

TEMA 6 Redes de Neuronas Artificiales

Francisco José Ribadas Pena

INTELIGENCIA ARTIFICIAL

5º Informática

ribadas@uvigo.es

29 de noviembre de 2005

6.1 Introducción

- **Objetivo:** Usar los principios de organización de los cerebros biológicos para construir sistemas inteligentes. (*IA subsimbólica*)
 - RNA → Emulación (modelo matemático) del funcionamiento del cerebro a bajo nivel.
 - IA simbólica → Simula el comportamiento inteligente (interesa el resultado inteligente)
- **Cerebro/RNAs:** Sistemas masivamente paralelos formados por un gran número de elementos de procesos (EPs) simples (neuronas) interconectados

| | CEREBRO | ORDENADOR |
|---------------------|--|--|
| EPs | Muchos EPs simples (10^{11} neuronas + 6×10^{14} conex.) Baja velocidad proceso (ms.) | Pocos (1..miles) EPs muy complejos y potentes Alta velocidad proceso (μs) |
| Memoria | Distribuida entre EPs Direccionable por contenido | Separada de EPs No direcc. por contenido |
| Procesam. | Altamente distribuido y paralelo Puede aprender/mejorar Alta tolerancia fallos (pérdida neuronas no afecta) | Programas secuenciales y centralizados Proceso fijo y precodificado Poco robusto ante fallos |
| Aplicaciones | Percepción y predicción | Proceso numérico y simbólico. |

- Aplicaciones típicas.
 - En dominios difíciles de formalizar (necesidad aprendizaje)
 - Entradas/salidas muchas dimensiones. Entradas con ruido.
 - Tareas clasificación/reconocimiento de patrones.
 - Comprensión por humanos poco importante.
 - Percepción: {
 - reconocim./generación de voz
 - reconocim. formas (OCR, ...)
 - identificación personas (voz, huellas, iris,...)
 - Predicción: Predicción series temporales: ciclos consumo energía, valores bursátiles, ...)
 - Otras: clasificación, aproximación funciones, filtrado adaptativo de señales

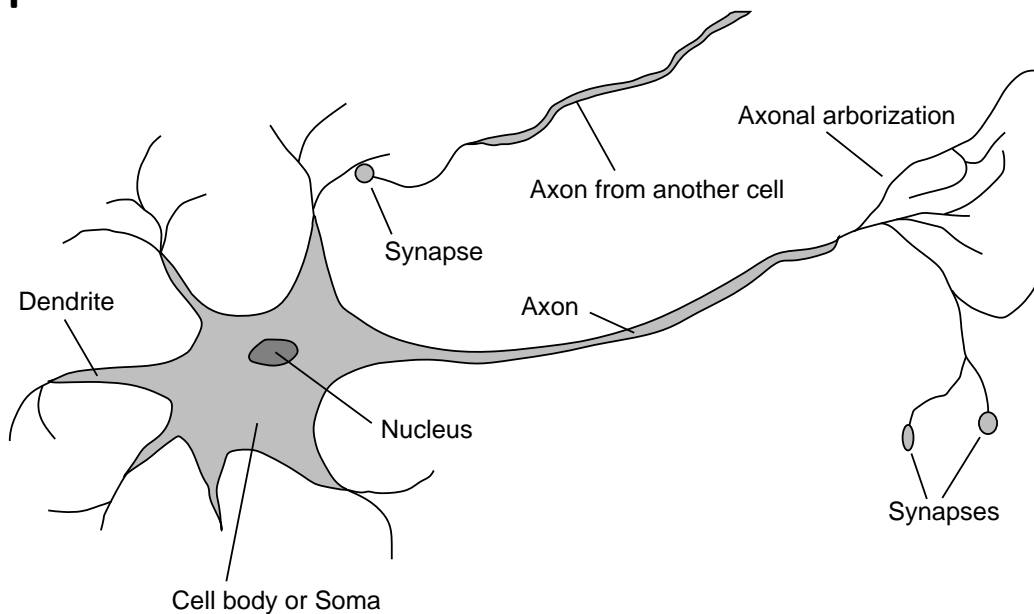
6.2 Neuronas Biológicas y Artificiales

(a) Fundamentos Biológicos

- **Neurona biológica:** Procesador de información muy simple basado en procesos electroquímicos.

