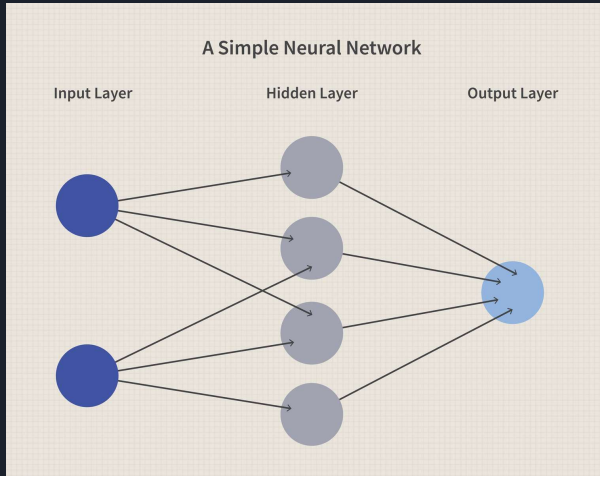
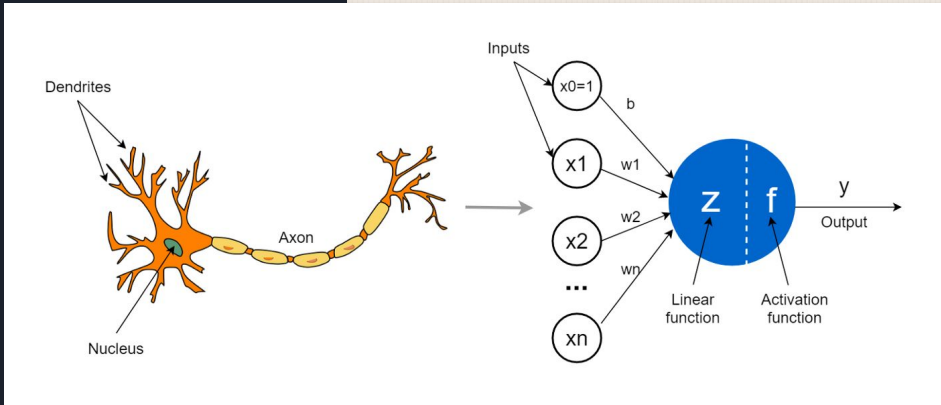
Ayudantía 10: Redes Neuronales (NN)

¿Qué es una Red Neuronal?

Redes Neuronales:

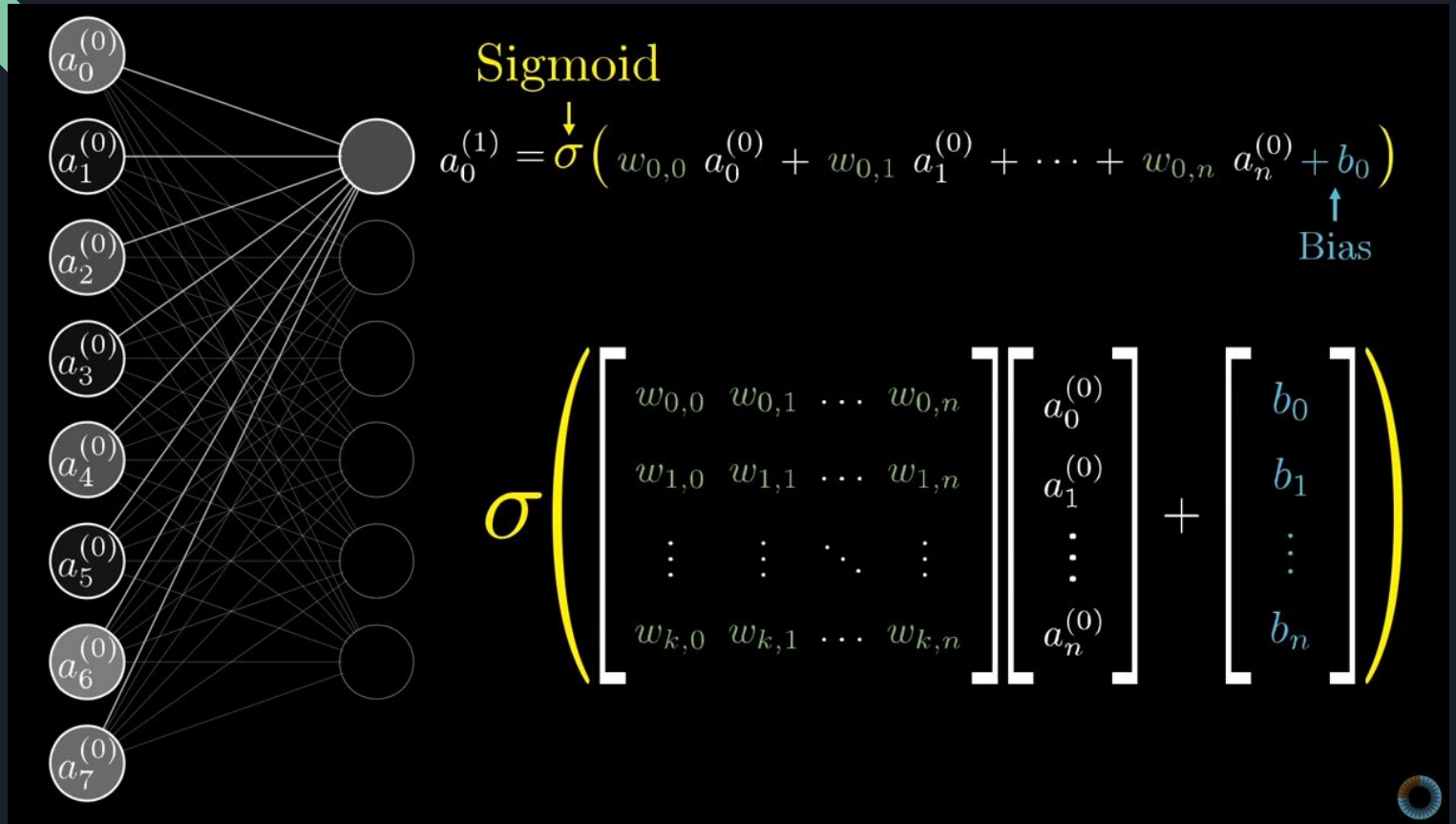
- Son un conjunto de nodos **(neuronas)** que intentan “imitar” el funcionamiento de un cerebro biológico
- Las neuronas están conectadas por flechas o enlaces, los cuales tienen un **peso** asociado
- Adicionalmente pueden haber **bias** que son simplemente valores constantes que se le aplican como input a una neurona
- Luego se hace la sumatoria de todos los inputs y eso se “pasa” por una **función de activación**
- Esto finalmente entrega un **valor de activación** para cada neurona





Pregunta!

Respuesta

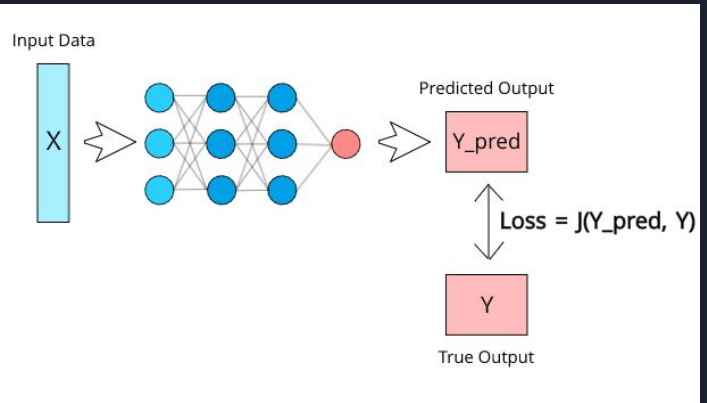

Sigmoid
$a_0^{(1)} = \sigma\left(w_{0,0}\ a_0^{(0)} + w_{0,1}\ a_1^{(0)} + \cdots + w_{0,n}\ a_n^{(0)} + b_0 \right)$
Bias
$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} \right)$

Pregunta!

