

# Redes Neurais

Perceptron

# Perceptron

Perceptron - Primeira rede neural escrita como algoritmo

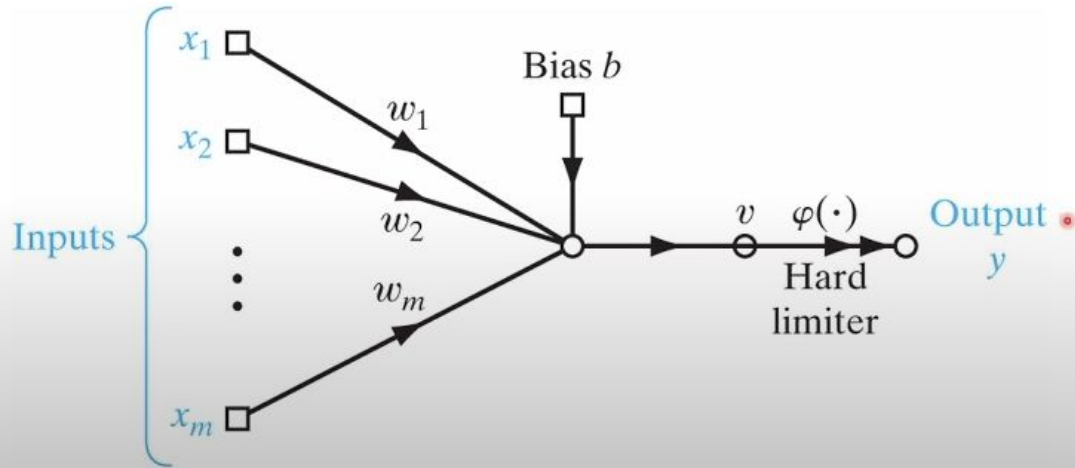
-A maneira mais simples de se utilizar uma rede neural para classificar exemplos linearmente separáveis, em que pode-se dividir os elementos com apenas um hiperplano de separação.

-Um único neurônio e um bias

-Foi provado que se os exemplos mostrados no treinamento, pertencerem às classes linearmente separáveis, o algoritmo converge

# Perceptron

- Rosenblatt's Perceptron uses the McCulloch-Pitts neuron model



Pesos sinápticos  $w$   
Entradas  $X$   
Bias  $b$

# Perceptron

$$v = \sum_{i=1}^m w_i x_i + b$$

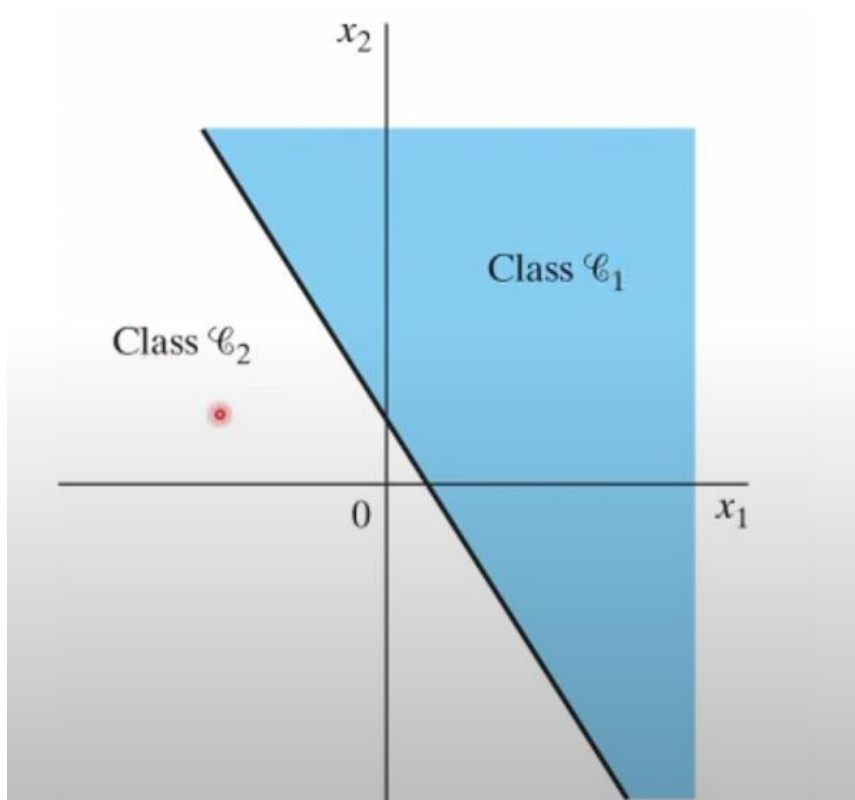
o OBJETIVO do perceptron é classificar corretamente um conjunto de exemplos.

