Universidad de Granada E.T.S. Ingeniería Informática

Diseño, Análisis y Aplicaciones de Sistemas Inteligentes

Aprendizaje de Sistemas

Redes Neuronales Artificiales

José Manuel Benítez Sánchez

J.M.Benitez@decsai.ugr.es

Granada, enero de 2001

Índice

Introducción

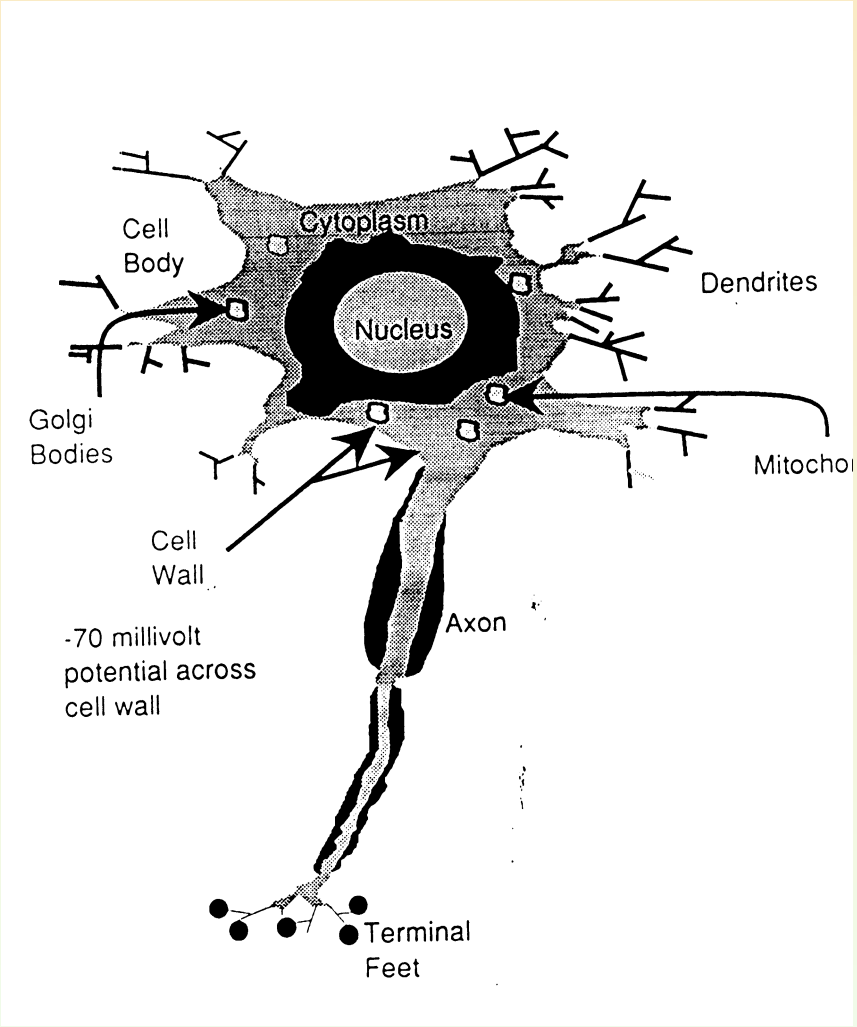
- Inteligencia Artificial:
  - Aproximar el proceso
  - Aproximar el órgano
- Órgano de la Inteligencia: Cerebro

RNA: Disciplina de carácter técnica con raíces en muchos campos: Neurociencia, Matemáticas (especialmente Estadística), Física, Biología, Psicología Cognitiva, Filosofía, Informática e Ingeniería.

Redes Neuronales Artificiales

- Origen biológico
- Definición
- Aplicaciones
- Componentes de una RNA

Origen biológico: neurona



Definición

(De Haykin, 1999):

RNA: Procesador distribuido masivamente paralelo construido a partir de unidades de procesamiento simples que tiene una propensión natural para almacenar conocimento experimental y utilizarlo. Se parece al cerebro en dos aspectos:

1. Extrae conocimiento del entorno a través de un proceso de **aprendizaje**

2. Las fuerzas de interconexión neuronal o pesos son usadas para almacenar el conocimiento adquirido

Aplicaciones

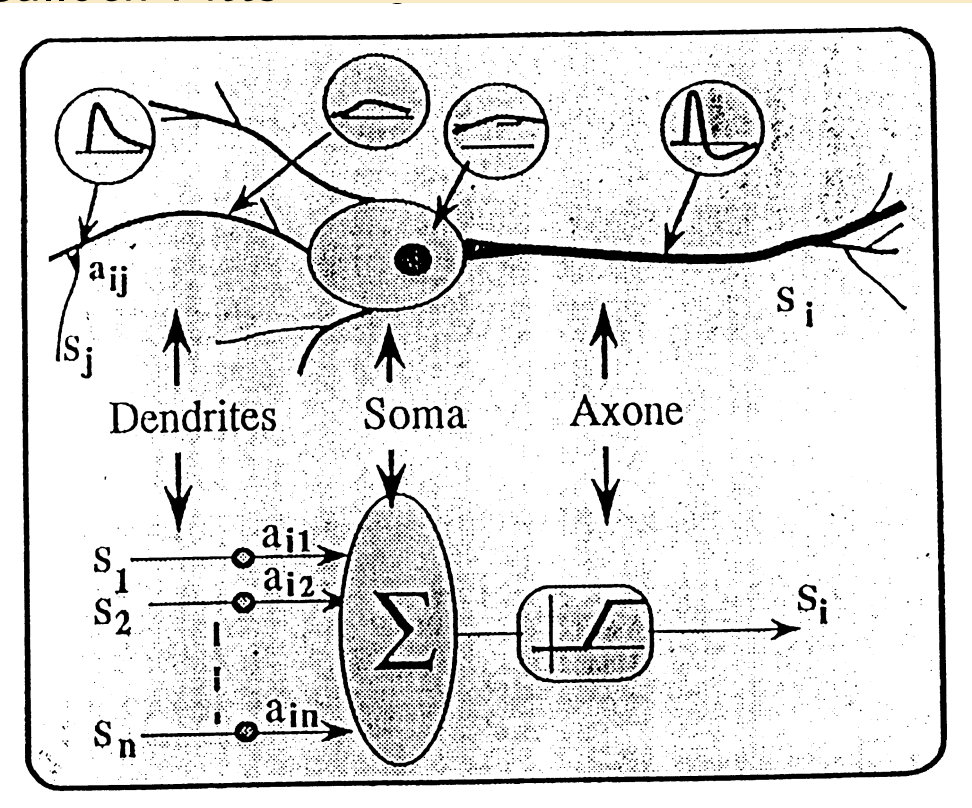
1. Reconocimiento de patrones (p.ej.: dígitos manuscritos)
2. Agrupamiento
3. Aproximación de funciones
4. Predicción
5. Asociación de patrones
6. Memoria asociativa
7. Control
8. Optimización

Componentes de una RNA

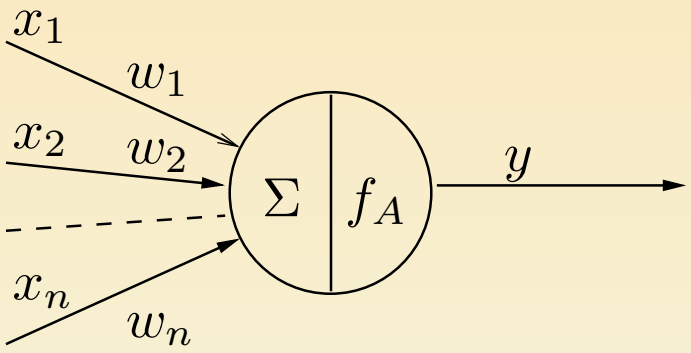
1. Arquitectura
   (a) Neuronas
   (b) Enlaces
   (c) Topología
2. Aprendizaje

Neuronas

Formulación de McCulloch-Pitts



Neurona (II)



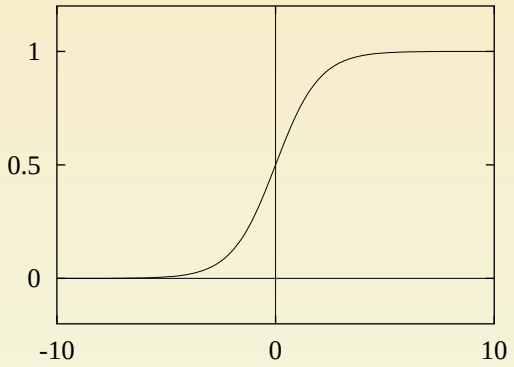
Entrada efectiva a la neurona:

$$N = \sum_i w_i x_i \tag{1}$$

Funciones de activación

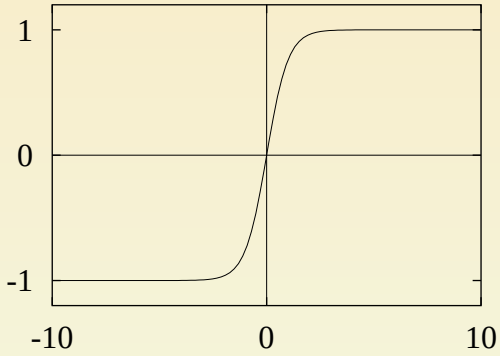
Funciones sigmoides: continuas y derivables con forma de "S".