10^{11} neuronas, con miles de conex. de entrada y cientos de salida (6×10^{14} conex.)

- **Componentes:**



- **SOMA:** Cuerpo de la célula
 - Realiza el "procesamiento"
- **AXON:** Elemento de salida con múltiples ramificaciones.
 - Transporta impulsos nerviosos a otras neuronas
- **DENDRITAS:** Elementos de entrada
 - Reciben señales de excitación/inhibición de otras neuronas a través de las sinapsis
- **SINAPSIS:** Áreas de contacto entre neuronas
 - Conexiones unidireccionales, dos tipos

| | |
|---|-------------|
| { | excitadoras |
| | inhibidoras |
 - No hay contacto físico, (separación)
 - Transmisión de info. en forma electroquímica (iones +/-), gobernada por *neurotransmisores*

■ **Funcionamiento**

- Neurona (soma) "acumula" todos los potenciales positivos y negativos que recibe en sus entradas (dendritas)
- Si la suma de esos impulsos es "suficiente", cambia su potencia y genera su salida en el axón que se propagará a otras neuronas.

■ **Aprendizaje**

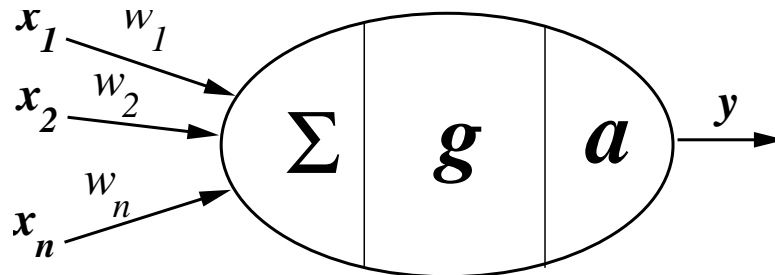
- La intensidad de las sinapsis (conexiones) puede modificarse.
 - Conexiones más o menos fuertes.
 - Permite modificar comportamiento de la neurona para adaptarse a nuevas situaciones (aprendizaje)
- También es posible modificar el comportamiento mediante creación/destrucción de sinapsis y por la muerte de neuronas

(b) Redes de Neuronas Artificiales

- **Objetivo:** Emular funcionamiento de neuronas biológicas
- Una RNA está formada por un conjunto de EPS (neuronas artificiales) unidas por conexiones unidireccionales ponderadas (con un peso asociado)
 - Normalmente se organizan en capas.
- El procesamiento en cada EP es local
 - Depende sólo de $\left\{ \begin{array}{l} \text{entradas} + \text{pesos conexión} \\ \text{estado anterior del EP (opcional)} \end{array} \right.$
- La red se adapta mediante un aprendizaje que modifica los pesos de las conexiones.
 - El conocimiento se almacena en los pesos sinápticos.
- Elementos clave de las RNAs:
 - Procesamiento paralelo: EPs/neuronas operan en paralelo
 - Memoria distribuida: info. almacenada en las conexiones
 - Aprendizaje: modificación pesos de conexiones en base a ejemplos de aprendizaje

(c) Modelo de Neurona Artificial

- **Neurona Artificial/EP:** Dispositivo que genera una salida única a partir de un conjunto de entradas con pesos asociados.
 - Entradas: binarias ($\{0, 1\}$), bipolares ($\{-1, +1\}$), continuas ($[0, 1]$), etc
 - Pesos sinápticos: representan intensidad de la conexión ($[0, 1]$)



$$y = a (g (\Sigma (\vec{x}, \vec{w})))$$

- Caracterizadas por tres funciones:
 1. **Función de transferencia (Σ):** Calcula la entrada al EP.
 - Combina valores de pesos y entradas
 - Ejemplos:

| | | |
|---|-----------------|--|
| { | suma ponderada: | $\sum_{i=1}^n w_i x_i$ (más frecuente) |
| | distancia: | $\sqrt{\sum_{i=1}^n (x_i - w_i)^2}$ |
 2. **Función de activación (g):** Calcula estado de activación de la neurona en función de las entradas y, opcionalmente, del estado de activación actual.