# Perceptron

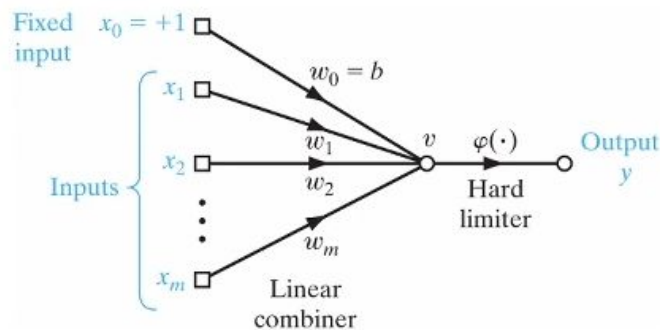
- An example represented by  $x_1, x_2, \dots, x_m$  is classified as  $\mathcal{C}_1$  if the output  $y$  is  $+1$  and as  $\mathcal{C}_2$  if output  $y$  is  $-1$ . There are two regions separated by a hyperplane:

$$\sum_{i=1}^m w_i x_i + b = 0$$

# Perceptron



# Perceptron



Upper Saddle River, New Jersey  
All rights reserved

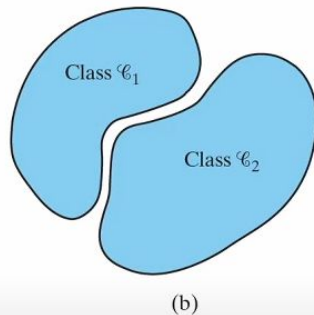
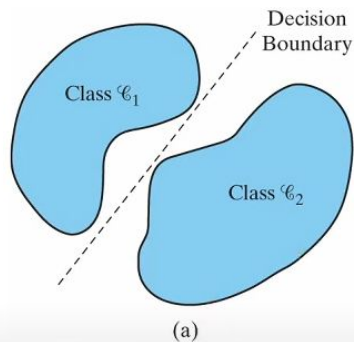
- The bias  $b(n)$  is treated as a weight associated to an input  $+1$
- Input vector:  $\mathbf{x}(n) = [+1, x_1(n), x_2(n), \dots, x_m(n)]^T$
- Weight vector:  $\mathbf{w}(n) = [b, w_1(n), w_2(n), \dots, w_m(n)]^T$

$$v(n) = \sum_{i=0}^m w_i(n) x_i(n) = \mathbf{w}^T(n) \mathbf{x}(n)$$

# Perceptron

Os pesos sinápticos são ajustados em um processo iterativo usando a algoritmo que converge de Perceptron

Como a ilustração mostra, o bias é tratado com o peso associado a entrada +1



Copyright © 2009 by Pearson  
Upper Saddle River, NJ

- $\mathbf{w}^T \mathbf{x} = 0$  defines a separation hyperplane
- $\mathbf{w}^T \mathbf{x} > 0$  for all vector  $\mathbf{x}$  belonging to class  $\mathcal{C}_1$
- $\mathbf{w}^T \mathbf{x} \leq 0$  for all vector  $\mathbf{x}$  belonging to class  $\mathcal{C}_2$



# Perceptron

Caso o  $n$ -ésimo vetor  $\mathbf{x}$  é classificado pelo vetor  $\mathbf{w}$  na  $n$ -ésima iteração do algoritmo, nenhuma correção é realizada no peso do vetor

■  $\mathbf{w}(n+1) = \mathbf{w}(n)$  if  $\mathbf{w}^T \mathbf{x}(n) > 0$  e  $\mathbf{x}(n)$  belongs to class  $\mathcal{C}_1$

■  $\mathbf{w}(n+1) = \mathbf{w}(n)$  if  $\mathbf{w}^T \mathbf{x}(n) \leq 0$  e  $\mathbf{x}(n)$  belongs to class  $\mathcal{C}_2$

□ On the contrary, the weight vector is updated.

■  $\mathbf{w}(n+1) = \mathbf{w}(n) - \eta(n)\mathbf{x}(n)$  if  $\mathbf{w}^T(n)\mathbf{x}(n) > 0$  and  $\mathbf{x}(n)$  belongs to class  $\mathcal{C}_2$

■  $\mathbf{w}(n+1) = \mathbf{w}(n) - \eta(n)\mathbf{x}(n)$  if  $\mathbf{w}^T(n)\mathbf{x}(n) \leq 0$  and  $\mathbf{x}(n)$  belongs to class  $\mathcal{C}_1$

□  $\eta(n)$  is the learning rate that controls the weight adjustment

# Perceptron

A saída do neurônio é calculada usando uma função, nesse caso pode ser a função sinal,

ou seja, se  $v$  é maior que 0 saída 1, se menor saída é -1,  $v$  é o produto interno

**We express the output  $y(n)$  in a compact way:**

$$y(n) = \text{sgn}[\mathbf{w}^T(n)\mathbf{x}(n)]$$

# Perceptron

*Variables and Parameters:*

$\mathbf{x}(n)$  =  $(n + 1)$ -by-1 input vector  
=  $[+1, x_1(n), x_2(n), \dots, x_m(n)]^T$

$\mathbf{w}(n)$  =  $(m + 1)$ -by-1 weight vector  
=  $[b, w_1(n), w_2(n), \dots, w_m(n)]^T$

$b$  = bias

$y(n)$  = actual response (quantized)

$d(n)$  = desired response

$\eta$  = learning-rate parameter, a positive constant less than unity

1. *Initialization.* Set  $\mathbf{w}(0) = \mathbf{0}$ . Then perform the following computations for time-step  $n = 1, 2, \dots$
2. *Activation.* At time-step  $n$ , activate the perceptron by applying continuous-valued input vector  $\mathbf{x}(n)$  and desired response  $d(n)$ .
3. *Computation of Actual Response.* Compute the actual response of the perceptron as

$$y(n) = \text{sgn}[\mathbf{w}^T(n)\mathbf{x}(n)]$$

where  $\text{sgn}(\cdot)$  is the signum function.

4. *Adaptation of Weight Vector.* Update the weight vector of the perceptron to obtain

$$\mathbf{w}(n + 1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n)$$

where

$$d(n) = \begin{cases} +1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathcal{C}_1 \\ -1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathcal{C}_2 \end{cases}$$

5. *Continuation.* Increment time step  $n$  by one and go back to step 2.

