Redes Neuronales

Métodos Numéricos II

Introducción

Definición

Una red neuronal es un modelo computacional inspirado en la estructura y función del cerebro humano. Compuesta por capas de nodos interconectados, cada uno procesa información y transmite señales a través de conexiones ponderadas.

¿Cómo funcionan?

Utilizada en inteligencia artificial, las redes neuronales aprenden patrones no lineales y complejos mediante entrenamiento con datos, permitiendo la toma de decisiones y la resolución de problemas.

¿En qué se utilizan?

En diversas aplicaciones, desde reconocimiento de imágenes, generación de arte hasta procesamiento del lenguaje natural

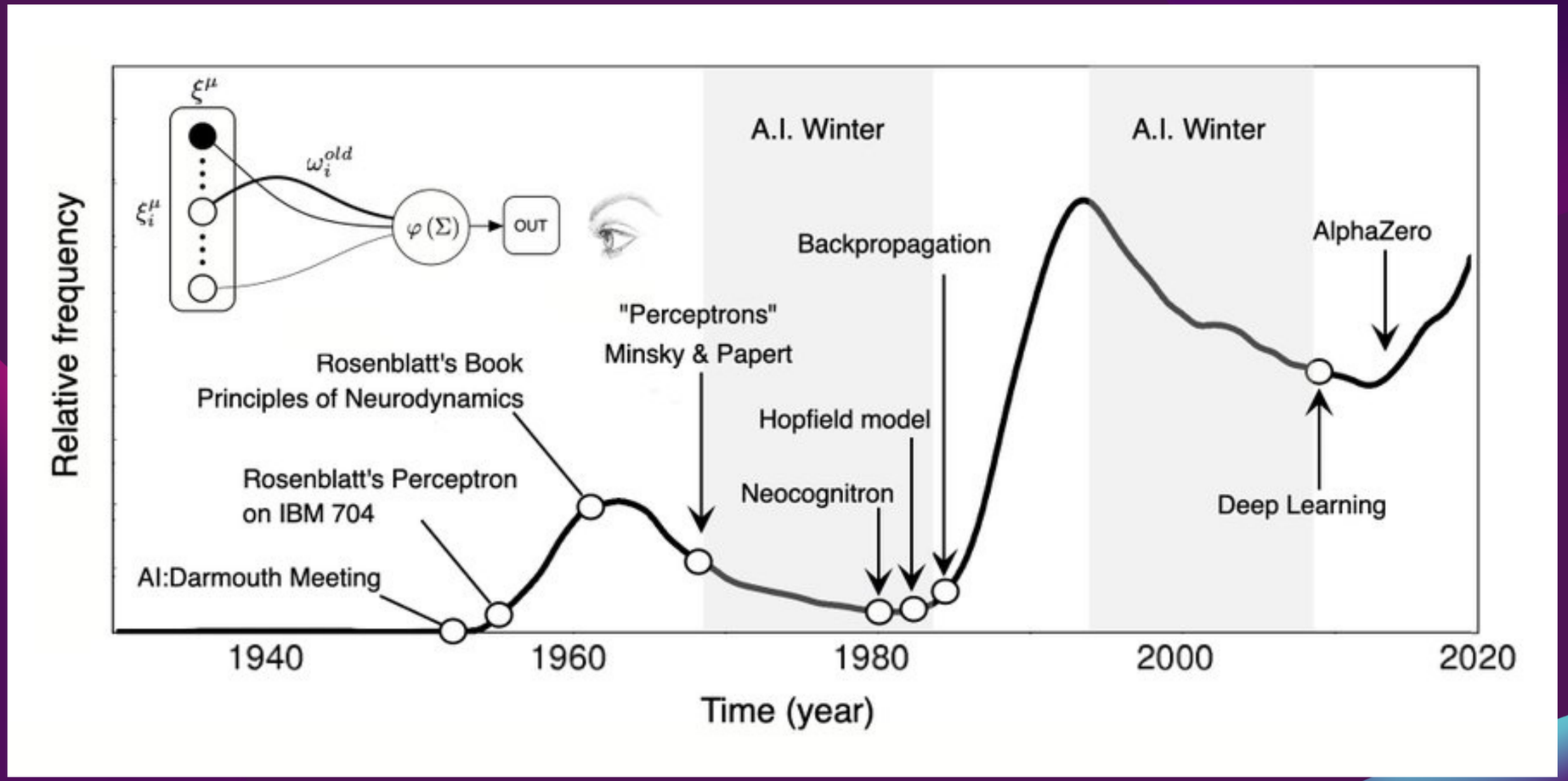
Historia

En la década de 1940, el neurólogo Warren McCulloch y el matemático Walter Pitts colaboraron para desarrollar el primer modelo conceptual de una red neuronal, inspirándose en la estructura y función del sistema nervioso.

En la década de 1950, el psicólogo Frank Rosenblatt creó el "Perceptrón", una forma primitiva de red neuronal de una sola capa capaz de aprender a reconocer patrones simples. Aunque tenía limitaciones para abordar problemas más complejos, marcó el inicio de la aplicación de conceptos neuronales en la informática.

Invierno de la IA

Las limitaciones del Perceptrón se destacaron, y se creía que las redes neuronales tenían restricciones fundamentales. Esto llevó a un período de desinterés conocido como "invierno de la inteligencia artificial" en las décadas de 1970 y 1980, donde otras técnicas predominaron.


A.I. Winter
A.I. Winter
Relative frequency
Time (year)
1940
1960
1980
2000
2020
AI:Darmouth Meeting
Rosenblatt's Perceptron
on IBM 704
Rosenblatt's Book
Principles of Neurodynamics
"Perceptrons"
Minsky & Papert
Neocognitron
Hopfield model
Backpropagation
Deep Learning
AlphaZero
OUT

Defense systems

Unmanned systems

Robotics

Spacecraft

Neural networks hardware

Internet of Things

Pattern recognition

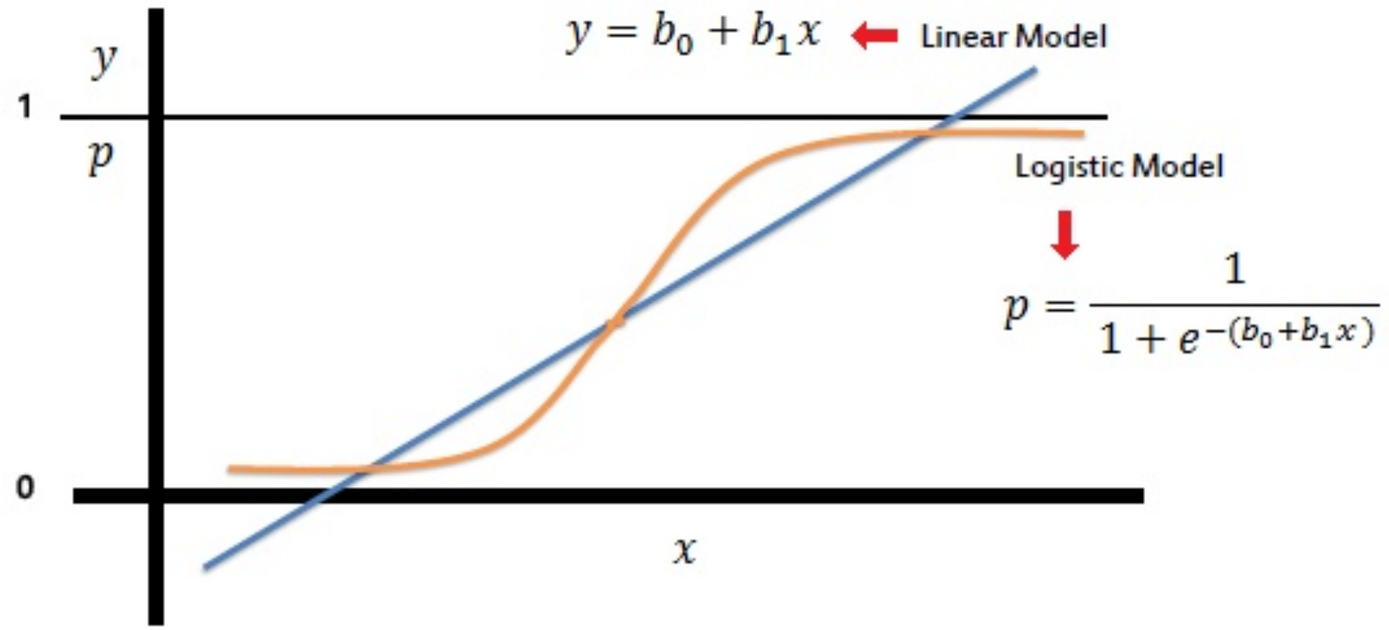
Transport control

Air Force

02

Regresión Logística

Regresión Logística



La función de regresión logística es una función matemática utilizada en estadísticas y aprendizaje automático para modelar la probabilidad de que una variable pertenezca a una categoría particular. A menudo, se emplea en problemas de clasificación binaria, donde la variable dependiente puede tomar solo dos valores, como 0 o 1

Forma General

La función de regresión logística transforma una combinación lineal de variables predictoras (características) mediante la función logística, también conocida como función sigmoide. La forma general de la función de regresión logística es la siguiente:

$$z = w_1 x_1 + w_2 x_2 + b \qquad \sigma(z) = \frac{1}{1+e^{-z}}$$

Donde:

x = variables
w = pesos
b = desplazamiento

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

Clasificación

$$= -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

La función de pérdida logarítmica, también conocida como log loss o cross-entropy loss, es una medida comúnmente utilizada en problemas de clasificación para evaluar la calidad de un modelo de clasificación de probabilidades. Es particularmente útil cuando se trabaja con modelos que generan probabilidades para cada clase en lugar de simplemente predecir la clase.

04

Función de Pérdida

La función de pérdida, también conocida como función objetivo o función de costo, mide la discrepancia entre las predicciones de un modelo y los valores reales del conjunto de datos durante el entrenamiento de una red neuronal. El objetivo durante el entrenamiento es minimizar esta función, ajustando los parámetros del modelo para mejorar la precisión de las predicciones.

Problemas de regresión

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

Mean Squared Error (MSE)

Ventajas

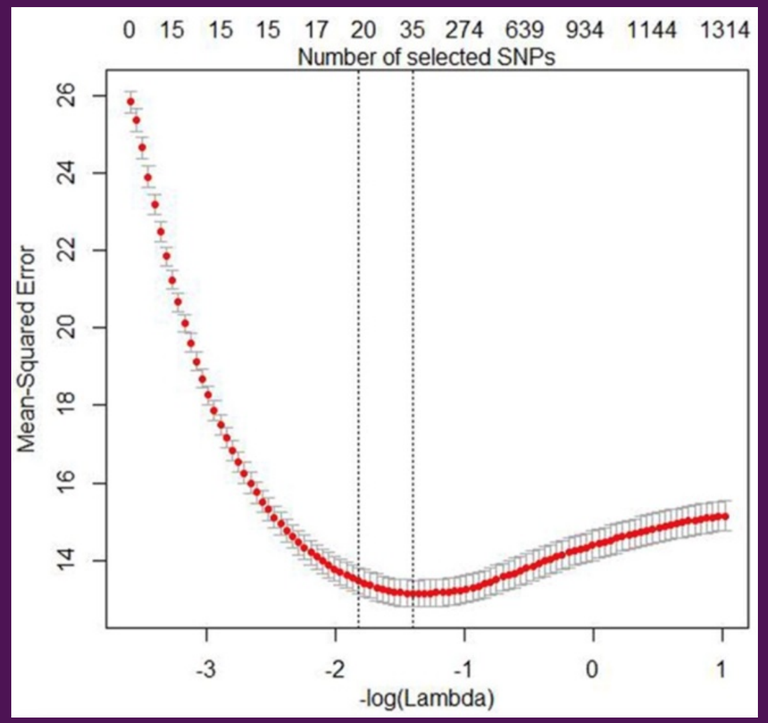
Penaliza los errores más grandes.

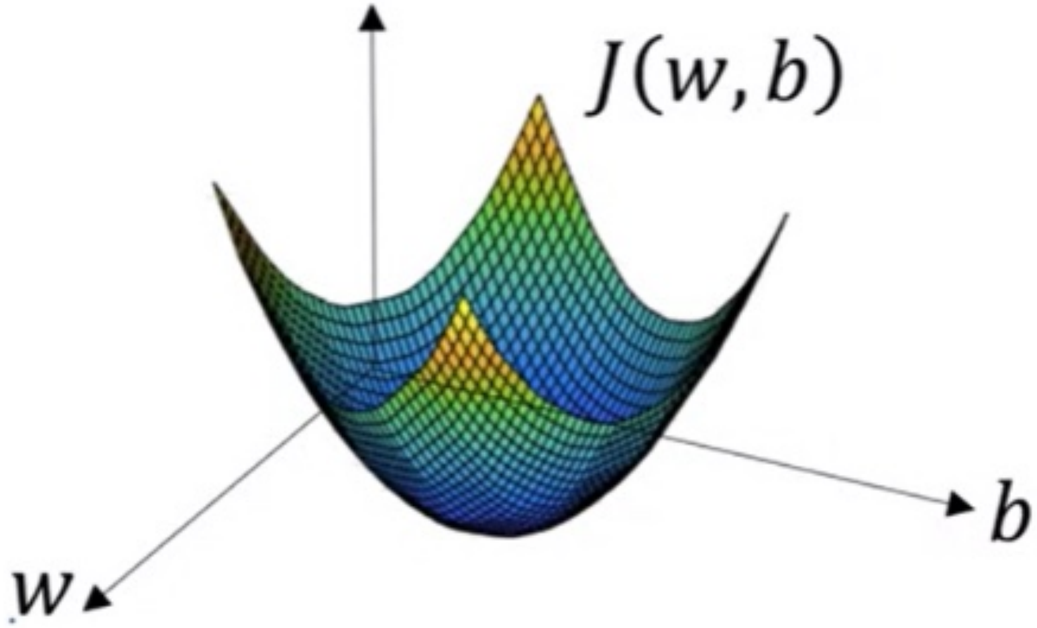
Es convexa.