Ejemplos:

| | | |
|----------------|--------------|--|
| <i>escalón</i> | <i>signo</i> | <i>sigmoideal</i> $\left[g(x) = \frac{1}{1+e^{-x}} \right]$ |
|----------------|--------------|--|

3. **Función de salida (a):** Genera la salida de la neurona en función del estado de activación.
 - Normalmente, función identidad, $a(g(x)) = g(x)$.

6.3 Evolución Histórica

■ Orígenes

- *McCulloch, Pitts*(1943): Primer modelo de neurona artificial

- *Hebb*(1949): Aprendizaje neuronal (regla de Hebb)

'' Una sinapsis aumenta su eficacia (peso) si las dos neuronas que conecta tienden a estar activas o inactivas simultáneamente. Ocurre en el caso contrario, una atenuación de ese peso sináptico''

''Si dos unidades están activas simultáneamente, entonces el peso de la conexión entre ellas debe ser incrementado en proporción a esa actividad conjunto''

- *Rosenthal*(1958): Desarrollo *perceptron* (red simple, 1 capa)

- *Widrow, Hoff*(1960): Desarrollo *adaline*

- Primera aplicación industrial real (cancelación ecos linea telef.)

■ Declive Finales 60's-80's

- *Minsky, Papert*: Estudio sobre limitaciones del perceptron.

- Reducción investigación.

- Falta de modelos de aprendizaje.

■ Resurgir

- *Hopfield*(principios 80s): Desarrollo red de *Hopfield*

- Memoria asociativa

- Sólida base teórica (fisiología cerebro + física (termodinámica))

- *Rumelhart, McLellan*(1986): Alg. aprendizaje retropropagación (*back-propagation*)

- Aplicable a perceptrones complejos (multicapa)

6.4 Tipos de R.N.A.

- **RNA:** Conjunto de EPs interconectados.
- **Arquitectura de RNA:** Topología de las conexiones entre EPs.
 - En general, estructura en capas de EPS homogéneos.
 - Capa de entrada: Reciben datos del entorno.
 - Capa de salida: Proporcionan salida de la red al entorno.
 - Capa oculta: No interacciona con el entorno.
 - En otros casos: Posibilidad de conexión intercapa, entre EPs de capas no contiguas, o en sentido inverso (realimentación)
- **Clasificación de RNAs**
 1. Por **estructura de capas:** $\left\{ \begin{array}{l} \text{monocapa} \\ \text{multicapa} \end{array} \right.$
 2. Por **flujo de info. de activación:**
 - a) Redes de propagación hacia adelante (unidireccionales):
 - Flujo de entrada a salida. Respetar orden de las capas.
 - Ej.: perceptron multicapa (percepción, predicción, ...), redes de base radial
 - b) Redes recurrentes:
 - Permiten conexiones en sentido contrario (realimentación).
 - Ej.: red Hopfield (monocapa con interconex. total) (memoria asociativa)
 3. Por **tipo de aprendizaje:**
 - a) Aprendizaje supervisado
 - Red se entrena con pares [entrada, salida_correcta].
 - Ajustar pesos para minimizar error ($salida_correcta - salida_red$)
 - Patrones entrenamiento: $[x_1, x_2, \dots, x_n, o_1, o_2, \dots, o_m]$
 n EPs entrada, m EPs salida
 - Ej.: perceptrones, redes de base radial

b) Aprendizaje no supervisado

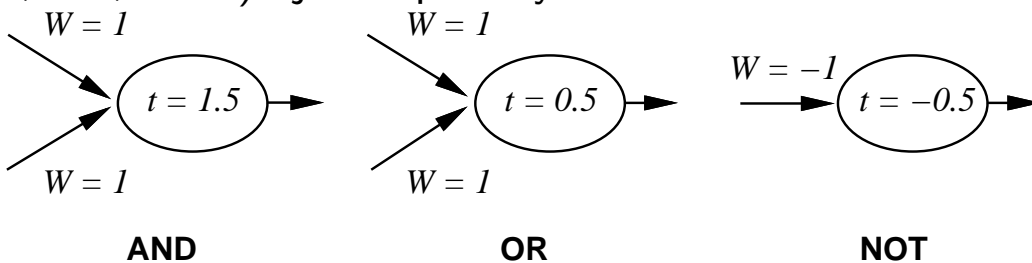
- En entrenamiento, red recibe sólo patrones de entrada.
- La red es capaz de identificar regularidades en esos patrones
- Aprende a extraer características y a clasificar/agrupar patrones de entrada
- Ej.: Mapas auto-organizativos de Kohonen.

