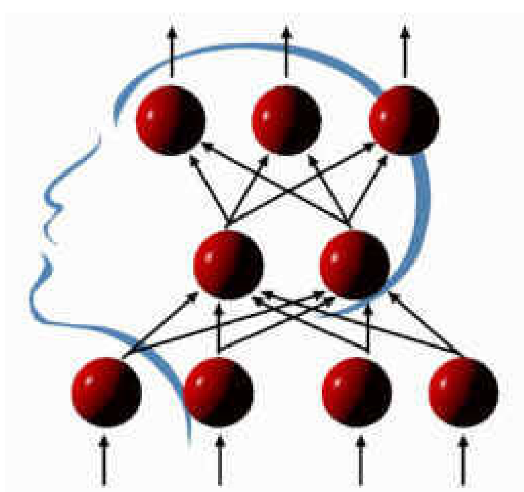
Procesamiento Digital de Imágenes

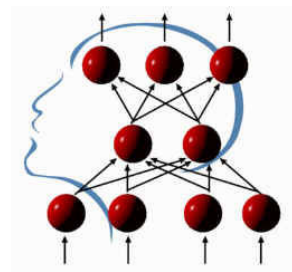
Pablo Roncagliolo B.
Nº 21

Redes Neuronales



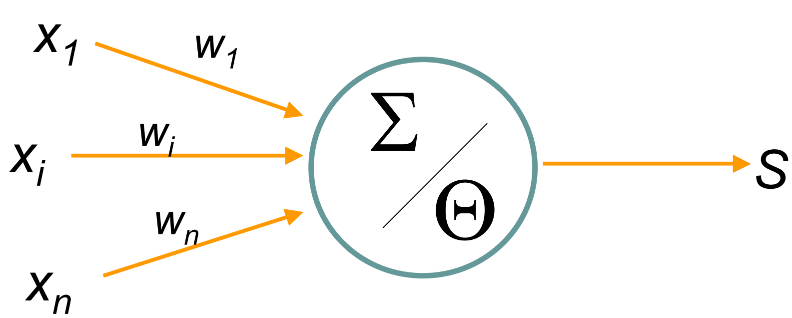
Redes Neuronales Básicas

- **Células de McCulloch&Pitts**
- **El Perceptrón**
- **ADALINE**
- **El Perceptrón Multicapa**



Celulas de McCulloch-Pitts

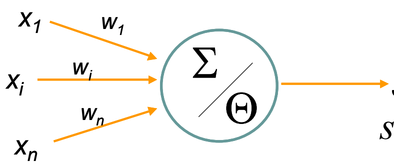
- 1943. Fueron un modelo simplificado del funcionamiento de las neuronas del cerebro.
- Cada célula puede tener dos estados de salida, 0 ó 1.



Celulas de McCulloch-Pitts

- Las células operan en lapsos discretos.

- Una red neuronal de células de McCulloch-Pitts tiene la capacidad de computo universal. Es decir, cualquier estructura que pueda ser programada en un computador, puede ser modelada con este tipo de redes.

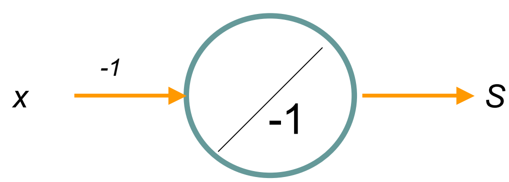


$$s(t+1)=\begin{cases}1 & \sum_i w_i x_i(t) > \theta \\ 0 & \textit{...en caso contrario}\end{cases}$$

- Sin embargo, el tamaño de estas redes para problemas complejos es muy elevado. Además el método de aprendizaje para redes muy grandes no es apropiado.

