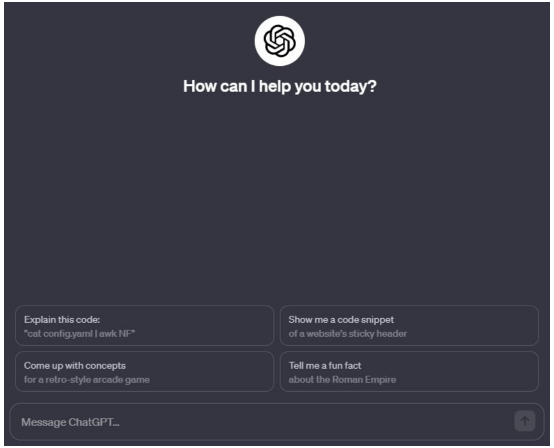
Deep Learning 101

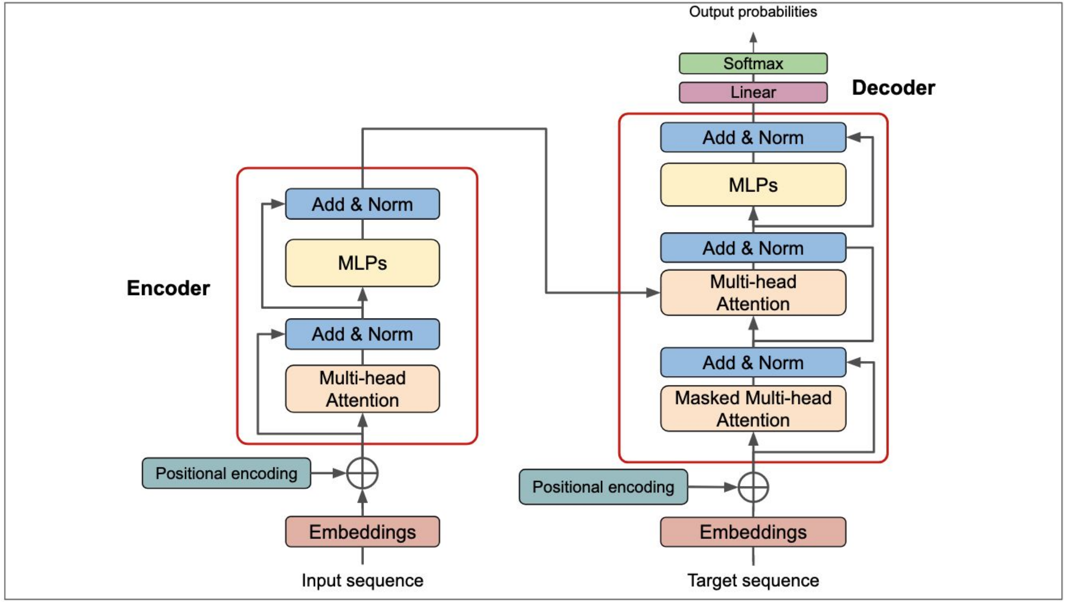
Capítulo 1: Neurona

Overview



ChatGPT

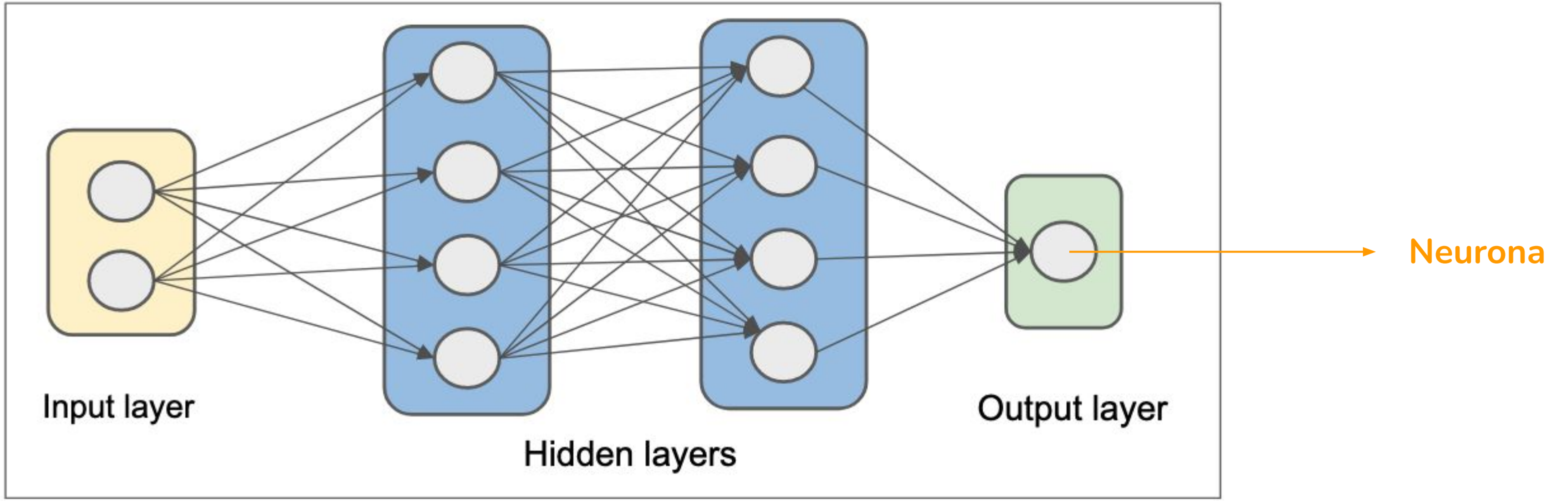
Modelo “Transformer” (2017)



https://deeprevision.github.io/posts/001-transformer/

Overview

MLP



https://deeprevision.github.io/posts/001-transformer/

Agenda

- Problemas que puede resolver una neurona.
- Modelo de neurona: perceptrón.
- Entrenamiento: qué significa que la neurona “aprenda”.
- Implementación desde cero, sin librerías.