# Perceptron

Os pesos são ajustados de acordo com o valor do produto interno  $w^T x$

Caso o produto interno em uma iteração tem um sinal errado os pesos são ajustados para classificar corretamente o exemplo da iteração  $n+1$

O que acontece é, na figura abaixo o produto interno entre o vetor de entrada e o vetor  $w$

deu positivo, mas o produto interno deveria ser negativo então o ajuste de peso é realizado,

puxando o vetor  $w$ , para longe do vetor  $x$ , para assim , o produto interno dar negativo.



# Perceptron

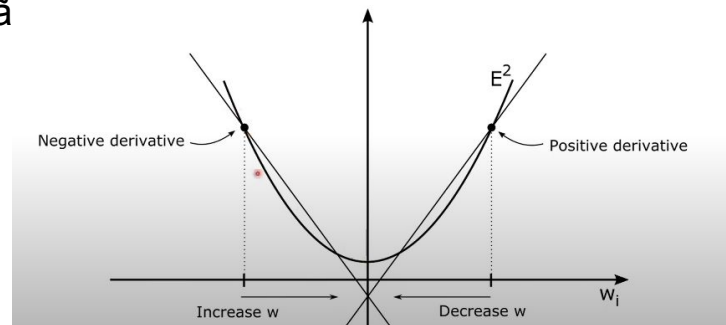
Para minimizar o erro, devemos utilizar derivadas.

Quando o Erro está do lado direito, a derivada é positiva com relação ao peso, então para isso o  $w$  deve diminuir para jogar o  $w$  para mais perto do erro correto.

Quando o erro está do lado esquerdo, a derivada é negativo com relação ao peso, então o  $w$  deve aumentar para jogar o  $w$  para mais perto do erro correto.

O algoritmo faz isso automaticamente.

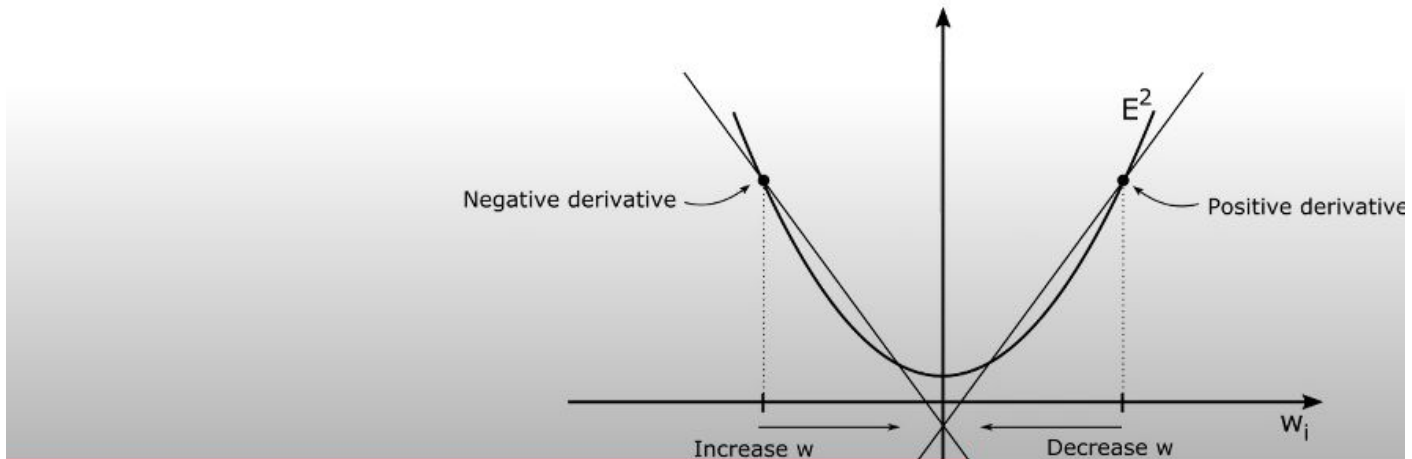
$$E^2 = (d(n) - y(n))^2 = (d(n) - \mathbf{w}^T(n)\mathbf{x}(n))^2$$



# Perceptron

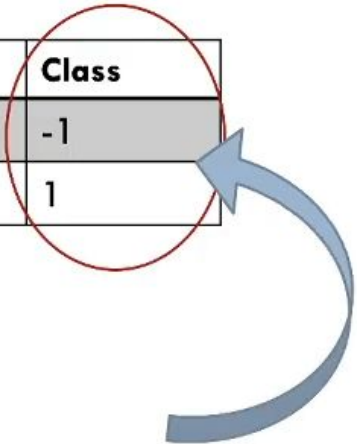
Gradiente Descendente:  $w_i(n+1) = w_i(n) - \eta \frac{dE^2}{dw_i}$

$$\frac{dE^2}{dw_i} = \frac{d(d(n) - y(n))^2}{dw_i(n)} = 2 \times (d(n) - \mathbf{w}^T(n)\mathbf{x}(n)) \times -x_i$$



