5.4 Redes neuronales para clasificación

Inteligencia Artificial

3er curso INF

Escuela Politécnica Superior

Aprendizaje supervisado: clasificación

Aprendizaje automático por **inducción a partir de datos etiquetados** $\mathcal{D} = \{(\mathbf{x}_n, c_n)\}_{n=1}^{N}$

$\mathbf{x} \in \mathcal{X}$: vector de **atributos**, características, variables independientes, variables de entrada, covariables …

Término bias
$x_{n0} = 1$

$$\mathbf{x}_n^T = (x_{n0}\ x_{n1}\ \ x_{n2}\ \dots x_{nD})$$ [$(D+1)$-vector dimensional]

$c \in \mathcal{C}$: (clase) **etiqueta**, dependent variable, outcome, target, …

Clasificación: **Etiquetas** c **discretas** $(e.g.\ c \in \{C_1, \dots, C_K\})$

Algoritmo de aprendizaje : $\mathcal{L}$

w: parámetros del modelo

$$\mathcal{L}\colon \mathcal{D} = \{(\mathbf{x}_n, c_n)\}_{n=1}^{N} \to h(\cdot\,; \mathbf{w})$$

$$h(\cdot\,; \mathbf{w})\colon \quad \mathbf{x} \in \mathcal{X} \ \to\ h(\mathbf{x}; \mathbf{w}) \in \{C_1, \dots, C_K\}$$

Clasificación binaria

n	x_{n1}	x_{n2}	...	x_{nD}	c_n
1	2.3	0	...	10.3	C_0
2	2.5	1	...	13.1	C_1
3	2.6	0	...	-2.7	C_1
4	2.7	-1	...	-5.4	C_0
5	2.9	0	...	2.1	C_1
6	3.1	0	...	-10.9	C_0

Clasificación binaria

Codificación 0-1 de los **datos etiquetados**

$$\mathcal{D} = \{(\mathbf{x}_n, c_n)\}_{n=1}^{N} \Rightarrow \mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^{N}; \;\; t_n = \begin{cases} 0 & \text{si} \;\; c_n = C_0 \\ 1 & \text{si} \;\; c_n = C_1 \end{cases}$$

$$\mathcal{L}\text{: } \mathcal{D} = \{(\mathbf{x}_n, c_n)\}_{n=1}^{N} \rightarrow o(\,\cdot\,; \mathbf{w})$$

$$z(\,\cdot\,; \mathbf{w})\text{:} \quad \mathbf{x} \in \mathcal{X} \; \rightarrow z(\mathbf{x}; \mathbf{w}) \in \mathbb{R}$$

$$o(\mathbf{x}; \mathbf{w}) = \varphi^{(o)}\big(z(\mathbf{x}; \mathbf{w})\big) \in [0,1]$$

- La salida $o\,(\mathbf{x};\, \mathbf{w})$ es una estimación de la clase C_1 posterior

Modelo discriminativo

$$\hat{p}(C_1 | \mathbf{x}, \mathbf{w}) = \varphi^{(o)}\big(z(\mathbf{x}; \mathbf{w})\big)$$

Regresión logística : $\varphi^{(o)}(z) = \sigma(z) = \frac{1}{1+e^{-z}}$

Regresión probit: $\varphi^{(o)}(z) = \Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{y^2}{2}} dy$

Clasificación binaria: codificación 0-1

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 probabilidad

Se suponen Muestras dii (distribuidas Idéntica e Independiente-mente)

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w})$$

- Probabilidad del modelo, dado $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^N$

$$\mathcal{L}(\mathbf{w}) = \hat{P}(\{t_n\}_{n=1}^N | \{\mathbf{x}_n\}_{n=1}^N, \mathbf{w}) = \prod_{n=1}^N \hat{p}(t_n|\mathbf{x}_n, \mathbf{w}) =$$

$$= \prod_{n=1}^N \left(\hat{p}(C_0|\mathbf{x}_n, \mathbf{w})\right)^{1-t_n} \left(\hat{p}(C_1|\mathbf{x}_n, \mathbf{w})\right)^{t_n}$$

Factoriza porque se asumen muestras independientes

Estimación de la clase posterior

Misma distribución: se asumen muestras idénticamente dist.

- Función de verosimilitud

$$\log \mathcal{L}(\mathbf{w}) = \sum_{n=1}^N \left((1 - t_n) \log \hat{p}(C_0|\mathbf{x}_n, \mathbf{w}) + t_n \log \hat{p}(C_1|\mathbf{x}_n, \mathbf{w})\right)$$

Mismo maximizador que $\mathcal{L}(\mathbf{w})$ (log is monotono)

Minimización del error de entropía cruzada

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} CE(\mathbf{w})$$

Mismo maximizador que $\mathcal{L}(\mathbf{w})$

- Error de entropía cruzada
- $CE(\mathbf{w}) = -\log \mathcal{L}(\mathbf{w})$

$$= -\sum_{n=1}^{N} \big((1-t_n)\log \hat{p}(C_0|\mathbf{x}_n,\mathbf{w}) + t_n \log \hat{p}(C_1|\mathbf{x}_n,\mathbf{w})\big)$$

$$= -\sum_{n=1}^{N} \Big((1-t_n)\log\big(1-\hat{p}(C_1|\mathbf{x}_n,\mathbf{w})\big) + t_n \log \hat{p}(C_1|\mathbf{x}_n,\mathbf{w})\Big)$$