Problema

Clasificación de especies de flores.

Longitud del Sépalo (cm)	Ancho del Sépalo (cm)	Es Iris Setosa
5.1	3.5	1
4.9	3.0	1
6.2	5.4	-1
x_1	x_2	y

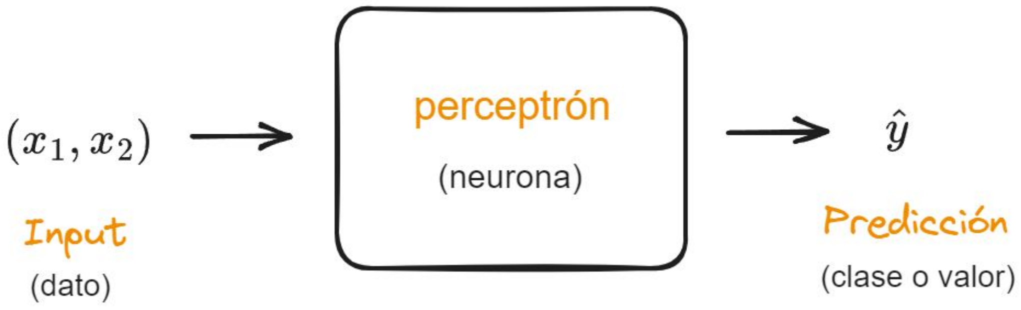
Problema

Estimación del precio de una casa.

Tamaño de la Casa (m^2)	Precio de la casa (miles de usd)
95	100
134	150
281	375
x_1	y

Modelo de neurona: Perceptrón

Su función es hacer predicciones sobre datos de entrada.



Conjunto de entrenamiento

Se otorgan al perceptrón inputs *(x1, x2)* con sus respectivos outputs "*y*" esperados.

x_1	x_2	y
1	1	1
2	2	1
1	2	-1

Predicciones

- El perceptrón calculará una salida (predicción) para cada input.
- Su objetivo será ajustar las predicciones para que coincidan con los datos de entrenamiento.

x_1	x_2	y
1	1	1
2	2	1
1	2	-1

Datos de entrenamiento

x_1	x_2	$\hat{y}$
1	1	-1
2	2	1
1	2	1

Predicciones

Al inicio, las predicciones serán aleatorias.

Perceptrón

- Datos de entrenamiento: $x = (x_1, x_2, \dots, x_m)$
- Pesos: $w = (w_1, w_2, \dots, w_m)$