Celulas de McCulloch-Pitts

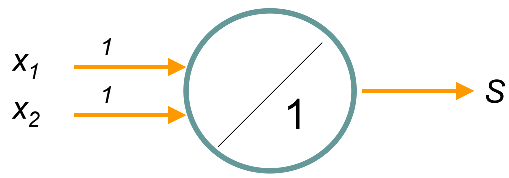
- Ejemplo: NOT



x_1	$\sum x_i w_i$	S
0	0	1
1	-1	0

Celulas de McCulloch-Pitts

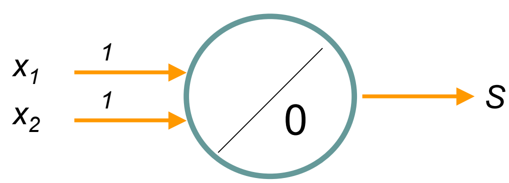
➢Ejemplo: AND



x_1	x_2	$\sum x_i w_i$	S
0	0	0	0
0	1	1	0
1	0	1	0
1	1	2	1

Celulas de McCulloch-Pitts

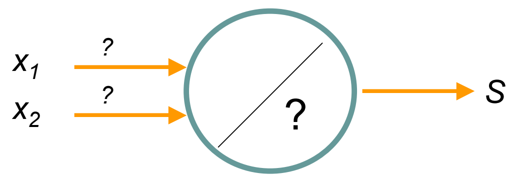
➢Ejemplo: OR



x_1	x_2	$\sum x_i w_i$	S
0	0	0	0
0	1	1	1
1	0	1	1
1	1	2	1

Celulas de McCulloch-Pitts

➢XOR ??? Con una celula no es posible.



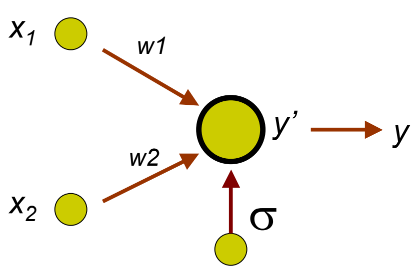
x_1	x_2	$\sum x_i w_i$	S
0	0		0
0	1		1
1	0		1
1	1		0

El Perceptrón

➢Rosenblat generalizó las células de McCulloch-Pitts

➢Se concibió como un sistema capaz de realizar tareas de clasificación de forma automática.

➢La idea era disponer de un sistema que a partir de un conjunto de ejemplos (patrones) de clases diferentes, fuera capaz de determinar las ecuaciones de las superficies que hacían de frontera de dichas clases.



$$y' = \sum_{i=1}^{n} w_i x_i$$

$$y = F(y', \sigma)$$

$$F(s,\sigma) = \begin{cases} 1 & si\, s > \sigma \\ -1 & en\ caso\ contrario \end{cases}$$

El Perceptrón

➢Se puede expresar la misma ecuación considerando SIGMA como parte de la sumatoria de entrada a la función:

$$y = F\left(\sum_{i=1}^{n} w_i x_i + \sigma\right)$$

$$F(s,\sigma) = \begin{cases} 1 & si\, s > 0 \\ -1 & en\ caso\ contrario \end{cases}$$

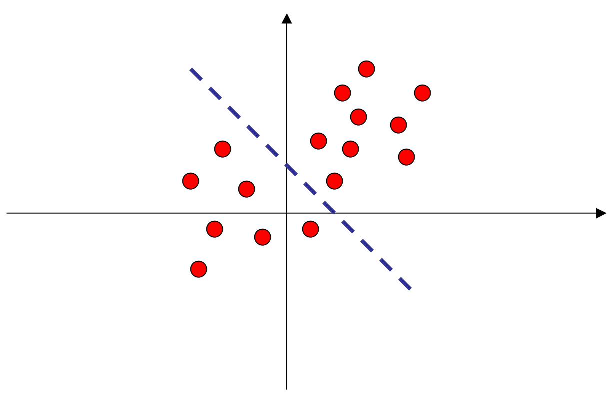
➢Ej. Para dos entradas:

$$y = F(w_1 x_1 + w_2 x_2 + \sigma)$$

➢Se observa que el umbral que separa las dos respuestas de la red 1 y –1, corresponde a una recta con pendiente –w1/w2 e intercepto -σ/w2

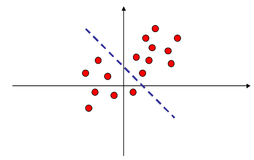
El Perceptrón

➢Graficamente, la separación de las dos clases:



El Perceptrón

➢En el caso general sería:



➢Dado el conjunto de puntos A=(a1,a2....an) y B=(b1,b2....bn). Obtener el conjunto W=(w1,w2....wn) tal que:

$$\forall \vec{a} \in A : w_1 a_1 + ... + w_n a_n + \sigma > 0$$

$$y$$

$$\forall \vec{b} \in B : w_1 b_1 + ... + w_n b_n + \sigma > 0$$

➢Esta es la base del aprendizaje del PERCEPTRON.

El Perceptrón

➢El proceso de aprendizaje:

$$Sea:$$

$$d(x) = clase\ del\ vector\ x = \begin{cases} 1 \\ -1 \end{cases}$$

➢PASO 0: Comenzar con valores aleatorios para pesos y umbral.

➢PASO 1: Seleccionar un ejemplo X del conjunto de entrenamiento.

