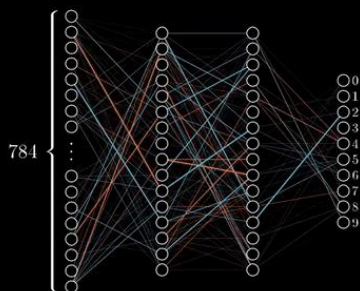




Training in  
progress...



# Inteligencia Artificial

**Unidad 2:** Redes Neuronales

**TEMA 3:** Algoritmos de IA Moderna-I

**Módulo 3:** Algoritmo Backpropagation

Unidad 2

# Redes Neuronales

**TEMA 3:** Algoritmos de IA Moderna-I

Sesión 13

## MÓDULO 3: Algoritmo Backpropagation



### Contenido

1. Backpropagation: principal ingrediente de una RNA
2. Backpropagation: Cálculo del Error
3. Backpropagation: Ejemplo para Función lógica XOR
4. Práctica: Modelo de Clasificación de Imágenes con RNA



### Preguntas

# 1. Backpropagation: principal ingrediente de una RNA

## Algoritmo Backpropagation

- El algoritmo de retropropagación es probablemente el bloque de construcción más importante en una red neuronal.
- Fue introducido por primera vez en la década de 1960 y casi 30 años después (1989) fue popularizado por Rumelhart, Hinton y Williams en un artículo titulado "Aprendizaje de representaciones mediante errores de retropropagación".

Pero, ¿para qué  
se utiliza este  
algoritmo?



El algoritmo **Backpropagation** se utiliza para entrenar eficazmente una red neuronal a través de un método llamado regla de cadena.

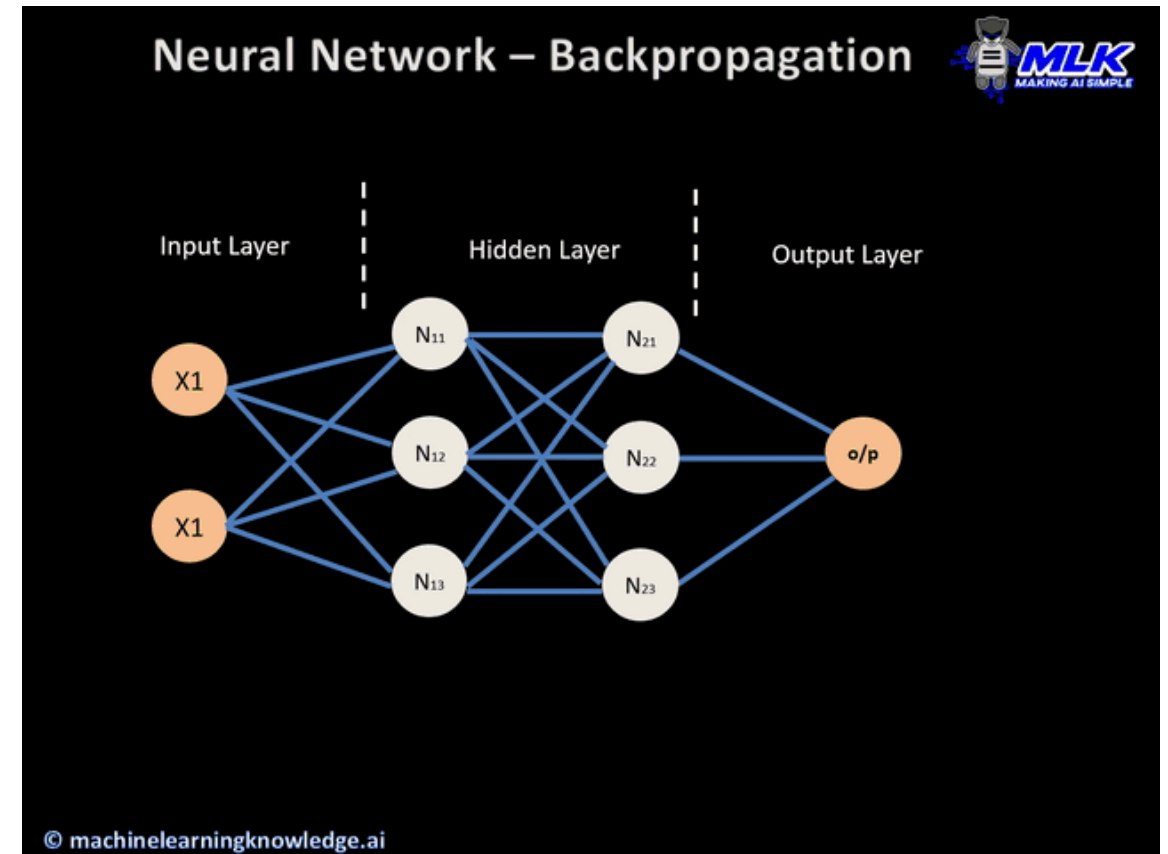
# 1. Backpropagation: principal ingrediente de una RNA

## Algoritmo Backpropagation

- **Backpropagation** es la esencia del entrenamiento de las redes neuronales.

En términos simples:

- Después de cada pase hacia adelante a través de una red, **Backpropagation** realiza un pase hacia atrás mientras ajusta los parámetros del modelo (pesos y sesgos).
- **Backpropagation** entonces, es la práctica de ajustar los pesos de una red neuronal en función de la tasa de error (es decir, pérdida) obtenida en la época anterior (es decir, iteración).
- El ajuste adecuado de los pesos garantiza tasas de error más bajas, lo que hace que el modelo sea confiable al aumentar su generalización.



## 2. Backpropagation: Cálculo del Error

- Hay muchas formas de medir el error. En ejemplos anteriores utilizamos la diferencia de la salida de la red (real) y la salida esperada (objetivo)

$$E = |d^p - y^p|$$

- El **Algoritmo Backpropagation** busca minimizar el error cuadrático medio cometido por la capa de salida.

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

m = numero de neuronas

- Al obtener el error cuadrático medio de cada salida, cambiamos los pesos capa por capa hacia atrás, hasta llegar a la capa de entrada. Esta tarea la realizamos hasta reducir el error de la capa de salida o hasta concluir con las iteraciones del conjunto de datos de entrenamiento.

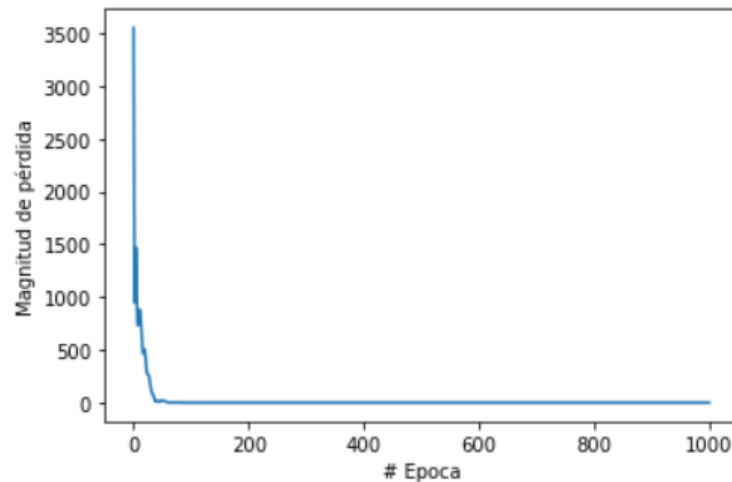
## 2. Backpropagation: Cálculo del Error

- En cada iteración del conjunto de entrenamiento, veremos que el error cuadrático medio se reduce por el cambio de los pesos en cada iteración, tratando de llegar a un error mínimo.

### 5. Visualización gráfica del entrenamiento

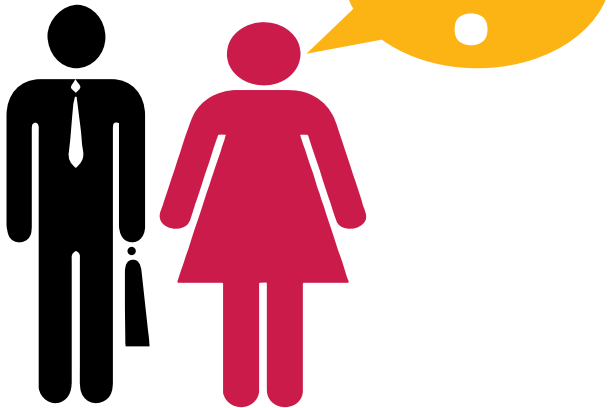
```
In [6]: plt.xlabel("# Epoca")  
plt.ylabel("Magnitud de pérdida")  
plt.plot(historial.history["loss"])
```

```
Out[6]: [<matplotlib.lines.Line2D at 0x1d4ffb42a90>]
```



## 2. Backpropagation: Cálculo del Error

Pero, ¿cómo  
Backpropagation  
cambia los pesos?

