

# Perceptrón Simple

Sistemas de Inteligencia Artificial

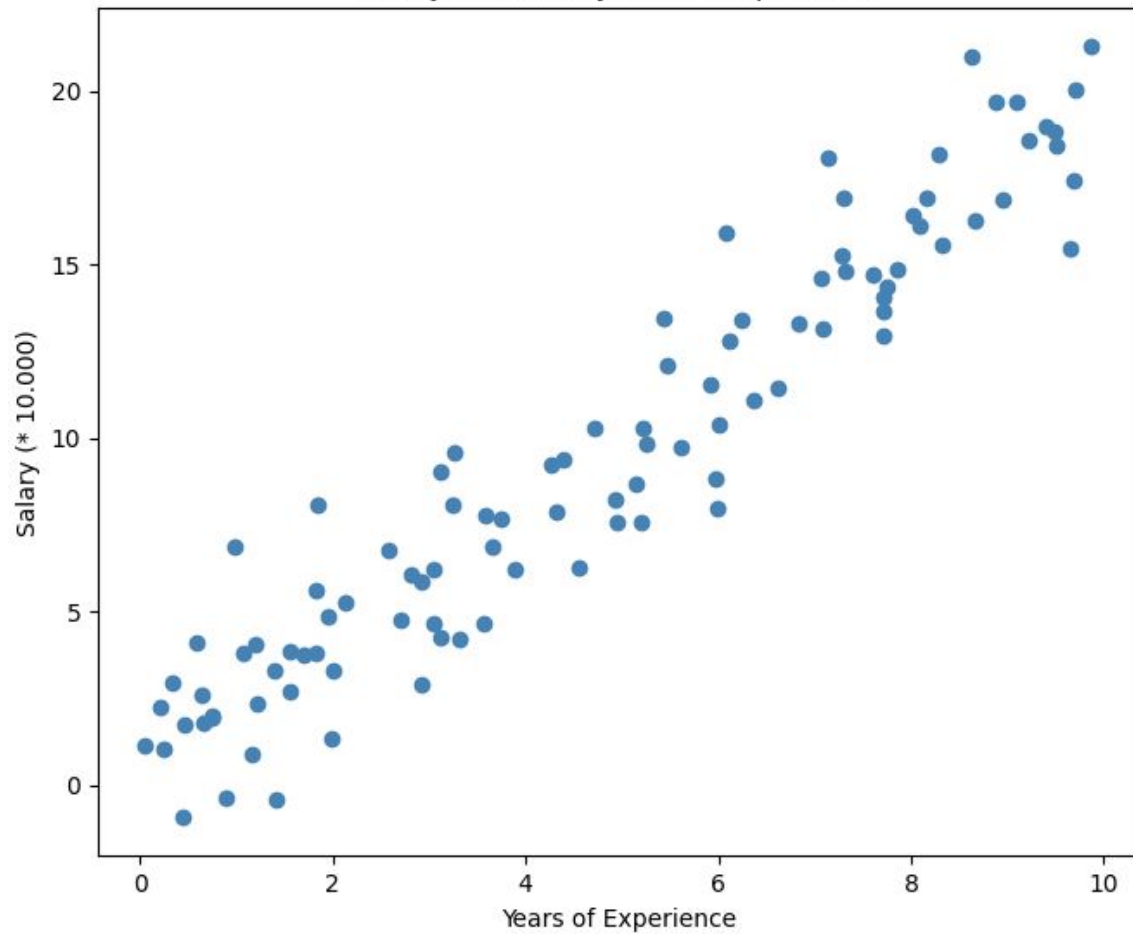
Segundo Cuatrimestre 2025

Alan Pierri  
Rodrigo Ramele  
Eugenia Piñeiro  
Marina Fuster  
Luciano Bianchi  
Santiago Reyes  
Marco Scilipoti  
Paula Oseroff  
Joaquín Girod

