Introducción

- **Muchos organismos biológicos poseen sistemas muy sofisticados de reconocimiento de patrones (RP).**
- **La sofisticación de estos sistemas se debe a que ofrecen ventajas de tipo evolutivo.**
  - Supervivencia
    - Reconocimiento de alimentos.
    - Reconocimiento de depredadores.
  - Reproducción
    - Reconocimiento de parejas.
    - Reconocimiento de la descendencia.
- **Hipótesis:**
  - La capacidad de los organismos biológicos para el RP se debe a la estructura del cerebro.
- **Reconocimiento Neuronal de Patrones**
  - Intentar reproducir las capacidades biológicas de RP mediante modelos cerebrales

El Cerebro: ¿Cómo funciona? (1)

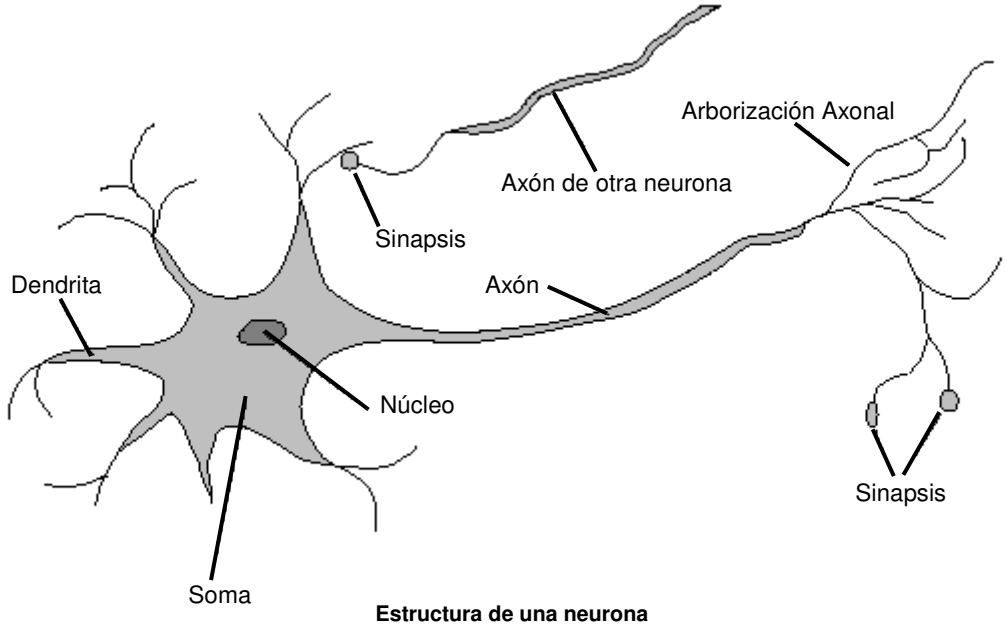
- **Estructura del cerebro:**
  - El cerebro está compuesto de unas células denominadas neuronas interconectadas (Ramón y Cajal, 1911)
  - Hay un gran número de neuronas e interconexiones entre ellas:
    - El número de neuronas es del orden de 10^{11} (del mismo orden que el número de estrellas en la Vía Láctea).
    - Cada una tiene del orden de 10^3 conexiones.
  - Son más lentas que los ordenadores:
    - El tiempo de un ciclo en el ordenador es del orden de 10^{-9} segundos mientras que en las neuronas es del orden de 10^{-3}.
    - La velocidad de transmisión de la información en las neuronas es de 2-120 m/seg. (10^6 veces más lento que un ordenador)
  - El elevado número de neuronas compensa la lentitud del procesamiento.
    - En el sistema visual humano las tareas de reconocimiento llevan del orden de 0.1 a 0.2 segundos. Algo inalcanzable para los ordenadores actuales

Cajal, S. R. (1911). Histologie du Système Nerveux de l'Homme et des Vertébrés, L. Azoulay, trans. Paris: Maloine.

Fernando Pérez Nava

El Cerebro: ¿Cómo funciona? (2)

- **Estructura de una neurona:**
  - Las neuronas se componen de:
    - Soma: cuerpo de la célula, contiene el núcleo.
    - Dendritas:varias fibras de entrada
    - Axón: una única fibra de salida
    - Sinapsis: unión del axón y la dendrita. Cada neurona forma sinapsis con otras 10^1- 10^5 neuronas.

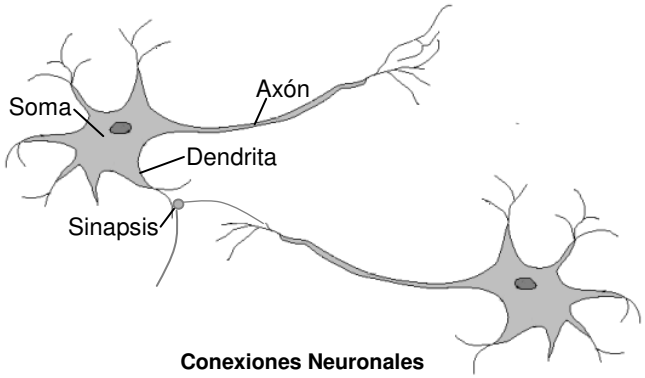


Estructura de una neurona

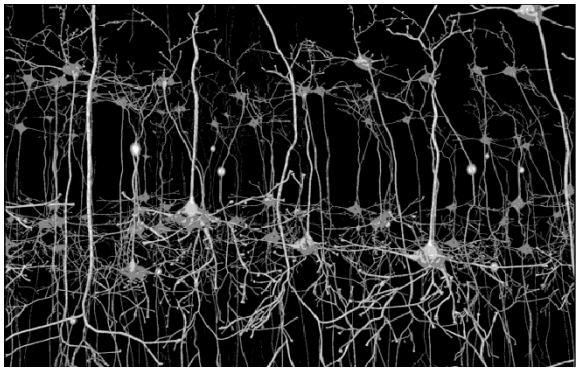
Fernando Pérez Nava

El Cerebro: ¿Cómo funciona? (3)

