

Redes Neuronales

Redes multicapa

Perceptrón simple (Diagrama)

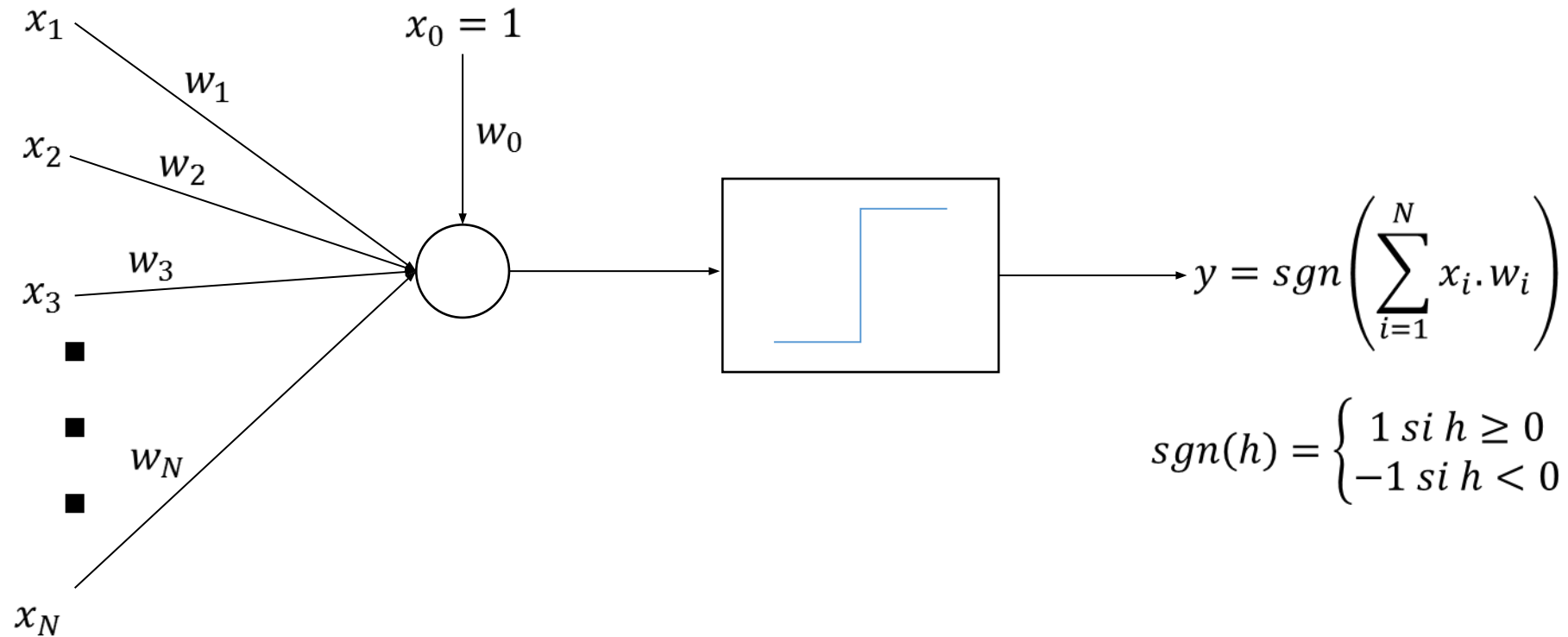


Diagrama del perceptron simple

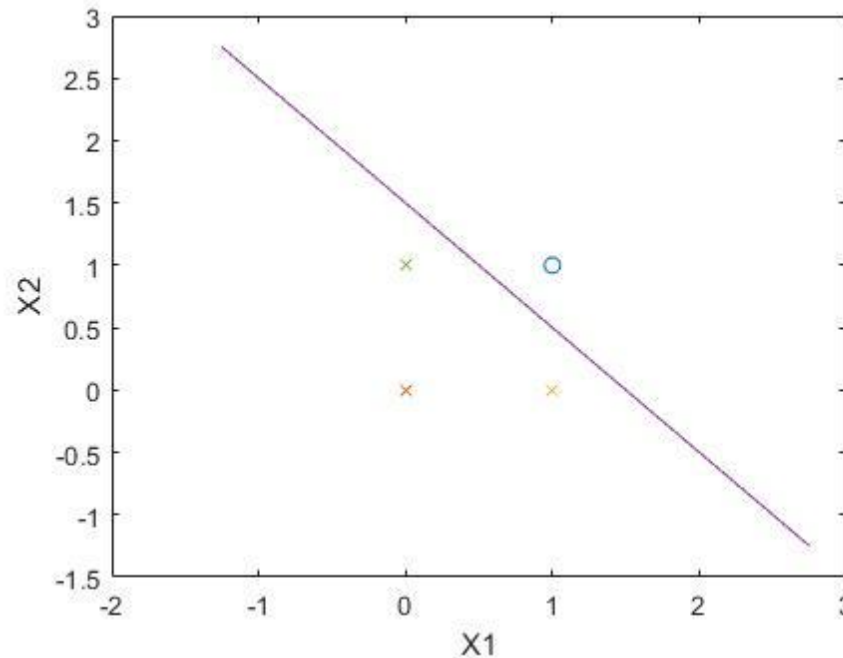
Perceptrón simple

- Es el modelo matemático mas simple de una neurona.
- Los canales $x \in \mathbb{R}$ de entrada corresponden biológicamente a las dendritas. Una de ellas (x_0) se utiliza como bias y su valor es siempre “1”.
- La salida y es una clasificación entre dos clases (1 o -1) y biológicamente corresponde al axón.
- Se utiliza como clasificador de los patrones de entrada, los cuales pueden pertenecer a una clase δ_1 ($y = 1$) o a una clase δ_2 ($y = -1$).
- Para realizar la clasificación se realiza el producto escalar entre el patrón de entrada (\mathbf{x}) y los pesos (\mathbf{w}) y a este resultado se le aplica la función signo. Si los pesos están definidos correctamente, el perceptrón dara a su salida un 1 para los patrones pertenecientes a la clase δ_1 o un -1 para los patrones pertenecientes a la clase δ_{-1} .
- Ejemplo: compuerta AND

Perceptrón simple (Ej: AND)

- Se utiliza para clasificar patrones de entrada en dos tipos. Un ejemplo de ello es la compuerta AND de dos variables de entrada:

X1	X2	Y
0	0	-1(o)
0	1	-1(x)
1	0	-1(x)
1	1	1(x)



La recta corresponde a los pesos:

$$w_0 = -1,5$$

$$w_1 = 1$$

$$w_2 = 1$$

$$x_1 w_1 + x_2 w_2 + w_0 = 0$$

Si $\mathbf{w}^T \mathbf{x} > 0 \rightarrow \in \text{clase } \delta_1$

Si $\mathbf{w}^T \mathbf{x} < 0 \rightarrow \in \text{clase } \delta_{-1}$