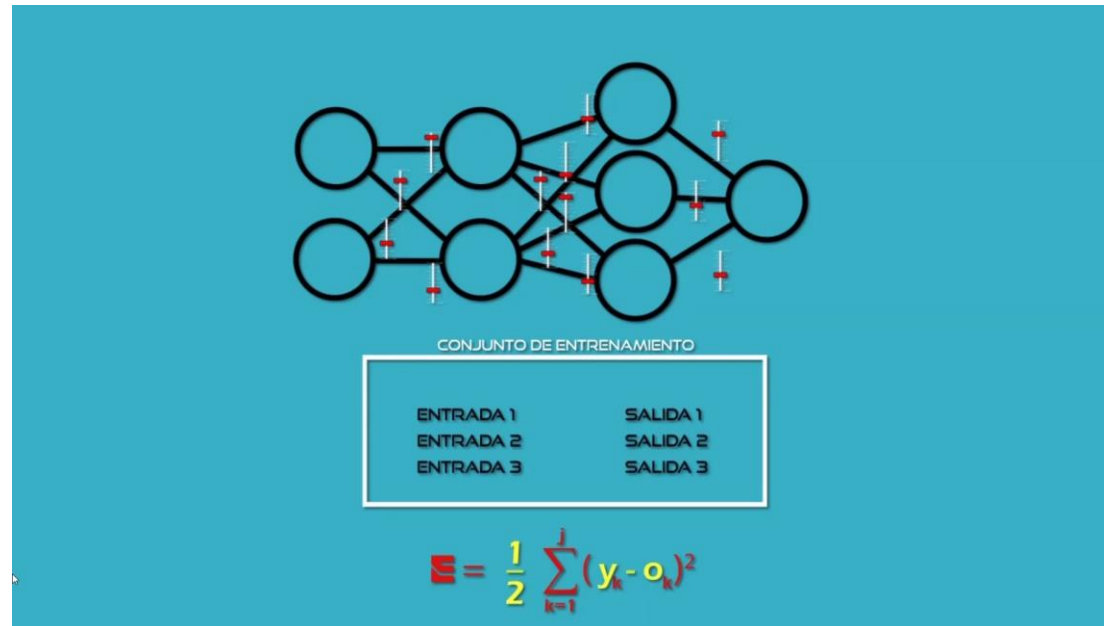


- Para cambiar los pesos, **Backpropagation** utiliza la Regla Delta  $\Delta$
- Esta regla, incrementa el peso con la dirección opuesta al gradiente del error.
- Usa la pendiente para cambiar el peso a otro que provoque menos error, pero también utiliza un factor o **tasa de aprendizaje**,  $\alpha$  que es un numero que reducirá ese cambio para asegurarnos llegar a un error mínimo.
- Para aplicar la Regla Delta, la función de activación debe ser derivable (debido que usamos el gradiente). Por ello utiliza la función Sigmoide porque su derivada es muy fácil de calcular (con la propia salida de la neurona).

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

## 2. Backpropagation: Cálculo del Error

**Backpropagation**  
¿Qué es y cómo funciona?



Enlace: <https://youtu.be/CldXLZUlewE> (Duración: 7 min.)



## 2. Backpropagation: Cálculo del Error

### Algoritmo Backpropagation: Seudocódigo

- Se asigna para la red neuronal pesos iniciales con valores aleatorios.
- Mientras “error” sea grande:
  - Para cada patrón (características) de entrenamiento:
    - Asignar las entradas a la red.
    - Calcular la salida de cada neurona desde la capa de entrada, a través de las capa(s) oculta(s) hasta la capa de salida.

Se utiliza la Función sigmoidea logística  $F(x) = \frac{1}{1 + e^{-x}}$  llamada también función de transferencia.

- Calcular el error en la(s) salida(s) mediante la Regla Delta:  $\Delta o_k = o_k * (1 - o_k) * (T_k - o_k)$
- Usar el error de salida para calcular las señales de error para las capas previas a la salida:  $W o_{1jk} = W o_{1jk} + \alpha * h_j * \Delta o_k$
- Usar las señales de error para recalcular los ajustes de peso.
- Aplicar los ajustes de peso.

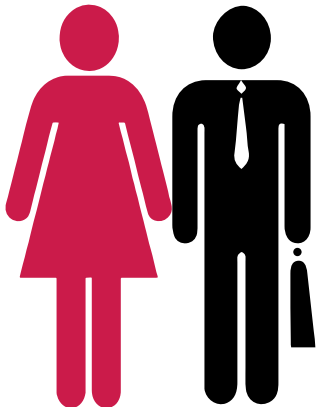
### 3. Backpropagation: Ejemplo para Función lógica XOR

#### Algoritmo Backpropagation

Con las vastas mini ejecuciones simultáneas involucradas,  
¡Aprendamos con un ejemplo!

Apliquemos el algoritmo backpropagation para hallar los pesos de la red neuronal de la **función XOR** (a valores iguales resultado 0, a valores diferentes resultado 1).

Entonces, ¿cómo  
funciona este  
proceso?



Patrones de  
entrenamiento

2 entradas		Salidas deseadas (objetivos)
$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Descripción de la red:

- ❑ 2 neuronas en la capa de entrada
- ❑ 1 capa oculta, con 2 neuronas en la capa oculta ( $h_1, h_2$ )
- ❑ 1 neurona en la capa de salida
- ❑ Tasa de aprendizaje  $\alpha = 0.25$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

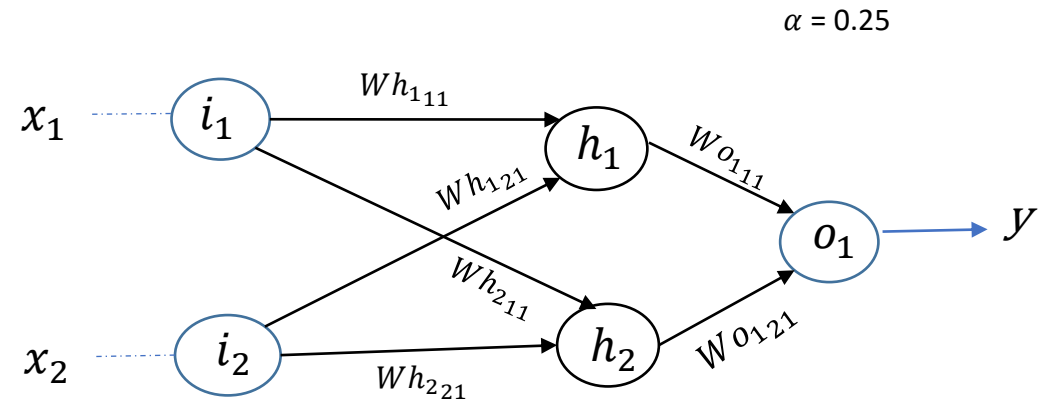
Datos:

		$d(x)$
$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Descripción de la red:

- ❑ 1 capa de entrada con 2 neuronas ( $i_1, i_2$ )
- ❑ 1 capa oculta, con 2 neuronas ( $h_1, h_2$ )
- ❑ 1 neurona en la capa de salida ( $o_1$ )
- ❑ Tasa de aprendizaje  $\alpha = 0.25$

Estructura (diagrama) de la red



### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

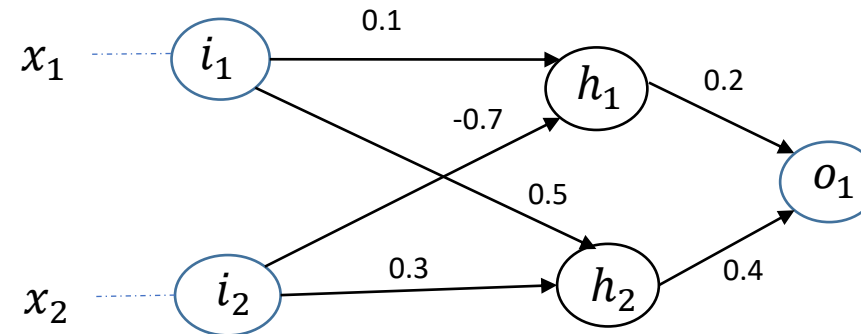
Datos:

		$d(x)$
$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Descripción de la red:

- ❑ 1 capa de entrada con 2 neuronas ( $i_1, i_2$ )
- ❑ 1 capa oculta, con 2 neuronas ( $h_1, h_2$ )
- ❑ 1 neurona en la capa de salida ( $o_1$ )
- ❑ Tasa de aprendizaje  $\alpha = 0.25$

**Paso #1:** Asignamos valores aleatorios para los pesos de las conexiones.



Pesos iniciales (aleatorios):

$$W_{o_{111}} = 0.2$$

$$W_{o_{121}} = 0.4$$

$$W_{h_{111}} = 0.1$$

$$W_{h_{121}} = -0.7$$

$$W_{h_{211}} = 0.5$$


$$W_{h_{221}} = 0.3$$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

$d(x)$



$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

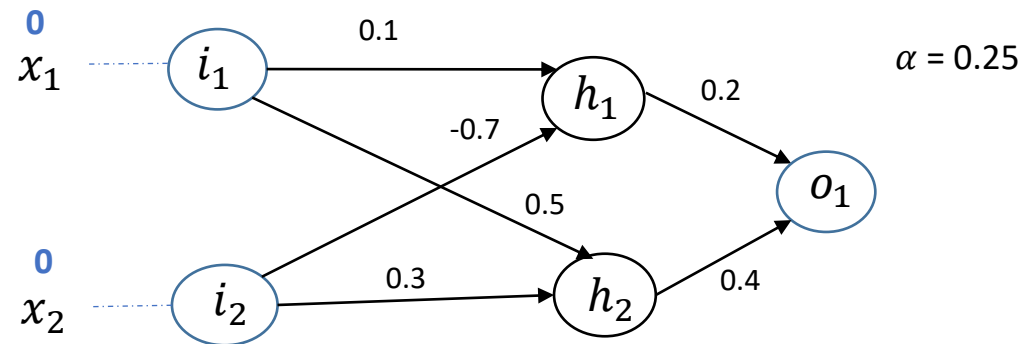
Cálculos para el patrón #1:

$$h_1 = 0.5$$

$$h_2 = 0.5$$

$$o_1 = o_k = 0.552$$

**Paso #2:** Para cada patrón de entrenamiento, calcular la salida de cada neurona desde la capa de entrada (i), a través de la capa oculta (h), hasta la capa de salida (o).



**(a) Neurona  $h_1$ :**

$$\text{Entrada: } 0.1 * 0 + (-0.7) * 0 = 0$$

$$\text{Salida (función de transferencia): } h_1 = 1/(1+e^0) = 0.5$$

**(b) Neurona  $h_2$ :**

$$\text{Entrada: } 0.5 * 0 + (0.3) * 0 = 0$$

$$\text{Salida: } h_2 = 1/(1+e^0) = 0.5$$

**(c) Neurona  $o_1$ :**

$$\text{Entrada: } 0.2 * 0.5 + (0.4) * 0.5 = 0.21$$

$$\text{Salida: } o_1 = 1/(1+e^{0.21}) = 0.552$$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

$d(x)$

$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Cálculos para el patrón #1:

$$h_1 = 0.5$$

$$h_2 = 0.5$$

$$o_1 = o_k = 0.552$$

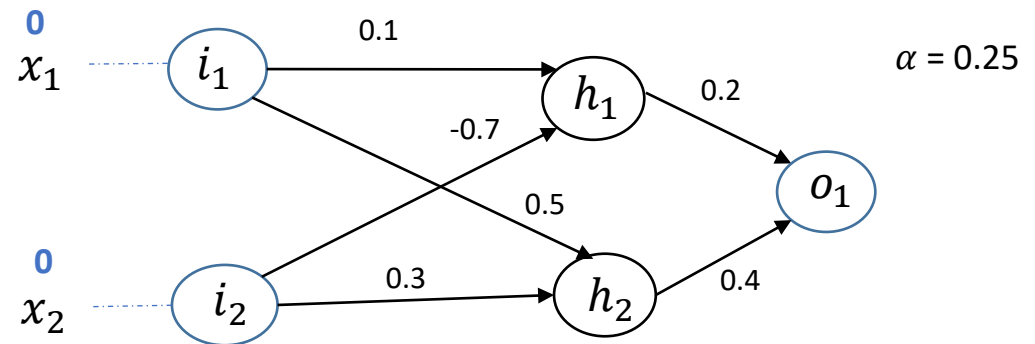
$$\Delta o_k = -0.136$$

Nuevos pesos:

$$W_{o_{11}} = 0.183$$

$$W_{o_{12}} = 0.383$$

**Paso #3: Calcular el error** para la(s) neurona(s) de salida y sus nuevos pesos (tenemos una sola neurona de salida  $o_1$ ).



**Error Neurona  $o_1$ :** Objetivo o  $d(x)$

$$\Delta o_k = o_k * (1 - o_k) * (T_k - o_k)$$

$$\Delta o_k = 0.552 * (1 - 0.552) * (0 - 0.552) = -0.136$$

**Nuevos pesos para Neurona  $o_1$ :**  $W_{o_{1jk}} = W_{o_{1jk}} + \alpha * h_j * \Delta o_k$

$$W_{o_{11}} = 0.2 + 0.25 * 0.5 * -0.136 = 0.183$$

$$W_{o_{12}} = 0.4 + 0.25 * 0.5 * -0.136 = 0.383$$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

$d(x)$

$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Cálculos para el patrón #1:

$$h_1 = 0.5$$

$$h_2 = 0.5$$

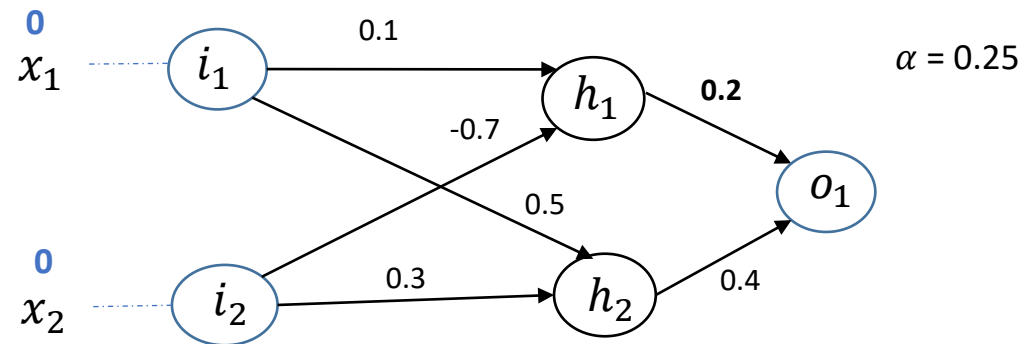
$$o_1 = 0.552$$

$$\Delta o_k = -0.136$$

Nuevos pesos:

$$\begin{array}{l|l} W_{o_{11}} = 0.183 & W_{h_{11}} = 0.1 \\ W_{o_{12}} = 0.383 & W_{h_{12}} = -0.7 \end{array}$$

**Paso #4a:** Calcular el error para la(s) neurona(s) de la(s) capa(s) oculta(s) y sus nuevos pesos (tenemos dos neuronas en la capa oculta  $h_1, h_2$ ).



