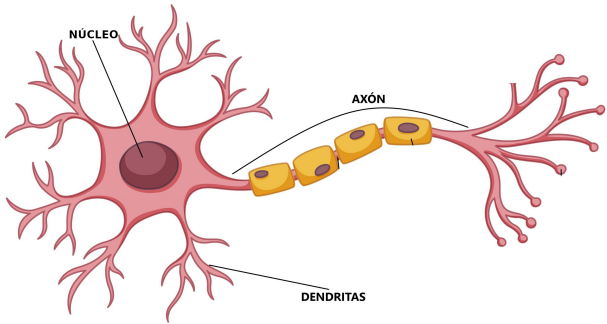
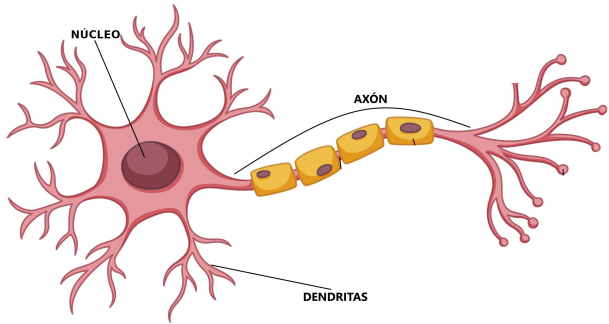
Redes Neuronales

Laboratorio de Datos, IC - FCEN - UBA - 1er. Cuatrimestre 2024

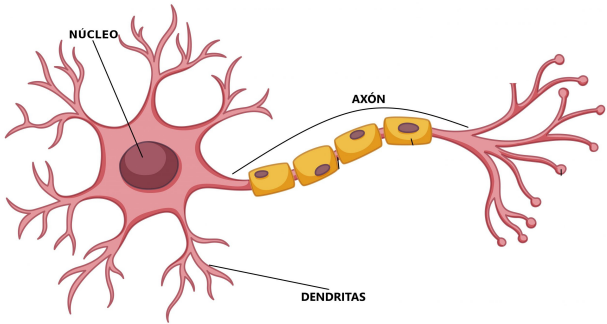
La neurona



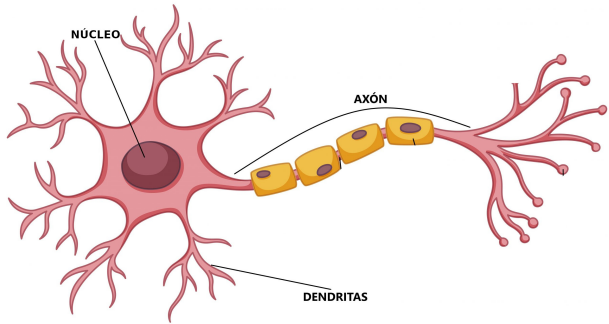
- Las neuronas se comunican a través de impulsos (sinapsis).



- Las neuronas se comunican a través de impulsos (sinapsis).
- Una neurona recibe los impulsos de las neuronas precedentes en la red. Estos impulsos pueden ser excitatorios (+) o inhibitorios (-).

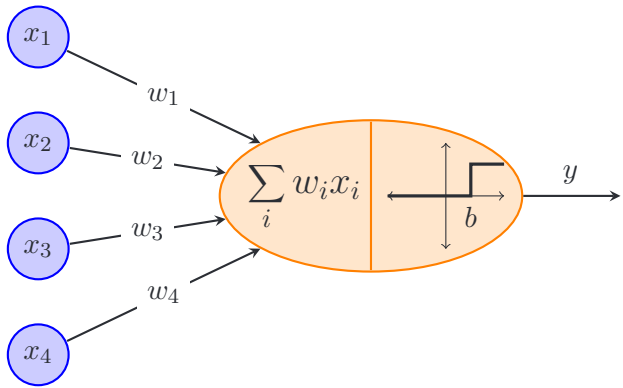


- Durante un corto periodo de tiempo, la neurona *suma* los impulsos recibidos.

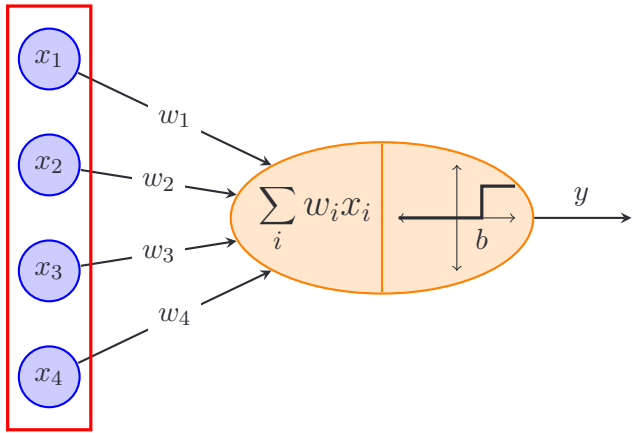


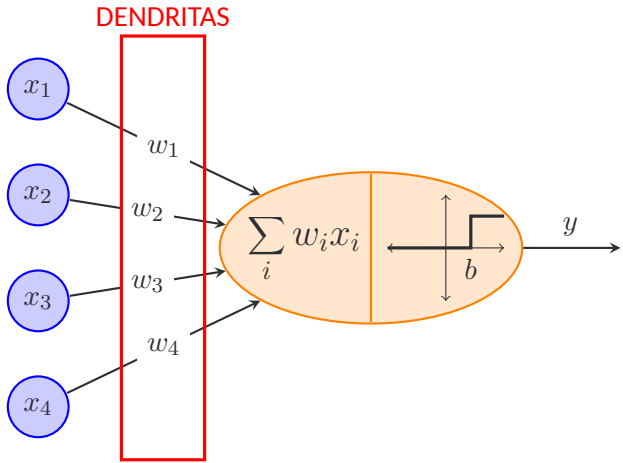
- Durante un corto periodo de tiempo, la neurona *suma* los impulsos recibidos.
- La neurona dispara un impulso si la suma supera cierto umbral.