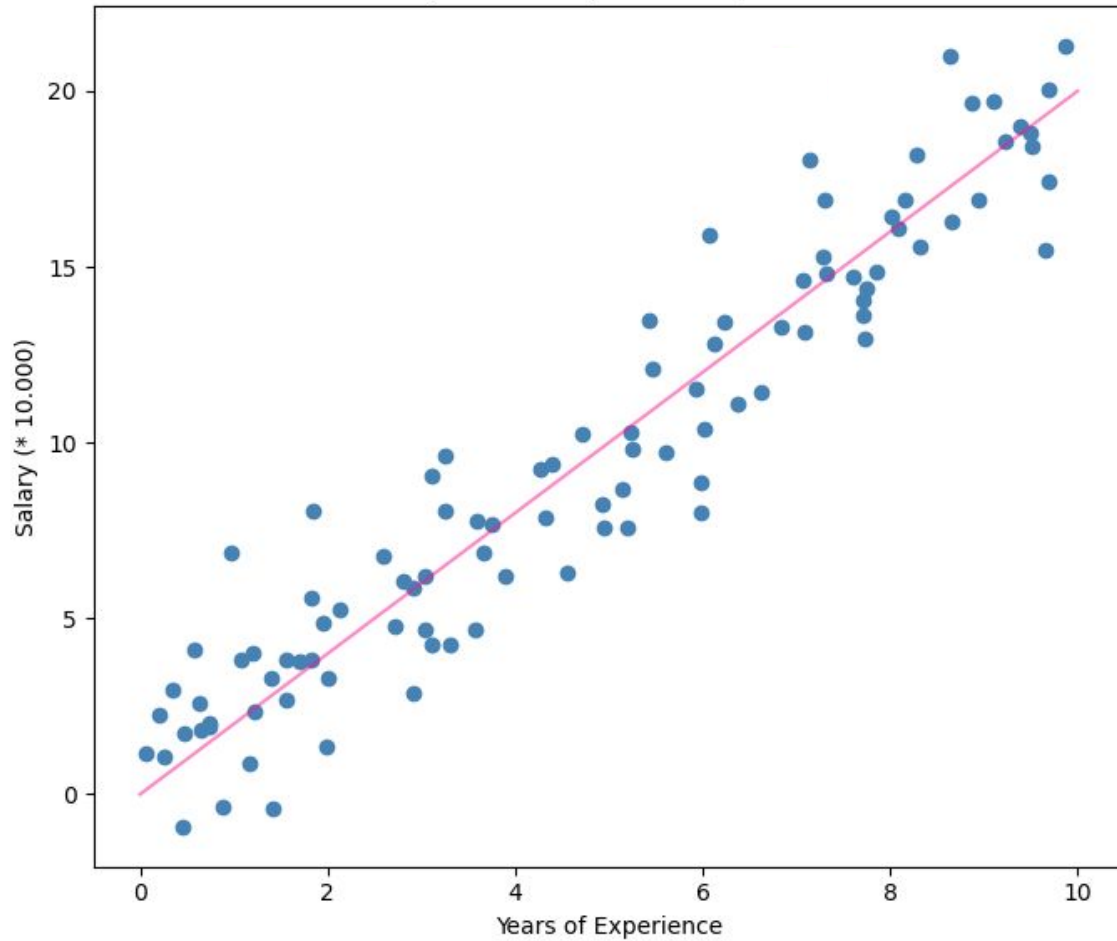
## RESUMEN DE PERCEPTRÓN SIMPLE

- McCulloch y Pitts sientan las bases del modelo de neurona que se utiliza en el área de redes neuronales. Este modelo se denomina **Perceptrón**.
- El modelo de McCulloch y Pitts permite resolver problemas linealmente separables.
- Rosenblatt provee el mecanismo que permite obtener los pesos del perceptrón de manera iterativa
- No es lo mismo aprendizaje que generalización

