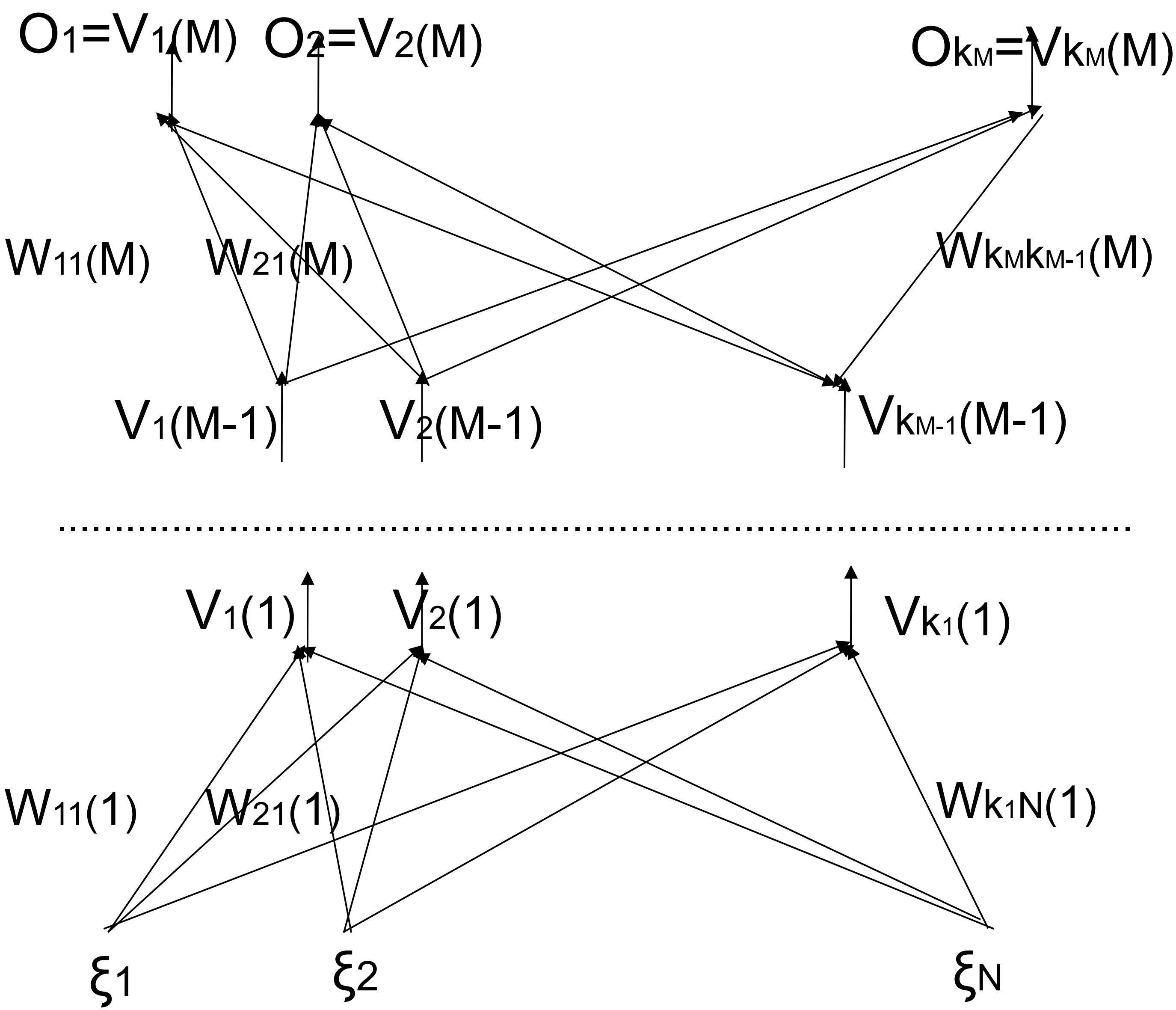
PERCEPTRON MULTICAPA



Si M = 1 (perceptrón simple)

- *Fácil de entrenar*
- *Soluciona familia de problemas restringida*

Si M ≥ 2 (perceptrón multicapa)

- *Mucho más difícil de entrenar (durante mucho tiempo no se supo cómo)*
- *Puede representar cualquier función booleana y, más aun, continua en general*

Teorema (Funahashi)