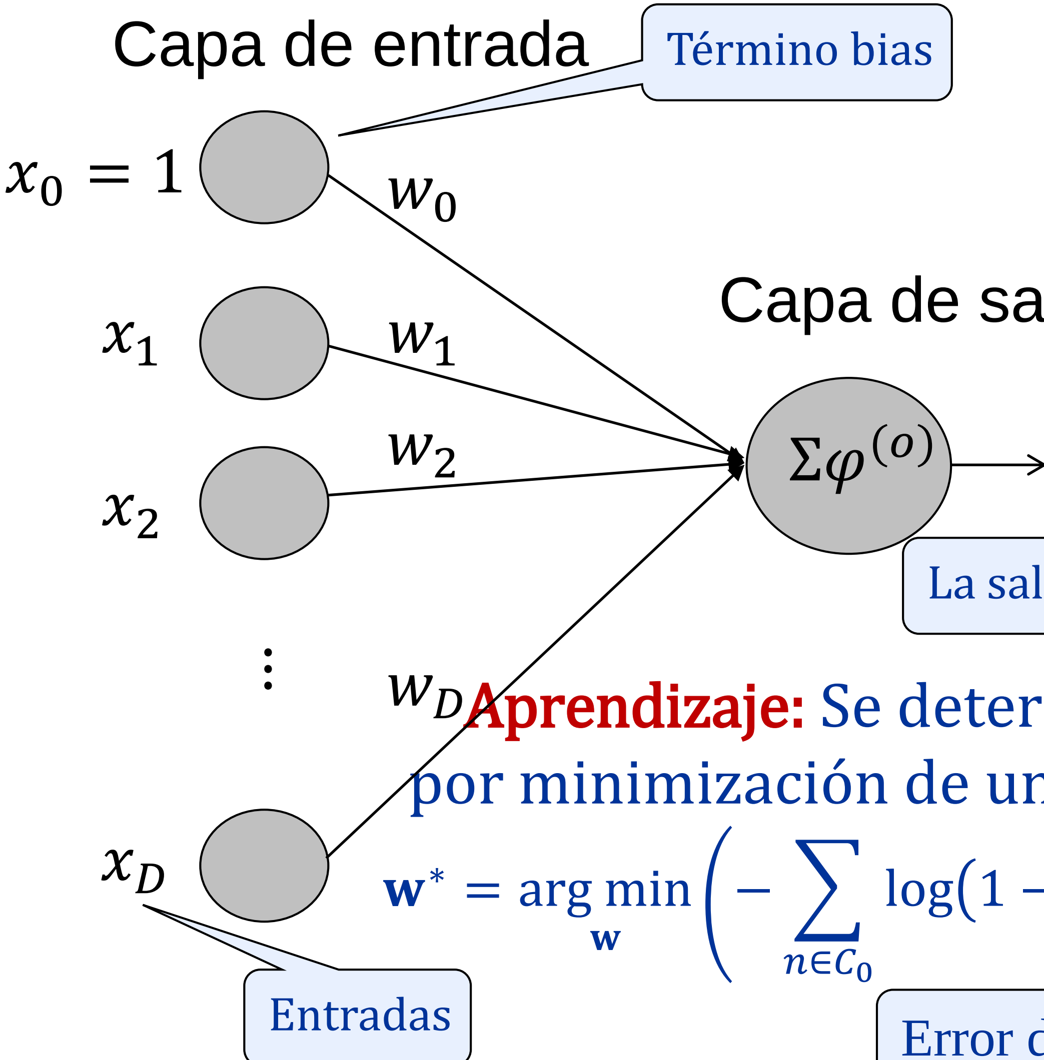
- Modelo discriminativo : $\hat{p}(C_1|\mathbf{x},\mathbf{w}) = \varphi^{(o)}\big(z(\mathbf{x}_n;\mathbf{w})\big)$

$$CE(\mathbf{w}) = -\sum_{n=1}^{N}\Big((1-t_n)\log\Big(\underbrace{1-\varphi^{(o)}\big(z(\mathbf{x}_n;\mathbf{w})\big)}_{\hat{p}(C_0|\mathbf{x},\mathbf{w})}\Big) + t_n\log\Big(\underbrace{\varphi^{(o)}\big(z(\mathbf{x}_n;\mathbf{w})\big)}_{\hat{p}(C_1|\mathbf{x},\mathbf{w})}\Big)\Big)$$

Perceptrón monocapa

Puede resolver solo problemas linealmente separables

$$\mathbf{w}^{\mathrm{T}} = (w_0, w_1, \dots, w_D)$$
$$\mathbf{x}^{\mathrm{T}} = (x_0\ x_1 \dots x_D)$$
$$\mathbf{w}^{\mathrm{T}}\mathbf{x} = \sum_{d=0}^{D} w_d x_d$$



$$o(\mathbf{x};\mathbf{w}) = \varphi^{(0)}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}\right)$$

La salida es una estimación de $p(C_1|\mathbf{x})$

Aprendizaje: Se determinan los pesos de la red. por minimización de una función de coste. Ej:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(-\sum_{n\in C_0} \log\left(1 - o(\mathbf{x}_n;\mathbf{w})\right) - \sum_{n\in C_1} \log\left(o(\mathbf{x}_n;\mathbf{w})\right) \right)$$

Error de entropía cruzada

Regresión logística

- Clase posteriores

$$p(C_1|\mathbf{x},\mathbf{w}) = \frac{1}{1+e^{-z(\mathbf{x};\mathbf{w})}}$$

$$p(C_0|\mathbf{x},\mathbf{w}) = 1 - p(C_1|\mathbf{x},\mathbf{w}) = \frac{e^{-z(\mathbf{x};\mathbf{w})}}{1+e^{-z(\mathbf{x};\mathbf{w})}} = \frac{1}{1+e^{z(\mathbf{x};\mathbf{w})}}$$

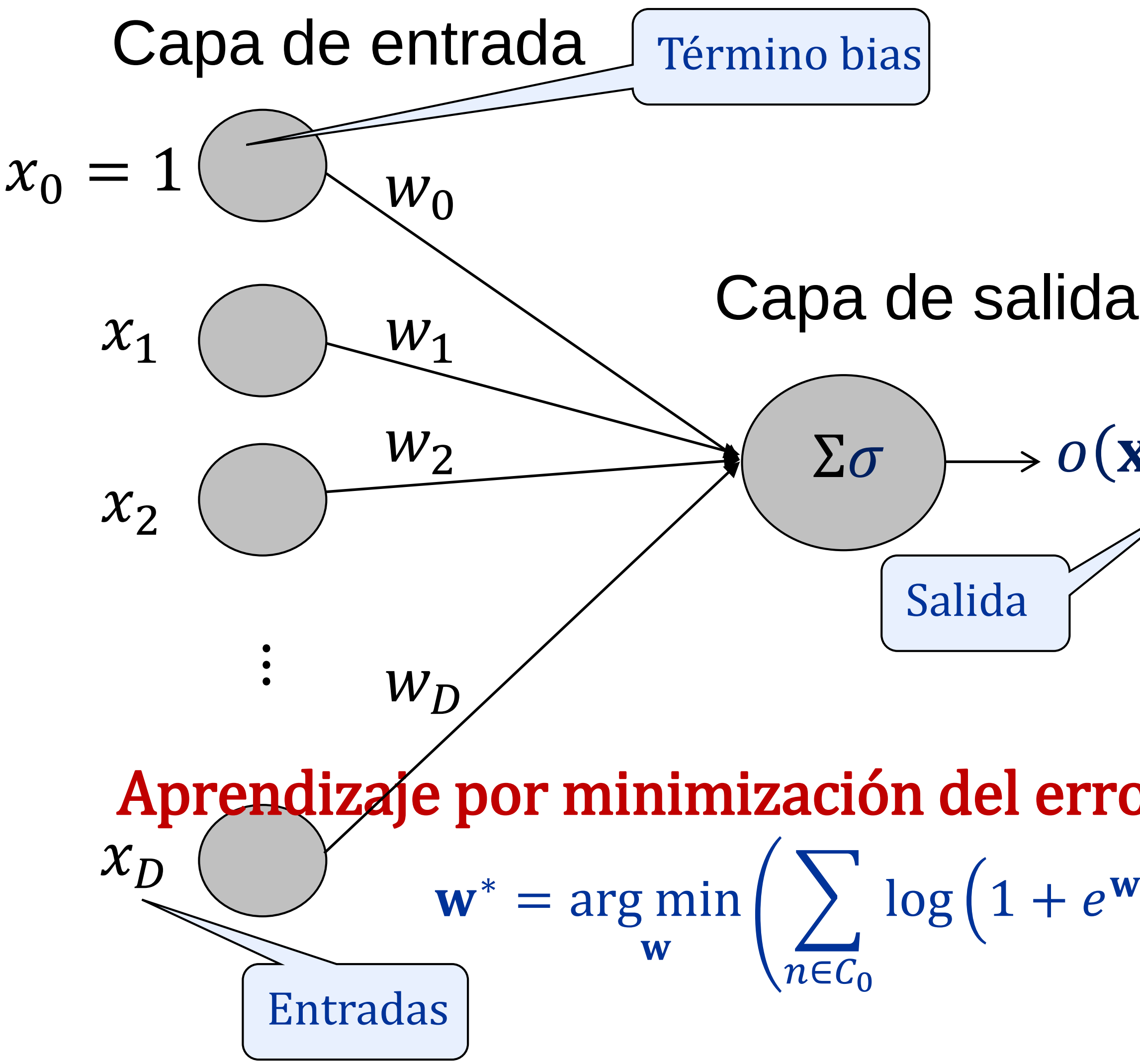
- Probabilidades-log (log-odds): $\mathrm{z}(\mathbf{x};\mathbf{w}) = \log\frac{p(C_1|\mathbf{x},\mathbf{w})}{p(C_0|\mathbf{x},\mathbf{w})}$

