

# IA y Robótica

## Redes Neuronales

Facultad de Ingeniería  
Instituto de Computación

# Presentación

- Las redes neuronales artificiales constituyen una de las primeras ramas de la IA, tanto por su temprana aparición como sus múltiples aplicaciones industriales.
- Muchos investigadores las ven más como una cuestión de arte que de ciencia.

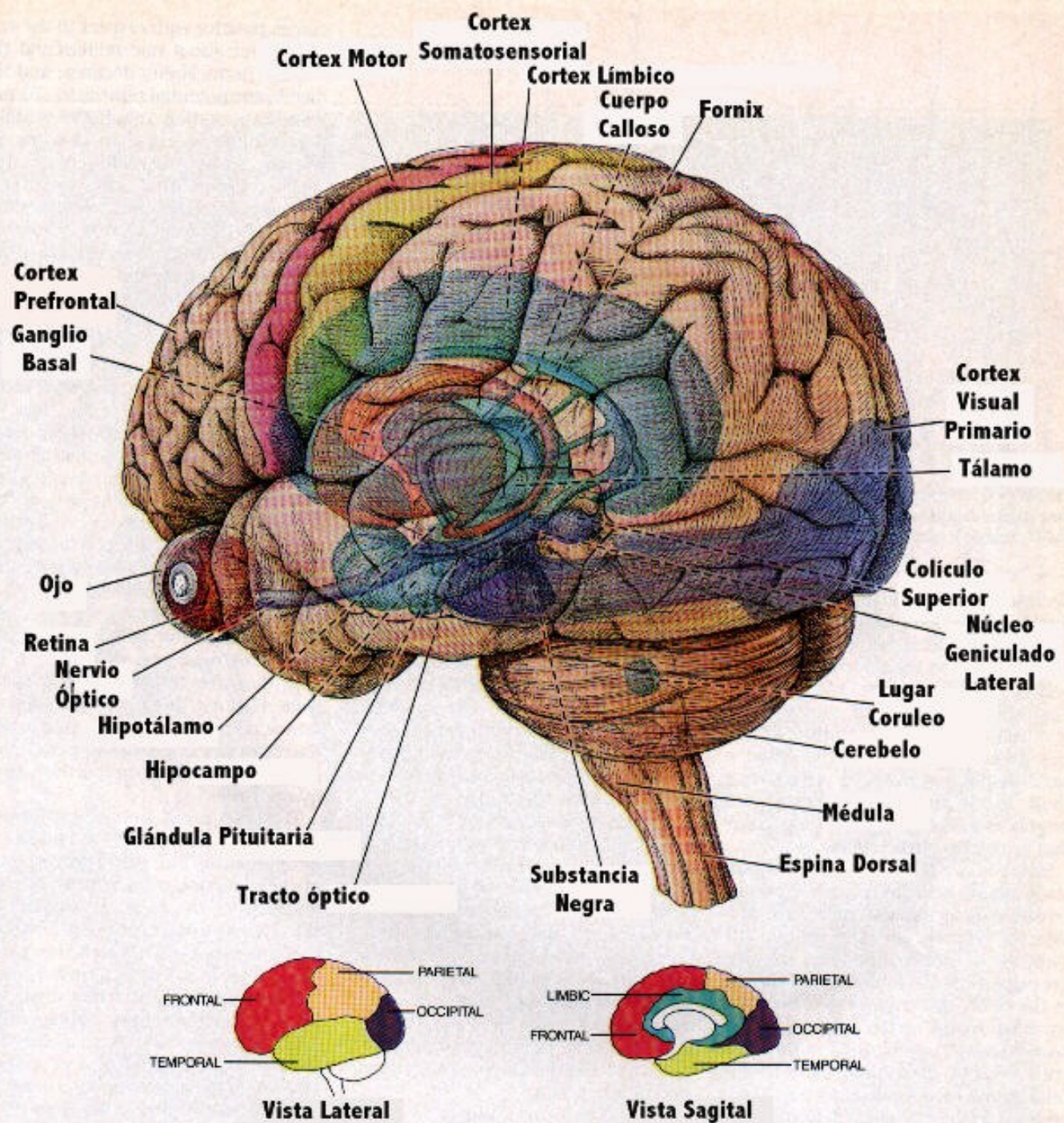
# Organización

- Cerebro y neuronas
  - Biológicas
  - Artificiales.
- Redes Neuronales Artificiales
  - Perceptron
  - Perceptron Multi Capa
  - Mapas Autoorganizados

# Introducción

- Las ANN surgen como un modelo inspirado en estructuras biológicas, en este caso el cerebro.
- Redes neuronales como grafos ponderados.
- Elaborar sistemas que aprendan de ejemplos.
- Etapas:
  - Diseño de topología.
  - Implementación.
  - Proceso de entrenamiento.
  - Validación.

# Cerebro (1/3)



# Cerebro (2/3)

- El tamaño y la complejidad de éste constituyen una diferencia crítica entre los humanos y los animales inferiores.
- Las diferentes regiones de la corteza están especializadas para tareas concretas como ser hablar, entendimiento del habla, analizar información visual, organización de actividades motoras y otros aspectos del comportamiento inteligente.

# Cerebro (3/3)

- El cerebro es una extremadamente compleja computadora paralela no lineal. Tiene la capacidad de realizar ciertas computaciones (reconocimiento de patrones, percepción y control motor) mejor y varias veces más rápido que una computadora.
- La experiencia se forma con los años, en el caso del hombre durante los dos primeros años de vida tienen lugar los cambios más drásticos. Durante esta etapa de desarrollo se forman cerca de 1 millón de sinapsis por segundo.
- En el cerebro adulto se pueden crear nuevas sinapsis y modificar nuevas sinapsis (plasticidad).