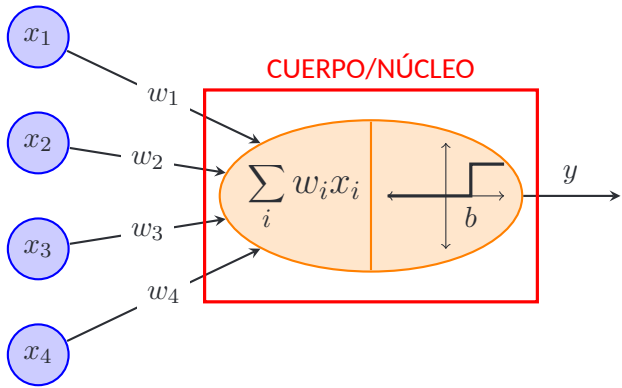
Perceptrón Simple

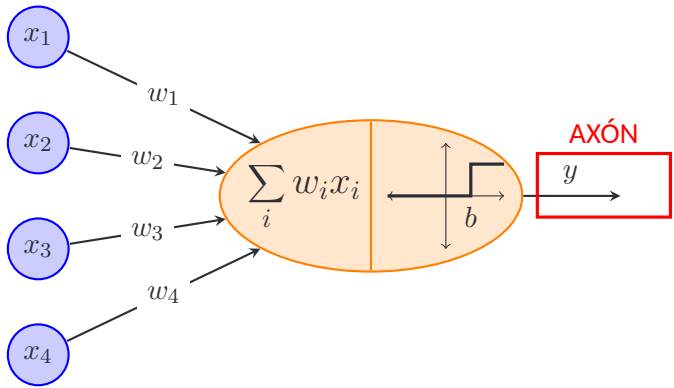

x_1
x_2
x_3
x_4
w_1
w_2
w_3
w_4
$\sum_i w_i x_i$
b
y

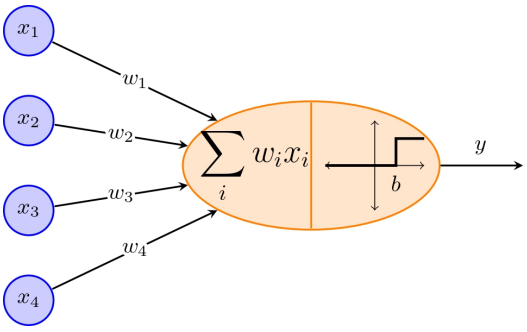
NEURONAS PREVIAS




DENDRITAS
x_1
x_2
x_3
x_4
w_1
w_2
w_3
w_4
$\sum_i w_i x_i$
b
y

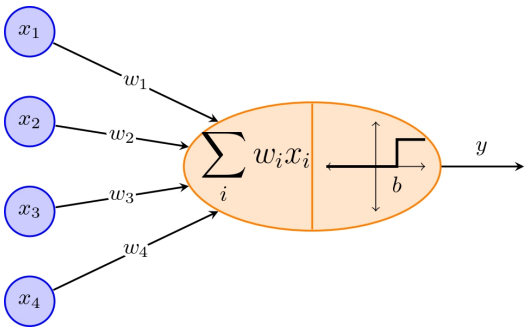

CUERPO/NÚCLEO
x_1
x_2
x_3
x_4
w_1
w_2
w_3
w_4
$\sum_i w_i x_i$
b
y


x_1
x_2
x_3
x_4
w_1
w_2
w_3
w_4
$\sum_i w_i x_i$
b
AXÓN
y



1. La neurona recibe los impulsos de las neuronas precedentes y los suma:

$$\sum_i w_i x_i$$



1. La neurona recibe los impulsos de las neuronas precedentes y los suma:

$$\sum_i w_i x_i$$

2. Si la suma supera el umbral b, la neurona *dispara*

$$y = \begin{cases} 1 & \text{si } \sum_i w_i x_i > b \\ 0 & \text{si no} \end{cases}$$

Obs:

$$\sum_i w_i x_i > b \iff \sum_i w_i x_i - b > 0 \iff \sum_i w_i x_i + b \cdot (-1) > 0$$

$\Rightarrow$ agregamos una neurona para el bias, la neurona se activa si el impulso que recibe es mayor que 0

Obs:

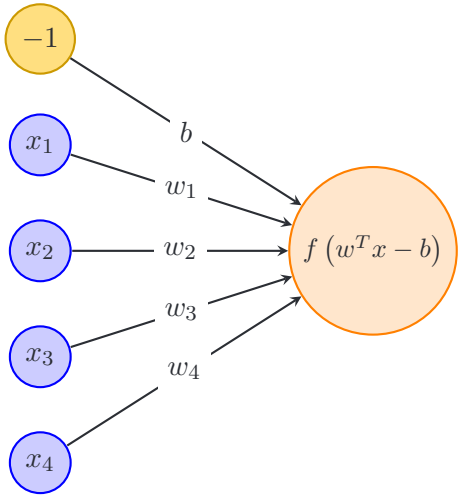
$$\sum_i w_i x_i > b \iff \sum_i w_i x_i - b > 0 \iff \sum_i w_i x_i + b \cdot (-1) > 0$$

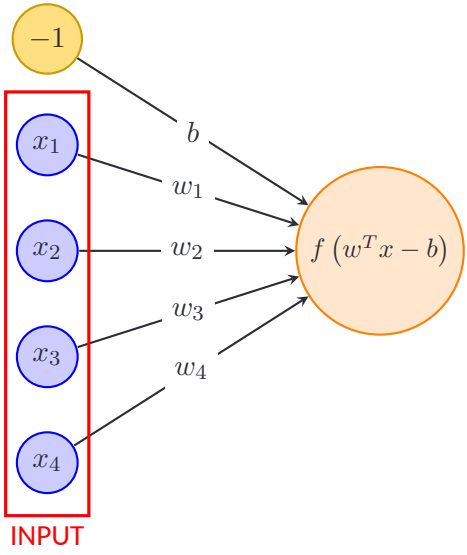
$\Rightarrow$ agregamos una neurona para el bias, la neurona se activa si el impulso que recibe es mayor que 0

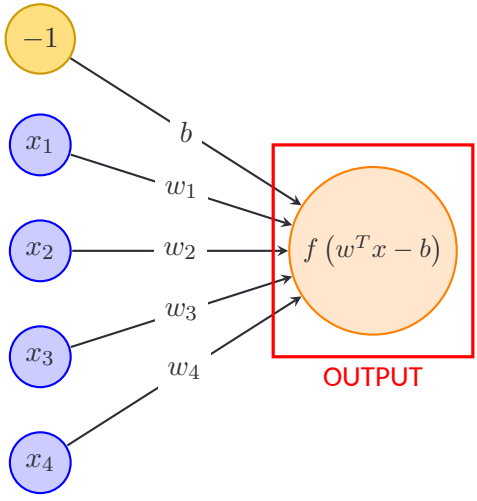
Consideramos la siguiente función:

$$f(t) = \begin{cases} 1 & \text{si } t > 0 \\ 0 & \text{si no} \end{cases}$$

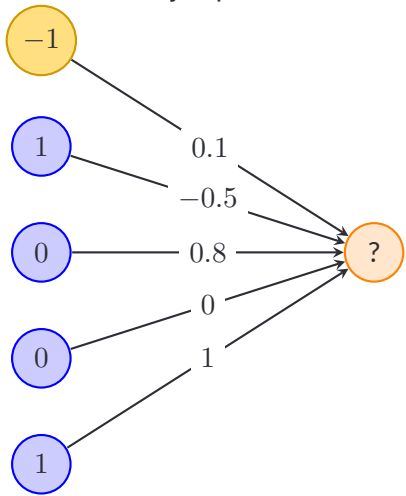
Que será nuestra **función de activación**.