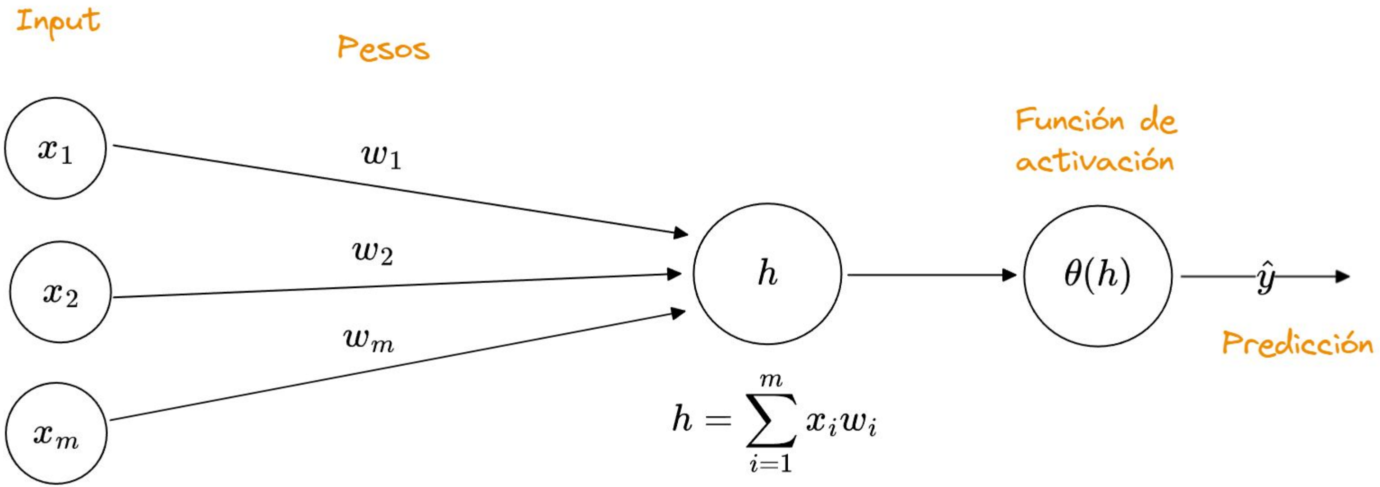
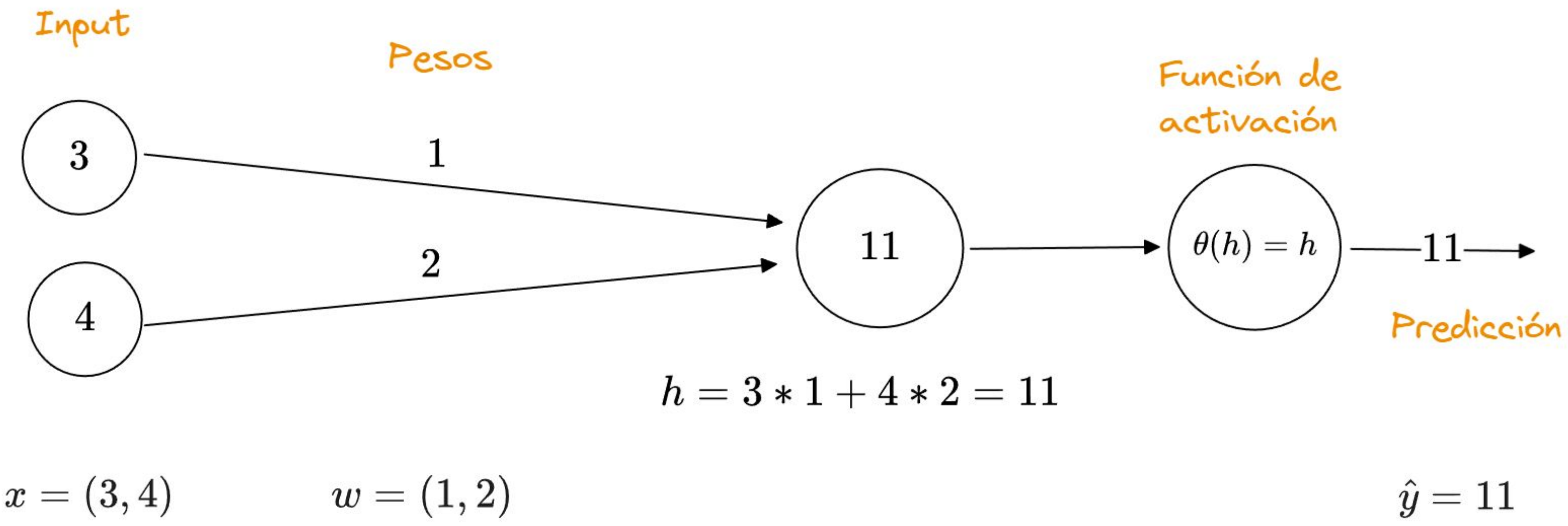
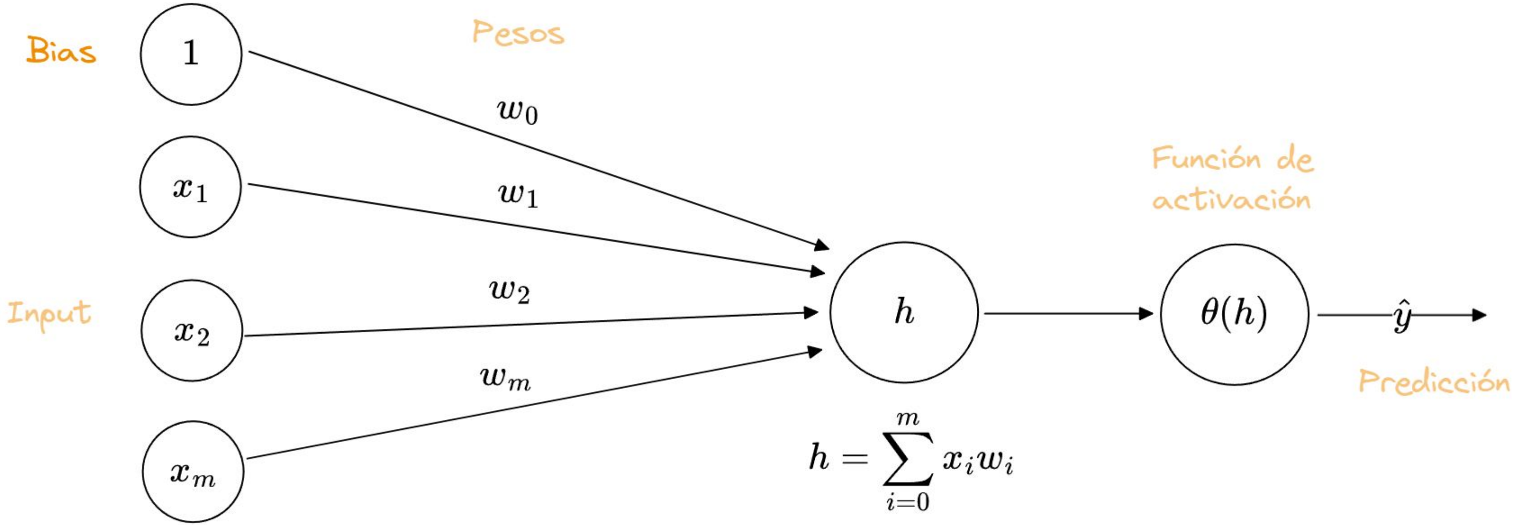


