

## **Redes de Neuronas Artificiales**

- Introducción
- Perceptrón simple. Limitaciones
- Perceptrón Multicapa: algoritmo de retropropagación
- Aplicaciones de las redes de neuronas artificiales

## **Redes de Neuronas Artificiales. Introducción**

- Durante varias décadas los científicos han perseguido la construcción de algoritmos capaces de procesar información al igual que el cerebro humano
- En la actualidad existe un gran número de estructuras diferentes de redes de neuronas artificiales.
- Se caracterizan porque gozan de propiedades como la capacidad de aprendizaje la capacidad de aproximación a partir de ejemplos
- Las redes de neuronas se utilizan para afrontar una gran variedad de problemas: Aproximación, Predicción, Clasificación  
reconocimiento de patrones (imagen, voz, caracteres)  
compresión y análisis de datos  
robótica

## Redes de Neuronas Artificiales. Introducción

### Breve historia

- Los primeros estudios fueron realizados por McCulloch y Pitts en 1949. Mostraron la habilidad de un grupo de neuronas conectadas para llevar a cabo la implementación de ciertas funciones lógicas.
- Durante la década de los 50 hubo un considerable crecimiento en este campo. Rosenblatt en 1958 introdujo la primera arquitectura de red de neuronas artificial con capacidad de aprendizaje: **El perceptrón simple**.
- En 1969 y debido a que se demostraron las serias limitaciones de dicha red, la mayor parte de los investigadores en este área abandonaron su trabajo.
- A principios de la década de los 80, las redes de neuronas artificiales volvieron a renacer  
  
En 1986, Rumelhart, Mc Celland and Williams propusieron el **perceptrón multicapa** y el algoritmo de retropropagación

## Redes de Neuronas Artificiales. Introducción

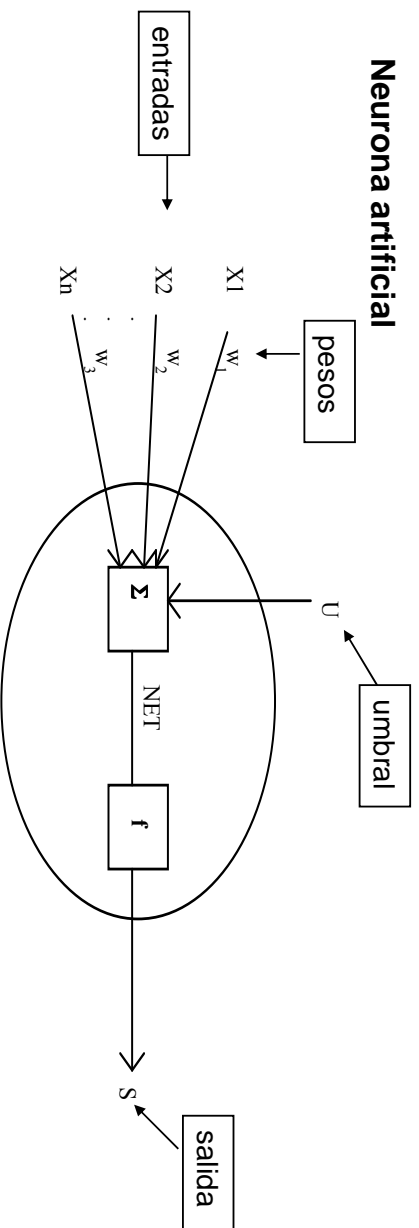
### Neurona artificial

Unidad elemental de una red de neuronas artificiales

- Recibe un conjunto de señales de entradas procedentes del mundo exterior o de otras neuronas.
- Las señales de entrada se reciben a través de unas conexiones, las cuales tienen un número real asociado llamado peso
- Procesa la información recibida, mediante una serie de operaciones simples
- Emite una señal de salida como respuesta a las señales de entrada

## Redes de Neuronas Artificiales. Introducción

### Neurona artificial



La salida de la neurona es:

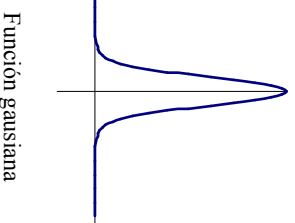
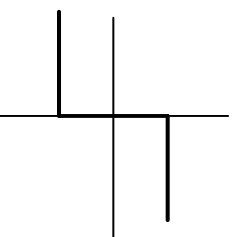
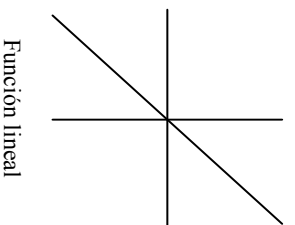
$$S = f(\text{NET})$$

$f$  es la función de activación

$$\text{NET} = X_1 * w_1 + X_2 * w_2 + \dots + X_n * w_n + U = \sum_{i=1}^n X_i * w_i + U$$

## Redes de Neuronas Artificiales. Introducción

### Funciones de activación



$$f(x) = x$$

$$f(x) = \begin{cases} f_1 & / & x > 0 \\ -f_1 & / & x \leq 0 \end{cases}$$

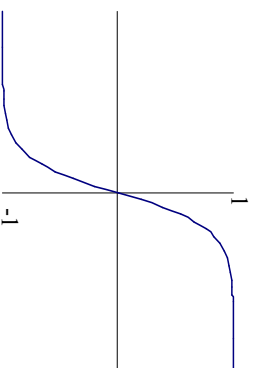
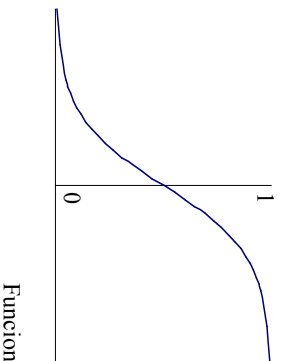
$$f(x) = e^{-\frac{x^2}{2}}$$

Función en (0,1)

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

Función en (-1,1)

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



## Redes de Neuronas Artificiales. Introducción

### Otros conceptos

- **Redes de neuronas**: es un conjunto de neuronas artificiales conectadas entre sí mediante una serie de arcos llamados conexiones. Estas conexiones tienen números reales asociados, llamados peso de la conexión.
- Las neuronas generalmente se distribuyen en **capas** de distintos niveles, con conexiones que unen las neuronas de distintas capas y/o neuronas de una misma capa.
- **Aprendizaje de la red**: es el proceso mediante el cual la red modifica sus respuestas ante las entradas para que sus pesos se vayan adaptando de manera paulatina al funcionamiento que se considera correcto.  
La modificación de los pesos se realiza en base a un criterio establecido y que permite que la red “aprenda” a dar las respuestas adecuadas.  
Supervisado  
No Supervisado

## Redes de Neuronas Artificiales. Introducción

### Otros conceptos

- **Aprendizaje supervisado**: Para cada patrón (ejemplo) presentado a la red existe una respuesta deseada. La respuesta de la red se compara con su salida deseada y en base a esa comparación se ajustan los pesos de la red.
- **Patrones de entrenamiento**: Conjunto de muestras o ejemplos para realizar el aprendizaje (determinación de pesos y umbrales).
- **Patrones de test o validación**: Conjunto de ejemplos utilizados para evaluar la capacidad de **generalización** de la red.

