

# Perceptron Multicapas

Aprendizaje maquina

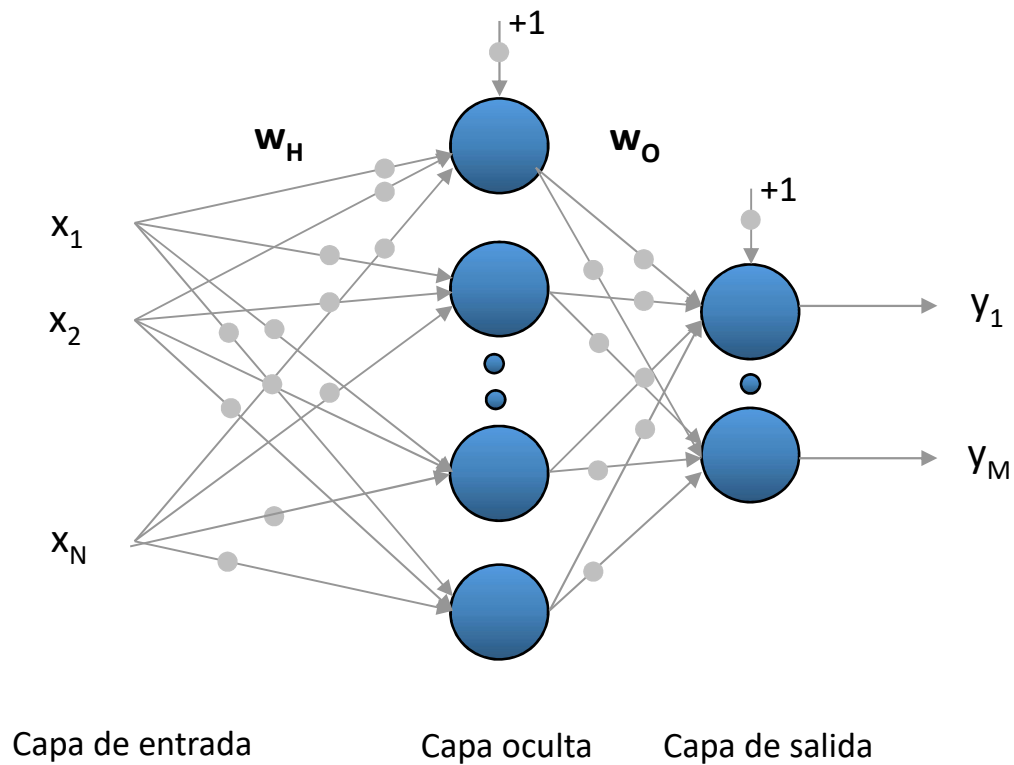
Dr. Rubén Acevedo  
ruben.acevedo@uner.edu.ar

Tecnicatura Universitaria  
Procesamiento y Exploración de Datos

# Redes neuronales artificiales

Introducción

Estructura



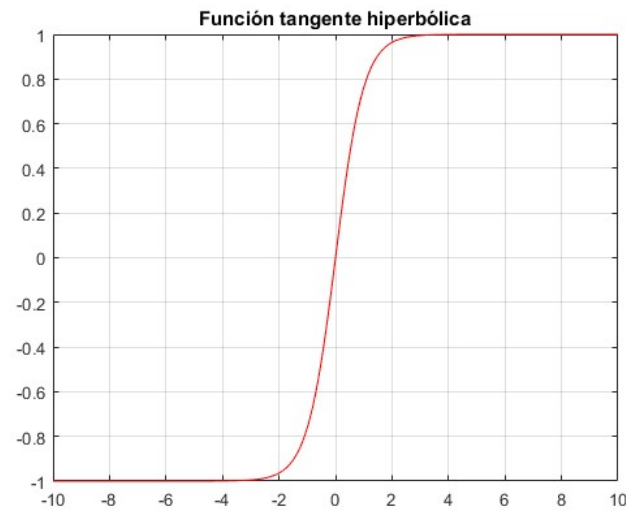
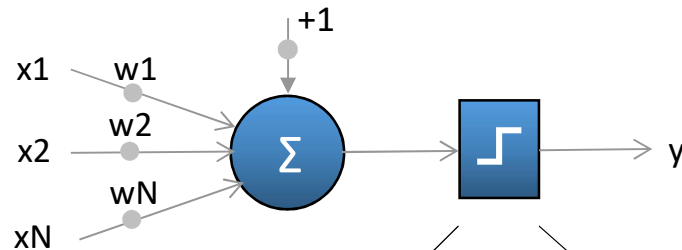
# Redes neuronales artificiales

Introducción

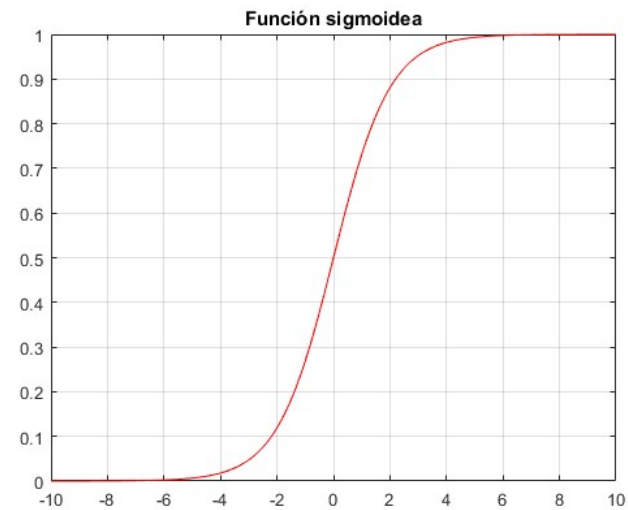
Estructura

Función de activación

Nodo de un PMC:



$$y(\theta) = \frac{\sinh(\theta)}{\cosh(\theta)}$$



$$y(\theta) = \frac{1}{1 + e^{-\theta}}$$

# Redes neuronales artificiales

Algoritmo de aprendizaje

Retropropagación del error

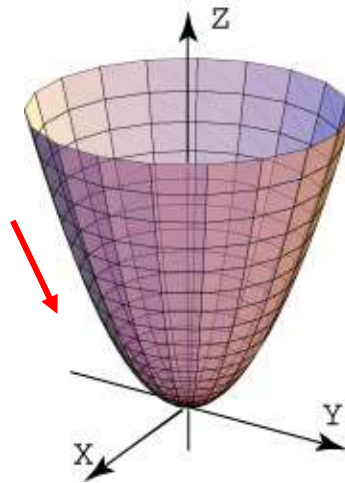
$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{pmatrix}$$

Salida

$$yd = \begin{pmatrix} yd_1 \\ yd_2 \\ \vdots \\ yd_M \end{pmatrix}$$

Salida deseada

$$e(n)^2 = \sum_{j=1}^M (yd_j(n) - y_j(n))^2$$



$$ECM = \frac{1}{N} \cdot \sum_{n=1}^N e(n)^2$$

# Redes neuronales artificiales

Algoritmo de aprendizaje

Retropropagación del error

El error cuadrático medio (ECM) es función de los parámetros libres (pesos) del perceptron multicapas (PMC), y para un conjunto de patrones de entrada representa el parámetro de medida del proceso de aprendizaje (*función de costo*).