Gráfico por x.com/lvvittor

Perceptrón: ejemplo



Perceptrón: bias

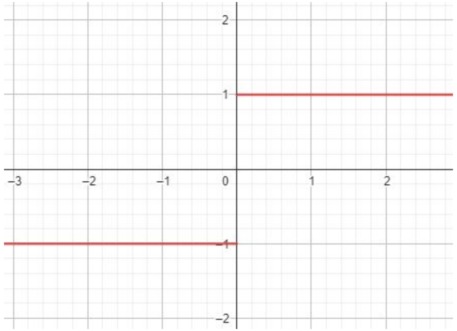


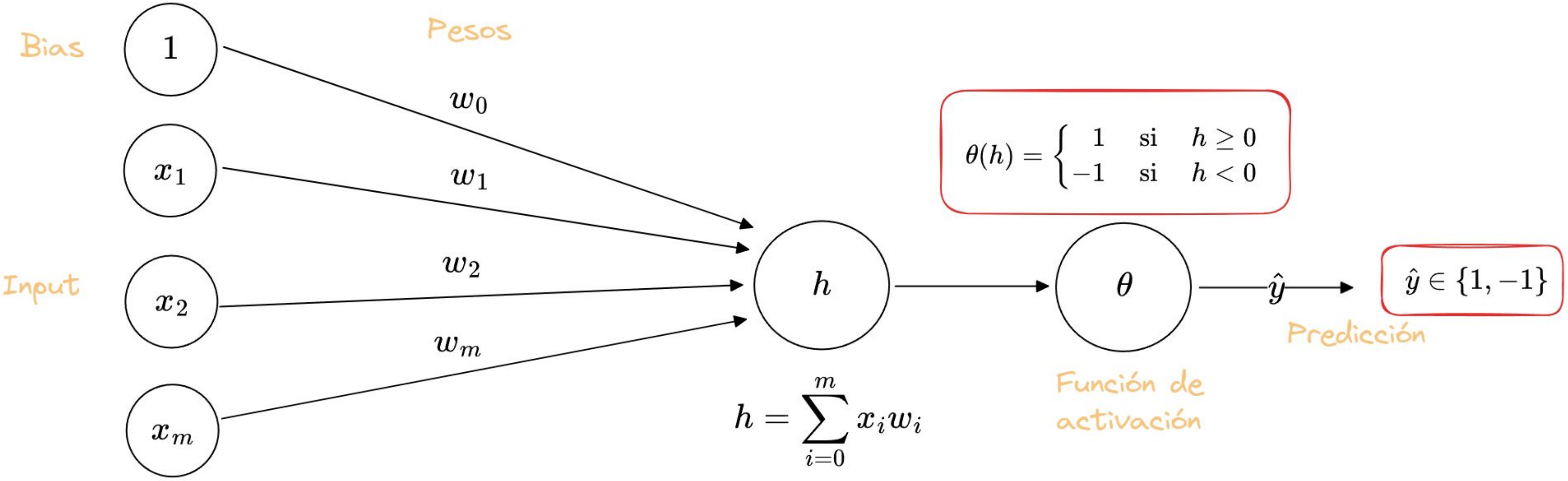
$$x = (1, x_1, x_2, \ldots, x_m) \qquad w = (w_0, w_1, w_2, \ldots, w_m)$$

Tipos de perceptrones

Según su función de activación

Perceptrón escalón



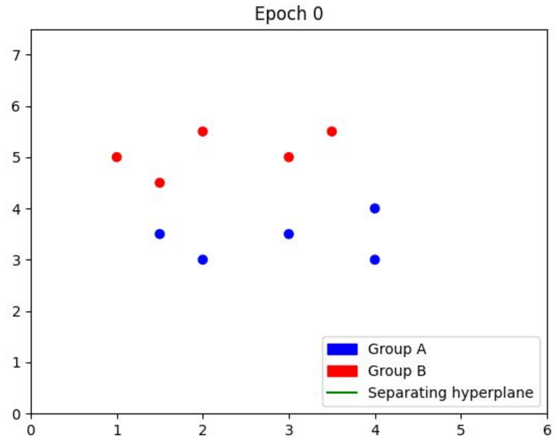


Perceptrón escalón: usos

- Clasificación binaria.

