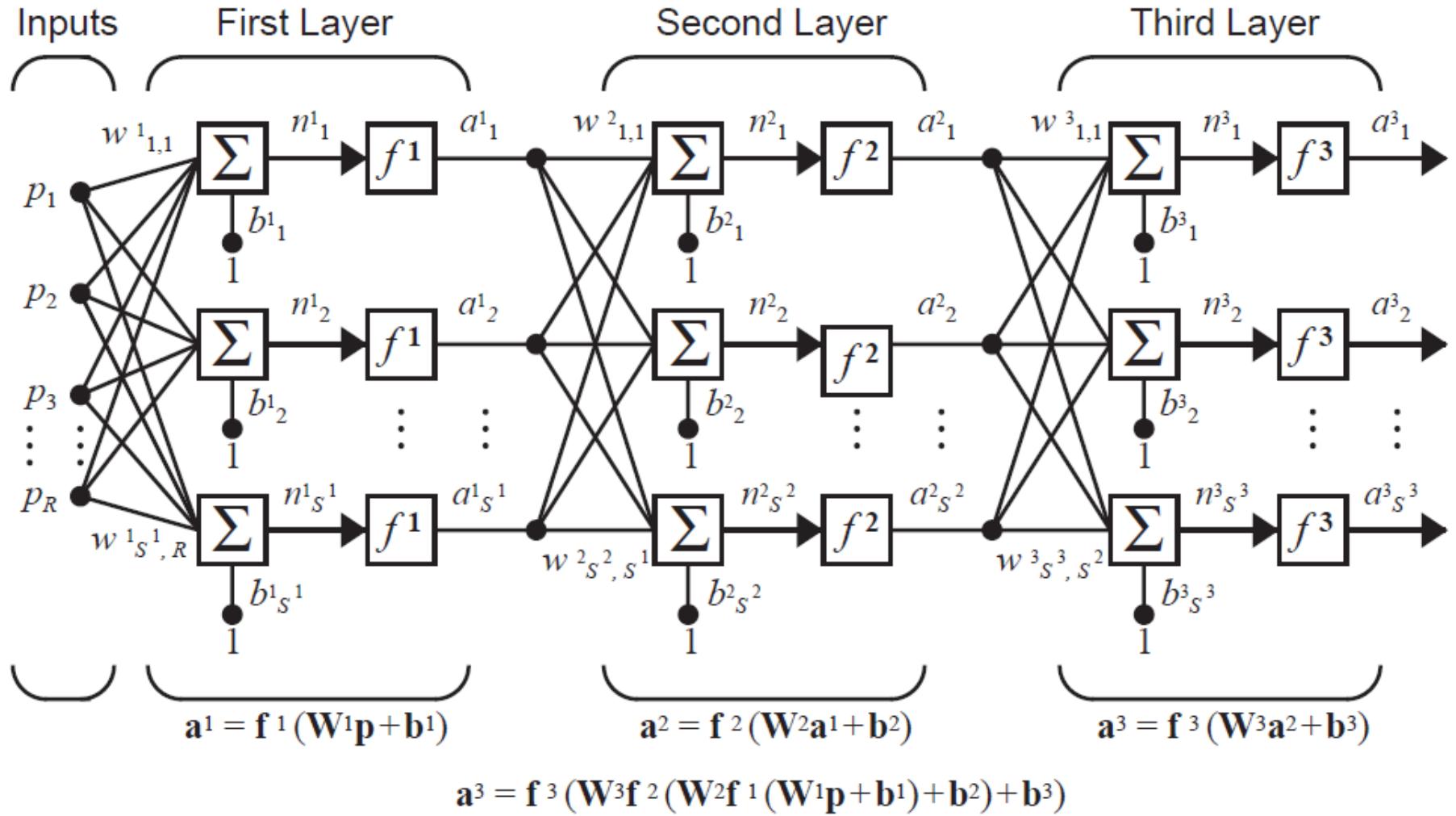
¿Cómo sería una red de neuronas en una capa?

# **Arquitecturas de RNAs:**

# Una capa de neuronas:

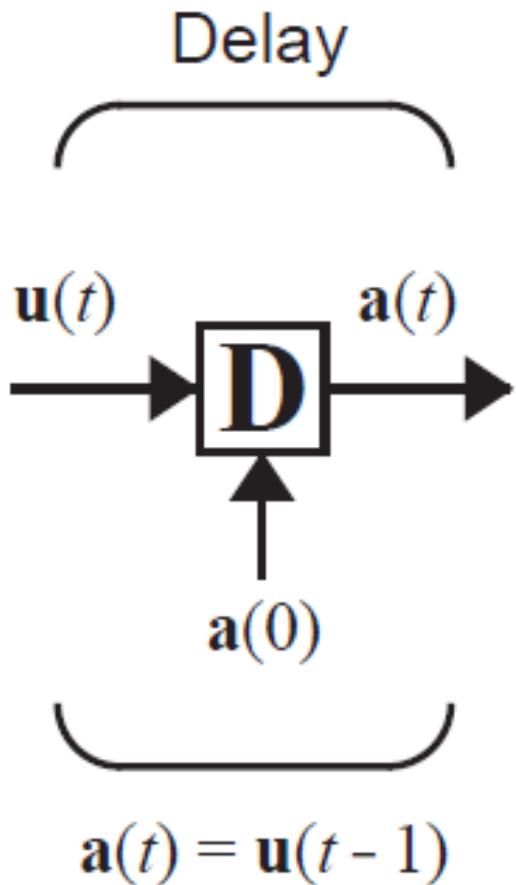


# Red neuronal en capa:

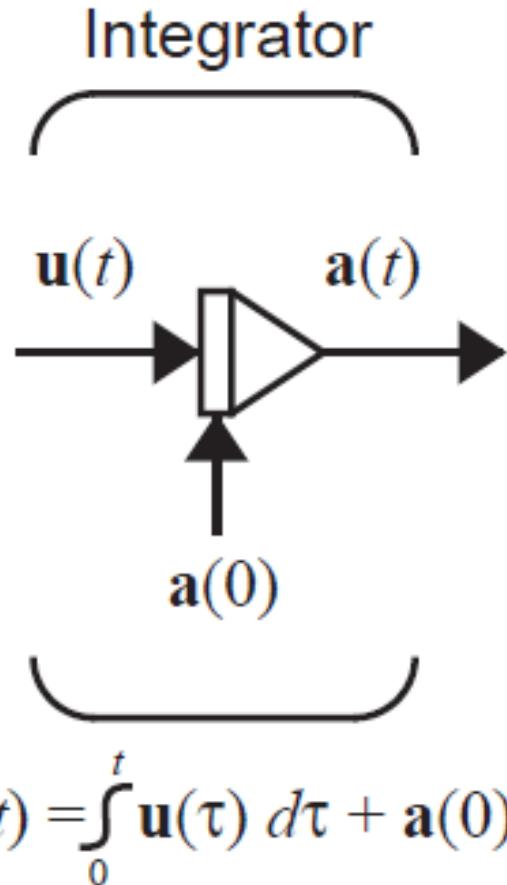


# Redes recurrentes:

Bloque retraso:



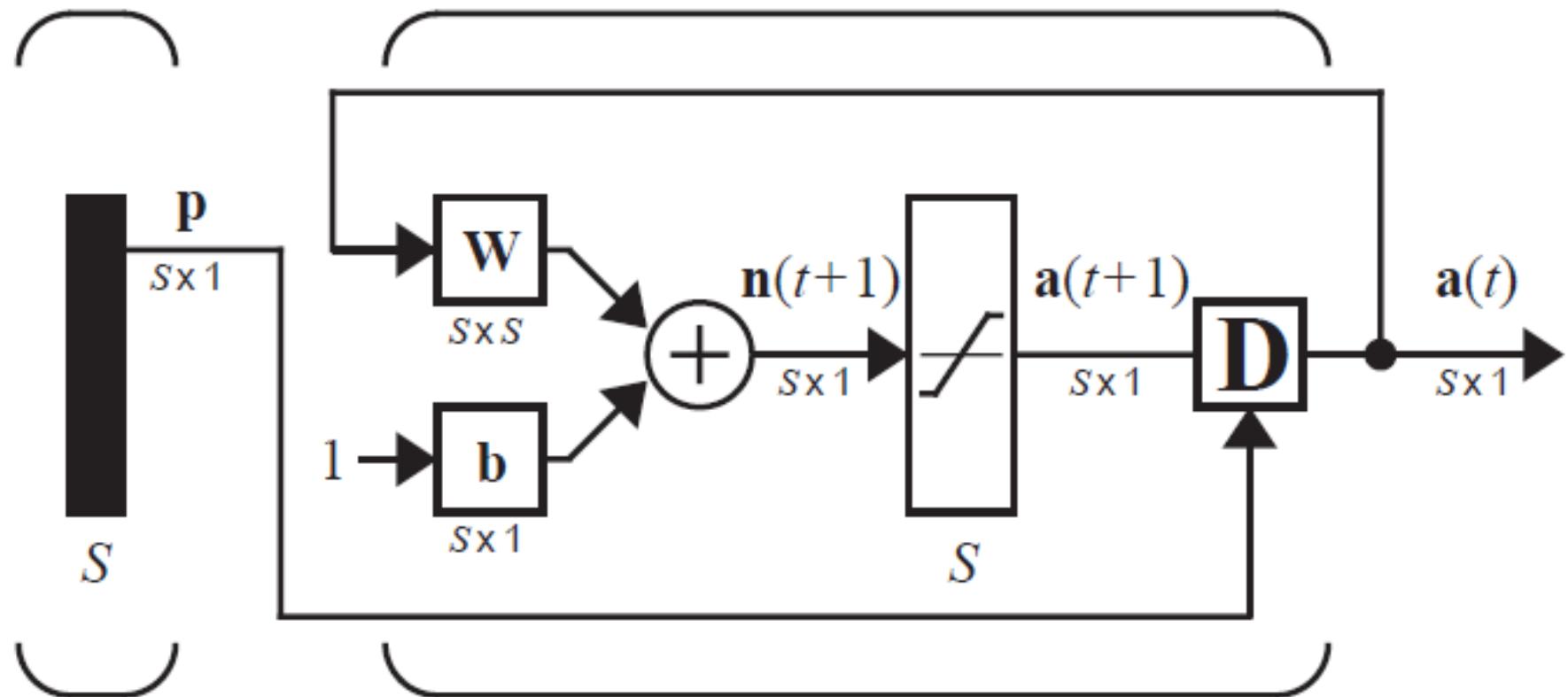
Bloque integrador:



# Redes recurrentes:

Initial  
Condition

Recurrent Layer



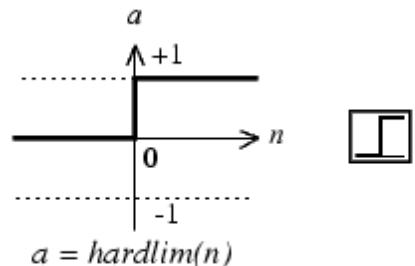
$$a(0) = p \quad a(t+1) = \text{satlins}(W a(t) + b)$$

**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.

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

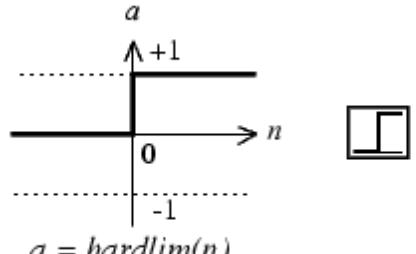
# Funciones de activación típicas:



Límite duro:

Hard-Limit Transfer Function

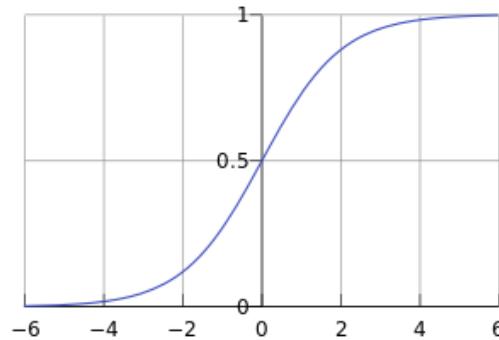
# Funciones de activación típicas:



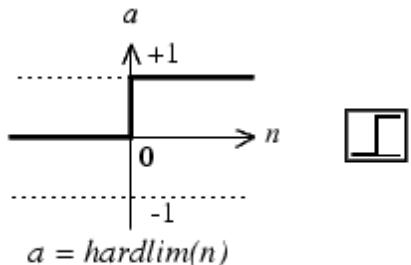
Límite duro:

Hard-Limit Transfer Function

Sigmoidea o semilineal:  $S(t) = \frac{1}{1 + e^{-t}}$ .