El objetivo del proceso de aprendizaje es ajustar los pesos del PMC con el objetivo de minimizar esta función error.

# Redes neuronales artificiales

Algoritmo de aprendizaje

Retropropagación del error

Si bien las neuronas de las capas ocultas no están directamente accesibles, comparten la responsabilidad del error que se genera en las neuronas de salida.

Cuando una neurona pertenece a una capa oculta, no tiene predefinido el valor de la salida deseada para cada presentación de un patrón de entrada.

# Redes neuronales artificiales

Algoritmo de aprendizaje

Retropropagación del error

En consecuencia, la señal de error de una neurona de una capa oculta se debe calcular recursivamente en términos de los errores de todas las neuronas a las que la neurona intermedia está directamente conectada.

La idea central de este algoritmo es calcular los errores para las unidades de las capas ocultas a partir de los errores de las unidades de la capa de salida siendo propagados capa tras capa hacia la entrada.

# Redes neuronales artificiales

## Algoritmo de aprendizaje

## Retropropagación del error

- a) Inicializar los pesos aleatoriamente  $\mathbf{W}_H$  y  $\mathbf{W}_O$  entre (-0.5 , 05).
- b) Escoger un patrón del conjunto de entrenamiento  $\mathbf{x}$ , calcular la salida y el error asociado en cada nodo de la capa de salida.

- c) Adaptar los pesos: empezando con la capa de salida, y “*modificando hacia atrás*”, según la ecuación

$$w_{ij}(n + 1) = w_{ij}(n) + \mu \cdot \delta_{pj} \cdot y_{pj}$$

donde  $y_{pj}$  es la salida real de la capa anterior y  $\delta_{pj}$  es el gradiente cuya expresión difiere según la capa que se esté adaptando.

Nodos de la capa de salida:

$$\delta_{pj} = y_{pj} \cdot (1 - y_{pj}) \cdot (d_{pj} - y_{pj})$$

donde  $d_{pj}$  es la respuesta deseada.

Nodos en la capa escondida:

$$\delta_{pj} = y_{pj} \cdot (1 - y_{pj}) \cdot \sum_{j=1}^K \delta_{pN} \cdot w_{jN}$$

- d) Volver al paso b) y repetir el proceso hasta que los pesos converjan.