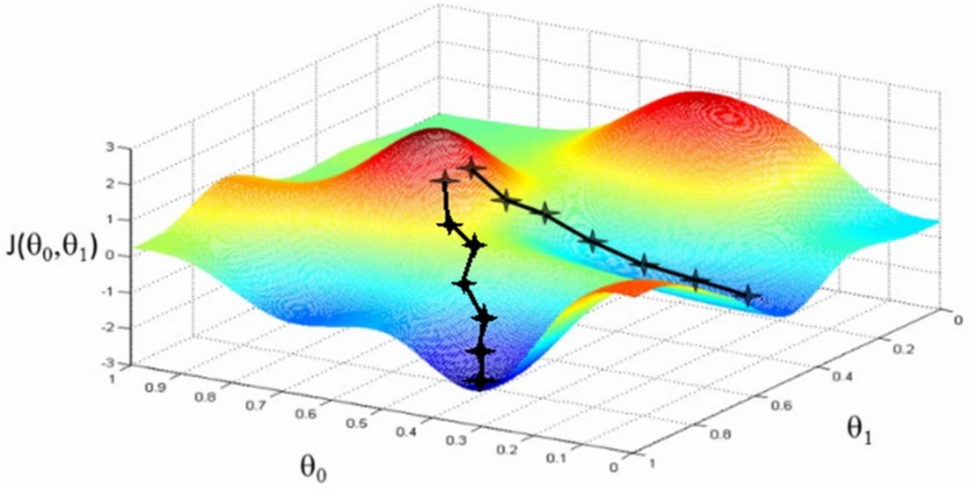
Desventajas

Sensible a valores atípicos

Problemas en predicciones en 0 en el gradiante





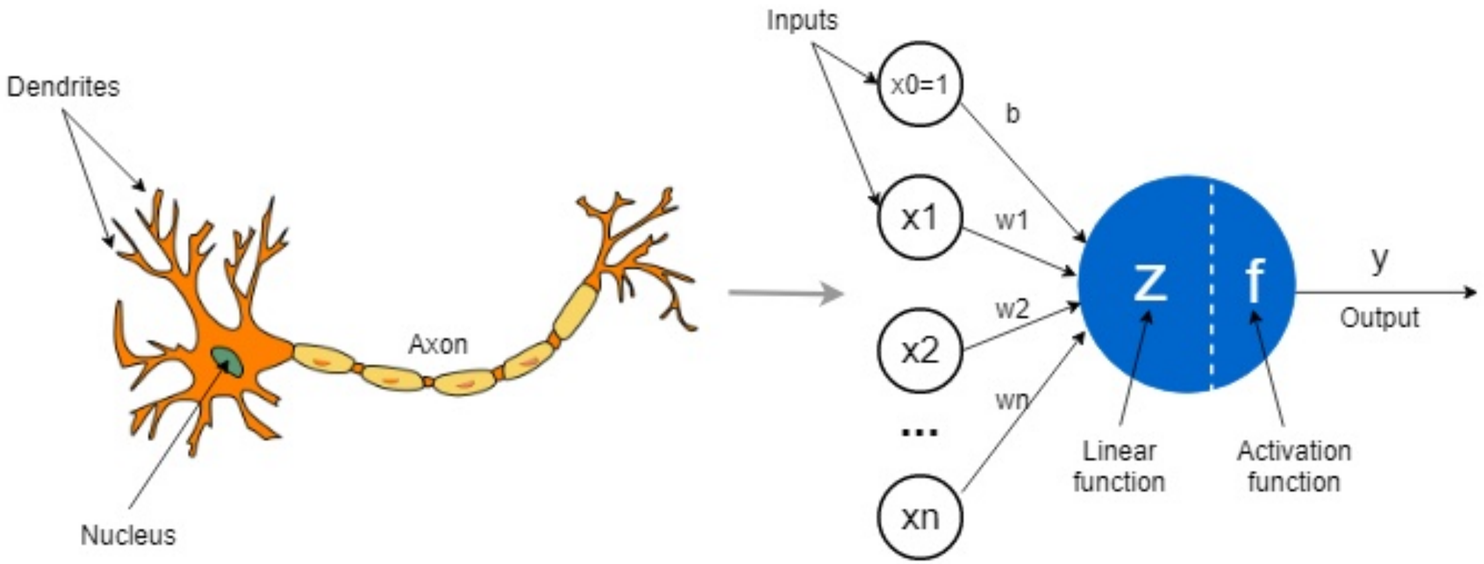


$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

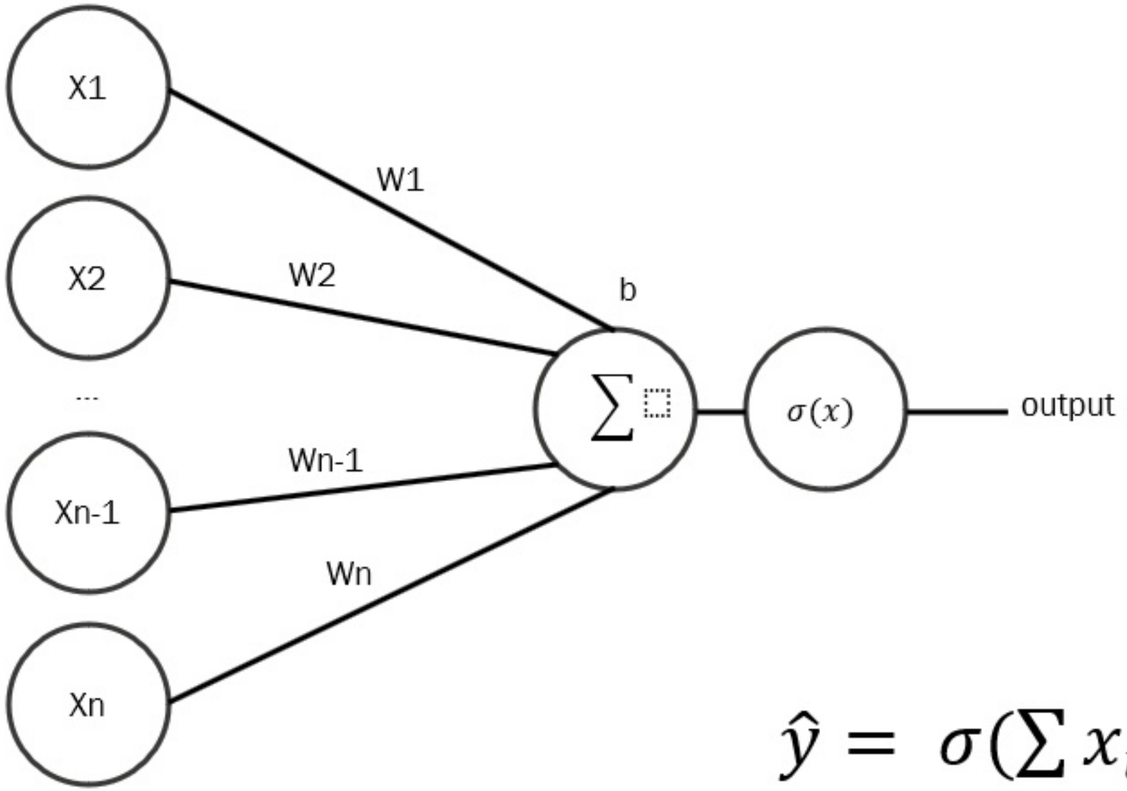
03

Redes Neuronales

Forward y Backward propagation

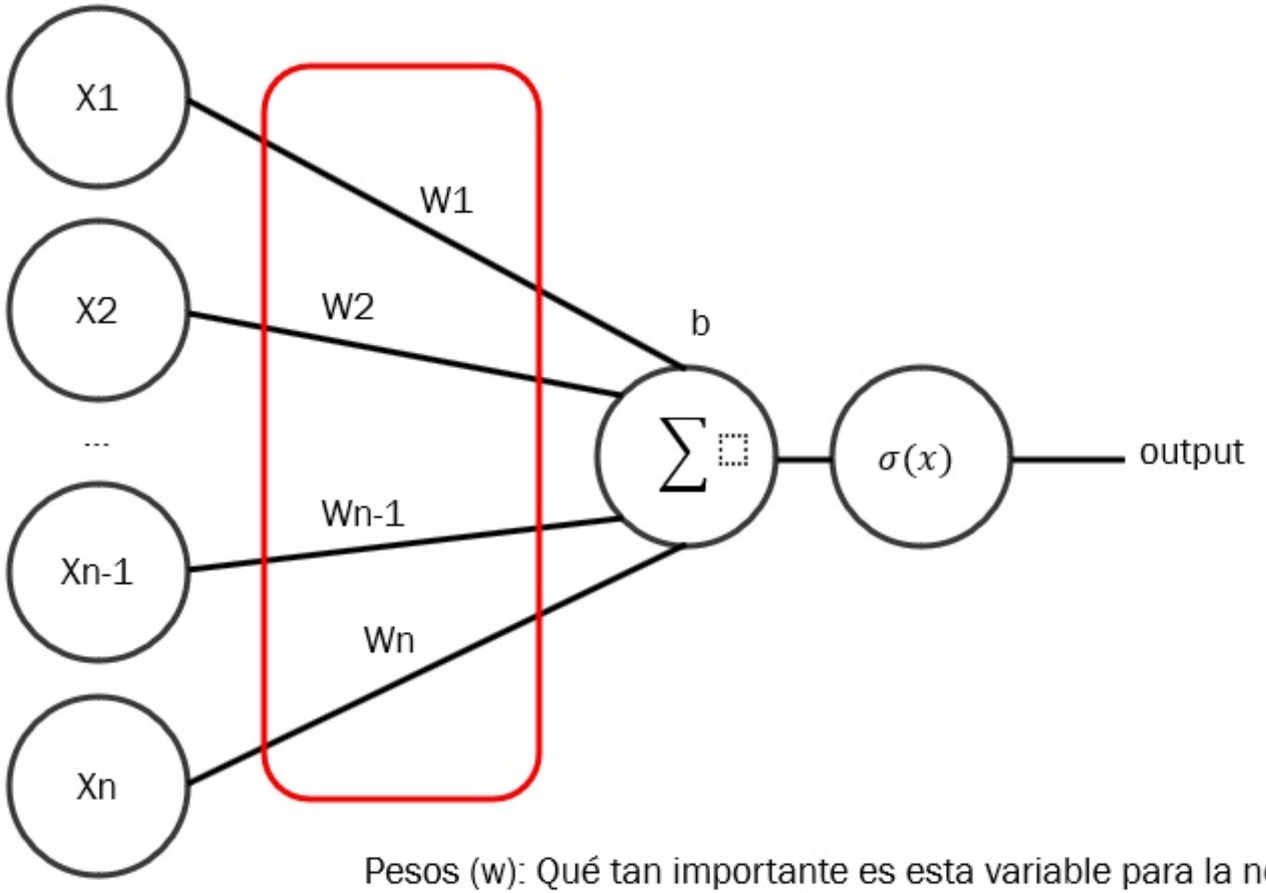

Dendrites
Axon
Nucleus
Inputs
x0=1
b
x1
w1
x2
w2
...
wn
xn
z
f
y
Output
Linear function
Activation function

Perceptrón



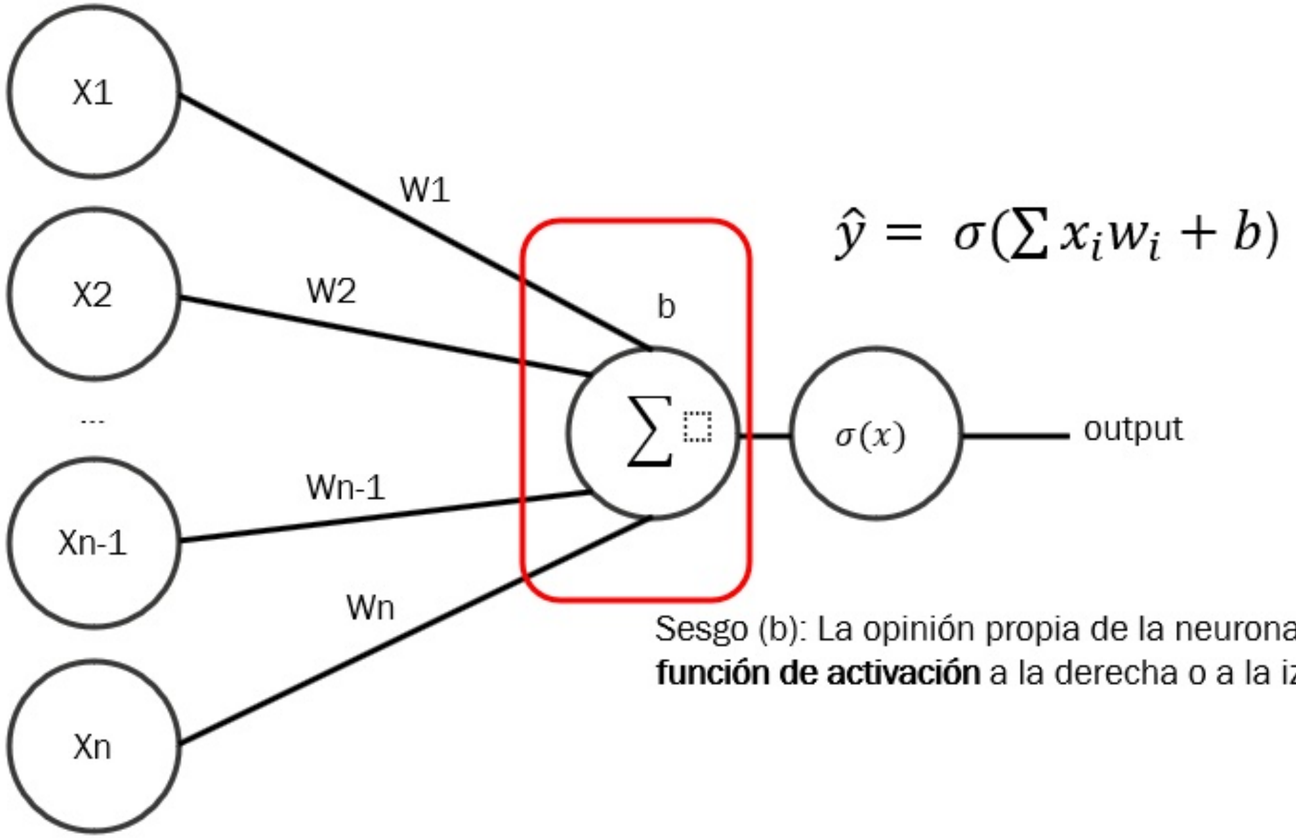
$$\hat{y} = \sigma(\sum x_i w_i + b)$$

Perceptrón



Pesos (w): Qué tan importante es esta variable para la neurona?