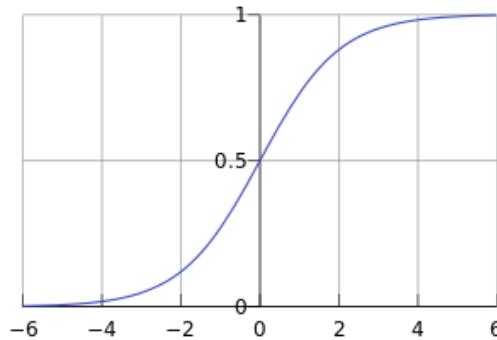
# Funciones de activación típicas:



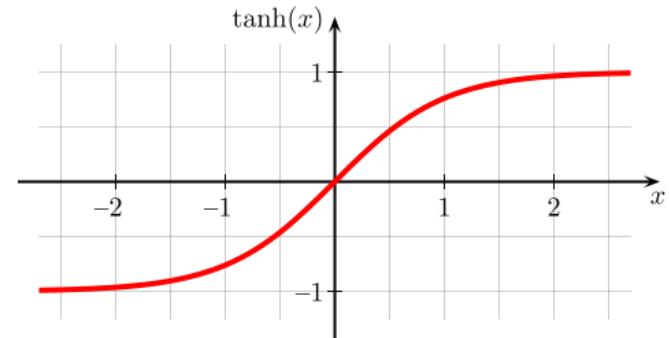
Límite duro:

Hard-Limit Transfer Function

Sigmoidea o semilineal:  $S(t) = \frac{1}{1 + e^{-t}}$ .

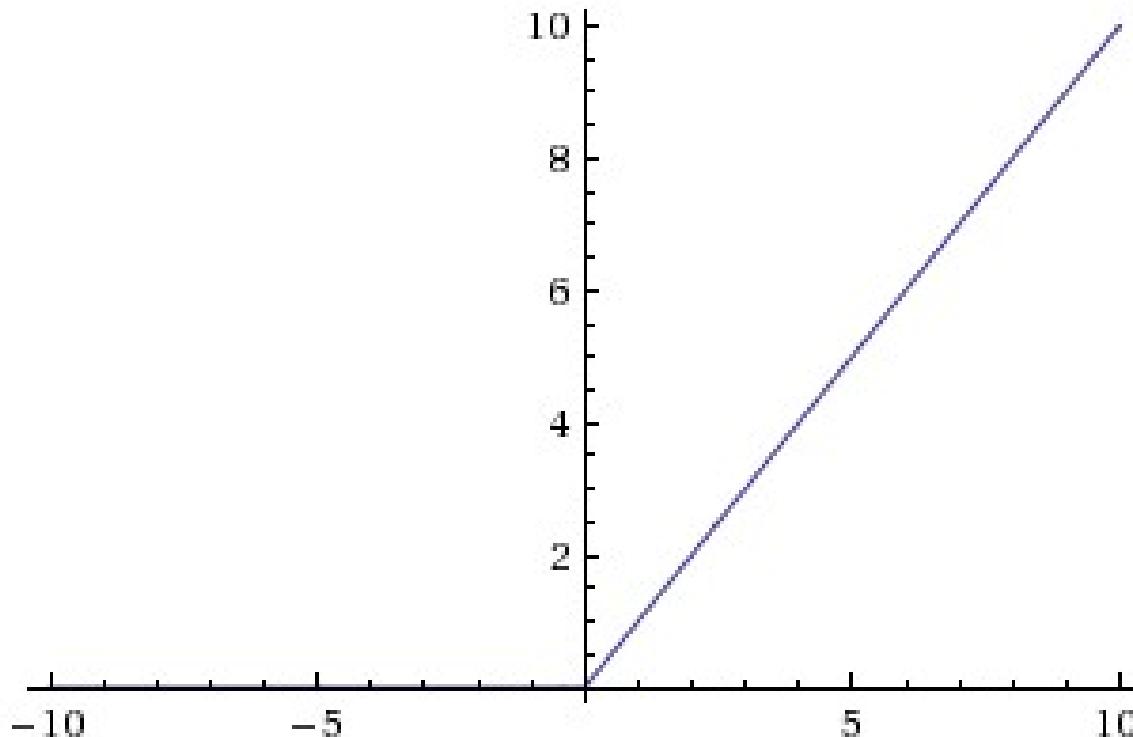


Tangente hiperbólica:  $\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$



Función ReLU (Unidad lineal restringida):  $f(x) = \max(0, x)$

Se ha convertido en la selección para el entrenamiento de RNAs para muchas tareas (visión por computadora) \*

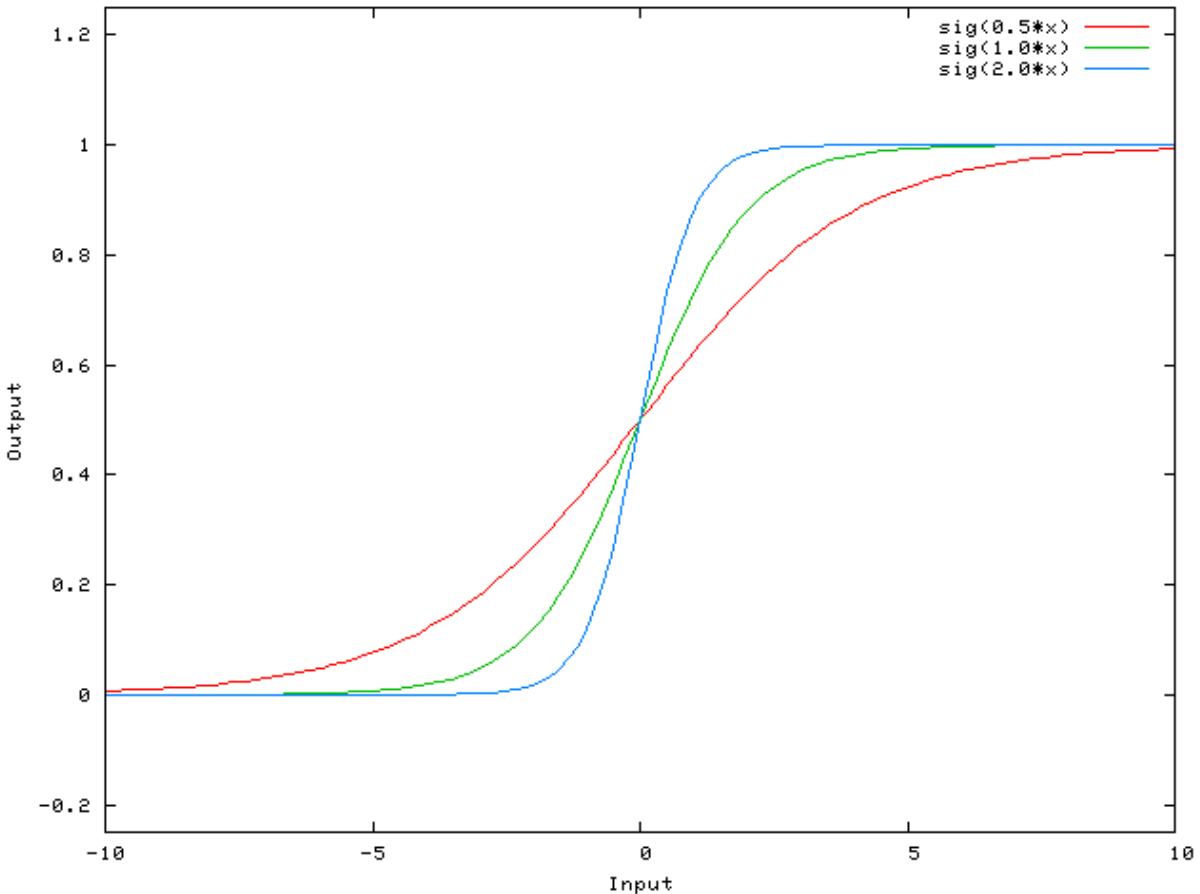


\* V. Nair and G. E. Hinton (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. Proceedings of the 27th International Conference on Machine Learning (ICML-10), Pp. 807-814.