¿Cómo entrenamos a la Red?

Función de Costo o Loss

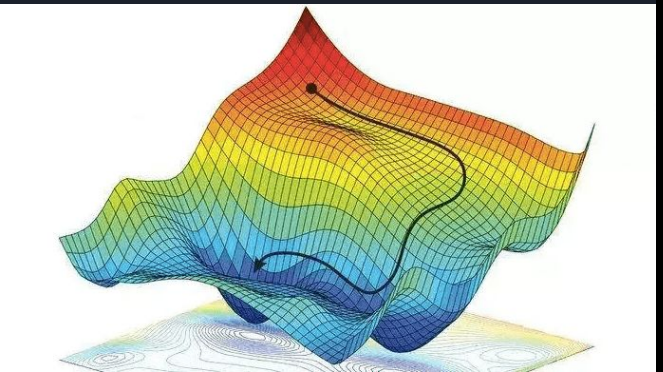
- Función que cuantifica la diferencia promedio entre los valores predecidos y los valores reales
- Sumatoria de las diferencias al cuadrado

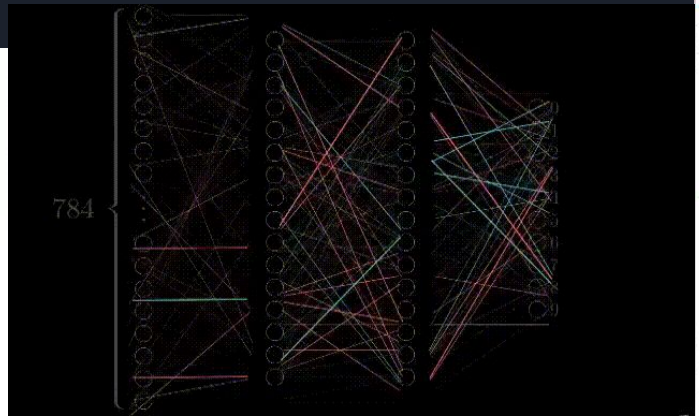


$$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$$

Gradient Descent

- La idea es calcular el gradiente sobre la función Loss, con el objetivo de ir disminuyendo su valor de forma iterativa
- La NN predice → calcula el Loss → Obtiene su gradiente → actualiza los parámetros → repite





Batch mode Gradient Descent:
Do until satisfied

1. Compute the gradient $\nabla E_D[\vec{w}]$
2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_D[\vec{w}]$

Incremental mode Gradient Descent:
Do until satisfied

- For each training example d in D
  1. Compute the gradient $\nabla E_d[\vec{w}]$
  2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_d[\vec{w}]$