# Redes neuronales artificiales

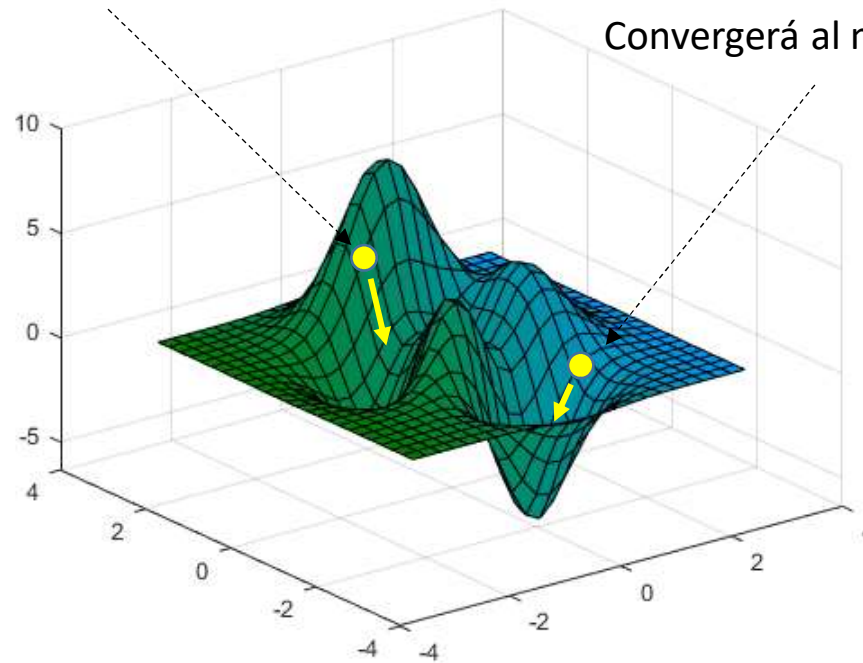
Algoritmo de aprendizaje

Retropropagación del error

Inicialización de los pesos

No convergerá al mínimo global

Convergerá al mínimo global

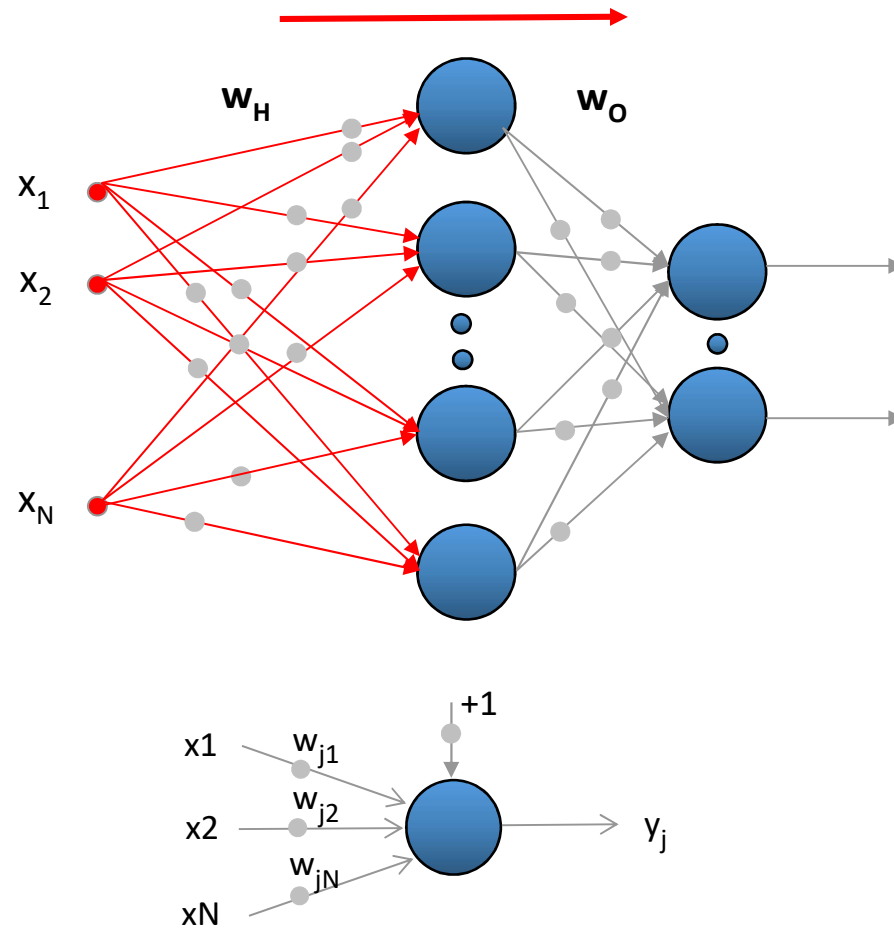


# Redes neuronales artificiales

Algoritmo de aprendizaje

Retropropagación del error

Presentación patrón

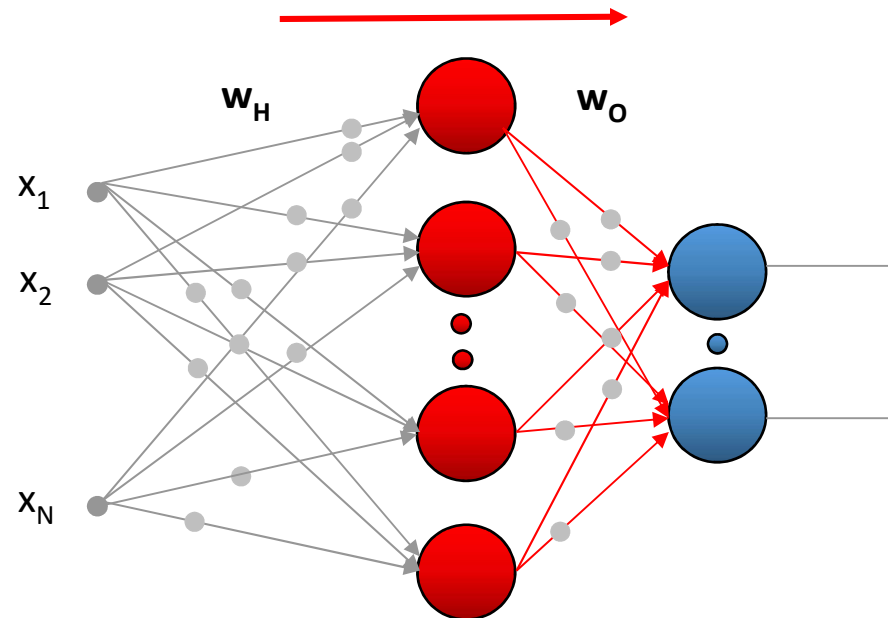


# Redes neuronales artificiales

Entrenamiento

Retropropagación del error

Cálculo salida capa oculta



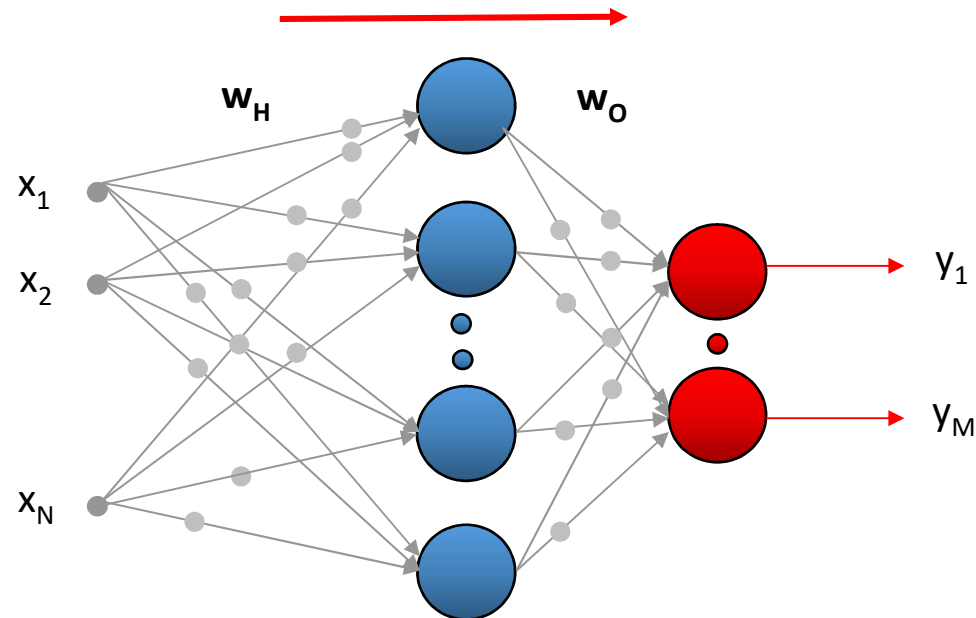
$$o_j = \frac{1}{1 + e^{-(\sum_{i=0}^K w_{ji} \cdot x_i)}}$$

# Redes neuronales artificiales

Entrenamiento

Retropropagación del error

Cálculo salida



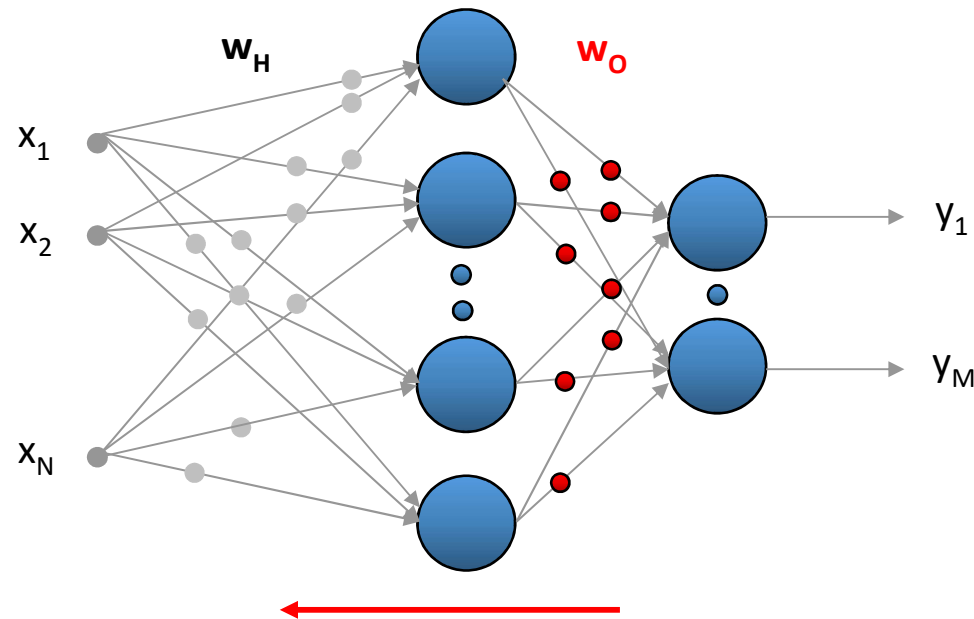
$$y_j = \frac{1}{1 - e^{-(\sum_{i=0}^K w_{ji} \cdot o_i)}}$$

