Redes neuronales

Lecturas:

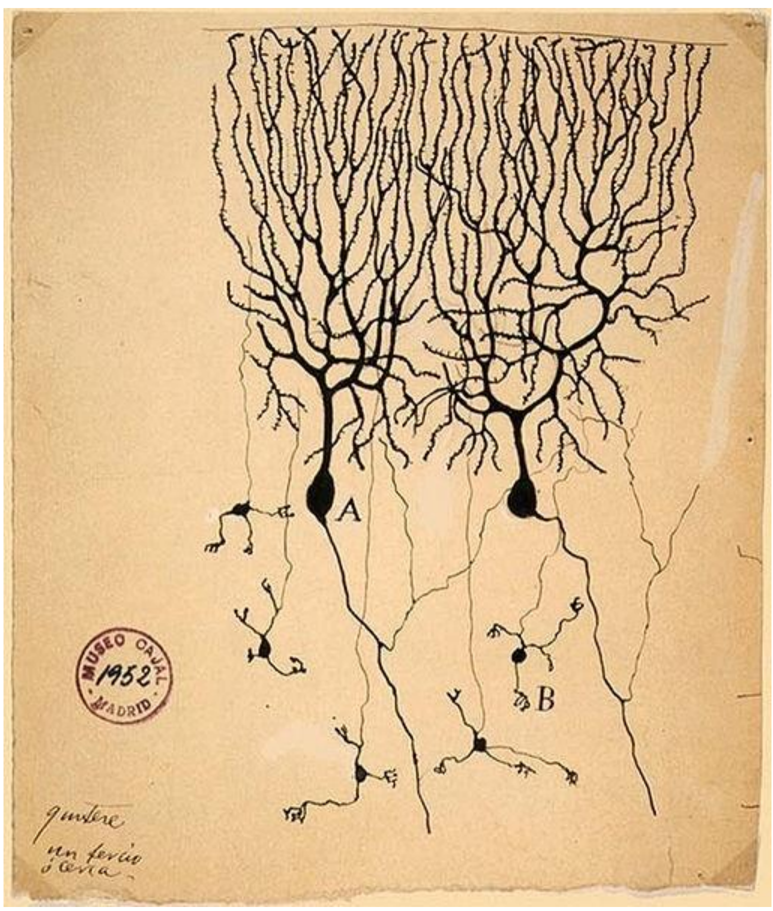
- Secciones 5.1,5.2, 5.3 Bishop
- Capítulos 1,2,3,4 Haykin
- Capítulos 1,3,4 Fausett

El cerebro

El cerebro es una red interconectada de neuronas.
Las neuronas se comunican mediante señales eléctricas y químicas

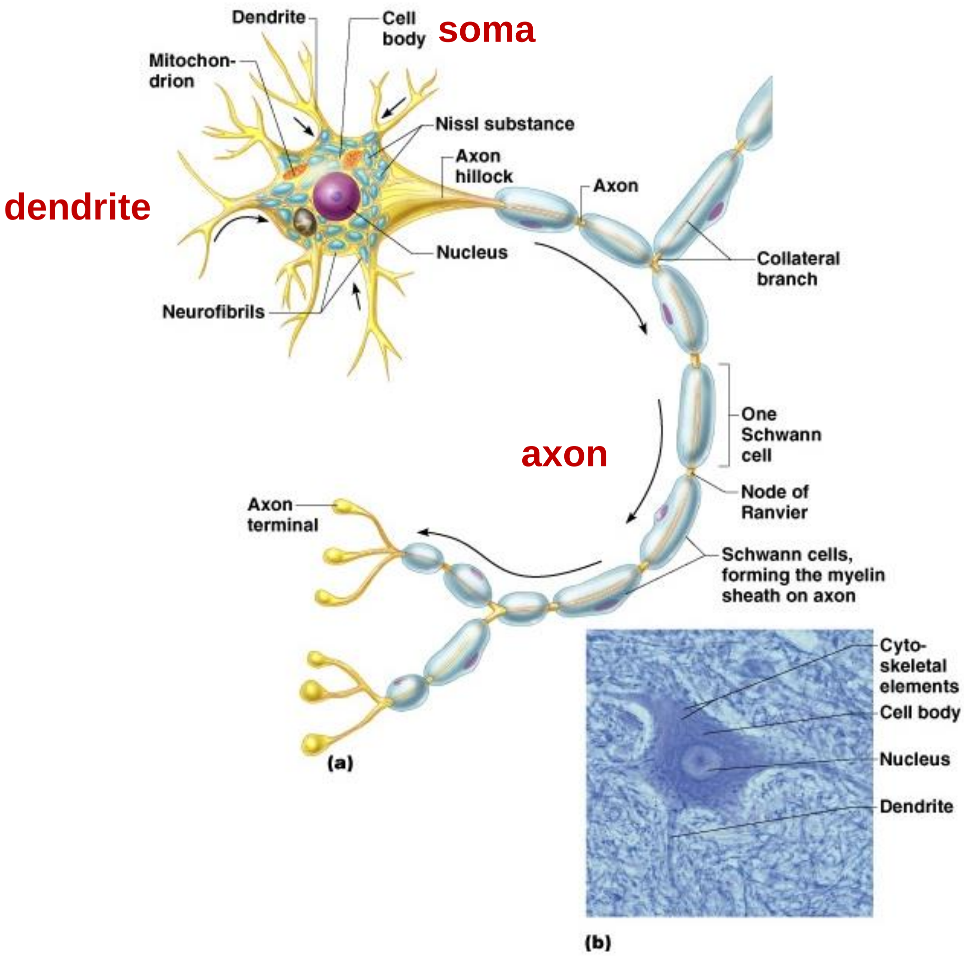
~ 10^{11} neuronas

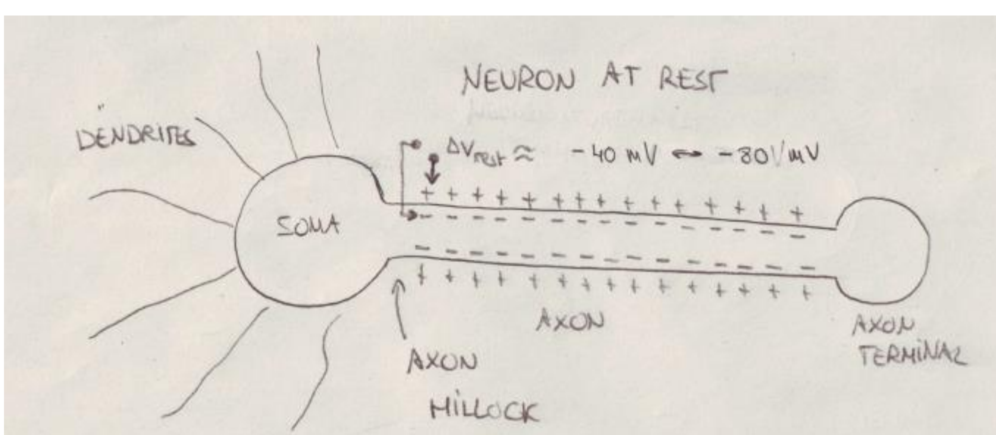
~1000 sinapsis por neurona



Dibujo de Ramón y Cajal de neuronas en el córtex del cerebelo de una paloma.
La letra A señala las células de Purkinje con sus características ramificaciones dendríticas.
[Santiago Ramón y Cajal, 1899. Instituto Santiago Ramón y Cajal, Madrid]

Neuronas biológicas



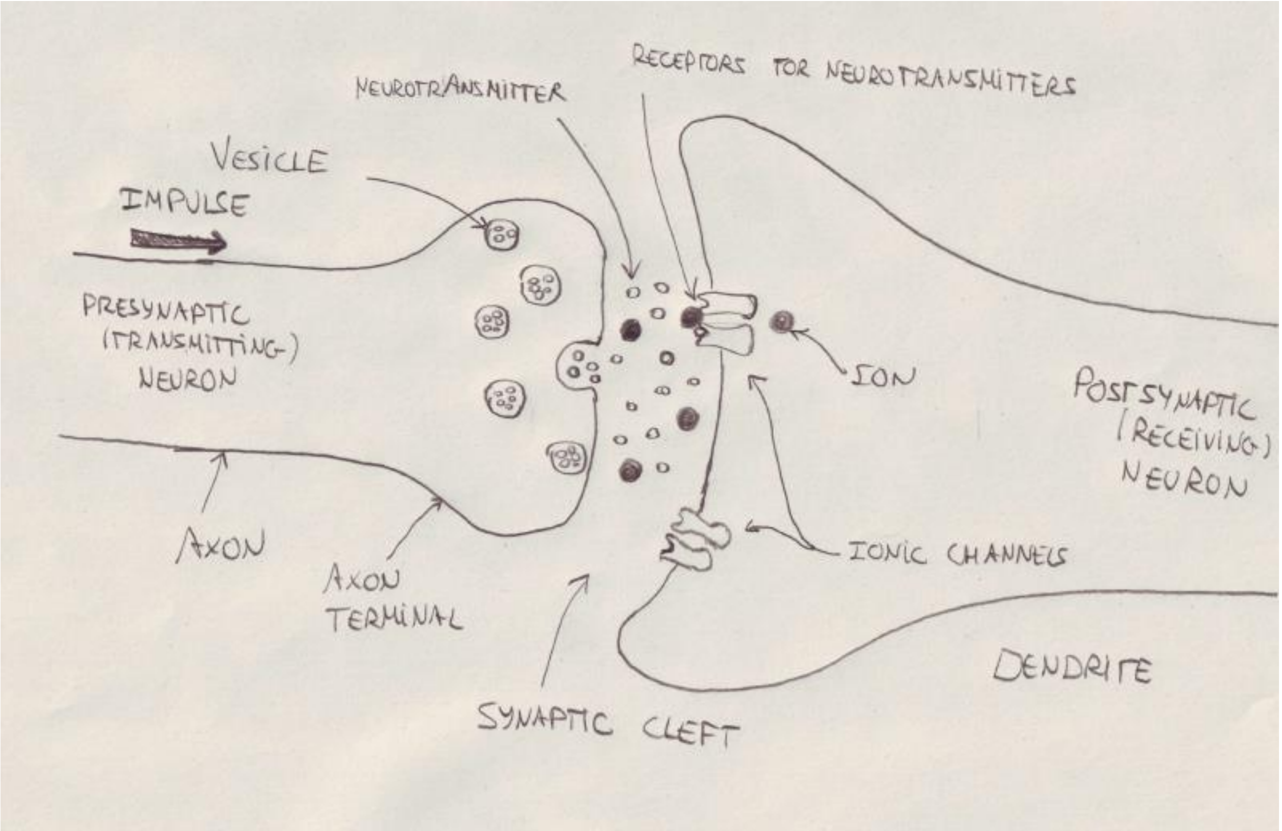


La sinapsis

» **La comunicación entre neuronas** tiene lugar cuando el **potencial de acción** llega al **extremo terminal del axón** de **la neurona presináptica,** causando el flujo de iones de calcio (Ca^{++}) hacia el interior de la célula. Este flujo inicia la liberación de unos compuestos químicos llamados **neurotransmisores** que transportan la señal por el **espacio sináptico** a la **neurona postsináptica.**

» **Los neurotransmisores pueden ser tanto inhibitorios como excitatorios**, y pueden o bien bloquear o bien estimular la generación de un potencial de acción en la neurona postsináptica.

