2016-2017

Aprendizaje Automático

6. Redes Neuronales Multicapa

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

Enrique Vidal Ruiz
(evidal@dsic.upv.es)

Departament de Sistemas Informàtics i Computació (DSIC)

Universitat Politècnica de València (UPV)

Septiembre, 2016

Página intencionadamente en blanco

Index

Modelo conexionista

- Un conjunto de procesadores elementales densamente interconectados
- Nombres alternativos:
  - Modelo conexionista
  - Red neuronal artificial
  - Procesado distribuido y paralelo
- Neurocomputador: conjunto de procesadores hardware interconectados operando concurrentemente que implementan un modelo conexionista

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. En 1950 Rochester realiza una simulación en un computador IBM
- 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 perceptones 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 propuests 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
- 2006: *Hinton* publica en *Science* un artículo que inagura una nueva tendencia denominada *“redes profundas”*. Para el aprendizaje de los pesos de estas redes se recurre a una técnica de randomización conocida como “máquina de Boltzmann restringida” y se usa la técnica de retropropagación del error para un ajuste final. Posteriormente, en 2015 *LeCun, Bengio y Hinton* publican un artículo sobre estas técnicas en *Nature*

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

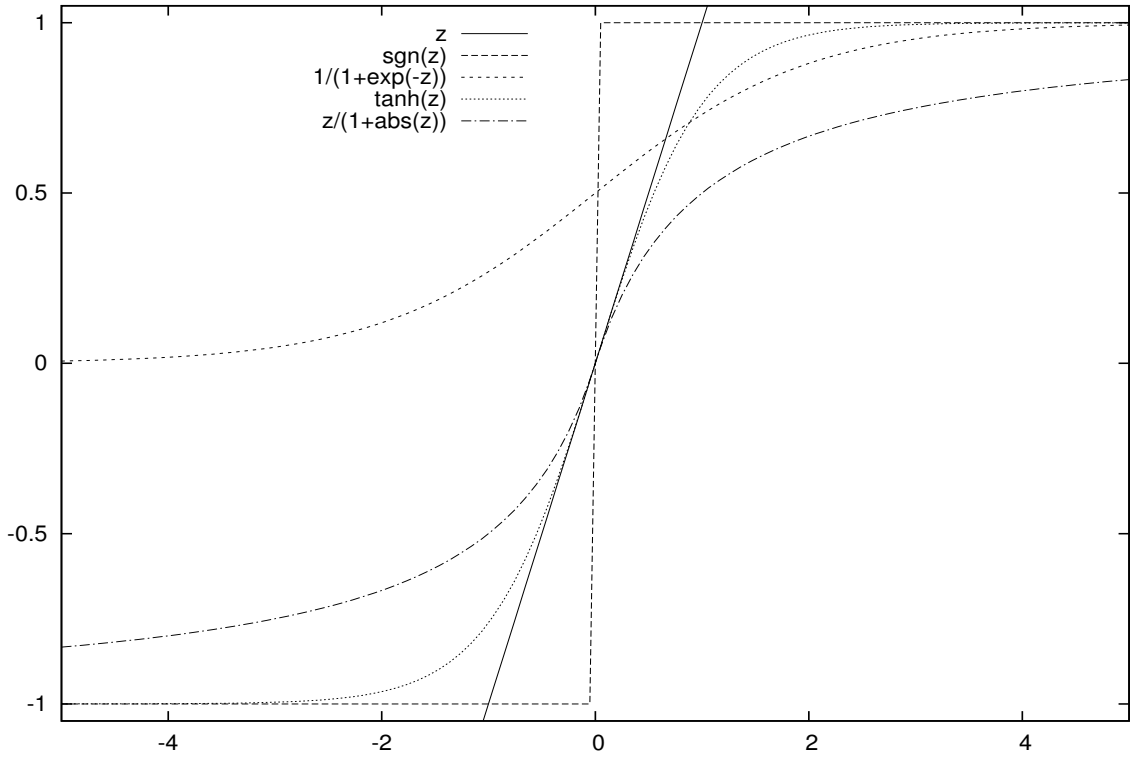
Funciones de activación†

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

1. LINEAL: $g_L(z) = z$.
2. SALTO O ESCALÓN: $g_E(z) = \text{sgn}(z) = \begin{cases} +1 & \text{si } z > 0 \\ -1 & \text{si } z < 0 \end{cases}$
3. RAMPA: $g_R(z) = \begin{cases} z & \text{si } -1 < z < +1 \\ \text{sgn}(z) & \text{si } |z| \geq 1 \end{cases}$
4. SIGMOID: $g_S(z) = \dfrac{1}{1 + \exp(-z)}$
5. TANGENTE HYPERBÓLICA: $g_T(z) = \tanh(z) = \dfrac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$
6. RÁPIDA: $g_F(z) = \dfrac{z}{1 + \mid z \mid}$
7. SOFTMAX: Dados $z_1, \ldots, z_M \in \mathbb{R}, \quad g_M(z_i) = \dfrac{\exp(z_i)}{\sum_{j=1}^{M} \exp(z_j)}$ para $1 \leq i \leq M$

