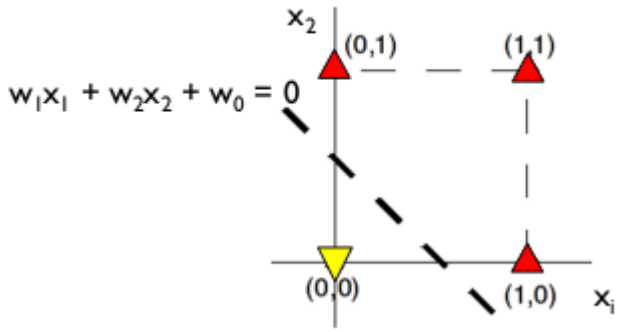
Tema 6-Sistemas conexionistas alimentados hacia delante

Es un modelo anterior al Adaline. Es el primer modelo de RNA. Se diferencia del Adaline en la función de transferencia y en el algoritmo de entrenamiento.

Tiene una única neurona (EP) en la capa de salida. Sirve para resolver problemas linealmente separables. Por ejemplo, una puerta OR:

- 4 patrones:
  - (0,0) (0,1) (1,0) (1,1)

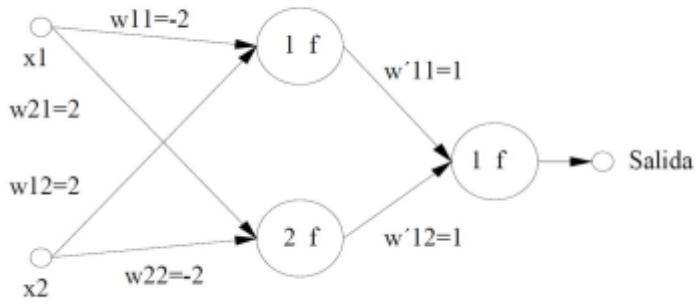


La regla más utilizada para la actualización de pesos es la Regla Delta:
$wi(t + 1) = wi(t) + \mu(d(t) - y(t))xi(t)$

Se obtiene una convergencia finita si el conjunto es linealmente separable.

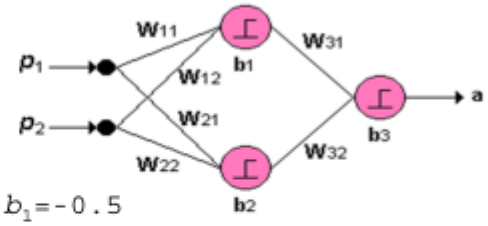
Perceptrón multicapa

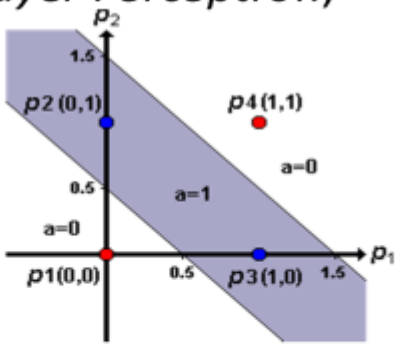
Pero existen problemas, como el XOR, que no son linealmente separables. La solución es añadir más capas.



XI	X2	Salida neurona oculta I	Salida neurona oculta 2	Salida
0	0	0	0	0
0	I	I	0	I
I	0	0	I	I
I	I	0	0	0

- Perceptrón multicapa (*Multilayer Perceptron*, MLP)