Perceptrón simple (Separabilidad lineal)

- La función del perceptrón simple es separar un espacio vectorial en dos subespacios vectoriales, uno cuyos vectores pertenecen a la clase δ_1 y otro cuyos vectores pertenecen a la clase δ_{-1} , a través del hiperplano definido por:

$$\mathbf{w}^T \mathbf{x} = 0$$

- Se dice que un problema de clasificación es linealmente separable cuando existe un hiperplano definido por \mathbf{w} tal que todos los vectores \mathbf{x}^{δ_1} pertenecientes a la clase δ_1 cumplen con:

$$\mathbf{w}^T \mathbf{x}^{\delta_1} > 0$$

- Y todos los vectores $\mathbf{x}^{\delta_{-1}}$ pertenecientes a la clase δ_{-1} cumplen con:

$$\mathbf{w}^T \mathbf{x}^{\delta_{-1}} < 0$$

Perceptron simple (aprendizaje)

- Para un problema linealmente separable, ¿cómo hallo los valores de \mathbf{w} ?
 - - 1) Se selecciona un conjunto P de patrones de entrenamiento \mathbf{x}^μ . A cada uno de ellos le corresponde una salida deseada (clase) δ^μ .
 - 2) Se inicializan los pesos $\mathbf{w}=0$ (o valores aleatorios pequeños con respecto a la potencia de los patrones de entrenamiento).
 - 3) η es una constante y su valor varía entre $0 < \eta < 1$.
 - 4) $error = 0$
 - 5) Para μ desde 1 hasta P:
 - 1) Se presenta el patrón \mathbf{x}^μ a la entrada de la red y se obtiene la salida y
 - 2) Se actualizan los pesos: $w'_i = w^i + \eta(\delta^\mu - y)x_i$
 - 3) Se acumula el error: $error = error + |(\delta - y)|$
 - 6) Si $error > cte$, se vuelve al paso 3, si $error \leq cte$ fin del algoritmo.