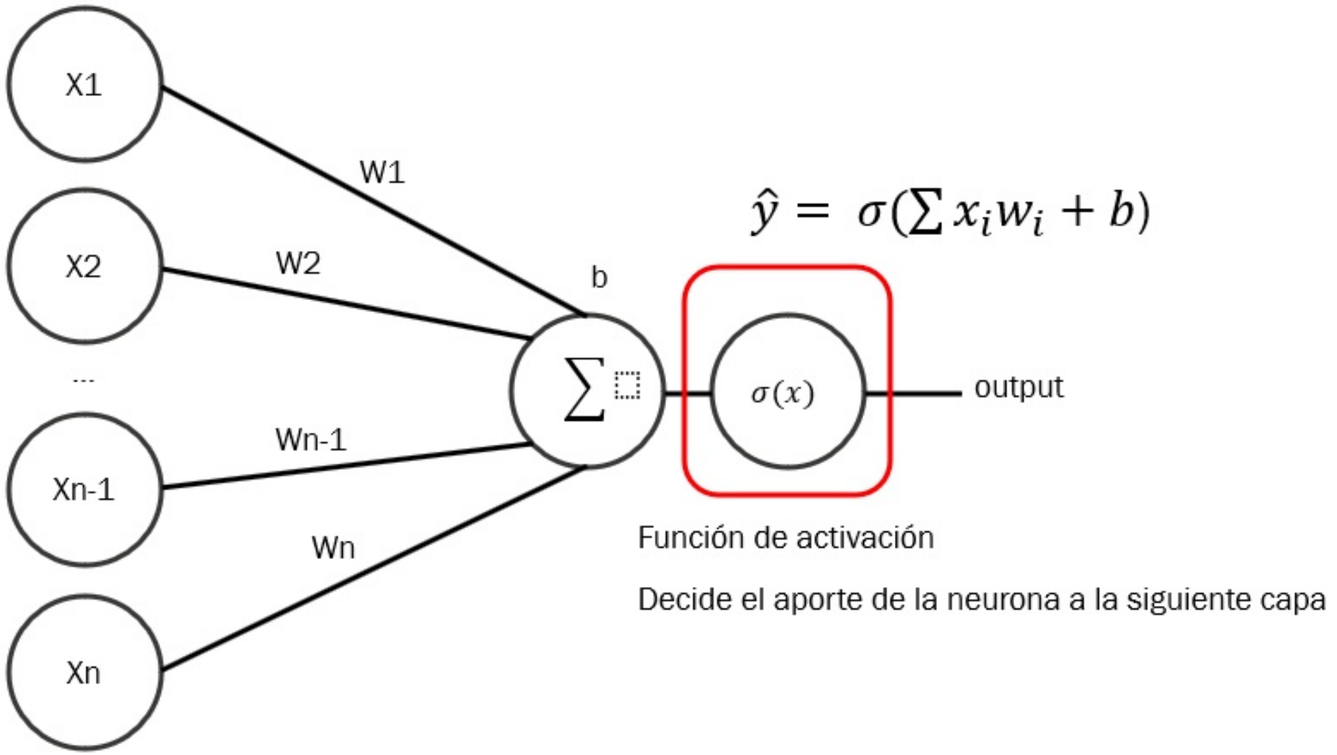
Perceptrón



$$\hat{y} = \sigma(\sum x_i w_i + b)$$

Sesgo (b): La opinión propia de la neurona. Puede mover la **función de activación** a la derecha o a la izquierda.

Perceptrón



Función de activación

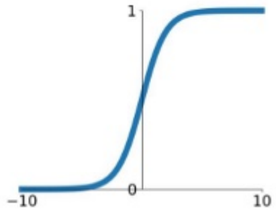
Decide el aporte de la neurona a la siguiente capa

Funciones de Activación

Las funciones de activación introducen no linealidades en las redes neuronales, lo cual es crucial para que la red pueda aprender patrones y relaciones no lineales en los datos. Permiten a las redes neuronales aprender representaciones jerárquicas y complejas de los datos.

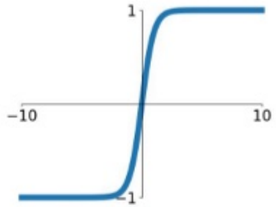
Sigmoid

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



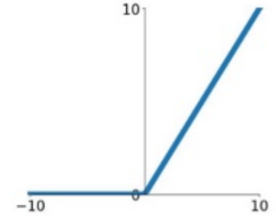
tanh

$$\tanh(x)$$



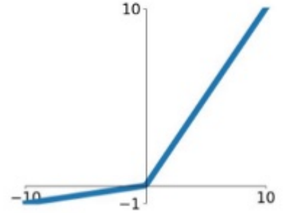
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

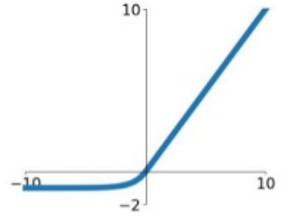


Maxout

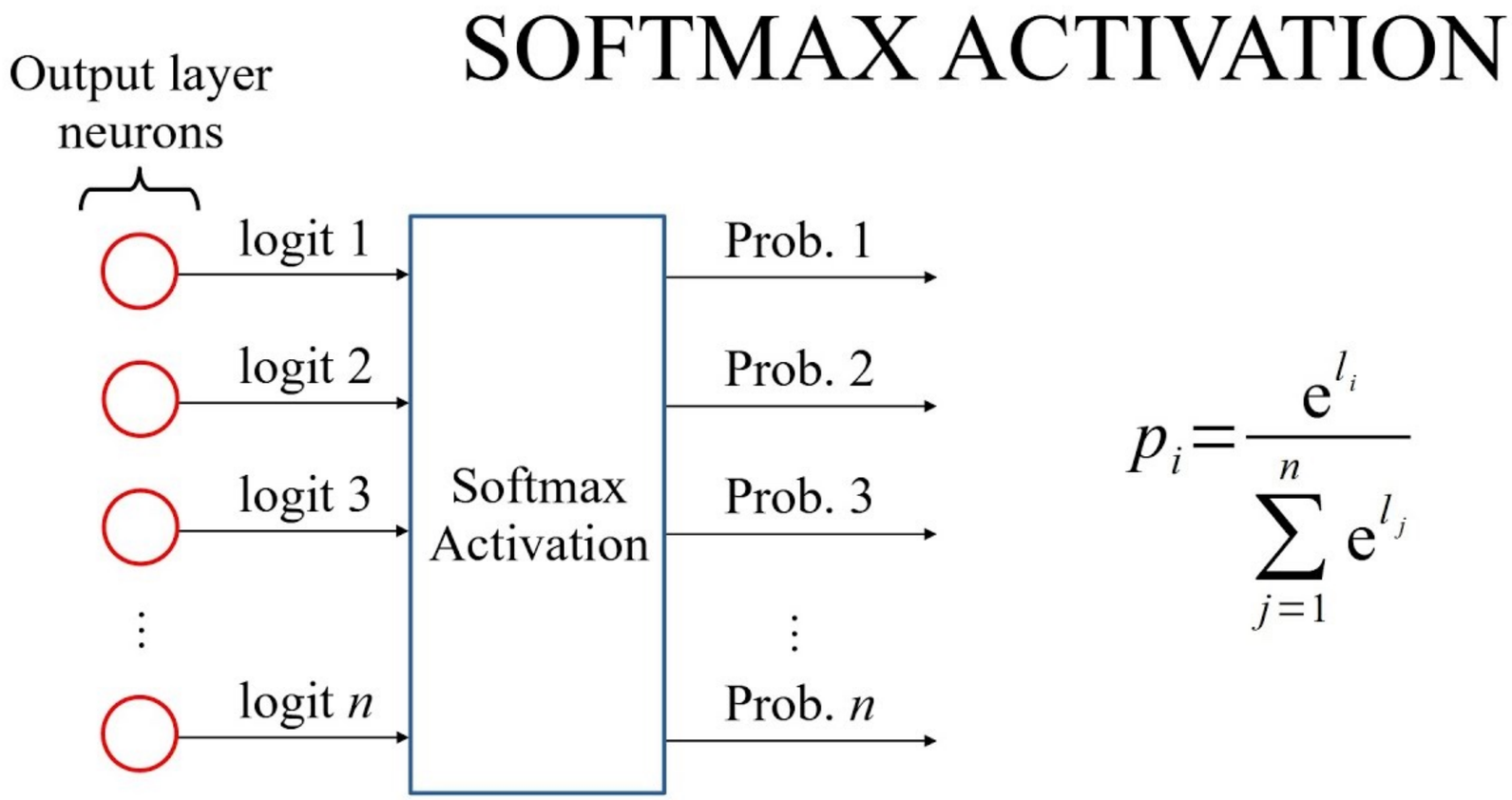
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

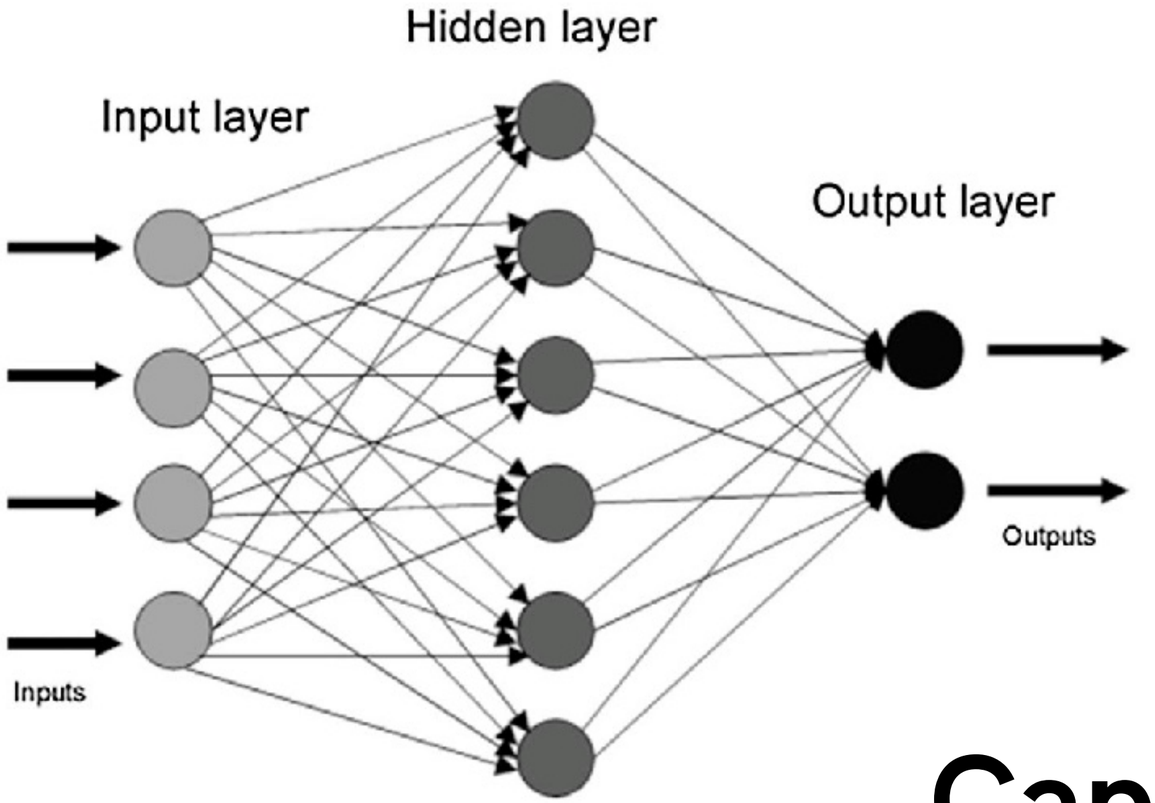
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



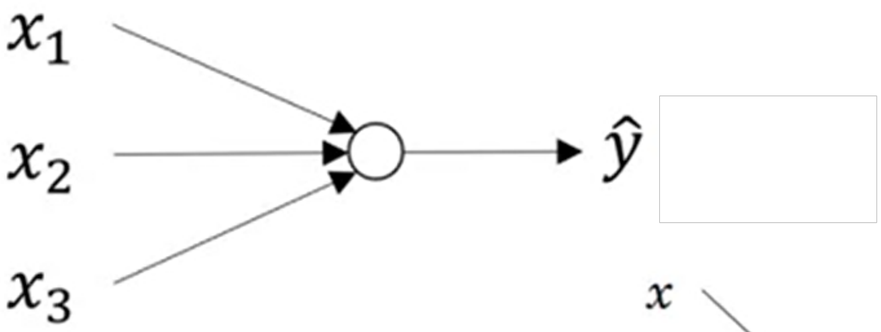
Funciones de Activación

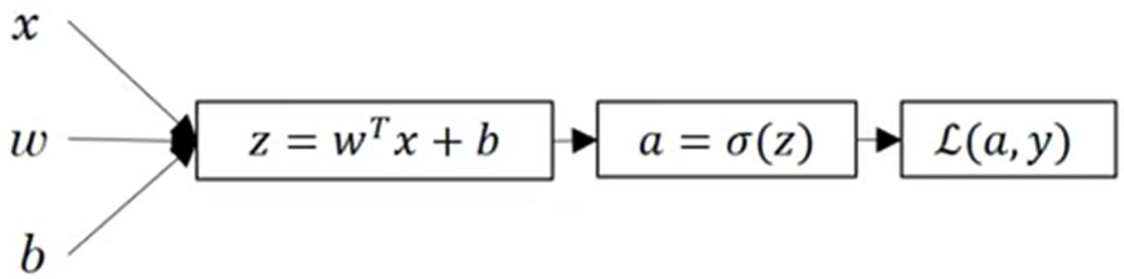




Capas

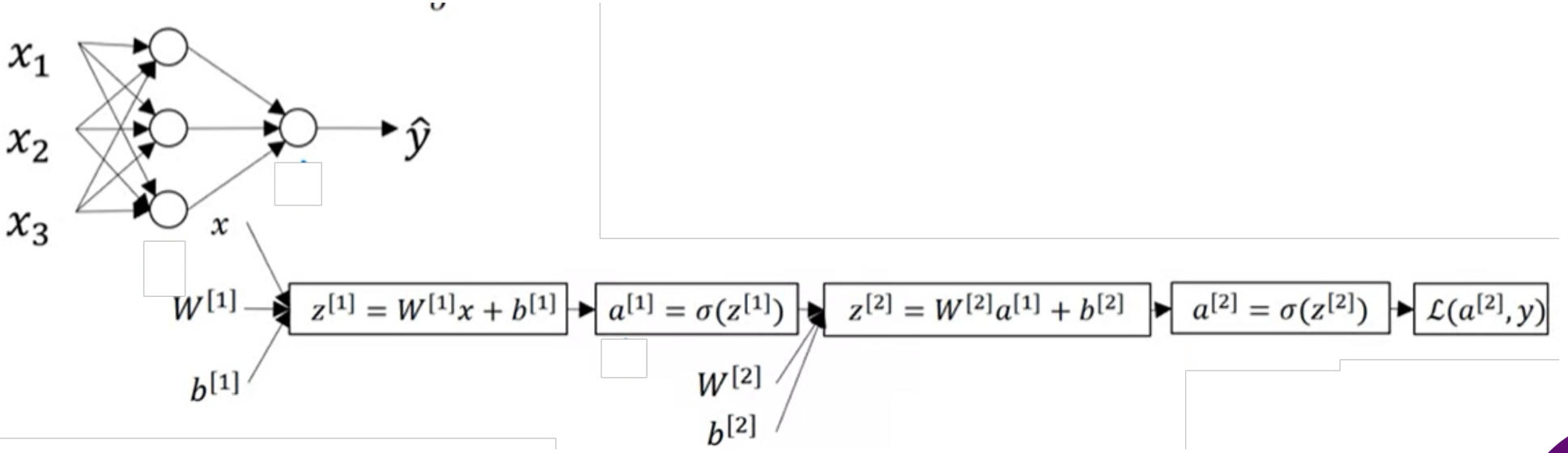
Las capas de una red neuronal son unidades estructurales que procesan la información durante el entrenamiento. La capa de entrada recibe los datos, las capas ocultas realizan cálculos intermedios, y la capa de salida produce las predicciones o resultados finales.



