Machine Learning en Raspberry Pi

Introducción a las redes neuronales convolucionales

Lic. Alexis Luszczak
Lic. Lucas Finazzi

¿Qué es Machine Learning?

"La ciencia de lograr que las computadoras actúen sin ser programadas explícitamente"[1]

[1] Brain Tumor MRI Image Segmentation Using Deep Learning Techniques, 2022.

¿Qué es Machine Learning?

"La ciencia de lograr que las computadoras actúen sin ser programadas explícitamente"[1]

“Técnicas para identificar relaciones entre datos basadas en algoritmos libres de modelo que dada una serie de parámetros libres que estructuran esas relaciones, pueden ajustarse mediante algún proceso de optimización matemático”

[1] Brain Tumor MRI Image Segmentation Using Deep Learning Techniques, 2022.

Terminología

- Features (Características): Son las variables o atributos que se utilizan como entradas en un modelo de machine learning.
- Target (Objetivo): Se refiere a la variable que se intenta predecir en un modelo. Es el resultado o la salida que queremos obtener a partir de las características.
- Labels (Etiquetas): Es resultados que el modelo intenta predecir. Cada entrada puede tener una etiqueta asociada.

Terminología

Ej. Quiero predecir el género de una persona con el siguiente dataset.

Target Feature

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

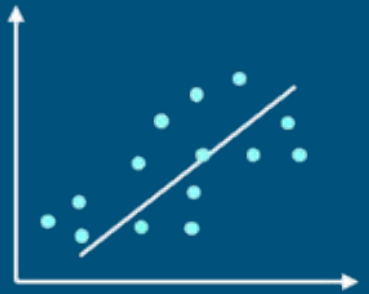
Label

Tipos de Modelos de Machine learning

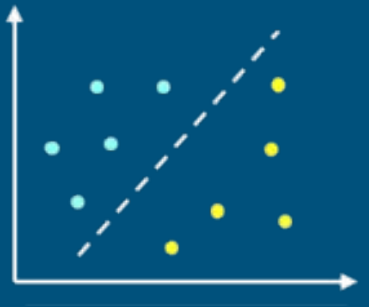
Tipos de Modelos de Machine learning

Supervisado

Regresión: Predecir un valor continuo.
Ejemplo: estimar el precio de una casa en función de sus características.



Clasificación: Predecir una categoría.
Ejemplo: determinar si un correo electrónico es spam o no.

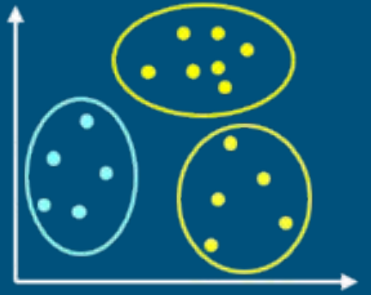


Tipos de Modelos de Machine learning

No Supervisado

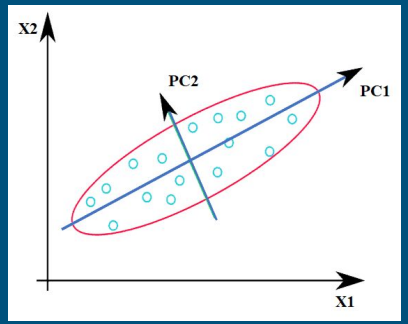
Clustering: Agrupar datos similares.

Ejemplo: segmentar clientes en grupos basados en sus comportamientos de compra.



Reducción de Dimensionalidad: Simplificar los datos sin perder información importante.

Ejemplo: usar PCA (Análisis de Componentes Principales) para visualizar datos de alta dimensión.



Conceptos clave

Conjunto de Entrenamiento (Train Set):

- Usado para entrenar el modelo.
- Permite que el modelo aprenda patrones.

Conjunto de Prueba (Test Set):

- Usado para evaluar el rendimiento del modelo.
- No se utiliza durante el entrenamiento, lo que permite una evaluación objetiva.

Importancia

- **Previene el Sobreajuste**: Asegura que el modelo no solo memorice los datos de entrenamiento.
- **Evaluación Realista**: Mide cómo se comportará el modelo en datos nuevos.

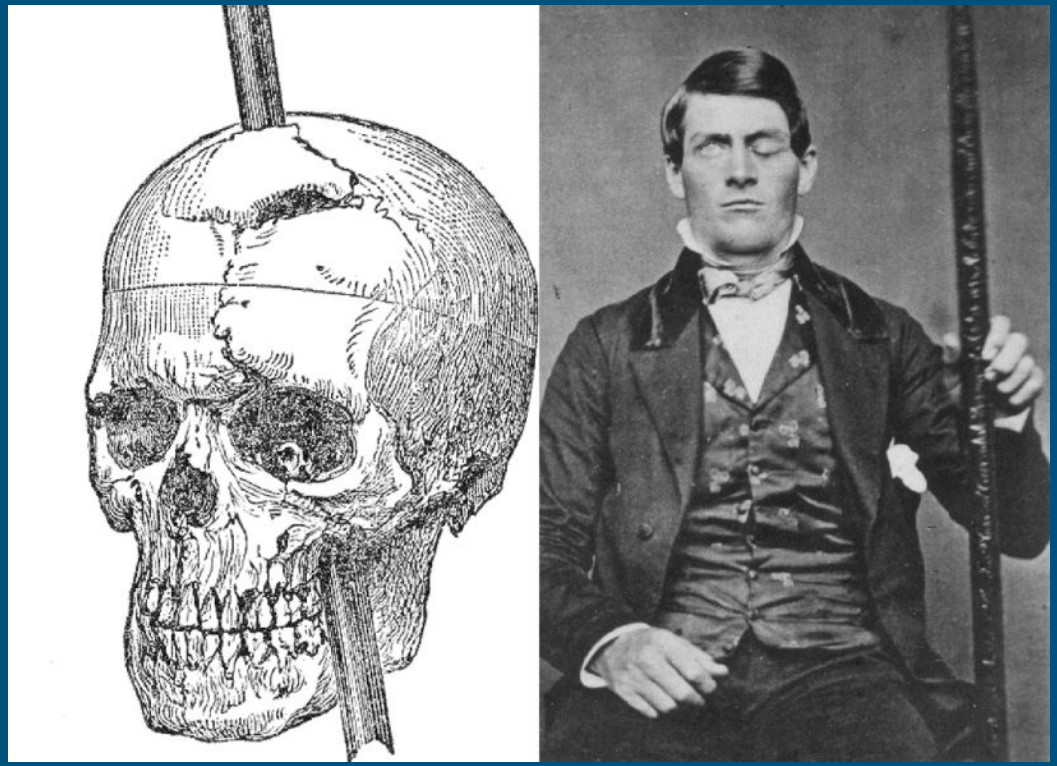
Conceptos clave

- **Parámetro: Es un valor interno que el modelo aprende durante el entrenamiento. Por ejemplo, los pesos y bias de una red neuronal.**