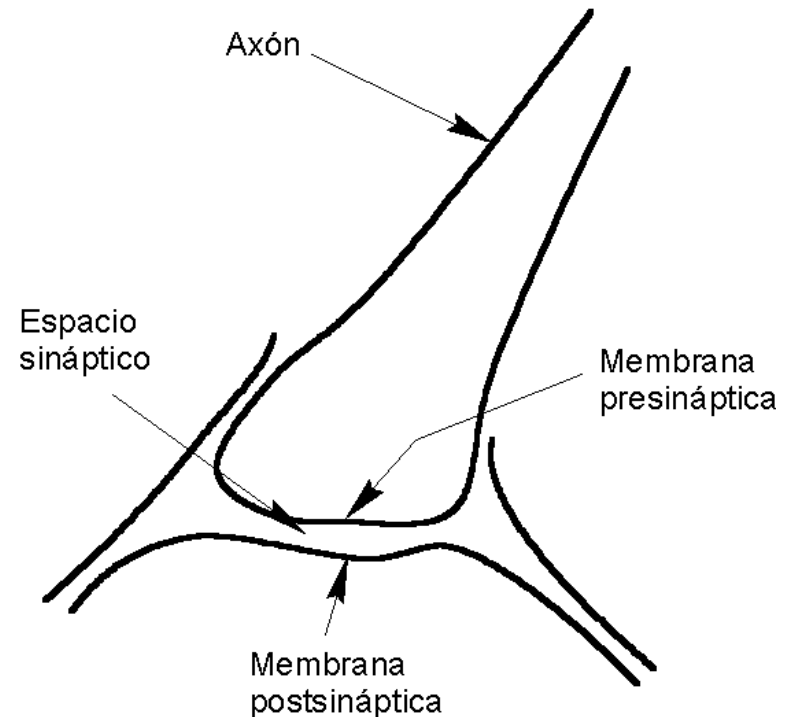
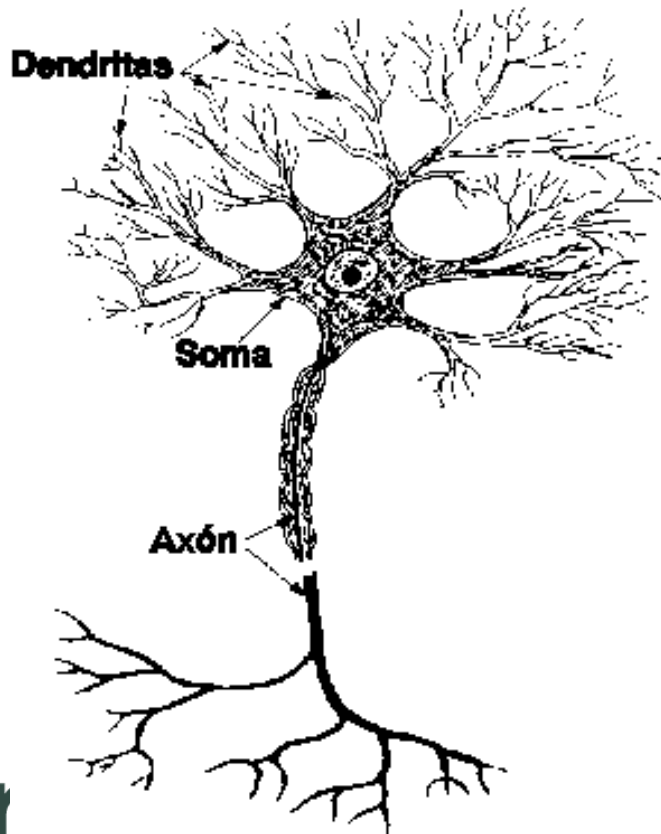
# Características del cerebro

- Robusto y Tolerante.
- Flexible (aprende).
- Soporta información con ruido.
- Es paralelo.
- Es pequeño, compacto y consume poca energía.



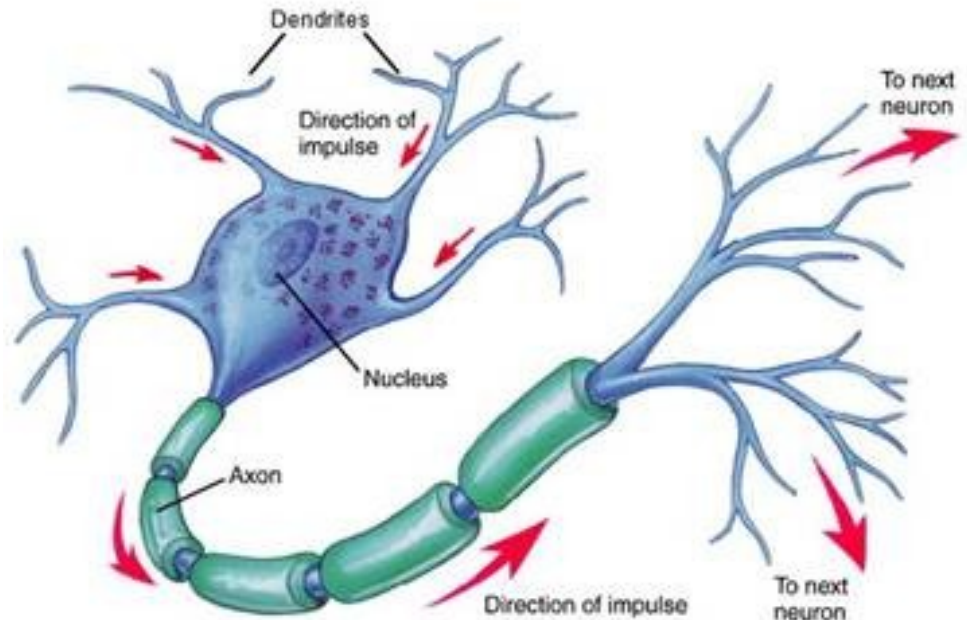
# Neurona <sup>(1/5)</sup>

- El pionero en el entendimiento del cerebro fue Ramón y Cajal (1911), que introduce el concepto de neurona.



# Neurona (2/5)

- Típicamente, las neuronas son 5 o 6 veces más lentas que las compuestas lógicas de silicio.
- Se estima que la cantidad de neuronas es  $10^{11}$  en la corteza cerebral y existen 60 trillones de sinapsis.
- El axón es más largo y definido con algunas ramificaciones. Mientras que las dendritas tienen una superficie irregular con más ramificaciones.

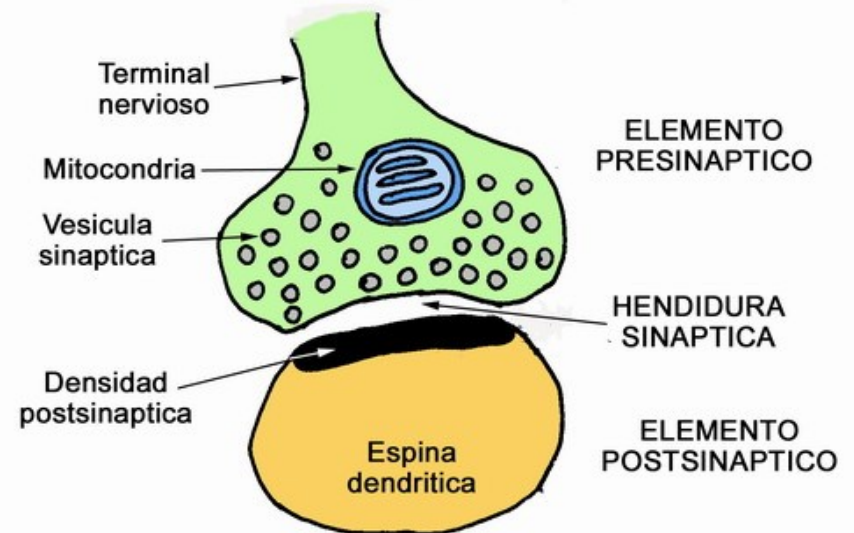
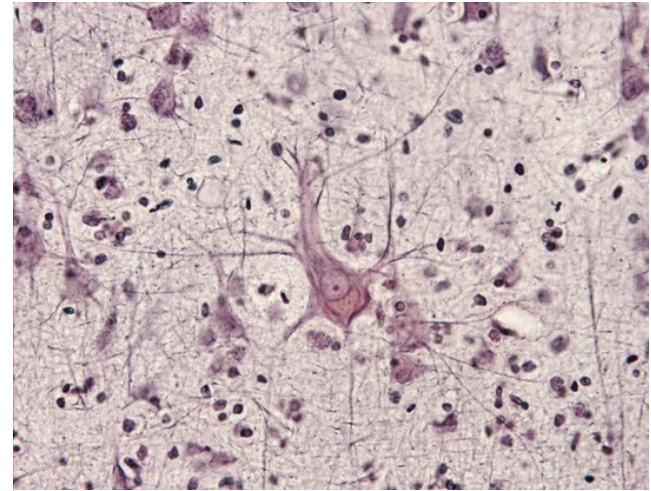


