Ayudantía 12: Redes Neuronales

¿Qué es una red neuronal?

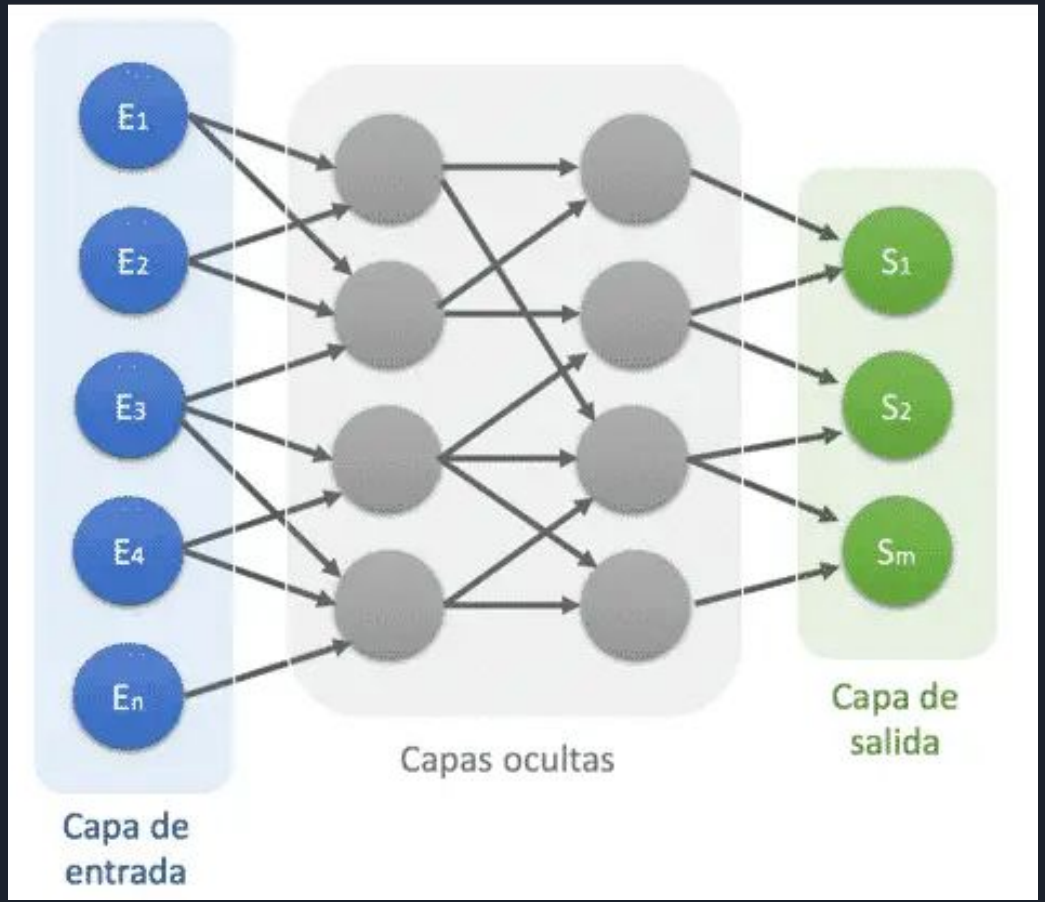
Modelo de aprendizaje de máquina inspirado en el funcionamiento del cerebro humano.

Consta de un grupo de nodos (neuronas) que reciben una entrada y entregan una salida que son enviadas como entradas a otras neuronas, conformando una red.

Las redes neuronales pueden reconocer patrones, realizar predicciones y tomar decisiones en base a datos de entrada, entre otras cosas.

¿Qué compone una red neuronal?

- Capa de entrada
- Capas ocultas
- Capa de salida
- Parámetros (Pesos y bias)
- Funciones de activación
- Función de costo o Loss
- Optimizador (Descenso de gradiente)
- Learning rate


E1
E2
E3
E4
En
S1
S2
Sm
Capa de entrada
Capas ocultas
Capa de salida

Pesos y Bias

Buscan obtener un valor de activación para una neurona, a partir de los valores recibidos de las neuronas anteriores.

El valor de activación se pasa como input a una función de activación, para obtener el valor “final” de la neurona.