Ejercicio: Demostrar que $g_T(z) = 2 g_S(2z) - 1 \;\; \forall z \in \mathbb{R}$

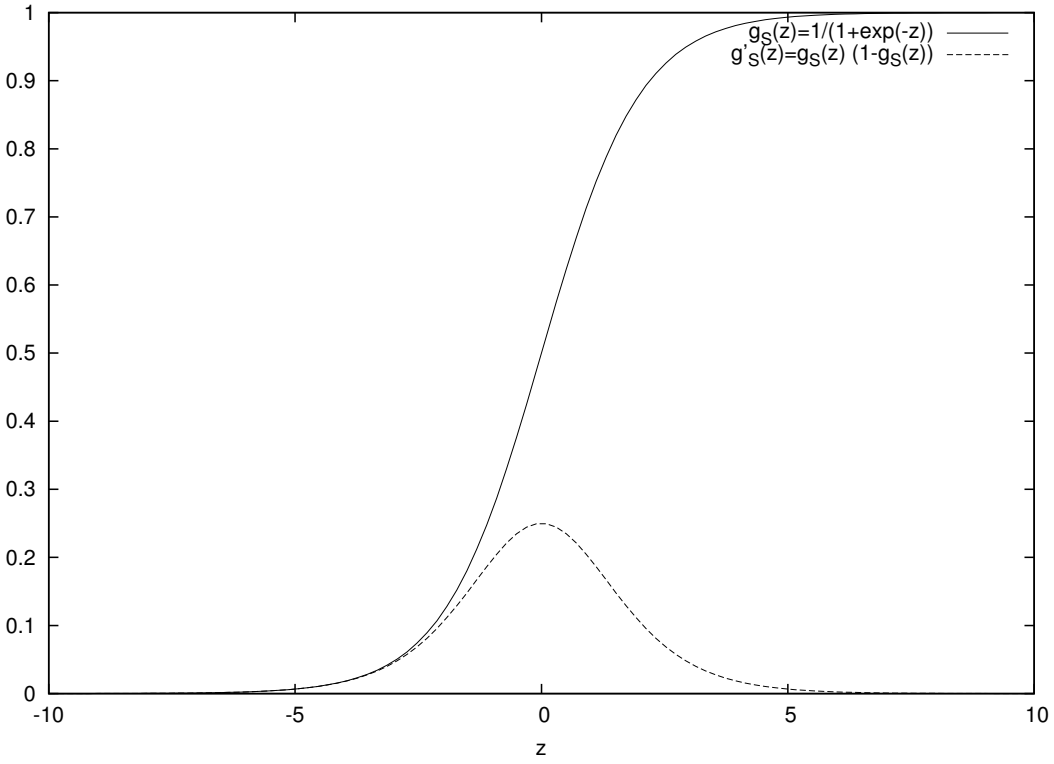
Funciones de activación



Derivadas de las funciones de activación

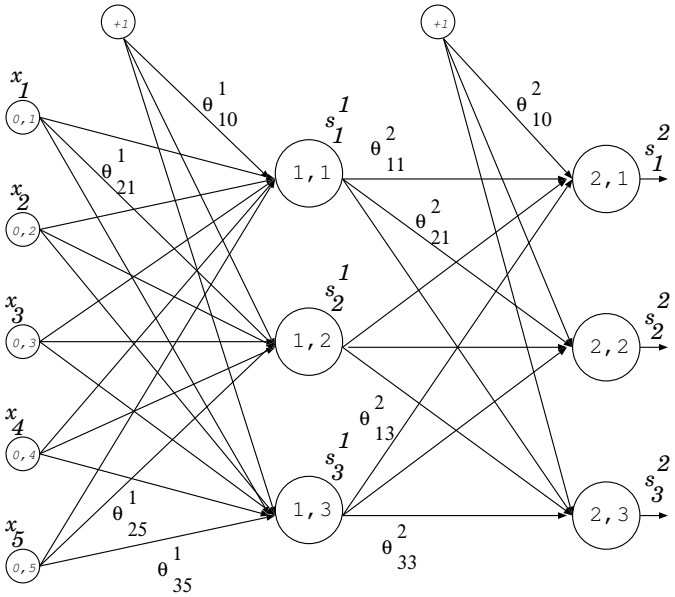
- LINEAL: $g_L(z) = z \;\Rightarrow\; g'_L(z) = \dfrac{d\, g_L}{d\, z} = 1$
- ESCALÓN: $g_E(z) = \begin{cases} 1 & \text{si } z > 0 \\ 0 & \text{si } z < 0 \end{cases} \;\Rightarrow\; g'_E(z) = \begin{cases} 0 & \text{si } z \neq 0 \\ \text{no definida} & \text{si } z = 0 \end{cases}$
- RAMPA: $g_R(z) = \begin{cases} z & -1 < z < +1 \\ \text{sgn}(z) & |z| \geq 1 \end{cases} \Rightarrow g'_R(z) = \begin{cases} 1 & -1 \leq z \leq +1 \\ 0 & |z| \geq 1 \\ \text{no def.} & |z| = 1 \end{cases}$
- SIGMOID: $g_S(z) = \dfrac{1}{1+\exp(-z)} \;\Rightarrow\; g'_S(z) = \dfrac{d\, g_S}{d\, z} = g_S(z)\,(1 - g_S(z))$
- TANGENTE HIPERBÓLICA: $g_T(z) = \dfrac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} \Rightarrow g'_T(z) = \dfrac{d\, g_T}{d\, z} = 1 - (g_T(z))^2$
- RÁPIDA: $g_F(z) = \dfrac{z}{1+\mid z \mid} \;\Rightarrow\; g'_F(z) = \dfrac{d\, g_F}{d\, z} = \dfrac{1}{(1+\mid z \mid)^2}$
- SOFTMAX: Para $z_1, \ldots, z_n \in \mathbb{R}$, $g_M(z_i) = \dfrac{\exp(z_i)}{\sum_j \exp(z_j)} \;\Rightarrow\; g'_M(z_i) = g_M(z_i)\,(1 - g_M(z_i))$

Derivada de la función de activación sigmoid



Un perceptrón de dos capas: ejemplo y notación

Topología



Dinámica

Capa de entrada

$1 \leq i \leq M_0 \equiv d = 5$

$x_i \in \mathbb{R}$

Capa oculta

$1 \leq i \leq M_1 = 3$

$$s_i^1(\boldsymbol{x};\boldsymbol{\Theta}) = g(\sum_{j=0}^{M_0} \theta_{ij}^1 \; x_j)$$

Capa de salida

$1 \leq i \leq M_2 = 3$

$$s_i^2(\boldsymbol{x};\boldsymbol{\Theta}) = g(\sum_{j=0}^{M_1} \theta_{ij}^2 \; s_j^1(\boldsymbol{x}))$$

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 $\boldsymbol{s}^2(\boldsymbol{x}_n;\boldsymbol{\Theta}) = \boldsymbol{t}_n\ \forall n, 1 \leq n \leq N$.

- *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 $\mathbb{R}^{M_0} \to \mathbb{R}^{M_2}$:

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

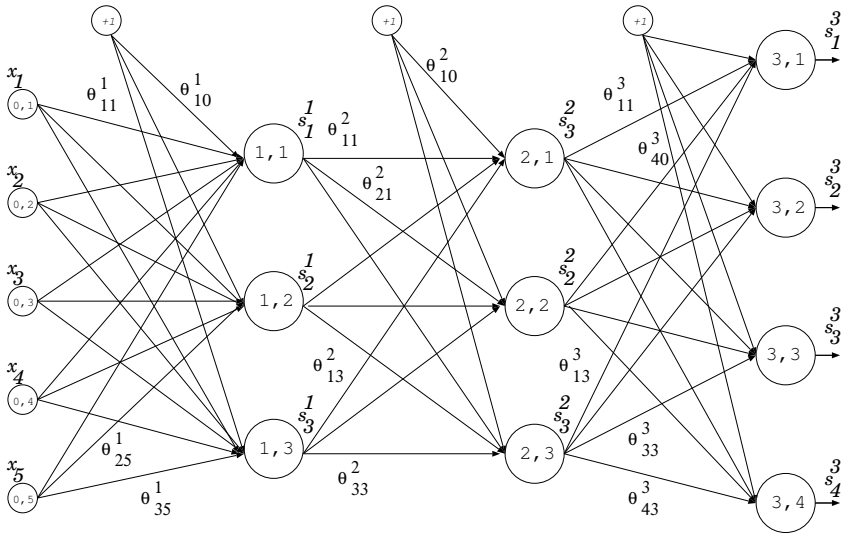
- Si todas las funciones de activación son lineales, un perceptrón multicapa define UNA FUNCIÓN DISCRIMINANTE LINEAL, $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \ldots, \phi_{M_2}(\boldsymbol{x}))^t$:

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

- 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:

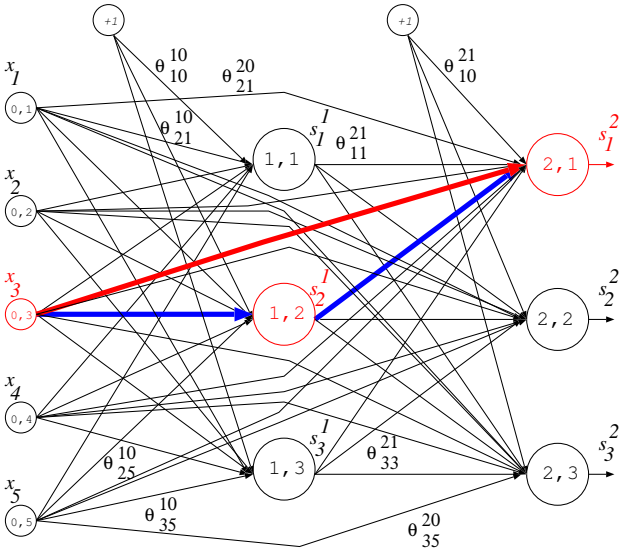
$$\phi_i(\boldsymbol{x}) \equiv s_i^2(\boldsymbol{x}) = \sum_{j=0}^{M_1} \theta_{ij}^2\ g(\sum_{j'=0}^{M_0} \theta_{jj'}^1\ x_{j'}) = \sum_{j=0}^{M_1} \theta_{ij}^2\ \psi_j(\boldsymbol{x}) \text{ para } 1 \leq i \leq M_2$$

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 $d' \equiv M_2$)

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

- 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*

El perceptrón multicapa como clasificador

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

$$f : \mathbb{R}^d \rightarrow \{1, \ldots, C\} : f(\boldsymbol{x}) = \arg\max_{1 \leq c \leq M_2} s_c^2(\boldsymbol{x})$$

- 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]

Index

Aprendizaje de los pesos de un perceptrón multicapa

PROBLEMA: Dada la topología de un perceptrón multicapa y 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 $\boldsymbol{\Theta} \in \mathbb{R}^D : \boldsymbol{s}^2(\boldsymbol{x}_n; \boldsymbol{\Theta}) = \boldsymbol{t}_n$ para $1 \leq n \leq N$ (o $\boldsymbol{s}^2(\boldsymbol{x}_n; \boldsymbol{\Theta}) \approx \boldsymbol{t}_n$).

- Resolver el sistema de ecuaciones no lineales $\boldsymbol{s}^2(\boldsymbol{x}_n) = \boldsymbol{t}_n, 1 \leq n \leq N$: raramente tiene solución.
- Problema reformulado: Dada la topología de un perceptrón multicapa y 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 $\boldsymbol{\Theta}$ que minimice el error cuadrático medio:

$$q_S(\boldsymbol{\Theta}) = \frac{1}{2N} \sum_{n=1}^{N} \sum_{i=1}^{M_2} \left(t_{ni} - s_i^2(\boldsymbol{x}_n)\right)^2$$

- Solución: DESCENSO POR GRADIENTE:

$$\Delta\theta_{ij}^l = -\rho \, \frac{\partial q_S(\boldsymbol{\Theta})}{\partial \theta_{ij}^l}$$

$$1 \leq l \leq 2, \;\; 1 \leq i \leq M_l, \;\; 0 \leq j \leq M_{l-1}$$

Algoritmo de retropropagación del error (BackProp)

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

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

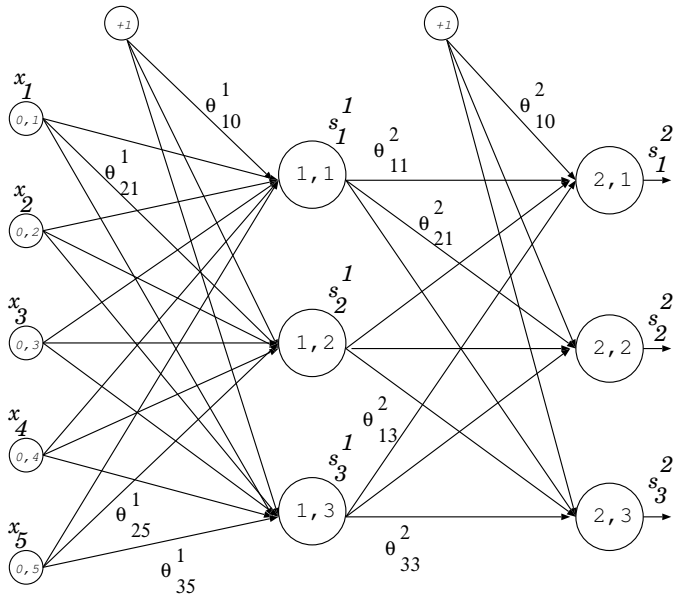
$$\delta^2_i(\boldsymbol{x}_n) = \left(t_{ni} - s^2_i(\boldsymbol{x}_n)\right)\ g'(\phi^2_i(\boldsymbol{x}_n)) \ \text{ con } \ \phi^2_i(\boldsymbol{x}_n) = \sum_{j=0}^{M_1} \theta^2_{ij} s^1_j(\boldsymbol{x}_n)$$