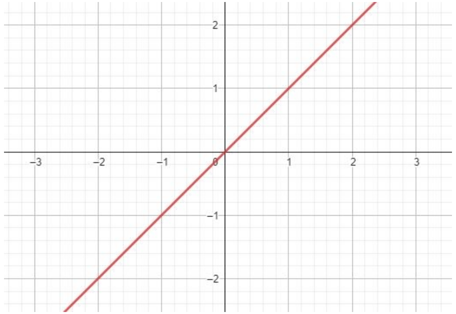
Input $X = (x_1, x_2) = (x, y)$

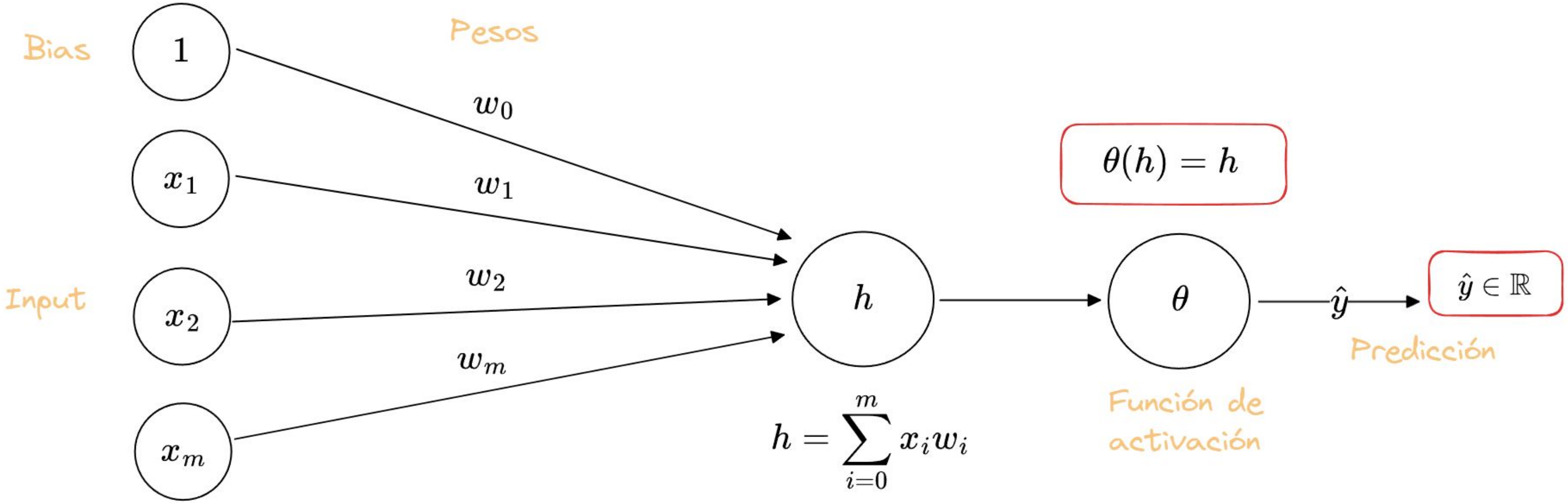
Predicción
$\hat{Y} = 1 =$
$\hat{Y} = -1 =$



Los grupos deben ser linealmente separables.