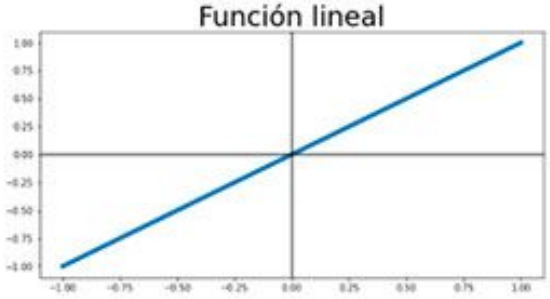
Función de activación

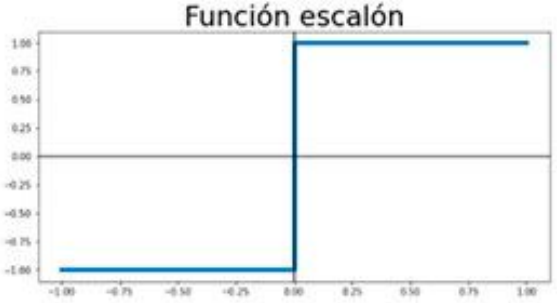
Filtro, función limitadora o umbral, que modifica el valor resultado o impone un límite que se debe sobrepasar para poder proseguir a otra neurona.

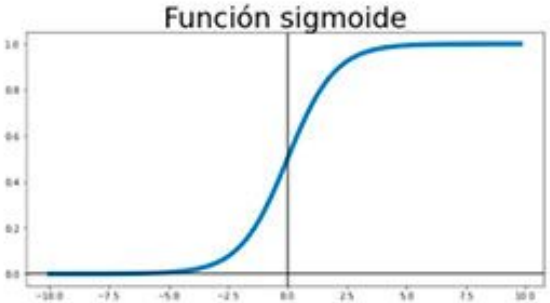
Se utilizan para introducir no linealidad en la red, permitiendo a la red aprender relaciones y patrones complejos en los datos.

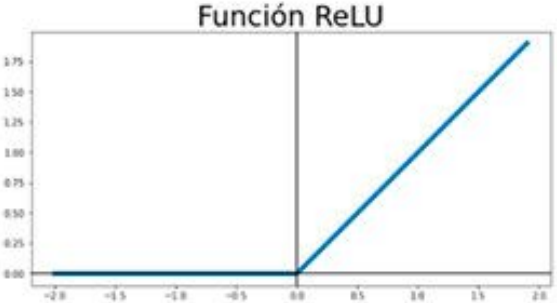
Existen distintas funciones y si se usa una u otra depende de la estructura y objetivo de la red neuronal.

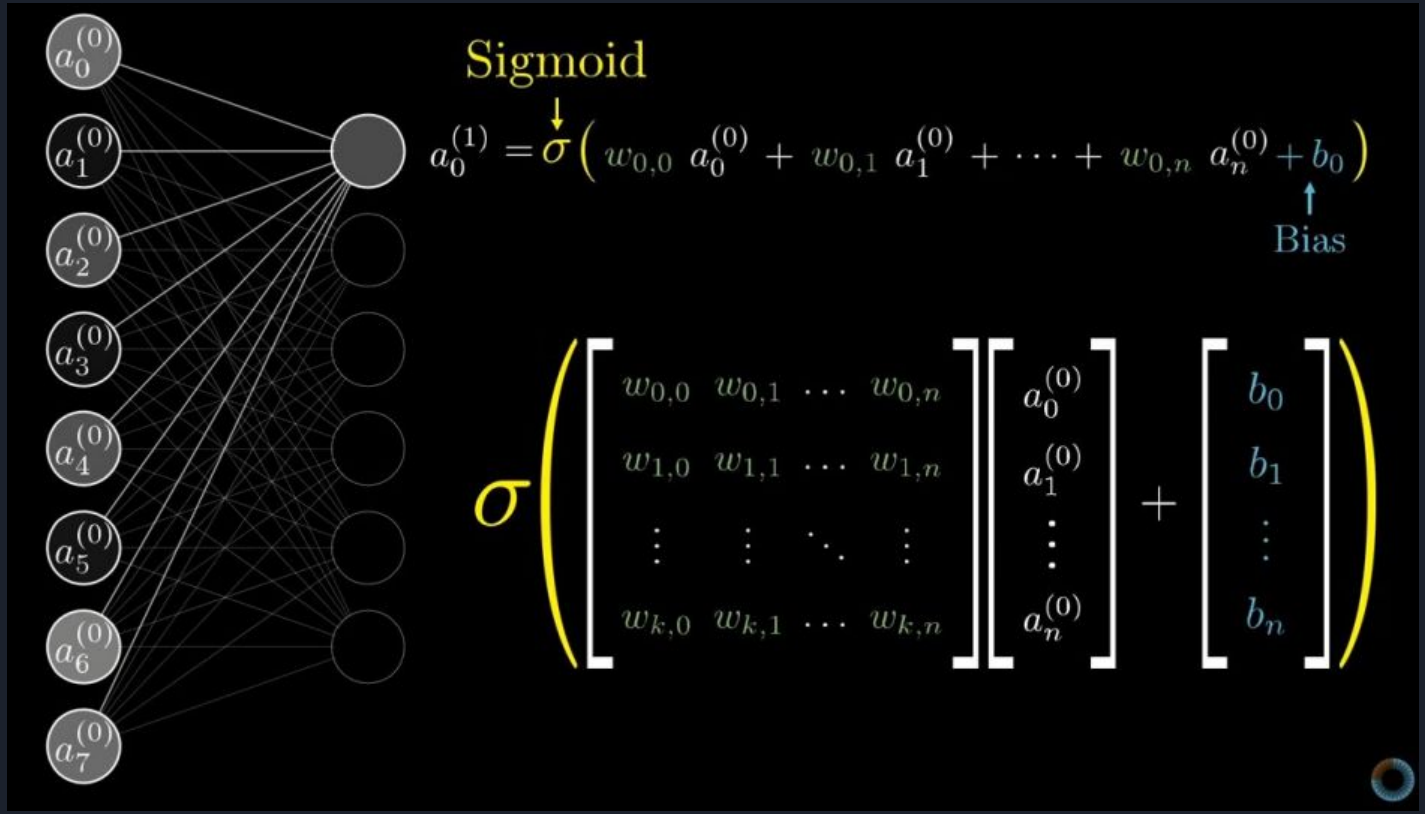
Por lo general, las neuronas de la capa de entrada no tienen función de activación, pero pueden llegar a tener.