# Exemplo Perceptron

Data	x1	x2	x3	Class
E1	0	0	1	-1
E2	1	0	0	1

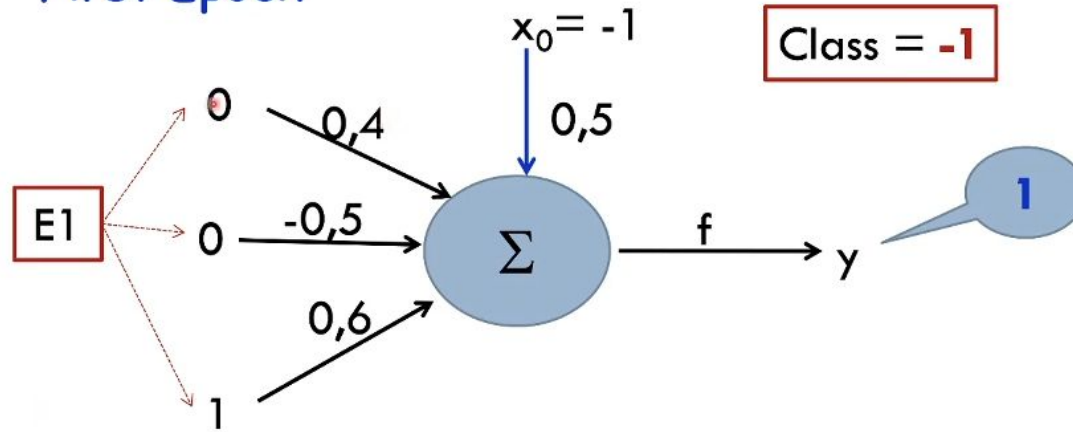


**Supervised  
Learning**

$$f(X) = \begin{cases} 1 & \text{se } X \geq 0 \\ -1 & \text{se } X < 0 \end{cases}$$

# Exemplo Perceptron

First Epoch



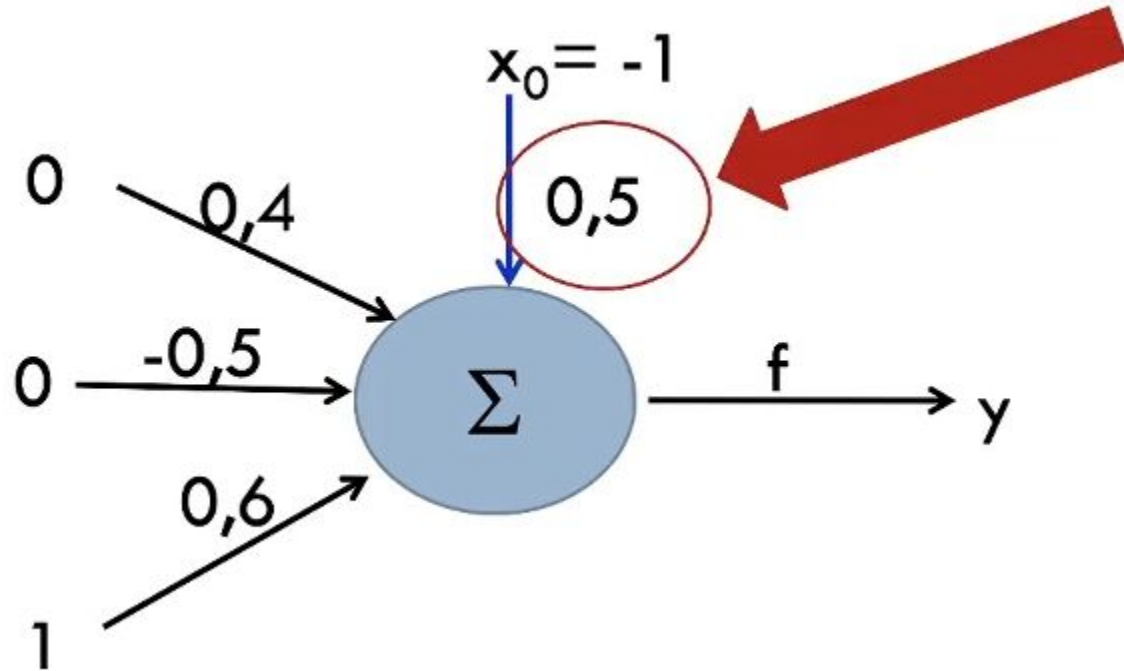
$$X = (-1 * 0,5) + (0 * 0,4) + (0 * -0,5) + (1 * 0,6) = 0,1$$