- **Hiper Parámetro: Es un valor que se configura antes de entrenar el modelo y que no se aprende. Por ejemplo la tasa de aprendizaje, número de epoch, tamaño del batch, número de capas, etc.**

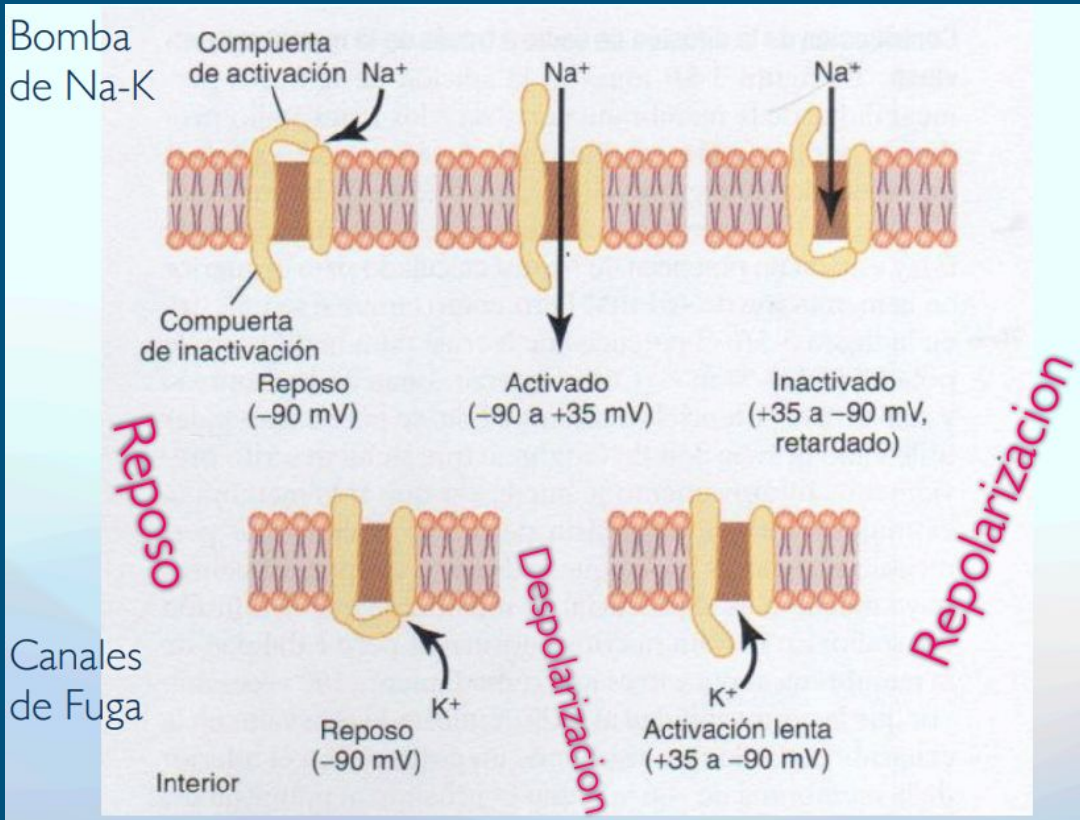
Redes neuronales

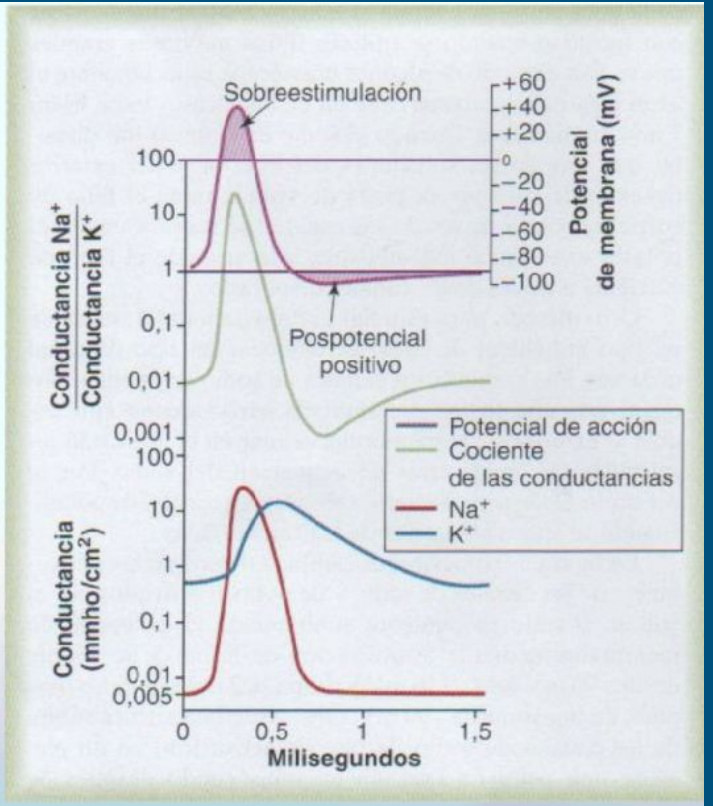
Phineas Gage (1848)