Función Logística



$$f_A(N) = \frac{1}{1+\exp(-N)}$$

Función Tangente Hiperbólica



$$f(N) = \tanh(N)$$

A veces, también función lineal en las salidas

Topología

Nodos simples $\Longrightarrow$ Poca capacidad de cálculo

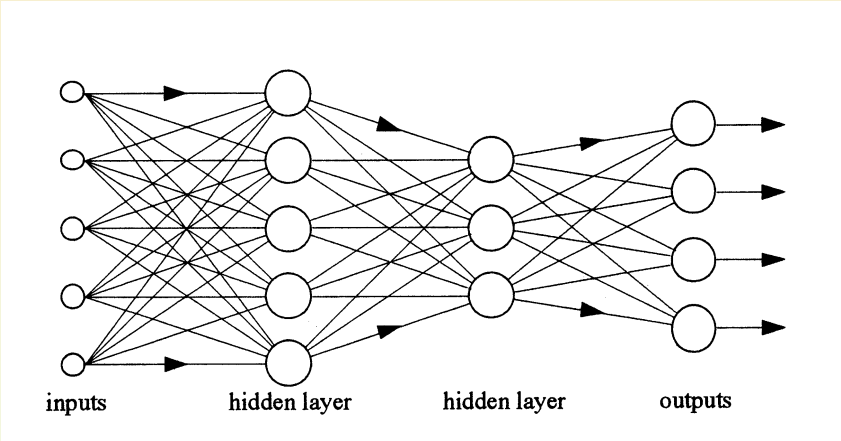
Potencia real de una RNA: Interconexión

Topología: Patrón de Interconexión.

Topología "hacia adelante"

Propagación de las señales en un solo sentido (*feedforward*).

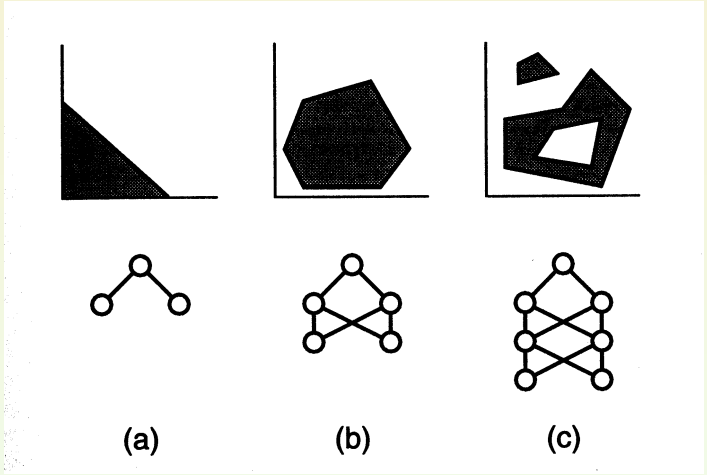
Organización habitual por *capas*, con neuronales iguales. El número de neuronas por capa puede ser distinto en cada capa



No es necesaria una estructuración perfecta en capas: Grafo Dirigido Acíclico

El papel de las capas ocultas

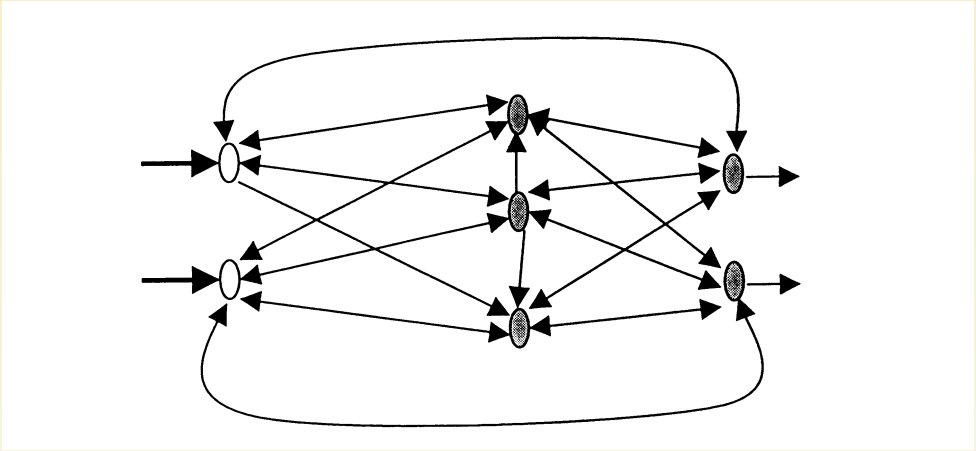
- Proporcionar un cambio de variable para hacer linealmente separable el problema
- Extracción de características
- Construir superficies de decisión más complejas



Topología con realimentación

Sí se permiten conexiones hacia atrás e intracapa

Evolución paso a paso hasta lograr la convergencia



Aprendizaje

- Concepto
- Aprendizaje supervisado y no supervisado
- Reglas de aprendizaje:
  - Corrección de error
  - Aprendizaje hebbiano
  - Aprendizaje competititvo
  - Aprendizaje probabilístico
  - Aprendizaje por refuerzo
- El Aprendizaje como optimización

Concepto de Aprendizaje

- *Aprendizaje*: Capacidad para modificar el comportamiento mediante la experiencia.
- *Aprendizaje Automático*: Disciplina encarga de estudiar y desarrollar programas informáticos que mejoran con la experiencia
- Aprendizaje de Redes Neuronales: Proceso por el que los parámetros libros de una red neuronal son adaptados de acuerdo con los estímulos de su entorno

Algoritmo de aprendizaje: algoritmo para ajustar los pesos de una R.N.A.

Tipos de aprendizaje

- **Supervisado**: Se conoce la respuesta correcta de cada ejemplo y se utiliza. El ajuste persigue acercar la respuesta de la red a la esperada.

  Variante importante: *aprendizaje por refuerzo*. No se dispone de respuesta exacta. Sólo de una señal de refuerzo que indica si la respuesta es correcta o no. Puede ser con retraso.

- **No supervisado**: No se conocen las salidas correctas. La red debe encontrar regularidades entre los datos de entrada y tiende a agrupar los ejemplos con entradas "similares". Habitualmente debe realizar algún tipo de compresión de datos.

Aprendizaje por corrección de error

El comportamiento de una RNA viene definido por una función: $\mathrm{F}(\mathbf{x}, \mathbf{w})$. $\mathbf{x}$, entrada; $\mathbf{w}$, parámetros libres.

Sea el conjunto de datos: $\{(\mathbf{x}^k, \delta^k)\}$

Error cometido por la red:

$$E = \sum_k \|\delta^k - \mathbf{y}^k\| \tag{2}$$

Aprender = Corregir el error:

$$\Delta \mathbf{w} = G(\mathbf{w}, \mathbf{x}, \mathbf{y}\delta) \tag{3}$$

Aprendizaje hebbiano

Dos reglas:

1. Si las dos neuronas conectadas por un enlace están activas simultáneamente, el enlace es reforzado

2. Si las dos neuronas conectadas por un enlace se activan asíncronamente, el enlace se debilita

Características:

- Dependencia temporal
- Regla local
- Mecanismo interactivo

$$\Delta \mathbf{w} = \eta \mathbf{x} \mathbf{y} \tag{4}$$

Aprendizaje competitivo

Las neuronas de salida **compiten** entre sí por la activación

- Neuronas con respuestas distintas
- Límite a la activación de cada neurona
- Mecanismo de competición: *Winner-takes-all*

Aprendizaje probabilístico

Ajuste NO determinístico

Función de energía:

$$E = -\frac{1}{2}\sum_j \sum_i w_{ij} x_k x_j \tag{5}$$

Probabilidad del cambio de estado:

$$p(x_k \rightarrow -x_k) = \frac{1}{1 + \exp(-\Delta E/T)} \tag{6}$$

Aprendizaje como optimización

La inmensa mayoría de los métodos de aprendizaje emplean un algoritmo de optimización subyacente como método de ajuste.

Suele ser una función fuertemente no lineal $\Longrightarrow$ métodos aproximados:

- Descenso en Gradiente
- Gradiente Conjugado
- Enfriamiento Simulado
- Algoritmos evolutivos

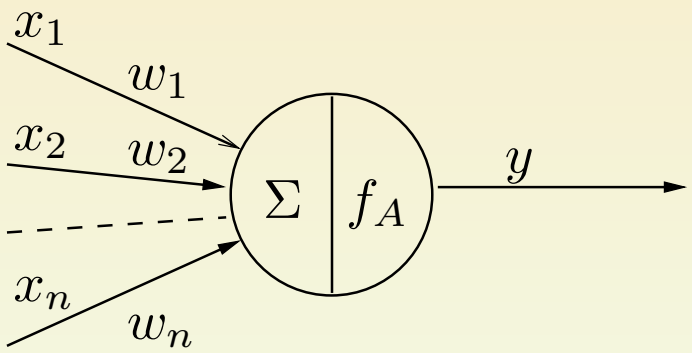
Perceptrones

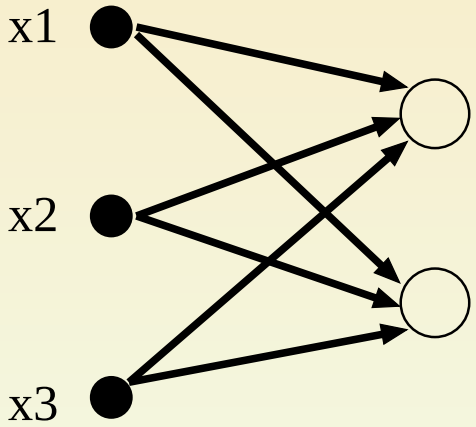
- El perceptron
  - Arquitectura
  - Algoritmo de entrenamiento de Rosenblatt
  - La regla delta
  - Limitaciones del perceptron
  - La función XOR
- El perceptron multicapa
  - Arquitectura
  - La neurona
  - Topología
  - El algoritmo de retropropagación de errores
  - Métodos avanzados

El perceptron

Porpuesto por Rosenblatt en 1985

Red de propagación hacia adelante con una sola capa:





con f_A la función umbral o signo.