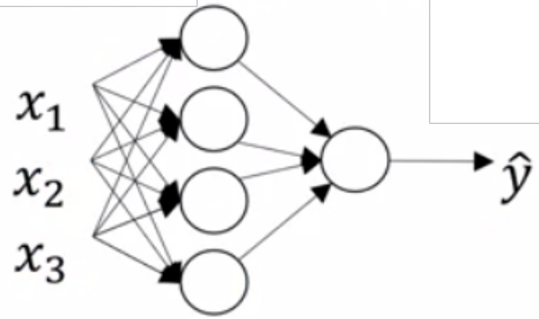


Feed Forward

Feed Forward 2 Capas



Feed Forward 2 Capas



$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

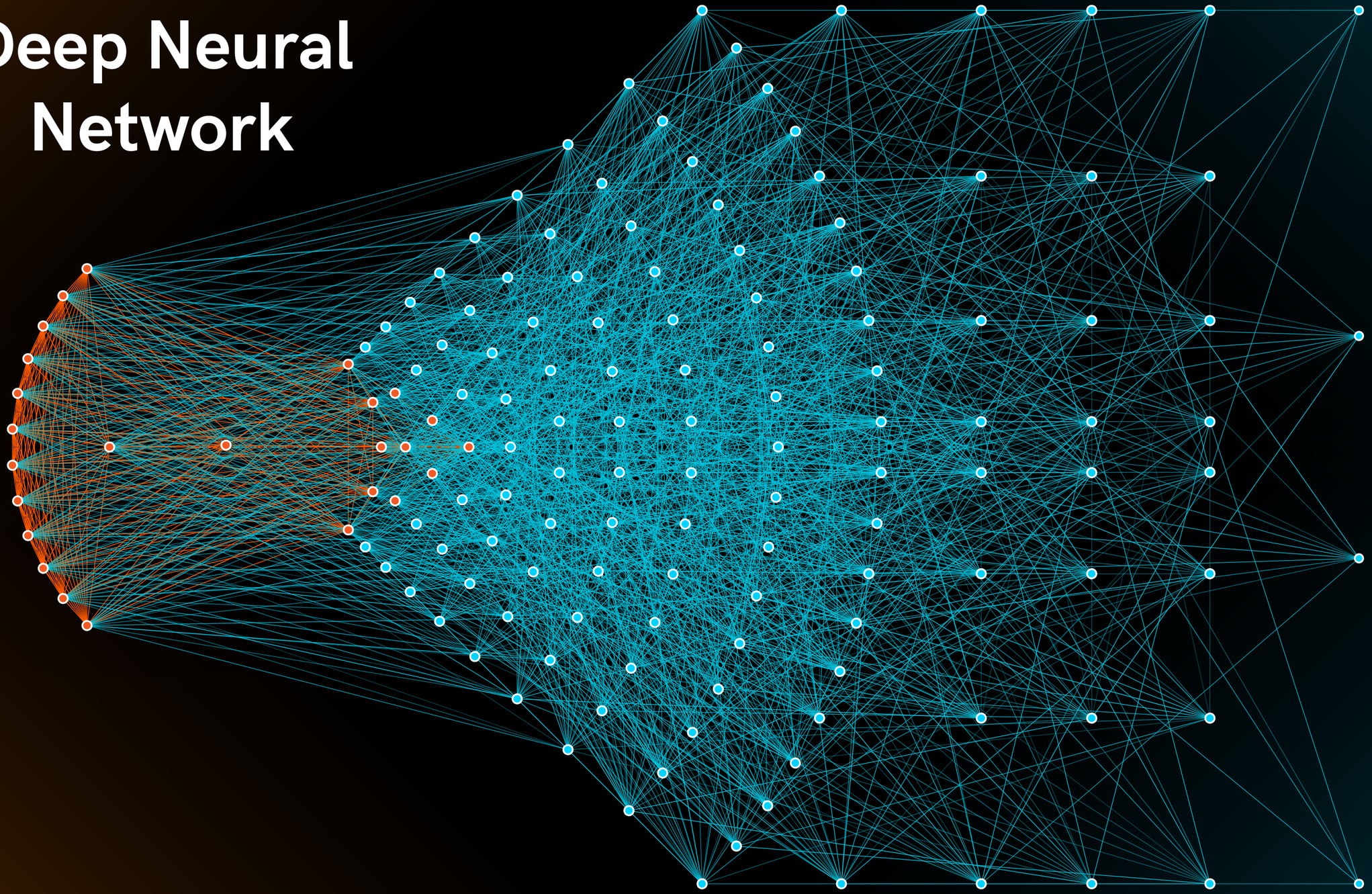
$$A^{[1]} = \begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \dots & a^{[1](m)} \\ | & | & & | \end{bmatrix}$$

```
for i = 1 to m
```

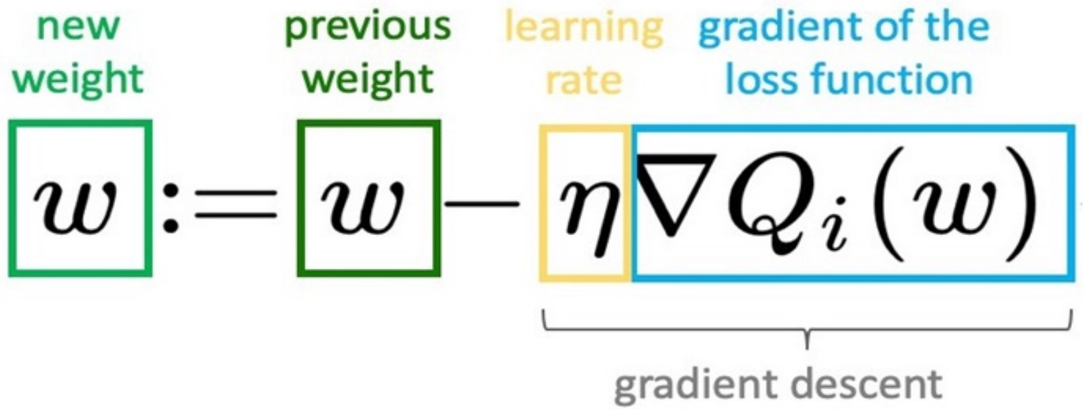
$$z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$$
$$a^{[1](i)} = \sigma(z^{[1](i)})$$
$$z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$$
$$a^{[2](i)} = \sigma(z^{[2](i)})$$

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$
$$A^{[1]} = \sigma(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = \sigma(Z^{[2]})$$

Deep Neural Network



Back Propagation



$$w := w - \alpha \frac{dJ(w,b)}{dw} = w - \alpha \frac{\partial J(w,b)}{\partial w}$$

$$b := b - \alpha \frac{dJ(w,b)}{db} = b - \alpha \frac{\partial J(w,b)}{\partial b}$$

$$\frac{\partial J}{\partial z} = \frac{\partial J}{\partial a} \times \frac{\partial a}{\partial z} = \frac{a-y}{a(1-a)} \times a(1-a) = a - y$$

Finalmente para calcular $\frac{\partial J}{\partial w_i}$ y $\frac{\partial J}{\partial b}$ sabemos que:
Para w_i:

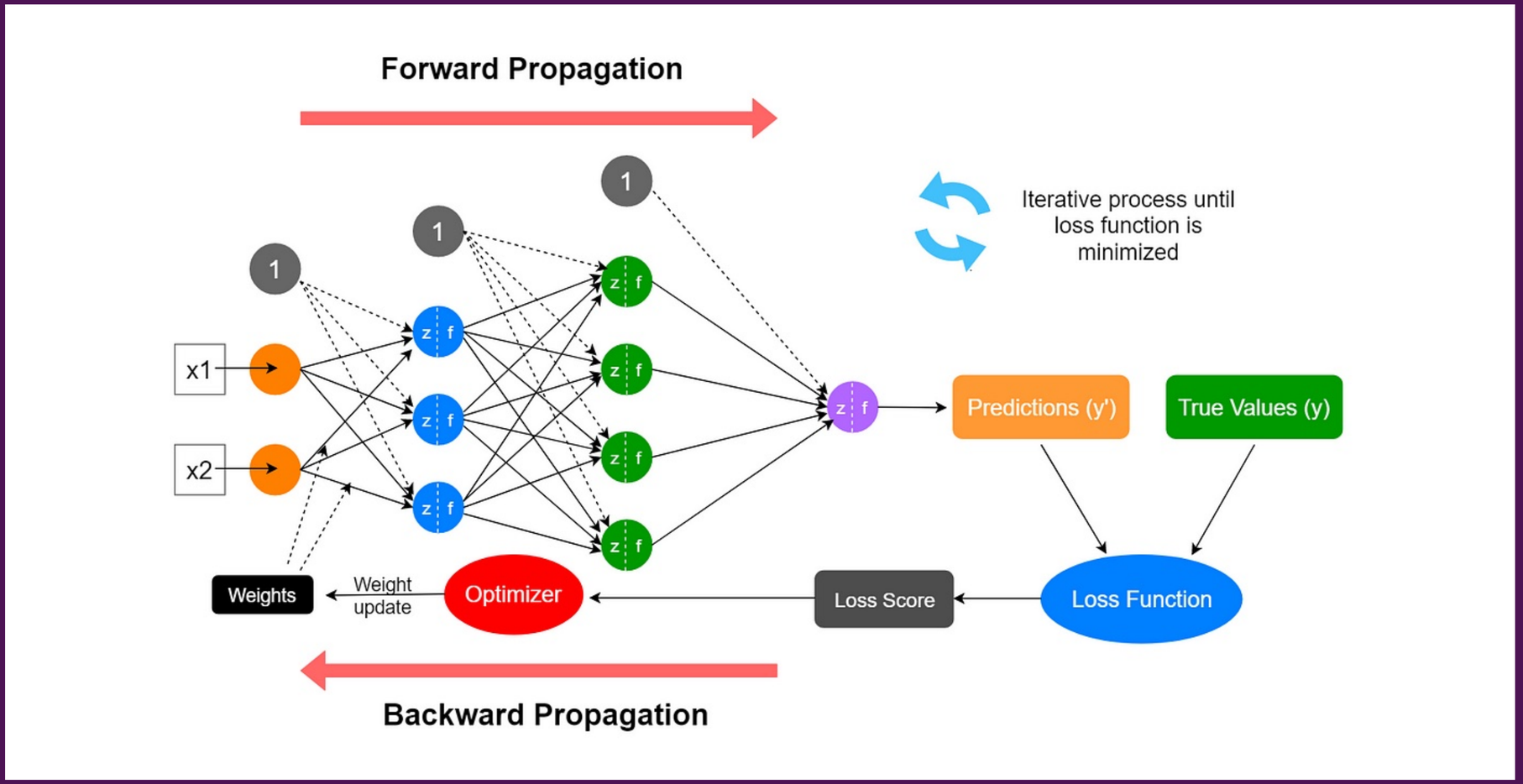
$$\frac{\partial J}{\partial w_i} = \frac{\partial J}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w_i}$$

Calculemos : $\frac{\partial w_i}{\partial z} = x_1$ por lo que: $\frac{\partial J}{\partial w_i} = x_1 \times \partial z = x_1 \times (a - y)$

Para b:

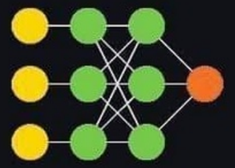
$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial b}$$

Calculemos : $\frac{\partial b}{\partial z} = 1$ por lo que: $\frac{\partial J}{\partial b} = 1 \times \partial z = 1 \times (a - y)$


Forward Propagation
1
1
1
x1
x2
z f
z f
z f
z f
z f
z f
z f
Iterative process until loss function is minimized
Predictions (y')
True Values (y)
Loss Function
Loss Score
Optimizer
Weight update
Weights
Backward Propagation

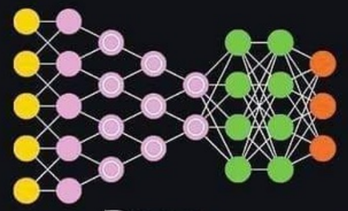
TYPES OF NEURAL NETWORKS

Deep-Feed Forward (DFF)

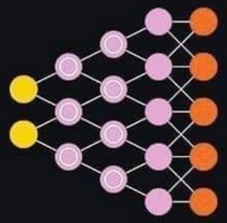


Support Vector Machine (SVM)

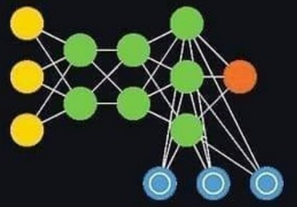
Boltzmann Machine (BM)



Deep Convolutional Network (DCN)