-1
x_1
x_2
x_3
x_4
b
w_1
w_2
w_3
w_4
$f\left(w^T x - b\right)$

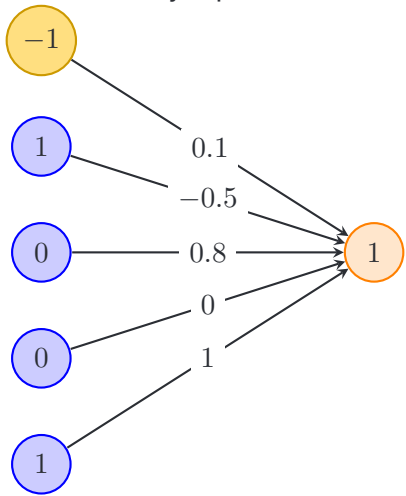

-1
b
x_1
w_1
x_2
w_2
$f\left(w^T x - b\right)$
w_3
x_3
w_4
x_4
INPUT


-1
x_1
x_2
x_3
x_4
b
w_1
w_2
w_3
w_4
$f\left(w^T x - b\right)$
OUTPUT

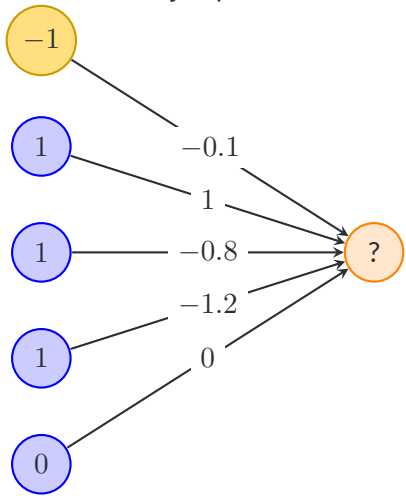
Ejemplo



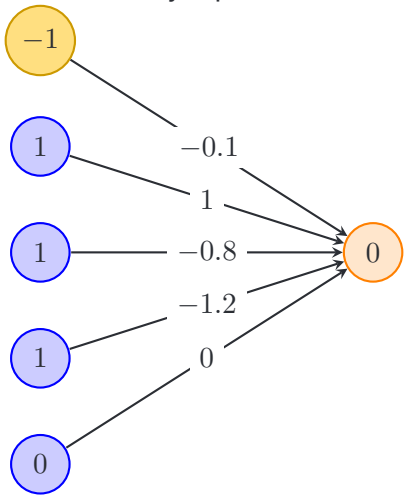
Ejemplo



Ejemplo



Ejemplo



Entrenando a la Red Neuronal

Las redes neuronales tienen **muchísimas** aplicaciones:

- Clasificación

Las redes neuronales tienen **muchísimas** aplicaciones:

- Clasificación
- Regresión

Las redes neuronales tienen **muchísimas** aplicaciones:

- Clasificación
- Regresión
- Clustering

Las redes neuronales tienen **muchísimas** aplicaciones:

- Clasificación
- Regresión
- Clustering
- Reducción de dimensionalidad

Las redes neuronales tienen **muchísimas** aplicaciones:

- Clasificación
- Regresión
- Clustering
- Reducción de dimensionalidad
- Compresión (*auto-encoders*)

Las redes neuronales tienen **muchísimas** aplicaciones:

- Clasificación
- Regresión
- Clustering
- Reducción de dimensionalidad
- Compresión (*auto-encoders*)
- Reconocimiento de patrones

Las redes neuronales tienen **muchísimas** aplicaciones:

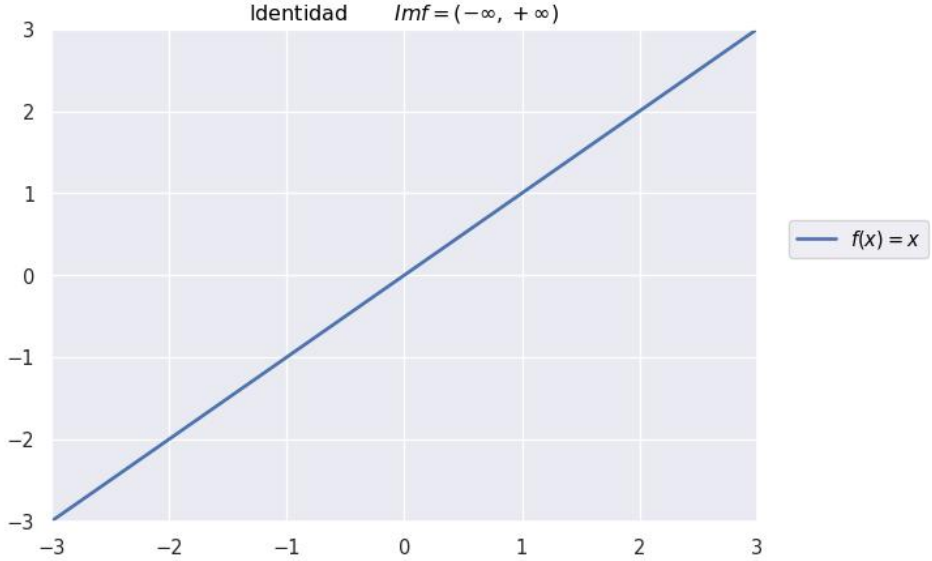
- Clasificación
- Regresión
- Clustering
- Reducción de dimensionalidad
- Compresión (*auto-encoders*)
- Reconocimiento de patrones
- Y muchas más!

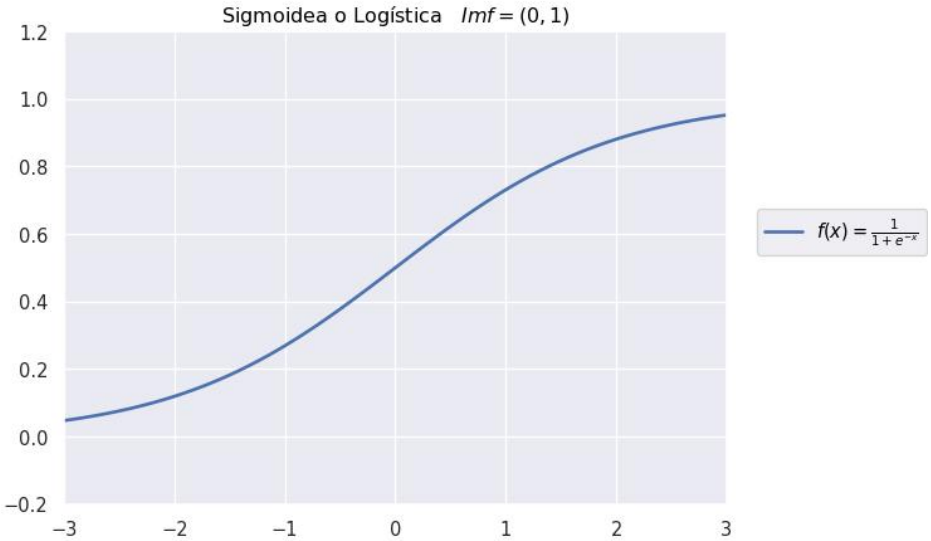
Vamos a centrarnos en **clasificación** y **regresión**.

