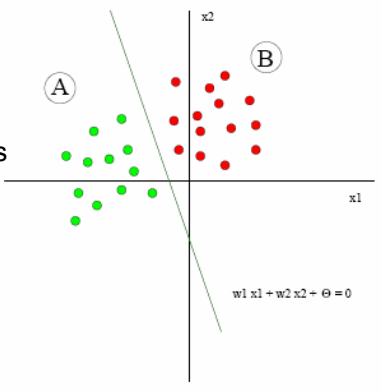
Perceptron y Adaline

Perceptron simple

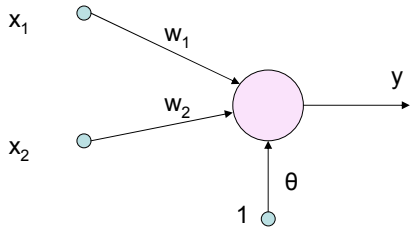
- Sistema capaz de realizar tareas de clasificación de forma automática

- A partir de un número de ejemplos etiquetados, el sistema determina la ecuación del hiperplano discrimininante



Arquitectura

- Red monocapa con una o más neuronas de salida conectadas a todas las entradas



$$y = \begin{cases} 1, \text{ si } & w_1 x_1 + w_2 x_2 + \theta > 0 \\ -1, \text{ si } & w_1 x_1 + w_2 x_2 + \theta \leq 0 \end{cases}$$

Arquitectura

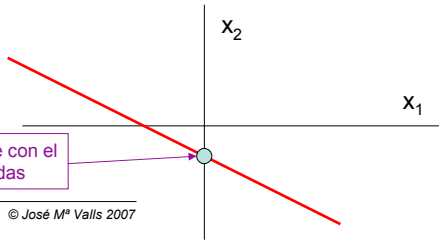
- El perceptron equivale a un hiperplano de dimensión n-1 capaz de separar las clases
  - Si la salida del perceptron es +1, la entrada pertenecerá a una clase (estará situada a un lado del hiperplano)
  - Si la salida es -1, la entrada pertenecerá a la clase contraria (estará situada al otro lado del hiperplano)
- La ecuación del hiperplano es: $w_1 x_1 + w_2 x_2 + \theta = 0$

En dos dimensiones es una recta

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{\theta}{w_2}$$

Pendiente de la recta

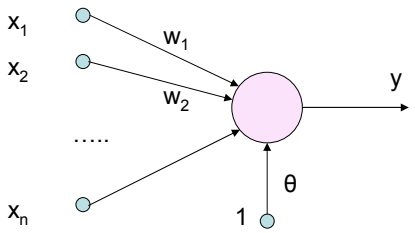
Punto de corte con el eje de ordenadas



Arquitectura

- En general, para n dimensiones:

$$w_1 x_1 + w_2 x_2 + ... + w_n x_n + \theta = 0$$



Proceso

- Los datos son puntos en un espacio multidimensional
- Se dispone de un conjunto de observaciones (puntos) de los que se sabe su categoría o clase
- Hay que determinar la ecuacion del hiperplano que deja a un lado los ejemplos de un tipo y a otro los del otro
- La ecuacion del hiperplano se deduce a partir de los ejemplos
- Ejemplos: Puntos de $\Re^n : (x_1, x_2, ...x_n)$
- Hiperplano: $w_1 x_1 + w_2 x_2 + ... + w_n x_n + \theta = 0$

Aprendizaje

- Proceso iterativo supervisado
- Modicacion de los parámetros de la red (pesos y umbral), hasta encontrar el hiperplano discriminante
- Numero finito de iteraciones

❑ Dado:

- Conjunto de ejemplos de entrenamiento etiquetados (clase A o B) y distribuidos en un espacio multidimensional $\Re^n$

  Ejemplos: patrón p: $x^p = (x_1^p, x_2^p, ...x_n^p)$ Salida deseada para p: $d(x^p)$