Sean $\Phi(x)$ continua, no constante, acotada y monótona creciente

K subconjunto compacto de Rn

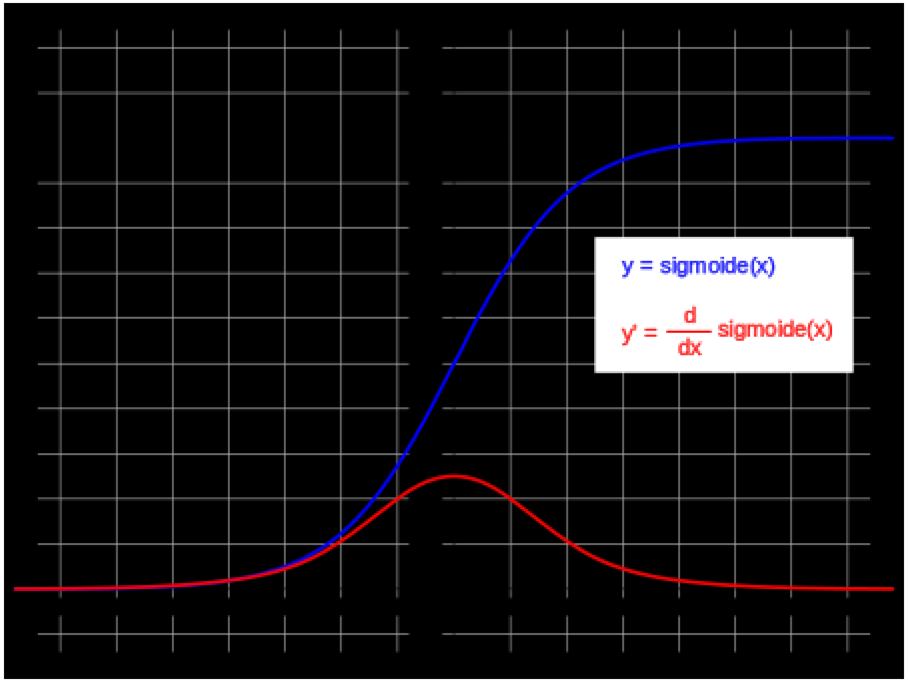
$f : K \to$ R continua

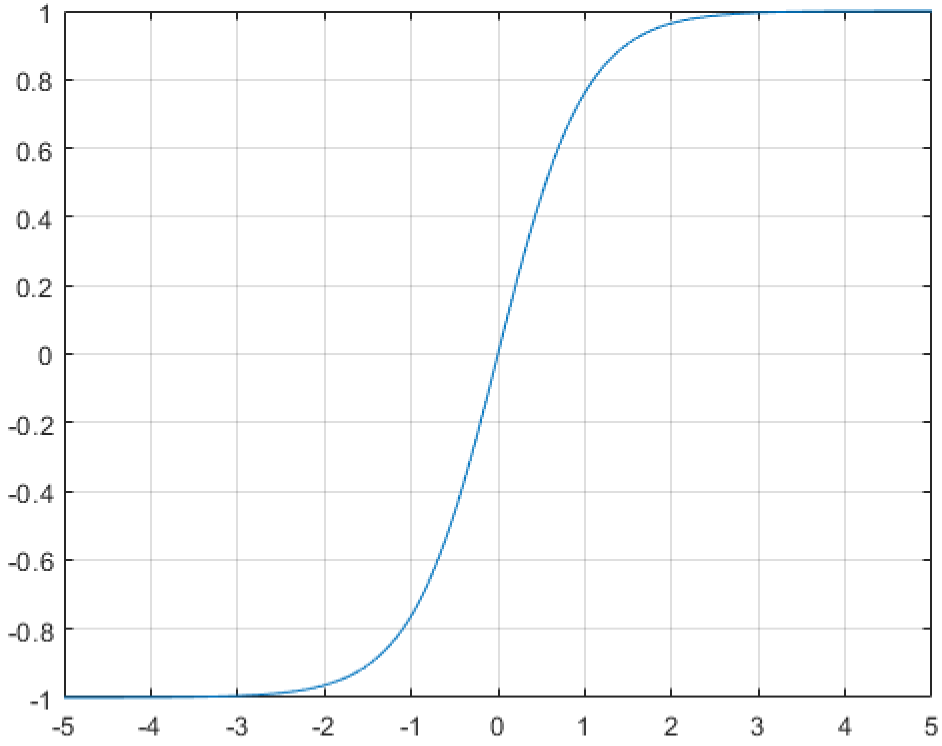
Dado $\varepsilon > 0$ arbitrario, existe N entero y constantes reales C_i, θ_i (i = 1, . . ., N), w_{ij} (i = 1, . . ., N; j = 1, . . ., n) tales que