# Redes de Neuronas Artificiales. Introducción

## Tipos de redes de neuronas

- Redes feedforward  
Conexiones en un sólo sentido  
**Perceptron simple y multicapa**, Redes de Base Radial
- Redes recurrentes  
Conexiones en todas las direcciones
- Redes parcialmente recurrentes  
Unas pocas conexiones recurrentes  
Red de Jordan, Red de Elman
- Redes no supervisadas  
Kohonen, ART

## Redes de Neuronas Artificiales. Perceptrón simple

### Introducción

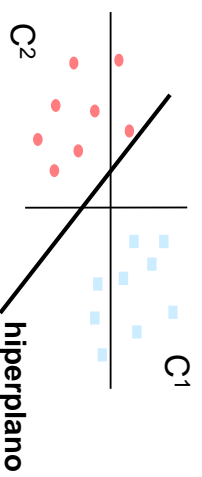
- Forma más simple de red de neuronas
- Estuvo inspirada en el modelo de célula de McCulloch-Pitts
- Modelo propuesto por Rosenblatt en 1959
- Adaptación supervisada
- Tareas de clasificación lineal: Dado un conjunto de **ejemplos o patrones**, determinar el **hiperplano** capaz de discriminar los patrones en dos clases

#### Ejemplos

Puntos de  $\mathbb{R}^n$ :  $(x_1, \dots, x_n)$

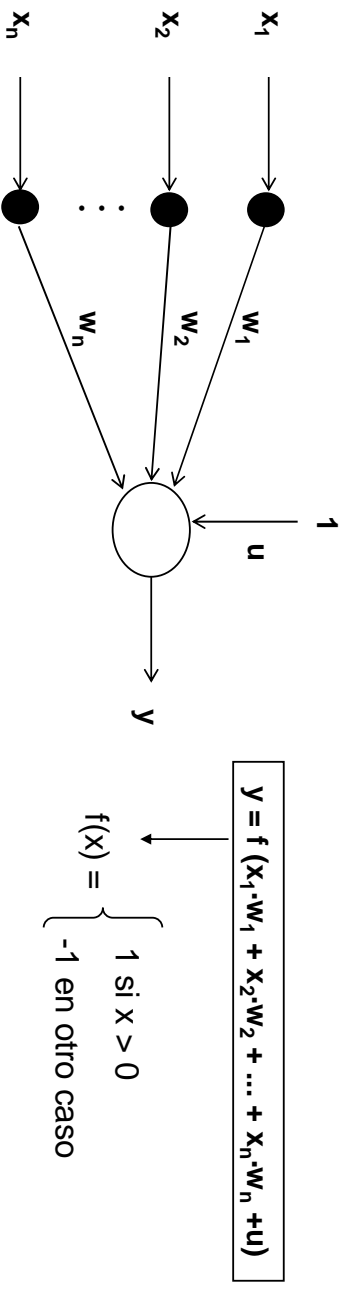
#### Hiperplano

$$w_1 \cdot x_1 + \dots + w_n \cdot x_n + w_0 = 0$$



## Redes de Neuronas Artificiales. Perceptr3n simple

### Arquitectura



La red puede utilizarse para clasificar los patrones de entrada

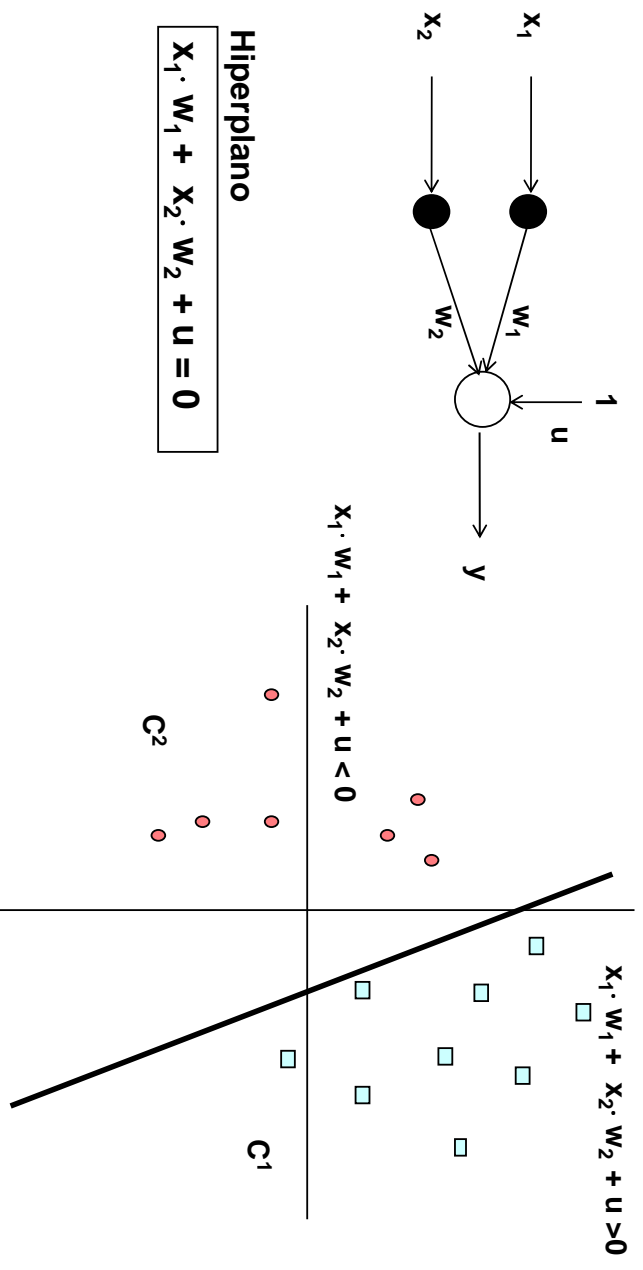
$$\text{Si } x_1 \cdot w_1 + x_2 \cdot w_2 + \dots + x_n \cdot w_n + u > 0 \Rightarrow y = 1 \Rightarrow (x_1, \dots, x_n) \in C^1$$

$$\text{Si } x_1 \cdot w_1 + x_2 \cdot w_2 + \dots + x_n \cdot w_n + u \leq 0 \Rightarrow y = -1 \Rightarrow (x_1, \dots, x_n) \in C^2$$

$$\text{Hiperplano: } x_1 \cdot w_1 + x_2 \cdot w_2 + \dots + x_n \cdot w_n + u = 0$$