$$f(0,1) = 1$$

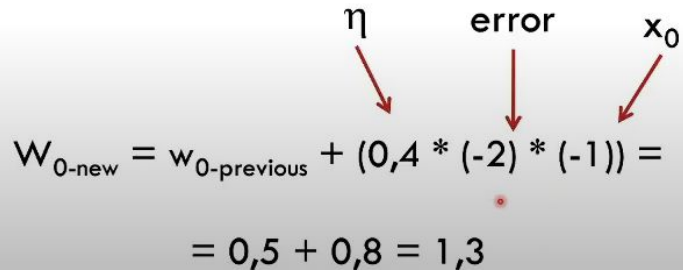
$$e = -1 - 1 = -2$$



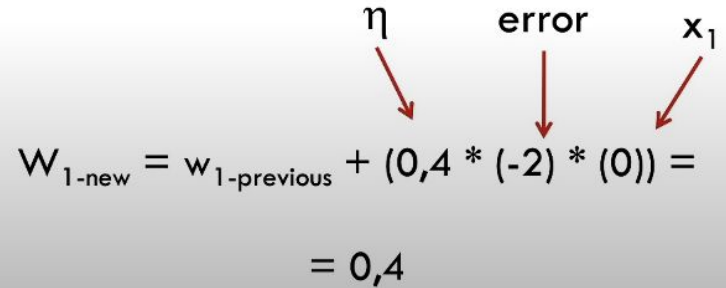
# Exemplo Perceptron

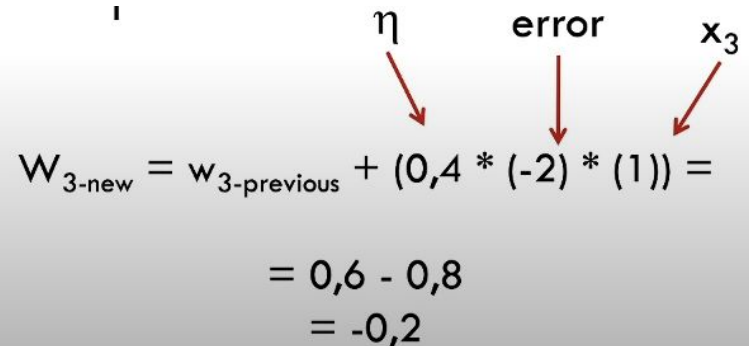


# Exemplo Perceptron

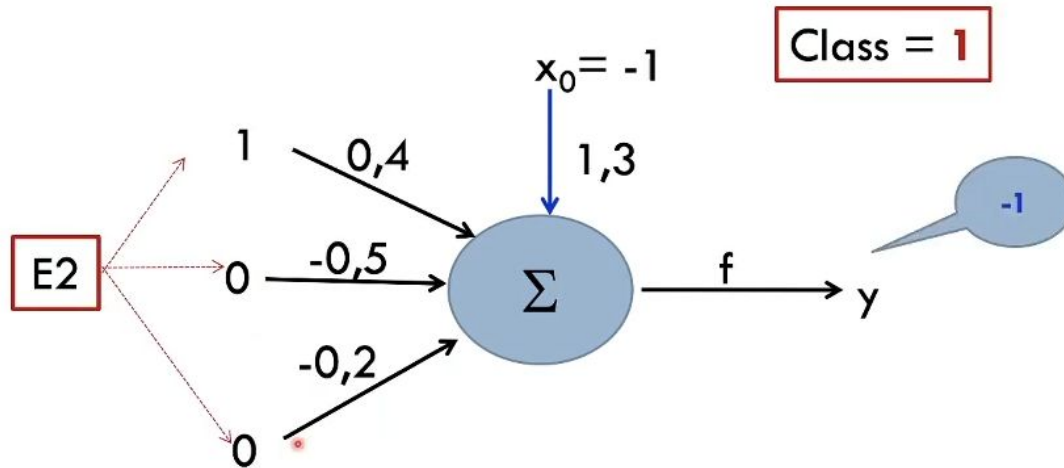

$$\begin{aligned}w_{0\text{-new}} &= w_{0\text{-previous}} + (\eta * \text{error} * x_0) = \\&= 0,5 + 0,8 = 1,3\end{aligned}$$

w2 não está pois sua entrada também é 0


$$\begin{aligned}w_{1\text{-new}} &= w_{1\text{-previous}} + (\eta * \text{error} * x_1) = \\&= 0,4\end{aligned}$$


$$\begin{aligned}w_{3\text{-new}} &= w_{3\text{-previous}} + (\eta * \text{error} * x_3) = \\&= 0,6 - 0,8 \\&= -0,2\end{aligned}$$