Función lineal


Función escalón


Función sigmoide


Función ReLU


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

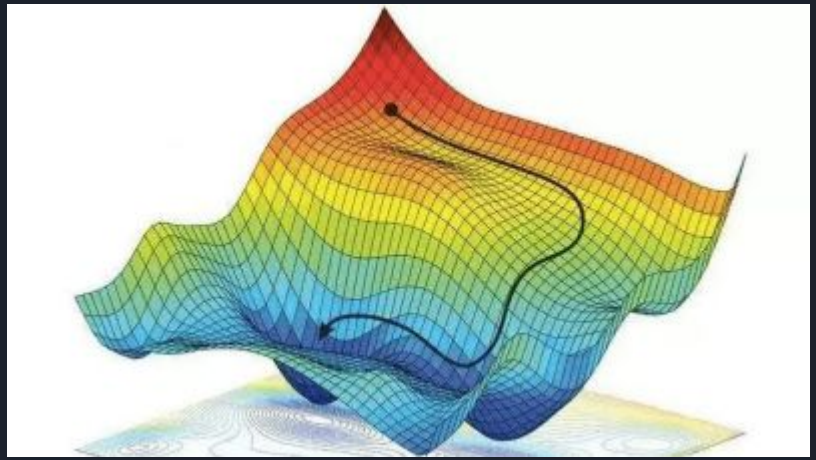
Función de costo o Loss

Cuantifica diferencia promedio entre los valores predecidos y los valores reales.

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

Gradient Descent

Optimizador que se basa en buscar mínimos locales (globales en el caso óptimo) en la función de costo, para ajustar los pesos y los bias.



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. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_d[\vec{w}]$

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

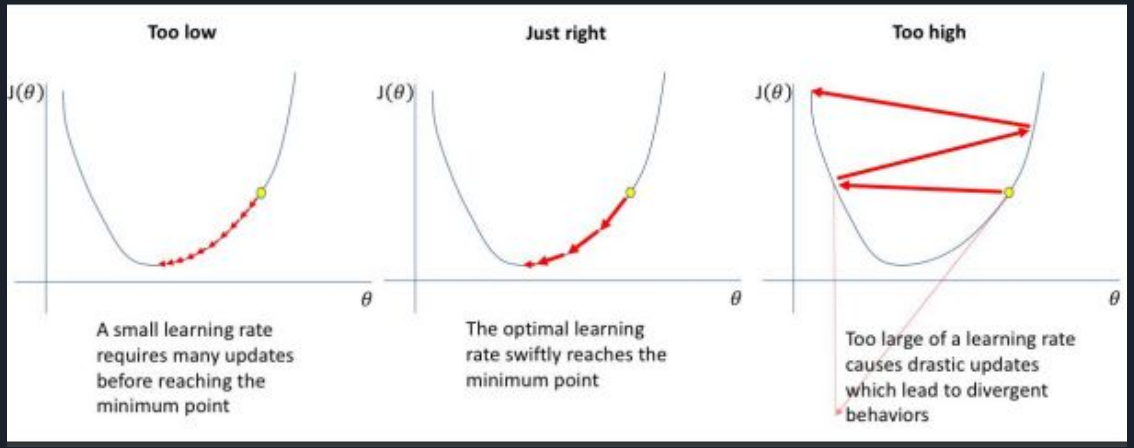
$$E_d[\vec{w}] \equiv \frac{1}{2}(t_d - o_d)^2$$