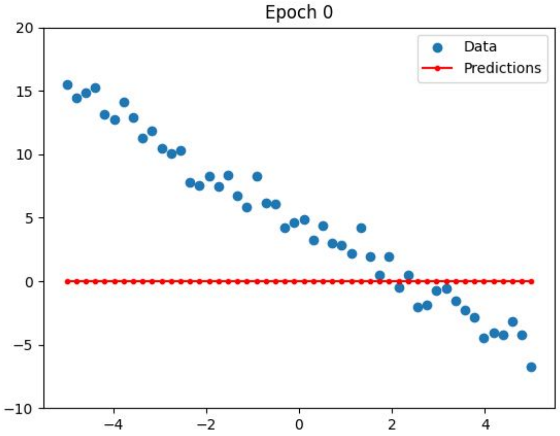
Perceptrón lineal



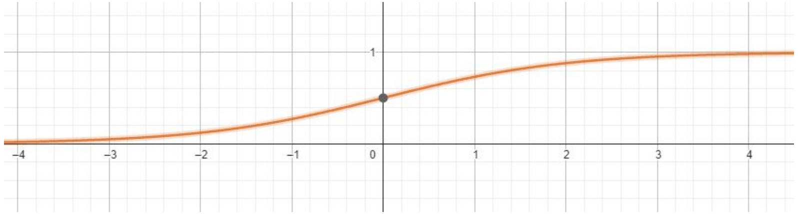


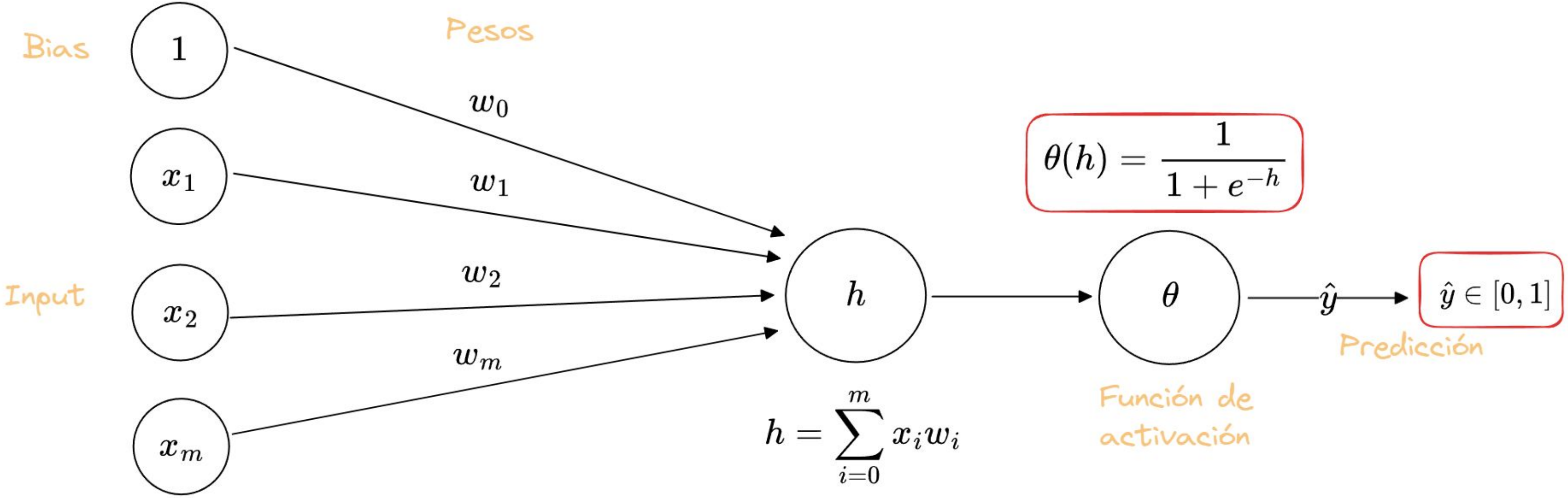
Perceptrón lineal: usos

- Regresión (lineal).



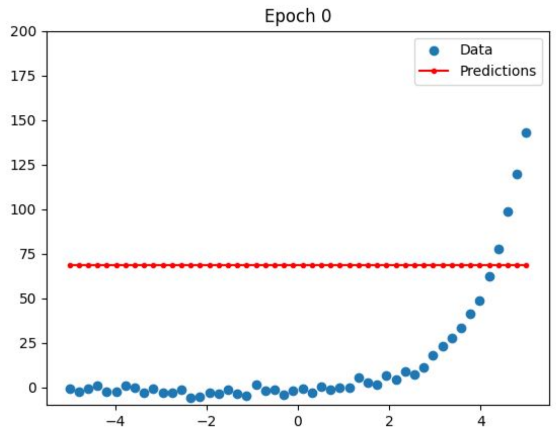
Perceptrón no lineal





Perceptrón no lineal: usos

- Clasificación binaria.
- Regresión (lineal y no lineal).



Aprendizaje y entrenamiento

Problema

Estimación del precio de una casa.

Al inicio, las predicciones
serán aleatorias.

Tamaño de la Casa (m^2)	Precio de la casa (miles de usd)	Predicción del precio
95	100	70
134	150	200
281	375	300
x_1	y	$\hat{y}$

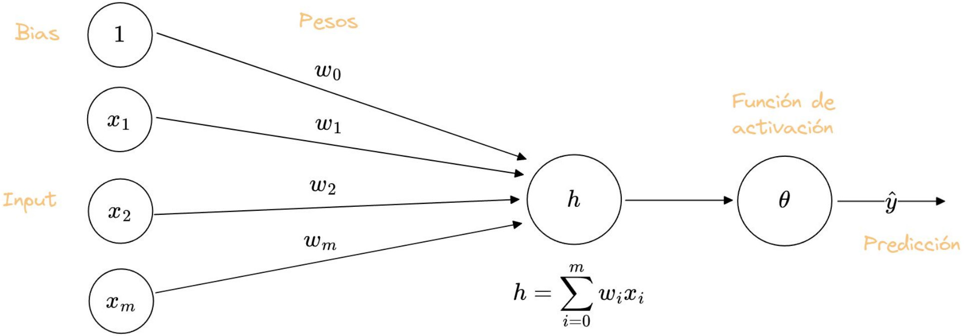
Función de costo

Se utiliza para medir qué tan mal está la predicción actual de la neurona, en comparación con la salida esperada.

Error cuadrático medio (MSE)

$$J = \frac{1}{n}\sum_{i=1}^{n} \overset{L}{\boxed{(y^i - \hat{y}^i)^2}}$$

Aprendizaje



Cómo ajustamos las predicciones de la neurona para alcanzar las salidas esperadas?

$$w_i = (w_0, w_1, \ldots, w_m)$$

$$\Delta w = (\Delta w_0, \Delta w_1, \ldots, \Delta w_m)$$

$$w_{i+1} = w_i + \Delta w$$ → **"aprendizaje"**

$$w_i = (w_0, w_1, \ldots, w_m)$$

$$\Delta w = (\Delta w_0, \Delta w_1, \ldots, \Delta w_m)$$

$$w_{i+1} = w_i + \Delta w$$

Actualización de pesos

Para el **perceptrón escalón**, la actualización de pesos estará dada por:

$$\Delta w = \eta(y - \hat{y})x$$

$$\eta \in (0, 1)$$

Tasa de aprendizaje

Actualización de pesos

$$w_i = (w_0, w_1, \ldots, w_m)$$

$$\Delta w = (\Delta w_0, \Delta w_1, \ldots, \Delta w_m)$$

$$w_{i+1} = w_i + \Delta w$$

Ejemplo:

$$\Delta w = \eta(y - \hat{y})x$$

$$\eta = 0.1$$
$$y = 1$$
$$\hat{y} = -1$$
$$x = (3, 4)$$
$$w_i = (1, 2)$$

$$\Delta w = 0.1(1 - (-1))(3, 4)$$
$$\Delta w = 0.2(3, 4)$$
$$\Delta w = (0.6, 0.8)$$