# Redes neuronales artificiales

Algoritmo de aprendizaje

Retropropagación del error

Actualización pesos capa salida



$$w_{ij}(n + 1) = w_{ij}(n) + \mu \cdot \delta_{pj} \cdot o_{pj}$$

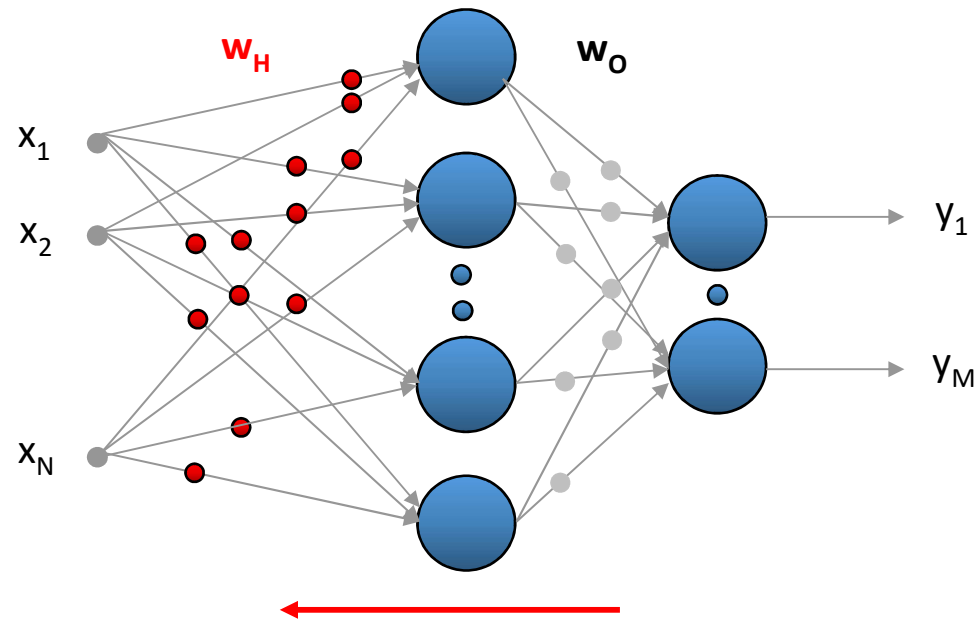
Actualización de pesos de la  $p$ -ésimo nodo de la capa de salida.

# Redes neuronales artificiales

Algoritmo de aprendizaje

Retropropagación del error

Actualización pesos capa oculta



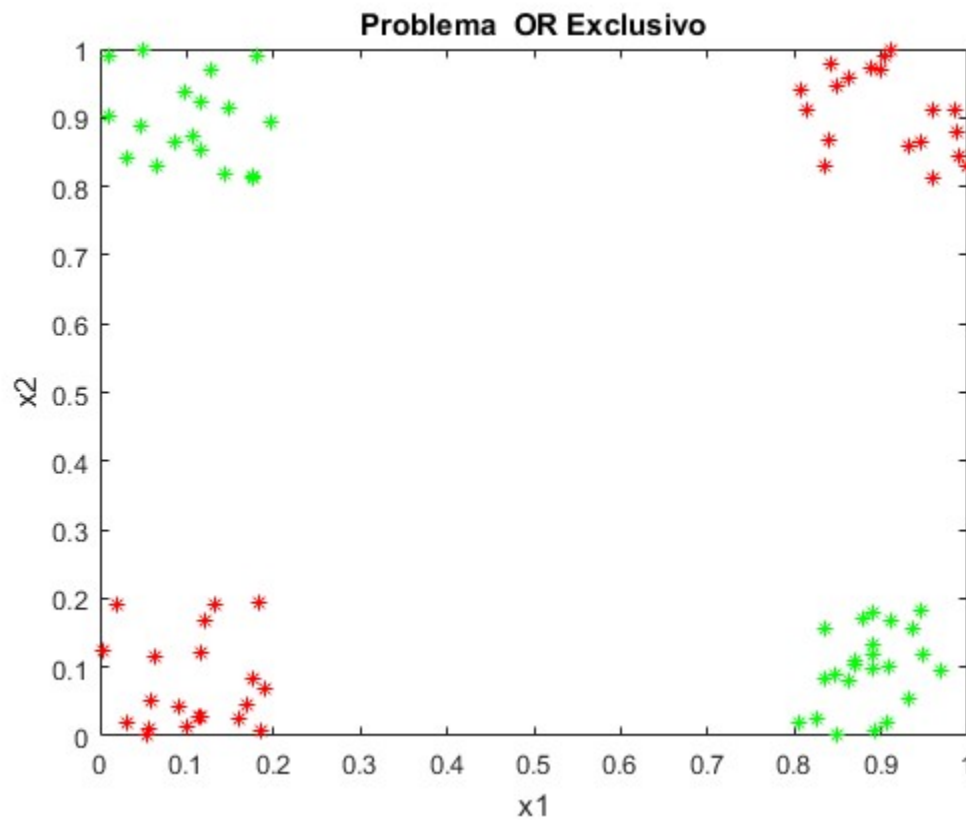
$$w_{ij}(n + 1) = w_{ij}(n) + \mu \cdot \delta_{pj} \cdot x_{pj}$$

Actualización de pesos de la  $p$ -ésimo nodo de la capa oculta.

# Redes neuronales artificiales

Ejemplo

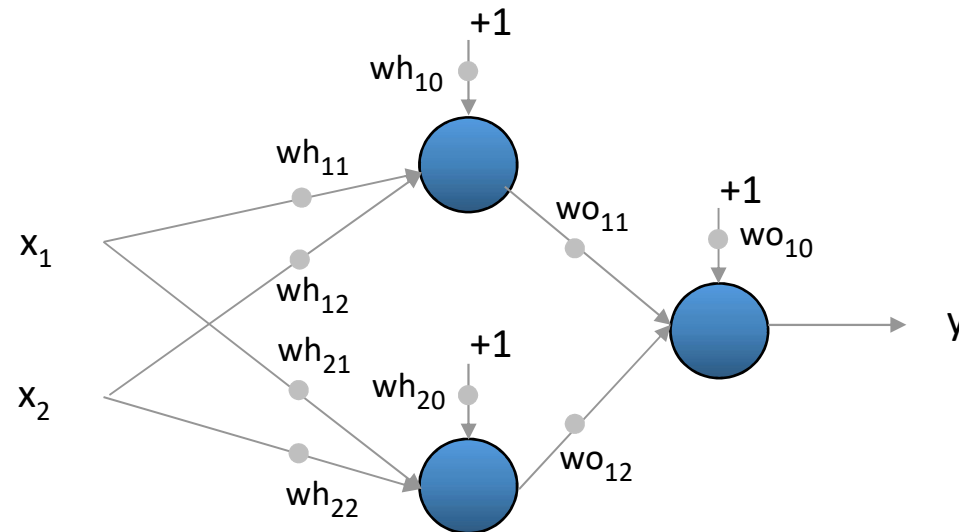
OR exclusivo



# Redes neuronales artificiales

Ejemplo

OR exclusivo



$$W_H = \begin{pmatrix} wh_{10} & wh_{11} & wh_{12} \\ wh_{20} & wh_{21} & wh_{22} \end{pmatrix}$$