- **Operaciones de las neuronas biológicas**
  - Una neurona recibe información en forma de pulsos eléctricos, los procesa y los envía a las neuronas vecinas.
    - Los pulsos eléctricos provenientes de otras neuronas viajan a través sus axones.
    - Cuando los pulsos recibidos son los suficientemente elevados la sinapsis provoca el cambio del potencial eléctrico de la dendrita.
    - Cada uno de los potenciales generados en las dendritas se difunde en el soma.
    - En el soma se suman los efectos de miles de estos potenciales. Si la suma excede un cierto umbral la neurona genera un pulso a través de su axón hacia otras neuronas



Conexiones Neuronales



Organización de las neuronas en el cortex visual

El Cerebro: ¿Cómo funciona? (4)

- **Plasticidad Neuronal**
  - Se cree que las sinapsis son las principales responsables del aprendizaje.
  - Las sinapsis del cerebro pueden crearse o modificarse mediante aprendizaje (plasticidad neuronal).
  - En las primeras etapas del desarrollo del cerebro humano el factor fundamental del aprendizaje es la creación de nuevas sinapsis (En los primeros dos años de vida se forman aproximadamente 10^6 sinapsis por segundo).
  - En un adulto el factor fundamental de aprendizaje es la modificación de las sinapsis existentes (aunque siguen creándose nuevas conexiones sinápticas entre las neuronas).
  - Este proceso continuo de readaptación cerebral es el responsable de la tolerancia a fallos (muerte de neuronas) del cerebro.

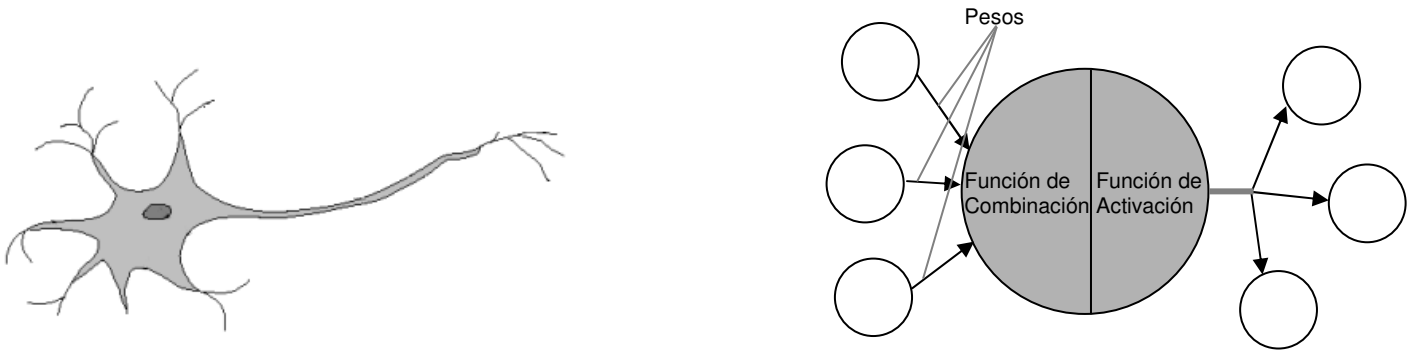
Redes Neuronales (RN)

- **Las RN son un modelo (extremadamente) simplificado del cerebro.**
  - El modelado ayuda a evitar los detalles no esenciales
  - Nos permite aplicar las herramientas matemáticas
  - Proporciona la esperanza de que la comprensión de un modelo básico ayude en el estudio de un modelo más complejo.
  - Se sabe que los modelos de RN actuales no son correctos (por ejemplo se trabaja con señales continuas y no con pulsos).
- **Una RN está compuesta por una gran cantidad de nodos simples (neuronas) interconectados y que operan en paralelo.**
- **Una Red Neuronal (RN) se caracteriza por:**
  - Nodos: características y propiedades.
  - Arquitectura: la forma de conexión de los nodos
  - Pesos: fuerza de las conexiones entre los nodos.
  - Plasticidad: regla de modificación (aprendizaje) para los pesos

Fernando Pérez Nava

Modelos Neuronales: Unidades (1)

- **Funcionamiento de una unidad:**
  - Recibe las entradas de otras neuronas multiplicadas por los pesos.
  - Combina las entradas mediante una función de combinación
  - Esta combinación se pasa a una función de activación que calcula la activación de la unidad.
  - Esta activación se envía a las unidades a las que se conecta.
- **Analogías:**
  - Neurona: Unidad
  - Dendrita: Conexiones
  - Cuerpo: Función de Combinación, Función de activación.
  - Axón: Conexión a otras neuronas
  - Pulso: Salida de la neurona
  - Sinapsis: Pesos de la conexión



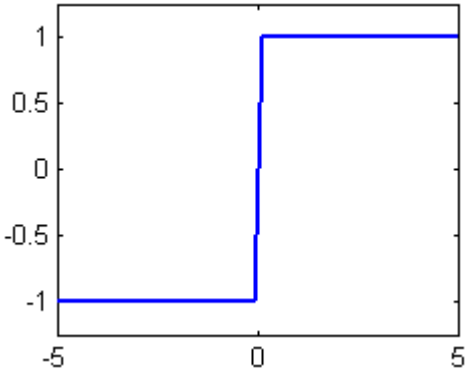
Neurona biológica y neurona formal

Modelos Neuronales: Unidades (2)

- **Función de combinación:**
  - Suele ser la suma de las entradas
- **Función de activación τ:**
  - Las más utilizadas son:

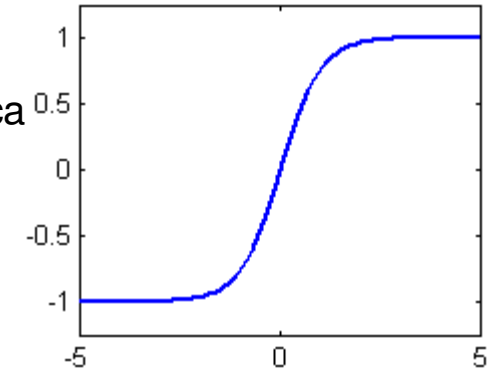
Umbral

$$\tau(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$



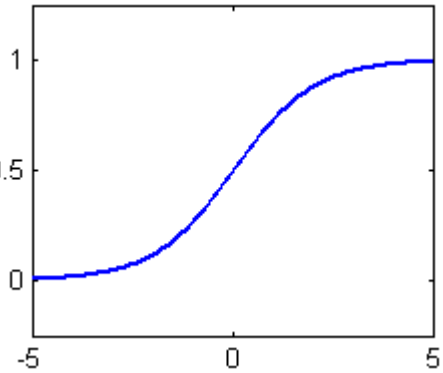
Tang. Hiperbólica

$$\tau(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



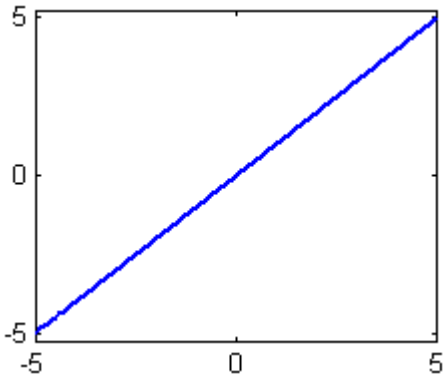
Logística

$$\tau(x) = \frac{1}{1 + \exp(-x)}$$



Lineal

$$\tau(x) = x$$

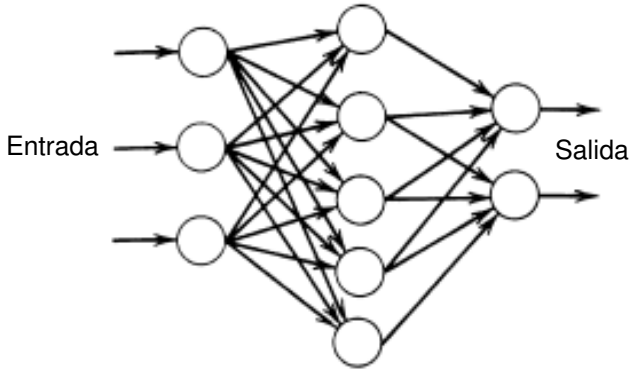


Funciones de Activación

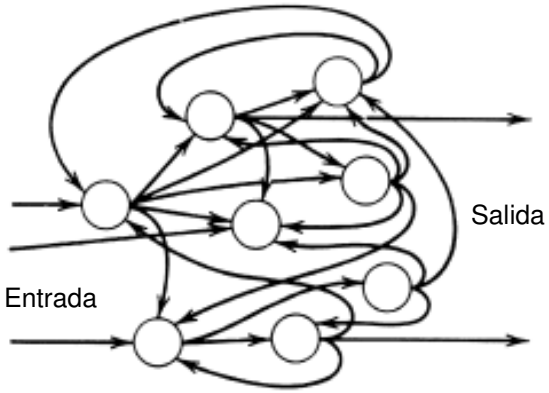
Fernando Pérez Nava

Modelos Neuronales:Arquitecturas

- **Se pueden encontrar dos tipos fundamentales de arquitecturas**
  - Redes con Propagación hacia delante
    - Las unidades se dividen en:
      - Unidades de entrada: Reciben los datos del entorno
      - Unidades de salida: Devuelven los resultados de la red al entorno
      - Unidades ocultas: Sin relación directa con el entorno
    - No se permiten ciclos en las conexiones entre las unidades
    - Las unidades suelen estar dispuestas en capas. Las unidades de cada capa reciben información de la capa anterior y la envían a la siguiente.
  - Redes Recurrentes
    - Se permiten ciclos en las conexiones.
    - Son más realistas desde el punto de vista biológico.



Red con Propagación hacia delante



Red recurrente

Fernando Pérez Nava

Modelos Neuronales: Aprendizaje

- **A diferencia del ordenador las RN no se programan para realizar las tareas requeridas.**
- **El aprendizaje se lleva a cabo mediante la modificación de los pesos de las conexiones.**
- **El aprendizaje en las RN puede ser:**
  - Supervisado: Se muestra a la red los datos de entrada y las salidas deseadas. El objetivo es que la red genere las salidas deseadas para los datos de entrada
  - No Supervisado: Se entrena a la red para que encuentre agrupaciones en los datos de entrada
- **Se pueden encontrar diversos métodos para realizar el aprendizaje en RN:**
  - Supervisado:
    - Regla de Hebb, Retropropagación, etc.
  - No Supervisado:
    - Regla de Kohonen, Cuantización Vectorial, etc.