- Modelo lineal para log-odds: $z(\mathbf{x};\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} = \sum_{d=0}^{D} w_d x_d$

$\mathbf{w}^{\mathrm{T}} = (w_0, w_1, \dots, w_D)$
$\mathbf{x}^{\mathrm{T}} = (x_0\ x_1 \dots x_D)$

Perceptrón monocapa!

Perceptrón monocapa: regresión logística



$$\mathbf{w}^{\mathrm{T}} = (w_0, w_1, \dots, w_D)$$
$$\mathbf{x}^{\mathrm{T}} = (x_0\ x_1 \dots x_D)$$
$$\mathbf{w}^{\mathrm{T}}\mathbf{x} = \sum_{d=0}^{D} w_d x_d$$

$$o(\mathbf{x};\mathbf{w}) = \sigma\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}\right)$$

Aprendizaje por minimización del error de entropía cruzada :

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_{n\in C_0} \log\left(1 + e^{\mathbf{w}^{\mathrm{T}}\mathbf{x}_n}\right) + \sum_{n\in C_1} \log\left(1 + e^{-\mathbf{w}^{\mathrm{T}}\mathbf{x}_n}\right) \right)$$

Error de entropía cruzada para regresión logística

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} CE(\mathbf{w})$$

$$\frac{\partial \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n)}{\partial \mathbf{w}} = \mathbf{x}_n \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n)\left(1 - \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n)\right)$$

- Error de entropía cruzada

$$CE(\mathbf{w}) = -\sum_{n=1}^{N}\left((1-t_n)\log\left(1-\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n)\right) + t_n \log \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n)\right)$$

- Gradiente del error de entropía cruzada

$$\frac{\partial}{\partial \mathbf{w}} CE(\mathbf{w}) = \sum_{n=1}^{N}\left((1-t_n)\mathbf{x}_n \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n) - t_n \mathbf{x}_n\left(1-\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n)\right)\right)$$

$$\delta_n = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n) - t_n$$

$$= \sum_{n=1}^{N}\left(\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n) - t_n\right)\mathbf{x}_n = \sum_{n=1}^{N} \delta_n \, \mathbf{x}_n$$

Perceptrón de una capa: aprendizaje por lotes

Los atributos deberían
Estar escalados!

ENTRADA: Instancias de entrenamiento : $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^{N}$

Parámetro de aprendizaje : $\eta > 0$

Similar a Rosenblatt!
$\delta_n = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n) - t_n$
predicción error

SALIDA: $\mathbf{w}^* = \arg\min_{\mathbf{w}} CE(\mathbf{w})$

1. Inicializar aleatoriamente $\mathbf{w} \sim U[-0.5{,}0.5]^{(D+1)}$
2. $n_{epoch} = 0$
3. Mientras no se cumplan los criterios de convergencia
    3.1 Incrementa contador de épocas: $n_{epoch} = n_{epoch} + 1$
    3.2 Calcula las salidas de red : $\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n)$; $n = 1, \ldots, N$
    3.3 Calcula el gradiente : $\frac{\partial}{\partial \mathbf{w}} CE(\mathbf{w}) = \sum_{n=1}^{N} \delta_n \, \mathbf{x}_n$
    3.4 Actualizar pesos : $\mathbf{w} = \mathbf{w} - \eta \sum_{n=1}^{N} \delta_n \, \mathbf{x}_n$

Perceptrón de una capa: aprendizaje en línea

ENTRADA: Instancias de entrenamiento : $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^{N}$

Parámetro de aprendizaje : $\eta > 0$

SALIDA: $\mathbf{w}^* = \arg\min_{\mathbf{w}} CE(\mathbf{w})$

1. Inicializa aleatoriamente $\mathbf{w} \sim U[-0.5, 0.5]^{(D+1)}$
2. $n_{epoch} = 0$
3. Mientras no se cumplan los criterios de convergencia

    3.1 Incrementa contador de épocas : $n_{epoch} = n_{epoch} + 1$

    3.2 Para $n = 1, \dots, N$

    Calcula la salida de red : $\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n)$

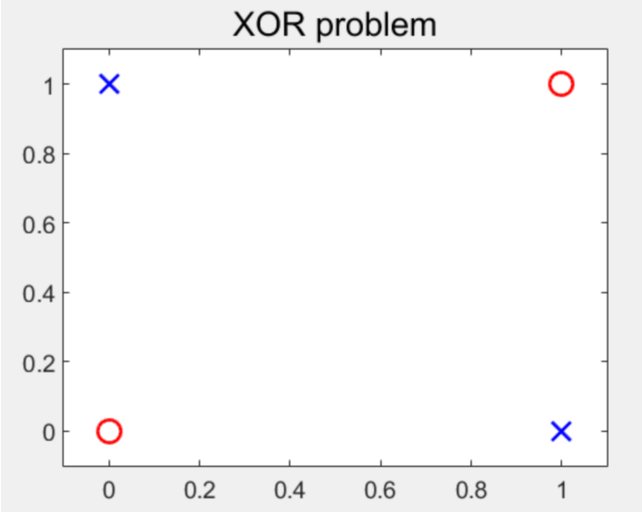
    Calcula error de predicción : $\delta_n = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n) - t_n$

    Actualiza pesos: $\mathbf{w} = \mathbf{w} - \eta\delta_n \mathbf{x}_n$

XOR: Un problema no separable linealmente

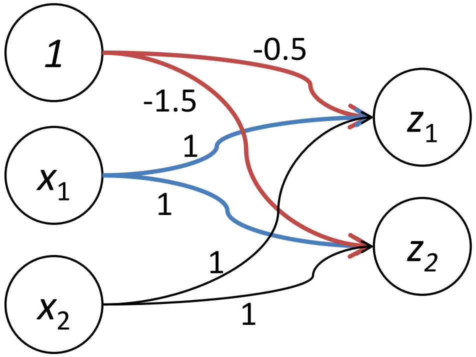
- El perceptrón de una sola capa solo puede abordar problemas que son linealmente separables
- Por lo tanto, el simple problema XOR, que no es linealmente separable, no se puede resolver con esta máquina de aprendizaje.