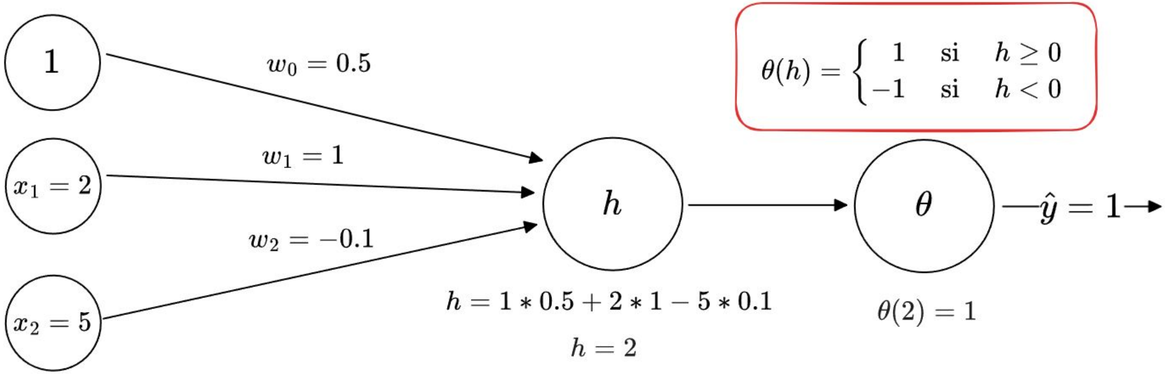
$$w_{i+1} = (1.6, 2.8)$$

$$\Delta w = \eta(y - \hat{y})x$$

Actualización de pesos: ejemplo 1

x	y
(2, 5)	-1

$$w = [0.5, 1, -0.1] \qquad \eta = 0.1$$



$$\Delta w = 0.1(-1 - 1)[1, 2, 5]$$

$$\Delta w = -0.2[1, 2, 5] = [-0.2, -0.4, -1]$$

$$w' = [0.5, 1, -0.1] + [-0.2, -0.4, -1]$$

$$w' = [0.3, 0.6, -1.1]$$

$$h' = 1 * 0.3 + 2 * 0.6 - 5 * 1.1$$

$$h' = -4$$

$$\theta(-4) = -1$$

$$\hat{y} = -1 = y$$

$$w_i = (w_0, w_1, \ldots, w_m)$$

$$\Delta w = (\Delta w_0, \Delta w_1, \ldots, \Delta w_m)$$

$$w_{i+1} = w_i + \Delta w$$

Actualización de pesos

Para el **perceptrón lineal y no lineal**, la actualización de pesos estará dada por:

$$J = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\boxed{(y^i - \hat{y}^i)^2}}_{L}$$

Error cuadrático medio (MSE)

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

Gradiente descendente

$$\theta(h) = \hat{y}$$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \theta}\frac{\partial \theta}{\partial h}\frac{\partial h}{\partial w}$$

Regla de la cadena

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \theta}\frac{\partial \theta}{\partial h}\frac{\partial h}{\partial w}$$

Regla de la cadena

Gradiente: ejemplo

$$\theta(h) = \hat{y}$$

$$L = (y - \hat{y})^2 \longrightarrow \frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \hat{y}} = 2(y - \hat{y})(-1) = 2(\hat{y} - y)$$

Error cuadrático

$$\theta(h) = h \longrightarrow \frac{\partial \theta}{\partial h} = 1$$

Función identidad

$$h = \sum_{i=0}^{m} w_i x_i \longrightarrow \frac{\partial h}{\partial w} = (x_0, x_1, \ldots, x_m)$$