» La respuesta de la neurona postsináptica depende del tipo de señales que (inhibitorias o excitatorias) dominen.



http://science.education.nih.gov/supplements/nih2/addiction/guide/lesson2-1.htm

http://science.education.nih.gov/supplements/nih2/addiction/activities/lesson2_neurotransmission.htm

Rasgos generales de las neuronas biológicas

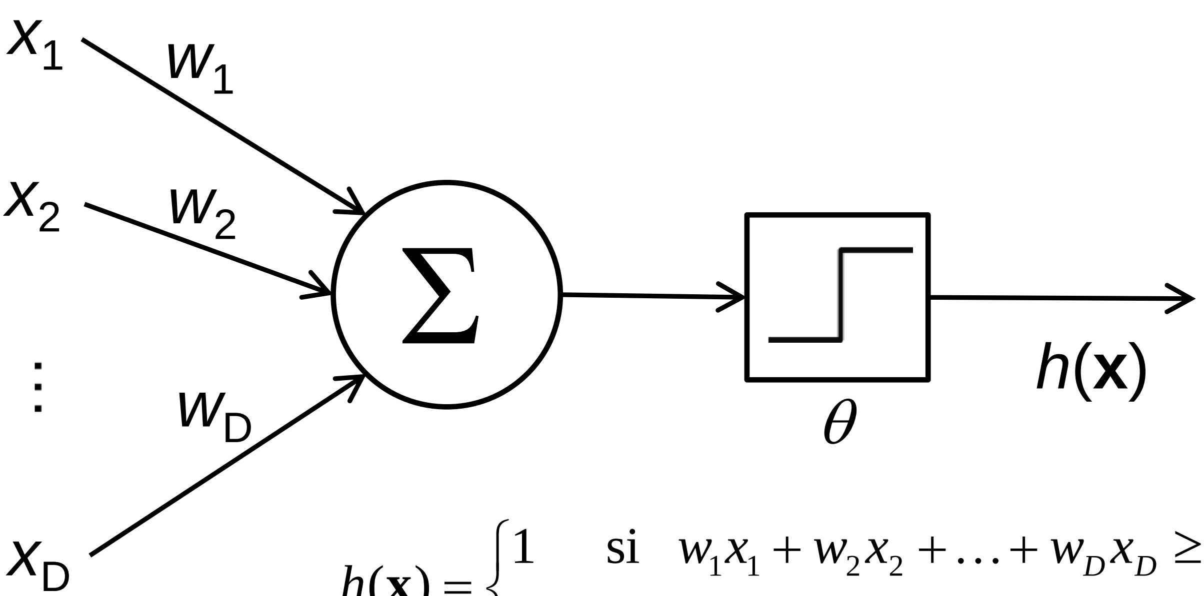
<u>Propiedades de las neuronas biológicas</u>

» Una neurona recibe impulsos (tanto excitatorios como inhibitorios) de distinta intensidad de muchas otras neuronas.
» La neurona integra (suma) los impulsos recibidos en el tiempo y en el espacio (de sus diferentes dendritas).
» Si la señal integrada resultante está por encima de un umbral, la neurona genera un potencial de acción (se dice que la neurona “dispara”).
» La respuesta de la neurona es del tipo “todo o nada”
» El potencial de acción, generado en la parte del axón más próximo al cuerpo de la neurona, es transportado a lo largo del axón hasta el extremo terminal del axón.
» La señal es transmitida a la siguiente neurona en la red a través de la sinapsis mediante neurotransmisores.
» La intensidad de la sinapsis puede modificarse (aprendizaje).

<u>Lecciones para sistemas de reconocimiento de patrones</u>

» **Conexionismo:** Se puede obtener comportamiento complejo a partir de unidades simples que interactúan en una red compleja.
» **Representación del conocimiento distribuida**. Paralelismo.
» Uso de **umbrales** para **clasificación robusta**.
» **Mecanismos de aprendizaje**: se puede aprender modificando los pesos de las conexiones sinápticas.

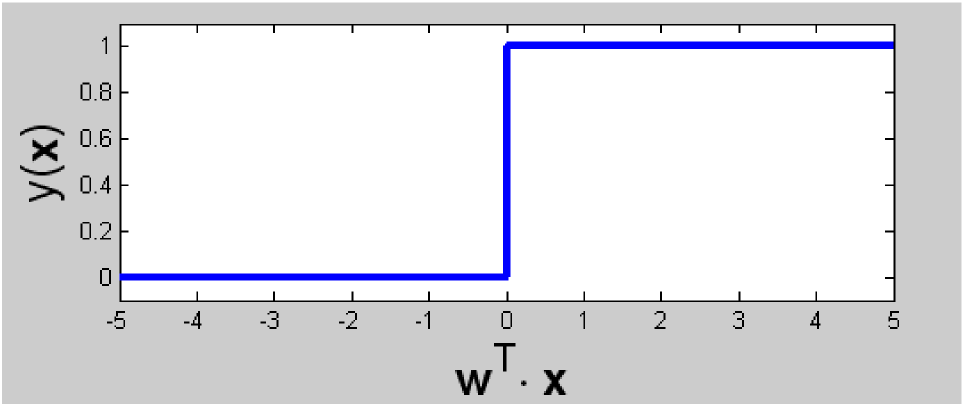
Neurona artificial



$$h(\mathbf{x}) = \begin{cases} 1 & \text{si} \quad w_1 x_1 + w_2 x_2 + \ldots + w_D x_D \geq \theta \\ 0 & \text{si} \quad w_1 x_1 + w_2 x_2 + \ldots + w_D x_D < \theta \end{cases}$$

umbral : θ

En notación vectorial: $$h(\mathbf{x}) = \begin{cases} 1 & \text{si} \quad \mathbf{w}^T \cdot \mathbf{x} \geq \theta \\ 0 & \text{si} \quad \mathbf{w}^T \cdot \mathbf{x} < \theta \end{cases}$$



McCulloch, W. S. y Pitts, W. "A Logical Calculus of Ideas Immanent in Nervous Activity." Bulletin of mathematical biophysics, 5, pp. 115-133 (1943). Reimpreso en McCulloch, W. S., *Embodiments of mind*. Cambridge, MA: MIT Press.

La función de activación sigmoidal (logit)

$$\sigma(z) \equiv \frac{1}{1+e^{-z}}; \quad \textbf{[sigmoidal logística]}$$

Propiedades :

1. $\sigma(-\infty) = 0; \quad \sigma(0) = 0.5; \quad \sigma(\infty) = 1;$

2. Monótona creciente :

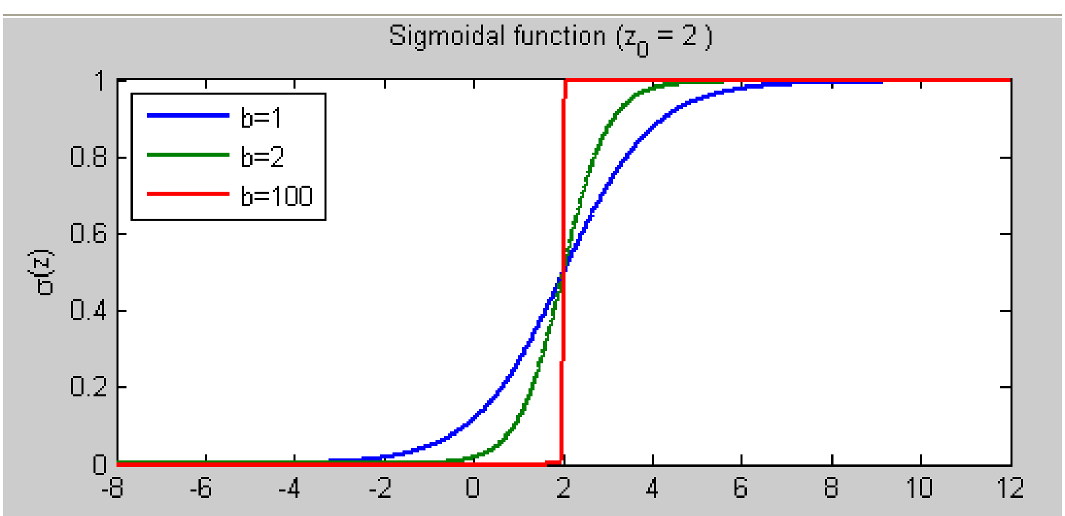
$$z_1 > z_2 \Rightarrow \sigma(z_1) > \sigma(z_1)$$

3. $1 - \sigma(z) = \sigma(-z)$

4. $\sigma'(z) = \dfrac{d\sigma(z)}{dz} = \sigma(z)$

Posición (z_0) **y escala** $(1/b)$ **:** $\quad \sigma(z; z_0, b) \equiv \dfrac{1}{1+e^{-b(z-z_0)}}$

$$\lim_{b\to\infty} \sigma(z; z_0, b) = \theta(z - z_0) \equiv \begin{cases} 1 & \text{si } z > z_0 \\ 0.5 & \text{si } z = z_0 \\ 0 & \text{si } z < z_0 \end{cases}$$

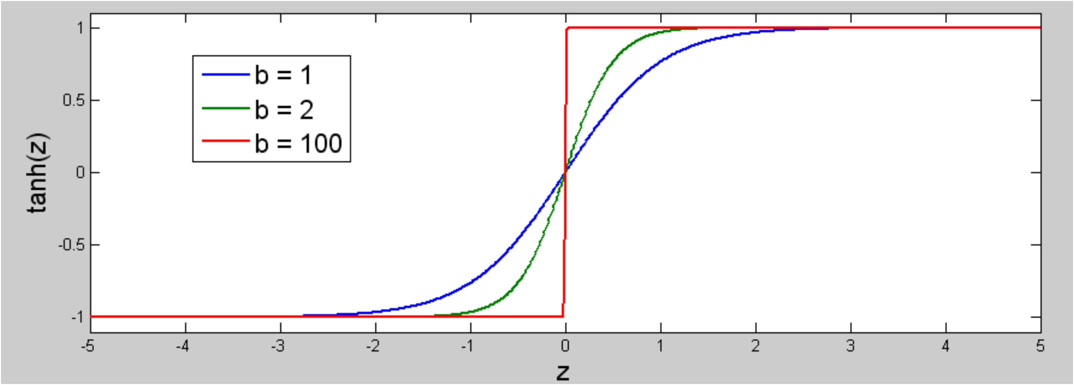


Otras funciones de activación

» **Activación lineal** $f(z) = z$

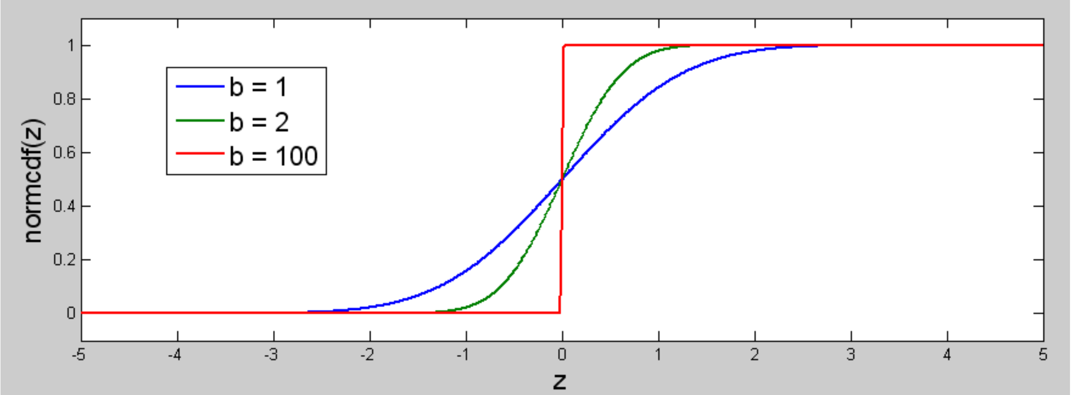
[Neurona de salida para regresión]

» **Tangente hiperbólica** $\tanh(z) = \dfrac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$



» **Función de activación probit**

$$normcdf(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{z} dy \exp\left(-\frac{y^2}{2\sigma^2}\right)$$

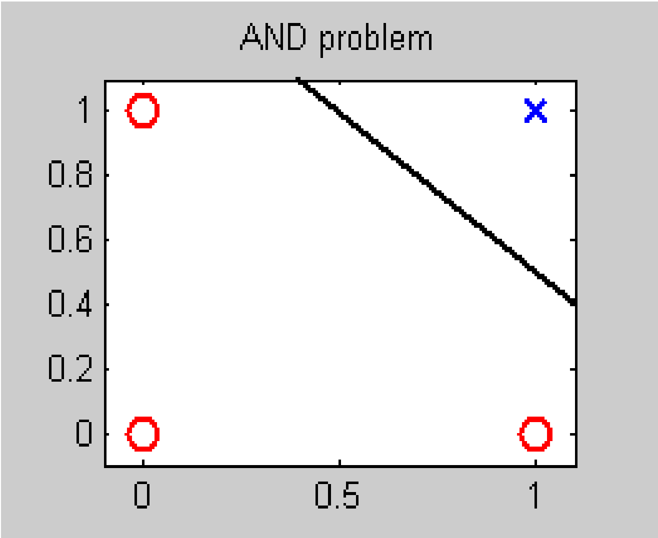


Una neurona como clasificador: Puerta lógica AND

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \qquad h(\mathbf{x}) = \begin{cases} 1 & \text{si} \quad x_1 + x_2 \geq 1.5 \\ 0 & \text{si} \quad x_1 + x_2 < 1.5 \end{cases}$$

$w_1 = 1 \quad w_2 = 1 \quad \theta = 1.5$

x_1	x_2	$w_1x_1 + w_2x_2$	t
0	0	0	0
0	1	1	0
1	0	1	0
1	1	2	1

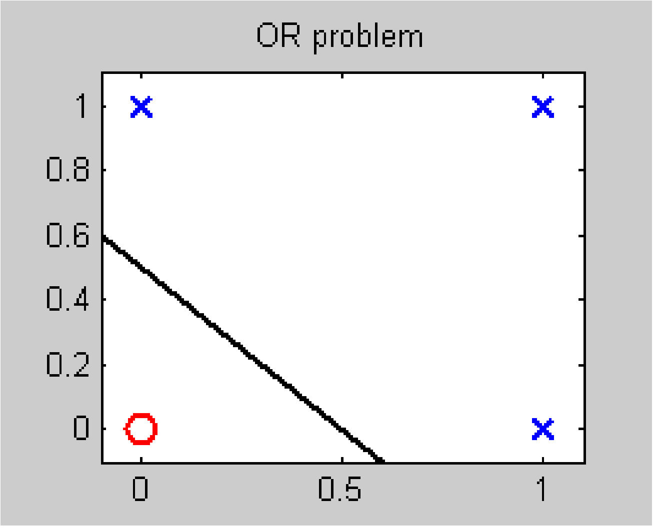


Una neurona como clasificador: Puerta lógica OR

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \qquad h(\mathbf{x}) = \begin{cases} 1 & \text{si} \quad x_1 + x_2 \geq 0.5 \\ 0 & \text{si} \quad x_1 + x_2 < 0.5 \end{cases}$$