# Ajuste de un peso multiplicativo: ( $\text{sig}(w \times x)$ )



Permite alterar la  
“**inclinación**” de la  
función sigmoide



**Favorece el aprendizaje de patrones.**

## Adición de un **bias** o **desplazamiento**:

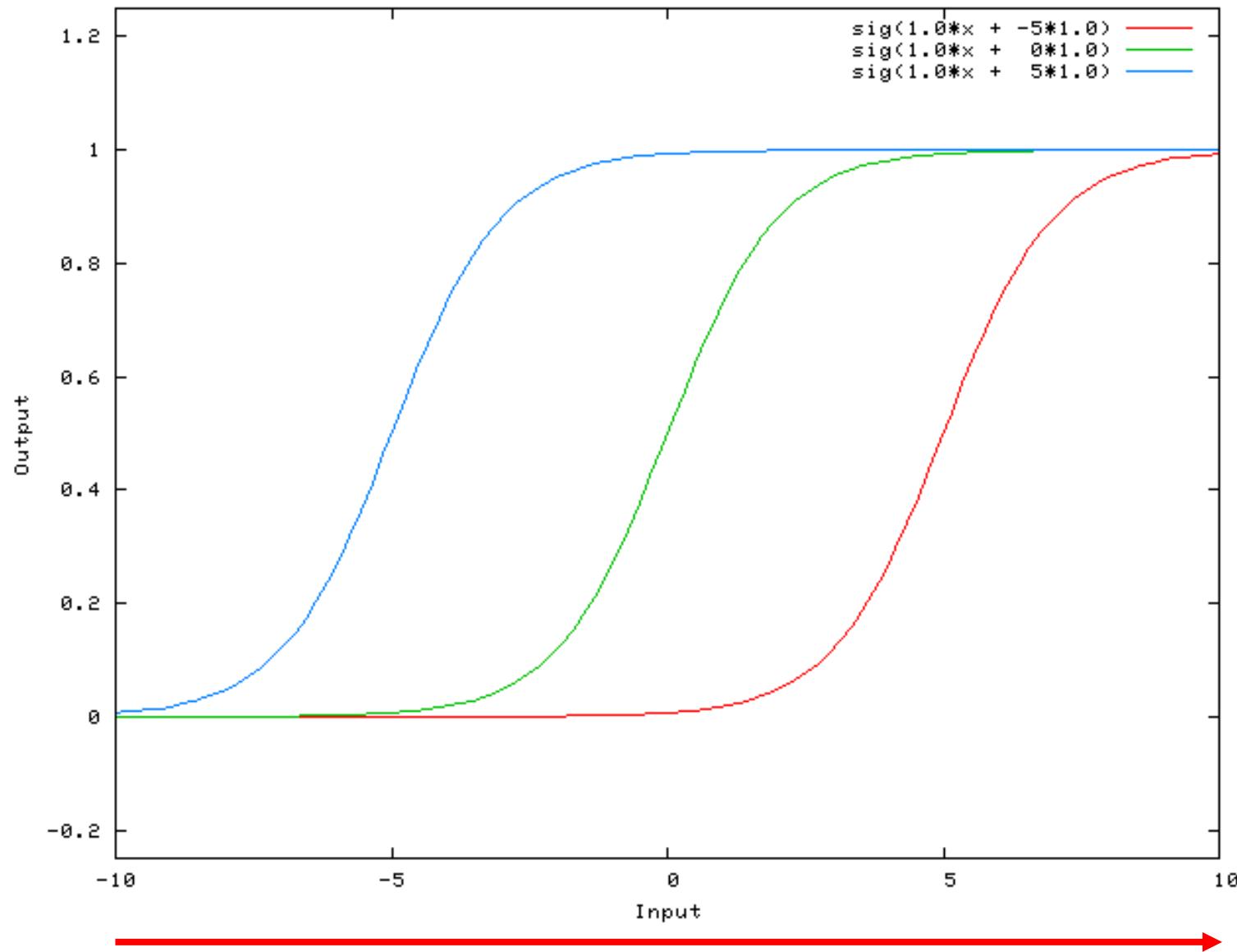
¿Cómo se lograría que la neurona emitiera un 0 cuando  $x$  toma un valor diferente de 0, por ejemplo 3?

Adición de un **bias** o **desplazamiento**:

¿Cómo se lograría que la neurona emitiera un 0 cuando  $x$  toma un valor diferente de 0, por ejemplo 3?

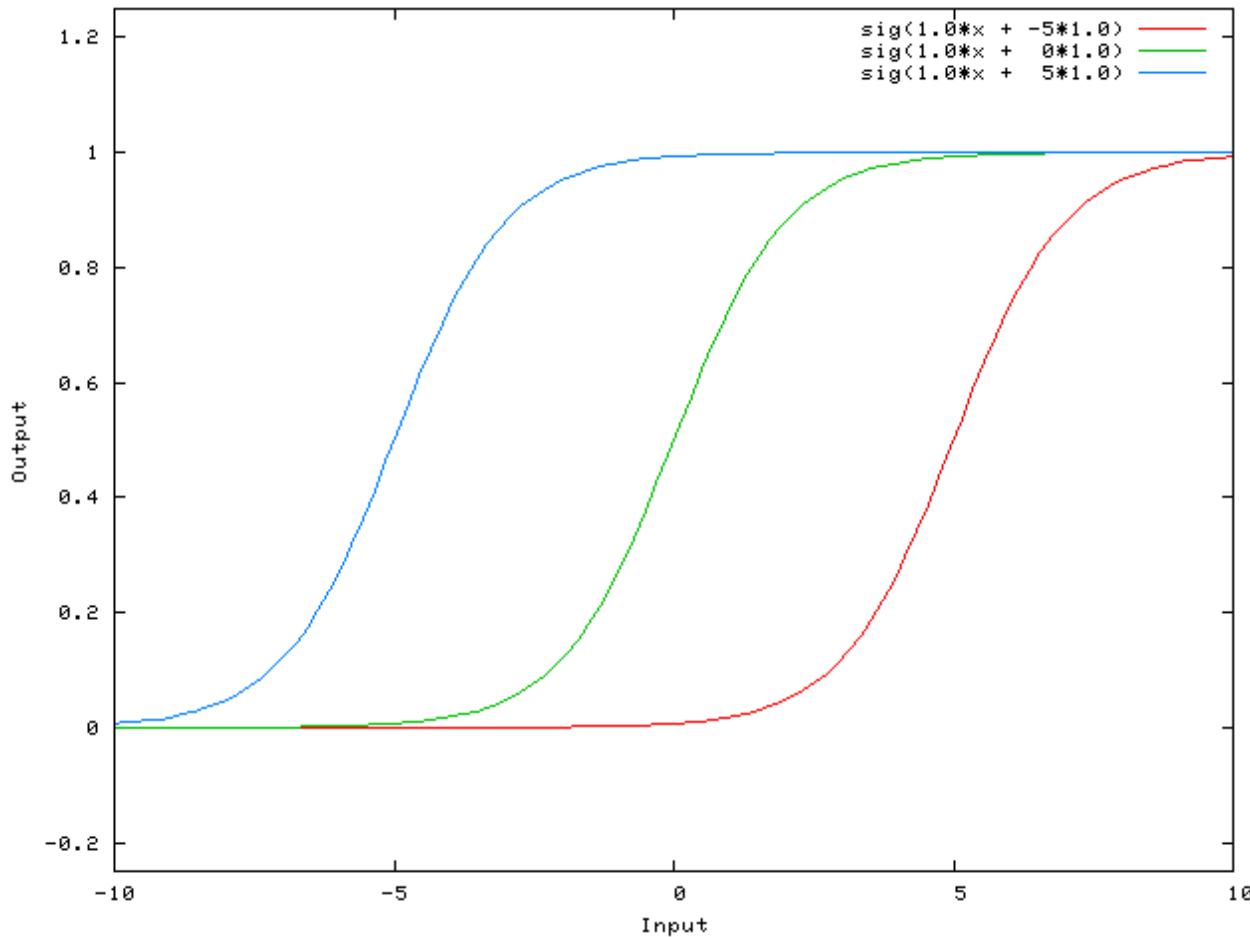
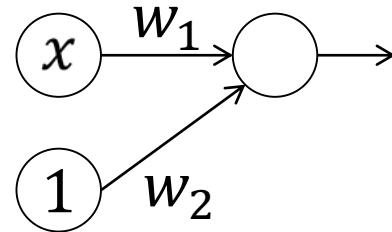
El cambio de inclinación no permite esto.

Para esto se requiere un **desplazamiento** de la curva completa a la derecha:



Este es el propósito de un **bias**.

Ahora la RNA aparece como:



## Resumen:

Las RNAs de primera y segunda generación utilizan dos operaciones concatenadas para realizar su función:

Un **producto punto** seguido de la aplicación de una función de activación (FA).

La FA agrega **no-linealidades** a la neurona, lo cual, a su vez, permite a la neurona aprender patrones.

La adición de un peso **multiplicativo** permite variar la inclinación de la FA.

La adición de un **bias** permite a la neurona emitir valores deseados a la salida.

# Resumen de funciones de activación:

Name	Input/Output Relation	Icon	MATLAB Function
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlim
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$		hardlims
Linear	$a = n$		purelin
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$		satlin
Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$		satlins

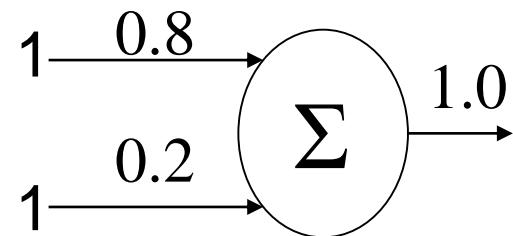
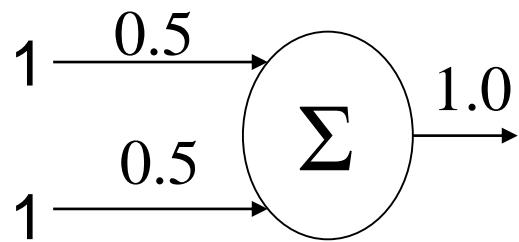
# Resumen de funciones de activación:

Log-Sigmoid	$a = \frac{1}{1 + e^{-n}}$		logsig
Hyperbolic Tangent Sigmoid	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$		tansig
Positive Linear	$\begin{aligned} a &= 0 && n < 0 \\ a &= n && 0 \leq n \end{aligned}$		poslin
Competitive	$\begin{aligned} a &= 1 && \text{neuron with max } n \\ a &= 0 && \text{all other neurons} \end{aligned}$		compet

# **Métodos para el cálculo del error:**

Una RNA emitirá valores **en función** de los pesos seleccionados.

Ejemplo con una neurona con dos entradas:



**Entrenamiento** es el proceso a través del cual los pesos iniciales **son ajustados** de forma que la RNA emita **resultados deseables**.

Se requiere de una manera de medir la efectividad en que una RNA trabaja.

A esta medida se le conoce como error de cálculo.

En entrenamiento supervisado la salida de una RNA es comparada con la salida ideal respecto a la base de datos.

La **diferencia** entre la **salida actual** y la **ideal** determina el **error** de la RNA.

Ocurre a dos niveles:


$$\text{Error\_RNA} = \text{Salida ideal} - \text{Salida actual}$$

El **error local** es la diferencia entre la salida actual de una neurona y la salida ideal esperada.

Estos errores locales se suman para obtener el **error global** de la RNA.

Este **error global** permite medir que tan bien la RNA se desempeña para una BD dada.

## Ejemplos de funciones de error:

- Suma de los cuadrados de los errores (ESS):
- El error cuadrático medio (MSE):
- La raíz cuadrada promedio (RMS):

## Función error:

El error local viene de la función de error (FE).

La FE recibe como entrada las salidas actual e ideal para una neurona.

Produce un número que representa el error de dicha neurona.

Todo método de entrenamiento busca minimizar este error.

## Función de error lineal:

$$E = (i - a)$$

$i$  salida ideal.

$a$  salida actual.

Ejemplos:

La neurona produce una salida de 0.9 cuando se requiere que produjera 0.8; entonces el error sería de -0.1.

La neurona produce una salida de 1.2 cuando se requiere que produjera 1.4; entonces el error sería de 0.2.

## Cálculo del error global:

El error global se mide en términos de los errores locales.

El MSE es el más usado:

$$MSE = \frac{1}{n} \sum_{i=1}^n E^2 = \frac{1}{n} \sum_{i=1}^n (i - a)^2$$

Este error es similar al promedio clásico, sólo que el error local es elevado al cuadrado.

Esto permite negar el efecto de errores locales positivos y negativos.

## Otros métodos para el cálculo del error global:

La suma de los errores cuadrados (ESS):

$$ESS = \frac{1}{2} \sum_{\nu} E^2 = \frac{1}{2} \sum_{\nu} (i - a)^2$$

No es porcentaje; es simplemente **un número que más grande** dependiendo de que tan severo es el error.

## Otros métodos para el cálculo del error global:

La raíz del error cuadrático medio (MSE):

$$MSE = \sqrt{\frac{1}{n} \sum_{i=1}^n E^2} = \sqrt{\frac{1}{n} \sum_{i=1}^n (i - a)^2}$$

Este error siempre será mayor que el MSE.