Salary based on years of experience

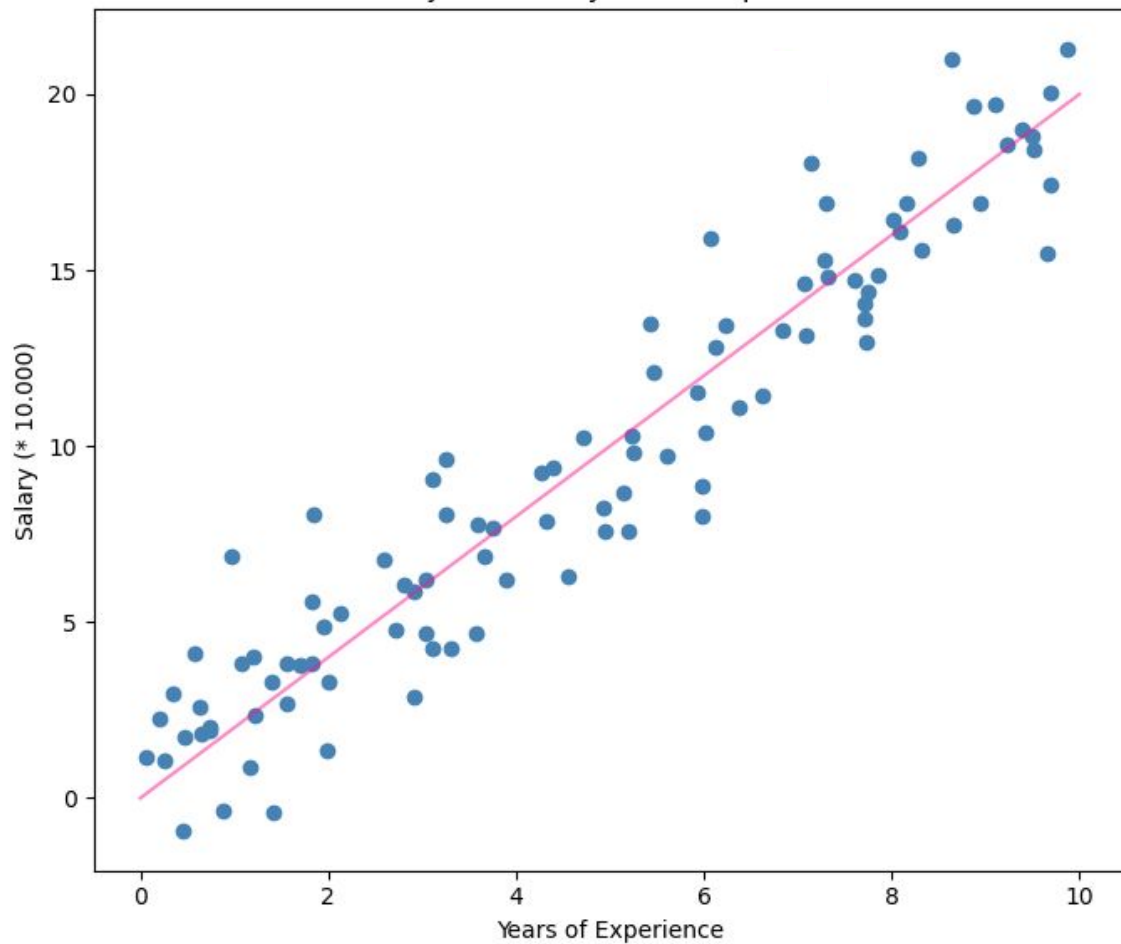


Salary based on years of experience



```
salary(years_of_experience):  
    return a * years_of_experience + b
```

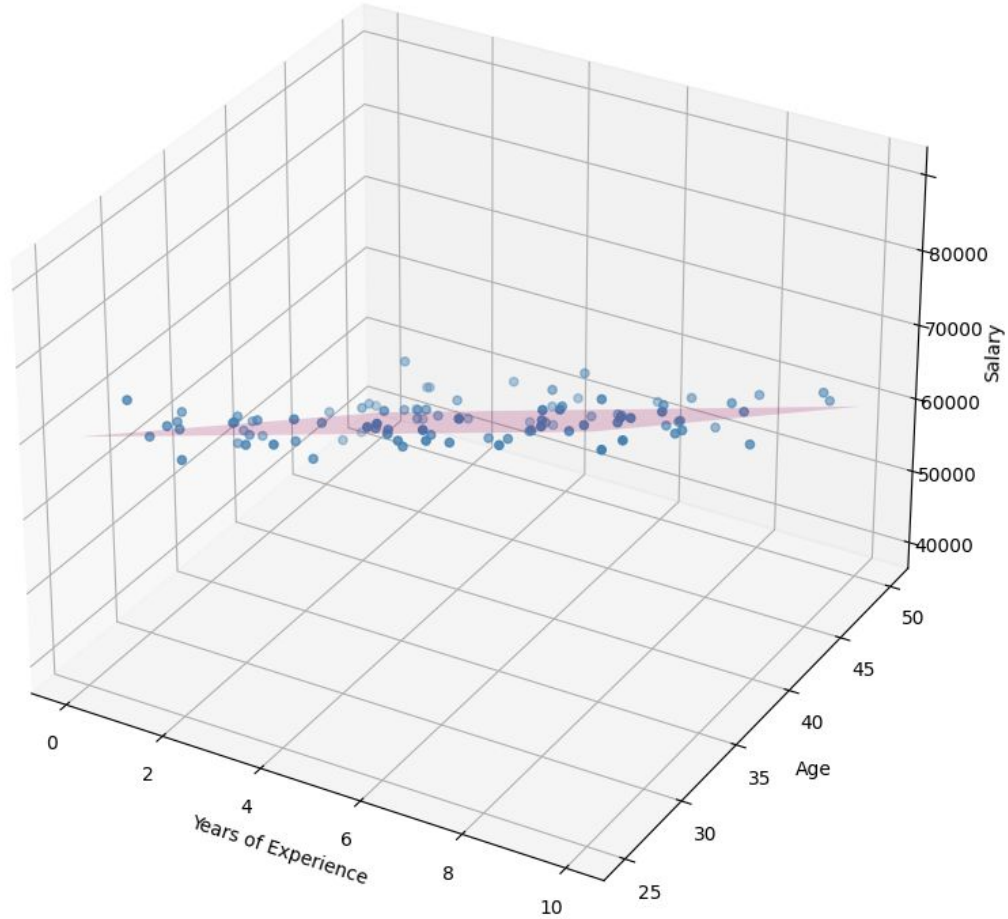
Salary based on years of experience



```
salary(years_of_experience):  
    return a * years_of_experience + b
```

```
salary(x1):  
    return  $w_1 * x_1 + w_0$ 
```

Salary based on age and years of experience



```
salary(x1, x2):  
    return w1 * x1 + w2 * x2 + w0
```