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

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

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

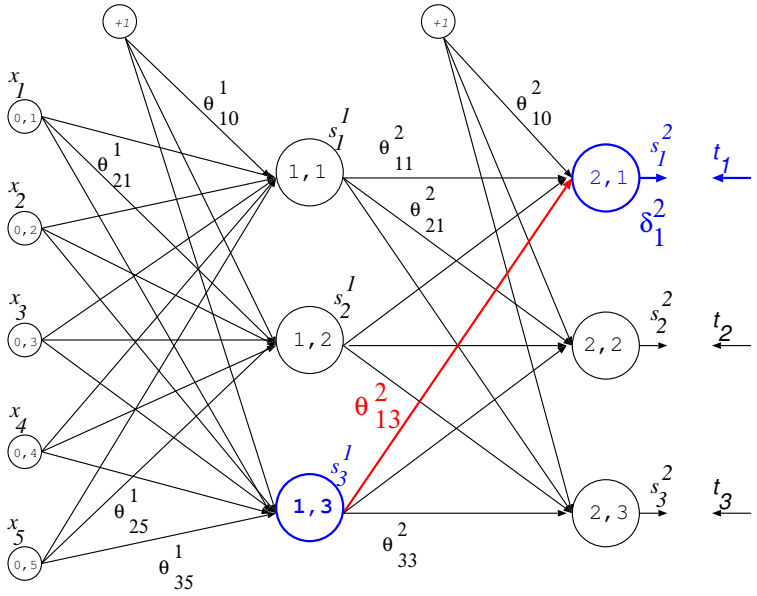
Algoritmo BackProp: cálculo hacia adelante



$$s^1_i(\boldsymbol{x}) = g(\phi^1_i) = g(\sum_{j=0}^{M_0} \theta^1_{ij}\, x_j)\ \ 1 \leq i \leq M_1; \quad s^2_j = g(\phi^2_j) = g(\sum_{k=0}^{M_1} \theta^2_{jk}\ s^1_k(\boldsymbol{x})),\ 1 \leq j \leq M_2$$

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

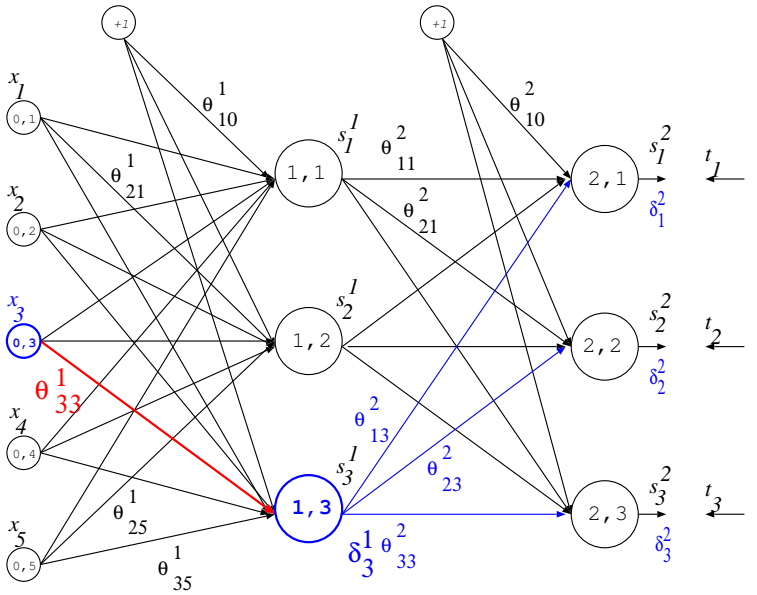
BackProp: Actualización pesos de la capa de salida



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

(para solo una muestra de entrenamiento)

BackProp: Actualización pesos de la capa oculta



$$\Delta\theta^1_{3\,3} = \rho\ \delta^1_3\ x_3 = \rho\ (\textstyle\sum_{r=1}^{M_2} \delta^2_r \theta^2_{r3}) g'(\phi^1_3)\ x_3$$

(para solo una muestra de entrenamiento)

Regla de la cadena en el cálculo de derivadas

- Función simple de otra función de una variable: $f, g : \mathbb{R} \to \mathbb{R}$:

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

- Función de dos funciones de M variables: $g_1, g_2 : \mathbb{R}^M \to \mathbb{R}$, $f : \mathbb{R}^2 \to \mathbb{R}$

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

- Función de N funciones de M variables: $g_1, \dots, g_N : \mathbb{R}^M \to \mathbb{R}$, $f : \mathbb{R}^N \to \mathbb{R}$:

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

Derivación del algoritmo BackProp (I)

Actualización de los pesos de la capa de salida θ^2_{kl} (una sola muestra, $N = 1$)

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

$$\frac{\partial q_S(\mathbf{\Theta})}{\partial \theta^2_{ij}} = \frac{\partial q_S(\mathbf{\Theta})}{\partial \phi^2_i}\frac{\partial \phi^2_i}{\partial \theta^2_{ij}} = \frac{\partial q_S(\mathbf{\Theta})}{\partial s^2_i}\frac{\partial s^2_i}{\partial \phi^2_i}\frac{\partial \phi^2_i}{\partial \theta^2_{ij}} = \left[(-1)\ (t_i - s^2_i)\right]\ g'(\phi^2_i)\ s^1_j$$

$$\delta^2_i \stackrel{\text{def}}{=} (t_i - s^2_i)\ g'(\phi^2_i)\ \left(= -\frac{\partial q_S(\mathbf{\Theta})}{\partial \phi^2_i}\right) \implies \frac{\partial q_S(\mathbf{\Theta})}{\partial \theta^2_{ij}} = -\delta^2_i\ s^1_j$$

$$\Delta\theta^2_{ij} = -\rho\ \frac{\partial q_S(\mathbf{\Theta})}{\partial \theta^2_{ij}} = \rho\ \delta^2_i\ s^1_j \quad 1 \le i \le M_2, 0 \le j \le M_1$$

Derivación del algoritmo BackProp (II)

Actualización de los pesos de la capa de la capa oculta θ^1_{ij} (para $N = 1$)

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

$$\frac{\partial q_S(\boldsymbol{\Theta})}{\partial \theta^1_{ij}} = \frac{\partial q_S(\boldsymbol{\Theta})}{\partial \phi^1_i}\,\frac{\partial \phi^1_i}{\partial \theta^1_{ij}} = \frac{\partial q_S(\boldsymbol{\Theta})}{\partial s^1_i}\,\frac{\partial s^1_i}{\partial \phi^1_i}\,\frac{\partial \phi^1_i}{\partial \theta^1_{ij}}$$

$$= \left(\sum_{l=0}^{M_2}\frac{\partial q_S(\boldsymbol{\Theta})}{\partial \phi^2_l}\,\frac{\partial \phi^2_l}{\partial s^1_i}\right) g'(\phi^1_i)\, x_j = \left(\sum_{l=0}^{M_2} -\,\delta^2_l\,\theta^2_{li}\right) g'(\phi^1_i)\, x_j$$

$$\delta^1_i \overset{\text{def}}{=} \left(\sum_{l=1}^{M_2}\delta^2_l\,\theta^2_{li}\right) g'(\phi^1_i) \quad\Longrightarrow\quad \frac{\partial q_S(\boldsymbol{\Theta})}{\partial \theta^1_{ij}} = -\delta^1_i\, x_j$$

$$\Delta\theta^1_{ij} = -\rho\,\frac{\partial q_S(\boldsymbol{\Theta})}{\partial \theta^1_{ij}} = \rho\,\delta^1_i\, x_j \qquad 1 \le i \le M_1, 0 \le j \le M_0$$

