2023-2024

Aprendizaje Automático

3. Redes Neuronales

Francisco Casacuberta Nolla
(fcn@dsic.upv.es)

Alfons Juan Císcar
(ajuan@dsic.upv.es)

(con material de Enrique Vidal Ruiz)

Departament de Sistemes Informàtics i Computació (DSIC)

Universitat Politècnica de València (UPV)

Septiembre, 2023

Index

1 Redes neuronales multicapa ▷ 2

2 Algoritmo de retropropagación del error (BackProp) ▷ 19

3 Aspectos de uso y propiedades del BackProp ▷ 34

4 Notación ▷ 49

Index

◦ 1 *Redes neuronales multicapa* ▷ 2

2 Algoritmo de retropropagación del error (BackProp) ▷ 19

3 Aspectos de uso y propiedades del BackProp ▷ 34

4 Notación ▷ 49

Modelo conexionista

- Un conjunto de procesadores elementales densamente interconectados.

- Nombres alternativos:
  - Modelo conexionista.
  - Red neuronal artificial.
  - Procesado distribuido y paralelo.

- Perceptrón multicapa:
  - Modelo conexionista simple.
  - Fronteras de decisión complejas.
  - Optimización no convexa.
  - Entrenamiento de los pesos mediante descenso por gradiente: algoritmo de retropropagación del error.

Introducción: historia (I)

- 1943: McCulloch y Pitt introducen un modelo matemático simple de “neurona”.
- 1949: Hebb propone una regla que modela el aprendizaje en las neuronas. Rochester realiza una simulación en un computador IBM en 1950.
- 1957: *Rosenblatt* introduce *el Perceptrón* como un dispositivo hardware con capacidad de autoaprendizaje y Widrow y Hoff proponen el *Adaline* para la cancelación de ecos en redes telefónicas.
- 1969: *Minsky y Papert* demuestran que un perceptrón solo puede implementar funciones discriminantes lineales y que esta limitación no se puede superar mediante multiples perceptrones organizados en cascada: Para ello sería necesario introducir funciones no-lineales.
- 1970-1975: Diversos autores tratan de desarrollar algoritmos de descenso por gradiente adecuados para multiples perceptrones en cascada con funciones no-lineales. El cálculo de derivadas parciales se muestra esquivo.

Introducción: historia (II)

- 1986: *Rumelhart, Hinton y Williams* popularizan la técnica de *retropropagación del error*. Se basa en el uso de cierto tipo de funciones no lineales, llamadas "funciones de activación", con las que se simplifica el cálculo de derivadas parciales necesarias para descenso por gradiente. Al parecer, técnicas similares habían sido ya propuestas por *Linnainmaa* en 1970 y *Werbos* en 1974.

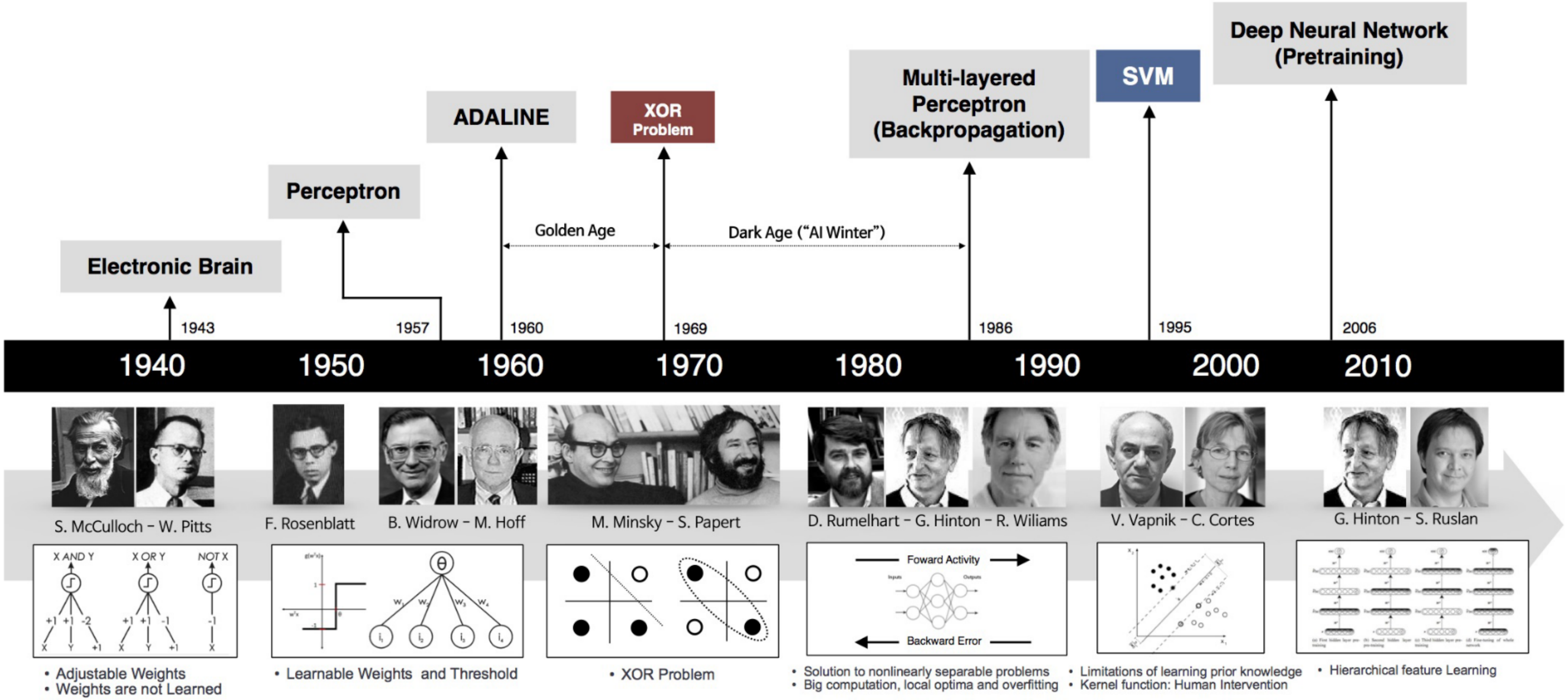
- 1996: *Bishop*, *Rippley*, *Ney*, entre otros, dan una interpretación probabilística a las redes neuronales y al algoritmo de retropropagación del error.

- 2000: Limitaciones en el uso del algoritmo de retropropagación del error en redes de muchas capas.

- 2006: *Hinton* publica en *Science* un artículo que inagura una nueva tendencia denominada *"redes profundas"*. Posteriormente, en 2015 *LeCun, Bengio y Hinton* publican un artículo sobre estas técnicas en *Nature*. Se desarrollan diversas técnicas que mejoran el uso del algoritmo de retropropagación en redes profundas y en redes recurrentes.

- 2015-: Aplicación con gran éxito en multiples problemas. Uso de grandes redes pre-entrenadas.

Introducción: historia (III)



(Serengil, Evolution of Neural Networks 2017)

Funciones discriminantes lineales y función de activación

- FUNCIONES DISCRIMINANTES LINEALES (FDL)

$$\phi : \mathbb{R}^d \to \mathbb{R} : \quad \phi(\boldsymbol{x};\boldsymbol{\theta}) \;=\; \boldsymbol{\theta}^t \boldsymbol{x} \;=\; \sum_{i=0}^{d} \theta_i \, x_i$$

Notación en coordenadas homogéneas:

– $\boldsymbol{x} \in \mathbb{R}^{d+1}, \qquad \boldsymbol{x} = x_0, x_1, \ldots, x_d\,, \qquad x_0 \stackrel{\text{def}}{=} 1$

– $\boldsymbol{\theta} \in \mathbb{R}^{D}, \qquad \boldsymbol{\theta} = \theta_0,\, \theta_1, \ldots,\, \theta_d \qquad D \stackrel{\text{def}}{=} d+1$

La componente 0 del *vector de pesos* es el *umbral*, $\theta_0 \in \mathbb{R}$

Funciones discriminantes lineales y función de activación

- FUNCIONES DISCRIMINANTES LINEALES (FDL)

$$\phi : \mathbb{R}^d \to \mathbb{R} : \quad \phi(\boldsymbol{x};\boldsymbol{\theta}) \ = \ \boldsymbol{\theta}^t \boldsymbol{x} \ = \ \sum_{i=0}^{d} \theta_i \, x_i$$

Notación en coordenadas homogéneas:

- $\boldsymbol{x} \in \mathbb{R}^{d+1}, \qquad \boldsymbol{x} = x_0, x_1, \ldots, x_d\,, \qquad x_0 \stackrel{\text{def}}{=} 1$
- $\boldsymbol{\theta} \in \mathbb{R}^{D}, \qquad \boldsymbol{\theta} = \ \theta_0,\ \theta_1, \ldots,\ \theta_d \qquad D \stackrel{\text{def}}{=} d+1$

La componente 0 del *vector de pesos* es el *umbral*, $\theta_0 \in \mathbb{R}$

- FUNCIONES DISCRIMINANTES LINEALES CON ACTIVACIÓN (FDLA)

$$g \circ \phi : \mathbb{R}^d \to \mathbb{R} : \quad g \circ \phi(\boldsymbol{x};\boldsymbol{\theta}) \ = \ g(\boldsymbol{\theta}^t \boldsymbol{x})$$

$g : \mathbb{R} \to \mathbb{R}$ es una función de activación †

† también denominada función logística y a la FDLA función discriminante lineal logística.

Funciones de activación y sus derivadas

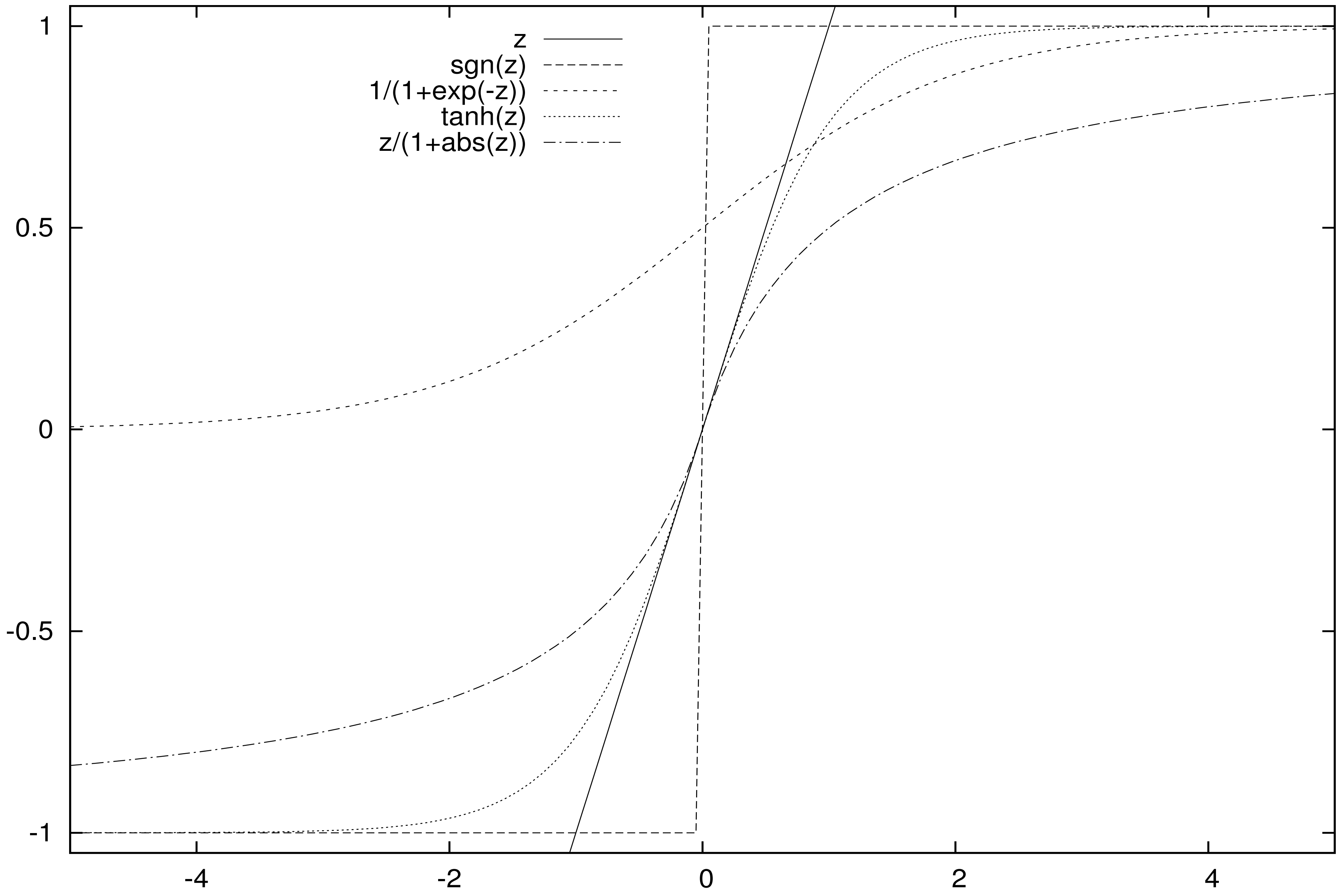
Sea $g : \mathbb{R} \to \mathbb{R}$ y $z \in \mathbb{R}$:

- LINEAL: $g_L(z) = z \;\Rightarrow\; g'_L(z) = \dfrac{d\, g_L}{d\, z} \;=\; 1$
- RELU (rectified linear unit): $g_U(z) = \max(0, z) \;\Rightarrow\; g'_U(z) = \begin{cases} 0 & \text{si } z < 0 \\ 1 & \text{si } z > 0 \\ \text{no definida} & \text{si } z = 0 \end{cases}$
- ESCALÓN†: $g_E(z) = \operatorname{sgn}(z) \overset{\text{def}}{=} \begin{cases} +1 & \text{si } z > 0 \\ -1 & \text{si } z < 0 \end{cases} \;\Rightarrow\; g'_E(z) = \begin{cases} \text{no definida} & \text{si } z = 0 \\ 0 & \text{si } z \neq 0 \end{cases}$
- SIGMOID: $g_S(z) = \dfrac{1}{1 + \exp(-z)} \;\Rightarrow\; g'_S(z) = g_S(z)\,(1 - g_S(z))$
- TANGENTE HIPERBÓLICA: $g_T(z) = \dfrac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} \;\Rightarrow\; g'_T(z) = 1 - (g_T(z))^2$
- RÁPIDA: $g_F(z) = \dfrac{z}{1 + |z|} \;\Rightarrow\; g'_F(z) = \dfrac{1}{(1 + |z|)^2} = \left(\dfrac{g_F(z)}{z}\right)^2$
- SOFTMAX: Para $z_1, \ldots, z_n \in \mathbb{R}$, $g_M(z_i) = \dfrac{\exp(z_i)}{\sum_j \exp(z_j)} \;\Rightarrow\; g'_M(z_i) = g_M(z_i)\,(1 - g_M(z_i))$

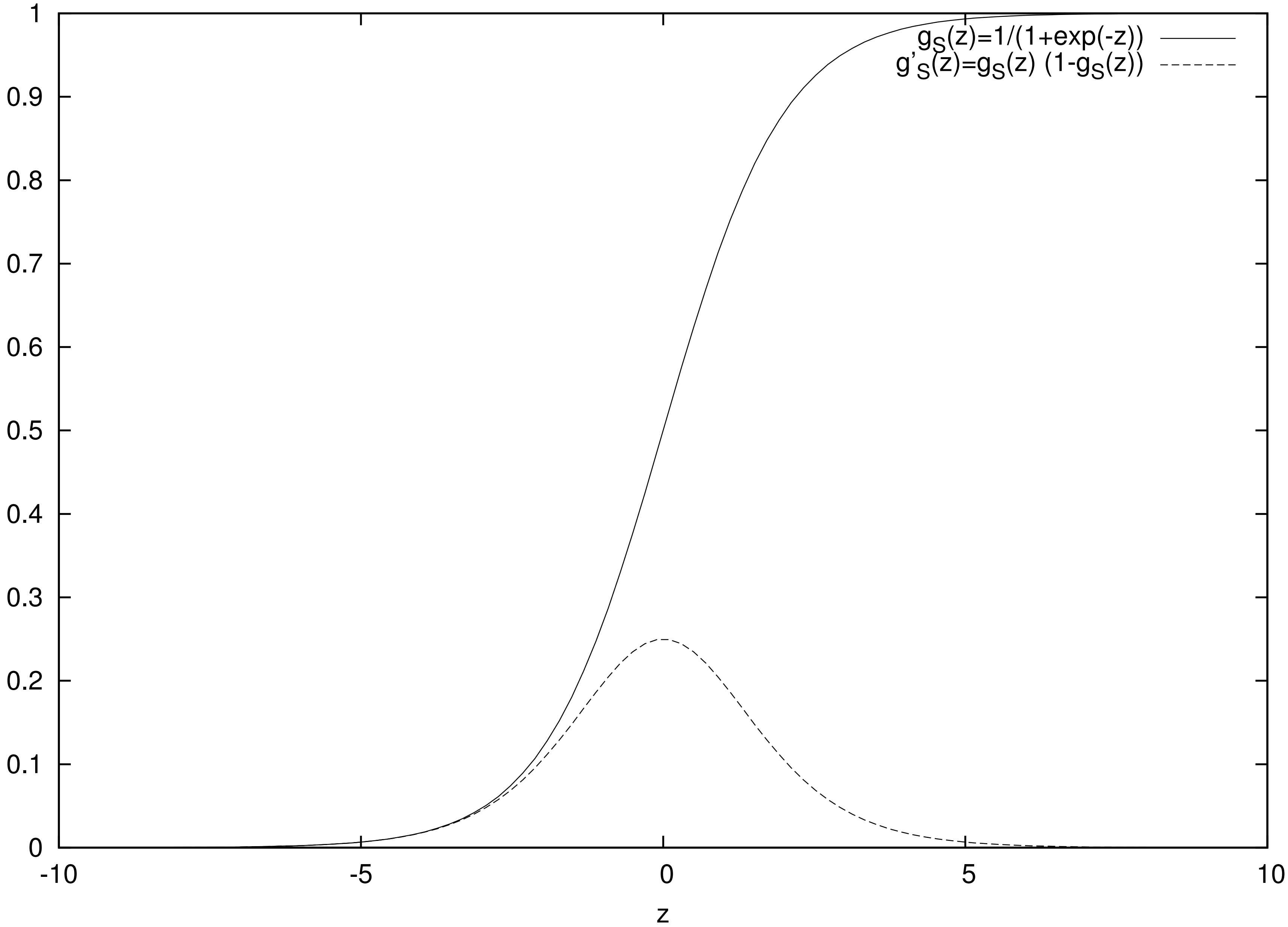
Ejercicios: Demostrar $\;g'_S(z) = g_S(z)\,(1 - g_S(z))\,,\;\; g'_T(z) = 2g_S(2z) - 1 \quad \forall z \in \mathbb{R}$

† sgn es la *función signo*

Funciones de activación

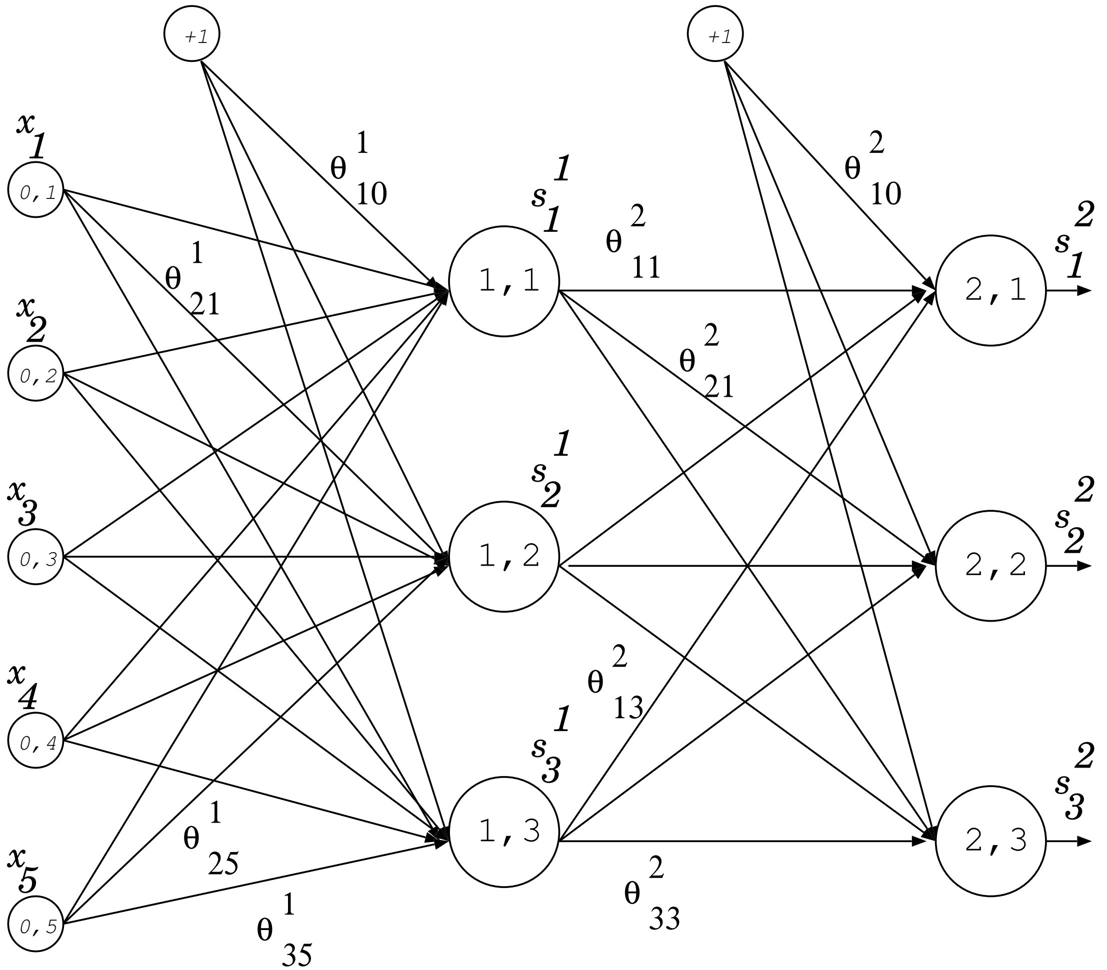


Derivada de la función de activación sigmoid



Un perceptrón de dos capas: ejemplo y notación

Topología



Dinámica

Capa de entrada	Capa oculta	Capa de salida
$1 \leq i \leq M_0 \equiv d = 5$	$1 \leq i \leq M_1 = 3$	$1 \leq i \leq M_2 = 3$
$x_i \in \mathbb{R}$	$s_i^1(\boldsymbol{x};\boldsymbol{\Theta}) = g(\sum_{j=0}^{M_0} \theta_{ij}^1 \; x_j)$	$s_i^2(\boldsymbol{x};\boldsymbol{\Theta}) = g(\sum_{j=0}^{M_1} \theta_{ij}^2 \; s_j^1(\boldsymbol{x}))$

Perceptrón de dos capas

- *Un perceptrón de dos capas* consiste en una combinación de FDLA agrupadas en 2 capas de tallas M_1 (capa *oculta*) y M_2 (capa de salida), más una capa de entradas de talla $M_0 = d$ (por simplicidad no se contabilizan los umbrales):

$$s_i^2(\boldsymbol{x};\boldsymbol{\Theta}) = g(\sum_{j=0}^{M_1} \theta_{ij}^2 \; s_j^1(\boldsymbol{x};\boldsymbol{\Theta})) = g(\sum_{j=0}^{M_1} \theta_{ij}^2 \; g(\sum_{j'=0}^{M_0} \theta_{jj'}^1 \; x_{j'})) \quad 1 \leq i \leq M_2$$

  Los parámetros son $\boldsymbol{\Theta} = [\theta_{10}^1, \ldots, \theta_{M_1 M_0}^1, \theta_{10}^2, \ldots, \theta_{M_2 M_1}^2] \in \mathbb{R}^D$

- *Problema*: Dado un conjunto de entrenamiento $S = \{(\boldsymbol{x}_1, \boldsymbol{t}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{t}_N)\}$, con $\boldsymbol{x}_n \in \mathbb{R}^{M_0}$, $\boldsymbol{t}_n \in \mathbb{R}^{M_2}$, encontrar $\boldsymbol{\Theta}$ tal que $\boldsymbol{s}^2(\boldsymbol{x}_n;\boldsymbol{\Theta})$ aproxime lo mejor posible a $\boldsymbol{t}_n \; \forall n, 1 \leq n \leq N$.