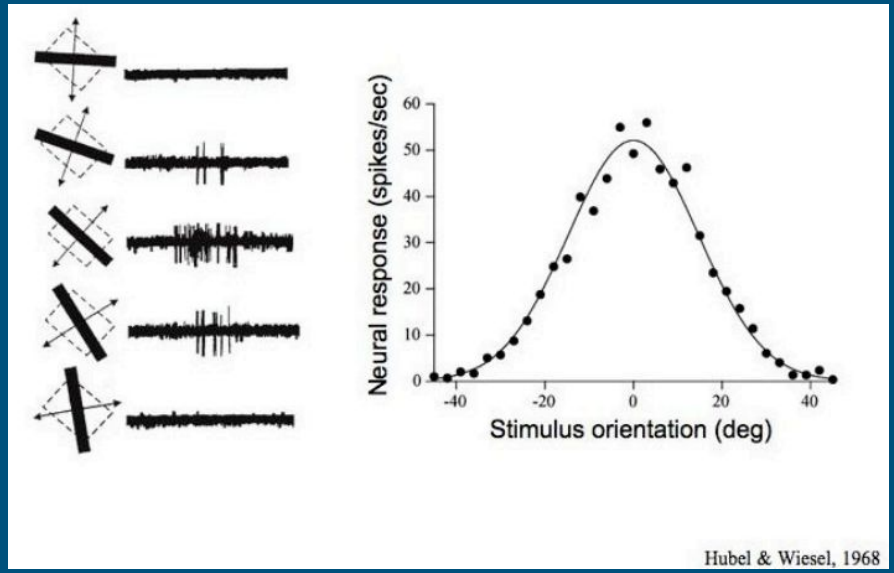
https://slate.com/technology/2014/05/phineas-gage-neuroscience-case-true-story-of-famous-frontal-lobe-patient-is-better-than-textbook-accounts.html

Acción potencial



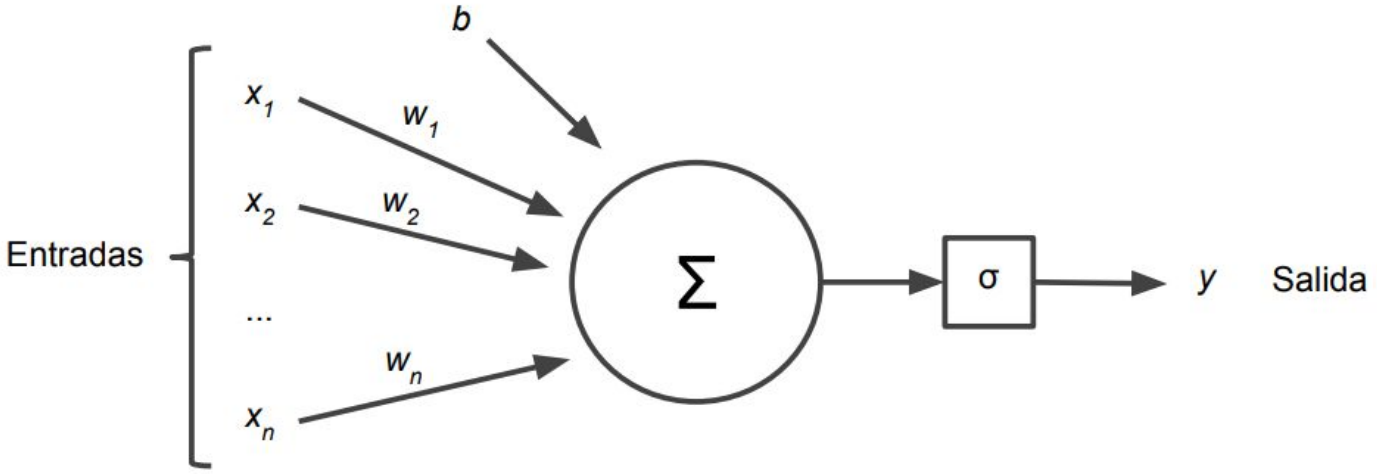


Los gatos de Hubel & Wiesel



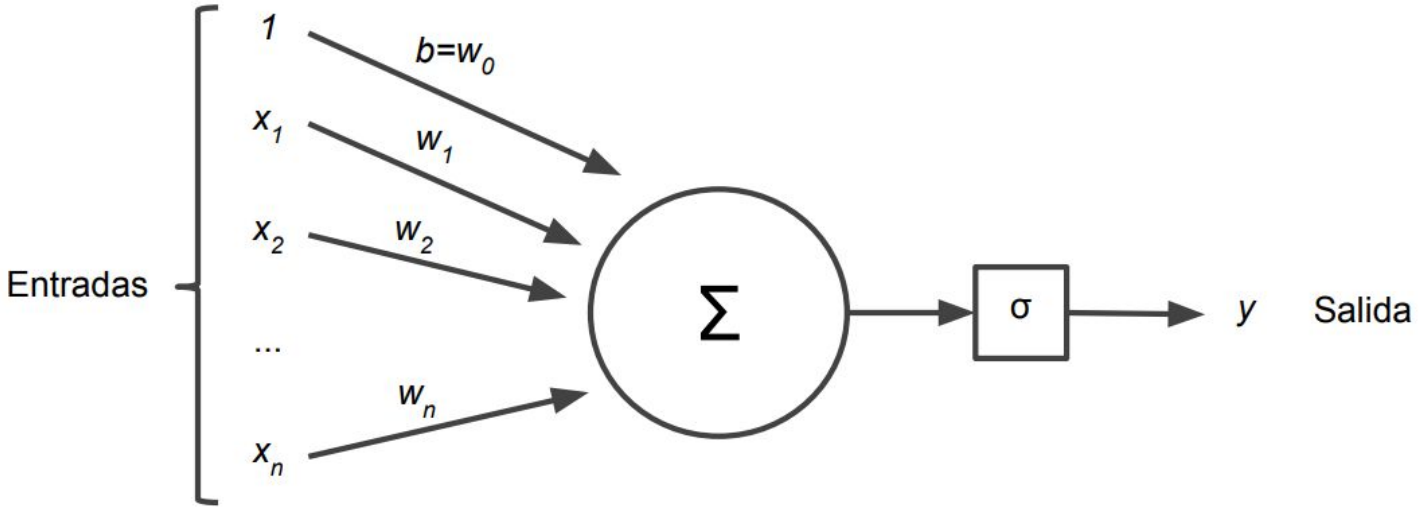
La neurona se activa dependiendo la orientación del objeto