BackProp para perceptrones de tres capas

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

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

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

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

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

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

Algoritmo BACKPROP

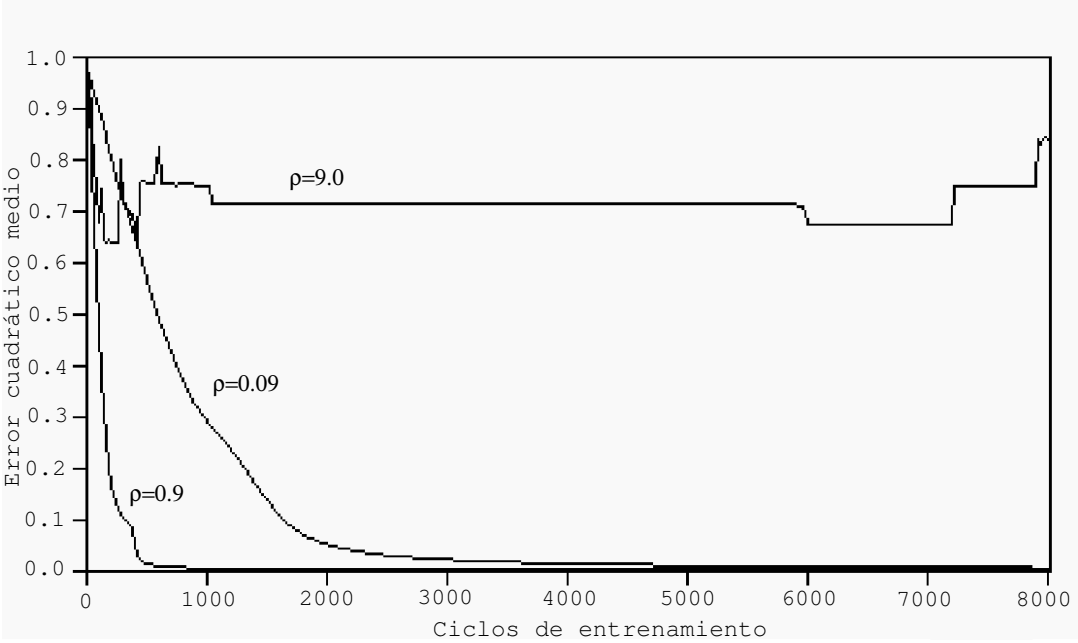
Entrada: Una topología, datos de entrenamiento S, un factor de aprendizaje ρ, pesos iniciales θ_{ij}^l $1 \leq l \leq L, 1 \leq i \leq M_l, 0 \leq j \leq M_{l-1}$ y condiciones de convergencia
Salidas: Pesos de las conexiones que minimizan el error cuadrático medio de S
Método:

Mientras no converja
  Para $1 \leq l \leq L, 1 \leq i \leq M_l, 0 \leq j \leq M_{l-1}$, inicializar $\Delta\theta_{ij}^l = 0$
  Para toda muestra de aprendizaje $(\boldsymbol{x}_n, \boldsymbol{t}_n) \in S$
    Desde la capa de entrada a la capa de salida ($l = 1, \ldots, L$):
      Calcular $\phi_i^l(\boldsymbol{x}_n)$ y $s_i^l(\boldsymbol{x}_n) = g(\phi_i^l(\boldsymbol{x}_n))$ para $1 \leq i \leq M_l$
    Desde la salida a la entrada ($l = L, \ldots, 1$) ,
      Para cada nodo ($1 \leq i \leq M_l$)
        Calcular $\delta_i^l(\boldsymbol{x}_n) = \begin{cases} g'(\phi_i^l(\boldsymbol{x}_n)) \; (t_{ni} - s_i^L(\boldsymbol{x}_n)) & \text{si } l == L \\ g'(\phi_i^l(\boldsymbol{x}_n)) \; (\sum_r \delta_r^{l+1}(\boldsymbol{x}_n) \; \theta_{ri}^{l+1}) & \text{en otro caso} \end{cases}$
        Para cada peso θ_{ij}^l $(0 \leq j \leq M_{l-1})$ calcular: $\Delta\theta_{ij}^l$ += $\delta_i^l(\boldsymbol{x}_n) \; s_j^{l-1}(\boldsymbol{x}_n)$
      Fin para
  Fin para
  Actualizar pesos: $\theta_{ij}^l = \theta_{ij}^l + \rho \; \Delta\theta_{ij}^l$
Fin mientras

El coste computacional del algoritmo BackProp es $O(N \; D)$ en cada iteración.