## Redes de Neuronas Artificiales. Perceptr3n simple

### Arquitectura. Ejemplo para patrones bidimensionales



## Redes de Neuronas Artificiales. Perceptrón simple

### Definición del proceso de aprendizaje

Proceso iterativo supervisado: modificación de los parámetros de la red (pesos y umbral) hasta encontrar el hiperplano discriminante

*Número finito de iteraciones*

Dado

Conjunto de patrones
Vector de entrada: $x=(x_1, x_2, \dots, x_n)$
Salida deseada: $d(x)$
$d(x) = 1$ si $x \in C^1$
$d(x) = -1$ si $x \in C^2$

Encontrar

Hiperplano discriminante
$(w_1, w_2, \dots, w_n, u)$ tales que
$x_1 \cdot w_1 + x_2 \cdot w_2 + \dots + x_n \cdot w_n + u = 0$
separe las clases $C^1$ y $C^2$

## Redes de Neuronas Artificiales. Perceptrón simple

### Pasos del proceso de aprendizaje

**Paso 1:** Inicialización aleatoria de los pesos y el umbral de la red

$$\{w_i(0)\}_{i=0, \dots, n} \quad u(0)$$

**Paso 2:** Se toma un patrón entrada-salida  $[x=(x_1, x_2, \dots, x_n), d(x)]$

**Paso 3:** Se calcula la salida de la red:  $y = f(x_1 \cdot w_1 + x_2 \cdot w_2 + \dots + x_n \cdot w_n + u)$

Si  $y = d(x)$  (clasificación correcta) se vuelve al paso 2

Si  $y \neq d(x)$  (clasificación incorrecta) se modifican los parámetros y se vuelve al paso 2

$w_i(t+1) = w_i(t) + d(x) \cdot x_i \quad u(t+1) = u(t) + d(x)$
---

### Ley de aprendizaje

$$\text{Si } x \in C^1, d(x) = 1 \Rightarrow w_i(t+1) = w_i(t) + x_i \quad u(t+1) = u(t) + 1$$

$$\text{Si } x \in C^2, d(x) = -1 \Rightarrow w_i(t+1) = w_i(t) - x_i \quad u(t+1) = u(t) - 1$$

## Redes de Neuronas Artificiales. Perceptrón simple

### Regla de aprendizaje de Windrow-Hoff (1960)

$$w_i(t+1) = w_i(t) + (d(x) - y(x)) \cdot x_i \quad u(t+1) = u(t) + (d(x) - y(x))$$

- Para casos en los que la función de activación sea entre 0 y 1
- Comportamiento similar a la ley anterior
- La idea es utilizar el error cometido por la red para adaptar los pesos

Esta idea hizo que surgiera la arquitectura conocida como: **ADALINE**

La única diferencia con el perceptrón simple es que el valor  $y(x)$  que aparece en la ley de aprendizaje es la respuesta de la red, antes de aplicarle la función de activación umbral.

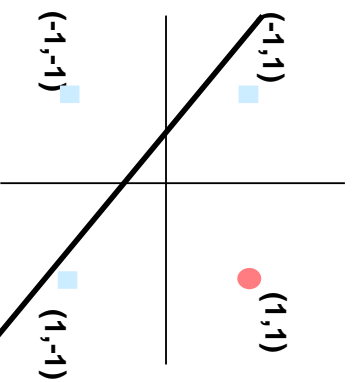
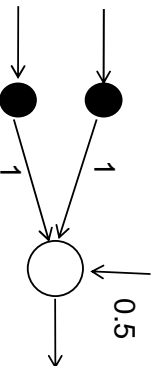
$$y(x) = x_1 \cdot w_1 + x_2 \cdot w_2 + \dots + x_n \cdot w_n + u$$

$$\text{La salida de la red } v(x) = f(y(x))$$

## Redes de Neuronas Artificiales. Perceptrón simple

### Ejemplo: Función lógica AND

$x_1$	$x_2$	AND
-1	-1	-1
1	-1	-1
-1	1	-1
1	1	1





## Redes de Neuronas Artificiales. Perceptrón simple

### Ejemplo: Función lógica AND

$$X=(-1,-1), d(x)=-1 \quad \Rightarrow \quad Y=f(-1.5)=-1 \quad \Rightarrow \quad \text{Bien clasificado}$$

$$X=(1,-1), d(x)=-1 \quad \Rightarrow \quad Y=f(0.5)=1 \quad \Rightarrow \quad \text{Mal clasificado}$$

**Nuevos parámetros**

$$w_1(1) = 1 - 1 = 0$$

$$w_2(1) = 1 - (-1) = 2$$

$$u(1) = 0.5 - 1 = -0.5$$

$$\Rightarrow Y=f(-2.5)=-1 \quad \Rightarrow \quad \text{Bien clasificado}$$

$$X=(-1,1), d(x)=-1 \quad \Rightarrow \quad Y=f(1.5)=1 \quad \Rightarrow \quad \text{Mal clasificado}$$

**Nuevos parámetros**

$$w_1(1) = 0 - (-1) = 1$$

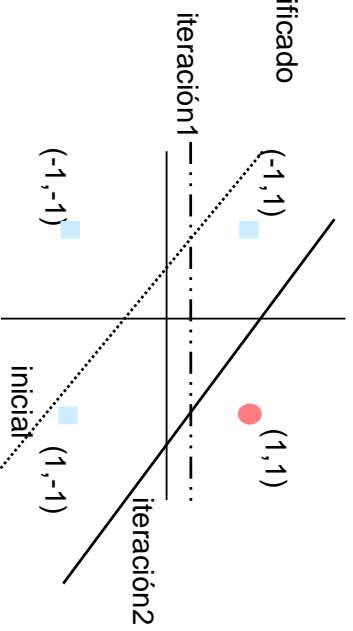
$$w_2(1) = 2 - 1 = 1$$

$$u(1) = -0.5 - 1 = -1.5$$

$$\Rightarrow Y=f(-1.5)=-1 \quad \Rightarrow \quad \text{Bien clasificado}$$

$$X=(1,1), d(x)=1 \quad \Rightarrow \quad Y=1 \quad \Rightarrow \quad \text{Bien clasificado}$$