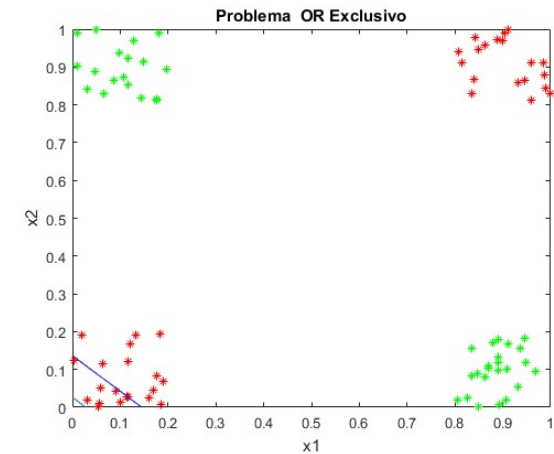
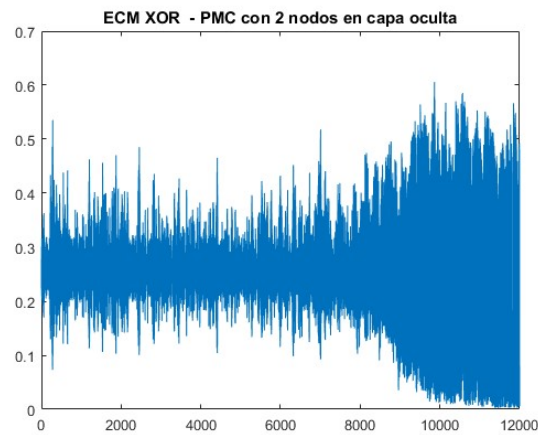
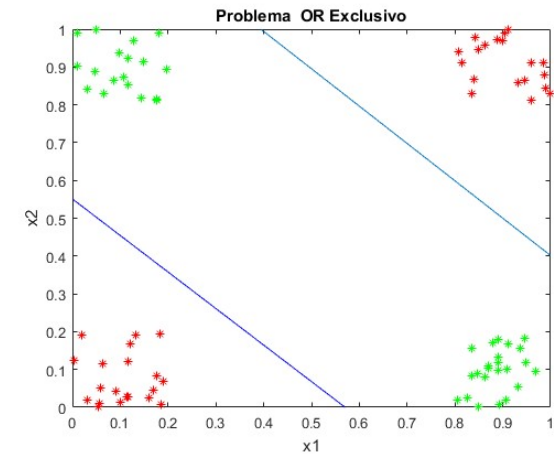
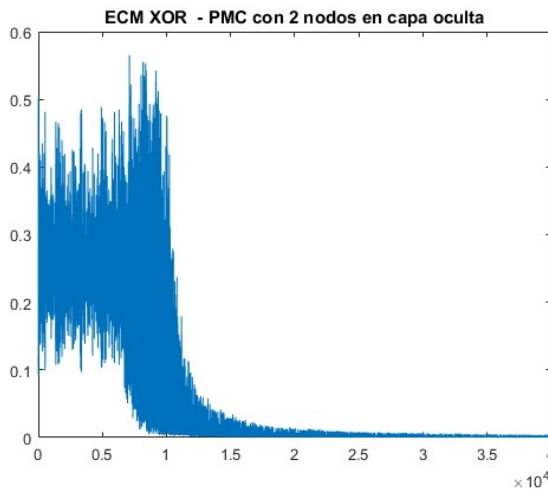
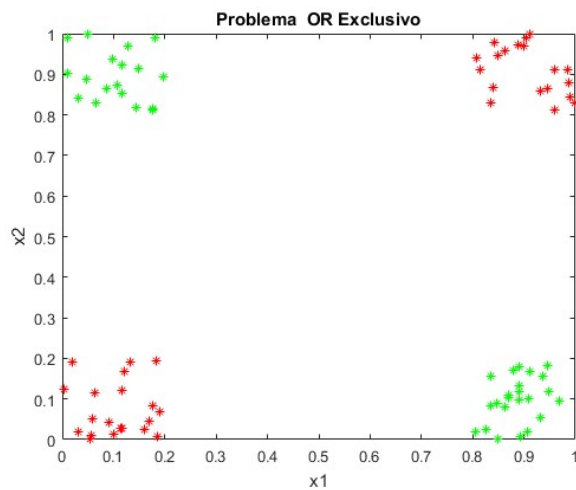
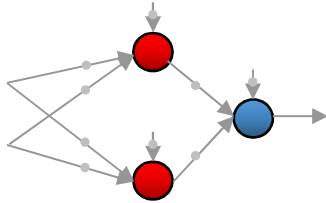
$$W_O = \begin{pmatrix} wo_{10} & wo_{11} & wo_{12} \end{pmatrix}$$



# Redes neuronales artificiales

Ejemplo

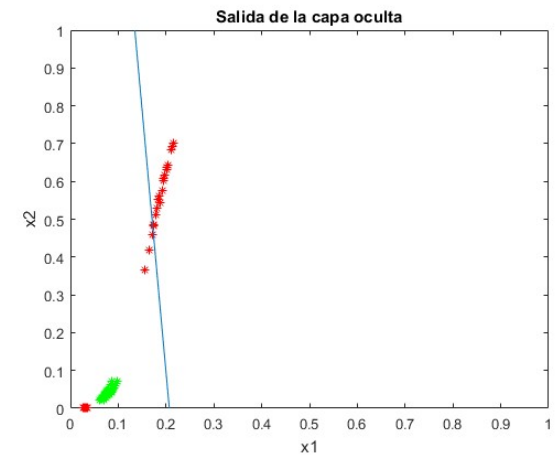
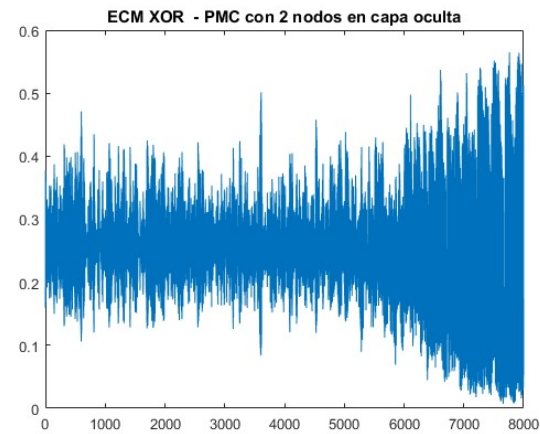
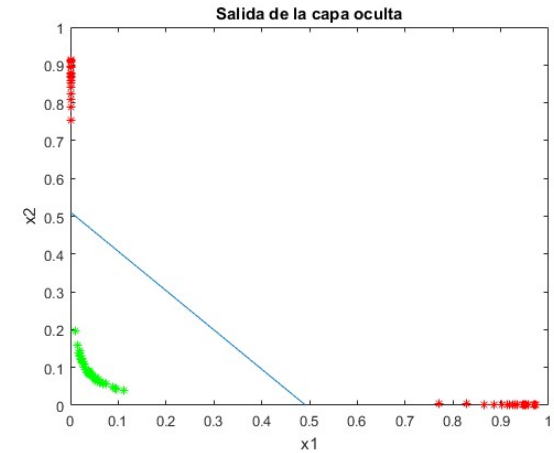
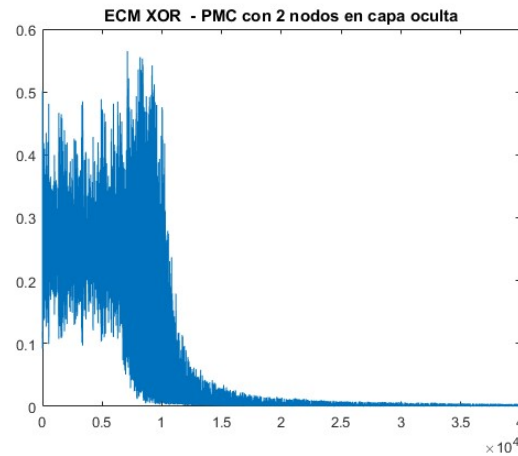
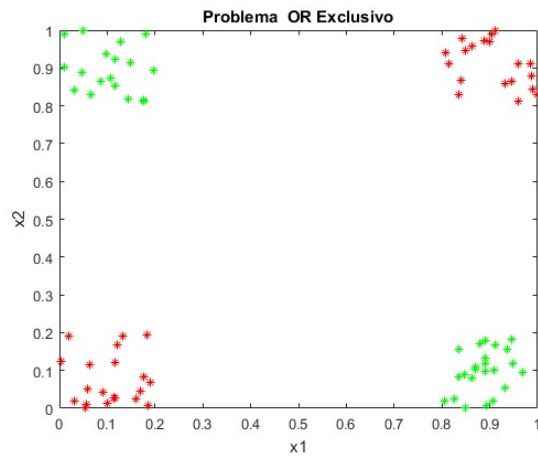
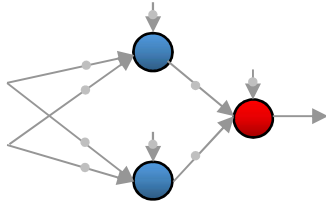
OR exclusivo