# Neurona (3/5)

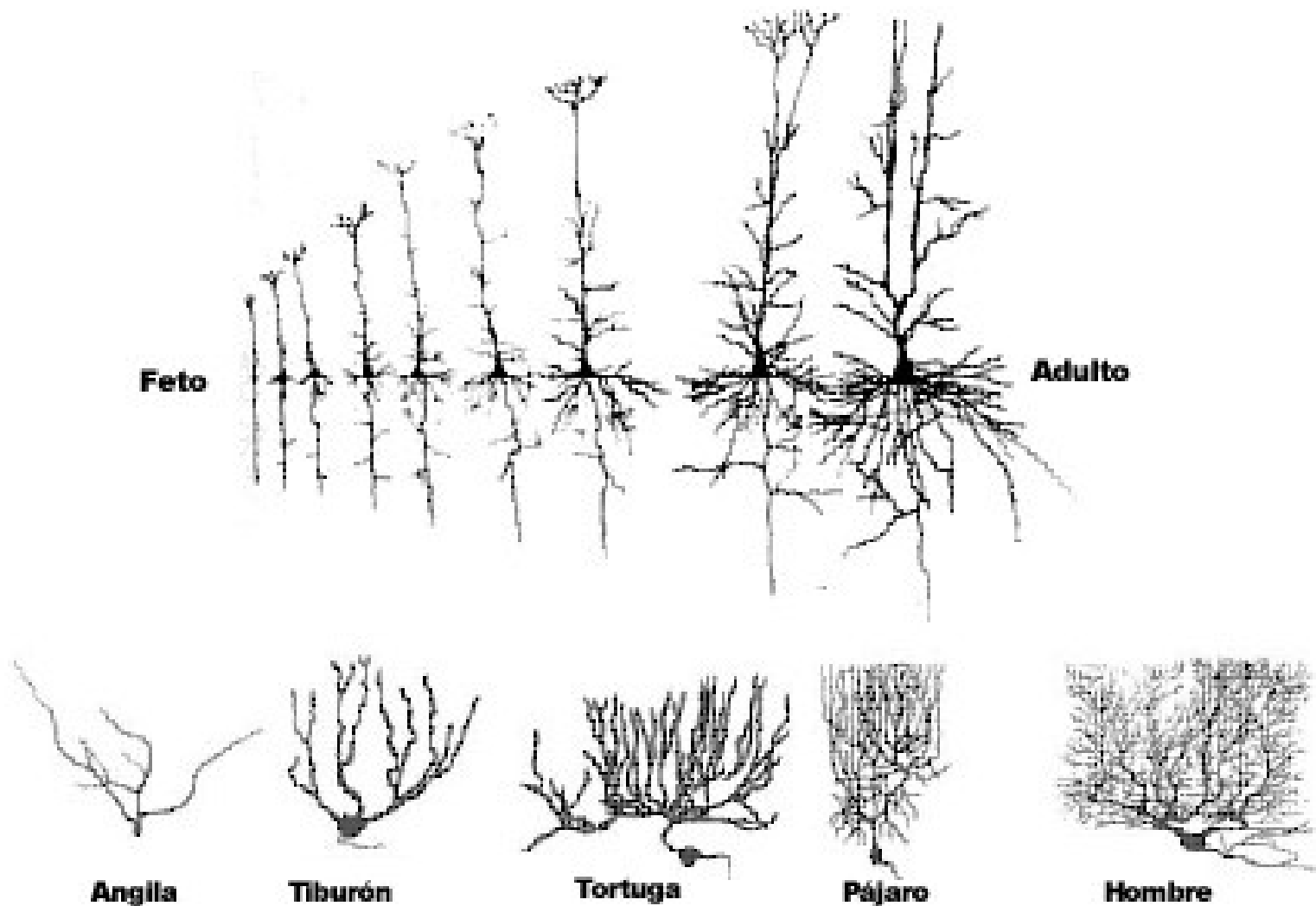
- Un proceso sináptico libera una sustancia química que se difunde a través de la unión sináptica entre neuronas y que luego actuará en un proceso postsináptico.
- Una sinapsis convierte una señal eléctrica en una señal química y luego en una señal eléctrica postsináptica.
- Las neuronas tienen varias formas y tamaños en las distintas partes del cerebro.

# Neurona (4/5)

- Las neuronas piramidales son las más comunes en la corteza cerebral. Como la mayoría de las neuronas, recibe la mayoría de sus entradas a través de las espinas dendríticas. Una célula piramidal puede recibir 10000 o más contactos sinápticos y puede proyectarse sobre miles de células destino.



# Neurona (5/5)

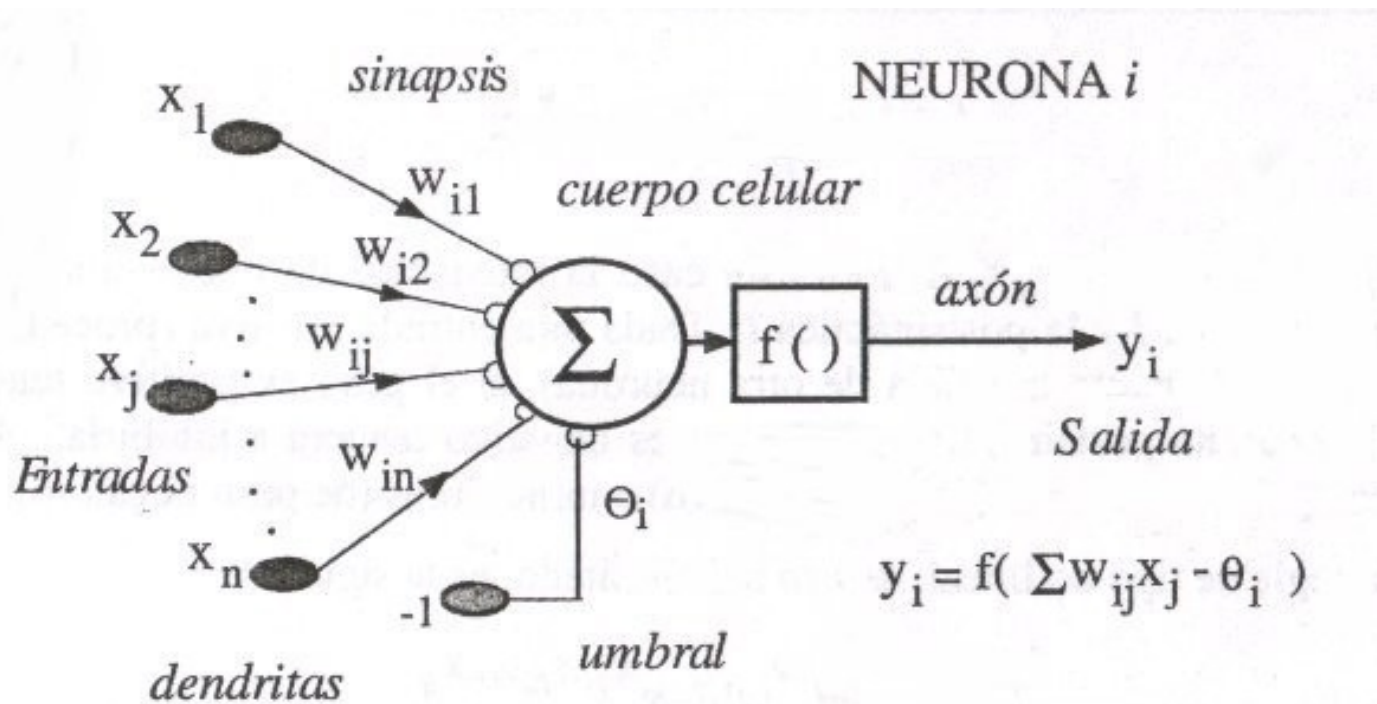


# Redes Neuronales

- Generalidades.
- Modelo de neurona.
- Funciones de activación.
- Topologías.
- Sistemas biológicos vs artificiales.