**Error Neurona  $h_1$ :**

$$\Delta h_1 = h_1 * (1 - h_1) * (W_{o_{11}} * \Delta o_k)$$

$$\Delta h_1 = 0.5 * (1 - 0.5) * (0.2 * -0.136) = -0.006$$

**Nuevos pesos para Neurona  $h_1$ :**  $Wh_{1jk} = Wh_{1jk} + \alpha * i_j * \Delta h_1$

$$Wh_{111} = 0.1 + 0.25 * 0 * -0.006 = 0.1$$

$$Wh_{121} = -0.7 + 0.25 * 0 * -0.006 = -0.7$$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

$d(x)$

$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Cálculos para el patrón #1:

$$h_1 = 0.5$$

$$h_2 = 0.5$$

$$o_1 = 0.552$$

$$\Delta o_k = -0.136$$

Nuevos pesos:

$$W_{o_{111}} = 0.183$$

$$W_{h_{111}} = 0.1$$

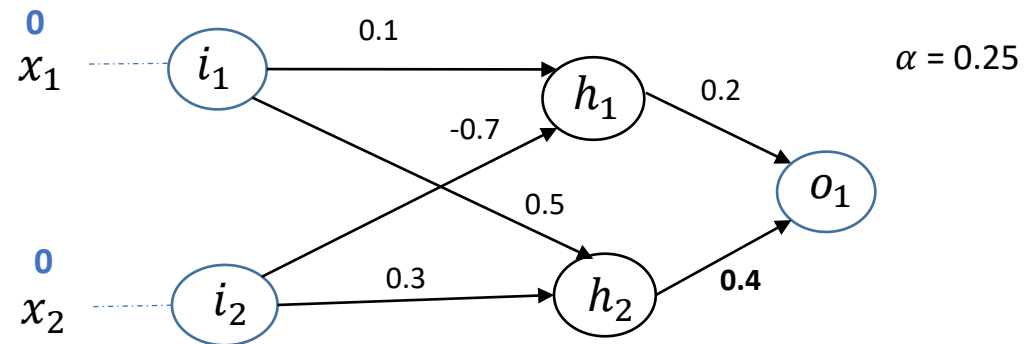
$$W_{h_{211}} = 0.5$$

$$W_{o_{121}} = 0.383$$

$$W_{h_{121}} = -0.7$$

$$W_{h_{221}} = 0.3$$

**Paso #4b:** Calcular el error para la(s) neurona(s) de la(s) capa(s) oculta(s) y sus nuevos pesos (tenemos dos neuronas en la capa oculta  $h_1, h_2$ ).



**Error Neurona  $h_2$ :**

$$\Delta h_2 = h_2 * (1 - h_2) * (W_{o_{121}} * \Delta o_k)$$

$$\Delta h_2 = 0.5 * (1 - 0.5) * (0.4 * -0.136) = -0.013$$

**Nuevos pesos para Neurona  $h_2$ :**  $Wh_{2jk} = Wh_{2jk} + \alpha * i_j * \Delta h_2$

$$Wh_{211} = 0.5 + 0.25 * 0 * -0.013 = 0.5$$

$$Wh_{221} = 0.3 + 0.25 * 0 * -0.013 = 0.3$$



### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

$d(x)$

$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Cálculos para el patrón #1:

$$h_1 = 0.5$$

$$h_2 = 0.5$$

$$o_1 = 0.552$$

$$\Delta o_k = -0.136$$

Nuevos pesos:

$$W_{o_{11}} = 0.183$$

$$W_{h_{11}} = 0.1$$

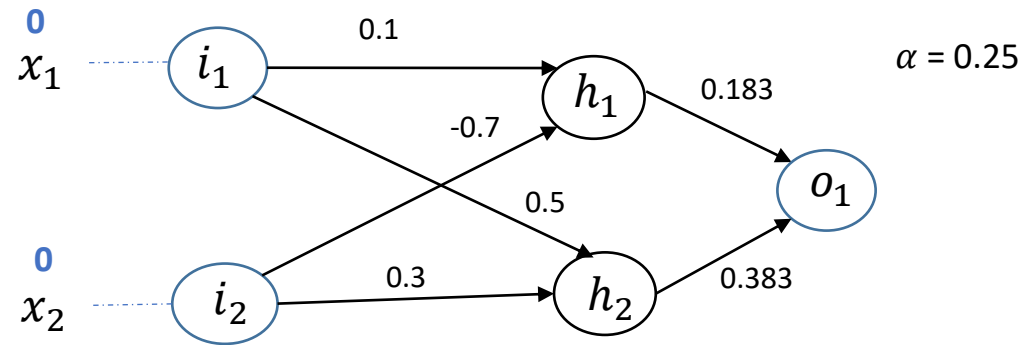
$$W_{h_{21}} = 0.5$$

$$W_{o_{12}} = 0.383$$

$$W_{h_{12}} = -0.7$$

$$W_{h_{22}} = 0.3$$

**Paso #5:** Aplicar los ajustes de peso. Obtenemos una red con nuevos pesos.



- Repetimos desde el **paso #2** para el siguiente patrón de datos de entrada (hasta reducir el error o llegar a la ultima iteración).

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

$d(x)$

$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

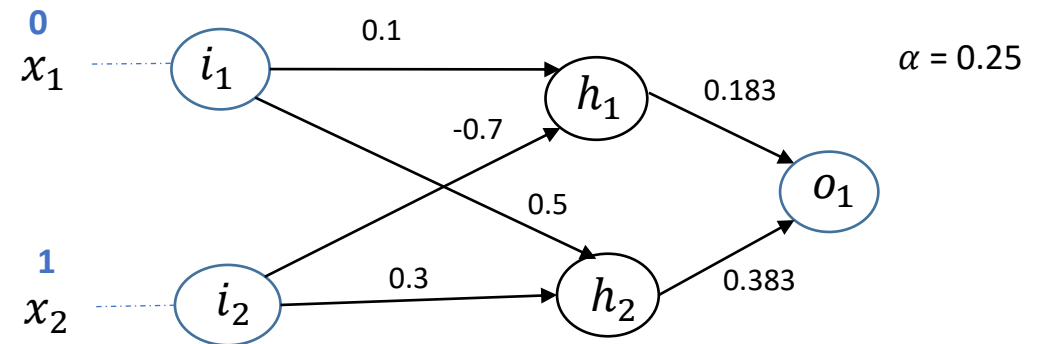
Cálculos para el patrón #2:

$$h_1 = 0.331$$

$$h_2 = 0.574$$

$$o_1 = 0.569$$

**Paso #2:** Para cada patrón de entrenamiento, calcular la salida de cada neurona desde la capa de entrada (i), a través de la capa oculta (h), hasta la capa de salida (o).



**(a) Neurona  $h_1$ :**

$$\text{Entrada: } 0.1 * 0 + (-0.7) * 1 = -0.7$$

$$\text{Salida: } h_1 = 1/(1+e^{-0.7}) = 0.331$$

**(b) Neurona  $h_2$ :**

$$\text{Entrada: } 0.5 * 0 + (0.3) * 1 = 0.3$$

$$\text{Salida: } h_2 = 1/(1+e^{-0.3}) = 0.574$$

**(c) Neurona  $o_1$ :**

$$\text{Entrada: } 0.183 * 0.331 + (0.383) * 0.574 = 0.28$$

$$\text{Salida: } o_1 = 1/(1+e^{-0.28}) = 0.569$$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

		$d(x)$
$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Cálculos para el patrón #2:

$$h_1 = 0.331$$

$$h_2 = 0.574$$

$$o_1 = o_k = 0.569$$

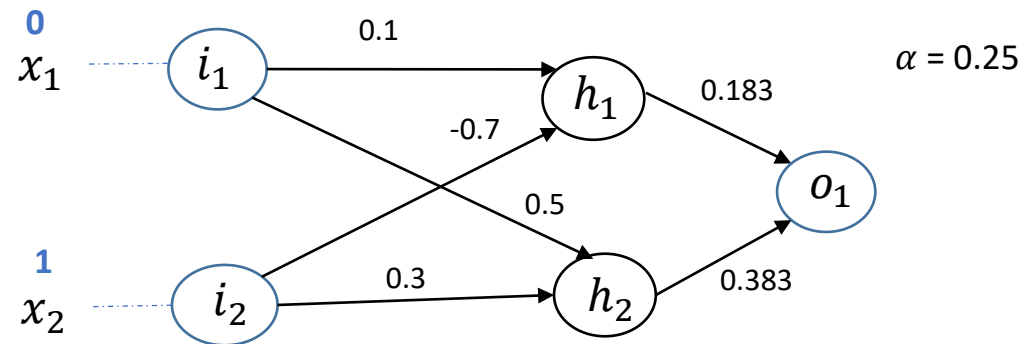
$$\Delta o_k = 0.105$$

Nuevos pesos:

$$W_{o_{11}} = 0.191$$

$$W_{o_{12}} = 0.398$$

**Paso #3: Calcular el error** para la(s) neurona(s) de salida y sus nuevos pesos (tenemos una sola neurona de salida  $o_1$ ).