- *En clasificación*: $M_2 \equiv C$ y las etiquetas $\boldsymbol{t}_n$ para $1 \leq n \leq N$ son de la forma

$$1 \leq c \leq C \quad t_{nc} = \begin{cases} 1 & \boldsymbol{x}_n \text{ es de la clase } c \\ 0 \;\; (\text{o } -1) & \boldsymbol{x}_n \text{ no es de la clase } c \end{cases}$$

Simplificaciones de notación: $s_i^k(\boldsymbol{x};\boldsymbol{\Theta}) \equiv s_i^k(\boldsymbol{x}) \equiv s_i^k \;\; \forall k, i$

El perceptrón multicapa y las funciones de activación

- Un perceptrón multicapa de dos capas define una función $\mathbb{R}^{M_0} \to \mathbb{R}^{M_2}$:

$$s_i^2(\boldsymbol{x}) = g(\sum_{j=0}^{M_1} \theta_{ij}^2 \; s_j^1(\boldsymbol{x})) = g(\sum_{j=0}^{M_1} \theta_{ij}^2 \; g(\sum_{j'=0}^{M_0} \theta_{jj'}^1 \; x_{j'})) \text{ para } 1 \leq i \leq M_2$$

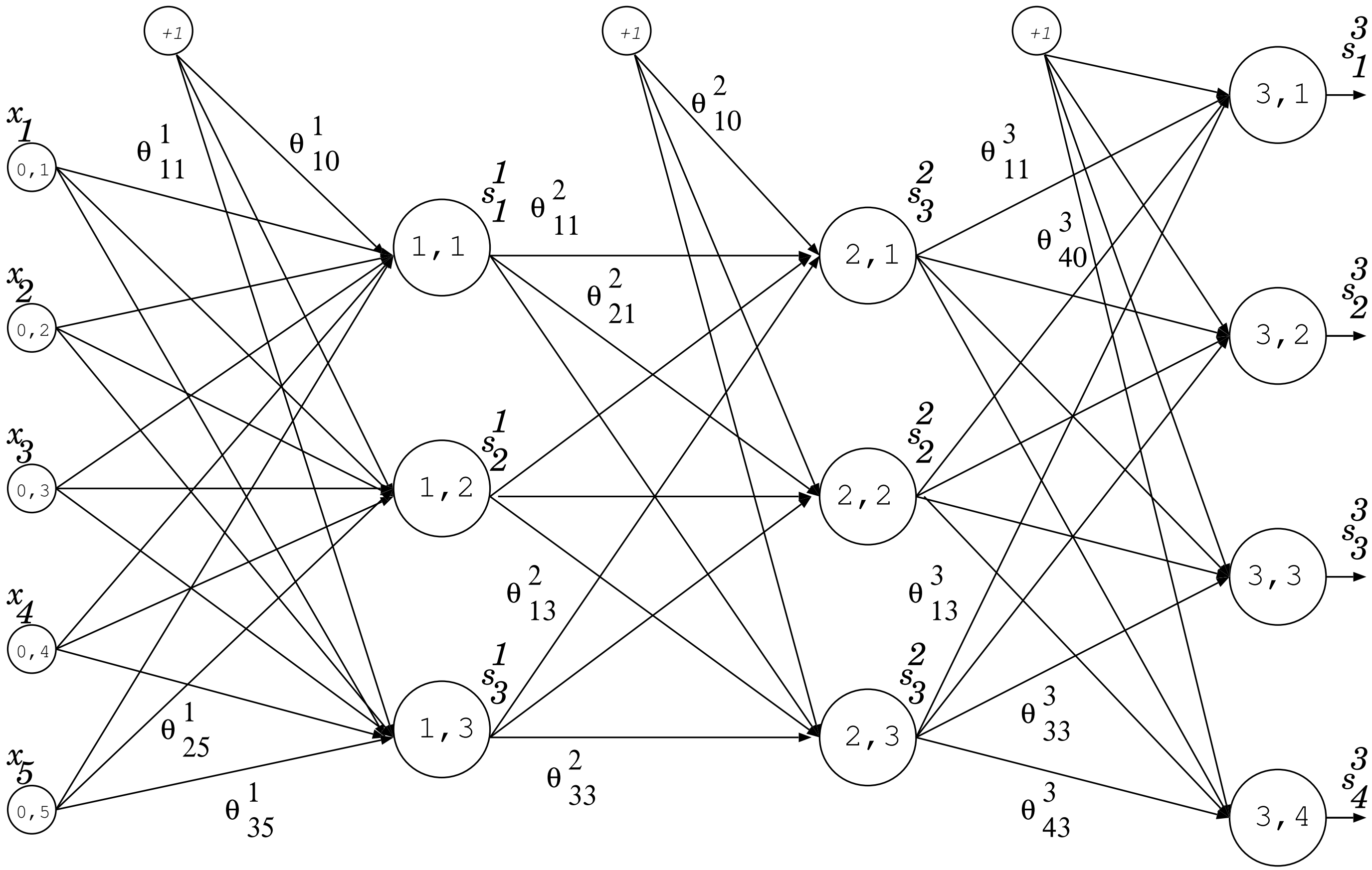
- Si todas las funciones de activación son lineales, un perceptrón multicapa define UNA FUNCIÓN DISCRIMINANTE LINEAL, $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \ldots, \phi_{M_2}(\boldsymbol{x}))^t$:

$$\phi_i(\boldsymbol{x}) \equiv s_i^2(\boldsymbol{x}) \;=\; \sum_{j=0}^{M_1} \sum_{j'=0}^{M_0} \theta_{ij}^2 \theta_{jj'}^1 \, x_{j'} \;=\; \sum_{j'=0}^{M_0} (\sum_{j=0}^{M_1} \theta_{ij}^2 \theta_{jj'}^1) \, x_{j'} \;=\; \sum_{j'=0}^{M_0} \theta_{ij'} \, x_{j'}$$

- Si al menos una función de activación no es lineal (y, sin pérdida de generalidad, todas las funciones de activación de la capa de salida son lineales) un perceptrón multicapa define UNA FUNCIÓN DISCRIMINANTE LINEAL GENERALIZADA:

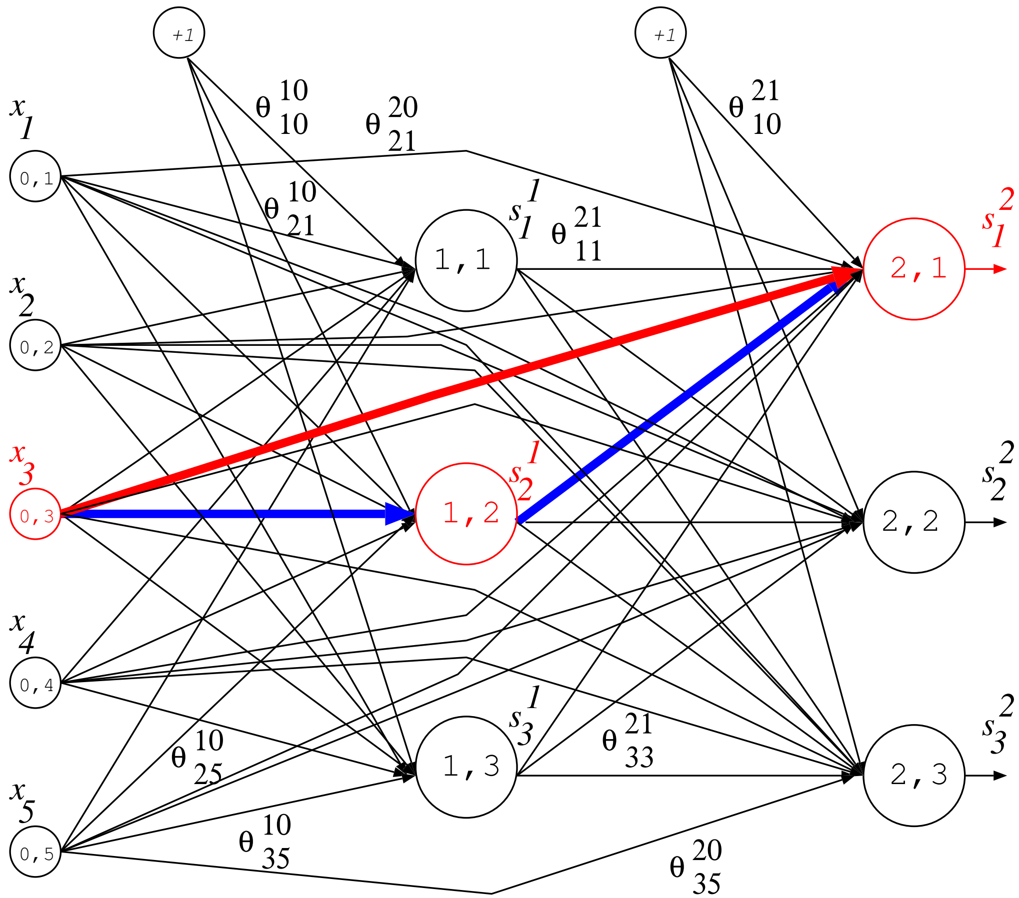
$$\phi_i(\boldsymbol{x}) \equiv s_i^2(\boldsymbol{x}) = \sum_{j=0}^{M_1} \theta_{ij}^2 \; g(\sum_{j'=0}^{M_0} \theta_{jj'}^1 \; x_{j'}) = \sum_{j=0}^{M_1} \theta_{ij}^2 \; \psi_j(\boldsymbol{x}) \text{ para } 1 \leq i \leq M_2$$

Un perceptrón de tres capas



- Primera capa oculta: $s_i^1 = g(\sum_{j=0}^{M_0} \theta_{ij}^1 \; x_j)$ para $1 \leq i \leq M_1$
- Segunda capa oculta: $s_i^2 = g(\sum_{j=0}^{M_1} \theta_{ij}^2 \; s_j^1)$ para $1 \leq i \leq M_2$
- Capa de salida: $s_i^3 = g(\sum_{j=0}^{M_2} \theta_{ij}^3 \; s_j^2)$ para $1 \leq i \leq M_3$

Redes hacia adelante de dos capas



- Primera capa oculta: $s_i^1 = g(\sum_{j=0}^{M_0} \theta_{ij}^{1,0}\; x_j)$ para $1 \leq i \leq M_1$
- Capa de salida: $s_i^2 = g(\sum_{j=0}^{M_1} \theta_{ij}^{2,1}\; s_j^1 + \sum_{j=0}^{M_0} \theta_{ij}^{2,0}\; x_j)$ para $1 \leq i \leq M_2$

El perceptrón multicapa es un caso particular

El perceptrón multicapa como regresor

Regresión de $\mathbb{R}^d$ a $\mathbb{R}^{d'}$

(p.e. un PM de dos capas con $d \equiv M_0$ y $d' \equiv M_2$)

$$\boldsymbol{f} : \mathbb{R}^{M_0} \rightarrow \mathbb{R}^{M_2} : f_i(\boldsymbol{x}) \equiv s_i^2(\boldsymbol{x}) = \sum_{j=0}^{M_1} \theta_{ij}^2 \; g(\sum_{j'=0}^{M_0} \theta_{jj'}^1 \; x_{j'}) \quad 1 \leq i \leq M_2$$

- Cualquier función se puede aproximar con precisión arbitraria mediante un perceptrón de *una* o más capas ocultas con un número de nodos suficientemente grande
- En general, para alcanzar una precisión dada, el número de nodos necesarios suele ser *mucho menor* si el número de capas ocultas es mayor o igual que *dos*
- Si se utiliza una función de activación softmax en la capa de salida, entonces el perceptrón implementa una distribución de probabilidad.