Mc Culloch & Pitts - Modelo de neurona



$$y = \sigma(\sum_i w_i x_i + b)$$

Mc Culloch & Pitts - Modelo de neurona



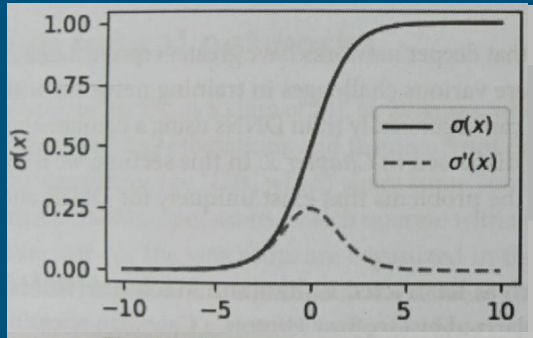
$$\hat{x} = [1, x_1, x_2, ..., x_n]$$
$$\hat{w} = [w_0, w_1, w_2, ..., w_n]$$

$$y = \sigma(\hat{w}.\hat{x})$$

Función de activación

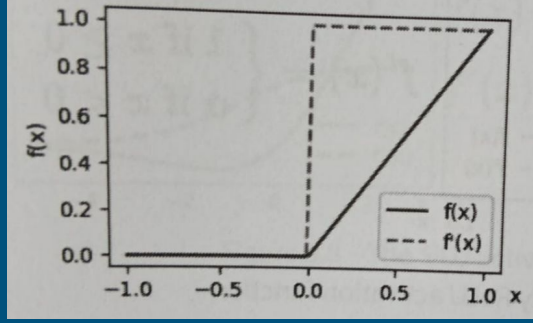
Función preferentemente derivable

Función logística



$$\sigma(z) = \frac{1}{1+e^{-z}}$$

ReLu



$$f(x) = \begin{cases} x \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

$$f'(x) = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ if } x < 0 \end{cases}$$

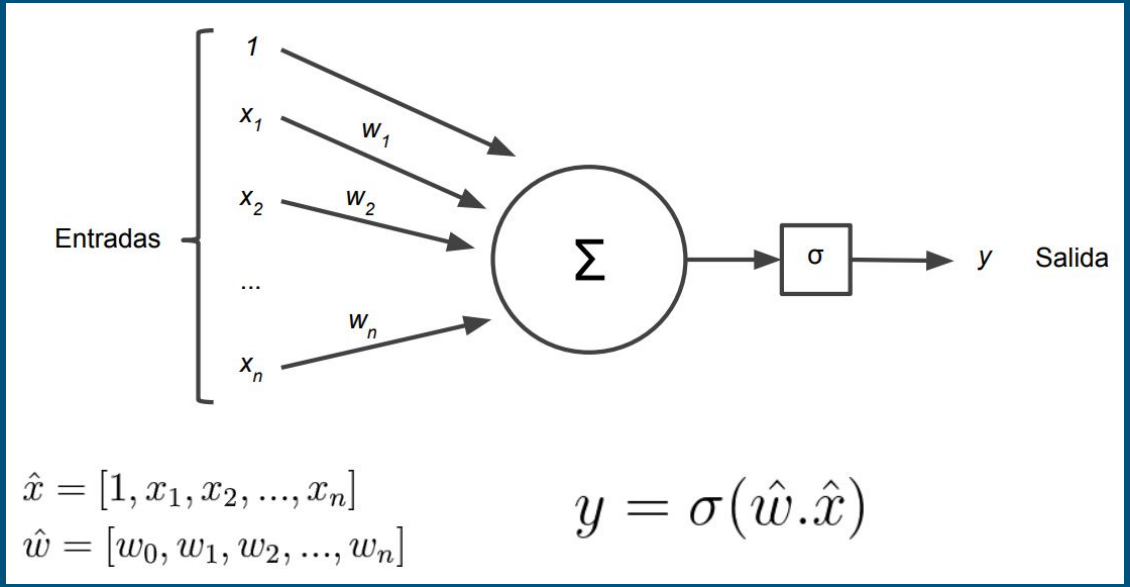
Ejemplo: Análisis de Sentimiento

Sea un conjunto de comentarios (reviews) de películas, cada uno categorizado como positivo o negativo (0 o 1)

Review	Clasificación
Buenísima. Entretenida y bien lograda.	1
Escenas traídas de los pelos. No la recomiendo.	0
No me gustó la película.	0
Horrible! Me aburrí como un hongo.	0
Muy buena la película. La super recomiendo.	1
Muy linda película.	1
Me gustó! La recomiendo totalmente.	1
No la recomiendo. Es un divague.	0
Una historia que es un mamarracho.	0

Ejemplo: Análisis de Sentimiento

Las entradas de mi red son números, y la salida también ¿Cómo hago para representar mi texto mediante números?



$$\hat{x} = [1, x_1, x_2, ..., x_n]$$
$$\hat{w} = [w_0, w_1, w_2, ..., w_n]$$

$$y = \sigma(\hat{w}.\hat{x})$$

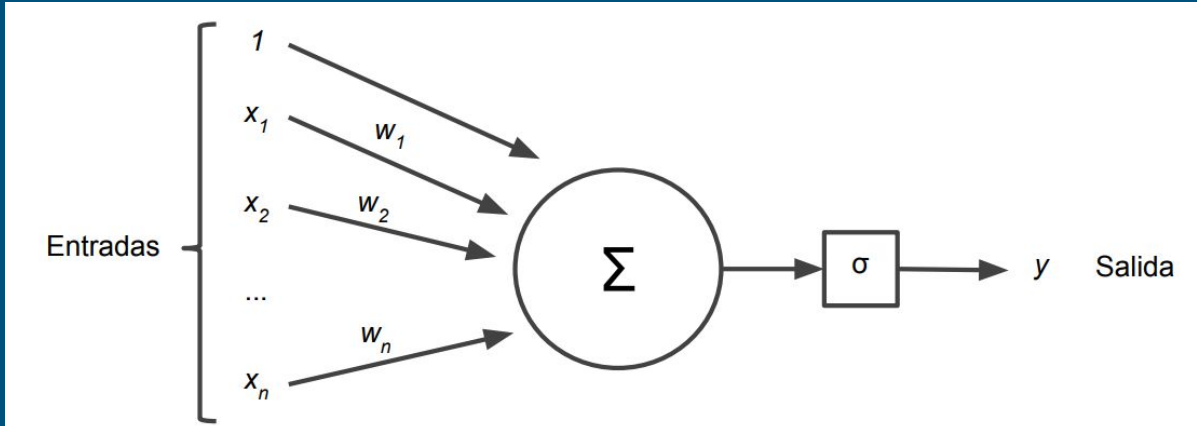
Ejemplo: Análisis de Sentimiento

Las entradas de mi red son números, y la salida también ¿Cómo hago para representar mi texto mediante números?

Primera idea: Extracción de features (manuales)
Lista de palabras positivas P={buenísima, buena, gustó, linda, recomiendo}
Lista de palabras negativas N={horrible, divague, mamarracho}

x1 = cantidad de palabras de la lista P en el texto
x2 = cantidad de palabras de la lista N en el texto
x3 = la palabra “no” aparece en el texto
x4 = el signo de exclamación “!” aparece en el texto
x5 = cantidad de palabras en el texto

Ejemplo: Análisis de Sentimiento



$$\hat{x} = [1, x_1, x_2, ..., x_n]$$
$$\hat{w} = [w_0, w_1, w_2, ..., w_n]$$
$$y = \sigma(\hat{w}.\hat{x})$$

Por ejemplo, eligiendo σ como la sigmoide tenemos que este clasificador es exactamente una regresión logística.