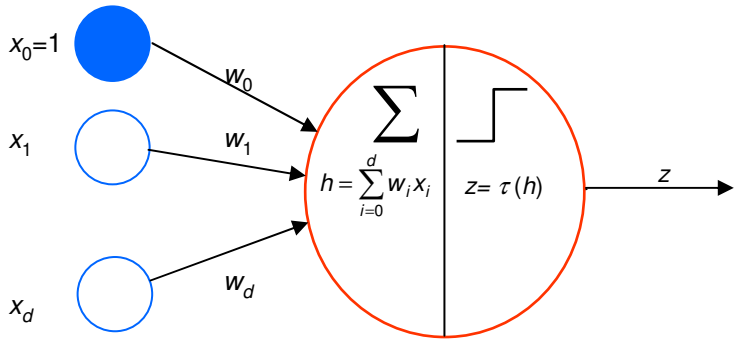
El Perceptrón (1)

- **Desarrollado por Rosenblatt (1962)**
- **Modelo Neuronal:**
  - Unidades:
    - Función de combinación:suma
    - Función de activación τ: de tipo umbral

      Este tipo de unidad fue el primer modelo formal de neurona (McCulloch y Pitts, 1944)

$$\tau(h) = \begin{cases} +1 & h \geq 0 \\ -1 & h < 0 \end{cases}$$

  - Arquitectura:
    - Red de propagación hacia delante con dos capas.



Representación de un perceptrón

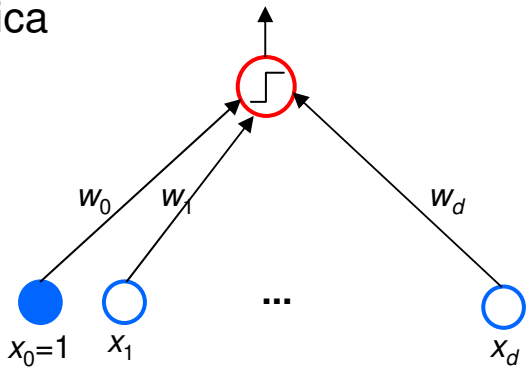
Fernando Pérez Nava

El Perceptrón (2)

- **De forma matemática:**

$$z(\boldsymbol{x}) = \tau\left(\sum_{j=0}^{d} w_j x_j\right) = \tau\left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}\right), \quad \tau(h) = \begin{cases} 1 & t \geq 0 \\ -1 & t < 0 \end{cases}$$

  - La salida del perceptrón se obtiene multiplicando las entradas por los pesos y devolviendo +1 si el resultado es positivo o nulo y –1 si es negativo.
- **Clasificación con el Perceptrón.**
  - Dado un vector de características $\boldsymbol{x}$ si la salida del perceptrón $z(\boldsymbol{x})$ es positiva o nula se asigna a la primera clase si es negativa a la segunda.
- **El Perceptrón no es más que un caso especial de función discriminante lineal.**
  - Representación gráfica



- **Por tanto, las fronteras de decisión que genera son lineales.**

Fernando Pérez Nava

Entrenamiento del Perceptrón (1)

- **El entrenamiento del Perceptrón difiere de las FDL usuales.**
  - La razón es debida a que el método del gradiente no se puede aplicar de forma directa porque la función de transferencia (umbral) no es diferenciable.
- **El entrenamiento del Perceptrón se realiza calculando el mínimo de una función de error.**
  - Representaremos el signo deseado para cada elemento del conjunto de entrenamiento como: $y_i = 1$ si $\boldsymbol{x}_i \in w_1$, $y_i = -1$ si $\boldsymbol{x}_i \in w_2$
  - Para que todo el conjunto de entrenamiento esté bien clasificado es necesario que el signo deseado y el obtenido $z(x_i)$ coincidan.
  - Se define entonces la función de error:

$$E_P(\boldsymbol{w}) = \sum_{i=1}^{n} -\delta_i \, \boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_i$$

$\boldsymbol{x}_i$ es la muestra i de H

y_i es el signo deseado del perceptrón para $\boldsymbol{x}_i$: $y_i = \begin{cases} 1 & \boldsymbol{x}_i \in w_1 \\ -1 & \boldsymbol{x}_i \in w_2 \end{cases}$

δ_i es la diferencia entre la salida deseada y_i y la obtenida $z(\boldsymbol{x}_i)$:

$\delta_i = y_i - z(\boldsymbol{x}_i)$

Fernando Pérez Nava

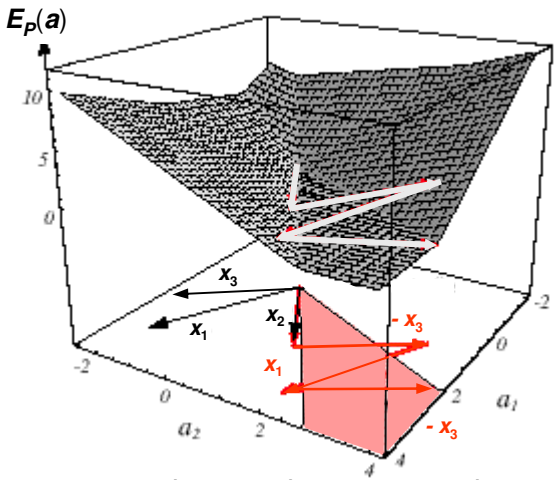
Entrenamiento del Perceptrón (2)

- **Optimización**
  - Se basa en aplicar el método del gradiente a la nueva función de error.
  - Entrenamiento por época:

$$\boldsymbol{w}^{(r+1)} = \boldsymbol{w}^{(r)} + \rho_r \sum_i \delta_i \, \boldsymbol{x}_i$$

  - Entrenamiento por muestra:

$$\boldsymbol{w}^{(r+1)} = \boldsymbol{w}^{(r)} + \rho_r \delta_i \, \boldsymbol{x}_i$$