# Redes neuronales artificiales

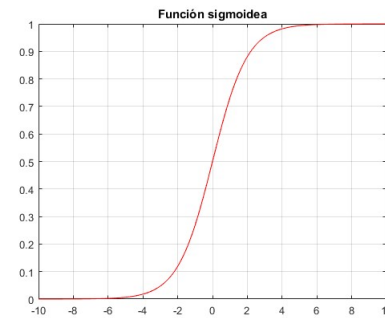
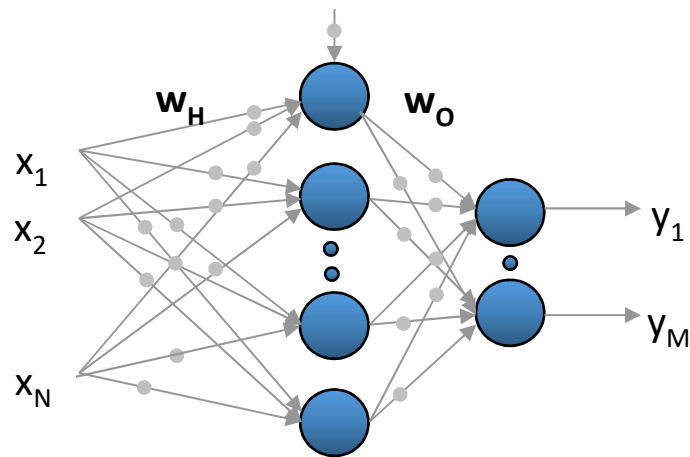
## Ejemplo

## OR exclusivo



# Redes neuronales artificiales

Salida PMC



$$y(\theta) = \frac{1}{1 + e^{-\theta}}$$

$$y = [0.24, 0.81, 0.18]$$

$$y_d = [0, 1, 0]$$

¿  $y = y_d$  ?

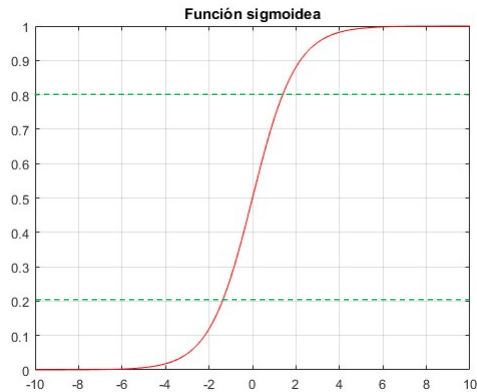
# Redes neuronales artificiales

## Salida PMC – Etapa de testeo

$$y = [0.24, 0.81, 0.18]$$

$$y_d = [0, 1, 0]$$

Umbrales



$$\text{Si } y > 0.8 \rightarrow y = 1$$

$$\text{Si } y < 0.2 \rightarrow y = 0$$

$$y = [0.24, 1, 0]$$

Máximo

$$y = [0.24, 0.81, 0.18]$$

$$y = [0, 1, 0]$$

# Redes neuronales artificiales

## Regiones de decisión


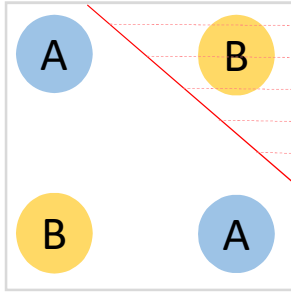
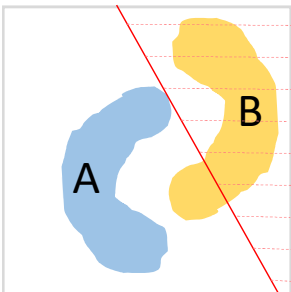
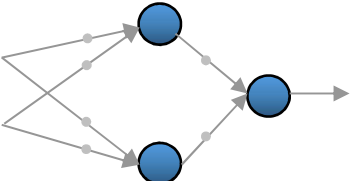
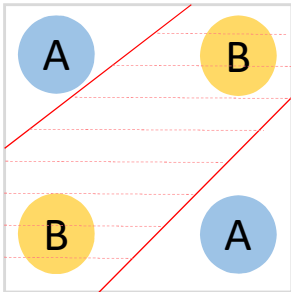
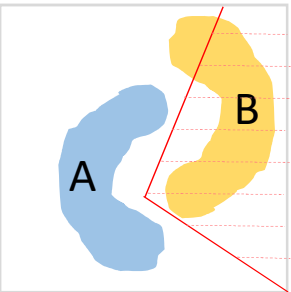
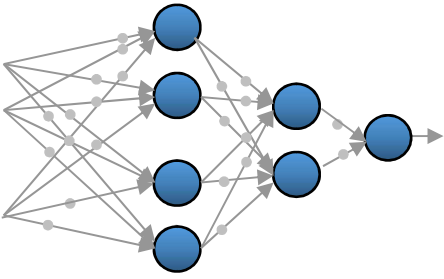
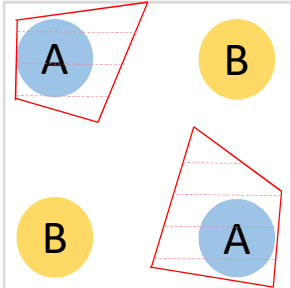
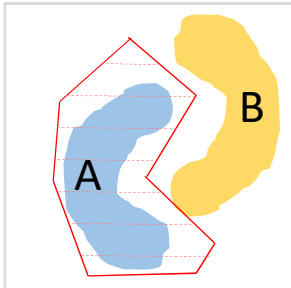
La capacidad para resolver distintos problemas de una red multicapa, viene dada por la nolinealidad que existe en cada uno de los nodos.

Una red con una capa intermedia genera una región de decisión *convexa* que está limitada por la intersección de los hiperplanos generados por cada una de las neuronas de la capa intermedia.

Para generar regiones de decisión *no convexas*, se deben utilizar redes con dos capas intermedias.