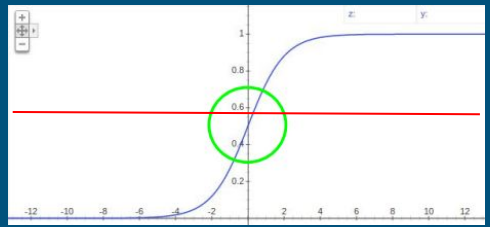
Ejemplo: Análisis de Sentimiento

Escenas traídas de los pelos. No la recomiendo.

P={buenísima, buena, gustó, linda, recomiendo}
N={horrible, divague, mamarracho}

x1 = cantidad de palabras de la lista P en el texto = 1
x2 = cantidad de palabras de la lista N en el texto = 0
x3 = la palabra “no” aparece en el texto = 1
x4 = el signo de exclamación “!” aparece en el texto = 0
x5 = cantidad de palabras en el texto = 8
w = [0.5, 0.6, -0.8, 0.3, -0.1]

$$\frac{1}{1+e^{-w\cdot x}} = ?$$



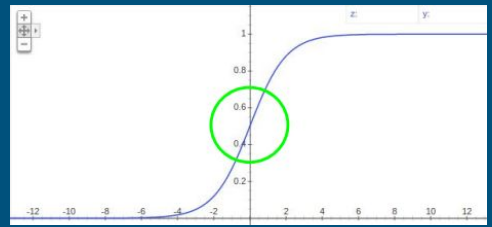
Ejemplo: Análisis de Sentimiento

Escenas traídas de los pelos. No la recomiendo.

P={buenísima, buena, gustó, linda, recomiendo}
N={horrible, divague, mamarracho}

x1 = cantidad de palabras de la lista P en el texto = 1
x2 = cantidad de palabras de la lista N en el texto = 0
x3 = la palabra “no” aparece en el texto = 1
x4 = el signo de exclamación “!” aparece en el texto = 0
x5 = cantidad de palabras en el texto = 8
w = [0.5, 0.6, -0.8, 0.3, -0.1]

$$\frac{1}{1+e^{-w\cdot x}} = 0.2497$$



Problema del XOR

Perceptron de Rosenblatt (1958) Una neurona como la que vimos, pero donde la función de activación es la función escalón

$$y = \begin{cases} 0, \text{si } w \cdot x + b \leq 0 \\ 1, \text{si } w \cdot x + b > 0 \end{cases}$$

¿Podemos modelar los operadores lógicos usando esta neurona?

AND			OR			XOR		
x1	x2	y	x1	x2	y	x1	x2	y
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

Problema del XOR

AND

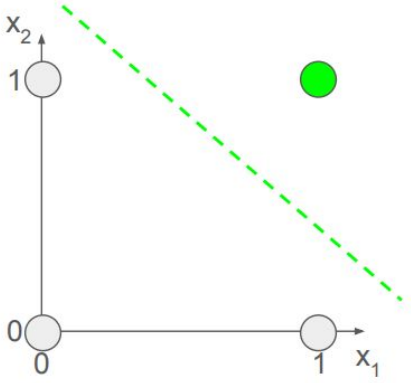
x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

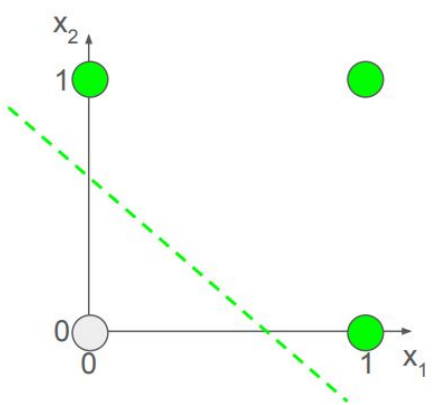
OR

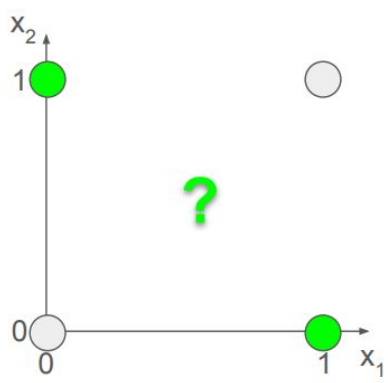
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

XOR

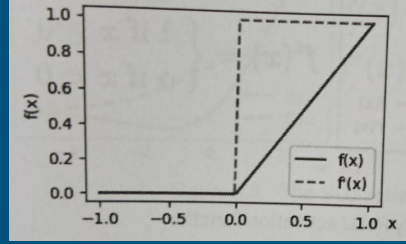
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0







Problema del XOR



$$f(x) = \begin{cases} x \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

$$f'(x) = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ if } x < 0 \end{cases}$$