  $d(x^p) = 1, si\ x^p \in A; \quad d(x^p) = -1, si\ x^p \in B$

❑ Encontrar hiperplano discriminante que separa las clases A y B

$$w_1 x_1 + w_2 x_2 + ... + w_n x_n + \theta = 0$$

Encontrar los parámetros $w_1, w_2, ...w_n, \theta$

Algoritmo de Aprendizaje

- Se dispone de un conjunto de ejemplos de entrenamiento; cada uno de ellos será un vector de entrada y una salida: $(\chi, d(\chi)), d(\chi) = \{-1,1\}$

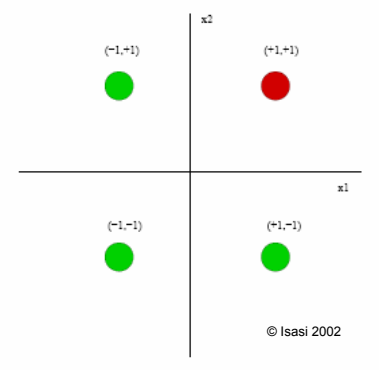
1. Empezar con valores aleatorios para los pesos y el umbral
2. Seleccionar un vector de entrada del conjunto de ejemplos de entrenamiento
3. Si $y \neq d(\chi)$ la red da una respuesta incorrecta.
   Modicar w_i de acuerdo a:

$$\Delta w_i = x_i d(\chi)$$
$$\Delta \theta = d(\chi)$$

4. Si no se ha cumplido el criterio de finalización, volver a 2

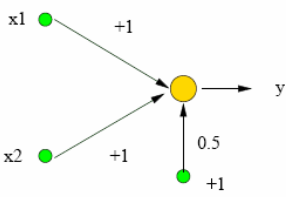
Ejemplo: función AND

	x_1	x_2	AND
1	-1	-1	-1 (A)
2	+1	-1	-1 (A)
3	-1	+1	-1 (A)
4	+1	+1	+1 (B)

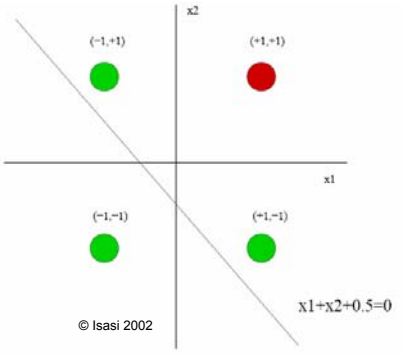


Función AND

- Valores iniciales:



© Isasi 2002



© Isasi 2002

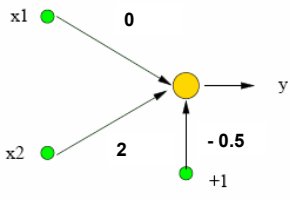
	Patron	Salida	Clasifica
1	(-1,-1|-1)	f(-1 + (-1) + 0.5)=f(-1.5)=-1	Bien [-1=d(X)]
2	(+1,-1|-1)	f(1 + (-1) + 0.5)=f(0.5)=+1	Mal [+1 ≠ d(X)]

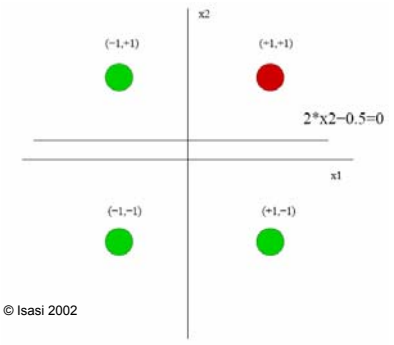
Hay que ajustar pesos

Función AND

2

Aprendizaje:		
Antes	Incremento	Después
$w_1 = 1$	$d(\chi) \cdot x_1 \equiv -1 \cdot +1 = -1$	$w_1 = 0$
$w_2 = 1$	$d(\chi) \cdot x_2 \equiv -1 \cdot -1 = 1$	$w_2 = 2$
$\theta = 0.5$	$d(\chi) \equiv -1$	$w_1 = -0.5$