**Error Neurona  $o_1$ :** Objetivo o  $d(x)$

$$\Delta o_k = o_k * (1 - o_k) * (T_k - o_k)$$

$$\Delta o_k = 0.569 * (1 - 0.569) * (1 - 0.569) = 0.105$$

**Nuevos pesos para Neurona  $o_1$ :**  $W_{o_{1jk}} = W_{o_{1jk}} + \alpha * h_j * \Delta o_k$

$$W_{o_{11}} = 0.183 + 0.25 * 0.331 * 0.105 = 0.191$$

$$W_{o_{12}} = 0.383 + 0.25 * 0.574 * 0.105 = 0.398$$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

		$d(x)$
$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Cálculos para el patrón #2:

$$h_1 = 0.331$$

$$h_2 = 0.574$$

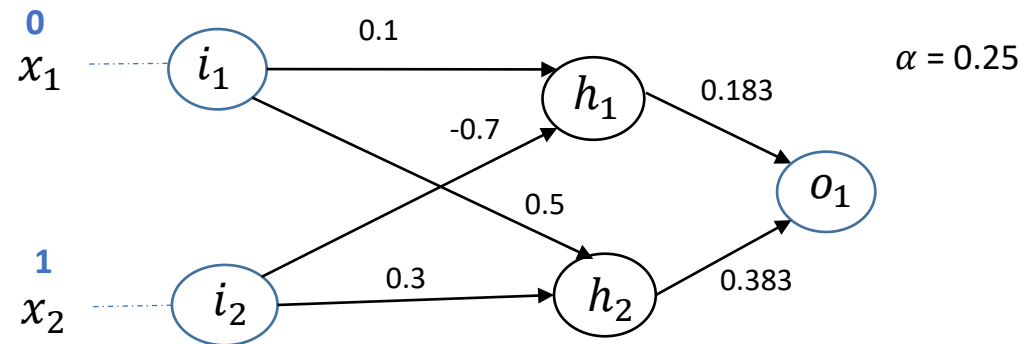
$$o_1 = 0.569$$

$$\Delta o_k = 0.105$$

Nuevos pesos:

$$\begin{array}{l|l} W_{o_{11}} = 0.191 & Wh_{11} = 0.1 \\ W_{o_{12}} = 0.398 & Wh_{12} = -0.7 \end{array}$$

**Paso #4a:** Calcular el error para la(s) neurona(s) de la(s) capa(s) oculta(s) y sus nuevos pesos (tenemos dos neuronas en la capa oculta  $h_1, h_2$ ).



**Error Neurona  $h_1$ :**

$$\Delta h_1 = h_1 * (1 - h_1) * (W_{o_{11}} * \Delta o_k)$$

$$\Delta h_1 = 0.331 * (1 - 0.331) * (0.183 * 0.105) = 0.004$$

**Nuevos pesos para Neurona  $h_1$ :**  $Wh_{1jk} = Wh_{1jk} + \alpha * i_j * \Delta h_1$

$$Wh_{111} = 0.1 + 0.25 * 0 * 0.004 = 0.1$$

$$Wh_{121} = -0.7 + 0.25 * 1 * 0.004 = -0.7$$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

$d(x)$

$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Cálculos para el patrón #2:

$$h_1 = 0.331$$

$$h_2 = 0.574$$

$$o_1 = 0.569$$

$$\Delta o_k = 0.105$$

Nuevos pesos:

$$W_{o_{11}} = 0.191$$

$$W_{h_{11}} = 0.1$$

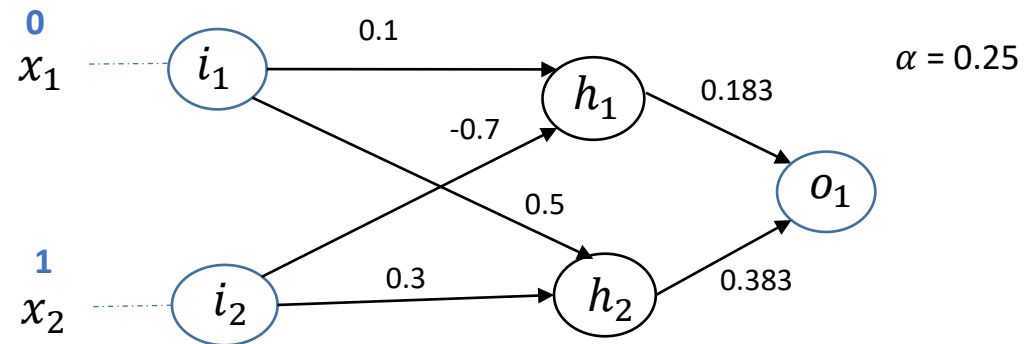
$$W_{h_{21}} = 0.5$$

$$W_{o_{12}} = 0.398$$

$$W_{h_{12}} = -0.7$$

$$W_{h_{22}} = 0.302$$

**Paso #4b:** Calcular el error para la(s) neurona(s) de la(s) capa(s) oculta(s) y sus nuevos pesos (tenemos dos neuronas en la capa oculta  $h_1, h_2$ ).



**Error Neurona  $h_2$ :**

$$\Delta h_2 = h_2 * (1 - h_2) * (W_{o_{12}} * \Delta o_k)$$

$$\Delta h_2 = 0.574 * (1 - 0.574) * (0.383 * 0.105) = 0.009$$

**Nuevos pesos para Neurona  $h_2$ :**  $W_{h_{2jk}} = W_{h_{2jk}} + \alpha * i_j * \Delta h_2$

$$W_{h_{211}} = 0.5 + 0.25 * 0 * 0.009 = 0.5$$

$$W_{h_{221}} = 0.3 + 0.25 * 1 * 0.009 = 0.302$$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

$d(x)$

$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Cálculos para el patrón #2:

$$h_1 = 0.331$$

$$h_2 = 0.574$$

$$o_1 = 0.569$$

$$\Delta o_k = 0.105$$

Nuevos pesos:

$$W_{o_{11}} = 0.191$$

$$W_{h_{11}} = 0.1$$

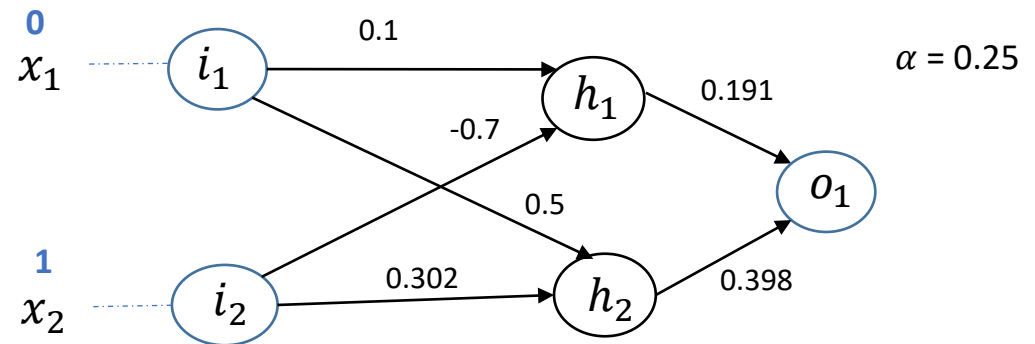
$$W_{h_{21}} = 0.5$$

$$W_{o_{12}} = 0.398$$

$$W_{h_{12}} = -0.7$$

$$W_{h_{22}} = 0.302$$

**Paso #5:** Aplicar los ajustes de peso. Obtenemos una red con nuevos pesos.



- Repetimos desde el **paso #2** para el siguiente patrón de datos de entrada (hasta reducir el error o llegar a la última iteración).

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

		$d(x)$
$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0



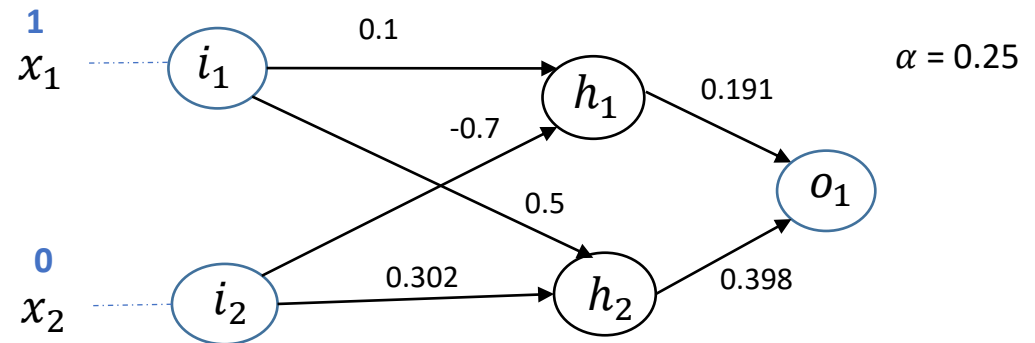
Cálculos para el patrón #3:

$$h_1 = 0.524$$

$$h_2 = 0.622$$

$$o_1 = 0.585$$

**Paso #2:** Para cada patrón de entrenamiento, calcular la salida de cada neurona desde la capa de entrada (i), a través de la capa oculta (h), hasta la capa de salida (o).