$$\tilde{f}(x_1, \ldots, x_n) = \sum_{i=1}^{N} c_i \phi\left(\sum_{j=1}^{n} w_{ij} x_j - \theta_i\right)$$

satisface

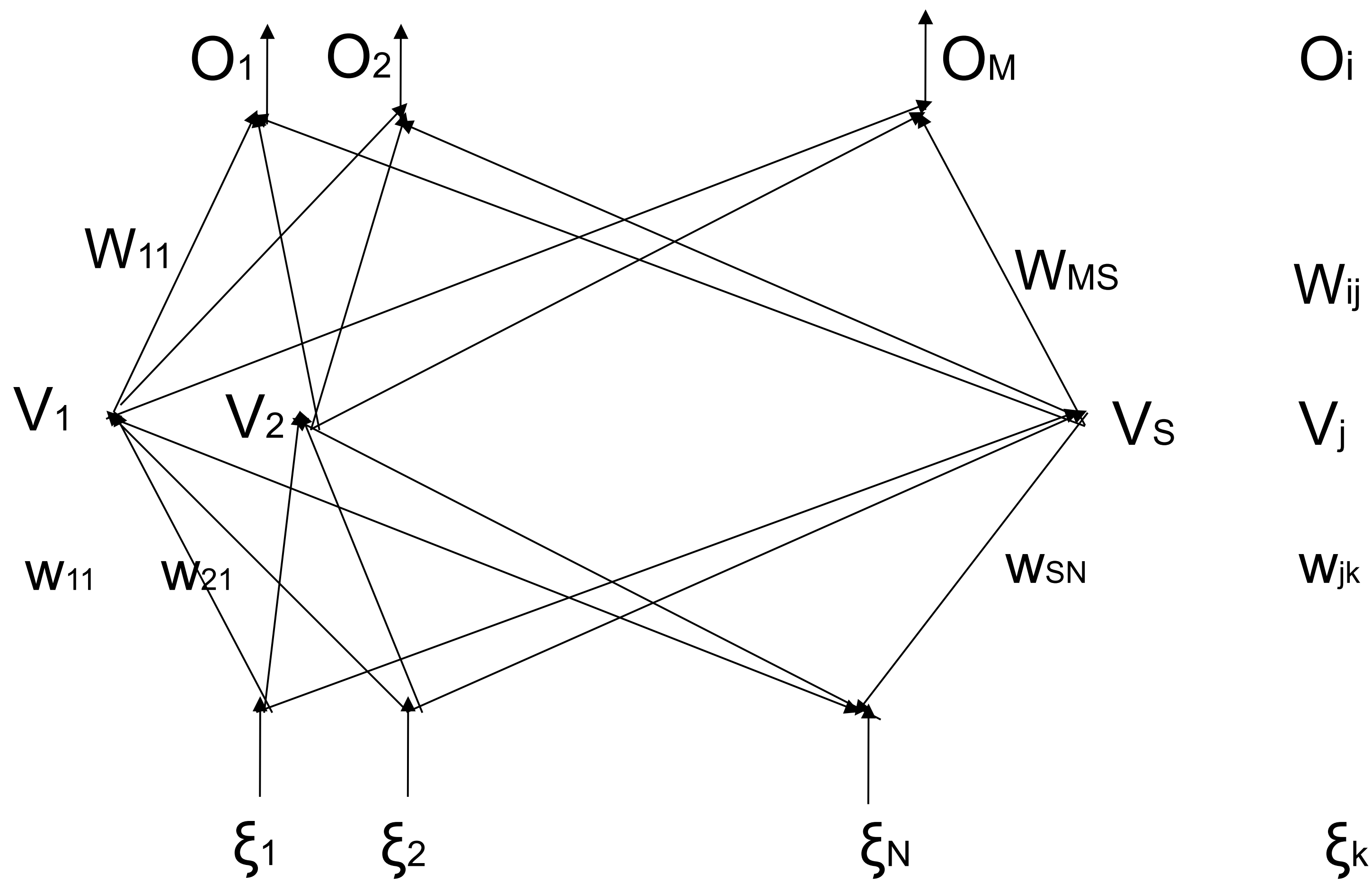
$$\max_{\mathbf{x} \in K} \left| f(x_1, \ldots, x_n) - \tilde{f}(x_1, \ldots, x_n) \right| < \epsilon$$


y = sigmoide(x)
y' = d/dx sigmoide(x)