➢PASO 2: Si y<>d(x), modificar w_i de acuerdo con:

$$\Delta w_i = d(x) x_i$$

➢PASO 3: Si no se ha cumplido el criterio de finalización, volver a **1**

El Perceptrón

➢El proceso de aprendizaje:

$$\Delta w_i = d(x) x_i$$

➢Se observa que si la salida y=d(x)=1 para un vector x de clase –1, entonces

$$\Delta w_i = -x_i$$

➢El delta W es proporcional al nodo de entrada y en la dirección de clasificación del vector x.

El Perceptrón

$$y = F(w_1 x_1 + w_2 x_2 + \sigma)$$

$$F(s,\sigma) = \begin{cases} 1 & si\ s > 0 \\ -1 & en\ caso\ contrario \end{cases}$$

1

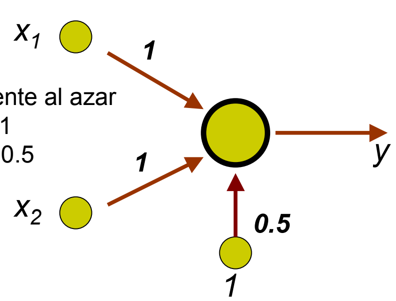
➢Ejemplo: AND

x_1	x_2	AND
0	0	$0(A)$
1	0	$0(A)$
0	1	$0(A)$
1	1	$1(B)$

Patron	*Salida*	*Clasifica*
$(0,0 \mid 0)$	$+1$	*Mal*

Actualizo pesos

Inicialmente al azar
w1=w2=1
Umbral=0.5



$\Delta w_i = d(x) x_i$

$w_1 = w_1 + \Delta w_1 = 1 + (-1) * 0 =$ **1**

$w_2 = w_2 + \Delta w_2 = 1 + (-1) * 0 =$ **1**

$w_0 = w_0 + \Delta w_0 = 0.5 + (-1) * 1 =$ **-0.5**

El Perceptrón

$$y = F(w_1x_1 + w_2x_2 + \sigma)$$

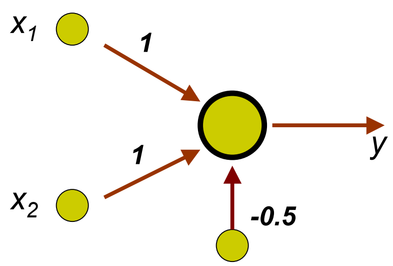
$$F(s,\sigma) = \begin{cases} 1 & si\ s > 0 \\ -1 & en\ caso\ contrario \end{cases}$$

2

➢Ejemplo: AND

x_1	x_2	AND
0	0	$0(A)$
1	0	$0(A)$
0	1	$0(A)$
1	1	$1(B)$

Patron	*Salida*	*Clasifica*
$(0,1 \mid 0)$	$+1$	*Mal*



$\Delta w_i = d(x) x_i$

$w_1 = w_1 + \Delta w_1 = 1 + (-1)*0 = 1$

$w_2 = w_2 + \Delta w_2 = 1 + (-1)*1 = 0$

$w_0 = w_0 + \Delta w_0 = -0.5 + (-1)*1 = -1.5$

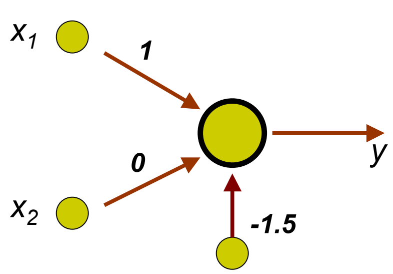
El Perceptrón

3

➢Ejemplo: AND

x_1	x_2	AND
0	0	$0(A)$
1	0	$0(A)$
0	1	$0(A)$
1	1	$1(B)$

Patron	*Salida*	*Clasifica*
$(1,0 \mid 0)$	-1	*Bien*
$(1,1 \mid 1)$	-1	*Mal*



$\Delta w_i = d(x) x_i$

$w_1 = w_1 + \Delta w_1 = 1 + (+1)*1 = 2$

$w_2 = w_2 + \Delta w_2 = 0 + (+1)*1 = 1$

$w_0 = w_0 + \Delta w_0 = -1.5 + (+1)*1 = -0.5$

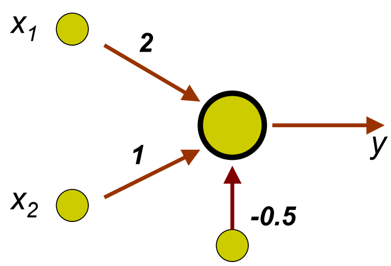
El Perceptrón

4

➢Ejemplo: AND

x_1	x_2	AND
0	0	$0(A)$
1	0	$0(A)$
0	1	$0(A)$
1	1	$1(B)$

Patron	*Salida*	*Clasifica*
$(1,1\|1)$	$+1$	*Bien*
$(0,0\|0)$	-1	*Bien*
$(0,1\|0)$	$+1$	*Mal*