# Exemplo Perceptron

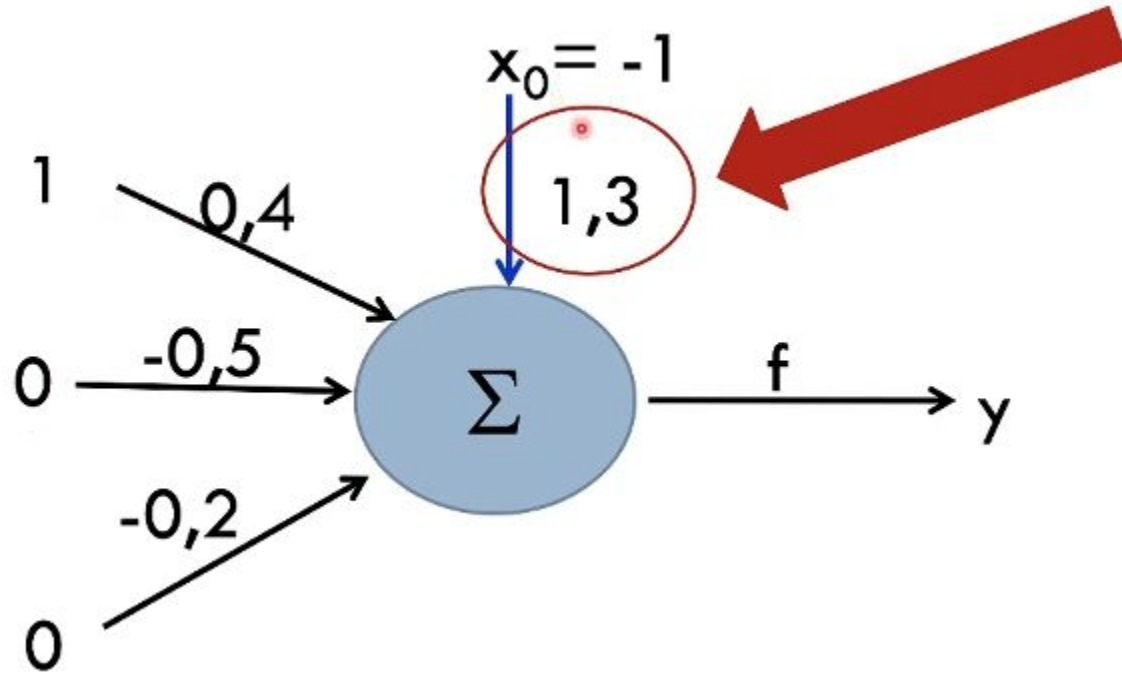


$$X = (-1 * 1,3) + (1 * 0,4) + (0 * -0,5) + (0 * -0,2) = -0,9$$

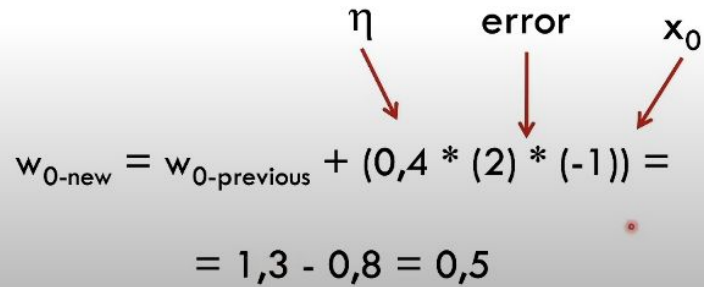
$$f(-0,9) = -1$$

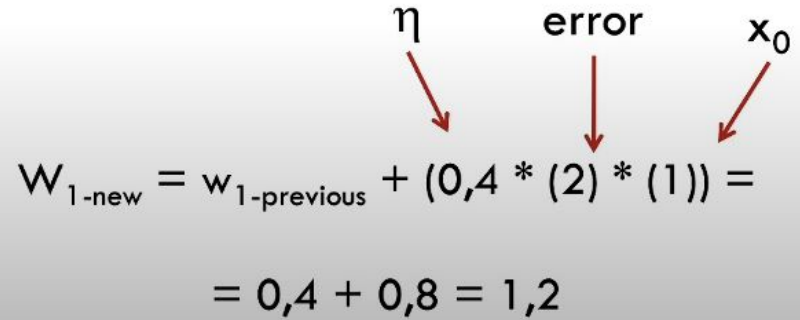
$$e = 1 - (-1) = 2$$

# Exemplo Perceptron



# Exemplo Perceptron

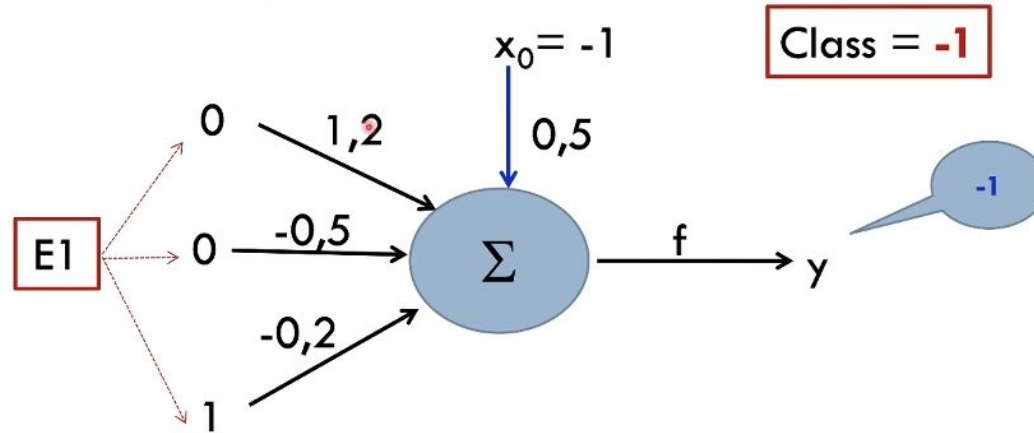
$$\begin{aligned} w_{0\text{-new}} &= w_{0\text{-previous}} + (\eta * \text{error} * x_0) = \\ &= 1,3 - 0,8 = 0,5 \end{aligned}$$


$$\begin{aligned} w_{1\text{-new}} &= w_{1\text{-previous}} + (\eta * \text{error} * x_0) = \\ &= 0,4 + 0,8 = 1,2 \end{aligned}$$


w2 nem w3 estão pois suas entradas são 0

# Exemplo Perceptron

## Second Epoch



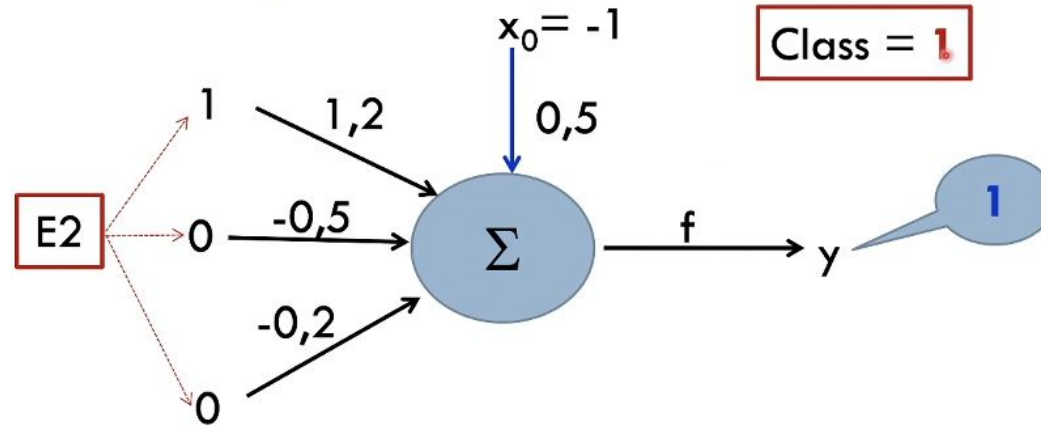
$$X = (-1 * 0,5) + (0 * 1,2) + (0 * -0,5) + (1 * -0,2) = -0,7$$