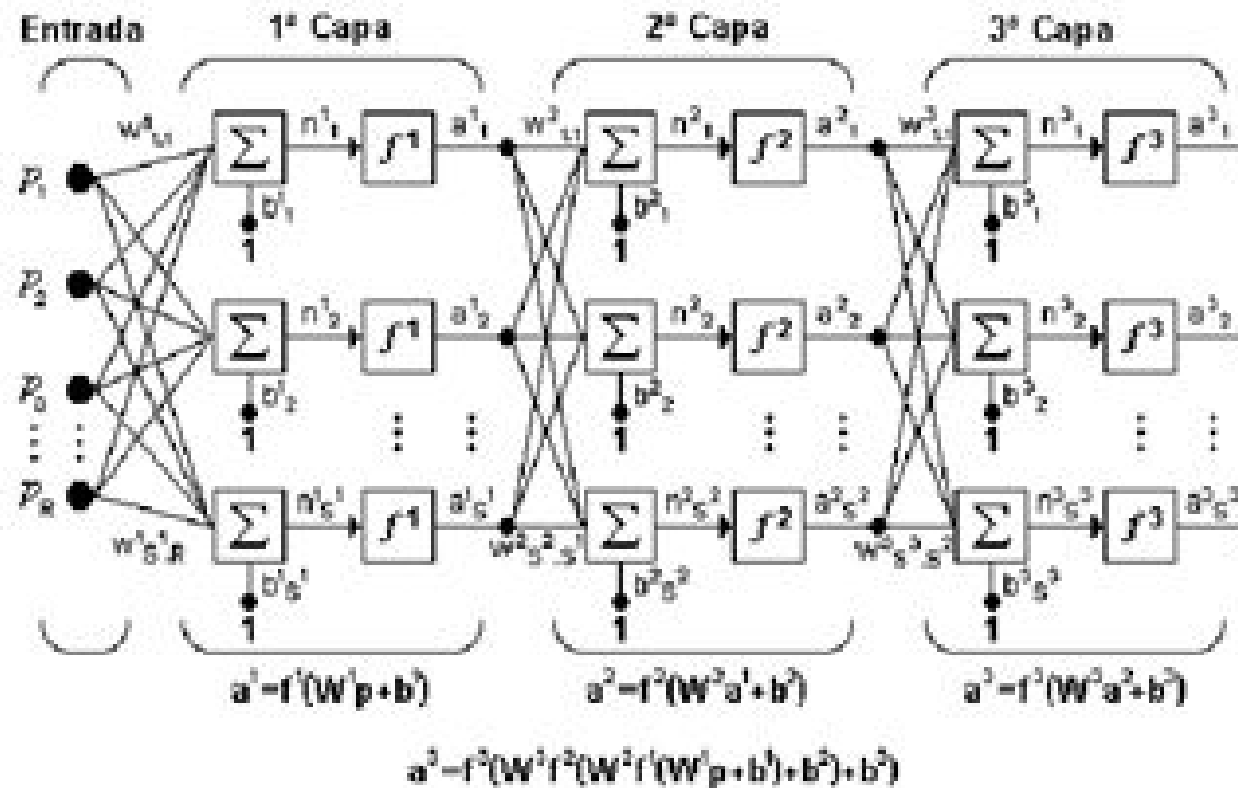
Nótese que en el caso de que la salida sea igual al valor deseado ($\delta = y$), no se actualizan los pesos

- Si el problema es linealmente separable, se puede elegir $cte=0$ para el error.
- Se suele definir una cantidad máxima de veces en la que se puede volver al punto 3, en el caso de que el algoritmo no alcance una solución ($error \leq cte$).

Perceptrón simple (aprendizaje)

- $$w'_i = w_i + \eta(\delta^\mu - y)x_i$$
- η es una constante de entrenamiento.
 - Valores altos hacen que el algoritmo converja mas rápido, con mas riesgo de que el aprendizaje se vuelva inestable.
 - Valores bajos hacen que el algoritmo converja mas lento, pero el aprendizaje se hace mas estable.
 - Se suele buscar una solución de compromiso, según el problema.
 - Se puede hacer que η varíe en el proceso de entrenamiento, comenzando por valores altos para tener un rápido acercamiento a la solución, y luego valores bajos, para tener precisión.
- Si $\delta^\mu = y$ entonces $(\delta^\mu - y)$ es cero y los pesos no se actualizan
- Si $\delta^\mu = 1$ y $y = -1 \rightarrow x_i$. w_i debe crecer $\rightarrow x_i \cdot (w_i + 2\eta x_i) = x_i \cdot w_i + 2\eta x_i^2$
- Si $\delta^\mu = -1$ y $y = 1 \rightarrow x_i$. w_i debe decrecer $\rightarrow x_i \cdot (w_i - 2\eta x_i) = x_i \cdot w_i - 2\eta x_i^2$
- Novikoff (1962) probó que el algoritmo de aprendizaje converge después de un número finito de iteraciones si los datos son separables linealmente. Se suele dar esta demostración ya que expone una forma de representar el estado interno de la red y permite comprender de dónde sale la regla de aprendizaje. Por motivos de tiempo no se dará.