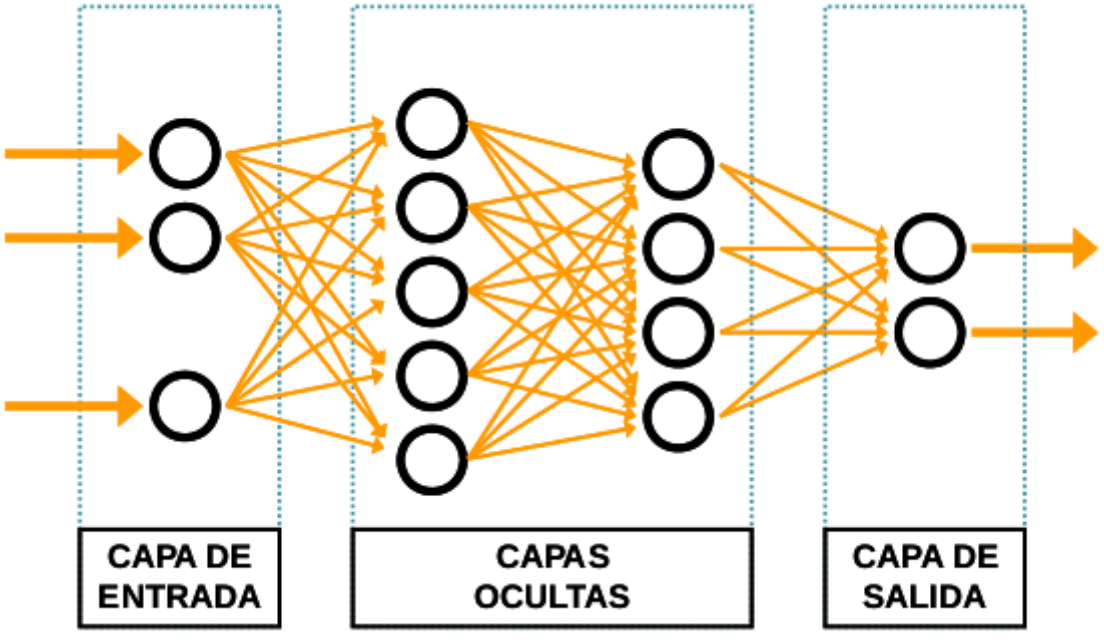




$w_{11}=1 \quad w_{12}=1 \quad b_1=-0.5$
$w_{21}=1 \quad w_{22}=1 \quad b_2=-1.5$
$w_{31}=1 \quad w_{32}=-1.5 \quad b_3=-0.5$

XI	X2	Neta neurona oculta I	Salida neurona oculta I	Neta neurona oculta 2	Salida neurona oculta 2	Neta neurona salida	Salida
0	0	0*I + 0*I - 0.5 = -0.5	0	0*I + 0*I - I.5 = -I.5	0	0*I – 0*I.5 - 0.5 = -0.5	0
0	I	0*I + I*I - 0.5 = 0.5	I	0*I + I*I - I.5 = -0.5	0	I*I – 0*I.5 – 0.5 = 0.5	I
I	0	I*I + 0*I - 0.5 = 0.5	I	I*I + 0*I - I.5 = -0.5	0	I*I – 0*I.5 – 0.5 = 0.5	I
I	I	I*I + I*I - 0.5 = I.5	I	I*I + I*I - I.5 = 0.5	I	I*I – I*I.5 – 0.5 = -I	0

La arquitectura general de un perceptrón multicapa es:



- **Capa de entrada**: no computa nada, solo almacenan las entradas. 1 neurona por entrada
- **Capas ocultas**: neuronas ocultas. n neuronas, prueba y error
- **Capa de salida**: emite la salida de la RNA. 1 neurona por salida

El conocimiento de la red reside en los pesos de las conexiones y bias. Es imposible explicar su funcionamiento.

Las funciones de transferencia se asume que son la mismas para todas las neuronas de la red o, al menos, para las neuronas de la misma capa. Las más típicas son la función umbral, la lineal, la logarítmica sigmoidal o la tangente sigmoidal. Generalmente la función de transferencia en las capas internas no puede ser lineal.

Un perceptrón puede aprender cualquier tipo de función o relación continua entre un grupo de variables de entrada y salida.

Se aplica en ajuste de funciones y curvas, problemas de clasificación y problemas de regresión.

Algoritmo de backpropagation

No se puede utilizar la Regla Delta porque no se conocen las salidas deseadas de las capas ocultas. Par ello se utiliza la Regla Delta Generalizada o backpropagation.

En la capa de salida puede haber más de una neurona, por tanto no basta con calcular un único error. Hay que minimizar el error de la suma de los cuadrados de los errores. Por tanto, podemos modificar los pesos en la capa de salida. Pero en las capas ocultas nos faltan como parámetros las salidas deseadas. Lo que se puede hacer es propagar el error capa a capa para atrás. Se repite este proceso hasta llegar a las emtradas.

El algoritmo es el siguiente:

1. Se inicializan los pesos aleatoriamente
2. Se calculan los errores y se modifican los pesos
3. Se repite este proceso n ciclos (épocas)
4. Si no se ha mejorado el error durante una serie de ciclos, para evitar el sobrentrenamiento, se para

Entrenamiento

No hace falta emplear todos los datos para entrenar una red, pero si un subconjunto que cubra todo el espacio. Es importante que sean representativos. Permite generalización, es decir, incluír nuevos vectores de datos que no pertenezcan al conjunto de entrenamiento.

El conjunto de datos debe ser:

- **Significativo**: suficiente número de ejemplo
- **Representativo**: cubrir todas las regiones del espacio

Si un conjunto de aprendizaje contiene muchos más ejemplo de un tipo que del resto, la red se especializará demasiado en dicho subconjunto.

Las entradas deben estar entre 0 y 1 o entre -1 y 1. Las salidas deben ser también normalizadas.

En cuanto al control de convergencia, no se puede asegurar que a lo que se llega sea un mínimo global. Puede llegar a mínimos locales, no es determinista. Por tanto, si alcanzamos un mínimo local, pero el error es satisfactorio, el entrenamiento es un éxito.

Después del entrenamiento se pasa otro conjunto de patrones: el conjunto de test. Son valores que no están presentes en el entrenamiento e indican si la red está bien entrenada o no.

Cuando hay una mala generalización es por culpa del sobreentrenamiento. Para evitarlo podemos establecer que pare cuando lleve cierto número de ciclos sin mejorar la validación o establecer un número de ciclos muy alto y memorizar la red con mejor resultado de validación.

Hacen falta 3 conjuntos de patrones:

- **Entrenamiento**: guía el proceso de entrenamiento
- **Validación**: supervisa el entrenamiento y lo para si es necesario
- **Test**: evalúa el resultado una vez terminado el entrenamiento. Es la únic forma de saber si la red está bien entrenada.