$\Delta w_i = d(x)x_i$

$w_1 = w_1 + \Delta w_1 = 2 + (-1)*0 = 1$

$w_2 = w_2 + \Delta w_2 = 1 + (-1)*1 = 0$

$w_0 = w_0 + \Delta w_0 = -0.5 + (-1)*1 = -1.5$

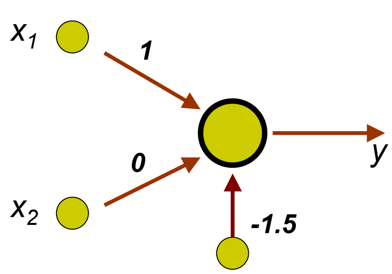
El Perceptrón

5

➢Ejemplo: AND

x_1	x_2	AND
0	0	$0(A)$
1	0	$0(A)$
0	1	$0(A)$
1	1	$1(B)$

Patron	*Salida*	*Clasifica*
$(1,0\|0)$	$+1$	*Mal*



$\Delta w_i = d(x)x_i$

$w_1 = w_1 + \Delta w_1 = 1 + (-1)*1 = 0$

$w_2 = w_2 + \Delta w_2 = 0 + (-1)*0 = 0$

$w_0 = w_0 + \Delta w_0 = -1.5 + (-1)*1 = -2.5$

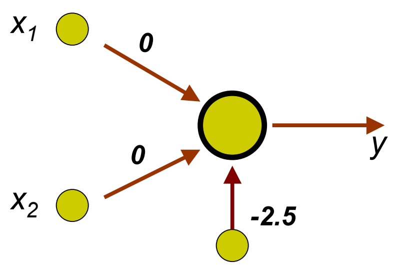
El Perceptrón

6

➢Ejemplo: AND

x_1	x_2	AND
0	0	$0(A)$
1	0	$0(A)$
0	1	$0(A)$
1	1	$1(B)$

$Patron$	$Salida$	$Clasifica$
$(1,1\|1)$	-1	Mal



$\Delta w_i = d(x) x_i$

$w_1 = w_1 + \Delta w_1 = 0 + (+1)*1 = 1$

$w_2 = w_2 + \Delta w_2 = 0 + (+1)*1 = 1$

$w_0 = w_0 + \Delta w_0 = -2.5 + (+1)*1 = -1.5$

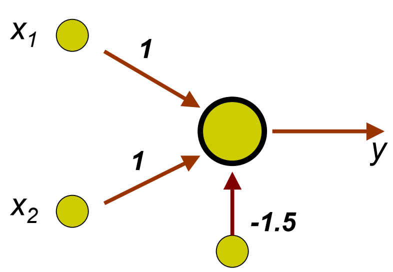
El Perceptrón

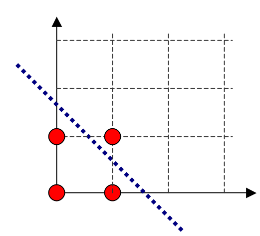
7

➢Ejemplo: AND

x_1	x_2	AND
0	0	$0(A)$
1	0	$0(A)$
0	1	$0(A)$
1	1	$1(B)$

$Patron$	$Salida$	$Clasifica$
$(0,0\|0)$	-1	$Bien$
$(0,1\|0)$	-1	$Bien$
$(1,0\|0)$	-1	$Bien$
$(1,1\|1)$	$+1$	$Bien$

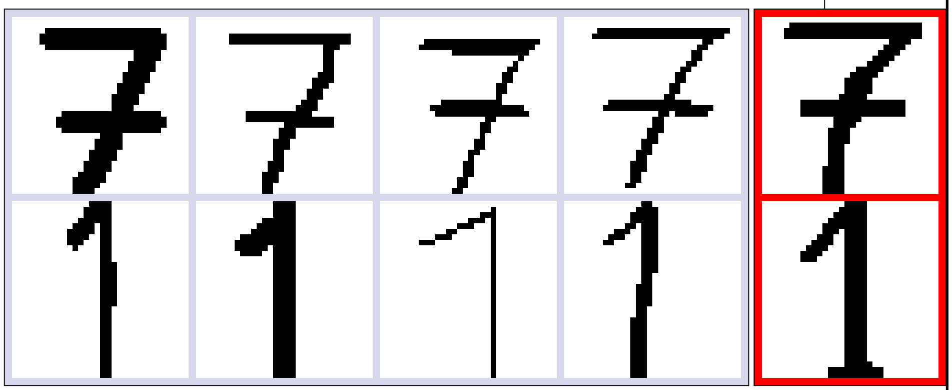




$$x_1 \cdot 1 + x_2 \cdot 1 - 1.5 = 0 \quad \Rightarrow \quad x_2 = -x_1 - 1.5$$

El Perceptrón

➢Ejemplo práctico...