Algoritmo de Rosenblatt

1: Iniciar w aleatoriamente
2: **while** haya ejemplos mal clasificados **do**
3: Seleccionar aleatoriamente un ejemplo (x, d)
4: Calcular la salida y y comparar con d
5: $\Delta w = \eta(d - y)$
6: **end while**

TEOREMA:

El conjunto de datos $\{(x, d)\}$ es finito y linealmente separable, el algoritmo anterior encuentra una solución en un tiempo finito (converge).

La regla delta

Regla basada en corrección del error: considerar el entrenamiento como minimización del error

$$E(w) = \frac{1}{2}\sum_k (o^k - y^k)^2 \tag{7}$$

$$w \leftarrow w + \Delta w \tag{8}$$

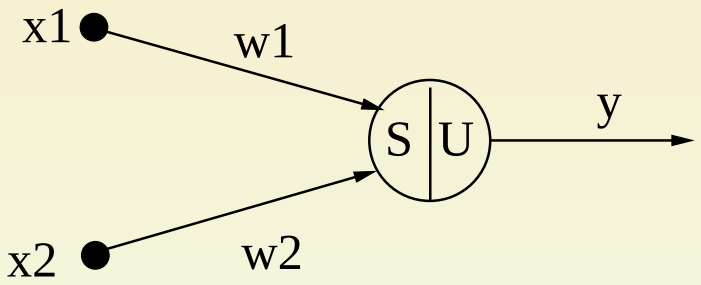
$$\Delta w = -\eta \nabla E(w) \tag{9}$$

$$= \eta \sum_k (o^k - y^k) x^k \tag{10}$$

Limitaciones del Perceptron

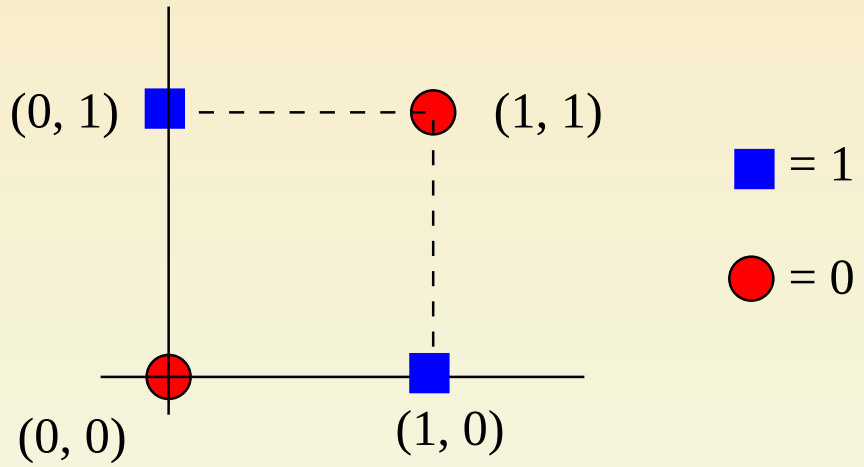
Minsky y Papert 1969: El perceptron sólo puede reseprentar problemas linealmente separables.

Existen funciones sencillas que **NO** puede representar el perceptron: función O-exclusiva (XOR).



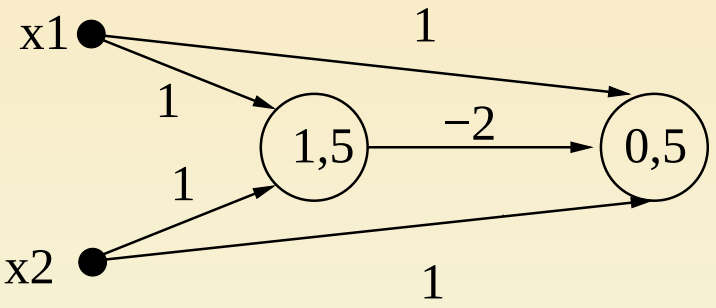
La función O-Exclusiva (XOR)

Función lógica de dos variables $\Longrightarrow$ Muy simple



Un perceptron para calcular la función XOR

El problema se puede resolver usando una unidad intermedia:



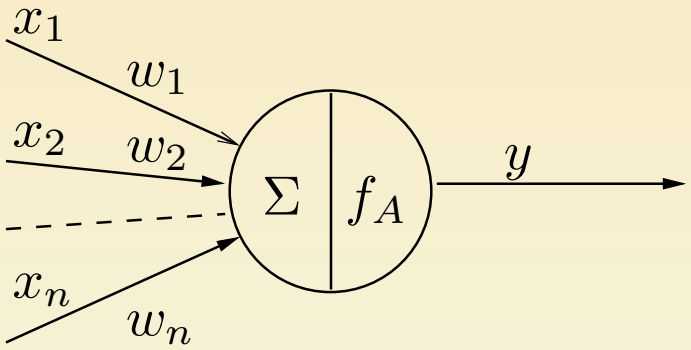
Solución a las limitaciones del perceptron: **usar redes neuronales con más de una capa** y unidades que puedan procesar señales continuas.

El Perceptron Multicapa: Arquitectura

- La neurona básica
- Topología de la red: "hacia adelante"
- El papel de las capas ocultas

La neurona básica

Variación continua de la neurona de McCulloch-Pitts: función de activación continua



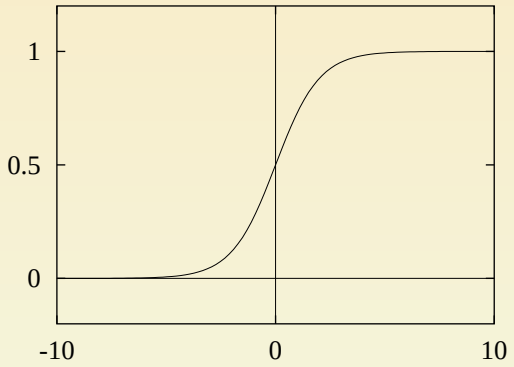
Entrada efectiva a la neurona:

$$N = \sum_i w_i x_i \tag{11}$$

Funciones de activación

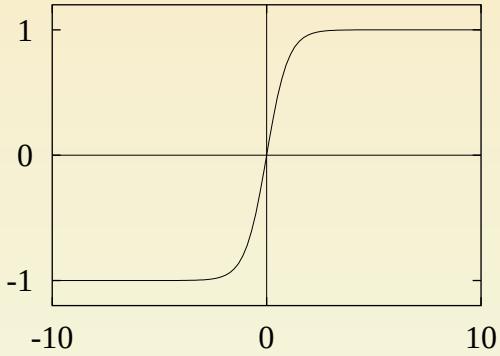
Funciones sigmoides: continuas y derivables con forma de "S".

Función Logística



$$f_A(N) = \frac{1}{1+\exp(-N)}$$

Función Tangente Hiperbólica



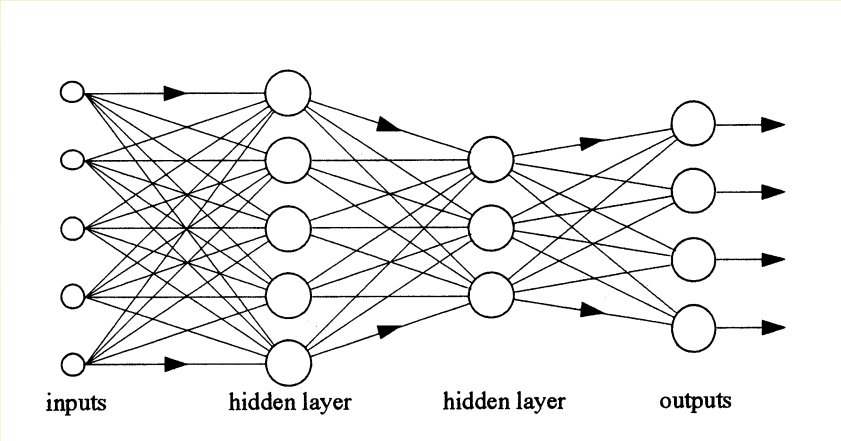
$$f(N) = \tanh(N)$$

A veces, también función lineal en las salidas

Topología "hacia adelante"

Propagación de las señales en un solo sentido (*feedforward*).

