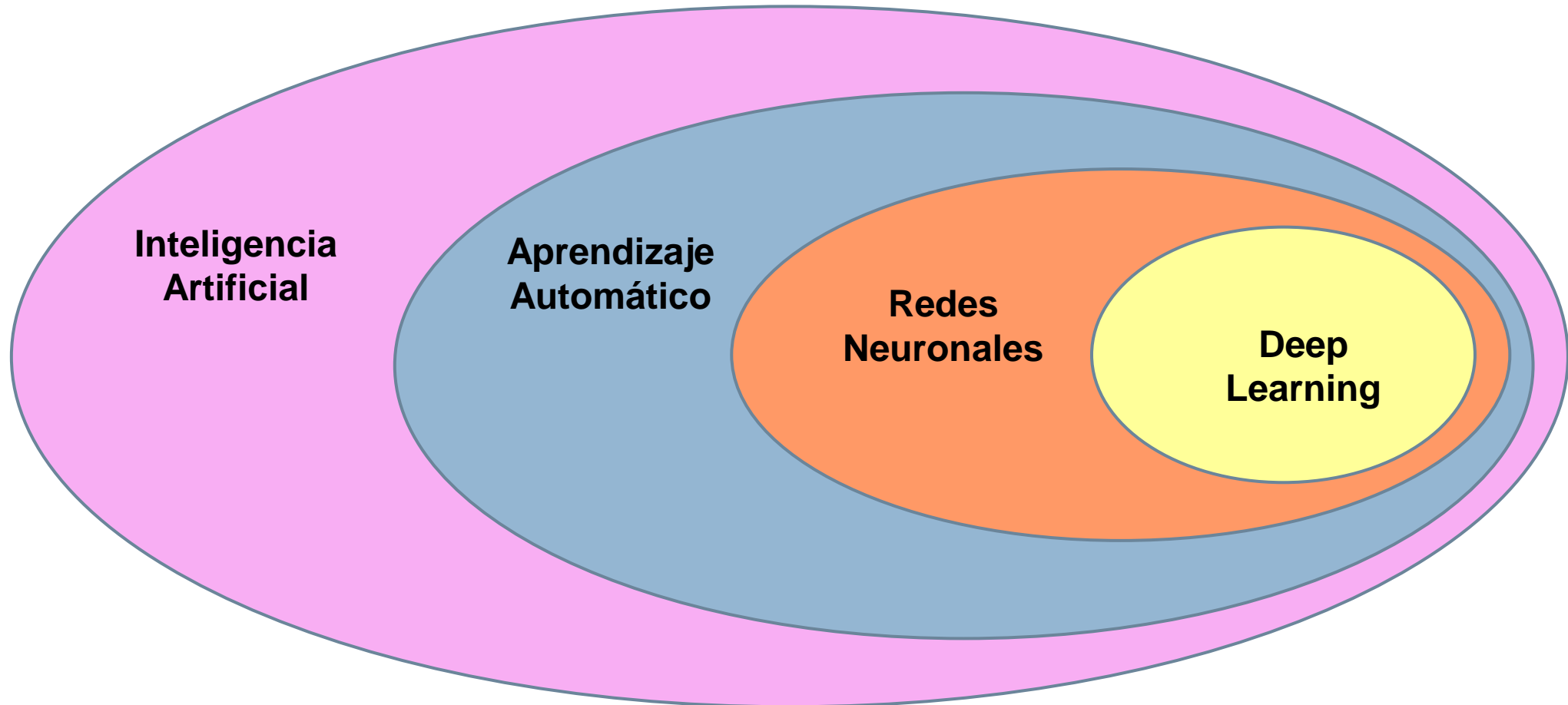
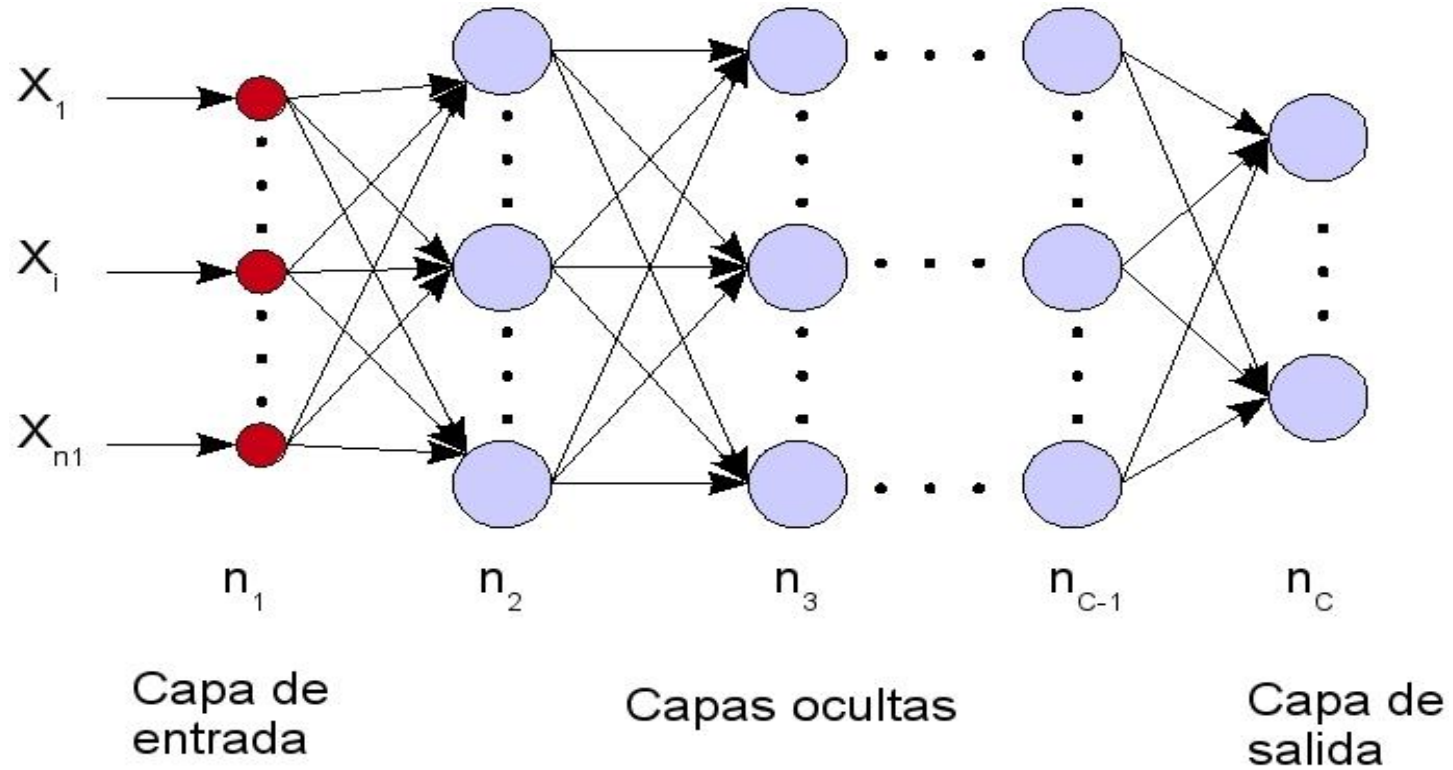


Redes Neuronales y Aprendizaje Profundo

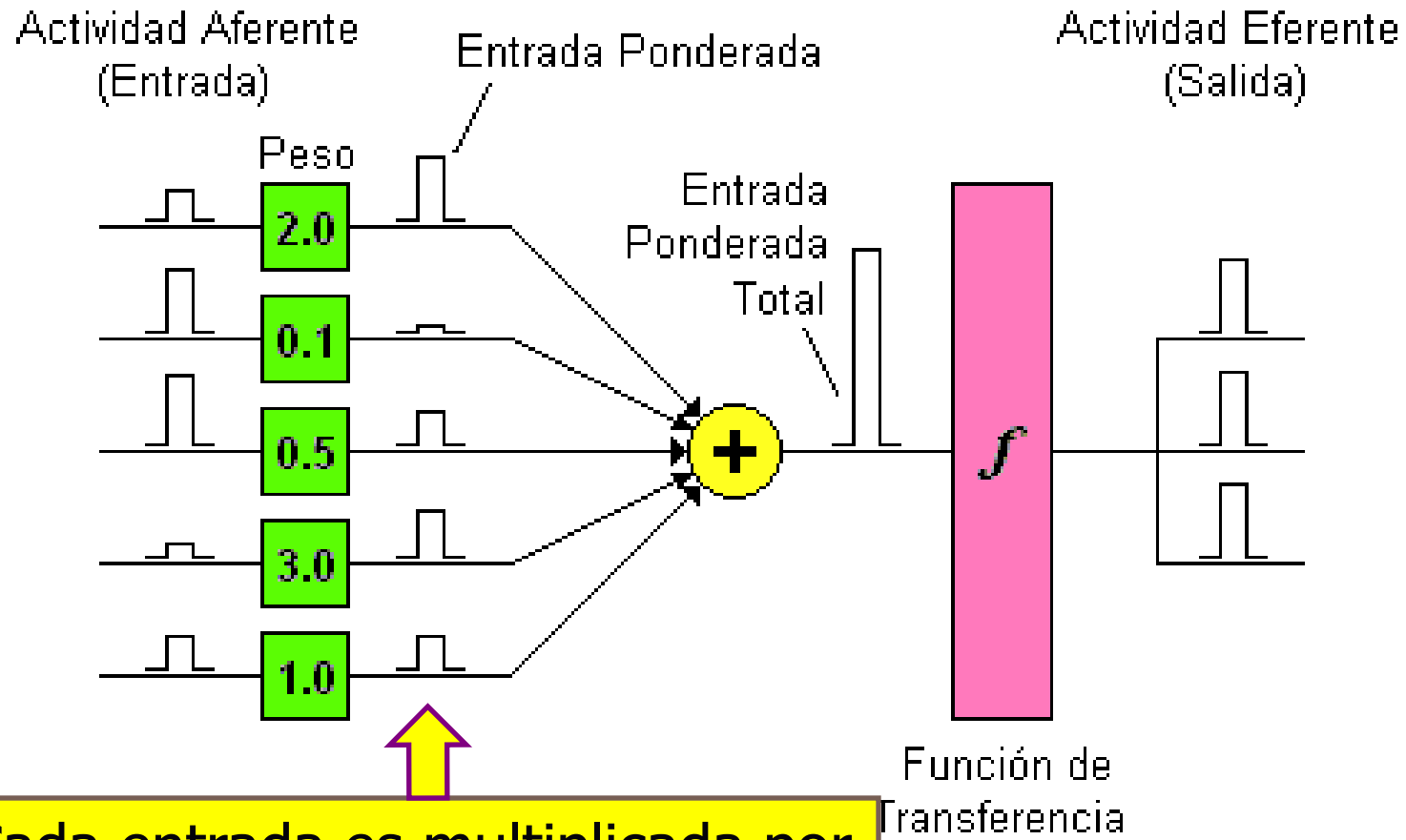


Red neuronal feedforward



Neurona Artificial

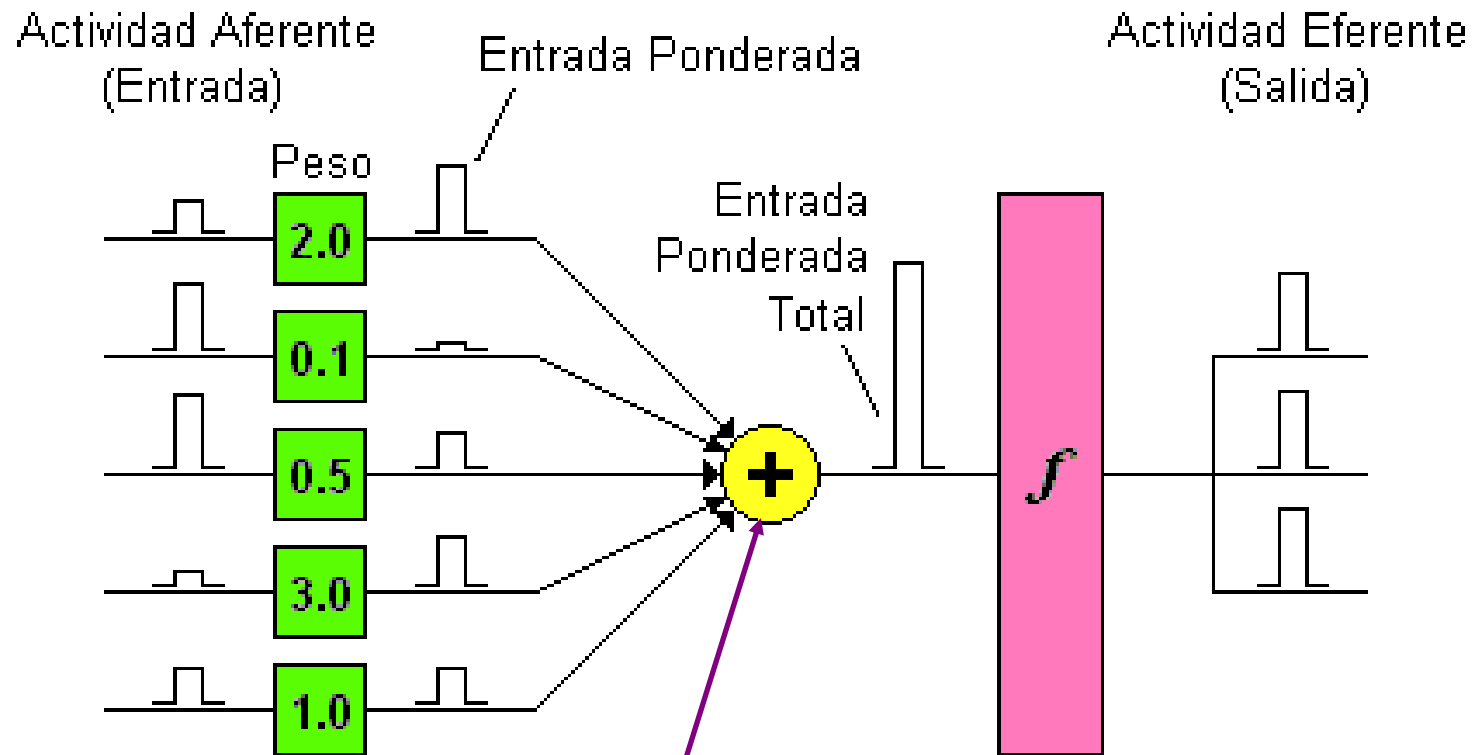
3



Cada entrada es multiplicada por el peso de arco correspondiente.

Neurona Artificial

4

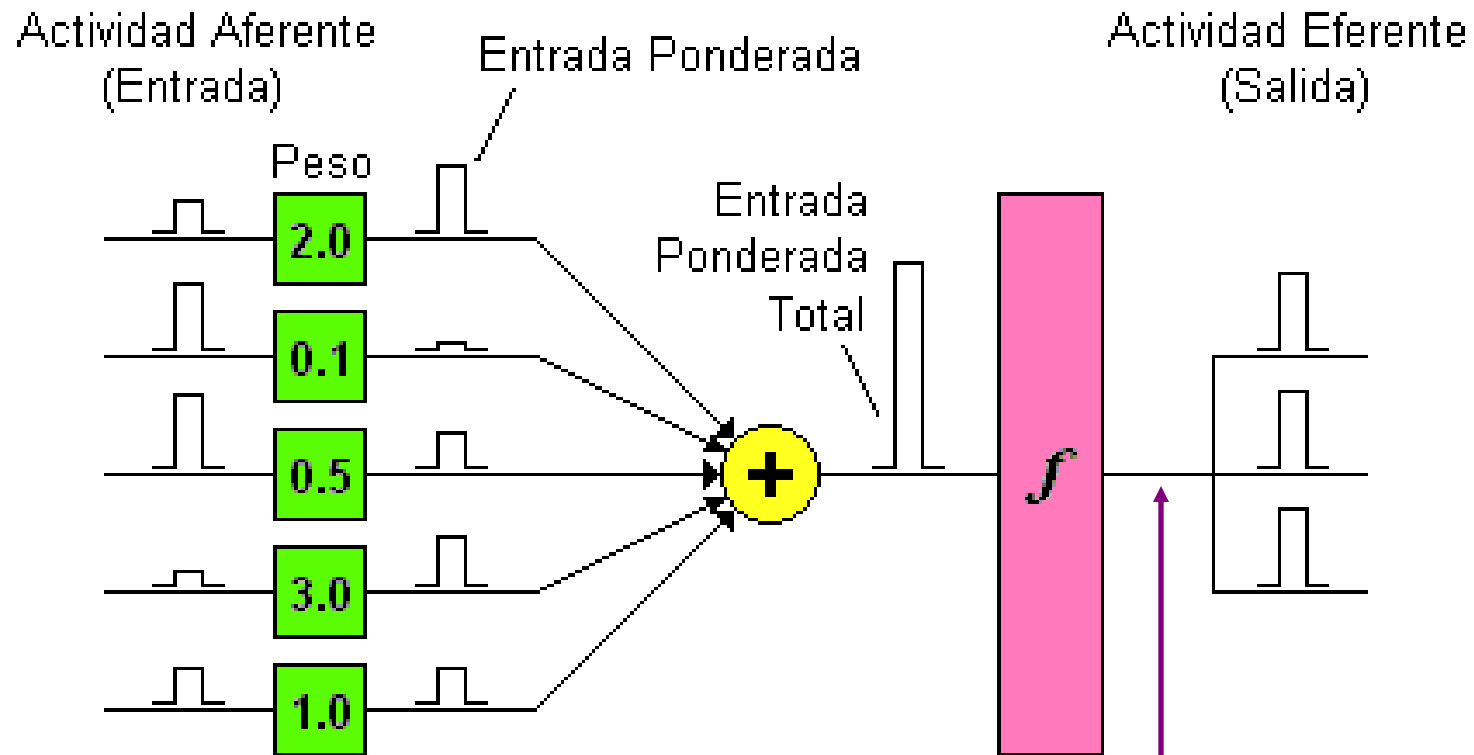


Cada neurona calcula su entrada neta como:

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

Neurona Artificial

5

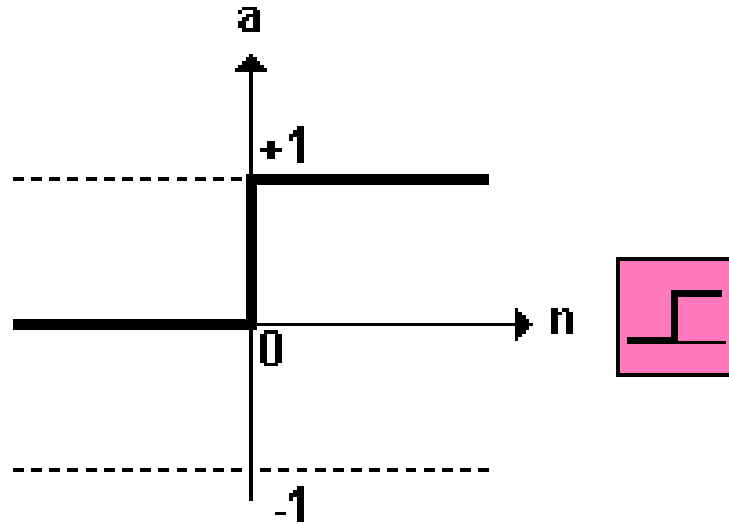


El valor de salida (único) se
obtiene como $y = f(neta)$

Funciones de Transferencia

6

□ Función umbral binaria

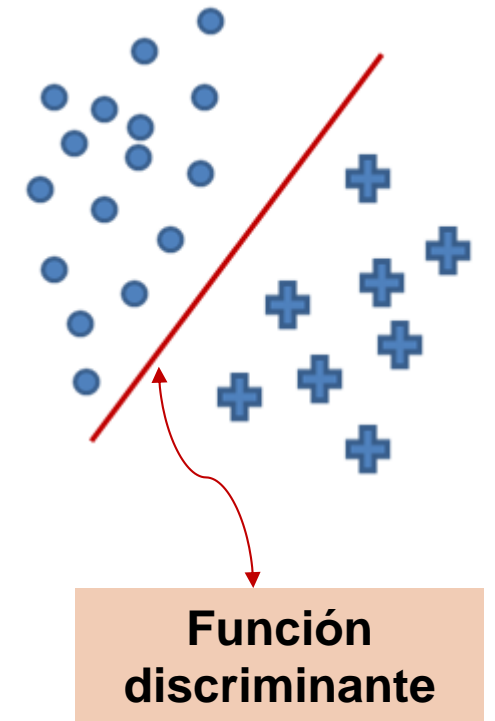


$$a = \begin{cases} 1 & \text{si } n \geq 0 \\ 0 & \text{si } n < 0 \end{cases}$$

Esta función crea neuronas que clasifican las entradas en dos categorías diferentes.