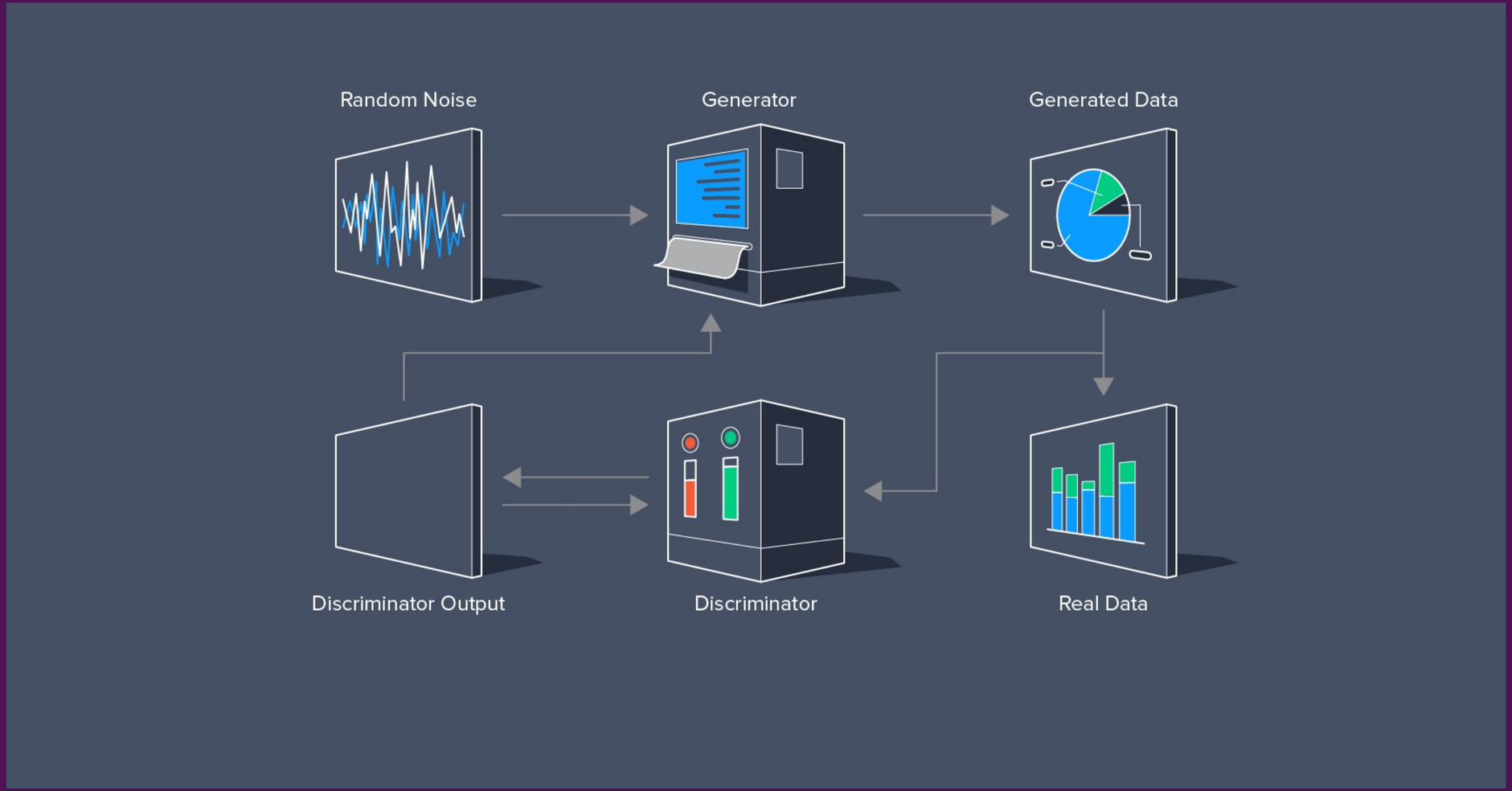


Deconvolutional Network (DN)

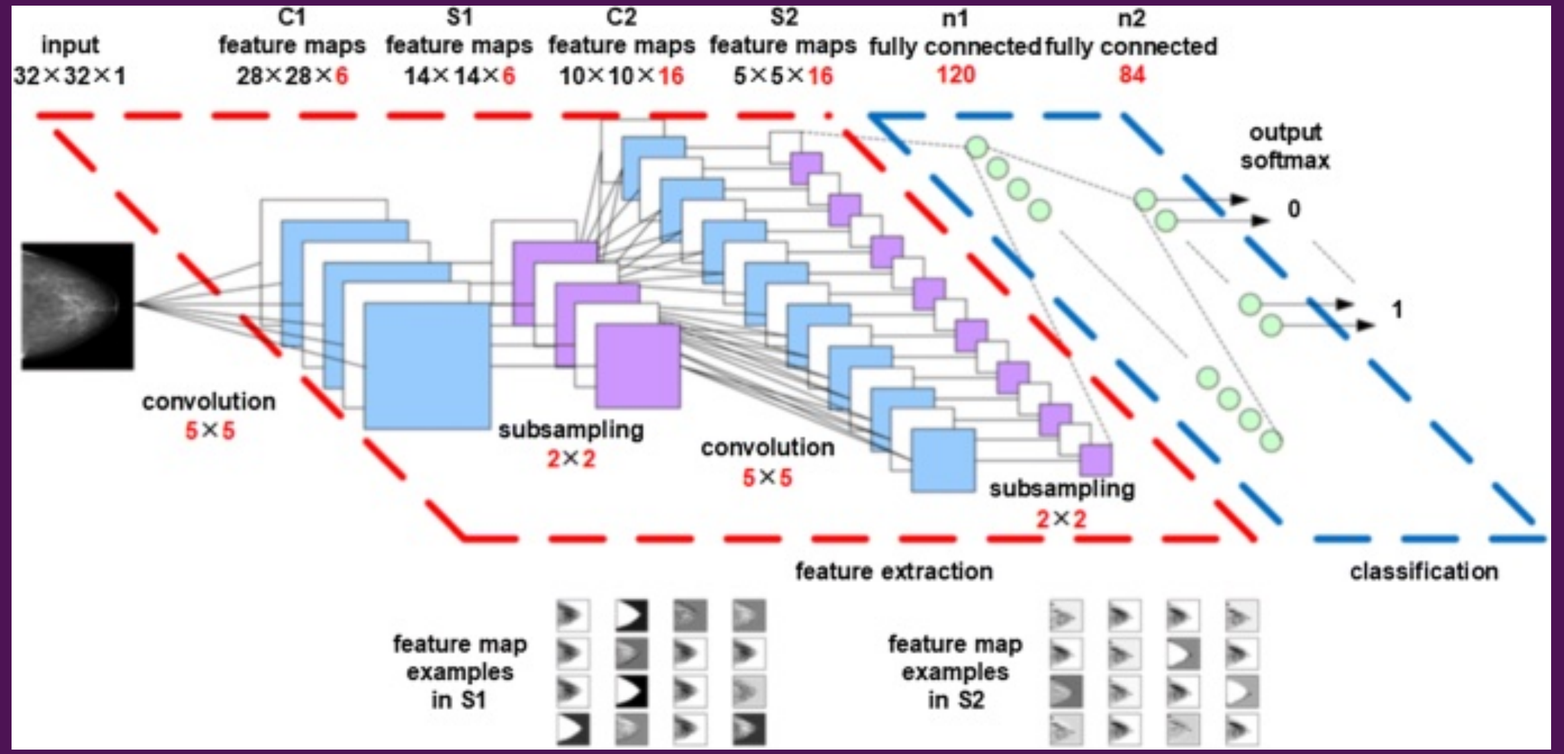


Neural Turing Machine (NTM)

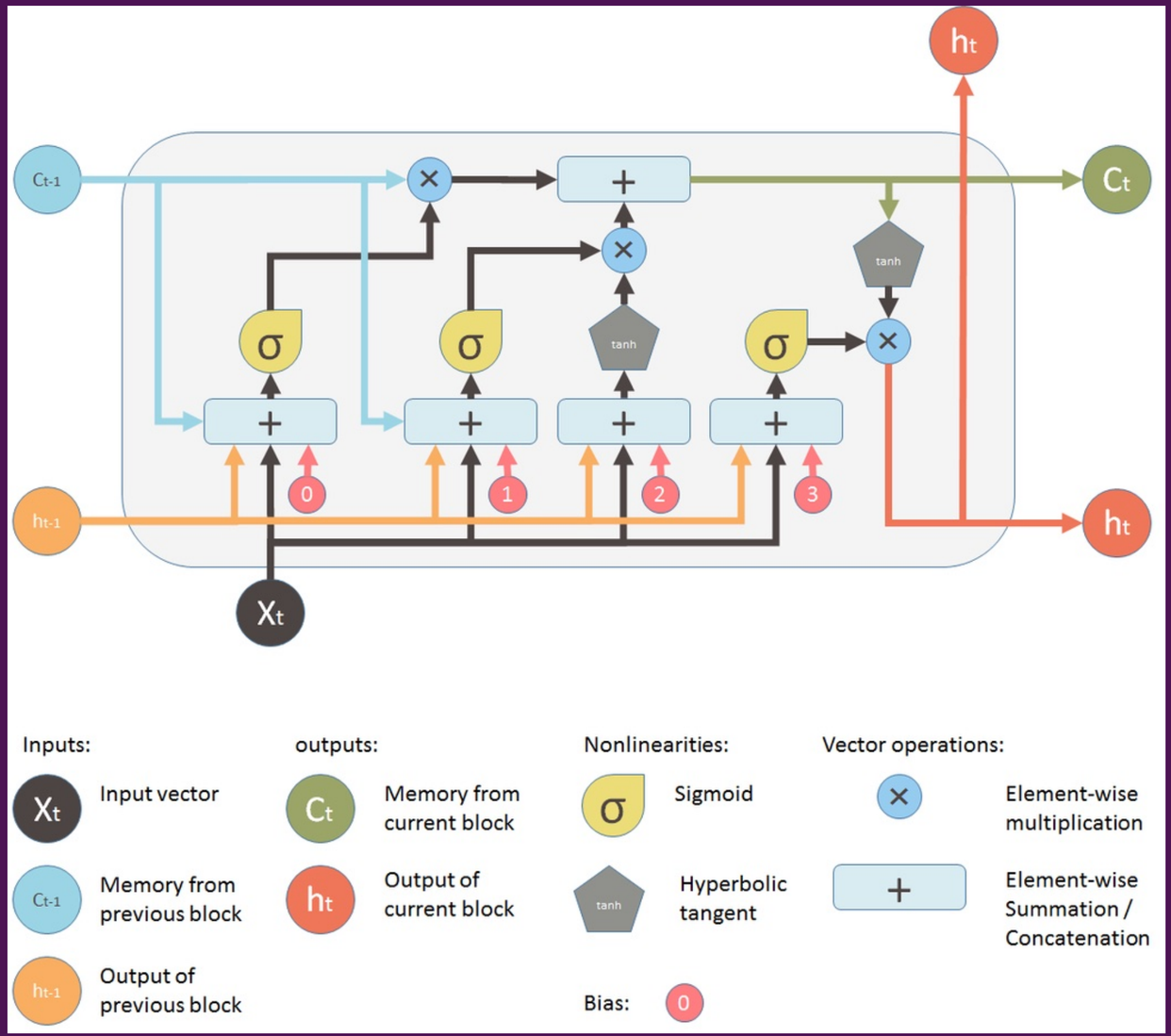
GAN


Random Noise
Generator
Generated Data
Discriminator Output
Discriminator
Real Data

Convolutional NN

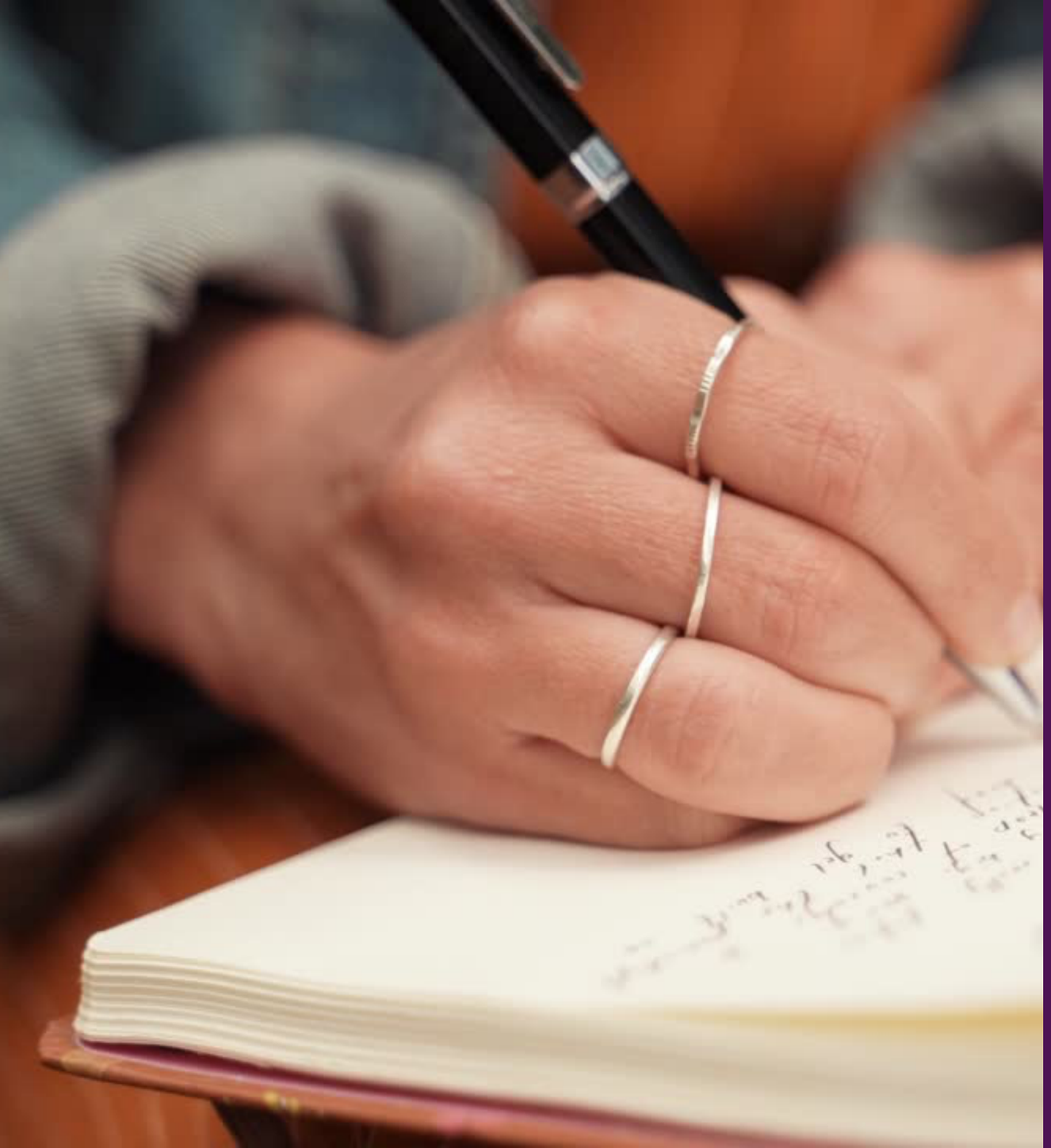


LSTM

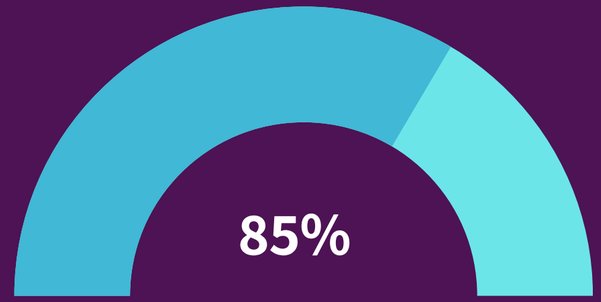

ht
Ct-1
Ct
tanh
σ
σ
tanh
σ
tanh
+
+
+
+
0
1
2
3
ht-1
ht
Xt
Inputs:
Input vector
Memory from previous block
Output of previous block
outputs:
Memory from current block
Output of current block
Nonlinearities:
Sigmoid
Hyperbolic tangent
Bias:
Vector operations:
Element-wise multiplication
Element-wise Summation / Concatenation

Optimizadores

Optimizer	State Memory [bytes]	# of Tunable Parameters	Strengths	Weaknesses
SGD	0	1	Often best generalization (after extensive training)	Prone to saddle points or local minima Sensitive to initialization and choice of the learning rate α
SGD with Momentum	$4n$	2	Accelerates in directions of steady descent Overcomes weaknesses of simple SGD	Sensitive to initialization of the learning rate α and momentum β
AdaGrad	$\sim 4n$	1	Works well on data with sparse features Automatically decays learning rate	Generalizes worse, converges to sharp minima Gradients may vanish due to aggressive scaling
RMSprop	$\sim 4n$	3	Works well on data with sparse features Built in Momentum	Generalizes worse, converges to sharp minima
Adam	$\sim 8n$	3	Works well on data with sparse features Good default settings Automatically decays learning rate α	Generalizes worse, converges to sharp minima Requires a lot of memory for the state
AdamW	$\sim 8n$	3	Improves on Adam in terms of generalization Broader basin of optimal hyperparameters	Requires a lot of memory for the state
LARS	$\sim 4n$	3	Works well on large batches (up to 32k) Counteracts vanishing and exploding gradients Built in Momentum	Computing norm of gradient for each layer can be inefficient



Calificaciones de Ensayos



Caso Práctico

Estructura del Problema

Prompt

Egyptian society was structured like a pyramid. At the top were the gods, such as Ra, Osiris, and Isis. Egyptians believed ...