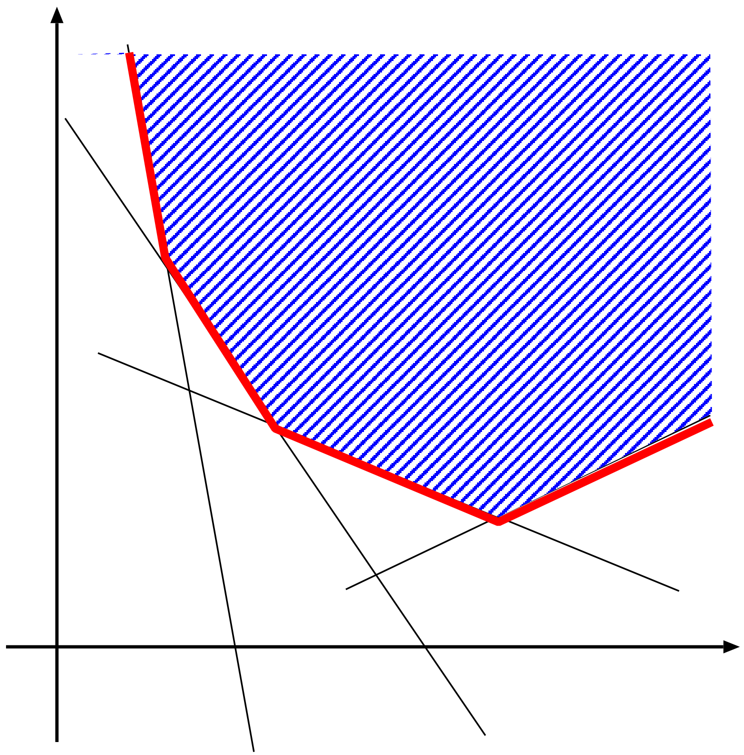
El perceptrón multicapa como clasificador

Clasificación en C clases de puntos de $\mathbb{R}^d$ (PM con $M_0 \equiv d$, $M_2 \equiv C$)

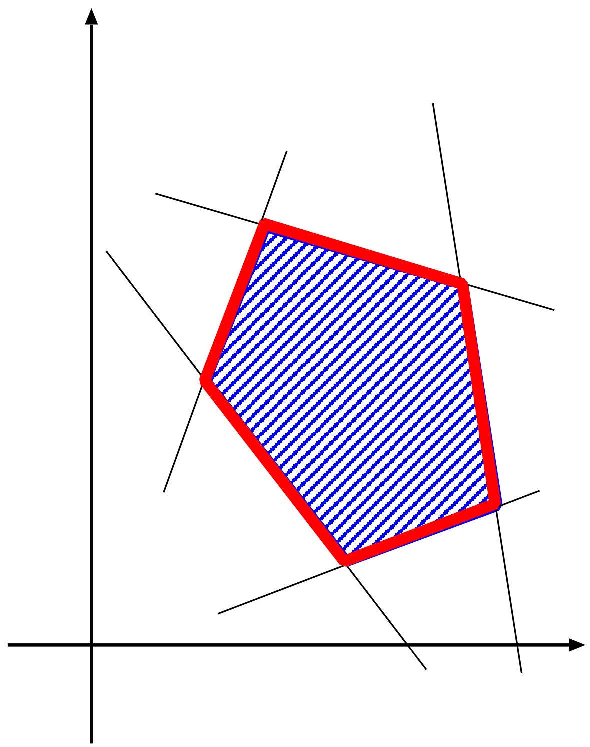
$$f : \mathbb{R}^d \to \{1, \ldots, C\} : f(\boldsymbol{x}) = \underset{1 \le c \le C}{\arg\max}\, \phi_c(\boldsymbol{x}) = \underset{1 \le c \le M_2}{\arg\max}\, s_c^2(\boldsymbol{x})$$

- Si un conjunto de entrenamiento es linealmente separable existe un perceptrón sin capas ocultas que lo clasifica correctamente.
- Un PM con *una* capa oculta de $N - 1$ nodos puede clasificar correctamente las muestras de cualquier conjunto de entrenamiento de talla N. ¿Con qué poder de generalización?.
- Cualquier frontera de decisión basada en trozos de hiperplanos puede obtenerse mediante un PM con *una* capa oculta y un número de nodos adecuado [Huang & Lippmann, 1988], [Huang, Chen & Babri, 2000].
- En general, el número de nodos necesarios para aproximar una frontera dada suele ser *mucho menor* si el número de capas ocultas es mayor o igual que *dos*.

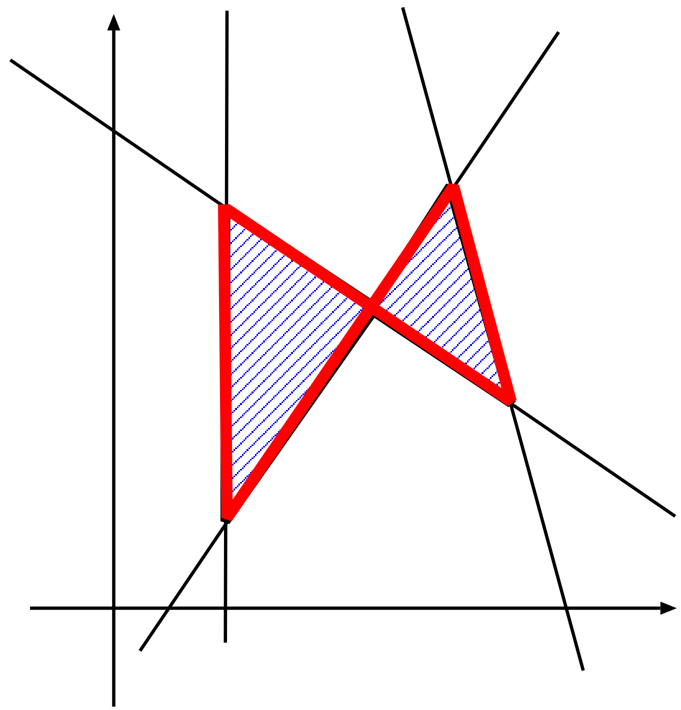
El perceptrón multicapa como clasificador



Fronteras lineales a intervalos



Regiones cerradas



Regiones no convexas

Index

1 Redes neuronales multicapa ▷ 2

○ 2 *Algoritmo de retropropagación del error (BackProp)* ▷ 19

3 Aspectos de uso y propiedades del BackProp ▷ 34

4 Notación ▷ 49

Aprendizaje de los pesos de un perceptrón multicapa

PROBLEMA: dada la topología de un perceptrón multicapa con L capas y un conjunto de entrenamiento:

$$S \;=\; \{(\boldsymbol{x}_1, \boldsymbol{t}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{t}_N)\}\,, \quad \boldsymbol{x}_n \in \mathbb{R}^{M_0},\; \boldsymbol{t}_n \in \mathbb{R}^{M_L}$$

obtener $\boldsymbol{\Theta}$ que minimice una función objetivo o "de pérdida" (*"loos"*), $q_S(\boldsymbol{\Theta})$, que mida adecuadamente la discrepancia entre las salidas de la red y los valores deseados (o *"targets"*) dados por S:

$$q_S(\boldsymbol{\Theta}) = \frac{1}{N}\sum_{n=1}^{N} q_n(\boldsymbol{\Theta})$$

donde $q_n(\boldsymbol{\Theta})$ es la pérdida de la muestra n-ésima.

SOLUCIÓN: descenso por gradiente ("algoritmo BACKPROP")

$$\Delta\theta_{ij}^{l} = -\rho\,\frac{\partial q_S(\boldsymbol{\Theta})}{\partial\theta_{ij}^{l}} \;=\; \frac{1}{N}\sum_{n=1}^{N} -\rho\,\frac{\partial q_n(\boldsymbol{\Theta})}{\partial\theta_{ij}^{l}} \;=\; \frac{1}{N}\sum_{n=1}^{N}\Delta_n\theta_{ij}^{l}$$

Por tanto, para cada muestra n-ésima, hay que calcular:

$$\Delta_n\theta_{ij}^{l} \stackrel{\text{def}}{=} -\rho\,\frac{\partial q_n(\boldsymbol{\Theta})}{\partial\theta_{ij}^{l}}\,,\;\; 1\le i\le M_l,\; 0\le j\le M_{l-1},\; 1\le l\le L,\; 1\le n\le N$$

Funciones objetivo o de pérdida

ERROR CUADRÁTICO: Adecuado en general para regresión y clasificación:

$$q_n(\mathbf{\Theta}) = \frac{1}{2} \sum_{i=1}^{M_L} \left(t_{ni} - s_i^L(\boldsymbol{x}_n; \mathbf{\Theta})\right)^2$$

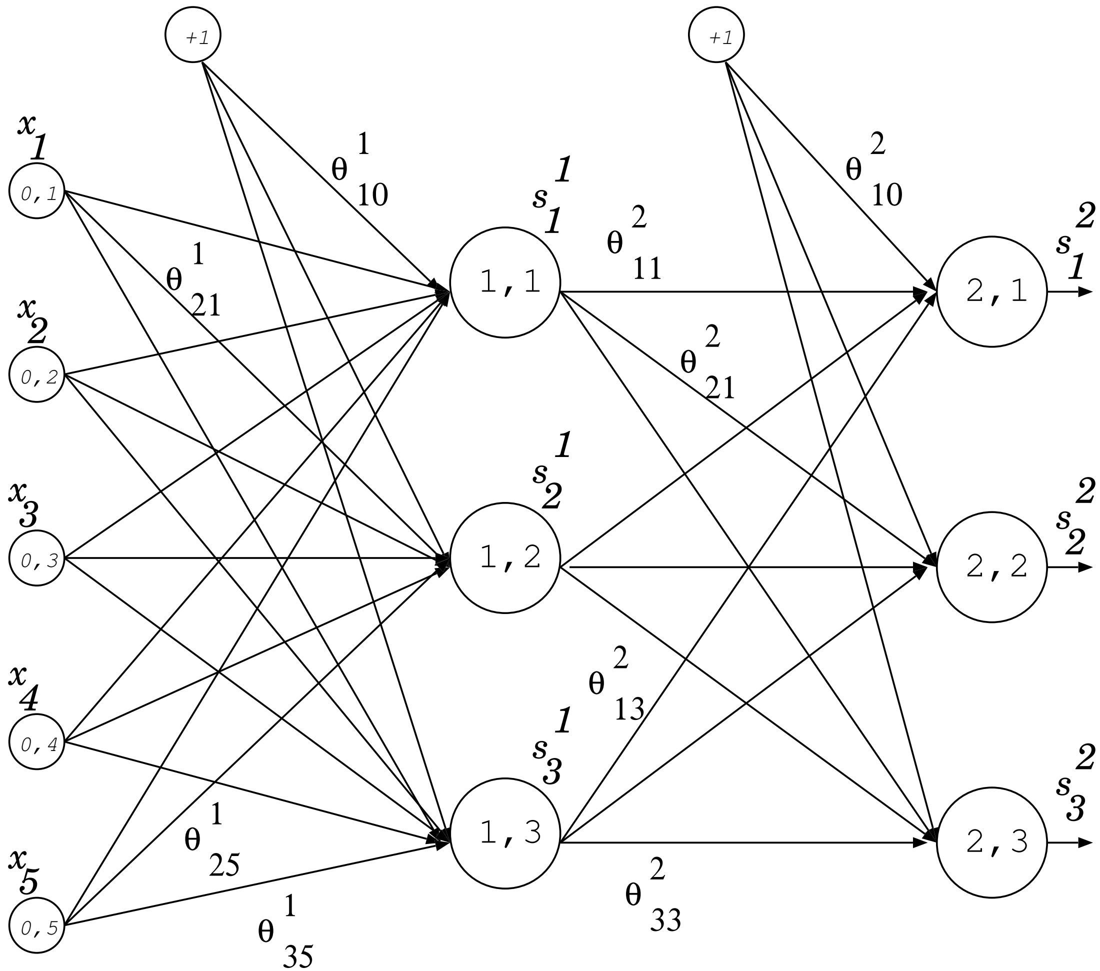
ENTROPÍA CRUZADA: Específicamente adecuada para clasificación, con activación *softmax* en la capa de salida y $M_L = C$.

$$q_n(\mathbf{\Theta}) = -\sum_{i=1}^{C} t_{ni} \, \log s_i^L(\boldsymbol{x}_n; \mathbf{\Theta})$$

En este caso, las C salidas se consideran como aproximaciones a las probabilidades a posteriori de clase. Los valores objetivo (*targets*) de cada muestra de aprendizaje en S tienen valores en $\{0, 1\}$ y se cosnideran probabilidades a posteriori de clase de la distribución "empírica" dada por S.

Para simplificar, pero sin pérdida de generalidad, en lo que sigue se asume $L = 2$.