Un hiperplano solución es:  $x_1 + x_2 - 1.5 = 0$



## Redes de Neuronas Artificiales. Perceptrón simple

### Razón de aprendizaje

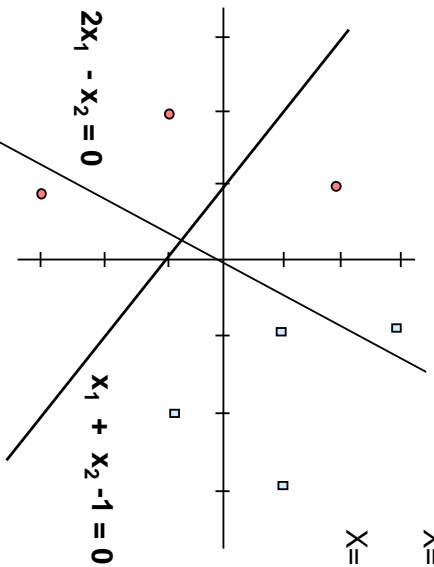
Sea  $w_1=w_2=U=1$

$$X=(-2,-1), d(x)=-1 \quad \Rightarrow \quad Y=-1 \quad \Rightarrow \quad \text{Bien clasificado}$$

$$X=(-1,2), d(x)=-1 \quad \Rightarrow \quad Y=1 \quad \Rightarrow \quad \text{Mal clasificado}$$

$$w_1(1) = 1 + 1 = 2 \quad w_2(1) = 1 - 2 = -1 \quad u(1) = 1 - 1 = 0$$

Los nuevos parámetros no clasifican correctamente a patrones que anteriormente estaban bien clasificados: (1, 3) y (-1, -3)



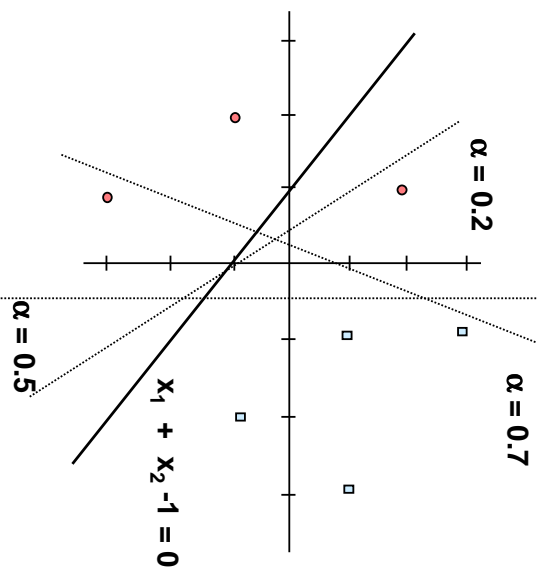
Estas situaciones pueden evitarse introduciendo la razón de aprendizaje

$$w_i(t+1) = w_i(t) + \alpha \cdot d(x) \cdot x_i \quad u(t+1) = u(t) + \alpha \cdot d(x)$$

$\alpha$ : controla la brusquedad de las modificaciones de los parámetros  $0 < \alpha < 1$

## Redes de Neuronas Artificiales. Perceptrón simple

Razón de aprendizaje



$$\alpha = 0.7$$

$$w_1(1) = 1 - \alpha \cdot (-1) = 1.7$$

$$w_2(1) = 1 - \alpha \cdot 2 = -0.4$$

$$u(1) = 1 - \alpha \cdot 1 = 0.3$$

$$\alpha = 0.5$$

$$w_1(1) = 1 - \alpha \cdot (-1) = 1.5$$

$$w_2(1) = 1 - \alpha \cdot 2 = 0$$

$$u(1) = 1 - \alpha \cdot 1 = 0.5$$

$$\alpha = 0.2$$

$$w_1(1) = 1 - \alpha \cdot (-1) = 1.2$$

$$w_2(1) = 1 - \alpha \cdot 2 = 0.6$$

$$u(1) = 1 - \alpha \cdot 1 = 0.8$$

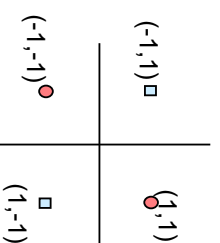
## Redes de Neuronas Artificiales. Perceptrón simple

### Limitaciones del perceptrón

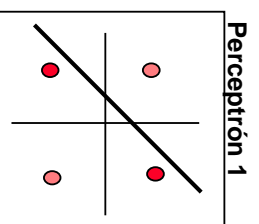
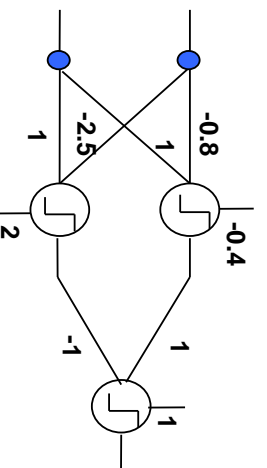
Si no existe un hiperplano  $\Rightarrow$  La ley de aprendizaje no encuentra la solución

Función XOR (OR exclusivo)

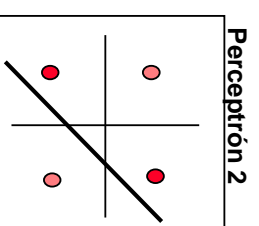
$x_1$	$x_2$	$d(x)$
-1	-1	1
-1	1	-1
1	-1	-1
1	1	1



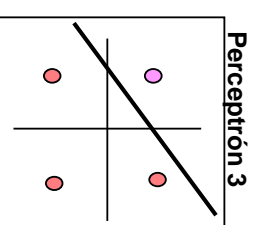
Una solución: **combinando varios perceptrones**