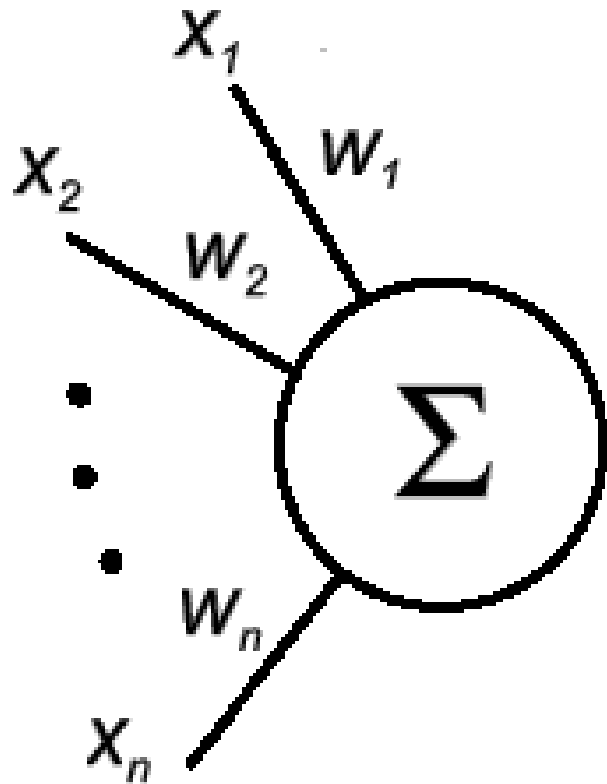
La primera RN – El Perceptrón

- Está formada por una única neurona.
- Utiliza aprendizaje supervisado.
- Su regla de aprendizaje es una modificación de la propuesta por Hebb.
- Se adapta teniendo en cuenta el error entre la salida que da la red y la salida esperada.
- Representa una única función discriminante que separa **linealmente** los ejemplos en **dos** clases.

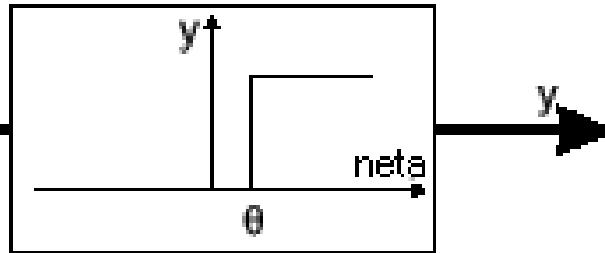


Perceptrón

8

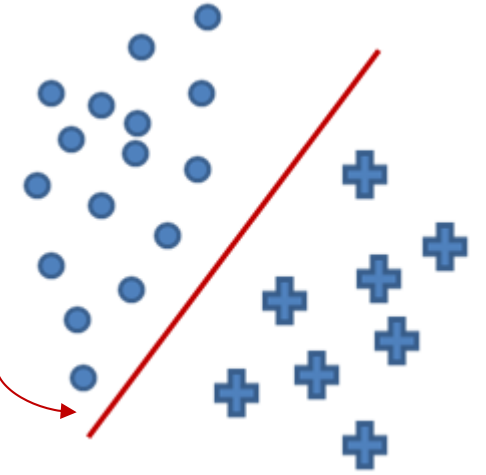


$$neta = \sum_i x_i w_i$$



$$y = \begin{cases} 1 & \text{si } neta \geq \theta \\ 0 & \text{si } neta < \theta \end{cases}$$

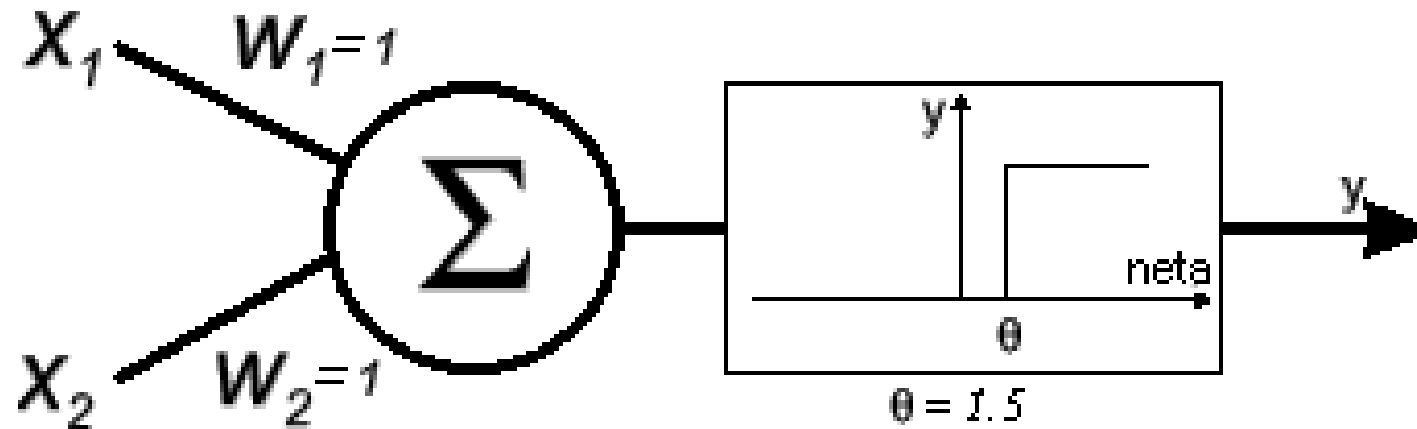
$$\sum_i x_i w_i = \theta$$



Ejemplo

9

- Verifique si la siguiente red neuronal se comporta como la función lógica AND



AND

10

$$\theta = 1.5$$

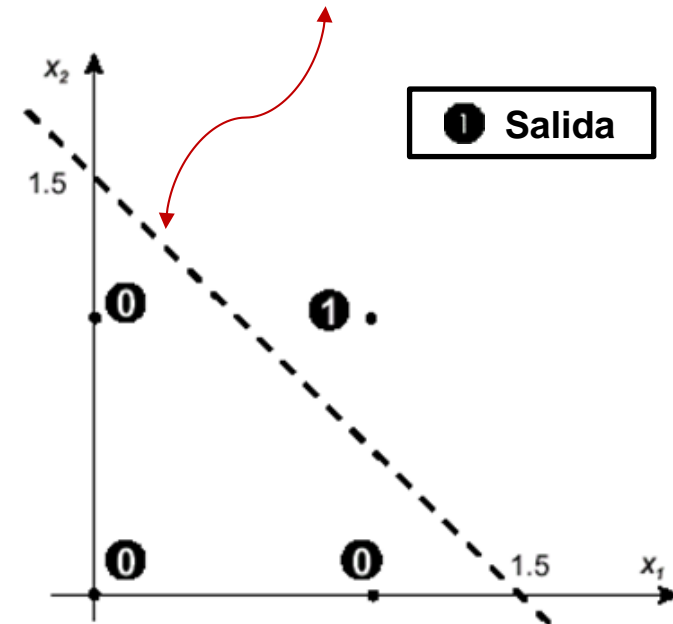
x_1	x_2	neta	salida
0	0	0	0
0	1	1	0
1	0	1	0
1	1	2	1

Graficar la función discriminante
(recta)

Función discriminante

$$x_1 w_1 + x_2 w_2 = \theta$$

$$x_1 + x_2 = 1.5$$



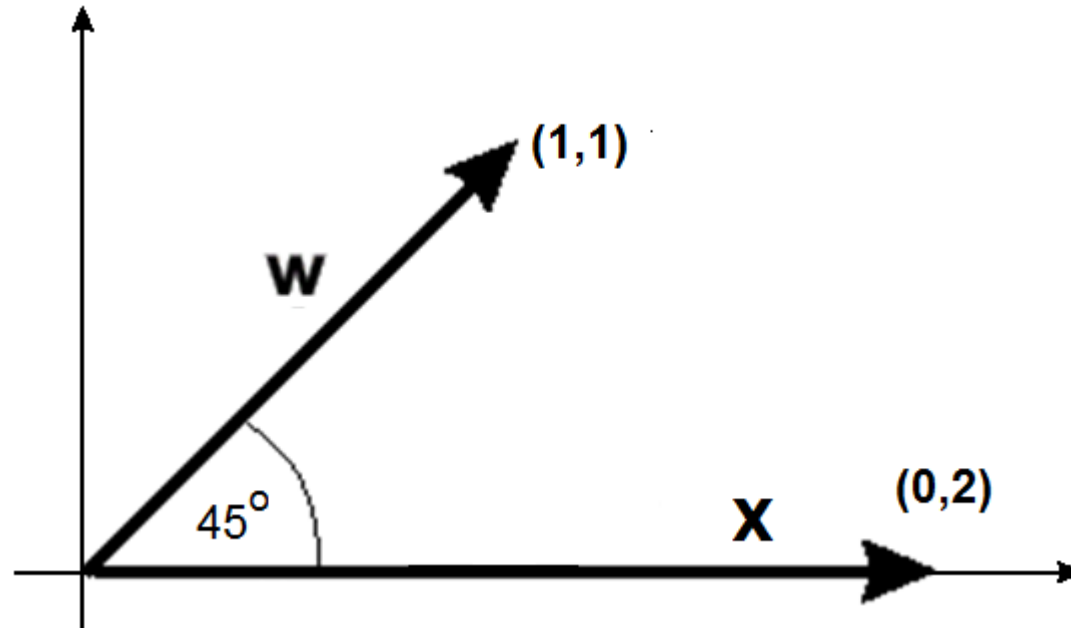
Entrenamiento del perceptrón

11

- Se busca una estrategia iterativa que permita adaptar los valores de las conexiones a medida que se presentan los datos de entrada.
- Ver que el estímulo de entrada se corresponde con el producto interior de los vectores X y W .

Producto interior

12

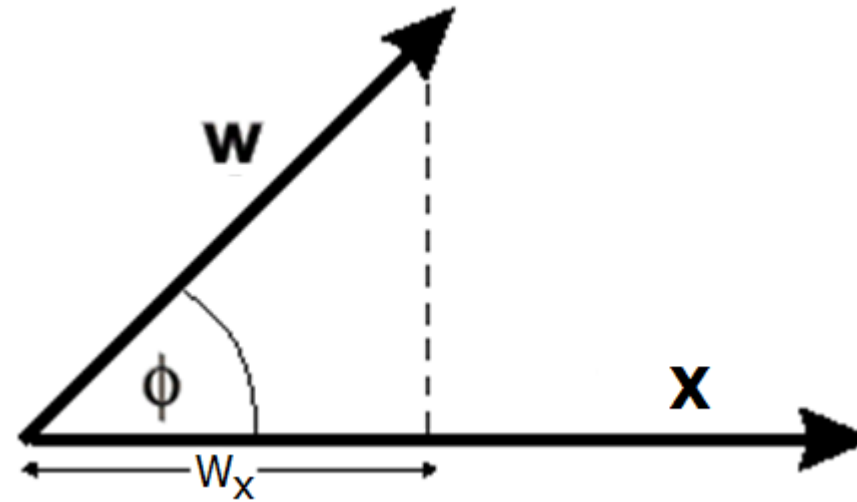


$$w \cdot x = \|w\| \cdot \|x\| \cdot \cos(\phi)$$

$$w \cdot x = \sum_{i=1}^n w_i x_i$$

Vector de proyección

13

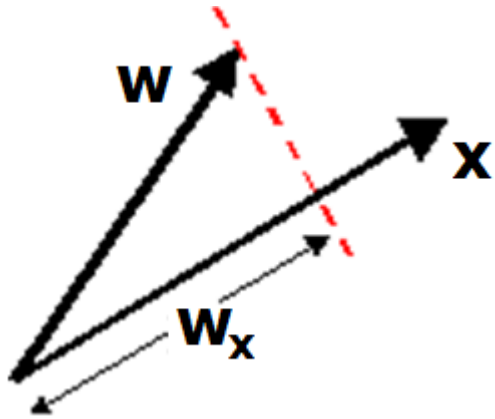


$$w_x = \|w\| \cdot \cos(\phi)$$

$$w_x \cdot \|x\| = w \cdot x$$

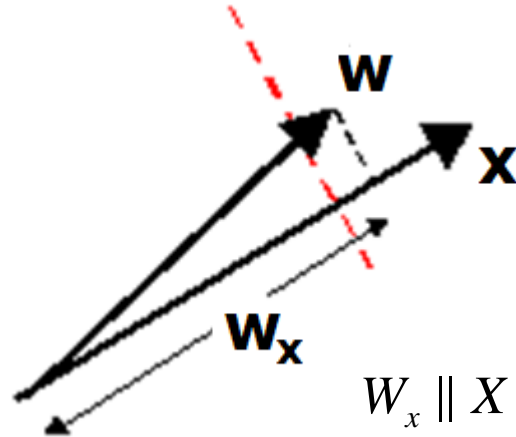
Uso del vector de proyección

14



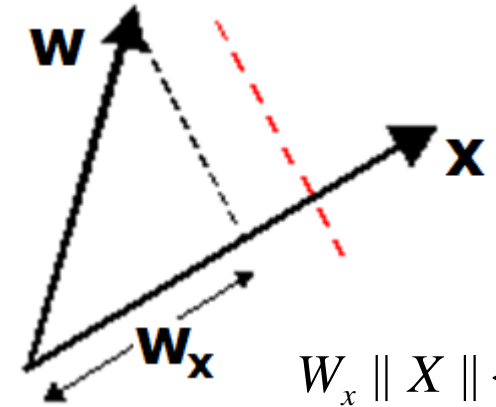
$$w_x \| x \| = \theta$$

$$w \cdot x = \theta$$



$$w_x \| x \| > \theta$$

$$w \cdot x > \theta$$



$$w_x \| x \| < \theta$$

$$w \cdot x < \theta$$

Entrenamiento del Perceptrón

- Inicializar los pesos de las conexiones con valores random (vector W)
- □ Mientras no se clasifiquen todos los ejemplos correctamente
 - ▣ Ingresar un ejemplo a la red.
 - ▣ Si fue clasificado incorrectamente
 - Si esperaba obtener $W \cdot X > \theta$ y no lo logró, "acerque" el vector W al vector X .
 - Si esperaba obtener $W \cdot X < \theta$ y no lo logró, "aleje" el vector W al vector X .