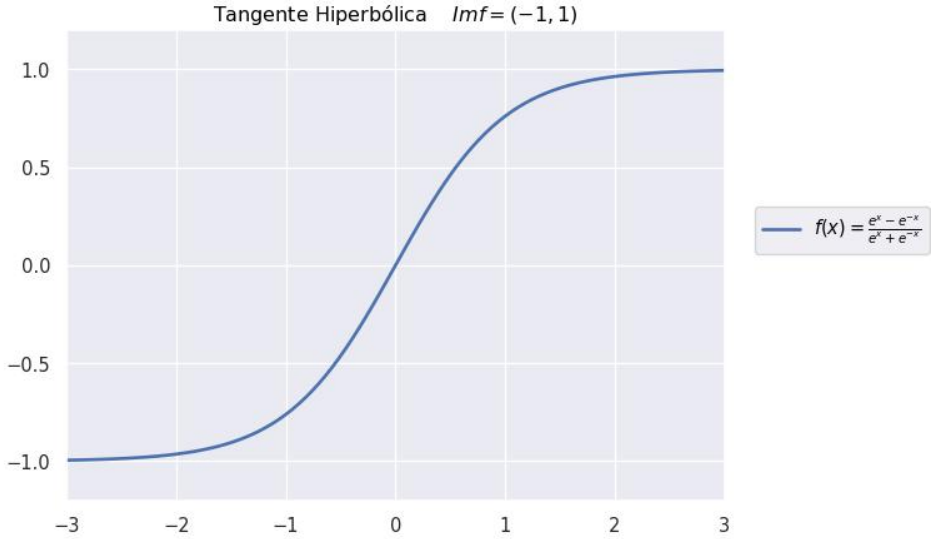
Para poder ampliar la variedad de problemas los que puede aplicarse nuestro modelo de Perceptrón Simple, vamos a suponer que el impulso que transmiten las neuronas no es necesariamente binario.

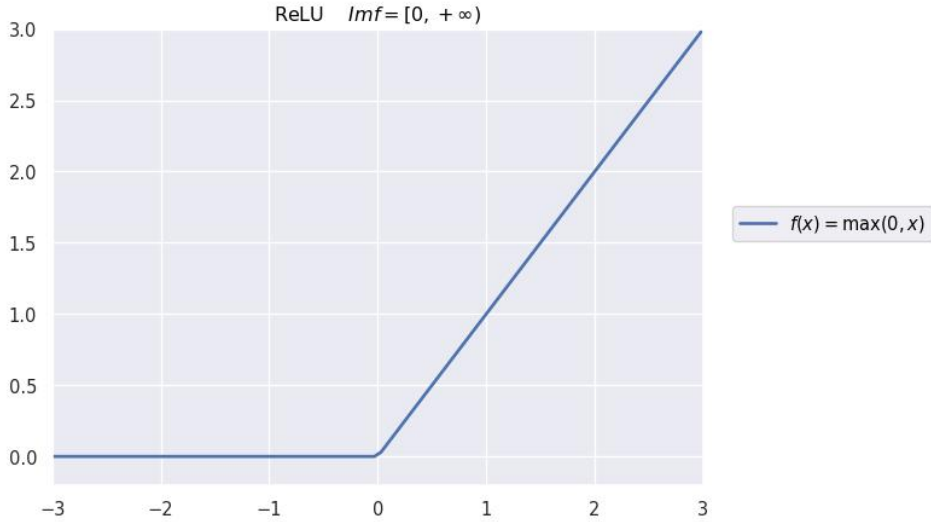
Para poder ampliar la variedad de problemas los que puede aplicarse nuestro modelo de Perceptrón Simple, vamos a suponer que el impulso que transmiten las neuronas no es necesariamente binario.

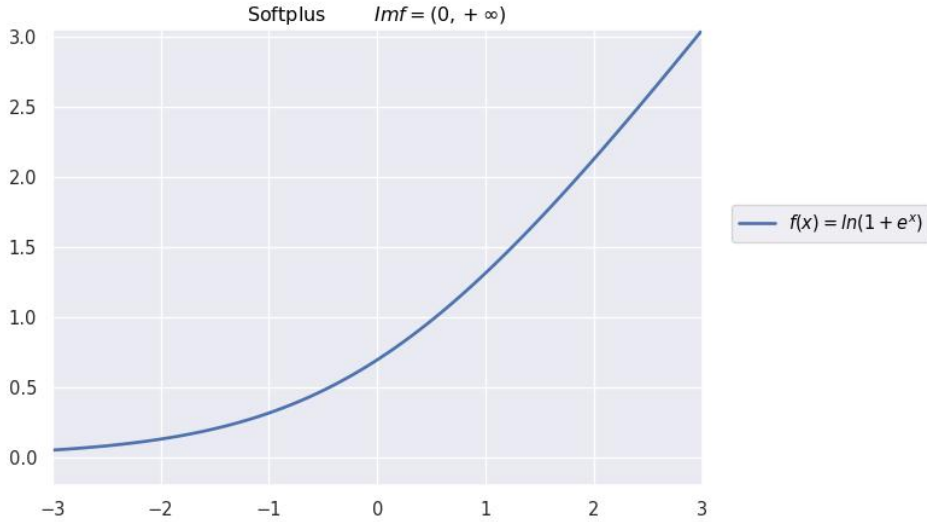
$\rightarrow$ consideramos **otras funciones de activación** para la neurona de output.


Identidad
$Imf = (-\infty, +\infty)$
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3
$f(x) = x$


Sigmoidea o Logística $Imf = (0, 1)$
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−3
−2
−1
0
1
2
3
$f(x) = \frac{1}{1+e^{-x}}$


Tangente Hiperbólica $Imf = (-1, 1)$
1.0
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3
$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$


ReLU $Imf = [0, +\infty)$
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−3
−2
−1
0
1
2
3
$f(x) = \max(0, x)$


Softplus
$Imf = (0, +\infty)$
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−3
−2
−1
0
1
2
3
$f(x) = ln(1 + e^x)$

IMPORTANTE: la imagen de la función de activación de la neurona de output tiene que ser coherente con los datos de salida (y).

Entrenando a la red

Como en todos los modelos de aprendizaje supervisado, queremos minimizar la discrepancia (medida por la función de pérdida) que existe entre lo que devuelve el modelo y el dato real.

Entrenando a la red

Como en todos los modelos de aprendizaje supervisado, queremos minimizar la discrepancia (medida por la función de pérdida) que existe entre lo que devuelve el modelo y el dato real.

Si vamos a minimizar una función, ¿qué algoritmo podemos usar?

Entrenando a la red

Como en todos los modelos de aprendizaje supervisado, queremos minimizar la discrepancia (medida por la función de pérdida) que existe entre lo que devuelve el modelo y el dato real.

Si vamos a minimizar una función, ¿qué algoritmo podemos usar?

✨ Descenso por Gradiente Estocástico ✨

El entrenamiento consiste en ir ajustando los pesos y el bias para minimizar la función de pérdida.