Perceptrón 1



Perceptrón 2

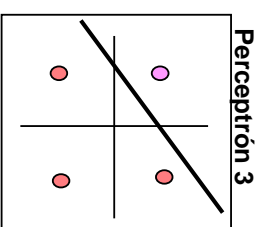
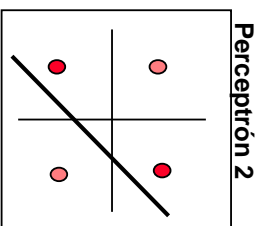
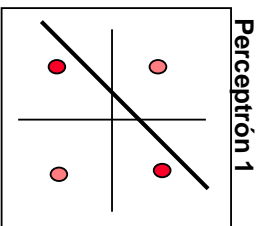


Perceptrón 3

## Redes de Neuronas Artificiales. Perceptrón simple

### Limitaciones del perceptrón

$x_1$	$x_2$	$d(x)$
-1	-1	1
-1	1	-1
1	-1	-1
1	1	1

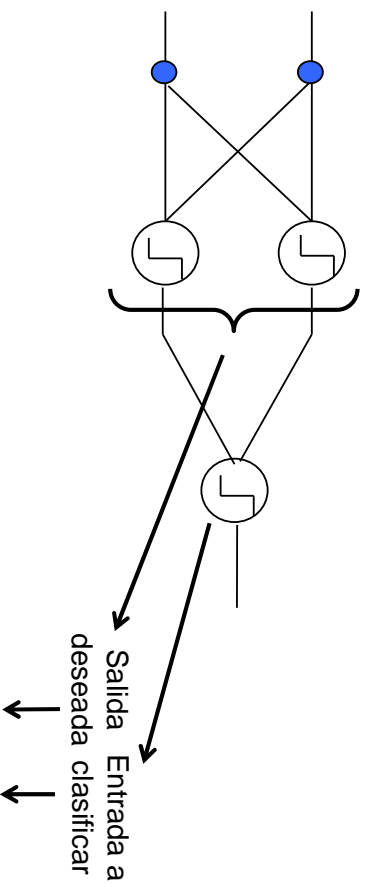


	Perceptron1	Perceptron2	Perceptron3
$(-1 -1)$	-1	1	1
$(-1 1)$	1	1	-1
$(1 -1)$	-1	-1	-1
$(1 1)$	-1	1	1

## Redes de Neuronas Artificiales. Perceptrón simple

### Limitaciones del perceptrón

Esta aproximación puede ser complicada de llevar a cabo en la práctica, pues la ley de aprendizaje no es aplicable y los pesos tendrían que ser determinados mediante un proceso manual



La ley de aprendizaje no es aplicable

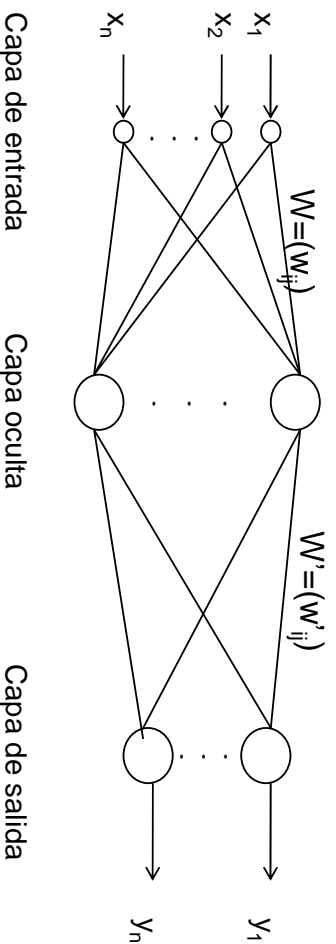
$$w_i(t+1) = w_i(t) + d(x) \cdot x_i$$

## Redes de Neuronas Artificiales. Perceptrón multicapa

Propuesto por Rumelhart, Hinton y Williams en 1986 para solventar las limitaciones del perceptrón simple: no linealidad

### Arquitectura

- El perceptrón multicapa tiene sus neuronas agrupadas en capas.
- Cada neurona en cada capa está conectada a todas las neuronas de la siguiente capa.
- Cada neurona procesa la información recibida y la respuesta se propaga a través de la conexión correspondiente y actúa como entrada para todas las neuronas de la siguiente capa.



## Redes de Neuronas Artificiales. Perceptrón multicapa

### Arquitectura

Dado un perceptrón multicapa con 3 capas,  $n$  neuronas en la capa de entrada,  $m$  neuronas en la capa de salida y  $r$  neuronas ocultas, las activaciones de las neuronas se calculan del siguiente modo:

**Capa de entrada:** recibe los patrones del exterior

$$a_i = x_i \quad i=1,2,\dots,n$$

**Capa oculta:**

$$b_j = f \left( \sum_{i=1}^n w_{ij} \cdot a_i + u_j \right) \quad j=1,2,\dots,r \quad f: \text{función sigmoideal}$$

**Capa de salida:** proporciona la respuesta de la red para cada patrón de entrada

$$y_j = f \left( \sum_{i=1}^r w'_{ij} \cdot b_i + v_j \right) \quad j=1,2,\dots,m$$

**Extensión a varias capas ocultas**

## Redes de Neuronas Artificiales. Perceptrón multicapa

### Arquitectura

#### Número de neuronas en la red

- Número de neuronas de entrada y salida viene dado por el problema a resolver, dependiendo de la codificación de la información
- Número de neuronas ocultas y/o número de capas ocultas: por prueba y error

#### **Teorema (Aproximadores universales)**

Toda función continua puede aproximarse utilizando un perceptrón multicapa con una única capa

El resultado no dice nada acerca del número de neuronas ocultas

## Redes de Neuronas Artificiales. Perceptrón multicapa

### Aprendizaje

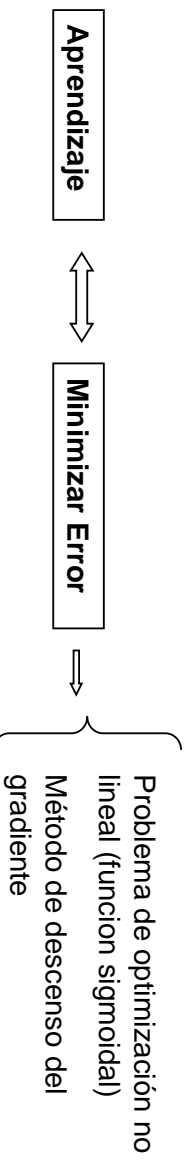
Proceso iterativo supervisado: modificación paulatina de los parámetros de la red (pesos y umbrales) hasta que la salida de la red sea lo más próxima posible a la salida deseada o esperada para cada patrón de entrenamiento