x_1	x_2	h
0	0	1
0	1	0
1	0	0
1	1	1

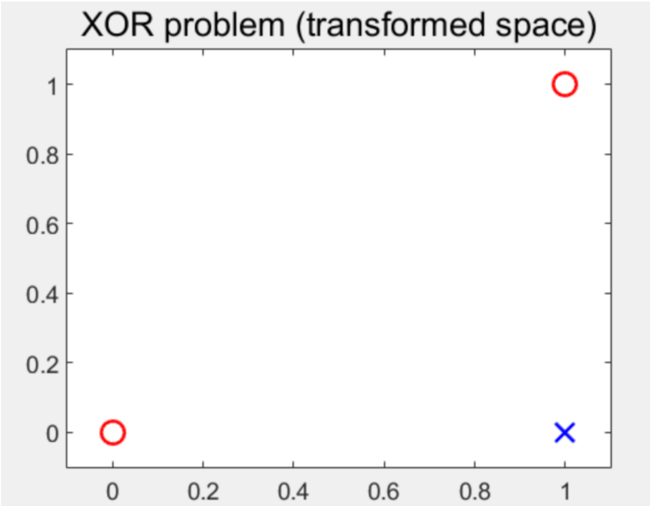


XOR: Construcción de características no lineales

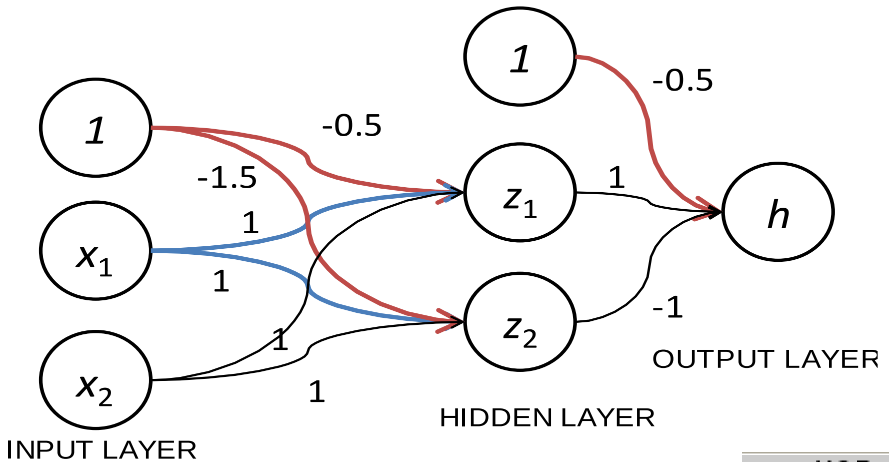
- Considere el problema XOR en un espacio de características transformado



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



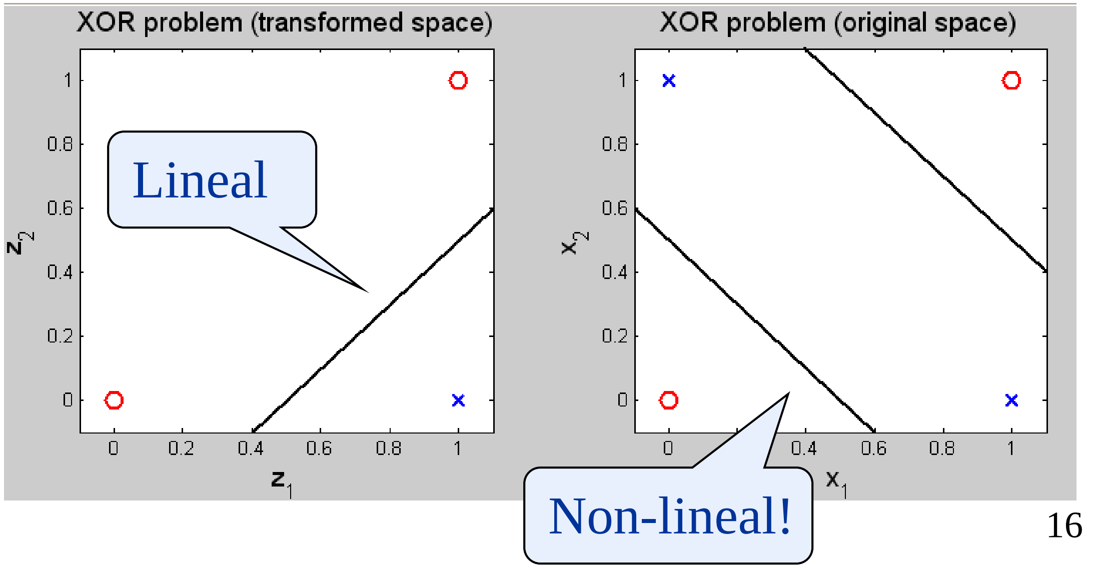
XOR: Un modelo lineal en espacio de características



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

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

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



Perceptrón multicapa

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

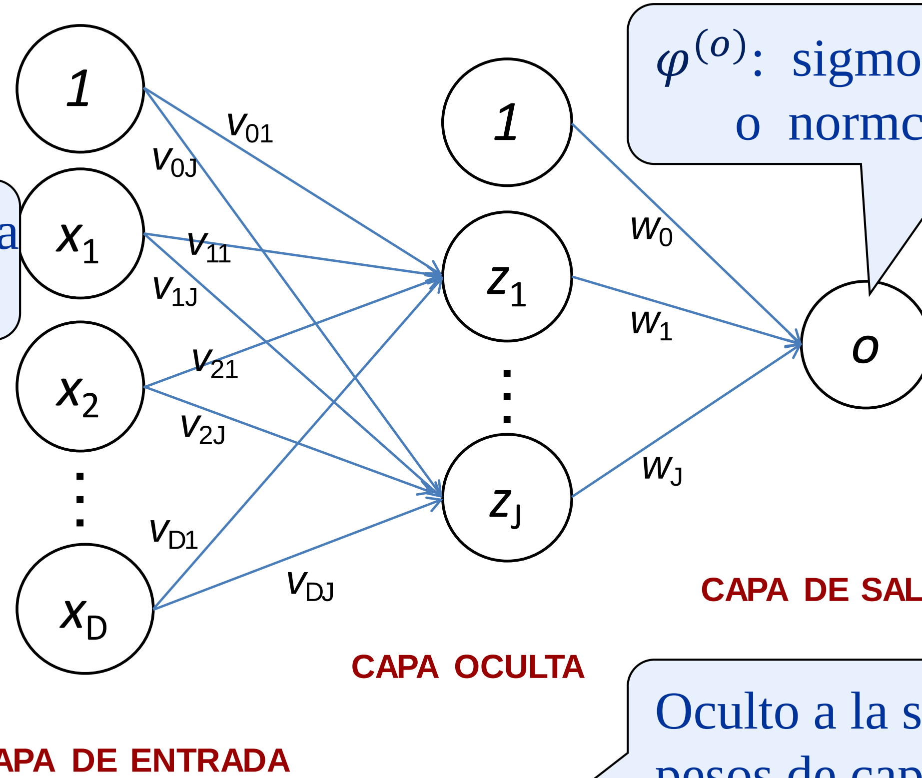
vector de atributos (de entrada)

Oculto a la entrada pesos de capa

$$\mathbf{V} = (\mathbf{v}_1 \ \mathbf{v}_2 \dots \mathbf{v}_J)$$

$$\mathbf{v}_j^{\mathrm{T}} = (\mathrm{v}_{0j} \ \mathrm{v}_{1j} \dots \mathrm{v}_{Dj});$$