Optimización por el método del perceptrón

Fernando Pérez Nava

Convergencia del Algoritmo del Perceptrón

- **El algoritmo converge en un número finito de pasos siempre que:**
  - El conjunto de entrenamiento sea linealmente separables (en cuyo caso proporciona una solución que los separa)
  - La sucesión de parámetros de entrenamiento ρ_r sea constante $\rho_r=\rho$ o la sucesión de parámetros de entrenamiento ρ_r sea variable y cumpla:

$$\rho_r \geq 0, \quad \lim_{n\to\infty} \sum_{r=1}^{n} \rho_r = \infty, \quad \lim_{n\to\infty} \sum_{r=1}^{n} \rho_r^2 < \infty$$

  - Un ejemplo de sucesión que cumple esas condiciones es la:

$$\rho_r = \frac{1}{r}$$

Fernando Pérez Nava

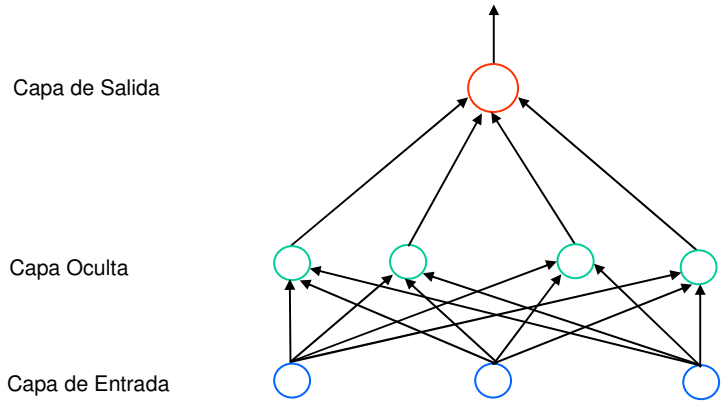
Perceptrón: Extensiones

- **El caso no separable linealmente**
  - Se obtiene la solución con menor número de errores de clasificación con probabilidad 1 mediante el algoritmo del "bolsillo" (Gallant 1990)
- **Caso multiclase:**
  - Mediante la construcción de Kesler.
- **Otras funciones de activación τ**
  - Con τ lineal se obtiene el modelo de regresión lineal (Tema 4)
  - Con τ logística se obtiene el modelo de regresión logística (Tema 4)
- **Caso no lineal**
  - Colocar varias capas de neuronas para obtener un clasificador no lineal

Fernando Pérez Nava

Perceptrón Multicapa

- **Un Perceptrón es un tipo de Función Discriminante Lineal**
  - Por tanto genera fronteras de decisión lineales
  - Para problemas de clasificación más complejos se necesita un clasificador capaz de generar fronteras de decisión no lineales.
- **Solución:**
  - Colocar varias capas de neuronas. Así se obtiene el denominado Perceptrón Multicapa.

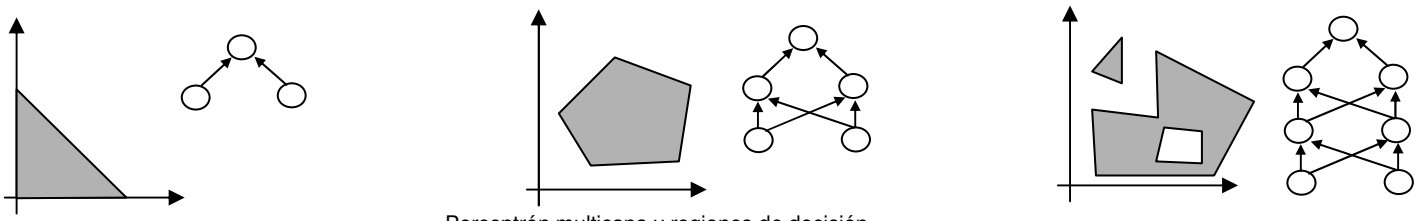


Perceptrón multicapa

Fernando Pérez Nava

Capacidad de Representación

- **Un Perceptrón Multicapa (PMC) con función de activación de tipo umbral:**
  - Con una capa genera regiones de decisión conexas cuya frontera de decisión es lineal.
  - Con dos capas puede generar regiones de decisión conexas cuyas fronteras son lineales.
  - Con tres capas genera regiones de decisión arbitrarias.

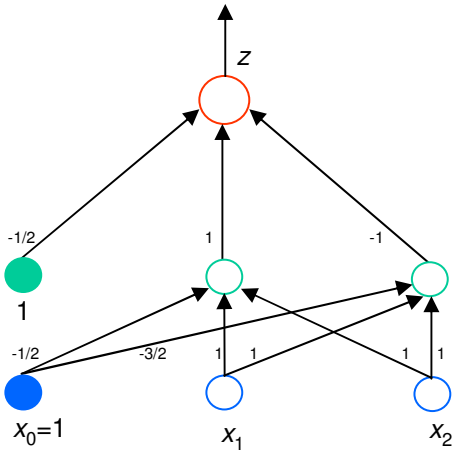


Perceptrón multicapa y regiones de decisión

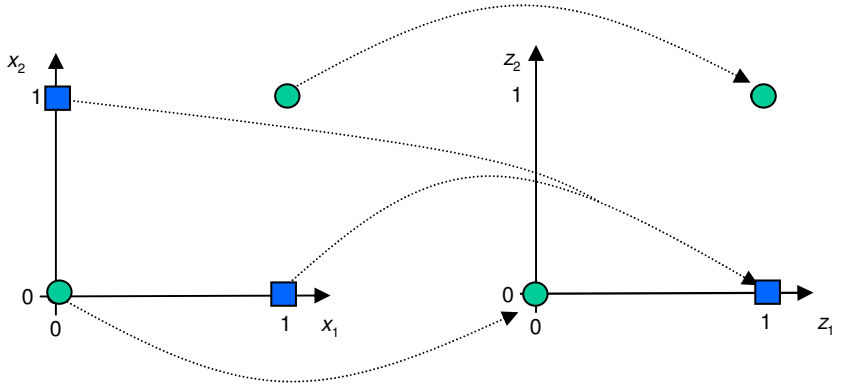
- **Un PMC con función de activación sigmoidal (logística, tangente hiperbólica):**
  - Con dos capas puede aproximar cualquier función continua (y por tanto cualquier frontera continua) con precisión arbitraria con un número suficientemente grande de nodos en la capa oculta.