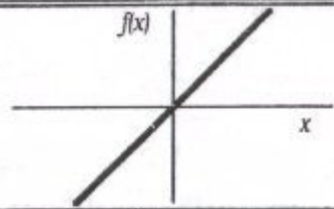
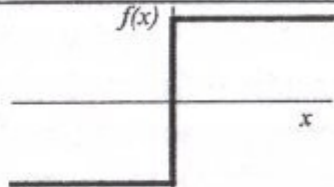
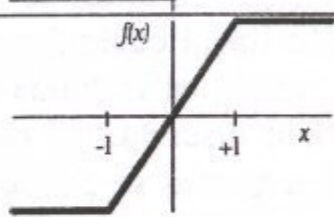
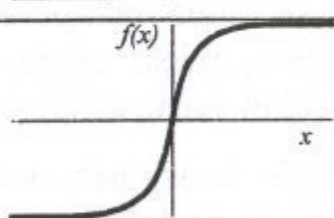
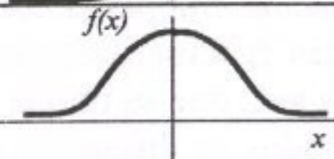
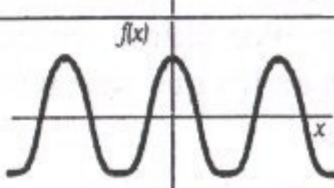
# Modelos de neurona

- Una neurona es una unidad de procesamiento de la información que es fundamental para el funcionamiento de la red neuronal.



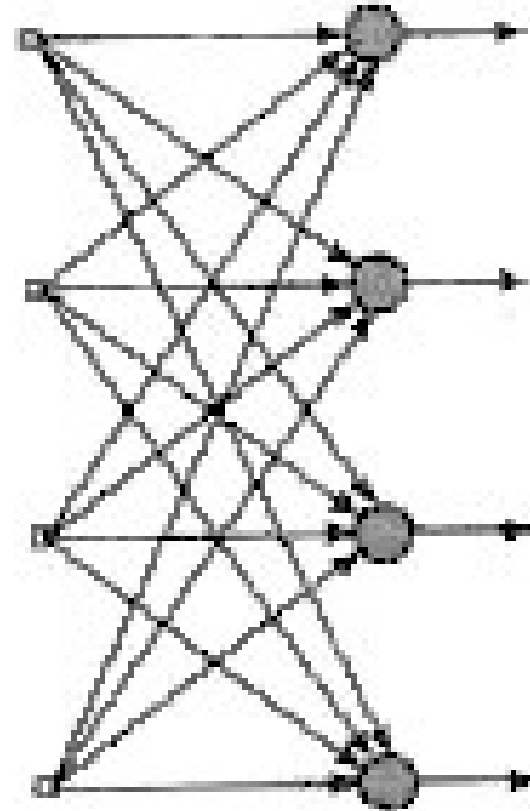


# Funciones de activación

	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = \text{sign}(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & \text{si } x < -1 \\ x, & \text{si } -1 \leq x \leq 1 \\ +1, & \text{si } x > 1 \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = \text{tgh}(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0, +1]$	
Sinusoidal	$y = A \text{sen}(\omega x + \varphi)$	$[-1, +1]$	

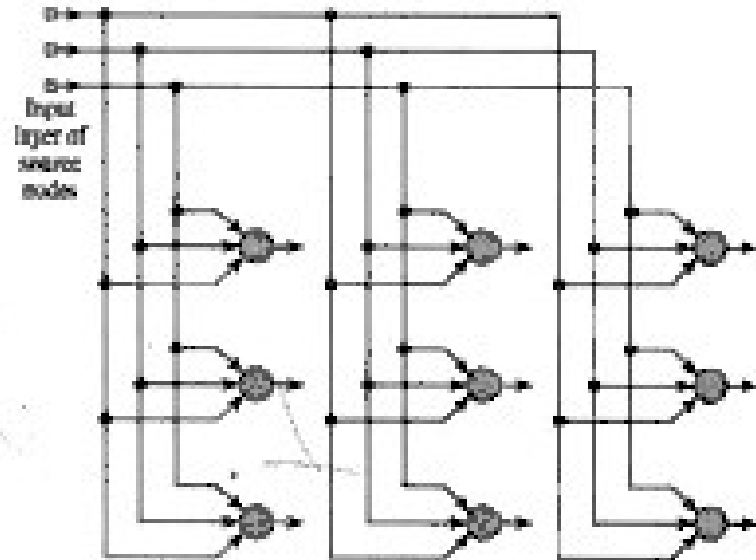
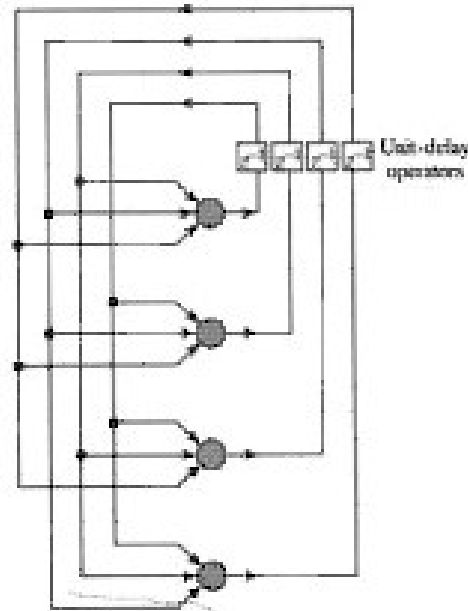
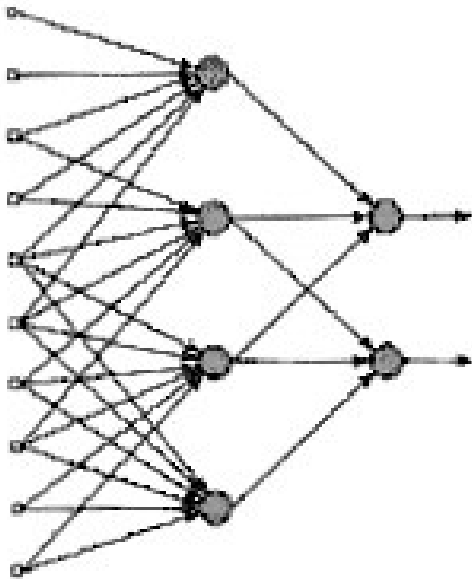
# Arquitecturas (1/2)

- Grafo dirigido.
- Capa única con alimentación hacia delante (feed forward)
  - Capa de entrada
  - Capa de salida
- Buenas para problemas de clasificación y filtrado de ruido.
- Rápida convergencia.
- Algoritmia simple.