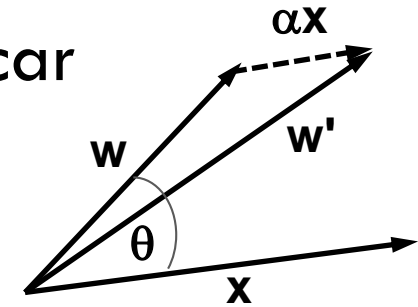
Aprendizaje supervisado

Ajuste del vector de pesos

16

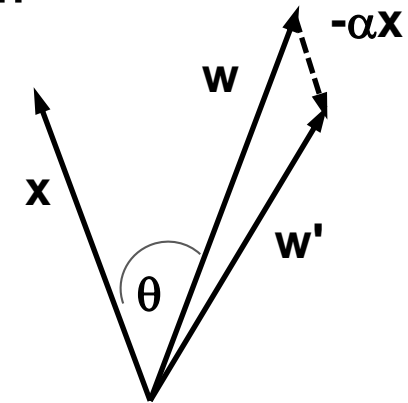
- Si $W.X < \theta$ no es el valor esperado entonces acercar W a X de la siguiente forma

$$w' = w + \alpha x$$



- Si $W.X > \theta$ no es el valor esperado entonces alejar W a X de la siguiente forma

$$w' = w - \alpha x$$



La velocidad de aprendizaje α es un valor real perteneciente a $(0,1]$

Ajuste del vector de pesos

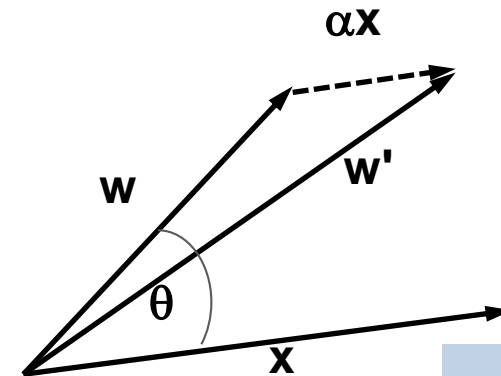
17

- La salida del perceptrón es $y = \begin{cases} 1 & \text{si } W.X \geq \theta \\ 0 & \text{si } W.X < \theta \end{cases}$
- La actualización de los pesos puede calcularse como

$$w_{nuevo} = w + \alpha (t - y) x$$

donde

- t es valor esperado
- y es valor obtenido



$$\begin{aligned} t &= 1 \\ y &= 0 \end{aligned}$$

Ajuste del vector de pesos

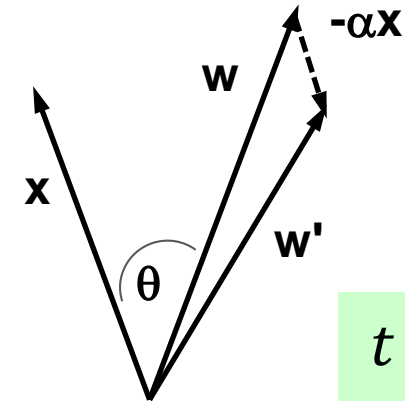
18

- La salida del perceptrón es $y = \begin{cases} 1 & \text{si } W.X \geq \theta \\ 0 & \text{si } W.X < \theta \end{cases}$
- La actualización de los pesos puede calcularse como

$$w_{nuevo} = w + \alpha (t - y) x$$

donde

- t es valor esperado
- y es valor obtenido



$$\begin{aligned} t &= 0 \\ y &= 1 \end{aligned}$$

Entrenamiento del perceptrón

- Seleccionar el valor de α y θ
- Inicializar los pesos de las conexiones con valores random (vector W)
- Mientras no se clasifiquen todos los ejemplos correctamente
 - ▣ Ingresar un ejemplo a la red.
 - ▣ Si fue clasificado incorrectamente
 - $W_{\text{nuevo}} = W + \alpha (t - y) x$

Ejemplo 1

20

- Entrenar un perceptrón para que se comporte como la función lógica AND.

Utilice

$$\alpha = 0.3$$

$$\theta = 1.5$$

W_1 y W_2 comienzan con valores aleatorios

AND

$$W1_{t+1} = W1_t + 0.3 (T-Y) X1$$

$$W2_{t+1} = W2_t + 0.3 (T-Y) X2$$

21

X1	X2	T	$W1_t$	$W2_t$	Y	$W1_{t+1}$	$W2_{t+1}$
0	0	0	0.00	0.25	0	0.00	0.25
1	0	0	0.00	0.25	0	0.00	0.25
0	1	0	0.00	0.25	0	0.00	0.25
1	1	1	0.00	0.25	0	0.30	0.55

Repetir hasta que sean iguales

AND

$$W1_{t+1} = W1_t + 0.3 (T - Y) X1$$

$$W2_{t+1} = W2_t + 0.3 (T - Y) X2$$

22

X1	X2	T	$W1_t$	$W2_t$	Y	$W1_{t+1}$	$W2_{t+1}$
<hr/>							
0	0	0	0.00	0.25	0	0.00	0.25
1	0	0	0.00	0.25	0	0.00	0.25
0	1	0	0.00	0.25	0	0.00	0.25
1	1	1	0.00	0.25	0	0.30	0.55
<hr/>							
0	0	0	0.30	0.55	0	0.30	0.55
1	0	0	0.30	0.55	0	0.30	0.55
0	1	0	0.30	0.55	0	0.30	0.55
1	1	1	0.30	0.55	0	0.60	0.85
<hr/>							
0	0	0	0.60	0.85	0	0.60	0.85
1	0	0	0.60	0.85	0	0.60	0.85
0	1	0	0.60	0.85	0	0.60	0.85
1	1	1	0.60	0.85	0	0.90	1.15
<hr/>							
0	0	0	0.90	1.15	0	0.90	1.15
1	0	0	0.90	1.15	0	0.90	1.15
0	1	0	0.90	1.15	0	0.90	1.15
1	1	1	0.90	1.15	1	0.90	1.15

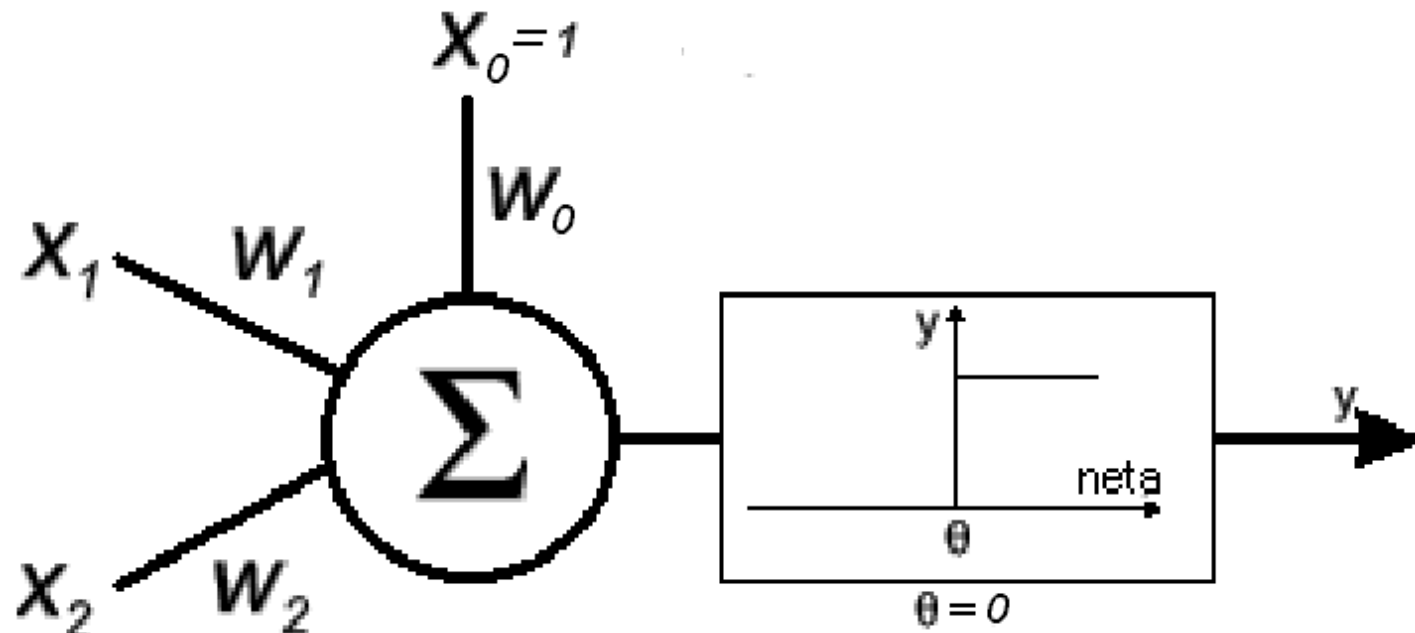
AND

El proceso se repite hasta comprobar que todos los ejemplos son clasificados correctamente

23

X1	X2	T	$W1_t$	$W2_t$	Y	$W1_{t+1}$	$W2_{t+1}$
0	0	0	0.00	0.25	0	0.00	0.25
1	0	0	0.00	0.25	0	0.00	0.25
0	1	0	0.00	0.25	0	0.00	0.25
1	1	1	0.00	0.25	0	0.30	0.55
0	0	0	0.30	0.55	0	0.30	0.55
1	0	0	0.30	0.55	0	0.30	0.55
0	1	0	0.30	0.55	0	0.30	0.55
1	1	1	0.30	0.55	0	0.60	0.85
0	0	0	0.60	0.85	0	0.60	0.85
1	0	0	0.60	0.85	0	0.60	0.85
0	1	0	0.60	0.85	0	0.60	0.85
1	1	1	0.60	0.85	0	0.90	1.15
0	0	0	0.90	1.15	0	0.90	1.15
1	0	0	0.90	1.15	0	0.90	1.15
0	1	0	0.90	1.15	0	0.90	1.15
1	1	1	0.90	1.15	1	0.90	1.15

Perceptrón



$$neta = \sum_i x_i w_i$$

$$y = \begin{cases} 1 & \text{si } neta \geq 0 \\ 0 & \text{si } neta < 0 \end{cases}$$

Ejemplo 2

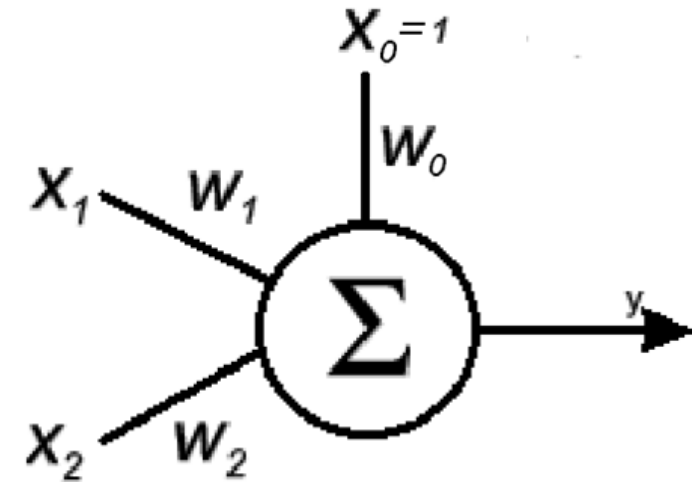
25

- Entrenar un perceptrón para que se comporte como la función lógica AND.

Utilice

$$\alpha = 0.25$$

W_0 , W_1 y W_2 comienzan con valores aleatorios



Ejemplo 2

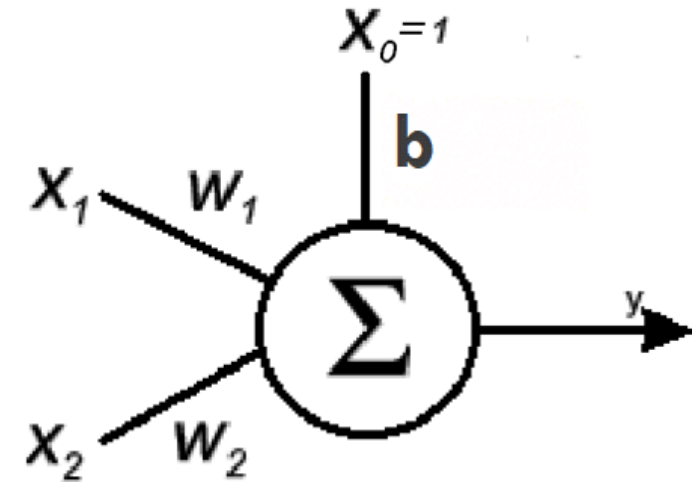
26

- Entrenar un perceptrón para que se comporte como la función lógica AND.