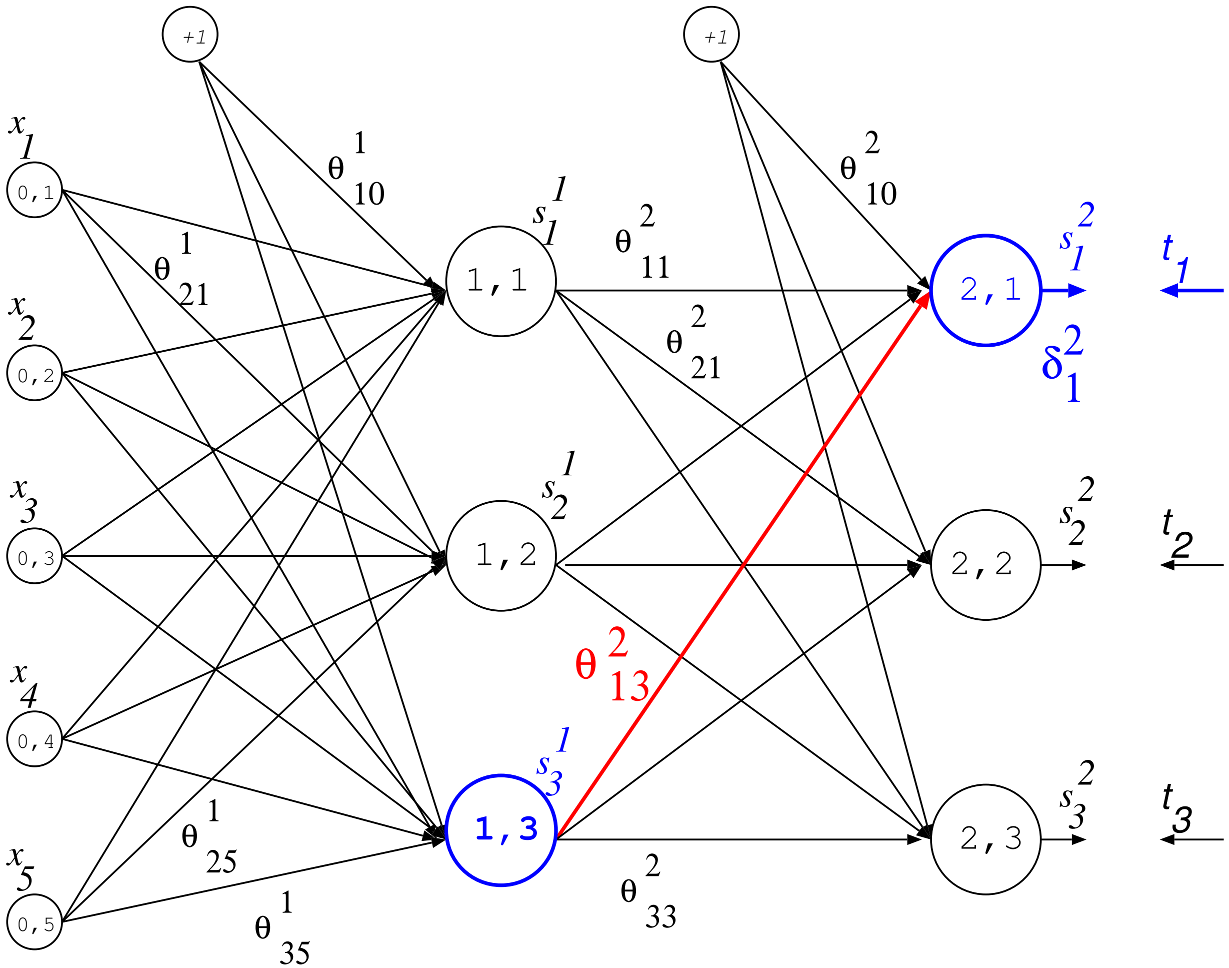
Retropropagación del error (BackProp): cálculo hacia adelante



$$s_i^1(\boldsymbol{x}) = g(\phi_i^1) = g(\sum_{j=0}^{M_0} \theta_{ij}^1\, x_j)\ \ 1 \le i \le M_1; \quad s_j^2 = g(\phi_j^2) = g(\sum_{k=0}^{M_1} \theta_{jk}^2\, s_k^1(\boldsymbol{x})),\ 1 \le j \le M_2$$

(para una muestra de entrenamiento genérica $(\boldsymbol{x}, \boldsymbol{t})$, y $M_0 = 5, M_1 = 3, M_2 = 3$)

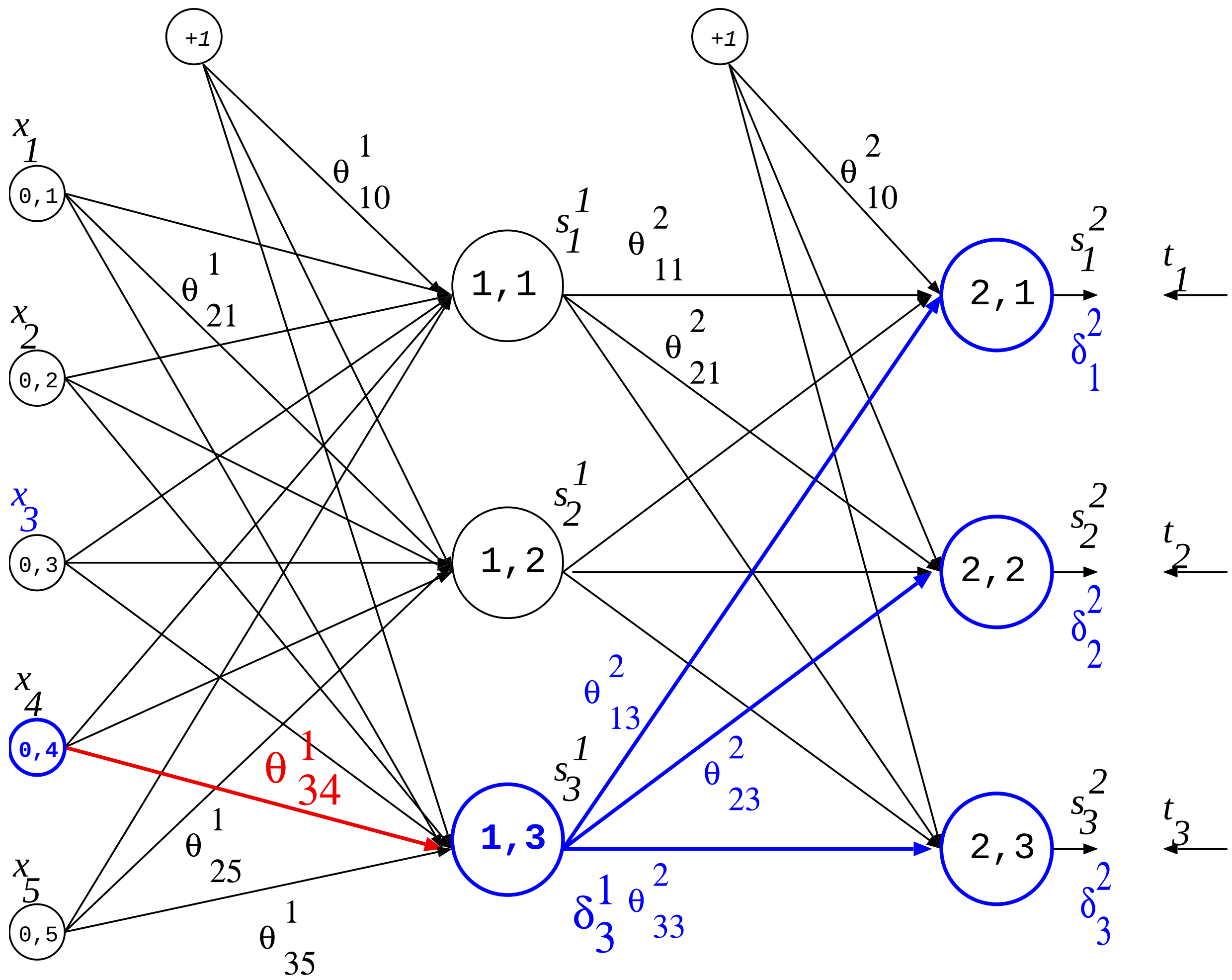
BackProp: ilustración de actualización pesos de la capa de salida



$$\Delta\theta^2_{1\,3} = \rho\, \delta^2_1\, s^1_3 = \rho\,(t_1 - s^2_1)\, g'(\phi^2_1) s^1_3$$

(para la muestra $(\boldsymbol{x}, \boldsymbol{t})$ y en el caso de regresión)

BackProp: ilustración de actualización pesos de la capa oculta



$$\Delta\theta^1_{3\,4} = \rho\,\delta^1_3\,x_4 = \rho\left(g'(\phi^1_3)\sum_{r=1}^{M_2}\delta^2_r\,\theta^2_{r3}\right)x_4$$

(para la muestra $(\boldsymbol{x}, \boldsymbol{t})$ y en el caso de regresión)

Regla de la cadena para el cálculo de derivadas

- Función simple de otra función

  $f, g : \mathbb{R} \to \mathbb{R}$:

$$\frac{d\, f(g(x))}{d\, x} = \frac{d\, f}{d\, g} \frac{d\, g}{d\, x}$$

- Función de otras dos funciones de n (o más) variables

  $g_1, g_2 : \mathbb{R}^{n \geq 2} \to \mathbb{R}$, $\quad f : \mathbb{R}^{M \geq 2} \to \mathbb{R}$

$$\frac{\partial f(g_1(x, \ldots),\ g_2(x, \ldots))}{\partial x} = \frac{\partial f}{\partial g_1} \frac{\partial g_1}{\partial x} + \frac{\partial f}{\partial g_2} \frac{\partial g_2}{\partial x}$$

- Función de N (o más) funciones de n (o más) variables

  $g_1, \ldots, g_N : \mathbb{R}^{n \geq N} \to \mathbb{R}$, $\quad f : \mathbb{R}^{M \geq N} \to \mathbb{R}$:

$$\frac{\partial f(g_1(x, \ldots), \ldots, g_N(x, \ldots))}{\partial x} = \sum_{i=1}^{N} \frac{\partial f}{\partial g_i} \frac{\partial g_i}{\partial x}$$

Derivación del algoritmo BackProp para regresión (I)

Actualización de los pesos de la capa de salida θ^2_{ij}, para una muestra genérica $(\boldsymbol{x}, \boldsymbol{t}) \equiv (\boldsymbol{x}_n, \boldsymbol{t}_n)$:

$$q(\boldsymbol{\Theta}) \equiv q_n(\boldsymbol{\Theta}) \;=\; \frac{1}{2}\sum_{l=1}^{M_2}\left(t_l - s^2_l\right)^2; \quad s^2_l = g\left(\phi^2_l\right); \quad \phi^2_l = \sum_{m=0}^{M_1} \theta^2_{lm} s^1_m$$

$$\frac{\partial q}{\partial \theta^2_{ij}} \;=\; \frac{\partial q}{\partial s^2_i}\;\frac{d\, s^2_i}{d\,\phi^2_i}\;\frac{\partial \phi^2_i}{\partial \theta^2_{ij}}$$

$$\downarrow \qquad \downarrow \qquad \downarrow$$

$$= \; -\big((t_i - s^2_i)\, g'(\phi^2_i)\big)\, s^1_j \;\stackrel{\text{def}}{=}\; -\delta^2_i\, s^1_j$$

$$\frac{\partial q}{\partial \theta^2_{ij}} = -\delta^2_i\, s^1_j\,, \qquad \delta^2_i \stackrel{\text{def}}{=} (t_i - s^2_i)\, g'(\phi^2_i)$$

$$\boxed{\Delta_n \theta^2_{ij} \;=\; -\rho\, \frac{\partial q_n}{\partial \theta^2_{ij}} \;=\; \rho\, \delta^2_i\, s^1_j \qquad 1 \le i \le M_2,\; 0 \le j \le M_1}$$

Derivación del algoritmo BackProp para regresión (II)