$w_1 = 1 \quad w_2 = 1 \quad \theta = 0.5$

x_1	x_2	$w_1x_1 + w_2x_2$	t
0	0	0	0
0	1	1	1
1	0	1	1
1	1	2	1

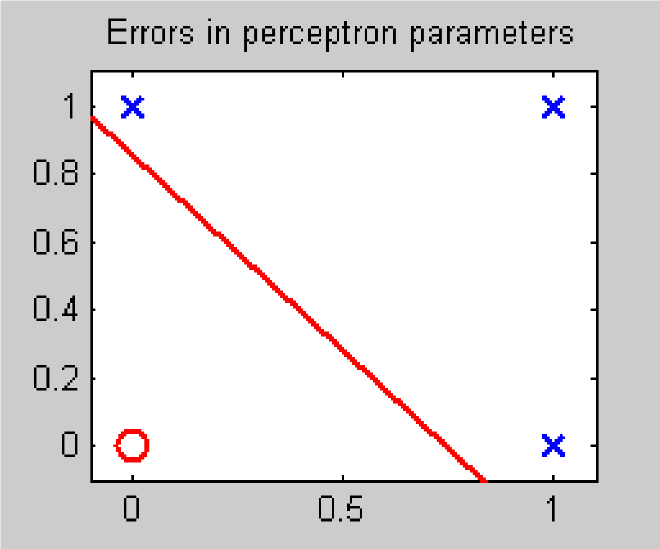


Errores en los parámetros de la neurona

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad h(\mathbf{x}) = \begin{cases} 1 & \text{si } \; 0.8\,x_1 + 0.7x_2 \geq 0.6 \\ 0 & \text{si } \; 0.8\,x_1 + 0.7x_2 < 0.6 \end{cases}$$

$w_1 = 0.8 \quad w_2 = 0.7 \quad \theta = 0.6$

x_1	x_2	$w_1x_1 + w_2x_2$	t
0	0	0	0
0	1	0.7	1
1	0	0.8	1
1	1	1.5	1

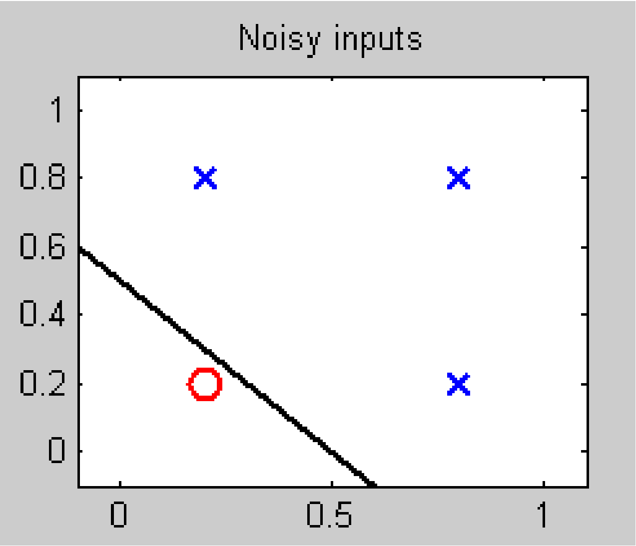


Errores en las entradas

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad h(\mathbf{x}) = \begin{cases} 1 & \text{si} \quad x_1 + x_2 \geq 0.5 \\ 0 & \text{si} \quad x_1 + x_2 < 0.5 \end{cases}$$

$w_1 = 1 \quad w_2 = 1 \quad \theta = 0.5$

x_1	x_2	$w_1x_1 + w_2x_2$	t
0.2	0.2	0.4	0
0.2	0.8	1	1
0.8	0.2	1	1
0.8	0.8	1.6	1

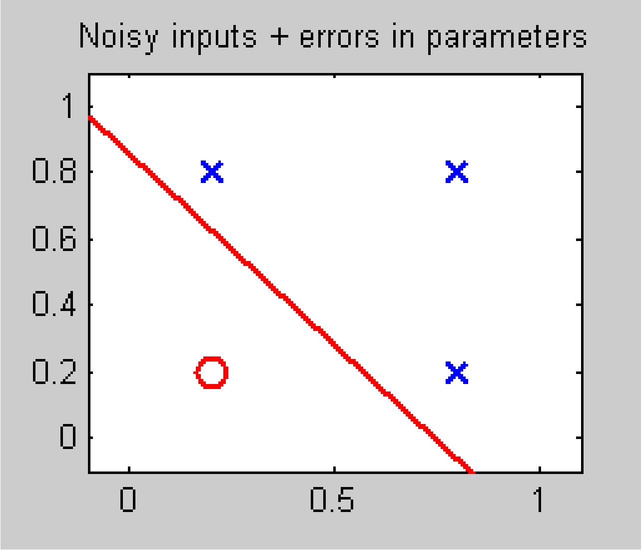


Errores en parámetros y entradas

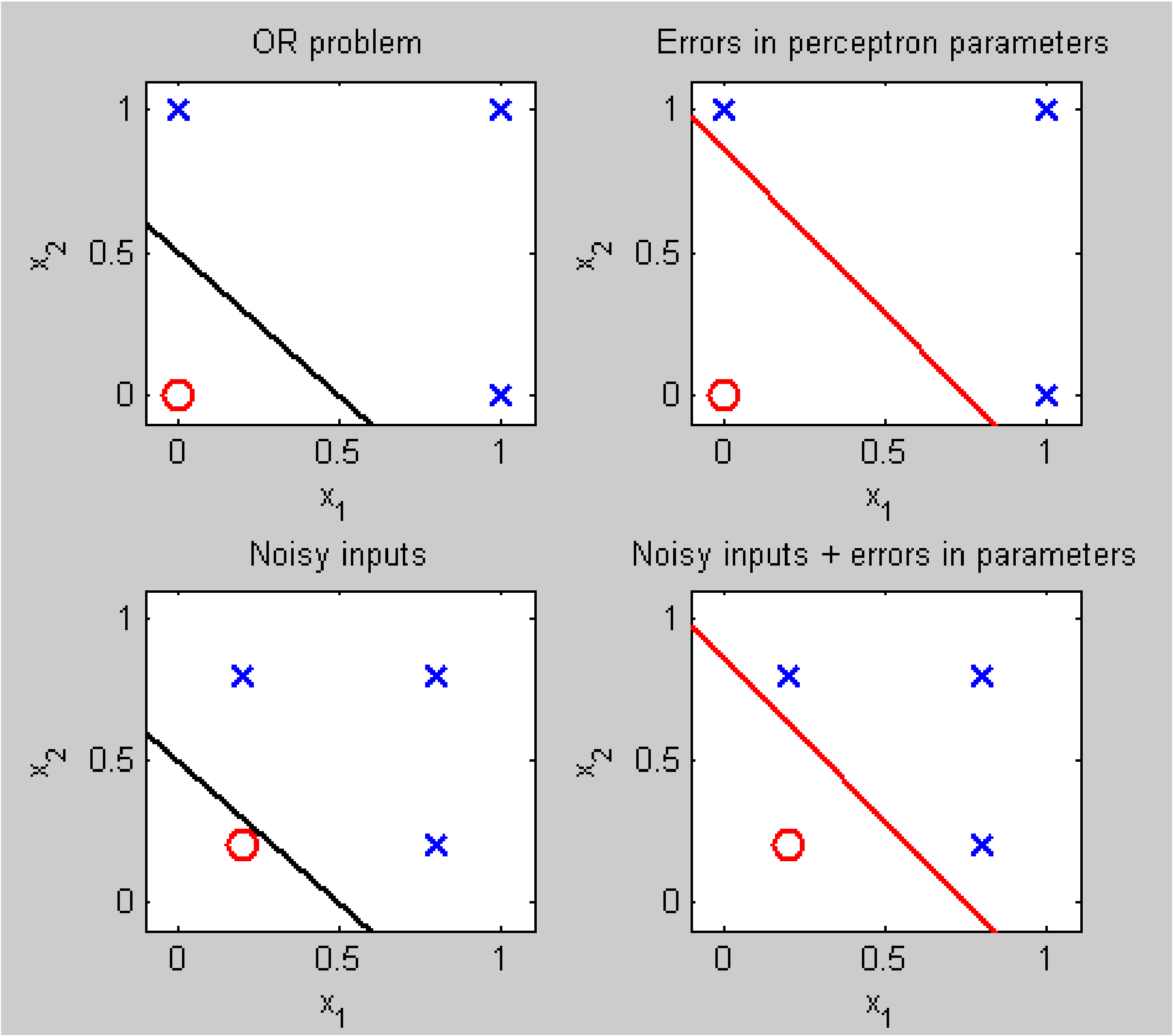
$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad h(\mathbf{x}) = \begin{cases} 1 & \text{si} \quad 0.8x_1 + 0.7x_2 \geq 0.6 \\ 0 & \text{si} \quad 0.8\,x_1 + 0.7x_2 < 0.6 \end{cases}$$

$w_1 = 0.8 \quad w_2 = 0.7 \quad \theta = 0.6$

x_1	x_2	$w_1x_1 + w_2x_2$	t
0.2	0.2	0.3	0
0.2	0.8	0.72	1
0.8	0.2	0.78	1
0.8	0.8	1.2	1



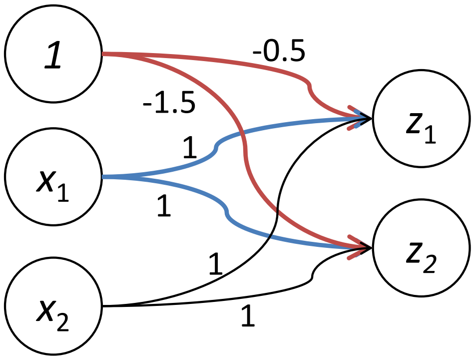
Resumen problema OR



Problema XOR

» El problema XOR no es separable linealmente
» No se puede resolver mediante una sola neurona
» Tampoco se puede resolver mediante una red neuronal de una capa.

» Sin embargo puede ser resuelto mediante una red neuronal de dos capas.
» Consideremos la siguiente construcción



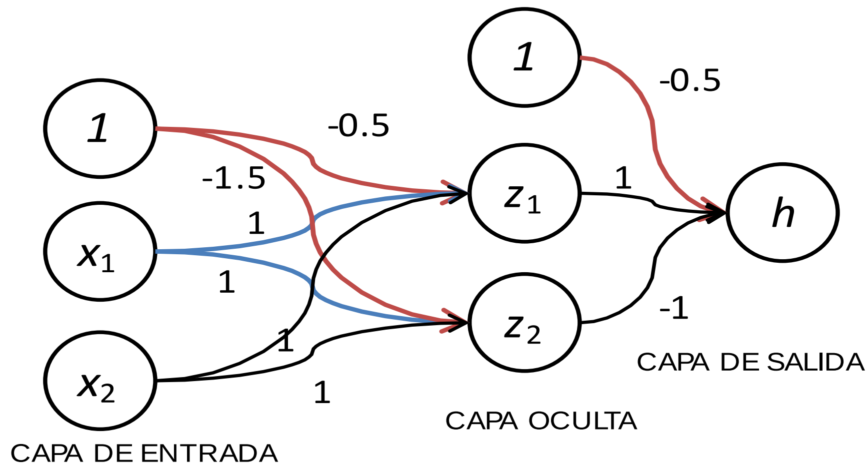
$$z_1 = \theta(x_1 + x_2 - 0.5)$$
$$z_2 = \theta(x_1 + x_2 - 1.5)$$

x_1	x_2	z_1	z_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

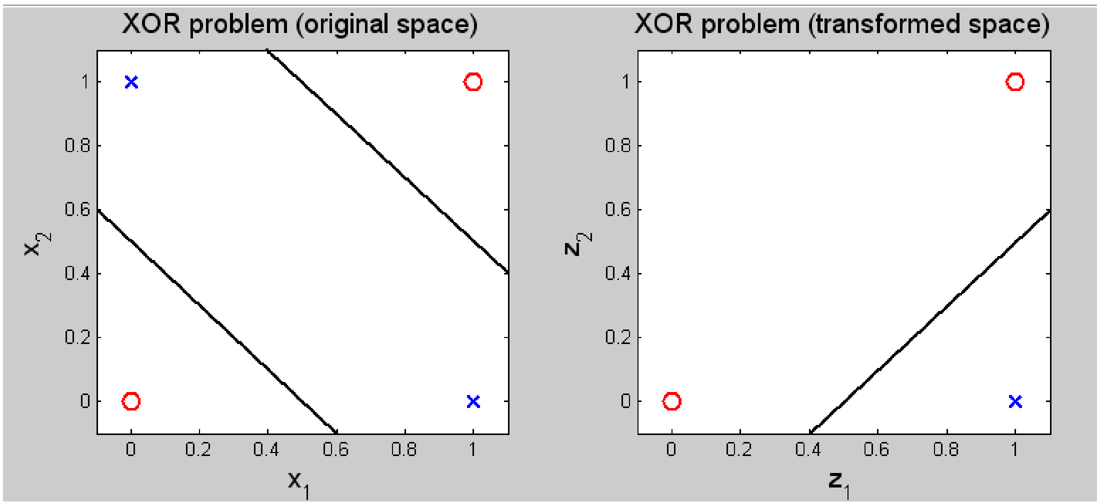
Red neuronal con varias capas para el problema XOR

$$\begin{cases} z_1 = \theta(x_1 + x_2 - 0.5) \\ z_2 = \theta(x_1 + x_2 - 1.5) \end{cases}$$

$$h = \theta(z_1 - z_2 - 0.5)$$



x_1	x_2	z_1	z_2	t
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0



Arquitectura de una red neuronal

- **Una red neuronal** es una **red de neuronas interconectadas con una cierta estructura.**