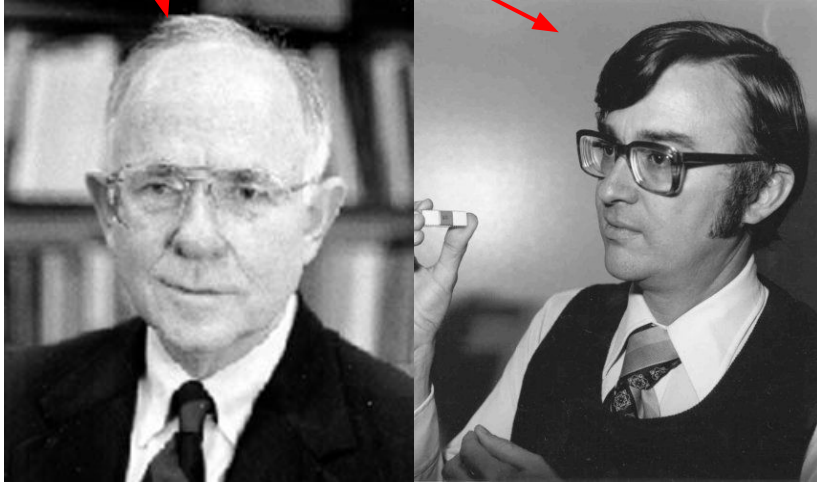
## ¿CÓMO RESUELVO EL PROBLEMA?



Necesito encontrar los valores de  $w$  que me permitan encontrar un hiperplano que ajuste lo mejor posible al conjunto de datos

$$salario = \sum_{i=1}^n x_i \cdot w_i + w_0$$

Widrow-Hoff - 1960



ADALINE: ADaptive LINear Element  
(o perceptrón simple ***lineal***)

Cambiamos la función de activación por la identidad:

$$O(h) = \theta(h(x)) = \sum_{i=1}^n x_i \cdot w_i + w_0$$

↓

$$\theta(h) = h$$

La salida del perceptrón ya no está confinada a ser binaria: ***toma valores en los reales***



# ¿CÓMO RESUELVO EL PROBLEMA?



Necesito encontrar los valores de  $\mathbf{w}$  que me permitan encontrar un hiperplano que ajuste lo mejor posible al conjunto de datos

$$O(h) = \theta(h(x)) = \sum_{i=1}^n x_i \cdot w_i + w_0$$

## APRENDIZAJE PARA EL PERCEPTRÓN LINEAL

$$O(x) = \sum_{i=1}^n x_i \cdot w_i + w_0$$

**Rosenblatt:** Cada vez que la neurona recibe un estímulo, los pesos sinápticos pueden actualizarse (proceso iterativo):

$$w^{nuevo} = w^{anterior} + \boxed{\Delta w} \text{ ?}$$

# ¿CÓMO DEFINIMOS QUE EL PERCEPTRÓN SE EQUIVOCA?



```
salary(x1):  
    return  $w_1 * x_1 + w_0$ 
```

## ¿CÓMO DEFINIMOS QUE EL PERCEPTRÓN SE EQUIVOCA?



Podemos usar la siguiente función de error:

$$E(O) = \frac{1}{2} \sum_{\mu=0}^{p-1} (\zeta^{\mu} - O^{\mu})^2$$

$$E(\cdot) = \frac{1}{2} \sum_{\mu=1}^p (\zeta^{\mu} - g(\xi_i^{\mu}))$$

La salida del perceptrón depende, a su vez, de los pesos sinápticos.

$$E(w) = \frac{1}{2} \sum_{\mu=0}^{p-1} (\zeta^{\mu} - \theta(\sum_{i=0}^n x_i^{\mu} \cdot w_i))^2$$

## ¿CÓMO DEFINIMOS QUE EL PERCEPTRÓN SE EQUIVOCA?

Podemos usar la siguiente función de error o costo:

$$E(O) = \frac{1}{2} \sum_{\mu=0}^{p-1} (\zeta^{\mu} - O^{\mu})^2$$

La salida del perceptrón depende, a su vez, de los pesos sinápticos.

$$E(w) = \frac{1}{2} \sum_{\mu=0}^{p-1} (\zeta^{\mu} - \theta(\sum_{i=0}^n x_i^{\mu} \cdot w_i))^2$$



Incluye el “umbral” o “bias”

¿CÓMO “**APRENDE**” EL PERCEPTRÓN CON ESTA FUNCIÓN DE COSTO?

$$E(w) = \frac{1}{2} \sum_{\mu=0}^{p-1} (\zeta^{\mu} - \theta(\sum_{i=0}^n x_i^{\mu} \cdot w_i))^2$$