Actualización de los pesos de la capa oculta θ^1_{ij}, para una muestra genérica $(\boldsymbol{x}, \boldsymbol{t}) \equiv (\boldsymbol{x}_n, \boldsymbol{t}_n)$:

$$q(\boldsymbol{\Theta}) = \frac{1}{2}\sum_{l=1}^{M_2}\left(t_l - s^2_l\right)^2; \quad s^2_l = g(\phi^2_l); \quad \phi^2_l = \sum_{m=0}^{M_1}\theta^2_{lm}\, s^1_m; \quad s^1_m = g(\phi^1_m); \quad \phi^1_m = \sum_{k=0}^{M_0}\theta^1_{mk} x_k$$

$$\frac{\partial q}{\partial \theta^1_{ij}} = \sum_{r=1}^{M_2} \frac{\partial q}{\partial s^2_r}\,\frac{d\, s^2_r}{d\,\phi^2_r}\,\frac{\partial \phi^2_r}{\partial s^1_i}\,\frac{d\, s^1_i}{d\,\phi^1_i}\,\frac{\partial \phi^1_i}{\partial \theta^1_{ij}}$$

$$\downarrow \qquad \downarrow \qquad \downarrow \qquad \downarrow$$

$$= \sum_{r=1}^{M_2} -\delta^2_r \qquad \theta^2_{ri}\;\; g'(\phi^1_i)\; x_j \;= -\left(g'(\phi^1_i)\sum_{r=1}^{M_2}\delta^2_r\;\theta^2_{ri}\right)x_j \overset{\text{def}}{=} -\,\delta^1_i\, x_j$$

$$\frac{\partial q}{\partial \theta^1_{ij}} = -\delta^1_i\; x_j\,, \qquad \delta^1_i \overset{\text{def}}{=} g'(\phi^1_i)\sum_{r=1}^{M_2}\delta^2_r\;\theta^2_{ri}$$

$$\boxed{\Delta_n\theta^1_{ij} = -\rho\,\frac{\partial q_n}{\partial \theta^1_{ij}} = \rho\,\delta^1_i\, x_j \qquad 1\leq i\leq M_1,\; 0\leq j\leq M_0}$$

Ecuaciones BackProp para N muestras de aprendizaje

- Actualización de los pesos de la capa de salida: $(1 \leq i \leq M_2,\ 0 \leq j \leq M_1)$:

$$\Delta\theta^2_{ij} = -\rho \, \frac{\partial q_S(\boldsymbol{\Theta})}{\partial \theta^2_{ij}} = \frac{\rho}{N} \sum_{n=1}^{N} \delta^2_i(\boldsymbol{x}_n)\; s^1_j(\boldsymbol{x}_n)$$

$$\delta^2_i(\boldsymbol{x}_n) = \big(t_{ni} - s^2_i(\boldsymbol{x}_n)\big)\; g'(\phi^2_i(\boldsymbol{x}_n)) \;\text{ con }\; \phi^2_i(\boldsymbol{x}_n) = \sum_{j=0}^{M_1} \theta^2_{ij} s^1_j(\boldsymbol{x}_n)$$

- Actualización de los pesos de la capa oculta $(1 \leq i \leq M_1,\ 0 \leq j \leq M_0)$:

$$\Delta\theta^1_{ij} = -\rho \, \frac{\partial q_S(\boldsymbol{\Theta})}{\partial \theta^1_{ij}} = \frac{\rho}{N} \sum_{n=1}^{N} \delta^1_i(\boldsymbol{x}_n)\; x_{nj}$$

$$\delta^1_i(\boldsymbol{x}_n) = \Big(\sum_{r=1}^{M_2} \delta^2_r(\boldsymbol{x}_n)\; \theta^2_{ri}\Big)\; g'(\phi^1_i(\boldsymbol{x}_n)) \;\text{ con }\; \phi^1_i(\boldsymbol{x}_n) = \sum_{j=0}^{M_0} \theta^1_{ij} x_{nj}$$

Ejercicio: derivar las ecuaciones BackProp para el caso de clasificación.

Ecuaciones BackProp para perceptrones de tres capas

- Actualización de los pesos de la capa de salida $(1 \le i \le M_3,\ 0 \le j \le M_2)$

$$\Delta\theta^3_{ij} = -\rho\,\frac{\partial q_S(\boldsymbol{\Theta})}{\partial\theta^3_{ij}} = \frac{\rho}{N}\sum_{n=1}^{N}\delta^3_i(\boldsymbol{x}_n)\ s^2_j(\boldsymbol{x}_n) \qquad \delta^3_i(\boldsymbol{x}_n) = \Big(t_{ni} - s^3_i(\boldsymbol{x}_n)\Big)\ g'(\phi^3_i(\boldsymbol{x}_n))$$

- Actualización de los pesos de la segunda capa oculta ($1 \le i \le M_2,\ 0 \le j \le M_1$)

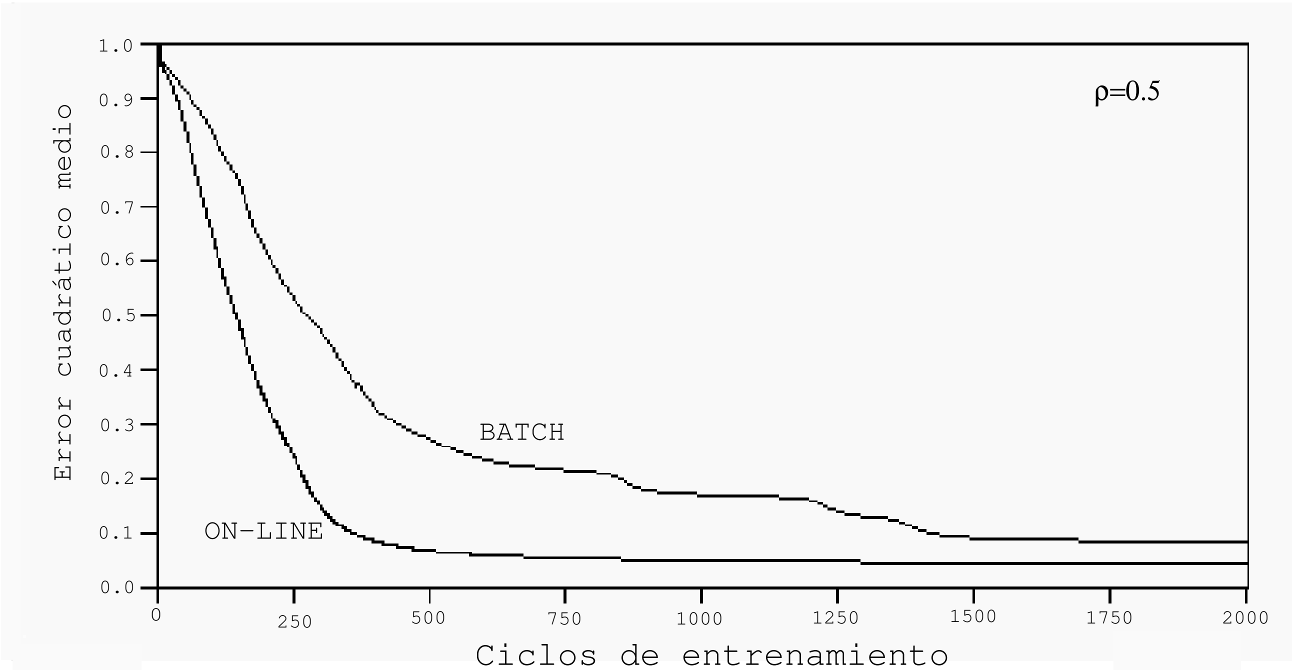
$$\Delta\theta^2_{ij} = -\rho\,\frac{\partial q_S(\boldsymbol{\Theta})}{\partial\theta^2_{ij}} = \frac{\rho}{N}\sum_{n=1}^{N}\delta^2_i(\boldsymbol{x}_n)\ s^1_j(\boldsymbol{x}_n) \qquad \delta^2_i(\boldsymbol{x}_n) = \Big(\sum_{r=1}^{M_3}\delta^3_r(\boldsymbol{x}_n)\ \theta^3_{ri}\Big)\ g'(\phi^2_i(\boldsymbol{x}_n))$$

- Actualización de los pesos de la primera capa oculta ($1 \le i \le M_1,\ 0 \le j \le M_0$)

$$\Delta\theta^1_{ij} = -\rho\,\frac{\partial q_S(\boldsymbol{\Theta})}{\partial\theta^1_{ij}} = \frac{\rho}{N}\sum_{n=1}^{N}\delta^1_i(\boldsymbol{x}_n)\ x_{nj} \qquad \delta^1_i(\boldsymbol{x}_n) = \Big(\sum_{r=1}^{M_2}\delta^2_r(\boldsymbol{x}_n)\ \theta^2_{ri}\Big)\ g'(\phi^1_i(\boldsymbol{x}_n))$$

BackProp incremental o *"batch"*

- BackProp "batch": en cada iteración se procesan las N muestras de entrenamiento (" epoch") y los pesos del PM se actualizan una sola vez.
- BackProp "*mini-batch*": el conjunto de entrenamiento se divide en B bloques, en cada uno se procesan las muestras que están contenidas y luego se actualizan los pesos del PM. Por tanto en un epoch los pesos se actualizan N/B veces.
- BackProp "*incremental*": en cada iteración se procesa solo una muestra de entrenamiento (aleatoria) y se actualizan los pesos del PM. Por tanto en un epoch los pesos se actualizan N veces.



Algoritmo BACKPROP

Entrada: Topología, pesos iniciales $\theta^l_{ij},\ 1 \leq l \leq L,\ 1 \leq i \leq M_l,\ 0 \leq j \leq M_{l-1}$,
factor de aprendizaje ρ, condiciones de convergencia, N datos de entrenamiento S

Salidas: Pesos de las conexiones que minimizan el error cuadrático medio de S

Mientras no se cumplan las condiciones de convergencia

Para $1 \leq l \leq L, 1 \leq i \leq M_l, 0 \leq j \leq M_{l-1}$, inicializar $\Delta\theta^l_{ij} = 0$

Para cada muestra de entrenamiento $(\boldsymbol{x}, \boldsymbol{t}) \in S$

Desde la capa de entrada a la de salida ($l = 0, \ldots, L$):

Para $1 \leq i \leq M_l$ si $l = 0$ entonces $s^0_i = x_i$ sino calcular ϕ^l_i y $s^l_i = g(\phi^l_i)$

Desde la capa de salida a la de entrada ($l = L, \ldots, 1$),

Para cada nodo ($1 \leq i \leq M_l$)

Calcular $\delta^l_i = \begin{cases} g'(\phi^l_i)\ (t_{ni} - s^L_i) & \text{si } l == L \\ g'(\phi^l_i)\ (\sum_r \delta^{l+1}_r\ \theta^{l+1}_{ri}) & \text{en otro caso} \end{cases}$

Para cada peso θ^l_{ij} ($0 \leq j \leq M_{l-1}$) calcular: $\Delta\theta^l_{ij} = \Delta\theta^l_{ij} + \rho\ \delta^l_i\ s^{l-1}_j$

Para $1 \leq l \leq L, 1 \leq i \leq M_l, 0 \leq j \leq M_{l-1}$, actualizar pesos: $\theta^l_{ij} = \theta^l_{ij} + \frac{1}{N}\ \Delta\theta^l_{ij}$

Coste computacional por cada iteración *mientras*: $O(N\ D)$, $N = |S|$, $D =$ número de pesos

DEMO: http://playground.tensorflow.org/

Algoritmo BACKPROP ("incremental")

Entrada: Topología, pesos iniciales $\theta^l_{ij},\ 1 \leq l \leq L,\ 1 \leq i \leq M_l,\ 0 \leq j \leq M_{l-1}$,
factor de aprendizaje ρ, condiciones de convergencia, N datos de entrenamiento S

Salidas: Pesos de las conexiones que minimizan el error cuadrático medio de S

Mientras no se cumplan las condiciones de convergencia

Para cada muestra de entrenamiento $(\boldsymbol{x}, \boldsymbol{t}) \in S$ (en orden aleatorio)

Desde la capa de entrada a la de salida ($l = 0, \ldots, L$):

Para $1 \leq i \leq M_l$ si $l = 0$ entonces $s^0_i = x_i$ sino calcular ϕ^l_i y $s^l_i = g(\phi^l_i)$

Desde la capa de salida a la de entrada ($l = L, \ldots, 1$),

Para cada nodo ($1 \leq i \leq M_l$)

Calcular $\delta^l_i = \begin{cases} g'(\phi^l_i)\ (t_{ni} - s^L_i) & \text{si } l == L \\ g'(\phi^l_i)\ (\sum_r \delta^{l+1}_r\ \theta^{l+1}_{ri}) & \text{en otro caso} \end{cases}$

Para cada peso θ^l_{ij} ($0 \leq j \leq M_{l-1}$) calcular: $\Delta\theta^l_{ij} = \rho\ \delta^l_i\ s^{l-1}_j$

Para $1 \leq l \leq L, 1 \leq i \leq M_l, 0 \leq j \leq M_{l-1}$, actualizar pesos: $\theta^l_{ij} = \theta^l_{ij} + \frac{1}{N}\ \Delta\theta^l_{ij}$

