Introducción a las Redes Neuronales Artificiales *(RNA's)*

Rodrigo Trejo

'Los cielos proclaman la gloria de Dios, Y el firmamento anuncia la obra de Sus manos.'
Salmo 19:1

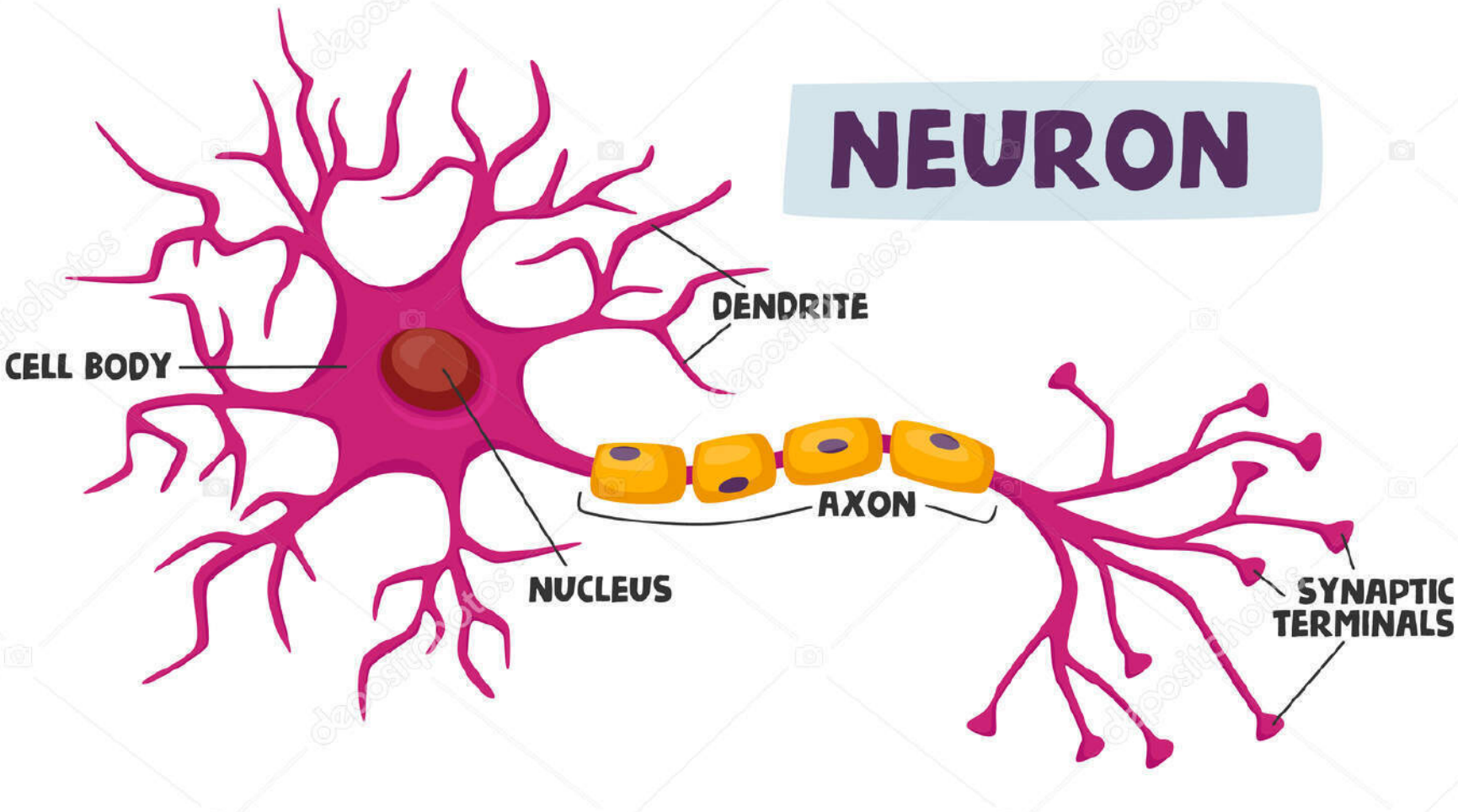
April 5, 2024

Presentation Overview

1. ¿Qué son las RNA's?
2. Predicción en una RNA
3. Entrenamiento de una RNA

Inspiración

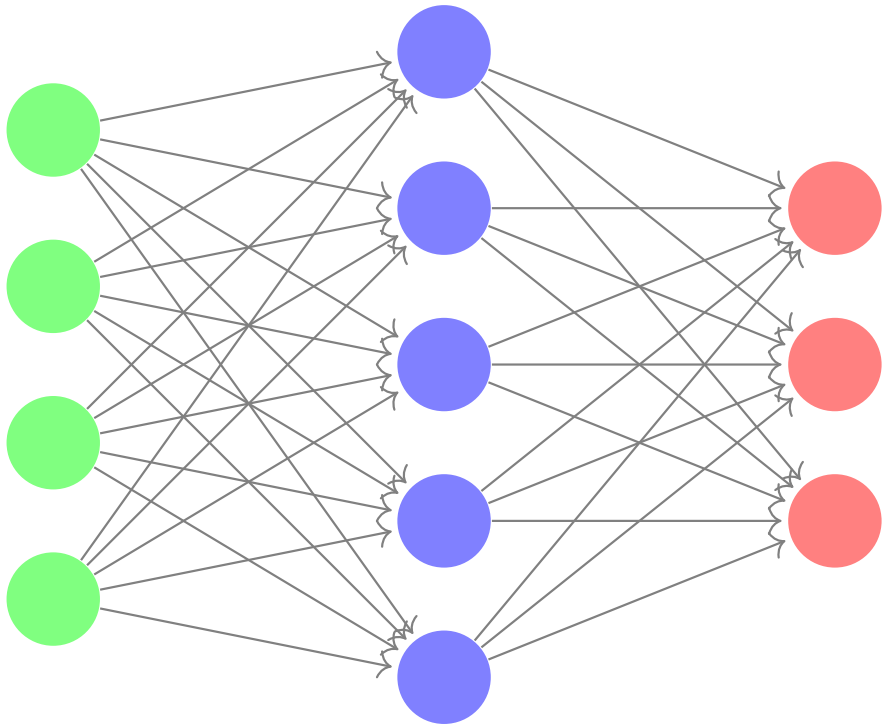
- Es el origen de pensamientos, sentimientos, decisiones y memorias.
- Coordina actividades como:
  - Movimiento
  - Visión
  - Comunicación (oído, habla).
  - Cantidad importante de ***conexiones*** *entre neuronas.*


NEURON
DENDRITE
CELL BODY
AXON
NUCLEUS
SYNAPTIC TERMINALS

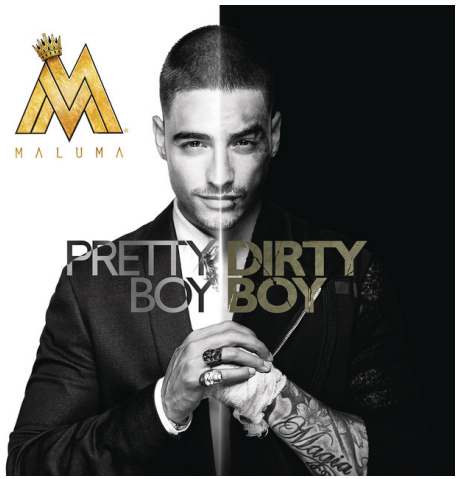
La Neurona Artificial

De manera similar, la conexión (red) de neuronas artificiales, pueden ***dotar a la máquina de***:

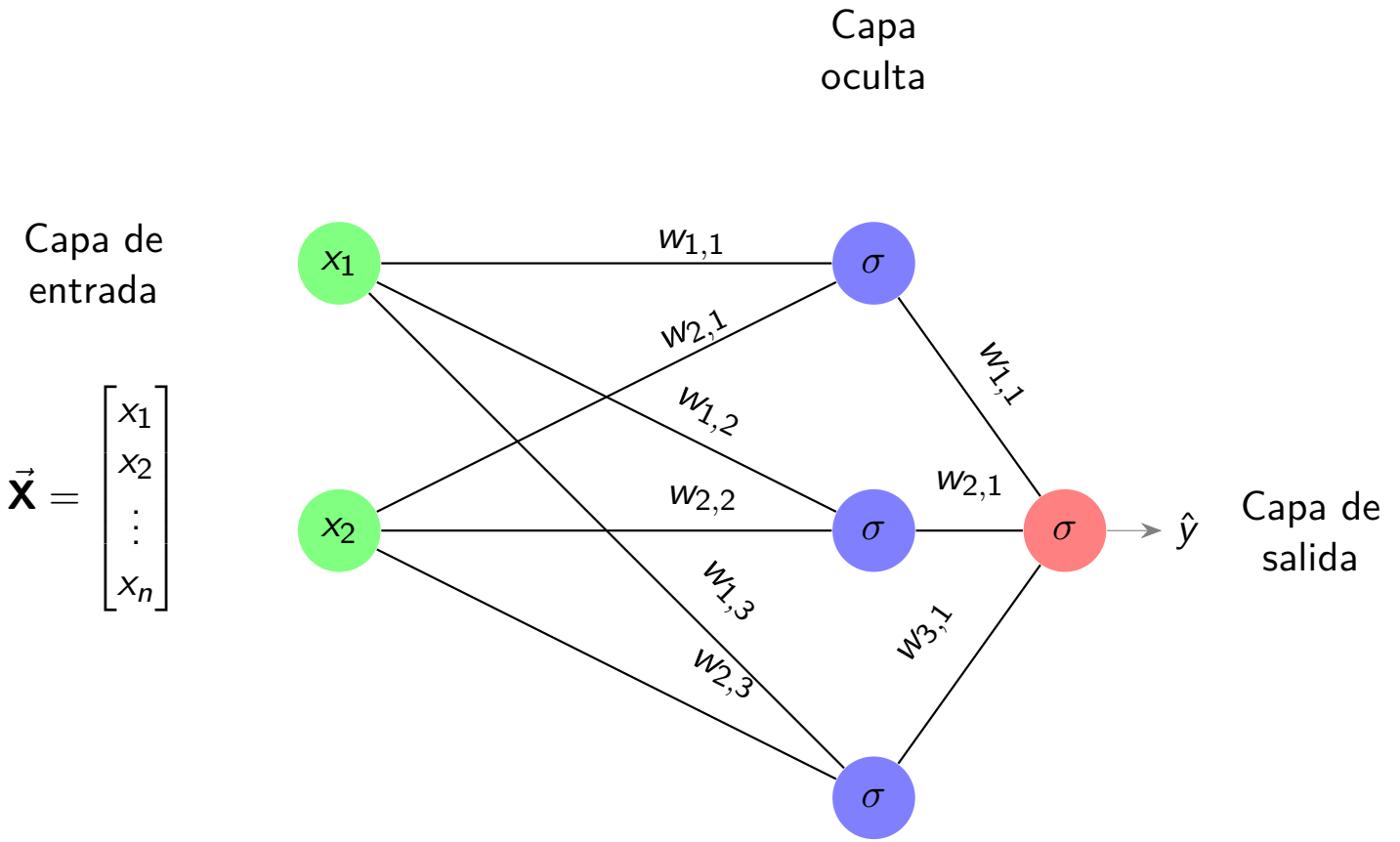
- Coordinación para el movimiento.
- Visión artificial.
- Entendimiento del lenguaje natural.
- Habla.
- Comunicación.
- Reconocimiento de patrones.
- Capacidad de aprendizaje.