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-5
-4
-3
-2
-1
0
1
2
3
4
5

NOTACION Y DEFINICIONES

Tomemos M = 2 $\longrightarrow$ Perceptrón bicapa (una capa oculta)



Patrones: pares $(\xi(\mu),\zeta(\mu))$ $\mu=1,\ldots,p$ $\xi(\mu)=(\xi_1(\mu),\xi_2(\mu),\ldots,\xi_N(\mu))$

$\zeta(\mu)=(\zeta_1(\mu),\zeta_2(\mu),\ldots,\zeta_M(\mu))$

$$h_j^\mu = \sum_k w_{jk}\xi_k^\mu \qquad\qquad V_j^\mu = g(h_j^\mu) = g\Big(\sum_k w_{jk}\xi_k^\mu\Big)$$

$$h_i^\mu = \sum_j W_{ij}V_j^\mu = \sum_j W_{ij}\,g\Big(\sum_k w_{jk}\xi_k^\mu\Big)$$

Para el patrón μ como entrada, la salida será:

$$O_i^\mu = g(h_i^\mu) = g\Big(\sum_j W_{ij}V_j^\mu\Big) = g\Big(\sum_j W_{ij}\,g\Big(\sum_k w_{jk}\xi_k^\mu\Big)\Big)$$

Función de costo o error:

$$E[\mathbf{w}] = \frac{1}{2}\sum_{\mu i}[\zeta_i^\mu - O_i^\mu]^2 \qquad E[\mathbf{w}] = \frac{1}{2}\sum_{\mu i}\Big[\zeta_i^\mu - g\Big(\sum_j W_{ij}\,g\Big(\sum_k w_{jk}\xi_k^\mu\Big)\Big)\Big]^2$$

Cuya continuidad y diferenciabilidad dependerán de g.

Pediremos g al menos derivable (en todo punto).

EL ALGORITMO

g derivable → E(w) derivable → puede aplicarse descenso por gradiente

→ $\Delta w = - \eta$ grad E

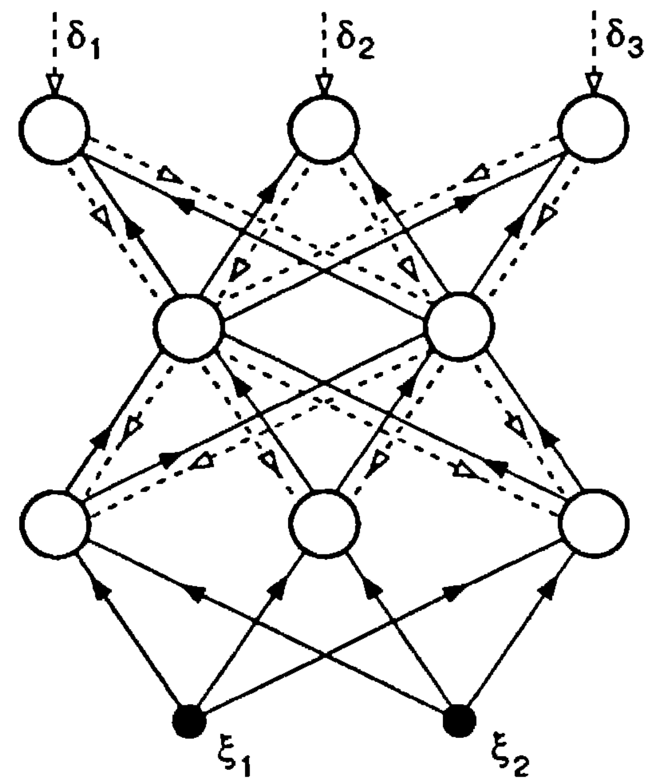
Conexiones capa oculta – capa de salida (1):

$$\Delta W_{ij} = -\eta \frac{\partial E}{\partial W_{ij}} = \eta \sum_{\mu} [\zeta_i^{\mu} - O_i^{\mu}] g'(h_i^{\mu}) V_j^{\mu}$$

$$= \eta \sum_{\mu} \delta_i^{\mu} V_j^{\mu}$$

donde

$$\delta_i^{\mu} = g'(h_i^{\mu})[\zeta_i^{\mu} - O_i^{\mu}]$$



Conexiones entrada - capa oculta (2):