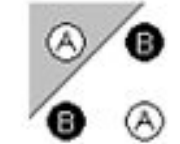
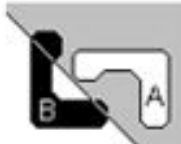

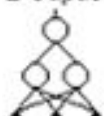
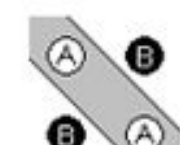






Perceptrón Multicapa



- Cada neurona procesa envía su resultado a las entradas de la capa posterior (funcionamiento feedforward) .
- Establece zonas de decisión mucho mas complejas que el perceptrón simple (2 subespacios complementarios), permitiendo resolver problemas que no sean linealmente separables.

Perceptrón Multicapa

- Las capacidades del perceptrón multicapa con dos y tres capas y con una única neurona en la capa de salida se muestran en la siguiente figura:

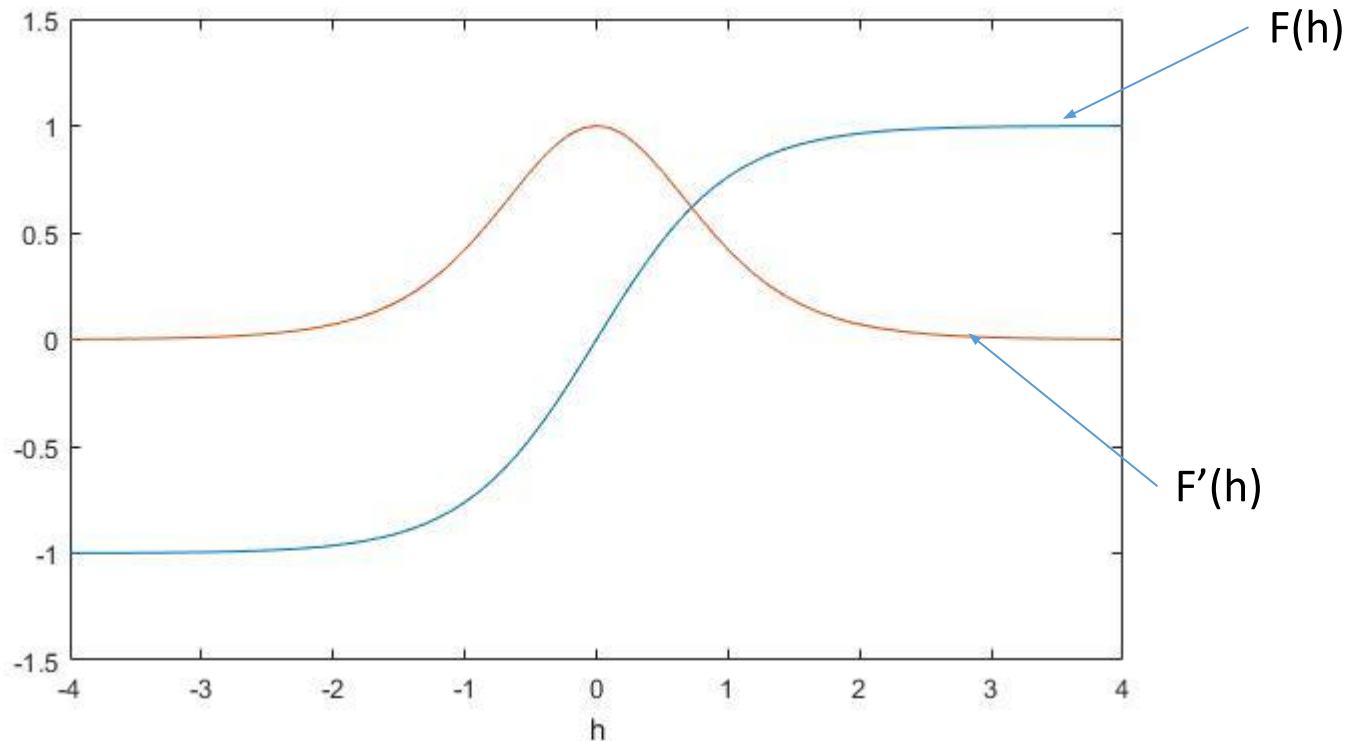
Estructura	Regiones de Decisión	Problema de la XOR	Clases con Regiones Mezcladas	Formas de Regiones más Generales
<p>1 Capa</p> 	Medio Plano Limitado por un Hiperplano			
<p>2 Capas</p> 	Regiones Cerradas o Convexas			
<p>3 Capas</p> 	Complejidad Arbitraria Limitada por el Número de Neuronas			