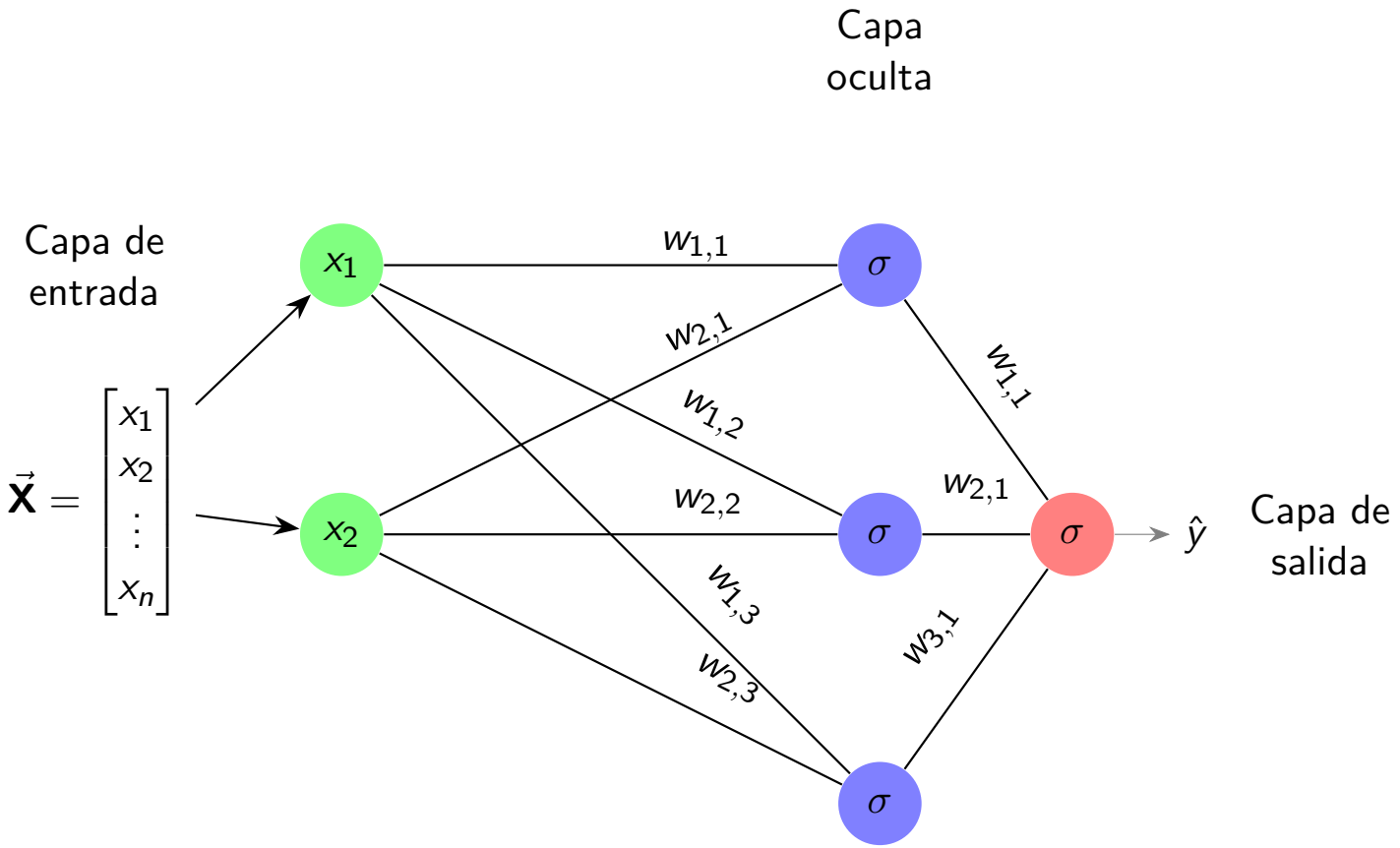
Aplicaciones



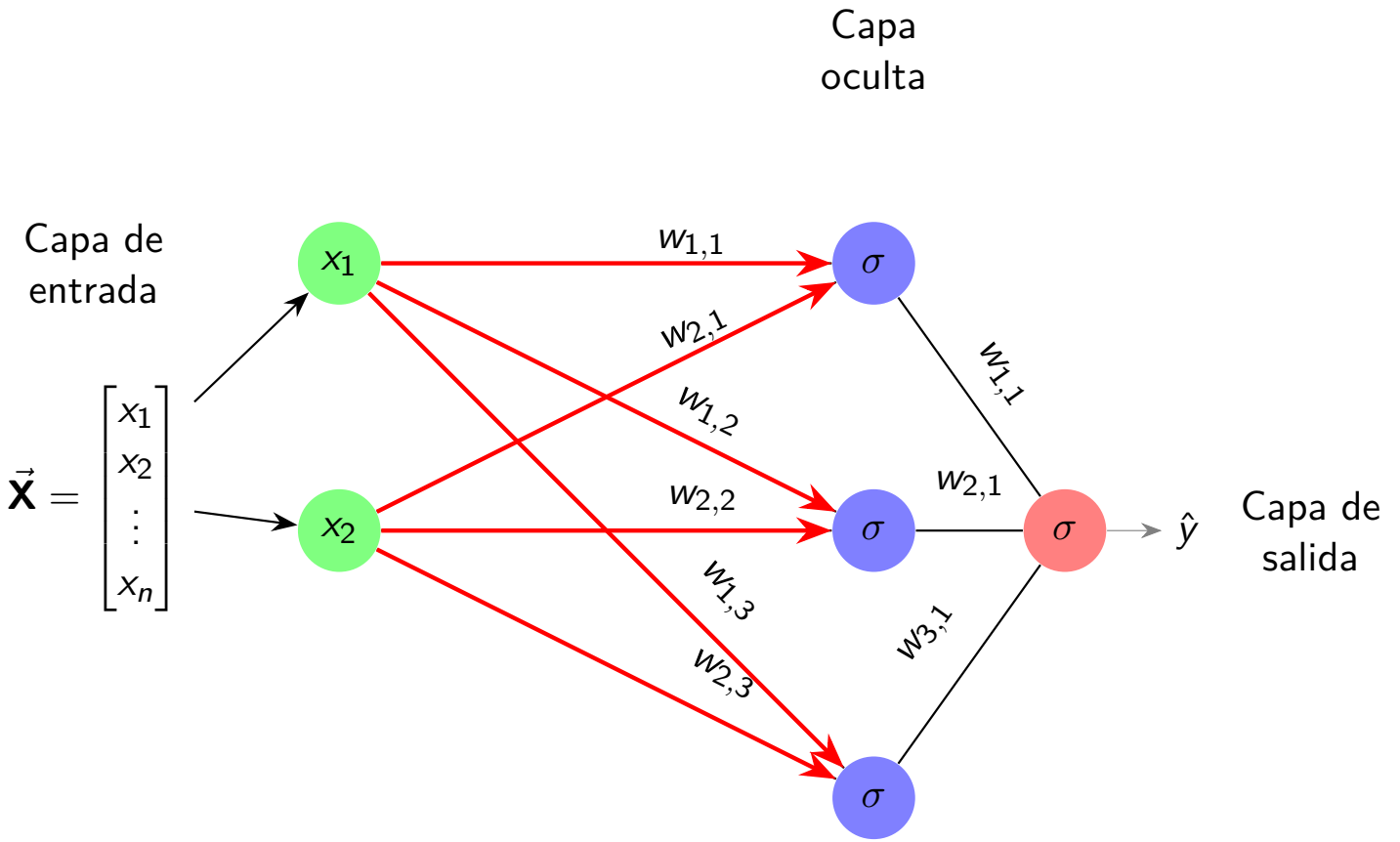
Elementos de una RNA



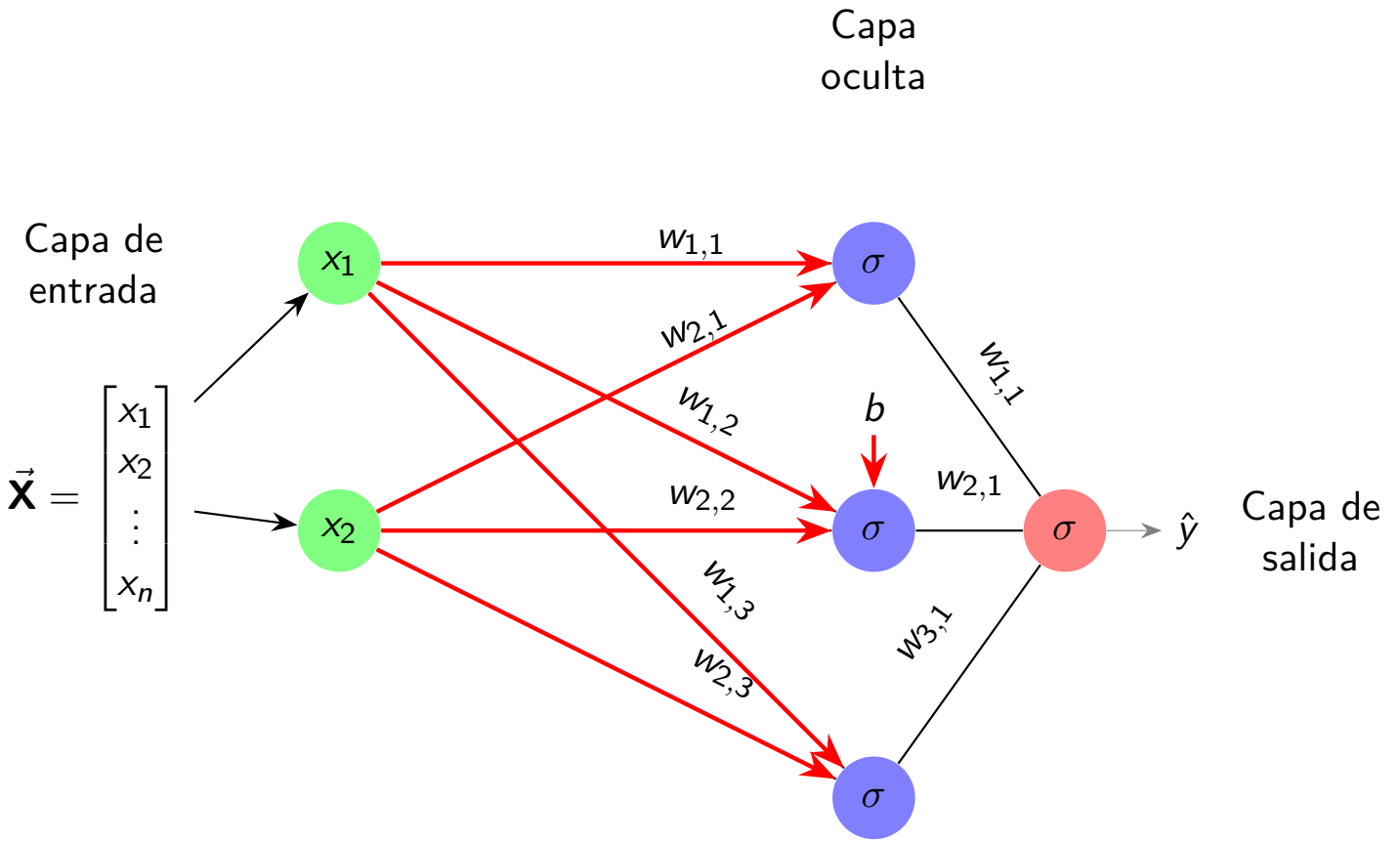
Elementos de una RNA



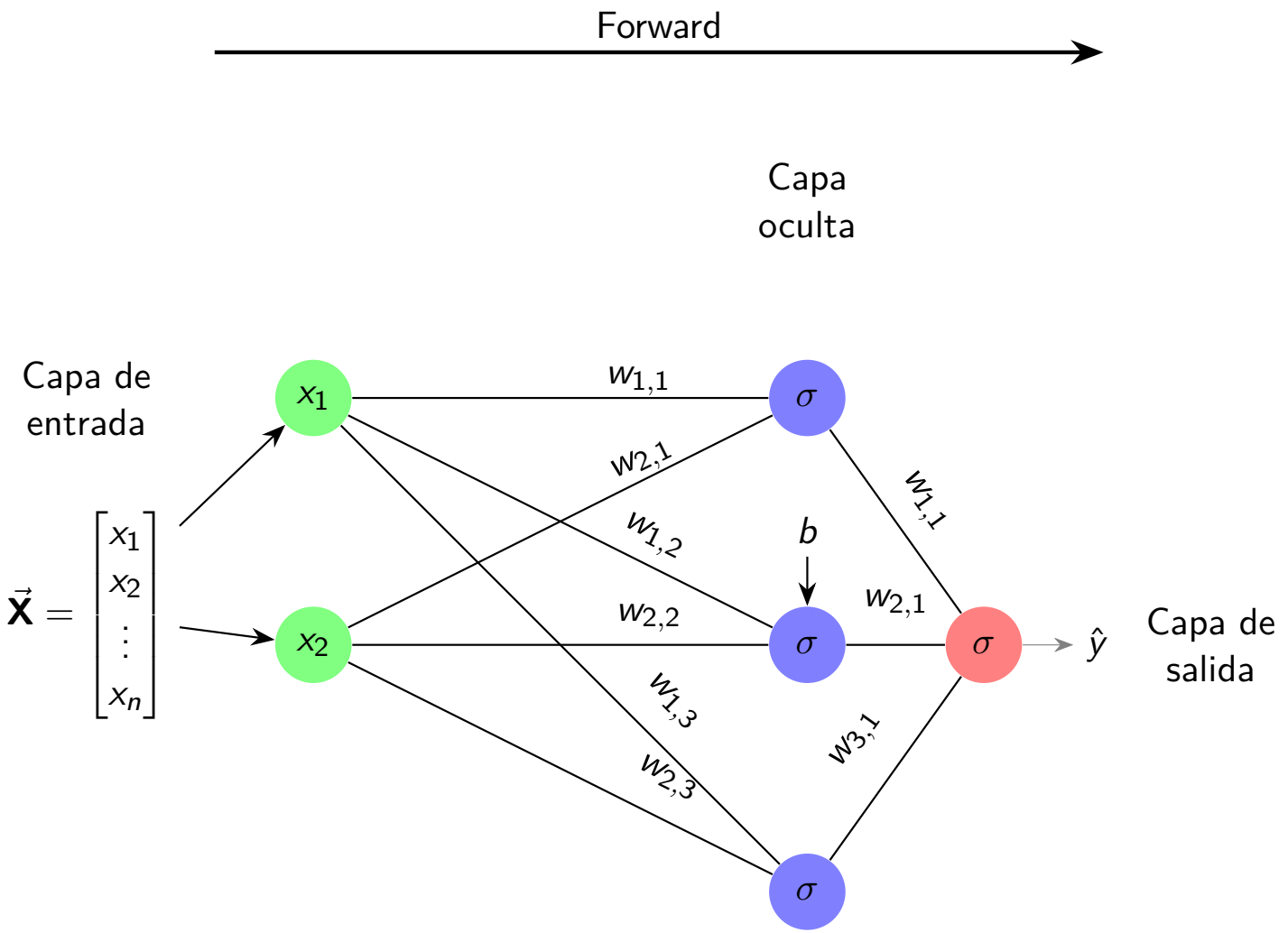
Elementos de una RNA



Elementos de una RNA

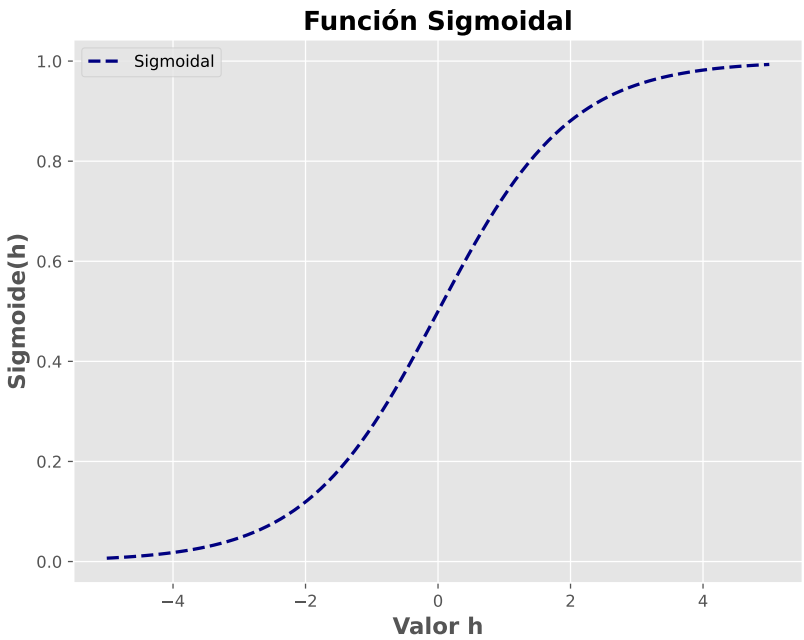


Forward propagation: Realizar la predicción



Funciones de Activación: Sigmoidal