6.4.1 Modelo de McCulloch-Pitts

- Primer modelo de neurona artificial (1943).
- Entradas (x_i) y salida (o) binarias $\{0, 1\}$.
- Función de activación de tipo escalón con umbral, T .

$$o = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_i x_i \geq T \\ 0 & \text{si } \sum_{i=1}^n w_i x_i < T \end{cases}$$

- Una neurona puede "simular" cualquier función booleana simple (AND, OR, NOT) fijando pesos y umbrales adecuados.



Para AND:

| x_1 | x_2 | $\sum w_i x_i$ | o |
|-------|-------|----------------|-----|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 2 | 1 |

Para OR:

| x_1 | x_2 | $\sum w_i x_i$ | o |
|-------|-------|----------------|-----|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 2 | 1 |

- Funciones complejas = combinación de funciones simples.

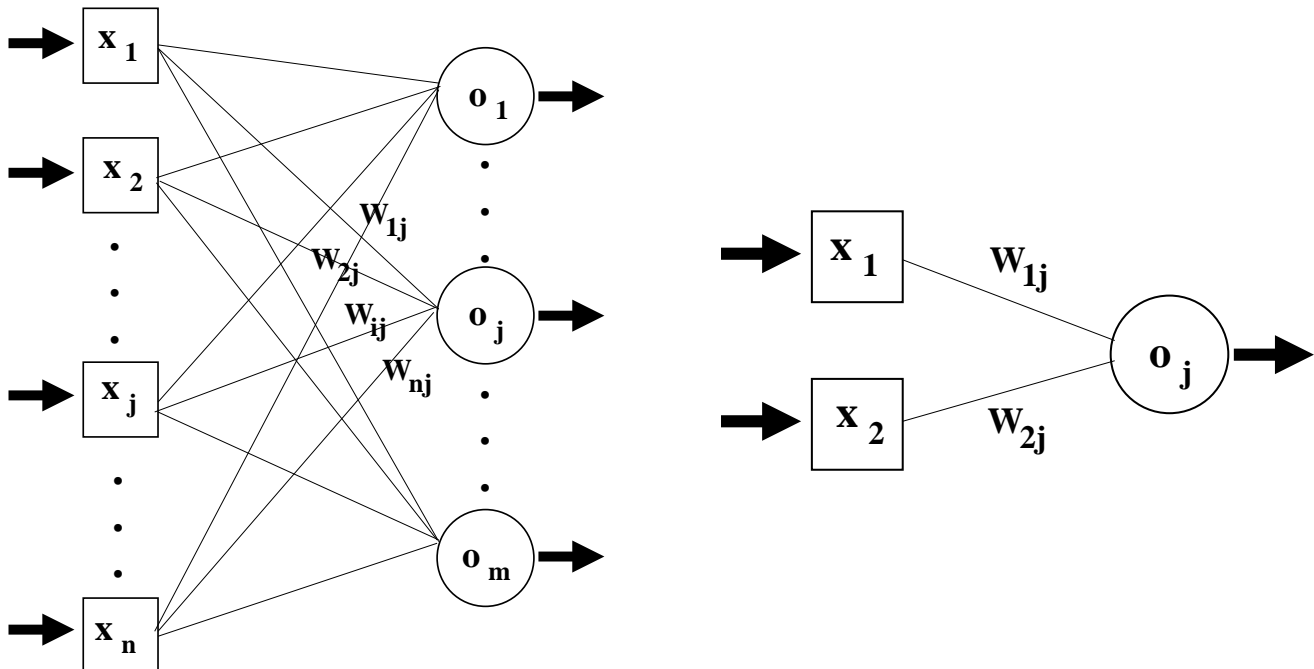
6.4.2 Perceptrón Simple

- Modelo de red neuronal propuesto por Roseblatt (1959)
 - Red monocapa con alimentación hacia adelante.
 - Colección de neuronas similares a McCulloch-Pitts, con entradas continuas $[0, 1]$, umbral T_j y salida bipolar $\{-1, 1\}$.

$$o_j = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_{ij}x_i \geq T_j \\ -1 & \text{en caso contrario} \end{cases}$$