$$f(-0,7) = -1$$

$$e = -1 - (-1) = 0$$

# Exemplo Perceptron

Second Epoch



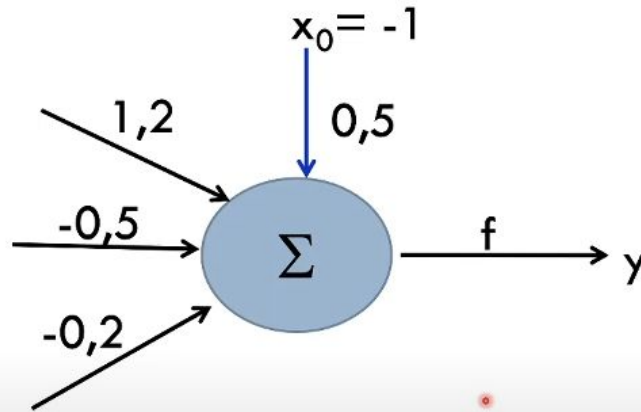
$$X = (-1 * 0,5) + (1 * 1,2) + (0 * -0,5) + (0 * -0,2) = 0,7$$

$$f(0,7) = 1$$

$$e = 1 - 1 = 0$$

# Exemplo Perceptron

## Final Network



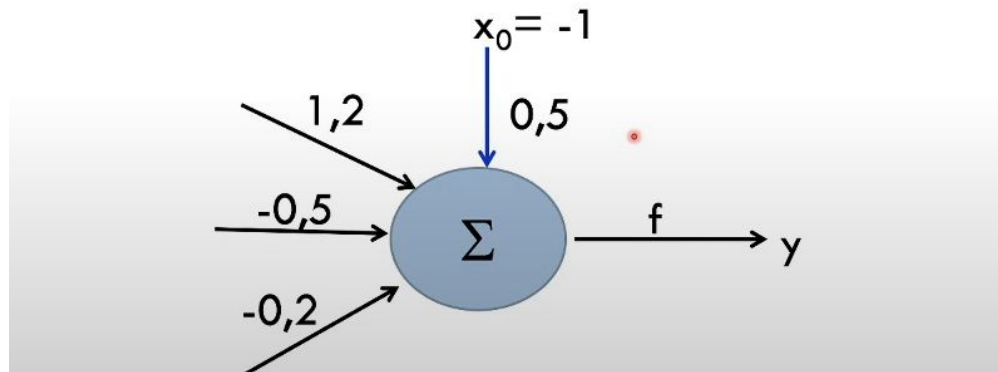
Não ocorreram erros durante a última época.