Función AND

3

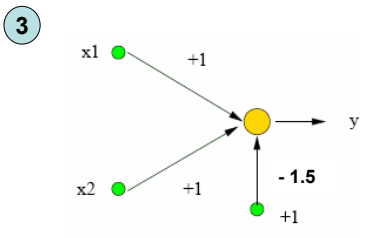
Patron	Salida	Clasifica
(-1,+1|-1)	f(0 + 2 - 0.5)=f(1.5)=1	Mal [1≠ d(X)]

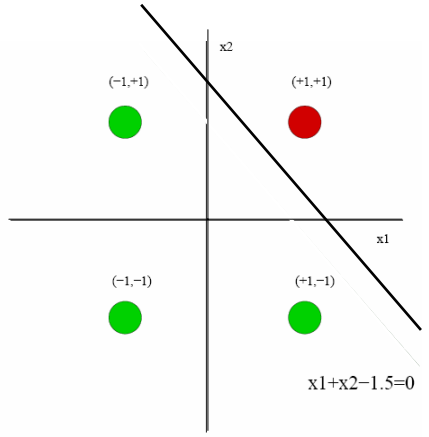
Hay que ajustar pesos

Aprendizaje:		
Antes	Incremento	Después
$w_1 = 0$	$d(\chi) \cdot x_1 \equiv -1 \cdot -1 = 1$	$w_1 = 1$
$w_2 = 2$	$d(\chi) \cdot x_2 \equiv -1 \cdot 1 = -1$	$w_2 = 1$
$\theta = -0.5$	$d(\chi) \equiv -1$	$w_1 = -1.5$

Función AND

3





Función AND

4

Patron	Salida	Clasifica
(+1,+1|+1)	f(1 + 1 - 1.5)=f(0.5)=1	Bien [1= d(X)]

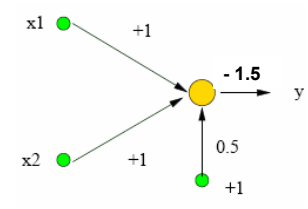
Ya se ha completado un ciclo.

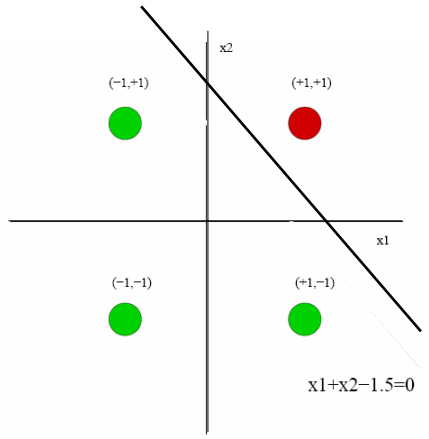
Hay que completar otro ciclo sin que se realicen ajustes:

	Patron	Salida	Clasifica
1	(-1,-1|-1)	f(-1 + (-1) - 1.5)=f(-3.5)=-1	Bien [-1=d(X)]
2	(+1,-1|-1)	f(1 + (-1) - 1.5) =f(-1.5)=-1	Bien [-1=d(X)]
3	(-1,+1|-1)	f(-1 + 1 - 1.5) =f(-1.5)=-1	Bien [-1=d(X)]
4	(+1,+1|+1)	f(1 + 1 - 1.5) =f(0.5)=+1	Bien [+1=d(X)]

Función AND

Resultado final:





Adaline: ADAptive LInear NEuron

- El perceptrón realiza tareas de clasificación ⟹ salida binaria
- Si las salidas fueran números reales, estaríamos ante un problema de regresión

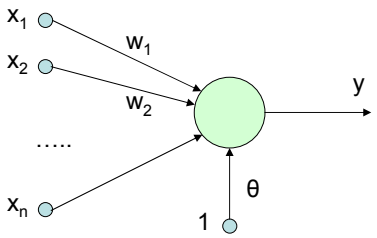
  (Applet de regresión: http://www.math.csusb.edu/faculty/stanton/m262/regress/regress.html)
- Aproximar una función cualquiera F(x) definida por un conjunto de datos de entrada y su salida real correspondiente
- Los ejemplos de entrenamiento son conjuntos de valores: (vector de entrada, salida real)

$$P = \{(\vec{x}^1, y^1), (\vec{x}^2, y^2), ...(\vec{x}^m, y^m)\}$$

- Habrá que buscar la función $F(\vec{x})$ tal que $F(\vec{x}^i) = y^i, \forall \vec{x}^i \in P$