PMC: Ejemplo

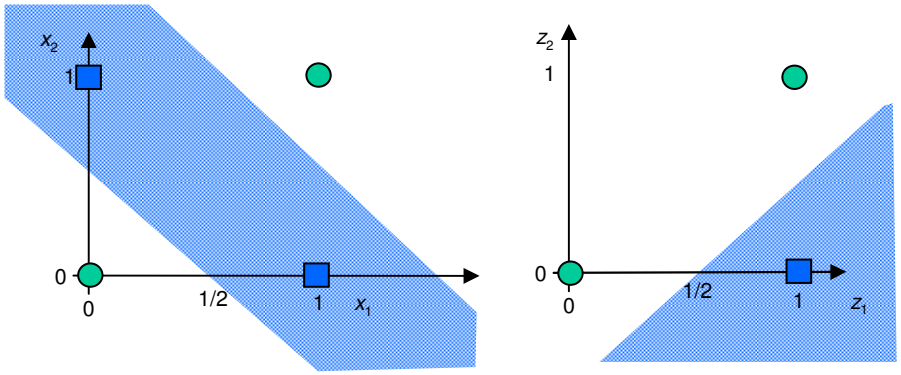
- **Un perceptrón de 2 capas que resuelve el problema del XOR**



Perceptrón de dos capas con función de activación de tipo umbral $\tau(x)=1$, $x\geq 0$; $\tau(x)=0$, $x<0$



Transformación de las entradas



Representación gráfica de las regiones de decisión (que no son acotadas)

Fernando Pérez Nava

Aprendizaje en el PMC

- **Aprendizaje en el PMC**
  - La presencia de unidades ocultas hace complejo el aprendizaje.
  - El aprendizaje de los pesos de la capa oculta a la de salida es fácil. La capa oculta proporciona las características transformadas. El problema que queda es el aprendizaje de una función discriminante lineal
  - El aprendizaje de los pesos de la capa de entrada a la oculta es difícil. Estos pesos transforman los datos de entrada de forma óptima para clasificarlos.
- **El método más extendido de aprendizaje en el PMC es el "BackPropagation" o "RetroPropagación" (RPR).**
  - Es un método de aprendizaje supervisado basado en el método del gradiente que minimiza una función de error entre las salidas deseadas por la red y las salidas obtenidas.
  - Requiere una función de activación diferenciable.

Fernando Pérez Nava

Aprendizaje mediante RPR

- **Funciones de error más usuales:**
  - Regresión:
    - Función de error: Error Cuadrático Medio:

$$E_{ECM}(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - z(\boldsymbol{x}_i;\boldsymbol{w})\right)^2$$

donde y_i es la salida deseada para x_i y $z(x_i;\boldsymbol{w})$ es la salida de la red para x_i con el conjunto de pesos $\boldsymbol{w}$.

  - Clasificación:
    - Inspirados en la regresión logística se asume que la probabilidad a posteriori de la clase se escribe como:

$$\mathrm{P}(w_i \mid \boldsymbol{x};\boldsymbol{w}) = \frac{1}{1+\exp(-y_i z(\boldsymbol{x};\boldsymbol{w}))} = \tau(-y_i z(\boldsymbol{x};\boldsymbol{w})), \quad y_i = \begin{cases} 1 & \text{para } w_1 \\ -1 & \text{para } w_2 \end{cases}$$

$$L = p(H \mid \boldsymbol{w}) = \prod_{k=1}^{n} \mathrm{P}(y_k \mid \boldsymbol{x}_k, \boldsymbol{w})$$

y se maximiza la verosimilitud de los datos. O de forma equivalente se minimiza la entropía cruzada:

$$E_{ENT}(\boldsymbol{w}) = -\sum_{i=1}^{n}\left(y_i \ln z(\boldsymbol{x}_i,\boldsymbol{w}) + (1-y_i)\ln(1-z(\boldsymbol{x}_i,\boldsymbol{w}))\right), \quad y_i = \begin{cases} 1 & \boldsymbol{x}_i \in w_1 \\ 0 & \boldsymbol{x}_i \in w_2 \end{cases}$$

Fernando Pérez Nava

Algoritmo RPR

- **Esquema general:**
  - Entrenamiento por época

    Paso 1

    › Para cada elemento del conjunto de entrenamiento
      - Calcular el incremento en los pesos debido a ese elemento
        - » Propagar hacia delante los datos
        - » Propagar hacia detrás los errores

    Paso 2

    › Actualizar los pesos añadiendo los incrementos debido a cada uno de los elementos.

    Paso 3

    › Parar si se verifica una regla de parada (por ejemplo el incremento de los pesos está por debajo de un umbral). En otro caso volver al paso 1.
  - Entrenamiento por muestra

    › Se actualizan los pesos en cuanto se calcula el incremento debido a un elemento.

Algoritmo RPR

- **Cálculo del incremento en los pesos debido a un elemento del conjunto de entrenamiento *x* con salida deseada *y*.**

  Llamaremos e a la entrada de una neurona y $s = \tau(e)$ a su salida.