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



$\varphi^{(o)}$: sigmoidal o normcdf

Local field

Oculto a la salida pesos de capa

$$z_j = \phi_j^{(H)}(\mathbf{v}_j^{\mathrm{T}}\mathbf{x}); \qquad j = 1, 2, \dots, J; \qquad \mathbf{w}^{\mathrm{T}} = (\mathrm{w}_0 \ \mathrm{w}_1 \dots \mathrm{w}_D);$$

Función de Activación

$$o(\mathbf{x}; \mathbf{w}) = \varphi^{(o)}(\mathbf{w}^{\mathrm{T}}\mathbf{z}) = \varphi^{(o)}\left(\sum_{j=0}^{J} w_j \, \phi_j^{(H)}(\mathbf{v}_j^{\mathrm{T}}\mathbf{x})\right)$$

Función de activación sigmoide (logit)

También: función de transferencia

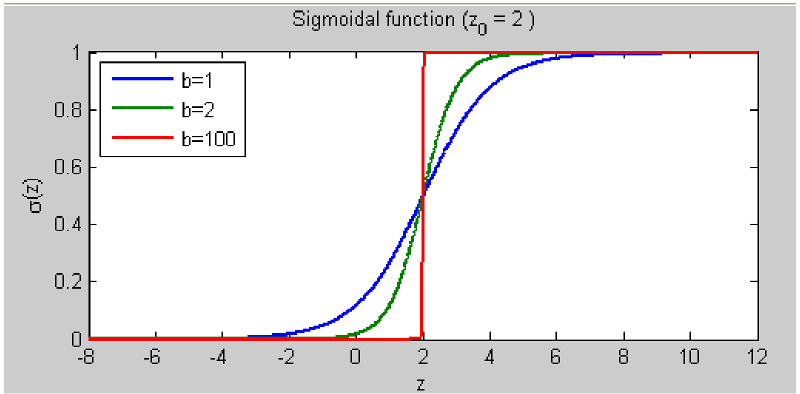
$$\sigma(z) = \frac{1}{1+e^{-z}};$$

- $\sigma(-\infty) = 0; \quad \sigma(0) = \frac{1}{2}; \sigma(+\infty) = 1;$
- Monotónamente creciente : $z_2 > z_1 \Rightarrow \sigma(z_2) > \sigma(z_1)$
- Simetría : $\sigma(-z) = 1 - \sigma(z)$
- Derivada: $\sigma'(z) = \frac{d\sigma(z)}{dz} = \sigma(z)\big(1-\sigma(z)\big)$

$$\sigma(z; z_0, b) = \frac{1}{1+e^{-b(z-z_0)}}$$

$\frac{1}{b}$: escala

z_0: centro



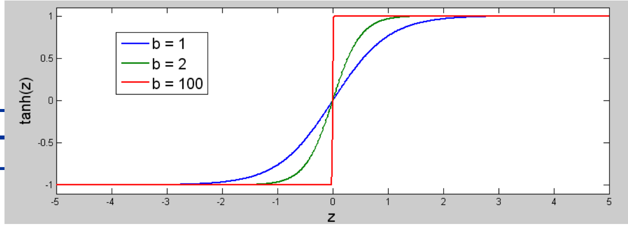
$$\lim_{b\to\infty} \sigma(z; z_0, b) = \begin{cases} 0 & \text{si} \quad z < z_0 \\ \frac{1}{2} & \text{si} \quad z = z_0 \\ 1 & \text{si} \quad z > z_0 \end{cases}$$

Función de paso de Heaviside

Otras funciones de activación

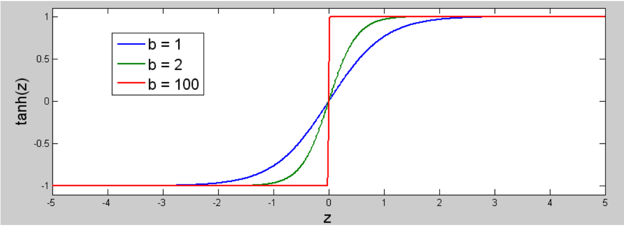
In regression output layer

- **Lineal**: $\varphi(z) = z$
- **Tangente hiperbolica** : $\tanh(z) = \frac{e^{z}-}{e^{z}+}$



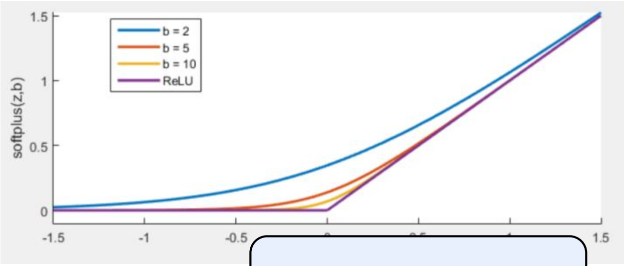
normcdf

- **Probit**: $\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{y^2}{2}} dy$



- **softplus**: $(z; z_0, b) = \frac{log\left(1+e^{-b(z-z_0)}\right)}{b}$

$$\lim_{b\to\infty} \mathrm{softplus}(z; 0, b) = \mathrm{ReLU}(z)$$



- **Unidad lineal rectificada** : $\mathrm{ReLU}(z) = \max(0, z)$

Ej. $a = 0.01$

- **Unidad lineal rectificada con fugas** : $\text{Leaky ReLU}(z) = \begin{cases} az & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}; \quad 0 < a \ll 1$

Evita el gradiente cero si $z < 0$

Perceptrón multicapa: aprendizaje en línea

ENTRADA: Training instances: $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^{N}$

Parámetro de aprendizaje : $\eta > 0$

SALIDA: $\mathbf{V}^*, \mathbf{w}^* = \arg\min_{\mathbf{V},\mathbf{w}} CE(\mathbf{V}, \mathbf{w})$

Solo una capa oculta.
Activaciones sigmoideas.

1. Inicializar aleatoriamente $\mathbf{V}, \mathbf{w} \sim U[-0.5, 0.5]^{(D+1)}$
2. $n_{epoch} = 0$
3. Mientras no se cumplan los criterios de convergencia
   - 3.1 Incrementar el contador de épocas : $n_{epoch} = n_{epoch} + 1$
   - 3.2 Para $n = 1, \dots, N$

$z_{nj} = \sigma(\mathbf{v}_j^{\mathrm{T}} \mathbf{x}_n), \quad j = 1, 2, \dots, J;$ **# propagación hacia adelante**

$o_n = \sigma(\mathbf{w}^{\mathrm{T}} \mathbf{z}_n)$ # salida de red

$\delta_n = o_n - t_n;$ # error de predicción

$\mathbf{w} = \mathbf{w} - \eta \delta_n \mathbf{z}_n$ # actualización de peso

$\Delta_{nj} = z_{nj}(1 - z_{nj})\delta_n, \quad j = 1, 2, \dots, J;$ **# error de propagación hacia atrás**

$\mathbf{v}_j = \mathbf{v}_j - \eta \Delta_{nj} \mathbf{x}_n$ # actualización de peso

Propiedad de aproximación universal

THEOREM: "A feed-forward network with a **single hidden layer** containing a **sufficiently large number of hidden neurons** can uniformly **approximate any continuous function** on compact subsets of $\mathbb{R}^D$, under mild conditions on the activation function"

- GOOD NEWS:

  A neural network can be used to make optimal predictions!