- **Neurona (unidad de computación)**
  - Conjunto de **conexiones sinápticas** lineales. Cada una de ella sirve es un canal para **una señal de entrada.** Esta entrada puede ser o bien una señal de entrada de la red o la salida de otra neurona en la red.
  - En general siempre hay una conexión sináptica con una **señal de entrada constante igual a +1** (**sesgo**)
  - Las conexiones sinápticas **ponderan** las señales de entrada correspondientes **multiplicándolas por un peso**
  - La **suma ponderada de las señales de entrada** define un **campo local.**
  - La neurona genera una **salida** que es el resultado de procesar el campo local mediante una **función de activación**.
  - La salida de la neurona puede ser utilizada como entrada de otra neurona o como salida final de la red neuronal.

- La **arquitectura de una red** está determinada por la forma en que las neuronas están conectadas entre sí.
  La red puede ser vista como un grafo dirigido compuesto por
  - **Nodos fuente**: Proporcionan la señal de entrada a la red.
  - **Nodos de computación**: neuronas.
  - **Enlaces dirigidos** que indican el sentido en el que la señal es procesada (se propaga) en el grafo.

Predicción de una red neuronal

- Supongamos que queremos **predecir** el valor de una **propiedad específica** para un ejemplo concreto **a partir de** los valores (conocidos) de otras propiedades, recogidos en **x**, el **vector de atributos** que caracteriza al ejemplo en cuestión

- La red neuronal recibe el vector de atributos D-dimensional **x** como **señal de entrada** a través de los nodos fuente.

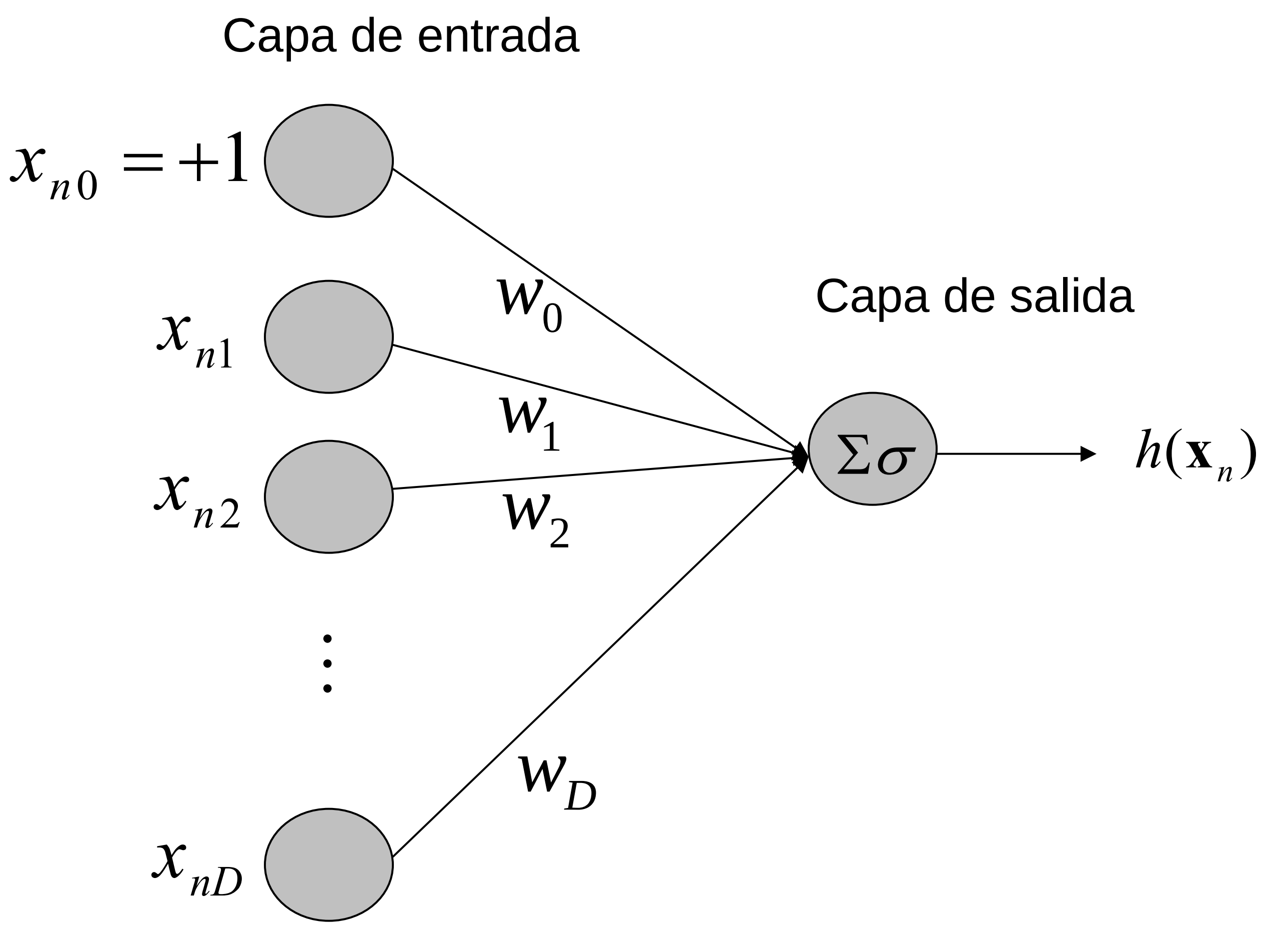
- La señal de entrada es **procesada** por la red, la cual genera, una vez finalizado dicho procesamiento, una **salida**.

- La **predicción de la red neuronal** es una interpretación de la **salida** de una neurona o de un grupo de neuronas en **la red**.

» **Regresión**:
- Regresión escalar: La salida es un número real
- Regresión vectorial: Se registran los números reales correspondientes a la salida de varias neuronas.

» **Clasificación**:
- Clasificación binaria [**Una neurona de salida**]
  - Puntuación / estimación de probabilidad de la clase.
  - Etiqueta de clase: 1 (clase 1) / 0 (clase 2)
- Problemas multiclase (K clases) [**K neuronas de salida**]
  - Vector de puntuaciones / probabilidades de clase
  - Codificación uno de K.

Regresión logística mediante un perceptrón lineal



Campo local: $\mathbf{w}^T \cdot \mathbf{x}_n = \sum_{d=0}^{D} w_d x_{nd}$

Función de activación: $\sigma(z) = \frac{1}{1+e^{-z}}$

Salida: $h(\mathbf{x}_n) = \sigma\left(\mathbf{w}^T \cdot \mathbf{x}_n\right)$

Clasificación binaria (2 clases)

$$\text{Datos de entrenamiento}: \left\{(\mathbf{x}_n, t_n); \ n = 1,2,\ldots,N\right\}; \quad t_n = \begin{cases} 1 \ \text{si} \ c_n = C_1 \\ 0 \ \text{si} \ c_n = C_2 \end{cases}$$

n	x_{n1}	x_{n2}	...	x_{nD}	t_n
1	2.3	0	...	10.3	0
2	2.5	1	...	13.1	1
3	2.6	0	...	-2.7	1
4	2.7	-1	...	-5.4	0
5	2.9	0	...	2.1	1
6	3.1	0	...	-10.9	0

Aprendizaje por máxima verosimilitud

Probabilidad posterior: $P\left(C_1 \mid \mathbf{x}, \mathbf{w}\right) = h(\mathbf{x})$

Función de verosimilitud:

$$\textbf{Datos}: \left\{\left(\mathbf{x}_n, t_n\right); n = 1,2,\ldots,N\right\}; \quad t_n = \begin{cases} 1 & \text{si } c_n = C_1 \\ 0 & \text{si } c_n = C_2 \end{cases}$$

$$\mathbf{w}_{ML} = \arg\max_{\mathbf{w}} \left\{L\left(\mathbf{w}; \left\{\left(\mathbf{x}_n, t_n\right)\right\}_{n=1}^{N}\right)\right\}$$

Verosimilitud

$$L\left(\mathbf{w}; \left\{\left(\mathbf{x}_n, t_n\right)\right\}_{n=1}^{N}\right) \equiv P\left(\left\{t_n\right\}_{n=1}^{N} \mid \left\{\mathbf{x}_n\right\}_{n=1}^{N}, \mathbf{w}\right) =$$

$$\prod_{n=1}^{N} \left[P(C_1 \mid \mathbf{x}_n, \mathbf{w})\right]^{t_n} \cdot \left[1 - P(C_1 \mid \mathbf{x}_n, \mathbf{w})\right]^{(1-t_n)} =$$

$$\prod_{n=1}^{N} \left[h(\mathbf{x}_n)\right]^{t_n} \cdot \left[1 - h(\mathbf{x}_n)\right]^{(1-t_n)}$$

Funcióndeerror de entropía cruzada

$$E(\mathbf{w}) \equiv -\log L\left(\mathbf{w}; \left\{\left(\mathbf{x}_n, t_n\right)\right\}_{n=1}^{N}\right) = \sum_{n=1}^{N} E_n(\mathbf{w})$$

$$E_n(\mathbf{w}) \equiv -t_n \log h(\mathbf{x}_n) - (1 - t_n) \log\left(1 - h(\mathbf{x}_n)\right)$$

$$\textbf{Aprendizaje ML} = \textbf{Minimizar } E(\mathbf{w})$$

Aprendizaje por lotes (*batch*)

$$E(\mathbf{w}) \equiv -\sum_{n=1}^{N}\left\{t_n \log P(C_1 \mid \mathbf{x}_n, \mathbf{w}) + (1-t_n)\log P(C_2 \mid \mathbf{x}_n, \mathbf{w})\right\}$$

$$\begin{cases} P(C_1 \mid \mathbf{x},\mathbf{w}) = h(\mathbf{x}) \equiv \sigma(\mathbf{w}^T \cdot \mathbf{x}) \\ P(C_2 \mid \mathbf{x},\mathbf{w}) = 1 - h(\mathbf{x}) \end{cases}$$

$$\text{Maximizar } \textbf{Log - verosimilitud} = \text{ Minimizar } E(\mathbf{w})$$

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \sum_{n=1}^{N}\big(h(\mathbf{x}_n) - t_n\big)\mathbf{x}_n$$

Nombre: **Aprendizaje por lotes para regresión logística**

Entrada: Datos de entrenamiento: $\{(\mathbf{x}_n, t_n); n = 1,2,\ldots,N\}$; $t_n = \begin{cases} 1 \text{ si } c_n = C_1 \\ 0 \text{ si } c_n = C_2 \end{cases}$

Parámetro de aprendizaje: η

Salida: Parámetros del modelo (pesos de la red) $\mathbf{w}_{ML}$

Código:

1. Inicializa los pesos aleatoriamente $\mathbf{w} \sim U[-0.5, 0.5]^{(D+1)}$
2. Mientras no se cumplan los criterios de convergencia

   2.1 Calcular salida de red para ejemplos de entrenamiento

   $$h(\mathbf{x}_n) = \sigma\big(\mathbf{w}^T \cdot \mathbf{x}_n\big), \quad n = 1,2,\ldots,N$$

   2.2 Calcular el gradiente: $\displaystyle \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \sum_{n=1}^{N}\big(h(\mathbf{x}_n) - t_n\big)\mathbf{x}_n$

   2.3 Actualizar los pesos: $\displaystyle \mathbf{w} = \mathbf{w} - \eta \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}$