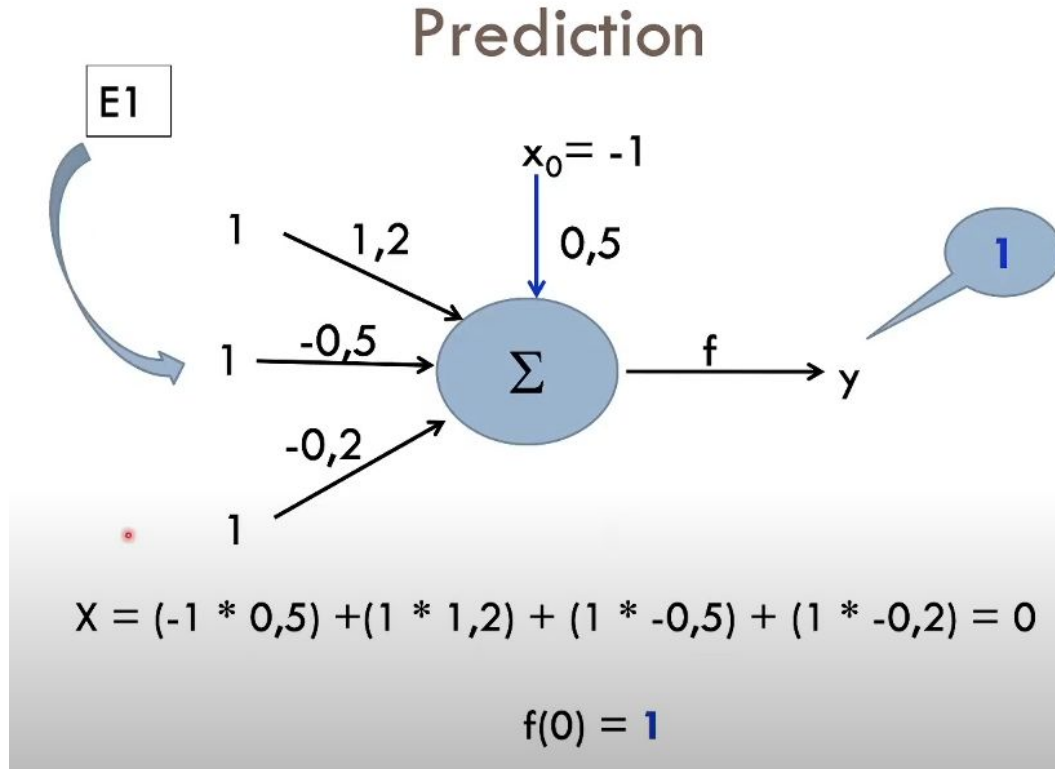
# Exemplo Perceptron

## Prediction

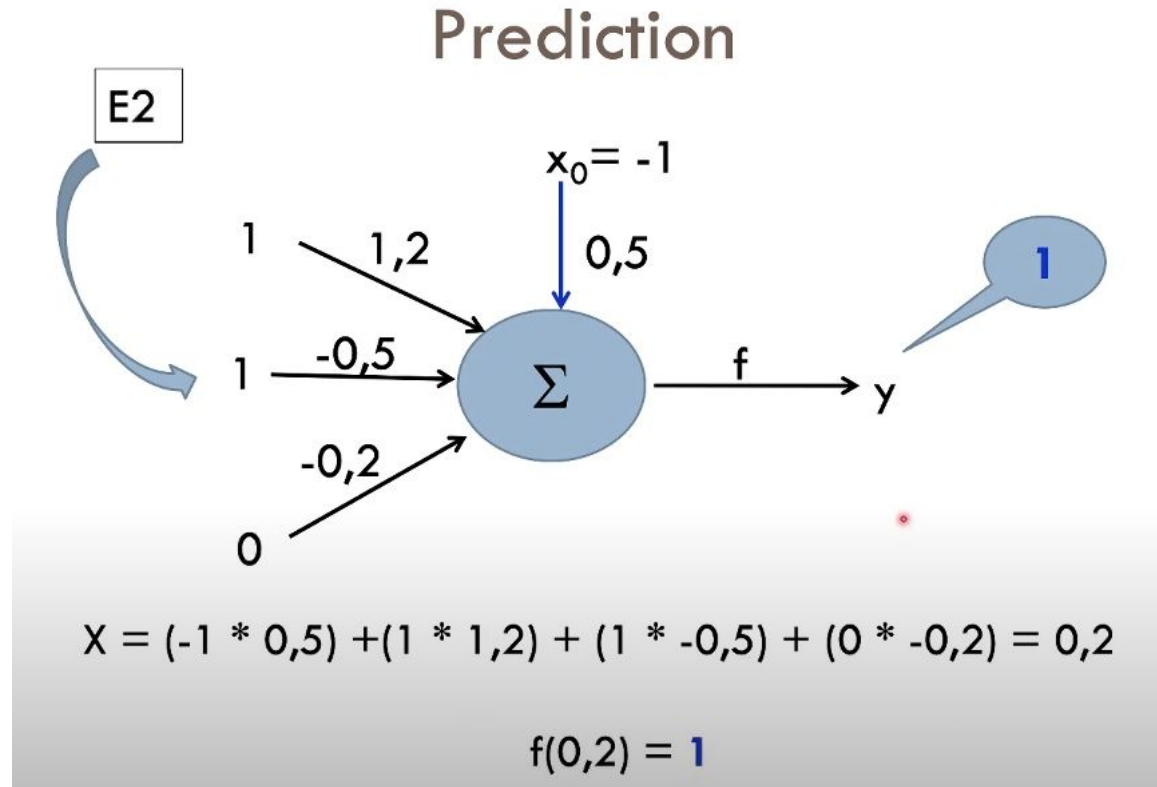
Data	X1	X2	x3	Class
E1	1	1	1	?
E2	1	1	0	?
E3	0	1	1	?



# Exemplo Perceptron

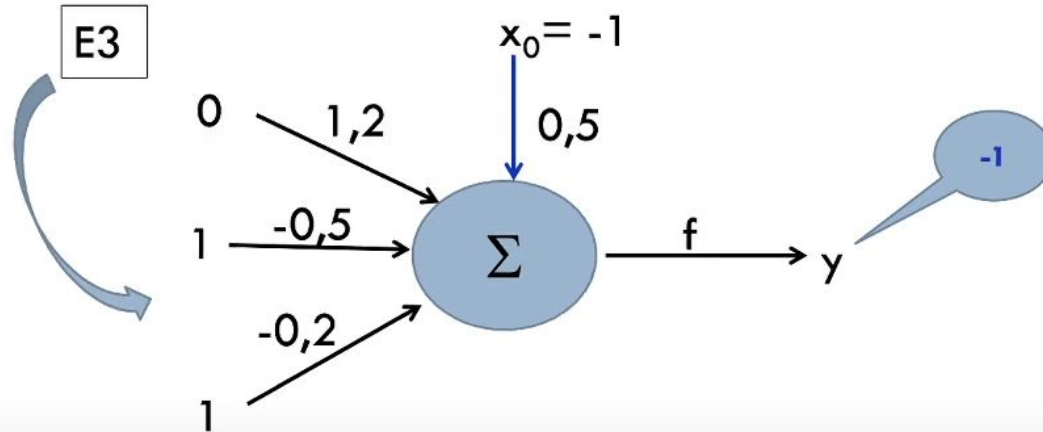


# Exemplo Perceptron



# Exemplo Perceptron

## Prediction



$$X = (-1 * 0,5) + (0 * 1,2) + (1 * -0,5) + (1 * -0,2) = -1,2$$

$$f(-1,2) = -1$$

# Exemplo Perceptron

## Prediction

Data	X1	X2	x3	Class
E1	1	1	1	1
E2	1	1	0	1
E3	0	1	1	-1