  Propagación hacia delante de los datos

  › Obtener $z = s_{salida}$ la salida de la red para $\boldsymbol{x}$ con el conjunto actual de pesos $\boldsymbol{w}^{(r)}$

  Propagación hacia atrás de los errores

  › Obtener el error de la neurona de salida:

  – E_{ECM}: $\delta_{salida} = (y - z)\ \tau'(e_{salida})$

  – E_{ENT} : $\delta_{salida} = (y - z)$. (Utilizando como función de transferencia en la neurona de salida la sigmoide logística)

  » $\tau'(e_{intermedia,}) = s_{intermedia,}(1 - s_{intermedia})$ para la función logística

  » $\tau'(e_{intermedia,}) = 1 - (s_{intermedia})^2$ para la tangente hiperbólica

  » $\tau'(e_{intermedia,}) = 1$ para la función identidad

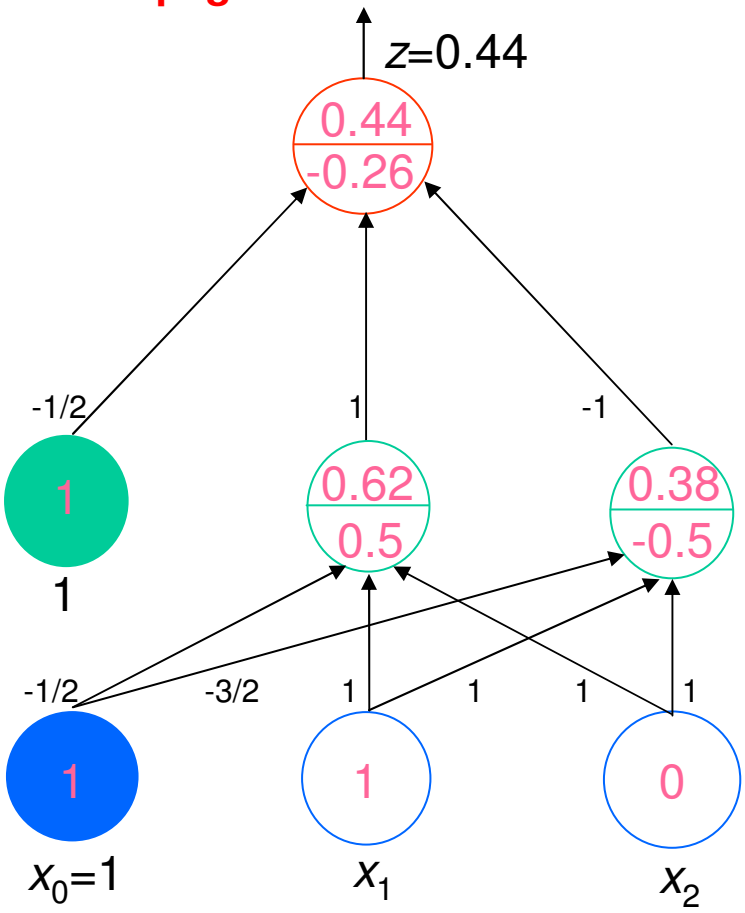
  › Para cada neurona intermedia calcular su error como:

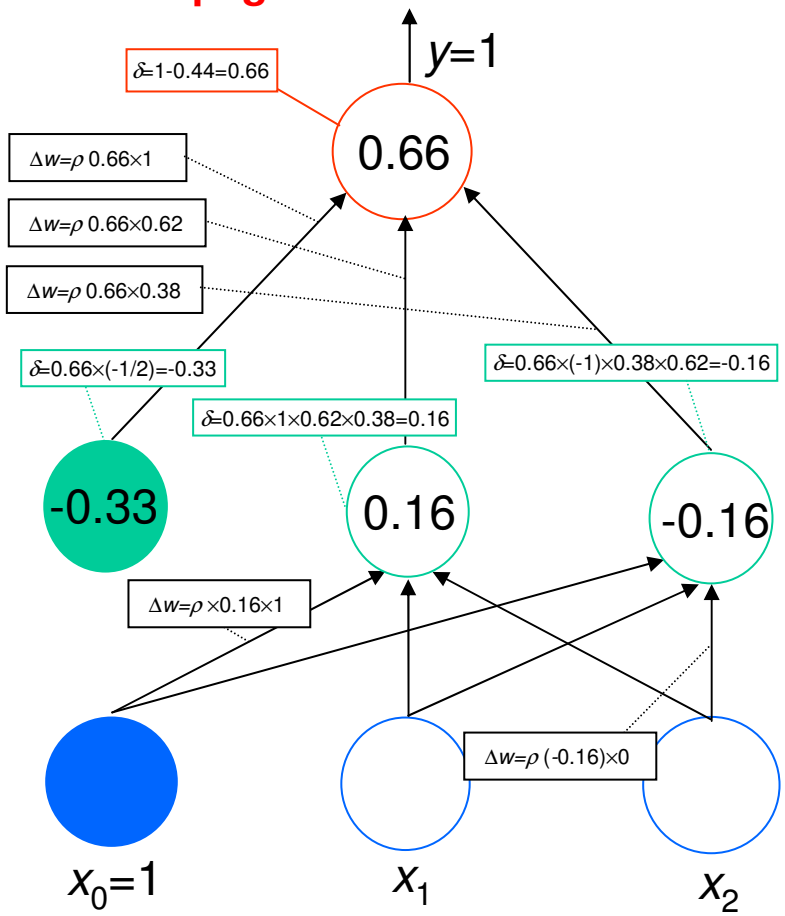
  $\delta_{intermedia} = \delta_{salida}\, w_{intermedia,salida}\, \tau'(e_{intermedia})$ donde $w_{intermedia,salida}$ es el peso que une la neurona intermedia con la de salida.

  › El incremento del peso de cada enlace debido a $\boldsymbol{x}$ es el producto del parámetro de aprendizaje ρ_r por el valor del δ de la neurona a la que apunta y por el valor de salida s de la neurona de la que parte.