# Arquitecturas (2/2)

- Multicapa con alimentación hacia delante
- Redes recurrentes
- Estructura de grilla



# Aprendizaje

- El proceso para realizar el aprendizaje es conocido como algoritmo de entrenamiento, su función es modificar los pesos sinápticos de modo ordenado para obtener el objetivo deseado.
- Es posible que la red neuronal modifique su topología, lo cual está motivado por el hecho que las neuronas en el cerebro humano pueden morir y que nuevas sinapsis puedan aparecer.

# Diferencias entre las redes neuronales biológicas y artificiales

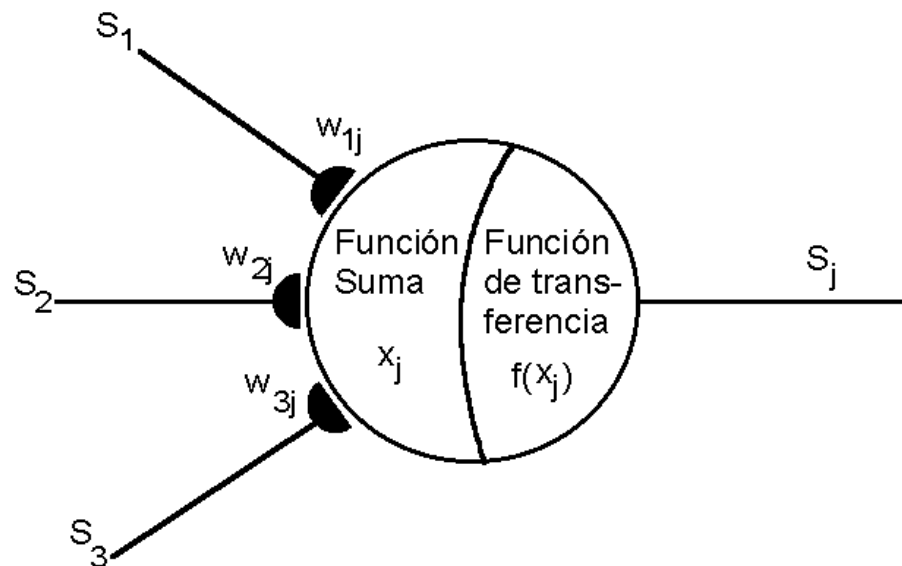
Red Biológica	Red Artificial
Sinapsis compleja	Sinapsis simple
Aprende en una pasada	Convergencia lenta
10 billones de neuronas	cientos o miles de neuronas
10000 interconexiones por neurona	entre 10 y 10000 interconexiones
actualización asíncrona o continua	usualmente sincrónica

# Perceptron Simple

- Introducción
- Aprendizaje
- Problemas linealmente separables
- Adaline vs Perceptron

# Introducción

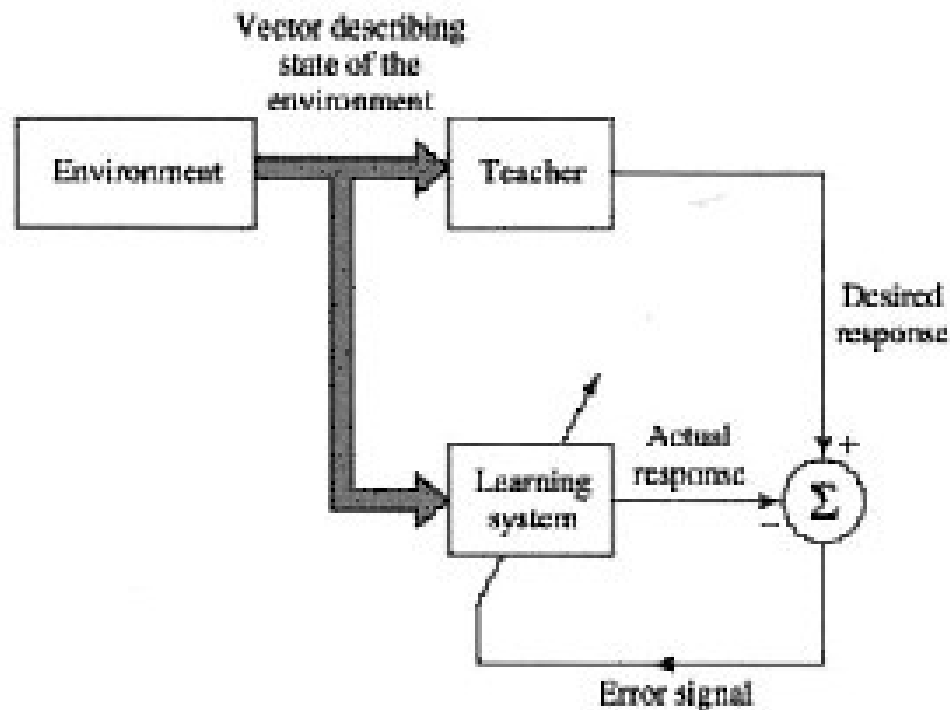
- El perceptron es la forma más simple de red neuronal utilizado para la clasificación de patrones linealmente separables.
- La operación básica del perceptron de Rosenblatt (1960) es la neurona de McCulloch-Pitts (1943).





# Aprendizaje supervisado

- La arquitectura perceptron aprende a clasificar patrones a través de aprendizaje supervisado.



# Regla de aprendizaje

- El objetivo del entrenamiento es obtener un único juego de pesos que permitan realizar satisfactoriamente el mapeo entrada-salida.
- Una regla de aprendizaje simple:

$$w_{ij_{new}} = w_{ij_{old}} + c(t_j - x_j)a_i$$

$t_j$  = valor deseado de salida para la unidad j.

$x_j$  = valor obtenido de salida para la unidad j.

$c$  = constante de aprendizaje.

$a_i$  = valor de la entrada de la unidad i.

# Algoritmo de aprendizaje

- Se le presenta a la red cada uno de los elementos del conjunto y luego de ello se reajustan los pesos. Luego de presentar todo el conjunto, el conjunto se vuelve a presentar y así varias veces.
- El rendimiento de la red es medido por mínimos cuadrados (root-mean-square)

$$RMS = \sqrt{\frac{\sum_p \sum_j (t_{jp} - x_{jp})^2}{n_p n_0}}$$

$t_{jp}$  = valor deseado de salida para la unidad  $j$  luego de presentado el caso  $p$ .

$x_{jp}$  = valor obtenido de salida para la unidad  $j$  luego de presentado el caso  $p$ .

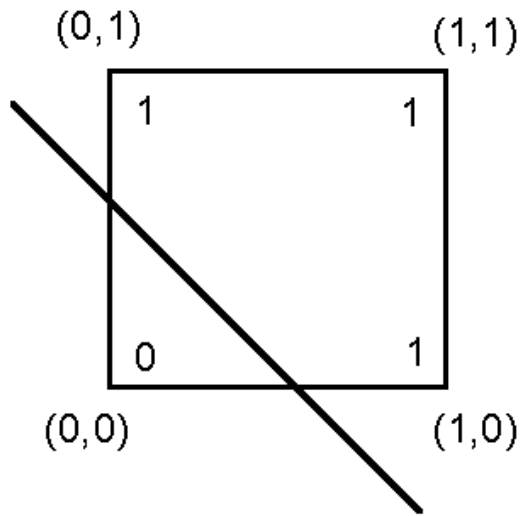
$n_p$  = cantidad de patrones en el conjunto de entrenamiento.