NOTA: normalmente se evita usar el umbral directamente y se sustituye por una entrada ficticia x_0 con valor fijo -1 , cuyo peso w_{0j} se corresponderá con el umbral T_j .

$$o_j = \begin{cases} 1 & \text{si } \sum_{i=0}^n w_{ij}x_i \geq 0 \\ -1 & \text{en caso contrario} \end{cases}$$



- Capaz de aprender a clasificar patrones linealmente separables.
 - Salida +1 si pertenece a la clase, -1 si no pertenece
 - Pesos determinan zonas de clasificación separadas por un hiperplano (en el caso de 2 entradas, la separación será una línea)
 - Ejemplo: con dos entradas:

$$\text{Salida unidad } o_j = 1 \text{ si : } w_{1j}x_1 + w_{2j}x_2 \geq T_j$$

$$\text{Recta separación : } x_2 \geq - \left(\frac{w_{1j}}{w_{2j}} \right) x_1 + \frac{T_j}{w_{2j}}$$

- Minsky y Papert (1969) mostraron esta limitación del perceptrón para representar/aprender funciones no separables linealmente (ejemplo: XOR no separable linealmente, necesita mas de 1 capa)

Aprendizaje en Perceptrón Simple

- Aprendizaje supervisado.
 - Asociación: [patrón entrada, patrón salida].
 - Ajuste de pesos de conexión (aprendizaje) en base al error (salida deseada vs. salida obtenida)

Inicializar pesos aleatoriamente

Repetir hasta tener salidas "suficientemente" correctas (EPOCH)

Para cada patrón de entrenamiento (E,T)

Calcular salida actual (O)

Comparar con salida deseada (T-O)

Actualizar pesos según regla de aprendizaje

- Actualización de pesos: **Regla del Perceptron**

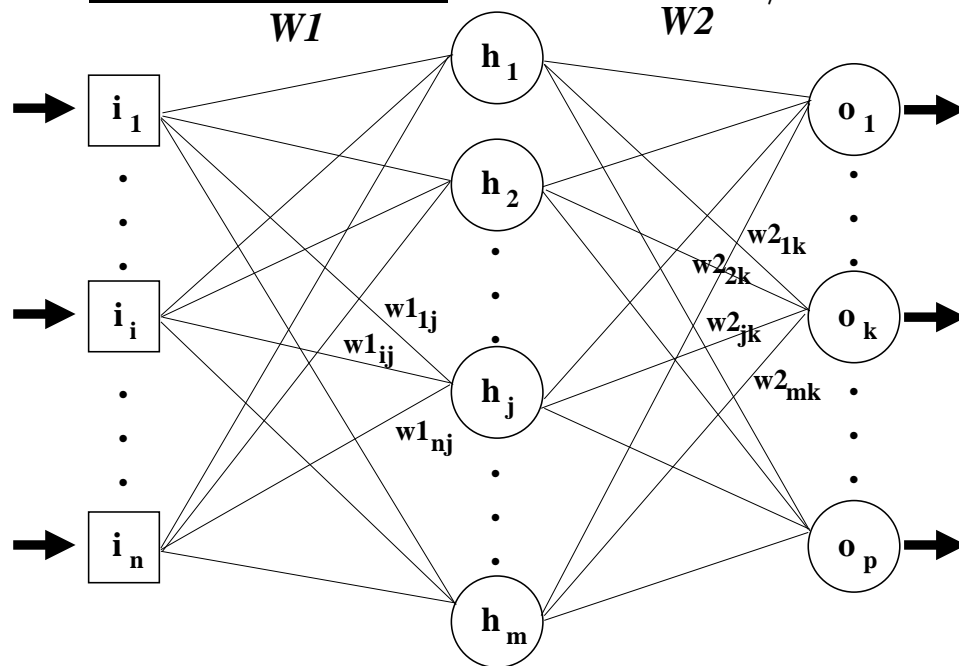
- Para la neurona j , sean: $\begin{cases} t_j : \text{salida } \underline{\text{deseada}} \\ o_j : \text{salida } \underline{\text{obtenida}} \end{cases}$
- Nuevos pesos en neurona j :