#### Dado

Conjunto de patrones o ejemplos
Vector de entrada: $x = (x_1, x_2, \dots, x_n)$
Vector de salida deseada: $t(x)$

#### Encontrar

Pesos $W = (w_{ij})$ , $W' = (w'_{ij})$ y umbrales $U = (u_i)$ , $V = (v_j)$ tales que $t(x) \approx y(x) \quad \forall$ patrón $x \Leftrightarrow  t(x) - y(x)  \approx 0 \quad \forall x$ $\Leftrightarrow$ Minimizar $E = \sum  t(x) - y(x) $
---



## Redes de Neuronas Artificiales. Perceptrón multicapa

### Aprendizaje

Sea un perceptrón multicapa, el error para el patrón  $[x=(x_1, x_2, \dots, x_n), t(x)]$  se expresa como:

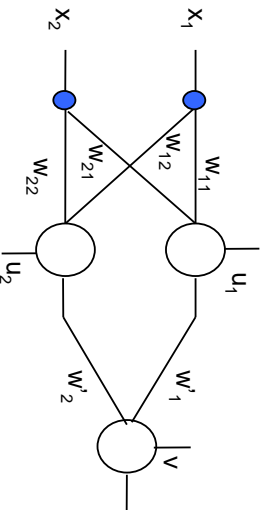
m salidas:

$$e(x) = \|t(x) - y(x)\| = \frac{1}{2} \sum_{i=1}^m (t_i(x) - y_i(x))^2 \quad \text{1 salida: } e(x) = \frac{1}{2} (t(x) - y(x))^2$$

El método de descenso del gradiente consiste en modificar los parámetros de la red siguiente la dirección negativa del gradiente del error:

$$w_{\text{nuevo}} = w_{\text{anterior}} + \alpha \cdot \left( -\frac{\partial e}{\partial w} \right) = w_{\text{anterior}} - \alpha \cdot \frac{\partial e}{\partial w} \quad \text{V parámetro } w$$

El algoritmo de retropropagación es el resultado de aplicar dicho método al perceptrón multicapa



$$b_1 = f(w_{11} \cdot x_1 + w_{21} \cdot x_2 + u_1)$$

$$b_2 = f(w_{12} \cdot x_1 + w_{22} \cdot x_2 + u_2)$$

$$y = f(w'_{11} \cdot b_1 + w'_{22} \cdot b_2 + v)$$

➔ **Pesos de la capa oculta a la de salida:  $w'_{11}, w'_{22}$**

$$w'_{11 \text{ nuevo}} = w'_{11 \text{ anterior}} - \alpha \cdot \frac{\partial e}{\partial w'_{11}}$$

Como:  $e(x) = \frac{1}{2} (t(x) - y)^2 \iff \frac{\partial e}{\partial w'_{11}} = -(t(x) - y) \cdot \frac{\partial y}{\partial w'_{11}}$

Teniendo en cuenta la expresión de la salida y que  $f'(x)=f(x)(1-f(x))$ :

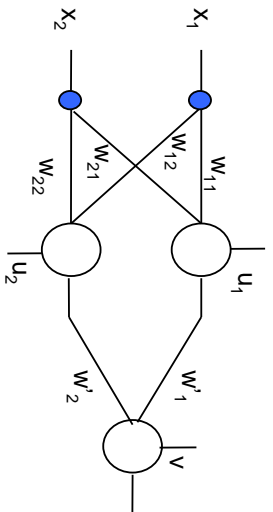
$$\frac{\partial y}{\partial w'_{11}} = f'(w'_{11} \cdot b_1 + w'_{22} \cdot b_2 + v) \cdot b_1 = y \cdot (1 - y) \cdot b_1$$

Por tanto:

$$w'_{11 \text{ nuevo}} = w'_{11 \text{ anterior}} + \alpha \cdot (t(x) - y) \cdot y \cdot (1 - y) \cdot b_1 = w'_{11 \text{ anterior}} + \alpha \cdot \delta(x) \cdot b_1$$

## Redes de Neuronas Artificiales. Perceptrón multicapa

### Aprendizaje: Algoritmo de retropropagación



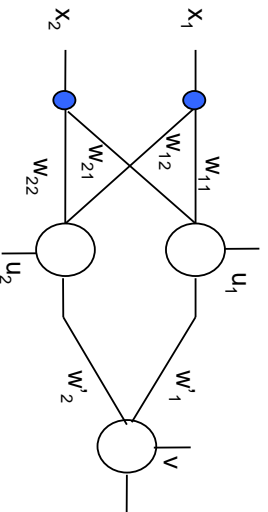
$$\begin{aligned} \text{Para } w'_{1i}: \quad w'_{1i \text{ nuevo}} &= w'_{1i \text{ anterior}} + \alpha \cdot \delta(x) \cdot b_i \\ \text{con} \quad \delta(x) &= (t(x) - y) \cdot y \cdot (1 - y) \\ \text{entonces} \quad w'_{2 \text{ nuevo}} &= w'_{2 \text{ anterior}} + \alpha \cdot \delta(x) \cdot b_2 \\ v^{\text{nuevo}} &= v^{\text{anterior}} + \alpha \cdot \delta(x) \end{aligned}$$

Para una red con r neuronas ocultas y una neurona de salida, los pesos de la capa oculta a la neurona de salida ( $w'_j$ ) se modifican siguiendo las ecuaciones:

$$\begin{aligned} w'_{ij \text{ nuevo}} &= w'_{ij \text{ anterior}} + \alpha \cdot \delta(x) \cdot b_i \quad i=1,2, \dots, r \\ \text{con} \quad \delta(x) &= (t(x) - y) \cdot y \cdot (1 - y) \end{aligned}$$

## Redes de Neuronas Artificiales. Perceptrón multicapa

### Aprendizaje: Algoritmo de retropropagación



$$\begin{aligned} b_1 &= f(w_{11} \cdot x_1 + w_{21} \cdot x_2 + u_1) \\ b_2 &= f(w_{12} \cdot x_1 + w_{22} \cdot x_2 + u_2) \\ y &= f(w'_{11} \cdot b_1 + w'_{21} \cdot b_2 + v) \end{aligned}$$

➔ **Pesos de la capa de entrada a la oculta:**  $w_{ij}$   $w_{12 \text{ nuevo}} = w_{12 \text{ anterior}} - \alpha \cdot \frac{\partial e}{\partial w_{12}}$