Algoritmo RPR:Ejemplo de iteración

- **Problema del XOR. Función de transferencia logística**
- **Propagación hacia delante**
- **Propagación hacia detrás**





Fernando Pérez Nava

Extensiones del RPR

- **El algoritmo anterior se puede extender:**
  - A más de dos clases
    - Para ello se coloca una neurona en la capa de salida por clase.
    - Si e_i es la entrada a la neurona i de la capa de saliday $s_j = \tau(e_j)$ a su salida.
    - La función de transferencia de cada neurona para problemas de clasificación debe ser:
    - La clase ganadora es aquella para la que se obtiene el mayor valor de salida en su neurona correspondiente.
  - A más de una capa oculta
  - A distintas funciones de transferencia
  - A un parámetros de entrenamiento por peso

Fernando Pérez Nava

Aspectos Prácticos del RPR (1)

- **Utilizar el entrenamiento por muestra**
  - Suele converger mucho más rápido
  - Suele proporcionar mejores soluciones
- **Aleatorizar el orden de presentación de los patrones en el entrenamiento por muestra.**
  - A este tipo de entrenamiento se le llama estocástico
- **Reescalar las entradas**
  - Hacer que las entradas tengan media 0 y varianza 1
- **Número de capas**
  - Puesto que dos capas son suficientes para representar cualquier función éste suele ser su número. Un número de mayor de capas puede utilizase para efectuar algún preprocesamiento explícito.
  - De forma empírica se observa que un mayor número de capas hace que el RPR sea más propenso a converger a óptimos locales.

Aspectos Prácticos del RPR (2)

- **Inicialización de los pesos**
  - Los valores iniciales de los pesos suelen ponerse de forma aleatoria y su magnitud suele ser pequeña.
  - Una regla utilizada es utilizar los rangos $\left[-1/\sqrt{d},1/\sqrt{d}\right]$ y $\left[-1/\sqrt{M},1/\sqrt{M}\right]$ para las conexiones entrada-oculta y oculta-salida donde *d* es la dimensión de los datos y *M* el número de nodos en la capa oculta.
- **Parámetro de aprendizaje**
  - Su valor determina la velocidad de convergencia. Si es muy pequeño la convergencia es lenta mientras que si es muy grande el método del RPR no converge. Un valor típico es $\rho = 0.1$
- **Momentos**
  - Suelen acelerar el proceso de aprendizaje. Se basa en combinar el cambio en los pesos del RPR $\Delta \boldsymbol{w}_{RPR}$ con los cambios anteriores:

$$\boldsymbol{w}^{(r+1)}=\boldsymbol{w}^{(r)} + (1-\alpha)\,(\Delta \boldsymbol{w}_{RPR})+ \alpha\,(\boldsymbol{w}^{(r)} - \boldsymbol{w}^{(r-1)})$$

  Un valor típico para α es 0.9

Fernando Pérez Nava

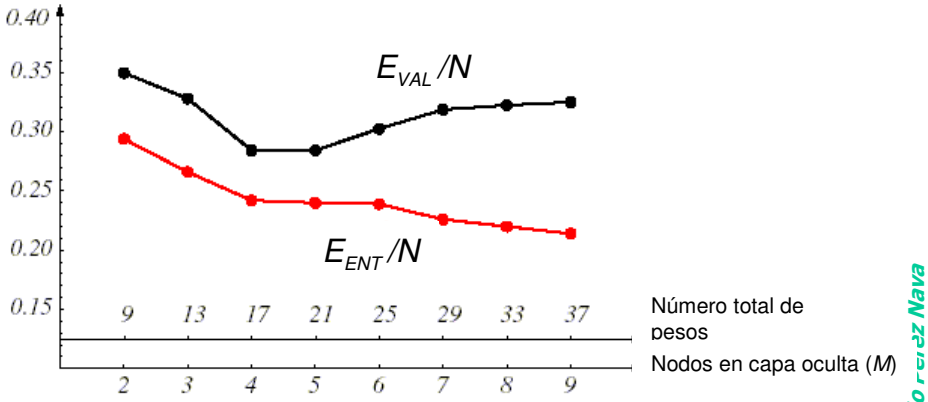
Aspectos Prácticos del RPR (3)

- **Métodos de segundo orden**
  - Son métodos que aceleran la convergencia mediante la utilización de la información proporcionada por la segunda derivada de la función de error *E.*
  - Están basados en métodos de optimización que utilizan las segundas derivadas. Su principal inconveniente es el mayor costo computacional.
  - La mayor parte de los métodos sólo funcionan para aprendizaje por lotes. Esto hace que no puedan aplicarse en la práctica para redes de gran tamaño o conjuntos de entrenamiento con un número elevado de elementos.
- **Número de neuronas en la capa oculta**
  - El número de neuronas en la capa oculta determina la complejidad del clasificador. Un número muy grande provoca sobreajuste y un número muy pequeño provoca malos resultados en la clasificación.
  - Una regla genérica es determinar los nodos a partir de un número de pesos igual a la décima parte del número de datos

Fernando Pérez Nava

Número de Nodos en la Capa Oculta

- **Regla Genérica**
  - Se suele determinar el número de nodos en la red a partir de un número de pesos igual a la décima parte del número de datos
- **Entrenamiento y Validación**
  - Se divide el conjunto de entrenamiento en dos subconjuntos de entrenamiento H_{ENT} y validación H_{VAL}. El entrenamiento se hace solamente sobre H_{ENT} y una vez converja el RPR se comprueban los errores E_{ENT} y E_{VAL} sobre los conjuntos H_{ENT} y H_{VAL} .
  - La complejidad óptima del PMC se obtiene si los errores E_{ENT} y E_{VAL} se mantienen bajos tras la convergencia del RPR.



Errores medios de entrenamiento y validación para una red 2-*M*-1 entrenada con 180 muestras. Los valores óptimos de M son 4,5