Utilice

$$\alpha = 0.25$$

b , W_1 y W_2 comienzan con valores aleatorios



AND – Iteración 1

$$\alpha = 0.25$$

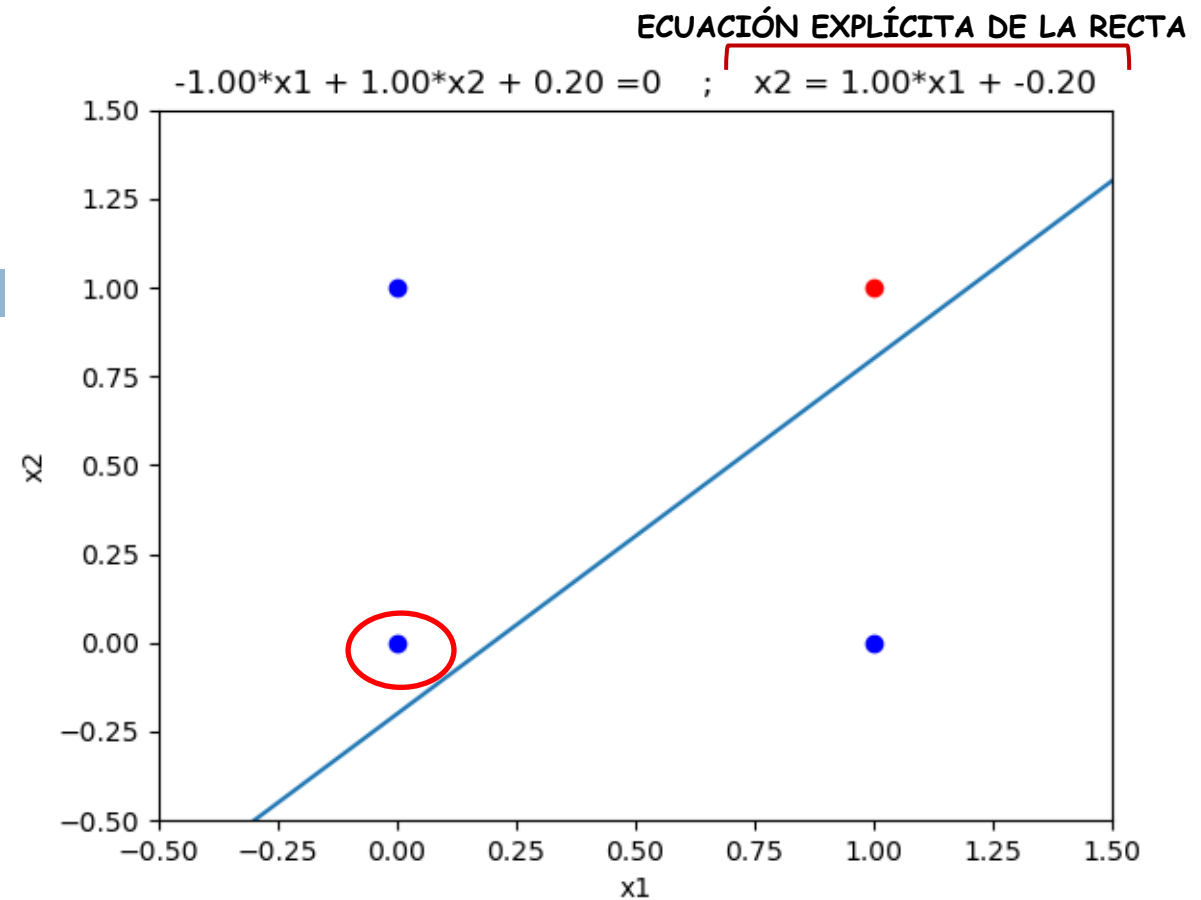
W1	W2	b
-1	1	0.2

X1	X2	T	Neta	Y
0	0	0	0.2	1

$$W1 = W1 + \alpha * (T - Y) * X1 = -1 + 0.25 * (0 - 1) * 0 = -1$$

$$W2 = W2 + \alpha * (T - Y) * X2 = 1 + 0.25 * (0 - 1) * 0 = 1$$

$$b = b + \alpha * (T - Y) * 1 = 0.2 + 0.25 * (0 - 1) = -0.05$$



AND – Iteración 1

$$\alpha = 0.25$$

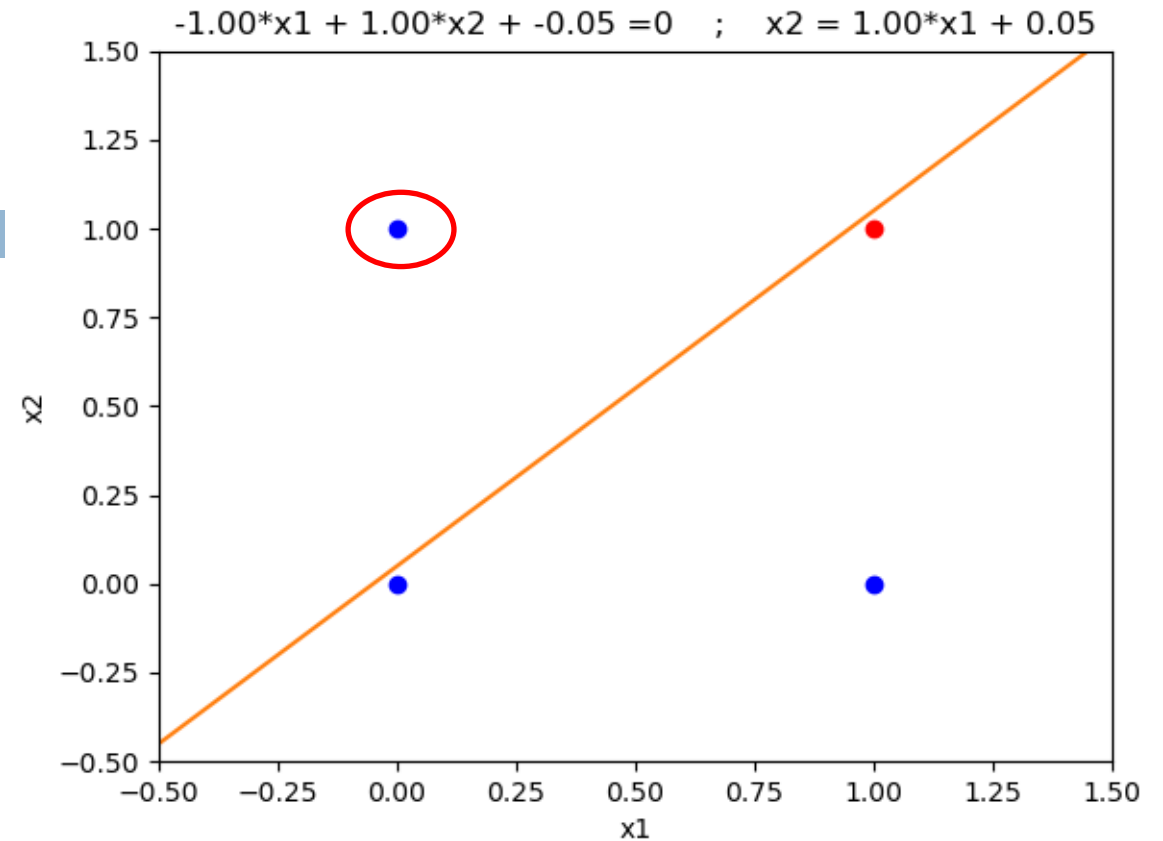
W1	W2	b
-1	1	-0.05

X1	X2	T	Neta	Y
0	1	0	0.95	1

$$W1 = W1 + \alpha * (T - Y) * X1 = -1 + 0.25 * (0 - 1) * 0 = -1$$

$$W2 = W2 + \alpha * (T - Y) * X2 = 1 + 0.25 * (0 - 1) * 1 = 0.75$$

$$b = b + \alpha * (T - Y) * 1 = -0.05 + 0.25 * (0 - 1) = -0.3$$

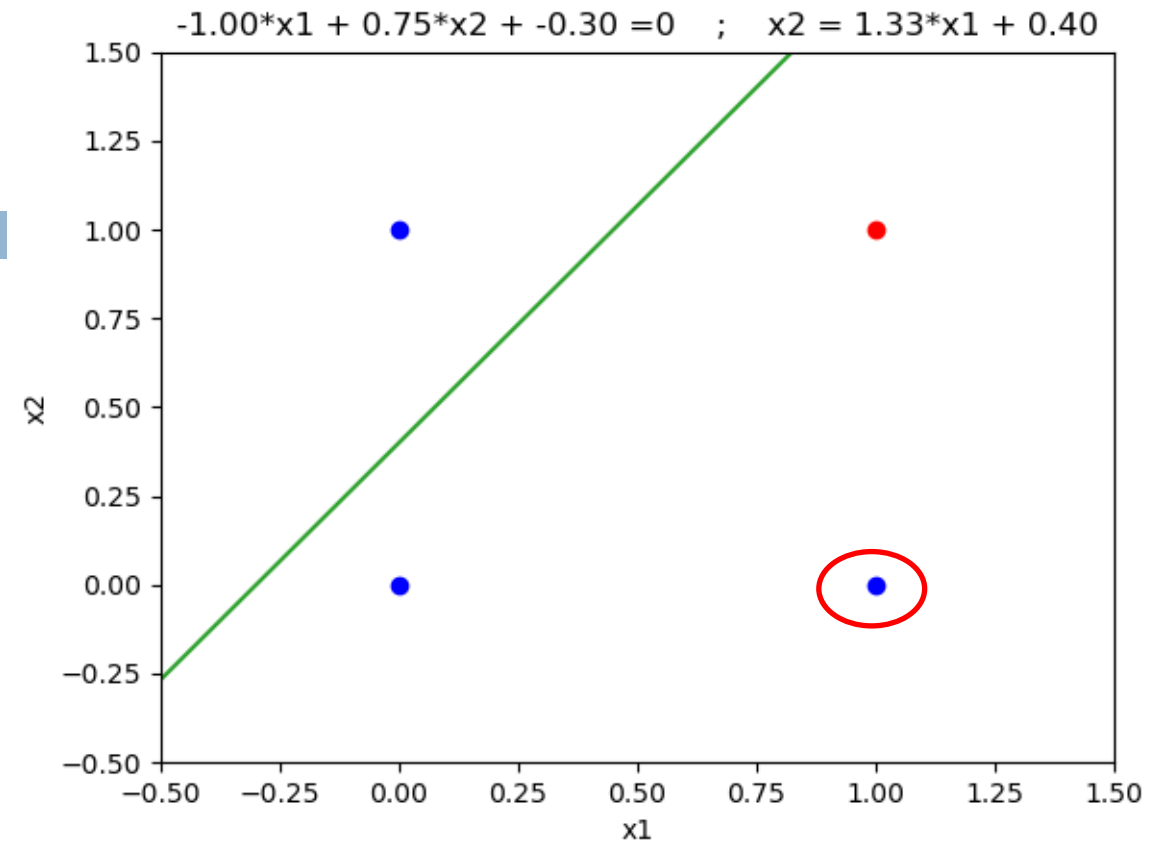


AND – Iteración 1

$$\alpha = 0.25$$

w1	w2	b
-1	0.75	-0.3

x1	x2	T	Neta	Y
1	0	0	-1.3	0



AND – Iteración 1

$$\alpha = 0.25$$

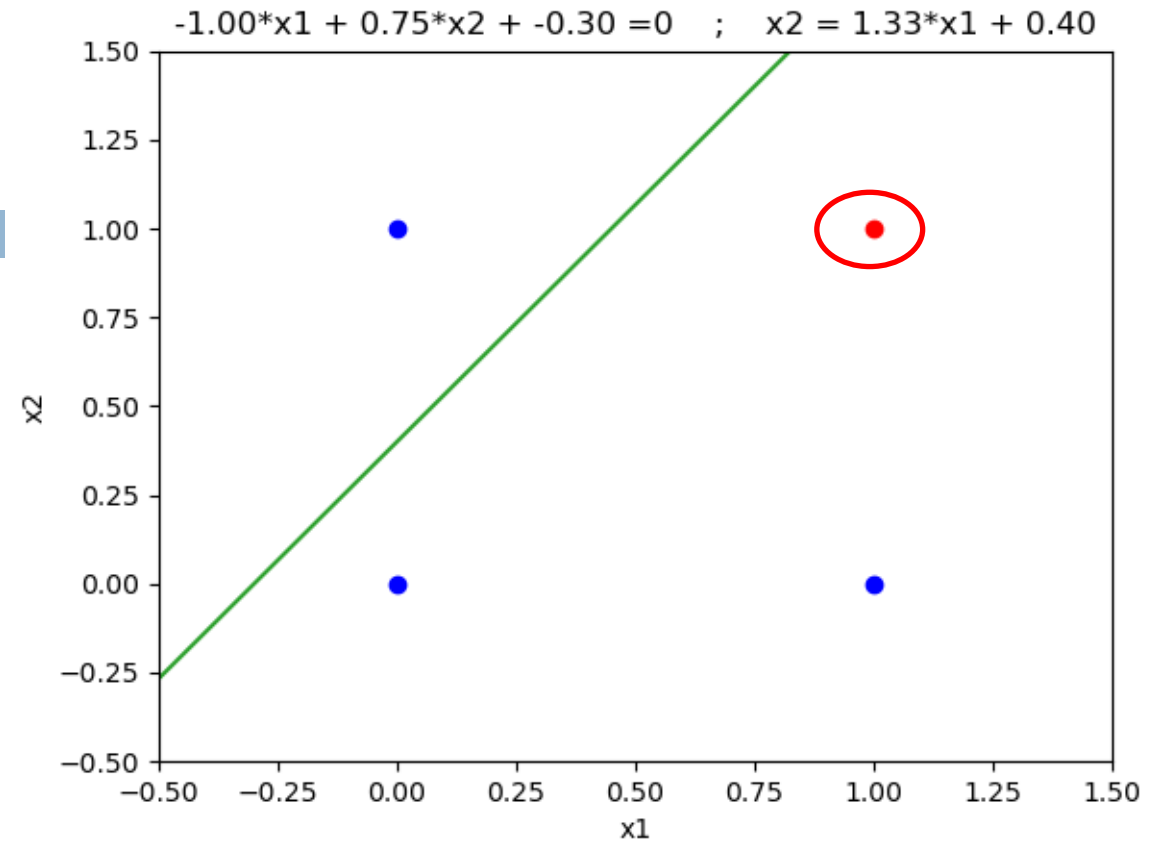
W1	W2	b
-1	0.75	-0.3

X1	X2	T	Neta	Y
1	1	1	-0.55	0

$$W1 = W1 + \alpha * (T - Y) * X1 = -1 + 0.25 * (1 - 0) * 1 = -0.75$$

$$W2 = W2 + \alpha * (T - Y) * X2 = 0.75 + 0.25 * (1 - 0) * 1 = 1$$

$$b = b + \alpha * (T - Y) * 1 = -0.3 + 0.25 * (1 - 0) = -0.05$$

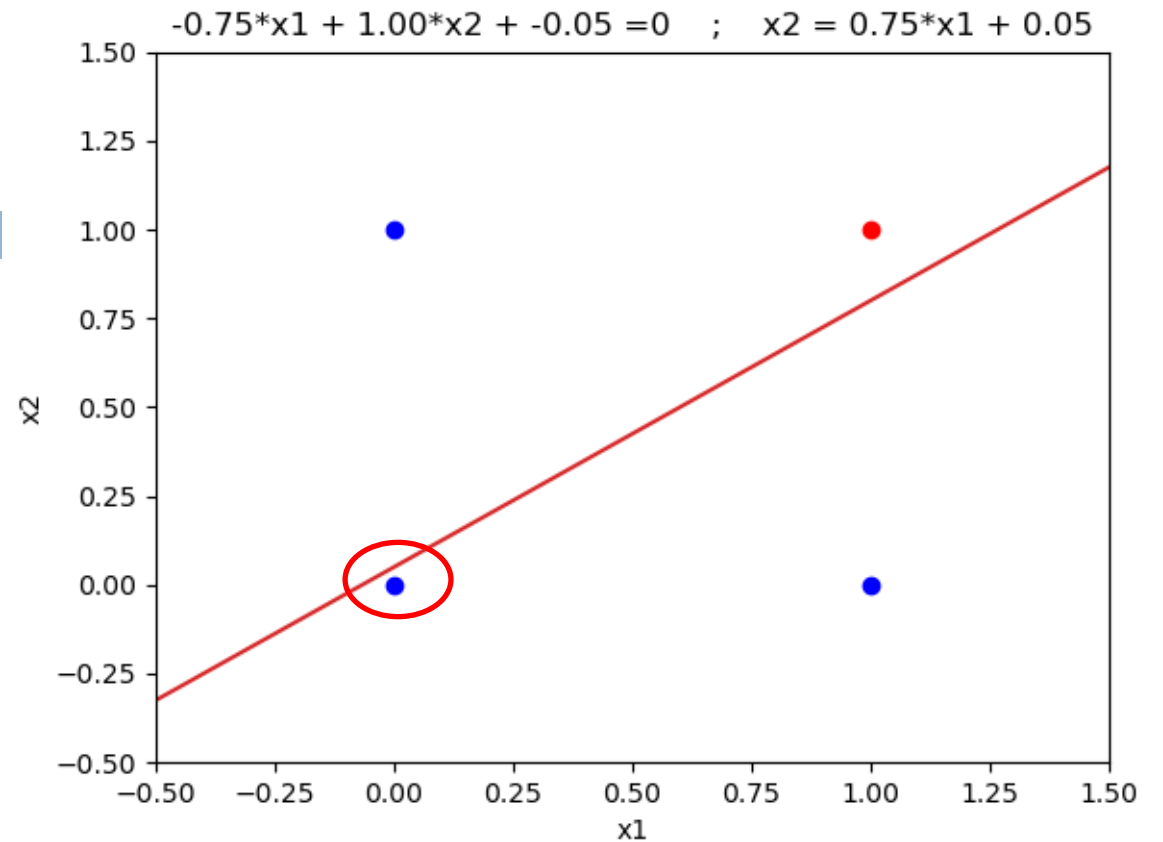


AND – Iteración 2

$$\alpha = 0.25$$

w1	w2	b
-0.75	1	-0.05

x1	x2	T	Neta	Y
0	0	0	-0.05	0



AND – Iteración 2

$$\alpha = 0.25$$

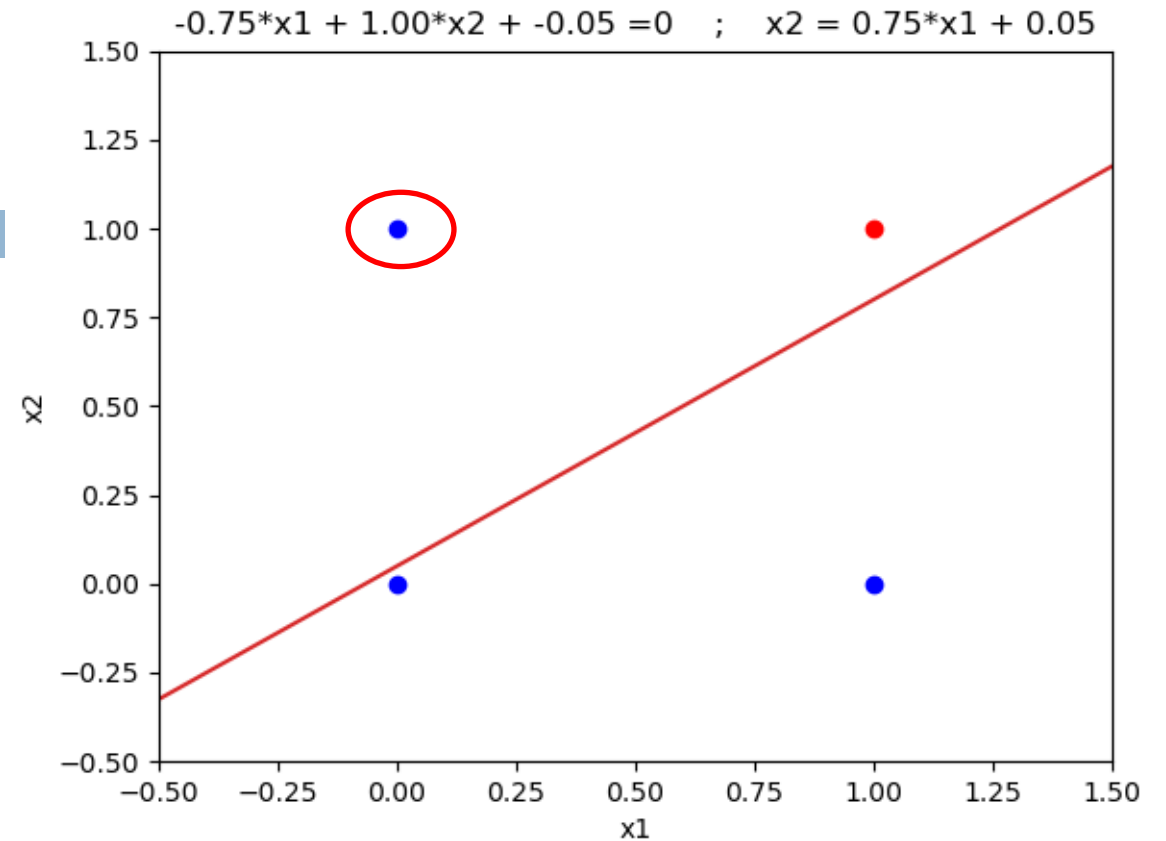
W1	W2	b
-0.75	1	-0.05

X1	X2	T	Neta	Y
0	1	0	0.95	1

$$W1 = W1 + \alpha * (T - Y) * X1 = -0.75 + 0.25 * (0 - 1) * 0 = -0.75$$

$$W2 = W2 + \alpha * (T - Y) * X2 = 1 + 0.25 * (0 - 1) * 1 = 0.75$$

$$b = b + \alpha * (T - Y) * 1 = -0.05 + 0.25 * (0 - 1) = -0.3$$