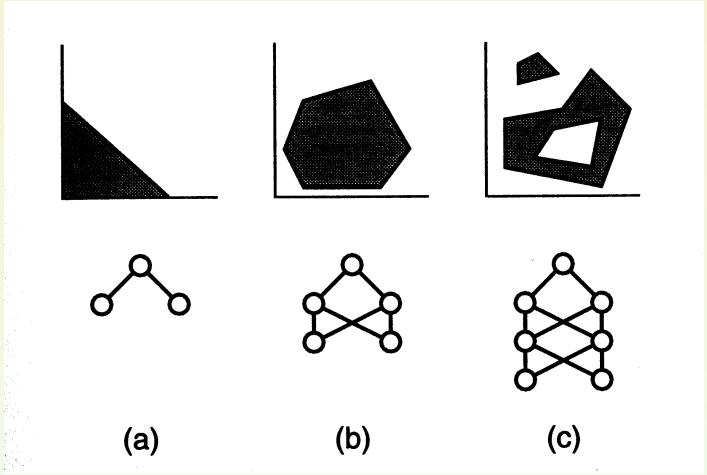
Organización habitual por *capas*, con neuronales iguales. El número de neuronas por capa puede ser distinto en cada capa



No es necesaria una estructuración perfecta en capas: Grafo Dirigido Acíclico

El papel de las capas ocultas

- Proporcionar un cambio de variable para hacer linealmente separable el problema
- Extracción de características
- Construir superficies de decisión más complejas



El algoritmo de retropropagación de errores

- Reseña Histórica
- Idea del algoritmo
- El algoritmo BP
- Consideraciones
- Particularizaciones
- Tasa de aprendizaje y momento
- Aprendizaje en línea y en "batch"
- Condiciones de parada
- Limitaciones del algoritmo

BP: Reseña Histórica

Autores

- Rumelhart, Hinton y Williams en 1986

BP: Reseña Histórica

Autores

- Rumelhart, Hinton y Williams en 1986
- Parker en 1982

BP: Reseña Histórica

Autores

- Rumelhart, Hinton y Williams en 1986
- Parker en 1982
- Werbos 1974

BP: Reseña Histórica

Autores

- Rumelhart, Hinton y Williams en 1986
- Parker en 1982
- Werbos 1974
- Bryson y Ho, 1969

BP: Reseña Histórica

Autores

- Rumelhart, Hinton y Williams en 1986
- Parker en 1982
- Werbos 1974
- Bryson y Ho, 1969
- ¿?

El algoritmo de retropropagación de errores

Entrenamiento = minimización de una función de error

Error: Diferencia entre salidas obtenidas y esperadas.

Regla δ: usa el error para ajustar los pesos entre las dos últimas capas, pesos de salida. Pero **no** es válida para los demás pesos: no conocemos su aportación al error.

Idea del algoritmo

Consideración: Las salidas de una capa son las entradas de la siguiente; propagar hacia atrás el error

Esquema iterativo en dos etapas:

1. Propagación hacia adelante: Evaluar el nivel de activación de las neuronas y calcular el error de la red

2. Propagar el error hacia atrás, capa a capa, modificando los pesos

Retropropagación de errores: *BACKPROPAGATION* (BP).

Notación

Generalización de la Regla δ

Supongamos un sistema con n entradas y m salidas: diseñamos un perceptron con n entradas y m neuronas en la capa de salida

Conjunto de pares de entrenamiento $(\mathbf{x}^k, \mathbf{t}^k)$, $\mathbf{x}^k \in \mathbb{R}^n$, $\mathbf{t}^k \in \mathbb{R}^m$, $k = 1, 2, \ldots, K$

$$\min_{\mathbf{w}} \; E(\mathbf{w}) = \sum_k \|\mathbf{t}^k - F(\mathbf{x}^k, \mathbf{w})\| \tag{12}$$

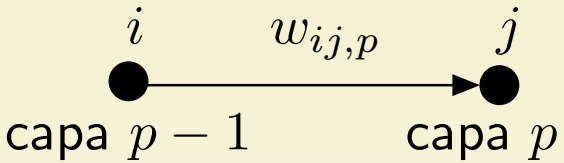
$$\mathbf{w}(n+1) = \mathbf{w}(n) + \Delta\mathbf{w}(n) \tag{13}$$

$$= \mathbf{w}(n) - \eta\nabla E \tag{14}$$

Notación (II)

Componentes del vector gradiente: derivadas parciales

$$\nabla E = \left[\frac{\partial E}{\partial w_{ij,p}} \right]_{ij,p} \tag{15}$$



Ajuste de los pesos:

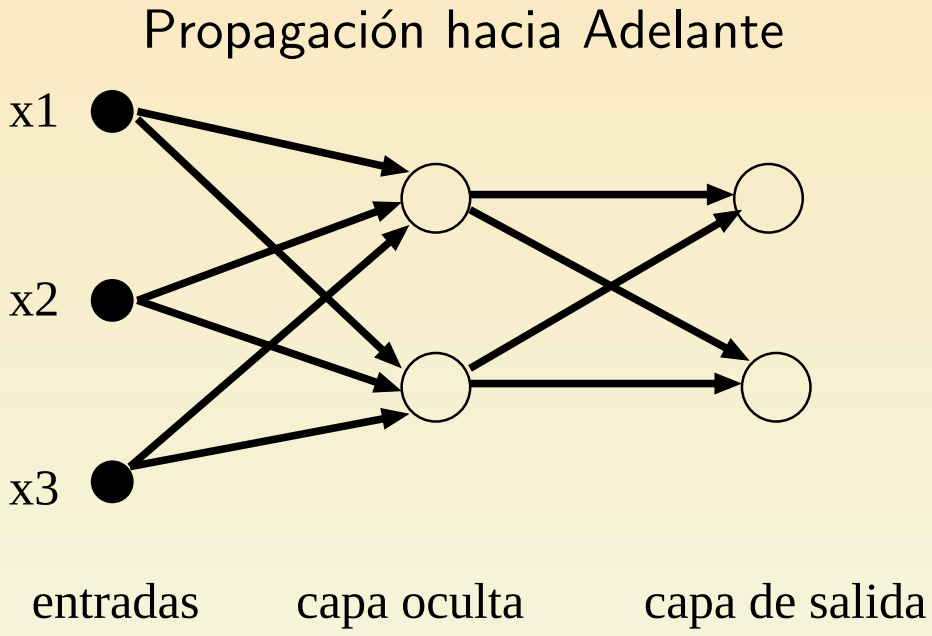
$$\Delta w_{ij,p}(n) = +\eta \delta_{j,p} y_{i,p-1} \tag{16}$$

$\delta_{j,p}$: valor asociado a cada unidad

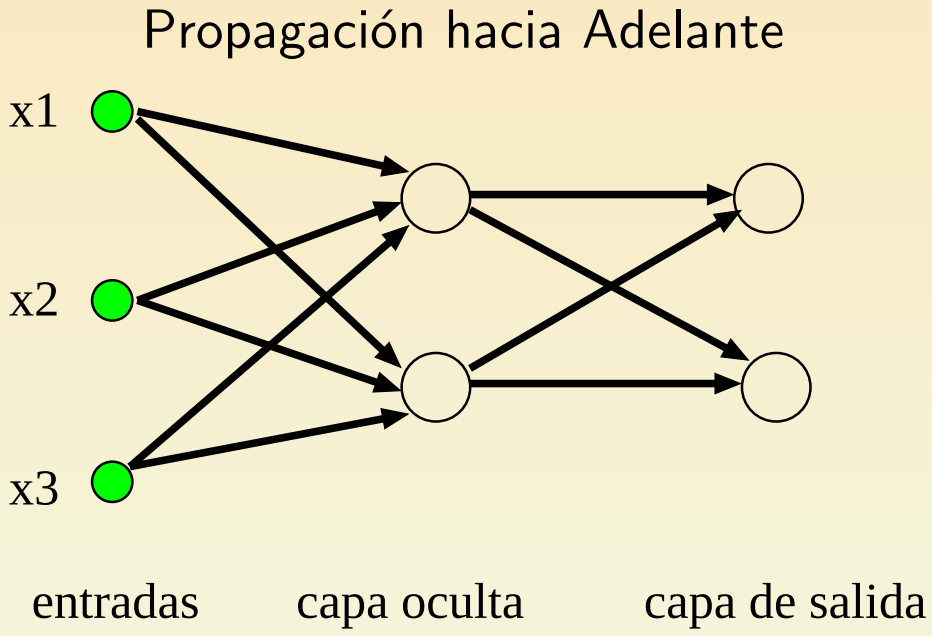
BP: El algoritmo

1: **repeat**
2: Seleccionar el siguiente par de entrenamiento.
3: Introducir la entrada en la red y calcular la salida que le corresponde.
4: Calcular el error (en términos de norma cuadrática) entre la salida obtenida y la salida deseada (el vector objetivo del par de entrenamiento).
5: Ajustar los pesos de la red de modo que se minimice el error.
6: **until** que se cumpla el criterio de parada