1. Inicializá los pesos y el bias aleatoriamente
2. En cada época:
   - 2.1 Para cada dato $x^{(i)}$:
     - 2.1.1 *Feedforward*: alimentamos a la red con $x^{(i)}$ como input, obtenemos $\hat{y}^{(i)}$ como output
     - 2.1.2 *Corrección*: medimos el error entre $y^{(i)}$ y $\hat{y}^{(i)}$, ajustamos los pesos para disminuirlo
   - 2.2 mezclá el dataset

Clasificando pingüinos

Vamos a entrenar un perceptrón para predecir el sexo de un pingüino a partir de cuatro características:

- peso (X_1)
- longitud de aleta (X_2)
- longitud de pico (X_3)
- profundidad del pico (X_4)

Vamos a entrenar un perceptrón para predecir el sexo de un pingüino a partir de cuatro características:

- peso (X_1)
- longitud de aleta (X_2)
- longitud de pico (X_3)
- profundidad del pico (X_4)

▶Arquitectura de la red neuronal:

- Neuronas en la capa de Input:
- Neuronas en la capa de Output:
- Función de activación:
- Función de pérdida (L):

Vamos a entrenar un perceptrón para predecir el sexo de un pingüino a partir de cuatro características:

- peso (X_1)
- longitud de aleta (X_2)
- longitud de pico (X_3)
- profundidad del pico (X_4)

▶Arquitectura de la red neuronal:

- Neuronas en la capa de Input: 4
- Neuronas en la capa de Output:
- Función de activación:
- Función de pérdida (L):

Vamos a entrenar un perceptrón para predecir el sexo de un pingüino a partir de cuatro características:

- peso (X_1)
- longitud de aleta (X_2)
- longitud de pico (X_3)
- profundidad del pico (X_4)

▶Arquitectura de la red neuronal:

- Neuronas en la capa de Input: 4
- Neuronas en la capa de Output: 1
- Función de activación:
- Función de pérdida (L):

Vamos a entrenar un perceptrón para predecir el sexo de un pingüino a partir de cuatro características:

- peso (X_1)
- longitud de aleta (X_2)
- longitud de pico (X_3)
- profundidad del pico (X_4)

▶Arquitectura de la red neuronal:

- Neuronas en la capa de Input: 4
- Neuronas en la capa de Output: 1
- Función de activación: Sigmoidea
- Función de pérdida (L):

Vamos a entrenar un perceptrón para predecir el sexo de un pingüino a partir de cuatro características:

- peso (X_1)
- longitud de aleta (X_2)
- longitud de pico (X_3)
- profundidad del pico (X_4)

▶Arquitectura de la red neuronal:

- Neuronas en la capa de Input: 4
- Neuronas en la capa de Output: 1
- Función de activación: Sigmoidea
- Función de pérdida (L): Binary Cross Entropy

Vamos a entrenar un perceptrón para predecir el sexo de un pingüino a partir de cuatro características:

- peso (X_1)
- longitud de aleta (X_2)
- longitud de pico (X_3)
- profundidad del pico (X_4)

▶Arquitectura de la red neuronal:

- Neuronas en la capa de Input: 4
- Neuronas en la capa de Output: 1
- Función de activación: Sigmoidea
- Función de pérdida (L): Binary Cross Entropy

▶Codificación y:

Female: 1 *Male:* 0

Vamos a entrenar un perceptrón para predecir el sexo de un pingüino a partir de cuatro características:

- peso (X_1)
- longitud de aleta (X_2)
- longitud de pico (X_3)
- profundidad del pico (X_4)

▶Arquitectura de la red neuronal:

- Neuronas en la capa de Input: 4
- Neuronas en la capa de Output: 1
- Función de activación: Sigmoidea
- Función de pérdida (L): Binary Cross Entropy

▶Codificación y:

Female: 1 *Male:* 0

▶Inicialización aleatoria de w y b

Vamos a entrenar un perceptrón para predecir el sexo de un pingüino a partir de cuatro características:

- peso (X_1)
- longitud de aleta (X_2)
- longitud de pico (X_3)
- profundidad del pico (X_4)

▶Arquitectura de la red neuronal:

- Neuronas en la capa de Input: 4
- Neuronas en la capa de Output: 1
- Función de activación: Sigmoidea
- Función de pérdida (L): Binary Cross Entropy

▶Codificación y:

Female: 1 *Male:* 0

▶Inicialización aleatoria de w y b ▶$\eta_k = 0.1$ constante

Para quien desee seguir las cuentas

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

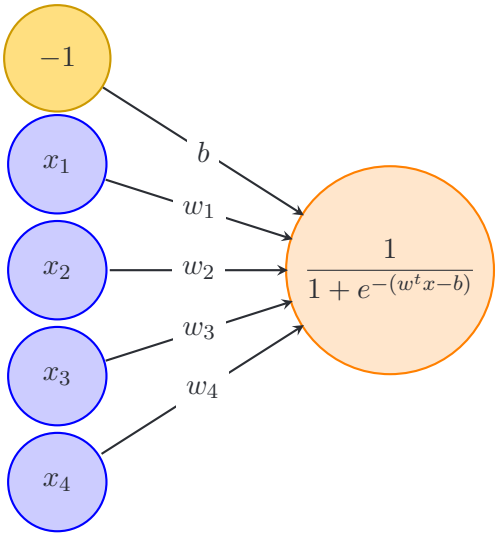
$$f_i(b, w) = \frac{1}{1 + e^{-(b + w^t x^{(i)})}} \qquad \text{vale que: } f'(x) = f(x)(1 - f(x))$$

Función de error *Binary Cross-Entropy Loss* o *log loss* para Regresión Logística:

$$L(b, w) = \frac{1}{n} \sum_{i=1}^{n} \underbrace{-\left(y^{(i)} \log(f_i(b, w)) + (1 - y^{(i)}) \log(1 - f_i(b, w)) \right)}_{F_i(b, w_1, \ldots, w_m)}$$

Tenemos que:

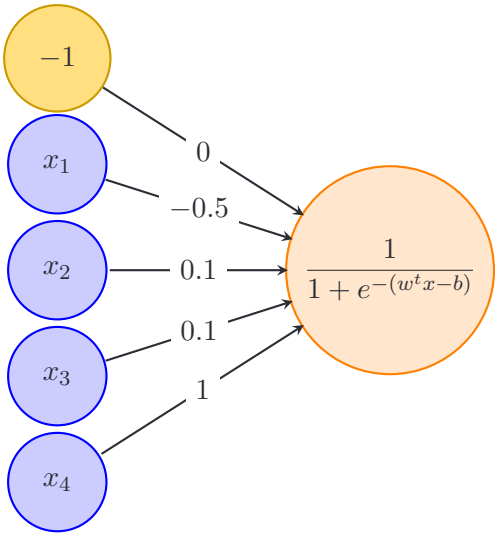
$$\frac{\partial F_i}{\partial w_j}(b, w_1, \ldots, w_m) = w_j(f_i(b, w_1, \ldots, w_m) - y_i)$$
$$\frac{\partial F_i}{\partial b}(b, w_1, \ldots, w_m) = -(f_i(b, w_1, \ldots, w_m) - y_i)$$