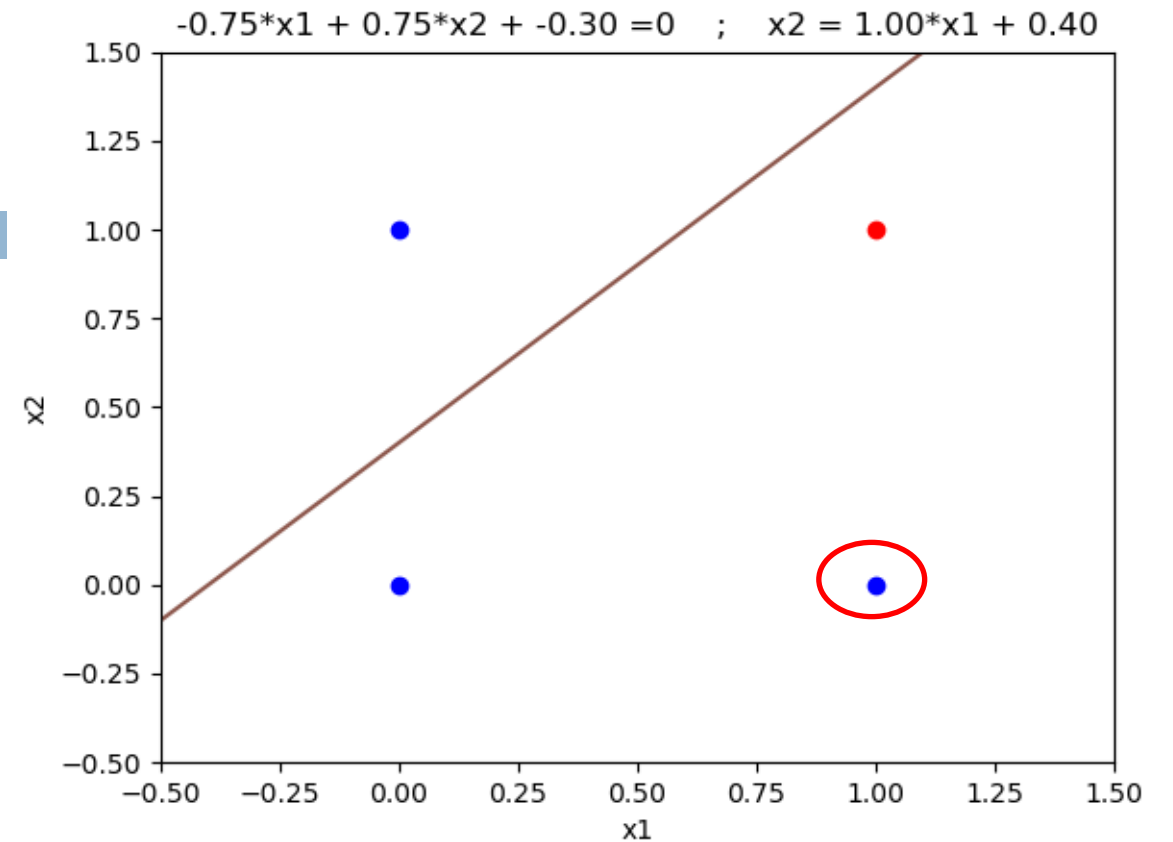


AND – Iteración 2

$$\alpha = 0.25$$

w1	w2	b
-0.75	0.75	-0.3

x1	x2	T	Neta	Y
1	0	0	-1.05	0



AND – Iteración 2

$$\alpha = 0.25$$

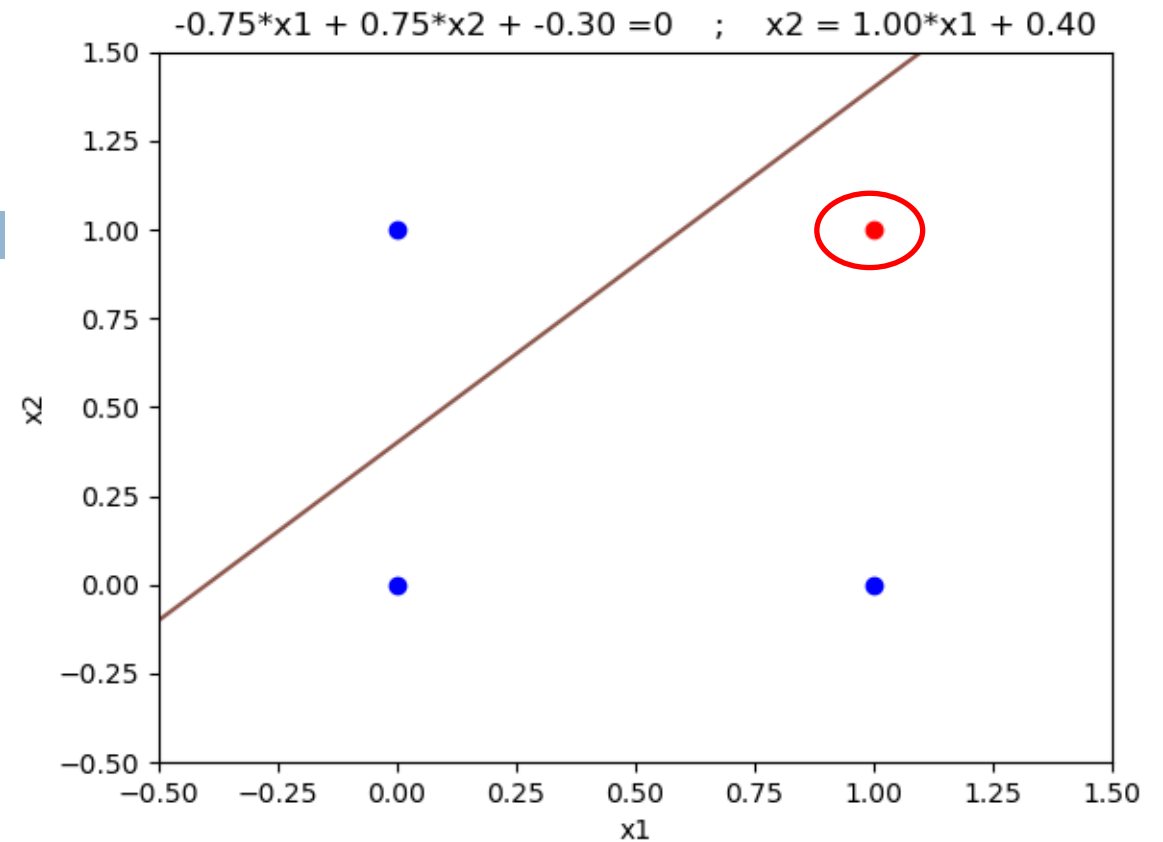
W1	W2	b
-0.75	0.75	-0.3

X1	X2	T	Neta	Y
1	1	1	-0.3	0

$$W1 = W1 + \alpha * (T-Y) * X1 = -0.75 + 0.25 * (1-0) * 1 = -0.5$$

$$W2 = W2 + \alpha * (T-Y) * X2 = 0.75 + 0.25 * (1-0) * 1 = 1$$

$$b = b + \alpha * (T-Y) * 1 = -0.3 + 0.25 * (1-0) = -0.05$$

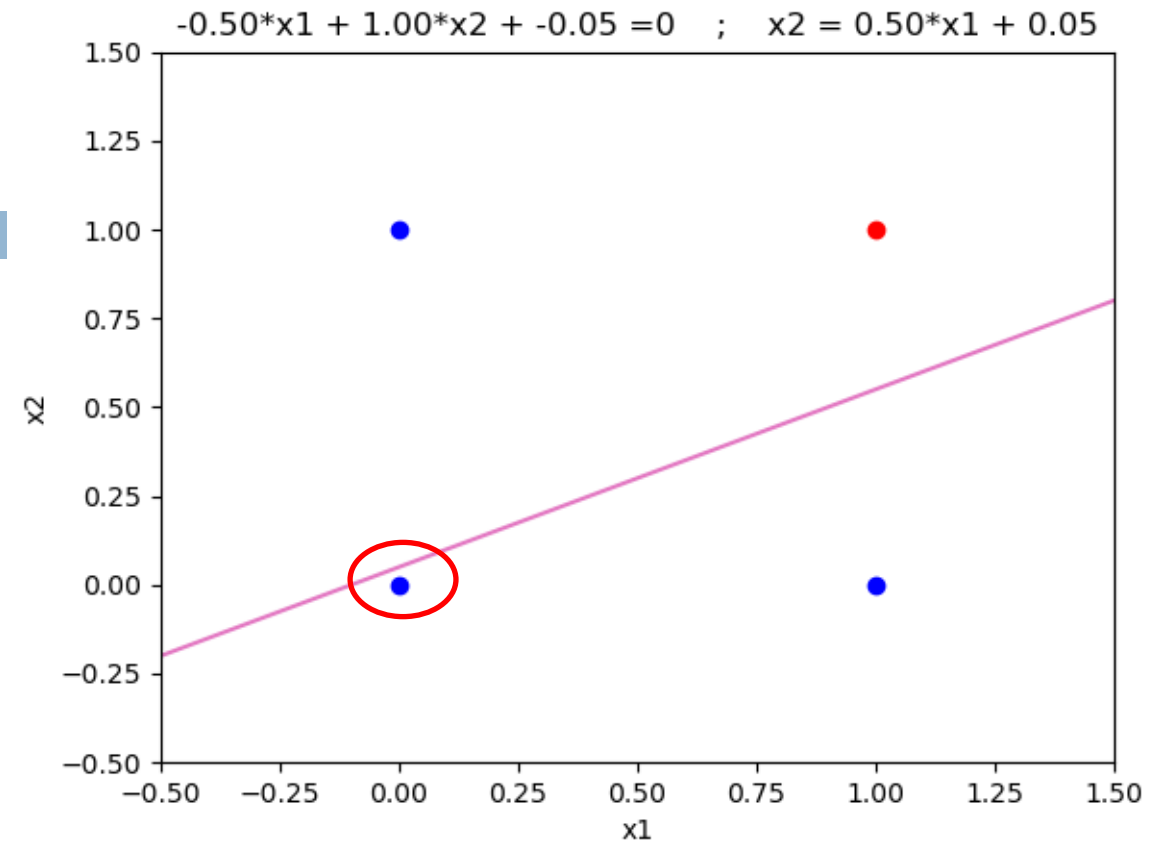


AND – Iteración 3

$$\alpha = 0.25$$

w1	w2	b
-0.5	1	-0.05

x1	x2	T	Neta	Y
0	0	0	-0.05	0



AND – Iteración 3

$$\alpha = 0.25$$

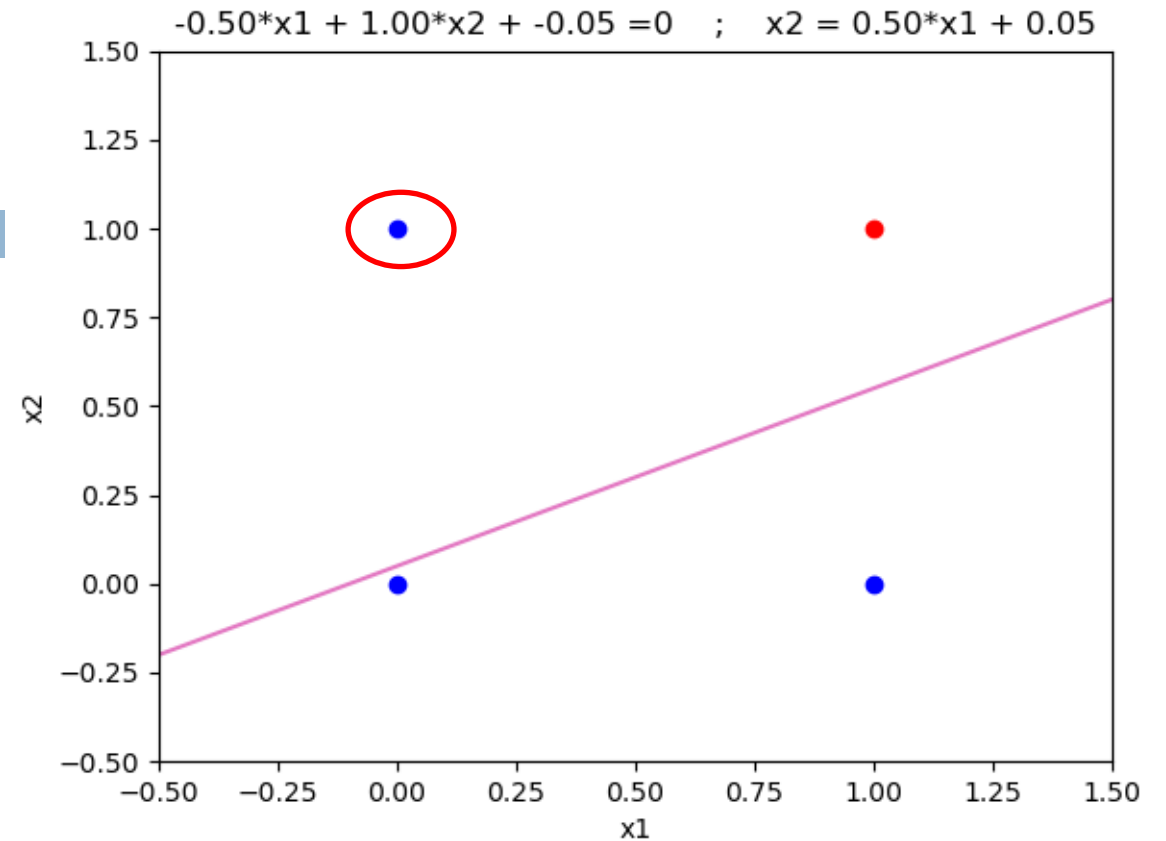
W1	W2	b
-0.5	1	-0.05

X1	X2	T	Neta	Y
0	1	0	0.95	1

$$W1 = W1 + \alpha * (T - Y) * X1 = -0.5 + 0.25 * (0 - 1) * 0 = -0.5$$

$$W2 = W2 + \alpha * (T - Y) * X2 = 1 + 0.25 * (0 - 1) * 1 = 0.75$$

$$b = b + \alpha * (T - Y) * 1 = -0.05 + 0.25 * (0 - 1) = -0.3$$

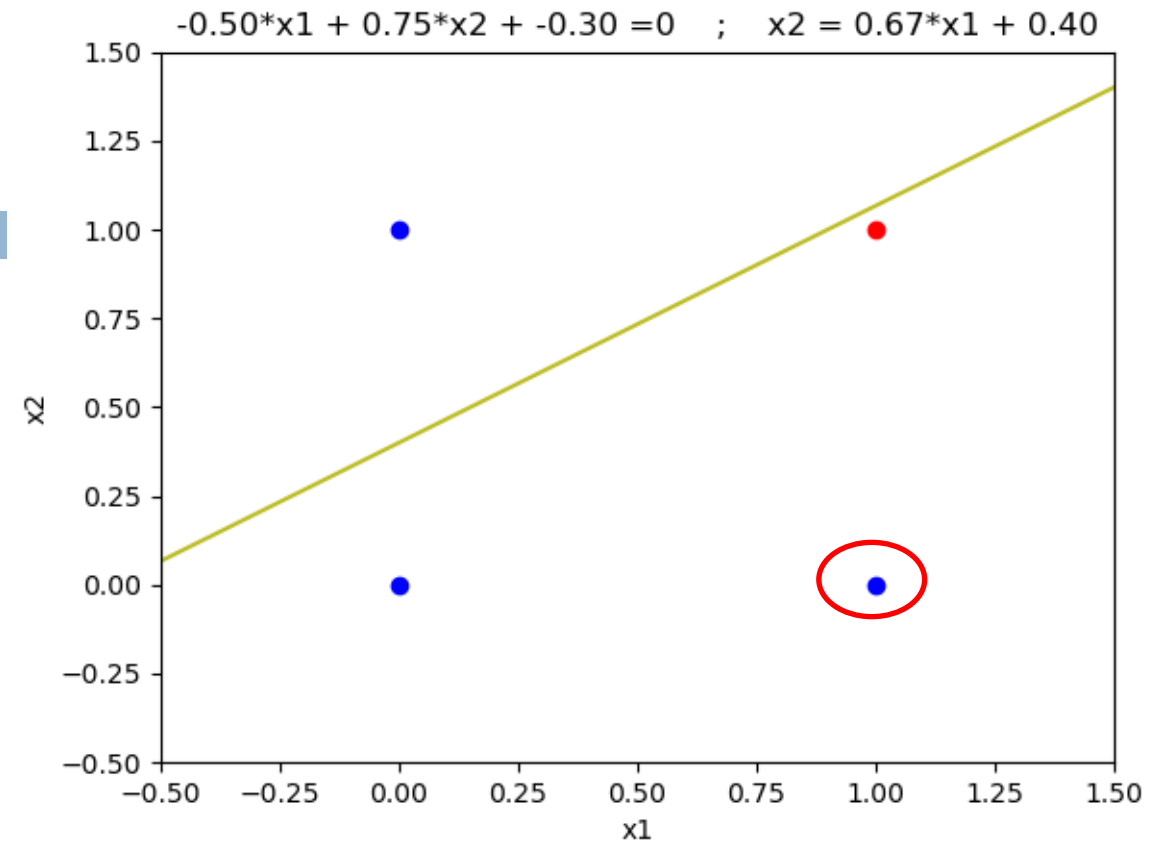


AND – Iteración 3

$$\alpha = 0.25$$

w1	w2	b
-0.5	0.75	-0.3

x1	x2	T	Neta	Y
1	0	0	-0.8	0



AND – Iteración 3

$$\alpha = 0.25$$

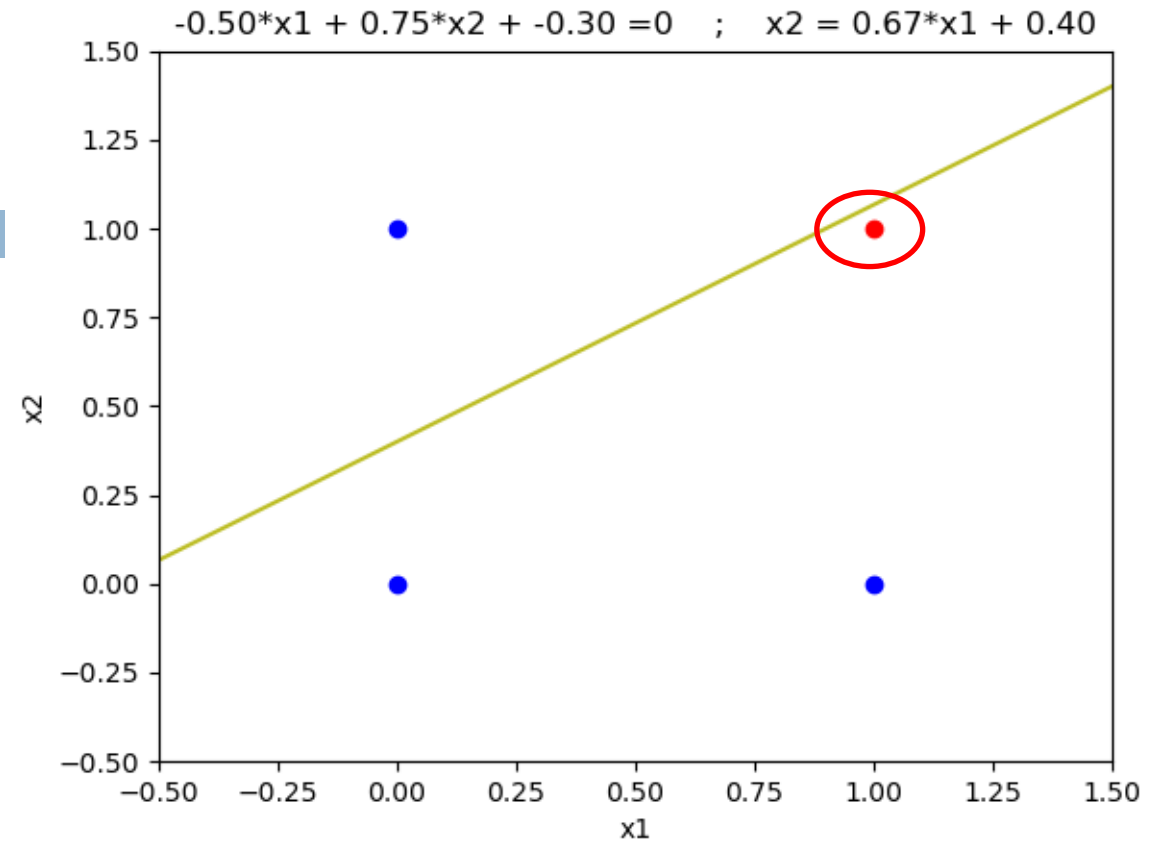
W1	W2	b
-0.5	0.75	-0.3

X1	X2	T	Neta	Y
1	1	1	-0.05	0

$$W1 = W1 + \alpha * (T-Y) * X1 = -0.5 + 0.25 * (1-0) * 1 = -0.25$$

$$W2 = W2 + \alpha * (T-Y) * X2 = 0.75 + 0.25 * (1-0) * 1 = 1$$

$$b = b + \alpha * (T-Y) * 1 = -0.3 + 0.25 * (1-0) = -0.05$$

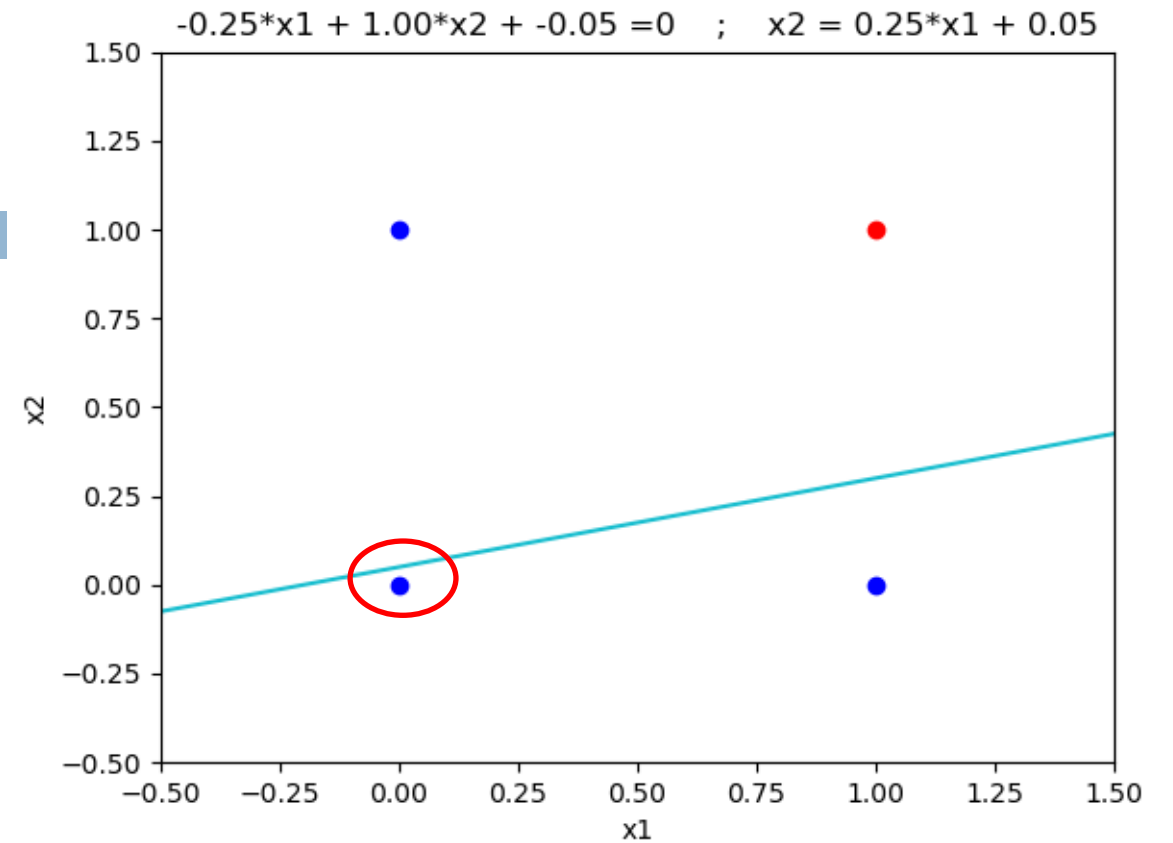


AND – Iteración 4

$$\alpha = 0.25$$

w1	w2	b
-0.25	1	-0.05

x1	x2	T	Neta	Y
0	0	0	-0.05	0



AND – Iteración 4

$$\alpha = 0.25$$

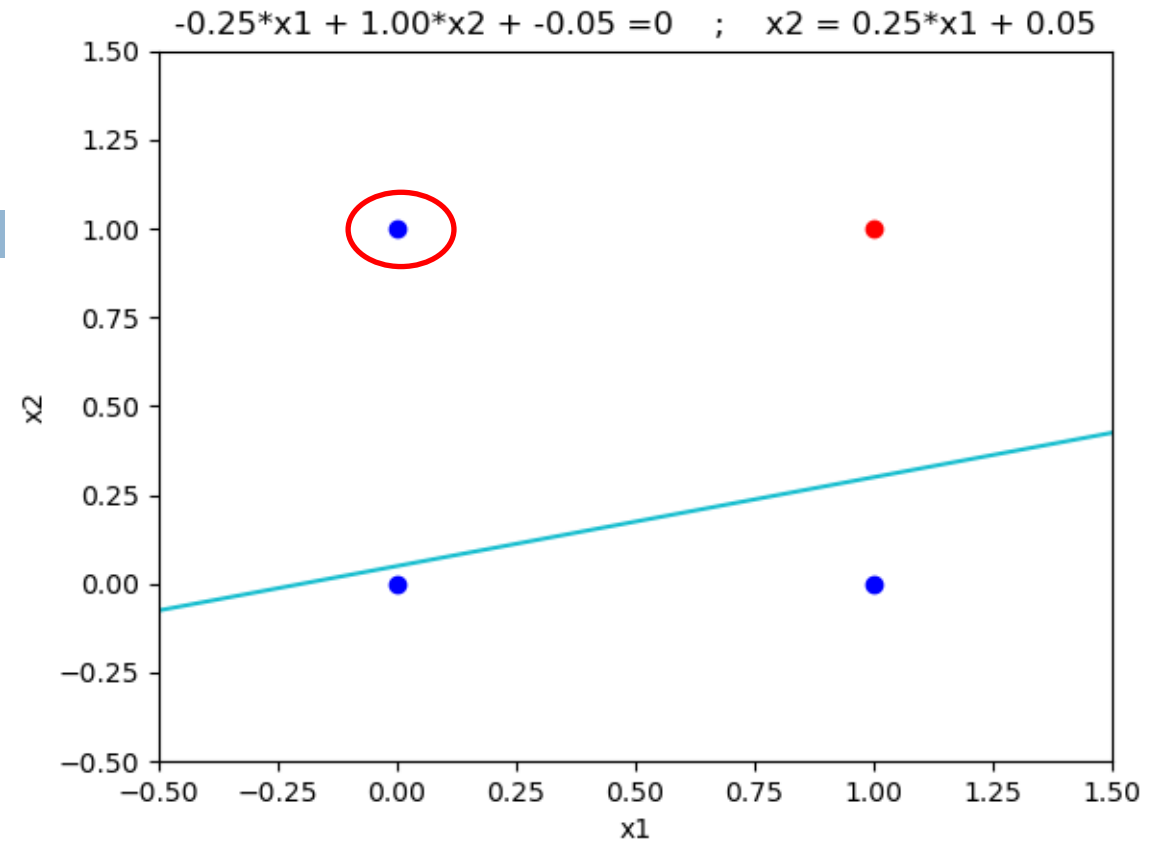
W1	W2	b
-0.25	1	-0.05

X1	X2	T	Neta	Y
0	1	0	0.95	1