$$W_{ij} = W_{ij} + \alpha x_i (t_j - o_j)$$

- Con: $\alpha =$ tasa/velocidad de aprendizaje
 - Gobierna la "brusquedad" de los cambios de pesos.
 - Al inicio del entrenamiento alta (0.9). Al final baja (0.05).
- Interpretación intuitiva:
 - Repartir el error $(t_j - o_j)$ entre los pesos de las entradas (x_i) implicadas.
 - Salida correcta: no cambia pesos
 - Salida *baja* ($t_j = +1; o_j = -1$)
 - ◇ Aumentar pesos de las entradas no nulas.
 - ◇ Variación peso: $w_{ij} = w_{ij} + \beta x_i$ ($\beta = 2\alpha$)
 - Salida *alta* ($t_j = -1; o_j = +1$)
 - ◇ Decrementar pesos de las entradas no nulas.
 - ◇ Variación peso: $w_{ij} = w_{ij} - \beta x_i$ ($\beta = 2\alpha$)
- El umbral se ajusta como un peso mas (w_{0j}).

6.4.3 Perceptrones Multicapa (I)

- Extensión del perceptron simple → añade capas adicionales.
- Capas con interconexión total: entrada, oculta/s, salida.



- Propagación hacia adelante:
 - Patrones de entrada se presentan en capa de entrada.
 - Se propagan hasta generar salida.
- Función activación neuronas: **sigmoidal**.

$$g\left(\sum w_{ij}x_i\right) = \frac{1}{e^{-\sum w_{ij}x_i}}$$

- Entradas y salidas continuas $[0, 1]$
- Pesos de conexión determinan una función que relaciona entradas con salidas.
 - Sin capa oculta: funciones lineal^{te} separables (perceptron simple)
 - Una capa oculta: funciones continuas
 - Dos capas ocultas: funciones no continuas
- Método de entrenamiento: **retropropagación** (*backpropagation*)

6.4.3 Perceptrones Multicapa (Método Retropropagación)

- Propuesto por Rumelhart (1984)
- Objetivo: ajustar pesos para reducir el error cuadrático de las salidas.
- Funcionamiento aprendizaje:

Inicializar pesos aleatoriamente

Repetir hasta tener salidas "suficientemente" correctas (EPOCH)

Para cada patrón de entrenamiento (E,T)

Propagar E por la red para obtener salida actual (O)

Comparar con salida deseada (T-O)

Actualizar pesos hacia atrás, capa a capa.

- Exige función de activación (g) continua y derivable \Rightarrow sigmoideal.

$$g(x) = \frac{1}{e^{-x}} \quad g'(x) = g(x)(1 - g(x))$$

- (1) Ajuste pesos CAPA SALIDA

Nuevos pesos para neurona de salida o_k :

$$W_{2_{jk}} = W_{2_{jk}} + \alpha h_j \Delta_k$$

- Idea base: "Repartir" error obtenido para cada neurona de salida (Δ_k) entre los pesos de sus conexiones de forma proporcional a la intensidad de la entrada recibida (h_j).

- Usa: $\left\{ \begin{array}{l} h_j : \text{activación neurona oculta } h_j \\ (T_k - o_k) : \text{error (salida deseada - salida obtenida)} \\ g'(ent_k) : \text{derivada func. activ. (sigmoideal: } g'(ent_k) = o_k(1 - o_k)) \\ \alpha : \text{tasa de aprendizaje} \end{array} \right.$

$$\Delta_k = g'(ent_k) (T_k - o_k) = o_k (1 - o_k) (T_k - o_k)$$

■ (2) Ajuste pesos CAPA/S OCULTA/S

Nuevos pesos para neurona oculta h_j :

$$W1_{ij} = W1_{ij} + \alpha i_i \Delta_j$$

- Problema: Cuantificar error en las capas ocultas (Δ_j).
- Idea: Propagar la parte proporcional del error en la capa de salida (Δ_k) del cual es "responsable" cada neurona oculta h_j .
 - **NOTA**: En este caso se considera *solo 1 capa oculta* que realimenta hacia atras el error presente en la capa de salida.
Si hubiera mas de 1 capa oculta, la idea seria la misma: realimentar el error presente en la siguiente capa [oculta o de salida].
- **Estimacion del error en neurona oculta h_j** :

$$\Delta_j = g'(ent_j) \sum_{k=1}^p (W2_{jk} \Delta_k) = h_j (1 - h_j) \sum_{k=1}^p (W2_{jk} \Delta_k)$$