**(a) Neurona  $h_1$ :**

$$\text{Entrada: } 0.1 * 1 + (-0.7) * 0 = 0.1$$

$$\text{Salida: } h_1 = 1/(1+e^{-0.1}) = 0.524$$

**(b) Neurona  $h_2$ :**

$$\text{Entrada: } 0.5 * 1 + (0.302) * 0 = 0.5$$

$$\text{Salida: } h_2 = 1/(1+e^{-0.5}) = 0.622$$

**(c) Neurona  $o_1$ :**

$$\text{Entrada: } 0.191 * 0.524 + (0.398) * 0.622 = 0.347$$

$$\text{Salida: } o_1 = 1/(1+e^{-0.347}) = 0.585$$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

		$d(x)$
$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0



Cálculos para el patrón #3:

$$h_1 = 0.524$$

$$h_2 = 0.622$$

$$o_1 = 0.585$$

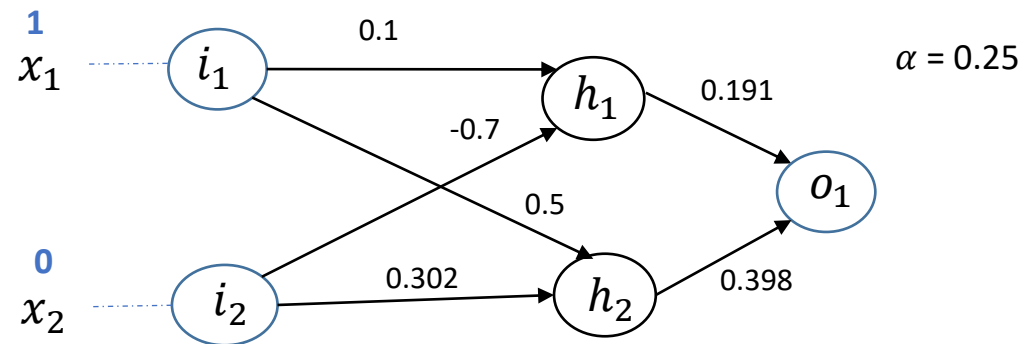
$$\Delta o_k = 0.1$$

Nuevos pesos:

$$W_{o_{11}} = 0.204$$

$$W_{o_{12}} = 0.413$$

**Paso #3: Calcular el error** para la(s) neurona(s) de salida y sus nuevos pesos (tenemos una sola neurona de salida  $o_1$ ).



**Error Neurona  $o_1$ :** Objetivo o  $d(x)$

$$\Delta o_k = o_k * (1 - o_k) * (T_k - o_k)$$

$$\Delta o_k = 0.585 * (1 - 0.585) * (1 - 0.585) = 0.1$$

**Nuevos pesos para Neurona  $o_1$ :**  $W_{o_{1jk}} = W_{o_{1jk}} + \alpha * h_j * \Delta o_k$

$$W_{o_{11}} = 0.191 + 0.25 * 0.524 * 0.1 = 0.204$$

$$W_{o_{12}} = 0.398 + 0.25 * 0.622 * 0.1 = 0.413$$



### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

		$d(x)$
$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Cálculos para el patrón #3:

$$h_1 = 0.524$$

$$h_2 = 0.622$$

$$o_1 = 0.585$$

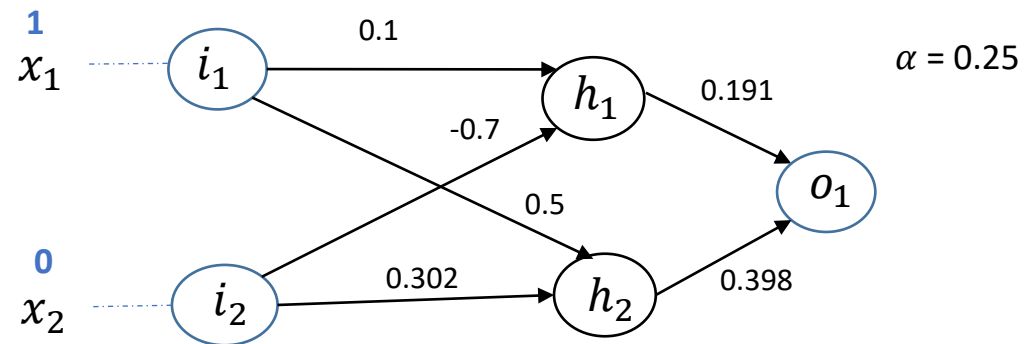
$$\Delta o_k = 0.1$$

Nuevos pesos:

$$W_{o_{11}} = 0.204 \quad Wh_{111} = 0.101$$

$$W_{o_{12}} = 0.413 \quad Wh_{121} = -0.7$$

**Paso #4a:** Calcular el error para la(s) neurona(s) de la(s) capa(s) oculta(s) y sus nuevos pesos (tenemos dos neuronas en la capa oculta  $h_1, h_2$ ).



**Error Neurona  $h_1$ :**

$$\Delta h_1 = h_1 * (1 - h_1) * (W_{o_{11}} * \Delta o_k)$$

$$\Delta h_1 = 0.524 * (1 - 0.524) * (0.191 * 0.1) = 0.005$$

**Nuevos pesos para Neurona  $h_1$ :**  $Wh_{1jk} = Wh_{1jk} + \alpha * i_j * \Delta h_1$

$$Wh_{111} = 0.1 + 0.25 * 1 * 0.005 = 0.101$$

$$Wh_{121} = -0.7 + 0.25 * 0 * 0.005 = -0.7$$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

		$d(x)$
$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Cálculos para el patrón #3:

$$h_1 = 0.524$$

$$h_2 = 0.622$$

$$o_1 = 0.585$$

$$\Delta o_k = 0.1$$

Nuevos pesos:

$$W_{o_{111}} = 0.204$$

$$W_{h_{111}} = 0.101$$

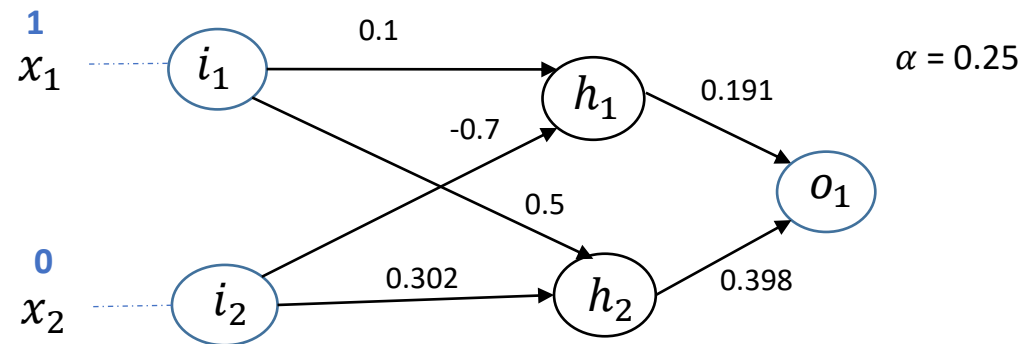
$$W_{h_{211}} = 0.502$$

$$W_{o_{121}} = 0.413$$

$$W_{h_{121}} = -0.7$$

$$W_{h_{221}} = 0.302$$

**Paso #4b:** Calcular el error para la(s) neurona(s) de la(s) capa(s) oculta(s) y sus nuevos pesos (tenemos dos neuronas en la capa oculta  $h_1, h_2$ ).