Pregunta

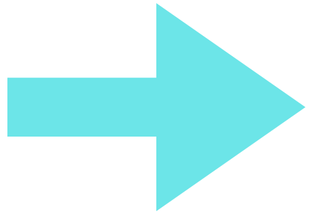
In complete sentences, summarize the structure of the ancient Egyptian system of government. How were different social classes ...

Ensayo

In Egypt, there were many occupations and social classes involved in day-to-day living. In many instances if you were at ...

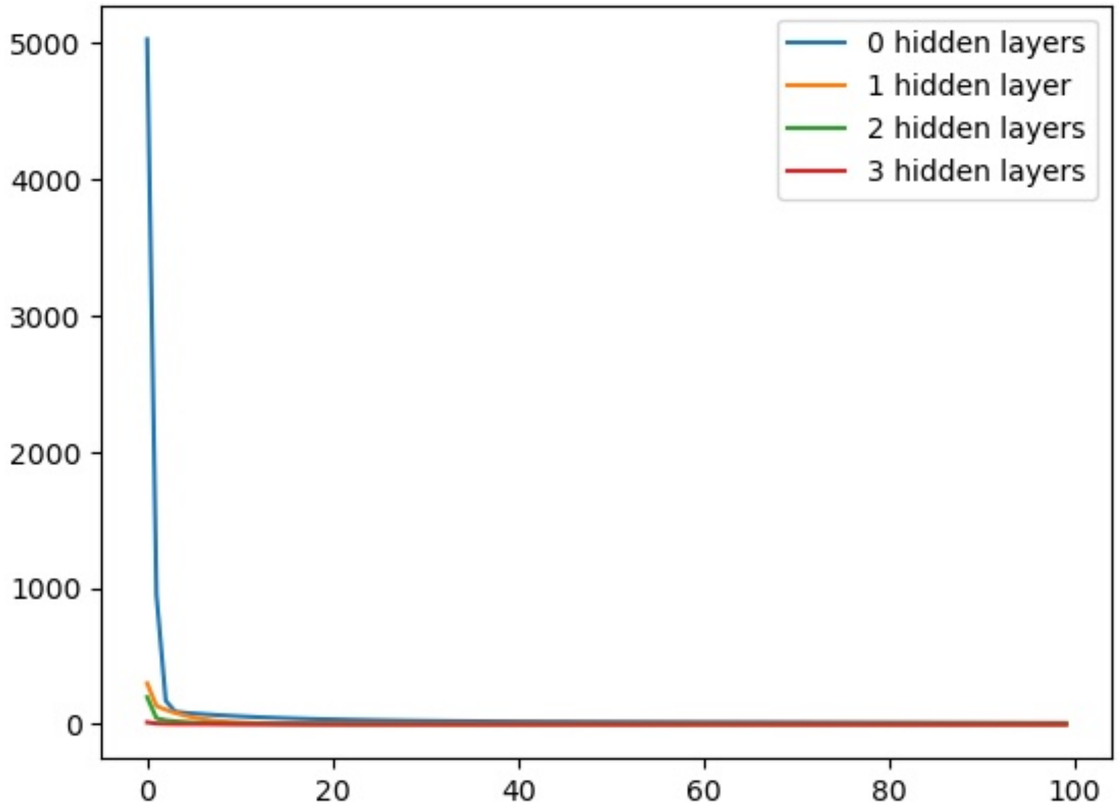
Calculamos 82 variables ...

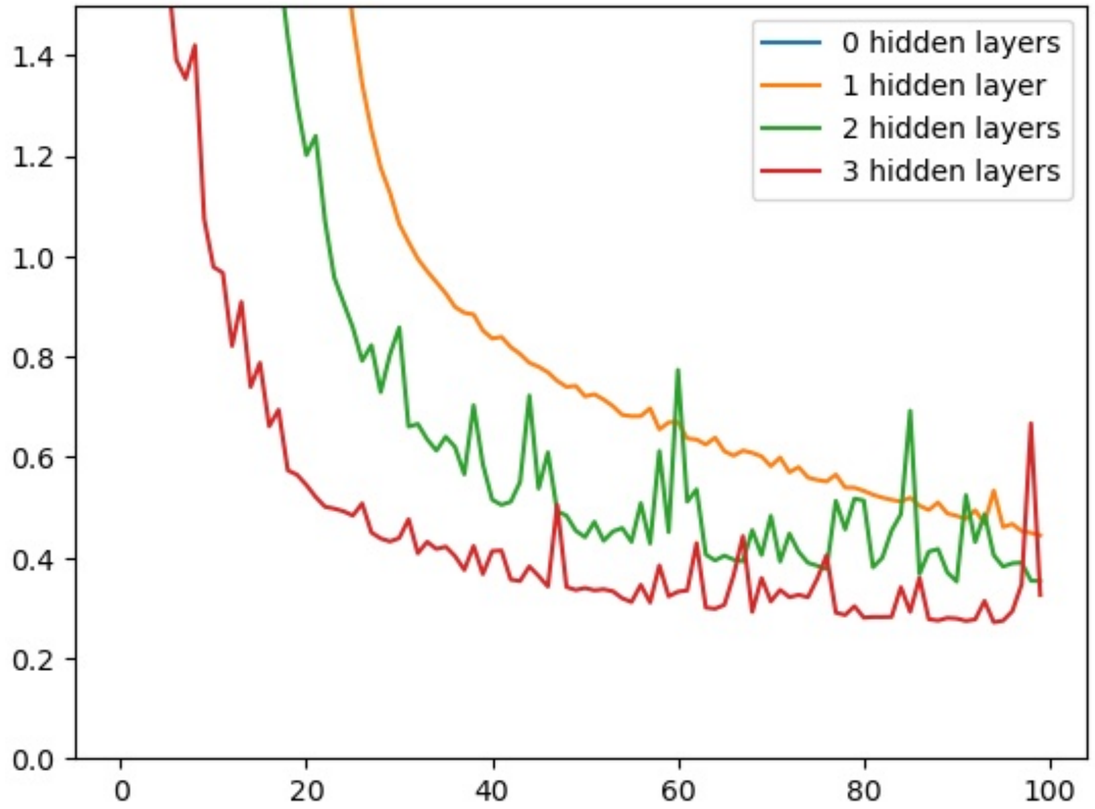
1. Caraterísticas del prompt
2. Características de la pregunta
3. Caracteísticas del ensayo
4. Relación entre el ensayo y el prompt