Son las que 'prenden o apagan' las neuronas



Expresión matemática:

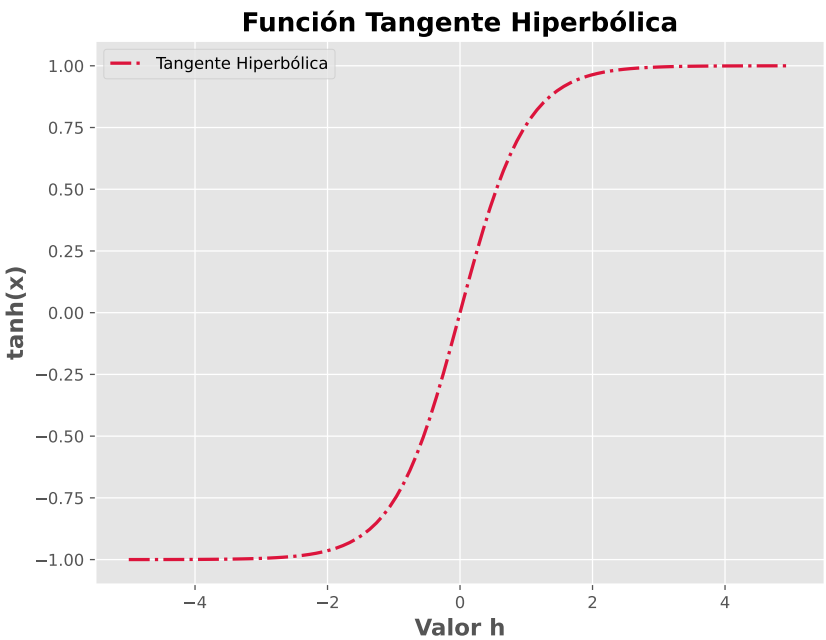
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Dominio y Rango:

- **Dominio:** Todos los números reales ($\mathbb{R}$)
- **Rango:** Entre 0 y 1 (0, 1)

Descripción: Mapea cualquier valor real a un valor entre 0 y 1, de ahí su capacidad para interpretar las predicciones como probabilidades.