Adaline: ADAptive LInear NEuron

- Desarrollado en 1960 por Widrow y Hoff
- Estructura prácticamente idéntica al perceptron, pero es un mecanismo físico capaz de realizar aprendizaje
- Elemento combinador adaptativo lineal, que recibe todas las entradas, las suma ponderadamente, y produce una salida



$$y = \sum_{i=1}^{n} w_i \cdot x_i + \theta$$

Adaline

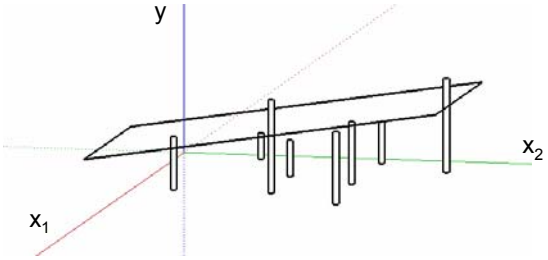
- La diferencia con el perceptron es la manera de utilizar la salida en la regla de aprendizaje
- El perceptron utiliza la salida de la funcion umbral (binaria) para el aprendizaje. Sólo se tiene en cuenta si se ha equivocado o no.
- En Adaline se utiliza directamente la salida de la red (real) teniendo en cuenta cuánto se ha equivocado.
- Se utiliza la diferencia entre el valor real esperado ***d*** y la salida producida por la red ***y***.

  Para un patrón de entrada x^p, se tendrá en cuenta el error producido (d^p-y^p)

- El objetivo es obtener una red tal que $y^p=d^p$ para todos los patrones ***p***

Adaline

- Será imposible conseguir una salida exacta porque ***y*** es una función lineal, pero se minimizará el error cometido para todos los patrones de entrenamiento
- Hay que elegir una medida de dicho error, p.ej.el error cuadrático
- La regla de aprendizaje será la REGLA DELTA



Regla Delta

- Los patrones de entrenamiento estan constituidos por pares de valores $(\vec{x}, d)$ que son el vector de entrada y su salida deseada
- La regla Delta utiliza la diferencia entre la salida producida para cada patron (p) y la deseada $(d^p - y^p)$
- Se calcula una funcion de error para todo el conjunto de patrones:

$$E = \sum_{p=1}^{m} E^p = \frac{1}{2} \sum_{p=1}^{m} (d^p - y^p)^2$$

Error global — Error cuadrático por patrón

Regla Delta

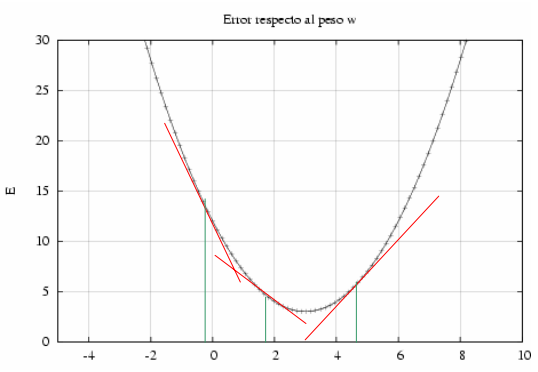
- La regla Delta busca el conjunto de pesos que minimiza la funcion de error
- Se hará mediante un proceso iterativo donde se van presentando los patrones uno a uno y se van modificando los parámetros de la red mediante la regla del descenso del gradiente