$n_0$  = cantidad de unidades en la capa de salida.

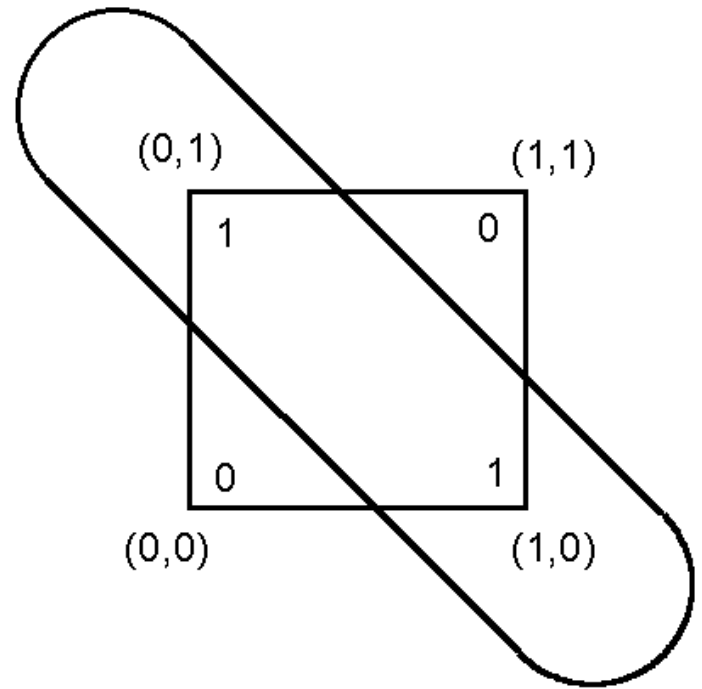
# Qué es capaz de aprender un perceptron

- Aun cuando el perceptron es capaz de realizar una gran variedad de tareas de clasificación de patrones con éxito, el aprendizaje no siempre ocurre o el tiempo de aprendizaje puede ser extremadamente largo. Las redes perceptron pueden clasificar patrones linealmente separables.
- No puede resolver el XOR.

# Linealmente separables



Función OR



Función XOR

# Adaline vs Perceptron

- Las unidades de procesamiento tienen entradas que pueden ser booleanas en como ocurre en la ADALINE (ADaptive LINear Element) o reales como en el perceptron.
- Las neuronas ADALINE usan como función de transferencia purelin o escalón.
- Las ADALINE sufren de separabilidad lineal.

# Perceptron Milticapa

- Introducción
- Back Propagation
- Criterios de parada
- Ventajas
- Limitaciones



# Introducción

- El perceptron multicapa (PMC) es una red multicapa con alimentación hacia adelante. Típicamente formado por una capa de entrada, una o más capas ocultas y una capa de salida.
- MPL han sido aplicado exitosamente en gran cantidad de problemas diversos entrenándola con el conocido algoritmo back-error-propagation.

# Back Propagation (1/3)

1. Inicialización. Comenzar con una configuración razonable de red y setear todos los pesos y umbrales a niveles pequeños uniformemente distribuidos.
2. Presentar a la red los ejemplos de entrenamiento. Realizar los pasos 3 Y 4.

# Back Propagation (2/3)

1. Computación hacia adelante. Sea el elemento del conjunto de entrenamiento  $[x(n), d(n)]$ .

$$v_j^{(l)}(n) = \sum_{i=0}^p w_{ji}^{(l)}(n) y_i^{(l-1)}(n), \text{ donde el superíndice indica la capa.}$$

$$y_j^{(l)}(n) = \frac{1}{1 + \exp(-v_j^{(l)}(n))}, \quad 1 < l < L$$

$$y_j^{(0)}(n) = x_j(n)$$

$$y_j^{(L)}(n) = o_j(n)$$

$$e_j(n) = d_j(n) - o_j(n)$$

# Back Propagation (3/3)

- Computación hacia atrás. Computar  $\delta$  (gradiente local) de la red de forma iterativa hacia atrás capa por capa.

$\delta_j^{(L)}(n) = e_j^{(L)}(n) o_j(n)(1 - o_j(n))$ , para neuronas en la capa de salida.

$\delta_j^{(l)}(n) = y_j^{(l)}(n)(1 - y_j^{(l)}(n)) \sum_k \delta_{kj}^{(l+1)}(n) w_{kj}^{(l+1)}(n)$ , para neuronas en las capas ocultas.

Ajustamos los pesos según la regla delta generalizada:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha(w_{ji}^{(l)}(n) - w_{ji}^{(l)}(n-1)) + \eta(\delta_j^{(l)}(n)y_i^{(l-1)}(n))$$

# Momento

- Para  $\eta$  chico el aprendizaje es lento y para  $\eta$  grande es rápido pero inestable. Para incrementar el  $\eta$  evitando la inestabilidad Rumelhart, 1986, propone modificar la regla delta incluyendo un termino de momento (Regla delta generalizada).

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \cdot \delta_j(n) \cdot y_i(n)$$

# Criterio de parada

- La idea del entrenamiento es reducir el error en sucesivas iteraciones.
- Debe existir un método de parada y aplicarse en el momento adecuado.
- La red puede caer en over-training.
- Método:
  - Error absoluto.
  - Error peor caso.
  - Cantidad de aciertos.
  - Cantidad de épocas.
  - Validación cruzada.
  - Desviación estándar.

# Validación cruzada

- Trata de medir la capacidad predictiva de la red.
- Se divide al juego de datos en dos conjuntos:
  - Conjunto de entrenamiento (70-80%).
  - Conjunto de validación(30-20%).
- Es posible utilizarlo on-line u off-line.



# Inicialización

- Una mala elección de los pesos sinápticos puede llevar a un fenómeno conocido como saturación prematura.
- Este fenómeno se refiere a la situación en la cual la suma instantánea de los errores cuadráticos permanece constante por algunos periodos de tiempo en la etapa de entrenamiento.
- Dicho fenómeno puede considerarse un mínimo local.
- Se mostró que al elegir valores iniciales pequeños aleatorios uniformemente distribuidos se evita la saturación.

# Modos de entrenamiento

- La presentación del conjunto de entrenamiento completa a la red se denomina época.
- Es sano presentar los elementos del conjunto en orden aleatorio.
- Formas de realizar el aprendizaje:
  - Modo patrón: los pesos se ajusta toda vez que se presenta un nuevo ejemplo de entrenamiento.
  - Modo batch: los pesos se actualizan al final de una época.

# Generalización

- Decimos que una red generaliza bien cuando la relación entrada-salida computada por ella es correcta (o cerca de serlo) para patrones entrada-salida nunca antes utilizados. La generalización está influenciada por los siguientes factores:
  - Tamaño y eficiencia del conjunto de entrenamiento.
  - Arquitectura de la red.
  - Complejidad del problema a resolver.
- Las unidades ocultas juegan un papel muy importante en la generalización del MLP con el algoritmo back-prop en el que actúan como detectores de característica.