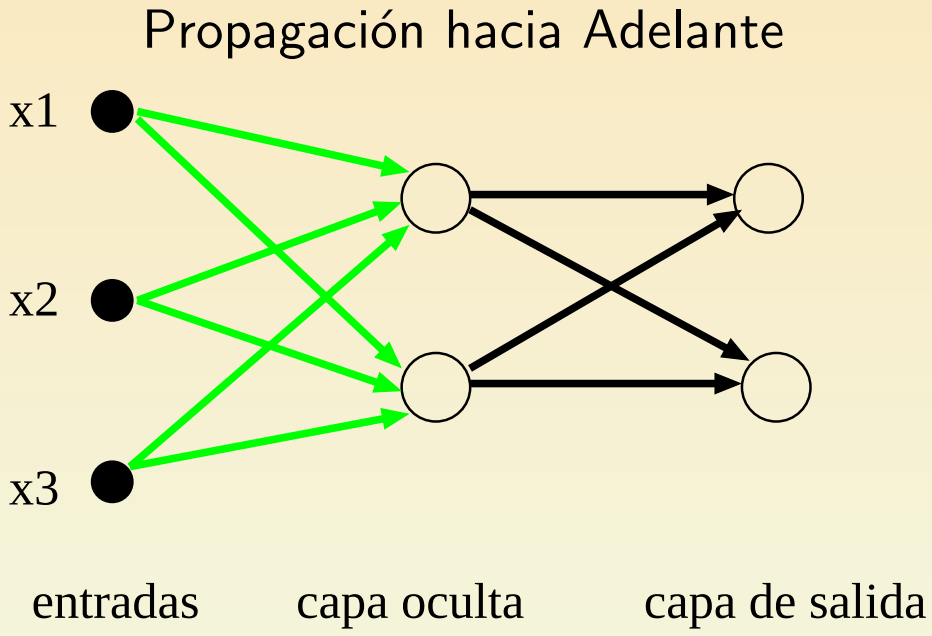
El algoritmo en acción



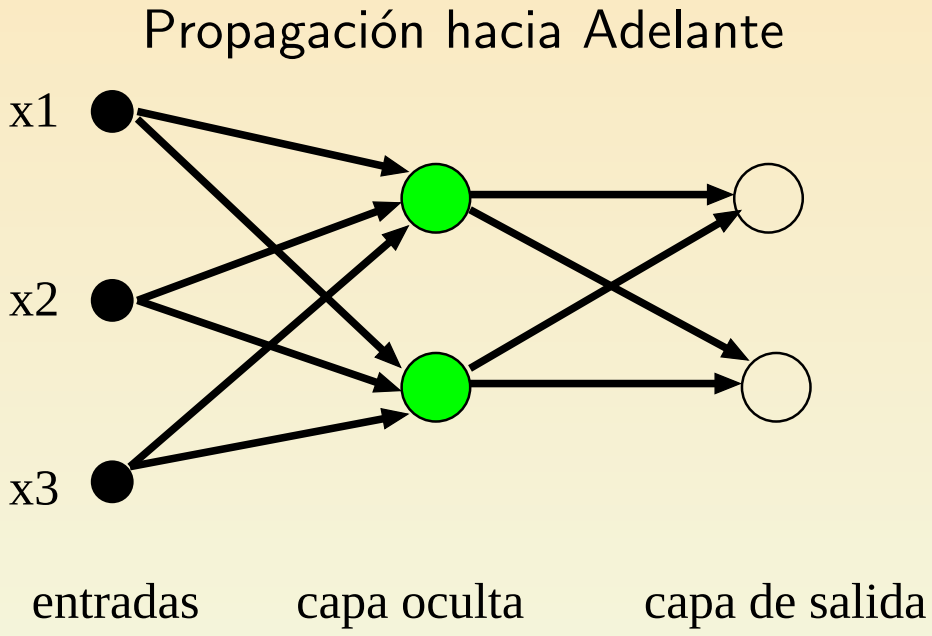
El algoritmo en acción



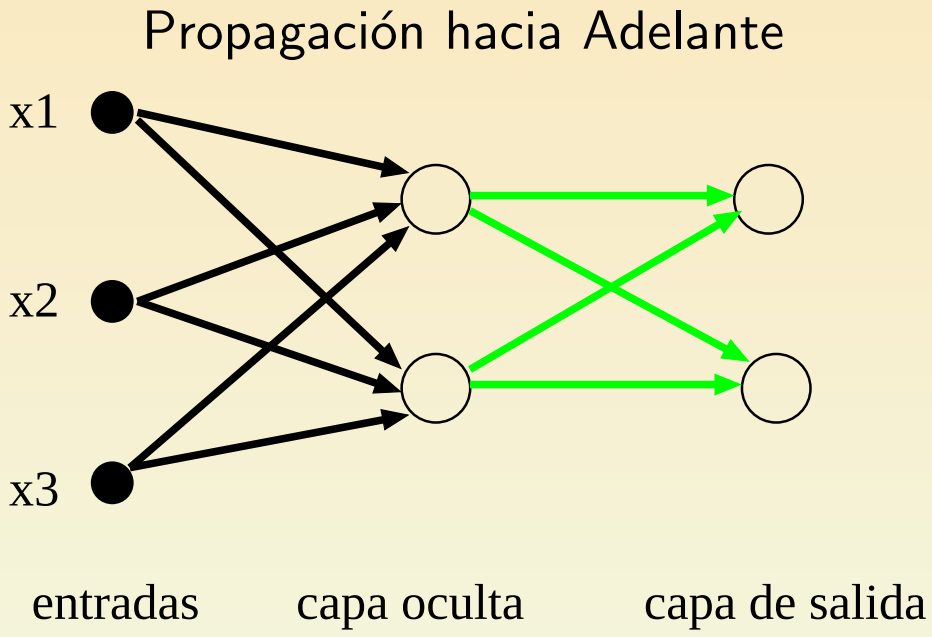
El algoritmo en acción



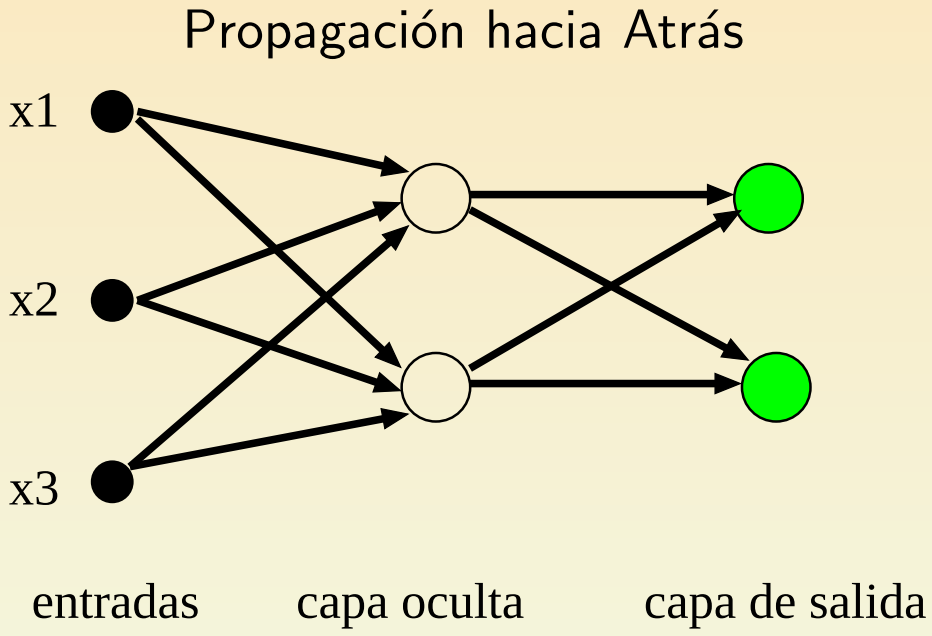
El algoritmo en acción



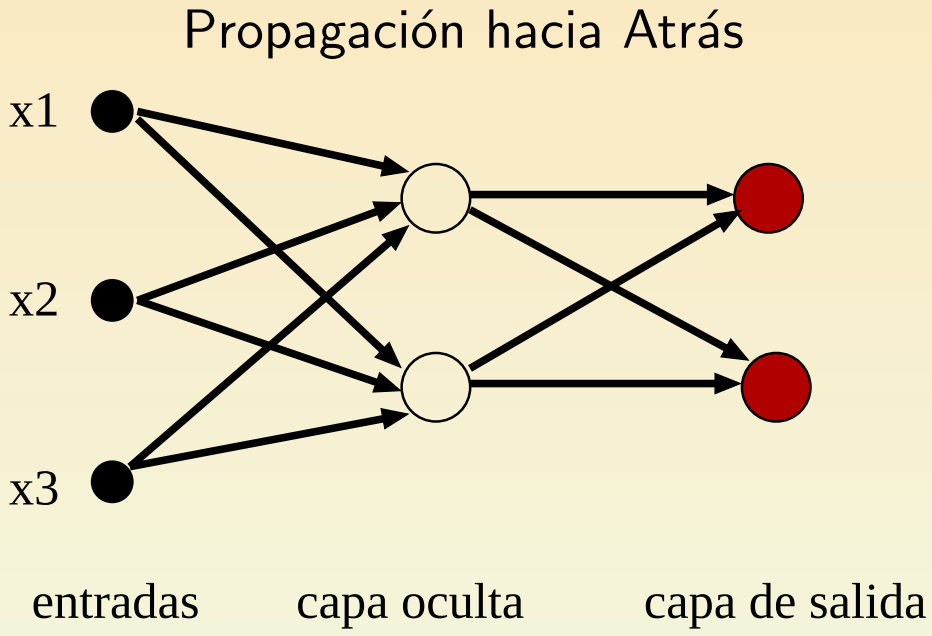
El algoritmo en acción



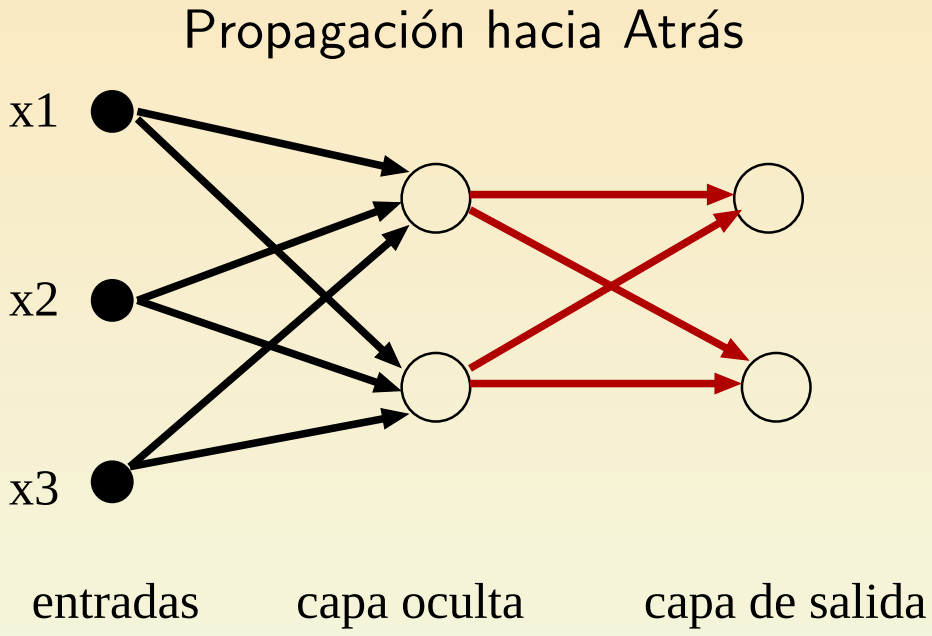
El algoritmo en acción



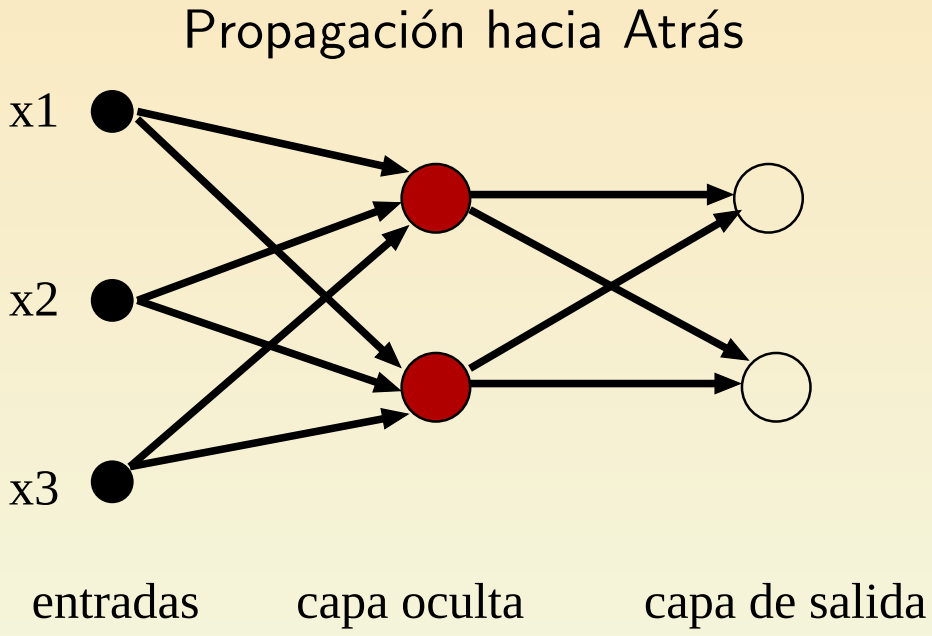
El algoritmo en acción



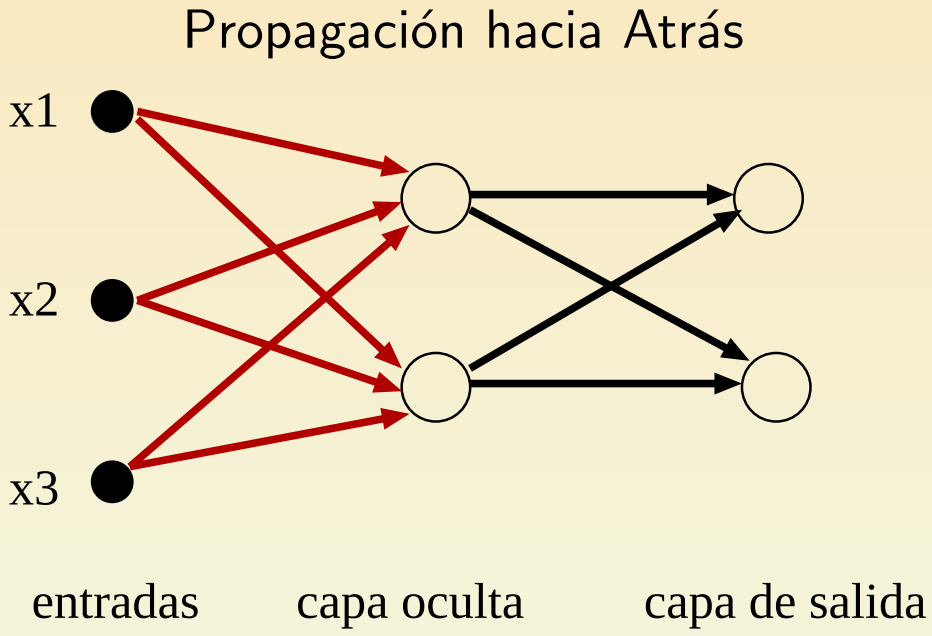
El algoritmo en acción



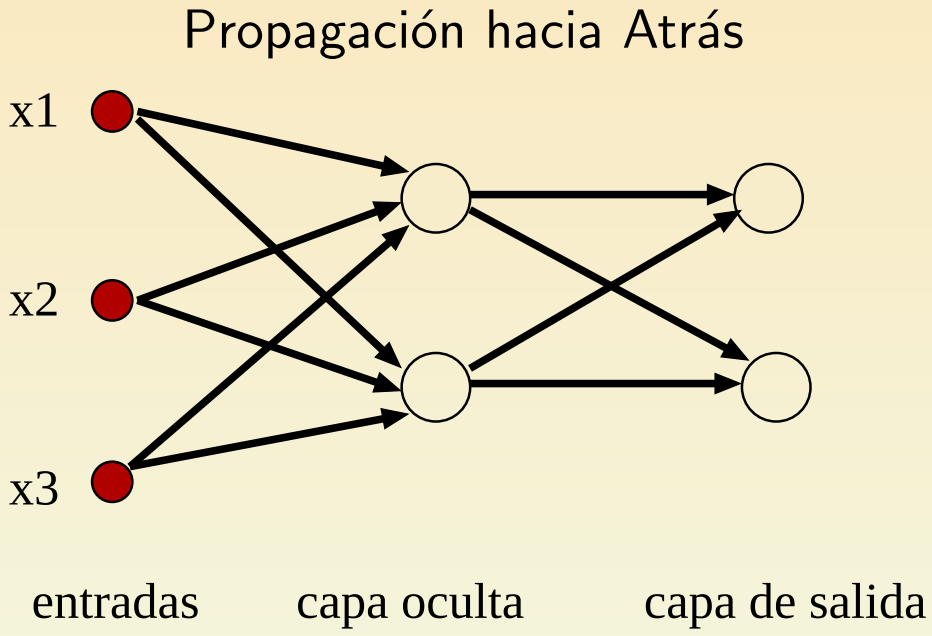
El algoritmo en acción



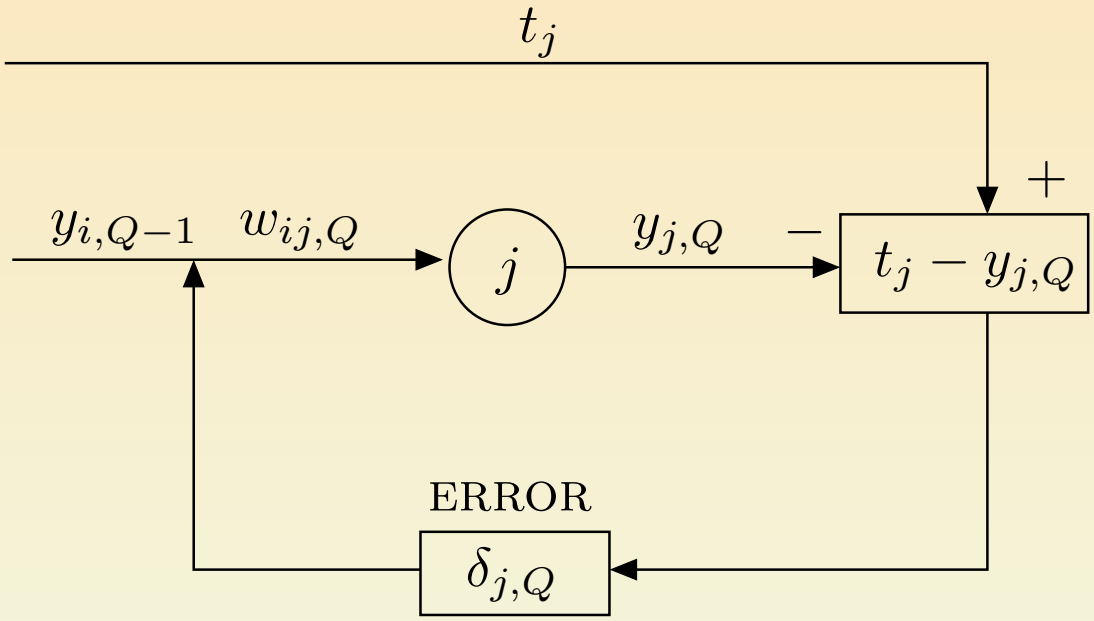
El algoritmo en acción



El algoritmo en acción

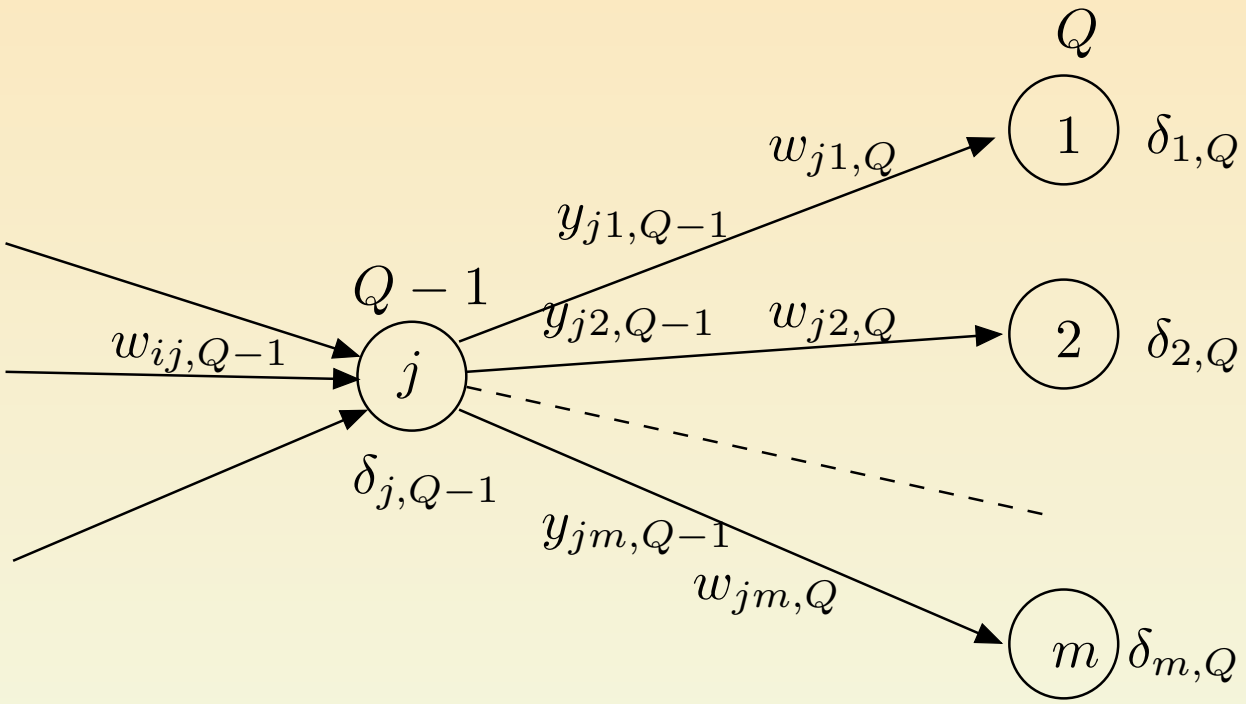


BP: Ajuste de pesos de salida



$$\delta_{j,Q} = f'_A(N_{j,Q})(t_j - y_{j,Q}) \tag{17}$$

BP: Ajuste de pesos intermedios



$$\delta_{j,Q-1} = f'_A(N_{j,Q-1}) \sum_i \delta_{i,Q} w_{ji,Q} \tag{18}$$

BP en notación matricial

Con notación vectorial, la operación del BP es más compacta.

δ_q, vector de valores delta
$\mathbf{y}_q$, salidas obtenidas
$\mathbf{w}_q$, pesos

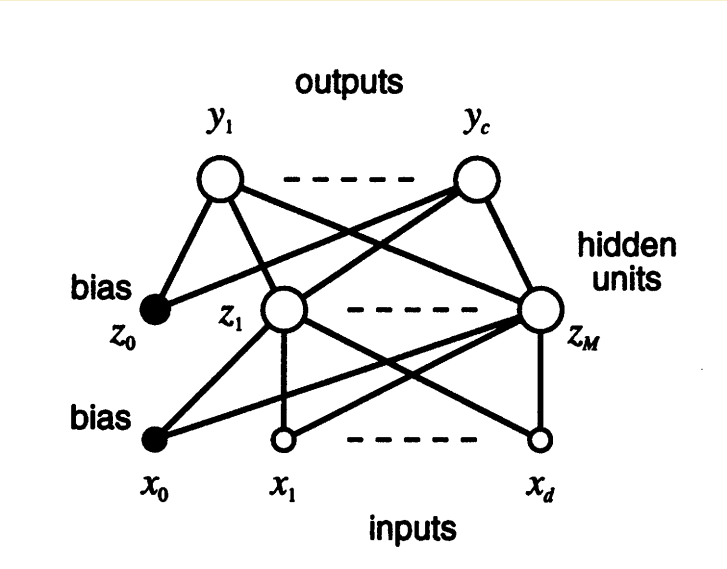
$$\delta_q = D_{q+1}\mathbf{w}_{q+1}^t \otimes [\mathbf{y}_q \otimes (1 - \mathbf{y}_q)] \tag{19}$$