$$W1 = W1 + \alpha * (T-Y) * X1 = -0.25 + 0.25 * (0-1) * 0 = -0.25$$

$$W2 = W2 + \alpha * (T-Y) * X2 = 1 + 0.25 * (0-1) * 1 = 0.75$$

$$b = b + \alpha * (T-Y) * 1 = -0.05 + 0.25 * (0-1) = -0.3$$

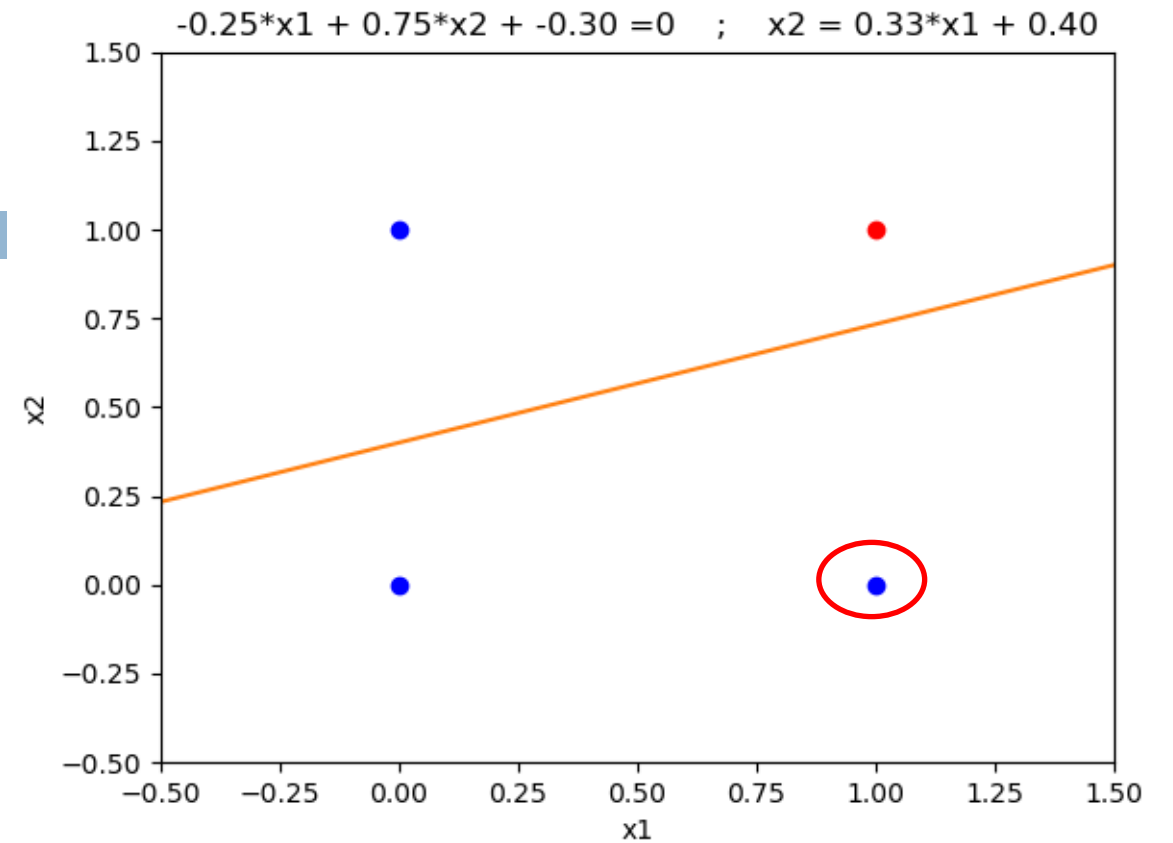


AND – Iteración 4

$$\alpha = 0.25$$

w1	w2	b
-0.25	0.75	-0.3

x1	x2	T	Neta	Y
1	0	0	-0.55	0

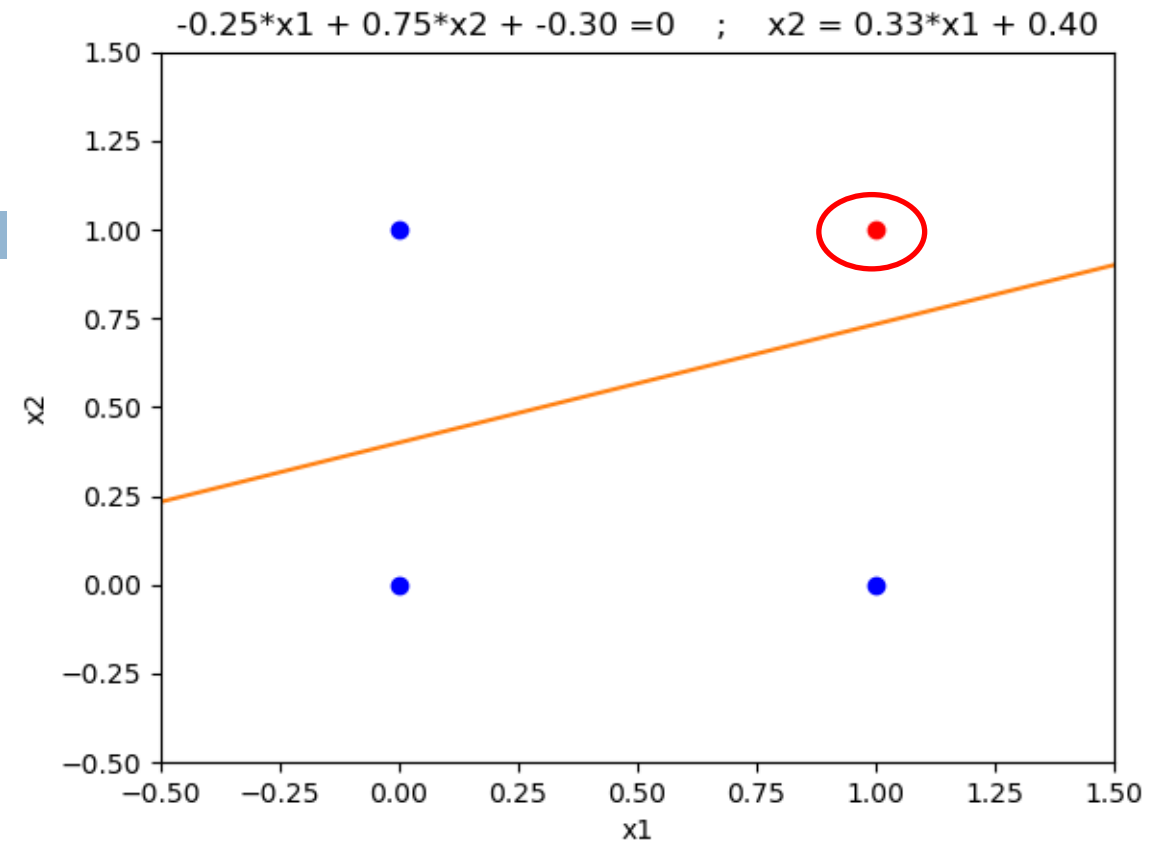


AND – Iteración 4

$$\alpha = 0.25$$

w1	w2	b
-0.25	0.75	-0.3

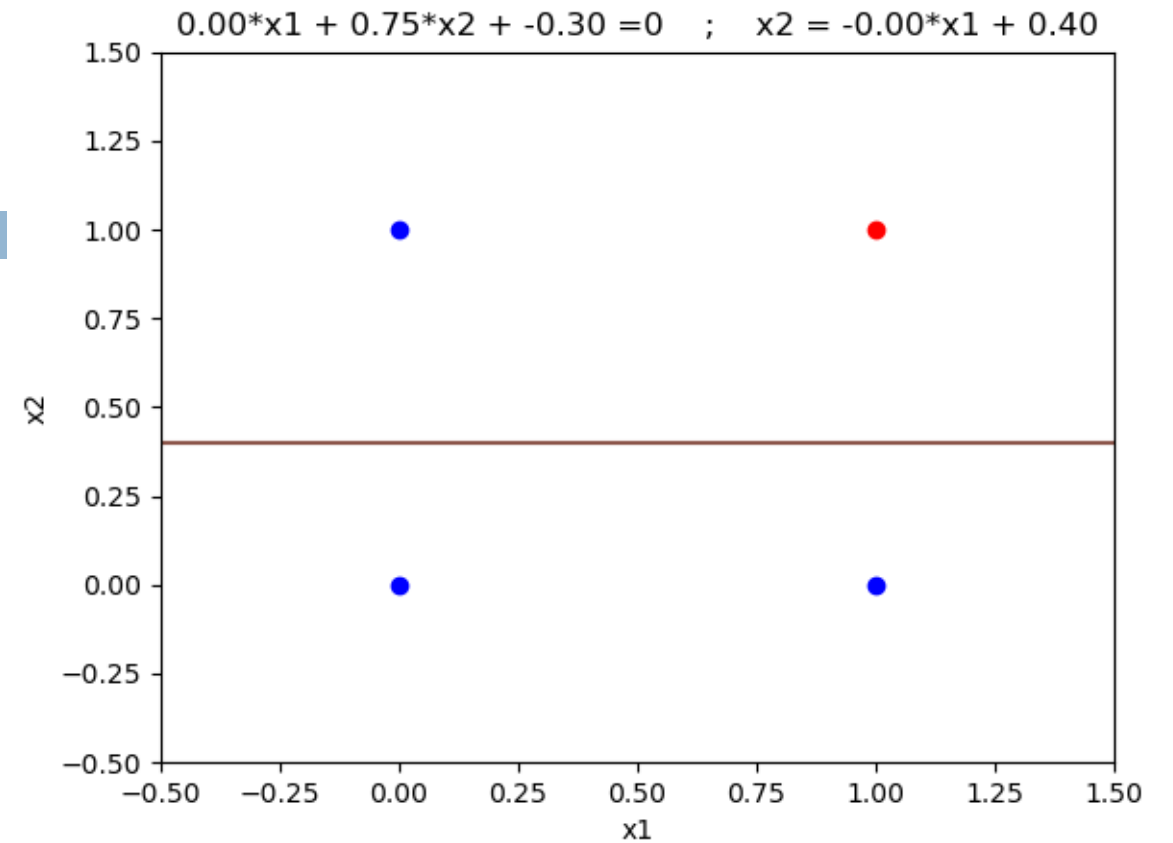
x1	x2	T	Neta	Y
1	1	1	0.2	1



AND – Iteración 5

$$\alpha = 0.25$$

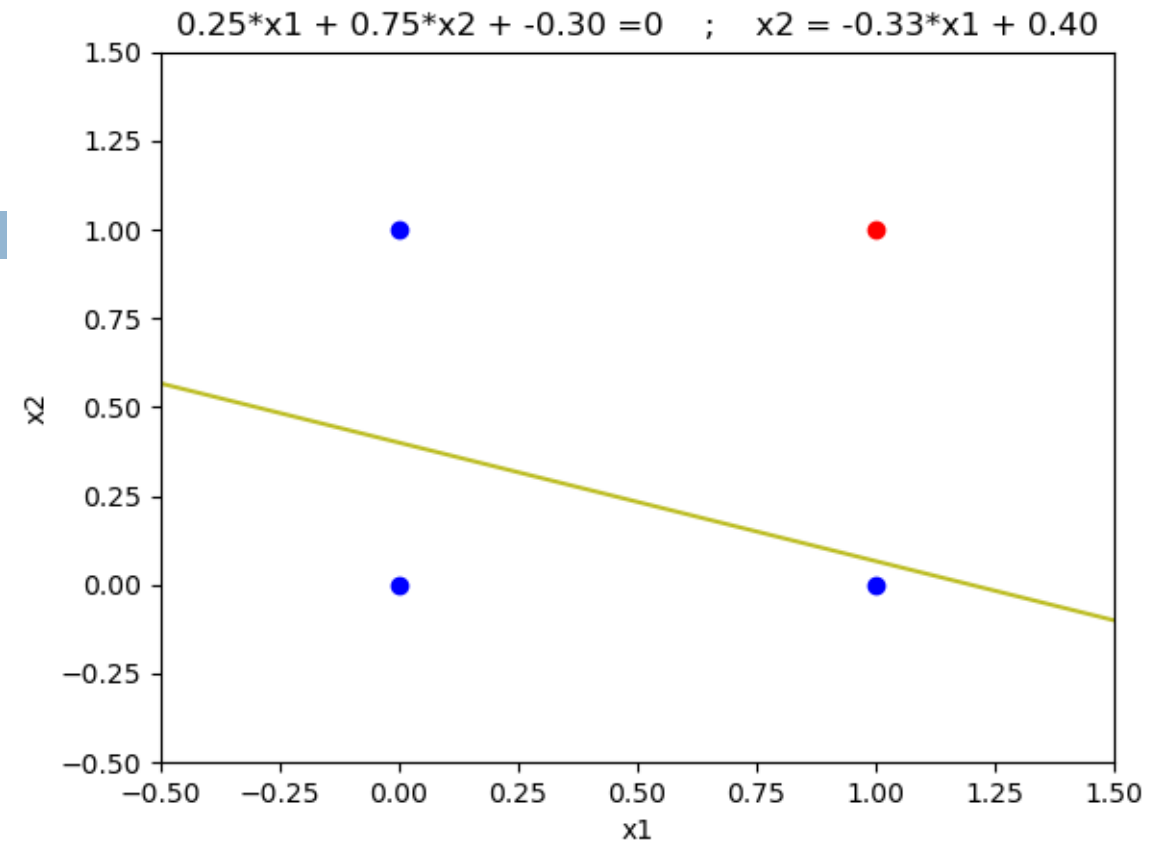
w1	w2	b
0	0.75	-0.3



AND – Iteración 6

$$\alpha = 0.25$$

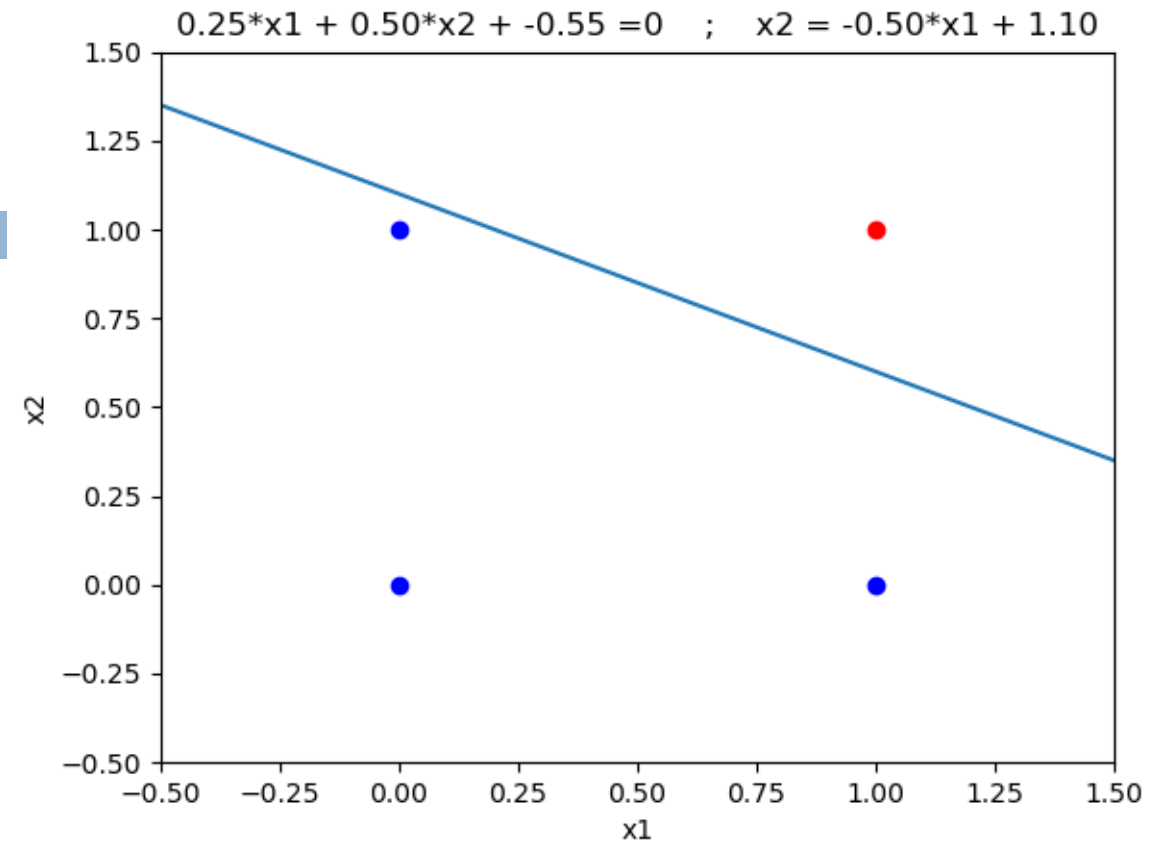
w1	w2	b
0.25	0.75	-0.3



AND – Iteración 7

$$\alpha = 0.25$$

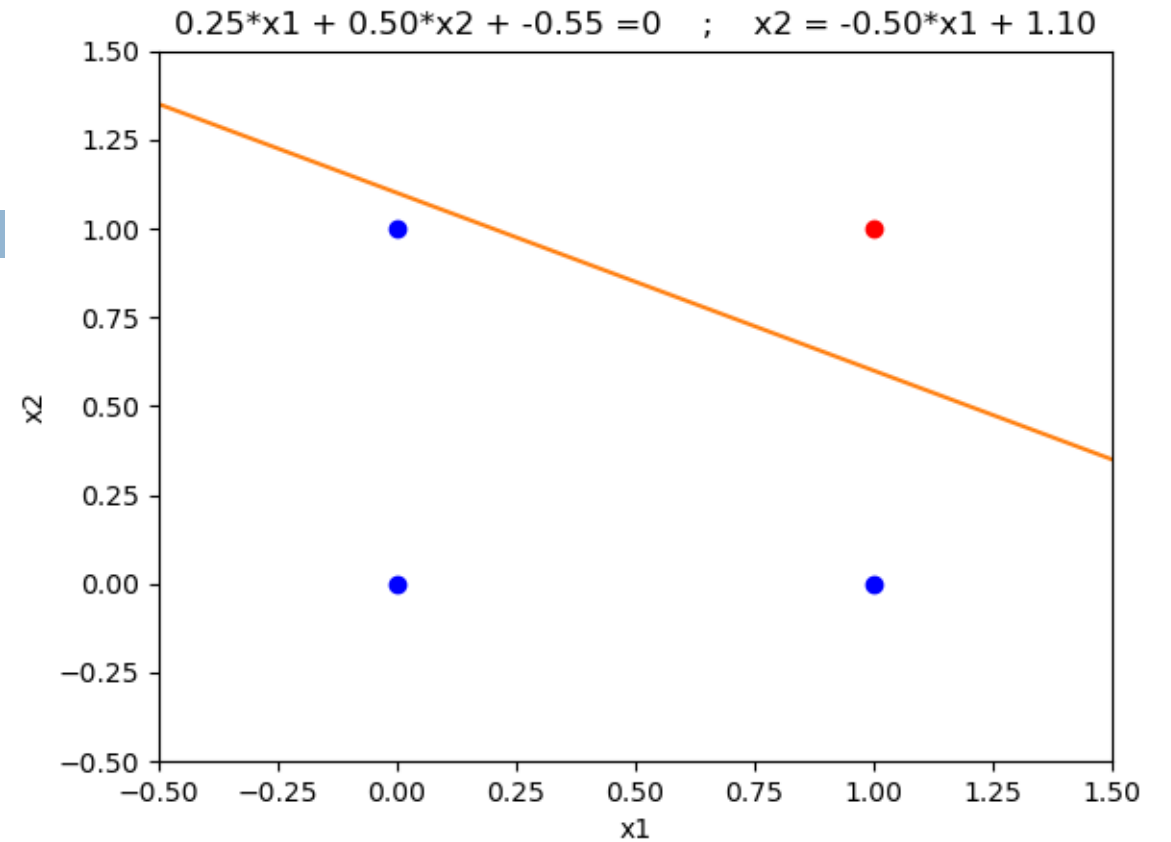
w1	w2	b
0.25	0.5	-0.55



AND – Iteración 8

$$\alpha = 0.25$$

w1	w2	b
0.25	0.5	-0.55



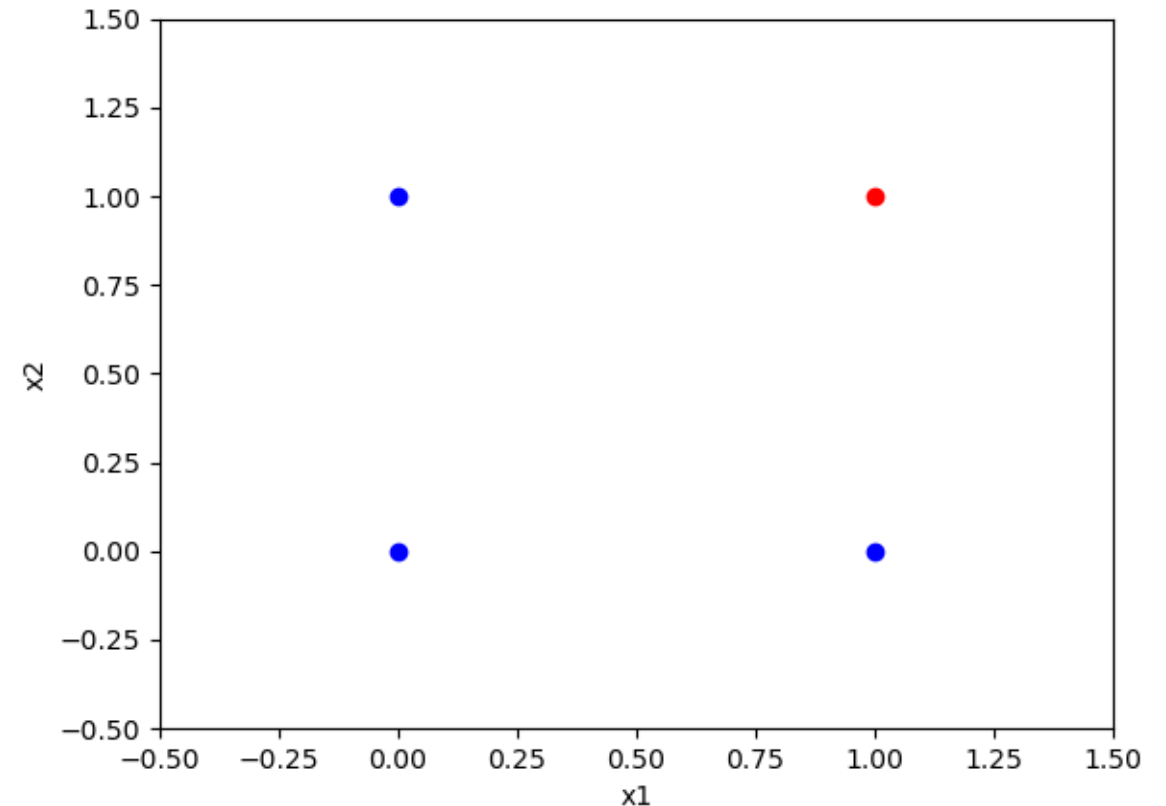
*Veamos cómo implementar el
algoritmo de entrenamiento
del **Perceptrón***

```
import numpy as np
import grafica as gr
```

```
# ---- FUNCION AND ----
```

```
X = np.array([[0, 1], [1, 0], [0, 0], [1, 1]])
```

```
T = np.array([0, 0, 0, 1])
```



```
import numpy as np
import grafica as gr
```

```
# ---- FUNCION AND ----
```

```
X = np.array([[0, 1], [1, 0], [0, 0], [1, 1]])
```

```
T = np.array([0, 0, 0, 1])
```

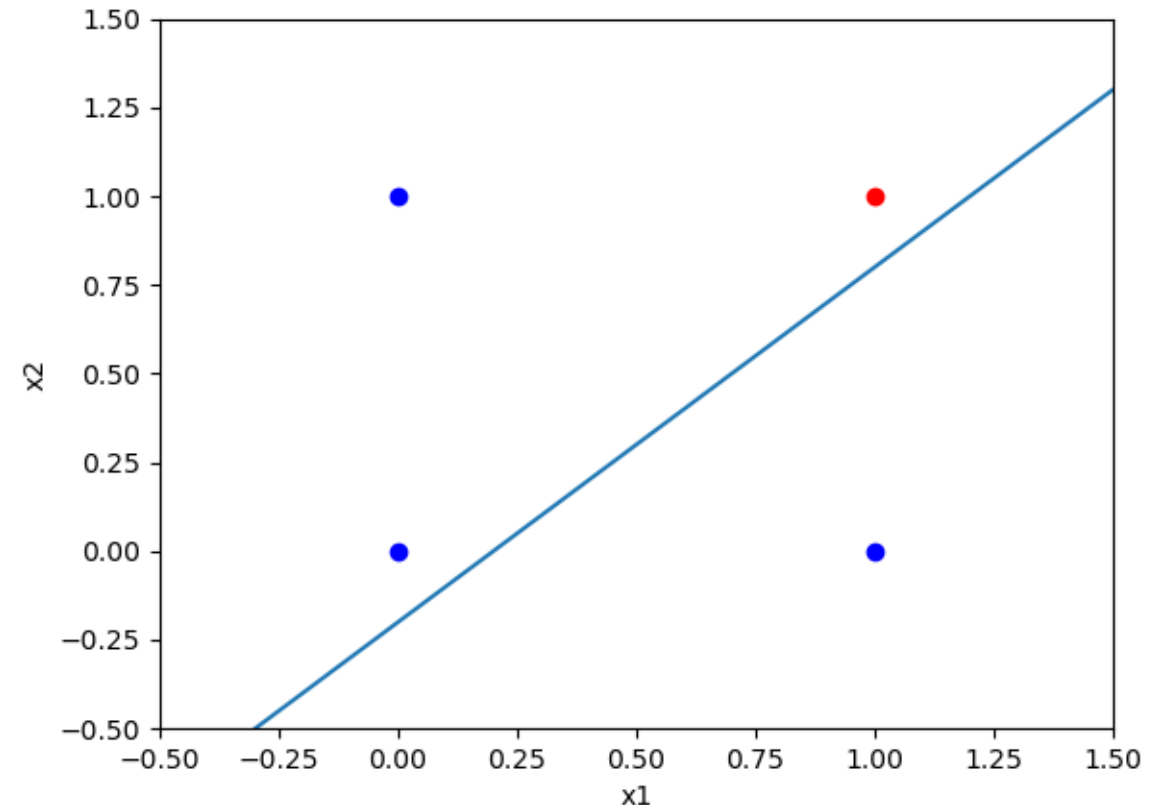
```
# --- Pesos iniciales ---
```

```
W = np.array([-1.0, 1.0])
```

```
b = 0.2
```

```
# Recta -->  $w_1*x_1 + w_2*x_2 + b = 0$ 
```

```
gr.dibuPtosRecta(X, T, W, b)
```



Entrenamiento del perceptrón

- Seleccionar el valor de α
- Inicializar los pesos de las conexiones con valores random (vector W y el bias b)