Fórmula de actualización de los pesos al evaluar un dato de entrada:

$$w^{nuevo} = w^{anterior} + \Delta w$$



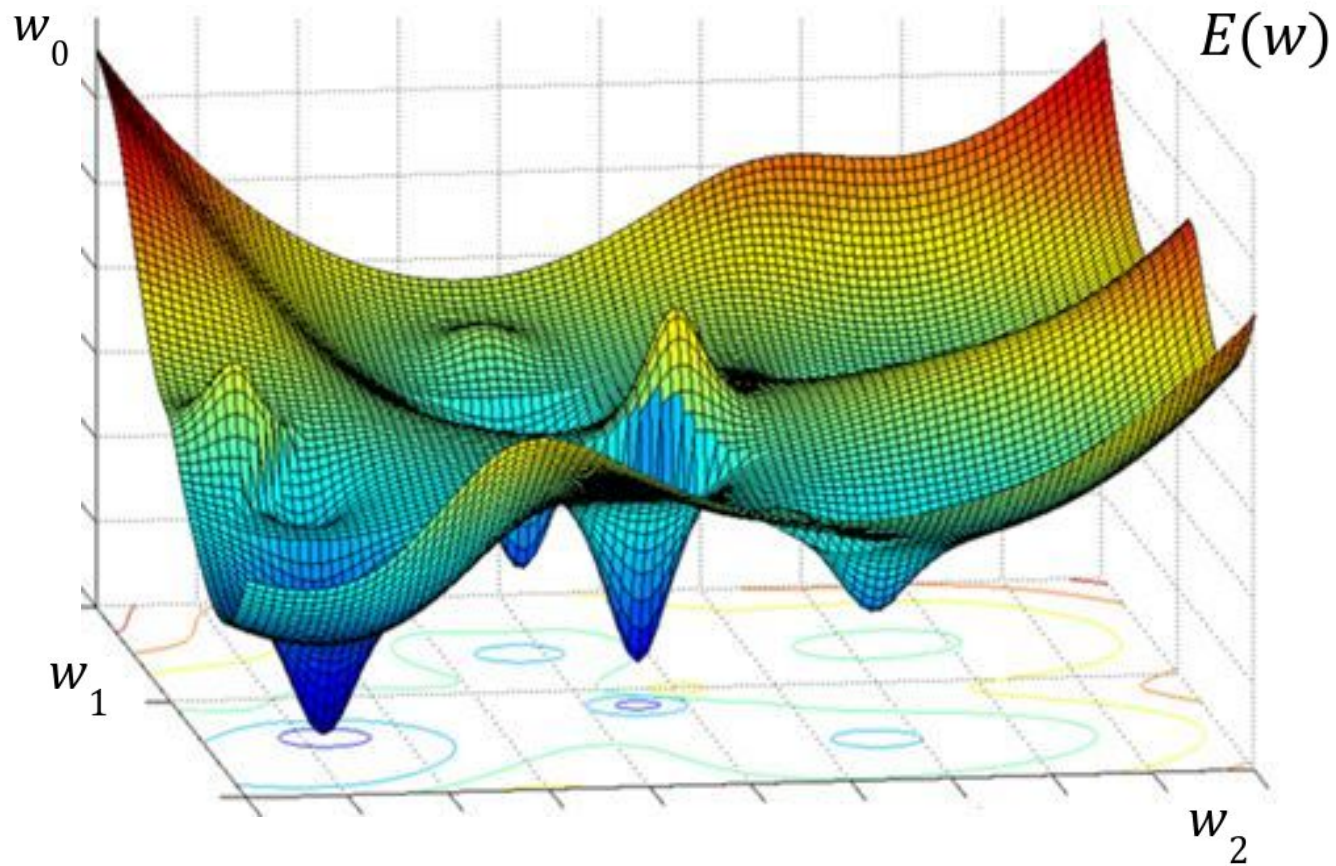
$$\Delta w = -\eta \frac{\partial E}{\partial w}$$

#### Condiciones sobre la dirección de movimiento

Por supuesto, la idea es que sea descendente, o sea  $d_k^t \nabla f(x_k) < 0$ .

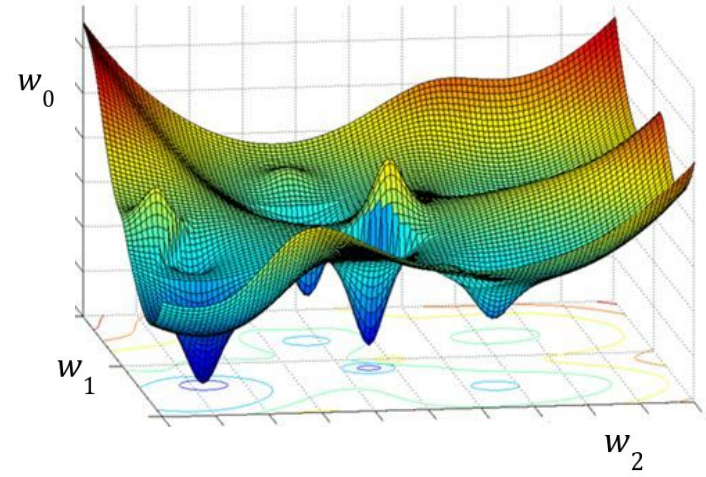
- Recordar que el gradiente es la dirección de máximo crecimiento de una función.
- Cualquier dirección contraria a la del gradiente, es una dirección de decrecimiento de la función.