6.4.3 Perceptrones Multicapa (Interpretación aprendizaje retroprop.)

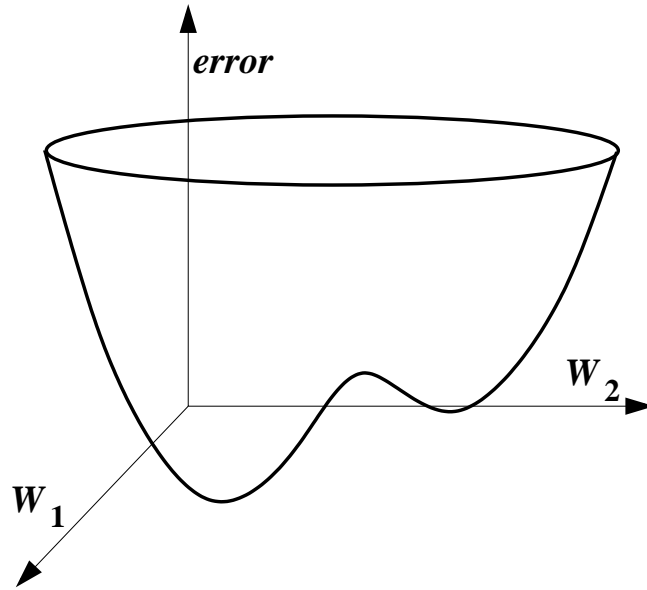
- Aprendizaje retropropag. \approx búsqueda por descenso de gradiente.
- Búsqueda en espacio de estados de tipo "ascenso a colinas"
 - Espacio de búsqueda = conjunto de posibles valores de pesos
 - Métrica de evaluación = error cuadrático (en función de pesos)

$$E(red) = \frac{1}{2} \sum_{k=1}^p (T_k - O_k)^2$$

$$E(pesos) = \frac{1}{2} \sum_{k=1}^p \left(T_k - g \left(\sum_{j=1}^m W_{2jk} g \left(\sum_{i=1}^n W_{1ij} i_i \right) \right) \right)^2$$

- En cada ejemplo de entrenamiento:
 - A partir del error calculado \rightarrow se definen los pesos de la nueva red.
 - Se ajustan pesos en la dirección de **mayor pendiente**.
 \Rightarrow Uso de la derivada g' en el ajuste pesos.
 - Objetivo: Minimizar valor de la función de error.
 - ◇ En realidad se pretende hacer que su derivada ($E'(red)$) sea 0.
 - ◇ Se buscan unos pesos para los que esa función tenga un mínimo ($E'(pesos) = 0$)
- Inconveniente: No garantizada convergencia en una red óptima.
 - Problemas con mínimos locales.
 - Funciones separables linealmente sólo tienen un mínimo local \Rightarrow se puede asegurar que se encontrará ese único mínimo.

- Ejemplo: Superficie de error con 2 pesos/entradas.



6.5 Aprendizaje supervisado en RNAs.

- En general, no se garantiza convergencia.
- En la práctica, es posible entrenamiento adecuado para problemas reales.
 - Entrenamiento lento.
 - Puede requerir muchos ciclos de entrenamiento (*epoch*).
 - Se suele usar: conj. entrenamiento + conj. validación.
 - Verificar si realmente ha aprendido y puede generalizar.
- Criterios de parada:
 - Número fijo de ciclos de entrenamiento (*epochs*)
 - Umbral de error sobre conjunto entrenamiento.
 - Umbral de error sobre conjunto de validación.
- Problemas del aprendizaje.
 1. **¿Cuál es la topología de red adecuada?**
 - Pocas neuronas/capas → incapaz de aprender función.
 - Muchas neuronas/capas → posibilidad de sobreajuste.
 - Además: ¿Qué funciones activación usar?, ¿Cómo codificar entradas/salidas?, etc,...
 2. **Sobreentrenamiento y Sobreajuste**
 - Error pequeño conj. entrenamiento, grande conj. validación.
 - Demasiados ciclos entrenamiento ⇒ Pérdida capacidad generalización.
 - Red demasiado compleja ⇒ Se ajusta a cualquier función.
 - Se memorizan patrones, no se generaliza con otros nuevos.
 3. **Mínimos locales**
 - Se mitigan con múltiples fases de entrenamiento, inicializadas con distintos pesos aleatorios.