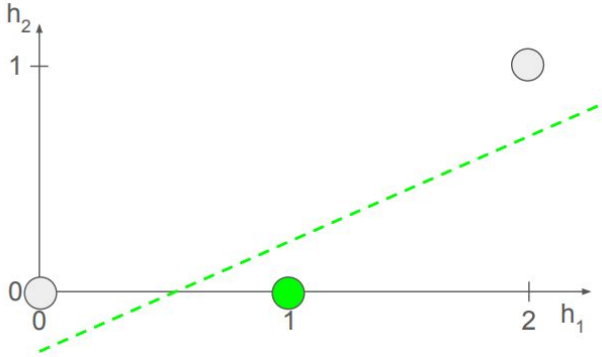
$h_1 = \text{ReLU}(x_1 + x_2)$

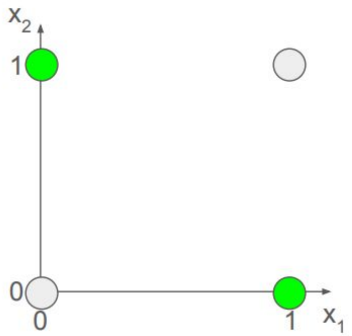
$h_2 = \text{ReLU}(x_1 + x_2 - 1)$

$y = h_1 - 2\, h_2$

XOR

x1	x2	h1	h2	y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	2	1	0





Problema del XOR

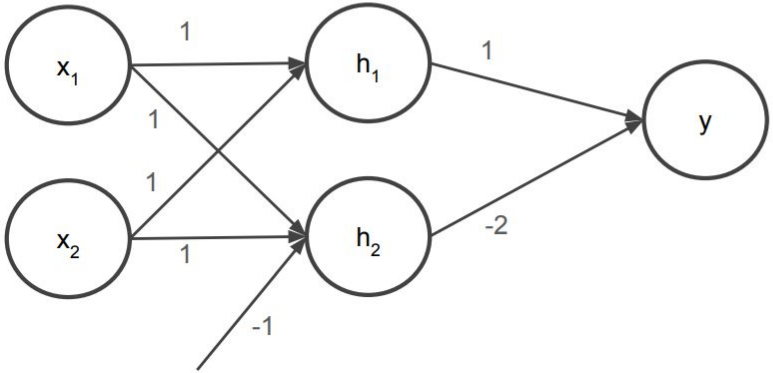
$h_1 = \text{ReLU}(x_1 + x_2)$

$h_2 = \text{ReLU}(x_1 + x_2 - 1)$

$y = h_1 - 2\, h_2$

XOR

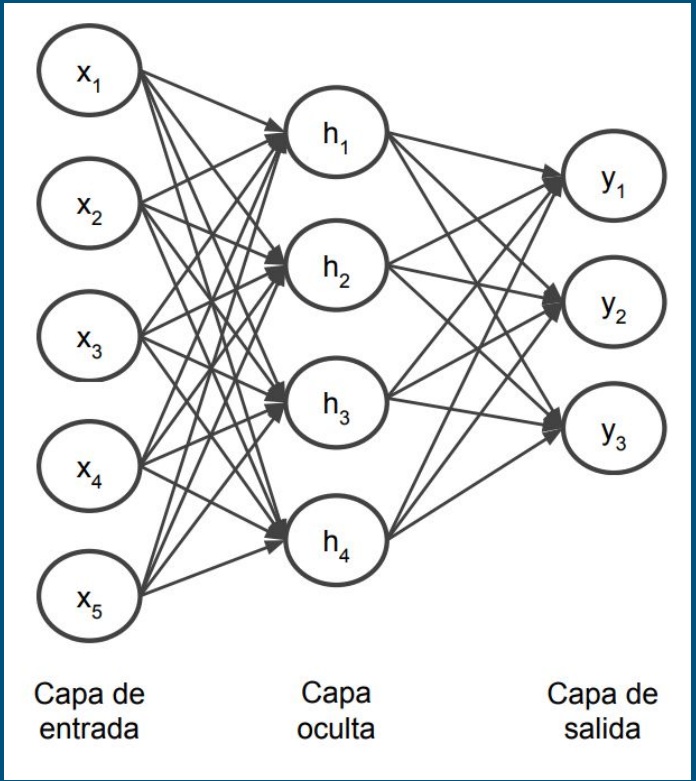
x1	x2	h1	h2	y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	2	1	0



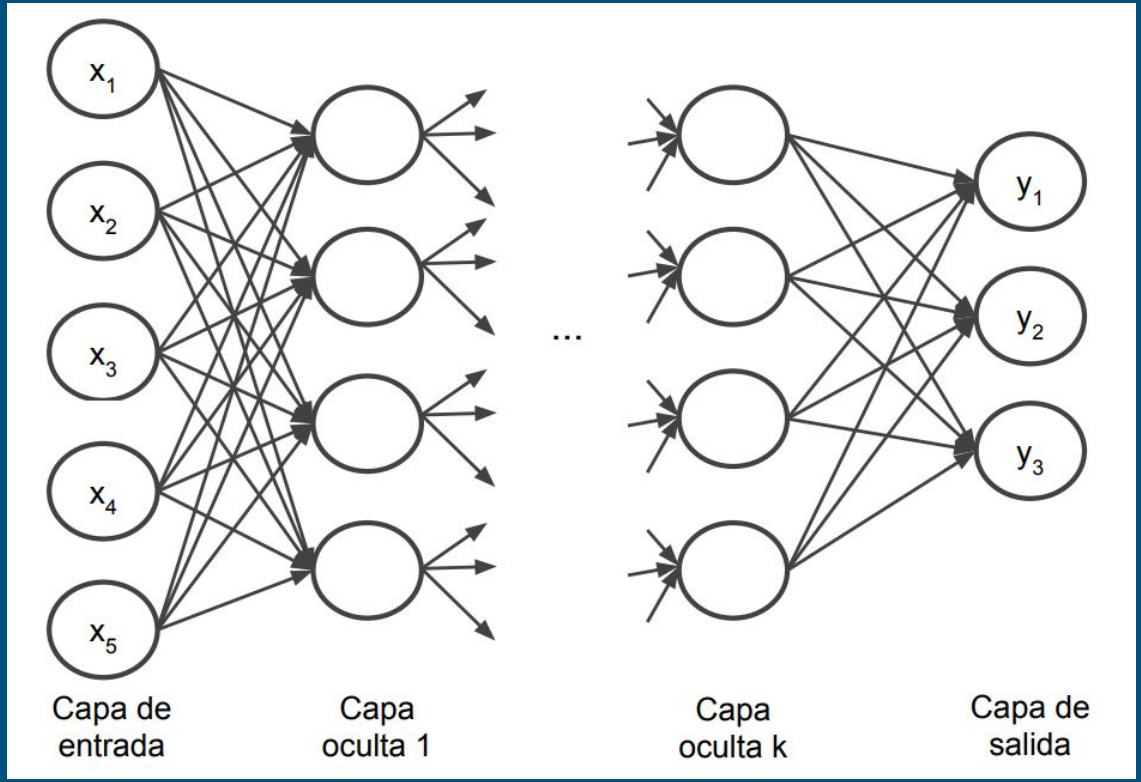
Capa oculta

Para poder modelar funciones más complejas necesitamos al menos una capa más que haga una transformación
Llamamos capa oculta (hidden layer) a la capa que está luego de la entrada pero antes de la capa de salida