¿QUÉ FORMA TIENE LA FUNCIÓN DE COSTO?



DESARROLLO DE  $\frac{\partial E}{\partial w}$

$$\frac{\partial E}{\partial w_i} = \frac{\partial \left( \frac{1}{2} \sum_{\mu=0}^{p-1} \left( \zeta^\mu - \theta \left( \sum_{i=0}^n x_i^\mu \cdot w_i \right) \right)^2 \right)}{\partial w_i}$$





DESARROLLO DE  $\frac{\partial E}{\partial w}$

$$\frac{\partial E}{\partial w_i} = \frac{\partial \left( \frac{1}{2} \sum_{\mu=0}^{p-1} \left( \zeta^\mu - \theta \left( \sum_{i=0}^n x_i^\mu \cdot w_i \right) \right)^2 \right)}{\partial w_i} = \sum_{\mu=0}^{p-1} \left( \left( \zeta^\mu - \theta \left( \sum_{i=0}^n x_i^\mu \cdot w_i \right) \right) (-1) \theta' \left( \sum_{i=0}^n x_i^\mu \cdot w_i \right) x_i^\mu \right)$$

DESARROLLO DE  $\frac{\partial E}{\partial w}$

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \frac{\partial \left( \frac{1}{2} \sum_{\mu=0}^{p-1} \left( \zeta^\mu - \theta \left( \sum_{i=0}^n x_i^\mu \cdot w_i \right) \right)^2 \right)}{\partial w_i} = \sum_{\mu=0}^{p-1} \left( (\zeta^\mu - \overset{o^\mu}{\theta \left( \sum_{i=0}^n x_i^\mu \cdot w_i \right)}) (-1) \theta' \left( \overset{h^\mu}{\sum_{i=0}^n x_i^\mu \cdot w_i} \right) x_i^\mu \right) \\ &= \sum_{\mu=0}^{p-1} (\zeta^\mu - o^\mu) (-1) \theta'(h^\mu) x_i^\mu = - \sum_{\mu=0}^{p-1} (\zeta^\mu - o^\mu) \theta'(h^\mu) x_i^\mu\end{aligned}$$

DESARROLLO DE  $\frac{\partial E}{\partial w}$

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \frac{\partial \left( \frac{1}{2} \sum_{\mu=0}^{p-1} \left( \zeta^\mu - \theta \left( \sum_{i=0}^n x_i^\mu \cdot w_i \right) \right)^2 \right)}{\partial w_i} = \sum_{\mu=0}^{p-1} \left( \left( \zeta^\mu - \theta \left( \sum_{i=0}^n x_i^\mu \cdot w_i \right) \right) (-1) \theta' \left( \sum_{i=0}^n x_i^\mu \cdot w_i \right) x_i^\mu \right) \\ &= \sum_{\mu=0}^{p-1} (\zeta^\mu - o^\mu) (-1) \theta'(h^\mu) x_i^\mu = - \sum_{\mu=0}^{p-1} (\zeta^\mu - o^\mu) \theta'(h^\mu) x_i^\mu\end{aligned}$$

Fórmula de actualización de los pesos al evaluar un dato de entrada:

$$\Delta w = - \eta \frac{\partial E}{\partial w} = \boxed{\eta (\zeta^\mu - o^\mu) \theta'(h) x^\mu}$$

# EL ALGORITMO PARA EL PERCEPTRÓN LINEAL ES IGUAL AL ANTERIOR

```
Initialize weights w to small random values
Set learning rate  $\eta$ 

for a fixed number of epochs:
    For each training example  $\mu$  in the dataset:
        1. Calculate the weighted sum:
            
$$h^\mu = w_0 * x_0^\mu + w_1 * x_1^\mu + w_2 * x_2^\mu + \dots + w_n * x_n^\mu$$


        2. Compute activation given by  $\theta$ :
            
$$o(h^\mu) = \theta(h^\mu) = h^\mu$$


        3. Update the weights and bias:
            For each weight  $w_i$ :
                
$$w_i = w_i + \eta * (y - \text{output}) * \theta'(h^\mu) * x_i^\mu$$


        4. Calculate perceptron error:
            
$$\text{error} = f(x_1^\mu, x_2^\mu, \dots, x_n^\mu)$$

            
$$\text{convergence} = \text{True if error} < \epsilon \text{ else False}$$

            if convergence: break

End
```

## Tip de implementación

Construirse un conjunto de datos que sean puntos pertenecientes a una recta y ajustarlos!