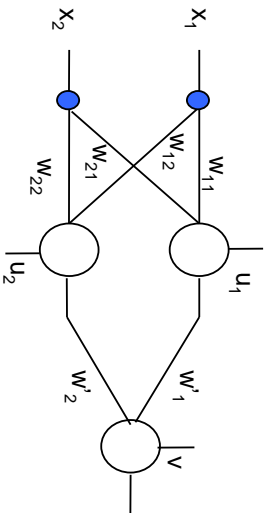
$$\frac{\partial e}{\partial w_{12}} = -(t(x) - y(x)) \cdot \frac{\partial y}{\partial w_{12}} = -(t(x) - y) \cdot y \cdot (1 - y) \cdot w'_{21} \cdot \frac{\partial b_2}{\partial w_{12}} = \delta(x) \cdot w'_{21} \cdot \frac{\partial b_2}{\partial w_{12}}$$

$$\frac{\partial b_2}{\partial w_{12}} = f'(w_{12} \cdot x_1 + w_{22} \cdot x_2 + u_1) \cdot x_1 = b_2 \cdot (1 - b_2) \cdot x_1$$

Por tanto:  $\frac{\partial e}{\partial w_{12}} = \delta(x) \cdot b_2 \cdot (1 - b_2) \cdot w'_{21} \cdot x_1$   $w_{12 \text{ nuevo}} = w_{12 \text{ anterior}} + \alpha \cdot a_2(x) \cdot x_1$

## Redes de Neuronas Artificiales. Perceptrón multicapa

### Aprendizaje: Algoritmo de retropropagación



$$\text{Para } w_{12}: \quad w_{12}^{\text{nuevo}} = w_{12}^{\text{anterior}} + \alpha \cdot a_2(x) \cdot x_1$$

$$\text{con } a_2(x) = \delta(x) \cdot b_2 \cdot (1 - b_2) \cdot w'_{12}$$

$$\text{entonces } w_{11}^{\text{nuevo}} = w_{11}^{\text{anterior}} + \alpha \cdot a_1(x) \cdot x_1$$

$$w_{21}^{\text{nuevo}} = w_{21}^{\text{anterior}} + \alpha \cdot a_1(x) \cdot x_2$$

$$w_{22}^{\text{nuevo}} = w_{22}^{\text{anterior}} + \alpha \cdot a_2(x) \cdot x_2$$

$$u_1^{\text{nuevo}} = u_1^{\text{anterior}} + \alpha \cdot a_1(x)$$

$$u_2^{\text{nuevo}} = u_2^{\text{anterior}} + \alpha \cdot a_2(x)$$

**Para una red con r neuronas ocultas y una neurona de salida, los pesos de la capa de entrada a la capa oculta ( $w_{ij}$ ) se modifican siguiendo las ecuaciones:**

$$w_{ij}^{\text{nuevo}} = w_{ij}^{\text{anterior}} + \alpha \cdot a_j(x) \cdot x_i \quad u_j^{\text{nuevo}} = u_j^{\text{anterior}} + \alpha \cdot a_j(x)$$

$$i=1,2, \dots, n; j=1,2, \dots, r \quad a_j(x) = \delta(x) \cdot b_j \cdot (1 - b_j) \cdot w'_j$$

## Redes de Neuronas Artificiales. Perceptrón multicapa

### Aprendizaje: Algoritmo de retropropagación

#### Resumen de las ecuaciones

Dado un perceptrón multicapa con 3 capas, n neuronas en la capa de entrada, 1 neurona en la capa de salida y r neuronas ocultas, los parámetros se modifican siguiendo las siguientes ecuaciones:

→ **Pesos de la capa oculta a la neurona de salida ( $w'_j$ ) y umbral de la salida**

$$w'_j{}^{\text{nuevo}} = w'_j{}^{\text{anterior}} + \alpha \cdot \delta(x) \cdot b_j$$

$$v^{\text{nuevo}} = v^{\text{anterior}} + \alpha \cdot \delta(x)$$

$$i=1,2, \dots, r \quad \delta(x) = (t(x) - y) \cdot y \cdot (1 - y)$$

→ **Pesos de la capa de entrada a la capa oculta ( $w_{ij}$ ) y umbrales de las ocultas**

$$w_{ij}^{\text{nuevo}} = w_{ij}^{\text{anterior}} + \alpha \cdot a_j(x) \cdot x_i \quad u_j^{\text{nuevo}} = u_j^{\text{anterior}} + \alpha \cdot a_j(x)$$

$$i=1,2, \dots, n; j=1,2, \dots, r \quad a_j(x) = \delta(x) \cdot b_j \cdot (1 - b_j) \cdot w'_j$$

Los errores cometidos en la capa de salida se propagan (hacia atrás) a las neuronas de la capa oculta



## Redes de Neuronas Artificiales. Perceptrón multicapa

### Proceso de aprendizaje o entrenamiento

**Paso 1:** Inicialización aleatoria de los pesos y umbrales

**Paso 2:** Dado un patrón del conjunto de entrenamiento  $(x, t(x))$ , se presenta el vector  $x$  a la red y se calcula la salida de la red para dicho patrón, y

**Paso 3:** Se evalúa el error  $e(x)$  cometido por la red

**Paso 4:** Se modifican todos los parámetros de la red utilizando las ecuaciones anteriormente descritas

**Paso 5:** Se repiten los pasos 2, 3, y 4 para todos los patrones de entrenamiento, completando así un ciclo de aprendizaje

**Paso 6:** Se realizan  $n$  ciclos de aprendizaje (pasos 2,3,4 y 5) hasta que se verifique el criterio de parada establecido

## Redes de Neuronas Artificiales. Perceptrón multicapa

### Proceso de aprendizaje o entrenamiento

Criterio de parada del proceso: Combinación de los siguientes factores

**Evaluación del error de entrenamiento:** Una vez realizado un ciclo de aprendizaje, se evalúa el error cometido por la red para todos los patrones de entrenamiento.

$$E = \sum_{A \text{ } x} e(x) \quad \text{Si } E \text{ es constante de un ciclo a otro, parar el aprendizaje.}$$

$E$  es constante de un ciclo a otro  $\iff$  Los parámetros dejan de sufrir modificaciones  $\iff \frac{\partial E}{\partial w} = 0 \iff \text{Min } E$

**Evaluación del error de validación o de test:** Cada cierto número de ciclos de entrenamiento, se presenta a la red todos los patrones de test, calculando la salida de la red para dichos patrones y evaluando el error cometido por la red.