Perceptrón multicapa

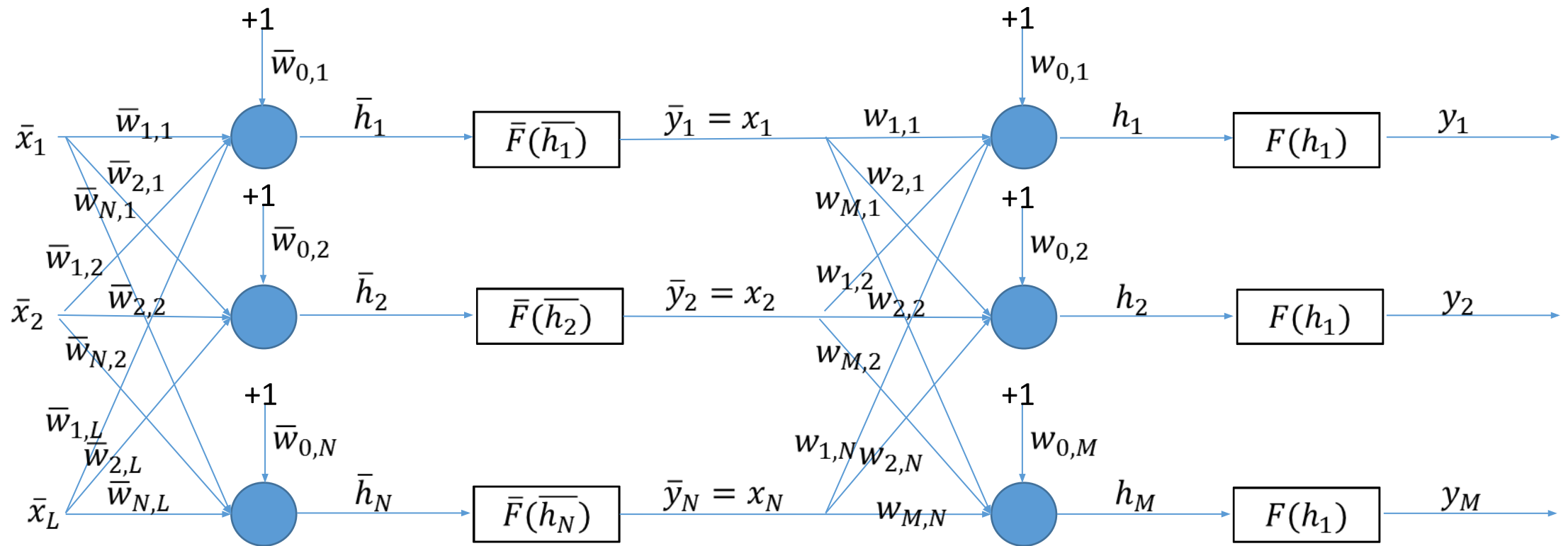
- Derivaremos la regla de aprendizaje para el perceptron multicapa
- El procedimiento consiste en:
 - Calcular el error cuadrático instantáneo para el número de iteración n : $e^2(n)$
 - Hacer la derivada parcial con respecto a cada uno de los pesos (gradiente)
 - Modificar el vector de pesos en dirección opuesta al vector gradiente.
 - Para poder calcular $\frac{\partial e^2(n)}{\partial w_i}$, necesitamos que la función de activación sea derivable. Es por ello que en este tipo de redes se suele utilizar como función transferencia una función sigmoidea (tanh) en vez de la función signo.
 - Existen otras funciones de activación derivables, pero la función sigmoidea además tiene la particularidad de que su derivada es fácilmente calculable

Función sigmoidea

- $F(h) = \tanh(h)$
- $F'(h) = 1 - \tanh^2(h) = 1 - F^2(h)$



Derivación de la regla de aprendizaje



Derivación de la regla de aprendizaje

- Definimos el error de la salida como:

$$e = \sum_{m=1}^M e_m^2$$

donde:

$$e_m^2 = (\delta_m - y_m)^2$$

Nos interesa hallar el valor de $\frac{\partial e}{\partial w_{ij}}$ para luego modificar los w_{ij} en dirección contraria a $\frac{\partial e}{\partial w_{ij}}$

Derivación de la regla de aprendizaje

- Para los pesos w_{ij} de la capa de salida:

$$\frac{\partial e}{\partial w_{ij}} = \frac{\partial \sum_{m=1}^M e_m^2}{\partial w_{ij}} = \frac{\partial e_i^2}{\partial w_{ij}} = \frac{\partial e_i^2}{\partial e_i} \frac{\partial e_i}{\partial w_{ij}} = 2e_i \frac{\partial e_i}{\partial w_{ij}} = 2e_i \frac{\partial (\delta_i - y_i)}{\partial w_{ij}} =$$

$$2e_i \frac{\partial (-y_i)}{\partial w_{ij}} = -2e_i \frac{\partial (y_i)}{\partial h_i} \frac{\partial (h_i)}{\partial w_{ij}} = -2e_i F'(h_i) \frac{\partial (\sum_{n=1}^N x_n w_{i,n})}{\partial w_{ij}} = -2e_i F'(h_i) x_j$$