$$\begin{aligned}\Delta w_{jk} &= -\eta\frac{\partial E}{\partial w_{jk}} = -\eta\sum_{\mu}\frac{\partial E}{\partial V_j^\mu}\frac{\partial V_j^\mu}{\partial w_{jk}} \\ &= \eta\sum_{\mu i}[\zeta_i^\mu - O_i^\mu]g'(h_i^\mu)W_{ij}\,g'(h_j^\mu)\xi_k^\mu \\ &= \eta\sum_{\mu i}\delta_i^\mu W_{ij}\,g'(h_j^\mu)\xi_k^\mu \\ &= \eta\sum_{\mu}\delta_j^\mu\xi_k^\mu\end{aligned}$$

siendo ahora $\delta_j^\mu = g'(h_j^\mu)\sum_i W_{ij}\delta_i^\mu$

En general, para cualquier número de capas, vale $\Delta w_{pq} = \eta\sum_{\text{patterns}}\delta_{\text{output}}\times V_{\text{input}}$

V_{in} entradas de la capa anterior o entradas reales

δ_{output} como en (1) o en (2), dependiendo de si es la última capa de conexiones o una anterior.

Observación: los δ de una capa oculta se calculan a partir de los de las unidades que esa capa alimenta (de ahí el nombre de error backpropagation)

IMPLEMENTACION

Aprendizaje:

<u>*sincrónico*</u>: primero se calculan todas las salidas (para todo μ) y luego todos los δ → batch

<u>*asincrónico*</u>: se entrena con un patrón por vez

g habituales:

$$g(h) = f_\beta(h) = \frac{1}{1+\exp(-2\beta h)} \qquad (1) \qquad \rightarrow \text{rango } (0,1)$$

$$g(h) = \tanh \beta h \qquad (2) \qquad \rightarrow \text{rango } (-1,1)$$

Se cumple entonces $\quad g'(h) = 2\beta g(1-g) \quad$ para (1)

$$g'(h) = \beta(1-g^2) \quad \text{para (2)}$$

→ *facilitan el cálculo de los* δ_i

Pasos a seguir (versión asincrónica o secuencial)

(mantenemos la notación: *M* número de capas

Vi(m) salida de la i-ésima neurona de la m-ésima capa

Vi(0) = ξi

wij(m) conexión de *Vj*(m-1) a *Vi*(m)

1- *Inicializar los w (pequeños y al azar)*

2- *Elegir un patrón (μ)* $V_k^0 = \xi_k^\mu$ **para todo** k .

3- *Etapa forward* $V_i^m = g(h_i^m) = g\left(\sum_j w_{ij}^m V_j^{m-1}\right)$ **para todo *i*, *m* hasta los *V* finales**

4- $\delta_i^M = g'(h_i^M)[\zeta_i^\mu - V_i^M]$ **deltas de la capa de salida para el patrón considerado**

5- *Etapa backward: retropropagación de errores*

$$\delta_i^{m-1} = g'(h_i^{m-1}) \sum_j w_{ji}^m \delta_j^m$$ ***m* = *M, M*-1, . . . , 2**

6- *Actualización* $$\Delta w_{ij}^m = \eta \delta_i^m V_j^{m-1}$$

$$w_{ij}^{\text{new}} = w_{ij}^{\text{old}} + \Delta w_{ij}$$

7- *ir a 2- (seleccionar patrón)*

EXTENSIONES Y VARIANTES

Dos defectos principales de BP: *lento*
Mínimos locales

Algunas mejoras:

- *Momento*:

$$\Delta w_{ij}(n+1) = -\eta \partial E/\partial w_{ij} + \alpha\, \Delta w_{ij}(n)$$

--> Si superficie de costo plana, acelera en un factor $1/(1-\alpha)$
Si hay oscilaciones, las fluctuaciones son escaladas por η

- *Parámetros adaptivos*:

$$\Delta\eta = \begin{cases} +a & \text{si } \Delta E < 0 \text{ en los últimos pasos} \rightarrow \text{crece aritméticamente} \\ -b\eta & \text{si } \Delta E > 0 \rightarrow \text{decrece geométricamente} \\ 0 & \text{en otro caso} \end{cases}$$

Si $\Delta E > 0 \rightarrow \eta$ decrece $\rightarrow$ se anula la modificación
a = 0 hasta un paso exitoso (si se estaba usando momento)

- *Otras técnicas determinísticas*: steepest descent
Gradientes conjugados
Quasi-Newton

- *Técnicas estocásticas*