Funciones de Activación: Tanh



Expresión Matemática:

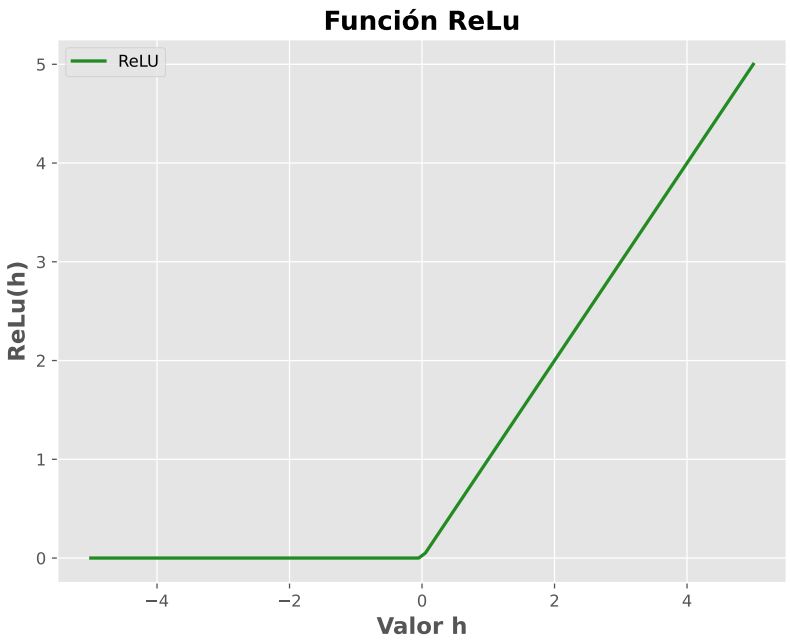
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Dominio y Rango:

- **Dominio:** $\mathbb{R}$
- **Rango:** $(-1, 1)$

Descripción: Transforma valores en un rango de -1 a 1. Esto la hace útil para modelar decisiones donde una característica aporta un valor positivo/negativo a la predicción.

Funciones de Activación: ReLU



Expresión Matemática:

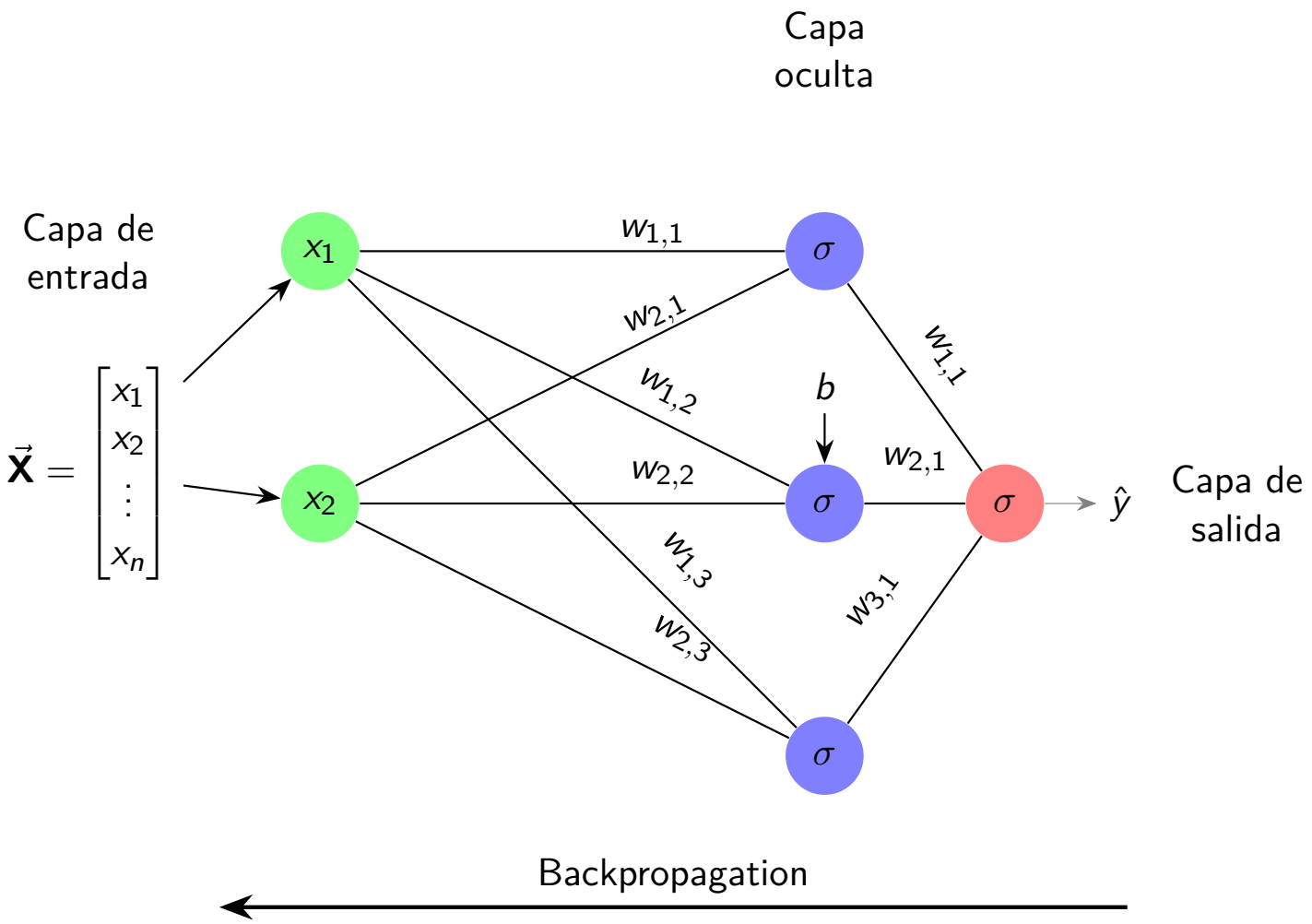
$$\text{ReLU}(x) = \max(0, x)$$

Dominio y Rango:

- **Dominio:** $\mathbb{R}$
- **Rango:** $[0, \infty)$

Descripción: Su simplicidad acelera el proceso de entrenamiento debido a la eficiencia computacional de su cálculo. Además, permite que las neuronas tengan salidas nulas. Enfrenta el problema de las neuronas muertas, lo que puede limitar la capacidad de aprendizaje de la red.

Backpropagation: Corrección de erroes



Función de Error (MSE)

MSE: Mean Square Error

Expresión Matemática

La fórmula para calcular el MSE es:

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

donde:

- n es el número total de observaciones (o ejemplos de entrenamiento),
- Y_i es el valor real del i-ésimo ejemplo,
- $\hat{Y}_i$ es la predicción del modelo para el i-ésimo ejemplo.

Función de Error (MSE)

Explicación: ¿Qué hace el MSE?

El MSE mide el promedio de los cuadrados de los errores; es decir, la diferencia cuadrática promedio entre los valores estimados y los valores reales. Al elevar al cuadrado las diferencias:

- Penaliza más fuertemente los errores grandes, lo que puede ser deseable en muchos casos.
- Asegura que solo tengamos valores positivos, simplificando el análisis de los errores.

Esto lo convierte en una herramienta poderosa para guiar al modelo en el proceso de aprendizaje, buscando minimizar estas diferencias y, por ende, el error de predicción.

Función de Error (MSE)

¿Por qué es importante el MSE?

Una función de error baja indica que el modelo tiene una buena precisión en sus predicciones. Reducir el MSE es fundamental en el proceso de optimización del modelo, llevándolo a hacer predicciones cada vez más cercanas a los valores reales.

Resumen

Elemento	Descripción	Origen
Entradas ($\vec{x}$)	Variables de entrada proporcionadas al modelo (por ejemplo, características de los datos de entrada)	P
Pesos ($\vec{\mathbf{W}}$)	Parámetros aprendidos por el modelo durante el entrenamiento	C
Sesgo ($\vec{b}$)	Término aditivo introducido para ajustar la salida de cada neurona	P
Función de activación	Función aplicada a la salida de cada neurona para introducir no linealidades en el modelo	P

Table: Elementos básicos de una red neuronal (Parte 1)

Resumen

Elemento	Descripción	Origen
Función de error	Función que mide la discrepancia entre las predicciones del modelo y los valores reales	P
Capas ocultas	Capas intermedias entre la capa de entrada y la capa de salida que contienen neuronas	P
Neuronas por capa	Número de unidades de procesamiento en cada capa oculta	P
Predicción ($\hat{y}$)	Valor calculado por el modelo como salida final	C
Target (y)	Valor real al que se compara la predicción durante el entrenamiento	P

Table: Elementos básicos de una red neuronal (Parte 2)

Resumen

Entrenar una Red Neuronal implica aplicar el método de *Prueba y Error*. Es un algoritmo que nos puede ayudar a dominar el arte de tener paciencia.