- La idea es realizar un cambio en cada peso proporcional a la derivada del error, medida en el patron actual, respecto del peso:

$$\Delta_p w_j = -\gamma \frac{\partial E^p}{\partial w_j}$$

Regla Delta

$$\Delta_p w_j = -\gamma \frac{\partial E^p}{\partial w_j}$$



Regla Delta

- Aplicando la regla de la cadena queda:

$$\frac{\partial E^p}{\partial w_j} = \frac{\partial E^p}{\partial y^p}\frac{\partial y^p}{\partial w_j}$$

$$E^p = \frac{1}{2}(d^p - y^p)^2$$

$$y^p = w_1 x_1 + w_1 x_1 + \ldots w_j x_j + \ldots + \theta$$

$$2 \times \frac{1}{2}(d^p - y^p) \times (-1) = -(d^p - y^p)$$

$$\frac{\partial y^p}{\partial w_j} = x_j$$

$$\Delta_p w_j = \gamma (d^p - y^p) x_j$$

Algoritmo de aprendizaje

1. Inicializar los pesos de forma aleatoria
2. Introducir un patrón de entrada
3. Calcular la salida, compararla con la deseada y obtener la diferencia: (d^p - y^p)
4. Para todos los pesos, multiplicar dicha diferencia por la entrada correspondiente y ponderarla por la tasa de aprendizaje

$$\Delta_p w_j = \gamma (d^p - y^p) x_j$$

5. Modicar el peso sumando al valor antiguo la cantidad obtenida en 4)

$$w_j^p = w_j^{p-1} + \Delta_p w_j$$

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

Perceptron vs. Adaline

- En Perceptron la salida es binaria, en Adaline es real
- En Adaline existe una medida de cuánto se ha equivocado la red, en Perceptron sólo de si se ha equivocado o no
- En Adaline hay una razón de aprendizaje γ para regular lo que va a afectar cada equivocacion a la modicacion de los pesos. Es siempre un valor entre 0 y 1 para ponderar el aprendizaje

Ejemplo 1: Decodificador binario-decimal

Aproximar con Adaline la función que realiza la decodificación binario-decimal

Vamos a utilizar como tasa de aprendizaje 0.3 y los siguientes valores de pesos:

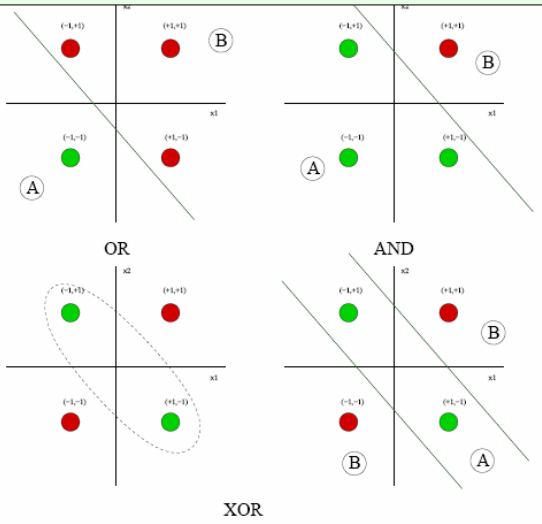
w1=0.84

w2=0.39

w3=0.78

x1	x2	x3	d
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

Problemas no linelamente separables: XOR



Problema XOR

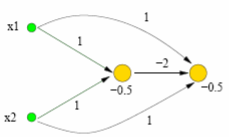
- Este simple problema no se puede resolver por un perceptron (Minsky,Papert 1969)
- La ecuación de un perceptron para una θ constante es: $w_1 \cdot x_1 + w_2 \cdot x_2 + \theta = 0$
- Como los puntos tienen sus signos negativos cualquier solución de pesos no produce el mismo signo a la salida:

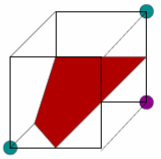
Clase	Patrón 1	Patrón 2
A	$-w_1 - w_2 + \theta > 0$	$+w_1 + w_2 + \theta > 0$
B	$-w_1 + w_2 + \theta < 0$	$+w_1 - w_2 - \theta < 0$

- Para que se cumplan las ec. del patrón A $\theta < 0$, para que se cumplan las del B $\theta > 0$, lo cual es **IMPOSIBLE**
- Esto es debido a que el perceptron solo soluciona problemas separables linealmente

Problema XOR

- Hay que transformar la entrada para que se convierta en linealmente separable
- Para ello hay que introducir una capa oculta:





- La introducción de capas ocultas puede separar cualquier región
- El problema es cómo entrenar las capas ocultas, pues se desconoce su salida deseada