- Por lo tanto, para la capa de salida:

$$\Delta w_{ij} = \eta e_i F'(h_i) x_j$$

Derivación de la regla de aprendizaje

- Para la capa oculta:

$$\frac{\partial e}{\partial \bar{w}_{nl}} = \frac{\partial \sum_{m=1}^M e_m^2}{\partial \bar{w}_{nl}} = \sum_{m=1}^M \frac{\partial e_m^2}{\partial \bar{w}_{nl}}$$

$$\frac{\partial e_m^2}{\partial \bar{w}_{nl}} = \frac{\partial e_m^2}{\partial e_m} \frac{\partial e_m}{\partial \bar{w}_{nl}} = 2e_m \frac{\partial e_m}{\partial \bar{w}_{nl}} = 2e_m \frac{\partial e_m}{\partial h_m} \frac{\partial h_m}{\partial \bar{w}_{nl}} = -2e_m F'(h_m) \frac{\partial h_m}{\partial \bar{w}_{nl}}$$

$$\frac{\partial h_m}{\partial \bar{w}_{nl}} = \frac{\partial h_m}{\partial x_n} \frac{\partial x_n}{\partial \bar{w}_{nl}} = w_{mn} \frac{\partial \bar{y}_n}{\partial \bar{w}_{nl}} = w_{mn} \frac{\partial \bar{y}_n}{\partial \bar{h}_n} \frac{\partial \bar{h}_n}{\partial \bar{w}_{nl}} = w_{mn} \bar{F}'(h_n) \frac{\partial \sum_{i=0}^L \bar{x}_i \bar{w}_{ni}}{\partial \bar{w}_{nl}}$$

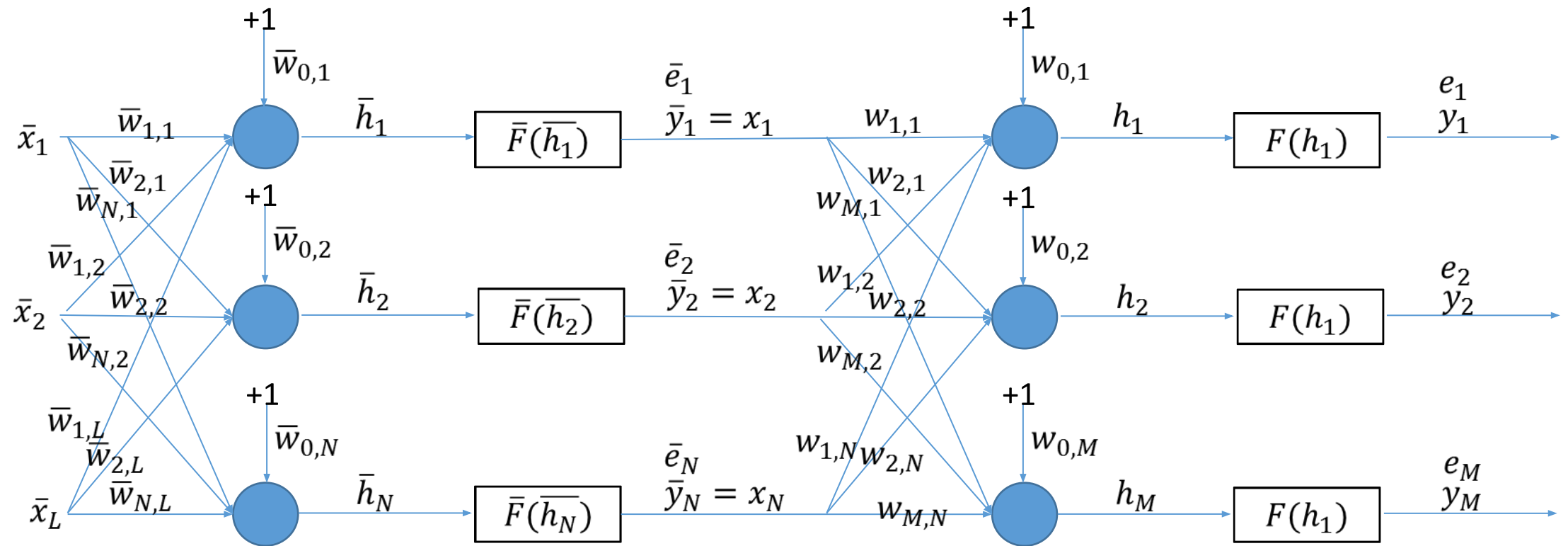
$$\frac{\partial h_m}{\partial \bar{w}_{nl}} = w_{mn} \bar{F}'(h_n) \bar{x}_l$$