Entrenamiento de una RNA

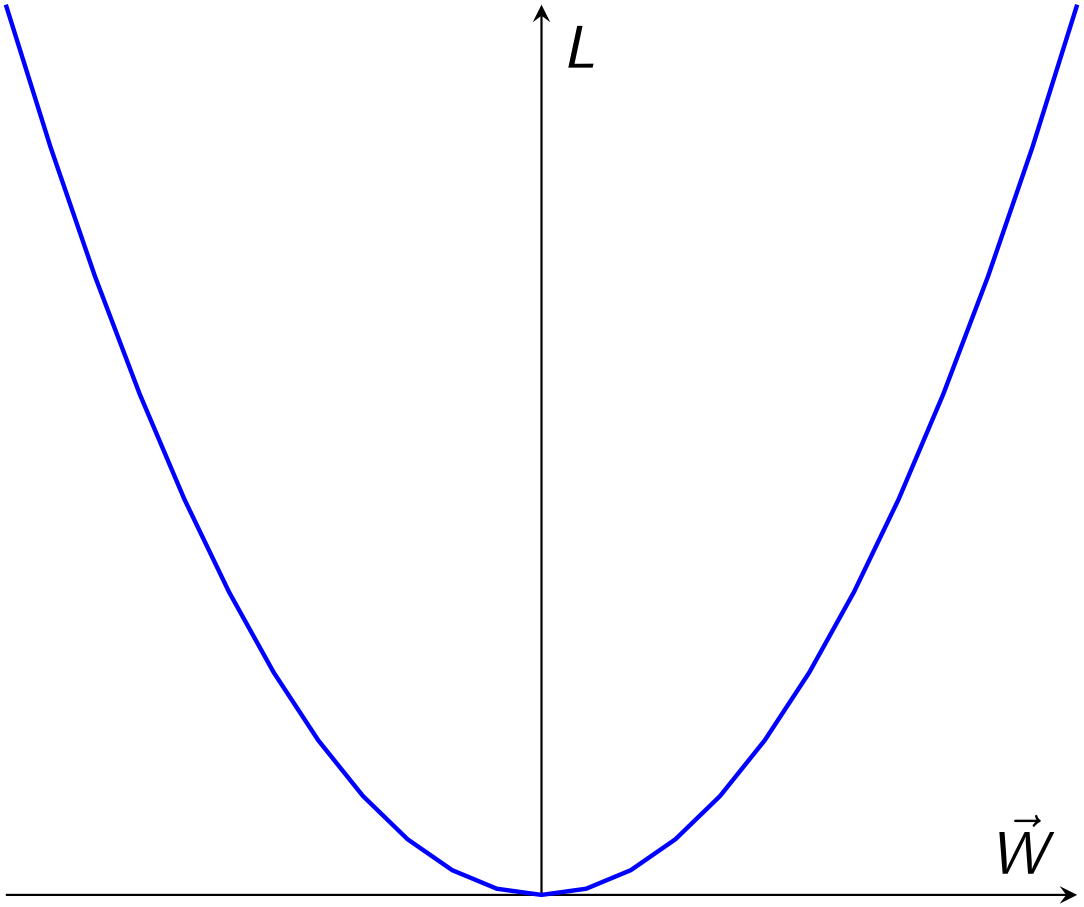
Objetivo: *Minimizar la función de error.*

¿CÓMO?

- Aplicando derivadas
- Para el caso de una función multivariable: **Aplicando el gradiente**
- Como es difícil usar un método analítico (con fórmulas) para determinar el mínimo de la función de error:
  - Aplicamos un método numérico iterativo conocido como: ***Descenso de gradiente.***

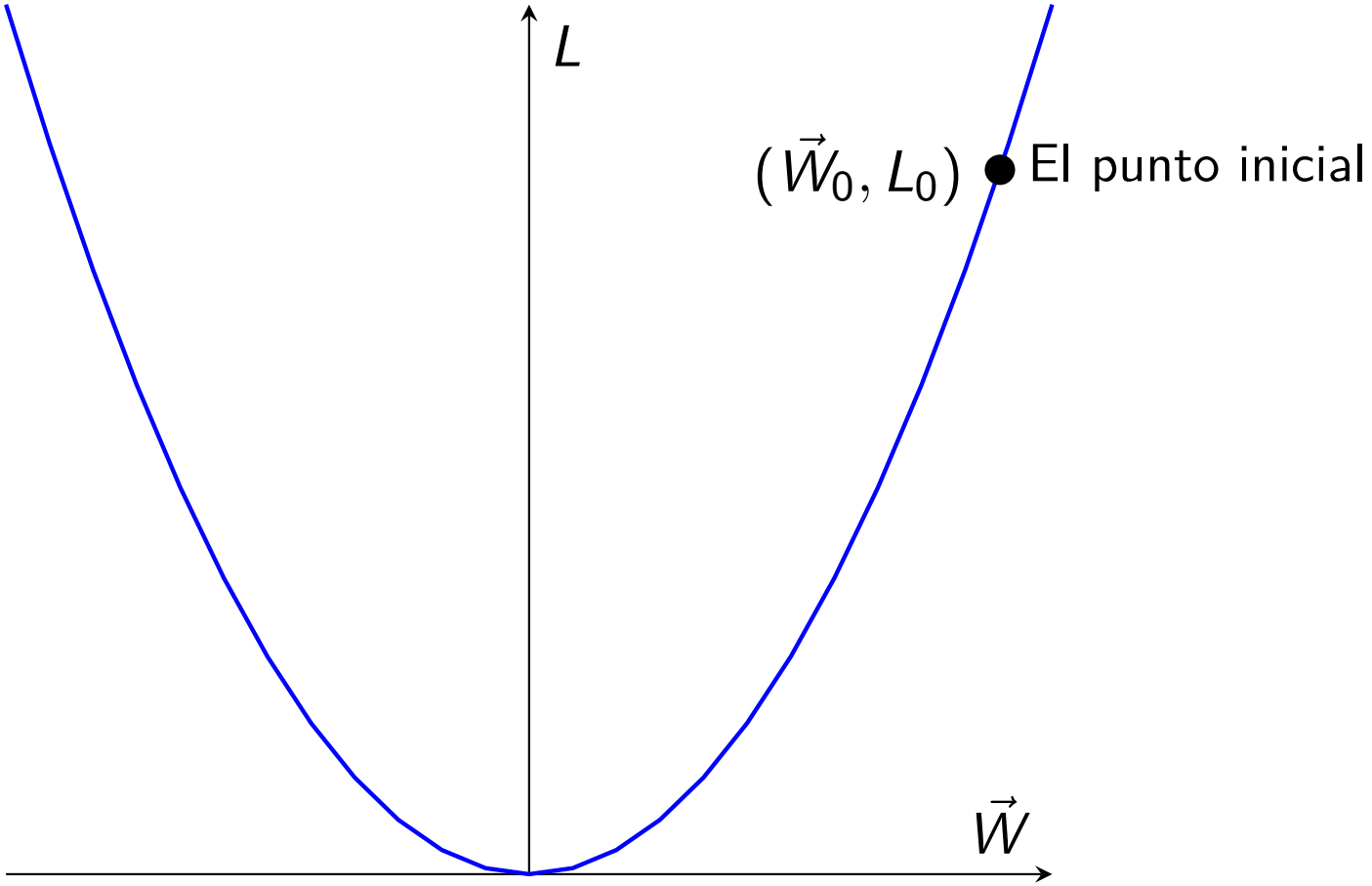
Descenso de gradiente - El parámetro learning_rate (α)

Descenso de Gradiente en $L = \vec{W}^2$: $\vec{W}_{n+1} = \vec{W}_n - \alpha \cdot \nabla L(\vec{W}_n)$