Coste computacional por cada iteración *mientras*: $O(N\ D)$, $N = |S|$, $D =$ número de pesos

Algoritmo BACKPROP (“on-line”)

Entrada: Topología, pesos iniciales $\theta^l_{ij},\ 1\leq l\leq L,\ 1\leq i\leq M_l,\ 0\leq j\leq M_{l-1}$,
factor de aprendizaje ρ, dato de entrenamiento $(\boldsymbol{x}, \boldsymbol{t})$

Salidas: Pesos de las conexiones actualizados mediante $(\boldsymbol{x}, \boldsymbol{t})$

Desde la capa de entrada a la de salida ($l = 0, \ldots, L$):

Para $1 \leq i \leq M_l$ si $l = 0$ entonces $s^0_i = x_i$ sino calcular ϕ^l_i y $s^l_i = g(\phi^l_i)$

Desde la capa de salida a la de entrada ($l = L, \ldots, 1$),

Para cada nodo ($1 \leq i \leq M_l$)

Calcular $\delta^l_i = \begin{cases} g'(\phi^l_i)\ (t_{ni} - s^L_i) & \text{si } l == L \\ g'(\phi^l_i)\ \left(\sum_r \delta^{l+1}_r\ \theta^{l+1}_{ri}\right) & \text{en otro caso} \end{cases}$

Para cada peso θ^l_{ij} ($0 \leq j \leq M_{l-1}$) calcular: $\Delta\theta^l_{ij} = \rho\ \delta^l_i\ s^{l-1}_j$

Para $1 \leq l \leq L, 1 \leq i \leq M_l, 0 \leq j \leq M_{l-1}$, actualizar pesos: $\theta^l_{ij} = \theta^l_{ij} + \Delta\theta^l_{ij}$

Coste computacional por cada muestra procesada: $O(D)$, $D =$ número de pesos

Index

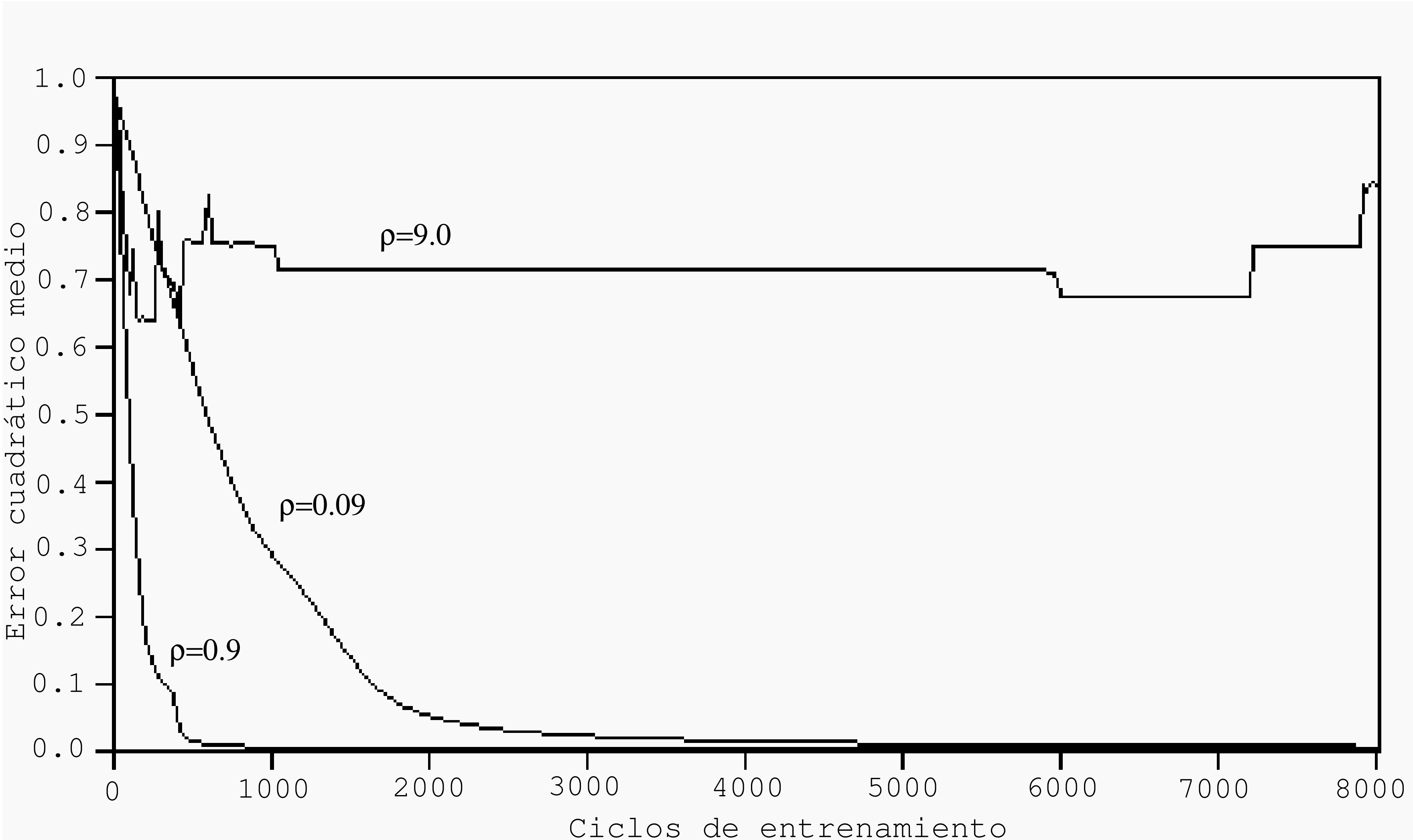
1 Redes neuronales multicapa ▷ 2

2 Algoritmo de retropropagación del error (BackProp) ▷ 19

○ 3 *Aspectos de uso y propiedades del BackProp* ▷ 34

4 Notación ▷ 49

Selección del factor de aprendizaje

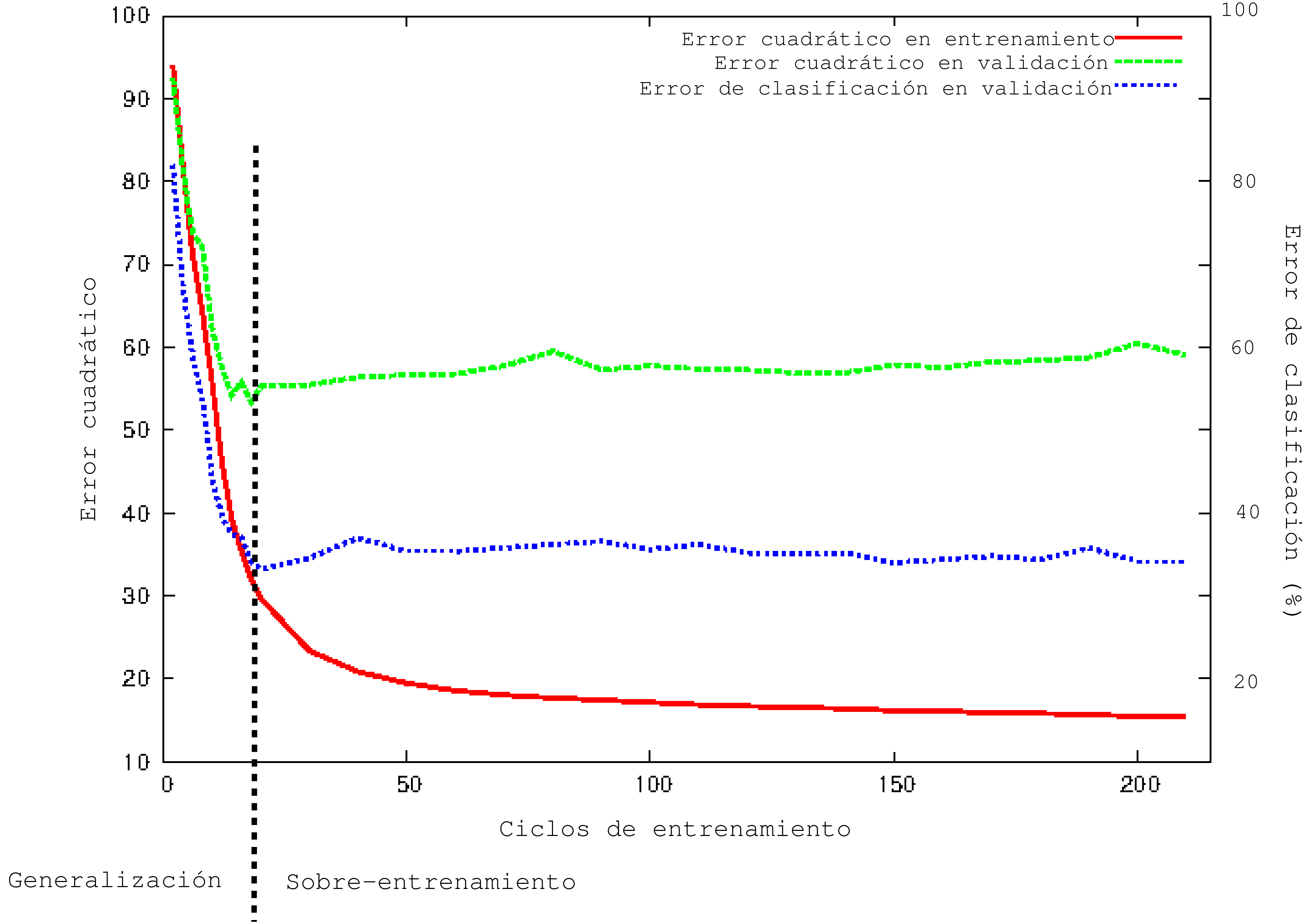


http://playground.tensorflow.org/

Algoritmos de optimización

- SDG (stochastic gradient descent).
- SGD with momentum
- Adagrad (Adaptive Gradient)
- Adadelta (an extension of Adagrad)
- ADAM (Adaptive Moment Estimation)
- NAG, RMSProp, AdaMax, Nadam, ...

Condiciones de convergencia



Algunos problemas y soluciones con el BackProp

- El problema de la anulación o explosión del gradiente en el caso de muchas capas:
  - Uso de funciones de activación como Relu, LeakyRelu/PreLU, Maxout, ELU, ...
  - Regularización.
  - Drop-out: Durante el entrenamiento se seleccionan aleatoriamente un subconjunto de nodos y no son utilizados en una iteración.
  - Conexiones residuales.
  - Normalización a nivel de batch y/o de capa.
  - Gradiente recortado (clipping).

- Evitar “malos” mínimos locales:
  - Barajar los datos de entrenamiento.
  - Aprender primero las muestras más “fáciles” (Curriculum learning).
  - Regularización.
  - Añadir ruido durante el entrenamiento.