Aprendizaje en línea (*online*)

$$E(\mathbf{w}) \equiv -\sum_{n=1}^{N}\left\{t_n \log P(C_1 \mid \mathbf{x}_n, \mathbf{w}) + (1-t_n)\log P(C_2 \mid \mathbf{x}_n, \mathbf{w})\right\}$$

$$\begin{cases} P(C_1 \mid \mathbf{x}, \mathbf{w}) = h(\mathbf{x}) \equiv \sigma(\mathbf{w}^T \cdot \mathbf{x}) \\ P(C_2 \mid \mathbf{x}, \mathbf{w}) = 1 - h(\mathbf{x}) \end{cases}$$

$$\text{Maximizar } \textbf{Log-verosimilitud} \ = \ \text{Minimizar } E(\mathbf{w})$$

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \sum_{n=1}^{N}\big(h(\mathbf{x}_n) - t_n\big)\mathbf{x}_n$$

Nombre: **Aprendizaje en línea para regresión logística**

Entrada: Datos de entrenamiento : $\{(\mathbf{x}_n, t_n); n = 1,2,\ldots,N\}$; $\quad t_n = \begin{cases} 1 \text{ si } c_n = C_1 \\ 0 \text{ si } c_n = C_2 \end{cases}$

Parámetro de aprendizaje : η

Salida: Parámetros del modelo (pesos de la red) $\mathbf{w}_{\text{ML}}$

Código:

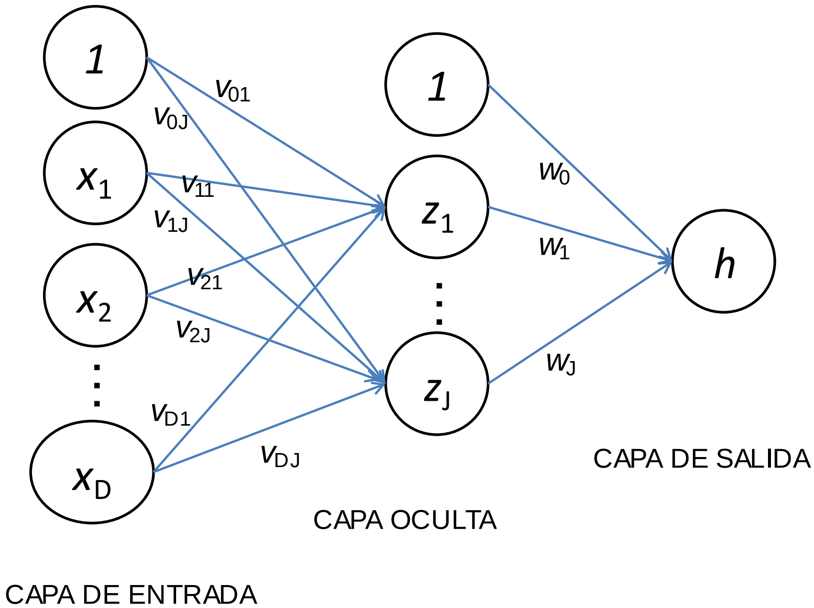
1. Inicializar los pesos aleatoriamente $\quad \mathbf{w} \sim U[-0.5, 0.5]^{(D+1)}$
2. Initializar el contador de épocas $\quad \mathrm{k} = 0$
3. Mientras no se cumplan los criterios de convergencia [**épocas**]
   - 3.1 Actualiza el contador de épocas $\quad \mathrm{k} = \mathrm{k} + 1$
   - 3.2 For $\mathrm{n} = 1,2,\ldots,\mathrm{N}$

     $h(\mathbf{x}_n) = \sigma\big(\mathbf{w}^T \cdot \mathbf{x}_n\big)$ [probabilidad posterior de C_1]

     $\mathbf{w} = \mathbf{w} - \eta\big(h(\mathbf{x}_n) - t_n\big)\mathbf{x}_n$

Perceptrón multicapa para clasificación binaria

Perceptrón multicapa



$$\mathbf{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_D \end{pmatrix}$$

Parámetros de la red: $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_J) \quad [(D+1) \times J]$

$\mathbf{w} \quad [(J+1) \times 1]$

v_{dj} : Peso entre las neuronas d (entrada) y j (oculta)

w_j : Peso entre las neuronas j (oculta) y la de salida

$x_0 = 1, \quad z_0 = 1$

$$\mathbf{z} = \begin{pmatrix} 1 \\ z_1 \\ z_2 \\ \vdots \\ z_D \end{pmatrix}$$

$$z_j = \sigma\left(\sum_{d=0}^{D} x_d v_{dj}\right) = \sigma\left(\mathbf{v}_j^T \cdot \mathbf{x}\right), j = 1, 2, \ldots, J$$

$$h(\mathbf{x}) = \sigma\left(\sum_{j=0}^{J} w_j z_j\right) = \sigma\left(\mathbf{w}^T \cdot \mathbf{z}\right)$$

Probabilidad posterior: $P(C_1 \mid \mathbf{x}, \mathbf{V}, \mathbf{w}) = h(\mathbf{x})$

Clasificación binaria

$$\text{Datos de entrenamiento}: \{(\mathbf{x}_n, t_n); n = 1,2,\ldots,N\}; \quad t_n = \begin{cases} 1 \ \text{si}\ c_n = C_1 \\ 0 \ \text{si}\ c_n = C_2 \end{cases}$$

n	x_{n1}	x_{n2}	...	x_{nD}	t_n
1	2.3	0	...	10.3	0
2	2.5	1	...	13.1	1
3	2.6	0	...	-2.7	1
4	2.7	-1	...	-5.4	0
5	2.9	0	...	2.1	1
6	3.1	0	...	-10.9	0

Aprendizaje por máxima verosimilitud

Probabilidad posterior: $P\left(C_1 \mid \mathbf{x}, \mathbf{V}, \mathbf{w}\right) = h(\mathbf{x})$

Función de verosimilitud:

$$\textbf{Datos}: \quad \left\{\left(\mathbf{x}_n, t_n\right); n = 1,2,\ldots,N\right\}; \quad t_n = \begin{cases} 1 & \text{si } c_n = C_1 \\ 0 & \text{si } c_n = C_2 \end{cases}$$

$$\begin{pmatrix} \mathbf{V}_{ML} \\ \mathbf{w}_{ML} \end{pmatrix} = \underset{\mathbf{V},\mathbf{w}}{\arg\max} \left\{ L\left(\mathbf{V}, \mathbf{w}; \left\{\left(\mathbf{x}_n, t_n\right)\right\}_{n=1}^{N}\right) \right\}$$

Verosimilitud

$$L\left(\mathbf{V}, \mathbf{w}; \left\{\left(\mathbf{x}_n, t_n\right)\right\}_{n=1}^{N}\right) \equiv P\left(\left\{t_n\right\}_{n=1}^{N} \mid \left\{\mathbf{x}_n\right\}_{n=1}^{N}, \mathbf{V}, \mathbf{w}\right) =$$

$$\prod_{n=1}^{N} \left[P(C_1 \mid \mathbf{x}_n, \mathbf{V}, \mathbf{w})\right]^{t_n} \cdot \left[1 - P(C_1 \mid \mathbf{x}_n, \mathbf{V}, \mathbf{w})\right]^{(1-t_n)} =$$

$$\prod_{n=1}^{N} \left[h(\mathbf{x}_n)\right]^{t_n} \cdot \left[1 - h(\mathbf{x}_n)\right]^{(1-t_n)}$$

Función de error de entropía cruzada

$$E(\mathbf{V}, \mathbf{w}) \equiv -\log L\left(\mathbf{V}, \mathbf{w}; \left\{\left(\mathbf{x}_n, t_n\right)\right\}_{n=1}^{N}\right) = \sum_{n=1}^{N} E_n(\mathbf{V}, \mathbf{w})$$

$$E_n(\mathbf{V}, \mathbf{w}) \equiv -t_n \log h(\mathbf{x}_n) - (1 - t_n) \log\left(1 - h(\mathbf{x}_n)\right)$$

Aprendizaje ML = Minimizar $E(\mathbf{V}, \mathbf{w})$

Propagación hacia atrás, online

Nombre: **Propagación hacia atrás (clasificación binaria)**

Entrada: Datos de entrenamiento : $\{(\mathbf{x}_n, t_n); n = 1,2,\ldots,N\}$; $\quad t_n = \begin{cases} 1 \text{ si } c_n = C_1 \\ 0 \text{ si } c_n = C_2 \end{cases}$

Parámetro de aprendizaje : η

Salida: Pesos del perceptrón multicapa **V**, **w**

Código:

1. Inicializar de manera aleatoria los pesos $\quad \mathbf{V}, \mathbf{w} \sim U[-0.5, 0.5]$
2. Inicializar el contador de épocas $\quad$ nÉpoca = 0
3. Mientras no se cumplan los criterios de convergencia $\quad$ [**épocas**]
   - 3.1 Actualizar el contador de épocas nÉpoca = nÉpoca + 1
   - 3.2 For n = 1,2,…, N

$$z_{nj} = \sigma\left(\mathbf{v}_j^T \cdot \mathbf{x}_n\right), \quad j = 1,2,\ldots,J$$

$$h(\mathbf{x}_n) = \sigma\left(\mathbf{w}^T \cdot \mathbf{z}_n\right) \quad \text{[probabilidad de clase posterior]}$$

$$\delta_n = h(\mathbf{x}_n) - t_n \quad \text{[error de predicción]}$$

$$\Delta_{nj} = z_{nj}\left(1 - z_{nj}\right) w_j \delta_n, \quad j = 1,2,\ldots,J \quad \text{[asigna error a ocultas]}$$

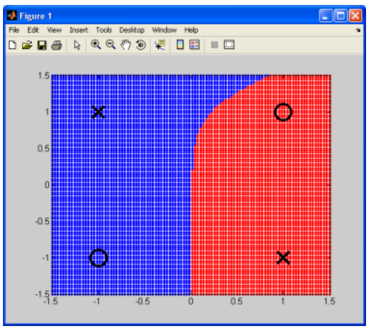
$$\mathbf{w} = \mathbf{w} - \eta \delta_n \mathbf{z}_n \quad \text{[actualizar pesos de oculta a salida]}$$

$$\mathbf{v}_j = \mathbf{v}_j - \eta \Delta_{nj} \mathbf{x}_n, \quad j = 1,2,\ldots,J$$

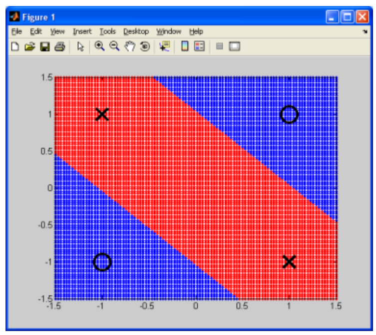
[actualizar pesos de entrada a ocultas]

Perceptrón multicapa para el problema XOR

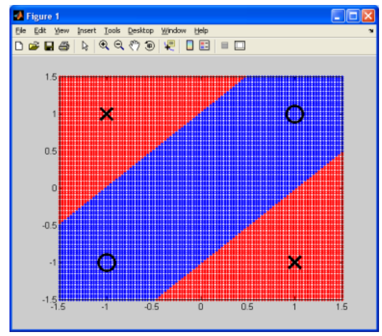
Perceptrón multicapa J = 2 (2 neuronas en la capa oculta)



No convergido

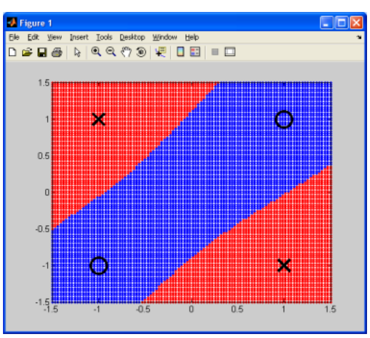


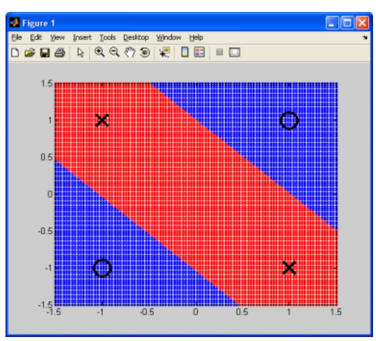
Solución 1

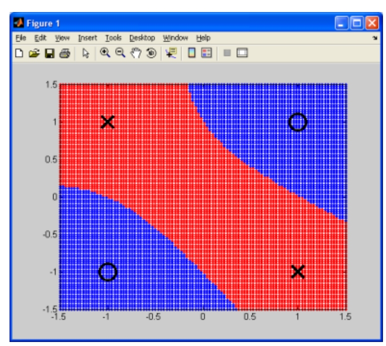


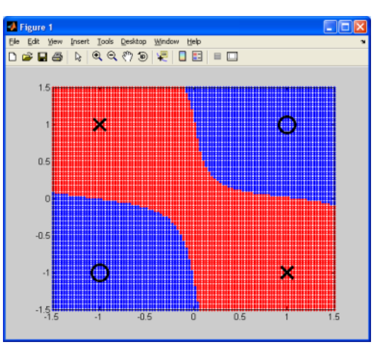
Solución 2

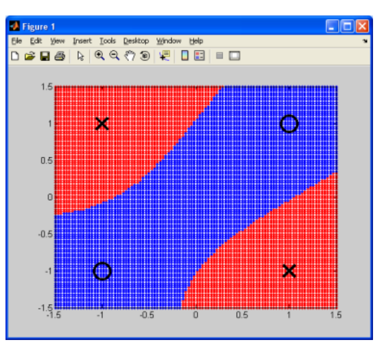
Perceptrón multicapa J = 5 (5 neuronas en la capa oculta)