Set de ejemplos durante el entranamiento

Ejemplos fuera del entrenamiento para verificar capacidad de generalización

El Perceptrón

➢Paso 1

Respuestas correctas para los 8 ejemplos de entrenamiento

```
clear;

map1=(0:255)/255;
map=[map1' map1' map1'];
figure(1);
COLORMAP(map);

CLASE=[1 1 1 1 -1 -1 -1 -1];

PESOS=rand(1,1024);
UMB=rand(1,1);
```

$$y = F\left(\sum_{i=1}^{n} w_i x_i + \sigma\right)$$

El Perceptrón

El ciclo while se debería realizar hasta que todos los ejemplos sean clasificados correctamente de manera consecutiva.

➢Paso 1

Lee imagen 32x32

Transforma en vector de 1024

$$y = F\left(\sum_{i=1}^{n} w_i x_i + \sigma\right)$$

```
index=1;
i=0;
while i<200
  A=double(imread(['f' num2str(index) '.bmp']));
  subplot(2,1,1); image(A);

  CAPA_ENTRADA=[];
  for f=1:32, CAPA_ENTRADA=[CAPA_ENTRADA A(f,:)];end;

  Y=sum(CAPA_ENTRADA.*PESOS+UMB);
  if (Y>0 & CLASE(index)<0) | (Y<0 & CLASE(index)>0)
      disp('ERROR')
      PESOS=PESOS+CLASE(index)*CAPA_ENTRADA;
      UMB=UMB+CLASE(index);
  end;

  index=index+1;
  if index>8, index=1; end;
  i=i+1;
end;
```

$w_i = w_i + \Delta w_i = w_i + d(x) x_i$

25

El Perceptrón ➔ ADALINE

➢ADALINE (ADAptive LInear NEuron): Neuron Lineal Adaptativa

- ➢La salida del perceptrón en binaria.
- ➢La regla de aprendizaje del perceptrón no mide el grado de "error".
- ➢Widrow & Hoff, 1960 proponen ADALINE.
  - ➢Consiste simplemente en una transformación que permite adaptar una entrada X a una salida Y.

$$\vec{y} = \sum_{i=1}^{n} w_i x_i + \sigma$$

prb@2007

26

ADALINE

➢La regla de aprendizaje de ADALINE considera el error entre la salida lograda **y** versus la salida deseada **d**

$$\left| \vec{d} - \vec{y} \right|$$

➢Esta regla se conoce como REGLA DELTA

$$\Delta w_i = \alpha \sum_{\forall p} (d_p - y_p) x_i$$

➢La constante α se denomina TASA DE APRENDIZAJE

ADALINE

➢ Al igual que en el perceptrón los pasos son:

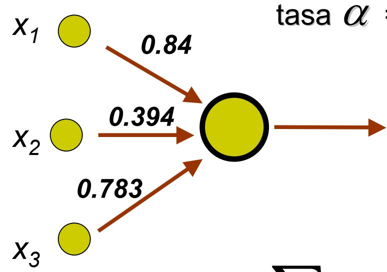
1. Inicializar los pesos en forma aleatoria
2. Introducir PATRON de entrada
3. Calcular salida Y, y obtener diferencia $\sum_{\forall p} (d_p - y_p)$
4. Para todos los pesos, multiplicar dicha diferencia por la entrada correspondiente y ponderarla por la tasa α
5. Actualizar todos los pesos $w_i = w_i + \Delta w_i$
6. Si no se ha cumplido el criterio de convergencia, regresar a **2**. Si se han acabado todos los patrones, empezar de nuevo a introducir patrones.