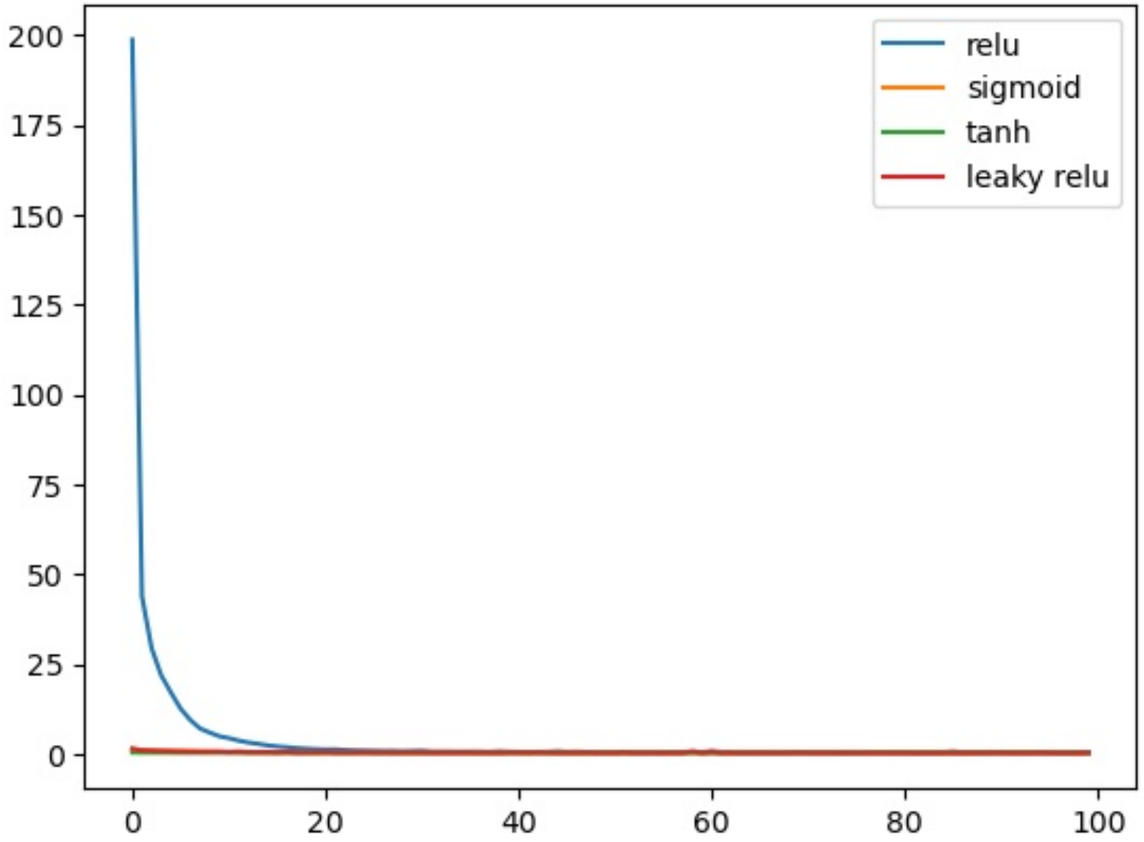
36 variables

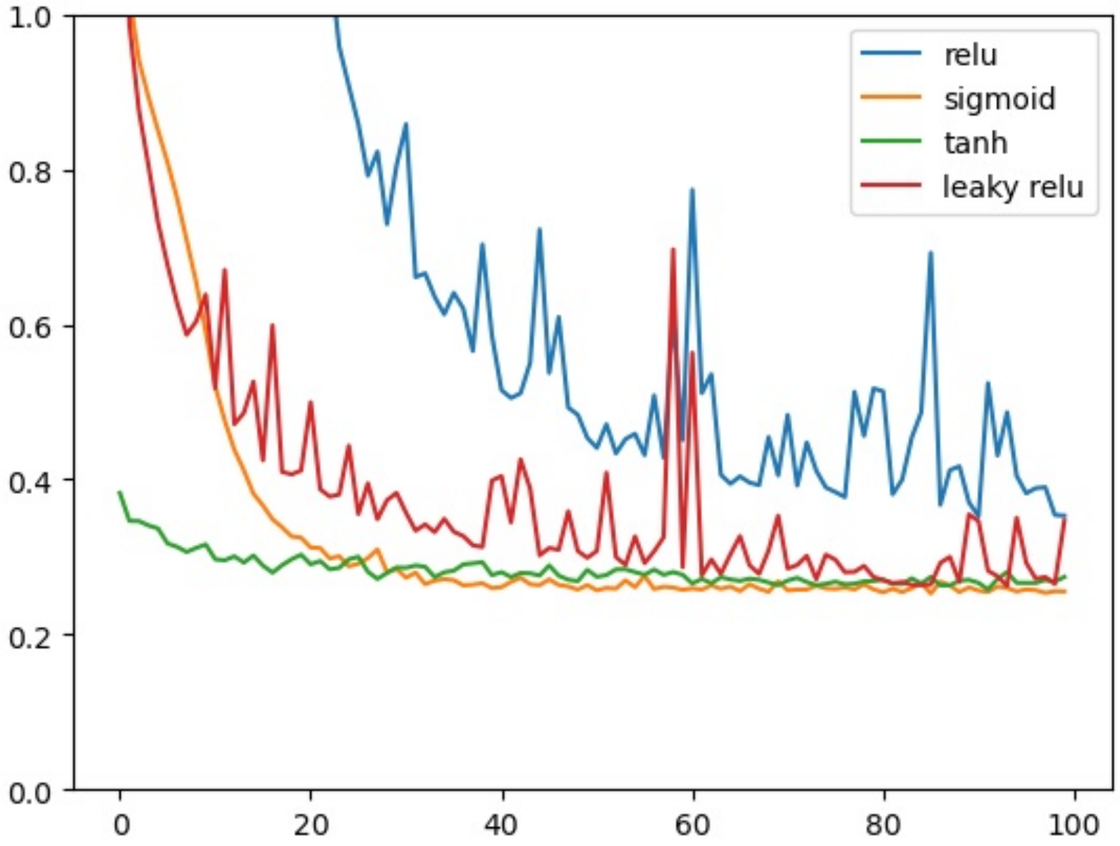
Número de Capas



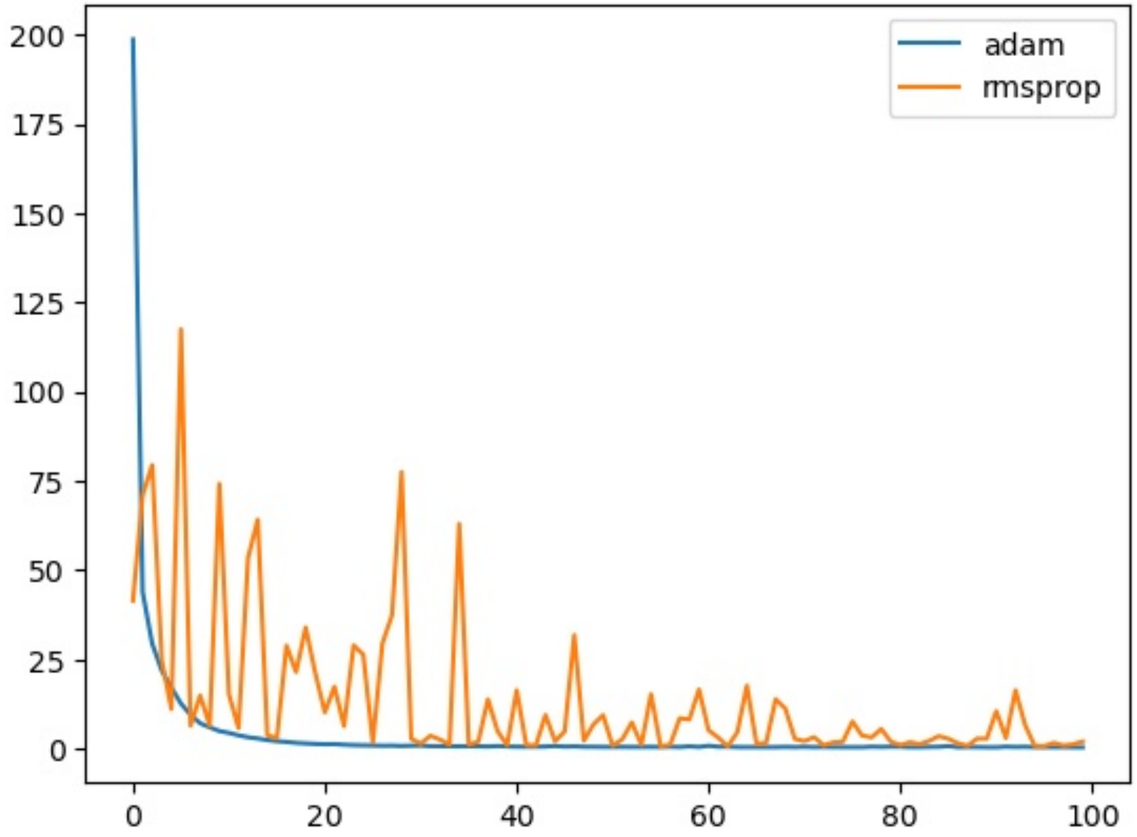


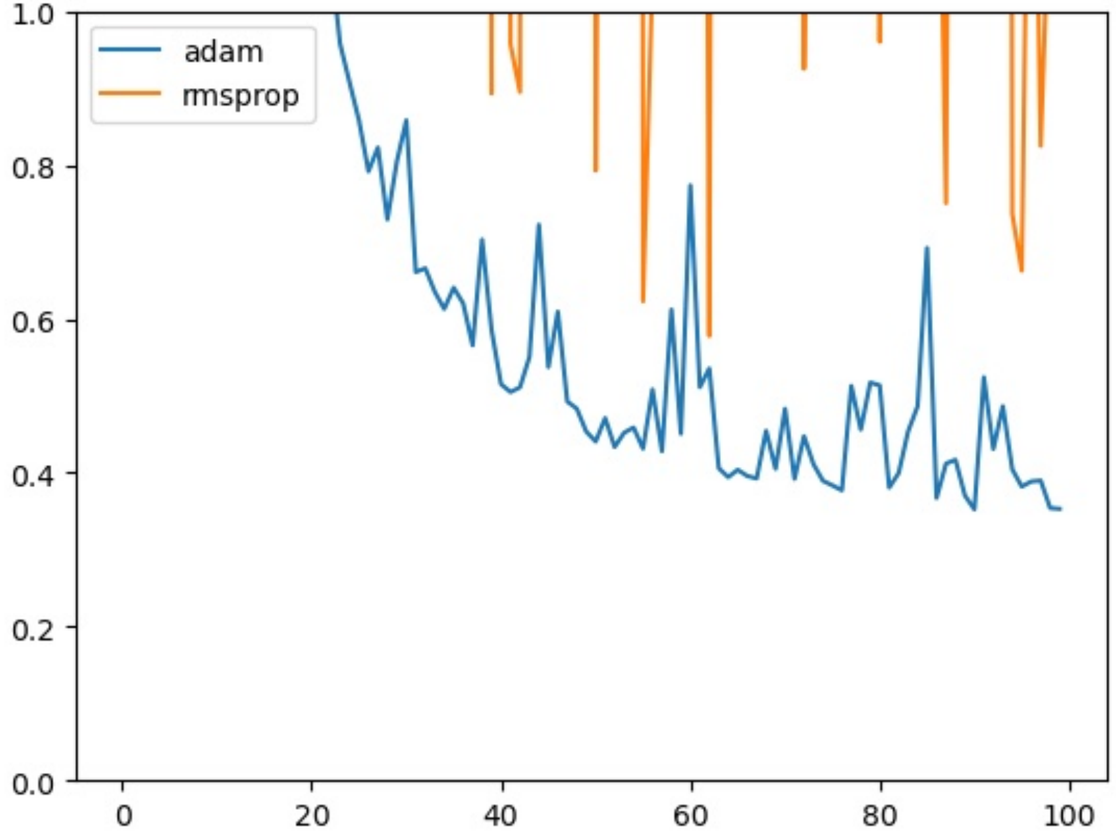
Funciones de Activación



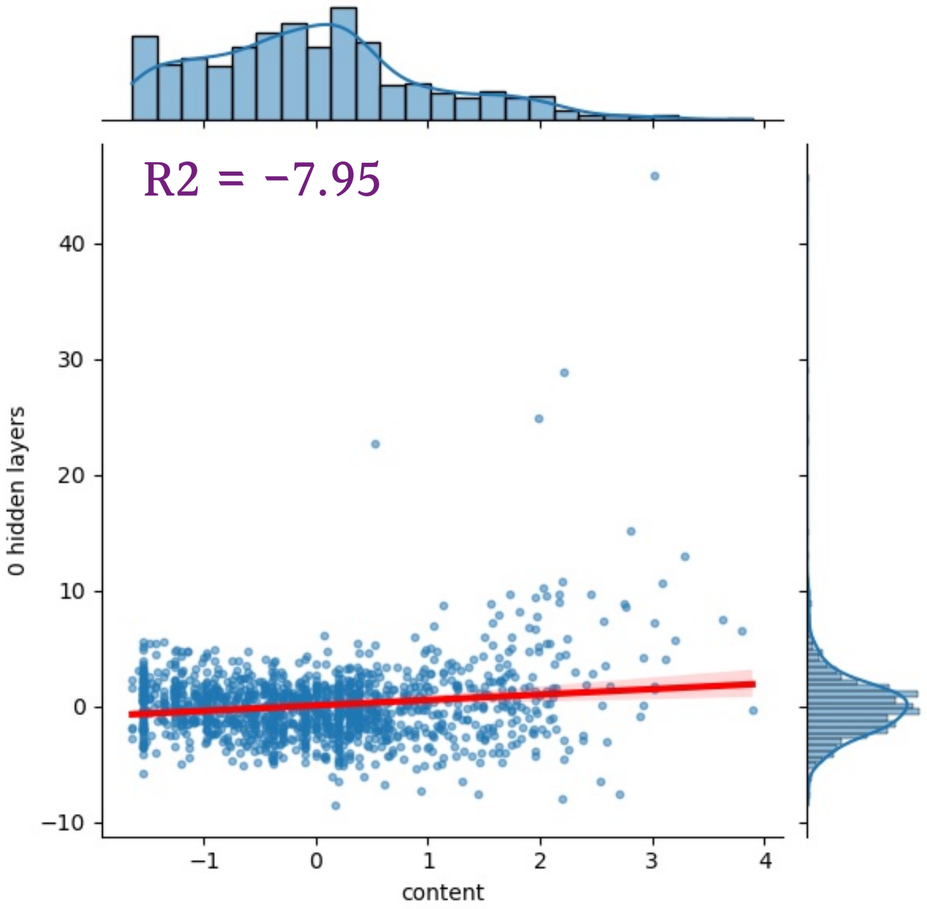


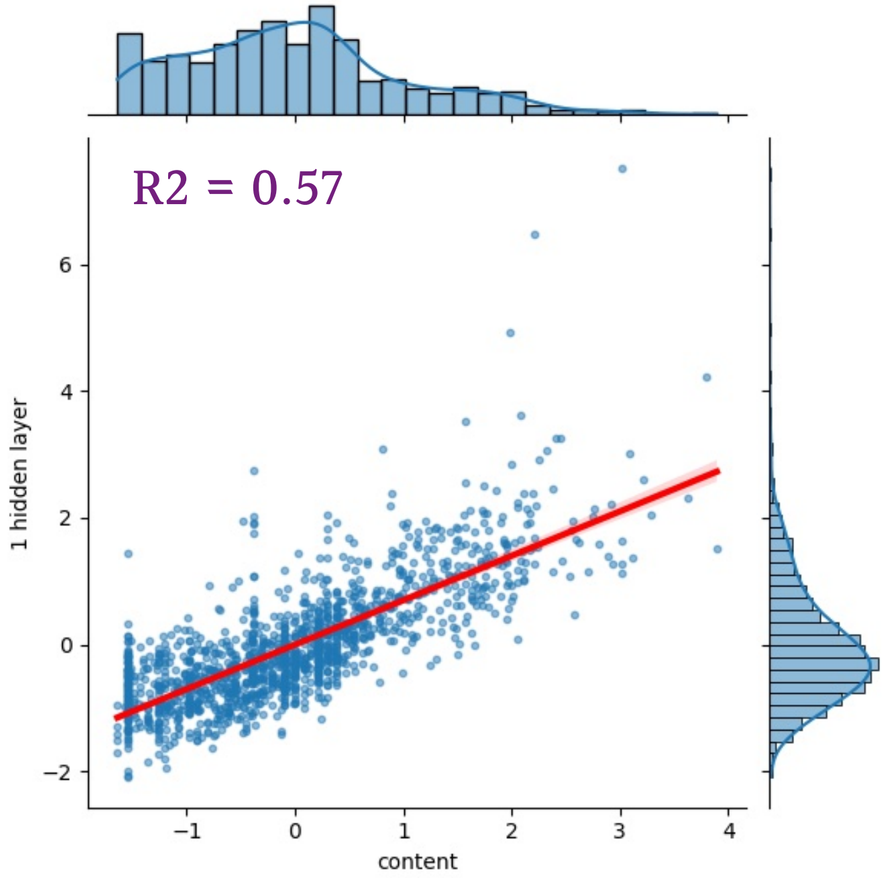
Optimizadores



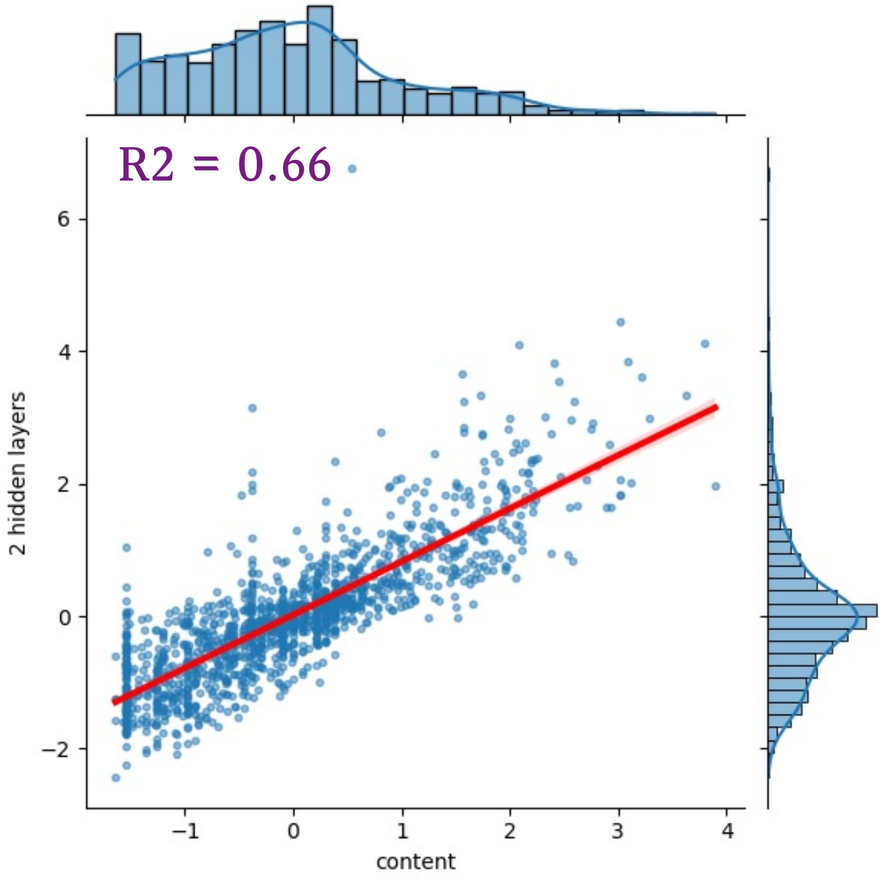