Derivación de la regla de aprendizaje

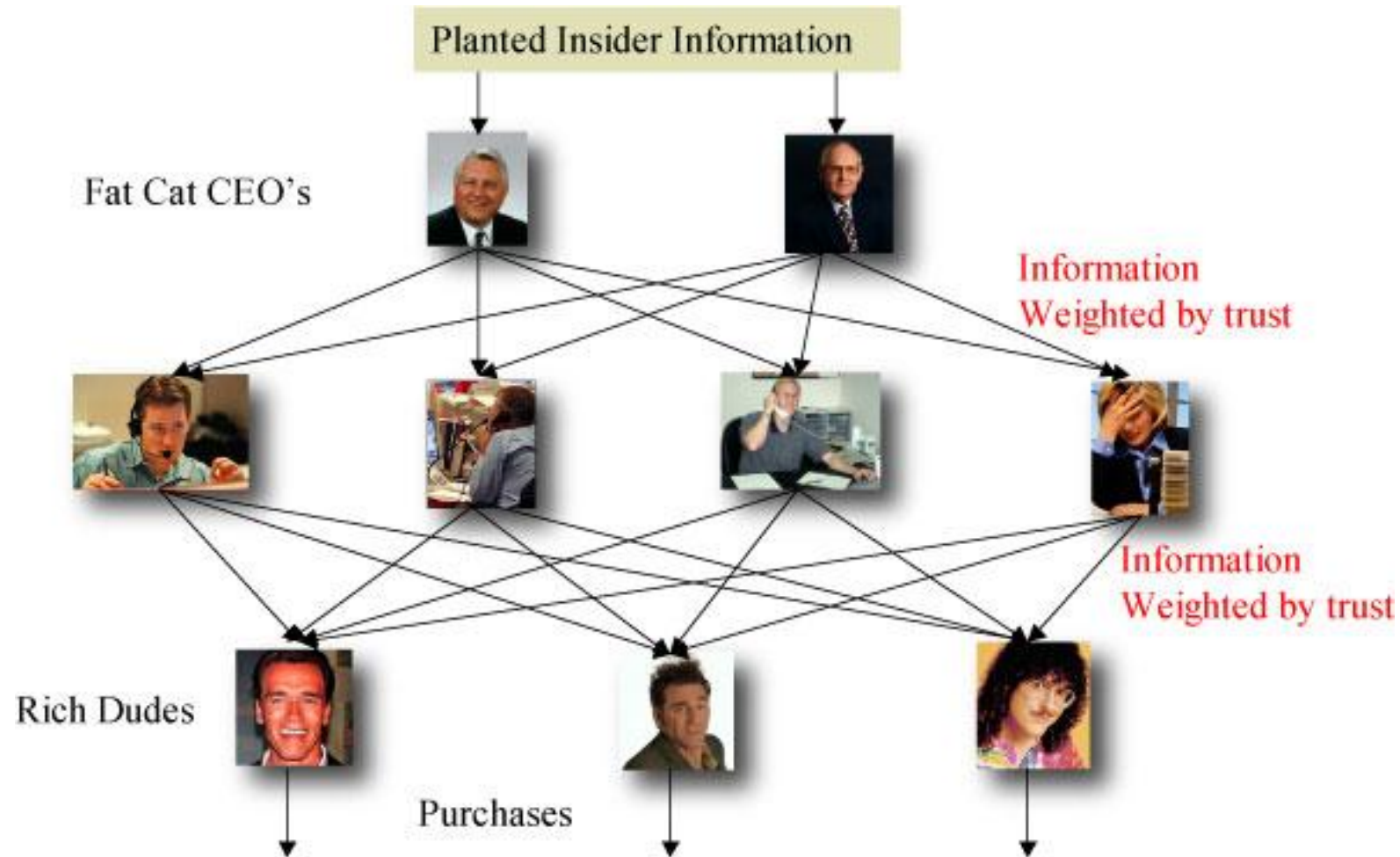
- $$\frac{\partial e}{\partial \bar{w}_{nl}} = \frac{\partial \sum_{m=1}^M e_m^2}{\partial \bar{w}_{nl}} = -2\bar{F}'(h_n)\bar{x}_l \sum_{m=1}^M e_m F'(h_m) w_{mn}$$
$$\frac{\partial e}{\partial \bar{w}_{nl}} = -2\bar{F}'(h_n)\bar{x}_l \bar{e}_n$$

Al término $\bar{e}_n = \sum_{m=1}^M e_m F'(h_m) w_{mn}$ se lo conoce como propagación hacia atrás del error (backpropagation) y puede ser propagado indefinidamente considerando la capa oculta como capa de salida cuyos errores e_m son ahora los \bar{e}_n