# Redes neuronales artificiales

## Regiones de decisión

Estructura	Regiones	OR exclusivo	General
	Hiperplanos		
	Convexas		
	Complejidad arbitraria – No convexas		

# Redes neuronales artificiales

## Término de momento

El algoritmo de aprendizaje provee una aproximación a la trayectoria en el espacio de los pesos, calculada por el método del gradiente descendente.

$$w_{ij}(n + 1) = w_{ij}(n) + \mu \cdot \delta_{pj} \cdot y_{pj}$$

Cuanto más pequeño se toma el parámetro de aprendizaje  $\mu$  más pequeños son los cambios de pesos entre una iteración y la otra, y más suave es la trayectoria en el espacio de los pesos.

En cambio si se aumenta el valor del parámetro  $\mu$ , aumenta la velocidad de convergencia del método pero la red puede resultar inestable.

# Redes neuronales artificiales

## Término de momento

Una manera de aumentar la velocidad de convergencia, sin poner en peligro la estabilidad de la red, es agregar un término a la ecuación de modificación de los pesos.

$$w_{ij}(n+1) = w_{ij}(n) - \alpha \cdot \frac{\partial e(n)}{\partial w} + \mu \cdot \Delta w_{ij}(n-1)$$

$\alpha$  es el parámetro o término de momento y  $\Delta w_{ij}(n-1) = w_{ij}(n-1) - w_{ij}(n-2)$

$$0 \leq \alpha < 1$$



# Redes neuronales artificiales

## Término de momento

Cuando  $\frac{\partial e(n)}{\partial w}$  tiene el mismo signo durante varias iteraciones la variación de los pesos aumenta en magnitud. Por lo tanto, en estas condiciones, la incorporación del término de momento acelera el descenso en el espacio de pesos.

Cuando  $\frac{\partial e(n)}{\partial w}$  tiene signos opuestos en iteraciones consecutivas, la variación de los pesos disminuye, por lo tanto la incorporación del momento genera un factor estabilizante cuando las direcciones oscilan en el signo.

# Redes neuronales artificiales

## Generalización

Concluido el proceso de entrenamiento de una red, es necesario verificar su comportamiento frente a patrones no incluidos en el conjunto de patrones de entrenamiento.

Esta etapa se denomina proceso de test de la red y permite determinar la capacidad de generalización de la red.

Una red se dice que generaliza en forma correcta cuando la relación entrada/salida es satisfactoria para un grupo de patrones diferente al utilizado durante el proceso de entrenamiento.

# Redes neuronales artificiales

## Generalización

## Factores que influyen

- ✓ Tamaño y calidad del conjunto de patrones de entrenamiento.
- ✓ Complejidad del problema a resolver.

# Redes neuronales artificiales

## Generalización

## Factores que influyen

✓ Arquitectura de la red.

Una regla práctica para obtener una buena generalización es utilizar la menor arquitectura de la red que resuelva eficientemente el problema a considerar.

Una primer aproximación sería comenzar con una arquitectura mínima e ir aumentando la misma a medida que los procesos de entrenamiento no verifiquen las condiciones de convergencia.

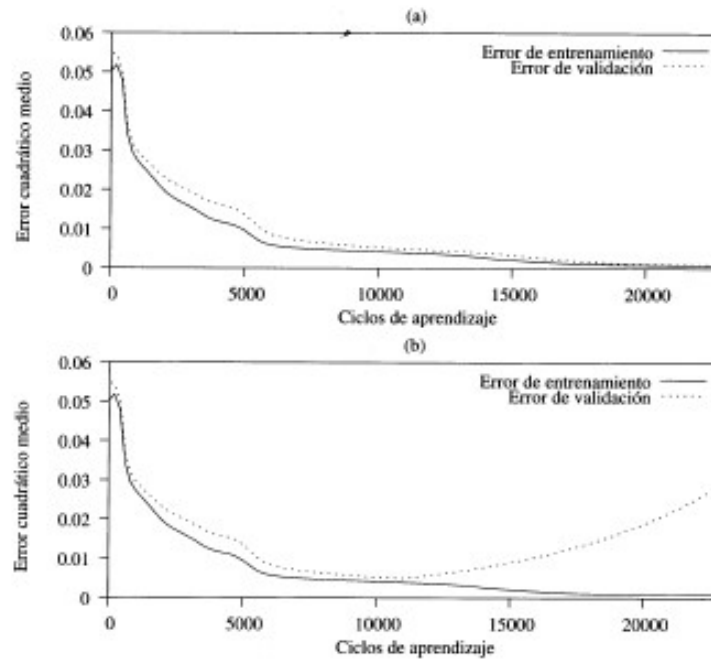
Otra aproximación a una arquitectura de óptima generalización es comenzar con una red sobredimensionada, para reducirla, eliminando los pesos con menor incidencia en la solución.