**Error Neurona  $h_2$ :**

$$\Delta h_2 = h_2 * (1 - h_2) * (W_{o_{121}} * \Delta o_k)$$

$$\Delta h_2 = 0.622 * (1 - 0.622) * (0.398 * 0.1) = 0.009$$

**Nuevos pesos para Neurona  $h_2$ :**  $Wh_{2jk} = Wh_{2jk} + \alpha * i_j * \Delta h_2$

$$Wh_{211} = 0.5 + 0.25 * 1 * 0.009 = 0.502$$

$$Wh_{221} = 0.302 + 0.25 * 0 * 0.009 = 0.302$$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

$d(x)$

$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Cálculos para el patrón #3:

$$h_1 = 0.524$$

$$h_2 = 0.622$$

$$o_1 = 0.585$$

$$\Delta o_k = 0.1$$

Nuevos pesos:

$$W_{o_{11}} = 0.204$$

$$W_{h_{11}} = 0.101$$

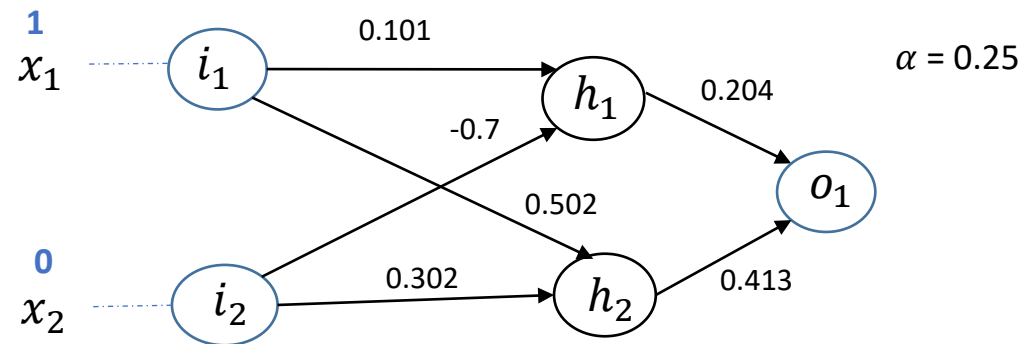
$$W_{h_{21}} = 0.502$$

$$W_{o_{12}} = 0.413$$

$$W_{h_{12}} = -0.7$$

$$W_{h_{22}} = 0.302$$

**Paso #5:** Aplicar los ajustes de peso. Obtenemos una red con nuevos pesos.



- Repetimos desde el **paso #2** para el siguiente patrón de datos de entrada (hasta reducir el error o llegar a la ultima iteración).

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

		$d(x)$
$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0



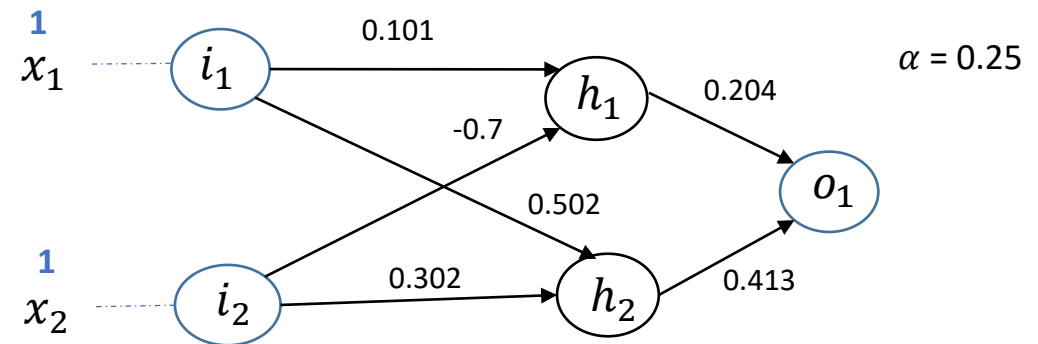
Cálculos para el patrón #4:

$$h_1 = 0.354$$

$$h_2 = 0.690$$

$$o_1 = 0.588$$

**Paso #2:** Para cada patrón de entrenamiento, calcular la salida de cada neurona desde la capa de entrada (i), a través de la capa oculta (h), hasta la capa de salida (o).



**(a) Neurona  $h_1$ :**

$$\text{Entrada: } 0.101 * 1 + (-0.7) * 1 = -0.599$$

$$\text{Salida: } h_1 = 1/(1+e^{-0.599}) = 0.354$$

**(b) Neurona  $h_2$ :**

$$\text{Entrada: } 0.502 * 1 + (0.302) * 1 = 0.804$$

$$\text{Salida: } h_2 = 1/(1+e^{-0.804}) = 0.690$$

**(c) Neurona  $o_1$ :**

$$\text{Entrada: } 0.204 * 0.354 + (0.413) * 0.69 = 0.357$$

$$\text{Salida: } o_1 = 1/(1+e^{-0.357}) = 0.588$$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

		$d(x)$
$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0



Cálculos para el patrón #4:

$$h_1 = 0.354$$

$$h_2 = 0.690$$

$$o_1 = 0.588$$

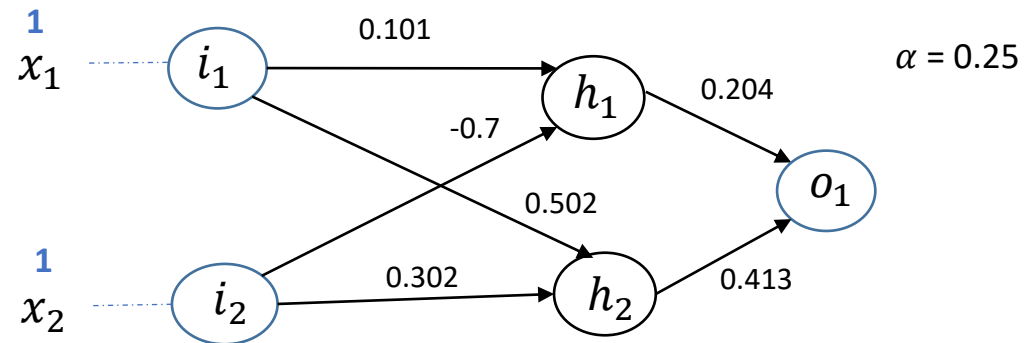
$$\Delta o_k = -0.142$$

Nuevos pesos:

$$W_{o_{111}} = 0.191$$

$$W_{o_{121}} = 0.388$$

**Paso #3: Calcular el error** para la(s) neurona(s) de salida y sus nuevos pesos (tenemos una sola neurona de salida  $o_1$ ).



**Error Neurona  $o_1$ :** Objetivo o  $d(x)$

$$\Delta o_k = o_k * (1 - o_k) * (T_k - o_k)$$

$$\Delta o_k = 0.588 * (1 - 0.588) * (0 - 0.588) = -0.142$$

**Nuevos pesos para Neurona  $o_1$ :**  $W_{o_{1jk}} = W_{o_{1jk}} + \alpha * h_j * \Delta o_k$

$$W_{o_{111}} = 0.204 + 0.25 * 0.354 * -0.142 = 0.191$$

$$W_{o_{121}} = 0.413 + 0.25 * 0.690 * -0.142 = 0.388$$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

		$d(x)$
$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0



Cálculos para el patrón #4:

$$h_1 = 0.354$$

$$h_2 = 0.690$$

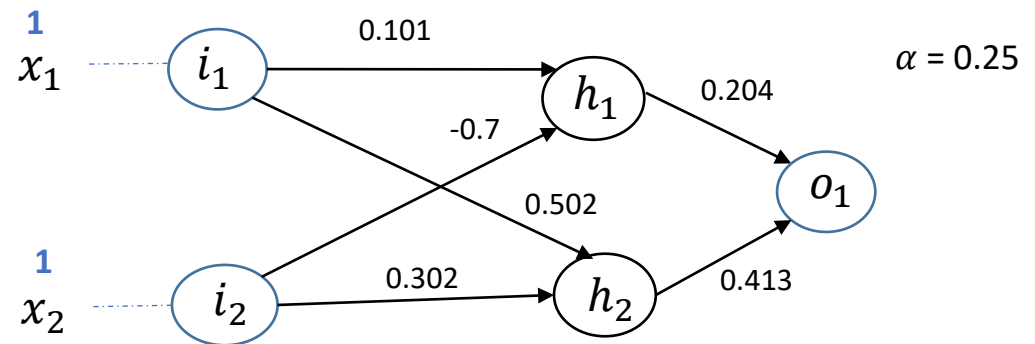
$$o_1 = 0.588$$

$$\Delta o_k = -0.142$$

Nuevos pesos:

$$\begin{array}{l|l} W_{o_{111}} = 0.191 & Wh_{111} = 0.099 \\ W_{o_{121}} = 0.388 & Wh_{121} = -0.701 \end{array}$$

**Paso #4a:** Calcular el error para la(s) neurona(s) de la(s) capa(s) oculta(s) y sus nuevos pesos (tenemos dos neuronas en la capa oculta  $h_1, h_2$ ).