- Mientras no se clasifiquen todos los ejemplos correctamente
 - ▣ Ingresar uno a uno los ejemplos a la red.
 - ▣ Para cada ejemplo incorrectamente clasificado
 - $W_{nuevo} = W + \alpha (t - y) X$
 - $b_{nuevo} = b + \alpha (t - y)$

COMPLETAR
PERCEPTRON_AND_enClase.ipynb

ClassPerceptron.py

```
ppn = Perceptron(alpha=0.1, n_iter=30, draw=1, title=['X1', 'X2'],  
                 random_state=1)
```

□ Parámetros de entrada

- **alpha**: valor en el intervalo $(0, 1]$ que representa la velocidad de aprendizaje.
- **n_iter**: máxima cantidad de iteraciones a realizar.
- **draw**: valor distinto de 0 si se desea ver el gráfico y 0 si no. Sólo si es 2D.
- **title**: lista con los nombres de los ejes para el gráfico. Se usa sólo si **draw** no es cero.
- **random_state**: None si los pesos se inicializan en forma aleatoria, un valor entero para fijar la semilla

ClassPerceptron.py

```
ppn = Perceptron(alpha=0.1, n_iter=30)
```

```
ppn.fit(X, T)
```

- Parámetros de entrada

- **X** : arreglo de $N \times M$ donde N es la cantidad de ejemplos y M la cantidad de atributos.
- **T** : arreglo de N elementos siendo N la cantidad de ejemplos

- Retorna

- **w_** : arreglo de M elementos siendo M la cantidad de atributos de entrada
- **b_** : valor numérico continuo correspondiente al bias.
- **errors_** : errores cometidos en cada iteración.

ClassPerceptron.py

$Y = \text{ppn.predict}(X)$

□ Parámetros de entrada

▣ **X** : arreglo de $N \times M$ donde N es la cantidad de ejemplos y M la cantidad de atributos.

□ Retorna: un arreglo con el resultado de aplicar el perceptrón entrenado previamente con `fit()` a la matriz de ejemplos X .