ADLINE

0

- Ejemplo: Decodificador Binario a Decimal

0	0	1	[1]
0	1	0	[2]
0	1	1	[3]
1	0	0	[4]
1	0	1	[5]
1	1	0	[6]
1	1	1	[7]



tasa α =0.3

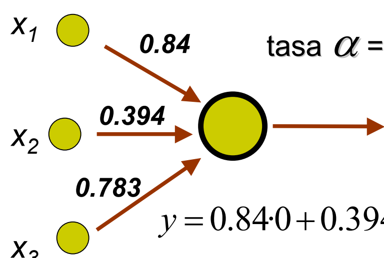
$$y = \sum_{\forall i} w_i x_i$$

ADALINE

1

- Ejemplo: Decodificador Binario a Decimal

0	0	1	[1]
0	1	0	[2]
0	1	1	[3]
1	0	0	[4]
1	0	1	[5]
1	1	0	[6]
1	1	1	[7]



tasa α =0.3

$$y = 0.84 \cdot 0 + 0.394 \cdot 0 + 0.783 \cdot 1 = 0.783$$

$$E = \sum_{\forall p} (d_p - y_p) = 1 - 0.783 = 0.217$$

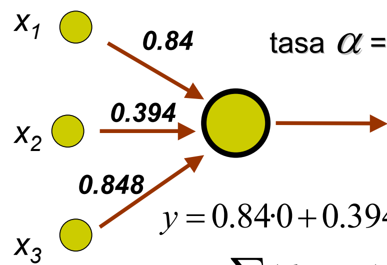
$$w_1 = w_1 + \alpha E \cdot x_1 = 0.84$$
$$w_2 = w_2 + \alpha E \cdot x_2 = 0.394$$
$$w_3 = w_3 + \alpha E \cdot x_3 = 0.783 + 0.3 \cdot 0.217 \cdot 1 = 0.848$$

ADLINE

2

- Ejemplo: Decodificador Binario a Decimal

$$\begin{matrix} 0 & 0 & 1 & [1] \\ 0 & 1 & 0 & [2] \\ 0 & 1 & 1 & [3] \\ 1 & 0 & 0 & [4] \\ 1 & 0 & 1 & [5] \\ 1 & 1 & 0 & [6] \\ 1 & 1 & 1 & [7] \end{matrix}$$



tasa α =0.3

$$y = 0.84 \cdot 0 + 0.394 \cdot 1 + 0.848 \cdot 0 = 0.394$$

$$E = \sum_{\forall p} (d_p - y_p) = 2 - 0.394 = 1.61$$

$$w_1 = w_1 + \alpha \cdot E x_1 = 0.84$$
$$w_2 = w_2 + \alpha \cdot E x_2 = 0.394 + 0.3 \cdot 1.61 \cdot 1 = 0.876$$
$$w_3 = w_3 + \alpha \cdot E x_3 = 0.848$$

prb@2007

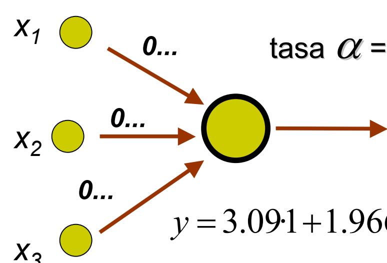
31

ADALINE

7

- Ejemplo: Decodificador Binario a Decimal

$$\begin{matrix} 0 & 0 & 1 & [1] \\ 0 & 1 & 0 & [2] \\ 0 & 1 & 1 & [3] \\ 1 & 0 & 0 & [4] \\ 1 & 0 & 1 & [5] \\ 1 & 1 & 0 & [6] \\ 1 & 1 & 1 & [7] \end{matrix}$$



tasa α =0.3

$$y = 3.09 \cdot 1 + 1.966 \cdot 1 + 1.825 \cdot 1 = 6.881$$

$$E = \sum_{\forall p} (d_p - y_p) = 7 - 6.881 = 0.12$$

Resultado despues de la primera iteración del entrenamiento

$$w_1 = w_1 + \alpha E \cdot x_1 = 3.09 + 0.3 \cdot 0.12 = 3.126$$
$$w_2 = w_2 + \alpha E \cdot x_2 = 1.966 + 0.3 \cdot 0.12 = 2.002$$
$$w_3 = w_3 + \alpha E \cdot x_3 = 1.825 + 0.3 \cdot 0.12 = 1.861$$

32

ADALINE

- Ejemplo: visualización de los pesos según iteraciones..

Iteración	*Pesos*		
1	3.12	2.00	1.86
2	3.61	1.98	1.42
3	3.82	1.98	1.2
4	3.92	1.98	1.1
5	3.96	1.99	1.02
6	3.99	2.00	1.01
7	4.00	2.00	1.00
8	4.00	2.00	1.00
9	4.00	2.00	1.00
10	4.00	2.00	1.00

> La tasa de aprendizaje α también puede ser adaptativa.

> Por ejemplo al inicio el valor puede ser alto, para dar "grandes pasos" de corrección del error y para salir de mínimos locales.

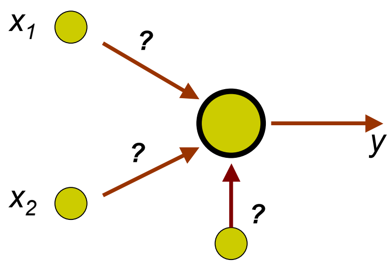
> Sin embargo al final del entrenamiento debe disminuir para hacer correcciones finas.