Datos de entrenamiento

	x_1	x_2	x_3	x_4	y
0	0.32	-0.07	0.31	-0.13	0
1	0.04	0.15	-0.33	0.05	1
2	-0.21	-0.34	-0.02	-0.29	1

Arquitectura del perceptrón



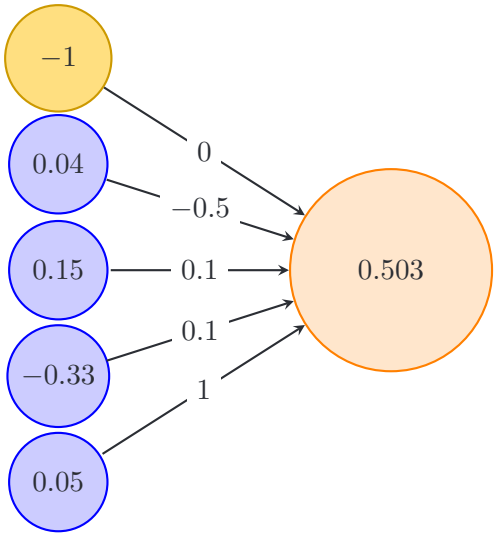
Datos de entrenamiento

	x_1	x_2	x_3	x_4	y
0	0.32	-0.07	0.31	-0.13	0
1	0.04	0.15	-0.33	0.05	1
2	-0.21	-0.34	-0.02	-0.29	1

Inicialización de los pesos y del bias

$$w = (-0.5, 0.1, 0.1, 1)$$

$$b = 0$$

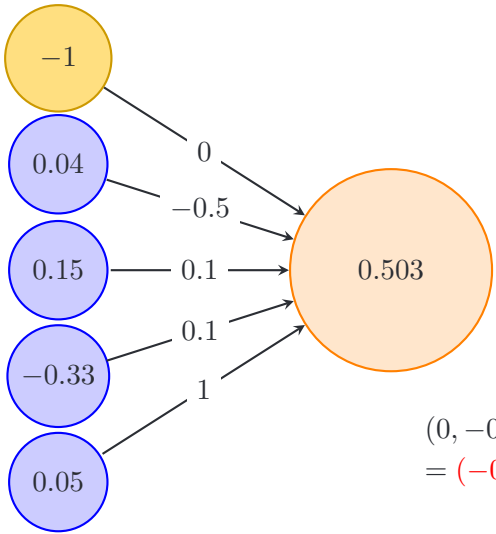


Datos de entrenamiento

	x_1	x_2	x_3	x_4	y
0	0.32	-0.07	0.31	-0.13	0
1	0.04	0.15	-0.33	0.05	1
2	-0.21	-0.34	-0.02	-0.29	1

Época: 1
Feedforward $x^{(1)}$

$$w^t x^{(1)} - 0 = 0.012$$
$$\hat{y}^{(1)} = f(0.012) = 0.503$$



Datos de entrenamiento

	x_1	x_2	x_3	x_4	y
0	0.32	-0.07	0.31	-0.13	0
1	0.04	0.15	-0.33	0.05	1
2	-0.21	-0.34	-0.02	-0.29	1

Época: 1

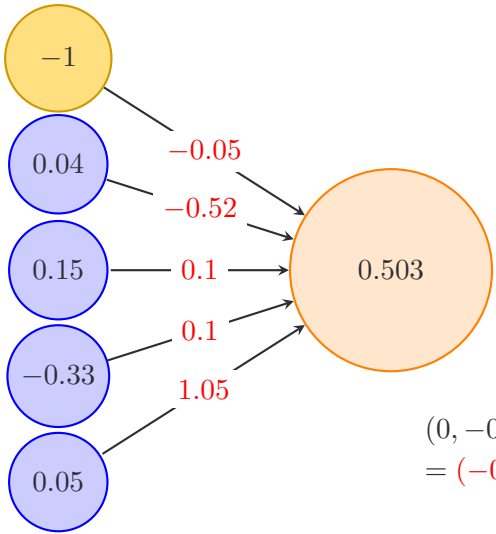
Corrección $x^{(1)}$

Se actualizan el bias y los pesos como indica SGD:

$$(b, w_1, \ldots, w_4) - 0.1 \cdot \nabla F_1(b, w_1, \ldots, w_4)$$

$$(0, -0.5, 0.1, 0.1, 1) - 0.1 \cdot \nabla F_1(0, -0.5, 0.1, 0.1, 1) =$$

$$= (-0.05, -0.52, 0.1, 0.1, 1.05) \leftarrow \text{Nuevos valores de } b \text{ y } w$$



Datos de entrenamiento

	x_1	x_2	x_3	x_4	y
0	0.32	-0.07	0.31	-0.13	0
1	0.04	0.15	-0.33	0.05	1
2	-0.21	-0.34	-0.02	-0.29	1

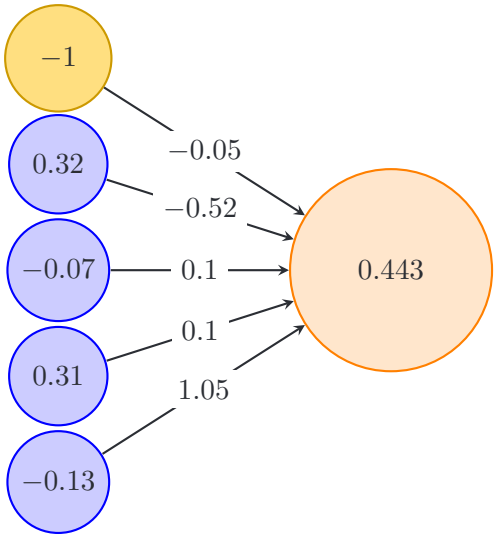
Época: 1

Se actualizan el bias y los pesos como indica SGD:

$(b, w_1, \ldots, w_4) - 0.1 \cdot \nabla F_1(b, w_1, \ldots, w_4)$

$(0, -0.5, 0.1, 0.1, 1) - 0.1 \cdot \nabla F_1(0, -0.5, 0.1, 0.1, 1) =$

$= (-0.05, -0.52, 0.1, 0.1, 1.05)$ ← Nuevos valores de b y w



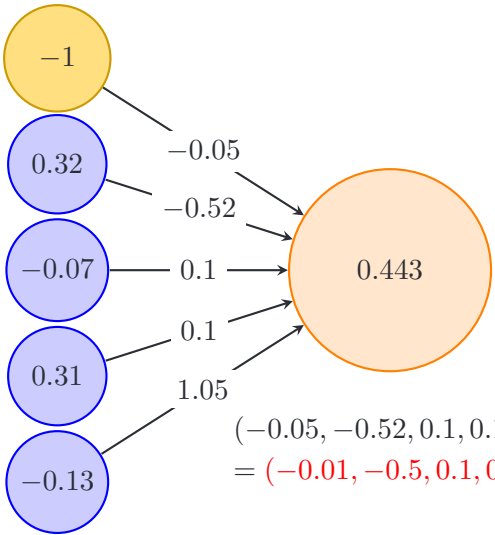
Datos de entrenamiento

	x_1	x_2	x_3	x_4	y
0	0.32	-0.07	0.31	-0.13	0
1	0.04	0.15	-0.33	0.05	1
2	-0.21	-0.34	-0.02	-0.29	1