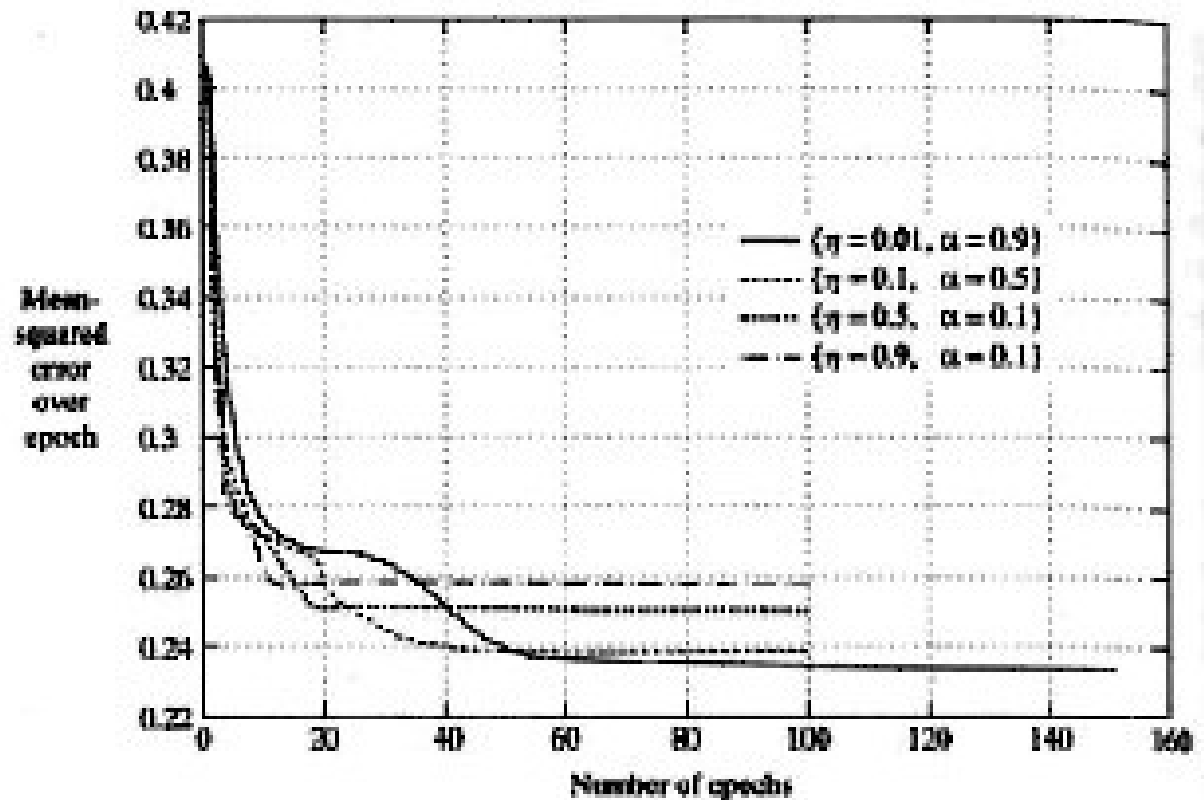
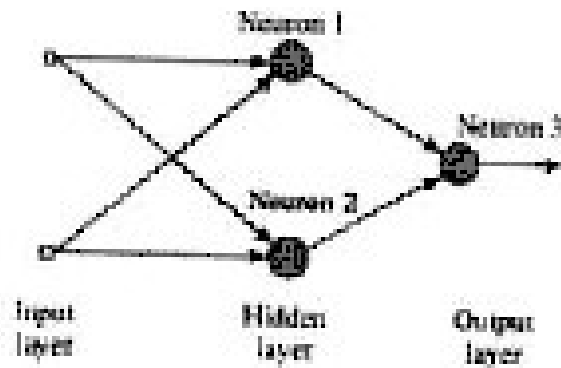
# Limitaciones

- Se requieren miles o cientos de iteraciones para problemas simples, por esto, el tiempo de convergencia para completar el entrenamiento es grande, pudiendo llegar a varios días.
- No fue diseñado para modelar los sistemas biológicos y no incluye estructuras biológicas encontradas en la sinapsis y en la interconexión de nervios. El MLP ignora la comunicación global (por ejemplo hormonal) existente en el sistema nervioso.
- Mínimos locales y oscilaciones.

# Ventajas

- Capacidad general de aprender una gran variedad de mapeos de patrones.
- Amplio espectro de aplicaciones, por su flexible arquitectura (número de capas, interconexiones, número de unidades, constante de aprendizaje y representación de los datos).
- Sólida base matemática.
- Muy estudiado.

# Experimentos con XOR



# Mapas Autoorganizados (SOM)

- Introducción
- Topología
- Algoritmo de adaptación

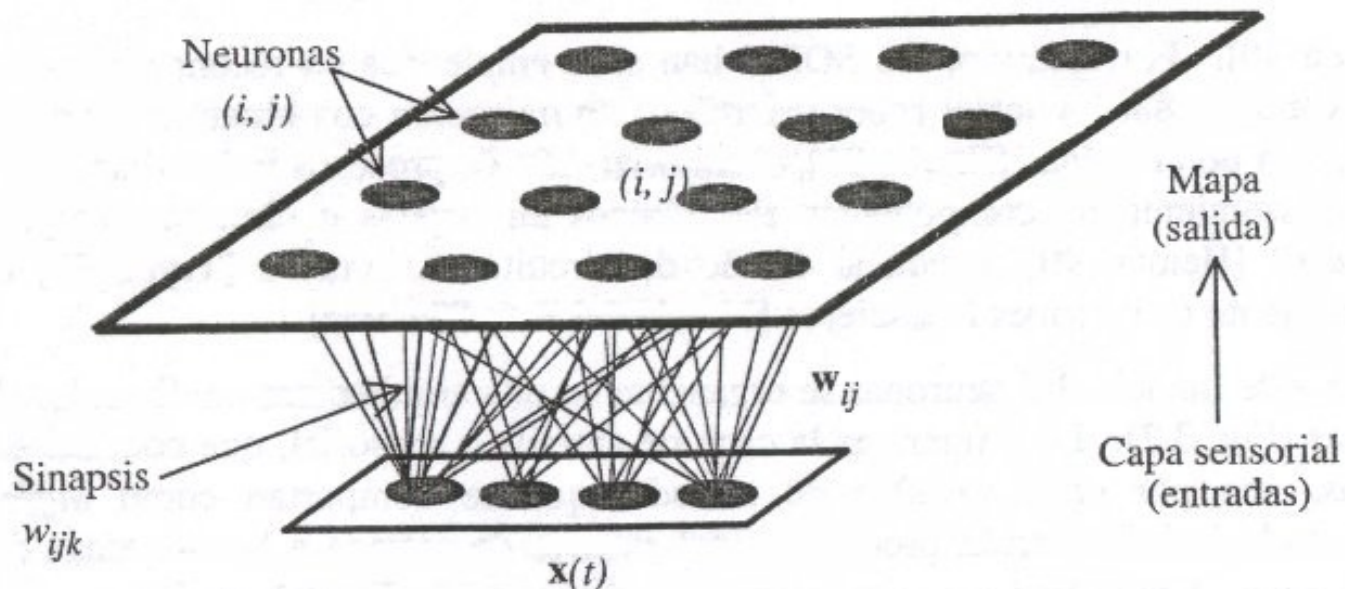
# Introducción

- Un mapa característico auto-organizado de Kohonen es una red de dos capas que puede organizar un mapa topológico a partir de una iniciación aleatoria.
- El mapa resultante muestra la relación natural entre los patrones entregados a la red.
- La red combina una capa de neuronas de entrada con una capa competitiva de neuronas, y es entrenada mediante aprendizaje no supervisado.