# Redes neuronales artificiales

Generalización

Factores que influyen

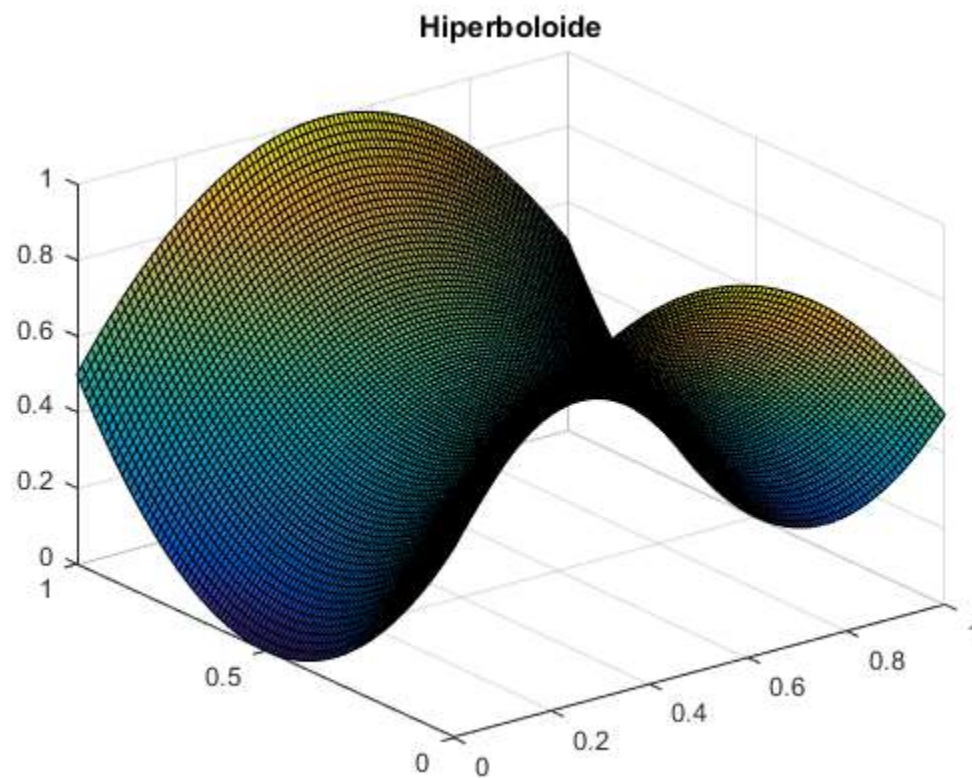
✓ Duración del entrenamiento



# Redes neuronales artificiales

Otras aplicaciones

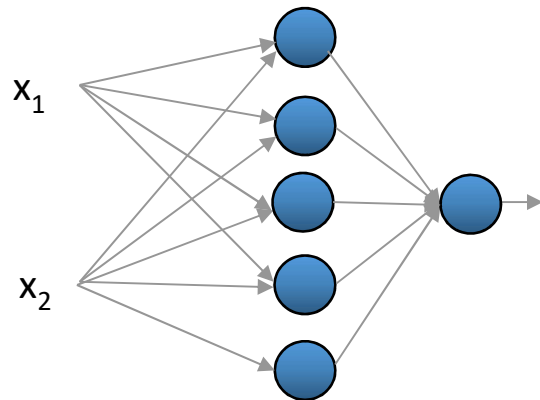
Aproximación de funciones



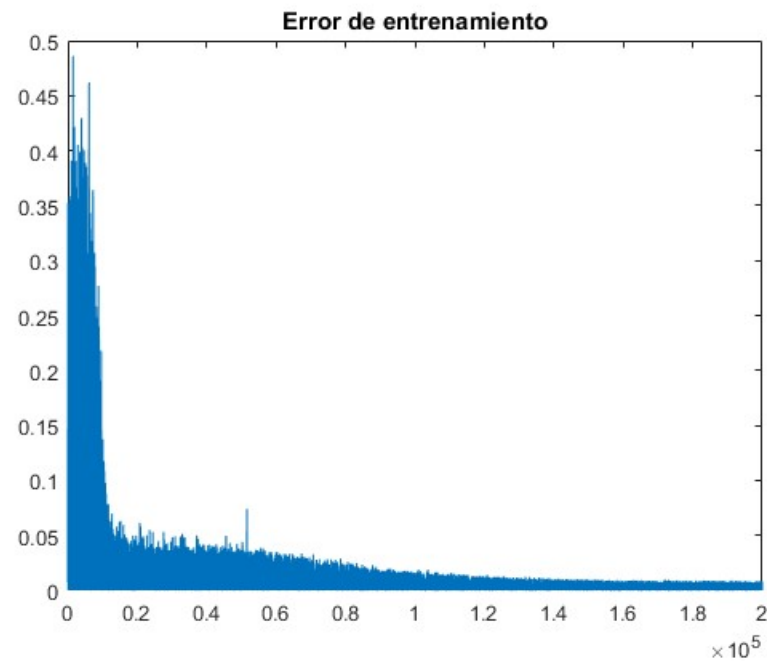
# Redes neuronales artificiales

Otras aplicaciones

Aproximación de funciones



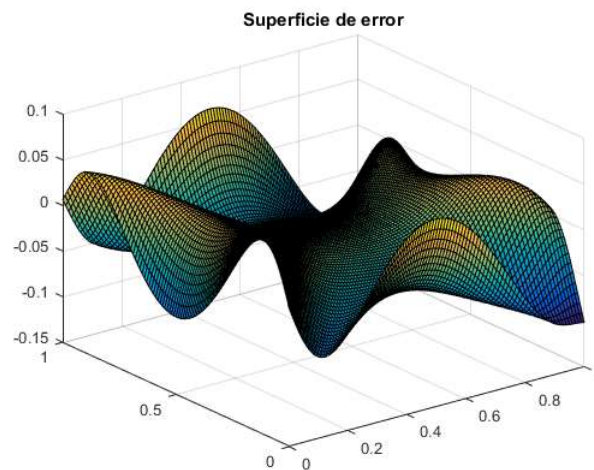
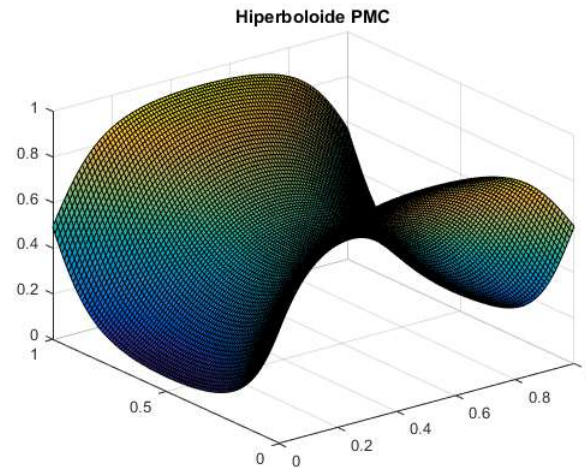
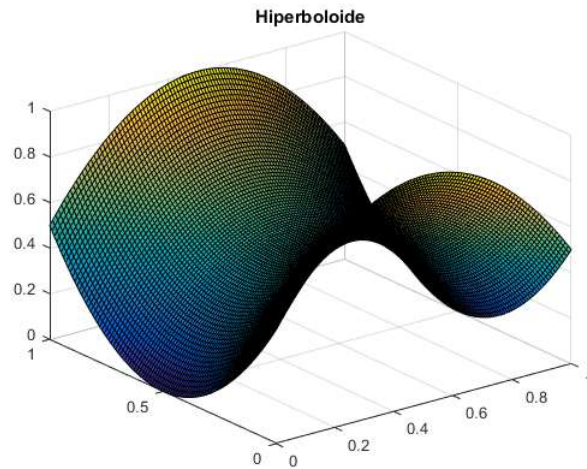
$$f(X): R^2 \rightarrow R^1$$



# Redes neuronales artificiales

Otras aplicaciones

Aproximación de funciones



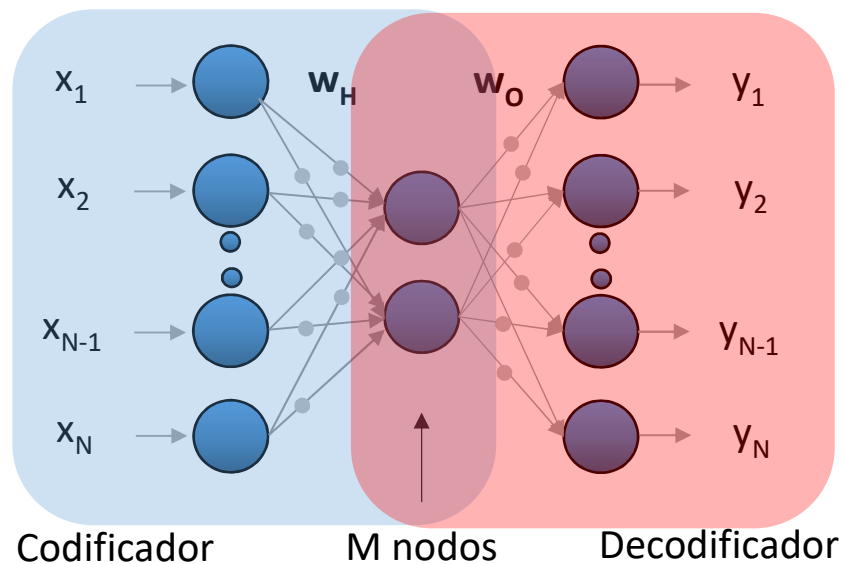


# Redes neuronales artificiales

Otras aplicaciones

Autoencoders

Un autoencoder es una red neuronal utilizada para aprender codificaciones, reducción de dimensionalidad, compresión de información y denosing de imágenes (deep learning).



$$X = Y$$

$$M < N$$

# Redes neuronales artificiales

Fin de la clase!