

**IMPLEMENTACION DE LA FUNCION LOGICA XOR, MEDIANTE UN MODELO NEURONAL  
Y EL ALGORITMO BACKPROPAGATION**

**EXPOSITORES:**

**ANDRES FELIPE RESTREPO LONDOÑO**

**BIBIANA JANETH GALLEGO MORA**

**FACULTAD DE INGENIERIA DE SISTEMAS**

**DIRECTOR: JAIRO PERTUZ CAMPO**

**UNIVERSIDAD DE MEDELLIN**

**MEDELLIN, MAYO DE 2005**



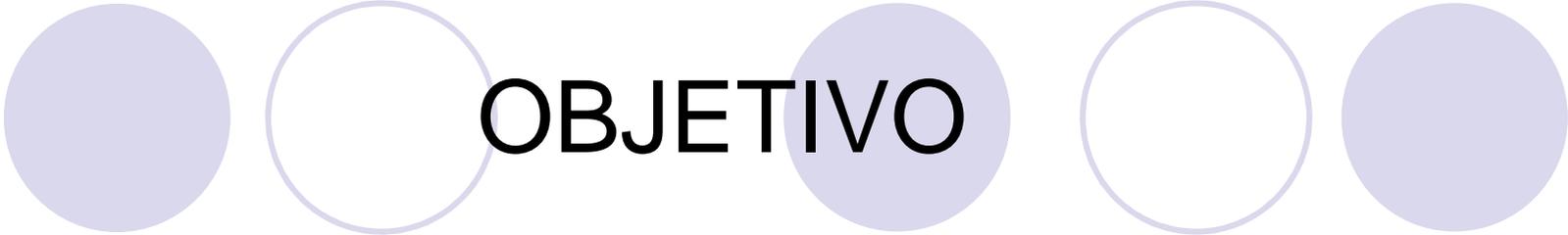
**COMPONENTES  
ELECTRONICAS LTDA**

Distribuidor Autorizado de



**The MathWorks**





# OBJETIVO

- Ilustrar el proceso de entrenamiento de una red neuronal con el algoritmo Backpropagation, mediante la implementación de la función lógica XOR.
- Demostrar que la velocidad de entrenamiento para una representación bipolar es mucho más rápida que una representación binaria

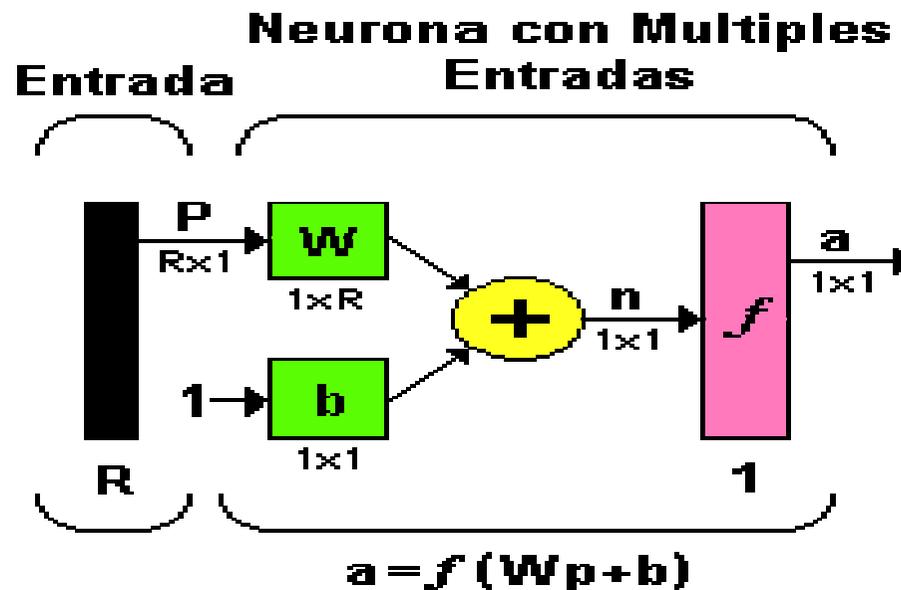
# REDES FEEDFORWARD

Las interacciones son siempre hacia adelante

Se suele hablar de:

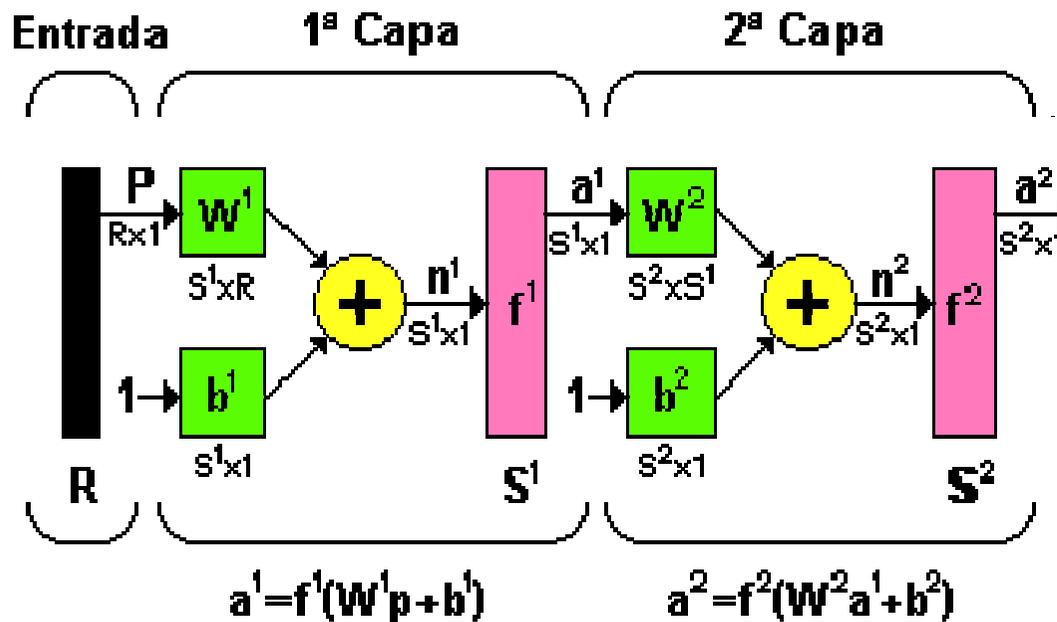
Redes de una sola capa (cuando no hay capas ocultas),

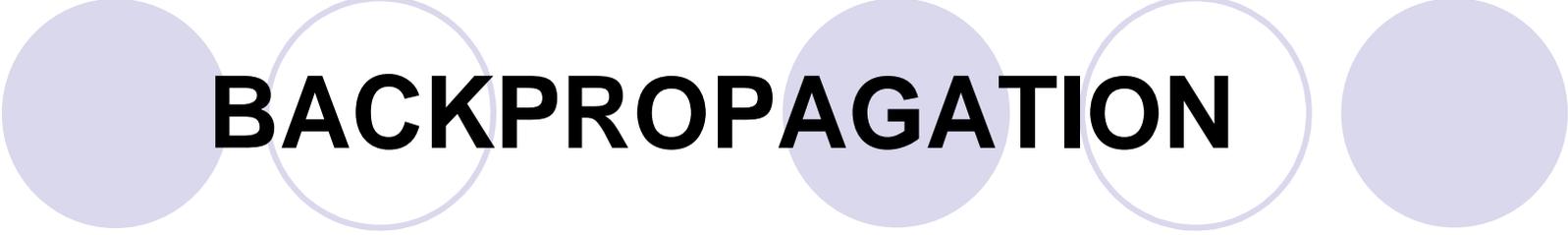
Ejemplo: El perceptrón simple



# REDES FEEDFORWARD

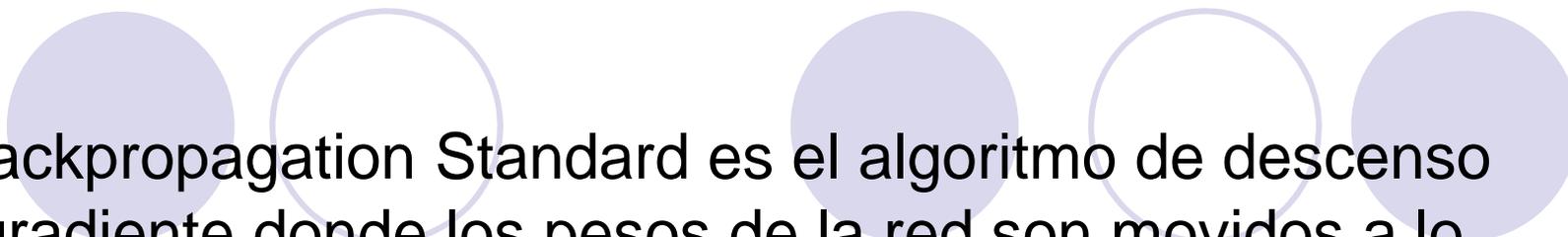
Redes de múltiples capas (cuando hay capas ocultas): Ejemplos: el perceptrón multicapa





# BACKPROPAGATION

- Procedimiento para encontrar el vector gradiente de una función error asociada a la salida de la red con respecto a los parámetros de la misma.
- El nombre backpropagation surge pues del cálculo que se hace en el sentido inverso de la red, propagándose desde los nodos de salida hacia los nodos de entrada
- Esto permite poder aplicar a posteriori alguno de los muchos métodos de optimización con gradiente para obtener el comportamiento deseado de la red



El Backpropagation Standard es el algoritmo de descenso del gradiente donde los pesos de la red son movidos a lo largo de la dirección negativa del gradiente (Es donde se minimiza para obtener T).