- NOT SO GOOD NEWS:

  How does one find such a network?

  - How many neurons do we need?
  - How easy is it to learn the network parameters (weights)?

Use CV

Having more than one layer can help (deep networks)

K. Hornik, M. Stinchcombe and H. White, "Multi-layer Feedforward Networks are Universal Approximators," Neural Networks, Vol. 2, pp. 359-366 (1989).

Propiedad de aproximación universal

TEOREMA: "Una red de retroalimentación con una sola capa oculta que contiene un **número suficientemente grande de neuronas ocultas** puede **aproximar** de manera uniforme **cualquier función continua** en subconjuntos compactos de $\mathbb{R}^D$, bajo condiciones suaves en la función de activación"

- BUENAS NOTICIAS :

¡Se puede usar una red neuronal para hacer predicciones óptimas!

- NO TAN BUENAS NOTICIAS :

¿Cómo se encuentra una red así?

Usar VC

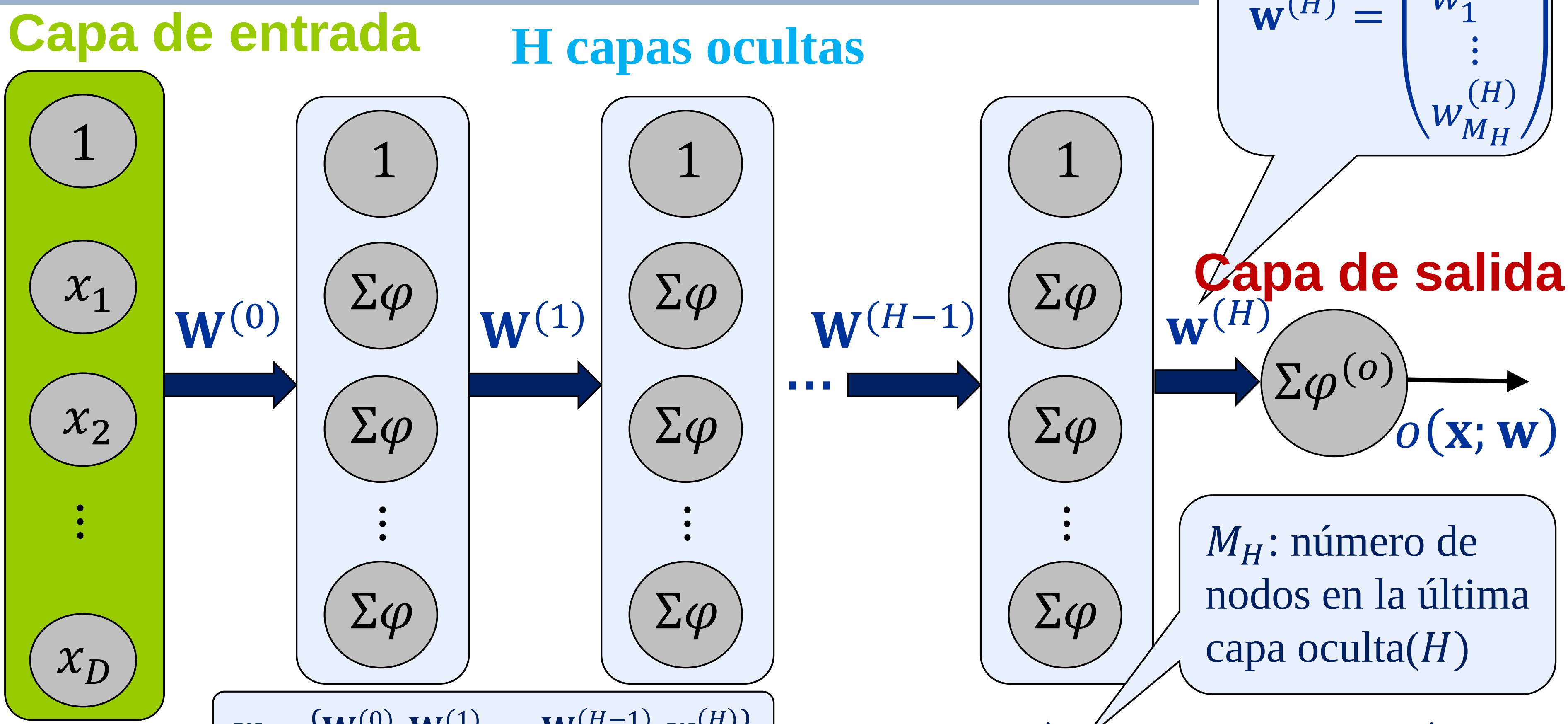
Tener más de una capa puede ayudar (redes profundas)

  - ¿Cuántas neuronas necesitamos?
  - ¿Qué tan fácil es aprender los parámetros de red (pesos)?

K. Hornik, M. Stinchcombe and H. White, "Multi-layer Feedforward Networks are Universal Approximators," Neural Networks, Vol. 2, pp. 359-366 (1989).

Redes neuronales profundas: clasificación

$$\mathbf{w}^{(H)} = \begin{pmatrix} w_0^{(H)} \\ w_1^{(H)} \\ \vdots \\ w_{M_H}^{(H)} \end{pmatrix}$$



M_H: número de nodos en la última capa oculta(H)

$$\mathbf{w} = \{\mathbf{W}^{(0)}, \mathbf{W}^{(1)}, \dots, \mathbf{W}^{(H-1)}, \mathbf{w}^{(H)}\}$$

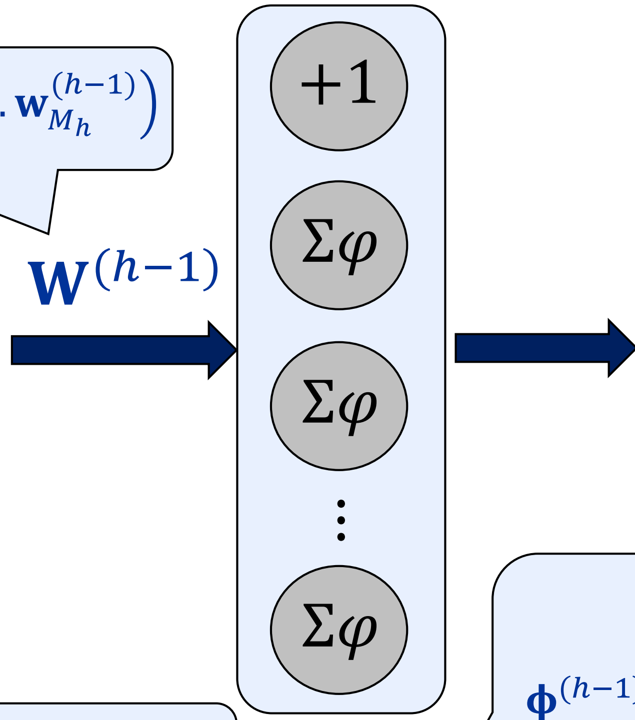
$$o(\mathbf{x};\mathbf{w}) = \varphi^{(o)}\left(\left(\mathbf{w}^{(H)}\right)^T \boldsymbol{\phi}^{(H)}(\mathbf{x})\right) = \varphi^{(o)}\left(\sum_{m=0}^{M_H} w_m^{(H)} \phi_m^{(H)}(\mathbf{x})\right)$$