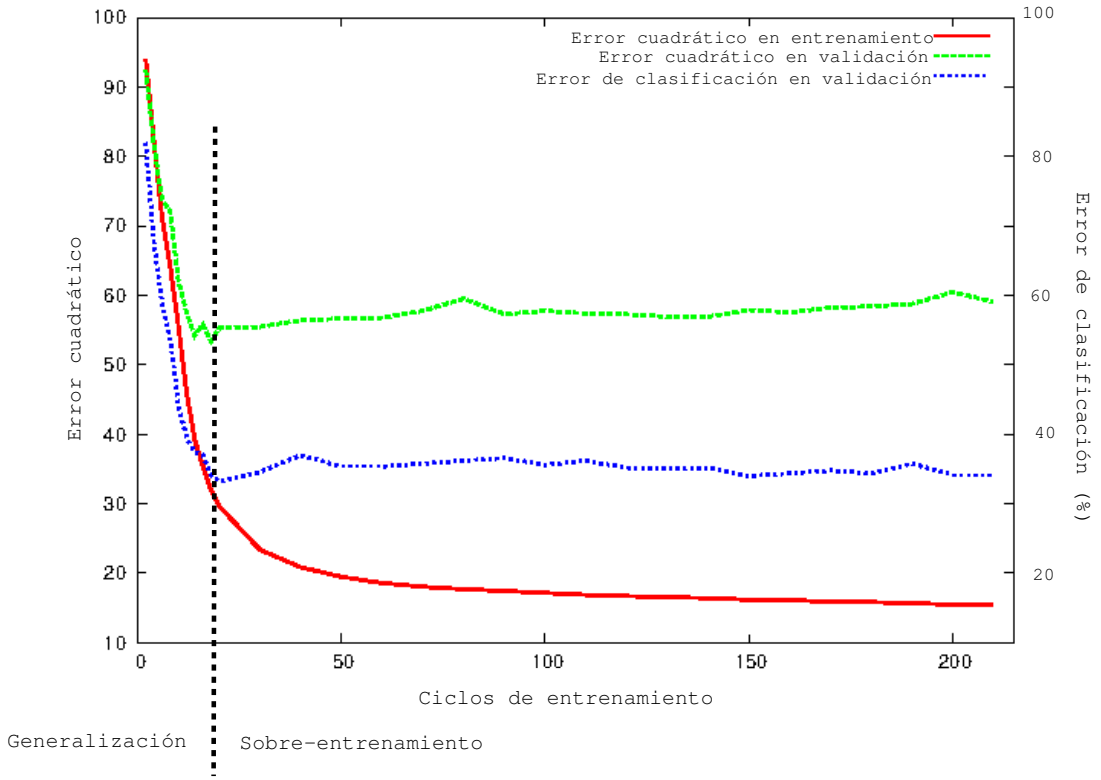
DEMO

La selección del factor de aprendizaje



DEMO

Condiciones de convergencia



Codificación

- Los valores ± 1 (o $0, 1$) solo se alcanzan asintóticamente cuando se utilizan la mayoría de las funciones de activación como la sigmoid:

$$\text{Valores deseados de salida: } t_i = \begin{cases} -0.9 \\ +0.9 \end{cases}$$

- 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.

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

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

Propiedades del BackProp

- Convergencia: teorema general del descenso por gradiente (Tema 3)
- Elección del factor de aprendizaje: Tema 3
- Coste computacional: $O(N\ D)$ en cada iteración
- Perceptrones de una o dos capas ocultas [Villiars and Basnard, 92]
  - En el mejor de los casos, los resultados de clasificación no presentan diferencias estadísticas significativas.
  - Generalmente, un perceptrón de una capa oculta suele producir mejores resultados de clasificación que los de dos capas ocultas.
  - Los perceptrones de dos capas ocultas suelen converger más rápidamente que los perceptrones de una capa oculta.
- *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.*

  En ocasiones, se suele utilizar las salidas del perceptrón como una aproximación a esa distribución a-posteriori (p.e. para un perceptrón de una capa oculta): $s_i^2(\boldsymbol{x};\boldsymbol{\Theta}) \approx P(i \mid \boldsymbol{x})$ para lo que se suele utilizar la función "softmax" como función de activación en la capa de salida, lo que permite su normalización.

Estimación de la topología

- MÉTODOS DE PODA:
  - Poda de pesos:
    - basada en RELEVANCIA
    - basada en en CASTIGO
  - Poda de FDLA.
- MÉTODOS INCREMENTALES:
  - Búsqueda incremental (Moddy & Utans, 1995)
  - "Cascade-correlation" (Fahlman & Lebiers, 1990)
  - Adaptación estructural (Lee et al. 1990)
- POR TRANSFORMACIÓN
  - Árboles de decisión

Aspectos computacionales del aprendizaje con el PM

- Sobre la intratabilidad del aprendizaje de redes neuronales (Blum and Rivest, 1992): La búsqueda de un conjunto de pesos (racionales) tal que el perceptrón multicapa (con funciones de activación en escalón) genere la salida deseada (en el caso de que sea posible) para cada muestra de entrenamiento es **NP**-duro.

- Sobre el tamaño del conjunto de entrenamiento (Ripley, 1993 y Tema 2.1): Si el perceptrón multicapa tiene M nodos y un número arbitrario de capas, el número de muestras de entrenamiento debe ser proporcional a $\frac{D}{\epsilon}\log\frac{M}{\epsilon}$ donde ϵ es el error tolerado. Si el perceptrón multicapa tiene solo una capa, el número de muestras de entrenamiento debe ser proporcional a $\frac{D}{\epsilon}$.

Index

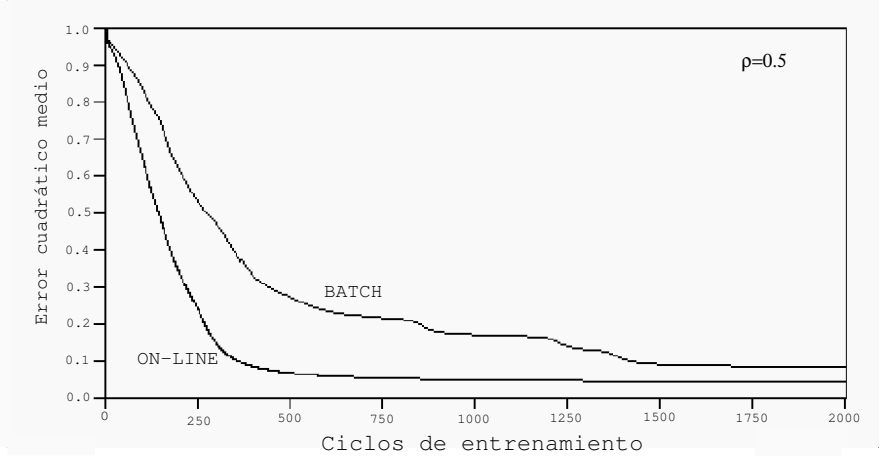
BackProp incremental u *"on-line"*

- BackProp "batch" (en cada iteración k se procesan todas las $\boldsymbol{x}_n, 1 \leq n \leq N$):

$$\Delta\theta_{ij}^l(k) \;=\; \rho_k \sum_{n=1}^{N} \delta_i^l(\boldsymbol{x}_n)\; s_j^{l-1}(\boldsymbol{x}_n) \quad 1 \leq l \leq L, 1 \leq i \leq M_l, 1 \leq j \leq M_{l-1}$$