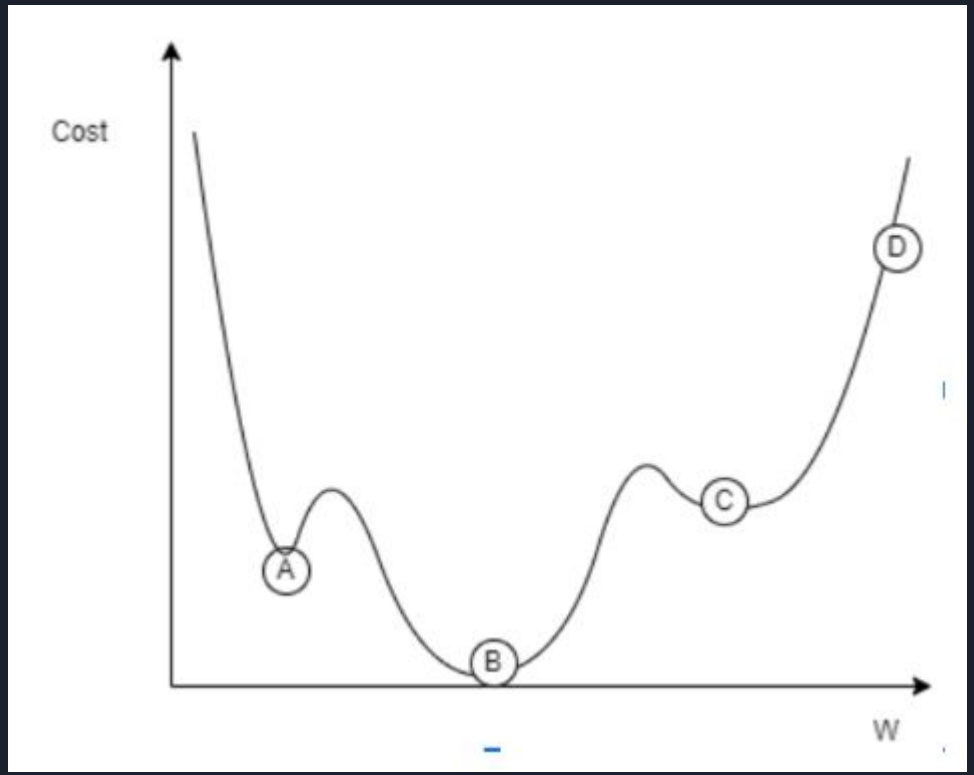
Incremental Gradient Descent can approximate *Batch Gradient Descent* arbitrarily closely if η made small enough

Learning rate

Ponderador según el cual se ajustan los pesos y el bias según un optimizador como el “gradient descent”.

En otras palabras, este ponderador indica que tanto se ajustan los parámetros y luego se aplica el gradient descent sobre los parámetros ajustados hasta volver a alcanzar un mínimo local.