▣ **Y** : arreglo de N elementos siendo N la cantidad de ejemplos

```
import numpy as np
from grafica import *
from ClassPerceptron import Perceptron
```

```
X = np.array([[0, 1], [1, 0], [0, 0], [1, 1]])
T = np.array([0, 0, 0, 1])
```

```
#--- ENTRENAMIENTO ---
```

```
ppn = Perceptron(alpha=0.1, n_iter=30, draw=1, title=['X1', 'X2'])
ppn.fit(X, T)
```

```
#--- Uso del perceptrón ---
```

```
Y = ppn.predict(X)

aciertos = sum(Y == T)
print("aciertos = ", aciertos)

nAciertos = sum(Y==T)
print("% de aciertos = %.2f %" % (100*nAciertos/X.shape[0]))
```

Ejemplo 1

56

- Sobre una cinta transportadora circulan naranjas y melones. Se busca obtener un clasificador de frutas que facilite su almacenamiento. Para cada fruta se conoce su diámetro, en centímetros y su intensidad de color naranja, medida entre 0 y 255.
- Utilice la información del archivo **FrutasTrain.csv** para entrenar un perceptrón que permita resolver el problema.
- Analice la performance de la red obtenida utilizando las muestras del archivo **FrutasTest.csv**

Ejemplo 1

Perceptron_Frutas_RN.ipynb

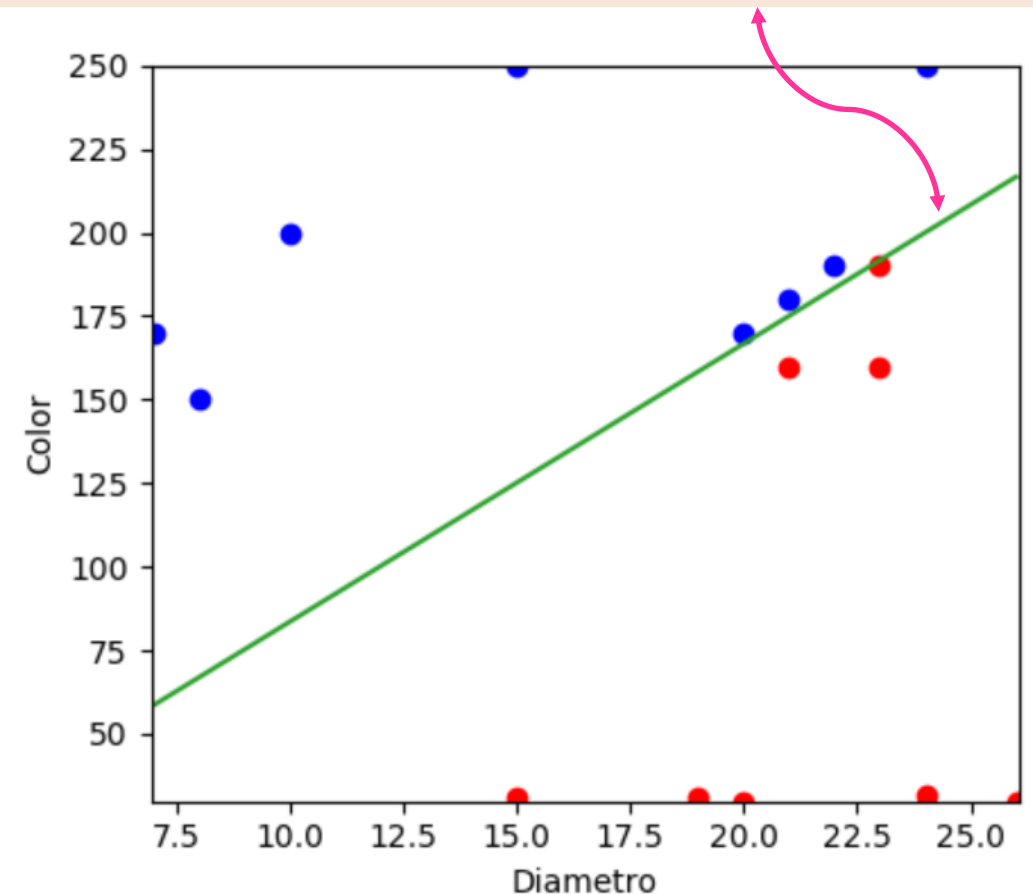
```
titulos  
['Diametro', 'Color']
```

```
W  
array([ 3.51034542, -0.41478193])
```

```
b  
0.2536257868554866
```

- El resultado es una función discriminante lineal (en este caso una recta) que separa los datos de entrada en dos clases

$$3.51034542 * \text{diametro} - 0.41478193 * \text{color} + 0.253625787 = 0$$



$$3.51034542 * \text{diametro} - 0.41478193 * \text{color} + 0.253625787$$



Diametro	Color	Clase	Neta	Predice	Corresponde a
10	200	Naranja	-47.5993	0	Naranja
20	30	Melon	58.0171	1	Melon
8	150	Naranja	-33.8809	0	Naranja
26	30	Melon	79.0791	1	Melon
7	170	Naranja	-45.6869	0	Naranja
24	32	Melon	71.2289	1	Melon
20	170	Naranja	-0.0524	0	Naranja
21	160	Melon	7.6058	1	Melon
21	180	Naranja	-0.6899	0	Naranja
23	160	Melon	14.6265	1	Melon
22	190	Naranja	-1.3273	0	Naranja
23	190	Melon	2.1830	1	Melon
24	250	Naranja	-19.1936	0	Naranja
15	31	Melon	40.0506	1	Melon
15	250	Naranja	-50.7867	0	Naranja
19	31	Melon	54.0919	1	Melon

Uso del perceptrón

□ Ejemplos del archivo **FrutasTest.csv**

$$3.51034542 * \text{diametro} - 0.41478193 * \text{color} + 0.253625787$$



Diametro	Color	Clase	Neta	Predice	Corresponde a
7	100	Naranja	-16.6521	0	Naranja
20	20	Melon	62.1649	1	Melon
25	70	Melon	58.9775	1	Melon
10	210	Naranja	-51.7471	0	Naranja

```
import pandas as pd
import numpy as np
from sklearn import preprocessing

datos = pd.read_csv("../Datos/FrutasTrain.csv")
xTrain = np.array(datos.iloc[:,0:2])

datosTest = pd.read_csv("../Datos/FrutasTest.csv")
xTest = np.array(datosTest.iloc[:,0:2])

#--- Escala los valores entre 0 y 1 ---
normalizador = preprocessing.MinMaxScaler()

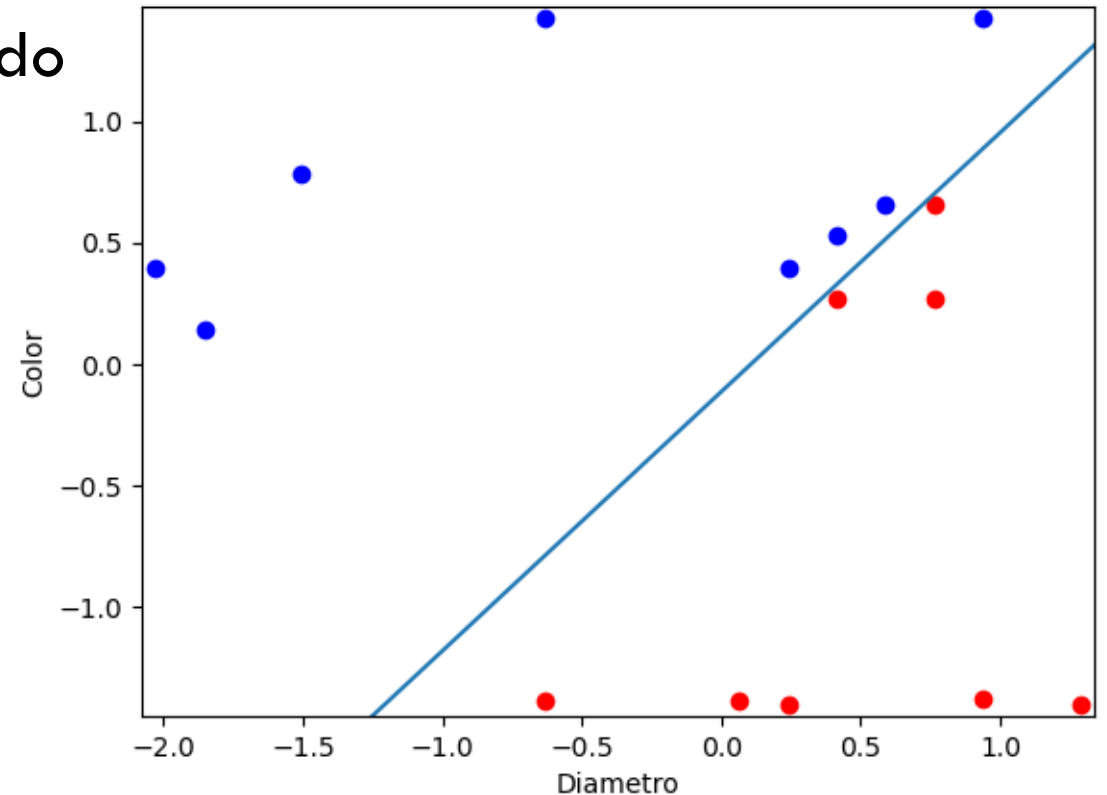
xTrain = normalizador.fit_transform(xTrain)

#--- normalizando los datos de testeo ---
xTest = normalizador.transform(xTest)
```



Ejemplo 1

- Entrene el perceptrón
 - ▣ Normalizando los ejemplos linealmente
 - ▣ Normalizando los ejemplos utilizando los valores de media y desvío
- Pruebe ingresando
 - ▣ Las frutas en orden aleatorio
 - ▣ Las naranjas primero
 - ▣ Los melones primero



Evaluación del modelo

- Matriz de confusión
- Métricas
 - ▣ Accuracy
 - ▣ Precisión
 - ▣ Recall
 - ▣ F1 -score
 - ▣ AUC, Curva ROC

Matriz de Confusión

	Predice Clase 1	Predice Clase 2	Recall
True Clase 1	A	B	$A/(A+B)$
True Clase 2	C	D	$D/(C+D)$
Precision	$A/(A+C)$	$D/(B+D)$	$(A+D)/(A+B+C+D)$ accuracy

- Los **aciertos** del modelo están sobre la **diagonal** de la matriz.
- **Precision**: la proporción de **predicciones correctas** sobre **una clase**.
- **Recall**: la proporción de **ejemplos** de **una clase** que son **correctamente clasificados**.
- **Accuracy**: la performance general del modelo, sobre **todas las clases**. Es la cantidad de **aciertos** sobre el **total** de ejemplos.

Sklearn.metrics.accuracy_score

```
from sklearn import metrics

Y_train = [0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]
Y_pred  = [0, 2, 1, 3, 0, 1, 2, 0, 0, 1, 2, 3]

aciertos = metrics.accuracy_score(Y_train, Y_pred)
print("%% accuracy = %.3f" % aciertos)
```

Sklearn.metrics.accuracy_score

```
from sklearn import metrics

Y_train = [0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]
Y_pred  = [0, 2, 1, 3, 0, 1, 2, 0, 0, 1, 2, 3]

aciertos = metrics.accuracy_score(Y_train, Y_pred)
print("%% accuracy = %.3f" % aciertos)
```

Rtas esperadas

Rtas obtenidas

- De los 12 valores sólo 9 fueron identificados correctamente.
- La tasa de aciertos es $9/12 = 0.75$

Sklearn.metrics.confusion_matrix

```
Y_train = [0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]
Y_pred  = [0, 2, 1, 3, 0, 1, 2, 0, 0, 1, 2, 3]
```

```
MM = metrics.confusion_matrix(Y_train, Y_pred)
print("Matriz de confusión:\n%s" % MM)
```

```
Matriz de confusión:
 0 [[3 0 0 0]
 1 [0 2 1 0]
 2 [0 1 2 0]
 3 [1 0 0 2]]
    0  1  2  3
  PREDICE
```

Esperaba obtener 1
como respuesta pero
la red respondió 2

Sklearn.metrics.confusion_matrix

```
Y_train = [0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]
Y_pred  = [0, 2, 1, 3, 0, 1, 2, 0, 0, 1, 2, 3]
```

```
MM = metrics.confusion_matrix(Y_train, Y_pred)
print("Matriz de confusión:\n%s" % MM)
```

```
Matriz de confusión:
 0 [[3 0 0 0]
 1 [0 2 1 0]
 2 [0 1 2 0]
 3 [1 0 0 2]]
    0  1  2  3
  PREDICE
```

La respuesta correcta
es 2 pero la red
respondió 1

Sklearn.metrics.confusion_matrix

```
Y_train = [0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]
Y_pred  = [0, 2, 1, 3, 0, 1, 2, 0, 0, 1, 2, 3]
```

```
MM = metrics.confusion_matrix(Y_train, Y_pred)
print("Matriz de confusión:\n%s" % MM)
```

```
Matriz de confusión:
 0 [[3 0 0 0]
 1 [0 2 1 0]
 2 [0 1 2 0]
 3 [1 0 0 2]]
   0  1  2  3
  PREDICE
```

Esperaba un 3 pero la red respondió 0

Sklearn.metrics.confusion_matrix

```
Y_train = [0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]
Y_pred  = [0, 2, 1, 3, 0, 1, 2, 0, 0, 1, 2, 3]
```

```
MM = metrics.confusion_matrix(Y_train, Y_pred)
print("Matriz de confusión:\n%s" % MM)
```

```
Matriz de confusión:
 0  [[3 0 0 0]
 1  [0 2 1 0]
 2  [0 1 2 0]
 3  [1 0 0 2]]
    0  1  2  3
    PREDICE
```

Los valores fuera de la diagonal principal son errores

Sklearn.metrics.classification_report

```
Y_train = [0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]
Y_pred  = [0, 2, 1, 3, 0, 1, 2, 0, 0, 1, 2, 3]
report = metrics.classification_report(Y_train, Y_pred)
print("Resultado de la clasificación:\n%s" % report)
```

Resultado de la clasificación:

	precision	recall	f1-score	support
0	0.75	1.00	0.86	3
1	0.67	0.67	0.67	3
2	0.67	0.67	0.67	3
3	1.00	0.67	0.80	3
accuracy			0.75	12
macro avg	0.77	0.75	0.75	12
weighted avg	0.77	0.75	0.75	12

Matriz de confusión:

```
[[3 0 0 0]
 [0 2 1 0]
 [0 1 2 0]
 [1 0 0 2]]
```

Sklearn.metrics.classification_report

```
Y_train = [0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]
Y_pred  = [0, 2, 1, 3, 0, 1, 2, 0, 0, 1, 2, 3]
report = metrics.classification_report(Y_train, Y_pred)
print("Resultado de la clasificación:\n%s" % report)
```

Resultado de la clasificación:



	precision	recall	f1-score	support
0	0.75	1.00	0.86	3
1	0.67	0.67	0.67	3
2	0.67	0.67	0.67	3
3	1.00	0.67	0.80	3
accuracy			0.75	12
macro avg	0.77	0.75	0.75	12
weighted avg	0.77	0.75	0.75	12

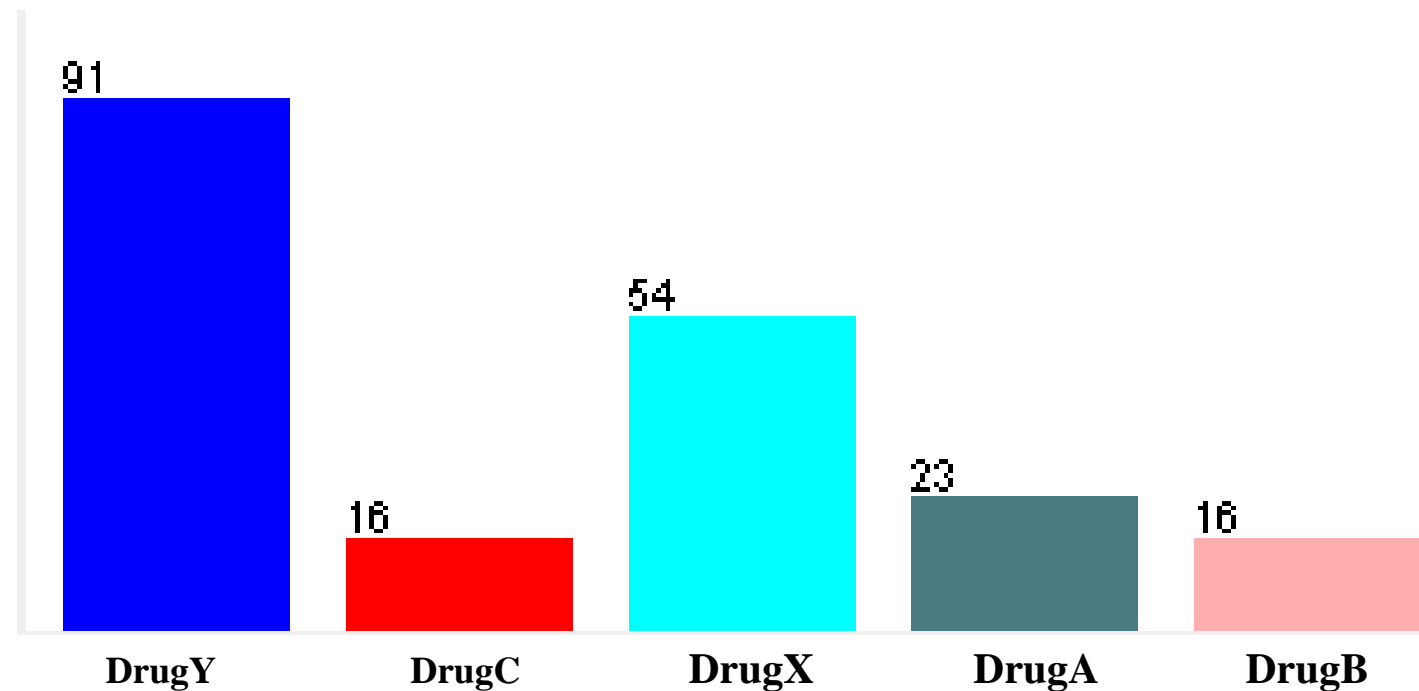
F1-score

$$F1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precisión} + \text{recall}}$$

Ejercicio: Predicción de fármacos

105

- Se busca predecir si el tipo de fármaco que se debe administrar a un paciente afectado de rinitis alérgica es el habitual (DrugY) o no.



Ejercicio: Predicción de fármacos

106

- Para ello se hará uso de la información disponible en las historias clínicas de pacientes atendidos previamente. Las variables relevadas son:
 - ▣ Age: Edad
 - ▣ Sex: Sexo
 - ▣ BP (Blood Pressure): Tensión sanguínea.
 - ▣ Cholesterol: nivel de colesterol.
 - ▣ Na: Nivel de sodio en la sangre.
 - ▣ K: Nivel de potasio en la sangre.
 - ▣ Cada paciente ha sido medicado con un único fármaco de entre cinco posibles: DrugA, DrugB, DrugC, DrugX, DrugY.

Ejercicio: Predicción de fármacos

107

- El archivo **DrugY.csv** contiene 200 muestras de pacientes atendidos previamente.

Nro.	Age	Sex	BP	Colesterol	Na	K	Drug
1	23	F	HIGH	HIGH	0,792535	0,031258	drugY
2	47	M	LOW	HIGH	0,739309	0,056468	drugC
3	47	M	LOW	HIGH	0,697269	0,068944	drugC
4	28	F	NORMAL	HIGH	0,563682	0,072289	drugX
5	61	F	LOW	HIGH	0,559294	0,030998	drugY
...
...
...
197	16	M	LOW	HIGH	0,743021	0,061886	drugC
198	52	M	NORMAL	HIGH	0,549945	0,055581	drugX
199	23	M	NORMAL	NORMAL	0,78452	0,055959	drugX
200	40	F	LOW	NORMAL	0,683503	0,060226	drugX

Ejercicio: Predicción de fármacos

108

- Entrene un perceptrón para predecir si el tipo de fármaco que se debe administrar a un paciente afectado de rinitis alérgica es el habitual (DrugY) o no. Utilice el 80% de los ejemplos para entrenar y el 20% para testear.
 - ▣ Pruebe ambas numerizaciones.
 - ▣ Pruebe resolver el problema
 - Sin normalizar los datos
 - Normalizando linealmente
 - Normalizando utilizando media y desvío