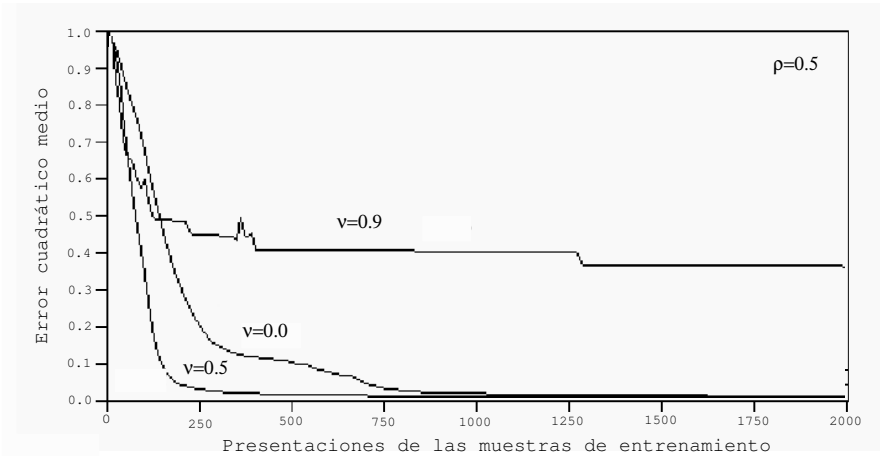
- BackProp "*on-line*" (en la iteración k-ésima se procesa solo $\boldsymbol{x}_{k'}$, $k' = 1 + k \bmod N$):

$$\Delta\theta_{ij}^l(k) \;=\; \rho_k\; \delta_i^l(\boldsymbol{x}_{k'})\; s_j^{l-1}(\boldsymbol{x}_{k'}) \qquad 1 \leq l \leq L, 1 \leq i \leq M_l, 1 \leq j \leq M_{l-1}$$



BackProp con momentum

- Problema: convergencia lenta en "plateaux"
- Posible solución: añadir una "inercia" o "momentum" con un peso $0 \leq \nu < 1$
- BP con momentum (batch): $\Delta\theta_{ij}^l(k) = \rho_k \sum_{n=1}^{N} \delta_i^l(\boldsymbol{x}_n)\; s_j^{l-1}(\boldsymbol{x}_n) + \nu\; \Delta\theta_{ij}^l(k-1)$
- TEOREMA (Phansalkar and Sartry, 1994): *Los puntos estables del algoritmo BackProp con momentum* $(\boldsymbol{\Theta}(k) = \boldsymbol{\Theta}(k+1))$ *son mínimos locales de* $q_S(\boldsymbol{\Theta})$



BackProp con amortiguamiento

- Problema: evitar que los pesos sean muy grandes y provoquen una parálisis de la red.
- Solución: minimizar el error cuadrático medio con regularización:

$$q_S(\mathbf{\Theta}) + \lambda \sum_{l,i,j} (\theta_{ij}^l)^2$$

- Algoritmo "batch', para $1 \leq l \leq L, 1 \leq i \leq M_l, 1 \leq j \leq M_{l-1}$

$$\Delta\theta_{ij}^l(k) = \rho_k \sum_{n=1}^{N} \delta_i^l(\boldsymbol{x}_n)\ s_j^{l-1}(\boldsymbol{x}_n) + 2\ \rho_k\ \lambda\ \theta_{ij}^l(k)$$

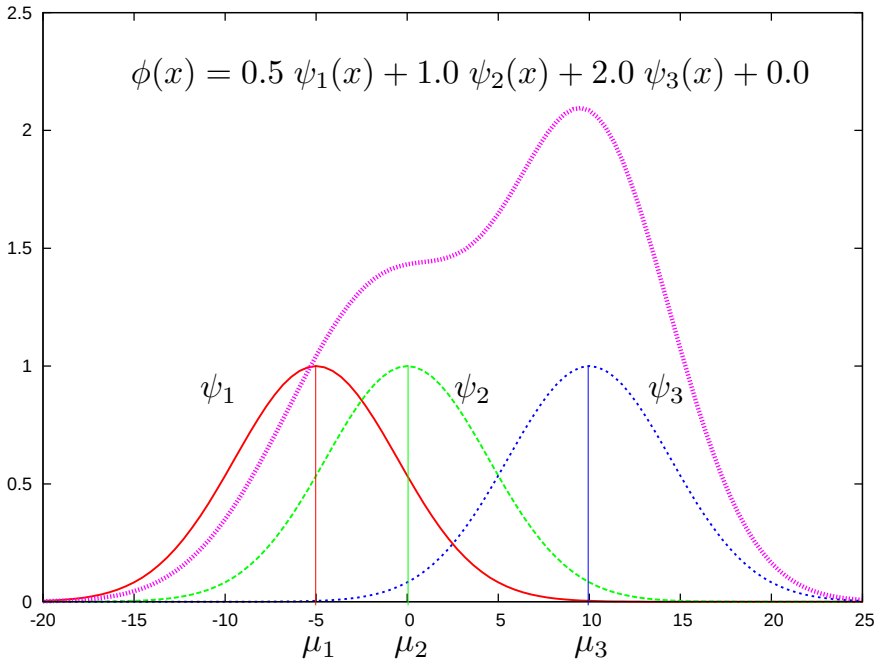
Redes neuronales radiales

- *Una red neuronal radial* (RNR) es una FDLG de la forma:

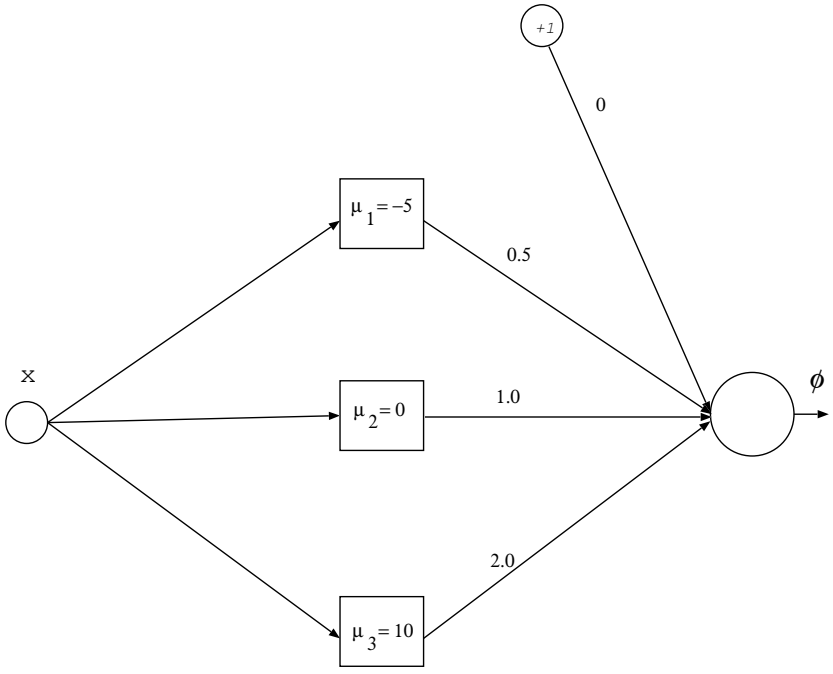
$$\phi_m(\boldsymbol{x}; \mathbf{\Theta}) \ = \ \sum_{i=1}^{d'} \theta_{mi}\, \psi(\| \boldsymbol{x} - \boldsymbol{\mu}_i \|) \ + \ \theta_{m0} \qquad 1 \leq m \leq M$$