$q = 1, 2, \ldots, Q$, donde 1 representa un vector con todas las componentes igual a 1 y $\otimes$ una multiplicación componente a componente.

Consideraciones

- **Ajuste de tendencias**

  Tratadas como pesos de los enlaces de unidades ficticias.



- Aplicación a **topologías más generales**

  Considerar las unidades en orden topológico.

Particularizaciones: Función logística

Su derivada es fácil de calcular:

$$f'_A(N) = f_A(N)(1 - f_A(N)) \tag{20}$$

δs para las neuronas de salida:

$$\delta_{j,Q} = (t_j - y_{j,Q})y_{j,Q}(1 - y_{j,Q}), \tag{21}$$

δs las unidades ocultas:

$$\delta_{j,p} = y_{j,p-q}(1 - y_{j,p-1}) \sum_i \delta_{i,p} w_{ji,p}. \tag{22}$$

Particularización: tangente hiperbólica

Derivada fácil de calcular:

$$f'_A(x) = 1 - f_A(x)^2 \tag{23}$$

δs para las neuronas de salida:

$$\delta_{j,Q} = (t_j - y_{j,Q})(1 - y_{j,Q}^2), \tag{24}$$

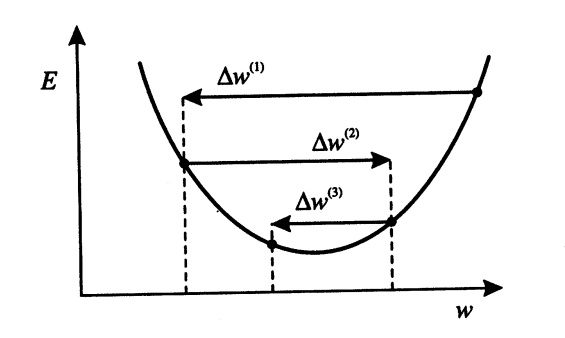
δs para las neuronas ocultas:

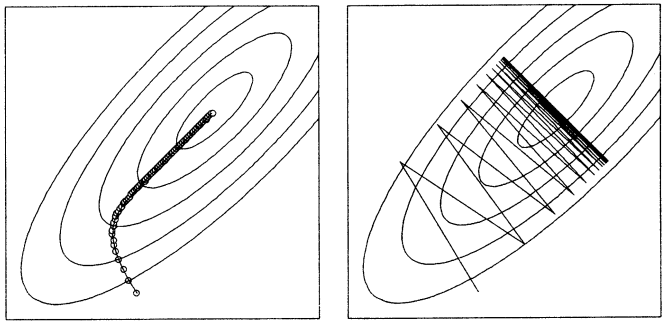
$$\delta_{j,p} = (1 - y_{j,p-1}^2) \sum_i \delta_{i,p} w_{ji,p}. \tag{25}$$

Tasa de Aprendizaje

Aproximación de la trayectoria en el espacio de pesos. Tamaño del paso regulado por η:

- η pequeña: trayectoria suave
- η grande: rapidez, pero también posibles oscilaciones





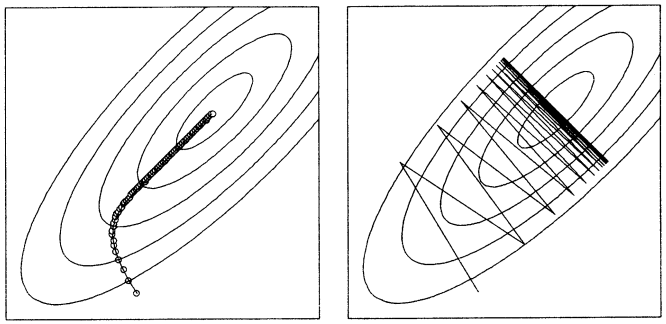
(a) small step-size (b) large step-size

Momento

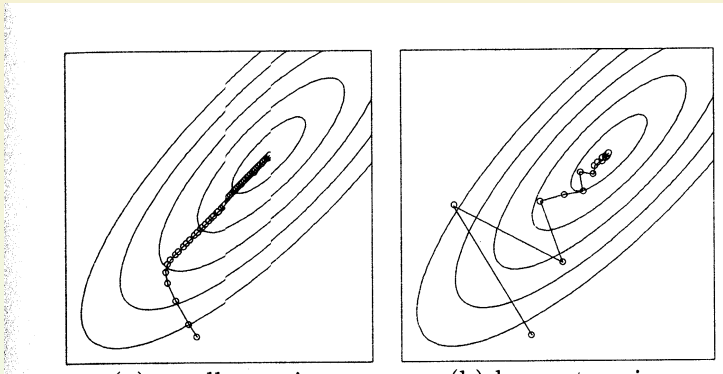
Uso del **momento**: para aumentar la velocidad y reducir la oscilación

$$\Delta w_{ij}(n) = \eta \delta_j(n) y_j(n) + \alpha \Delta w_{ij}(n-1) \tag{26}$$

α es la constante del momento, $\alpha \in [0, 1)$



(a) small step-size (b) large step-size



(a) small step-size (b) large step-size

BP en batch

No hay ajuste para cada ejemplo. Se acumulan los valores δ y se hace el ajuste cuando se han evaluado todos

Concepto de **época**

1: **repeat**
2: **for** cada par del conjunto de entrenamiento **do**
3: Introducir la entrada en la red y calcular la salida que le corresponde.
4: Calcular el error (en términos de norma cuadrática) entre la salida obtenida y la salida deseada (el vector objetivo del par de entrenamiento).
5: Calcular el δ de cada unidad y acumularlo
6: **end for**
7: Ajustar los pesos de la red de modo que se minimice el error.
8: **until** que se cumpla el criterio de parada

Condiciones de parada (I)

No se puede demostrar la convergencia del BP: Criterios heurísticos.