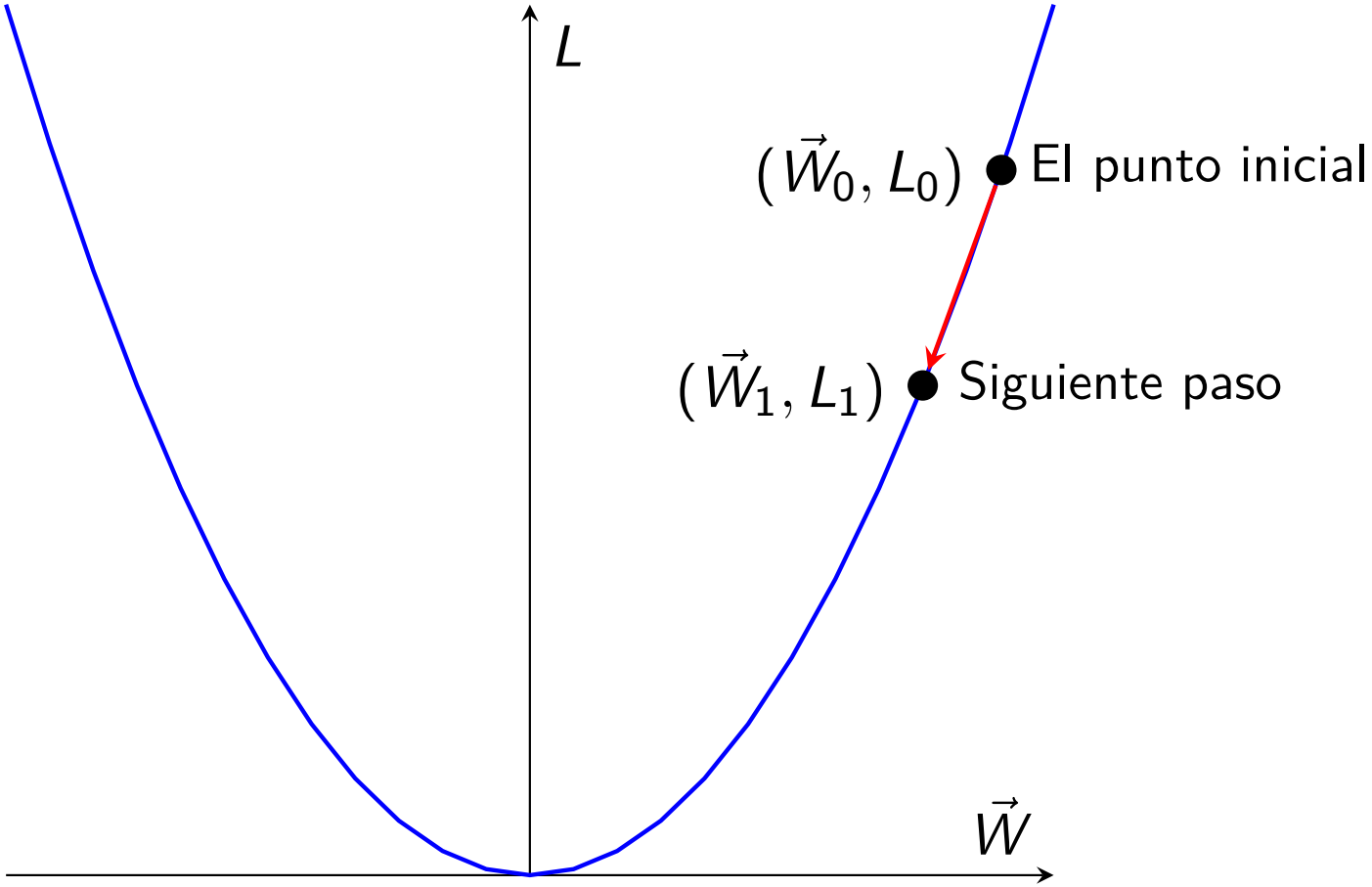
Descenso de gradiente - El parámetro learning_rate (α)

Descenso de Gradiente en $L = \vec{W}^2$: $\vec{W}_{n+1} = \vec{W}_n - \alpha \cdot \nabla L(\vec{W}_n)$



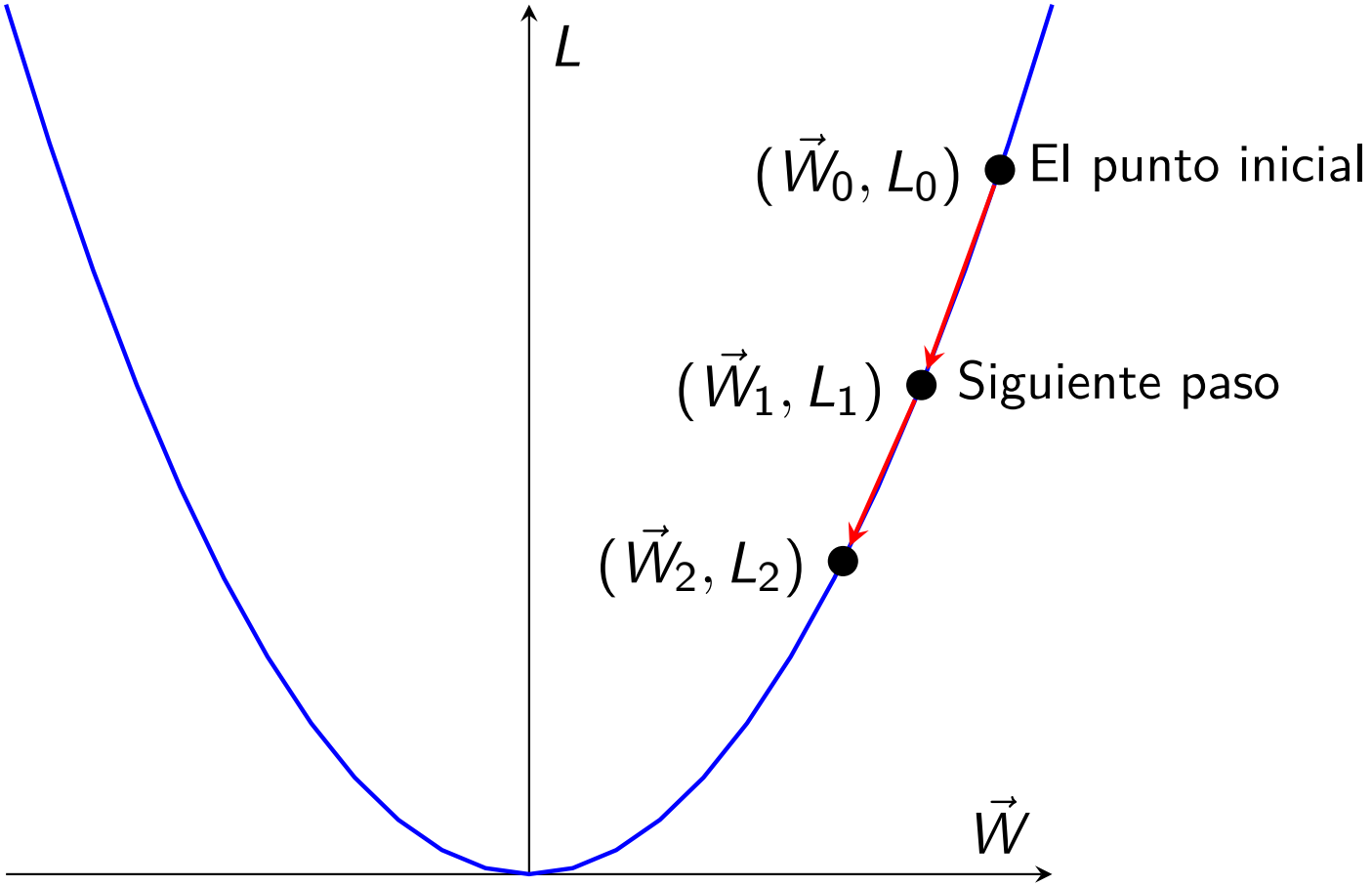
Descenso de gradiente - El parámetro learning_rate (α)

Descenso de Gradiente en $L = \vec{W}^2$: $\vec{W}_{n+1} = \vec{W}_n - \alpha \cdot \nabla L(\vec{W}_n)$



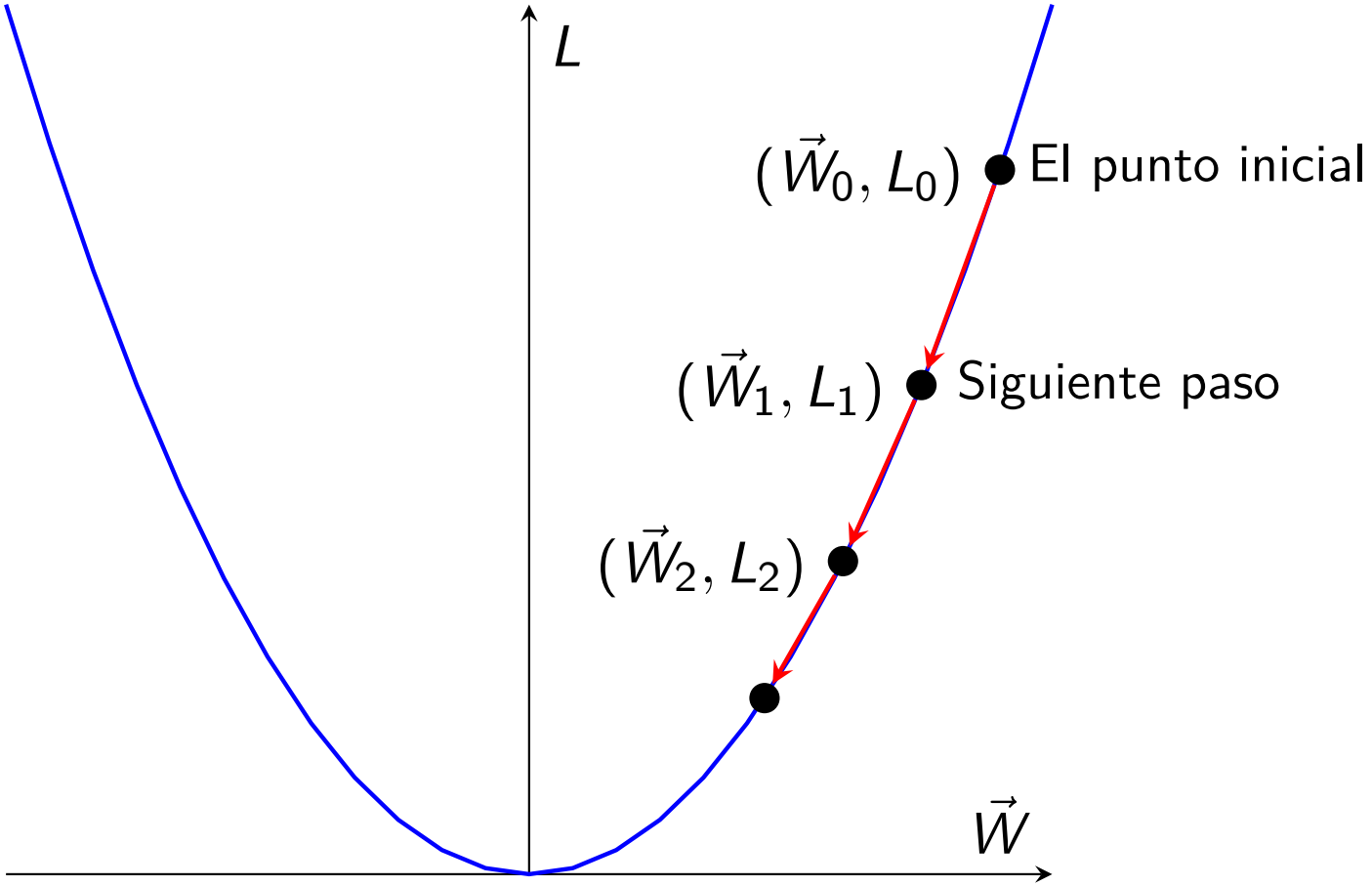
Descenso de gradiente - El parámetro learning_rate (α)