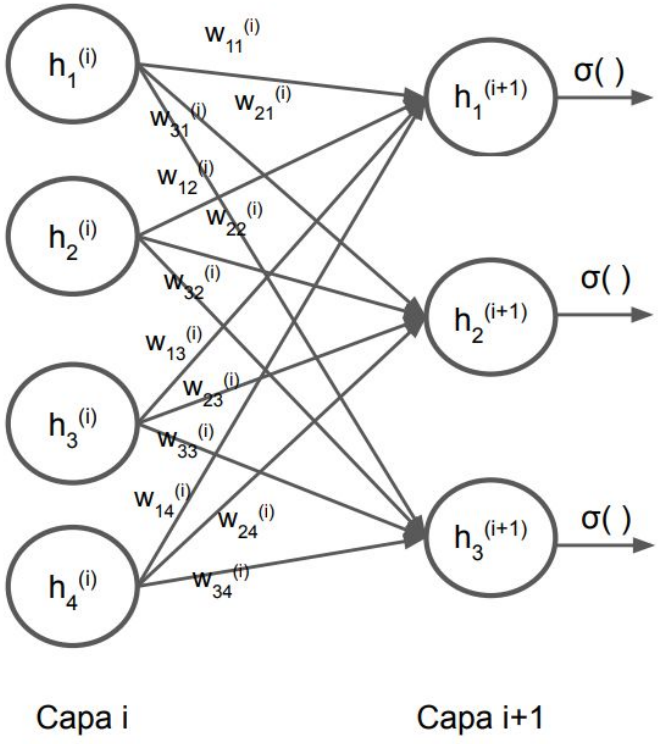
Teorema de la aproximación universal (Cybenko 1989) alcanza con una sola capa oculta para modelar cualquier función de un conjunto compacto de $R_n \rightarrow R_m$



Red Feedforward Completamente Conectada



Red Feedforward Completamente Conectada



Entrada: $h^{(i)} = [\, h_1^{(i)}, h_2^{(i)}, h_3^{(i)}, h_4^{(i)} \,]$

Salida: $h^{(i+1)} = [\, h_1^{(i+1)}, h_2^{(i+1)}, h_3^{(i+1)} \,]$

$$W^{(i)} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \end{bmatrix}$$

$$h^{(i+1)} = \sigma(W^{(i)} h^{(i)})$$

RED neuronal

Se debe elegir:

- Cantidad de capas
- Cantidad de unidades por capa
- Función de activación de cada capa

Clasificación supervisada

Tengo un conjunto de ejemplos, y el resultado para cada ejemplo

x (1) → y(1)
x (2) → y(2)
...
x (n) → y(n)

Quiero encontrar la red que prediga mejor el resultado de cada ejemplo ¿Cómo la construyo? ¿Y cómo la evalúo?

Clasificación supervisada

Queremos que luego de hacer una red (que es aplicar una función a los valores de entrada) la salida sea lo más parecido al valor que se que tiene que dar.

$x(1) \rightarrow y(1)$
$x(2) \rightarrow y(2)$
...
$x(n) \rightarrow y(n)$

$f(x(1)) \rightarrow \hat{y}(1)$
$f(x(2)) \rightarrow \hat{y}(2)$
...
$f(x(n)) \rightarrow \hat{y}(n)$

Clasificación supervisada

Queremos que luego de hacer una red (que es aplicar una función a los valores de entrada) la salida sea lo más parecido al valor que se que tiene que dar.

$x(1) \rightarrow y(1)$
$x(2) \rightarrow y(2)$
...
$x(n) \rightarrow y(n)$

$f(x(1)) \rightarrow \hat{y}(1)$
$f(x(2)) \rightarrow \hat{y}(2)$
...
$f(x(n)) \rightarrow \hat{y}(n)$

Para ello el algoritmo debe encontrar los mejores pesos!

Función de loss (pérdida)

Función que relaciona los ejemplos de entrenamiento $x^{(i)}$, sus resultados esperados $y^{(i)}$, y la salida de la red $\hat{y}^{(i)} = f(x^{(i)}; w)$

Existen varias funciones de loss, se elegirá una dependiendo del problema que se está modelando

Error cuadrático medio: $$L_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}^{(i)} - y^{(i)})^2$$

Entropía cruzada: $$L_{CE} = -\sum_{i=1}^{N} y^{(i)} log(\hat{y}_j^{(i)})$$

Otras…

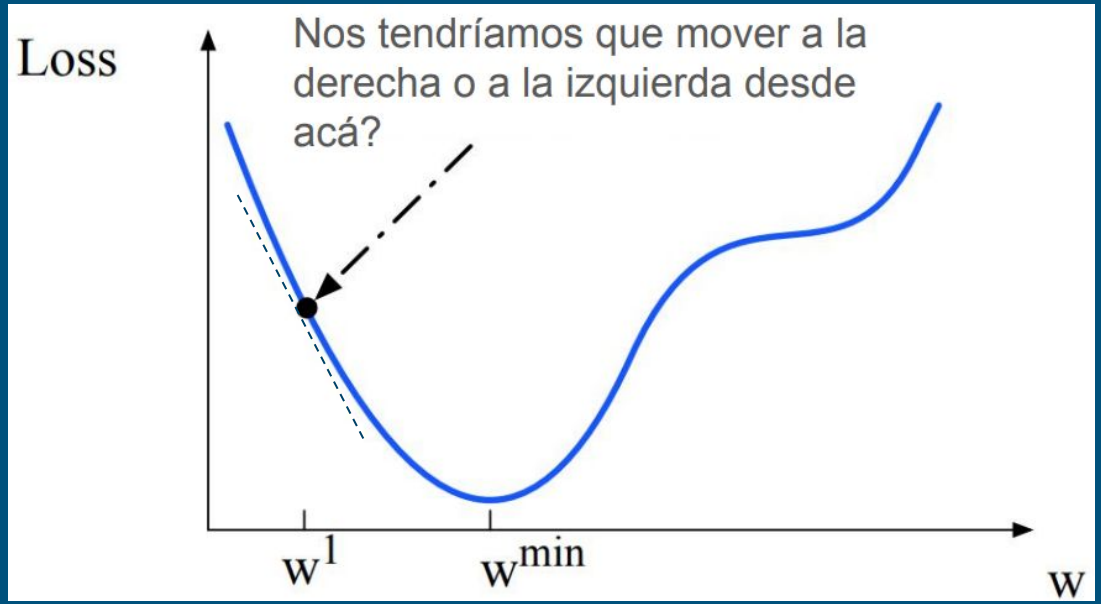
Función de loss (pérdida)

Relaciona los resultados esperados con los que obtiene la red

Cuanto más bajo es su valor, más parecidos son los valores obtenidos a los esperados