- **Gradiente cero**.

  Si $\mathbf{w}$ es extremo $\Longrightarrow \nabla E(\mathbf{w}) = 0$

  Parar cuando se alcance $\nabla E(\mathbf{w}) = 0$

- **Estado estacionario**. Parar cuando el cambio en la función de error E sea *suficientemente* pequeño.

- **Gasto computacional fijo**.

Condiciones de parada (II)

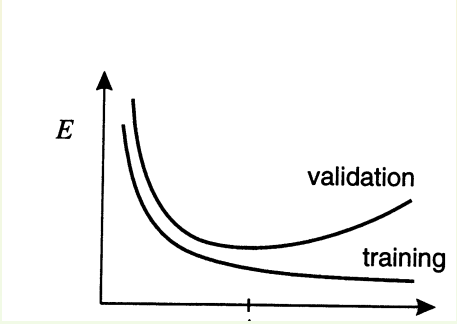
- **Parada temprana**.

  Dividir el conjunto de datos en

  – *entrenamiento*: usado para ajustar los pesos
  – *validación*: usado para valorar la capacidad de generalización

  Se mide el nivel de error en entrenamiento y en validación.

  Parar cuando empiece a crecer el error en validación.



Limitaciones del BP

- Presencia de mínimos locales
- Elección de la función de error
- Sobreajuste
- Lentitud
- Sin fundamento biológico

Retropropagación de errores

Ajuste de los pesos:

$$\Delta w_{ij,p}(n) = +\eta \delta_{j,p} y_{i,p-1} \tag{27}$$

δ para unidades de salida:

$$\delta_{j,Q} = f'_A(N_{j,Q})(t_j - y_{j,Q}) \tag{28}$$

δ para unidades ocultas:

$$\delta_{j,Q-1} = f'_A(N_{j,Q-1}) \sum_i \delta_{i,Q} w_{ji,Q} \tag{29}$$

Descenso en gradiente. Mejoras

- Adaptación de la tasa de aprendizaje

$$\eta_{\text{nuevo}} = \begin{cases} \rho\eta_{\text{anterior}} & \text{si}\Delta E < 0 \\ \sigma\eta_{\text{anterior}} & \text{si}\Delta E > 0. \end{cases}$$

$\rho > 1$; $\sigma \approx 0,5$

- Tasa de aprendizaje por cada peso

- QuickProp Aproximar la función de error por un polinomio cuadrático y emplear dos evaluaciones consecutivas de esta aproximación:

$$\Delta w_i^{(t+1)} = \frac{g_i^{(t)}}{g_i^{(t-1)} - g_i^{(t)}} \Delta w_i^{(t)}, \qquad g_i^{(t)} = \frac{\partial E}{\partial w_i^{(t)}}$$

Métodos más rápidos

- Gradiente Conjugado
- Método de Newton
- Método de Levenberg-Marquardt

Técnicas Heurísticas

- Enfriamiento Simulado
- Algoritmos Genéticos
- Programación Genética
- Controladores Difusos

Ingeniería de RNA

1. Seleccionar el conjunto de datos

   Entradas, salidas, tipo

2. Establecer el modelo

   Arquitectura, parámetros de aprendizaje

3. Entrenar la red con el conjunto de datos

4. Validar la red

5. Aplicarla

Preprocesamiento y Extracción de Características

- Necesidad de transformar los datos
- Transformaciones de entrada y salida
- Reducción de dimensionalidad
- Valores desconocidos o erróneos
- Extracción de características

Preprocesamiento

- Normalización y codificación de Entradas
- Escalado lineal; transformaciones no lineales
- Datos discretos:
  - ordinales
  - categóricos
- Datos desconocidos:
  - Reemplazar por media
  - Reemplazar por valores de regresión
  - Principio de máxima probabilidad

Selección de Características

- Comparar subconjuntos de características
  - Depende del problema
- Búsqueda:
  - Exhaustiva
  - Secuencial
  - Branch and Bound

Extracción de Características

- Análisis de Componentes Principales (PCA)
- Conocimiento a Priori

Funciones de Error

El aprendizaje de las RNAs es un problema de optimización: minimizar el error cometido sobre un conjunto de ejemplos.

- Suma de los cuadrados:

$$E = \frac{1}{2}\sum_{i=1}^{P} ||y_i(x;w) - t_i||^2$$

  Se emplea por simplicidad analítica

  Se deriva del principio de máxima probabilidad, suponiendo que la distribución de los datos objetivo es normal.

  Raíz media al cuadrado:

$$E = \frac{\sum_n ||y - t||^2}{\sum ||t - \bar{t}||^2}$$

Funciones de Error (II)

- Error de Minkowski

  Con ejemplos muy atípicos, su aportación al error puede ser demasiado determinante

$$E = \sum ||y - t||^R$$

  $R < 2$ atenúa este efecto.

Funciones de Error para Clasificación

Depende la codificación. Habitual: 1 de c.

- Suma de cuadrados
- Entropía cruzada:

$$E = -\sum t \ln y$$

Algoritmos para Optimización de Parámetros

- Superficies de Error
- Algoritmos Iterativos

$$w^{(t+1)} = w^{(t)} + \Delta w^{(t)}$$

- Orden de convergencia:

$$\varepsilon^{(t+1)} \propto \left(\varepsilon^{(t)}\right)^{L}$$

- Alto grado de simetría en el espacio de pesos

Aprendizaje y Generalización

- Objetivo del aprendizaje: construir un modelo estadístico del proceso que genera los datos
- Necesidad de controlar la complejidad del modelo
- Balance entre tendencia y varianza
- Regularización
- Estabilización estructural
- Entrenamiento con ruido

Tendencia y varianza

El error de generalización se puede descomponer en dos partes:

- tendencia: La diferencia en media de la función que calcula la red y la que pretende aproximar
- varianza: Mide la sensibilidad respecto al conjunto de datos empleado

Existe una relación de conflicto natural entre tendencia y varianza. Es necesario encontrar un punto de equilibrio

Minimizando la tendencia y la varianza:

- Usar más datos
- Conocimiento a priori

Regularización

Añadir un término a la función de error que regule la complejidad del modelo:

$$\tilde{E} = E + \nu\Omega$$

Ω: Penalización para modelos complejos
ν: regula el grado de aplicación de la penialización

Técnicas de regularización:

- Reducción de pesos:

$$\Omega = \frac{1}{2}\sum_i w_i^2$$

- Parada temprana

- Suavizamiento guiado por curvatura

Entrenamiento con ruido

Añadir ruido aleatorio (distribuido según una normal) a los datos de entrenamiento. Esto evita el sobreajuste.

Estabilización estructural

- Comparar redes de distinta complejidad
- Poda
- Crecimiento
- Combinar las salidas de distintas redes

Complejidad del Aprendizaje

El problema general de aprendizaje de una RNA consiste en encontrar los elementos desconocidos de una RNA dada (pesos, funciones de activación, conexiones).

Este problema es **NP-completo**

Aplicaciones

- Codificación/encriptación de información
- Lectura de textos
- Reconocimiento de escritura
- Problemas de decisión
- Predicción de series temporales: precios, bolsa, cambio de divisas
- Restauración de imágenes
- Diagnóstico médico

- Optimización combinatoria
- Control adaptativo
- Identificación de sistemas desconocidos

Propiedades

- Capacidad de aprendizaje y adaptación
- Capacidad de generalización
- Capacidad de almancenamiento y memoria
- "Aproximación Universal"
- Inmunidad frente al ruido
- Modelos de representación subsimbólica del conocimiento
- Naturaleza masivamente paralela: rapidez
- Modelos de cálculo

Inconvenientes

- Carácter de "Caja Negra"
- Diseño: selección de modelo y topología
- Selección del conjunto de ejemplos
- Problemas de entrenamiento

Bibliografía

- C. Bishop. *Neural Networks for Pattern Recognition*. Springer-Verlag, 1995.
- S. Haykin. *Neural Networks*. Prentice-Hall, 1999.
- J.R. Jang, C.-T. Sun y E. Mizutani. *Neuro-Fuzzy and Soft Computing*. Prentice-Hall, 1997.
- D. Nauck, F. Klawonn and R. Kruse. *Foundations of Neuro-Fuzzy*. Chichester: Wiley, 1997
- B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.
- R. D. Reed y R. J. M. II. *Neural Smithing. Supervised Learning in Feedforward Artificial Neural Networks*. The MIT Press, 1999.
- R. Rojas. *Neural Networks. A Systematic Introduction*. Springer-Verlag, 1995.

Revistas

- Neural Networks
- IEEE Trans. on Neural Networks
- Neurocomputing
- Neural Comptutation

Recursos en Internet

- http://ftp.sas.com/pub/neural/FAQ.html
- http://www.emsl.pnl.gov:2080/proj/neuron/neural/what.html
- http://www.cs.stir.ac.uk/ lss/NNIntro/InvSlides.html
- http://umtii.fme.vutbr.cz/MECH/nn.html
- news://comp.ai.neural-nets
- news://comp.ai.fuzzy
- news://es.comp.ai.neural