# EL ALGORITMO PARA EL PERCEPTRÓN LINEAL ES IGUAL AL ANTERIOR

```
Initialize weights w to small random values
Set learning rate  $\eta$ 

for a fixed number of epochs:
    For each training example  $\mu$  in the dataset:
        1. Calculate the weighted sum:
            
$$h^\mu = w_0 * x_0^\mu + w_1 * x_1^\mu + w_2 * x_2^\mu + \dots + w_n * x_n^\mu$$

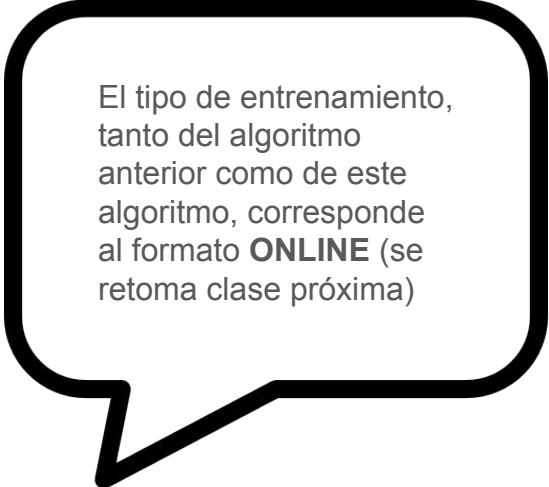

        2. Compute activation given by  $\theta$ :
            
$$o(h^\mu) = \theta(h^\mu) = h^\mu$$


        3. Update the weights and bias:
            For each weight  $w_i$ :
                
$$w_i = w_i + \eta * (y - \text{output}) * \theta'(h^\mu) * x_i^\mu$$

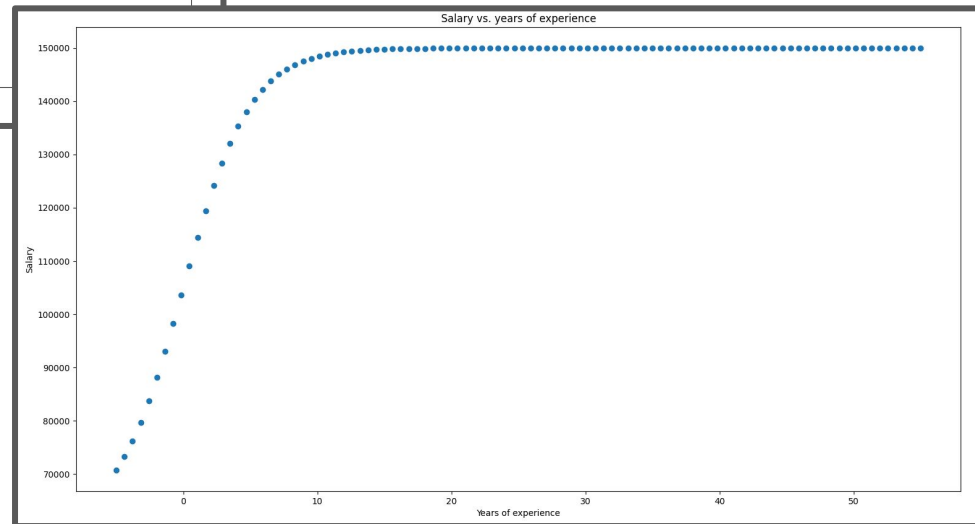
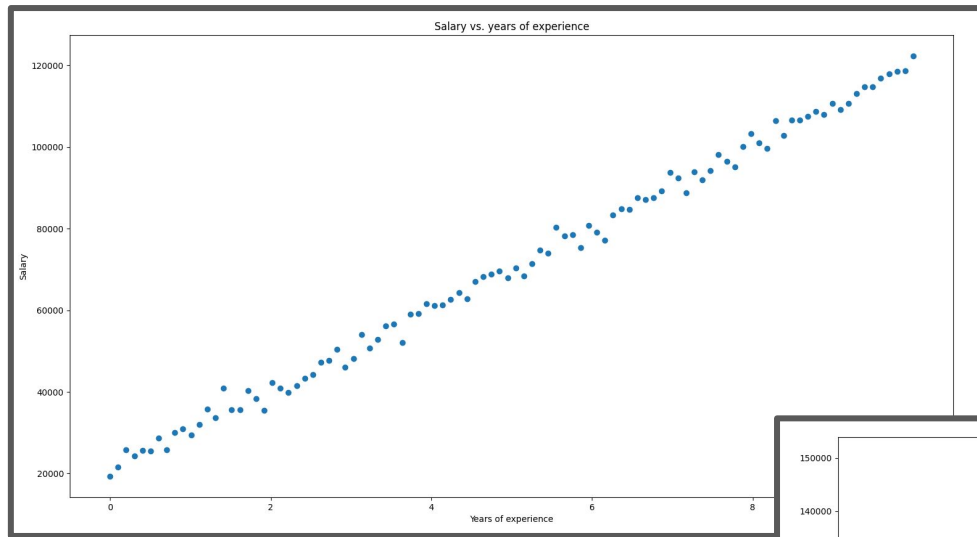

        4. Calculate perceptron error:
            
$$\text{error} = f(x_1^\mu, x_2^\mu, \dots, x_n^\mu)$$

            convergence = True if error <  $\epsilon$  else False
            if convergence: break

End
```