Existen varias técnicas como lo son:

- El algoritmo del gradiente conjugado.
- El algoritmo de Newton.

En ultima instancia lo que busca este algoritmo del Backporgation es el entrenamiento de Redes Neuronales Feedforward, con las cuales podemos resolver problemas específicos como: Aproximación de Funciones, clasificación, asociación de vectores, etc.

## ELEMENTOS ESENCIALES DE LA RED

ENTRADAS: P

CONEXIONES SINAPTICAS PONDERADAS: Matriz peso (W)

BIAS: Sesgo (b)

SUMADOR: (+)

Las funciones de transferencia utilizadas para este algoritmo son:

- Log – sigmoid transfer function.
- Tan - sigmoid transfer function.
- Linear transfer function.

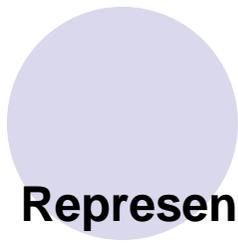
SALIDA DE LA RED: a

SALIDA DESEADA: T

Cuando hablamos de gradiente de una función, como requisito ésta debe ser derivable, por ende estas funciones antes mencionadas tienen derivada.

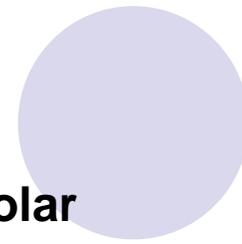
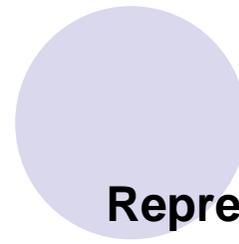
El proceso de entrenamiento:

Agrupar los datos, entrenar la red y posteriormente permite simular la respuesta de la red para nuevas entradas.



## Representación Binaria

P1	P2	<b>XOR</b>
0	0	<b>0</b>
0	1	<b>1</b>
1	0	<b>1</b>
1	1	<b>0</b>



## Representación Bipolar

P1	P2	<b>XOR</b>
-1	-1	<b>-1</b>
-1	1	<b>1</b>
1	-1	<b>1</b>
1	1	<b>-1</b>

## EJERCICIO 1 DE APLICACIÓN CON BACKPROPAGATION

**UTILIZAMOS COMPUERTA LOGICA XOR, ENTRADA BINARIA Y SALIDA BINARIA:**        2 - 4 - 1

“Entrada y T deseado

```
>> p= [0 0 1 1; 0 1 0 1];
```

```
>> T= [0 1 1 0];
```

“Creacion de la red

```
>> net = newff (minmax (p), [4 1], {'logsig','logsig'}, 'trainlm');
```

“Entrenamiento

```
>> net.trainParam.show = 25;
```

```
>> net.trainParam.lr = 0.02;
```

```
>> net.trainParam.epochs = 400;
```

```
>> net.trainParam.goal = 1e-8;
```

```
>> [net,tr] = train(net,p,T);
```

“Simulación

```
>> a = sim (net,p);
```

```
>> e = T - round(a)
```

## EJERCICIO 2 DE APLICACIÓN CON BACKPROPAGATION

UTILIZAMOS COMPUERTA LOGICA XOR, ENTRADA BIPOLAR Y SALIDA BIPOLAR: 2 - 4 - 1:

“Entradas y T deseado

```
>> p= [-1 -1 1 1; -1 1 -1 1];
```

```
>> T= [-1 1 1 -1];
```

“ Creación de la red

```
>> net = newff (minmax (p), [4 1], {'tansig','tansig'}, 'trainlm');
```

“Entrenamiento

```
>> net.trainParam.show = 25;
```

```
>> net.trainParam.lr =0.02;
```

```
>> net.trainParam.epochs = 400;
```

```
>> net.trainParam.goal = 1e-8;
```

```
>> [net,tr] = train(net,p,T);
```

“Simulación

```
>> a = sim (net,p);
```

```
>> e = T -round(a)
```



## COMENTARIOS FINALES

- La arquitectura neuronal (Red Feedforward) y el algoritmo de entrenamiento empleado (Backpropagation), resultaron ser muy eficientes en la ejecución del problema planteado.
- Pudimos comprobar que en esta aplicación, la velocidad de entrenamiento correspondiente a la representación bipolar, resulta ser mucho más rápida que la correspondiente a la representación binaria.