$$E_{\text{test}} = \sum_{V \text{ } x \text{ de test}} e_{\text{test}}(x) \quad \text{Si } E_{\text{test}} \text{ evoluciona favorablemente, continuar.}$$

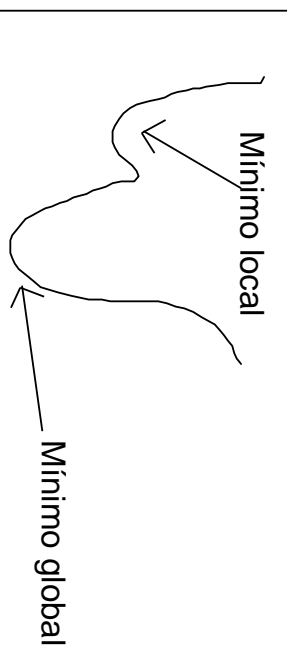
Si  $E_{\text{test}}$  no evoluciona favorablemente, detener el aprendizaje. No éxito

Si  $E_{\text{test}}$  ha alcanzado una cota deseada, detener el aprendizaje. Éxito

## Redes de Neuronas Artificiales. Perceptrón multicapa

### Proceso de aprendizaje o entrenamiento

#### Mínimo global y local



Desventaja del método: El proceso puede caer en un mínimo local del error

$$\text{El aprendizaje finaliza cuando } \frac{\partial E}{\partial w} \approx 0 \text{ ,}$$

hecho que también sucede en los mínimos locales.



El método detecta que cualquier pequeño cambio en los pesos, positivo o negativo, incrementa el error.

No es capaz de determinar en qué dirección deben moverse los pesos para que el error vuelva a decrecer.

## Redes de Neuronas Artificiales. Perceptrón multicapa

### Razón de aprendizaje

- El valor  $\alpha$  es el encargado de controlar cuánto se desplazan los pesos en la dirección negativa del gradiente.
- Influye en la velocidad de convergencia del algoritmo, puesto que determina la magnitud del desplazamiento.

#### Valores grandes de $\alpha$

- favorecen a una convergencia más rápida
- en las proximidades del mínimo existe el riesgo de saltar y oscilar alrededor de él

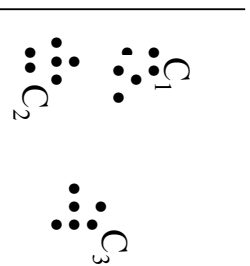
#### Valores pequeños de $\alpha$

- la convergencia es más lenta
- evita el problema de saltar el mínimo

## Redes de Neuronas Artificiales. Aplicaciones

### Clasificación. Formulación del problema

El problema de clasificación consiste en establecer una correspondencia entre un conjunto de elementos dados y un conjunto de clases.



Tres clases totalmente separadas

Dado un conjunto de ejemplos, no siempre es posible conocer cuántas clases tienen que ser consideradas.

Para poder afrontar un problema de clasificación con redes de neuronas supervisadas es condición necesaria y suficiente conocer el número de clases

Si el número de clases es desconocido es necesario utilizar técnicas no supervisadas

## Redes de Neuronas Artificiales. Aplicaciones

### Clasificación. Resolución utilizando el perceptrón multicapa

Sean  $X_i=(X_{i1},\dots,X_{in})$   $i=1,\dots,K$ , un conjunto de patrones y  $C_1, C_2,\dots, C_m$   $m$  clases diferentes

La red tendrá  $n$  neuronas de entrada que reciben los patrones  $X_i=(X_{i1},\dots,X_{in})$

$m$  neuronas de salida que representan las  $m$  clases  $C_1, C_2,\dots, C_m$ ,

#### Aprendizaje

La salida deseada para cada patrón de entrada  $X_i$  es una  $m$ -tupla  $(a_{i1},\dots,a_{im})$  donde

$$a_j=1 \text{ si } X_i \text{ pertenece a la clase } C_j$$

$$a_k=0 \text{ para todo } k \text{ distinto de } j$$

#### Codificación de los patrones de entrada

Los ejemplos a clasificar pueden ser de cualquier naturaleza, imágenes, juegos, letras, etc. y tienen que ser codificados en un vector de  $R^n$ . La codificación de los patrones juega un papel muy importante en el problema de clasificación hasta el punto que una codificación no adecuada puede producir resultados no satisfactorios.

## Redes de Neuronas Artificiales. Aplicaciones

### Predicción de series temporales. Formulación

Una serie temporal se puede definir como una función  $x$  : 
$$\begin{cases} \mathcal{R}^+ & \rightarrow \mathcal{R} \\ t & \rightarrow x(t) \end{cases}$$

El comportamiento temporal viene dado por ec. diferenciales o ec. en diferencias

Ejemplo:  $x(t+1) = a x(t) (1 - x(t))$  Serie temporal logística

El problema de predicción surge cuando la relación entre  $x(t)$  y sus valores anteriores es desconocida.

Determinar  **$f$  tal que  $x(t+1) = f(x(t-d), \dots, x(t))$  para todo  $t=d, d+1, d+2, \dots$**

Necesidad de definir el **valor  $d$** , el cual dependerá de cada serie temporal. Se pueden utilizar técnicas para su determinación, como el análisis de los espectros de frecuencia, análisis de Fourier, prueba y error.

## Redes de Neuronas Artificiales. Aplicaciones

### Predicción de series temporales. Resolución utilizando el perceptrón multicapa

Dado  $x(0), x(1), \dots, x(N)$ , y dado el valor  $d$ , la red tendrá:

- $d$  neuronas de entrada que reciben los valores  $(x(t-d), x(t-d-1), \dots, x(t-1), x(t))$
- Una neurona de salida. La salida deseada para el patrón de entrada  $(x(t-d), \dots, x(t))$  es el valor  $x(t+1)$ .

Una vez entrenada la red, se utiliza con el propósito de predicción:

- **Un paso:** Se presenta a la red el patrón  $(x(t-d), x(t-d-1), \dots, x(t-1), x(t))$  y se calcula su salida,  $x'(t+1)$ .
- **Múltiples pasos:** Con el patrón  $(x(t-d), x(t-d-1), \dots, x(t-1), x(t))$  se predice  $x'(t+1)$   
Con el patrón  $(x(t-d-1), x(t-d-2), \dots, x(t), x'(t+1))$  se predice  $x'(t+2)$   
Con el patrón  $(x(t-d-2), x(t-d-3), \dots, x'(t+1), x'(t+2))$  se predice  $x'(t+3)$   
Y así sucesivamente hasta conseguir la predicción deseada