Gradiente

$$\boxed{\frac{\partial L}{\partial w} = 2(\hat{y} - y)(x_0, \ldots, x_m)}$$

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

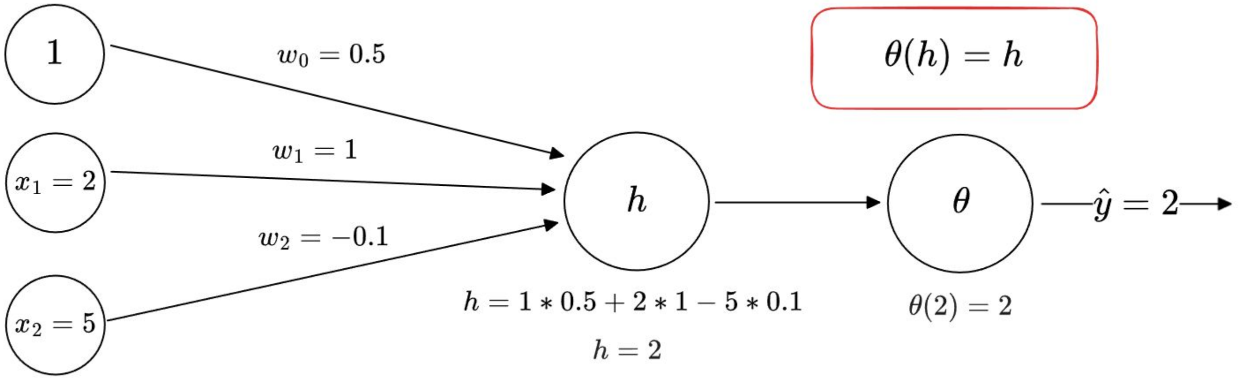
$$w_{i+1} = w_i + \Delta w$$

$$\Delta w = -\eta \frac{\partial L(w)}{\partial w}$$

Actualización de pesos: ejemplo 2

x	y
(2, 5)	-1

$w = [0.5, 1, -0.1] \quad \eta = 0.02 \quad J = MSE$



$$\boxed{\frac{\partial L}{\partial w} = 2(\hat{y} - y)(x_0, \ldots, x_m)}$$

$\Delta w = -0.02\ 2(2 - (-1))\ (1, 2, 5)$

$\Delta w = -0.12(1, 2, 5) = -(0.12, 0.24, 0.6)$

$w' = [0.5, 1, -0.1] - [0.12, 0.24, 0.6]$

$w' = [0.38, 0.76, -0.7]$

$h' = 1 * 0.38 + 2 * 0.76 - 5 * 0.7$

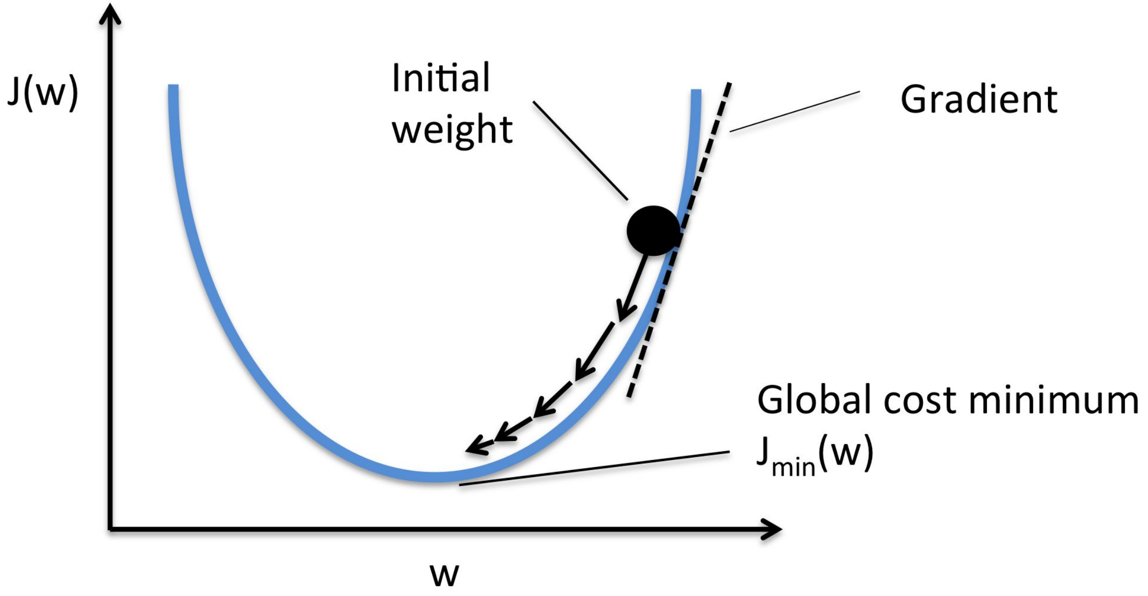
$h' = -1.6$

$\theta(-1.6) = -1.6$

$\hat{y} = -1.6$

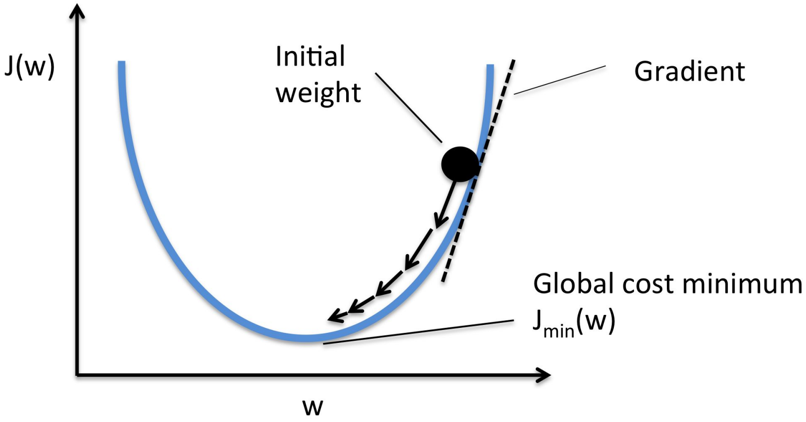
Bonus: Optimización

Optimización de pérdida



https://sebastianraschka.com/faq/docs/gradient-optimization.html

GD con Momentum



$$\boxed{\Delta w_{i+1} = -\eta \frac{\partial L}{\partial w} + \alpha \Delta w_i}$$

$$\alpha \in (0, 1)$$

https://sebastianraschka.com/faq/docs/gradient-optimization.html

Fin