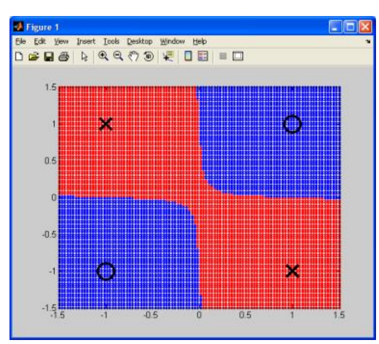












Otras referencias

Discovery Channel
http://www.dnatube.com/video/1298/Neurons-and-How-They-Work

Activación de una neurona
http://www.youtube.com/watch?v=G9rHAM0gIn8&feature=related

Entrenamiento de una red neuronal
http://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=gauss&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=4,2&seed=0.81663&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=true&xSquared=true&ySquared=true&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification

Material adicional

Operaciones con matrices

$\mathbf{A}$ $[M \times N$ matriz$]$

$\mathbf{B}$ $[N \times K$ matriz$]$

$\mathbf{u}$ $[N \times 1$ vector columna$]$

$\mathbf{z}^T$ $[1 \times N$ vector fila$]$

$\mathbf{v}^T$ $[1 \times M$ vector fila$]$

$$\mathbf{z}^T \cdot \mathbf{u} = \sum_{i=1}^{N} z_i u_j \qquad [\text{escalar}]$$

$$\mathbf{v}^T \cdot \mathbf{A} \cdot \mathbf{u} = Tr\left\{\mathbf{A} \cdot \left(\mathbf{u}\mathbf{v}^T\right)\right\} = \sum_{i=1}^{M}\sum_{j=1}^{N} v_i A_{ij} u_j \quad [\text{escalar }]$$

$\mathbf{A} \cdot \mathbf{u}$ $\quad [M \times 1$ vector columna : $\left(\mathbf{A} \cdot \mathbf{u}\right)_i = \sum_{j=1}^{N} A_{ij} u_j;$

$\qquad \mathrm{i} = 1,2,\ldots,\mathrm{M}\ ]$

$\mathbf{v}^T \cdot \mathbf{A}$ $\quad [1 \times N$ vector fila : $\left(\mathbf{v}^T \cdot \mathbf{A}\right)_j = \sum_{i=1}^{M} v_i A_{ij};$

$\qquad \mathrm{j} = 1,2,\ldots,\mathrm{N}\ ]$

$\mathbf{u}\mathbf{v}^T$ $\quad [N \times M$ matriz : $\left(\mathbf{u}\mathbf{v}^T\right)_{ij} = u_i v_j;$

$\qquad \mathrm{i} = 1,2,\ldots,\mathrm{N};\ \mathrm{j} = 1,2,\ldots,\mathrm{M}]$

$\mathbf{A} \cdot \mathbf{B}$ $\quad [M \times K$ matriz : $\left(\mathbf{A} \cdot \mathbf{B}\right)_{ij} = \sum_{l=1}^{M} A_{il} B_{lj}$

$\qquad \mathrm{i} = 1,2,\ldots,\mathrm{M};\ \mathrm{j} = 1,2,\ldots,\mathrm{K}]$

Aprendizaje por máxima verosimilitud (ML) [2 clases]

Derivación del gradiente para el perceptrón multicapa, aprendizaje por ML [2 clases]

$$\text{Datos}: \quad \left\{ \left(\mathbf{x}_n, t_n \right); n = 1,2,\ldots, N \right\}; \quad t_n = \begin{cases} 1 & \text{si } c_n = C_1 \\ 0 & \text{si } c_n = C_2 \end{cases}$$

$$\mathbf{w}_{ML} = \underset{\mathbf{w}}{\operatorname{argmax}} \left\{ L\left(\mathbf{w};\ \left\{ \left(\mathbf{x}_n, t_n \right) \right\}_{n=1}^{N} \right) \right\}$$

Verosimilitud

$$L\left(\mathbf{w}; \left\{ \left(\mathbf{x}_n, t_n \right) \right\}_{n=1}^{N} \right) \equiv P\left(\left\{ t_n \right\}_{n=1}^{N} \mid \left\{ \mathbf{x}_n \right\}_{n=1}^{N}, \mathbf{w} \right) =$$

$$\prod_{n=1}^{N} \left[P(C_1 \mid \mathbf{x}_n, \mathbf{w}) \right]^{t_n} \cdot \left[P(C_2 \mid \mathbf{x}_n, \mathbf{w}) \right]^{(1-t_n)}$$

Log - verosimilitud

$$LL\left(\mathbf{w}; \left\{ \left(\mathbf{x}_n, t_n \right) \right\}_{n=1}^{N} \right) \equiv \log P\left(\left\{ t_n \right\}_{n=1}^{N} \mid \left\{ \mathbf{x}_n \right\}_{n=1}^{N}, \mathbf{w} \right) =$$

$$\sum_{n=1}^{N} \left\{ t_n \log P(C_1 \mid \mathbf{x}_n, \mathbf{w}) + (1 - t_n) \log P(C_2 \mid \mathbf{x}_n, \mathbf{w}) \right\}$$

Función de error de entropía cruzada

$$E(\mathbf{w}) \equiv -\sum_{n=1}^{N} \left\{ t_n \log P(C_1 \mid \mathbf{x}_n, \mathbf{w}) + (1 - t_n) \log P(C_2 \mid \mathbf{x}_n, \mathbf{w}) \right\}$$

Aprendizaje por máxima verosimilitud $=$ **Minimizar error** $E(\mathbf{w})$

Cálculo del gradiente

$$E_n(\mathbf{V},\mathbf{w}) \equiv -t_n \log h(\mathbf{x}_n) - (1-t_n)\log\big(1-h(\mathbf{x}_n)\big) \Rightarrow$$

$$\frac{\partial E_n(\mathbf{V},\mathbf{w})}{\partial \mathbf{b}} = \left\{-t_n \frac{1}{h(\mathbf{x}_n)} + (1-t_n)\frac{1}{1-h(\mathbf{x}_n)}\right\}\frac{\partial h(\mathbf{x}_n)}{\partial \mathbf{b}} =$$

$$= \big(h(\mathbf{x}_n) - t_n\big)\frac{\partial\big(\mathbf{w}^T \cdot \mathbf{z}_n\big)}{\partial \mathbf{b}} = \delta_n \frac{\partial\big(\mathbf{w}^T \cdot \mathbf{z}_n\big)}{\partial \mathbf{b}}$$

Hemos usado: $h(\mathbf{x}_n) = \sigma\big(\mathbf{w}^T \cdot \mathbf{z}_n\big) \Rightarrow \dfrac{\partial h(\mathbf{x}_n)}{\partial \mathbf{b}} = h(\mathbf{x}_n)\big(1-h(\mathbf{x}_n)\big)\dfrac{\partial h(\mathbf{w}^T \cdot \mathbf{z}_n)}{\partial \mathbf{b}}$

y hemos definido : $\delta_n \equiv h(\mathbf{x}_n) - t_n$

Derivadas con respecto a los pesos de oculta a salida (**β → w**)

$$\frac{\partial}{\partial \mathbf{w}} E_n(\mathbf{V},\mathbf{w}) = \delta_n\, \mathbf{z}_n$$

Derivadas con respecto a los pesos de entrada a oculta (**β → v**$_j$)

$$\frac{\partial E_n(\mathbf{V},\mathbf{W})}{\partial \mathbf{v}_j} = \delta_n \frac{\partial\big(\mathbf{w}^T \cdot \mathbf{z}_n\big)}{\partial \mathbf{v}_j} = \delta_n \frac{\partial\left(\sum_{i=0}^{J} w_i\, z_{ni}\right)}{\partial \mathbf{v}_j} =$$

$$= \delta_n \sum_{i=0}^{J} w_i \frac{\partial z_{ni}}{\partial \mathbf{v}_j} = \delta_n w_j \frac{\partial z_{nj}}{\partial \mathbf{v}_j} = z_{nj}\big(1-z_{nj}\big)\delta_n w_j\, \mathbf{x}_n$$

$$z_{ni} = \sigma\big(\mathbf{v}_i^T \cdot \mathbf{x}_n\big) \Rightarrow \begin{cases} \dfrac{\partial z_{ni}}{\partial \mathbf{v}_j} = 0 & i \neq j; \\ & ; \qquad j = 1,2,\ldots,J \\ \dfrac{\partial z_{nj}}{\partial \mathbf{v}_j} = z_{nj}\big(1-z_{nj}\big)\mathbf{x}_n & \end{cases}$$

Aprendizaje por máxima verosimilitud (ML) [K clases, K >2]

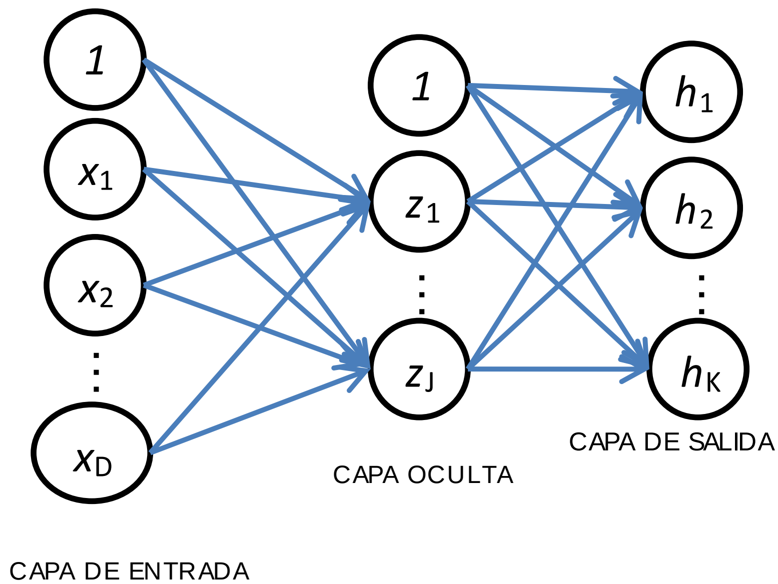
Problemas de clasificación multiclase (más de dos clases)

$$\text{Datos de entrenamiento}: \quad \left\{ \left(\mathbf{x}_n, c_n \right); n = 1,2,\ldots,N \right\}$$

n	x_{n1}	x_{n2}	...	x_{nD}	c_n [1 de K]
1	2.3	0	...	10.3	1 [1000]
2	2.5	1	...	13.1	3 [0010]
3	2.6	0	...	-2.7	2 [0100]
4	2.7	-1	...	-5.4	1 [1000]
5	2.9	0	...	2.1	4 [0001]
6	3.1	0	...	-10.9	3 [0010]

Perceptrón multicapa para problemas multiclase

Perceptrón multicapa para problemas multiclase [1 de K]



$$\mathbf{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_D \end{pmatrix}$$

Parámetros de la red: $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_J)$ $[(D+1) \times J]$

$\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_K)$ $[(J+1) \times K]$

v_{dj} : Peso entre las neuronas d (entrada) y j (oculta)

w_{jk} : Peso entre las neuronas j (oculta) y k (salida)

$x_0 = 1, z_0 = 1$

$$\mathbf{z} = \begin{pmatrix} 1 \\ z_1 \\ z_2 \\ \vdots \\ z_J \end{pmatrix}$$

$$z_j = \sigma\left(\sum_{d=0}^{D} x_d v_{dj}\right) = \sigma\left(\mathbf{v}_j^T \cdot \mathbf{x}\right), j = 1,2,\ldots,J$$

$$h_k(\mathbf{x}) = \frac{\exp\left(\mathbf{w}_k^T \cdot \mathbf{z}\right)}{\sum_{i=1}^{K} \exp\left(\mathbf{w}_i^T \cdot \mathbf{z}\right)}, k = 1,2,\ldots,K$$

Probabilidad posterior $P(C_k \mid \mathbf{x}, \mathbf{V}, \mathbf{W}) = h_k(\mathbf{x}), \quad k = 1,2,\ldots,K$

Propagación hacia atrás en problemas multiclase (online)

Nombre: Propagación en problemas multiclase (online)

Entrada:

$$\text{Datos de entrenamiento}: \{(\mathbf{x}_n, \mathbf{t}_n); n = 1,2,\ldots,N\};$$

$$t_{nk} = \begin{cases} 1 & \text{si } c_n = C_k \\ 0 & \text{si } c_n \neq C_k \end{cases}, k = 1,2,\ldots,K$$

$$\text{Parámetro de aprendizaje}: \ \eta$$

Salida: Pesos del perceptrón multicapa **V**, **W**

Código:

1. Inicializar aleatoriamente lo pesos $\mathbf{V}, \mathbf{W} \sim U[-0.5, 0.5]$
2. Inicializar el contador de épocas $\text{nÉpoca} = 0$
3. Mientras no se cumplan los criterios de convergencia [**épocas**]

   3.1 Actualizar el contador de épocas $\text{nÉpoca} = \text{nÉpoca} + 1$

   3.2 For $\text{n} = 1,2,\ldots,\text{N}$

$$z_{nj} = \sigma(\mathbf{v}_j^T \cdot \mathbf{x}_n), \quad j = 1,2,\ldots,J$$

$$h_k(\mathbf{x}_n) = \frac{\exp(\mathbf{w}_k^T \cdot \mathbf{z}_n)}{\sum_{i=1}^{K} \exp(\mathbf{w}_i^T \cdot \mathbf{z}_n)}, \quad k = 1,2,\ldots,K$$ [probabilidad posterior]

$$\delta_{nk} = h_k(\mathbf{x}_n) - t_{nk}, \qquad k = 1,2,\ldots,K$$ [error de predicción]

$$\Delta_{nj} = z_{nj}(1 - z_{nj}) \sum_{k=1}^{K} w_{jk} \delta_{nk}, \qquad j = 1,2,\ldots,J$$

[asignar error a las neuronas en la capa oculta]

$$\mathbf{w}_k = \mathbf{w}_k - \eta \delta_{nk} \mathbf{z}_n, \ k = 1,2,\ldots,K$$ [actualizar pesos]

$$\mathbf{v}_j = \mathbf{v}_j - \eta \Delta_{nj} \mathbf{x}_n, \quad j = 1,2,\ldots,J$$

Perceptrón multicapa, aprendizaje por ML [multiclase]

Probabilidades posteriores $P\left(C_k \mid \mathbf{x}, \mathbf{V}, \mathbf{W}\right) = h_k(\mathbf{x}), \quad k = 1,2,\ldots,K$

Función de verosimilitud:

$$\text{Datos}: \quad \left\{\left(\mathbf{x}_n, \mathbf{t}_n\right); n = 1,2,\ldots, N\right\}; \quad t_{nk} = \begin{cases} 1 & \text{si } c_n = C_k \\ 0 & \text{si } c_n \neq C_k \end{cases}$$

$$\begin{pmatrix} \mathbf{V}_{ML} \\ \mathbf{W}_{ML} \end{pmatrix} = \underset{\mathbf{V},\mathbf{W}}{\operatorname{argmax}} \left\{ L\left(\mathbf{V}, \mathbf{W}; \left\{\left(\mathbf{x}_n, \mathbf{t}_n\right)\right\}_{n=1}^{N}\right) \right\}$$

Verosimilitud

$$L\left(\mathbf{V}, \mathbf{W}; \left\{\left(\mathbf{x}_n, \mathbf{t}_n\right)\right\}_{n=1}^{N}\right) \equiv P\left(\left\{\mathbf{t}_n\right\}_{n=1}^{N} \mid \left\{\mathbf{x}_n\right\}_{n=1}^{N}, \mathbf{V}, \mathbf{W}\right) =$$

$$\prod_{n=1}^{N} \prod_{k=1}^{K} \left[P(C_k \mid \mathbf{x}_n, \mathbf{V}, \mathbf{W})\right]^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} \left[h_k(\mathbf{x}_n)\right]^{t_{nk}}$$

Función de error de entropía cruzada

$$E(\mathbf{V}, \mathbf{W}) \equiv -\log L\left(\mathbf{V}, \mathbf{W}; \left\{\left(\mathbf{x}_n, \mathbf{t}_n\right)\right\}_{n=1}^{N}\right) = \sum_{n=1}^{N} E_n(\mathbf{V}, \mathbf{W})$$

$$E_n(\mathbf{V}, \mathbf{W}) \equiv -\sum_{k=1}^{K} t_{nk} \log h_k(\mathbf{x}_n)$$

Aprendizaje ML $=$ **Minimizar** $E(\mathbf{V}, \mathbf{W})$

Cálculo del gradiente (I)

$$E_n(\mathbf{V},\mathbf{W}) = \quad -\sum_{k=1}^{K} t_{nk} \log h_k(\mathbf{x}_n) \quad \Rightarrow$$

$$\frac{\partial E_n(\mathbf{V},\mathbf{W})}{\partial \mathbf{b}} = \quad -\sum_{k=1}^{K} t_{nk} \frac{1}{h_k(\mathbf{x}_n)} \frac{\partial h_k(\mathbf{x}_n)}{\partial \mathbf{b}} =$$

$$= -\sum_{k=1}^{K} t_{nk} \left(\frac{\partial\left(\mathbf{w}_k^T \cdot \mathbf{z}_n\right)}{\partial \mathbf{b}} - \sum_{i=1}^{K} \frac{\partial\left(\mathbf{w}_i^T \cdot \mathbf{z}_n\right)}{\partial \mathbf{b}} h_i(\mathbf{x}_n) \right) =$$

$$= \sum_{k=1}^{K} \left(h_k(\mathbf{x}_n) - t_{nk}\right) \frac{\partial\left(\mathbf{w}_k^T \cdot \mathbf{z}_n\right)}{\partial \mathbf{b}} = \sum_{k=1}^{K} \delta_{nk} \frac{\partial\left(\mathbf{w}_k^T \cdot \mathbf{z}_n\right)}{\partial \mathbf{b}}$$

$$\delta_{nk} \equiv \left(y_k(\mathbf{x}_n) - t_{nk}\right)$$

Utilizando : $\displaystyle\sum_{k=1}^{K} t_{nk} = 1$

Utilizando : $\displaystyle h_k(\mathbf{x}_n) = \frac{\exp\left(\mathbf{w}_k^T \cdot \mathbf{z}_n\right)}{\sum_{i=1}^{K} \exp\left(\mathbf{w}_i^T \cdot \mathbf{z}_n\right)} \Rightarrow$

$$\frac{\partial h_k(\mathbf{x}_n)}{\partial \mathbf{b}} = \left(\frac{\partial\left(\mathbf{w}_k^T \cdot \mathbf{z}_n\right)}{\partial \mathbf{b}} - \sum_{i=1}^{K} \frac{\partial\left(\mathbf{w}_i^T \cdot \mathbf{z}_n\right)}{\partial \mathbf{b}} h_i(\mathbf{x}_n) \right) h_k(\mathbf{x}_n)$$

Cálculo del gradiente (II)

$$\frac{\partial E_n(\mathbf{V},\mathbf{W})}{\partial \mathbf{b}} = \sum_{k=1}^{K} \delta_{nk} \frac{\partial\left(\mathbf{w}_k^T \cdot \mathbf{z}_n\right)}{\partial \mathbf{b}}$$

Derivadas con respecto a los pesos de ocultas a salida: **b** → $\mathbf{w}_k$

$$\frac{\partial E_n(\mathbf{V},\mathbf{W})}{\partial \mathbf{w}_k} = \delta_{nk}\mathbf{z}_n; \quad k = 1,2,\ldots,K$$

Derivadas con respecto a los pesos de entrada a ocultas: **b** → $\mathbf{v}_j$

$$\frac{\partial E_n(\mathbf{V},\mathbf{W})}{\partial \mathbf{v}_j} = \sum_{k=1}^{K} \delta_{nk} \frac{\partial\left(\mathbf{w}_k^T \cdot \mathbf{z}_n\right)}{\partial \mathbf{v}_j} = \sum_{k=1}^{K} \delta_{nk} \frac{\partial\left(\sum_{i=0}^{J} w_{ki} z_{ni}\right)}{\partial \mathbf{v}_j} =$$

$$= \sum_{k=1}^{K} \delta_{nk} \sum_{i=0}^{J} w_{ki} \frac{\partial z_{ni}}{\partial \mathbf{v}_j} = \sum_{k=1}^{K} \delta_{nk} w_{kj} \frac{\partial z_{nj}}{\partial \mathbf{v}_j} =$$

$$= z_{nj}\left(1 - z_{nj}\right)\sum_{k=1}^{K} \delta_{nk} w_{kj} \mathbf{x}_n$$

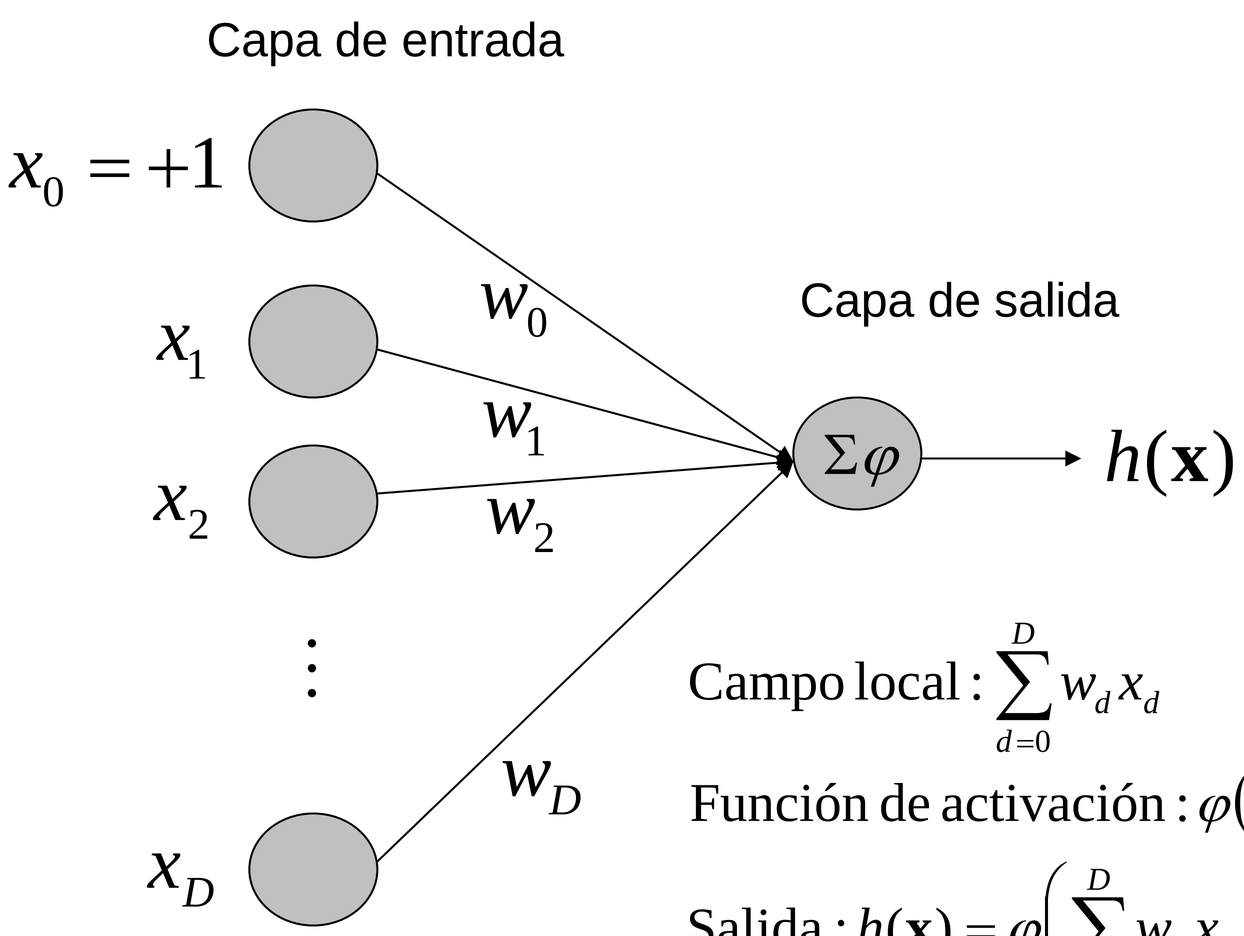
$$z_{ni} = \sigma\left(\mathbf{v}_i^T \cdot \mathbf{x}_n\right) \Rightarrow \begin{cases} \dfrac{\partial z_{ni}}{\partial \mathbf{v}_j} = 0 & i \neq j \\ \dfrac{\partial z_{nj}}{\partial \mathbf{v}_j} = z_{nj} \cdot \left(1 - z_{nj}\right) \cdot \mathbf{x}_n & \end{cases} ; j = 1,2,\ldots,J$$

Otras arquitecturas

Red de propagación hacia delante con una capa

» La red está compuesta por
  – **Capa de entrada:** nodos fuente
  – **Capa de salida:** neuronas (unidades de computación)