Ejemplo:

Actual=[-0.36,0.07,0.55,0.05,-0.37,0.34,-0.72,-0.10.-0.41,-0.32]

Ideal =[ -0.37,0.06,0.51,0.06,-0.36,0.35,-0.67,-0.09.-0.43,-0.33]

ESS = 0.00312453.

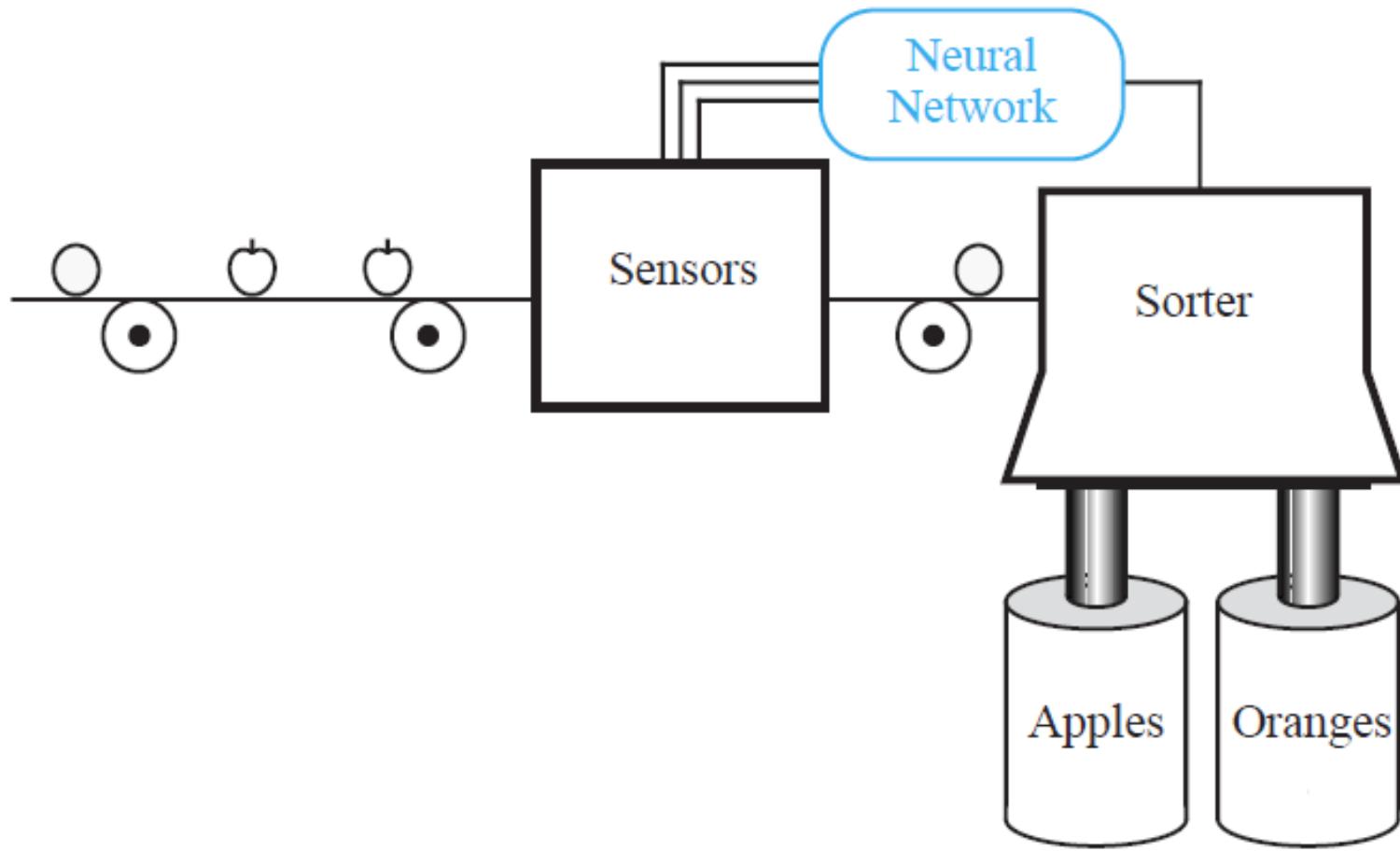
MSE = 0.062491%.

RMS = 2.499801%.

NOTA: El RMS no se adapta bien al entrenamiento de RNAS,  
se usa más en análisis de señales.

**Ejemplo de  
problema que  
puede ser resuelto  
mediante una  
RNA:**

# Planteamiento del problema:



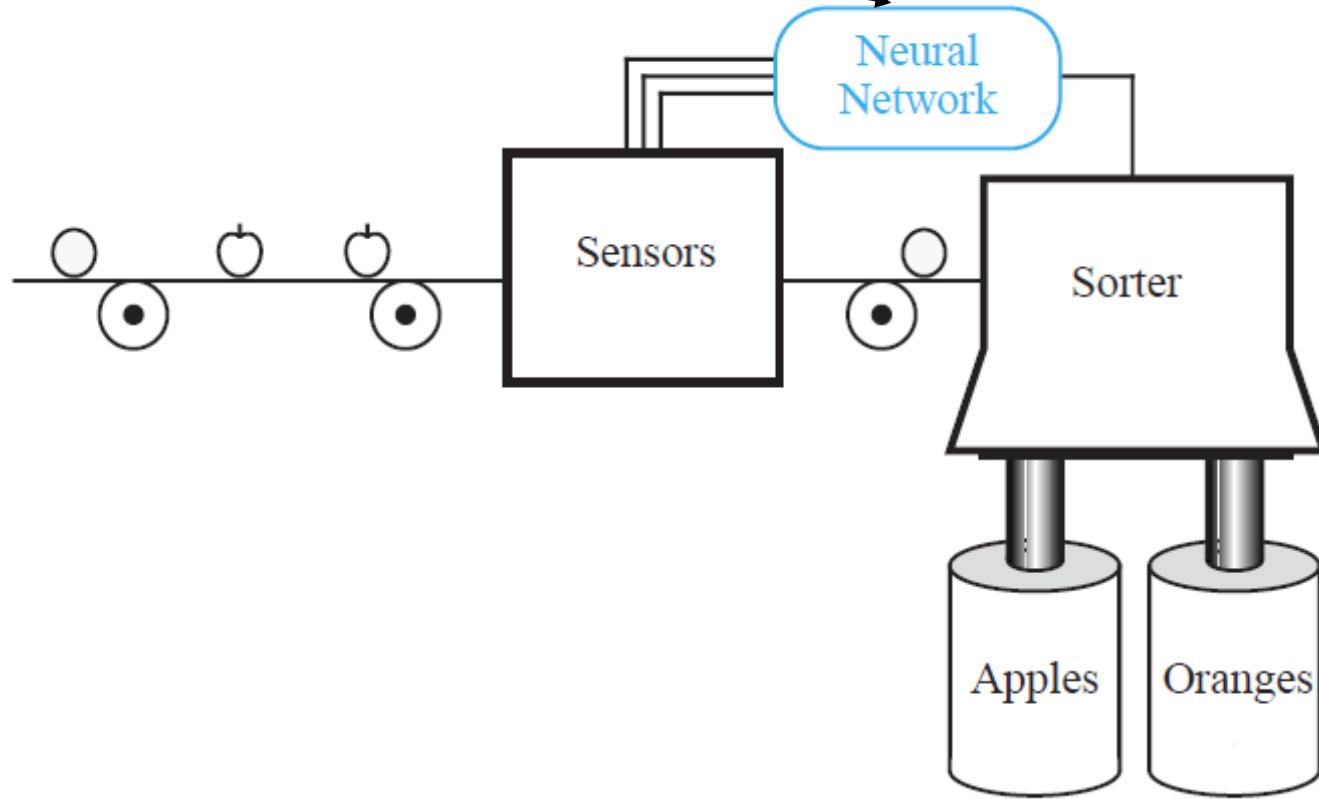
Rasgos medidos: forma, textura y peso.

El sensor de forma emitirá un 1 si la fruta es redonda y -1 si es elíptica.

El sensor de textura emitirá un 1 si superficie de fruta suave y -1 si es rugosa.

El sensor de peso emitirá un 1 si la fruta pesa más de medio kilo y -1 si pesa menos.

Las 3 entradas van a la RN:



Objetivo: Decidir que fruta va por la banda y clasificarla en su categoría.

Caso simple: Manzana o naranja.

Vector descriptor:

$$p = \begin{bmatrix} forma \\ textura \\ peso \end{bmatrix}$$

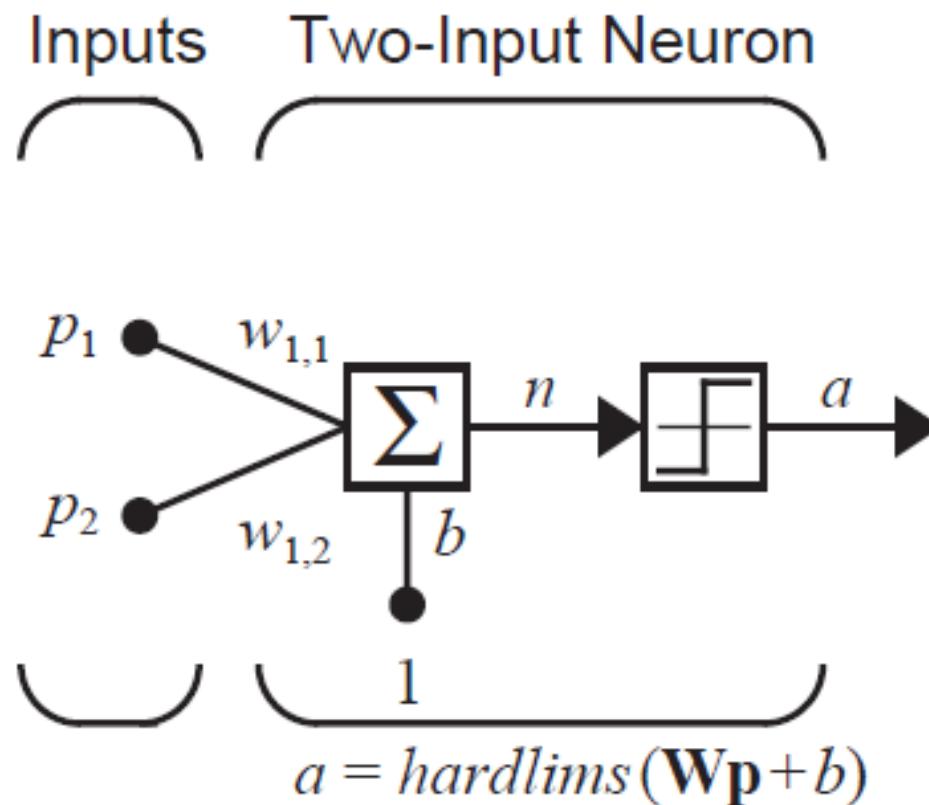
Prototipo para naranja:

$$p = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

Prototipo para manzana:

$$p = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

## Solución mediante perceptrón:

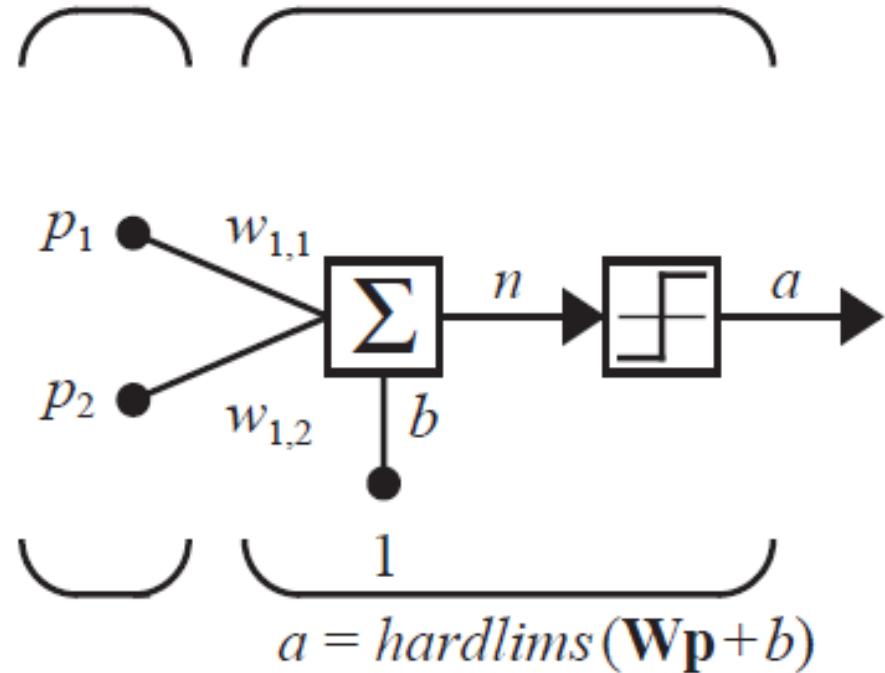


Name	Input/Output Relation	Icon	MATLAB Function
Hard Limit	$\begin{aligned} a &= 0 & n < 0 \\ a &= 1 & n \geq 0 \end{aligned}$		hardlim

## Solución mediante perceptrón: Inputs Two-Input Neuron

Suponer que:

$$w_{1,1} = -1 \text{ y } w_{1,2} = 1.$$

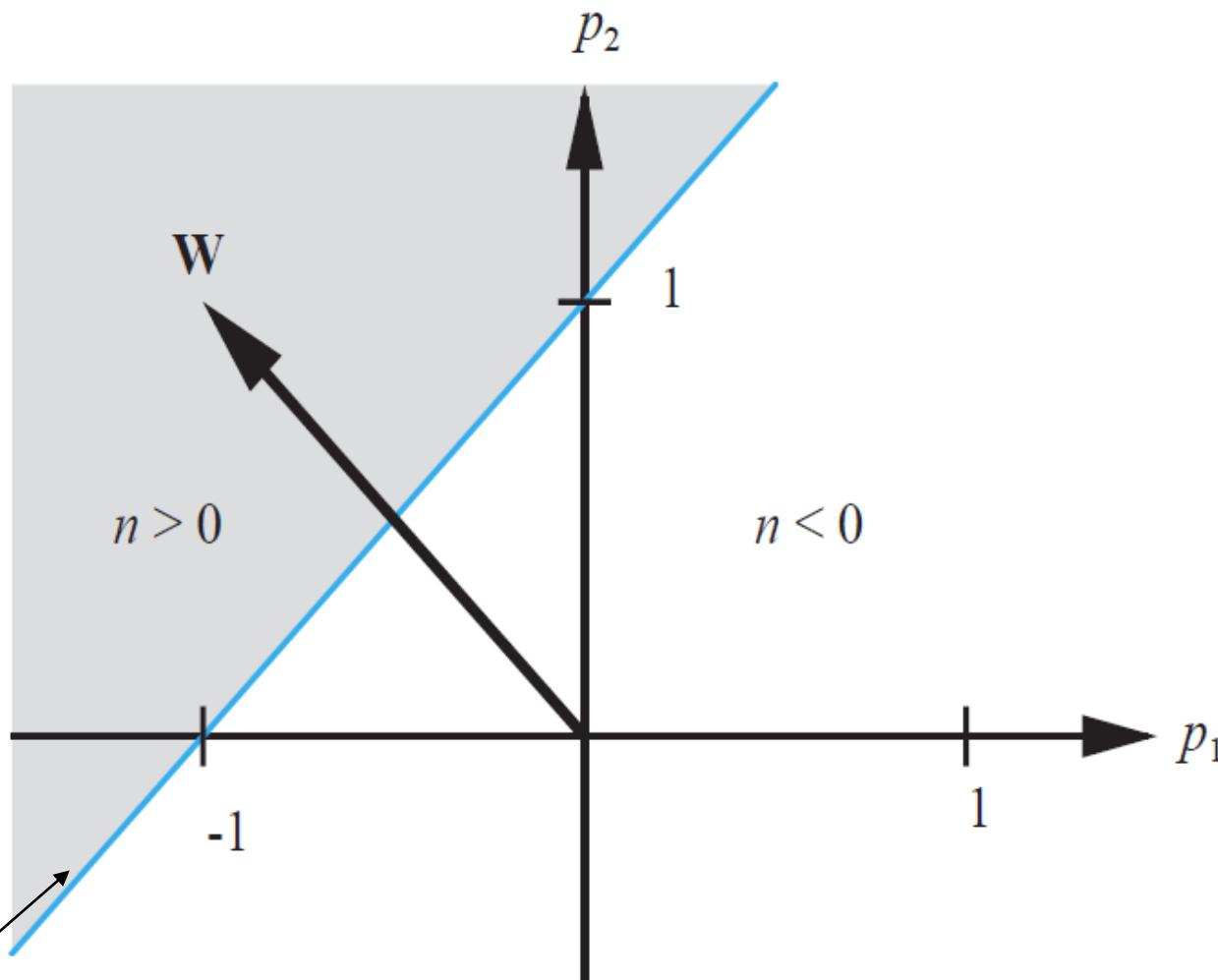


$$\text{Entonces: } a = \text{hardlims}(n) = \text{hardlims}([-1 \ 1]\mathbf{p} + b).$$

Si el producto punto es  $\geq$  que  $-b$ , entonces  $a = 1$ .

Si el producto punto es  $<$  que  $-b$  entonces  $a = -1$ .

Esto divide el espacio de entrada en dos partes: Caso para  $b = -1$ :



Puntos para los cuales  $n = [-1 \ 1]\mathbf{p} - 1 = 0$ .

## **Principio del perceptrón:**

Puede separar vectores de entrada en dos categorías.

La frontera de decisión entre las dos categorías queda determinada, en general, como:

$$\mathbf{Wp} + b = 0.$$

NOTA: Debido a que la frontera es siempre una línea, un solo perceptrón puede ser usado para clasificar patrones linealmente separables.

**Ejemplo de  
reconocimiento  
de patrones:**

Consideremos de nuevo el problema del reconocimiento de las manzanas y las naranjas:

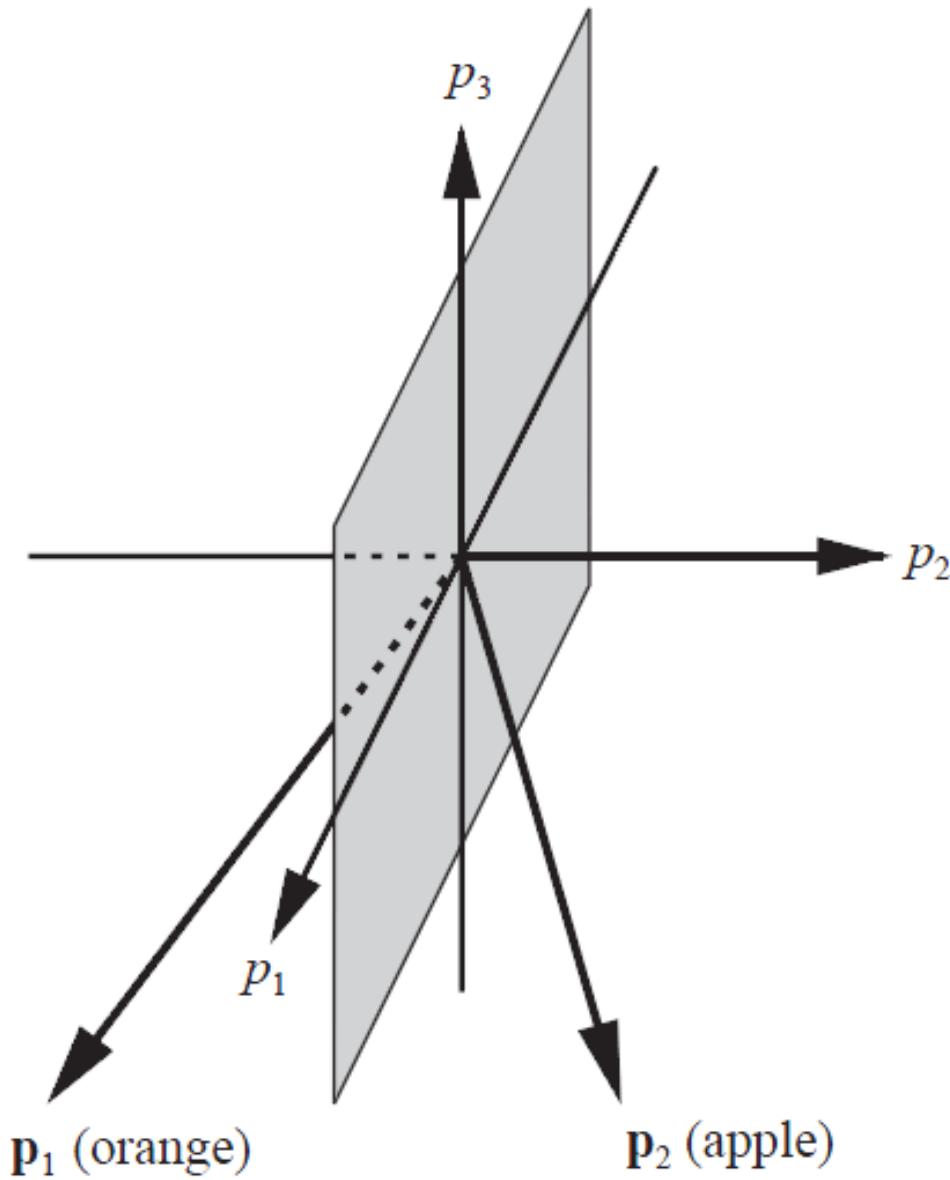
Como son dos clases, con un perceptrón será suficiente:

Ecuación del perceptrón:

$$a = \text{hardlims} \left( \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + b \right).$$

Se escogerá el valor de  $b$  de forma que los dos objetos sean clasificados, por ejemplo +1 cuando se trate de una manzana y -1 cuando se trate de una naranja.

En este caso la frontera de separación será el plano  $p_1p_3$ .