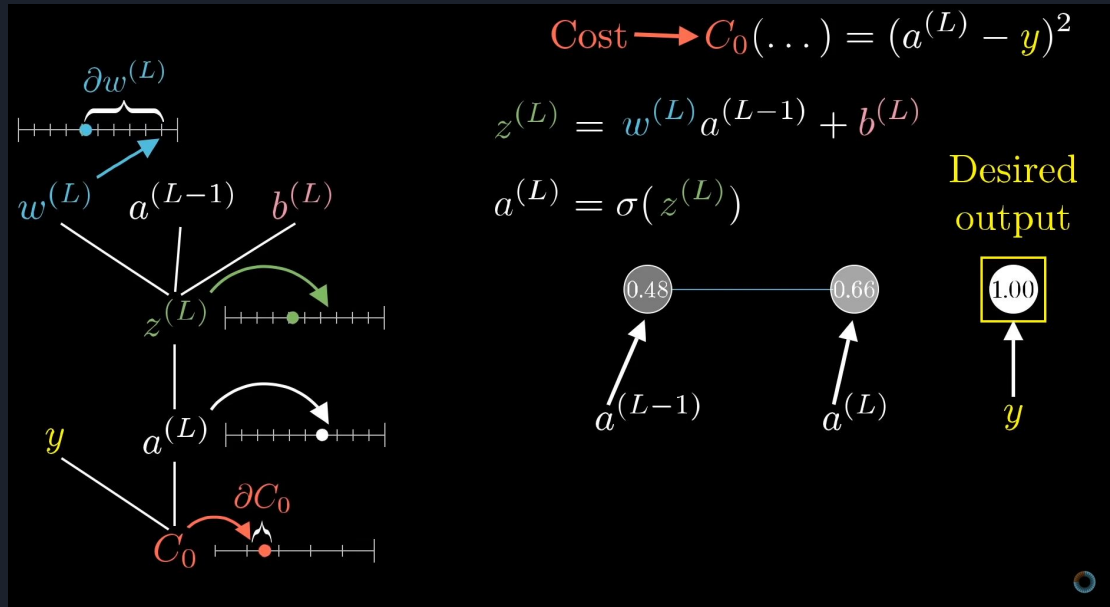
$$E_D[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

$$E_d[\vec{w}] \equiv \frac{1}{2}(t_d - o_d)^2$$

Incremental Gradient Descent can approximate *Batch Gradient Descent* arbitrarily closely if η made small enough

Backpropagation

- Backpropagation es simplemente aplicar la regla de la cadena a la función Loss
- Lo que queremos es ver cómo afecta variar cada uno de los diferentes parámetros de la Red Neuronal a la función Loss



Pregunta!

Learning rate

- El **learning rate** es un parámetro que usamos como ponderador para el gradiente, en el SGD (comúnmente entre 0 y 1)
- Su función es hacernos descender por la función de Costo (Loss) de la forma más “eficiente” posible

Batch mode Gradient Descent:
Do until satisfied

1. Compute the gradient $\nabla E_D[\vec{w}]$
2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_D[\vec{w}]$

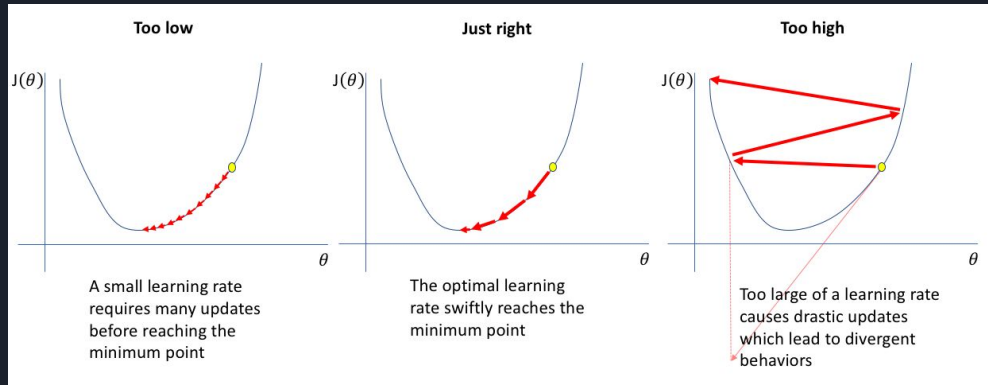
Incremental mode Gradient Descent:
Do until satisfied

- For each training example d in D
  1. Compute the gradient $\nabla E_d[\vec{w}]$
  2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_d[\vec{w}]$

$$E_D[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

$$E_d[\vec{w}] \equiv \frac{1}{2}(t_d - o_d)^2$$

Incremental Gradient Descent can approximate *Batch Gradient Descent* arbitrarily closely if η made small enough



Pregunta!

Playlist de YouTube muy buena sobre NNs

Neural networks de 3Blue1Brown:
https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

(harta info de la ay. sacada de ahí)