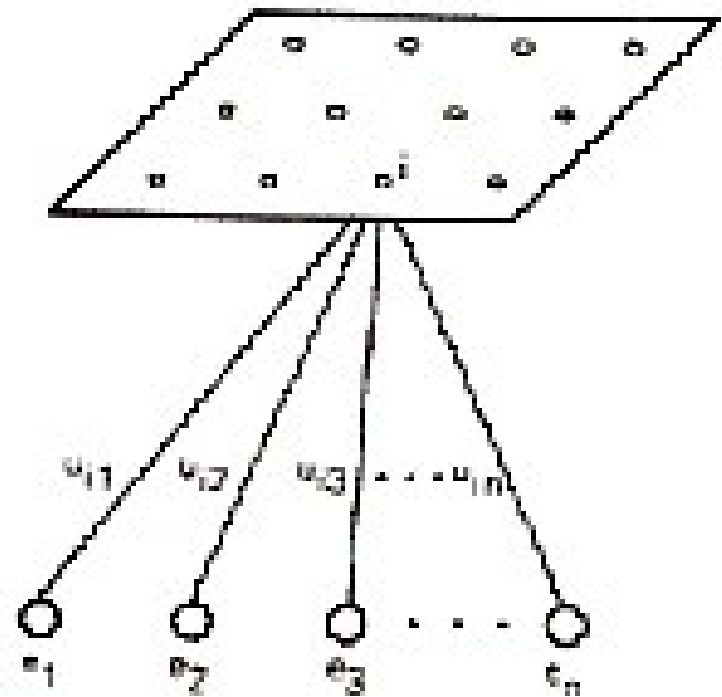
# Topología

- Es una red con dos capas. La primera capa es la de entrada. Típicamente la segunda capa está organizada como una cuadrícula 2D.
- Ambas capas están completamente interconectadas y dichas conexiones van de la primera capa a la segunda.



# Capa competitiva

- Cuando se presenta un patrón a la capa de entrada y todas las unidades en ella propagan el patrón hacia la capa competitiva.
- Las neuronas en la segunda capa compiten para encontrar a la única neurona ganadora.



# Inicialización

- Cada interconexión tiene asociado un peso inicialmente elegido al azar y ajustado durante la etapa de entrenamiento de la red.

# Algoritmo de adaptación (1/5)

- Un patrón de entrada a la red se denota como:

$$E = [e_1, e_2, \dots, e_n]$$

- Los pesos están dados por:

$$U_i = [u_{i1}, u_{i2}, \dots, u_{in}]$$

- El primer paso es computar valor de acierto para cada neurona en la capa competitiva. Este valor muestra cuan lejos están los pesos del patrón de entrada. El valor de acierto para la neurona  $i$  es:

$$\|E - U_i\| = \sqrt{\sum_j (e_j - u_{ij})^2}$$

# Algoritmo de adaptación (2/5)

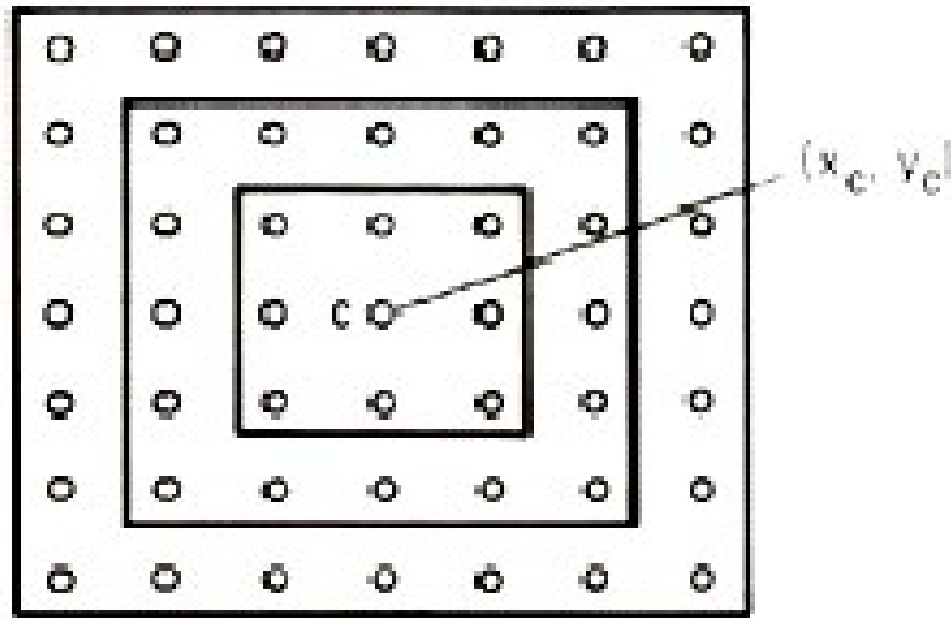
- La neurona con menor valor de acierto gana la competencia. Sea  $c$  la neurona ganadora dado que:

$$\|E - U_c\| = \min_i \|E - U_i\|$$

- Si dos unidades tienen igual valor de acierto, por convención, se elige como ganadora a la de menor índice. Luego de identificar a la ganadora, pasamos a buscar sus vecinos. Las neuronas vecinas son aquellas que se encuentran más cerca en la cuadrícula de la capa de competencia.

# Algoritmo de adaptación <sup>(3/5)</sup>

- Vecindad ( $N_c$ ) para  $d=1, 2$  y  $3$



# Algoritmo de adaptación (4/5)

- Se actualizan los pesos de todas las neuronas pertenecientes al conjunto de vecinos ( $N_c$ ) de la neurona ganadora. La ecuación de ajuste es:

$$\Delta u_{ij} = \begin{cases} \alpha(e_j - u_{ij}), & \text{si la neurona } i \text{ pertenece a } N_c. \\ 0, & \text{en otro caso.} \end{cases}$$

- Notamos que debemos especificar los parámetros  $d$  y  $\alpha$ . La razón de aprendizaje,  $\alpha$ , comienza con un valor grande y decrece durante el proceso de entrenamiento. Inicialmente  $\alpha \in [0.2, 0.5]$ . Una razón de decremento aceptable para  $\alpha$  esta dada por:

$$\alpha_t = \alpha_0 \left( 1 - \frac{t}{T} \right)$$

- Donde  $t$  es la iteración actual y  $T$  es el total de iteraciones a realizar.

# Algoritmo de adaptación (5/5)

- Típicamente el ancho inicial es grande y decrece durante el entrenamiento según:

$$d_t = d_0 \left[ 1 - \frac{t}{T} \right]$$

- $d_0$  puede ser la mitad o la tercera parte del ancho de la capa de competitiva.



# Bibliotecas

- Joone
- Toolbox WEKA
- Toolbox MatLab

# Algunas Ideas

- Clasificar colores (k-means).
- Aprender comportamientos.
- Evolucionar redes con evolutivos.

# Referencias

- Dayhoff J., Neural Network Architectures: An Introduction, Van Nostrand Reinhold, 1990.
- Hertz J.A., A. Krogh y R.G. Palmer, Introduction to the Theory of Neural Computation, Addison Wesley, Enero 1991.
- Haykin S., Neural Network: A Comprehensive Foundation, Macmillan, 1994.
- Teuvo Kohonen, Self-Organizing Maps, Springer, 3a edición, 2000.

Preguntas