Salida de capa oculta h

$$\mathbf{W}^{(h-1)} = \left(\mathbf{w}_1^{(h-1)} \mathbf{w}_2^{(h-1)} \dots \mathbf{w}_{M_h}^{(h-1)}\right)$$

$$\phi_0^{(h)} = 1$$

$$\boldsymbol{\phi}^{(h-1)}(\mathbf{x}) \xrightarrow{\mathbf{W}^{(h-1)}}$$



$$\boldsymbol{\phi}^{(h)}(\mathbf{x}) = \begin{pmatrix} 1 \\ \phi_1^{(h)}(\mathbf{x}) \\ \vdots \\ \phi_{M_h}^{(h)}(\mathbf{x}) \end{pmatrix}$$

$$\boldsymbol{\phi}^{(0)}(\mathbf{x}) = \mathbf{x} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_D \end{pmatrix}$$

$$M_0 = D$$

$$\boldsymbol{\phi}^{(h-1)}(\mathbf{x}) = \begin{pmatrix} 1 \\ \phi_1^{(h-1)}(\mathbf{x}) \\ \vdots \\ \phi_{M_h}^{(h-1)}(\mathbf{x}) \end{pmatrix}$$

$$\left(\mathbf{w}_m^{(h-1)}\right)^{\mathrm{T}} = \left(w_{m0}^{(h-1)} \; w_{m1}^{(h-1)} \dots w_{mM_{h-1}}^{(h-1)}\right)$$

$$\phi_m^{(h)}(\mathbf{x}) = \varphi^{(h)}\left(\left(\mathbf{w}_m^{(h-1)}\right)^{\mathrm{T}} \boldsymbol{\phi}^{(h-1)}(\mathbf{x})\right); \qquad m = 1, \dots, M_h$$

$$M_{H+1} = 1$$

$$h = 1, \dots, H, H+1$$

En la ultima capa: $\varphi^{(H+1)}(z) = \varphi^{(o)}(z)$ (logistico, probit,..)

Aprendiendo los pesos de la red

Los pesos de la red se determinan minimizando una función de coste:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\left[-\sum_{n\in C_0} \log(1 - o(\mathbf{x}_n;\mathbf{w})) - \sum_{n\in C_1} \log(o(\mathbf{x}_n;\mathbf{w})) \right] + \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \|\mathbf{w}\|_2^2 \right)$$

Error de entropía cruzada

Uso VC para seleccionar

$\lambda_1 > 0$

L_1 penalización

$\lambda_2 > 0$

L_2 penalización

- **Arquitectura:** número de capas ocultas / neuronas en capa oculta
- **Método de optimización:** cuasi-Newton, descenso de gradiente, descenso de gradiente estocástico, uso de impulso, …

Gradientes necesarios!

- **Hiperparámetros:** magnitud de las penalizaciones, del plazo de impulso, # máximo de iteraciones, …

Impulso/momentum es acelerar la velocidad de aprendizaje

Clasificación de clases múltiples (clases K)

Codificación 1-de K de los datos etiquetados

$$\mathcal{D} = \{(\mathbf{x}_n, c_n)\}_{n=1}^{N} \Rightarrow \mathcal{D} = \{(\mathbf{x}_n, \mathbf{t}_n)\}_{n=1}^{N}; \quad \boldsymbol{t}_n^{\mathrm{T}} = (t_{n1}\ t_{n2} \dots t_{nK})$$

$$t_{nk} = \mathbb{I}[c_n = C_k] = \begin{cases} 0 & \text{if} \quad y_n \neq C_k, \\ 1 & \text{if} \quad y_n = C_k \end{cases}; \quad k = 1, 2, \dots, K$$

$$\mathcal{L}\colon \mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N} \rightarrow o(\,\cdot\,; \mathbf{w})$$

$$z(\,\cdot\,; \mathbf{w})\colon \quad \mathbf{x} \in \mathcal{X} \rightarrow \mathbf{z}(\mathbf{x}; \mathbf{w}) \in \mathbb{R}^K$$

$$\mathbf{o}(\mathbf{x}; \mathbf{w}) = \boldsymbol{\varphi}^{(\mathrm{o})}\big(\mathbf{z}(\mathbf{x}; \mathbf{w})\big) \in \Delta^K$$

Probabilidad simplex en K dimensiones

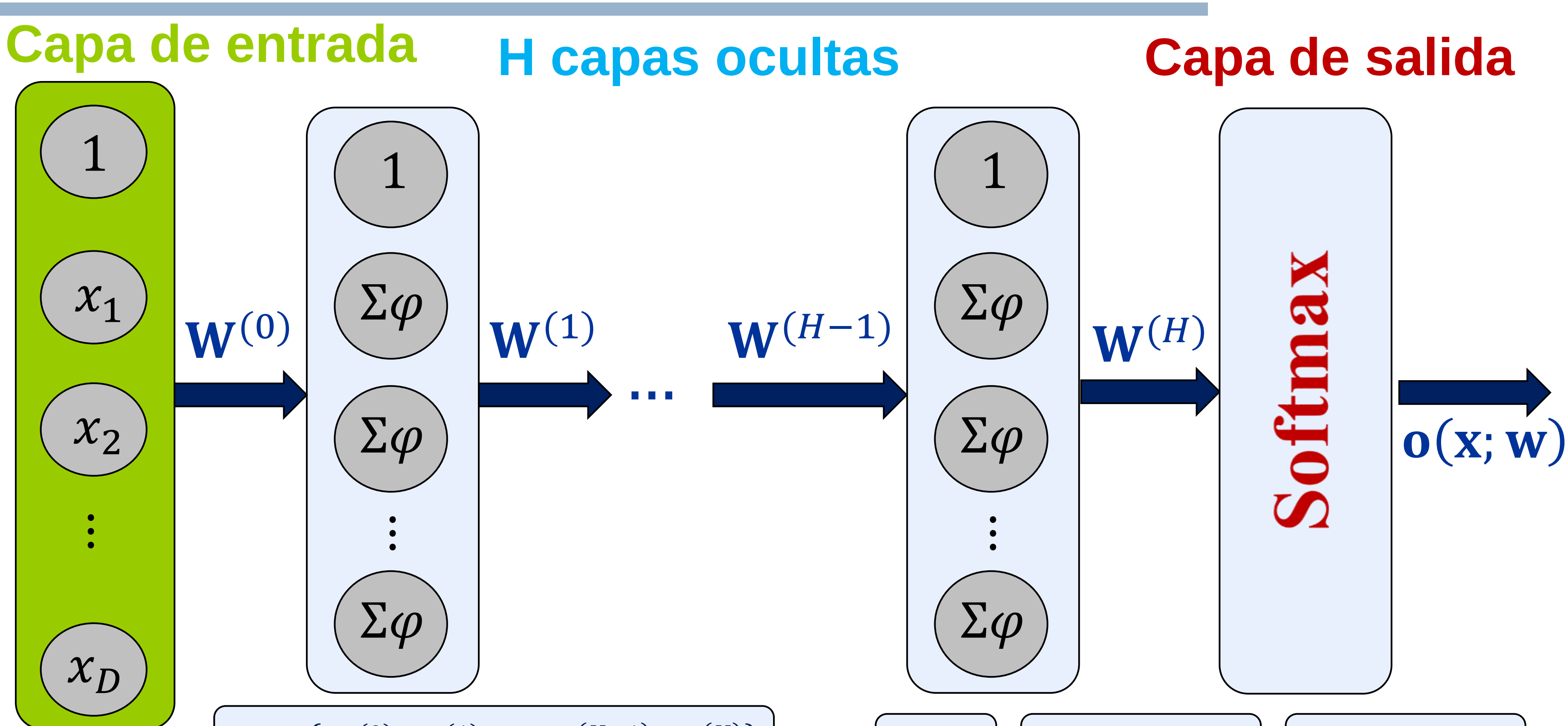
model Discriminativo $\hat{p}(C_k|\mathbf{x}, \mathbf{w}) = \varphi_K^{(o)}(\mathbf{z}(\boldsymbol{x}; \mathbf{w}))$