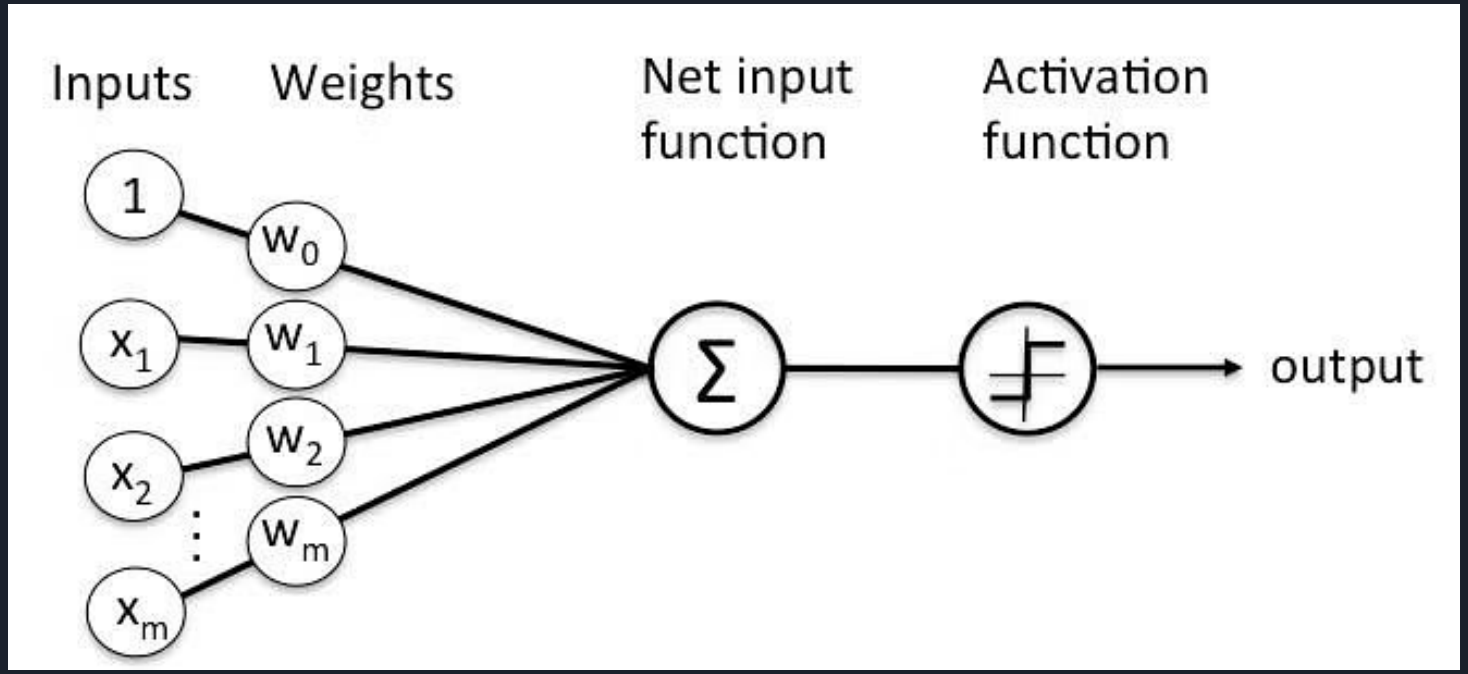

Cost
A
B
C
D
W

Perceptrón

Precursor redes neuronales.

Consta de una capa de entrada y una neurona de salida output.

Se podría decir que su funcionamiento es como la red más simple posible.


Inputs
Weights
Net input function
Activation function
1
x_1
x_2
x_m
w_0
w_1
w_2
w_m
Σ
output

Multilayer perceptron (MLP)

Como dice su nombre, es el resultado de combinar muchos perceptrones formando una red multicapa.

Resuelve problemas más complejos que los que resuelve un perceptrón.

Se puede definir una función de Loss al igual que en un perceptrón, pero se tienen muchos parámetros y se necesita minimizar los costos de todas las neuronas.

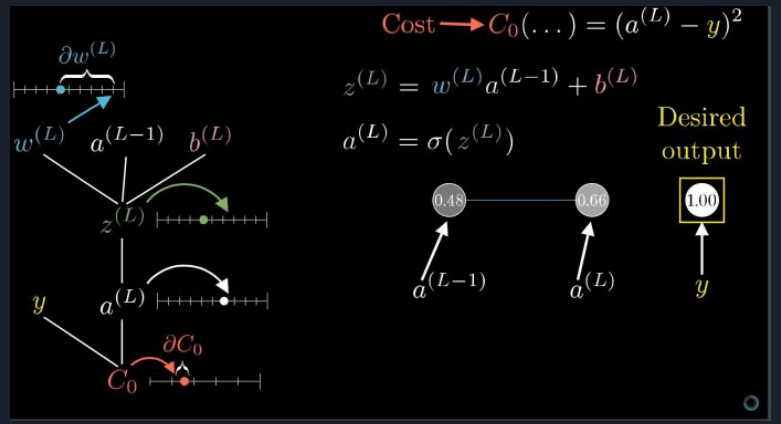
$$J(W) = \frac{1}{2n}\sum_{i=1}^{n}(f(x_i; W) - y_i)^2$$

Backpropagation

Método por el cual se propaga la señal de error por todos los nodos de una MLP (de una red).