Descenso de Gradiente en $L = \vec{W}^2$: $\vec{W}_{n+1} = \vec{W}_n - \alpha \cdot \nabla L(\vec{W}_n)$



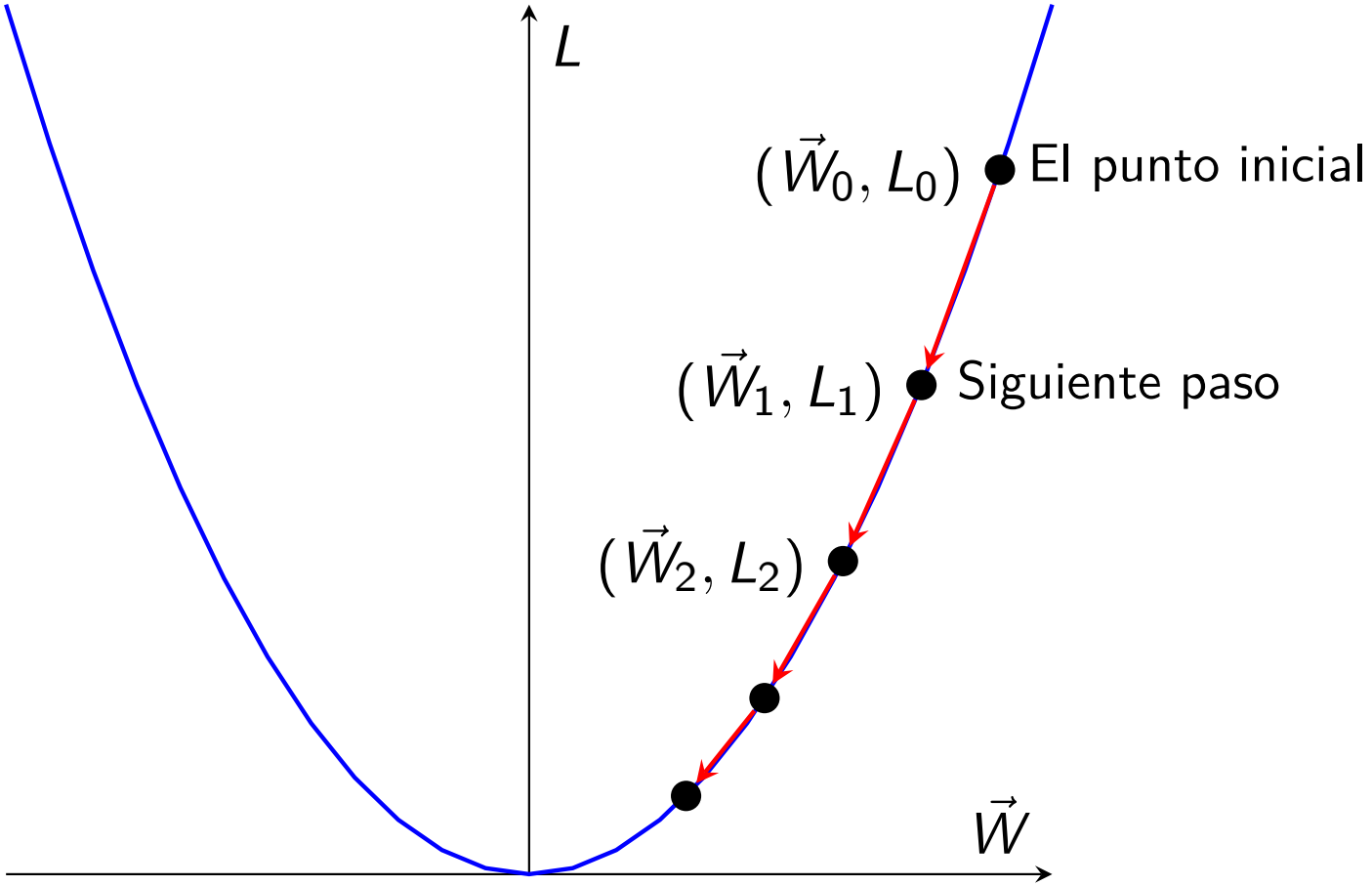
Descenso de gradiente - El parámetro learning_rate (α)

Descenso de Gradiente en $L = \vec{W}^2$: $\vec{W}_{n+1} = \vec{W}_n - \alpha \cdot \nabla L(\vec{W}_n)$



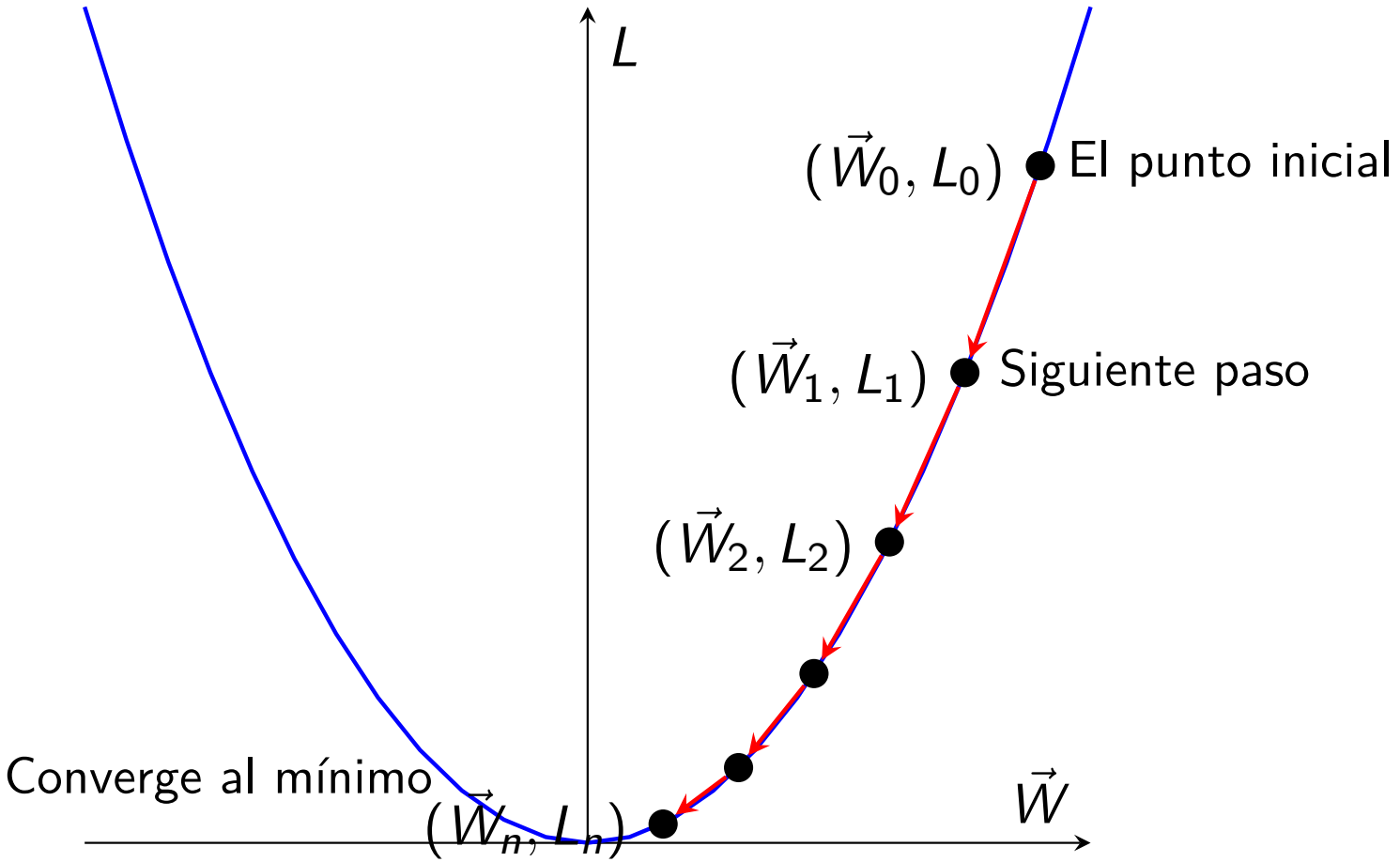
Descenso de gradiente - El parámetro learning_rate (α)

Descenso de Gradiente en $L = \vec{W}^2$: $\vec{W}_{n+1} = \vec{W}_n - \alpha \cdot \nabla L(\vec{W}_n)$