$$\varphi_k^{(o)}(\mathbf{z}) \geq 0 \qquad k = 1, \dots, K$$

$$\varphi_1^{(o)}(\mathbf{z}) + \cdots + \varphi_K^{(o)}(\mathbf{z}) = 1$$

$$\left(\boldsymbol{\varphi}^{(\mathrm{o})}(\mathbf{z})\right)^{\mathrm{T}} = \left(\varphi_1^{(o)}(\mathbf{z}) \dots \varphi_K^{(o)}(\mathbf{z})\right)$$

MLP para clasificación multiclase



$\mathbf{w} = \{\mathbf{W}^{(0)}, \mathbf{W}^{(1)}, \dots, \mathbf{W}^{(H-1)}, \mathbf{W}^{(H)}\}$

$K \times 1$ $\quad$ $K \times 1$ $\quad$ $K \times (M_H + 1)$ $\quad$ $(M_H + 1) \times 1$

$$\mathbf{o}(\mathbf{x};\mathbf{w}) = \boldsymbol{\varphi}^{(o)}\big(\mathbf{z}(\mathbf{x},\mathbf{w})\big) = \boldsymbol{\varphi}^{(o)}\left(\left(\mathbf{W}^{(H)}\right)^T \boldsymbol{\phi}^{(H)}(\mathbf{x})\right)$$

Capa Softmax

$$\mathbf{z}(\mathbf{x};\mathbf{w}) = \begin{pmatrix} z_1(\mathbf{x},\mathbf{w}) \\ z_2(\mathbf{x},\mathbf{w}) \\ \vdots \\ z_K(\mathbf{x},\mathbf{w}) \end{pmatrix} \Rightarrow \boxed{\textbf{Softmax}} \Rightarrow \boldsymbol{\varphi}^{(o)}\big(\mathbf{z}(\mathbf{x};\mathbf{w})\big) = \begin{pmatrix} \varphi_1^{(o)}\big(\mathbf{z}(\mathbf{x},\mathbf{w})\big) \\ \varphi_2^{(o)}\big(\mathbf{z}(\mathbf{x},\mathbf{w})\big) \\ \vdots \\ \varphi_K^{(o)}\big(\mathbf{z}(\mathbf{x},\mathbf{w})\big) \end{pmatrix}$$

$$\hat{p}(C_k|\mathbf{x},\mathbf{w}) = \varphi_k^{(o)}\big(\mathbf{z}(\mathbf{x},\mathbf{w})\big)$$
$$k = 1, \dots, K$$

probabilidades

$$\varphi_k^{(o)}(\mathbf{z}) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \geq 0 \qquad k = 1, \dots, K$$

$$\varphi_1^{(o)}(\mathbf{z}) + \cdots + \varphi_K^{(o)}(\mathbf{z}) = 1$$

Aprendizaje por máxima probabilidad

Se asumen Muestras dii (distribuidas Idéntica e Independiente-mente)

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w})$$

- Probabilidad del modelo, dado $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{t}_n)\}_{n=1}^{N}$

$$\mathcal{L}(\mathbf{w}) = \hat{P}(\{\mathbf{t}_n\}_{n=1}^{N} | \{\mathbf{x}_n\}_{n=1}^{N}, \mathbf{w}) = \prod_{n=1}^{N} \hat{p}(t_n | \mathbf{x}_n, \mathbf{w}) =$$

Factoriza porque se asumen muestras independientes

$$= \prod_{n=1}^{N} \prod_{k=1}^{K} \left(\hat{p}(C_k | \mathbf{x}_n, \mathbf{w})\right)^{t_{nk}}$$

Igualamos factores porque se asumen muestras idénticamente dist.

- Función de verosimilitud-log

$t_{nk} = \mathbb{I}[c_n = C_k]$

$$\log \mathcal{L}(\mathbf{w}) = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log \hat{p}(C_k | \mathbf{x}_n, \mathbf{w})$$

Minimización del error de entropía cruzada

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} CE(\mathbf{w})$$

- Error de entropía cruzada

$$CE(\mathbf{w}) = -\log \mathcal{L}(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \log \hat{p}(C_k|\mathbf{x}_n, \mathbf{w})$$

- Modelo discriminatorio: $\hat{p}(C_1|\mathbf{x}, \mathbf{w}) = \varphi_k^{(o)}\big(z(\mathbf{x}_n, \mathbf{w})\big)$

Salida de la k neurona en la última capa de la MLP.

$$\widehat{CE}(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \log \varphi_k^{(o)}\big(z(\mathbf{x}_n, \mathbf{w})\big)$$

Resumen de redes neuronales

- **Ventajas**
  - Excelentes predictores.
  - Adaptativo (aprendizaje en línea)
- **Desventajas**
  - Entrenamiento costoso.
  - Encontrar la arquitectura adecuada puede ser difícil
    - Número de capas / nodos ocultos en cada capa oculta
    - Función de activación
  - Determinando los hiperparámetros para la optimización
    - Tipo de optimización
    - En SGD: tasa de aprendizaje, tamaño de mini lotes, término de impulso, fuerza de los términos de regularización, ...
  - Difícil interpretación.

Estado del arte: redes neuronales profundas

Resumen de redes neuronales

- **Advantages**
  - Excellent predictors.
  - Adaptive (online learning)
- **Disadvantages**
  - Costly training.
  - Finding appropriate architecture can be difficult
    - Number of hidden layers / nodes in each hidden layer
    - Activation function
  - Determining the hyperparameters for the optimization
    - Type of optimization
    - In SGD: learning rate, size of mini-batches, momentum term, strength of regularization terms, …
  - Difficult interpretation.

State-of-the-art: Deep neural networks

Referencias

- Multilayer perceptron with TensorFlow
  https://playground.tensorflow.org/

- Scikit learn documentation
  - Neural networks
    http://scikit-learn.org/stable/modules/neural_networks_supervised.html

- Simon O. Haykin "Neural Networks and Learning Machines, 3rd edition", Pearson (2009)

- Ian Goodfellow and Yoshua Bengio and Aaron Courville "Deep Learning", MIT Press (2016)