**Error Neurona  $h_1$ :**

$$\Delta h_1 = h_1 * (1 - h_1) * (W_{o_{111}} * \Delta o_k)$$

$$\Delta h_1 = 0.354 * (1 - 0.354) * (0.204 * -0.142) = -0.006$$

**Nuevos pesos para Neurona  $h_1$ :**  $Wh_{1jk} = Wh_{1jk} + \alpha * i_j * \Delta h_1$

$$Wh_{111} = 0.101 + 0.25 * 1 * -0.006 = 0.099$$

$$Wh_{121} = -0.7 + 0.25 * 1 * -0.006 = -0.701$$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

		$d(x)$
$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0



Cálculos para el patrón #4:

$$h_1 = 0.354$$

$$h_2 = 0.690$$

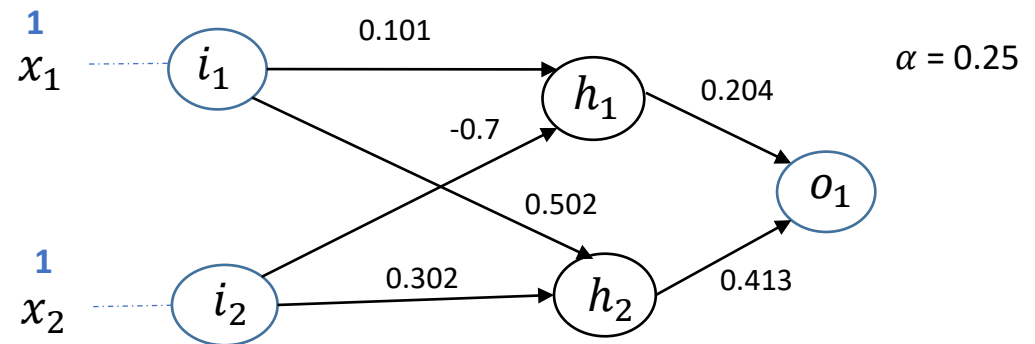
$$o_1 = 0.588$$

$$\Delta o_k = -0.142$$

Nuevos pesos:

$W_{o_{111}} = 0.191$	$W_{h_{111}} = 0.099$	$W_{h_{211}} = 0.499$
$W_{o_{121}} = 0.388$	$W_{h_{121}} = -0.701$	$W_{h_{221}} = 0.299$

**Paso #4b:** Calcular el error para la(s) neurona(s) de la(s) capa(s) oculta(s) y sus nuevos pesos (tenemos dos neuronas en la capa oculta  $h_1, h_2$ ).



**Error Neurona  $h_2$ :**

$$\Delta h_2 = h_2 * (1 - h_2) * (W_{o_{121}} * \Delta o_k)$$

$$\Delta h_2 = 0.690 * (1 - 0.690) * (0.413 * -0.142) = -0.012$$

**Nuevos pesos para Neurona  $h_2$ :**  $Wh_{2jk} = Wh_{2jk} + \alpha * i_j * \Delta h_2$

$$Wh_{211} = 0.502 + 0.25 * 1 * -0.012 = 0.499$$

$$Wh_{221} = 0.302 + 0.25 * 1 * -0.012 = 0.299$$

### 3. Backpropagation: Ejemplo para Función lógica XOR

#### EJEMPLO: Backpropagation para Función lógica XOR

Datos:

$d(x)$

$x_1$	$x_2$	XOR
0	0	0
0	1	1
1	0	1
1	1	0



Cálculos para el patrón #4:

$$h_1 = 0.354$$

$$h_2 = 0.690$$

$$o_1 = 0.588$$

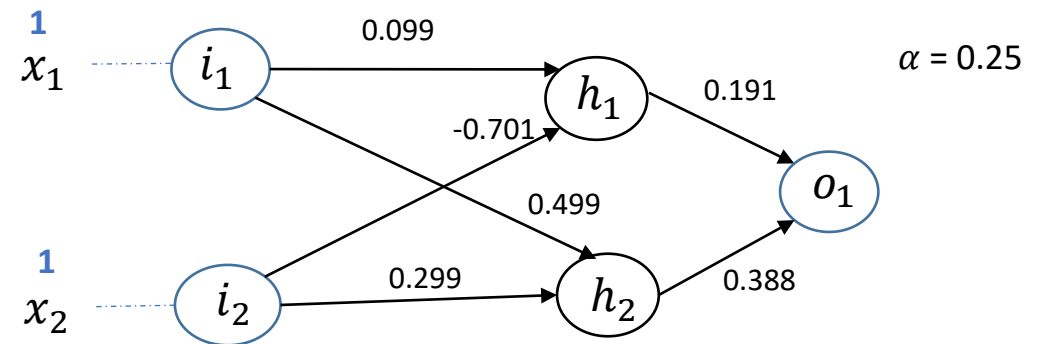
$$\Delta o_k = -0.142$$

Nuevos pesos:

$$W_{o_{11}} = 0.191 \quad W_{h_{11}} = 0.099 \quad W_{h_{21}} = 0.499$$

$$W_{o_{12}} = 0.388 \quad W_{h_{12}} = -0.701 \quad W_{h_{22}} = 0.299$$

**Paso #5:** Aplicar los ajustes de peso. Obtenemos una red con nuevos pesos.



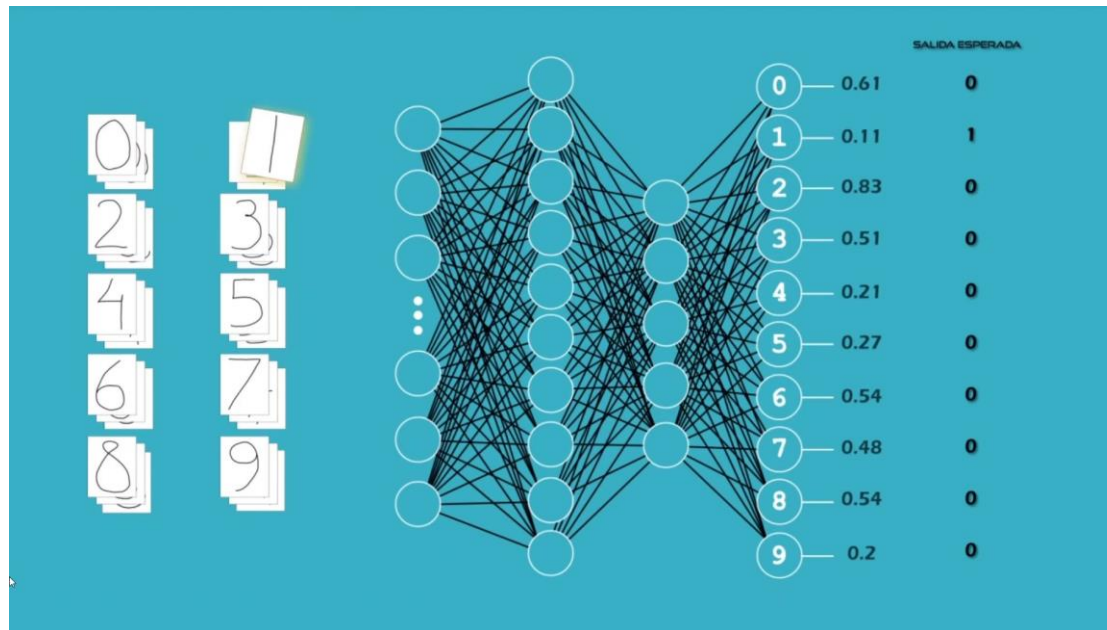
- Hemos concluido la interacción #1 para el conjunto de datos.
- Repetimos desde el **paso #2** para el primer patrón de datos de entrada (hasta reducir el error o llegar a la ultima iteración).



## 4. Práctica: Modelo de Clasificación de Imágenes con RNA

### Backpropagation

### Ejemplo MNIST: Predicción de dígitos numéricos manuscritos en Python



Enlace: <https://youtu.be/f0tBfcE13DY> (Duración: 3 min.)

# PREGUNTAS

Dudas y opiniones