Este plano  $p_1p_3$  puede ser descrito por la ecuación:

$$p_2 = 0$$

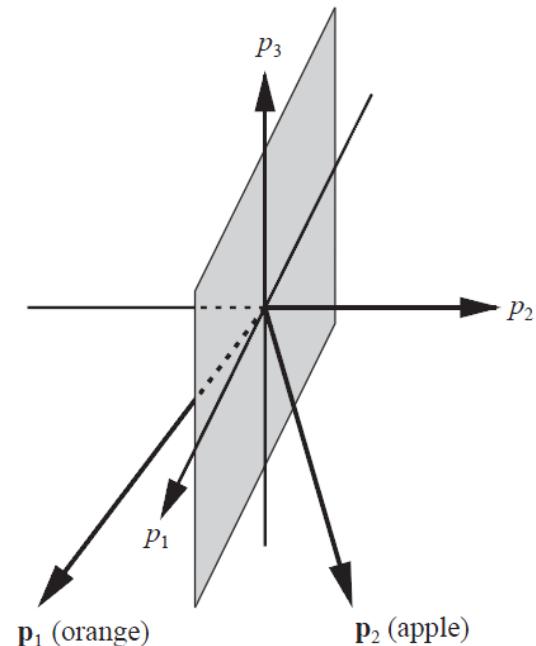
O

$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + 0 = 0.$$

La matriz de pesos y el bías serán,  
respectivamente:

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, b = 0.$$

$b = 0$ , ya que el plano pasa por el origen.



Prueba del perceptrón:

Naranja:

$$a = \text{hardlims} \left( \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0 \right) = -1(\text{orange}),$$

Manzana:

$$a = \text{hardlims} \left( \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + 0 \right) = 1(\text{apple}).$$

¿Qué pasa si se presenta una versión imperfecta de un patrón de entrada?

Por ejemplo, naranja un poco elíptica:  $\mathbf{p} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$ .

La respuesta del perceptrón es:

$$a = \text{hardlims} \left( \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + 0 \right) = -1(\text{orange}).$$

¡Que es lo que se desea!

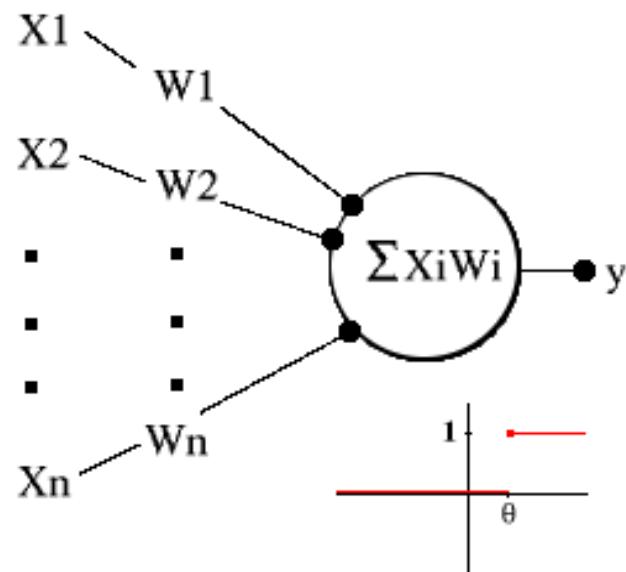
Cualquier vector de entrada cerca del patrón naranja será correctamente clasificado y viceversa.

# **El modelo de MaCulloch-Pitts**

## La TLU (McCulloch-Pitts en 1943).

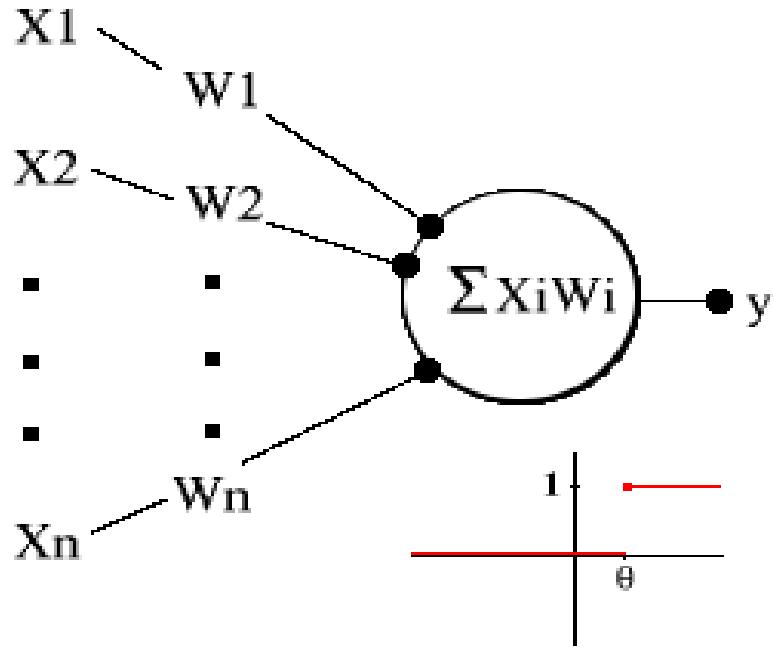
Un modelo muy sencillo de la neurona real estudiada en la sección anterior es la Unidad Lógica de Umbralado (ULU), del inglés (**Threshold Logic Unit, TLU**).

Una ULU recibe a la entrada  $n$  entradas:  $x_1, x_2, \dots, x_n$



El efecto de cada sinapsis es modelado por un número o peso  $w_i$  que multiplica a la entrada  $x_i$

Las acciones de excitación y de inhibición son modeladas, respectivamente por **valores positivos o negativos**.

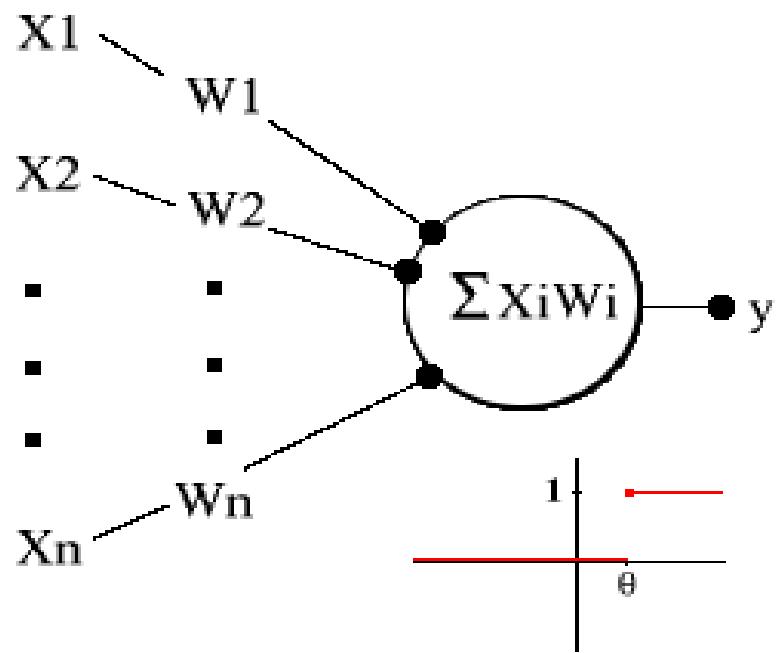


Con los  $n$  pesos:  $w_1, w_2, \dots, w_n$   
se forman  $n$  productos:

$$w_1 x_1, w_2 x_2, \dots, w_n x_n$$

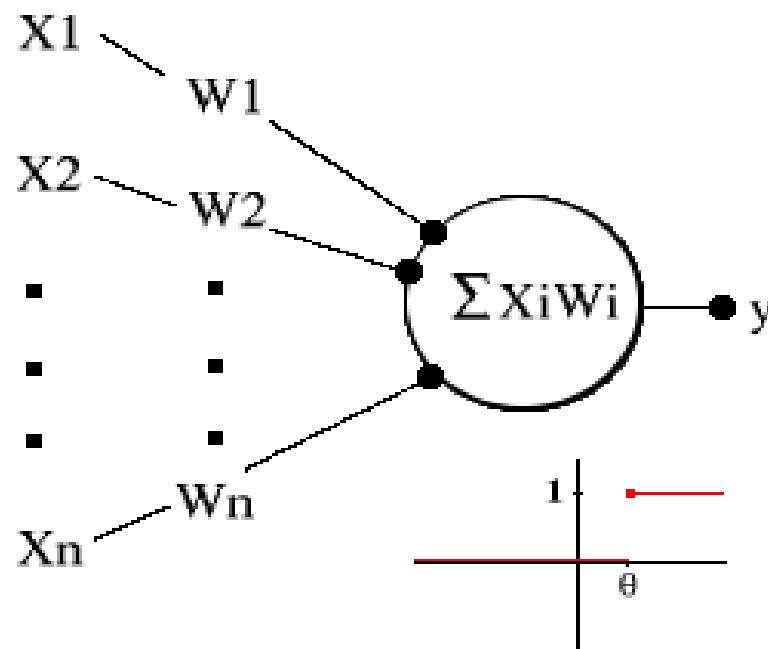
Cada uno de estos productos es el análogo a un PPS, y puede ser positivo o negativo dependiendo del signo del peso.

Estos productos son combinados (mediante una **suma**) tratando de simular el proceso que ocurre en la loma axonal.



En el caso de la TLU, esto se hace simplemente al sumar los productos, dando como resultado la activación (correspondiente al potencial de membrana):

$$a = w_1x_1, w_2x_2, \dots, w_nx_n$$



Ejemplo: Para el vector de entrada:

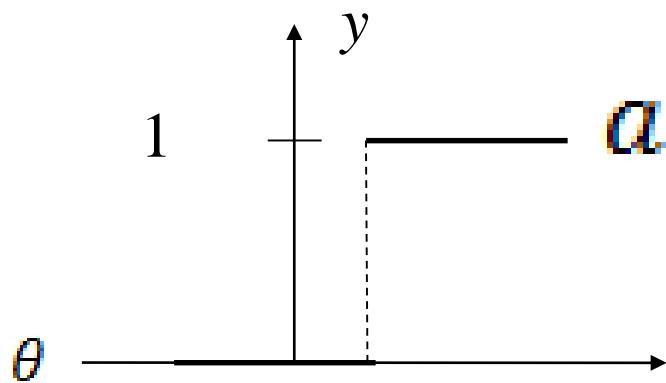
$$x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 0$$

y el vector de pesos:

$$(w_1 = 0.5 \ w_2 = 1.0 \ w_3 = -1.0 \ w_4 = -0.5 \ w_5 = 1.2).$$

$$a = 0.5 \times 1 + 1.0 \times 1 - 1.0 \times 1 - 0.5 \times 0 + 1.2 \times 0 = 0.5$$

Para simular la generación del correspondiente potencial de acción, se usa un valor de umbral, digamos  $\theta$  tal que si el valor de activación  $a$  excede o es igual a  $\theta$  entonces la neurona emite como salida  $y = 1$  (potencial de acción), si  $a < \theta$  la neurona emite un  $y = 0$



Ejemplo:  $\theta = 0.2$      $a = 0.5 > \theta = 0.2, y = 1$

Este tipo de neurona artificial se le conoce como TLU y fue propuesta por:

W. McCulloch, W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 7:115 - 133.

En forma más compacta la activación de la TLU se puede escribir como:

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

La salida umbralada se puede escribir como:

$$y = \begin{cases} 1 & \text{si } a \geq \theta \\ 0 & \text{si } a < \theta \end{cases}$$

Nótese que:

una TLU responde de manera **instantánea**, mientras que una neurona real integra tanto **en el tiempo como en el espacio**.

La generación del potencial de acción se representa por una simple función de umbralado.

## Características de una TLU: Resistencia al ruido y fallas de hardware.

Con este modelo simple se pueden demostrar dos propiedades de la RNA.