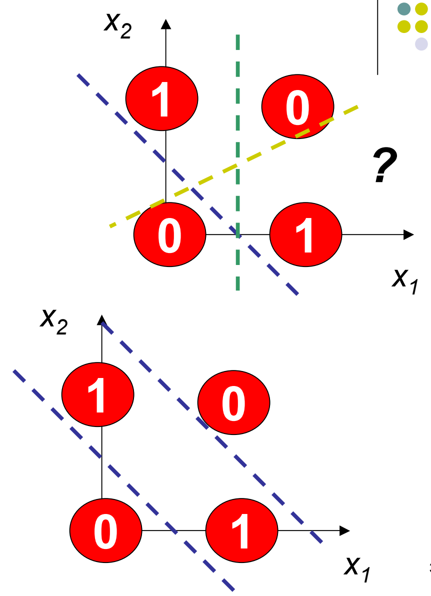
Sin embargo...

- El uso del Perceptrón o de las redes ADALINE permite aproximar de manera fácil, cualquier tipo de función o sistemas, sólo conociendo un conjunto de ejemplos.
- De esta manera cualquier sistema (caja negra), se puede representar por una red.
- Sin embargo, después de la década del 50 se demostró que estas técnicas poseen grandes limitaciones.
- Un ejemplo clásico es el OR Exclusivo.
- **CONCLUSION: éstas técnicas sólo pueden resolver sistemas donde los ejemplos son linealmente separables.**

Sin embargo...

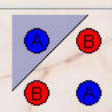
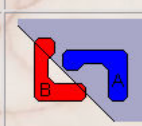
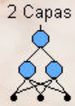
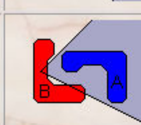
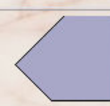
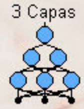
- OR Exclusivo.





- El Perceptrón o ADALINE nunca convergen!!
- Solución: varias redes en cascada ➔ complejidad!!

Sin embargo...

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

Perceptrón Multicapa

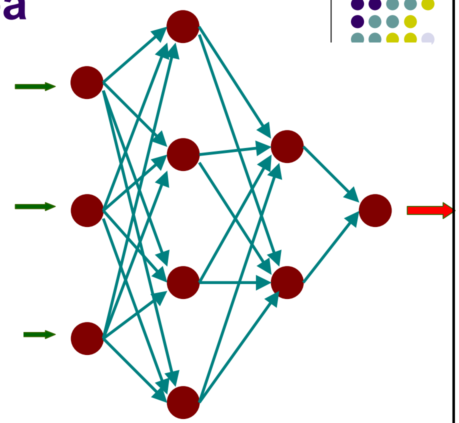
- Corresponde a una generalización del Perceptrón y Adaline
- 1969, Minsky & Papert mostraron que el uso de varios perceptrones simples (neuronas ocultas) puede ser una solución para problemas no lineales. Sin embargo no dejaron en claro como se puede entrenar (ajustar los pesos ocultos)
- 1986, Rumelhart & ..., presentó un método para retropropagar el error medido en la salida hacia las neuronas ocultas. Se denomina REGLA DELTA GENERALIZADA.
- 1989, Cybenko, Hornik, han demostrado que el Perceptrón Multicapa es un aproximador universal. Cualquier función continua sobre R^n, puede ser aproximada, con al menos una capa oculta.

Perceptrón Multicapa

- El Perceptrón Multicapa: puede aprender a partir de ejemplos, aproximar relaciones no lineales, filtrar ruido, modelar sistemas...
- Con éxito ha sido aplicado a:
  - Reconocimiento del habla (Cohen, 93)
  - Reconocimiento de caracteres (Sackinger, 92)
  - Reconocimiento de caracteres escritos (Guyon, 91)
  - Control de Procesos (Werbos, 89)
  - Modelamiento de Sistemas Dinámicos (Narendra, 90)
  - Conducción de vehículos (Pomerleau, 92)
  - Diagnósticos médicos (Baxt, 92)
  - Predicción de Series Temporales (Weiggend, 90)
  - Etc...

Perceptrón Multicapa

- Arquitectura:
  - CAPA DE ENTRADA
  - CAPAS OCULTAS
  - CAPA DE SALIDA



- Todas las neuronas transmiten información hacia delante: se denominan redes ***feedforward***
- Cada neurona posee un umbral independiente. Se considera como una entrada más cuya entrada es 1.
- Generalmente se utilizan resde completamente conectadas
- Función de activación de las neuronas: sigmoidal, hiperbólica.