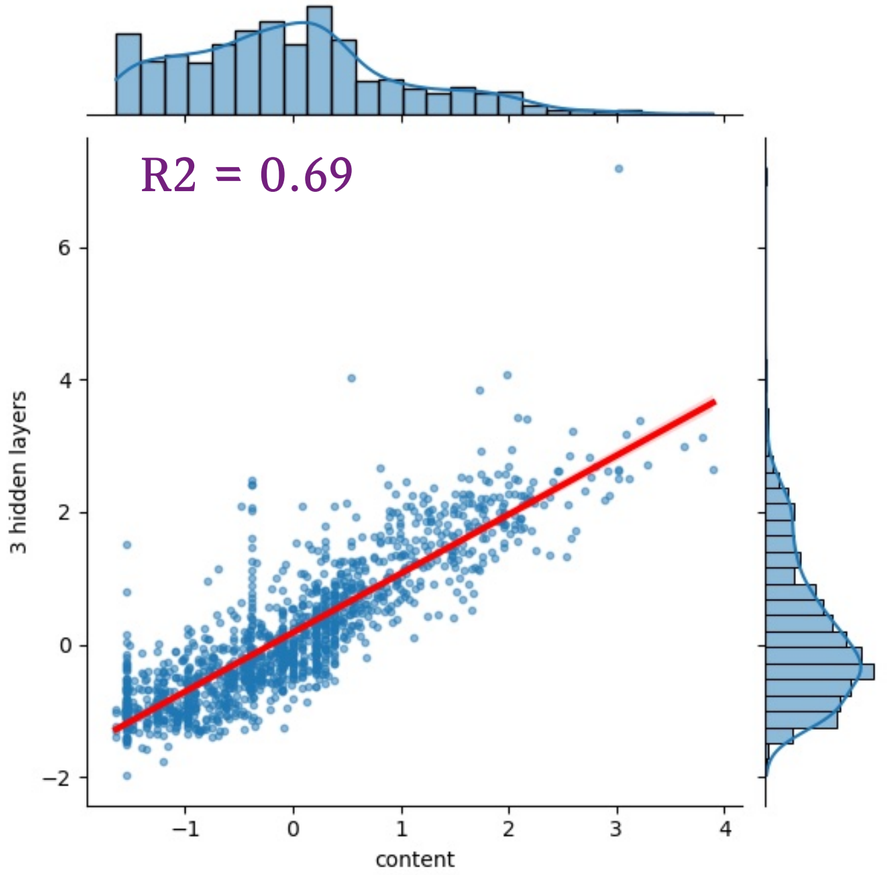
Número de Capas



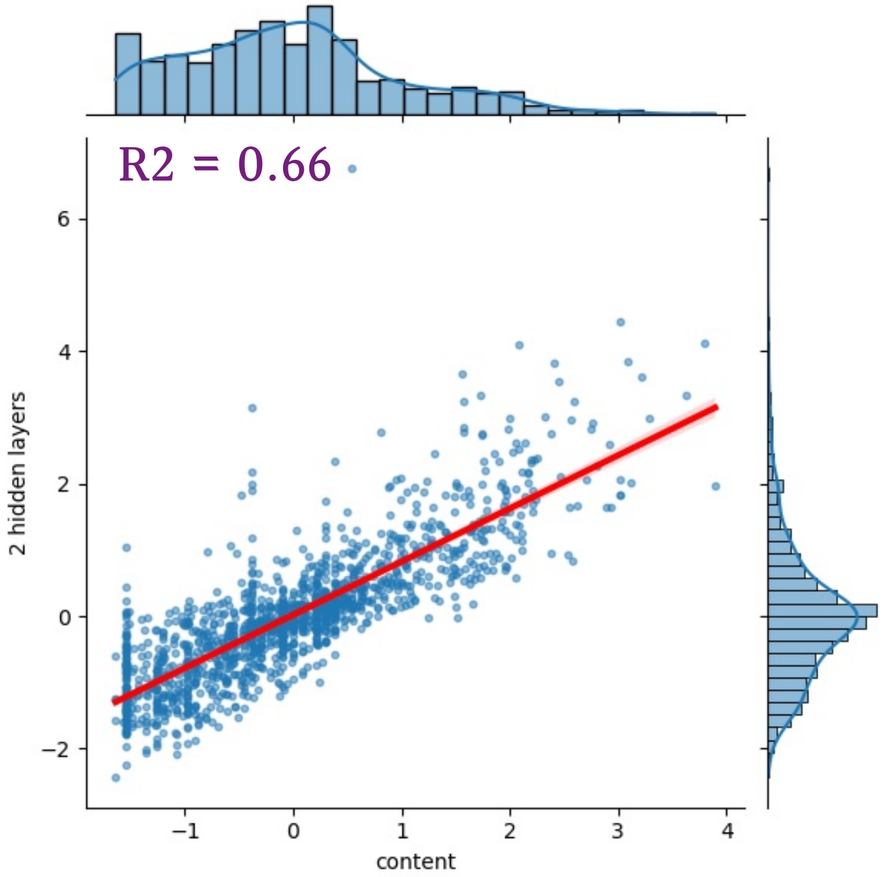


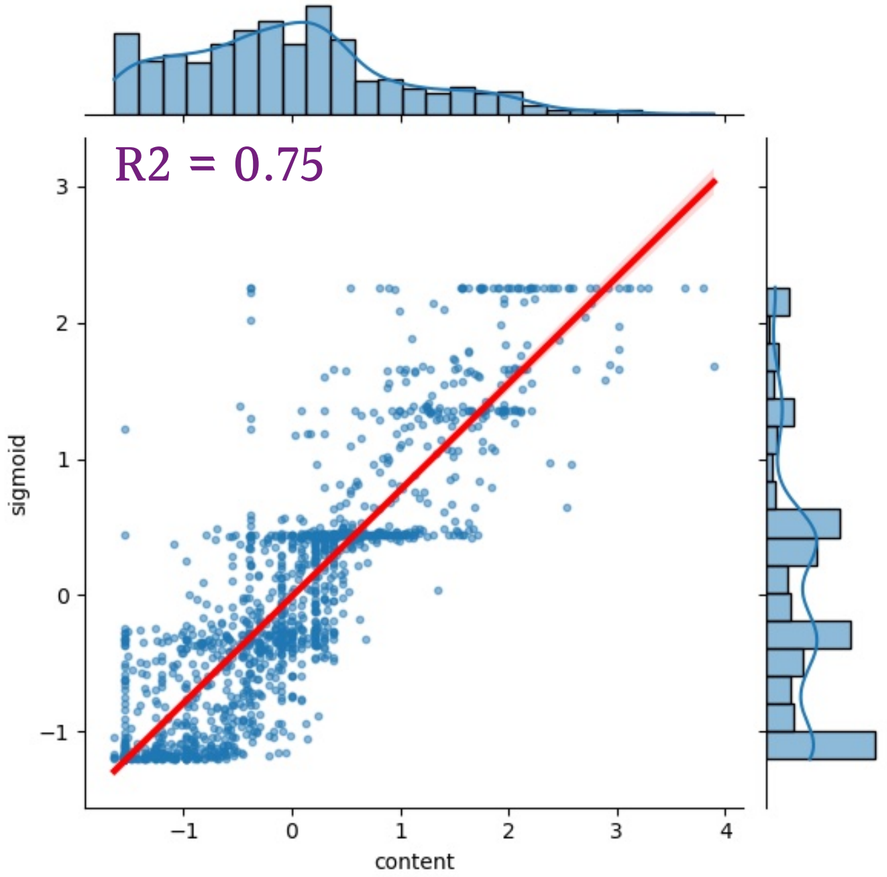
Número de Capas



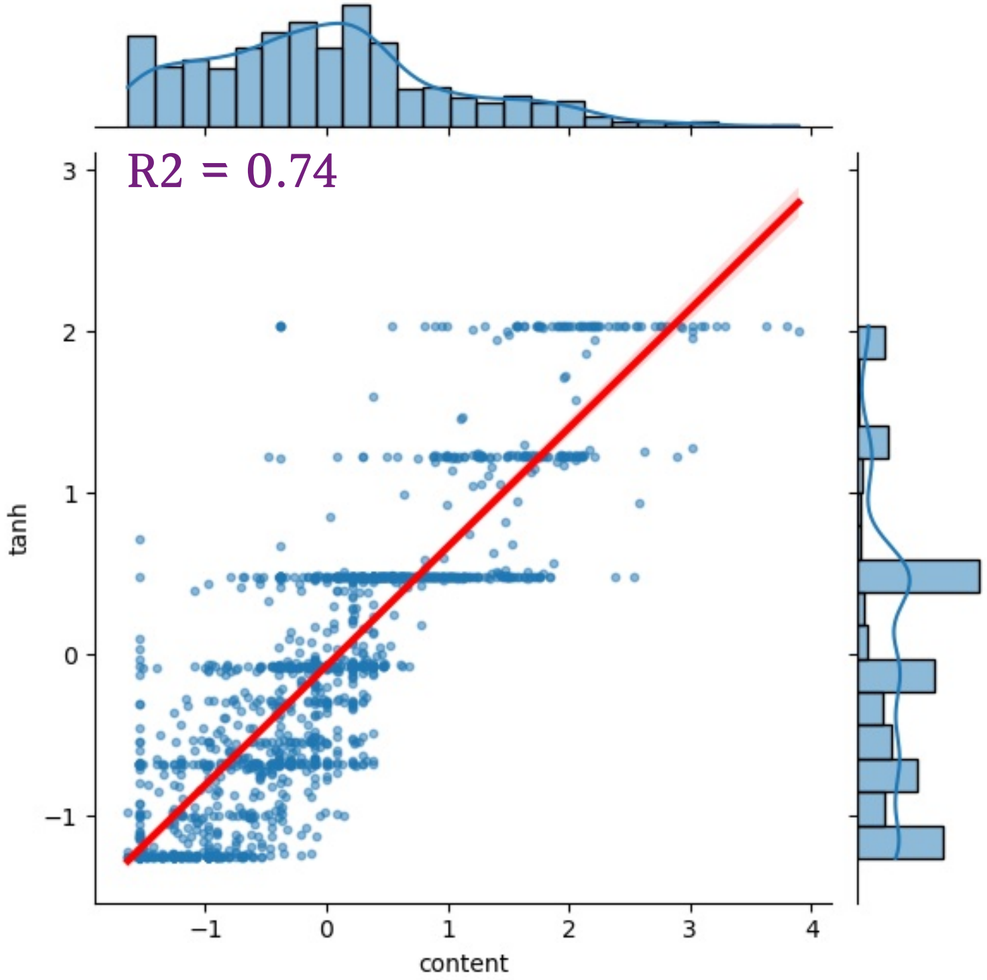


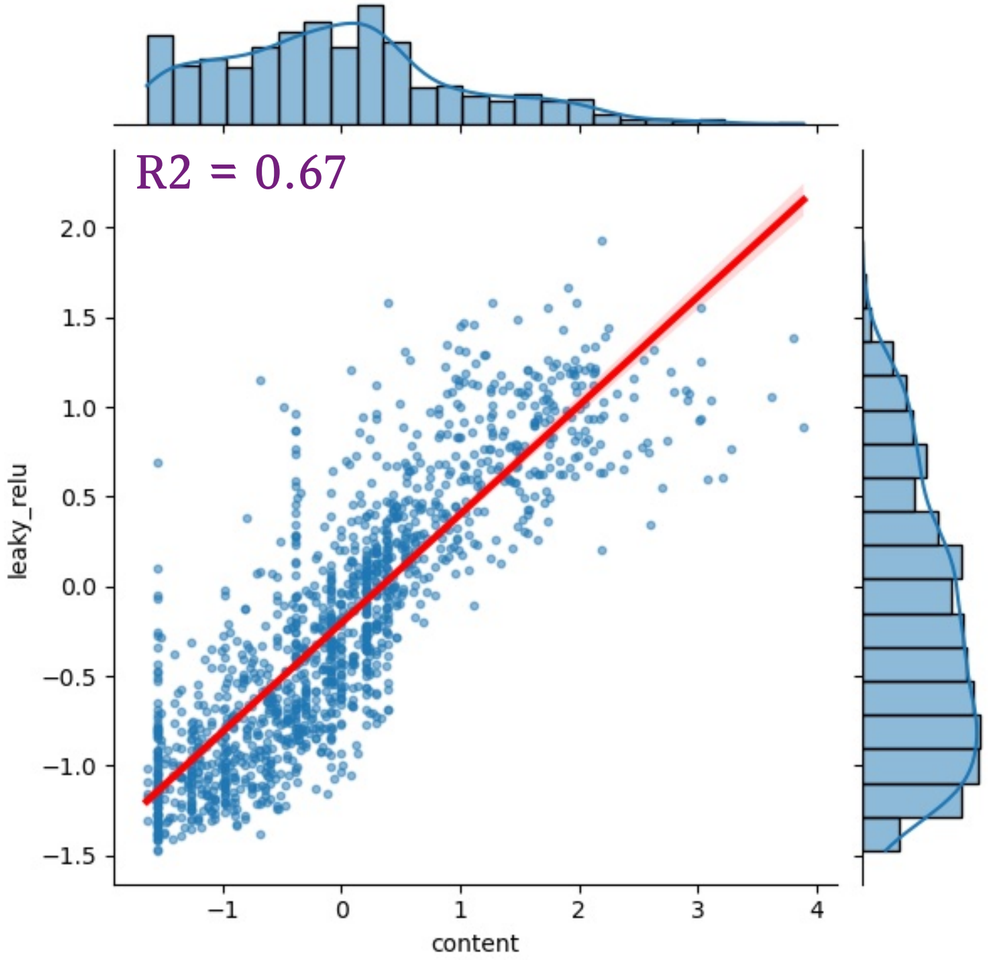
Funciones de Activación



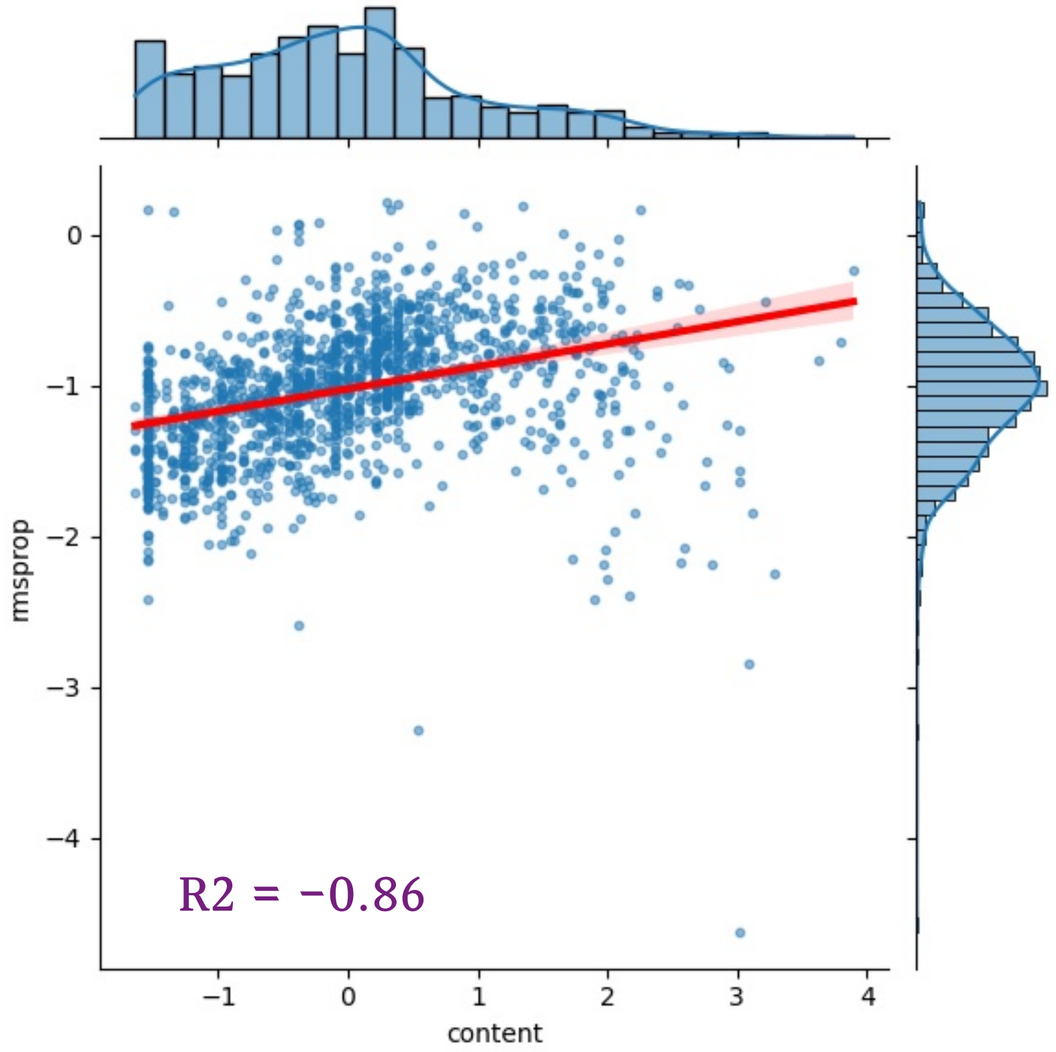


Funciones de Activación





RMS Prop



Otros Algoritmos