Consta de la aplicación sucesiva de la regla de la cadena sobre la función de Loss.

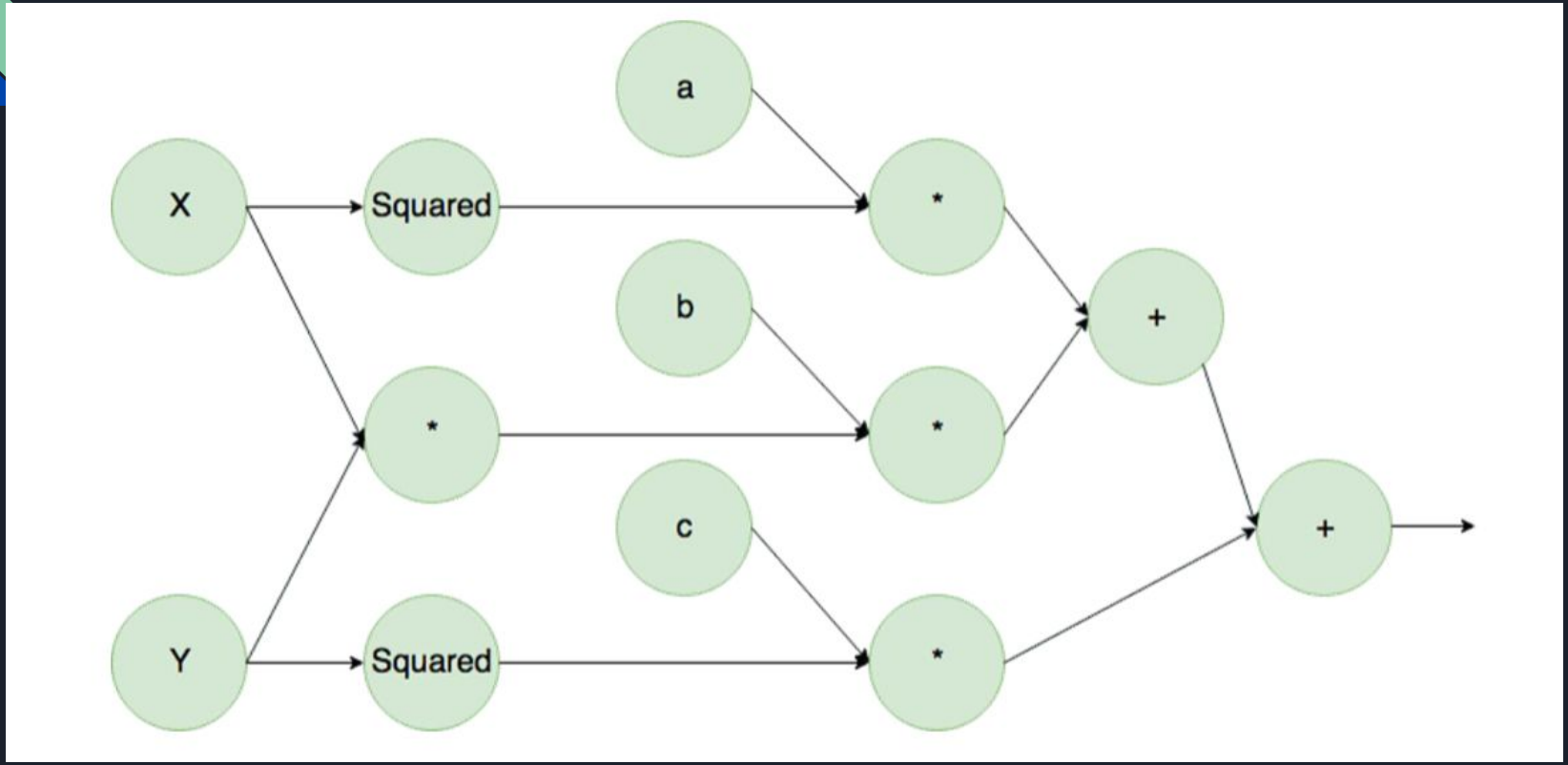
Permite aplicar el método de descenso de gradiente en las neuronas de cualquier red de forma eficaz y así entrenar una red.



Grafo de computo

Representa la secuencia de acciones necesarias para obtener un output.

Estructura y regla de la cadena permiten obtener el gradiente.


a
X
Squared
*
b
+
*
*
c
+
Y
Squared
*