Ejemplo. Consideremos una TLU de dos entradas con pesos (0,1) y un umbral de 0.5.

Su respuesta a las cuatro posibles entradas se muestra en la tabla:

$x_1$	$x_2$	Activación	Salida
0	0	$0 \times 0 + 0 \times 1 = 0$	0
0	1	$0 \times 0 + 1 \times 1 = 1$	1
1	0	$1 \times 0 + 0 \times 1 = 0$	0
1	1	$1 \times 0 + 1 \times 1 = 1$	1

Suponer por alguna razón que los pesos no son correctamente ajustados y que ahora (0.2, 0.8).

$x_1$	$x_2$	Activación	Salida
0	0	$0 \times 0.2 + 0 \times 0.8 = 0$	0
0	1	$0 \times 0.2 + 1 \times 0.8 = 0.8$	1
1	0	$1 \times 0.2 + 0 \times 0.8 = 0.2$	0
1	1	$1 \times 0.2 + 1 \times 0.8 = 1$	1

Nótese que la salida sigue siendo la misma.

Esto mientras la activación no cruce el umbral.

Este comportamiento es clásico de un sistema no-lineal.

Se puede concluir que TLUs son robustas en la presencia de fallas en el hardware.

Ejemplo. Considerar ahora que las entradas aparecen alteradas debido al ruido. Por ejemplo  $1 \rightarrow 0.8$  y  $0 \rightarrow 0.2$ .

$x_1$	$x_2$	Activación	Salida
0.2	0.2	$0.2x0+0.2x1=0.2$	0
0.2	0.8	$0.2x0+0.8x1=0.8$	1
0.8	0.2	$0.8x0+0.2x1=0.2$	0
0.8	0.8	$0.8x0+0.8x1=0.8$	1

Nótese que la salida sigue siendo la misma.

Conclusión: Las TLUs son robustas en la presencia de ruido en las entradas.

**Ejercicio:** Estudiar el caso en que tanto las entradas como los pesos son alterados.

**En resumen, las RNA  
basadas en TLUS  
son RNAS de  
primera generación.**

## Comunicación no-binaria:

La TLU originalmente fue diseñada para funcionar con valores “1” y “0”.

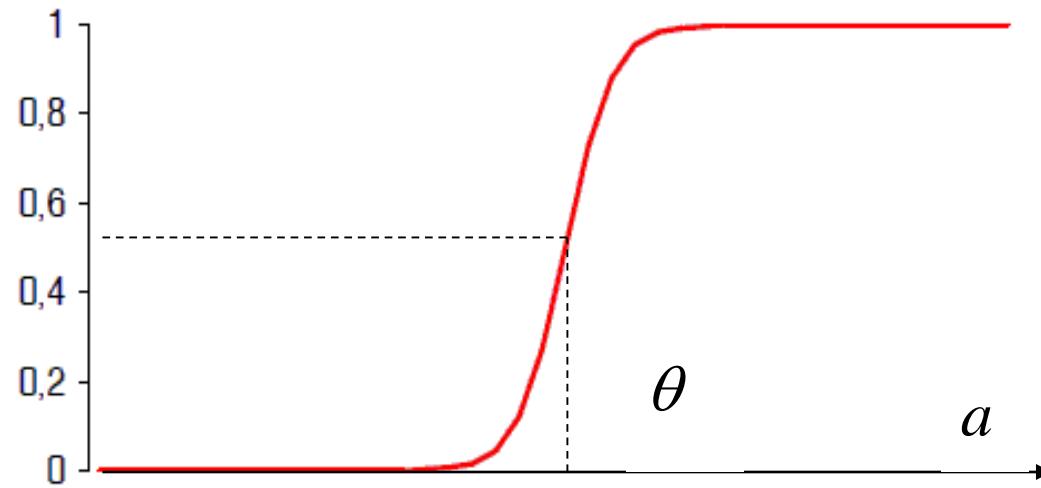
En el caso de las neuronas reales los valores de salida son codificados en la forma (patrón) de sus potenciales de acción, y no en la simple presencia o ausencia de un pulso.

Ya vimos que a la salida de la TLU se tiene un 1 o un 0.

Si se intentara conectar varias TLUs en red, no se permitiría **el manejo de señales continuas**.

Una manera de resolver este problema consiste en **alisar** la función escalón de manera que se obtenga una **función suave** de forma que y dependa suavemente de  $a$ .

Una función que hace esto es la llamada **sigmoide logística** o “**sigmoide**”.



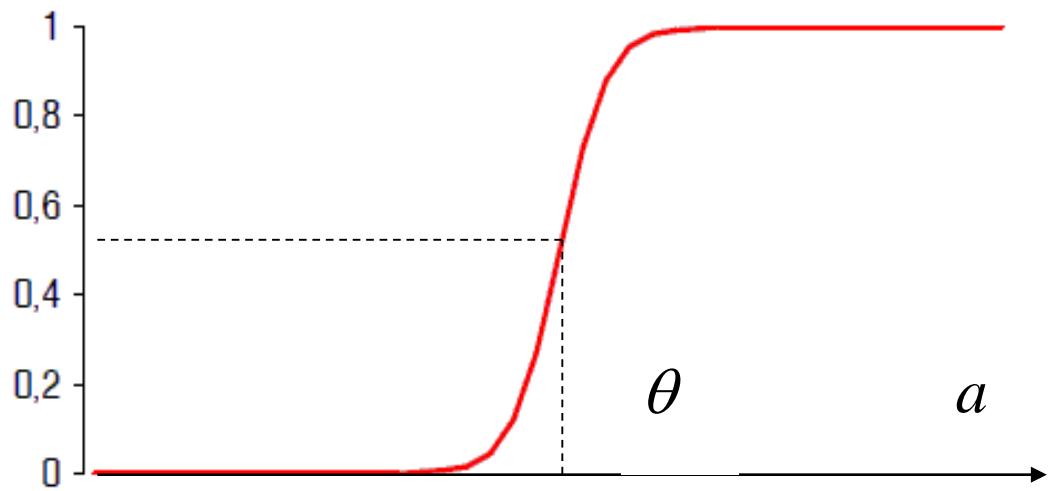
Conforme  $a$  es más positiva, la sigmoide tiende a 1.

Conforme  $a$  es más negativa, la sigmoide tiene a 0.

Notar que la sigmoide **es simétrica** respecto al eje y en 0.5.

La sigmoide se expresa mediante una de forma que:

$$y = \sigma(a) \equiv \frac{1}{1 + e^{-(a-\theta)/\rho}}$$



$\rho$  determina la forma de la sigmoide.

$\rho$  grande, la sigmoide es más plana.

$\rho$  pequeño, la sigmoide aproxima más la función escalón.

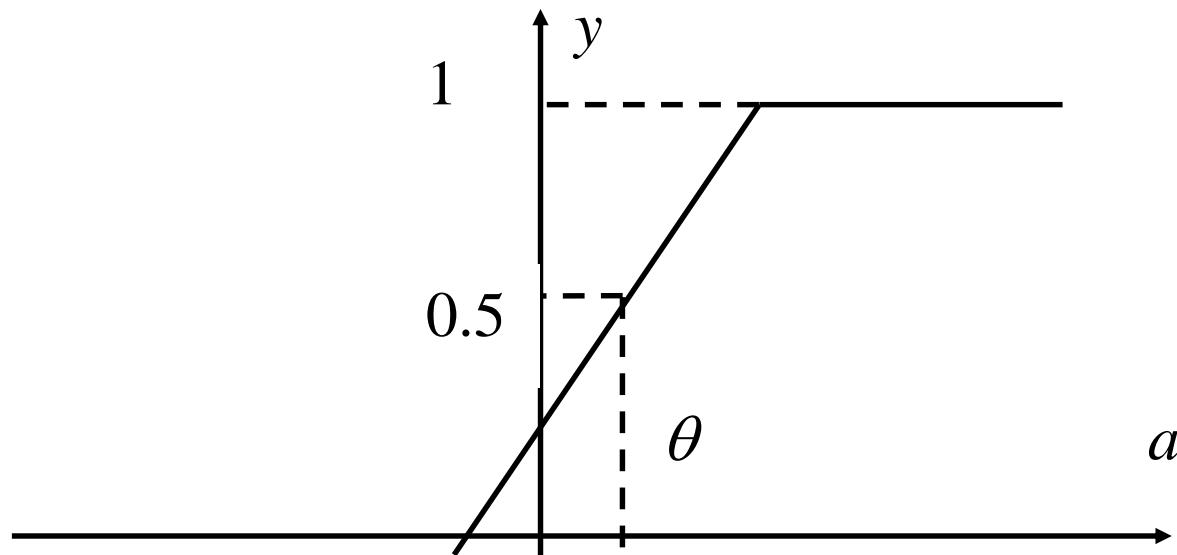
$$y = \sigma(a) \equiv \frac{1}{1 + e^{-(a - \theta)/\rho}}$$

La activación se calcula de la misma forma:

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

La función sigmoide se le llama “**semi-lineal**” ya que puede ser aproximada por la función lineal por partes:

Note que en una parte de interés significativa, a valores intermedios, la función es lineal con pendiente no cero.

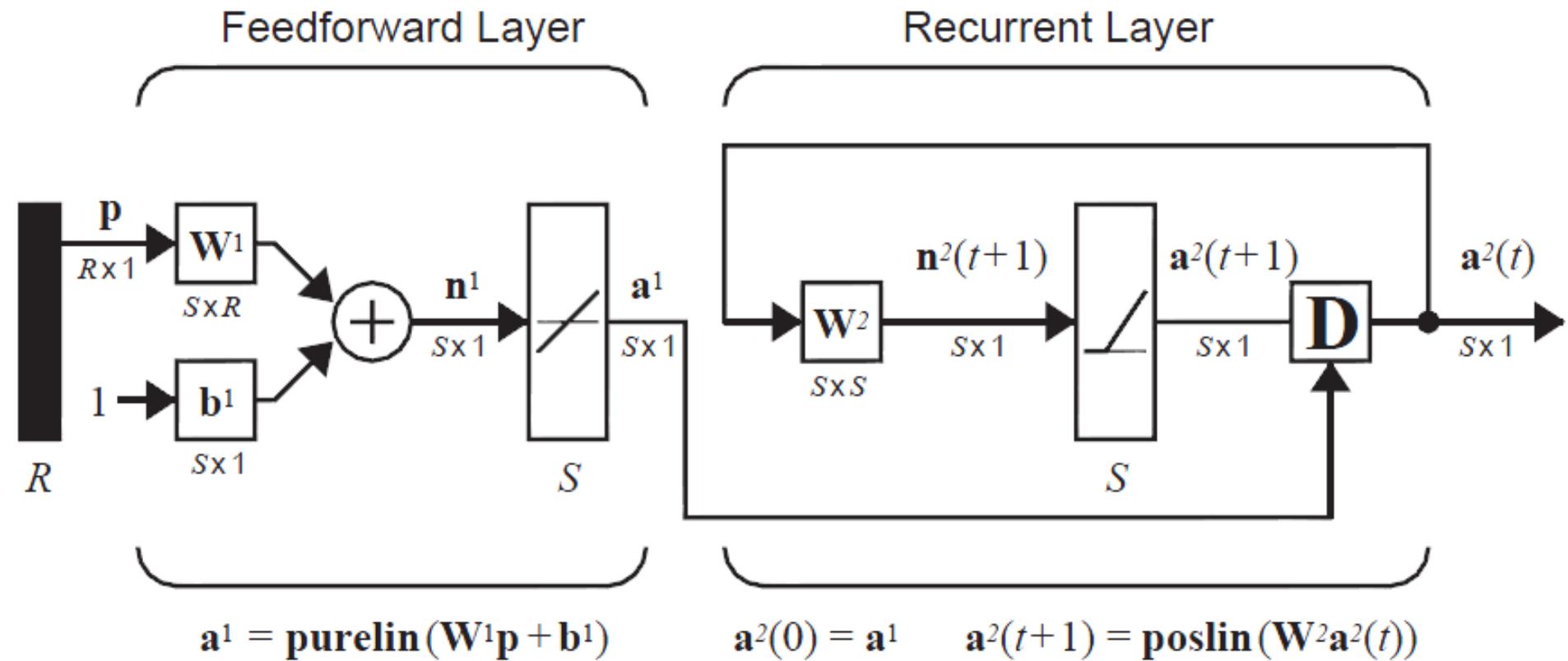


# **Otros ejemplos de RNAs:**

# RNA tipo Hamming:

Fue diseñada para resolver problemas de **reconocimiento de patrones binarios**.