» No hay conexiones **de realimentación**



$$\text{Campo local}: \sum_{d=0}^{D} w_d x_d$$

$$\text{Función de activación}: \varphi(\cdot)$$

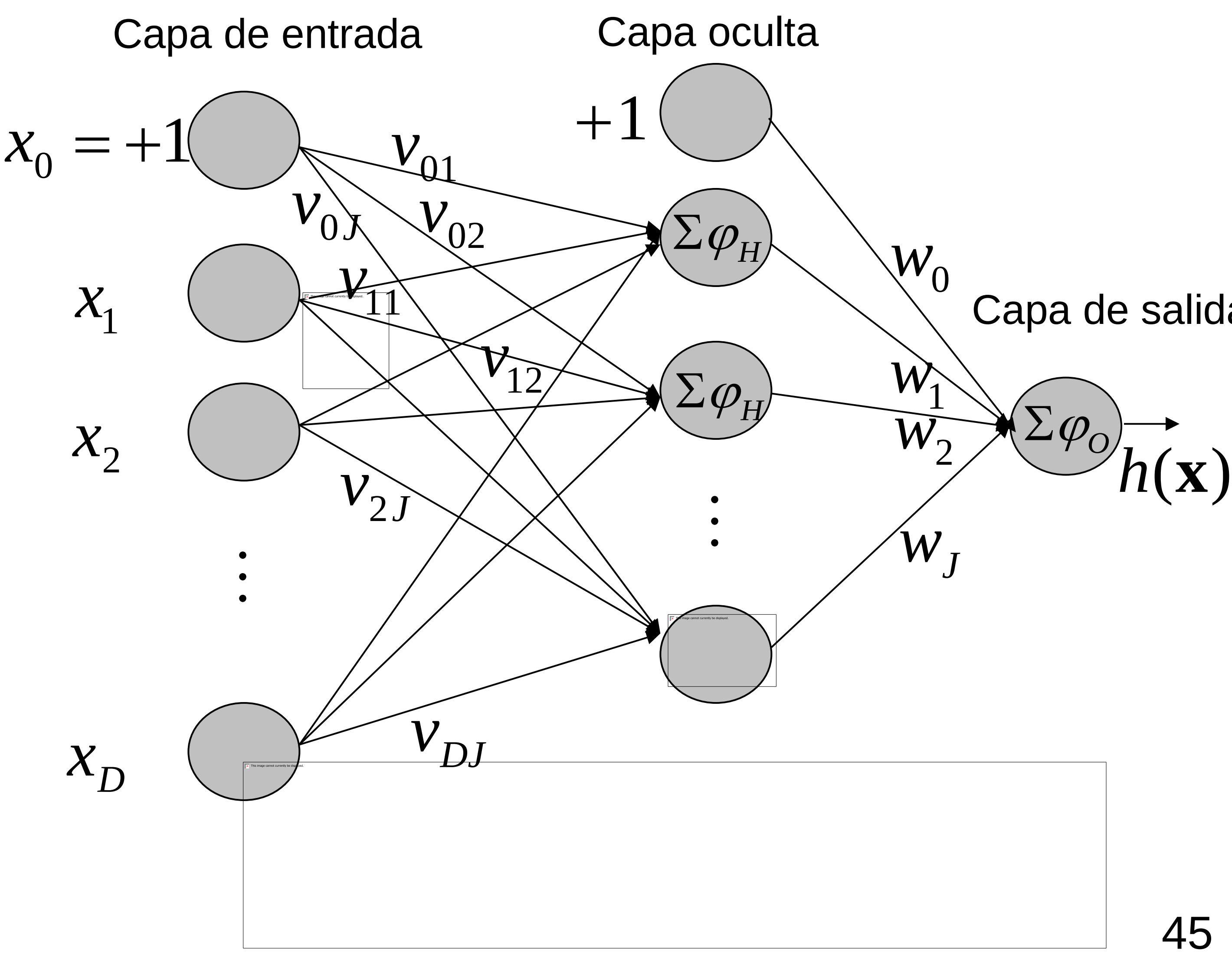
$$\text{Salida}: h(\mathbf{x}) = \varphi\left(\sum_{d=0}^{D} w_d x_d\right)$$

Red de propagación hacia delante multicapa

» La red está compuesta por
  - **Capa de entrada:** nodos fuente
  - **Una o varias capas ocultas: neuronas**
  - **Capa de salida: neurona(s)**

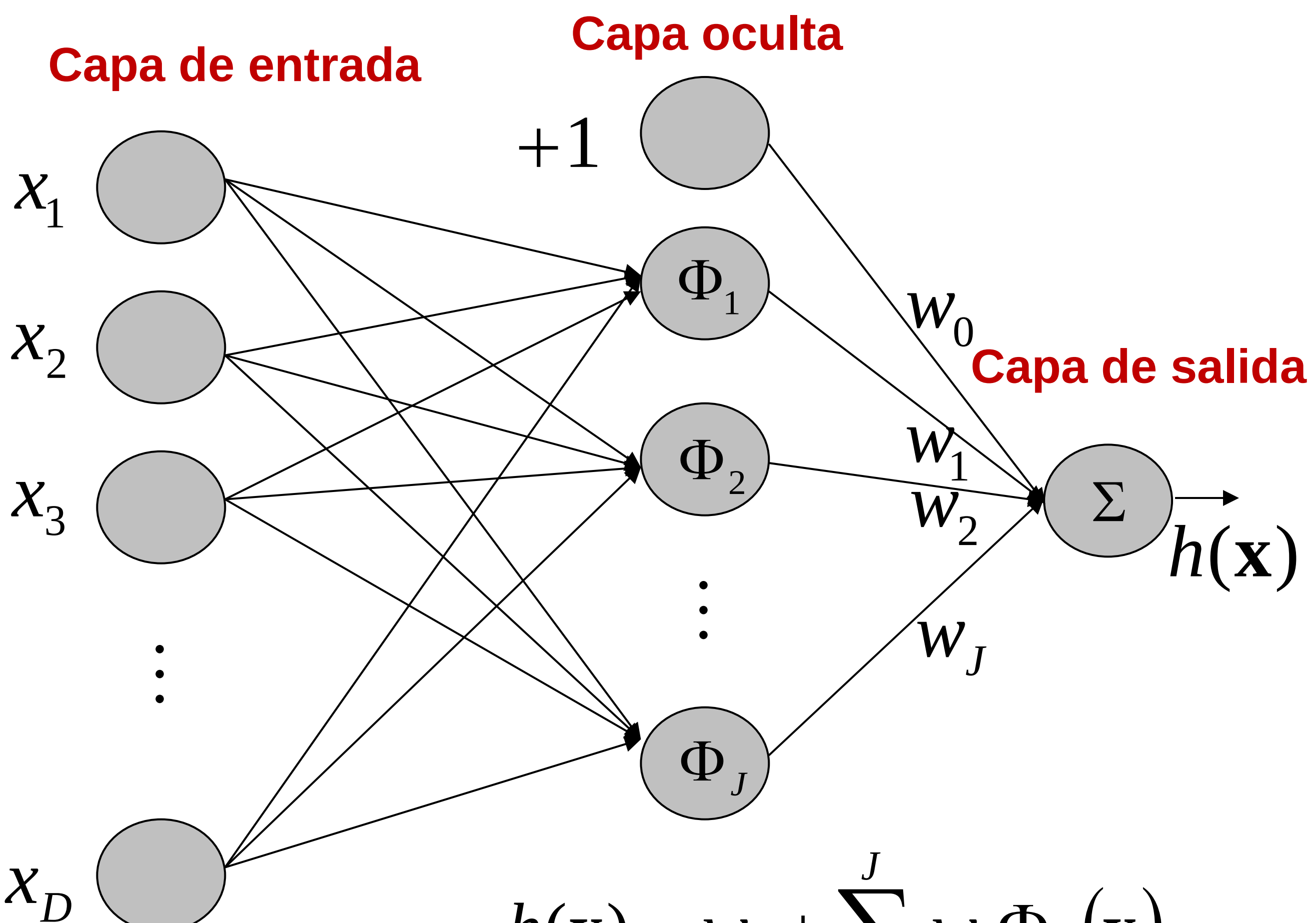
» Sólo hay conexiones entre **capas adyacentes**

» No hay conexiones de **realimentación**



Redes neuronales de funciones de base radial

- La red neuronal está formada por
  - **Capa de entrada** de nodos fuente
  - **Capa oculta**
  - **Capa de salida**
- No hay conexiones de **realimentación**

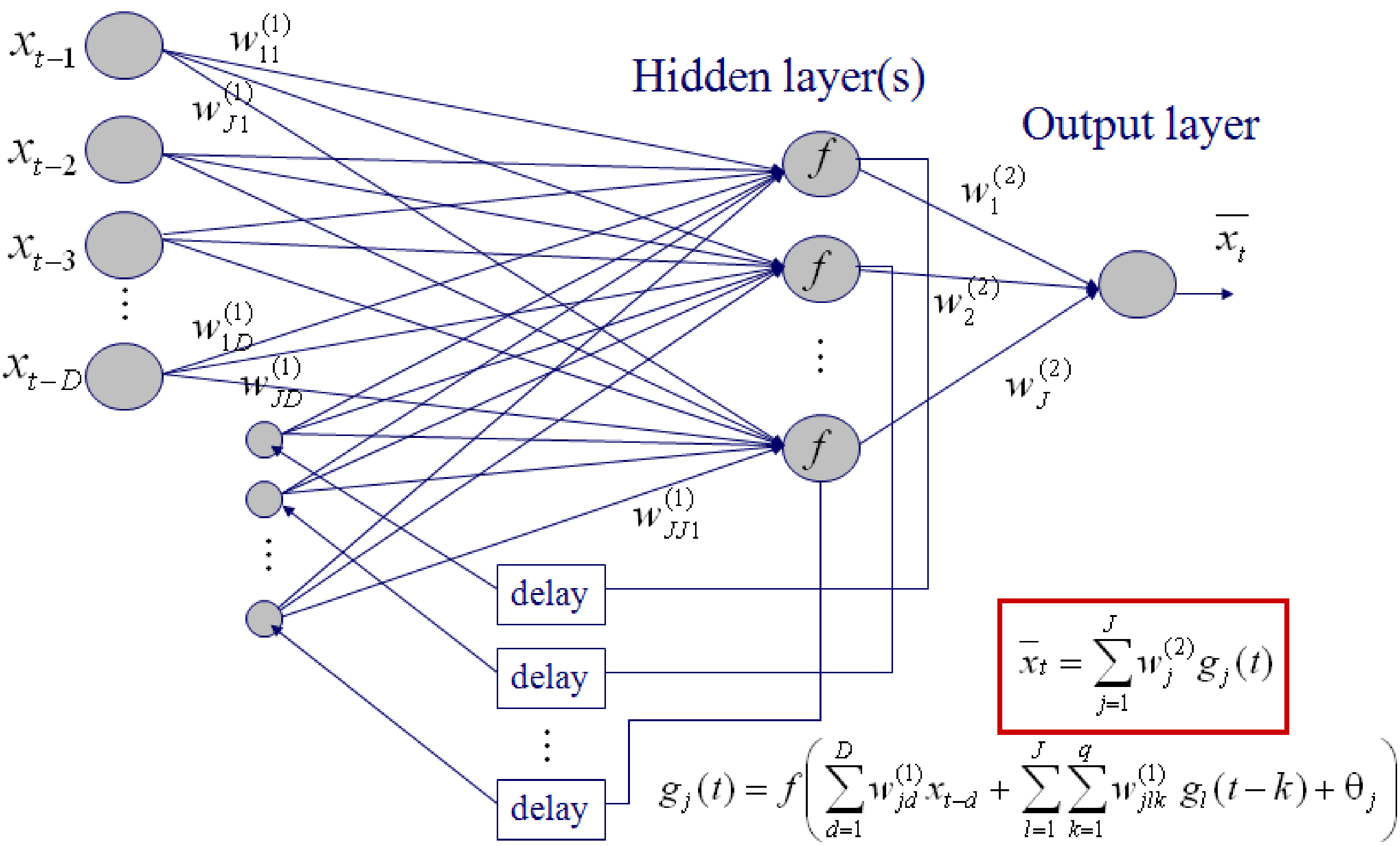


$$h(\mathbf{x}) = w_0 + \sum_{j=1}^{J} w_j \Phi_j(\mathbf{x})$$

$$\text{base gaussiana}: \Phi_j(\mathbf{x}) = \exp\left\{ -\frac{1}{2} \left(\mathbf{x} - \boldsymbol{\mu}_j\right)^{+} \cdot \boldsymbol{\Sigma}_j^{-1} \cdot \left(\mathbf{x} - \boldsymbol{\mu}_j\right) \right\}$$

Redes neuronales recurrentes

- Hay bucles de **realimentación**
- Se necesitan unidades para **retraso**
- **Modelos para series temporales**



$$\boxed{\overline{x}_t = \sum_{j=1}^{J} w_j^{(2)} g_j(t)}$$

$$g_j(t) = f\left(\sum_{d=1}^{D} w_{jd}^{(1)} x_{t-d} + \sum_{l=1}^{J} \sum_{k=1}^{q} w_{jlk}^{(1)} \, g_l(t-k) + \theta_j \right)$$