  donde: $\boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{\mu}_i \in \mathbb{R}^d$, para $1 \leq i \leq d'$ $\boldsymbol{\theta}_m \in \mathbb{R}^{d'}, \theta_{m0} \in \mathbb{R}$ y ψ es una función básica radial típicamente de la forma: $\psi(z) = \exp\left(-\frac{z^2}{2\sigma^2}\right)$ con $\sigma \in \mathbb{R}$.
- Para clasificación, $M \equiv C$, aunque las RNR son populares para regresión.
- Aprendizaje de RNR:
  - Aprendizaje secuencial, primero de las funciones básicas radiales (clustering) y a continuación los pesos (Adaline, . . .).
  - Aprendizaje integrado de las funciones básicas radiales y los pesos mediante la minimización del error cuadrático medio (similar al BackProp).

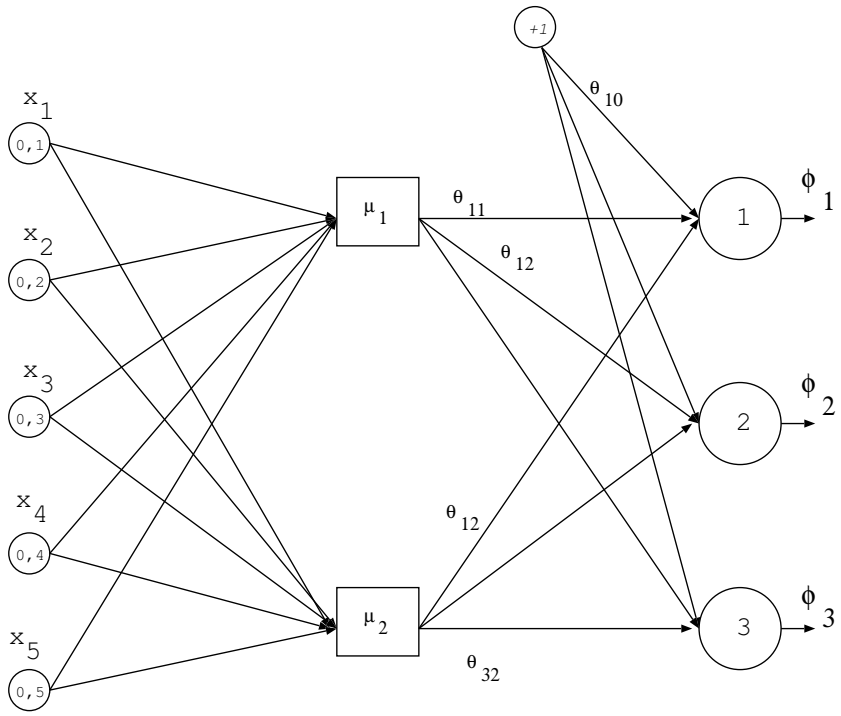
Ejemplo de funciones básicas radiales en $\mathbb{R}$



Ejemplo de funciones básicas radiales en $\mathbb{R}$



Redes neuronales radiales



Index

Ejemplos

Clasificación de dígitos I

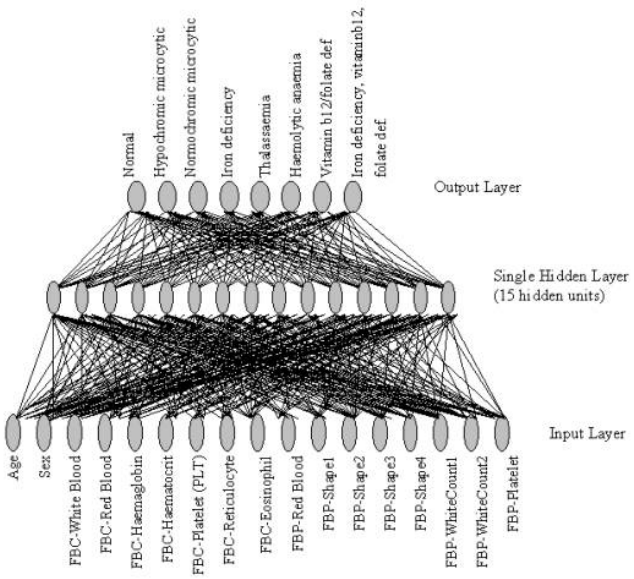
Clasificación de dígitos II

Predicción

Compresión de imágenes

http://neuron.eng.wayne.edu/bpImageCompression9PLUS/bp9PLUS.html

Ejemplo: Clasificación de anemias



http://www.generation5.org/content/2004/NNinAnaemia.asp

Algunas aplicaciones

- CLASIFICACIÓN
  - Reconocimiento de caracteres impresos y manuscritos
  - Exploración petrolífera.
  - Aplicaciones médicas: *detección de ataques de epilesia, ayuda al diagnóstico de la esclerosis múltiple*, etc.
  - Minería de datos
- PREDICCIÓN
  - Previsión de ocupación en los vuelos (Inc. BehavHeuristics.)
  - Evolución de los precios de solares (Ayuntamiento de Boston)
  - Predicción de consumo de bebidas refrescantes (Britvic)
  - Predicción metereológica (National Weather Service)
  - Predicción de stocks (Carl & Associates, Neural Applications Corporation, etc.)
  - Predicción de demanda electrica (Bayernwerk AG, Britvic, etc.)
  - Predicción de fallos en motores eléctricos (Siemens)
- CONTROL AND AUTOMATIZATION
  - Refinado del petróleo (Texaco).
  - Producción de acero (Fujitsu, Neural Applications Corporation, Nippon Steel)

Algunas aplicaciones

- Aplicaciones varias
  http://www.calsci.com/Applications.html
- Diagnóstico médico
  http://www.generation5.org/content/2004/MedicalDiagnosis.asp
- Anemias
  http://www.generation5.org/content/2004/NNinAnaemia.asp
- Aplicaciones en física de altas energías
  http://neuralnets.web.cern.ch/NeuralNets/nnwInHep.html
- Demos en java
  http://lcn.epfl.ch/tutorial/english/index.html
  http://www.aispace.org/downloads.shtml

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 $\mathbb{R}^d \equiv \mathbb{R}^{M_0}$: $f : \mathbb{R}^{M_0} \to \{1, \ldots, M_K\}$
- 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 y λ
- Media y desviación típica: μ y σ