Combina dos capas, una hacia adelante y una recurrente.



El objetivo de la red de Hamming es decidir **que prototipo es más cercano al vector de entrada.**

El resultado se ve reflejado en la salida de la capa recurrente.

**Por cada prototipo** se tiene una neurona de salida.

Cuando la red converge, sólo una neurona tendrá un valor no cero.

Esta neurona representará la salida del patrón deseado.

## Capa hacia adelante:

Esta capa realiza **una correlación (producto punto)** entre cada prototipo y el patrón de entrada.

Primero las filas de la matriz de pesos se igualan con las descripciones de los objetos de interés. Para nuestro ejemplo:

$$\mathbf{W}^1 = \begin{bmatrix} \mathbf{P}_1^T \\ \mathbf{P}_2^T \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix}.$$

La capa hacia adelante usa una función lineal, cada elemento del vector de desplazamientos se iguala con la dimensión del vector de entrada. Para nuestro ejemplo:

$$\mathbf{b}^1 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}.$$

Con estas selecciones, la salida de esta capa viene dada como:

$$\mathbf{a}^1 = \mathbf{W}^1 \mathbf{p} + \mathbf{b}^1 = \begin{bmatrix} \mathbf{P}_1^T \\ \mathbf{P}_2^T \end{bmatrix} \mathbf{p} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} \mathbf{P}_1^T \mathbf{p} + 3 \\ \mathbf{P}_2^T \mathbf{p} + 3 \end{bmatrix}.$$

**NOTA:** Las salidas de la capa hacia adelante igualan el **producto punto** entre los prototipos con la entrada.

Para vectores de igual tamaño, su producto punto será **más grande** cuando los vectores apunten en la misma dirección y **pequeños** cuando lo hagan en direcciones opuestas.

**¿Cuál es el efecto de agregar como número al producto punto la dimensionalidad del vector de entrada?**

Esta red se le llama de Hamming porque la neurona en la capa hacia adelante con la salida más grande corresponderá al prototipo más cercano en términos de distancia de Hamming al vector de entrada.

La distancia de Hamming entre dos vectores es igual al número de elementos diferentes, sólo para vectores binarios.

Problema: Demostrar que las salidas de la capa hacia adelante son iguales a  $2R$  menos 2 las distancias de Hamming de los patrones prototipo al patrón de entrada.

## Capa recurrente:

Esta capa se conoce como **capa competitiva**.

Sus neuronas se inician con las salidas de la capa hacia adelante.

Luego las neuronas compiten para determinar la ganadora.

Al final, sólo una neurona emitirá un salida no cero.

La neurona ganadora indica la clase de pertenencia del patrón de entrada.

Ecuaciones que describen la competencia:



poslin

$$\mathbf{a}^2(0) = \mathbf{a}^1 \quad (\text{Initial Condition}),$$

$$\mathbf{a}^2(t+1) = \text{poslin}(\mathbf{W}^2 \mathbf{a}^2(t)).$$

La matriz de pesos tiene la forma:

$$\mathbf{W}^2 = \begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix},$$

Con  $\varepsilon$  es cualquier número **menor** a  $1/(S - 1)$  con  $S$  el número de neuronas en la capa recurrente.

Problema. Demostrar porqué  $\varepsilon$  debe ser menor a  $1/(S - 1)$ .

Una iteración en la recurrente procede como sigue:

$$\mathbf{a}^2(t+1) = \text{poslin}\left(\begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix} \mathbf{a}^2(t)\right) = \text{poslin}\left(\begin{bmatrix} a_1^2(t) - \varepsilon a_2^2(t) \\ a_2^2(t) - \varepsilon a_1^2(t) \end{bmatrix}\right).$$

El efecto de esta operación es **ir reduciendo** las neuronas de salida hasta cero excepto por aquella neurona cuyo prototipo se acerca más al de entrada.

Operación (consideremos el objeto naranja alargada):

$$\mathbf{p} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}.$$

La salida de la capa hacia adelante:

$$\mathbf{a}^1 = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} (1 + 3) \\ (-1 + 3) \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix},$$

La matriz de pesos para la capa recurrente viene dada como:

$$\mathbf{W}^2 = \begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix},$$

Si  $\varepsilon = 0.5$ , la primera iteración es:

$$\mathbf{a}^2(1) = \mathbf{poslin}(\mathbf{W}^2 \mathbf{a}^2(0)) = \begin{cases} \mathbf{poslin}\left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \end{bmatrix}\right) \\ \mathbf{poslin}\left(\begin{bmatrix} 3 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \end{cases}.$$

La segunda iteración viene dada como:

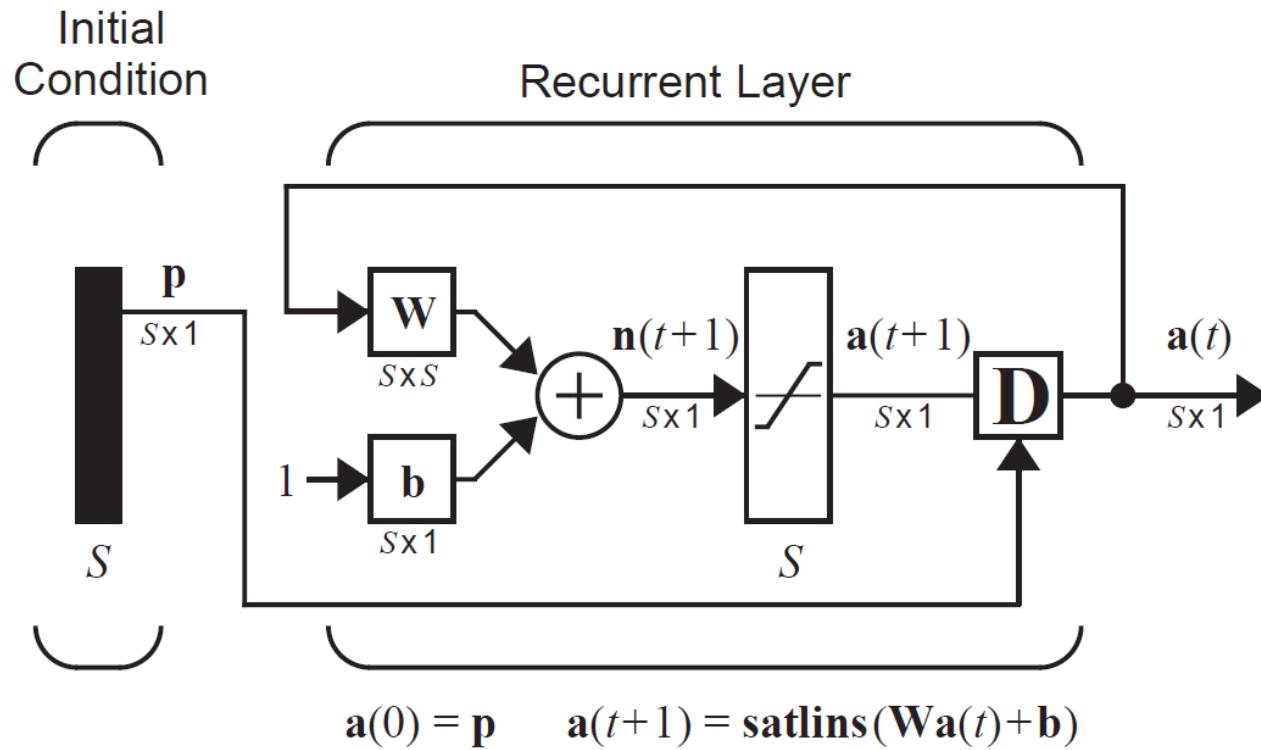
$$\mathbf{a}^2(2) = \text{poslin}(\mathbf{W}^2 \mathbf{a}^2(1)) = \begin{cases} \text{poslin}\left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 0 \end{bmatrix}\right) \\ \text{poslin}\left(\begin{bmatrix} 3 \\ -1.5 \end{bmatrix}\right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \end{cases}.$$

Como las salidas producen el mismo resultado, la red ha convergido.

El prototipo seleccionado es el primero que corresponde a la naranja, como se quería.

**Se escogió este porque su distancia de Hamming es de 1 mientras que el de la manzana es de 2.**

# Red de Hopfield:



Las neuronas de esta red se inician con el vector de entrada, luego la red **itera** hasta que su salida converge.

Cuando la RH opera correctamente la salida es uno de los prototipos.

## Operación de la red.

$$\mathbf{a}(0) = \mathbf{p}$$

$$\mathbf{a}(t + 1) = \textit{satlins}(\mathbf{W}\mathbf{a}(t) + \mathbf{b})$$

La función satlins es lineal en el rango [-1,+1].

Satura a +1 valores de entrada mayores a 1 y –a valores de entrada menores a -1.

El diseño de la red se verá más tarde.

Matrices de pesos de la red:

$$\mathbf{W} = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 1.2 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0.9 \\ 0 \\ -0.9 \end{bmatrix}$$

Ecuaciones de operación:

$$a_1(t + 1) = \text{satlins}(0.2a_1(t) + 0.9).$$

$$a_2(t + 1) = \text{satlins}(1.2a_1(t)).$$

$$a_3(t + 1) = \text{satlins}(0.2a_3(t) - 0.9).$$

Independientemente de los valores iniciales, el primer elemento aumentará hasta ser saturado al valor de +1, mientras que el tercer valor decrecerá hasta ser saturado a -1.

Si el segundo elemento es negativo irá a -1 y si +, irá a +1.

Ejemplo de operación con la naranja oblonga:

$$a_1(0) = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}.$$

Primera iteración:  $a_1(1) = \begin{bmatrix} 0.7 \\ -1 \\ -1 \end{bmatrix}.$

Segunda iteración:  $a_1(2) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}.$

Tercera operación:  $a_1(3) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}.$

La red a convergido al patrón deseado.

Note que las tres redes han entregado la solución deseada pero operando de manera distinta.

- 1) El perceptrón entrega un valor de -1 o +1.
- 2) En el caso de la red de Hamming, una sola neurona emite un valor superior a 0, el índice es la clase de pertenencia del patrón de entrada.
- 3) La RH converge de manera directa al prototipo deseado.