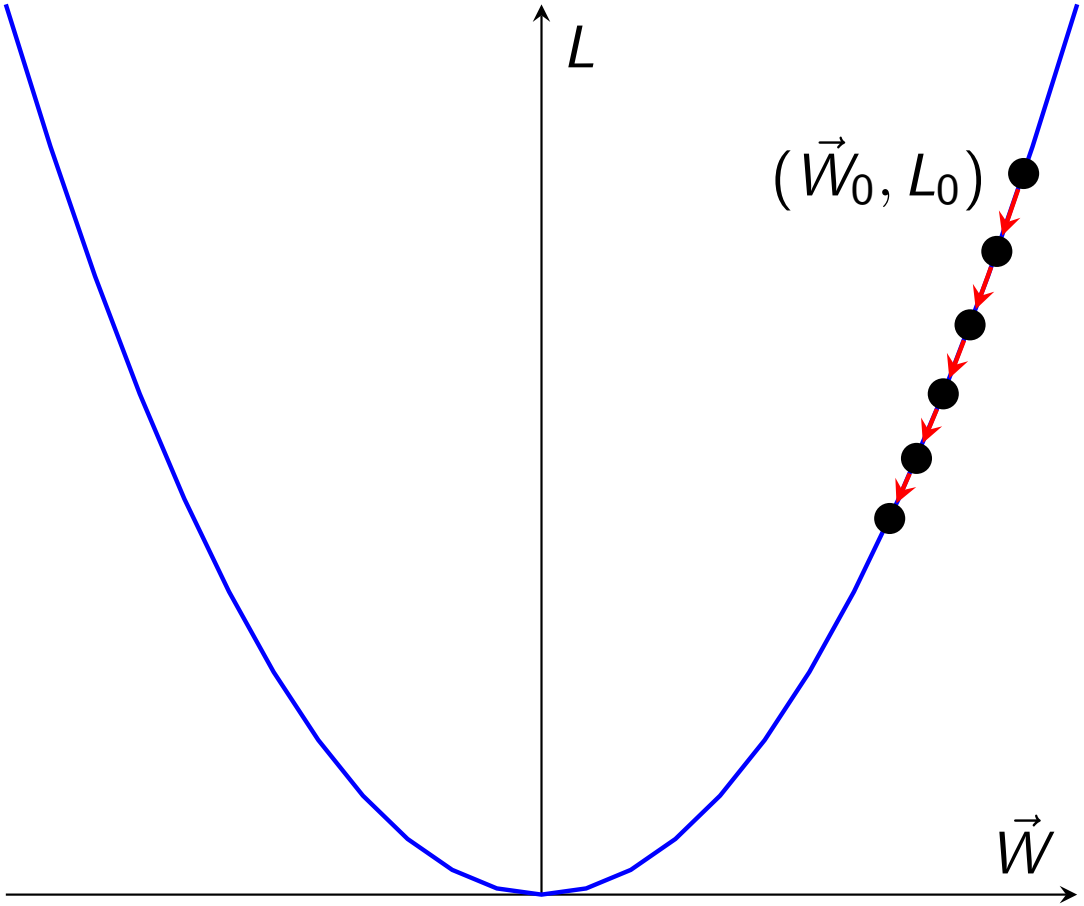
Descenso de gradiente - El parámetro learning_rate (α)

Descenso de Gradiente en $L = \vec{W}^2$: $\vec{W}_{n+1} = \vec{W}_n - \alpha \cdot \nabla L(\vec{W}_n)$



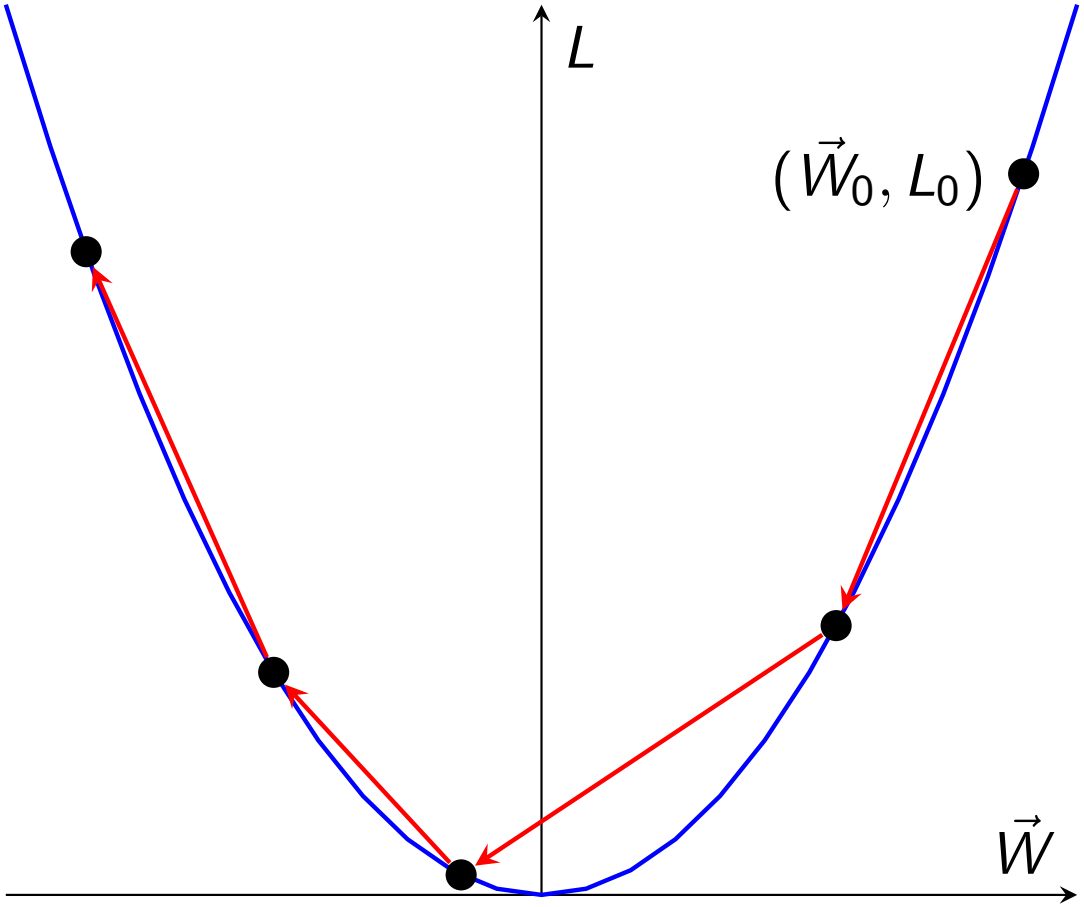
Cuando α es muy pequeño

Descenso de Gradiente en $L = \vec{W}^2$: $\vec{W}_{n+1} = \vec{W}_n - \alpha \cdot \nabla L(\vec{W}_n)$



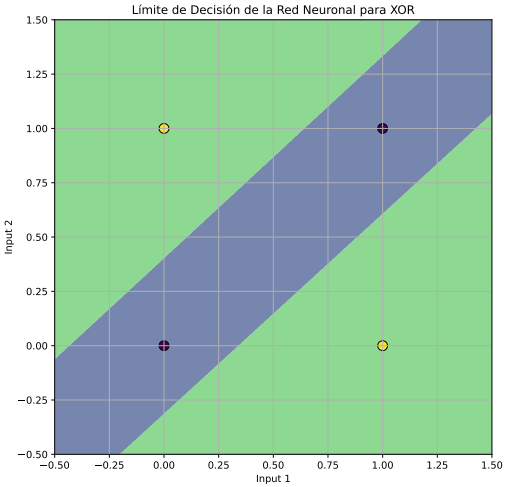
Cuando α es muy grande

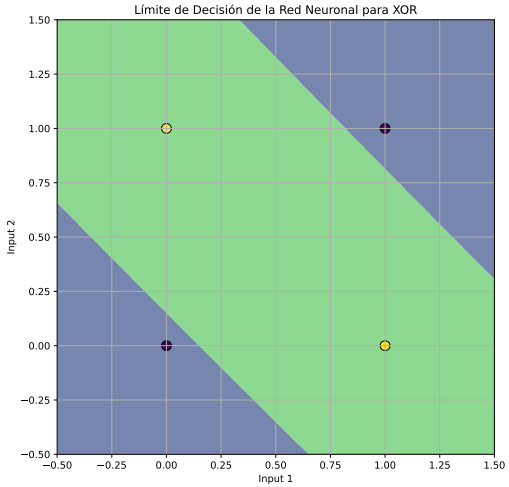
Descenso de Gradiente en $L = \vec{W}^2$: $\vec{W}_{n+1} = \vec{W}_n - \alpha \cdot \nabla L(\vec{W}_n)$

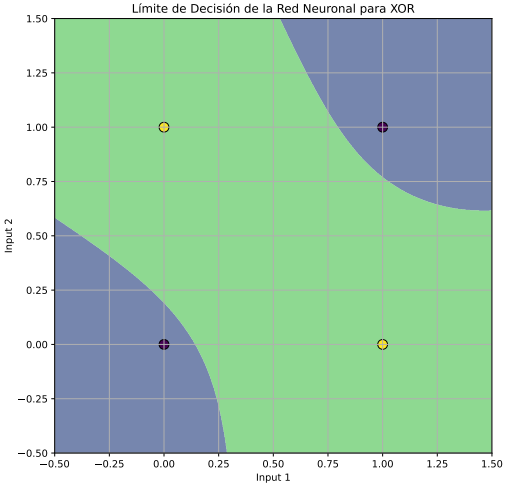


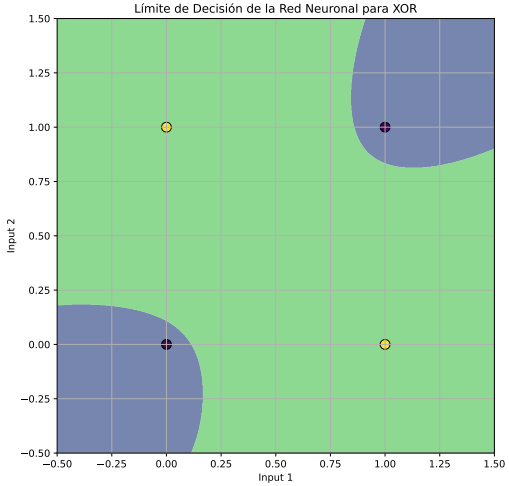
Y si... el learning_rate también lo define el programador /:

Ejemplito: Compuerta XOR









¡Gracias!