El tipo de entrenamiento, tanto del algoritmo anterior como de este algoritmo, corresponde al formato **ONLINE** (se retoma clase próxima)



## PERCEPTRÓN SIMPLE NO LINEAL

Cambiamos la función de activación por una sigmoidea, ***tanh*** o ***logística***

$$O = \theta\left(\sum_{i=0}^n x_i \cdot w_i\right)$$

$$\theta(x) = \tanh(\beta x) \quad Im = (-1, 1)$$

$$\theta(x) = \frac{1}{1 + \exp^{-2\beta x}} \quad Im = (0, 1)$$

La fórmula de error se mantiene igual al perceptrón lineal:

$$E(O) = \frac{1}{2} \sum_{\mu=0}^{p-1} (\zeta^{\mu} - O^{\mu})^2$$

¿Cómo “**aprende**”? ¡Igual que el anterior!

Cambiamos la función de activación por una sigmoidea, ***tanh*** o ***logística***

$$O = \theta\left(\sum_{i=0}^n x_i \cdot w_i\right)$$

Fórmula de actualización de los pesos al evaluar un dato de entrada:

$$w^{nuevo} = w^{anterior} + \Delta w$$

$$\Delta w = -\eta \frac{\partial E}{\partial w} = \eta \sum_{\mu=0}^{p-1} (\zeta^{\mu} - O^{\mu}) \theta'(h) x^{\mu}$$



## PERCEPTRÓN SIMPLE NO LINEAL

Cambiamos la función de activación por una sigmoidea, ***tanh*** o ***logística***

$$O = \theta\left(\sum_{i=0}^n x_i \cdot w_i\right)$$

### Hyperbolic Tangent Function

$$\theta(h) = \tanh(\beta h)$$

$$Im = (-1, 1)$$

$$\theta'(h) = \beta(1 - \theta^2(h))$$

### Logistic Function

$$\theta(h) = \frac{1}{1 + \exp^{-2\beta h}}$$

$$Im = (0, 1)$$

$$\theta'(h) = 2\beta\theta(h)(1 - \theta(h))$$

# Funciones sigmoideas

## Tangente Hiperbólica

Parámetro **beta** que cambia la forma:

$$\theta(h) = \tanh(\beta h)$$

$$\theta'(h) = \beta(1 - \theta^2(h))$$

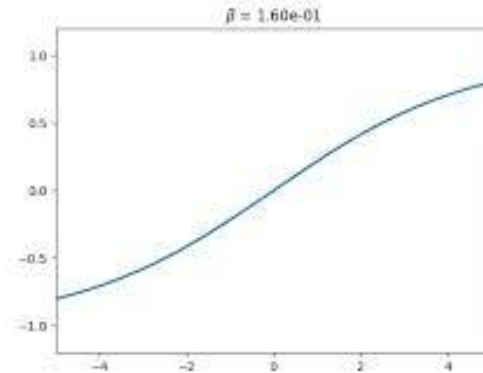
## Función Logística

Parámetro **beta** que cambia la forma:

$$\theta(h) = \frac{1}{1 + \exp^{-2\beta h}}$$

$$\theta'(h) = 2\beta\theta(h)(1 - \theta(h))$$

Variación de tanh de acuerdo a beta



## Con respecto a la inicialización de pesos

Inicializar los pesos en cero  
(ejercicio: ¿qué desventaja tiene arrancar los pesos en cero?)

Inicializar los pesos con valores aleatorios de una distribución uniforme/gaussiana

## [What are the best weight initialization techniques for deep neural networks?](#)

All / Engineering / Machine Learning

## What are the best weight initialization techniques for deep neural networks?

Powered by AI and the LinkedIn community

- 1 [Random initialization](#)
- 2 [Xavier initialization](#)
- 3 [He initialization](#)
- 4 [Orthogonal initialization](#)
- 5 [Sparse initialization](#)
- 6 [Here's what else to consider](#)

# BIBLIOGRAFÍA

Rodrigo Ramele (2024) *Reglamento y Apuntes de Sistemas de Inteligencia Artificial*, Capítulo 6.

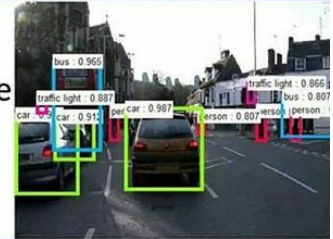
Bernard Widrow and Marcian E. Hoff, *Adaptive switching circuits*, 1960 IRE WESCON Convention Record, New York: IRE, pp. 96-104

## PARA LA PRÓXIMA CLASE

1. [What is a Neural Network?](#)
2. [Gradient Descent](#)
3. [What is backpropagation?](#)
4. [Backpropagation](#)

## Online Courses

What they promise  
you will learn



What you actually  
learn