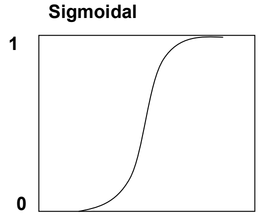
39

Perceptrón Multicapa

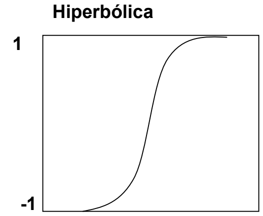
- Función de activación de las neuronas:

  sigmoidal, hiperbólica ➔ son equivalentes (f_2=2f_1-1)

Sigmoidal



Hiperbólica



$$f(x) = \frac{1}{1+e^{-x}} \qquad f(x) = \frac{1-e^{-x}}{1+e^{-x}}$$

La derivada de la f. Sigmoidal es: f(x)'=f(x)(1-f(x))

prb@2007

40

Perceptrón Multicapa

➢Algoritmo BACKPROPAGATION:

➢Cada neurona de salida distribuye hacia atrás su error δ a las neuronas ocultas que se conectan a ella, ponderado por el valor de la conexión.

➢Cada neurona oculta recibe un δ de cada neurona de salida. La suma de estas es el término δ de la neurona oculta.

➢Se repite el proceso hacia atrás... Por ello el nombre “retropropagación del error”.

Perceptrón Multicapa

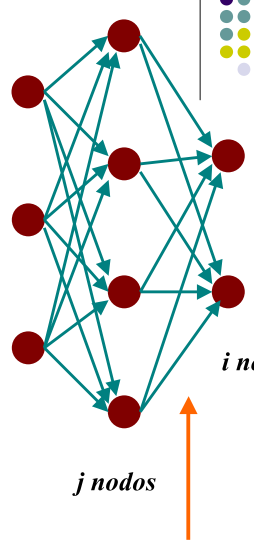
➢Algoritmo BACKPROPAGATION:

➢Pesos de la capa oculta 1, y umbrales de la capa de salida:

$$w_{ji} = w'_{ji} + \alpha \cdot \delta_i x_j$$

$$u_i = u'_i + \alpha \cdot \delta_i$$

$$\delta_i = (s_i - y_i) \cdot y_i \cdot (1 - y_i)$$



Perceptrón Multicapa

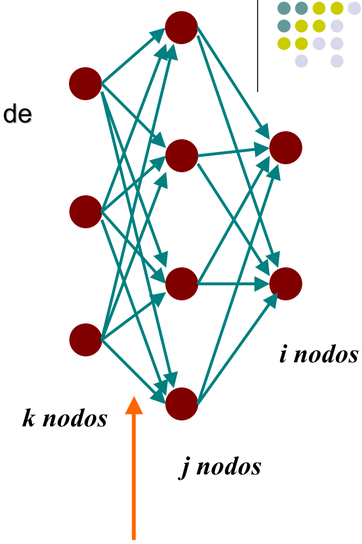
➢Algoritmo BACKPROPAGATION:

➢Pesos de la capa entrada, y umbrales de la capa de salida:

$$w_{kj} = w'_{kj} + \alpha \cdot \delta_j x_k$$

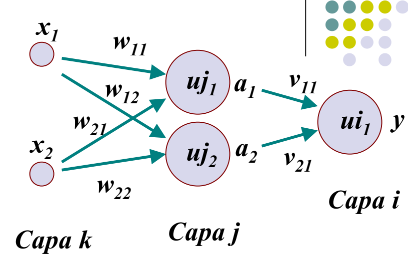
$$u_j = u'_j + \alpha \cdot \delta_j$$

$$\delta_j = x_j(1 - x_j)\sum_{\forall i} w_{ji}\delta_i$$



Perceptrón Multicapa

➢Ejemplo: XOR



$$v_{11} = v'_{11} + \alpha \cdot \delta^i \cdot a_1$$

$$v_{12} = v'_{12} + \alpha \cdot \delta^i \cdot a_2$$

$$ui_1 = ui'_1 + \alpha \cdot \delta^i$$

$$\delta^i = (s - y) \cdot y \cdot (1 - y)$$

$$w_{11} = w'_{11} + \alpha \cdot \delta_1^j \cdot x_1$$

$$\ldots$$

$$uj_1 = uj'_1 + \alpha \cdot \delta_1^j$$

$$\delta_1^j = a_1(1 - a_1)\sum_{\forall i} v_{1i}\delta^i$$

➢TAREA: Implementar esta pregunta de prueba.