Época: 1

Feedforward $x^{(0)}$

$$w^t x^{(0)} - (-0.05) = -0.23$$
$$\hat{y}^{(0)} = f(-0.23) = 0.443$$



Datos de entrenamiento

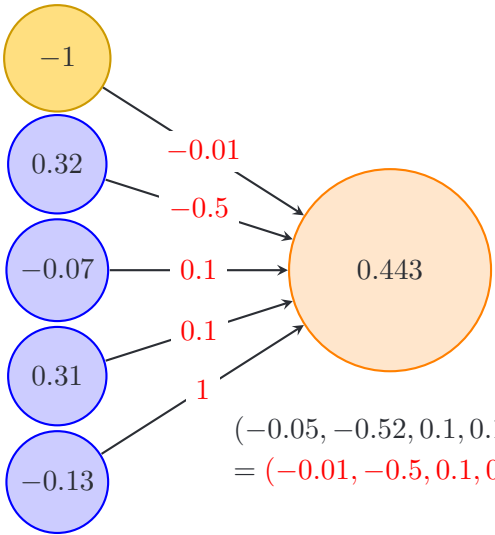
	x_1	x_2	x_3	x_4	y
0	0.32	-0.07	0.31	-0.13	0
1	0.04	0.15	-0.33	0.05	1
2	-0.21	-0.34	-0.02	-0.29	1

Época: 1
Corrección $x^{(0)}$

Se actualizan el bias y los pesos como indica SGD:

$$(b, w_1, \ldots, w_4) - 0.1 \cdot \nabla F_0(b, w_1, \ldots, w_4)$$

$$(-0.05, -0.52, 0.1, 0.1, 1.05) - 0.1 \cdot \nabla F_0(-0.05, -0.52, 0.1, 0.1, 1.05) =$$
$$= (-0.01, -0.5, 0.1, 0.1, 1) \leftarrow \text{Nuevos valores de } b \text{ y } w$$



Datos de entrenamiento

	x_1	x_2	x_3	x_4	y
0	0.32	-0.07	0.31	-0.13	0
1	0.04	0.15	-0.33	0.05	1
2	-0.21	-0.34	-0.02	-0.29	1

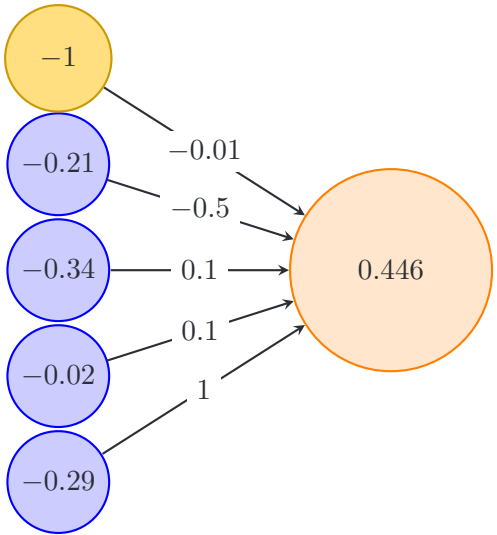
Época: 1

Corrección $x^{(0)}$

Se actualizan el bias y los pesos como indica SGD:

$(b, w_1, \ldots, w_4) - 0.1 \cdot \nabla F_0(b, w_1, \ldots, w_4)$

$$(-0.05, -0.52, 0.1, 0.1, 1.05) - 0.1 \cdot \nabla F_0(-0.05, -0.52, 0.1, 0.1, 1.05) =$$
$$= (-0.01, -0.5, 0.1, 0.1, 1) \leftarrow \text{Nuevos valores de } b \text{ y } w$$



Datos de entrenamiento

	x_1	x_2	x_3	x_4	y
0	0.32	-0.07	0.31	-0.13	0
1	0.04	0.15	-0.33	0.05	1
2	-0.21	-0.34	-0.02	-0.29	1

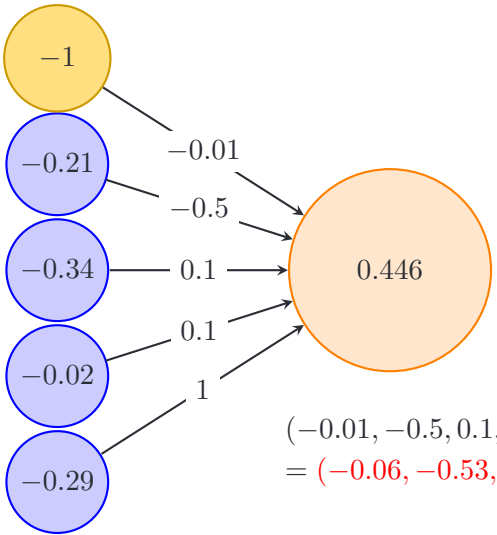
Época: 1

Feedforward $x^{(2)}$

$$w^t x^{(2)} - (-0.05) = -0.216$$
$$\hat{y}^{(2)} = f(-0.216) = 0.446$$

Datos de entrenamiento

	x_1	x_2	x_3	x_4	y
0	0.32	-0.07	0.31	-0.13	0
1	0.04	0.15	-0.33	0.05	1
2	-0.21	-0.34	-0.02	-0.29	1



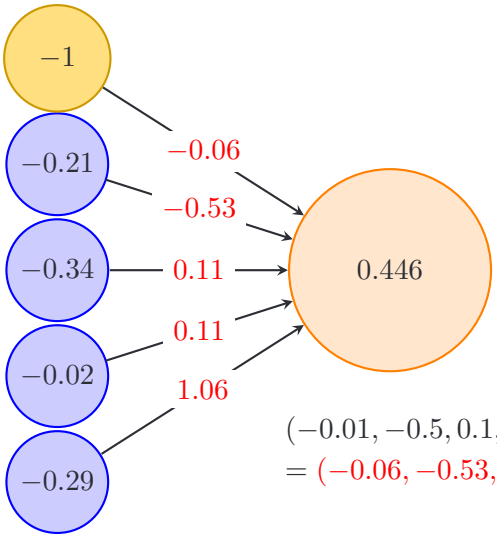
Época: 1

Corrección $x^{(2)}$

Se actualizan el bias y los pesos como indica SGD:

$(b, w_1, \ldots, w_4) - 0.1 \cdot \nabla F_2(b, w_1, \ldots, w_4)$

$$(-0.01, -0.5, 0.1, 0.1, 1) - 0.1 \cdot \nabla F_2(-0.01, -0.5, 0.1, 0.1, 1) =$$
$$= (-0.06, -0.53, 0.11, 0.11, 1.06) \leftarrow \text{Nuevos valores de } b \text{ y } w$$



Datos de entrenamiento

	x_1	x_2	x_3	x_4	y
0	0.32	-0.07	0.31	-0.13	0
1	0.04	0.15	-0.33	0.05	1
2	-0.21	-0.34	-0.02	-0.29	1

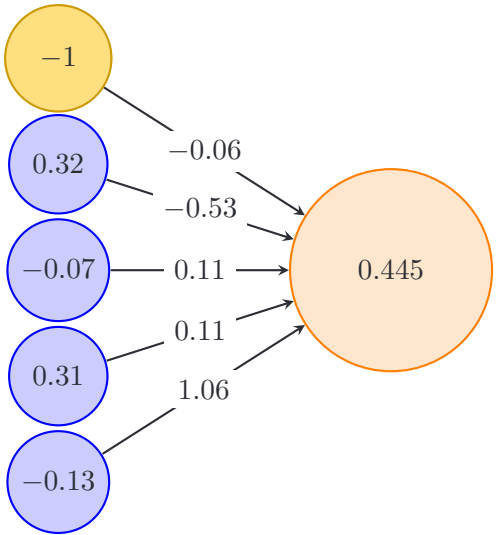
Época: 1

Corrección $x^{(2)}$

Se actualizan el bias y los pesos como indica SGD:

$(b, w_1, \ldots, w_4) - 0.1 \cdot \nabla F_2(b, w_1, \ldots, w_4)$

$$(-0.01, -0.5, 0.1, 0.1, 1) - 0.1 \cdot \nabla F_2(-0.01, -0.5, 0.1, 0.1, 1) =$$
$$= (-0.06, -0.53, 0.11, 0.11, 1.06) \leftarrow \text{Nuevos valores de } b \text{ y } w$$

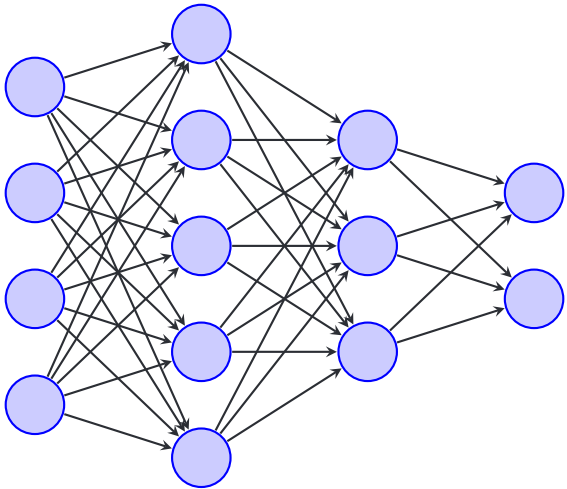


Datos de entrenamiento

	x_1	x_2	x_3	x_4	y
0	0.32	-0.07	0.31	-0.13	0
1	0.04	0.15	-0.33	0.05	1
2	-0.21	-0.34	-0.02	-0.29	1

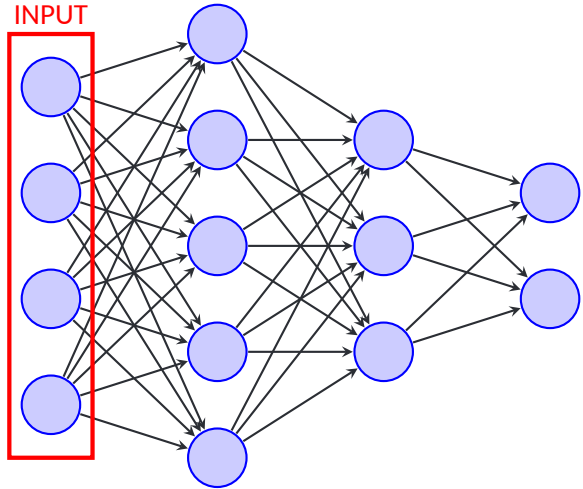
Comienza la Época: 2...

Redes feedforward multicapa



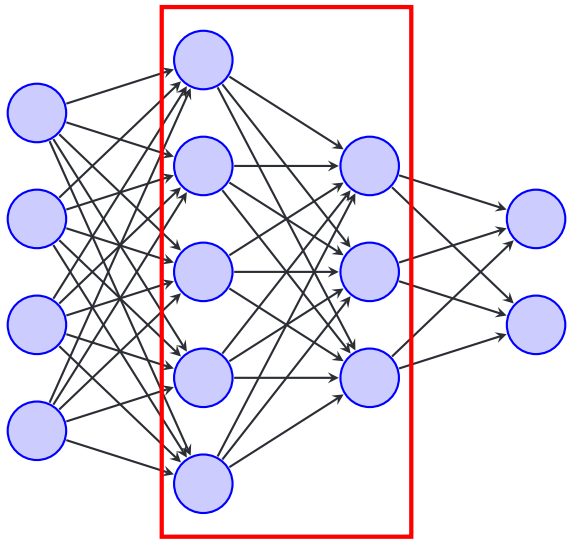
Para entender backpropagation:

https://remykarem.github.io/backpropagation-demo/



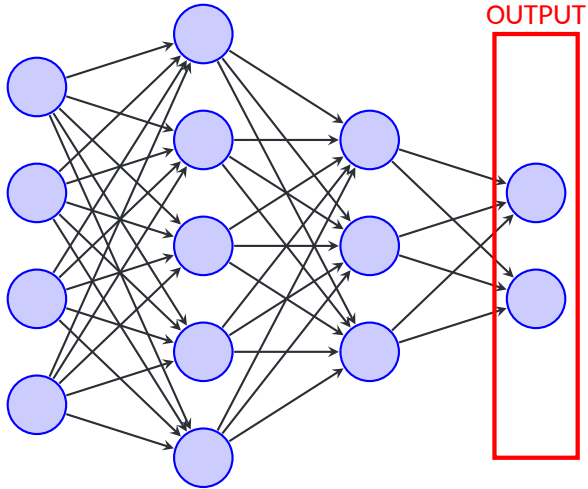
Para entender backpropagation:

https://remykarem.github.io/backpropagation-demo/



Para entender backpropagation:

https://remykarem.github.io/backpropagation-demo/



Para entender backpropagation:

https://remykarem.github.io/backpropagation-demo/

Existen muchos teoremas de aproximación universal, que demuestran que existen redes neuronales que convergen a ciertas clases de funciones.

Por ejemplo, el Teorema de Funahashi:
Sean $K \subset \mathbb{R}^n$ compacto, $f: K \to \mathbb{R}$ continua y $\varepsilon > 0$, entonces existe g una red neuronal de dos capas tal que $\|f - g\|_\infty < \varepsilon$


HIPERPARAMETROS
LIGHTYEAR
HIPERPARAMETROS POR TODOS LADOS
imgflip.com