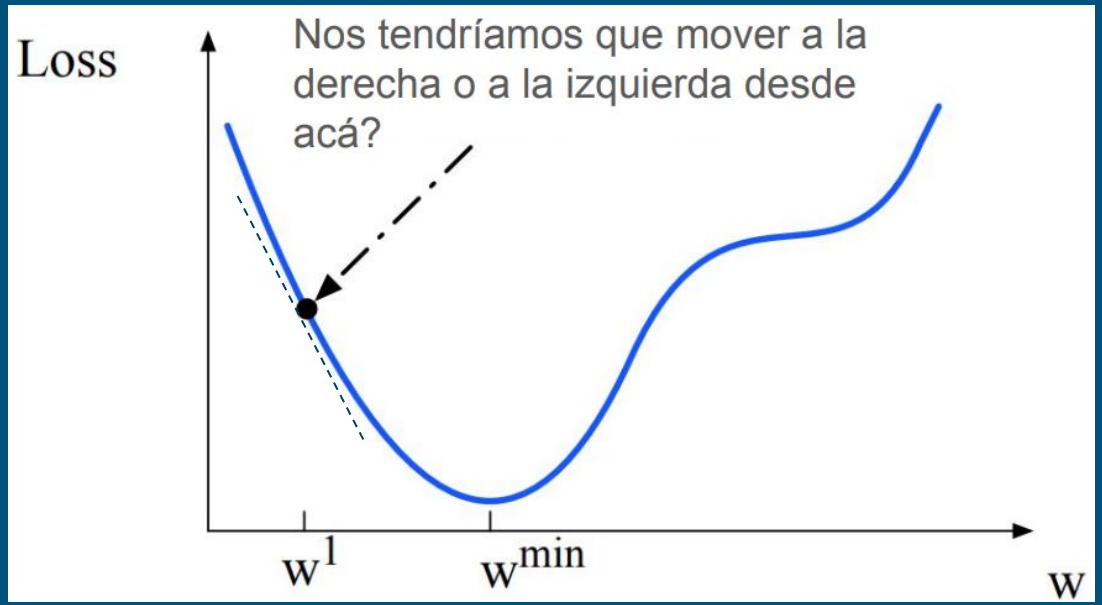
Tiene que ser derivable ya que varias técnicas usan la derivada para minimizar la función.
Por ejemplo: Gradient Descent Optimizador

Visualización en una sola dimensión



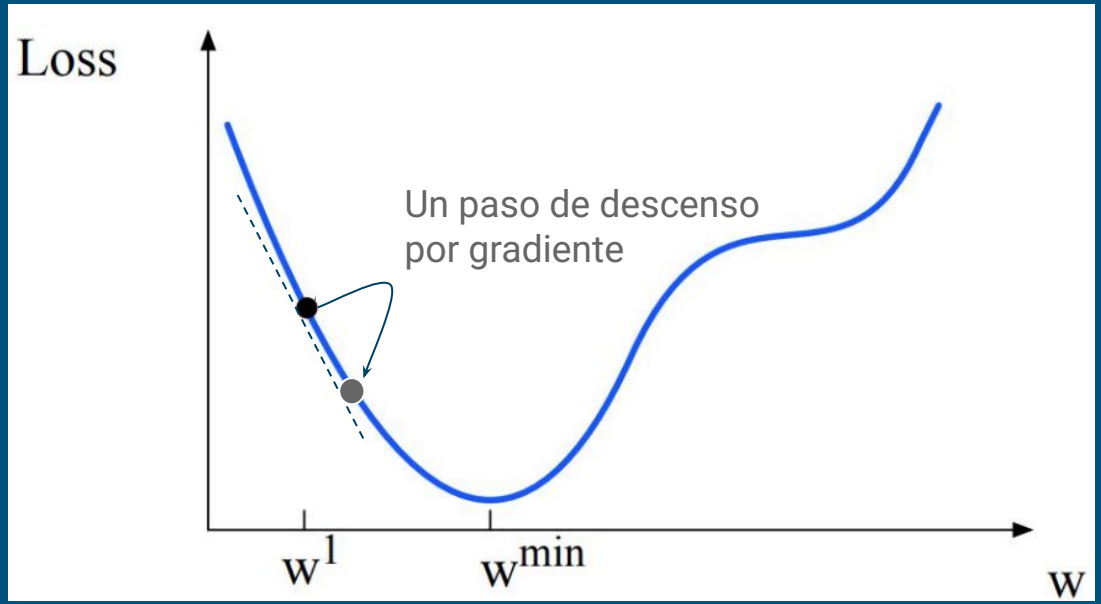
Visualización en una sola dimensión

La pendiente en w^1 es negativa
Así que hay que moverse en dirección positiva



Visualización en una sola dimensión

La pendiente en w^1 es negativa
Así que hay que moverse en dirección positiva



En varias variables: Gradiente

El gradiente de una función de varias variables es un vector que apunta en la dirección de mayor crecimiento.

Descenso por gradiente: Significa encontrar el gradiente de la función de loss en el punto actual y moverse en la dirección opuesta

¿Cuánto nos movemos? Un “paso” η

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x;w), y)$$

η es el Learning Rate

Entrenamiento

Se trata de encontrar los pesos que minimicen la función de pérdida

- Descenso por gradiente
- Estocástico / Mini-batch
- Backpropagation

Resumen: Arquitectura

- Dimensión de las características
- Tipo de red
- Número de capas
- Función de activación
- Función de pérdida
- Algoritmo de optimización

Agradecimientos

Parte de las explicaciones fueron modificadas de Rodrigo Ramele y Luis Chiruzzo. Mis agradecimientos a ellos.

MNIST

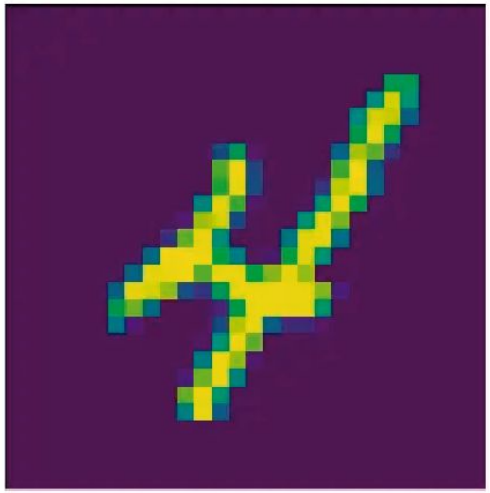


Imagen que se desea clasificar