Funciones de activación (1)

- Maxout: $f(z) = \max(w_1\, z + b_1, w_2\, z + b_2), \quad z \in \mathbb{R}$

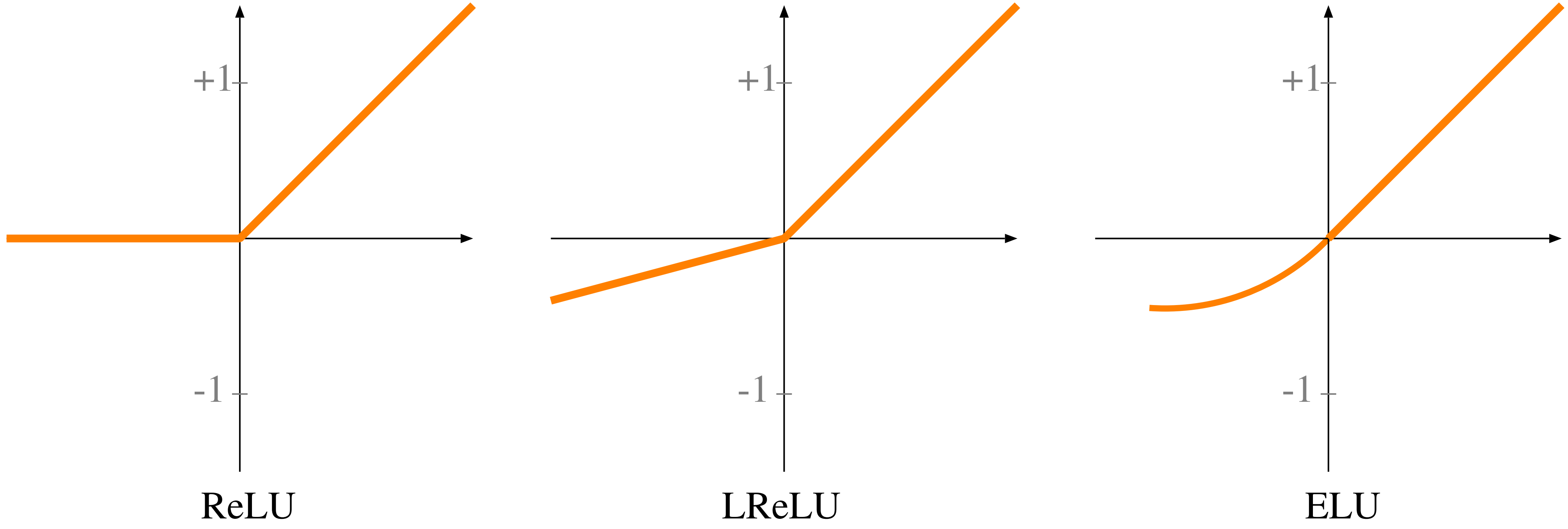
  Casos particulares relevantes:

  - Relu (Rectified Linear Unit): $f(z) = \max(0, z), \quad z \in \mathbb{R}$
  - PreLU (Parametric Rectified Linear Unit): $f(z) = \max(\alpha\, z, z), \quad z \in \mathbb{R}$

- ELU (Exponential Linear Units) : $f(z) = \begin{cases} z & z \geq 0 \\ \alpha\,(e^z - 1) & z < 0 \end{cases}, \quad z \in \mathbb{R}$

- Softmax: Para $z_1, \ldots, z_I$ con $z_i \in \mathbb{R}\ \ 1 \leq i \leq I,\ \ f(z_i) = \dfrac{e^{z_i}}{\sum_j e^{z_j}}$.

Funciones de activación (2)



Normalización de las entradas

- Parálisis de la red: para valores grandes de z la derivada $g'(z)$ es muy pequeña y por tanto, los incrementos de los pesos son muy pequeños. Una forma de disminuir este efecto es normalizar el rango de entrada.

$$S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^d \Rightarrow \begin{cases} \mu_j = \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} x_{ij} \;\; 1 \leq j \leq d \\ \sigma_j^2 = \dfrac{1}{N-1}\displaystyle\sum_{i=1}^{N}(x_{ij}-\mu_j)^2 \;\; 1 \leq j \leq d \end{cases}$$

$$\forall \boldsymbol{x} \in \mathbb{R}^d, \hat{\boldsymbol{x}} : \hat{x}_j = \frac{x_j - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}} \Rightarrow \begin{cases} \hat{\mu}_j = 0 \\ \hat{\sigma}_j = 1 \end{cases} \text{for } 1 \leq j \leq d$$

donde ϵ es una constante pequeña para evitar división por cero.

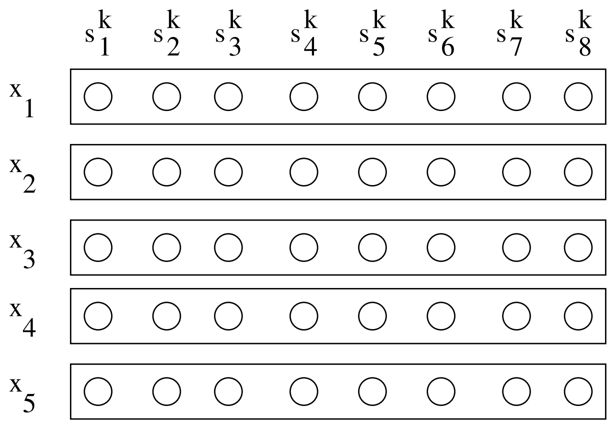
Normalización a nivel de capa

- En la capa k se dispone de las correspondiente salida $s^k \in \mathbb{R}^{M_k}$

$$\mu_L = \frac{1}{M_k} \sum_{i=1}^{M_k} s_i^k$$

$$\sigma_L^2 = \frac{1}{M_k - 1} \sum_{i=1}^{M_k} (s_i^k - \mu_L)^2$$

$$\forall s_i^k \in \mathbb{R}^{M_k}, \hat{s}_i^k := \frac{s_i^k - \mu_L}{\sqrt{\sigma_L^2 + \epsilon}} \text{ for } 1 \leq i \leq M_k$$

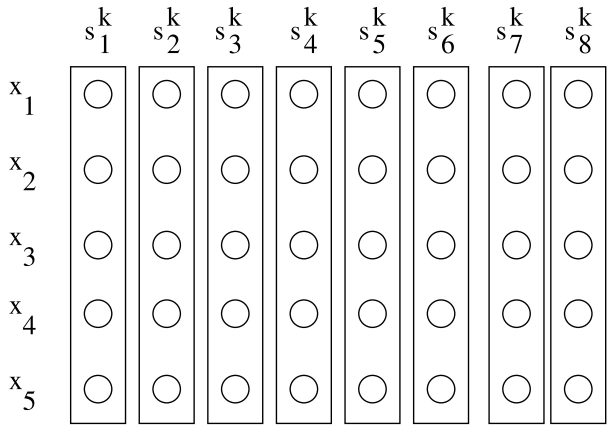


Normalización a nivel de minibatch

- En la capa k se dispone de las correspondientes salidas para un minibatch (subconjunto de S) de tamaño b: $B = \{\boldsymbol{s}_1^k, \ldots, \boldsymbol{s}_b^k\} \subset \mathbb{R}^{M_k}$

$$\boldsymbol{\mu}_B = \frac{1}{b}\sum_{i=1}^{b} \boldsymbol{s}_i^k$$
$$\sigma_B^2 = \frac{1}{b-1} \parallel \boldsymbol{s}_i^k - \boldsymbol{\mu}_B \parallel^2$$

$$\forall \boldsymbol{s}^k \in B, \quad \bar{\boldsymbol{s}}^k = \frac{\boldsymbol{s}^k - \boldsymbol{\mu}_B}{\sqrt{\sigma_B^2 + \epsilon}} \rightarrow \hat{\boldsymbol{s}}^k = \gamma\, \bar{\boldsymbol{s}}^k + \beta$$



Recorte del gradiente y Regularización

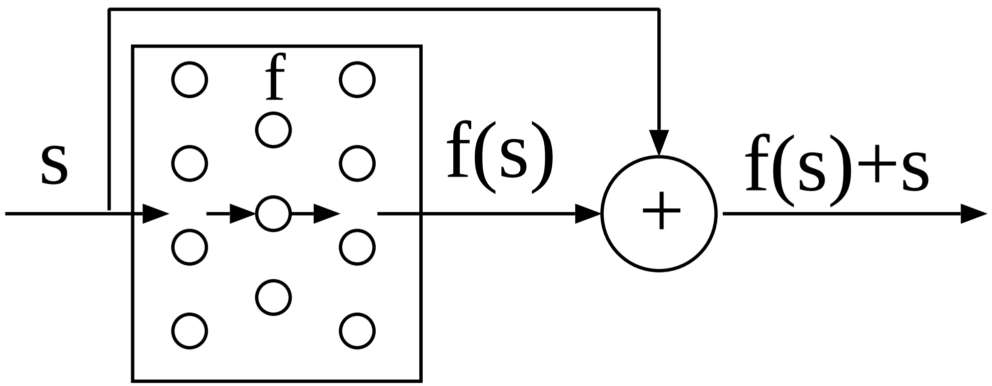
Problema: La actualización de los pesos durante el entrenamiento puede provocar un desbordamiento numérico, a menudo denominado ”gradiente explosivo” [Brownlee 2019]. Esto puede provocar una “parálisis de la red”.

Soluciones:

- Gradiente escalado: normalizar el vector de gradiente de manera que su norma (magnitud) sea igual a un valor definido, como 1.0.
- Recorte de gradiente: forzar los valores del gradiente (por elementos) a un valor mínimo o máximo si el gradiente excede un rango prefijado.
- Regularización: añadir a la función objetivo a minimizar un término relacionado con la suma de magnitudes de los pesos:
  - Regularización $L_2 : q_S(\boldsymbol{\Theta}) + \frac{\lambda}{2} \sum_{l,i,j} (\theta_{ij}^l)^2$
  - Regularización $L_1 : q_S(\boldsymbol{\Theta}) + \lambda \sum_{l,i,j} |\theta_{ij}^l|$

Conexión residual y drop-out

- Conexión residual: Si una serie de capas de una red neuronal implementa una función $s \rightarrow f(s)$, y asumiendo que s y $f(s)$ son de la misma dimensionalidad, un "conexión residual" permite implementar $f(s) + s$



- Drop-out: Durante el entrenamiento se seleccionan aleatoriamente un subconjunto de nodos y no son utilizados en una iteración.

Aumento de datos artificiales

- Aumentar el tamaño con muestras sintécticas a partir de reales mediante transformaciones.
  - En imágenes:
    - Giros horizontales.
    - Rotación.
    - Recortes.
    - Transformaciones de color.
    - Añadiendo ruido.
  - En texto:
    - Sustituyendo palabras.
    - Insertando palabras.
    - Borrando palabras.

Propiedades del BackProp

- Convergencia: teorema general del descenso por gradiente (Tema 3)
- Elección del factor de aprendizaje: Tema 3 (Adadelta, Adam, ...)
- Coste computacional: $O(N\ D)$ en cada iteración
- *En condiciones límites, las salidas de un perceptrón entrenado para minimizar el error cuadrático medio de las muestras de entrenamiento de un problema de clasificación aproximan la distribución a-posteriori subyacente en las muestras de entrenamiento.*

Reducción del tamaño de las redes neuronales

- MÉTODOS DE PODA:
  - Poda de pesos basadas en:
    - RELEVANCIA
    - CASTIGO
- Quantización de los parámetros.
- Entrenamiento de redes pequeñas a partir de datos generados por redes grandes (Knowledge distillation)

Index

1 Redes neuronales multicapa ▷ 2

2 Algoritmo de retropropagación del error (BackProp) ▷ 19

3 Aspectos de uso y propiedades del BackProp ▷ 34

○ 4 *Notación* ▷ 49

Notación

- Funciones discriminantes lineales: $\phi(\boldsymbol{x}; \boldsymbol{\Theta}) = \boldsymbol{\Theta}^t \boldsymbol{x}$ para una entrada $\boldsymbol{x}$ y parámetros $\boldsymbol{\Theta}$ compuestos por vector de pesos y umbral $(\boldsymbol{\theta}, \theta_0)$
- Funciones discriminantes lineales con activación: $g \circ \phi(\boldsymbol{x}; \boldsymbol{\theta})$ para una entrada $\boldsymbol{x}$, parámetros $(\boldsymbol{\theta}, \theta_0)$ y g una función de activación. g' es la derivada de la función de activación g
- Función de activación sigmoid: $g_S(z)$
- Salida del nodo i en la capa k: s_i^k en perceptrones multicapa y redes hacia adelante
- Pesos de la conexión que va del nodo j de la capa $k-1$ al nodo i de la capa k en un perceptrón multicapa: θ_{ij}^k. $\boldsymbol{\Theta}$ es un vector de talla D formado por todos los pesos θ_{ij}^k. Pesos de la conexión que va del nodo j de la capa k' al nodo i de la capa k en una red hacia adelante: $\theta_{ij}^{k',k}$
- Conjunto de N muestras de entrenamiento: $S = \{(\boldsymbol{x}_1, \boldsymbol{t}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{t}_N)\}$ con $\boldsymbol{x}_n \in \mathbb{R}^{M_0}$ y $\boldsymbol{t}_n \in \mathbb{R}^{M_K}$, siendo K el número de capas y M_k el número de nodos de la capa k
- Función a minimizar en el entrenamiento de un perceptrón multicapa: $q_S(\boldsymbol{\Theta}) \in \mathbb{R}$
- Clasificador en $C \equiv M_K$ clases de puntos de $\mathbb{R}^d \equiv \mathbb{R}^{M_0}$: $f : \mathbb{R}^{M_0} \rightarrow \{1, \ldots, M_K\}$
- Error en el nodo i de la capa k para la muestra $\boldsymbol{x}_n$: $\delta_i^k(\boldsymbol{x}_n)$
- Incremento del peso que va del nodo j en la capa $k-1$ al nodo i en la capa k: $\Delta\theta_{ij}^k$
- Factor de aprendizaje, momentum y factor de regularización: ρ, ν yλ
- Media y desviación típica: μ y σ