Derivación de la regla de aprendizaje



Interpretación de la red



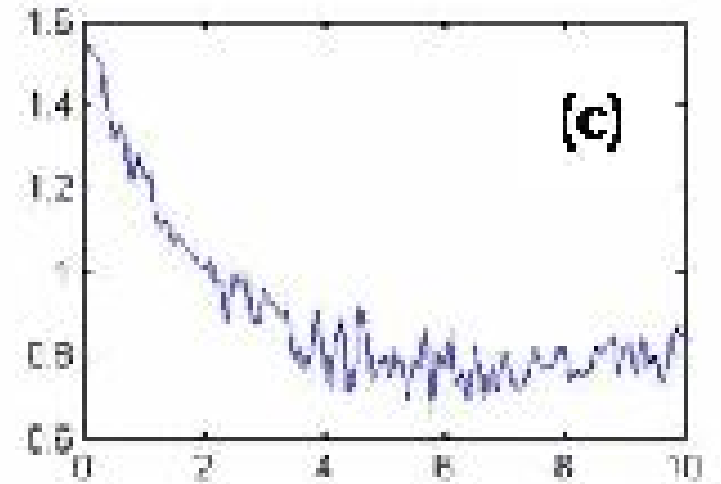
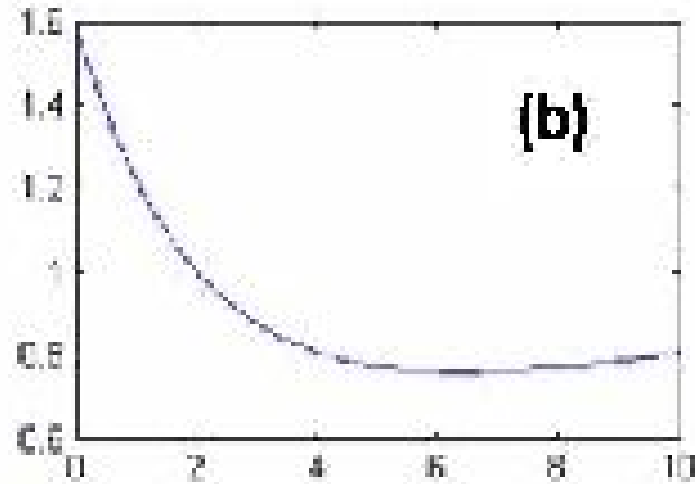
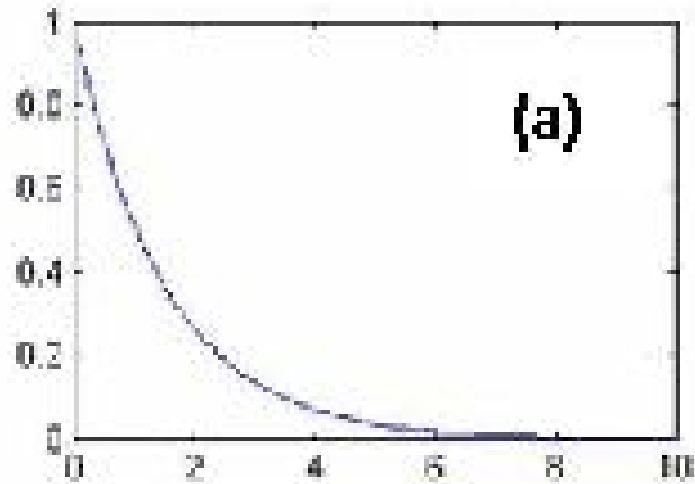
Intepretación

- El uso de sigmoideas hace que sea posible tener un grado de seguridad con el que el perceptrón nos da su resultado
- Cuando una función de activación nos da un resultado cercano a 1 o -1, su derivada se acercará a cero y ese peso será modificado muy poco.
- Para funciones de activación cuya salida tiende a cero, la derivada tiende a uno, generando una variación mayor de sus pesos.
- Esto hace que la red tenga memoria, y

Sobreentrenamiento

- Ocurre cuando entreno por demás a la red y esta se ajusta muy bien a los patrones de entrenamiento y no a los patrones de testeo.
- Cuando comienzo a entrenar la red, el error baja tanto para los patrones de entrenamiento como para los patrones de test.
- A partir de cierto punto, el error para los patrones de entrenamiento sigue bajando, mientras que el error para los patrones de test empieza a aumentar.
- El sobreentrenamiento conlleva la pérdida de capacidad de generalización por parte de la red.
- Una forma de evitarlo es medir el error para los patrones de test periódicamente mientras se entrena, y cuando el error para los patrones de test comienza a subir, cortar el entrenamiento.

Sobreentrenamiento



- a) Curva de error teórica para los patrones de entrenamiento
- b) Curva de error teórica para los patrones de testeo
- c) Curva de error real para los patrones de testeo -> Se debe confeccionar un criterio de corte.